МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: АНАЛІТИЧНИЙ ПРОГРАМНИЙ МОДУЛЬ ВЕБ-ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ ТА ПРОВЕДЕННЯ ТЕСТУВАННЯ У НАВЧАЛЬНОМУ ПРОЦЕСІ

Здобувача вищої освіти другого (магістерського) рівня,
ІІ курсу, групи КМІз-24-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»

_____ Скасків Лілії Василівни

Науковий керівник _____ Філоненко Михайло Миколайович, канд. фіз.-мат. наук, доцент

Керівник освітньо-професійної програми _____ Омельчук Антон Анатолійович, канд. техн. наук

Допущено до захисту за рішенням кафедри протокол від «09» січня 2026 № 11

Завідувач кафедри _____ (підпис) Філоненко Михайло Миколайович, канд. фіз.-мат. наук, доцент

Ірпінь – 2026

Реєстрація № 22 — 22.01.2026 (дата) — (підпис відповідальної особи)

Результати перевірки — до захисту (до захисту, на доопрацювання) — 09.01.2026 (дата) — (підпис)

Результат захисту — 5 (відмінно) (оцінка) — 11.02.2026 (дата) — (підпис)

Голова комісії: (підпис) О.С. Дюбанов

Члени комісії:

(підпис) А.Ю. Горбовий

(підпис) М.М. Філоненко

(підпис) І.Д. Погорєловська

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки вбалах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря,
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	27		11.02.2026 Дюбанов О.С. Горбовий А.Ю.
2. Дотримання вимог щодо оформлення	від 0 до 10	7		Філоненко М.М.
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	5		Погорєловська І.Д. Суховецька Л.П.
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	44		

Оцінювання кваліфікаційної роботи науковим керівником - 46 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100- бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національноюшкалою	Дата	П.І.Б.: голови, членів комісії, секретаря,
90	A	5	11.02.2026	Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погорєловська І.Д. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую
Завідувач кафедри _____ Філоненко М.М.
«11» грудня 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Скасків Лілії Василівні

1. **Тема кваліфікаційної роботи**: Аналітичний програмний модуль веб-додатку для організації та проведення тестування у навчальному процесі.
Затверджена наказом ректора від «10» грудня 2024 № 1852.
2. Строк здачі закінченої роботи «02» січня 2026.
3. **Мета роботи:** підвищення об'єктивності та інформативності оцінювання у навчальному процесі шляхом розробки та програмної реалізації аналітичних модулів у вебдодатку, що забезпечують автоматизований розрахунок психометричних характеристик тестів та детекцію аномальної поведінки користувачів у реальному часі.
Об'єкт дослідження: процес автоматизованого оцінювання результатів навчання та діагностики якості тестових матеріалів у веборієнтованих системах дистанційної освіти
Предмет дослідження: методи, математичні моделі, алгоритми та архітектурні патерни реалізації високопродуктивних аналітичних програмних модулів для обробки психометричних даних у веб-середовищі.
Орієнтовна база дослідження: сучасні веб-орієнтовані системи електронного навчання та тестування (LMS Moodle, Google Classroom, Microsoft Forms, Classtime, Testportal); програмні архітектури веб-додатків, що підтримують модульність, масштабованість та інтеграцію аналітичних компонентів.
4. **Вихідні дані до роботи**: результати тестування користувачів, часові характеристики виконання завдань, частотний аналіз правильних/неправильних відповідей, індивідуальні та групові показники успішності.
5. **Перелік графічного матеріалу**: рисунки, таблиці, які розміщені в додатках.
6. **Зміст кваліфікаційної роботи:**
Розділ 1. Теоретичні основи та огляд існуючих рішень для онлайн тестування та аналізу даних у навчальному процесі (поняття та значення онлайн

тестування; методи та підходи до аналізу даних онлайн тестування; огляд існуючих вебдодатків для онлайн тестування).

Розділ 2. Проєктування архітектури аналітичного програмного модуля у вебдодатку (вимоги до аналітичного модуля; функціональні вимоги та бізнес-логіка; нефункціональні вимоги; архітектура програмного комплексу).

Розділ 3. Розробка та реалізація аналітичних програмних модулів (реалізація моделей даних; програмна реалізація збору та трансформації даних; Реалізація алгоритмів; реалізація асинхронної обробки; реалізація інтерфейсу користувача).

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів роботи	
		За планом	Фактично
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2024	До 15.12.2024
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.05.2025	26.05.2025
3	**ІІ розділ** (аналіз експериментальної частини про методики досліджень)	02.09.2025	02.09.2025
4	**ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	24.11.2025	24.11.2025
5	**Висновки**	08.12.2025	08.12.2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	02.01.2025	02.01.2025
7	Отримання відгуку керівника	02.02.2026	02.02.2026
8	Отримання зовнішньої рецензії	05.02.2026	05.02.2026

8. Дата видачі завдання «11» грудня 2024

9. Керівник кваліфікаційної роботи _____ Філоненко М.М.

10. Гарант освітньо-професійної програми _____ Омельчук А.А.

11. Завдання до виконання отримав здобувач вищої освіти _____ Скасків Л.В.

АНОТАЦІЯ

Аналітичний програмний модуль веб-додатку для організації та проведення тестування у навчальному процесі

Сучасна парадигма освіти переживає фундаментальну трансформацію, зумовлену стрімкою цифровізацією всіх сфер суспільного життя. Перехід від традиційних, фізично локалізованих моделей навчання до гнучких, дистанційних та змішаних форм став процесом, який кардинально змінив вимоги до інструментарію педагогічного контролю. У цьому контексті системи онлайн тестування еволюціонували з простих механізмів доставки контенту та фіксації результатів у складні інформаційні екосистеми, що генерують величезні масиви даних. Якщо раніше тестування сприймалося переважно як підсумковий етап контролю знань, то сьогодні воно трансформується у безперервний діагностичний та формувальний процес, який здатний забезпечувати миттєвий зворотний зв'язок та адаптувати освітню траєкторію під індивідуальні потреби здобувача освіти.

Метою роботи є підвищення об'єктивності та інформативності оцінювання у навчальному процесі шляхом розробки та програмної реалізації аналітичних модулів у вебдодатку, що забезпечують автоматизований розрахунок психометричних характеристик тестів та детекцію аномальної поведінки користувачів у реальному часі.У роботі використано методи системного аналізу, теорію управління запасами, статистичний аналіз часових рядів, методи інженерії ознак та ансамблеві методи машинного навчання.

Об’єктом дослідження є процес автоматизованого оцінювання результатів навчання та діагностики якості тестових матеріалів у веборієнтованих системах дистанційної освіти.

Предметом дослідження є методи, математичні моделі, алгоритми та архітектурні патерни реалізації високопродуктивних аналітичних програмних модулів для обробки психометричних даних у веб-середовищі.

Наукова новизна отриманих результатів полягає у вдосконаленні методу інтеграції компонентів наукових обчислень у веб-орієнтовані інформаційні системи освітнього призначення. Запропоновано гібридну архітектуру обробки даних, яка, на відміну від існуючих підходів, використовує асинхронні черги повідомлень та векторизовані операції над матрицями пам'яті для розрахунку психометричних індексів у реальному часі, що дозволяє уникнути блокування інтерфейсу користувача при роботі з великими вибірками даних.

Практичне значення роботи визначається створенням дієздатного програмного продукту — вебдодатка з інтегрованими аналітичними модулями,

який готовий до впровадження у закладах вищої освіти. Розроблена система автоматизує рутинні процеси аналізу якості тестів, надає викладачам інструментарій для покращення навчального контенту та підвищує довіру до результатів онлайн оцінювання завдяки вбудованим механізмам виявлення академічної нечесності.

Кваліфікаційна робота викладена на 60 сторінках, містить вступ, три розділи, висновки, список використаних джерел із 52 найменувань та 17 додатків.

Ключові слова: веб-додаток, аналітичний програмний модуль, комп’ютерне тестування, навчальний процес, візуалізація даних, статистика успішності, архітектура програмного забезпечення, система контролю знань.

Рік виконання кваліфікаційної роботи: 2025.

Рік захисту роботи: 2026

ABSTRACT

Analytical software module of a web application for organizing and conducting testing in the educational process

The modern educational paradigm is undergoing a fundamental transformation driven by the rapid digitalization of all spheres of social life. The transition from traditional, physically localized learning models to flexible, remote, and blended forms has become a process that has radically changed the requirements for pedagogical control tools. In this context, online testing systems have evolved from simple mechanisms for content delivery and results recording into complex information ecosystems that generate massive datasets. While testing was previously perceived primarily as a final stage of knowledge assessment, today it is transforming into a continuous diagnostic and formative process capable of providing instant feedback and adapting the educational trajectory to the individual needs of the learner.

The aim of the research is to increase the objectivity and informativeness of assessment in the educational process by developing and implementing analytical software modules within a web application. These modules provide automated calculation of psychometric test characteristics and real-time detection of anomalous user behavior. The study utilizes methods of systems analysis, inventory management theory, statistical time series analysis, feature engineering techniques, and ensemble machine learning methods.

The object of the study is the process of automated assessment of learning outcomes and quality diagnostics of test materials in web-oriented distance education systems.

The subject of the study includes methods, mathematical models, algorithms, and architectural patterns for implementing high-performance analytical software modules for processing psychometric data in a web environment.

The scientific novelty of the results lies in the improvement of the method for integrating scientific computing components into web-oriented educational information systems. A hybrid data processing architecture is proposed which, unlike existing approaches, utilizes asynchronous message queues and vectorized operations on memory matrices for real-time psychometric index calculation. This approach prevents user interface blocking when working with large datasets.

The practical significance of the work is defined by the creation of a viable software product - a web application with integrated analytical modules ready for implementation in higher education institutions. The developed system automates routine test quality analysis, provides educators with tools to improve learning content,

and increases trust in online assessment results through built-in mechanisms for detecting academic dishonesty.

The master’s thesis consists of 60 pages, including an introduction, three chapters, conclusions, a list of 52 references, and 17 appendices.

Keywords: web application, analytical software module, computer-based testing, educational process, data visualization, academic performance statistics, software architecture, knowledge assessment system.

Year of completion: 2025.

Year of defense: 2026.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 4
ВСТУП 6
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОНЛАЙН ТЕСТУВАННЯ ТА АНАЛІЗУ ДАНИХ У НАВЧАЛЬНОМУ ПРОЦЕСІ 10
1.1. Поняття та значення онлайн тестування в сучасному освітньому просторі 10
1.2. Методи та підходи до аналізу даних онлайн тестування у навчальному процесі 17
1.3. Огляд існуючих вебдодатків для онлайн тестування та їх аналітичних можливостей 24
Висновки до розділу 1 28
РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ АНАЛІТИЧНОГО ПРОГРАМНОГО МОДУЛЯ У ВЕБДОДАТКУ 30
2.1. Вимоги до аналітичного модуля вебдодатка для онлайн тестування 30
2.2. Архітектура програмного комплексу 34
2.3. Проектування структури бази даних 38
2.4. Проектування архітектури аналітичного модуля 41
2.5. Алгоритмічне забезпечення та математичні моделі 43
Висновки до розділу 2 46
РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ АНАЛІТИЧНИХ ПРОГРАМНИХ МОДУЛІВ 47
3.1. Реалізація моделей даних та ORM оптимізація 47
3.2. Програмна реалізація збору та трансформації даних 49
3.3. Реалізація психометричних алгоритмів 50
3.4. Реалізація аналізу дистракторів 52
3.5. Програмна реалізація детекції аномалій 53
3.6. Реалізація асинхронної обробки (Celery & Redis) 54

3.7. Реалізація інтерфейсу користувача 56
3.8. Кастомізація управління користувачами..... 60
Висновки до розділу 3 62
ВИСНОВКИ..... 64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 66
ДОДАТКИ..... ……..72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

АМ - аналітичний модуль.

