Національний університет «Полтавська політехніка імені Юрія Кондратюка»

Навчально-науковий інститут інформаційний технологій і робототехніки

(повна назва інституту/факультету)

Кафедра комп'ютерних та інформаційних технологій і систем

(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)

на здобуття освітнього ступеня «магістр»

Спеціальність 122 «Комп'ютерні науки»

(код і назва)

на тему

«Розробка і організаційно-технічні методи забезпечення тестового контролю знань в системі сучасної шкільної освіти»

Виконав:
студент 6 курсу, групи 602-ТН

спеціальності 122 «Комп'ютерні науки»

(шифр і назва спеціальності)

Студент Бойко Ігор Віталійович

(прізвище та ініціали)

Керівник Митрофанов Павло Борисович

(прізвище та ініціали)

Полтава – 2025 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І РОБОТЕТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Спеціальність 122 «Комп'ютерні науки»

на тему

«Розробка і організаційно-технічні методи забезпечення тестового контролю знань в системі сучасної шкільної освіти»

Студент групи 602 ТН Бойко Ігор

Керівник роботи
кандидат технічних наук,
доцент Митрофанов П.Б.

Завідувач кафедри
к.ф.-м.н,
доцент Двірна О.А.

Полтава – 2025 рік

РЕФЕРАТ

Кваліфікаційна робота магістра: 100 с., 50 рисунків, 6 таблиці, 43 джерела.

Об'єкт дослідження: комп'ютерні системи тестування знань у сучасній освіті, зокрема платформи, що використовують штучний інтелект і системи захисту від списування.

Мета роботи: аналіз функціональних можливостей, архітектури та впровадження елементів штучного інтелекту в сучасні комп'ютерні системи тестування знань для підвищення ефективності оцінювання та забезпечення академічної доброчесності.

Методи: системний аналіз, порівняльний аналіз, статистичний аналіз результатів тестування, експертні оцінки, моделювання архітектури комп'ютерних систем, аналіз алгоритмів штучного інтелекту для адаптивного тестування та систем виявлення шахрайства.

Ключові слова: тестування знань, освітні платформи, адаптивні тести, штучний інтелект, системи захисту від списування, гейміфікація, аналітика, системи управління навчанням (LMS).

Результати виконання дипломної роботи були апробовані на Міжнародній науково-практичній конференції студентів, аспірантів та молодих вчених «Молодіжна наука: інновації та глобальні виклики»**,** 6 листопада 2024 року – Полтава: Полтавська політехніка.

А також на XVII Міжнародній науково-практичній конференції «Академічна й університетська наука: результати та перспективи» 12 – 13 грудня 2024 року – Полтава: Полтавська політехніка.

ABSTRACT

Master's thesis: 100 pages, 50 illustrations, 6 tables, 43 sources

Object of research: computer-based knowledge testing systems in modern education, specifically platforms utilizing artificial intelligence and anti-cheating systems.

Purpose of the work: to analyze the functional capabilities, architecture, and implementation of elements of artificial intelligence in modern computer-based knowledge testing systems to enhance assessment efficiency and ensure academic integrity.

Methods: system analysis, comparative analysis, statistical analysis of test results, expert evaluations, modeling of computer system architectures, analysis of artificial intelligence algorithms for adaptive testing and fraud detection systems.

Keywords: knowledge testing, educational platforms, adaptive tests, artificial intelligence, anti-cheating systems, gamification, analytics, learning management systems (LMS).

The results of the thesis were presented at the International scientific and practical conference of students, postgraduates and young scientists "Youth science: innovations and global challenges", November 6, 2024 – Poltava: Poltava Polytechnic.

And also, at the XVII International Scientific and Practical Conference "Academic and University Science: Results and Prospects" on December 12 – 13, 2024 – Poltava: Poltava Polytechnic.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект

LMS – система управління навчальною діяльністю (англ. Learning Management System)

IQ – кількісна оцінка рівня інтелекту людини (англ. Intelligence Quotient)

SAT – стандартизований тест для вступу до коледжів у США (англ. Scholastic Aptitude Test)

CAT – комп'ютерно-адаптивне тестування (англ. Computerized Adaptive Testing)

UI –дизайн інтерфейсу користувача (англ. User Interface)

VR – віртуальна реальність (англ.Virtual Reality)

AR – доповнена реальність (англ.Augmented Reality)

CRM – системи для управління відносинами з клієнтами (англ. Customer Relationship Management system)

ERP – системи для планування ресурсів підприємства (англ. Enterprise Resource Planning system)

GDPR – загальний регламент про захист даних (англ. General Data Protection Regulation)

ПЗ – програмне забезпечення.

БД – база даних.

API – інтерфейс програмування додатків (англ. Application Programming Interface).

ВСТУП

Сучасна освітня система активно трансформується під впливом технологічного прогресу, змін у суспільних цінностях і потребах ринку праці. Одним із ключових аспектів цієї трансформації є впровадження ефективних методів оцінювання знань, які дозволяють забезпечити об’єктивність, точність і прозорість. Тестовий контроль знань займає провідне місце серед таких методів, адже він дозволяє швидко й об’єктивно оцінювати рівень засвоєння матеріалу. Проте, існує низка викликів, пов’язаних із розробкою та впровадженням тестових систем, зокрема потреба у врахуванні індивідуальних особливостей учнів, адаптація до різних умов навчання, забезпечення академічної доброчесності, а також високої якості тестових завдань.

Актуальність теми дослідження обумовлена стрімким розповсюдженням дистанційної освіти та цифрових платформ. Застосування інноваційних підходів і технологій, таких як штучний інтелект, адаптивних алгоритмів та аналізу великих даних, відкриває нові можливості для підвищення ефективності цього процесу. Відмічені вище технології дозволяють не лише автоматизувати рутинні процеси, але й забезпечити індивідуальний підхід до кожного учня, що робить тестування не лише об'єктивним, але й гнучким та персоналізованим. Крім того, такі підходи сприяють створенню динамічних тестових завдань, які можуть змінюватися відповідно до рівня підготовки учня, що підвищує мотивацію до навчання. Таким чином, дослідження організаційно-технічних методів забезпечення тестового контролю знань у сучасній шкільній освіті є надзвичайно актуальним.

Мета дослідження полягає в розробці організаційно-технічних методів забезпечення тестового контролю знань, які відповідатимуть сучасним вимогам до якості, об’єктивності та ефективності освітнього процесу.

Завдання дослідження полягають в досягненні поставленої мети дослідження, а саме:

1. Провести аналіз існуючих методів тестового контролю знань у шкільній освіті.

2. Визначити основні проблеми та виклики, пов’язані з їх використанням.

3. Розробити рекомендації щодо вдосконалення тестових систем із використанням сучасних технологій.

4. Запропонувати організаційно-технічну модель впровадження тестового контролю в умовах цифровізації освіти та виконати програмну реалізацію з розробкою інтерфейсу користувача.

5. Оцінити ефективність запропонованих підходів на основі виконаного тестування та аналізу отриманих даних.

6. Провести порівняльний аналіз ефективності запропонованих підходів із традиційними методами тестування.

7. Створити рекомендації для педагогів щодо використання тестових систем у навчальному процесі.

Об’єктом дослідження є процес тестового контролю знань у сучасній шкільній освіті.

Предметом дослідження є організаційно-технічні методи забезпечення ефективності тестового контролю знань, враховуючи їхню адаптивність до змінних умов навчального процесу та специфіки учнівської аудиторії. Крім того, предмет дослідження включає вивчення можливостей впровадження елементів штучного інтелекту для створення індивідуалізованих навчальних траєкторій.

Наукова новизна дослідження полягає у розробці нових організаційно-технічних підходів до забезпечення тестового контролю знань, які враховують сучасні вимоги до адаптивності, інтерактивності та аналізу великих даних. Зокрема, запропоновано та програмно реалізовано інтерфейс, з інтегрованими технологіями штучного інтелекту для автоматизації процесу створення та

оцінювання тестових завдань. Особливістю цієї моделі є можливість її масштабування та використання для різних освітніх рівнів, починаючи з початкової школи і закінчуючи вищою освітою. Крім того, модель враховує необхідність регулярного оновлення тестових завдань відповідно до змін у навчальних програмах і потребах суспільства.

Практична значущість роботи полягає у можливості впровадження розроблених методів комп'ютерного тестування знань та інтеграції засобів виявлення та контролю випадків порушення академічної доброчесності під час процесу оцінювання знань учнів. Запропоновані рекомендації дозволять підвищити якість освітнього процесу, забезпечити індивідуальний підхід до учнів та оптимізувати витрати часу й ресурсів на оцінювання знань. Крім того, результати дослідження можуть бути використані при створенні цифрових платформ для дистанційного навчання. Це особливо важливо в умовах швидкої цифровізації освітньої сфери, коли школи та університети стикаються з необхідністю інтеграції нових технологій у повсякденну практику. Зокрема, розроблені рекомендації сприятимуть формуванню навичок самостійного навчання в учнів, що є важливим аспектом у підготовці до життя в умовах сучасного інформаційного суспільства.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ТЕСТУВАННЯ ЗНАНЬ

1.1. Історія розвитку методів тестування знань

Тестування знань як один із методів оцінювання розвивалось задовго до запровадження сучасних технологій. Ще у XIX столітті закладалися основи системного підходу до оцінювання знань, спочатку у персональних зустрічах, а згодом і у групових системах. Вже тоді виникала потреба в об'єктивному оцінюванні результатів навчання, що згодом стало основою для стандартизації.

Перші спроби систематичного оцінювання знань можна віднести до кінця XIX століття, коли у сфері освіти та психології почали використовуватися стандартизовані методики. Одним із піонерів у цій галузі був Джеймс Маккін Кеттелл, який у 1890 році вперше запровадив поняття "ментальних тестів" [1]. Ці ранні розробки здебільшого оцінювали реакції, увагу та пам'ять. Успіх цих тестів відкрив нові горизонти для вивчення людських когнітивних здібностей і став підґрунтям для подальших наукових досліджень.

Алфред Біне, французький психолог, у 1905 році розробив перший шкальний психометричний тест. Разом зі своїм колегою Теодором Сімоном вони створили тест, який мав на меті оцінювати інтелектуальні здібності дітей [2]. Основна мета полягала в ідентифікації дітей, які потребували додаткової підтримки у навчанні. Їхня методика передбачала використання завдань, спрямованих на оцінку логічного мислення, мовленнєвих і математичних здібностей. Цей тест став прототипом для сучасних IQ-тестів, що використовуються донині.

У Сполучених Штатах активну роль у розвитку тестування відіграв Ралф Транддайкер. Його роботи у 1910-1920-х роках заклали теоретичну базу для

педагогічного тестування [3]. Він розробив ідеї про об'єктивність тестування, які значною мірою вплинули на стандарти освітньої оцінки. У цей період також з'явилися тести для оцінювання академічних знань, такі як SAT (Scholastic Aptitude Test), який вперше був використаний у 1926 році. SAT став важливим інструментом для вступу до вищих навчальних закладів, адже він забезпечував об'єктивну оцінку здібностей абітурієнтів [4].

Під час Другої світової війни стандартизовані тести знайшли своє застосування у військовій сфері. Тести використовувались для оцінювання здібностей новобранців та їхнього розподілу за відповідними завданнями. Цей досвід показав ефективність і швидкість масового тестування, що сприяло його подальшому впровадженню у сферу освіти. Наприклад, Army General Classification Test дозволяв швидко визначати рівень знань і навичок солдатів, а також їхню придатність до виконання певних обов'язків [5].

До 1940-х років тести стали стандартною частиною освітніх програм багатьох країн. Їхнє застосування дозволяло швидко та об'єктивно оцінювати знання великих груп учнів. Водночас виникла необхідність удосконалення тестових методів для врахування культурних, соціальних та психологічних особливостей учасників. Це стало поштовхом для розвитку таких понять, як валідність, надійність та адаптивність тестів. Валідність тестів забезпечувала відповідність тестових завдань їхній меті, а надійність гарантувала стабільність отриманих результатів.

З розвитком комп’ютерних технологій у кінці XX століття тестові системи стали основою для онлайн-освіти та дистанційного навчання. У 1980-х роках з'явилися перші комп'ютеризовані системи тестування, які значно підвищили точність і швидкість оцінювання [6]. Наприклад, комп'ютерно-адаптивне тестування (CAT) дозволяє адаптувати складність питань залежно від відповідей користувача, що забезпечує більш точну оцінку знань. CAT став проривом у сфері освітнього тестування, адже його алгоритми дозволяють зменшити кількість питань без втрати точності.

Сучасні системи тестування базуються на передових технологіях, таких як штучний інтелект (ШІ) і машинне навчання (МН). Вони дозволяють враховувати індивідуальні особливості кожного учасника тестування та забезпечувати інтерактивність процесу. Наприклад, платформи для онлайн-освіти, такі як Coursera [8] чи Duolingo [7] (рис.1.1), активно використовують ці технології для підвищення ефективності навчання. Інтерактивні тести включають завдання, які сприяють активному залученню учнів, наприклад, через симуляції чи рольові ігри.

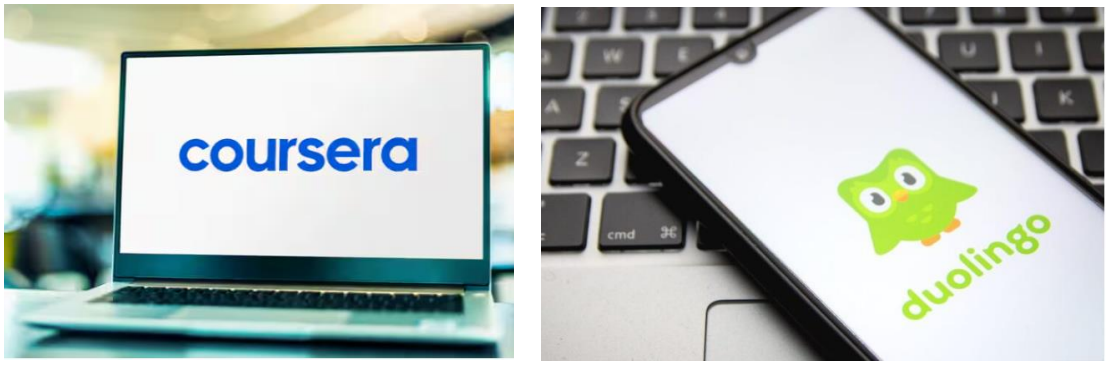

Рисунок 1.1 – Приклади платформ для онлайн-освіти

Також сучасні технології дозволяють проводити аналіз великих масивів даних, отриманих під час тестування, що відкриває можливості для постійного вдосконалення тестових завдань [9]. Наприклад, системи ШІ можуть автоматично виявляти та виправляти недоліки у формулюванні питань, забезпечуючи їхню більшу точність та зрозумілість.

Інший важливий аспект розвитку тестових систем — це їхня адаптація до культурного контексту та специфіки різних регіонів [10]. Для прикладу, в азійських країнах було запроваджено методики, які враховують традиційні освітні підходи та особливості мислення учнів. У Європі акцент зроблено на розвиток критичного мислення та здатності до аналізу, що знаходить відображення у відповідних форматах тестування.

Значну роль у вдосконаленні тестування відіграє розвиток мобільних технологій. Сьогодні тести можна проходити за допомогою смартфонів та планшетів, що робить процес більш доступним та зручним. Це також сприяє

розширенню аудиторії користувачів тестових систем, особливо у віддалених регіонах, де доступ до традиційної освіти може бути обмеженим.

Таким чином, історія розвитку методів тестування знань демонструє еволюцію від простих експериментів до високотехнологічних систем, які забезпечують точність, адаптивність і зручність у використанні. Від перших "ментальних тестів" до сучасних адаптивних платформ, тестування знань стало невід'ємною частиною освітнього процесу та наукових досліджень, сприяючи підвищенню якості навчання та забезпечуючи рівний доступ до освіти для людей з різними потребами та можливостями.

1.2. Сучасні методи оцінювання знань: переваги та недоліки

Сучасні методи оцінювання знань відіграють ключову роль у розвитку освітнього процесу. Вони спрямовані на забезпечення об’єктивності, точності та ефективності оцінювання результатів навчання. З появою нових технологій та методологій ця галузь зазнала значних змін, що дозволило інтегрувати новітні досягнення науки в освітню практику (рис.1.2). Проте кожен метод має як переваги, так і недоліки, які необхідно враховувати під час їхнього вибору та впровадження [12, 14].

Рисунок 1.2 – Приклади сучасних методів оцінювання знань

Перший сучасний метод оцінювання знань – це стандартизовані тести. Перевагами цього методу є об’єктивність, швидкість оцінювання та можливість масового застосування. Стандартизовані тести дозволяють забезпечити однакові умови для всіх учасників, що мінімізує суб’єктивний вплив на результати. Крім того, вони полегшують порівняння результатів на міжрегіональному або міжнародному рівнях, що є важливим для моніторингу ефективності освітніх систем. Недоліками таких тестів є їхня складність у створенні, адже необхідно забезпечити валідність, надійність та репрезентативність тестових завдань. Часто ці тести зосереджені на відтворенні фактичної інформації, що обмежує їхній потенціал для оцінювання критичного мислення. Крім того, цей метод часто критикують за відсутність гнучкості, адже він не враховує індивідуальні особливості учнів.

Другий популярний метод – це проєктна робота. Цей підхід дозволяє оцінити не лише теоретичні знання, але й практичні навички, творчий потенціал і здатність до самостійного мислення. Серед переваг проєктної роботи варто зазначити розвиток критичного мислення та мотивацію до навчання. Учні можуть проявити ініціативу, працюючи над реальними проблемами або моделюючи життєві ситуації. Такий підхід стимулює їхню допитливість і зацікавленість у навчанні. Водночас цей метод має недоліки: оцінювання проєктів є суб’єктивним і залежить від кваліфікації викладача, а також потребує значних витрат часу. Проєкти можуть бути занадто складними для учнів, які не мають достатнього рівня підготовки чи ресурсів. Проте, ми можемо звернутися до рейтингового способу оцінювання проектів, запропонованим науковим діячем В. В. Голобородьком [36]. Зміст полягає в тому, що автор проекту, учні та вчитель заповнюють анкету під час захисту, використовуючи для цього 11 критеріїв із трьома рівнями оцінок (5, 10, 20 балів) (Таблиця 1.1). Остаточна оцінка обчислюється як середнє арифметичне колективної оцінки, вчительської та самооцінки.

Таблиця 1.1 – Оцінювання проектної роботи (за [36])

Оцінка етапів	**Критерії оцінки**	**Бали**
Оцінка роботи	Актуальність теми і рішень, їхня складність	5, 10, 20
	Обсяг розробок і кількість рішень	5, 10, 20
	Практична цінність	5, 10, 20
	Рівень самостійності учасників	5, 10, 20
	Якість оформлення записів, наочності тощо	5, 10, 20
	Оцінка проекту рецензентом	5, 10
Оцінка захисту	Якість доповіді	5, 10, 20
	Глибина розкриття теми	5, 10, 20
	Глибина розкриття предмета	5, 10, 20
	Відповіді на питання журі	5, 10
	Відповіді на питання аудиторії (дискусія)	5, 10

Метод самостійного тестування, або self-assessment, стає дедалі популярнішим завдяки інтеграції цифрових платформ у навчальний процес. Цей підхід дозволяє учням самостійно оцінювати свої знання, отримувати зворотний зв’язок і коригувати власний навчальний план [18]. Серед переваг цього методу – розвиток навичок самоконтролю та підвищення відповідальності за результати навчання. Учні вчаться аналізувати свої сильні та слабкі сторони, що сприяє їхньому самовдосконаленню. Недоліками є можливість необ’єктивного оцінювання через низький рівень самосвідомості учня або недостатню точність автоматизованих систем. Окрім того, відсутність зовнішнього контролю може призводити до недооцінювання важливості регулярної перевірки знань.

Сучасні цифрові платформи, такі як адаптивне тестування, забезпечують індивідуалізований підхід до оцінювання знань. Ці системи використовують алгоритми ШІ для налаштування складності завдань відповідно до рівня підготовки учня. Перевагами адаптивного тестування є гнучкість, економія часу та точність оцінювання. Такі системи створюють персоналізоване середовище навчання, яке дозволяє зосередитися на слабких місцях учня. Проте вартість розробки таких систем є високою, що робить їх недоступними для багатьох освітніх закладів. Крім того, складність інтеграції цих платформ

у традиційний освітній процес часто вимагає додаткового навчання для викладачів.

Технології доповненої реальності (AR) та віртуальної реальності (VR) відкривають нові можливості для оцінювання знань у інтерактивному середовищі. Вони дозволяють моделювати реальні ситуації, що потребують застосування теоретичних знань на практиці. Це особливо корисно для природничих наук і професійної освіти, таких як медицина, інженерія або архітектура. Завдяки цим технологіям учні можуть взаємодіяти з віртуальними об’єктами, що робить процес навчання більш захопливим і практично орієнтованим. Однак впровадження таких технологій потребує значних фінансових і технічних ресурсів. Більше того, використання AR і VR вимагає наявності відповідної інфраструктури та обладнання, що обмежує їх доступність для деяких навчальних закладів. Окрім цього, як зазначає доктор педагогічних наук Хміль Н.А. [37], треба приділити окрему увагу підготовці викладачів до таких технологій та всебічної взаємодії з ними (рис. 1.3).

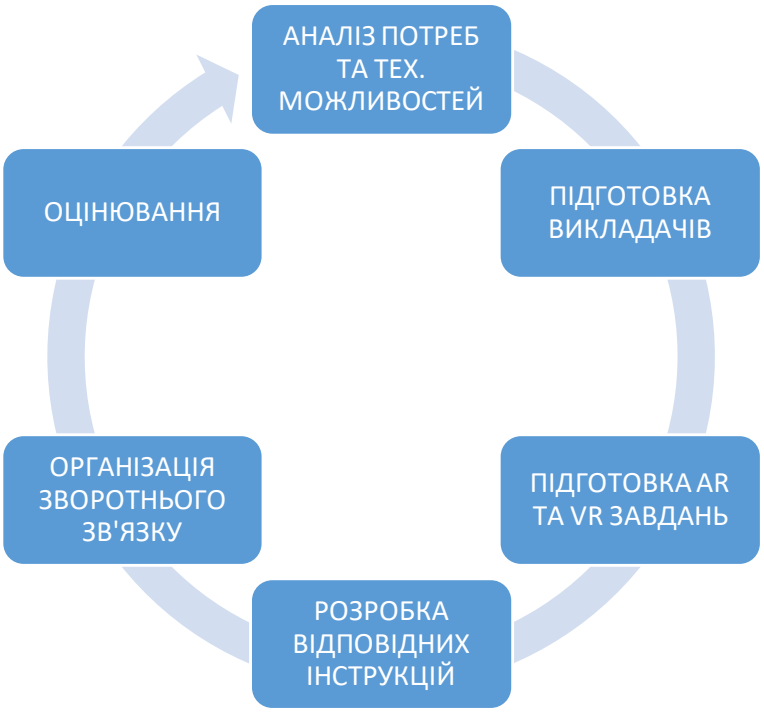

Рисунок 1.3 – Процедура реалізації технологій AR та VR для оцінювання знань учнів

Гейміфікація [16] є ще одним інноваційним підходом до оцінювання знань. Цей метод використовує ігрові елементи для залучення учнів і підвищення їхньої мотивації. Серед переваг гейміфікації – можливість створення цікавого та інтерактивного навчального середовища, яке стимулює учнів до активної участі [38] (рис. 1.4). Недоліки полягають у тому, що надмірна гейміфікація може знижувати серйозність навчального процесу, а розробка якісних гейміфікованих завдань вимагає значних зусиль і ресурсів.

Таким чином, сучасні методи оцінювання знань пропонують широкий спектр інструментів, які дозволяють забезпечити об’єктивність і точність освітнього процесу [13]. Вибір методу залежить від багатьох факторів, зокрема цілей навчання, наявних ресурсів і специфіки навчального контингенту. Балансування між перевагами та недоліками кожного методу є ключовим завданням для розробників освітніх програм. Інтеграція новітніх технологій у традиційні підходи до оцінювання дозволяє створити більш гнучку, адаптивну та ефективну систему навчання, яка відповідає викликам сучасного світу.