БД - база даних.

ЗВО - заклади вищої освіти.

ЗО - здобувачі вищої освіти.

МН - машинне навчання.

НП - навчальний процес.

ОП - освітній простір.

ОТ - онлайн тестування.

СОТ - системи онлайн тестування.

ШІ - штучний інтелект.

3NF (Third Normal Form) - третя нормальна форма.

ACID (Atomicity, Consistency, Isolation, Durability) - набір властивостей, що гарантують надійність транзакцій у БД.

AJAX (Asynchronous JavaScript and XML) - технологія фонового обміну даними між браузером та сервером.

API (Application Programming Interface) - програмний інтерфейс додатку.

CRUD (Create, Read, Update, Delete) - основні операції роботи з даними.

CTT (Classical Test Theory) - класична теорія тестів.

DOM (Document Object Model) - об'єктна модель документа HTML.

ETL (Extract, Transform, Load) - процес вилучення, трансформації та завантаження даних.

IRT (Item Response Theory) - теорія відповідей на завдання.

LMS (Learning Management System) - система управління навчанням.

MTV (Model-Template-View) - архітектурний патерн, адаптований фреймворком Django.

MVP (Minimum Viable Product) - мінімально життєздатний продукт.

ORM (Object-Relational Mapping) - технологія зв'язку об'єктів з реляційними базами даних.

SPA (Single Page Application) - односторінковий веб-додаток.

SQL (Structured Query Language) - мова структурованих запитів до БД.

SSR (Server-Side Rendering) - серверна генерація (рендеринг) сторінок.

UX (User Experience) - досвід користувача при взаємодії з інтерфейсом.

ВСТУП

Сучасна парадигма освіти переживає фундаментальну трансформацію, зумовлену стрімкою цифровізацією всіх сфер суспільного життя. Перехід від традиційних, фізично локалізованих моделей навчання до гнучких, дистанційних та змішаних форм став процесом, який кардинально змінив вимоги до інструментарію педагогічного контролю. У цьому контексті системи онлайн тестування еволюціонували з простих механізмів доставки контенту та фіксації результатів у складні інформаційні екосистеми, що генерують величезні масиви даних. Якщо раніше тестування сприймалося переважно як підсумковий етап контролю знань, то сьогодні воно трансформується у безперервний діагностичний та формувальний процес, який здатний забезпечувати миттєвий зворотний зв'язок та адаптувати освітню траєкторію під індивідуальні потреби здобувача освіти.

Актуальність теми зумовлена розривом між потенційними можливостями цифрових даних та реальним функціоналом існуючих платформ. Популярні веб-рішення, хоча й вирішують задачу масового охоплення та автоматичної перевірки, часто обмежуються наданням поверхневої описової статистики. Вони констатують факт успішності, але не пояснюють глибинних причин отриманих результатів та не дозволяють оцінити якість самого вимірювального інструментарію. Водночас, сучасна педагогічна наука та психометрія володіють потужним математичним апаратом який дозволяє з високою точністю визначати надійність тестів, їхню валідність, дискримінативну здатність та виявляти аномалії у поведінці респондентів.

Проблема, яку вирішує дане дослідження, лежить у площині інженерії програмного забезпечення та полягає у складності інтеграції ресурсомістких наукових обчислень у веб-архітектуру, орієнтовану на високу чутливість інтерфейсу. Виконання складних статистичних розрахунків, таких як обчислення коефіцієнта альфа Кронбаха, точково-бісеріальної кореляції або

аналіз дистракторів на великих вибірках, вимагає значних обчислювальних ресурсів. Традиційні архітектурні підходи, що базуються на синхронній обробці HTTP-запитів та використанні скриптових мов загального призначення, часто виявляються неспроможними забезпечити необхідну продуктивність при роботі з матрицями великої розмірності, що призводить до блокування інтерфейсу та деградації досвіду користувача.

Об’єктом дослідження є процес автоматизованого оцінювання результатів навчання та діагностики якості тестових матеріалів у веборієнтованих системах дистанційної освіти.

Предметом дослідження є методи, математичні моделі, алгоритми та архітектурні патерни реалізації високопродуктивних аналітичних програмних модулів для обробки психометричних даних у веб-середовищі.

Метою роботи є підвищення об'єктивності та інформативності оцінювання у навчальному процесі шляхом розробки та програмної реалізації аналітичних модулів у вебдодатку, що забезпечують автоматизований розрахунок психометричних характеристик тестів та детекцію аномальної поведінки користувачів у реальному часі.

Для досягнення поставленої мети сформульовано та вирішено наступні завдання:

1. Провести ґрунтовний аналіз предметної області, існуючих програмних рішень та теоретичних основ психометрики для визначення функціональних розривів та формування вимог до проектованої системи. Необхідно виявити обмеження існуючих платформ щодо глибини аналізу та можливостей кастомізації.

2. Спроектувати архітектуру вебдодатка, що базується на принципах модульності та асинхронної обробки даних. Обґрунтувати вибір архітектурного патерну та технологічного стеку, який дозволяє ефективно поєднувати логіку веб-сервера з потужністю наукових обчислень.

3. Адаптувати математичні моделі для розрахунку ключових

психометричних індексів: надійності тесту, складності завдань, дискримінації та аналізу ефективності дистракторів.

4. Розробити алгоритмічне забезпечення для виявлення академічної недоброчесності, включаючи методи детекції «швидкого вгадування» та аналізу схожості векторів відповідей для ідентифікації списування.

5. Виконати програмну реалізацію аналітичних модулів з використанням мови програмування Python, веб-фреймворку Django, бібліотеки векторизованих обчислень NumPy та системи управління чергами завдань Celery/Redis.

6. Реалізувати захищену рольову модель доступу до аналітичних даних.

Методи дослідження базуються на використанні теорії реляційних баз даних для проектування схеми зберігання даних, об'єктно-орієнтованого проектування для розробки програмної архітектури, методів математичної статистики та теорії ймовірностей для реалізації алгоритмів психометричного аналізу, а також методів емпіричного тестування для перевірки коректності роботи програмних модулів.

Наукова новизна отриманих результатів полягає у вдосконаленні методу інтеграції компонентів наукових обчислень у веб-орієнтовані інформаційні системи освітнього призначення. Запропоновано гібридну архітектуру обробки даних, яка, на відміну від існуючих підходів, використовує асинхронні черги повідомлень та векторизовані операції над матрицями пам'яті для розрахунку психометричних індексів у реальному часі, що дозволяє уникнути блокування інтерфейсу користувача при роботі з великими вибірками даних.

Практичне значення роботи визначається створенням дієздатного програмного продукту — вебдодатка з інтегрованими аналітичними модулями, який готовий до впровадження у закладах вищої освіти. Розроблена система автоматизує рутинні процеси аналізу якості тестів, надає викладачам інструментарій для покращення навчального контенту та підвищує довіру до

результатів онлайн оцінювання завдяки вбудованим механізмам виявлення академічної нечесності.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОНЛАЙН ТЕСТУВАННЯ ТА АНАЛІЗУ ДАНИХ У НАВЧАЛЬНОМУ ПРОЦЕСІ

1.1. Поняття та значення онлайн тестування в сучасному освітньому просторі

СОТ (система онлайн тестування) є фундаментальним інтерактивним інструментом для контролю та оцінки знань у сучасному ОП (освітній процес) і набула особливого значення в умовах дистанційного навчання, виступаючи ефективним засобом аналізу успішності та підвищення продуктивності ОП [1]. Цей підхід полягає у використанні веборієнтованих ресурсів для забезпечення оперативного зворотного зв'язку між ЗО (здобувач освіти) та навчальним матеріалом, що досягається завдяки автоматизованій обробці результатів тестування [2].

Цей метод оцінювання спричиняє фундаментальну зміну в педагогічній парадигмі. Моментальний зворотний зв'язок, який надається онлайн-системами, перетворює оцінювання з кінцевого, підсумкового етапу на безперервний, формувальний процес. Це сприяє адаптивному навчанню, оскільки ЗО можуть одразу бачити свої помилки та працювати над їх виправленням. Така оперативність дозволяє інтегрувати оцінювання безпосередньо в НП (навчальний процес), роблячи його інструментом для покращення засвоєння матеріалу.

Впровадження СОТ в ОП супроводжується низкою переваг, що сприяють підвищенню ефективності та доступності навчання. Однією з них є висока швидкість та універсальність. СОТ дозволяють за короткий час перевірити знання з великого обсягу вивченого матеріалу. Гнучкість налаштувань завдань, включаючи хронологічний порядок, рівень складності, а також можливість

вибору одного або кількох правильних варіантів відповідей, робить їх адаптивними до різноманітних педагогічних потреб.

Тестова форма перевірки мінімізує вплив людського фактора. Завдяки розробленій шкалі оцінювання та чітким критеріям відповідей, СОТ забезпечує прозорий та точний результат для всіх учасників ОП. Ця стандартизація сприяє справедливості оцінювання.

Автоматизація процесу перевірки є суттєвим фактором, що знижує навантаження на викладачів. Система може перевіряти тести, що є надзвичайно корисним при роботі з великими групами ЗО та в умовах дистанційного навчання.

Усі ЗВО (здобувачі вищої освіти) перебувають у рівних умовах, що дозволяє навіть слабкому ЗВО добре підготуватися та отримати гідний бал. Можливість передбачити результат та підготуватися заздалегідь робить СОТ менш стресовим процесом порівняно з усними іспитами та опитуванями. ЗВО отримують негайні результати та можливість працювати над помилками, що сприяє оперативній корекції знань та підвищує ефективність навчання.

Гнучкість та доступність СОТ дозволяють ЗВО проходити оцінювання з будь-якого місця та в будь-який час, що є особливо важливим для тих, хто має щільний графік або проживає у віддалених районах. Сучасні технології дозволяють адаптувати тести для ЗВО з особливими освітніми потребами, пропонуючи аудіо/візуальні формати або налаштування шрифтів, забезпечуючи рівні умови для всіх [3].

Ефективність та точність зворотного зв'язку сприяють глибшому навчанню, дозволяючи цілеспрямовано працювати над сильними та слабкими сторонами, що відображається у покращенні результатів наступних оцінювань.[4]

Зменшення адміністративного навантаження є наслідком автоматизації, що призводить до економії часу та коштів, пов'язаних з друком, логістикою та ручною перевіркою паперових тестів. Це робить СОТ більш економічним та ефективним. Крім того, онлайн-іспити можуть бути більш безпечними,

зменшуючи шанси на витік інформації, оскільки відсутні фізичні папери, які можуть загубитися [4].

Сукупність переваг СОТ є каталізатором для створення більш персоналізованих та інклюзивних освітніх середовищ. Ефективність, що вивільняє ресурси викладачів від рутинної перевірки, дозволяє їм зосередитися на індивідуальних потребах ЗВО.