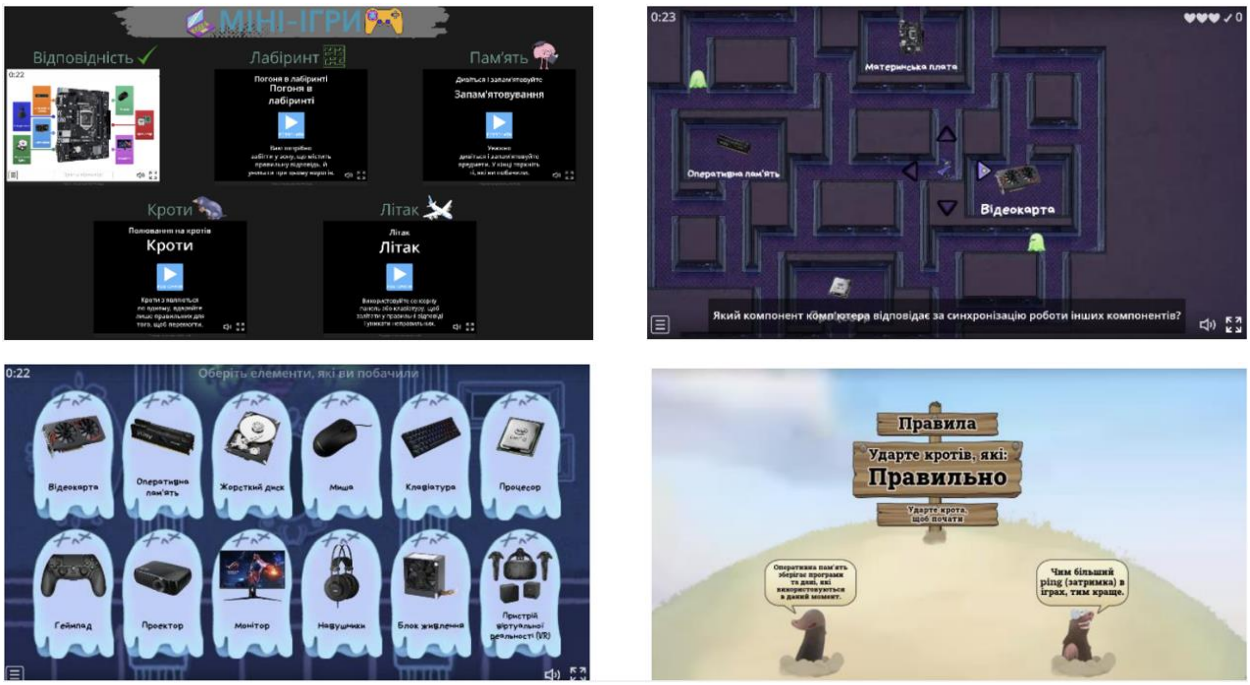

Рисунок 1.4 - Комплекс міні-ігор для вивчення комплектуючих персонального комп’ютера

1.3. Психолого-педагогічні аспекти тестування

Психолого-педагогічні аспекти тестування відіграють важливу роль у забезпеченні ефективності процесу оцінювання знань. Тестування як інструмент оцінювання не лише спрямоване на перевірку академічних досягнень, але й значною мірою впливає на психологічний стан учнів, їхню мотивацію до навчання та формування навчальних навичок. Урахування цих аспектів дозволяє створити сприятливе середовище для навчання та розвитку особистості [21].

Першим важливим аспектом є вплив тестування на мотивацію учнів. Добре сплановані та структуровані тести можуть стимулювати інтерес до навчання, сприяти формуванню навичок самоконтролю та відповідальності [18]. Вони мотивують учнів ставити перед собою чіткі цілі та працювати над їх досягненням. Водночас неякісні чи занадто складні тести можуть викликати тривожність, страх перед невдачею та знижувати впевненість у власних силах. Згідно з дослідженнями, учні, які отримують позитивний досвід тестування, краще адаптуються до академічних викликів і демонструють вищий рівень мотивації. Особливо важливо враховувати роль похвали та конструктивної критики у формуванні ставлення до тестування [17].

Другий важливий аспект – це психологічна готовність учнів до тестування. Для ефективного використання тестів необхідно враховувати вікові та індивідуальні особливості учнів, їхній когнітивний розвиток, рівень стресостійкості та емоційного інтелекту [14]. Наприклад, молодші школярі можуть потребувати більш гнучких форм тестування, які враховують їхню низьку концентрацію уваги. У старшій школі важливо застосовувати завдання, що сприяють розвитку аналітичного та критичного мислення. Також врахування попереднього досвіду тестування учнів допомагає уникнути надмірного стресу.

Третій аспект пов’язаний із педагогічною роллю тестування [20]. Тести повинні виконувати не лише контрольну, але й навчальну функцію. Для цього

важливо надавати учням детальний зворотний зв'язок, що допомагає їм зрозуміти свої помилки та планувати подальший навчальний процес. Наприклад, адаптивні системи тестування на базі Moodle (рис. 1.5) можуть автоматично генерувати рекомендації щодо покращення навичок [40] або повторення матеріалу. Крім того, тестові завдання можуть бути інтегровані в навчальний процес як елементи активного навчання, що підвищує зацікавленість учнів у тематиці [41].

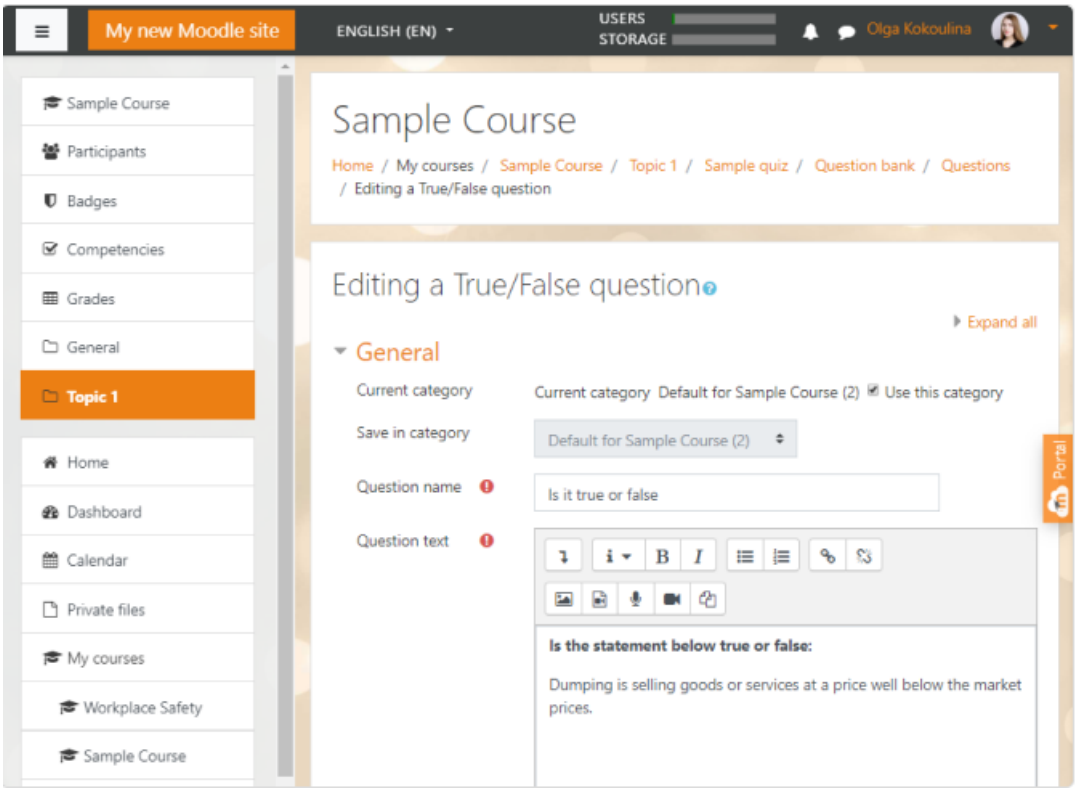

Рисунок 1.5 – Приклад інтерфейсу системи Moodle

Важливим є також вплив формулювання тестових завдань на сприйняття інформації. Педагогічно коректні тести повинні бути чіткими, зрозумілими та відповідати рівню підготовки учнів. Неоднозначні формулювання чи надто складні запитання можуть викликати плутанину і спотворювати результати тестування. Тому необхідно проводити ретельну перевірку тестових завдань перед їх використанням. Важливо також забезпечити різноманітність форм завдань, що сприяє розвитку різних когнітивних навичок.

Крім того, необхідно враховувати культурні та соціальні аспекти [**Помилка! Джерело посилання не знайдено.**]. Тести повинні бути адаптовані до контексту, у якому навчаються учні, і не містити упереджень чи стереотипів. Наприклад, завдання, які вимагають знань про специфічні культурні явища, можуть бути несправедливими для учнів із іншого середовища. Розробка завдань із урахуванням багатокультурного підходу сприяє створенню інклюзивного освітнього середовища.

Останній важливий аспект стосується використання результатів тестування. Вони повинні слугувати основою для індивідуалізації навчання, а не бути виключно інструментом оцінки успішності [15]. Наприклад, результати тестів можна використовувати для визначення сильних і слабких сторін учнів та планування додаткових занять або консультацій. Регулярний моніторинг результатів дозволяє відслідковувати прогрес учнів і вчасно коригувати навчальні стратегії.

Значну увагу також варто приділяти професійному розвитку вчителів, які займаються розробкою та використанням тестових завдань. Педагоги повинні володіти сучасними методиками тестування та враховувати психологічні аспекти в роботі з учнями. Це забезпечує якість і ефективність тестового процесу.

Рисунок 1.6 – Складові частини успішного тестувального процесу

Таким чином, психолого-педагогічні аспекти тестування є невід'ємною частиною освітнього процесу (рис. 1.6). Вони забезпечують гармонійне поєднання контролю знань із підтримкою мотивації, розвитку учнів і вдосконалення навчального процесу. Урахування цих аспектів сприяє досягненню високих результатів навчання та створенню сприятливих умов для розвитку освітньої системи.

1.4. Класифікація тестів: типи та їх застосування

Класифікація тестів є важливим кроком у розумінні їхньої структури, призначення та застосування в різних освітніх контекстах [15]. Існує кілька основних типів тестів (рис.1.7), кожен з яких має свої особливості та підходи до використання. Кожен із цих тестів виконує унікальну функцію, спрямовану на оцінювання різних аспектів знань і навичок учнів [17].

Перший тип – це тести із закритими питаннями. У таких тестах учні обирають правильну відповідь із запропонованих варіантів. Ці тести є найбільш поширеними завдяки своїй зручності у використанні та можливості швидкої перевірки. Вони ефективні для оцінки знань із конкретних тем і забезпечують об'єктивність оцінювання. До переваг цього типу тестів належить те, що вони дозволяють автоматизувати процес перевірки, що значно економить час. Проте обмеженням є складність оцінювання навичок критичного мислення чи творчих здібностей учнів. Наприклад, учень може вгадати правильну відповідь, що не завжди відображає реальний рівень знань.

Другий тип – тести з відкритими питаннями. Учні мають самостійно формулювати відповіді на поставлені запитання. Цей формат сприяє розвитку аналітичного мислення, вміння висловлювати думки та структурувати інформацію. Наприклад, запитання на кшталт "Опишіть основні причини кліматичних змін" вимагають від учня глибокого розуміння теми. Водночас

перевірка таких тестів потребує більше часу та може бути суб'єктивною. Для уникнення суб'єктивності важливо розробляти чіткі критерії оцінювання.

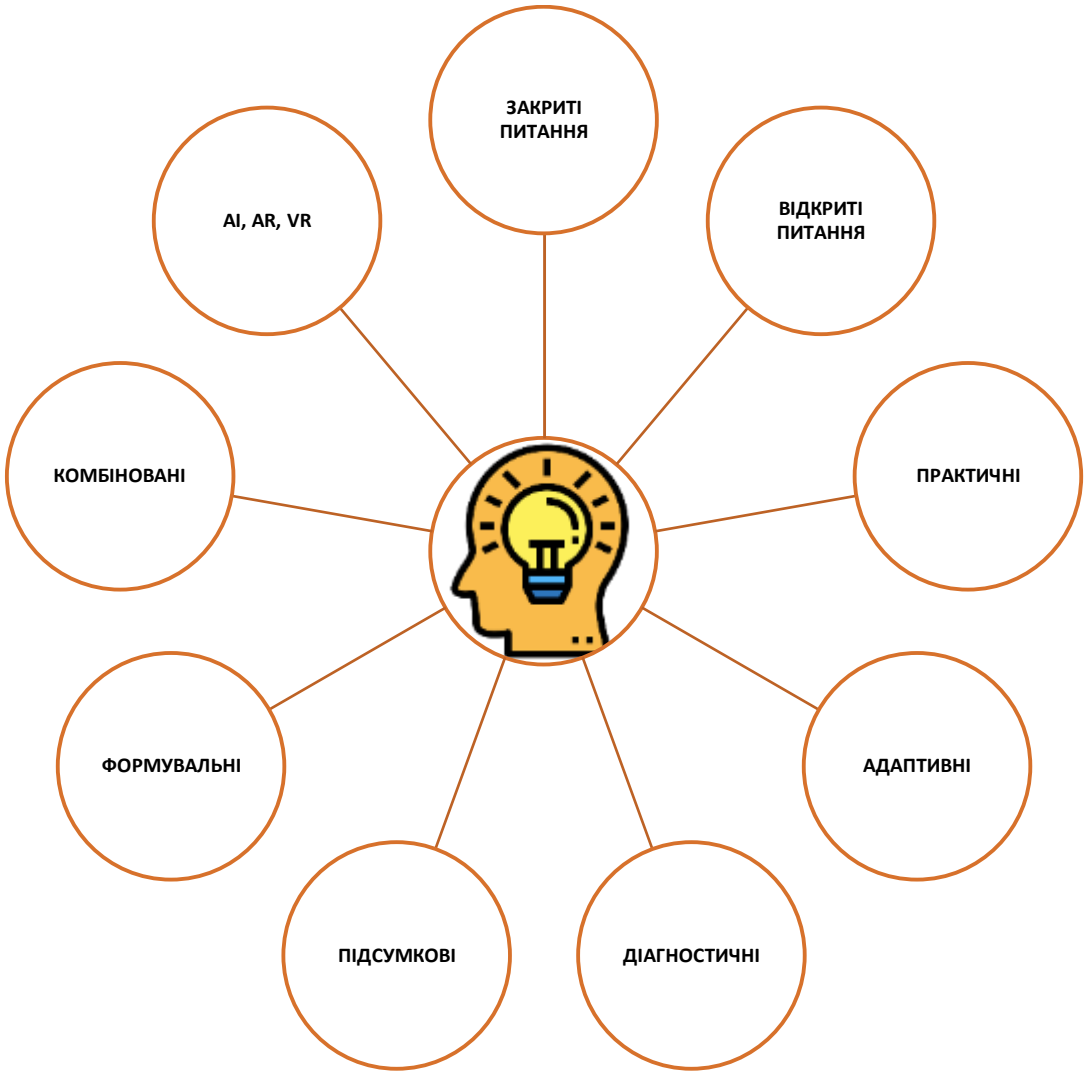

Рисунок 1.7 – Типи тестів

Третій тип – практичні або лабораторні тести. Вони використовуються для оцінки практичних навичок учнів у конкретних галузях, таких як хімія, фізика або біологія. Завдання можуть включати виконання експериментів, роботу з обладнанням або аналіз результатів. Цей тип тестів дозволяє оцінити здатність учнів застосовувати теоретичні знання на практиці. Наприклад, лабораторний тест може вимагати від учня приготувати хімічний розчин або провести фізичний експеримент. Незважаючи на їхню ефективність, такі тести є ресурсоємними та вимагають доступу до спеціалізованого обладнання.

Четвертий тип – адаптивні тести [23]. Вони використовують технології штучного інтелекту для налаштування складності завдань відповідно до рівня підготовки учня [16]. Адаптивні тести є надзвичайно ефективними для індивідуалізованого навчання, адже дозволяють швидко визначити сильні та слабкі сторони учня і адаптувати навчальний процес (рис. 1.8, 1.9). Наприклад, якщо учень правильно відповідає на запитання середньої складності, система може запропонувати складніше завдання [39]. Крім того, адаптивні тести дозволяють скоротити час оцінювання без втрати точності. Однак їх впровадження потребує значних фінансових та технологічних ресурсів.

Рисунок 1.8 – Схема проходження завдань при постійній адаптації (Х – хибна відповідь, П – правильна відповідь)

Рисунок 1.9 – Схема проходження завдань при блоковій адаптації
(Х – хибна відповідь, П – правильна відповідь)

П’ятий тип – діагностичні тести [21]. Їхня мета полягає у виявленні рівня знань учнів на початку навчального процесу або перед вивченням нової теми. Вони дозволяють визначити прогалини в знаннях і спрямувати навчання на їхнє усунення. Такі тести особливо важливі для початкових етапів навчання, коли необхідно оцінити базові знання та навички учнів. Наприклад, діагностичний тест із математики може показати, наскільки добре учні володіють основами арифметики, що є важливим для подальшого вивчення алгебри.

Шостий тип – підсумкові тести [9]. Вони проводяться наприкінці навчального курсу або модуля з метою оцінки загального рівня засвоєння матеріалу. Такі тести є важливим інструментом для оцінки ефективності навчального процесу. Їх результати використовуються для підведення підсумків і визначення успішності виконання навчальної програми. Наприклад, підсумковий тест із історії може включати як закриті, так і відкриті запитання для перевірки знань із ключових тем курсу.

Сьомий тип – формувальні тести [19]. Вони використовуються впродовж навчання для моніторингу прогресу учнів і коригування навчальних стратегій. Формувальні тести забезпечують зворотний зв’язок і сприяють активному залученню учнів у процес навчання. Їхнє основне завдання – виявлення прогресу учнів і надання їм рекомендацій для подальшого розвитку [14]. Наприклад, учитель може використовувати формувальні тести для перевірки розуміння окремих тем і визначення тих, які потребують додаткового пояснення.

Окрім зазначених типів, існують також комбіновані тести [22], які поєднують елементи різних форматів. Наприклад, тест може містити як закриті, так і відкриті питання, а також практичні завдання. Це дозволяє більш комплексно оцінювати знання та навички учнів. Наприклад, комбінований

тест із хімії може включати розрахункові завдання, питання на вибір правильної відповіді та лабораторний експеримент.

Також до сучасних підходів належать тести із використанням технологій віртуальної реальності (VR) (рис. 1.10) та доповненої реальності (AR). Вони дозволяють створювати інтерактивні завдання, що моделюють реальні ситуації. Наприклад, у медичній освіті VR-тести можуть моделювати проведення операцій, а в інженерії – тестування механізмів у віртуальному середовищі. Хоча такі тести є дорогими у впровадженні, вони значно підвищують практичну цінність оцінювання.

Рисунок 1.10 – Приклад використання технології VR у навчанні

Кожен із цих типів тестів має своє унікальне призначення і сферу застосування. Їхнє правильне використання сприяє ефективному контролю знань і розвитку учнів, забезпечуючи досягнення освітніх цілей. Залежно від завдань, які ставляться перед тестуванням, і специфіки аудиторії, вибір відповідного типу тесту є ключовим для досягнення високої якості оцінювання.

1.5. Критерії якості тестових завдань та їх оцінювання

Критерії якості тестових завдань відіграють важливу роль у забезпеченні ефективності та точності оцінювання знань учнів (рис.1.11). Якісні тестові завдання сприяють об’єктивності результатів і дозволяють точно виміряти рівень знань, умінь і навичок учасників тестування. Визначення цих критеріїв допомагає забезпечити відповідність тестів меті оцінювання та освітнім стандартам. Кожен із цих критеріїв є основою для забезпечення достовірності та справедливості тестового процесу.

Рисунок 1.11 – Критерії якості тестових завдань

Першим і найважливішим критерієм є валідність тестових завдань. Валідність відображає, наскільки тест відповідає цілям і завданням оцінювання [13]. Вона вказує на те, чи вимірює тест саме ті знання і навички, які потрібно оцінити. Наприклад, тест із математики повинен оцінювати математичні навички, а не здатність до запам’ятовування фактів. Існують різні види валідності: змістова валідність забезпечує відповідність змісту тесту навчальним цілям, конструктивна валідність оцінює те, чи відповідають завдання теоретичним концепціям, а критерійна валідність визначає зв’язок між результатами тесту і реальними досягненнями учнів. Впровадження

комплексного підходу до забезпечення валідності дозволяє уникнути спотворення результатів.

Другий важливий критерій – це надійність. Надійність тесту означає стабільність і послідовність результатів при повторному проходженні тесту за однакових умов [17]. Якщо результати одного й того самого тесту значно відрізняються в різний час, це свідчить про низьку надійність. Забезпечення надійності потребує ретельного підбору завдань, уникнення неоднозначних формулювань і перевірки тесту на репрезентативній вибірці учнів. Крім того, важливим інструментом є статистичний аналіз результатів, який дозволяє виявити завдання з низькою дискримінативністю.

Третім критерієм якості є об’єктивність. Об’єктивність тесту визначається відсутністю впливу суб’єктивних факторів, таких як упередженість екзаменатора або неякісно сформульовані завдання [14]. Наприклад, використання тестів із закритими питаннями знижує ризик суб’єктивної оцінки, оскільки правильна відповідь чітко визначена заздалегідь. Також об’єктивність можна забезпечити за допомогою автоматизованих систем перевірки, які виключають людський фактор у процесі оцінювання.

Четвертий критерій – це рівень складності тестових завдань. Тести повинні мати оптимальний баланс між легкими, середніми та складними завданнями. Завдання, які є надто простими, не стимулюють учнів до мислення, тоді як надмірно складні завдання можуть викликати тривожність і знизити впевненість у власних силах. Оптимальний рівень складності дозволяє отримати достовірну інформацію про рівень знань учасників тестування. Наприклад, у стандартизованих тестах рекомендується дотримуватися пропорції: 20% легких, 60% середніх і 20% складних завдань.

П’ятий критерій – це діагностична цінність завдань. Діагностична цінність визначає, наскільки завдання допомагають виявити сильні та слабкі сторони учнів. Завдання повинні бути сформульовані таким чином, щоб результати тестування надавали викладачам і учням корисну інформацію для

подальшого навчання та вдосконалення знань. Наприклад, завдання з багатоваріантними відповідями можуть містити варіанти, які вказують на типові помилки учнів, що дає змогу педагогу аналізувати недоліки в засвоєнні матеріалу.

Шостий критерій – це адаптивність. Адаптивність означає можливість тестових завдань підлаштовуватися під рівень підготовки учнів. Наприклад, адаптивні тести можуть пропонувати завдання різної складності залежно від правильності відповідей на попередні питання. Це дозволяє підвищити точність оцінювання та зменшити стрес для учнів. Такі підходи є особливо важливими в умовах дистанційного навчання, де індивідуалізація навчання є ключовим фактором успішності.

Сьомий критерій – це зрозумілість формулювання завдань. Завдання повинні бути чіткими, логічними і зрозумілими для учнів. Неоднозначні формулювання або складна мова можуть спотворювати результати тестування, адже учні можуть неправильно зрозуміти питання. Використання зрозумілої мови та уникнення технічних термінів без пояснення сприяє точності оцінювання.

Крім того, важливим критерієм є відповідність тестових завдань стандартам освіти [**Помилка! Джерело посилання не знайдено.**]. Завдання повинні відповідати навчальним планам і програмам, а також враховувати вимоги до знань і навичок, що визначені для певного освітнього рівня. Наприклад, тест із фізики для старшокласників має включати завдання, що відповідають їхньому рівню підготовки і темам, які вивчалися під час занять. Відповідність стандартам забезпечує зв’язок між тестуванням і реальними потребами навчання.

Для оцінювання якості тестових завдань використовується кілька підходів. Один із них – це статистичний аналіз результатів тестування. Аналіз успішності виконання завдань допомагає виявити занадто легкі чи складні завдання, а також ті, що не корелюють із загальними результатами тесту [23]. Наприклад, коефіцієнт дискримінативності дозволяє визначити, наскільки

завдання розрізняє учнів із різним рівнем знань. Інший підхід – це експертна оцінка, коли завдання аналізуються фахівцями в галузі освіти. Експерти оцінюють відповідність завдань критеріям якості, а також їхню практичну доцільність.

Також важливо враховувати технологічні аспекти оцінювання якості. Використання автоматизованих систем для аналізу відповідей учнів значно спрощує процес перевірки та дозволяє швидко отримати зворотний зв’язок. Сучасні програмні інструменти можуть аналізувати великі обсяги даних, визначати тенденції у відповідях і генерувати рекомендації для подальшого вдосконалення тестових завдань.

Таким чином, критерії якості тестових завдань є основою для створення ефективних інструментів оцінювання. Дотримання цих критеріїв дозволяє забезпечити об’єктивність, валідність і надійність тестів, а також сприяє досягненню високих результатів навчання. Комплексний підхід до оцінювання якості завдань дозволяє зробити тестування прозорим і сприятливим для учнів, водночас підвищуючи ефективність освітнього процесу.

РОЗДІЛ 2
АНАЛІЗ ІСНУЮЧИХ КОМП'ЮТЕРНИХ СИСТЕМ ТЕСТУВАННЯ ЗНАНЬ

2.1. Огляд популярних систем тестування знань

Комп'ютерні системи тестування знань є важливим інструментом у сучасній освіті. Вони забезпечують автоматизацію процесу оцінювання, зменшують витрати часу на перевірку знань і дозволяють проводити тестування у великому масштабі. Ці системи пропонують широкий спектр можливостей, починаючи від базових тестів із закритими питаннями і закінчуючи складними адаптивними платформами для оцінювання навичок та компетенцій. У цьому підрозділі розглядаються популярні системи тестування, їхні особливості, переваги та обмеження.

Однією з найпоширеніших платформ є Moodle, відкрита система управління навчанням (LMS). Moodle (рис.2.1, 2.2) дозволяє створювати курси, інтегрувати тести з різними форматами завдань і автоматично оцінювати відповіді. Система підтримує різні типи завдань, зокрема питання з вибором правильної відповіді, відкриті питання, завдання на відповідність та есе [27]. Платформа Moodle широко використовується в університетах завдяки своїй гнучкості. Перевагою є її безкоштовний характер і можливість інтеграції з іншими подібними за будовою системами. Проте недоліками є складність налаштування для користувачів без технічних знань та висока потреба в адмініструванні.

Під час навчання в Полтавській політехніці я активно використовував цю платформу. Вона є зручною при роботі як з лекційним і лабораторно-практичним матеріалом так і під час проходження тестових завдань при здачі диференційованих заліків та іспитів по курсам дисциплін.

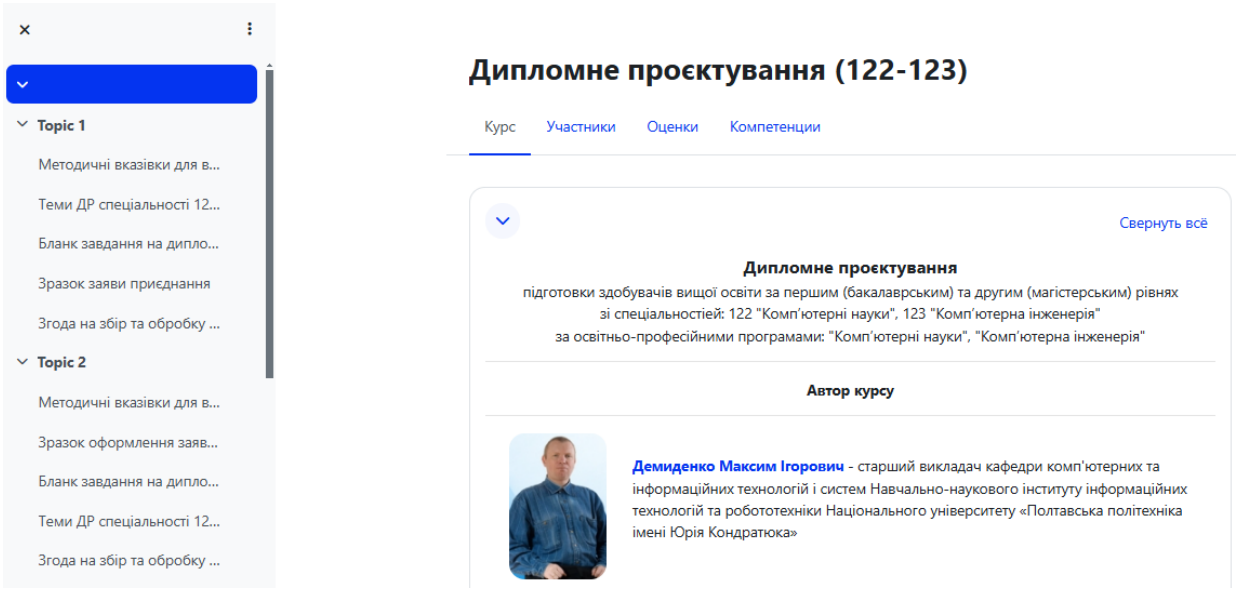

Рисунок 2.1 – Інтерфейс системи Moodle

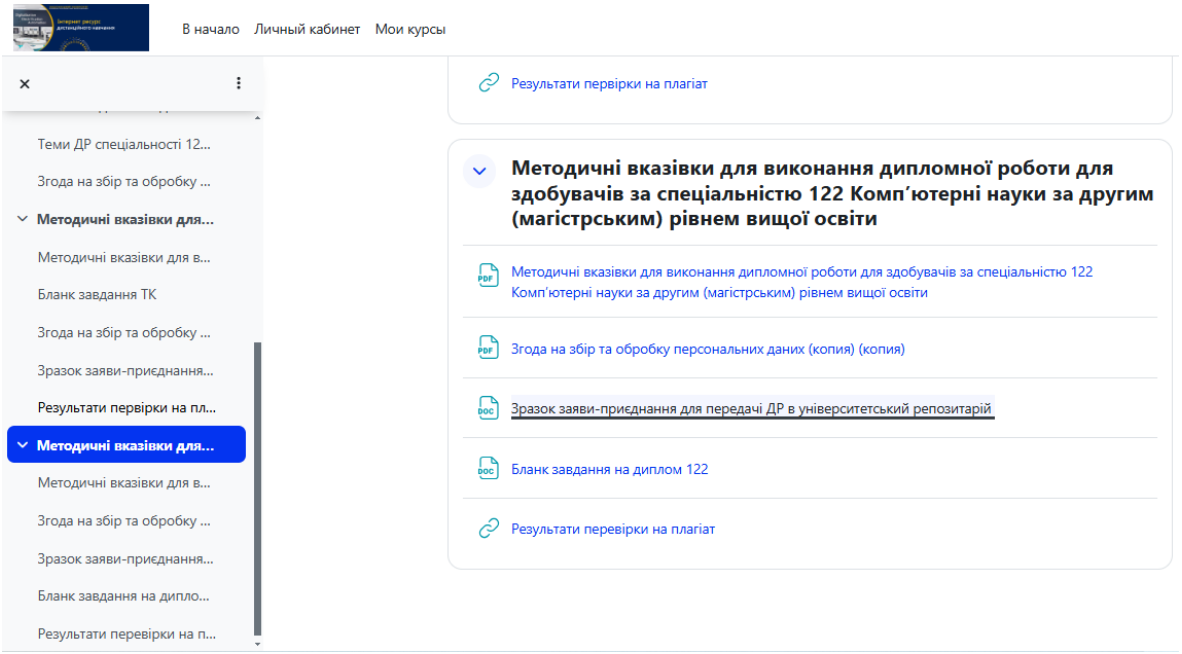

Рисунок 2.2 – Загальний вигляд сторінки курсу для студента в системі Moodle

Ще однією популярною платформою є Google Forms (рис.2.3), яка широко використовується для швидкого створення тестів. Ця платформа проста у використанні, підтримує автоматичну перевірку відповідей і є безкоштовною. Google Forms підходить для базових тестів і швидких опитувань, особливо в шкільних умовах [28]. Однак обмеженням є відсутність

інтерактивності та складних механізмів аналізу даних. Наприклад, вона не дозволяє створювати адаптивні тести або аналізувати навчальний прогрес учнів детально.

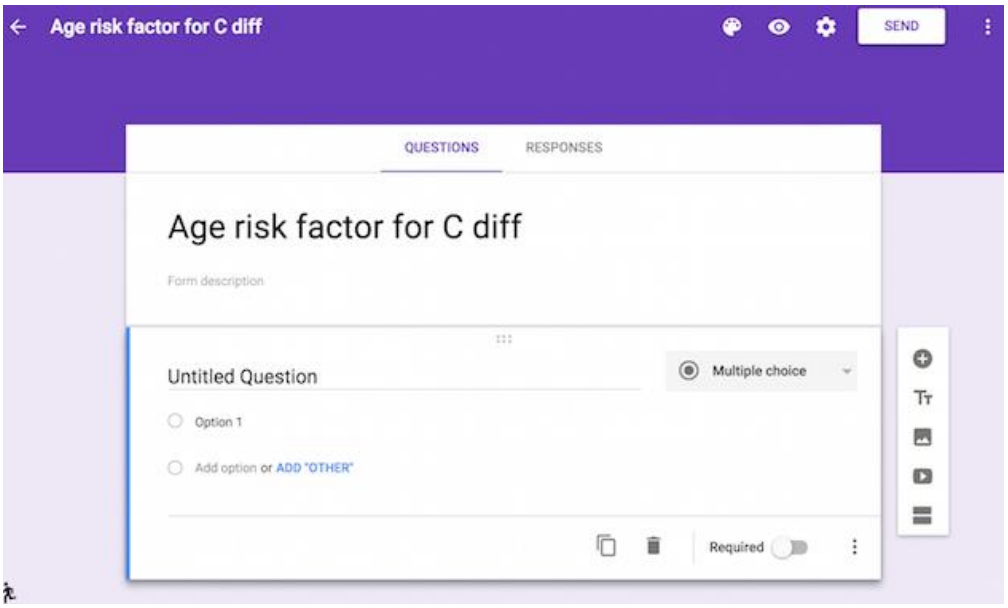

Рисунок 2.3 – Інтерфейс системи Google Forms

Microsoft Forms є аналогом Google Forms, але пропонує ширший набір функцій для корпоративного та освітнього середовища (рис.2.4). Вона дозволяє створювати опитування, анкети та тести, які автоматично перевіряються [29]. Платформа інтегрується з іншими сервісами Microsoft, такими як Teams і OneDrive. Однією з переваг є її можливість збирати відповіді в реальному часі та інтеграція з Office 365. Недоліками є обмеження безкоштовної версії, яка не включає розширену аналітику.

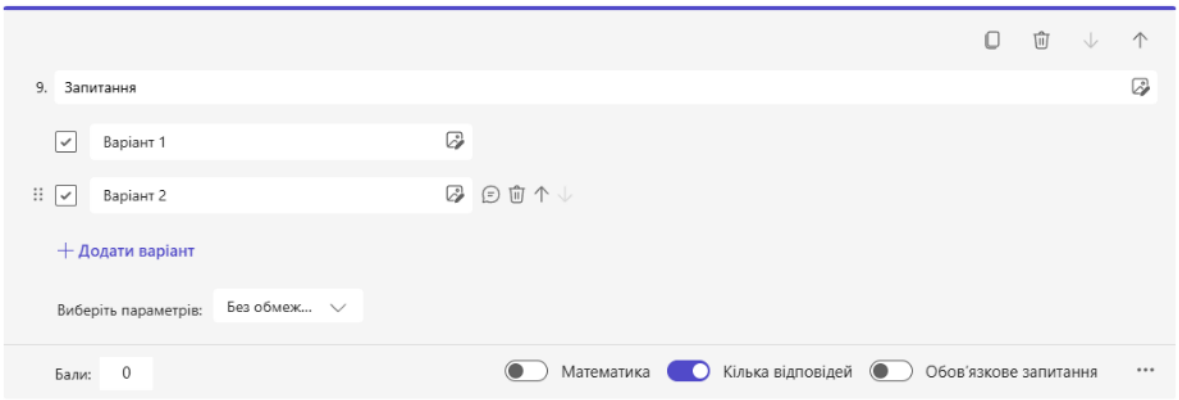

Рисунок 2.4 – Інтерфейс системи Microsoft Forms

Kahoot – інтерактивна платформа для створення тестів і вікторин, яка орієнтована на підвищення мотивації учнів (рис.2.5). Вона дозволяє проводити тести в ігровій формі, що сприяє залученню учасників. Kahoot підтримує як синхронне, так і асинхронне тестування, роблячи процес навчання веселим і динамічним [30]. Однак система менш підходить для глибокого оцінювання складних знань або великих обсягів інформації.

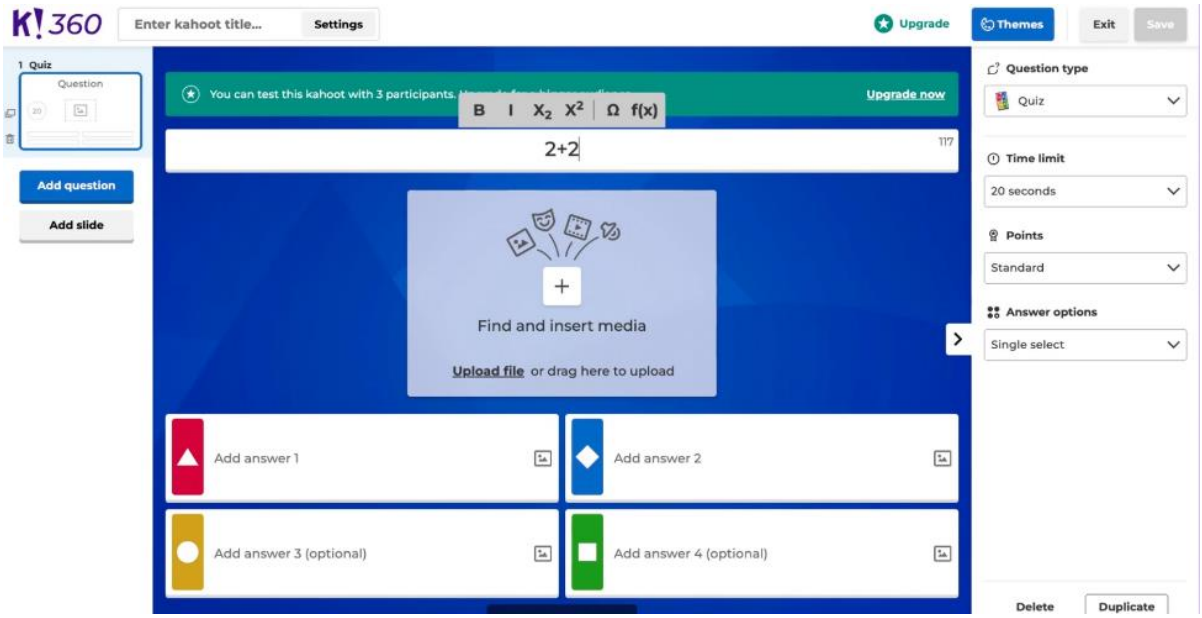

Рисунок 2.5 – Інтерфейс системи Kahoot

Blackboard – одна з провідних платформ у сфері вищої освіти. Вона забезпечує інтеграцію тестів із навчальними курсами, підтримує складні формати завдань, включаючи мультимедійні завдання, і пропонує розширену аналітику [31]. Blackboard (рис.2.6) дозволяє викладачам відслідковувати прогрес студентів у режимі реального часу. Недоліком є висока вартість використання, яка може бути недосяжною для багатьох навчальних закладів із обмеженим бюджетом.

Інноваційною платформою є ClassMarker, яка спеціалізується на проведенні професійних тестів (рис.2.7). Вона забезпечує високий рівень безпеки, можливість створення персоналізованих тестів і інтеграцію з іншими системами. ClassMarker дозволяє налаштовувати доступ до тестів і використовувати інструменти для захисту від шахрайства [35]. Основним

недоліком є те, що більшість функцій доступні лише у платній версії, що обмежує її використання для освітніх закладів із низькими ресурсами.

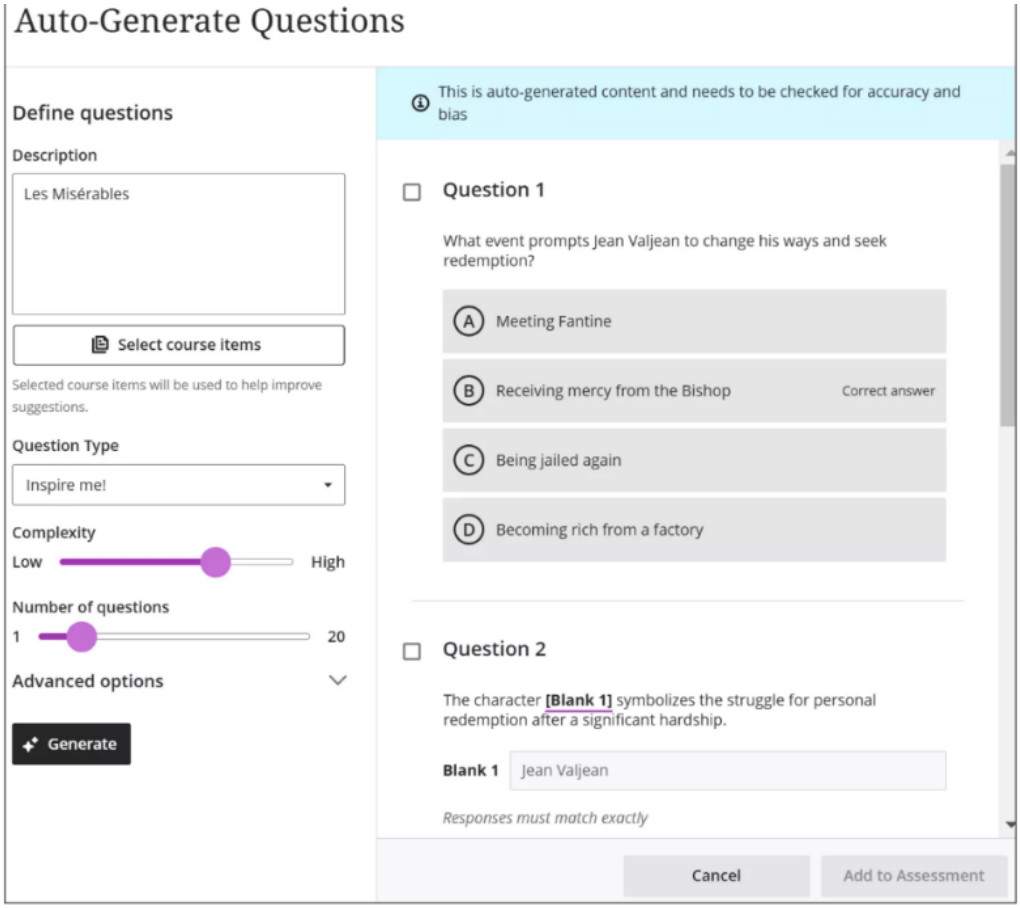

Рисунок 2.6 – Інтерфейс системи Blackboard

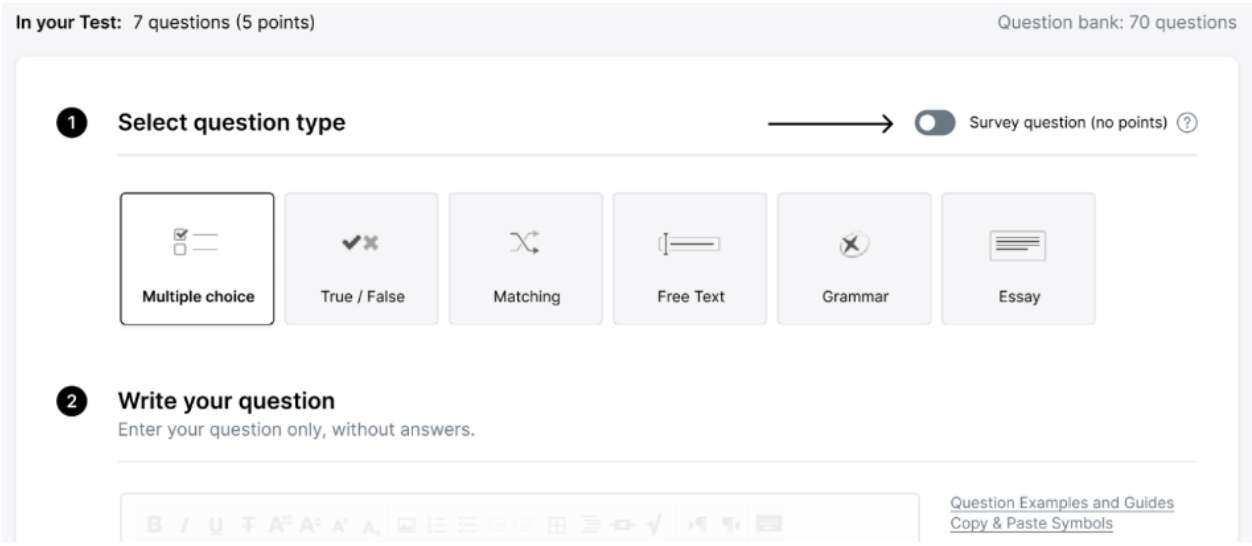

Рисунок 2.7 – Інтерфейс системи ClassMarker

Quizizz – інтерактивна платформа для створення тестів і вікторин із функцією гейміфікації (рис.2.8). Вона підтримує асинхронний формат тестування, що зручно для самостійного навчання учнів [32]. Quizizz дозволяє створювати кольорові та інтерактивні завдання, які заохочують учнів до активної участі. Вона також забезпечує аналітику результатів у реальному часі. Проте, як і у випадку з Kahoot, її використання може бути обмеженим для складних освітніх завдань.

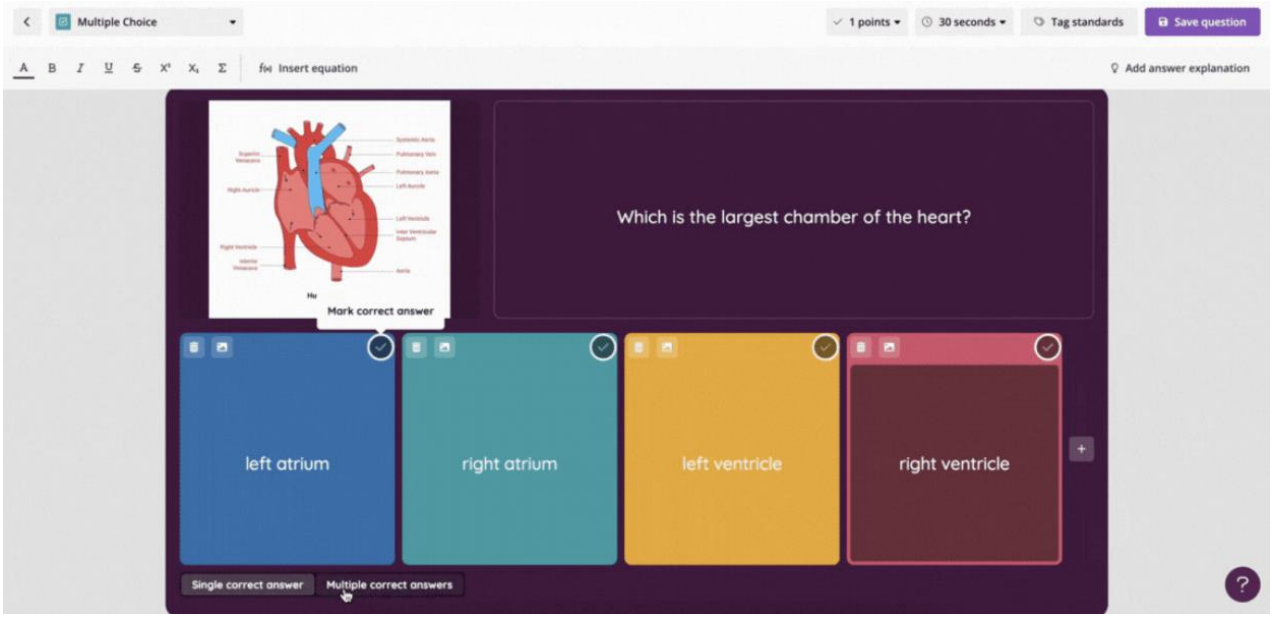

Рисунок 2.8 – Інтерфейс системи Quizizz

Canvas LMS – одна з найбільш функціональних платформ для управління навчанням (рис.2.9). Вона підтримує складні завдання, такі як відкриті питання, групові проєкти та адаптивні тести [33]. Canvas дозволяє викладачам створювати інтегровані курси, проводити тестування та отримувати детальну аналітику. Незважаючи на високу функціональність, вартість і складність навчання для нових користувачів можуть бути недоліками.