СОТ має певні недоліки. Одним з них є потенційний розвиток пам'яті, а не мислення. Формати завдань з варіантами відповідей можуть спонукати ЗВО до механічного запам'ятовування інформації або відгадування правильної відповіді шляхом виключення неможливих варіантів.

Розробка якісних тестів, особливо з правдоподібними дистракторами, які ефективно перевіряють знання та не дозволяють здогадатися, потребує значних зусиль та часу.

Комп'ютерне тестування не завжди може надати повну картину розуміння предмета, особливо для гуманітарних дисциплін. Воно часто обмежується оцінкою простих навчальних результатів на рівні знань, заснованих на асоціаціях.

Вплив навичок читання також може спотворювати результати. Рівень сформованості навичок читання ЗВО може впливати на результати оцінювання, а не лише на його знання предмета [5]. Завдання часто орієнтовані на переважно базовий рівень знань.

Відсутність безпосереднього контакту з ЗВО під час тестування збільшує ймовірність впливу випадкових факторів на результат оцінювання, оскільки викладач не може спостерігати за невербальними сигналами або уточнювати відповіді.

СОТ вимагає від ЗВО доступу до технологій (комп'ютерів/планшетів) та стабільного Інтернет-з'єднання. Це може бути проблемою для осіб з низькою комп'ютерною грамотністю або для тих, хто проживає у сільській місцевості чи регіонах з обмеженим доступом до мережі.

Не всі типи завдань, наприклад, ті, що вимагають розгорнутих відповідей або інтерпретації, піддаються автоматичній перевірці, що вимагає ручного втручання викладача.

Схильність до шахрайства є суттєвою проблемою. Попри використання технологій прокторингу, повністю усунути можливість шахрайства в онлайн-тестуванні складно. Різний час проведення іспитів може призвести до обміну відповідями між ЗВО, які вже пройшли тест, і тими, хто ще ні.

Недоліки СОТ виявляють фундаментальне протиріччя: прагнення до ефективності та автоматизації може призвести до поверхневого оцінювання та ігнорування потреб ЗВО з різними стилями навчання або обмеженим доступом до технологій. Це протиріччя між обіцянками «об'єктивності» та «автоматизації» та реальністю "розвитку пам'яті, а не мислення" або "обмеженої глибини розуміння" вказує на те, що поточні реалізації СОТ не завжди відповідають педагогічним цілям. Технологічні виклики та питання доступності створюють бар'єри для інклюзивності, незважаючи на заявлену гнучкість. Це означає, що розробка аналітичних модулів повинна не тільки використовувати переваги, але й активно працювати над подоланням цих недоліків, наприклад, через виявлення поведінкових аномалій або інтеграцію якісних методів оцінювання, а також розробку адаптивних інтерфейсів та складніших типів завдань, які оцінюють вищі порядки мислення.

Системи СОТ можуть бути класифіковані за різними критеріями, що відображає їхню багатогранність та адаптивність до різних освітніх потреб.

За педагогічною функцією [6]:

- діагностичні оцінювання - проводяться перед початком навчання для визначення існуючих знань та навичок ЗВО, виявлення сильних і слабких сторін;
- формувальні оцінювання - є поточними та використовуються протягом НП для моніторингу прогресу ЗВО та надання негайного зворотного зв'язку;

- підсумкові оцінювання - проводяться наприкінці навчального періоду для оцінки загального рівня засвоєння матеріалу;
- автентичні оцінювання - вимагають від ЗВО застосування знань та навичок до реальних завдань та проблем, що відображають виклики, з якими вони можуть зіткнутися поза аудиторією;
- бенчмаркінгові оцінювання - є періодичними та використовуються для вимірювання прогресу ЗВО у досягненні конкретних стандартів або навчальних цілей;
- оцінка продуктивності - вимагає від ЗВО демонстрації своїх знань та навичок через виконання комплексних завдань.

За режимом оцінювання, СОТ можуть підтримувати:

- самооцінювання, де ЗВО оцінює власні досягнення; [7]
- взаємооцінювання, коли ЗВО оцінюють роботи один одного;
- оцінювання викладачем, що є традиційним підходом; [8]
- автоматизоване оцінювання, яке виконується програмним забезпеченням. [9]

За технічною реалізацією, системи СОТ поділяються на [10]:

- комп'ютерні платформи, які вимагають встановлення нативного програмного забезпечення на пристрій користувача;
- веборієнтовані платформи, доступні через веббраузер, що усуває потребу в інсталяції додаткового програмного забезпечення та підвищує безпеку даних;
- хмарні платформи, що автоматично масштабують інфраструктуру для розміщення змінної кількості користувачів, забезпечуючи плавну роботу навіть при значній кількості ЗВО;
- мобільні платформи, оптимізовані для використання на мобільних пристроях, які пропонують зручність, але можуть мати менший рівень безпеки;

- платформи з прокторингом, які дозволяють адміністраторам спостерігати та контролювати ЗВО під час іспитів, забезпечуючи академічну доброчесність та запобігаючи шахрайству. [4]

За технологіями програмування, СОТ можуть бути класифіковані на основі:

- систем, що використовують Open Source скриптові мови;
- систем на базі CGI-скриптів;
- систем, реалізованих на технологіях Microsoft, включаючи.NET та ASP;
- систем, що використовують технології SUN;
- систем, що базуються на локальних додатках, які взаємодіють між собою;
- комбінованих рішень, що об'єднують різні технології. [11]

Різноманітність класифікацій підкреслює, що СОТ є не статичною технологією, а динамічною сферою, де технічний розвиток безпосередньо реагує на потреби. Перехід від комп'ютерних до веб- та хмарних платформ є прямою відповіддю на потреби у гнучкості та доступності. [6] Впровадження прокторингу є технологічною відповіддю на проблему шахрайства. [4] Ця взаємозалежність демонструє постійний цикл взаємодії між освітніми цілями та інженерними рішеннями, що відображає безперервне вдосконалення технічних рішень для кращого задоволення складних та різноманітних освітніх потреб.

Таблиця 1.1

Порівняльна таблиця переваг та недоліків онлайн тестування

Аспект	Переваги	Недоліки
Швидкість та універсальність	Швидка перевірка великого обсягу матеріалу, гнучке налаштування завдань	-
Об'єктивність та точність	Усунення людського фактора, прозорість оцінювання завдяки чітким критеріям та шкалам	-

Автоматизація	Автоматична перевірка тестів, зниження навантаження на викладачів	Обмеження автоматичного оцінювання для розгорнутих відповідей
Справедливість та передбачуваність	Рівні умови для всіх ЗВО, зменшення стресу	-
Миттєвий зворотний зв'язок	Негайні результати та можливість роботи над помилками	-
Гнучкість та доступність	Проходження тестів у будь-який час та з будь-якого місця, адаптація для осіб з особливими освітніми потребами	Доступ до технологій, стабільний Інтернет
Сприяння глибшому навчанню	Ефективність та точність зворотного зв'язку дозволяють зосередитися на сильних/слабких сторонах	Розвиток пам'яті, а не мислення; обмежена глибина розуміння
Зменшення адміністративного навантаження	Економія часу та коштів на друк, логістику та контроль	-
Безпека	Зменшення ризиків витоку інформації порівняно з паперовими тестами	Схильність до шахрайства
Якість завдань	-	Часові витрати на складання якісних завдань з правдоподібними дистракторами
Вплив зовнішніх факторів	-	Вплив навичок читання на результати оцінювання; відсутність безпосереднього контакту, вплив випадкових факторів
Рівень складності	-	Переважно базовий рівень завдань, заснований на асоціаціях

Таблиця 1.1 демонструє переваги та компроміси СОТ. Вона також вказує на сфери, де майбутні дослідження або розробки, такі як запропоновані аналітичні модулі, можуть бути зосереджені для пом'якшення виявлених недоліків.

1.2. Методи та підходи до аналізу даних онлайн тестування у навчальному процесі

Аналіз даних СОТ виходить за рамки простого підрахунку правильних відповідей, охоплюючи широкий спектр метрик та показників, що дозволяють отримати глибоке розуміння НП.

Цифрові навчальні середовища дозволяють відстежувати частоту та тривалість входів ЗВО на навчальні платформи, виявляти закономірності їхньої залученості протягом різних періодів часу та інших інтерактивних елементах. [12]

Результативність оцінювання включає детальний аналіз елементів тесту. Це дозволяє ідентифікувати конкретні питання або концепції, які викликають труднощі в окремих ЗВО або цілих груп. Відстеження динаміки успішності дає уявлення про ефективність навчальних втручань.

Аналітика залученості вимірює, як ЗВО взаємодіють з навчальним контентом, однолітками та викладачами. Ключові індикатори залученості охоплюють час, проведений за навчальними завданнями, патерни взаємодії з цифровими навчальними матеріалами, а також метрики соціальних навчальних мереж та співпраці. Дослідження показують, що рівень залученості є значним предиктором успішності ЗВО.

Показники адаптивного навчання фіксують, як ЗВО взаємодіють з персоналізованими освітніми траєкторіями. Ці метрики відстежують вибір та прогрес навчальних шляхів, реакції на зміни складності контенту, частоту та ефективність корекційних заходів, а також темп просування матеріалом. Вони є важливими для оптимізації індивідуальних навчальних досвідів.

Аналіз поведінкових тенденцій вивчає закономірності в діях та взаємодіях ЗВО, щоб ідентифікувати фактори, що впливають на результати навчання.

Перелічені метрики свідчать про еволюцію аналітики від простого вимірювання кінцевих оцінок до комплексного моніторингу самого НП. Такий підхід є основою для індивідуалізованого та проактивного педагогічного втручання. Збір та аналіз цих даних дозволяє викладачам не просто оцінювати, а активно впливати на навчальний шлях ЗВО, надаючи своєчасну підтримку та адаптуючи методику викладання.

Таблиця 1.2.

Основні метрики та показники для аналізу даних онлайн тестування

Категорія метрики/показника	Опис/Приклади
Рівень відвідуваності та участі	Частота та тривалість входів на навчальні платформи; патерни залученості протягом різних періодів; участь в обговореннях, спільних проєктах та інтерактивних елементах.
Результативність оцінювання	Аналіз елементів тесту; відстеження динаміки успішності; відображення опанування навичок.
Аналітика залученості	Час, проведений за навчальними завданнями; частота та якість внесків у класні дискусії; патерни взаємодії з цифровими навчальними матеріалами; показники соціальних навчальних мереж та співпраці.
Показники адаптивного навчання	Вибір та прогрес навчальних шляхів; реакції на зміни складності контенту; частота та ефективність корекційних заходів; темп просування матеріалом.
Аналіз поведінкових тенденцій	Вивчення патернів дій та взаємодій ЗВО для виявлення факторів, що впливають на результати навчання (звички до навчання, звернення за допомогою, індикатори ризику академічних труднощів або відсіву, наполегливість та мотивація).

Таблиця 1.2 надає структурований огляд різноманітних даних, що збираються в онлайн-тестуванні, які виходять за рамки простих оцінок. Вона слугує фундаментальним довідником щодо того, які дані можуть бути проаналізовані, безпосередньо підтримуючи аспект "аналітичних програмних модулів". Вона допомагає класифікувати сирий матеріал для аналізу, демонструючи систематичне розуміння навчальної аналітики.