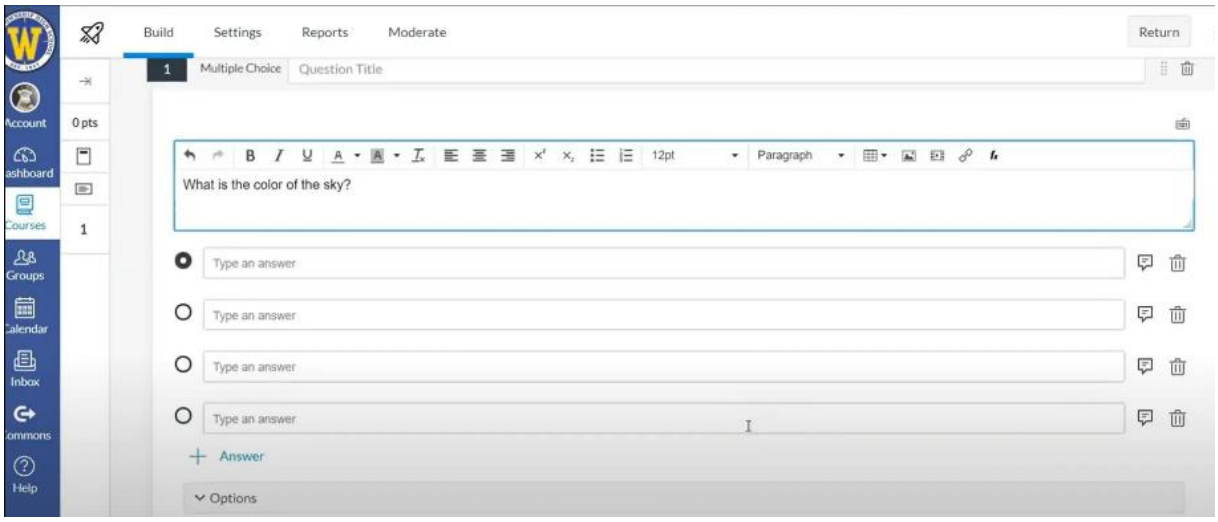

Рисунок 2.9 – Інтерфейс системи Canvas LMS

ProProfs – це ще одна популярна платформа, орієнтована на створення адаптивних тестів і навчальних курсів [34]. Вона пропонує інтеграцію з іншими інструментами та функціями зворотного зв'язку, що робить її корисною для викладачів, які хочуть надавати детальні рекомендації учням (рис.2.10). Основними перевагами є простота у використанні та широкий функціонал. Однак система є платною, і безкоштовна версія має суттєві обмеження.

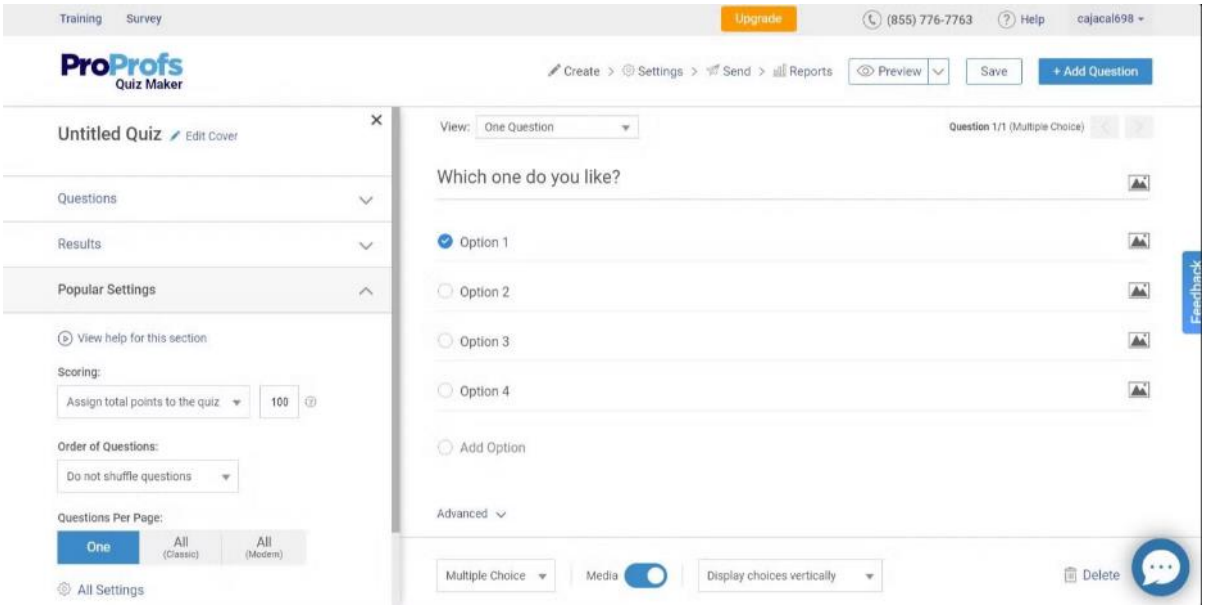

Рисунок 2.10 – Інтерфейс системи ProProfs

У Таблиці 2.1 представлено порівняльну таблицю основних систем тестування.

Таблиця 2.1 – Загальне порівняння популярних платформ для комп’ютерного тестування знань

Платформа	**Доступні функції**	**Безкоштовна / Платна**	**Цільова аудиторія**	**Особливості**
Moodle	Курси, різні типи тестів, аналітика	Безкоштовна	Школи, університети	Гнучкість, складність налаштувань
Google Forms	Тести, автоматична перевірка	Безкоштовна	Школи, малий бізнес	Простота, обмежена інтерактивність
Microsoft Forms	Тести, інтеграція з Office 365	Умовно безкоштовна	Корпоративне та освітнє середовище	Інтеграція з Microsoft
Kahoot	Інтерактивні вікторини, гейміфікація	Умовно безкоштовна	Школи, тренінги	Гра, підвищення мотивації
Blackboard	Курси, аналітика, складні формати завдань	Платна	Університети	Розширена аналітика
ClassMarker	Персоналізо-вані тести, безпека	Платна	Корпоративний сектор	Високий рівень безпеки
Quizizz	Гейміфіка-ція, асинхронне тестування	Умовно безкоштовна	Школи, дистанційне навчання	Зручність для самостійного навчання
Canvas LMS	Інтегровані курси, аналітика, адаптивні тести	Платна	Університети, школи	Широка функціональність, складність навчання
ProProfs	Адаптивні тести, навчальні курси	Платна	Освітні та корпоративні заклади	Детальний зворотний зв’язок

Таким чином, існує широкий вибір комп’ютерних систем тестування знань, кожна з яких має свої переваги та недоліки. Вибір платформи залежить від мети тестування, специфіки навчального процесу та технічних можливостей закладу освіти. Ретельний аналіз цих параметрів допоможе

підібрати найбільш відповідний інструмент, який максимально ефективно відповідає потребам освітнього процесу.

2.2. Архітектура комп'ютерних систем тестування

Архітектура комп'ютерних систем тестування визначає загальну структуру, принципи роботи та взаємодію між компонентами системи (рис.2.11).

Рисунок 2.11 – Основні компоненти архітектури

Вона забезпечує ефективність, масштабованість і надійність системи, дозволяючи адаптувати її до різних умов використання. У цьому підрозділі розглянемо основні компоненти архітектури таких систем, їхні функції та

взаємодію. Сучасні підходи до архітектури враховують вимоги до продуктивності, безпеки та гнучкості, дозволяючи реалізовувати інноваційні методи навчання й оцінювання знань.

Основні компоненти архітектури:

1. Інтерфейс користувача (User Interface) забезпечує взаємодію між системою та користувачем, включаючи адміністратора, викладача та учасника тестування. Він містить засоби для створення, налаштування та проходження тестів. Сучасні системи підтримують веб-інтерфейси, мобільні додатки, інтерактивні панелі та інтеграцію з іншими платформами, такими як Learning Management Systems (LMS). Зручність інтерфейсу є критичною для забезпечення високого рівня взаємодії користувачів із системою.

2. Серверна частина (Backend) відповідає за зберігання, обробку та управління даними. Вона включає:

- Базу даних, яка зберігає інформацію про користувачів, тестові завдання, результати та аналітику. Сучасні бази даних забезпечують високу швидкість доступу до даних і надійність зберігання.
- Логіку бізнес-процесів, яка обробляє запити користувачів, перевіряє відповіді, генерує звіти та забезпечує функціонування всіх сервісів системи.
- Систему безпеки, яка включає захист даних від несанкціонованого доступу, управління ролями користувачів і алгоритми виявлення шахрайства, наприклад, аналіз підозрілої поведінки під час тестування.

3. Модуль створення тестів дозволяє адміністраторам створювати та редагувати тестові завдання. Важливими функціями є:

- Підтримка різних форматів завдань (закриті питання, відкриті відповіді, адаптивні завдання, завдання з використанням мультимедіа).

- Інтеграція з банками завдань, які дозволяють автоматично формувати тести на основі певних критеріїв.
- Забезпечення варіативності завдань для унікальності кожного тесту.

4. Модуль тестування відповідає за проведення тестів у реальному часі. Він забезпечує:

- Стабільність роботи при одночасному доступі великої кількості користувачів.
- Адаптивність, що дозволяє змінювати складність завдань залежно від результатів попередніх питань.
- Підтримку офлайн-режиму для проходження тестів без підключення до мережі, з можливістю синхронізації результатів після відновлення підключення.
- Інтерактивні елементи, наприклад, таймери або підказки, які сприяють залученню учасників.

5. Модуль аналізу та звітності генерує детальні звіти про результати тестування, включаючи статистику відповідей, аналітику прогресу учасників і рекомендації для подальшого навчання. Сучасні системи часто інтегрують алгоритми машинного навчання для автоматизованого аналізу відповідей та прогнозування успішності учнів. Наприклад, аналітика може виявляти прогалини в знаннях і пропонувати відповідні навчальні матеріали.

Типи архітектур комп’ютерних систем тестування:

1. Клієнт-серверна архітектура. У цій моделі клієнт (користувач) взаємодіє із сервером через мережу. Сервер обробляє всі запити, зберігає дані та забезпечує виконання бізнес-логіки. Ця архітектура забезпечує централізоване управління і підходить для великих організацій, які потребують високої надійності й захищеності даних [25].

2. Розподілена архітектура. Цей тип архітектури використовує кілька серверів, які виконують різні функції, наприклад, сервер для бази даних, сервер для обробки бізнес-логіки та сервер для аналітики. Розподілена

архітектура підвищує масштабованість і стійкість системи, дозволяючи обробляти великі обсяги даних одночасно.

3. Хмарна архітектура. Сучасні системи тестування часто використовують хмарні рішення, які забезпечують доступність із будь-якого пристрою, високу масштабованість і зниження витрат на інфраструктуру. Такі системи дозволяють адаптувати ресурси залежно від потреб користувачів, що особливо корисно під час проведення масових тестувань.

4. Гібридна архітектура. Гібридна архітектура поєднує елементи локальних і хмарних рішень, забезпечуючи баланс між безпекою даних та гнучкістю доступу [24]. Наприклад, частина даних може зберігатися локально для забезпечення конфіденційності, тоді як аналітика виконується в хмарі для підвищення швидкодії.

Архітектура систем тестування повинна підтримувати інтеграцію з іншими освітніми платформами та сервісами, такими як LMS, CRM, ERP-системи та аналітичні інструменти. Наприклад, інтеграція з Moodle або Microsoft Teams дозволяє створювати єдине навчальне середовище, у якому викладачі можуть легко керувати навчальним процесом, тестуванням і звітністю.

У системах тестування особливу увагу приділяють безпеці [26]. Основні механізми включають:

1. Шифрування даних під час передачі та зберігання.

2. Аутентифікацію та авторизацію користувачів із використанням багатофакторної ідентифікації.

3. Виявлення шахрайства за допомогою аналізу поведінки учасників тестування (наприклад, використання систем розпізнавання обличчя або фіксації сторонніх звуків).

4. Резервне копіювання даних і створення механізмів відновлення після збоїв.

Архітектура комп’ютерних систем тестування є ключовим фактором їхньої ефективності та надійності. Вибір конкретного типу архітектури

залежить від масштабів системи, вимог до безпеки, функціональності та доступних ресурсів. Інноваційні підходи, такі як адаптивні алгоритми, інтеграція з хмарними сервісами та використання машинного навчання, роблять ці системи незамінними інструментами для сучасного навчання. У майбутньому розвиток архітектури таких систем буде орієнтований на ще більшу персоналізацію, автоматизацію процесів і підвищення рівня інтерактивності для користувачів.

2.3. Порівняльний аналіз функціональних можливостей систем

Комп'ютерні системи тестування знань мають значні відмінності у функціональних можливостях, гнучкості, інтеграції з іншими платформами та зручності використання (Таблиця 2.2). Вибір відповідної системи залежить від потреб користувача, масштабу застосування та особливостей навчального процесу. У цьому підрозділі розглянуто порівняльний аналіз популярних систем, таких як Moodle, Google Forms, Microsoft Forms, Kahoot, Blackboard, Quizizz, Canvas LMS та ProProfs. Оцінка базується на підтримці форматів завдань, інтеграції, безпеці, аналітичних можливостях, вартості та інших аспектах, які впливають на їх ефективність.

Ключові критерії аналізу:

1. Підтримка різних форматів завдань. Moodle, Blackboard і Canvas LMS є лідерами за підтримкою різноманітних форматів завдань, зокрема закритих питань, відкритих відповідей, мультимедійних завдань, адаптивних тестів і завдань есе. Google Forms і Microsoft Forms забезпечують базову функціональність, зосереджену на простих завданнях із вибором відповіді та текстових відповідях. Kahoot і Quizizz, орієнтовані на гейміфікацію, підтримують інтерактивні вікторини, що робить їх ідеальними для молодших учнів, але менш придатними для глибокого академічного аналізу.

2. Інтеграція з іншими платформами. Microsoft Forms демонструє відмінну інтеграцію з продуктами Microsoft, такими як Office 365, Teams та

OneDrive. Moodle, Canvas LMS і Blackboard забезпечують інтеграцію з іншими освітніми платформами, включаючи SCORM та інструменти для управління навчанням. Google Forms чудово працює в межах екосистеми Google Workspace, але має обмеження у взаємодії з складними освітніми платформами.

3. Безпека та захист даних. Blackboard, ProProfs і ClassMarker надають високий рівень безпеки завдяки шифруванню даних, багатофакторній аутентифікації та механізмам виявлення шахрайства. Moodle забезпечує базовий рівень безпеки, який залежить від налаштувань адміністратора. Google Forms і Microsoft Forms пропонують стандартний набір засобів захисту, підходячи для менш критичних тестів і опитувань.

4 Аналітика та звітність. Blackboard, Canvas LMS і ProProfs мають потужні аналітичні інструменти, що дозволяють створювати детальні звіти, аналізувати прогрес учнів і прогнозувати їхні результати. Quizizz і Kahoot надають базову статистику, яка орієнтована на швидке отримання результатів і візуалізацію основних даних.

5. Вартість. Moodle є безкоштовною платформою з відкритим кодом, що робить її привабливою для організацій із обмеженим бюджетом. Blackboard, Canvas LMS і ProProfs мають високу вартість, яка виправдовується їхньою розширеною функціональністю. Google Forms, Microsoft Forms, Quizizz і Kahoot пропонують безкоштовні базові версії з можливістю платного розширення.

6. Гнучкість і масштабованість (додатковий аспект аналізу). Moodle і Canvas LMS демонструють високу гнучкість, дозволяючи адаптувати системи до різних масштабів використання, від малих шкіл до великих університетів. Blackboard забезпечує масштабованість для великих організацій із численними користувачами. Quizizz і Kahoot найкраще підходять для індивідуального використання або невеликих груп через їхню простоту.

Таблиця 2.2 – Порівняння функціональних можливостей систем

Платформа	**Підтримка форматів завдань**	**Інтеграція**	**Безпека**	**Аналітика**	**Вартість**
Moodle	Широка	Висока	Середня	Розширена	Безкоштовна
Google Forms	Базова	Середня	Базова	Базова	Безкоштовна
Microsoft Forms	Базова	Висока	Середня	Базова	Умовно безкоштовна
Kahoot	Інтерактивна	Низька	Базова	Середня	Умовно безкоштовна
Blackboard	Широка	Висока	Висока	Розширена	Платна
ClassMarker	Широка	Середня	Висока	Середня	Платна
Quizizz	Інтерактивна	Низька	Базова	Середня	Умовно безкоштовна
Canvas LMS	Широка	Висока	Висока	Розширена	Платна
ProProfs	Широка	Середня	Висока	Розширена	Платна

7. Інтерфейс користувача (додатковий аспект аналізу). Простота використання є важливим фактором для користувачів із різними рівнями технічної підготовки. Google Forms і Microsoft Forms відзначаються інтуїтивно зрозумілим інтерфейсом, тоді як Blackboard і Canvas LMS можуть вимагати навчання для ефективного використання через їхній багатий функціонал.

8. Інноваційні можливості (додатковий аспект аналізу). Quizizz і Kahoot інтегрують елементи гейміфікації, що сприяє підвищенню мотивації учнів. ProProfs і ClassMarker підтримують персоналізацію навчання через адаптивні алгоритми, які підлаштовують завдання під рівень знань учня.

Порівняльний аналіз демонструє, що кожна система має свої унікальні переваги та недоліки. Moodle є найкращим вибором для організацій, які шукають гнучке та економічно ефективне рішення. Blackboard і Canvas LMS підходять для університетів, які потребують потужних інструментів аналітики. Google Forms і Microsoft Forms зручні для базових тестів, а Kahoot і Quizizz пропонують інтерактивні інструменти для підвищення залученості учнів. ProProfs і ClassMarker відзначаються високим рівнем безпеки й персоналізації, що робить їх ідеальними для професійного тестування.

2.4. Вимоги, проблеми та виклики у розробці та використанні систем тестування

Сучасні системи тестування знань стикаються з численними проблемами та викликами, які впливають на їхню ефективність, надійність та відповідність потребам освіти. Одночасно такі системи мають відповідати низці вимог, які забезпечують їхню адаптивність, безпеку та ефективність у швидко змінюваному середовищі. У цьому розділі перелічені та розкриті вимоги, проблеми та виклики, з якими стикаються розробники та користувачі систем тестування.

1. Технічні аспекти:
   - Сумісність і інтеграція. Системи тестування повинні інтегруватися з освітніми платформами, такими як LMS, CRM або ERP. Стандарти, як-от SCORM та xAPI, забезпечують сумісність із зовнішніми інструментами, але інтеграція часто ускладнюється технічними обмеженнями, такими як відсутність підтримки нових технологій або конфлікти між системами.
   - Масштабованість і продуктивність. Масові іспити вимагають підтримки великої кількості користувачів одночасно. Використання хмарних технологій і розподілених обчислень дозволяє вирішити цю проблему, проте потребує значних інвестицій у технічну інфраструктуру, включаючи сервери, мережеве обладнання та інструменти управління даними [25].
   - Безпека даних і конфіденційність. Захист даних учасників тестування є важливим компонентом [26]. Використання сучасних методів шифрування, багатофакторної аутентифікації та регулярного аудиту системи допомагає запобігати витокам інформації. Важливо дотримуватися міжнародних стандартів, таких як GDPR та ISO/IEC 27001, щоб забезпечити довіру до систем.

- Доступність і мобільність. Системи повинні забезпечувати підтримку різних пристроїв і операційних систем. Особлива увага має приділятися доступності для осіб із обмеженими можливостями відповідно до стандартів WCAG, що включає альтернативні способи доступу до контенту, адаптивний дизайн та підтримку допоміжних технологій.

2. Функціональні вимоги:
   - Адаптивність. Інтеграція адаптивних алгоритмів дозволяє системам підлаштовувати складність тестів до рівня знань учасників [24]. Це забезпечує індивідуалізований підхід, проте потребує ретельного тестування алгоритмів, щоб уникнути помилкових результатів і упереджень.
   - Різноманітність форматів завдань. Тести мають включати різні типи завдань: закриті питання, відкриті відповіді, мультимедійні елементи, інтерактивні завдання, есе та навіть симуляції. Різноманіття форматів сприяє більш об'єктивному оцінюванню знань і навичок учнів.
   - Аналітика та звітність. Системи повинні надавати розширену аналітику, включаючи статистику відповідей, прогрес учнів, успішність за конкретними темами та рекомендації для подальшого навчання. Застосування машинного навчання може забезпечити точніше прогнозування та адаптацію навчальних матеріалів.
   - Гнучкість налаштувань. Платформи повинні дозволяти налаштовувати параметри тестів, включаючи тривалість, кількість спроб, методи оцінювання, доступність завдань і параметри безпеки. Це забезпечує більшу адаптивність до потреб різних користувачів.
3. Педагогічні та етичні аспекти:

- Відповідність освітнім цілям. Завдання тестів повинні відповідати освітнім стандартам і перевіряти як базові знання, так і складні когнітивні навички: критичне мислення, аналітичні здібності, творчість та комунікацію. Розробка таких завдань вимагає міждисциплінарного підходу.
- Мотивація учасників. Елементи гейміфікації, такі як рейтинги, досягнення, візуалізація прогресу, допомагають залучити учнів до процесу тестування та зменшити стрес. Інтерактивні елементи, такі як відео або ігрові завдання, також сприяють підвищенню мотивації.
- Запобігання шахрайству. Технології виявлення шахрайства повинні включати моніторинг веб-камери, аналіз поведінки учасників, блокування сторонніх програм і автоматичне виявлення підозрілих відповідей. Водночас необхідно враховувати етичність і конфіденційність таких рішень.
- Захист конфіденційності. Дані учасників повинні зберігатися відповідно до норм конфіденційності, таких як GDPR. Системи мають забезпечувати прозорість використання даних і дозволяти учасникам контролювати обсяг інформації, яка збирається.

4. Проблеми та виклики:

- Високі витрати на розробку. Розробка систем тестування вимагає значних фінансових ресурсів. Ліцензії, обладнання, навчання персоналу та технічна підтримка є дорогими, а обмежені бюджети часто стають перепоною для їхнього впровадження.
- Навчання персоналу. Викладачі та адміністратори часто стикаються з труднощами в освоєнні нових систем через відсутність технічних знань. Програми підготовки персоналу займають час і потребують додаткових витрат, але є важливим компонентом успішного впровадження.

- Етичні проблеми алгоритмів. Алгоритми штучного інтелекту можуть містити вбудовані упередження, що призводить до несправедливих результатів. Регулярний аудит алгоритмів, прозорість їхньої роботи та використання відкритих моделей допоможуть мінімізувати ці ризики.
- Складність інтеграції. Нові системи тестування повинні інтегруватися з існуючою інфраструктурою навчальних закладів. Несумісність або застаріла інфраструктура створюють значні виклики для впровадження.

5. Додаткові аспекти:
   - Локалізація та багатомовність. Системи повинні підтримувати кілька мов і враховувати культурні особливості користувачів. Локалізація інтерфейсу та навчальних матеріалів є обов'язковою умовою для глобального впровадження.
   - Оновлюваність. Платформи повинні бути гнучкими для регулярних оновлень відповідно до технічного прогресу та освітніх потреб. Це включає інтеграцію нових форматів завдань, вдосконалення алгоритмів та оновлення заходів безпеки.
   - Покращений користувацький досвід. Інтерфейс має бути інтуїтивно зрозумілим і доступним для користувачів із різними рівнями технічної підготовки. Особлива увага приділяється адаптації для мобільних пристроїв, які стають основним інструментом навчання для багатьох користувачів.

Таким чином, сучасні системи тестування знань мають одночасно відповідати технічним, функціональним, педагогічним і етичним вимогам, а також долати численні проблеми та виклики, пов'язані з їхньою розробкою та використанням. Комплексний підхід, який включає інновації, міждисциплінарну співпрацю та фокус на користувацькому досвіді, дозволить створити платформи, які сприятимуть якісному оцінюванню знань і розвитку сучасної освіти.

РОЗДІЛ 3
ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ

3.1. Постановка задачі та визначення цілей системи

Розробка сучасних систем тестування знань є складним і багатогранним процесом, який починається з чіткого визначення задач і цілей. Цей етап формує основу для створення ефективної, безпечної та адаптивної платформи, здатної задовольнити сучасні потреби освіти й відповідати її викликам.

У нашому випадку, головна задача системи тестування знань полягає в автоматизації процесу оцінювання знань, навичок і компетенцій учасників навчального процесу. Для досягнення цієї мети система повинна виконувати такі завдання:

1. Проведення об'єктивної оцінки знань. Забезпечення прозорості, справедливості та виключення людських помилок під час оцінювання. Автоматизація перевірки відповідей мінімізує суб'єктивність та упередженість.

2. Підтримка різноманітних форматів завдань. Система має включати тести з вибором відповіді, відкриті запитання, мультимедійні завдання, симуляції, інтерактивні вправи та есе. Це дозволяє оцінити різні аспекти знань і навичок учасників.

3. Забезпечення безпеки та конфіденційності. Система повинна відповідати міжнародним стандартам захисту даних, зокрема GDPR, та використовувати шифрування для захисту результатів тестування.

4. Інтеграція інструментів для запобігання шахрайству. Система повинна мати механізми контролю, зокрема:

- Постійний запис процесу тестування через веб-камеру.

- Аналіз поведінки учасників, включаючи моніторинг руху очей та виявлення спроб обману.
- Блокування тесту при виявленні численних порушень правил.

5. Масштабованість. Система має працювати стабільно навіть за умов великої кількості одночасних користувачів, наприклад, під час вступних іспитів.

Цілі системи тестування формують її функціональні можливості та напрямки розвитку. Основні цілі включають:

1. Підвищення ефективності оцінювання. Автоматизація дозволяє швидко та точно перевіряти знання учасників, скорочуючи час і ресурси, необхідні для проведення тестування.