Статистичний аналіз є наріжним каменем для перетворення сирих даних СОТ на значущі та дієві висновки. Він охоплює широкий спектр методів, від простих узагальнень до складних прогностичних моделей.

Описова статистика формує основу аналізу освітніх даних, надаючи основні зведення про успішність ЗВО, демографічну інформацію та розподіл ресурсів. Ці методи перетворюють сирі дані на значущі відомості за допомогою показників центральної тенденції та розсіювання. [13]

Інференційна статистика дозволяє робити прогнози та узагальнення щодо більших сукупностей на основі вибіркових даних. Цей підхід є особливо цінним для освітніх досліджень, де вивчення цілих популяцій є непрактичним. Інференційні методи включають довірчі інтервали, перевірку гіпотез та рівні значущості. [14]

Багатовимірний аналіз досліджує взаємозв'язки між декількома змінними одночасно, надаючи глибше розуміння складних явищ. Його методи допомагають зрозуміти взаємодію різних факторів, що впливають на успішність ЗВО.

Також існують спеціалізовані методи:

- перевірка статистичних гіпотез використовує t-тест, критерій χ^2 та F-тест для оцінки значущості статистичних висновків; [15]
- кореляційний та регресійний аналіз досліджують зв'язки та взаємозалежності між змінними;
- кластеризація групує об'єкти за схожими характеристиками;

Аналіз елементів тесту — це процес вивчення відповідей ЗВО на окремі питання ОТ для оцінки якості цих питань та тесту в цілому. [16] Він допомагає ідентифікувати питання, які можуть бути занадто складними, занадто легкими або неефективними у вимірюванні того, що вони призначені вимірювати. [17] Аналіз включає визначення рівня складності (p-значення), дискримінаційної здатності (наскільки добре питання розрізняє ЗВО з високим та низьким рівнем знань) та ефективності дистракторів (неправильних варіантів відповідей).

Теорія відповідей на завдання є статистичною основою, що використовується для аналізу результатів тестів та покращення їх дизайну. Вона зосереджена на вимірюванні латентних рис, таких як здібності або особистісні характеристики, які не є безпосередньо спостережуваними, але можуть бути виведені з відповідей на тест. [18] IRT постулює, що ЗВО, які демонструють сильнішу присутність певної риси, з більшою ймовірністю правильно відповідають на відповідні тестові завдання. Ця методологія дає більш точні оцінки в широкому діапазоні здібностей, що робить її особливо корисною як для високоефективних, так і для низькоефективних ЗВО. Одним з ключових застосувань IRT є комп'ютерне адаптивне тестування, де складність тестових питань коригується в реальному часі на основі попередніх відповідей ЗВО. Порівняно з класичною теорією тестів, IRT краще оцінює ЗВО, чиї здібності є надзвичайно високими або надзвичайно низькими.

Психометричний аналіз в ОТ включає вимірювання таких психологічних метрик, як здібності, особистісні риси та ситуаційні судження. [19] Ці оцінювання мають відомі рівні валідності та надійності, пропонуючи об'єктивну оцінку кандидатів та їхніх рис і здібностей.

Еволюція статистичних методів від описової до інференційної та багатовимірної статистики, а також спеціалізованих психометричних підходів, дозволяє освітнім установам перейти від простого констатування фактів до розуміння причинно-наслідкових зв'язків та прогнозування майбутніх результатів. Це є ключовим для розробки проактивних АМ (аналітичний модуль), які можуть виявляти проблеми до їх загострення. IRT, зокрема, дозволяє адаптувати складність питань у реальному часі, що є прямою передумовою для адаптивного навчання. Ця прогресія від простого підрахунку до складного моделювання та прогнозування є фундаментальною для розробки інтелектуальних АМ.

Візуалізація даних є невід'ємною частиною аналізу результатів ОТ, оскільки вона перетворює складні числові дані на зрозумілі та дієві графічні

представлення. Основне призначення візуалізації полягає у покращенні розуміння складних даних, сприянні прийняттю рішень на основі даних та покращенні комунікації між усіма зацікавленими сторонами ОП. [20]

Для створення візуалізацій існує широкий спектр інструментів, від простих до професійних [21]:

- Canva та Piktochart є популярними інструментами для створення інфографіки завдяки своїй простоті у використанні та багатому набору шаблонів;
- Tableau та Infogram є потужними інструментами для візуалізації даних, що дозволяють створювати інтерактивні графіки та дашборди, особливо корисні для роботи з великими обсягами даних;
- Adobe Illustrator є професійним інструментом для створення високоякісних візуальних матеріалів, що вимагає певних навичок, але надає максимальний контроль над візуальним оформленням.

Особливе значення мають інтерактивні візуалізації, які сприяють залученості ЗВО та глибшому дослідженню даних, що призводить до кращого розуміння складних взаємозв'язків. [20]

Еволюція до інтерактивних та спеціалізованих інструментів візуалізації відображає зростаючу потребу в швидкій та ефективній інтерпретації даних для прийняття педагогічних рішень. Це означає, що візуалізація є кінцевим етапом у ланцюжку аналізу, який робить дані корисними для кінцевого користувача.

Застосування МН в аналізі даних СОТ є одним з найбільш перспективних напрямків розвитку освітньої аналітики. Воно дозволяє перетворити збір даних з ретроспективного звітування на проактивні, персоналізовані та адаптивні втручання.

Одним з ключових напрямків є прогнозна аналітика. Алгоритми МН (машинне навчання), такі як RandomForest, XGBoost, Logistic Regression та Decision Tree, використовуються для прогнозування майбутніх результатів навчання, виявлення ЗВО у групі ризику щодо низької успішності або відсіву, а

також для прогнозування загальної успішності. [22] Це дозволяє викладачам вживати своєчасних коригувальних заходів.

МН є рушійною силою для створення персоналізованих навчальних екосистем. Ці системи аналізують індивідуальні дані ЗВО для створення адаптованих навчальних шляхів, коригуючи контент, темп та методи викладання відповідно до потреб кожного ЗВО. [23] Це включає адаптивний зворотний зв'язок у реальному часі, предиктивне моделювання, безперервне відстеження взаємодій ЗВО та автоматизовану генерацію та курацію контенту.

Інтелектуальні навчальні системи є наступним етапом розвитку освітніх технологій. Вони надають індивідуальний супровід, використовуючи обробку природної мови, машинне навчання та когнітивне моделювання для взаємодії зі ЗВО. Ці системи не просто представляють матеріал, а ставлять запитання, коригують свій підхід до викладання на основі відповідей ЗВО. [24]

СОТ та зворотний зв'язок на основі ШІ трансформують процес оцінювання. Це включає автоматичну оцінку есе, аналіз підходів до вирішення проблем у реальному часі, виявлення плагіату та надання зворотного зв'язку. Системи можуть навіть ідентифікувати хибні уявлення ЗВО на основі патернів помилок. Генеративний ШІ використовується для створення різноманітних оцінювань, формулювання рубрик та для скраффолдингу (поетапної підтримки) навчання. [25]

Відстеження знань є методом оцінки стану знань ЗВО на основі його історичних записів вирішення завдань, прогнозуючи правильність наступної відповіді. [26] Розвиток цього напрямку включає Deep Knowledge Tracing та Option Tracing для питань з множинним вибором, що дозволяє детальніше аналізувати вибір варіантів відповідей. Модель Uncertainty-Aware Knowledge Tracing додатково враховує невизначеність у взаємодіях ЗВО, що підвищує точність прогнозування. [27]

Приховане оцінювання є інноваційним підходом, що передбачає ненав'язливе, безперервне формувальне оцінювання, вбудоване безпосередньо в

цифрові навчальні середовища. Система захоплює дані журналу активності ЗВО в реальному часі та використовує статистичні моделі, такі як байєсівські мережі або IRT, для оцінки рівнів майстерності та надання адаптивної підтримки без переривання навчального процесу. [28]

Виявлення шахрайства є важливим застосуванням МН в СОТ. Алгоритми глибокого навчання та МН використовуються для ідентифікації неетичних поведінкових патернів, що допомагає забезпечити академічну доброчесність. [29]

Важливою складовою впровадження МН є врахування етичних аспектів. Це включає проблеми конфіденційності даних, непрозорості алгоритмів, упередженості наборів даних, на яких навчаються моделі, та ризиків неточних прогнозів. [23]

Застосування МН перетворює аналітику навчання з ретроспективного звітування на проактивні, персоналізовані та адаптивні втручання. Це дозволяє не тільки глибше розуміти індивідуальний навчальний процес, а й автоматизувати складні завдання оцінювання та забезпечити надійні механізми академічної доброчесності. Предиктивна аналітика дозволяє ідентифікувати ЗВО у групі ризику [22], що є підставою для впровадження персоналізованого навчання та інтелектуальних навчальних систем. [24] Це створює прямий причинно-наслідковий ланцюжок: прогнозування веде до персоналізації, а та, своєю чергою, до адаптивного навчання. Методи, такі як Knowledge Tracing та Stealth Assessment дозволяють безперервно оцінювати та підтримувати навчання. Водночас ці технології не можуть бути впроваджені без уваги до соціальних та моральних наслідків, що є важливим аспектом для розробки АМ.

1.3. Огляд існуючих вебдодатків для онлайн тестування та їх аналітичних можливостей

Існуючий ринок вебдодатків для СОТ пропонує широкий спектр рішень, кожне з яких має свої особливості функціоналу та аналітичних можливостей.

Moodle є однією з найпоширеніших систем управління навчанням з відкритим вихідним кодом. Вона автоматично записує активність ЗВО у файли журналів, що є основою для подальшого аналізу. Moodle має вбудовані плагіни, які забезпечують швидкий та точний аналіз статистики навчання. [30] Платформа дозволяє проводити кореляційний аналіз між активністю ЗВО та їхніми оцінками. Також Moodle аналізує метрики тестів, такі як відповіді, оцінки та час, витрачений на питання, для дослідження стратегій навігації ЗВО під час тестування. [31] Щодо аналітичних можливостей, Moodle надає базову аналітику через вбудовані функції та плагіни, такі як 'Learning Analytics Moodle Plugin'. Однак для більш глибокого та розширеного аналізу часто потрібні сторонні інтеграції та відповідні технічні навички для їх налаштування та використання.

Google Forms є зручним та інтуїтивно зрозумілим інструментом для створення ОТ. Він підтримує різні типи питань, має інтерфейс drag-and-drop та дозволяє налаштовувати теми та логіку для відображення релевантних питань на основі попередніх відповідей. [32] Аналітичні можливості Google Forms включають автоматичну візуалізацію відповідей у вигляді діаграм у реальному часі. Для більш глибокого аналізу сирі дані можна експортувати безпосередньо до Google Sheets, що дозволяє подальші розрахунки. Також підтримується спільний аналіз даних, що дозволяє кільком користувачам працювати над інтерпретацією результатів.