2. Надання розширеної аналітики та звітності. Система повинна генерувати детальні звіти, що включають статистику результатів, аналіз прогалин у знаннях та рекомендації щодо подальшого навчання.

3. Підвищення залученості учасників. Використання елементів гейміфікації, таких як рейтинги, досягнення, інтерактивні завдання, допомагає мотивувати учнів і робить тестування цікавішим.

4. Забезпечення академічної доброчесності. Завдяки інструментам для запобігання шахрайству система підтримує високий рівень чесності під час тестування.

5. Інтеграція інноваційних технологій. Використання штучного інтелекту, машинного навчання та великих даних розширює можливості платформи, робить її більш адаптивною та ефективною.

6. Забезпечення інклюзивності. Система має враховувати потреби осіб із обмеженими можливостями, забезпечуючи доступність і комфорт для всіх користувачів незалежно від їхніх фізичних чи технічних обмежень.

3.2. Розробка технічного завдання

3.2.1 Загальні вимоги

Загальними вимогами для розробляємої системи є наступні

1. Клієнт-серверне рішення:
    - Система повинна бути реалізована у вигляді клієнт-серверного рішення.
    - Обґрунтування вибору програмного забезпечення та бази даних включає критерії продуктивності, безпеки та зручності розробки.
2. Гнучкість в настройці та керуванні тестами
3. Безпека та профілі:
    - Забезпечити розділення функцій між користувацькою (учнівською) та адміністраторською (викладацькою) частинами.
    - Реалізувати різні рівні доступу для різних типів профілів.
    - Інтеграція сучасних заходів захисту, таких як шифрування даних.

4. Захист від шахрайства - впровадження елементів штучного інтелекту для підвищення ефективності роботи системи, наприклад, автоматичного аналізу поведінки учасників.

3.2.2 Профілі користувачів

Кожен користувач (учень, викладач, адміністратор) має отримувати доступ через унікальний логін та пароль. Також передбачається 3 типи доступу: адміністраторський, викладацький та учнівський.

Адміністраторський профіль**:**

- Повний доступ до системи.
- Можливість створення, редагування та видалення профілів, груп і тестів.
- Управління параметрами системи.

Викладацький профіль**:**

- Створення, редагування та видалення тестів.

- Додавання доступу учням та групам до тестів.
- Перегляд статистики проходження тестів.

Учнівський профіль**:**

- Можливість проходження тестів, до яких надано доступ.
- Перегляд результатів пройдених тестів.
- Можливість перегляду детального зворотного зв’язку за результатами тестів.

3.2.3 Вимоги до створення тестів

Базові параметри (додавання та редагування)**:**

- Назви тесту.
- Опису тесту (опціонально).
- Списку учнів та груп у яких є доступ до тесту
- Кінцевої дати проходження тесту.
- Кількості спроб (опціонально).
- Часу на проходження тесту.

Додаткові параметри з елементами ШІ для запобігання шахрайству (опціонально)**:**

- Контроль за ввімкненою веб-камерою (запис процесу тестування).
- Виявлення відведення погляду або відключення веб-камери.
- Перехід на інші вкладки або монітори.
- Встановлення обмежень на кількість порушень з можливістю додати текст попередження під кожен тип, після яких доступ до тесту блокується.

Також слід звернути увагу на аналіз результатів і можливості з опрацювання його даних**:**

- Збір детальної статистики, включаючи відповіді, загальний бал та тривалість проходження тесту.
- Генерація звітів для аналізу успішності окремих учасників і груп.

3.2.4 Вимоги до створення питань тестів

Параметри питань:

- Кожне питання має містити текст або зображення.
- Поле “Відповідь”:
    - Відсутнє тільки для запитань типу “Передати файл” та “Довгий текст”.
    - Всі інші обов'язково мають поле “Варіант” (текст та/або зображення, проте “зображення” недоступне для типу запитання “Текст”)
- Обов’язкове поле для балів за кожне питання.

Типи питань:

- Вибір (одна правильна відповідь).
- Мультивибір (декілька правильних відповідей).
- Текст (коротка текстова відповідь).
- Довгий текст (відсутність правильної відповіді).
- Сортування (розташування елементів у правильному порядку).
- Передача файлу (надсилання файлу як відповіді).

Редагування питань:

- Можливість сортування, редагування та видалення питань.
- Встановлення зображень або відеоматеріалів як частини питання.

3.2.5 Вимоги до проходження тестів

Проходження тестів**:**

- Авторизація перед початком тестування.
- Вибір доступного тесту зі списку.
- Відображення пройдених тестів із підсумковим балом.

Процес тестування**:**

- Виведення опису тесту перед його початком (опціонально).
- Кнопка "Почати тестування" для старту.

- Завершення тесту через кнопку "Завершити тестування" із попередженням про можливість зарахування поточних відповідей.
    - Якщо відповіді на всі питання надані в повному обсязі - повідомлення на кшталт “Чи ви впевнені що хочете завершити тестування зараз? Ваші поточні відповіді будуть зараховані і ви вже не зможете їх виправити”;
    - Якщо відповіді на всі питання надані не в повному обсязі - повідомлення на кшталт “Чи ви впевнені що хочете завершити тестування зараз? Ви не надали відповіді на всі питання. Ваші поточні відповіді будуть зараховані і ви вже не зможете їх виправити”.

Автоматичне завершення**:**

- Тест завершується автоматично, якщо таймер спливає.
- Після завершення відображається підсумковий бал.

3.2.6 Інші вимоги

Безпека системи:

- Розробка рішень для захисту даних учасників і результатів тестування.
- Передбачення сценаріїв переривання тестування через проблеми з інтернетом чи енергопостачанням.

Зберігання даних:

- Встановлення максимального строку та обсягу зберігання даних (результати, відео, фото, файли).
- Впровадження механізмів автоматичного очищення старих даних.

Інтеграція з іншими системами:

- Реалізація API для інтеграції з освітніми платформами (LMS, ERP).
- Підтримка експорту даних у різних форматах (CSV, PDF).

Масштабованість і продуктивність:

- Забезпечення стабільної роботи при великій кількості одночасних користувачів.
- Використання хмарних технологій для розширення ресурсів під час пікових навантажень.

Доступність**:**

- Підтримка декількох мов і врахування особливих потреб користувачів з обмеженими можливостями.

Реалізація цього технічного завдання із зазначеними вимогами забезпечить створення системи тестування, яка відповідає сучасним стандартам якості, безпеки, функціональності та зручності для користувачів.

3.3. Вибір архітектури системи, технологій та інструментів для реалізації

Вибір архітектури, технологій і інструментів для реалізації системи тестування знань ґрунтується на потребі забезпечення високої продуктивності, безпеки, масштабованості та адаптивності. Далі детально розглянемо кожен елемент архітектури, обрані технології та інструменти, а також обґрунтуємо їхній вибір у порівнянні з альтернативами.

3.3.1 Вибір архітектури

Система побудована за клієнт-серверною архітектурою. Такий підхід розділяє функції між серверною (backend) і клієнтською (frontend) частинами, що забезпечує гнучкість у розробці та масштабуванні. Серверна частина відповідає за обробку запитів, взаємодію з базою даних і забезпечення безпеки, тоді як клієнтська частина забезпечує інтерактивний користувацький інтерфейс (рис. 3.1). Обраний підхід дозволяє реалізувати модульність, підвищує ефективність розробки та спрощує підтримку системи.

Рисунок 3.1 – Основні компоненти архітектури

3.3.2 Базові технології та інструменти

Вибір технологій та інструментів для реалізації системи базується на сучасних тенденціях у розробці програмного забезпечення та потребі забезпечення зручності, швидкості та безпеки:

1. TypeScript це мова програмування, яка позіціонується як засіб розробки вебзастосунків, що розширює можливості JavaScript. Використовується як основна мова програмування завдяки своїм численним перевагам:
   - Статична типізація яка допомагає виявляти помилки на етапі написання коду, забезпечуючи стабільність.
   - Модернізована екосистема яка підтримує сучасні стандарти JavaScript і дозволяє використовувати розширені функції без ризику зниження сумісності.
   - Інструментальна підтримка яка покращує досвід розробки завдяки інтеграції з популярними IDE та забезпечує автозаповнення, рефакторинг та перевірку коду в реальному часі.

У порівнянні з чистим JavaScript (рис. 3.2), TypeScript забезпечує вищу надійність і кращу організацію коду в середніх і великих проєктах, тоді як JavaScript більше підходить для невеликих застосунків.

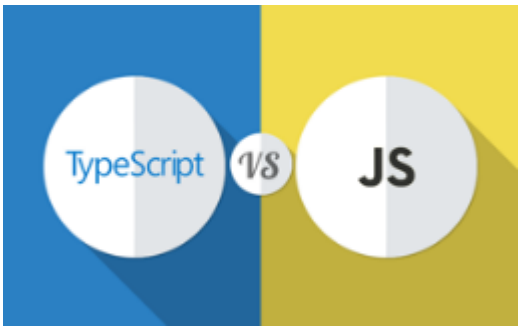

Рисунок 3.2 – Візуалізація порівняння TypeScript та JavaScript

2. Node.js це платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Є основним інструментом для розробки серверної частини завдяки:
   - Асинхронній обробці яка дозволяє працювати з великою кількістю одночасних запитів, забезпечуючи швидку відповідь.
   - Єдності технологій: дає можливість використовувати JavaScript як для клієнта, так і для сервера, що значно спрощує підтримку.
   - Масштабованості: дозволяє ефективно адаптувати систему до зростання навантаження.

На відміну від PHP (рис. 3.3), Node.js краще підходить для роботи з реальним часом, наприклад, чатів або систем моніторингу, завдяки своїй подієво-орієнтованій архітектурі.

Рисунок 3.3 – Візуалізація порівняння Node.js та PHP

3. MySQL це компактний багатопотоковий сервер баз даних (реляційний) який характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосунків. Саме його вибрано для роботи з базою даних через:

- Простоту використання яка підходить для швидкого впровадження в освітні проєкти.
- Підтримку транзакцій яка забезпечує надійність і консистентність даних.
- Оптимізацію під навантаження яка легко масштабується для великих проєктів.

Якщо порівнювати, то PostgreSQL (рис. 3.4) забезпечує більшу функціональність, але MySQL простіший у налаштуванні, що є важливим для проєктів зі швидкими термінами реалізації.

Рисунок 3.4 – Візуалізація порівняння MySQL та PostgreSQL

4. Nest.js це сучасний backend фреймворк для створення ефективних і масштабованих серверних додатків на Node. js. Він поєднує в собі принципи об'єктно-орієнтованого та функціонального програмування, надаючи розробникам інструменти для створення високопродуктивних додатків [42]. Використовується у цій роботі як фреймворк для серверної частини завдяки:

- Модульній архітектурі яка полегшує підтримку та розвиток системи.
- Інтеграції з TypeScript яка дозволяє використовувати переваги статичної типізації.
- Багатофункціональності яка включає підтримку WebSocket, GraphQL, інтеграцію з ORM і багато іншого.

Якщо порівнювати, то Express.js (рис. 3.5) забезпечує увесь базовий функціонал, але Nest.js надає готову архітектуру для складних застосунків, що прискорює розробку.

Рисунок 3.5 – Візуалізація порівняння Nest.js та Express.js

5. TypeORM це ORM (Object-Relational Mapping) для TypeScript і JavaScript, який дає змогу розробникам працювати з реляційними базами даних, такими як MySQL, PostgreSQL, SQLite та іншими. TypeORM надає можливість визначення моделей даних і виконання запитів мовою TypeScript [42]. Обрано для роботи з базою даних завдяки:

- Об'єктно-реляційному маппінгу який спрощує управління реляційними даними через код.
- Підтримці міграцій яка дозволяє легко оновлювати структуру бази даних.
- Інтеграції з TypeScript яка полегшує роботу з моделями даних.

Sequelize (рис. 3.6) має подібну функціональність, але TypeORM краще інтегрується з TypeScript і забезпечує більшу гнучкість у роботі з базами даних.

Рисунок 3.6 – Візуалізація порівняння TypeORM та Sequelize

6. React.js це відкрита JavaScript бібліотека для створення інтерфейсів користувача (frontend), яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці

односторінкових застосунків. В нашому випадку є базовим інструментом для клієнтської частини завдяки:

- Компонентній архітектурі яка полегшує повторне використання коду.
- Віртуальному DOM (це концепція програмування, в якій ідеальне чи «віртуальне» представлення інтерфейсу користувача зберігається в пам'яті і синхронізується зі «справжнім» DOM за допомогою бібліотеки, такої як ReactDOM.) який забезпечує швидке оновлення інтерфейсу без повного перезавантаження сторінки.
- Гнучкості яка легко інтегрується з іншими бібліотеками.

Якщо порівнювати, то Angular (рис. 3.7) пропонує ширший набір функцій «із коробки», але React.js забезпечує більше свободи у виборі сторонніх бібліотек і підходів до розробки.

Рисунок 3.7 – Візуалізація порівняння Angular та React.js

7. Redux це JavaScript-бібліотека, покликана спростити управління станом веб-додатку. Її основне призначення полягає в тому, щоб зробити управління даними більш організованим і передбачуваним. В нашому випадку використовується для централізованого управління станом застосунку та надає:

- Прогнозованість яка дозволяє легко відстежувати зміни в стані системи.
- Централізацію даних, при якій усі дані знаходяться в одному місці, що спрощує їх використання.

- Сумісність яка легко інтегрується з іншими бібліотеками, такими як React.

Якщо порівнювати, то Context API (рис. 3.8) підходить для невеликих проєктів, тоді як Redux є більш ефективним у великих застосунках із численними взаємодіями.

Рисунок 3.8 – Візуалізація порівняння Redux та Context API

3.3.3 Безпекові технології

Passport і JWT (JSON Web Token) — це інструменти, які використовуються для аутентифікації та авторизації в веб-застосунках.

Рисунок 3.9 – Візуальне представлення Passport і JWT

Passport — це бібліотека для Node.js, яка забезпечує гнучкий та розширюваний механізм аутентифікації. Вона підтримує безліч стратегій аутентифікації, таких як локальна (логін і пароль), OAuth, OpenID Connect, SAML тощо. Її особливостями є легка інтеграція з Node.js-застосунками, підтримка різних стратегій аутентифікації та розширюваність (можна створювати власні стратегії аутентифікації).

JWT (JSON Web Token) — це стандарт, який використовується для створення токенів, що містять зашифровану інформацію про користувача. Ці

токени використовуються для аутентифікації та передачі даних між клієнтом і сервером у безпечний спосіб. Серед особливостей цього стандарту можна відзначити:

- Його структуру яка кладається з трьох частин: Header (заголовок), Payload (корисне навантаження), Signature (цифровий підпис).
- Його масштабованість при якій токени зберігаються на стороні клієнта (наприклад, у браузері), тому сервер не потребує зберігати сесії.
- Його безпека, коли дані в JWT підписуються криптографічним алгоритмом, що гарантує їх цілісність.

Ці два інструменти (Passport і JWT) працюють разом за принципами аутентифікації та авторизації (таблиця 3.1).

Таблиця 3.1 – Опис взаємодії Passport і JWT

Аутентифікація	**Авторизація**
• Користувач вводить логін і пароль, які перевіряються через Passport. • Якщо аутентифікація успішна, сервер створює JWT і відправляє його клієнту.	• Клієнт зберігає JWT (зазвичай у cookies або локальному сховищі). • Для доступу до захищених ресурсів клієнт відправляє токен у кожному запиті. • Сервер перевіряє токен (наприклад, його підпис і термін дії) та дозволяє або відхиляє запит.

Таким чином, ця комбінація забезпечує безпеку, масштабованість і зручність у розподілених веб-застосунках.

3.3.4 Технологія з елементами ШІ

WebGazer.js це інструмент для аналізу поведінки користувачів на веб-сторінках, який дозволяє відстежувати напрямок погляду за допомогою звичайної веб-камери. Використовується для виявлення шахрайства шляхом моніторингу погляду та нетипової поведінки учнів (наприклад, постійний погляд у бік чи часті перевірки іншого монітора) під час проходження тестів.

Це бібліотека з відкритим кодом яка використовує JavaScript та створена для інтеграції в браузерні додатки, що не потребує спеціального апаратного забезпечення. Він використовується для визначення точок, на які користувач

дивиться на екрані, шляхом обробки відео, отриманого з веб-камери (рис. 3.10, 3.11).

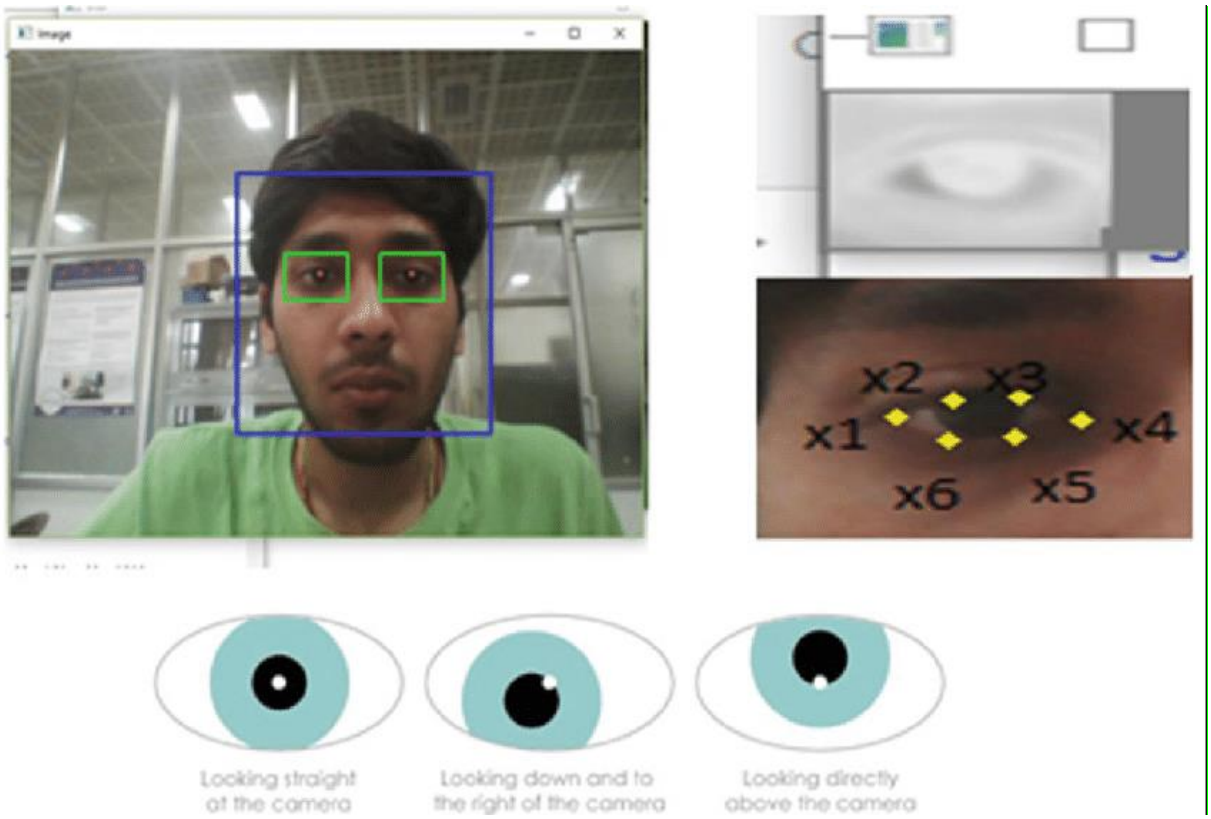

Рисунок 3.10 – Ілюстрація принципу роботи WebGazer.js

WebGazer.js використовує алгоритми машинного навчання для аналізу відео-потоку з веб-камери. Він розпізнає ключові риси обличчя (наприклад, очі, ніс, рот) і визначає напрямок погляду користувача, що дозволяє точно вказувати, на яку частину екрана дивиться користувач. Інструмент адаптується до індивідуальних особливостей обличчя кожного користувача, покращуючи точність розпізнавання з часом. Це також є елементом адаптивного навчання, що є частиною штучного інтелекту.

WebGazer.js аналізує послідовність зображень (відео) для створення моделей поведінки користувача в реальному часі. При цьому, алгоритми ШІ дозволяють ефективно фільтрувати шум і недоліки зображення, отриманого з веб-камери.

Основними перевагами використання WebGazer.js є:

1. Доступність, тому що працює з будь-якою звичайною веб-камерою без необхідності купівлі додаткового обладнання.

2. Легка інтеграція, тому що бібліотека легко додається у веб-застосунки завдяки простоті налаштування.

3. Відкритий код

Недоліком можна вважати те, що робота може погіршуватися при недостатньому освітленні.

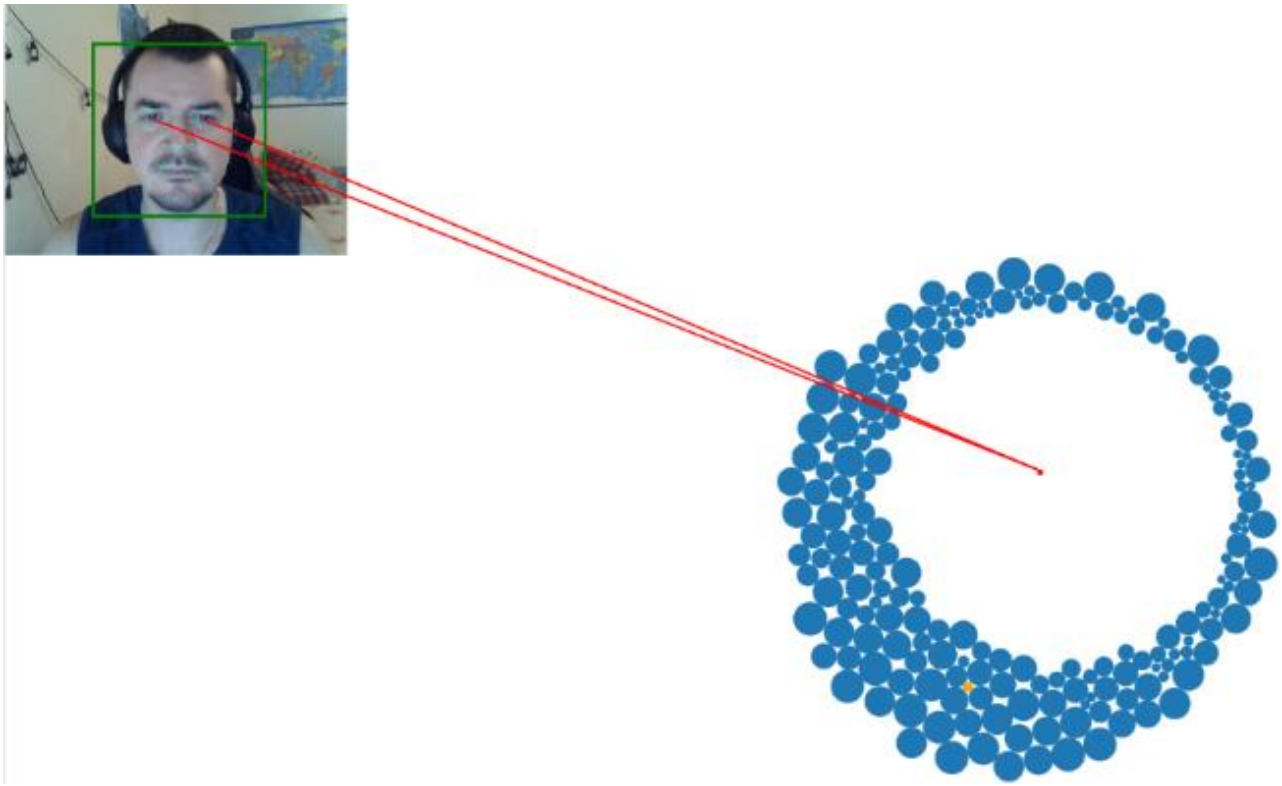

Рисунок 3.11 – Ілюстрація тренування ШІ в WebGazer.js

Отже, WebGazer.js є прикладом практичного використання технологій ШІ в освітніх застосунках, що працюють у браузері без спеціального апаратного забезпечення. Це перспективне рішення для інтеграції моніторингу погляду, яке особливо актуальне для систем із акцентом на забезпечення академічної доброчесності.

3.4. Проектування інтерфейсу користувача

Ефективний і інтуїтивно зрозумілий інтерфейс користувача є ключовим компонентом. Він також повинен бути функціональним та адаптованим до різних типів користувачів, забезпечуючи їх зручність під час роботи.

Для оптимізації процесу проектування був використаний так званий принцип «best practices», коли за основу береться багаторічний досвід успішних представників на ринку, аналізується та адапнується під цьогочасні потреби та можливості. Досвід використання таких платформ демонструє важливість інтеграції найкращих практик у дизайн інтерфейсу. Для цього були розглянуті основні елементи інтерфейсу цих платформ, їх переваги, недоліки, а також те, як ці елементи можуть бути адаптовані до нашої системи.