Kahoot! – навчальна платформа, що спеціалізується на гейміфікованих тестах. Вона активно використовує елементи гейміфікації, такі як бали, значки та таблиці лідерів, для підвищення залученості та мотивації ЗВО. [33] Платформа пропонує банки питань та зображень, а також підтримує режими реального часу

та домашньої роботи. Kahoot! відстежує індивідуальний прогрес ЗВО та дозволяє експортувати дані до Excel для подальшого аналізу. Аналітичні можливості включають побудову діаграм успішності для груп та відстеження індивідуальних відповідей ЗВО, а також надання негайного зворотного зв'язку аудиторії. Дослідження показують позитивний вплив Kahoot! на академічну успішність, мотивацію та зменшення тривожності. [34]

Blackboard Learn є комплексною системою управління навчанням, яка надає викладачам широкі можливості для моніторингу успішності та залученості ЗВО. Платформа дозволяє відстежувати детальні журнали активності ЗВО, включаючи IP-адреси для подань. [35] Аналітичні можливості системи включають звіти курсу, які аналізують відповідність контенту інституційним цілям та оцінюють прогрес ЗВО. Деталі активності ЗВО для оцінювань дозволяють бачити, коли ЗВО відкрили, почали та подали тести та завдання. Звіт про активність курсу допомагає зрозуміти продуктивність та взаємодію ЗВО, включаючи зміни загальної оцінки. Функція "Analytics for Learn" комбінує дані з Blackboard Learn з даними системи SIS установи, надаючи інтегровані звіти, зведення подань та матрицю активності. Дані можуть бути експортовані у форматі CSV для подальшого аналізу. Це дозволяє виявляти ЗВО у групі ризику, тих, хто покращує результати, або тих, чиї оцінки знижуються.

Порівняльний аналіз існуючих платформ для СОТ виявляє їхні сильні сторони та обмеження, особливо щодо глибини аналітичних можливостей.

Moodle, як відкрита платформа, пропонує високий ступінь кастомізації та гнучкості. Кожен аспект платформи, від макета до функціоналу, може бути адаптований до унікальних вимог організації. [36] Однак, її інтерфейс може бути складним та перевантаженим. Щодо аналітики, Moodle має вбудовані базові функції, але для розширеного аналізу та детального звітування часто потрібні сторонні плагіни.

Google Forms відзначається своєю простотою використання та інтуїтивно зрозумілим інтерфейсом. Вона чудово інтегрується з іншими сервісами Google

Workspace, дозволяючи спільну роботу та аналіз відповідей у реальному часі. Це робить її доступною та економічно ефективною. Проте, Google Forms має обмежені можливості кастомізації функціоналу та дизайну. Для складнішої логіки опитувань або розширеного аналізу даних може знадобитися експорт до сторонніх інструментів.

Kahoot! вирізняється гейміфікацією та інтерактивністю, що значно підвищує залученість та мотивацію ЗВО. [34] Вона дозволяє відстежувати прогрес групи та окремих ЗВО. Проте, її основний формат схильний до простих, фактологічних питань і менш ефективний для оцінки навичок вищого порядку мислення (критичне мислення, аналіз). [37] Надмірний акцент на конкуренції може викликати тривогу у ЗВО.

Blackboard Learn позиціонується як корпоративне рішення з акцентом на стабільність та вбудовані інтеграції. [36] Вона пропонує сучасний та професійний інтерфейс, має розширені вбудовані інструменти аналітики, що надають звіти в реальному часі про залученість та успішність. Це дозволяє отримувати відомості для коригування стратегій. Однак, Blackboard менш гнучкий у кастомізації та має обмеження щодо обробки великих файлів та детального відстеження активності ЗВО. [38]

Існуючі вебдодатки для ОТ, попри їхні численні переваги, мають низку обмежень, які впливають на глибину аналізу результатів та загальну ефективність ОП.

Загальні обмеження включають:

1. Проблеми з продуктивністю та масштабованістю. Деякі платформи можуть зіткнутися з проблемами продуктивності під час пікового навантаження.
2. Обмежена підтримка медіа та контенту. Це може знизити якість навчальних матеріалів та ускладнити їх використання. [39]
3. Складність інтеграції зі сторонніми системами. Хоча багато платформ підтримують інтеграцію, деякі зовнішні інструменти та системи

можуть бути несумісними або вимагати додаткових плагінів та кастомізації, що ускладнює розширення функціоналу.

4. Проблеми з інтерфейсом користувача. Деякі платформи можуть мати перевантажений інтерфейс з надмірною кількістю меню та налаштувань, що ускладнює навігацію.

5. Вимоги до технологій та доступність. Залежність від пристроїв та стабільного Інтернет-з'єднання є загальним обмеженням, що створює проблеми доступності для ЗВО з обмеженими можливостями або у віддалених регіонах. [37]

Найбільш критичні обмеження існуючих вебдодатків стосуються глибини аналізу результатів тестування:

1. Переважно описовий характер аналітики.

2. Відсутність прогностичних та прескриптивних моделей (лише описова аналітика). Ті, що мають прогностичні або проактивні функції, можуть ґрунтуватися на певних припущеннях про навчання, які не застосовуються до всіх ЗВО.

3. Обмеження збору та аналізу даних. Google Forms, наприклад, має обмеження, для поглибленого аналізу даних часто потрібен експорт до сторонніх інструментів або електронних таблиць. [40] Google Analytics, що використовується для глибшої вебаналітики, також має обмеження, такі як вибірка даних, неповний огляд шляху користувача та суворі обмеження збору даних, що може спотворювати точність та обмежувати детальний аналіз. [41]

4. Фокус на зовнішній мотивації та поверхневому засвоєнні. Платформи, такі як Kahoot!, через свій гейміфікований формат можуть надмірно акцентувати увагу на конкуренції, що зсуває фокус ЗВО.

5. Недостатня деталізація відстеження активності. Blackboard Learn може бути обмеженою у відстеженні детальної активності ЗВО, надаючи лише інформацію про останній вхід без повного відстеження всіх дій. [38] Це обмежує можливість побудови всебічної картини поведінки ЗВО.

6. Проблеми з валідністю та надійністю даних. Збір даних через опитування може страждати від проблем валідності, таких як соціальна бажаність, нестабільний настрій або упередження відповідей. Крім того, значна частина навчання може відбуватися офлайн, що не фіксується онлайн-системами, тому відсутність активності в онлайн-середовищі не завжди означає повну відсутність навчання. [42]

Хоча всі платформи збирають значний обсяг даних, існує розрив між можливостями збору даних та глибиною вбудованої аналітики. Більшість платформ надають лише базові описові звіти, що вимагає експорту даних та використання сторонніх інструментів для проведення комплексного, прогнозуючого або прескриптивного аналізу. Це підкреслює нагальну потребу в розробці спеціалізованих аналітичних модулів, які можуть заповнити цю прогалину, надаючи глибші, більш дієві та прогностичні відомості про навчальний процес.

Висновки до розділу 1

Проведений комплексний аналіз предметної області дозволяє констатувати наступне:

1. Сучасні платформи онлайн-тестування, успішно вирішуючи задачу логістики та доставки контенту, демонструють суттєвий функціональний дефіцит у площині валідизації вимірювань. Відсутність вбудованих механізмів розрахунку психометричних характеристик перетворює оцінювання на формальну процедуру, результати якої часто не відображають реальний рівень компетентності здобувача освіти.

2. Ефективна педагогічна діагностика вимагає переходу від описової статистики до інференційних методів аналізу. Застосування математичного апарату Класичної теорії тестів дозволяє перетворити "сирі" дані тестування на джерело доказової інформації для покращення якості навчального контенту. Це

вимагає інтеграції у веб-додатки спеціалізованих бібліотек для наукових обчислень.

3. Реалізація глибокого аналізу даних у реальному часі наштовхується на обмеження традиційних архітектурних патернів. Виконання ресурсомістких матричних операцій на інтерпретованих мовах загального призначення без використання асинхронних черг призводить до блокування інтерфейсу та деградації досвіду користувача.

4. В умовах дистанційного навчання критичного значення набувають алгоритмічні методи виявлення недоброчесної поведінки. Аналіз темпоральних патернів та структурної схожості відповідей стає необхідним компонентом аналітичної системи, компенсуючи відсутність фізичного прокторингу.

5. Виявлені теоретичні та технологічні розриви детермінують необхідність розробки спеціалізованого веб-додатку, побудованого на принципах модульності, асинхронності та використання високопродуктивних мов для роботи з даними. Система повинна забезпечувати не лише оцінювання студента, а й, перш за все, оцінювання якості самого тесту.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ АНАЛІТИЧНОГО ПРОГРАМНОГО МОДУЛЯ У ВЕБДОДАТКУ

Сучасна парадигма цифрової освіти переживає фундаментальну трансформацію, зміщуючи акценти від простої доставки контенту до глибокого аналізу якості навчального процесу. Системи управління навчанням еволюціонують у складні аналітичні платформи, здатні не лише фіксувати результати, але й діагностувати ефективність педагогічного інструментарію. У цьому контексті проектування підсистеми психометричного аналізу стає критичним етапом інженерії програмного забезпечення, що вимагає синтезу педагогічних теорій вимірювання та передових архітектурних патернів.

Розробка аналітичного модуля вимагає вирішення низки нетривіальних завдань: від забезпечення обробки великих масивів даних у реальному часі до імплементації складних математичних моделей класичної теорії тестів. Цей розділ присвячено комплексному проектуванню архітектури системи, формулюванню вимог, обґрунтуванню технологічного стеку та деталізації алгоритмічного забезпечення, яке ляже в основу програмної реалізації.

2.1. Вимоги до аналітичного модуля вебдодатка для онлайн тестування

Процес інженерії вимог є фундаментом для створення якісного програмного продукту. У контексті аналітичних систем освітнього призначення вимоги виходять за межі стандартних функціональних специфікацій, охоплюючи аспекти психометрії, безпеки даних та ергономіки взаємодії. Аналітичні модулі сучасних систем онлайн тестування повинні забезпечувати комплексний аналіз якості тестових завдань та результатів навчання. На основі аналізу предметної області, існуючих рішень, психометричних підходів та загальних принципів побудови освітніх вебсистем, можна сформулювати

наступні функціональні та нефункціональні вимоги.

Функціональні вимоги визначають поведінку системи та спектр можливостей, що надаються різним акторам системи. Специфіка АМ диктує необхідність чіткої сегрегації функціоналу відповідно до рольової моделі.

1. Забезпечення рольової моделі доступу. Архітектура системи повинна базуватися на жорсткій ієрархічній моделі прав доступу. Це не просто питання адміністративного порядку, а фундаментальна вимога до забезпечення валідності тестування та конфіденційності персональних даних. Проектування повинно передбачати три ключові ролі, кожна з яких має унікальний контекст взаємодії з даними.

Таблиця 2.1

Користувачі та рівень їх доступу

Роль користувача	Рівень доступу	Контекст аналітики
Адміністратор	Глобальний	Має повний доступ до всіх даних системи, налаштувань серверної частини та глобальної статистики використання платформи. Відповідає за технічний моніторинг.
Викладач	Обмежений	Доступ до створення контенту та перегляду аналітики *лише* тих курсів, де користувач є інструктором. Система повинна фільтрувати дані на рівні запитів до БД, щоб унеможливити доступ до результатів студентів чужих груп.
Студент	Ізольований	Доступ виключно до інтерфейсу проходження тесту та перегляду власних результатів. Критичною вимогою є блокування доступу до ключів відповідей та статистики групи до моменту закриття тесту.