Роботу з проектування нашої системи було розпочато з аналізу платформи Google Forms (рис. 3.12, 3.13).

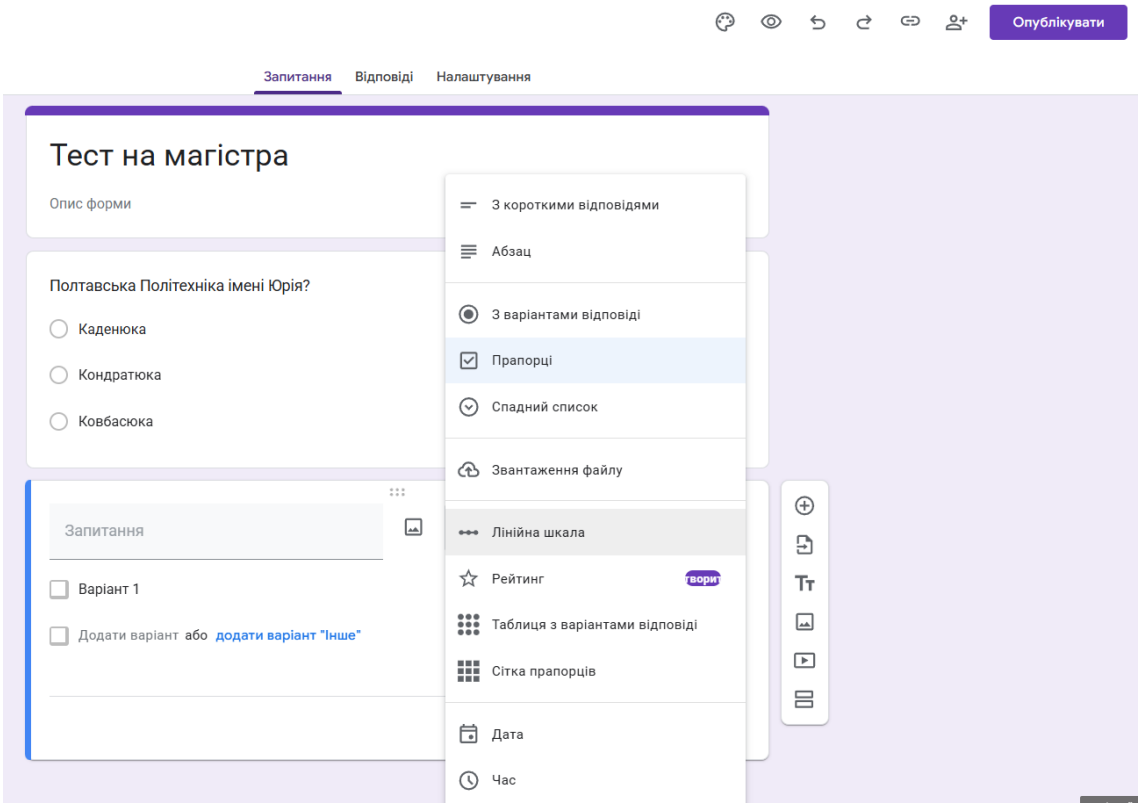

Рисунок 3.12 – Базові функції створення та редагування питань в системі Google Forms

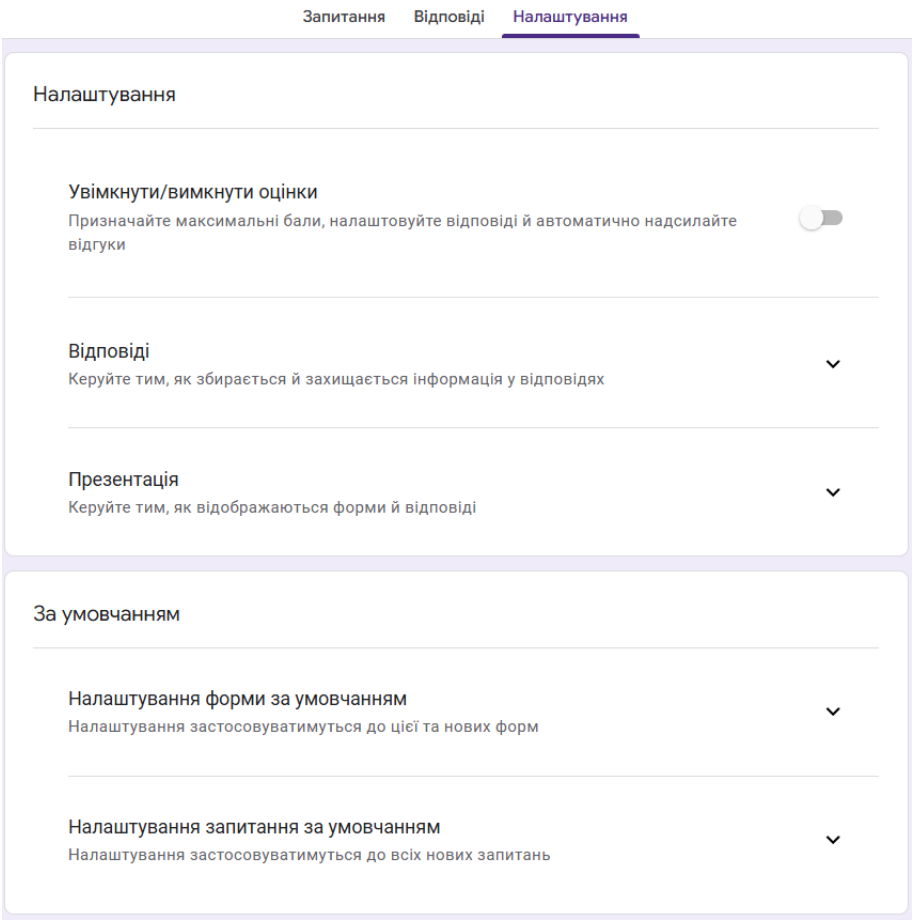

Рисунок 3.13 – Розширені функції налаштування тестів в системі Google Forms

Після чого, аналіз було продовжено, протк вже платформи Google Forms (рис. 3.14, 3.15, 3.16).

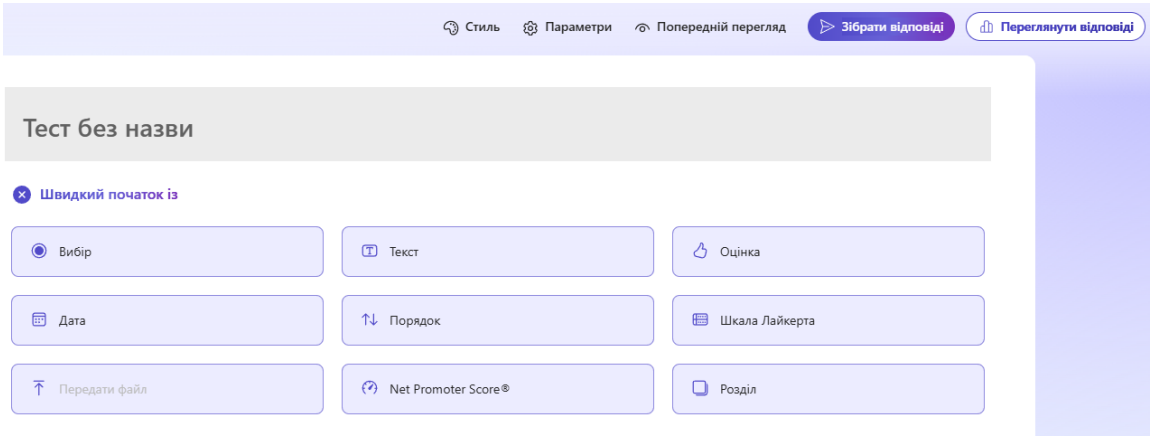

Рисунок 3.14 – Базові функції створення та питань в системі Microsoft Forms

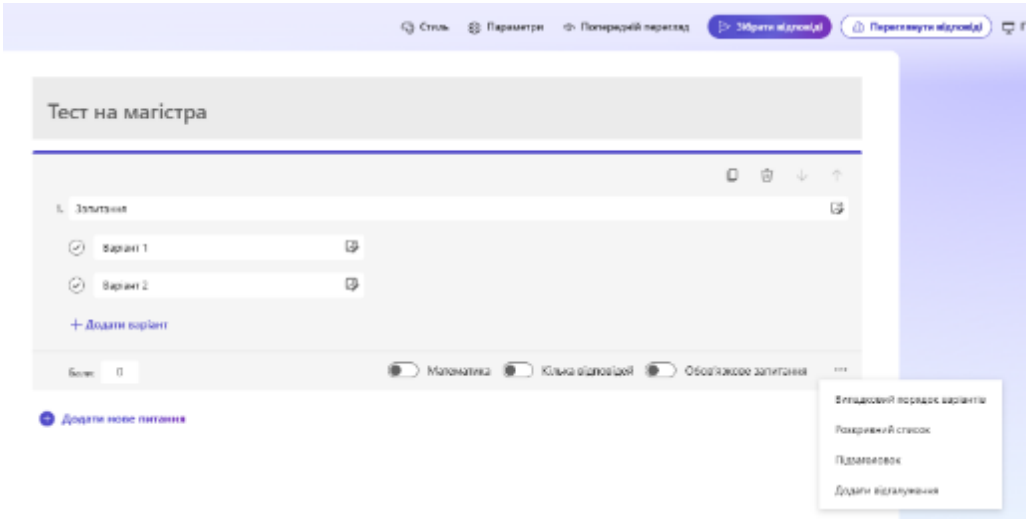

Рисунок 3.15 – Базові функції створення та налаштування питань в системі Microsoft Forms

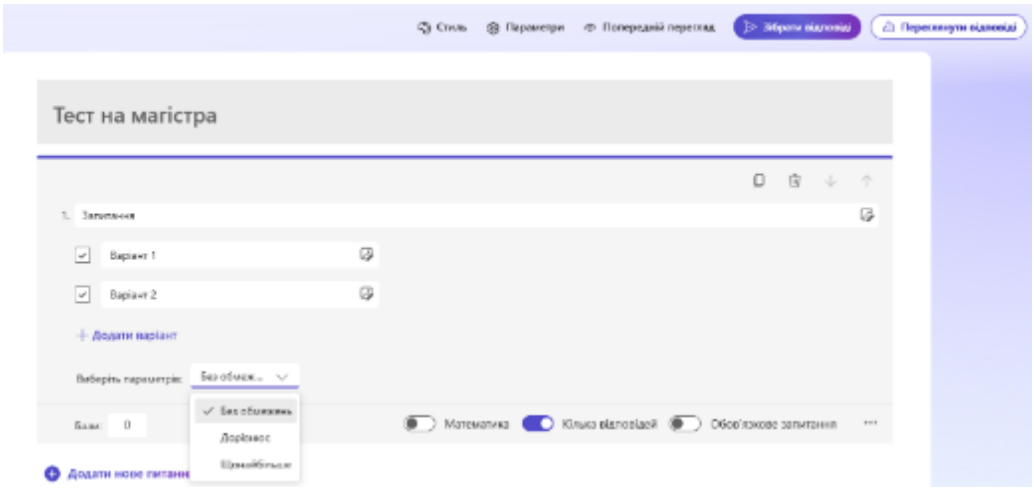

Рисунок 3.16 – Базові функції створення та налаштування питань в системі Microsoft Forms

На основі аналізу інтерфейсів Google Forms і Microsoft Forms було об'єднано ключові елементи, які необхідно впровадити в нашу систему тестування знань:

1. Інтуїтивний редактор форм з низьким «порогом входу» для створення, налаштування та проходження тестів
2. Використання тільки зрозумілих
3. Зрозумілий алгоритм додавання питань та їх сортування.
4. Можливість створення адаптивної логіки переходів між питаннями.

5. Підтримка мультимедійних елементів, таких як зображення.

6. Візуальна відмінність різніх типів питань (як при створенні, так і при проходженні).

7. Індикатор прогресу тестування, що показує завершеність тесту.

8. Миттєве відображення результатів після завершення тесту.

9. Зрозуміла нотифікація про порушення правил проходження тестування. У випадку розробляємою мною системи, основними є:

- Інтеграція з WebGazer.js для моніторингу поведінки учасників під час тестування.
- Контроль перемикання вкладок, відключення камери або виходу з повноекранного режиму.

Впровадження цих елементів інтерфейсу дозволить створити сучасну, зручну, інтуітивно зрозумілу та функціональну систему тестування знань. Комбінація найкращих практик із Google Forms і Microsoft Forms гарантує високу якість користувацького досвіду, ефективність управління тестами та забезпечення академічної доброчесності.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

4.1 Комплексний моніторинг поведінки користувача під час тестування

У сучасних системах тестування знань забезпечення доброчесності та надійності оцінювання є критично важливим завданням. Для досягнення цієї мети було впроваджено механізми комплексного моніторингу поведінки користувача, які дозволяють виявляти порушення та забезпечувати дотримання правил тестування (Додаток Б). Система моніторингу включає наступні функції:

1. Відстеження погляду (GAZE_AWAY) є ключовою функцією для виявлення випадків, коли користувач дивиться за межі екрану і базується на використанні WebGazer.js. Основні аспекти:

- Постійний моніторинг: система аналізує координати погляду в реальному часі, забезпечуючи безперервний контроль за поведінкою користувача.
- Толерантність до помилок: впроваджено алгоритми врахування межових випадків, які дозволяють уникнути фальшивих спрацьовувань.
- Збереження даних: координати погляду фіксуються для подальшого аналізу та формування звітів про потенційні порушення.

2. Моніторинг камери (CAMERA_DISABLED) – ця функція спрямована на забезпечення постійного доступу до веб-камери:

- Перевірка доступності: система перевіряє, чи увімкнена камера перед початком тестування.

- Контроль стану підключення: відстежується будь-яке відключення камери під час тесту.
- Обробка помилок: реалізовано обробку несправностей під час ініціалізації WebGazer.js, що дозволяє уникнути переривання тестування.

3. Відстеження фокусу (BROWSER_UNFOCUS) для забезпечення активності тестового вікна є критичним для запобігання шахрайству:

- Моніторинг активності: система відстежує, чи залишається вікно тесту активним протягом усього часу.
- Фіксація втрат *фокусу:* записується тривалість часу, коли користувач виходить з тестового вікна.
- Спроби відновлення: після втрати фокусу користувачу надсилається повідомлення з попередженням.

4. Перемикання вкладок (TAB_SWITCH) – цей механізм дозволяє контролювати зміну вкладок у браузері:

- Використання Visibility API: система виявляє, коли користувач переходить на іншу вкладку.
- Фіксація часу: зберігається час перемикання для аналізу частоти порушень.
- Багатоплатформенна підтримка: алгоритми адаптовані для роботи в різних браузерах.

5. Повноекранний режим (FULL_SCREEN_EXIT) – підтримання повноекранного режиму забезпечує контроль над тестовим середовищем:

- Примусове включення: тест починається лише в повноекранному режимі.
- Моніторинг виходу: система відстежує будь-які спроби виходу з режиму.
- Підтримка браузерів: реалізована сумісність із різними префіксами браузерів для коректної роботи функції.

6. Моніторинг дисплеїв (MULTIPLE_DISPLAYS) спрямований на запобігання використанню додаткових моніторів:

- Перевірка конфігурації: визначається кількість активних дисплеїв перед початком тесту.
- Динамічний моніторинг: система періодично перевіряє зміни конфігурації під час тестування.
- Запобігання порушенням: у разі виявлення нового дисплея користувач отримує відповідне попередження.

4.2 Фіксація та обробка порушень

Система автоматично фіксує кожне порушення, надаючи такі дані:

- Тип порушення: що саме було зафіксовано (втрата фокусу, відключення камери тощо).
- Час події: точний час виявлення порушення.
- Тривалість: (за наявності) тривалість некоректної поведінки.
- Технічні деталі: координати, розміри екрану, кількість дисплеїв тощо.
- Контекст браузера: інформація про платформу та середовище користувача.

Додаткові механізми безпеки:

- Автоматичне завершення тесту: якщо зафіксовано критичні порушення, система завершує тест.
- Відновлення безпечного стану: система намагається повернути тестове середовище у відповідний стан (наприклад, повторно вмикає повноекранний режим).
- Розширена діагностика: збирається детальна інформація для аналізу проблем та формування звітності.

Таким чином, запропонована система комплексного моніторингу забезпечує високий рівень доброчесності під час тестування, знижуючи ризики шахрайства та створюючи прозоре середовище для об'єктивного оцінювання знань.

4.3 Структура бази даних

У реалізованій системі тестування знань для зберігання та обробки даних використовується реляційна база даних MySQL (рис.4.1). Вибір MySQL обумовлений її високою продуктивністю, надійністю та широкими можливостями для масштабування. MySQL забезпечує ефективне управління великими обсягами даних і підтримує складні запити завдяки використанню SQL. Завдяки відкритому вихідному коду, ця база даних є економічно вигідним рішенням для освітніх платформ, які потребують гнучкості й стабільності.

Структура БД в форматі SQL наведена у Додатку А.

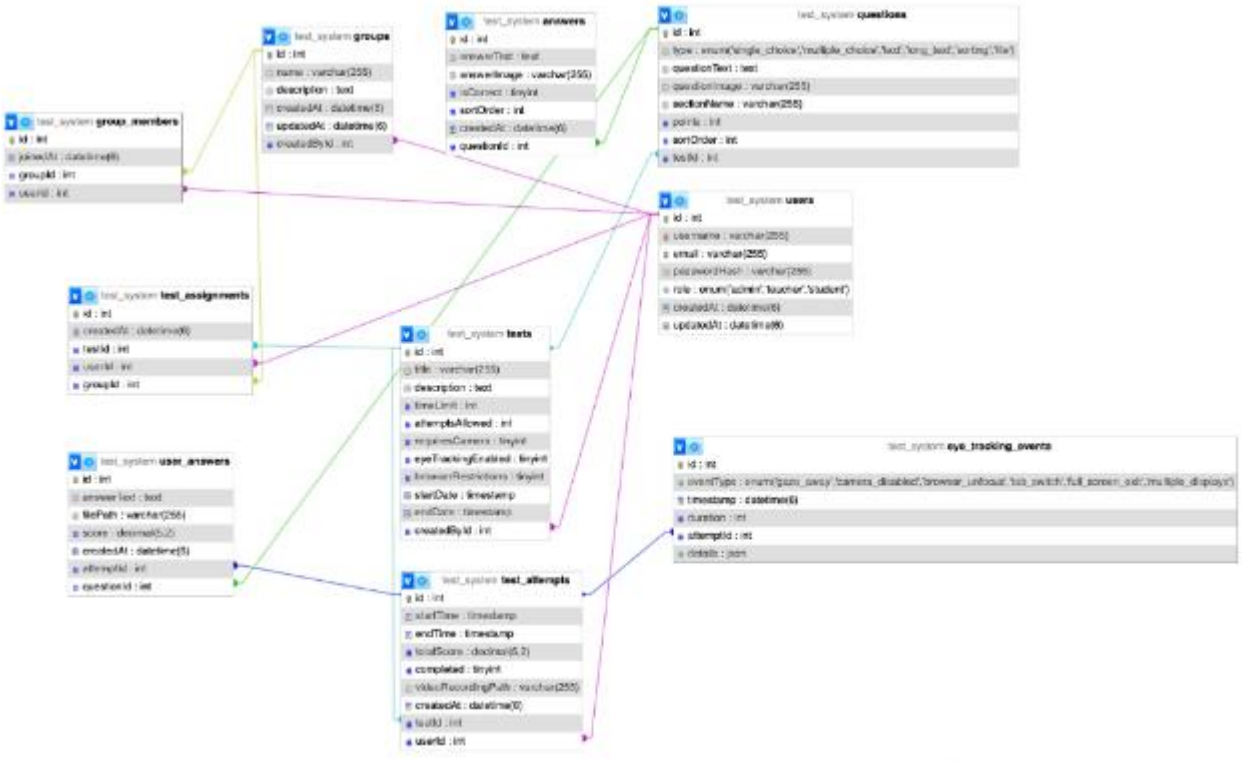

Рисунок 4.1 – Візуалізація таблиць БД та зв’язків між ними

4.4 Реалізація серверної частини системи

Серверна частина системи є основою для забезпечення надійного функціонування клієнт-серверної архітектури. Вона відповідає за обробку запитів клієнтів, зберігання даних, реалізацію бізнес-логіки та забезпечення безпеки. У нашій системі реалізовано серверну частину через REST API (Representational State Transfer Application Programming Interface), що дозволяє ефективно організувати взаємодію між клієнтською частиною та сервером. Якщо говорити більш конкретно, то це архітектурний стиль і набір принципів для побудови веб-сервісів, які дозволяють взаємодіяти між клієнтом і сервером через інтернет, використовуючи HTTP-протокол.

Таблиця 4.1 – Список використовуємих модулів, методів та ресурсів

Модуль	**Метод**	**Ресурс (endpoint)**	**Опис**
Auth (authentication endpoints)	POST	/auth/register	Регістрація нових юзерів (ALL)
	POST	/auth/login	Юзер логін (ALL)
	GET	/auth/profile	Отримати профіль юзера (AUTHORIZED)
Tests (test management endpoints)	GET	/tests	Отримати всі тести (ADMIN, OWNER)
	POST	/tests	Створити тест (ADMIN, TEACHER)
	GET	/tests/available	Отримати доступні тести (STUDENT)
	GET	/tests/{id}	Отримати тест по ID (STUDENT, ADMIN, OWNER)
	PUT	/tests/{id}	Оновити тест (ADMIN, TEACHER)
	DELETE	/tests/{id}	Видалити тест (ADMIN, TEACHER)
	POST	/tests/{id}/questions	Додати питання (ADMIN, TEACHER)
	PUT	/tests/questions/{id}	Оновити питання (ADMIN, TEACHER)
	DELETE	/tests/questions/{id}	Видалити питання (ADMIN, TEACHER)
	POST	/tests/{id}/assign	Назначити тест (ADMIN, TEACHER)
	GET	/attempts	Отримати всі спроби (ADMIN, TEACHER)

Attempts (test attempts endpoints)	GET	/attempts/my	Отримати спроби студента (STUDENT)
	GET	/attempts/{id}	Отримати спробу по ID (OWNER)
	POST	/attempts/start	Почати спробу (STUDENT)
	POST	/attempts/{id}/answer	Відправити відповідь (STUDENT)
	GET	/attempts/{id}/events	Отримати моніторинг подій (OWNER, TEACHER)
	POST	/attempts/{id}/events	Записати моніторинг події (STUDENT)
	PUT	/attempts/{id}/complete	Завершити спробу (STUDENT)
Users (User management endpoints)	GET	/users	Отримати всіх юзерів (ADMIN, TEACHER)
	POST	/users	Створити юзера (ADMIN)
	GET	/users/{id}	Отримати юзера по ID (ADMIN, TEACHER)
	PUT	/users/{id}	Оновити юзера (ADMIN)
	DELETE	/users/{id}	Видалити юзера (ADMIN)
Groups (Group management endpoints)	GET	/groups	Отримати всі групи (ADMIN, TEACHER)
	POST	/groups	Створити групу (ADMIN, TEACHER)
	GET	/groups/{id}	Отримати групу по ID (ADMIN, TEACHER)
	PUT	/groups/{id}	Оновити групу (ADMIN, TEACHER)
	DELETE	/groups/{id}	Видалити групу (ADMIN, TEACHER)
	POST	/groups/{id}/members	Додати юзера у групу (ADMIN, TEACHER)
	DELETE	/groups/{id}/members/{userId}	Видалити юзера з групи (ADMIN, TEACHER)

4.5 Реалізація клієнтської частини системи

Клієнтська частина системи є інтерфейсом, через який користувачі взаємодіють із платформою. Реалізація клієнтської частини орієнтована на забезпечення зручності, інтерактивності та безпеки.

Завдяки компонентно-орієнтованій архітектурі, основною бібліотекою для побудови користувацького інтерфейсу є React.js. Вона ж забезпечує динамічну взаємодію з користувачем без необхідності перезавантаження сторінки.

Уся клієнтська частина побудована з невеликих, незалежних компонентів, кожен із яких виконує окрему функцію (наприклад, форма авторизації, сторінка тесту, звіт результатів). Додаток працює як єдина сторінка, де всі зміни відбуваються динамічно без перезавантаження. Завдяки REST API взаємодія з серверною частиною здійснюється через стандартизовані запити API, сервер відповідає даними у форматі JSON, які обробляються на клієнтській стороні. Також впроваджено JWT (JSON Web Token) для авторизації користувачів та захисту від несанкціонованого доступу через перевірку токенів на кожному запиті.