2. Гнучка конфігурація параметрів тестування. Якість аналітичних даних безпосередньо залежить від стандартизації умов проведення вимірювання. Система повинна надавати інструментарій для налаштування параметрів, які згодом стануть змінними в аналітичних моделях. Зокрема, критичним є керування часовими обмеженнями. Система повинна дозволяти встановлювати

як глобальний ліміт часу на тест, так і мікро-ліміти на окремі питання. Ця функціональність є передумовою для виявлення патернів "швидкого вгадування" та аналізу поведінки студентів в умовах стресу.

3. Збір гранулярних метрик. Традиційні системи часто обмежуються фіксацією фінального балу. Для побудови глибокої психометричної аналітики цього недостатньо. Проектована система повинна забезпечувати логування метаданих на атомарному рівні взаємодії.

Вимоги до психометричного аналізу:

1. Оцінка надійності тесту – система повинна розраховувати коефіцієнт альфа Кронбаха для визначення внутрішньої узгодженості тестових завдань [43].

2. Аналіз складності завдань – реалізація розрахунку індексу складності кожного тестового питання за методикою класичної теорії тестування [44].

3. Оцінка диференціюючої здатності – обчислення індексів дискримінації для виявлення завдань, які не розрізняють сильних та слабких студентів [45].

4. Аналіз дистракторів – дослідження ефективності неправильних варіантів відповіді через розрахунок показників вибору різними групами студентів [46].

5. Темпоральний аналіз – фіксація та аналіз часових показників виконання завдань для виявлення швидкого вгадування та аномальних патернів [47].

Збереження повного «сліду» студента, включаючи обрані дистрактори. Це є необхідною умовою для проведення дистракторного аналізу та розрахунку коефіцієнтів кореляції.

4. Автоматизована алгоритмічна обробка. Ключовою функціональною вимогою є імплементація математичного апарату класичної теорії тестів. Система повинна автоматично, без втручання користувача, розраховувати матриці відповідей та обчислювати статистичні індекси: складність, дискримінацію, rpb та надійність. Це звільняє викладача від рутинних розрахунків і надає миттєвий зворотний зв'язок про якість тесту. Також має бути експорт даних – можливість вивантаження аналітичних результатів у форматі

CSV для подальшої обробки.

5. Виявлення академічної недоброчесності. В умовах віддаленого навчання система повинна компенсувати відсутність фізичного контролю програмними методами. Вимоги до виявлення академічної нечесності включають:

1. Детекція схожості відповідей – реалізація K-індексу для виявлення підозріло схожих патернів відповідей між студентами [48].

2. Аналіз темпоральних патернів – виявлення аномалій у часі виконання завдань через кореляційний аналіз.

3. Журналювання підозрілої активності – збереження всіх випадків потенційного порушення академічної доброчесності.

Нефункціональні вимоги визначають, *як* система виконує свої функції, і є критичними для забезпечення масштабованості та зручності використання.

1. Продуктивність та асинхронність. Розрахунок психометричних показників на великих вибірках (сотні студентів, десятки питань) вимагає значних обчислювальних ресурсів. Операції включають побудову матриць, їх транспонування та обчислення дисперсій. Виконання цих дій у синхронному режимі (блокуючи HTTP-запит) призведе до неприпустимих затримок інтерфейсу. Архітектура повинна передбачати асинхронну обробку даних через черги повідомлень, забезпечуючи час відгуку інтерфейсу в межах 200 мс, незалежно від навантаження на обчислювальне ядро [49].

2. Інтерактивність та досвід користувача (UX). Сучасні веб-стандарти диктують необхідність створення "безшовного" досвіду. Перезавантаження сторінки при кожній дії (наприклад, переході до наступного питання) порушує когнітивний потік користувача. Система повинна використовувати технології часткового оновлення DOM (Document Object Model) для забезпечення плавності роботи, наближеної до SPA (Single Page Application).

3. Цілісність даних. Структура бази даних повинна відповідати третій нормальній формі (3NF) для уникнення аномалій дублювання. Критичною є підтримка транзакційності: збереження результатів спроби повинно бути

атомарною операцією ("все або нічого"). Також система повинна забезпечувати каскадну цілісність, щоб видалення курсу чи тесту коректно очищало пов'язані аналітичні дані, не залишаючи "сирітських" записів. Система повинна коректно обробляти граничні випадки (малі вибірки, відсутні дані).

4. Масштабованість. Архітектура повинна підтримувати горизонтальне масштабування обчислювальних компонентів. Система повинна ефективно обробляти одночасне завершення тестів кількома студентами.

5. Підтримуваність. Модульна архітектура з чітким розділенням відповідальності компонентів. Документований код з описом алгоритмів обчислення метрик.

2.2. Архітектура програмного комплексу

Вибір архітектурного патерну та технологічного стеку є стратегічним рішенням, що визначає життєздатність системи. Для реалізації АМ обрано архітектуру монолітну архітектуру з асинхронною обробкою на базі фреймворку Django та черги завдань Celery.

Обґрунтування вибору монолітної архітектури:

1. Швидкість розробки – монолітний підхід дозволяє швидше створити MVP (Minimum Viable Product) без додаткової складності міжсервісної комунікації.

2. Простота розгортання – відсутність необхідності в складній оркестрації мікросервісів.

3. Консистентність даних – транзакційна цілісність забезпечується на рівні СУБД без використання розподілених транзакцій.

4. Достатність для середніх навантажень – для навчальних закладів з кількістю користувачів до 5000 монолітна архітектура є оптимальним рішенням.

На відміну від мікросервісної архітектури, яка часто вносить надмірну складність в інфраструктуру для систем середнього масштабу, модульний

моноліт дозволяє поєднати простоту розгортання з чіткою ізоляцією компонентів. Система структурується як набір слабкозв'язаних модулів, які взаємодіють через внутрішні програмні інтерфейси, а не через мережу.

В основі реалізації лежить патерн Model-Template-View (MTV), адаптований фреймворком Django.

Model (шар даних) - відповідає за структуру даних та бізнес-правила. Використання ORM (Object-Relational Mapping) дозволяє абстрагуватися від SQL, працюючи з даними як з об'єктами Python. Це підвищує безпеку (захист від SQL Injection) та швидкість розробки.

Template (шар представлення) - забезпечує генерацію інтерфейсу. Використання серверного рендерингу (SSR) гарантує швидке перше завантаження, що є критичним для користувачів зі слабким інтернет-з'єднанням.

View (шар логіки) - контролер, що обробляє запити, оркеструє взаємодію між моделлю та шаблоном, перевіряє права доступу та ініціює фонові задачі.

Вибір технологічного стеку для аналітичної системи є ключовим фактором, що впливає на можливість реалізації складних алгоритмів. Було проведено детальний порівняльний аналіз трьох підходів: використання традиційного веб-стеку (PHP, наприклад, Moodle), використання спеціалізованого статистичного ПЗ (SPSS/R) та використання універсальної мови з науковими бібліотеками (Python).

1. Порівняння з PHP (екосистема Moodle). Традиційні LMS, такі як Moodle, історично побудовані на PHP. Хоча PHP є чудовим інструментом для веб-розробки, він має суттєві обмеження у сфері наукових обчислень.

PHP не має вбудованих аналогів бібліотеки NumPy. Реалізація матричних операцій (наприклад, розрахунок дисперсій для Альфи Кронбаха) на чистому PHP вимагає написання вкладених циклів, що є обчислювально неефективним ($O(n^2)$) та повільним. Python, завдяки NumPy, виконує ці операції на низькому рівні (C/C++), забезпечуючи приріст продуктивності у десятки разів.

Кодова база PHP-проектів часто страждає від змішування логіки та

представлення ("spaghetti code"), тоді як Python примушує до структурованості та чистоти коду, що полегшує підтримку складних аналітичних алгоритмів.

2. Порівняння зі спеціалізованим ПЗ. Використання зовнішніх статистичних пакетів (SPSS, R) є стандартом у наукових дослідженнях, але є проблематичним для веб-автоматизації.

SPSS є пропрієтарним, дорогим ПЗ із закритим кодом. Його неможливо інтегрувати «всередину» безкоштовного веб-додатку. R є потужним, але його інтеграція з веб-сервером вимагає складних «мостів», що створює точки відмови та затримки на серіалізацію даних.

Python дозволяє розробнику повністю контролювати алгоритм. Наприклад, стандартні пакети SPSS можуть некоректно обробляти пропущені дані (видаляючи весь рядок), тоді як у Python можна легко імплементувати кастомну логіку обробки NaN значень, що є критичним для реальних тестів.

Таблиця 2.2

Підсумкова матриця вибору технологій:

Критерій	Python (Django + NumPy)	PHP (Laravel/Moodle)	R / SPSS
Швидкість веб-розробки	Висока	Висока	Низька (не призначені для Web)
Швидкість обчислень	Дуже висока	Середня/Низька	Висока
Вартість ліцензії	Open Source	Open Source (Free)	SPSS - Висока, R - Free
Масштабованість	Висока	Середня	Низька (Single threaded)
Екосистема ML/AI	Лідер ринку	Обмежена	R - сильна, SPSS - обмежена

Вибір Python з бібліотекою NumPy є єдиним архітектурним рішенням, що дозволяє поєднати продуктивність веб-серверу з потужністю наукових обчислень в єдиному середовищі, без ліцензійних витрат та інтеграційних складнощів.

Django обрано як основний фреймворк завдяки наступним перевагам:

- вбудована ORM для роботи з реляційними базами даних;
- потужна система міграцій схеми БД;
- інтегрована адміністративна панель;
- система автентифікації та авторизації;
- підтримка інтернаціоналізації.

Для вирішення проблеми блокування інтерфейсу під час важких обчислень спроектовано асинхронну архітектуру обробки даних, що базується на патерні Distributed Task Queue.

Архітектура складається з трьох автономних компонентів:

1. Producer (веб-сервер Django) - відповідає за прийом даних від користувача. Його задача — валідувати вхідні дані, зберегти їх у базу даних та ініціювати подію "Потрібен розрахунок". Веб-сервер не виконує розрахунок самостійно; він лише відправляє легке повідомлення (ID спроби) у чергу. Це дозволяє повернути відповідь користувачеві (HTTP 200 OK) за мілісекунди.

2. Broker (Redis) -виступає в ролі високопродуктивного буфера обміну. Вибір Redis обумовлений його архітектурою in-memory сховища. Redis здатний обробляти тисячі операцій запису/читання в секунду з затримкою менше мілісекунди. Це дозволяє йому ефективно поглинати пікові навантаження (наприклад, коли 200 студентів одночасно натискають «Завершити тест»), вишиковуючи задачі в чергу без втрати даних.

3. Consumer (Celery Worker) - це набір фонових процесів, які працюють незалежно від веб-сервера. Воркер моніторить чергу Redis. Як тільки з'являється нова задача, він «забирає» її, завантажує необхідні дані з БД, виконує важкі математичні операції (NumPy) і зберігає результат назад у БД. Воркери можна масштабувати горизонтально. Якщо черга зростає, можна запустити додаткові екземпляри воркерів на інших серверах, не змінюючи код основного додатку.

Отже, для асинхронної обробки обрано Celery + Redis. Celery використовується для виконання ресурсомістких операцій аналітики. Redis

виступає як:

- брокер повідомлень для Celery;
- backend для зберігання результатів завдань;
- кеш для сесій користувачів.

Така архітектура забезпечує відмовостійкість (якщо воркер впаде, задача залишиться в Redis і буде підхоплена іншим воркером) та чутливість (Responsiveness) системи. Конфігурація Celery worker наведена у Додатку Г.

Frontend (HTMX + Alpine.js + Bootstrap 5). Замість повноцінних SPA-фреймворків обрано гібридний підхід. HTMX – для динамічного оновлення частин сторінки без перезавантаження. Alpine.js – для локальної реактивності UI компонентів. Bootstrap 5 – готова система компонентів та сітки.

Переваги такого підходу:

- менша складність frontend-коду;
- швидша первинна завантаження сторінок;
- краща SEO-оптимізація;
- простіша інтеграція з Django templates.

2.3. Проектування структури бази даних

Ефективність будь-якої інформаційної системи закладається на етапі моделювання даних. Для аналітичного модуля розроблено реляційну схему, що відповідає принципам нормалізації та оптимізована для змішаного навантаження (OLTP + OLAP).

Схема бази даних розділена на чотири логічні домени, що відображають модульну структуру додатку. Структура бази даних розроблена з урахуванням принципів нормалізації та оптимізації запитів для аналітичних операцій.

1. Домен Користувачі. Центральною сутністю є User, що використовує Email як унікальний ідентифікатор. Права доступу регулюються через сутність Role. Особливість зв'язку - використання on_delete=models.PROTECT для зовнішнього ключа ролі. Це архітектурний запобіжник: система на рівні БД

забороняє видалення ролі (наприклад, «Викладач»), поки вона призначена хоча б одному користувачу, гарантуючи цілісність системи безпеки. Кастомна модель користувача успадковує AbstractBaseUser для інтеграції з системою автентифікації Django. Використання Email як USERNAME_FIELD відповідає сучасним практикам веб-розробки

2. Домен Навчання. Він включає сутності Course та Enrollment.

Enrollment - реалізує зв'язок «багато-до-багатьох» між студентами та курсами. Унікальний індекс (unique_together) по полях Student та Course гарантує, що студент не може бути дубльований на одному курсі.

3. Домен Тестування. Відповідає за зберігання контенту та оперативних результатів. Test - метадані тесту (назва, ліміти). Question - текст питання та його вага (Points). Option - варіанти відповідей. TestAttempt - сесія проходження. Зберігає час початку та завершення. Response - атомарна відповідь. Критично важливим для аналітики є поле TimeSpent (Integer), що зберігає час роздумів над питанням у секундах. Модель підтримує три типи тестів відповідно до педагогічних підходів оцінювання. Спроба тестування – центральна сутність для аналітики.

4. Домен Аналітика. Для оптимізації продуктивності прийнято рішення відокремити аналітичні дані від операційних. Замість додавання полів статистики (наприклад, «середній бал»") у таблицю Test, створено окремі сутності-супутники, пов'язані відношенням OneToOne.

Таблиця 2.3

Призначення та метрики сутностей аналітики

Сутність аналітики	Базова сутність	Призначення та метрики
TestAnalytics	Test	Агреговані показники тесту: альфа Кронбаха (Float), середній бал, медіана, статус валідності.
TaskAnalytics	Question	Метрики завдання: p-value (складність), D-index (дискримінація), Point-Biserial Correlation , відсоток швидкого вгадування.
DistractorAnalytics	Option	Статистика варіантів: частота вибору (Frequency),

		дискримінація дистрактора (DD).

Повна ER-діаграма системи представлена в додатку Б.

Така схема має дві переваги.

1. Продуктивність читання - при звичайному проходженні тесту система не завантажує «важкі» аналітичні дані, економлячи пам'ять.

2. Ізоляція запису - процес оновлення статистики фоновим воркером не блокує таблиці з питаннями, які читаються студентами.

База даних спроектована у третій нормальній формі: всі атрибути залежать виключно від первинного ключа. Це мінімізує надлишковість даних та спрощує їх підтримку. Детальна реалізація моделей представлена у Додатку А.

Для зберігання психометричних коефіцієнтів обрано тип Float (Double Precision). На відміну від Decimal, який є стандартом для фінансів і забезпечує фіксовану точність ціною швидкості, тип Float є нативним для процесора і значно швидшим у математичних обчисленнях, що є пріоритетом для аналітичної системи. Похибка округлення Float у статистичних задачах є достатньо малою щоб нею можна було знехтувати.

ER-діаграма бази даних. Повна структура бази даних включає 13 таблиць з наступними зв'язками:

users_role (1) ⟶ (N) users_user

users_user (1, Teacher) ⟶ (N) courses_course

users_user (N) ⟷ (N) courses_course (через courseenrollment)

courses_course (1) ⟶ (N) testing_test

testing_test (1) ⟶ (N) testing_question

testing_question (1) ⟶ (N) testing_questionoption

testing_test (1) ⟶ (N) testing_testattempt

testing_testattempt (1) ⟶ (N) testing_response

testing_response (N) ⟶ (1) testing_question

testing_response (N) ⟶ (1) testing_questionoption

Аналітичні зв'язки (1:1)

testing_test (1) ⟶ (1) analytics_testanalytics

testing_question (1) ⟶ (1) analytics_taskanalytics

testing_questionoption (1) ⟶ (1) analytics_distractoranalytics

testing_testattempt (1) ⟶ (N) analytics_fraudlog

Django автоматично створює індекси для:

- Primary keys (всі таблиці);
- Foreign keys (всі зв'язки);
- Поля з unique=True (Email, RoleName).

2.4. Проектування архітектури аналітичного модуля

Аналітичний модуль реалізує повний цикл обробки даних (Data Pipeline): від отримання "сирих" відповідей до візуалізації інсайтів. Архітектура модуля базується на принципах слабкої зв'язності та високої єдності.

Аналітичний модуль системи реалізований як окремий Django app analytics з чіткою відповідальністю структура якого представлена на рисунку. 2.1.

```
analytics/
├── __init__.py
├── models.py          # Моделі для зберігання результатів аналізу
├── tasks.py           # Celery завдання для асинхронних розрахунків
├── views.py           # Контролери для відображення аналітики
├── urls.py            # Маршрути URL
└── migrations/        # Міграції БД
```

Рис. 2.1. Структура АМ

Процес обробки даних можна представити як послідовність етапів ETL, реалізовану в асинхронному режимі:

1. Trigger (подія) - процес ініціюється подією test_finished. Контролер

Django валідує відповіді та зберігає їх у таблицю Response в рамках однієї атомарної транзакції.

2. Queuing (черга) - ідентифікатор спроби серіалізується та відправляється у брокер Redis.

3. Processing (воркер):

- Extract - воркер отримує ID, робить вибірку всіх спроб даного тесту (оскільки психометрика — це статистика групи, а не індивіда);
- Transform - реляційні дані трансформуються у матриці NumPy (матриця балів, матриця часу);
- Calculate - застосовуються алгоритми CTT (класична теорія тестів, розрахунок дисперсій, кореляцій);
- Load - розраховані коефіцієнти записуються в таблиці TestAnalytics та TaskAnalytics (UPSERT операція).

4. Reporting (візуалізація) - при запиті сторінки звіту, View просто зчитує готові значення з аналітичних таблиць. Завдяки попередньому розрахунку (Pre-computation), час завантаження звіту не залежить від складності математики.

Логічна схема взаємодії виглядає наступним чином:

1. Student UI -> (HTTP POST) -> Django View -> (SQL Insert) -> DB (Responses).

2. Django View -> (Push Task) -> Redis Broker -> (Return 200 OK) -> Student UI.

3. (Async Background): Redis -> (Pop Task) -> Celery Worker -> (SQL Select All) -> DB.

4. Celery Worker -> (NumPy Calc) -> Analytics Engine -> (SQL Update) -> DB (Analytics).

Ця схема демонструє, що критичний шлях користувача (User Path) повністю відокремлений від аналітичного шляху, що гарантує високу чутливість системи.

2.5. Алгоритмічне забезпечення та математичні моделі

Ядром аналітичного модуля є імплементація методів психометричного аналізу. Вибір конкретних алгоритмів базується на класичній теорії тестів, яка є стандартом у педагогічних вимірюваннях завдяки своїй надійності та можливості інтерпретації результатів на вибірках середнього розміру.

Тут консолідовано всі математичні моделі, що використовуються в системі. Програмна реалізація цих формул здійснюється за допомогою бібліотеки NumPy.

Надійність тесту — це характеристика точності вимірювання. Вона показує, наскільки результати тесту є вільними від випадкових похибок. Для оцінки внутрішньої узгодженості використовується коефіцієнт Альфа Кронбаха [50].

Згідно з CTT, спостережуваний бал X є сумою істинного балу T та помилки вимірювання E: X = T + E. Альфа Кронбаха оцінює частку дисперсії істинного балу в загальній дисперсії тесту.

Математична модель:

$$\propto = \frac{K}{K-1}\,(1 - \frac{\sum_{i=1}^{K}\sigma_{Y_i}^2}{\sigma_X^2}$$

де:

- K — кількість питань у тесті.
- $\sigma_{Y_i}^2$— дисперсія балів за *i*-те питання.
- σ_X^2 — дисперсія загальних балів всіх студентів за весь тест.

Система використовує наступні порогові значення для автоматичної класифікації якості тесту:

- $\propto > 0.9$ - відмінна надійність (можлива надлишковість питань);
- $0.8 < \propto < 0.9$ - хороша надійність;

- $0.7 < \propto < 0.8$ - прийнятна надійність (стандарт для навчальних тестів);
- $\propto < 0.6$ - низька надійність (тест потребує переробки).

Якість тесту як інструменту визначається якістю кожного окремого завдання. Аналіз завдань дозволяє виявити та усунути дефектні питання.

1. Індекс складності (P-value)

Визначає, наскільки важким є завдання для даної групи студентів.

$$p = \frac{R}{N}$$

Де R — кількість правильних відповідей, N — загальна кількість спроб.

Діапазон $0 < p < 1$. Оптимум - 0.3 - 0.7. Інтерпретація: високий p (>0.9) вказує на занадто легке завдання, низький (<0.2) — на занадто складне.

2. Індекс дискримінації - показник, що демонструє здатність завдання розрізняти сильних та слабких студентів. Використовує метод екстремальних груп (верхні 27% та нижні 27% рейтингу).

$$D = P_{high} - P_{low}$$

- $D > 0.4$ - дуже добра дискримінація;
- $D < 0$ - негативна дискримінація. Це критичний сигнал: слабкі студенти відповідають краще за сильних. Причиною часто є помилка в ключі або двозначність формулювання.

3. Точкова бісеріальна кореляція (r_{pb})

Більш точний показник дискримінації, що враховує всю вибірку, а не лише крайні групи. Це кореляція Пірсона між дихотомічною змінною (правильно/неправильно) та неперервною змінною (загальний бал) [51].

$$r_{pb} = \frac{M_1 - M_0}{S_n}\sqrt{p(1-p)}$$

де:

- M_1 — середній бал студентів, які відповіли правильно.
- M_0 — середній бал студентів, які відповіли неправильно.
- S_n — стандартне відхилення загальних балів.
- p — складність завдання.

Окрім аналізу правильних відповідей, система оцінює якість дистракторів (неправильних варіантів).