Нижче розглянемо основні елементи клієнтської частини:

1. Інтерфейс авторизації, а саме поля для введення логіна та пароля

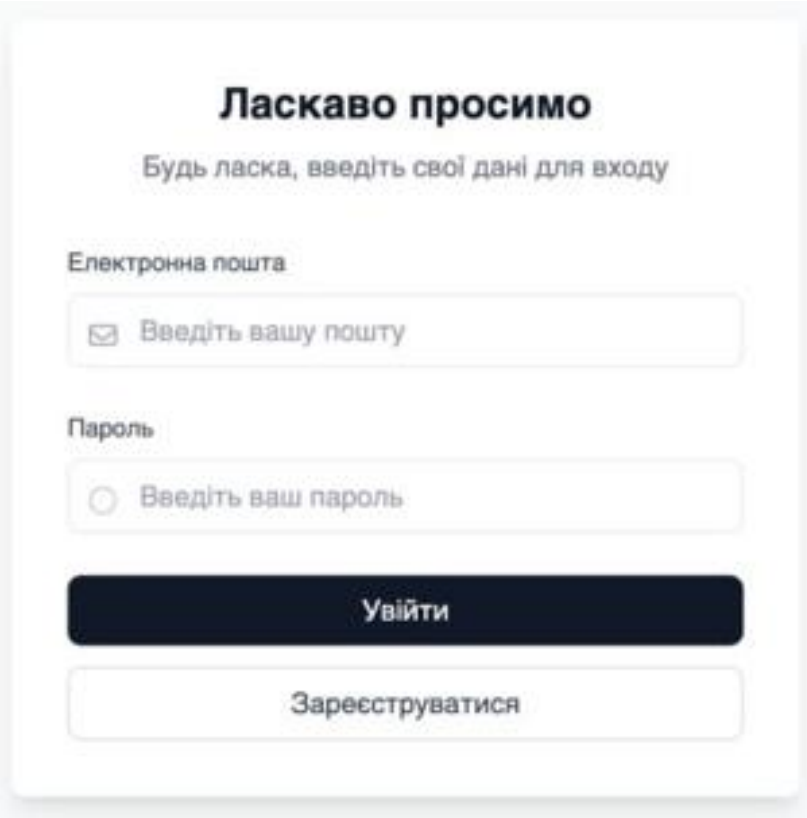

Рисунок 4.2 – Інтерфейс авторизації

2. Сторінка адміністратора, яка надає можливість переглядати, додавати, редагувати та видаляти як користувачів, так і групи.

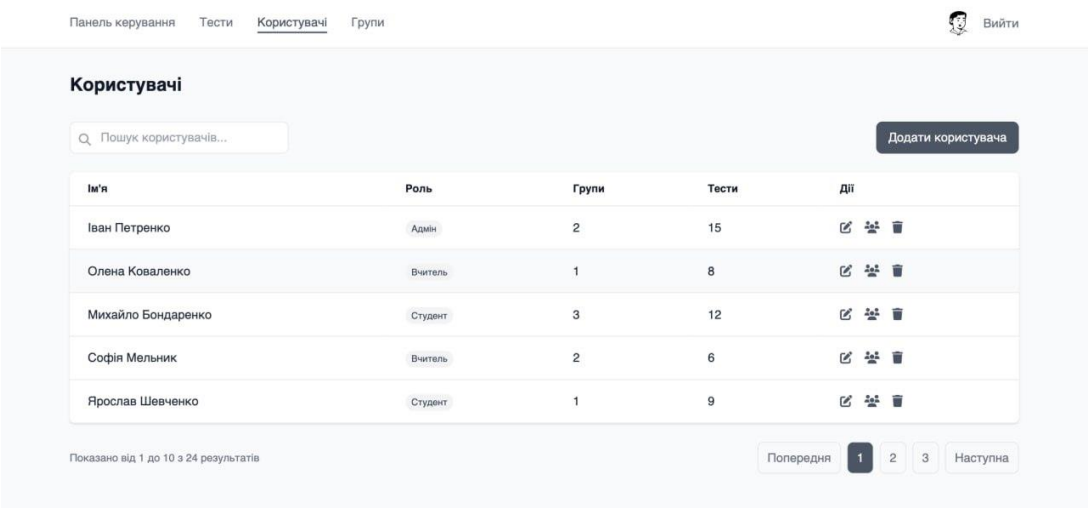

Рисунок 4.3 – Інтерфейс адміністрування користувачів

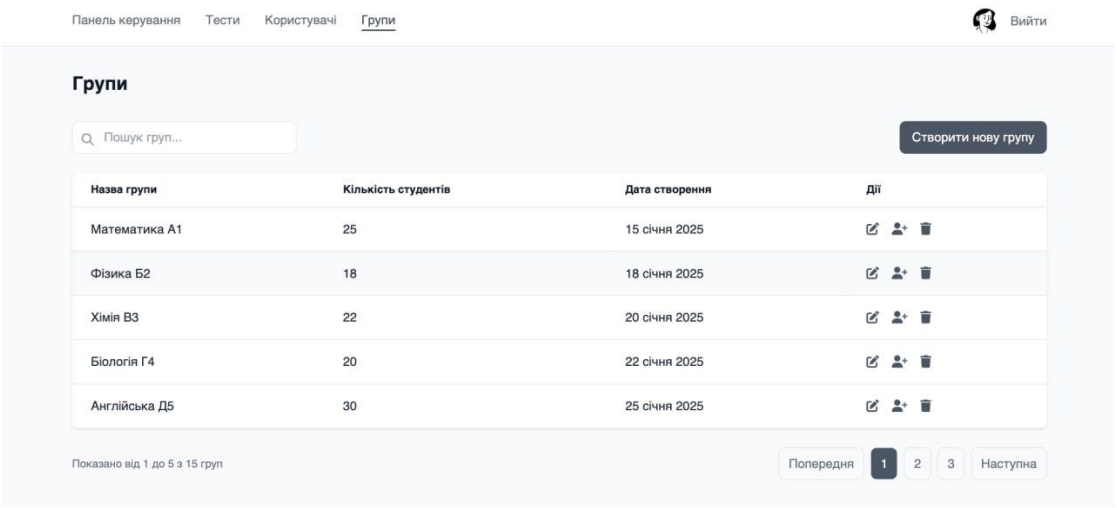

Рисунок 4.4 – Інтерфейс адміністрування груп

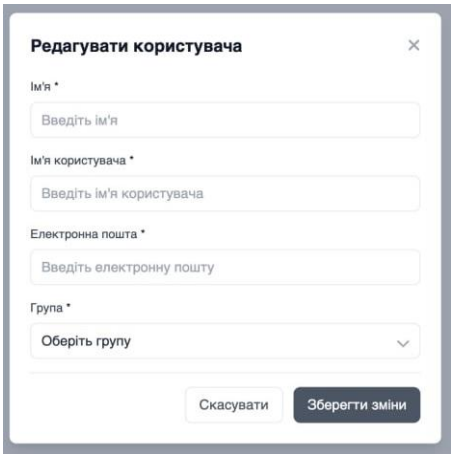

Рисунок 4.5 – Інтерфейс редагування користувача

3. Сторінка управління тестами, яка надає можливість переглядати, додавати, редагувати та видаляти всі доступні користувачу (викладачу чи адміністратору) тести, а також налаштовувати засоби запобігання порушенням академічної доброчесності (списування).

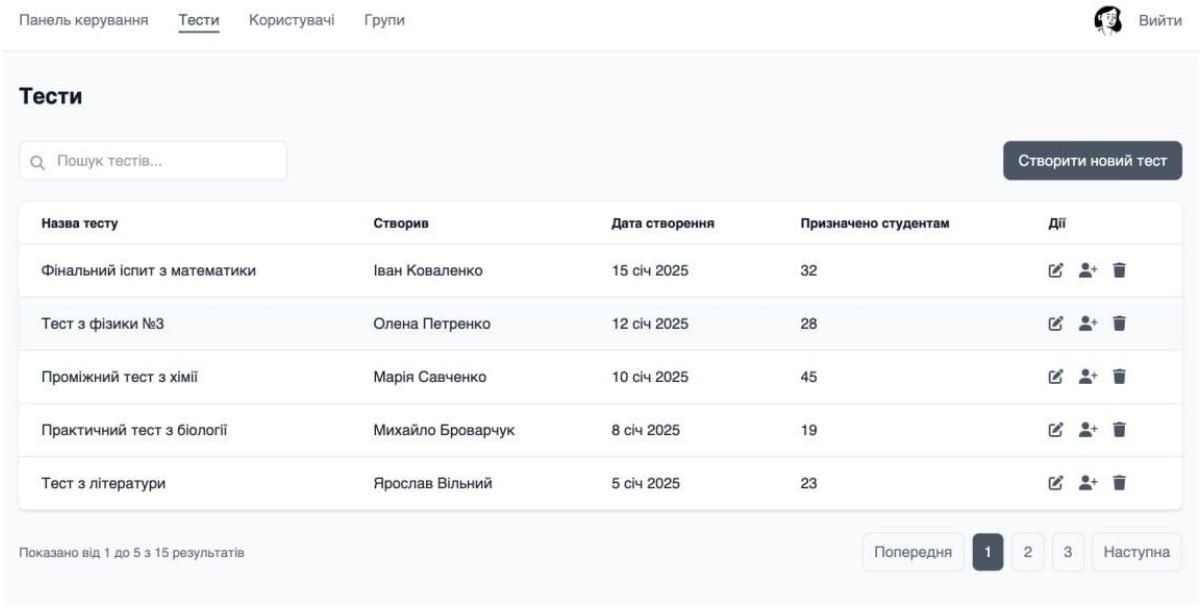

Рисунок 4.6 – Інтерфейс перегляду усих створених тестів

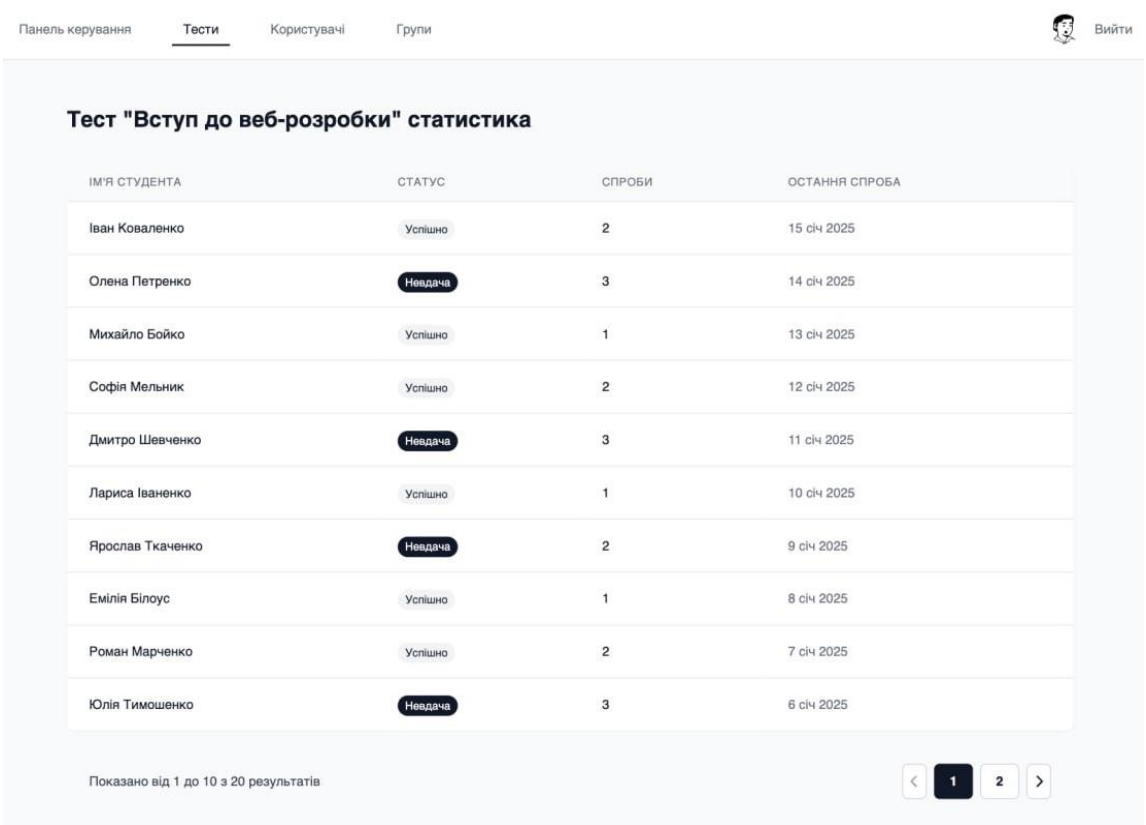

Рисунок 4.7 – Інтерфейс перегляду загальних результатів по тесту

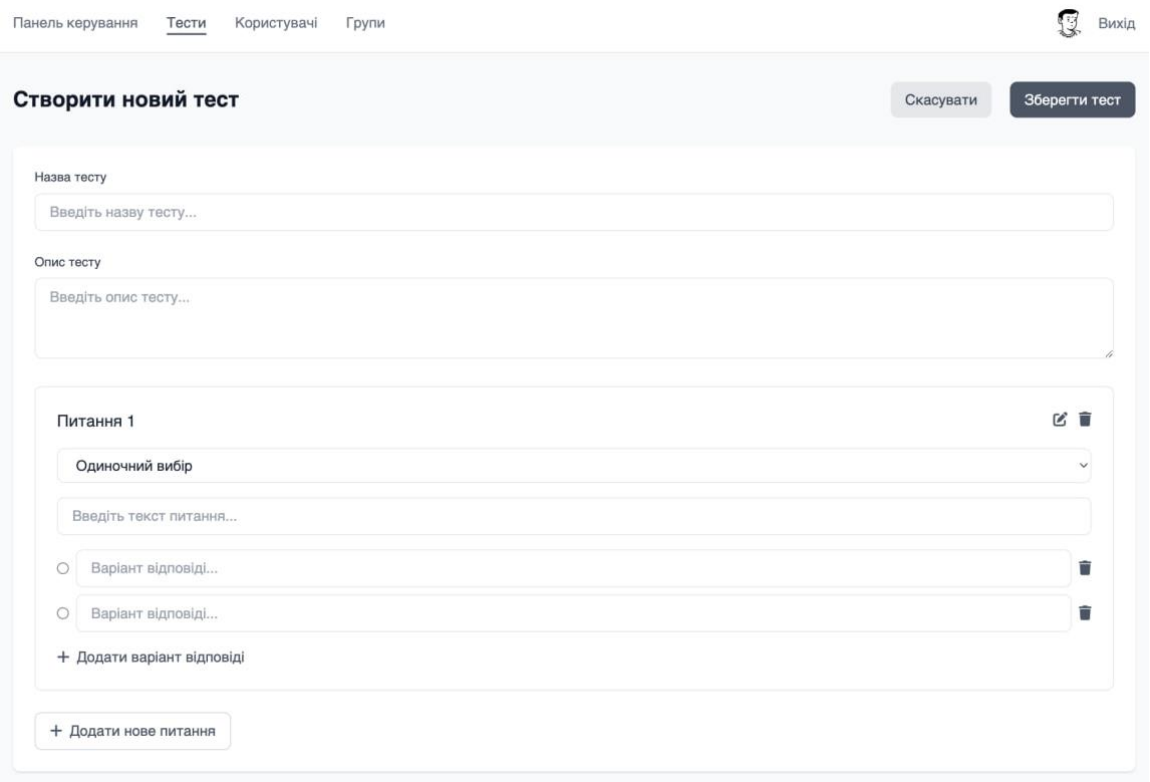

Рисунок 4.8 – Інтерфейс створення нового тесту

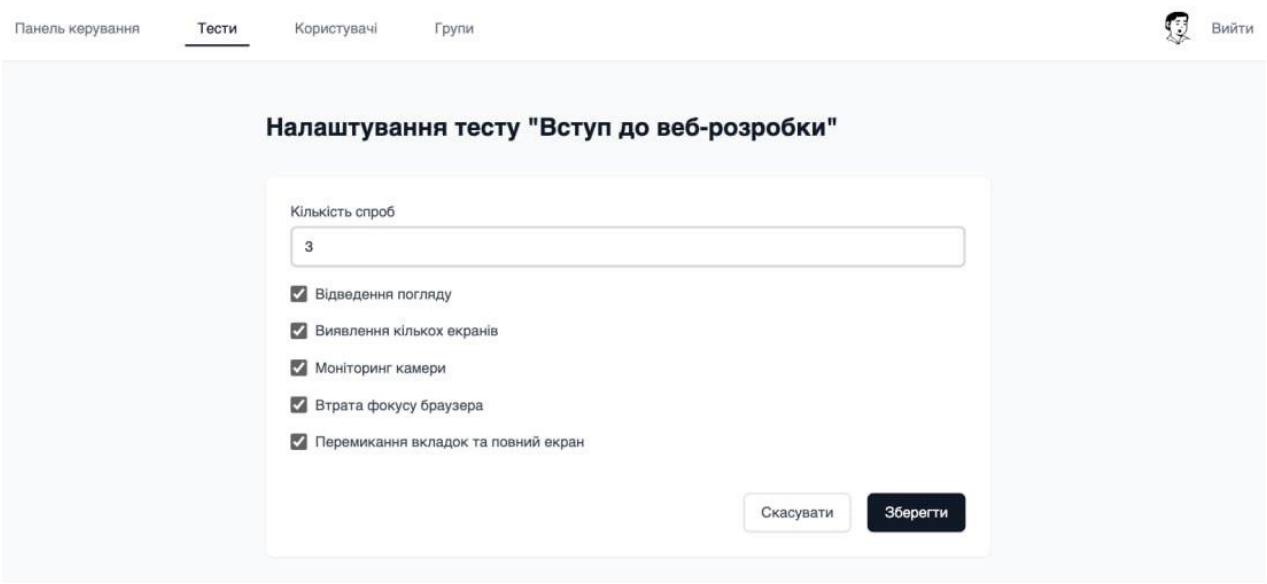

Рисунок 4.9 – Інтерфейс налаштування систем запобігання порушеням принципів академічної доброчесності

4. Сторінка проходження тесту, яка відображає питаня із можливістю вибору відповідей, інтерактивні елементи, такі як кнопка "Завершити тестування" та повідомлення про успішне завершення.

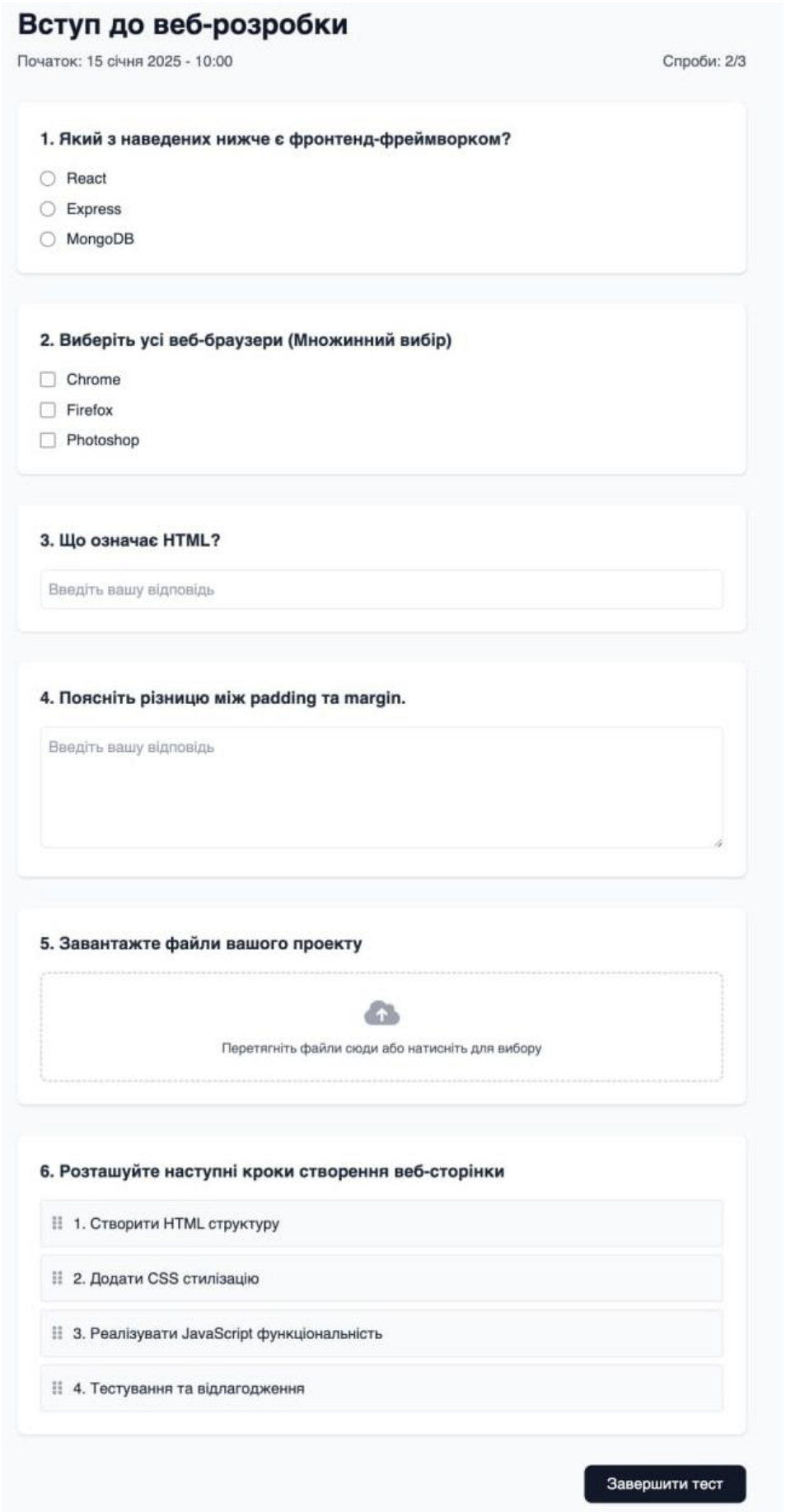

Рисунок 4.10 – Інтерфейс проходження тесту

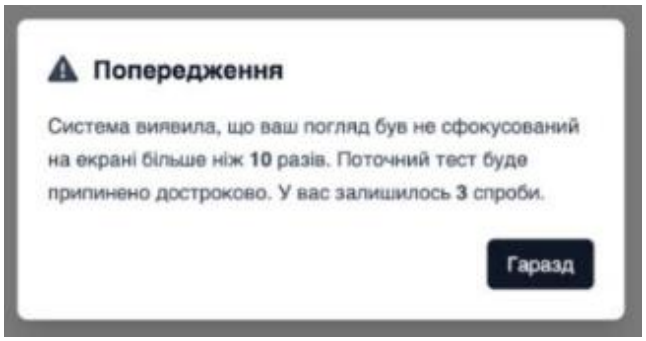

Рисунок 4.11 – Інтерфейс попередження про виявлення порушеня принципів академічної доброчесності

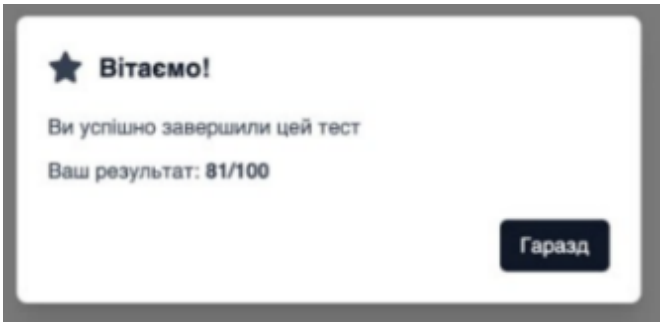

Рисунок 4.12 – Інтерфейс з результатами тестування

Таким чином, реалізація клієнтської частини системи базується на сучасних технологіях і підходах, що забезпечують зручність, продуктивність і безпеку. Компонентно-орієнтована архітектура та використання React.js дозволяють створювати масштабовані та легко підтримувані рішення, які відповідають вимогам сучасних освітніх платформ.

4.6 Проведення тестування системи: сценарії та результати

Тестування системи є важливим етапом розробки, що дозволяє перевірити її відповідність вимогам, знайти помилки та переконатися у правильності функціонування всіх компонентів. У цьому розділі описано процес проведення тестування системи, використані сценарії та отримані результати.

Для тестування системи було розроблено набір сценаріїв, що охоплюють ключові функції.

Таблиця 4.2 – Список тестових сценаріїв

Сценарій	**Аспекти**
Авторизація користувача	Перевірка входу з правильними даними
	Тестування входу з неправильними даними
Створення та управління користувачами і групами	Додавання користувачів і груп
	Редактування користувачів і груп
	Видалення користувачів і груп
Створення та управління тестами	Створення тесту з усіма типами питань
	Налаштування тесту
	Редагування тесту.
	Видалення тесту.
Проходження тесту користувачем	Перевірка коректності відображення питань
	Відстеження часу проходження тесту
	Обробка випадків, коли користувач залишає тестове вікно або закриває браузер
Аналіз результатів	Генерація звітів для викладачів
	Відображення підсумкового балу для користувача
	Перевірка коректності статистики
Функції безпеки	Відстеження поведінки через WebGazer.js
	Блокування доступу при критичних порушеннях (вихід із повноекранного режиму, вимкнення камери).

Результати тестування системи показали наступне:

1. Усі основні функції системи працюють коректно, включаючи авторизацію, створення тестів, проходження тестів та аналіз результатів.

2. Виявлено незначні помилки у валідації введених даних, які були виправлені.

3. Інтерфейс користувача є зручним та інтуїтивно зрозумілим.

4. Усі елементи інтерфейсу коректно відображаються в різних браузерах

5. Система успішно виявляє та реагує на шахрайську поведінку під час тестування.

Під час тестування були виявлені та виправлені такі проблеми:

1. Некоректна обробка випадків, коли користувач залишає тестове вікно.
2. Неповна валідація деяких типів відповідей у текстових полях.
3. Відсутність повідомлення про закінчення часу на тестування.

Проведене тестування підтвердило працездатність системи та її відповідність встановленим вимогам. Всі виявлені помилки були виправлені, і система готова до впровадження у навчальний процес. Результати тестування засвідчують високу надійність, безпеку та продуктивність системи тестування.

ВИСНОВКИ

Дослідження, проведене мною в межах магістерської роботи, дозволило розробити актуальну для ринку систему тестування знань TestGO! із використанням багатьох сучасних технологій, при цьому акцентувавши велику увагу на впровадженні підходів з елементами ШІ для запобігання недоброчесної поведінки під час тестування.

Проведений аналіз науково-методичної літератури та практичних підходів до тестування знань показав значний потенціал цифрових систем у підвищенні ефективності навчання. Вивчення систем Moodle, Google Forms, Microsoft Forms та інших виявило їх переваги та недоліки, що стало основою для розробки власної збалансованої системи TestGO!

Як вже зазначалося вище, важливою звитягою цієї роботи є впровадження моніторингу процесу тестування. Було використано технології розпізнавання поведінкових патернів з елементами ШІ, зокрема стеження за очима учня за допомогою веб-камери. Також було інтегровано аналіз переходів між вкладками браузера, іншими активними програмами, моніторами, вихід з повноекранного режиму для виявлення спроб маніпулювання академічною доброчесністю. Розроблені механізми дозволяють виявляти спроби списування в реальному часі та попереджати користувачів про порушення.

Важливим аспектом створеної мною системи тестування знань є реалізована клієнт-серверна архітектура, яка забезпечує чітке розділення функцій між учасниками процесу (учнями, викладачами та адміністраторами), високий рівень безпеки даних завдяки шифруванню, а також гарантує гнучкість, модульність та можливість подальшого масштабування функціоналу системи TestGO! В свою чергу, слід відзначити, що було обрано мову програмування TypeScript та середовище розробки IntelliJ WebStorm, які забезпечили ефективність і стабільність роботи продукту, а використані технології відповідають всім сучасним стандартам ринку.

Розроблена мною система тестування знань TestGO! відповідає сучасним стандартам якості і може бути використана у навчальних закладах для оцінювання знань учнів. Інтеграція рішень, в тому числі і на основі ШІ, дозволяє ефективно контролювати академічну доброчесність, що робить цю систему перспективною для впровадження на різних рівнях освіти та сприятиме покращенню навчального процесу завдяки об’єктивності оцінювання, підвищенню мотивації учнів та забезпеченню академічної доброчесності. Результати магістерської роботи є основою для подальших досліджень у сфері освітніх технологій та їх практичного застосування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cattell, J. M. (1890). Mental tests and measurements. Mind, 15(59), 373–381. https://doi.org/10.1093/mind/os-XV.59.373.
2. Binet, A., & Simon, T. (1905). Méthodes nouvelles pour le diagnostic du niveau intellectuel des anormaux. *L'Année Psychologique, 11,* 191–244.
3. Thorndike, R. L. (1910-1920). Educational Psychology. *Teachers College Press, Columbia University*.
4. College Board. (1926). Scholastic Aptitude Test: Technical Report.
5. U.S. Army. (1942). Army General Classification Test: Applications and results.
6. Parshall, C. G., Spray, J. A., Kalohn, J. C., & Davey, T. (2002). *Practical considerations in computer-based testing*. Springer. https://doi.org/10.1007/978-1-4613-0083-0 .
7. Duolingo. (2023). Adaptive learning technology in language education.
8. Coursera. (2023). AI-driven assessment technologies.
9. Reynolds, C. R., Livingston, R. B., & Willson, V. (2009). *Measurement and Assessment in Education*. Pearson Education.
10. OECD. (2020). *PISA 2022 results (Volume I): What students know and can do*. OECD Publishing. https://doi.org/10.1787/5f07c754-en.
11. OECD. (2013). *PISA 2012 results: What makes schools successful? Resources, policies and practices* (Volume IV). OECD Publishing. https://doi.org/10.1787/9789264201156-en.
12. OECD. (2019). *Trends Shaping Education.* OECD Publishing. https://doi.org/10.1787/trends_edu-2019-en.
13. Anderson, L. W., & Krathwohl, D. R. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.* Pearson.
14. Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education: Principles, Policy & Practice*, 5(1), 7-74. https://doi.org/10.1080/0969595980050102.

15. Brookhart, S. M. (2013). *How to create and use rubrics for formative assessment and grading.* ASCD.
16. Shute, V. J., & Ventura, M. (2013). *Stealth assessment: Measuring and supporting learning in video games.* MIT Press. https://doi.org/10.7551/mitpress/9589.001.0001.
17. Popham, W. J. (2008). *Classroom Assessment: What Teachers Need to Know.* Pearson Education.
18. Stiggins, R. J. (2001). *Student-involved classroom assessment.* (3rd ed.). Prentice Hall.
19. Heitink, M. C., Van der Kleij, F. M., Veldkamp, B. P., Schildkamp, K., & Kippers, W. B. (2016). A systematic review of the effects of formative assessment on student learning. *Educational Research Review*, 19, 158-166. https://doi.org/10.1016/j.edurev.2016.06.001.
20. Sadler, D. R. (1989). Formative assessment and the design of instructional systems. *Instructional Science*, 18(2), 119-144. https://doi.org/10.1007/BF00117714.
21. Vygotsky, L. S. (1978). *Mind in Society: The Development of Higher Psychological Processes.* Harvard University Press.
22. Pellegrino, J. W., Chudowsky, N., & Glaser, R. (2001). *Knowing What Students Know: The Science and Design of Educational Assessment.* National Academies Press. https://doi.org/10.17226/10019.
23. Miller, M. D., Linn, R. L., & Gronlund, N. E. (2012). *Measurement and Assessment in Teaching.*
24. Zhao, J., & Wagner, A. B. (2020). *A comprehensive review of adaptive learning systems*. Educational Technology Research and Development, 68(6), 1–28. https://doi.org/10.1007/s11423-020-09788-z.
25. Stallings, W. (2017). *Operating Systems: Internals and Design Principles.*
26. Chen, L., & Wang, Y. (2018). *Data Security in Distributed Systems.* Springer. https://doi.org/10.1007/978-3-319-75356-0.
27. Moodle Documentation (https://moodle.org/*)* (дата звернення 16.10.2024).

28. Google Workspace Help (https://support.google.com/) (дата звернення 16.10.2024).
29. Microsoft Forms Documentation (https://support.microsoft.com/forms) (дата звернення 09.11.2024).
30. Kahoot Resources (https://kahoot.com/) (дата звернення 23.10.2024).
31. Blackboard Help Center (https://help.blackboard.com/) (дата звернення 12.11.2024).
32. Quizizz Help Center (https://support.quizizz.com/) (дата звернення 12.11.2024).
33. Canvas LMS Documentation (https://community.canvaslms.com/) (дата звернення 09.10.2024).
34. ProProfs Knowledge Base (https://www.proprofs.com/) (дата звернення 15.11.2024).
35. ClassMarker Online Testing Documentation (https://www.classmarker.com/) (дата звернення 15.11.2024).
36. Залозна С.В. *Методика використання проектних технологій. "Проекти в школі"* (https://naurok.com.ua/metodika-vikoristannya-proektnih-tehnologiy-proekti-v-shkoli-288319.html) (дата звернення 10.12.2024).
37. Хміль Н.А. *Використання віртуальної та доповненої реальності в українській освіті* (https://zenodo.org/records/8251886) (дата звернення 19.12.2024).
38. Колективна монографія. *Діяльнісні засади підготовки майбутніх компетентних фахівців в умовах сучасних викликів* (http://eprints.zu.edu.ua/39737/1/%D0%90%D0%BD%D1%82%D0%BE%D0%BD%D0%BE%D0%B2_%D0%BC%D0%BE%D0%BD%D0%BE_1_%D0%A0%D0%BE%D0%B7%D0%B4%D1%96%D0%BB.pdf) (дата звернення 23.12.2024).
39. Сікора Я. *Методичні рекомендації до розробки та використання адаптивних тестових завдань*

http://eprints.zu.edu.ua/41797/1/metod_test.pdf) (дата звернення 23.11.2024).

40. С.Л. Загребельний *Використання комп'ютерного адаптивного тестування у ДДМА на платформі Moodle (https://ddpu.edu.ua/texel/index.php/TeXEL/article/view/18/15)* (дата звернення 24.12.2024).
41. Пасічник О.В. *Збірка статей присвячена платформі Moodle* (https://oksanapasichnyk.wordpress.com/category/moodle/) (дата звернення 24.12.2024).
42. Онлайн-платформа (https://foxminded.ua/) (дата звернення 19.12.2024).
43. WebGazer.js GitHub Repository (https://github.com/brownhci/WebGazer) (дата звернення 15.12.2024).

ДОДАТОК А

СТРУКТУРА БАЗИ ДАНИХ В ФОРМАТІ SQL

```sql
CREATE TABLE `answers` (
  `id` int NOT NULL,
  `answerText` text NOT NULL,
  `answerImage` varchar(255) DEFAULT NULL,
  `isCorrect` tinyint NOT NULL DEFAULT '0',
  `sortOrder` int DEFAULT NULL,
  `createdAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  `questionId` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `eye_tracking_events` (
  `id` int NOT NULL,
  `eventType` enum('gaze_away','camera_disabled','browser_unfocus',
    'tab_switch','full_screen_exit','multiple_displays')
    CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `timestamp` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  `duration` int DEFAULT NULL,
  `attemptId` int DEFAULT NULL,
  `details` json DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `groups` (
  `id` int NOT NULL,
  `name` varchar(255) NOT NULL,
  `description` text,
  `createdAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  `updatedAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6)
    ON UPDATE CURRENT_TIMESTAMP(6),
  `createdById` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `group_members` (
  `id` int NOT NULL,
  `joinedAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  `groupId` int DEFAULT NULL,
  `userId` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `questions` (
  `id` int NOT NULL,
  `type` enum('single_choice','multiple_choice','text','long_text','sorting','file')
    NOT NULL,
  `questionText` text NOT NULL,
  `questionImage` varchar(255) DEFAULT NULL,
  `sectionName` varchar(255) DEFAULT NULL,
  `points` int NOT NULL DEFAULT '1',
  `sortOrder` int NOT NULL,
  `testId` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `tests` (
  `id` int NOT NULL,
  `title` varchar(255) NOT NULL,
  `description` text,
  `timeLimit` int DEFAULT NULL,
  `attemptsAllowed` int NOT NULL DEFAULT '1',
  `requiresCamera` tinyint NOT NULL DEFAULT '0',
  `eyeTrackingEnabled` tinyint NOT NULL DEFAULT '0',
  `browserRestrictions` tinyint NOT NULL DEFAULT '0',
  `startDate` timestamp NULL DEFAULT NULL,
  `endDate` timestamp NULL DEFAULT NULL,
  `createdById` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `test_assignments` (
  `id` int NOT NULL,
  `createdAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  `testId` int DEFAULT NULL,
  `userId` int DEFAULT NULL,
  `groupId` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `test_attempts` (
  `id` int NOT NULL,
  `startTime` timestamp NOT NULL,
  `endTime` timestamp NULL DEFAULT NULL,
  `totalScore` decimal(5,2) DEFAULT NULL,
  `completed` tinyint NOT NULL DEFAULT '0',
  `videoRecordingPath` varchar(255) DEFAULT NULL,
  `createdAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  `testId` int DEFAULT NULL,
  `userId` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `users` (
  `id` int NOT NULL,
  `username` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `passwordHash` varchar(255) NOT NULL,
  `role` enum('admin','teacher','student') NOT NULL DEFAULT 'student',
  `createdAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  `updatedAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6)
    ON UPDATE CURRENT_TIMESTAMP(6)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```sql
CREATE TABLE `user_answers` (
  `id` int NOT NULL,
  `answerText` text NOT NULL,
  `filePath` varchar(255) DEFAULT NULL,
  `score` decimal(5,2) DEFAULT NULL,
  `createdAt` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  `attemptId` int DEFAULT NULL,
  `questionId` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


ALTER TABLE `answers`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_c38697a57844f52584abdb878d7` (`questionId`);

ALTER TABLE `eye_tracking_events`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_283eee49ae14aa9638d7464a59f` (`attemptId`);

ALTER TABLE `groups`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_e0522c4be8bab20520896919da0` (`createdById`);

ALTER TABLE `group_members`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_1aa8d31831c3126947e7a713c2b` (`groupId`),
  ADD KEY `FK_fdef099303bcf0ffd9a4a7b18f5` (`userId`);

ALTER TABLE `questions`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_94296641072b0f034d14e272cc6` (`testId`);

ALTER TABLE `tests`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_69bb0928b2d70079d6cc32267c0` (`createdById`);

ALTER TABLE `test_assignments`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_0a76f97d49cd2105f63827fbee8` (`testId`),
  ADD KEY `FK_423daa85d95ab116e4aa2657f89` (`userId`),
  ADD KEY `FK_1adf253b696efa8f8fc0f614aac` (`groupId`);

ALTER TABLE `test_attempts`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_117d71338d1aad502bf5927b1e6` (`testId`),
  ADD KEY `FK_c2d6de3ee467f1b42f22a30d5a4` (`userId`);

ALTER TABLE `users`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `IDX_fe0bb3f6520ee0469504521e71` (`username`),
  ADD UNIQUE KEY `IDX_97672ac88f789774dd47f7c8be` (`email`);
```

```
ALTER TABLE `user_answers`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_c8e6e678d62072038900fe3db5a` (`attemptId`),
  ADD KEY `FK_47a3ffddaba37b9707f93e4b140` (`questionId`);


ALTER TABLE `answers`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `eye_tracking_events`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `groups`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `group_members`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `questions`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `tests`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `test_assignments`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `test_attempts`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `users`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;

ALTER TABLE `user_answers`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;


ALTER TABLE `answers`
  ADD CONSTRAINT `FK_c38697a57844f52584abdb878d7` FOREIGN KEY (`questionId`)
    REFERENCES `questions` (`id`) ON DELETE CASCADE;

ALTER TABLE `eye_tracking_events`
  ADD CONSTRAINT `FK_283eee49ae14aa9638d7464a59f` FOREIGN KEY (`attemptId`)
    REFERENCES `test_attempts` (`id`) ON DELETE CASCADE;

ALTER TABLE `groups`
  ADD CONSTRAINT `FK_e0522c4be8bab20520896919da0` FOREIGN KEY (`createdById`)
    REFERENCES `users` (`id`) ON DELETE SET NULL;

ALTER TABLE `group_members`
  ADD CONSTRAINT `FK_1aa8d31831c3126947e7a713c2b` FOREIGN KEY (`groupId`)
    REFERENCES `groups` (`id`) ON DELETE CASCADE,
  ADD CONSTRAINT `FK_fdef099303bcf0ffd9a4a7b18f5` FOREIGN KEY (`userId`)
```

```
    REFERENCES `users` (`id`) ON DELETE CASCADE;

ALTER TABLE `questions`
  ADD CONSTRAINT `FK_94296641072b0f034d14e272cc6` FOREIGN KEY (`testId`)
    REFERENCES `tests` (`id`);

ALTER TABLE `tests`
  ADD CONSTRAINT `FK_69bb0928b2d70079d6cc32267c0` FOREIGN KEY (`createdById`)
    REFERENCES `users` (`id`);

ALTER TABLE `test_assignments`
  ADD CONSTRAINT `FK_0a76f97d49cd2105f63827fbee8` FOREIGN KEY (`testId`)
    REFERENCES `tests` (`id`) ON DELETE CASCADE,
  ADD CONSTRAINT `FK_1adf253b696efa8f8fc0f614aac` FOREIGN KEY (`groupId`)
    REFERENCES `groups` (`id`) ON DELETE CASCADE,
  ADD CONSTRAINT `FK_423daa85d95ab116e4aa2657f89` FOREIGN KEY (`userId`)
    REFERENCES `users` (`id`) ON DELETE CASCADE;

ALTER TABLE `test_attempts`
  ADD CONSTRAINT `FK_117d71338d1aad502bf5927b1e6` FOREIGN KEY (`testId`)
    REFERENCES `tests` (`id`) ON DELETE CASCADE,
  ADD CONSTRAINT `FK_c2d6de3ee467f1b42f22a30d5a4` FOREIGN KEY (`userId`)
    REFERENCES `users` (`id`) ON DELETE CASCADE;

ALTER TABLE `user_answers`
  ADD CONSTRAINT `FK_47a3ffddaba37b9707f93e4b140` FOREIGN KEY (`questionId`)
    REFERENCES `questions` (`id`),
  ADD CONSTRAINT `FK_c8e6e678d62072038900fe3db5a` FOREIGN KEY (`attemptId`)
    REFERENCES `test_attempts` (`id`);
```

ДОДАТОК Б

МОНІТОРИНГ ПОВЕДІНКИ КОРИСТУВАЧА ПІД ЧАС ТЕСТУВАННЯ

```
// src/monitoring/test-monitoring.ts
class TestMonitoring {
  private webgazer: any;
  private gazeCheckInterval: number = 500;
  private warningCount: number = 0;
  private maxWarnings: number = 3;
  private isFullScreen: boolean = false;
  private initialDisplayCount: number;

  constructor() {
    this.initialDisplayCount = window.screen.availWidth / window.screen.width;
  }

  async initialize() {
    // Ініціалізація всіх моніторингів
    await this.initializeWebGazer();
    this.setupDisplayMonitoring();
    this.setupFullScreenMonitoring();
    this.setupBrowserEvents();
    this.setupTabMonitoring();

    // Примусове включення повноекранного режиму
    await this.requestFullScreen();
  }

  /**
   * Ініціалізація WebGazer для відстеження погляду
   */
  private async initializeWebGazer() {
    try {
      this.webgazer = await WebGazer.setGazeListener((data: any, timestamp:
      number) => {
        if (data === null) {
          this.handleGazeAway();
        } else {
          this.checkGazePosition(data.x, data.y);
        }
      }).begin();

      // Калібрування
      await this.calibrate();
    } catch (error) {
      this.recordEvent({
        type: MonitoringEventType.CAMERA_DISABLED,
        timestamp: Date.now(),
```

```
        details: { error: error.message }
      });
    }
  }

  /**
   * Моніторинг підключених дисплеїв
   */
  private setupDisplayMonitoring() {
    // Перевірка при ініціалізації
    this.checkDisplays();

    // Відстеження змін конфігурації дисплеїв
    if ('screen' in window && 'orientation' in screen) {
      screen.orientation.addEventListener('change', () => {
        this.checkDisplays();
      });
    }

    // Періодична перевірка
    setInterval(() => this.checkDisplays(), 5000);
  }

  private checkDisplays() {
    if (window.screen.availWidth /
        window.screen.width !== this.initialDisplayCount) {
      this.recordEvent({
        type: MonitoringEventType.MULTIPLE_DISPLAYS,
        timestamp: Date.now(),
        details: {
          initialCount: this.initialDisplayCount,
          currentCount: window.screen.availWidth / window.screen.width
        }
      });
    }
  }

  /**
   * Моніторинг повноекранного режиму
   */
  private setupFullScreenMonitoring() {
    document.addEventListener('fullscreenchange', () => {
      const isCurrentlyFullScreen = !!document.fullscreenElement;

      if (this.isFullScreen && !isCurrentlyFullScreen) {
```

```
        this.recordEvent({
          type: MonitoringEventType.FULL_SCREEN_EXIT,
          timestamp: Date.now()
        });

        // Спроба відновити повноекранний режим
        this.requestFullScreen();
      }

      this.isFullScreen = isCurrentlyFullScreen;
    });
  }

  /**
   * Розширений моніторинг вкладок та фокусу
   */
  private setupTabMonitoring() {
    let unfocusStartTime: number;
    let isTabActive = true;

    // Відстеження активності вкладки
    document.addEventListener('visibilitychange', () => {
      if (document.hidden) {
        unfocusStartTime = Date.now();
        isTabActive = false;

        this.recordEvent({
          type: MonitoringEventType.TAB_SWITCH,
          timestamp: unfocusStartTime
        });
      } else {
        if (!isTabActive) {
          const duration = Date.now() - unfocusStartTime;
          this.recordEvent({
            type: MonitoringEventType.BROWSER_UNFOCUS,
            timestamp: unfocusStartTime,
            duration: duration
          });
        }
        isTabActive = true;
      }
    });

    // Відстеження фокусу вікна
    window.addEventListener('blur', () => {
```

```
      const timestamp = Date.now();
      unfocusStartTime = timestamp;

      this.recordEvent({
        type: MonitoringEventType.BROWSER_UNFOCUS,
        timestamp: timestamp
      });
    });


    window.addEventListener('focus', () => {
      if (unfocusStartTime) {
        const duration = Date.now() - unfocusStartTime;
        this.updateEvent
            (MonitoringEventType.BROWSER_UNFOCUS, { duration });
      }
    });
  }

  /**
   * Запит на повноекранний режим
   */
  private async requestFullScreen() {
    try {
      const elem = document.documentElement;
      if (elem.requestFullscreen) {
        await elem.requestFullscreen();
      } else if (elem.mozRequestFullScreen) {
        await elem.mozRequestFullScreen();
      } else if (elem.webkitRequestFullscreen) {
        await elem.webkitRequestFullscreen();
      } else if (elem.msRequestFullscreen) {
        await elem.msRequestFullscreen();
      }
      this.isFullScreen = true;
    } catch (error) {
      this.recordEvent({
        type: MonitoringEventType.FULL_SCREEN_EXIT,
        timestamp: Date.now(),
        details: { error: error.message }
      });
    }
  }

  /**
```

```
 * Перевірка позиції погляду в межах viewport
 */
private checkGazePosition(x: number, y: number) {
  const viewport = {
    left: 0,
    top: 0,
    right: window.innerWidth,
    bottom: window.innerHeight
  };

  const tolerance = 50; // пікселів толерантності для межових випадків

  if (!this.isWithinViewport(x, y, viewport, tolerance)) {
    this.recordEvent({
      type: MonitoringEventType.GAZE_AWAY,
      timestamp: Date.now(),
      details: {
        x,
        y,
        viewport: {
          width: window.innerWidth,
          height: window.innerHeight
        }
      }
    });
  }
}

private isWithinViewport(
  x: number,
  y: number,
  viewport: { left: number; top: number; right: number; bottom: number },
  tolerance: number
): boolean {
  return (
    x >= viewport.left - tolerance &&
    x <= viewport.right + tolerance &&
    y >= viewport.top - tolerance &&
    y <= viewport.bottom + tolerance
  );
}

/**
 * Надсилання події на сервер
 */
```

```
  private async recordEvent(event: MonitoringEvent) {
    try {
      const response = await axios.post(`/api/attempts/${this.attemptId}/event`, {
        ...event,
        details: {
          ...event.details,
          browserInfo: {
            userAgent: navigator.userAgent,
            screenResolution: `${window.screen.width}x${window.screen.height}`,
            windowSize: `${window.innerWidth}x${window.innerHeight}`,
            availableScreenSize: `${window.screen.availWidth}x${window.screen.availHeight}`
          }
        }
      });

      // Перевірка на необхідність завершення тесту
      if (response.data.shouldEndTest) {
        await this.endTestAttempt();
      }

    } catch (error) {
      console.error('Failed to record monitoring event:', error);
    }
  }

  /**
   * Примусове завершення тесту
   */
  private async endTestAttempt() {
    try {
      await axios.put(`/api/attempts/${this.attemptId}/complete`, {
        reason: 'Security violation detected',
        violations: this.warningCount
      });

      // Перенаправлення на сторінку результатів
      window.location.href = `/test-results/${this.attemptId}`;
    } catch (error) {
      console.error('Failed to end test attempt:', error);
    }
  }
}
```