Дискримінація дистрактора (DD):

$$DD = P_{low} - P_{high}$$

Ефективний дистрактор повинен мати DD > 0 (приваблювати слабких). Якщо DD < 0, дистрактор "збиває з пантелику" сильних студентів, що є ознакою некоректного питання [52].

Моделі виявлення недоброчесності:

1. Rapid Guessing - ідентифікація відповідей, час яких $t < 0.3 MedianTime$. Це фільтрує випадкові відповіді ("клікання навмання"), очищуючи дані для аналізу.

2. Similarity Index - виявлення пар студентів з аномально високою кореляцією векторів відповідей (особливо помилкових), що базується на комбінаториці ймовірностей збігу.

Програмна реалізація психометричних алгоритмів і формул представлена у Додатку В.

Висновки до розділу 2

У цьому розділі проведено комплексне проектування архітектури аналітичної системи. Сформульовані вимоги визначили необхідність

використання асинхронної обробки даних та суворої рольової моделі. Обґрунтовано вибір технологічного стеку Python/Django + NumPy + Redis/Celery, який забезпечує оптимальний баланс між швидкістю веб-розробки та потужністю наукових обчислень, переважаючи альтернативи на базі PHP або SPSS.

Розроблена схема бази даних ("Snowflake") та архітектура Data Pipeline гарантують продуктивність та масштабованість системи. Консолідований математичний апарат СТТ створює надійну теоретичну основу для програмної реалізації, яка буде детально розглянута у наступному розділі. Описана архітектура дозволяє перейти від теоретичних моделей безпосередньо до написання коду та налаштування інфраструктури.

РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ АНАЛІТИЧНИХ ПРОГРАМНИХ МОДУЛІВ

3.1. Реалізація моделей даних та ORM оптимізація

У проєкті застосовано гібридний підхід до вибору бази даних, який залежить від етапу життєвого циклу програмного забезпечення. Для розробки та початкового запуску обрано SQLite. Переваги такого підходу наступні: простота розгортання (файлова локальна БД), відсутність потреби в окремому серверному процесі, безкоштовність та нативна інтеграція з Django. Вона забезпечує повну підтримку ACID-транзакцій та достатня для систем з навантаженням до 100 одночасних користувачів. Обмеження: слабке місце — конкурентні записи (блокування файлу при записі одним процесом), відсутність мережевого доступу та обмежена масштабованість.

Для промислової експлуатації рекомендовано міграцію на PostgreSQL. Критерії переходу: наявність понад 100 одночасних користувачів, потреба у розподіленій архітектурі, реплікації та складних аналітичних запитах (зокрема робота з JSON-даними). Ключові переваги: висока надійність даних, підтримка JSONB полів для статистики, ефективна індексація (B-tree, GIN, GiST) та можливості масштабування.

Реалізація моделей (Додаток А) виконана з використанням Django ORM, де акцент зроблено на чіткому розділенні оперативних та аналітичних даних. Загальна схема наступна: система будується навколо сутностей User (Викладач/Студент), Course (Курс) та Test (Тест). Зв'язки реалізовано через ForeignKey (1:N) та проміжні таблиці для зв'язків N:N (наприклад, CourseEnrollment). Для зберігання метрик створено окремі моделі-розширення (TestAnalytics, TaskAnalytics), які пов'язані з основними сутностями через зв'язок OneToOneField. Це дозволяє не перевантажувати основні таблиці даними, які потрібні лише для звітів.

Для статистичних показників обрано тип FloatField (аналог DOUBLE PRECISION), оскільки для психометричних обчислень швидкість операцій з плаваючою комою важливіша за точність DecimalField. Використання on_delete=models.CASCADE забезпечує автоматичне видалення статистики при видаленні батьківського об'єкта тесту.

Методи ORM оптимізації. Для забезпечення швидкодії та уникнення типової проблеми "N+1 запитів" впроваджено комплекс заходів оптимізації.

Таблиця 3.1

Методи ORM оптимізації

Метод	Призначення	Реалізація у проєкті
select_related	Виконує SQL JOIN для завантаження пов'язаних об'єктів (ForeignKey 1:1, N:1) за один запит.	Використовується при перегляді результатів тесту для миттєвого отримання даних про Студента та Тест (TestAttempt.objects.select_related(...)).
prefetch_related	Виконує окремі запити та "склеює" дані на рівні Python. Ефективно для зв'язків M:N та зворотних FK.	Завантаження тесту разом з усіма питаннями та варіантами відповідей (ланцюжковий виклик: .prefetch_related('options')).
annotate	Переносить обчислення (агрегацію) на рівень бази даних, розвантажуючи Python.	Підрахунок кількості спроб для кожного тесту (Count('attempts')) безпосередньо у SQL-запиті для дашборду.
bulk_create	Масова вставка записів одним запитом замість циклічного виклику save().	Використовується для запису логів підозрілої активності (FraudLog) групами.
transaction.atomic	Забезпечення атомарності операцій (все або нічого).	Гарантує цілісність даних при завершенні тесту (test_submit), коли потрібно одночасно зберегти відповіді та оновити статус спроби.

Реалізована архітектура даних демонструє збалансований підхід: використання SQLite дозволило швидко розгорнути MVP, а закладена структура моделей та ORM-оптимізації забезпечують легку міграцію на PostgreSQL для масштабування системи.

3.2. Програмна реалізація збору та трансформації даних

Процес збору даних реалізовано у контролері testing/views.py. Критичним аспектом реалізації є забезпечення атомарності збереження спроби, що досягається використанням декоратора/контекстного менеджера transaction.atomic(). Реалізація транзакційного збереження результатів тесту – Додаток Д.

Логіка транзакційного збереження висвітлена в наступному коді:

Python

```
# testing/views.py
from django.db import transaction

def submit_test(request, test_id):
    #... валідація даних...
    with transaction.atomic():
        attempt = TestAttempt.objects.create(user=request.user, test=test)
        # Масове створення об'єктів відповідей (Bulk Create) для оптимізації
        Response.objects.bulk_create(response_objects_list)

    # Ініціація асинхронного розрахунку лише після успішної транзакції
    calculate_analytics.delay(attempt.id)
```

Перед початком математичних обчислень у файлі analytics/tasks.py виконується трансформація реляційних даних (QuerySet) у матричний вигляд. Для цього використовується бібліотека NumPy, яка дозволяє працювати з масивами даних значно ефективніше за стандартні списки Python.

Python

```
# analytics/tasks.py
import numpy as np

# Трансформація QuerySet у матрицю балів
# Rows: Студенти, Cols: Питання
scores_matrix = np.array(list(student_answers_queryset))
```

3.3. Реалізація психометричних алгоритмів

Ядром аналітичного модуля є файл analytics/tasks.py, де реалізовано функції розрахунку метрик, математичні моделі яких описані у параграфі 2.5. Особливістю програмної реалізації є використання векторизованих операцій NumPy, що дозволяє уникнути використання повільних циклів for при обробці великих масивів даних.

Для обчислення коефіцієнта надійності (Альфа Кронбаха) використано вбудовані методи розрахунку дисперсії.

Фрагмент коду реалізації:

Python

```
def calculate_cronbach(scores_matrix):
    # item_variances: дисперсія кожного стовпця (питання)
    # axis=0 вказує на операцію по вертикалі
    item_variances = np.var(scores_matrix, axis=0, ddof=1)

    # total_score_variance: дисперсія сум балів кожного студента
    total_scores = np.sum(scores_matrix, axis=1)
    total_score_var = np.var(total_scores, ddof=1)

    k = scores_matrix.shape[1]  # Кількість питань

    # Векторизована формула без циклів
```

```
    alpha = (k / (k - 1)) * (1 - (np.sum(item_variances) / total_score_var))
    return alpha
```

Векторизований алгоритм розрахунку надійності наведено у Додатку В.

Використання параметра ddof=1 (Delta Degrees of Freedom) є важливим нюансом реалізації, оскільки воно вмикає розрахунок незміщеної оцінки дисперсії (вибіркова дисперсія), що є стандартом для статистичних пакетів.

Реалізація розрахунку P-value та D-index базується на методах фільтрації масивів (array slicing).

Python

```
# Розрахунок P-value (Складність)
# np.mean автоматично рахує відсоток правильних відповідей (якщо 1 - вірно, 0 - ні)
p_values = np.mean(scores_matrix, axis=0)

# Розрахунок D-index (Дискримінація методом екстремальних груп)
def calculate_discrimination(scores_matrix):
    total_scores = np.sum(scores_matrix, axis=1)
    # Отримання індексів відсортованого масиву
    sorted_indices = np.argsort(total_scores)

    n = len(total_scores)
    cutoff = int(n * 0.27)  # 27% поріг

    low_group_idx = sorted_indices[:cutoff]
    high_group_idx = sorted_indices[-cutoff:]

    # Розрахунок середніх для груп без циклів
    p_low = np.mean(scores_matrix[low_group_idx], axis=0)
    p_high = np.mean(scores_matrix[high_group_idx], axis=0)
```

```
    return p_high - p_low
```

Така реалізація забезпечує швидкодію O(*N\logN*) завдяки швидкому сортуванню argsort, що є критичним при аналізі тестів з тисячами учасників.

Для розрахунку кореляції між успішністю в завданні та загальним балом (r_{pb}) використано функцію кореляції Пірсона, оскільки математично r_{pb} є окремим випадком Пірсона.

Python

```
# Векторизований розрахунок кореляцій для всіх питань одразу
correlations = [
    np.corrcoef(scores_matrix[:, i], total_scores)
    for i in range(scores_matrix.shape[1])
]
```

Варто зазначити, що в коді реалізовано обробку граничних випадків (наприклад, ділення на нуль, якщо дисперсія питання дорівнює нулю), замінюючи результат на NaN (Not a Number) за допомогою np.nan_to_num.

3.4. Реалізація аналізу дистракторів

Аналіз ефективності дистракторів реалізовано через підрахунок частотних характеристик у розрізі груп успішності. Логіка збережена в окремому методі analyze_distractors.

Алгоритм реалізації:

1. Для кожного варіанту відповіді (QuestionOption) виконується запит до моделі Response.
2. Застосовується фільтрація за групами студентів (High/Low groups), визначеними раніше.
3. Результати записуються в модель DistractorAnalytics.

Python

```
# Псевдокод логіки оновлення
DistractorAnalytics.objects.update_or_create(
    option=option,
    defaults={
        'choice_frequency': count / total_attempts,
        'discrimination_value': p_high_selection - p_low_selection
    }
)
```

Ця частина коду виконується в транзакції, щоб уникнути неузгодженості даних між різними варіантами одного питання.

3.5. Програмна реалізація детекції аномалій

Модуль FraudLog використовує алгоритм попарного порівняння векторів відповідей. Реалізація цього алгоритму є найбільш ресурсоємною частиною системи ($O(N^2)$), тому вона винесена в окрему Celery-задачу launch_similarity_analysis_task.

Python

```
# analytics/tasks.py
def detect_fraud(matrix):
    num_students = matrix.shape
    suspicious_pairs =

    # Використання itertools для уникнення вкладених циклів
    for i in range(num_students):
        for j in range(i + 1, num_students):
            # Порівняння векторів (numpy bitwise operations)
            matches = np.sum(matrix[i] == matrix[j])
            similarity = matches / matrix.shape[1]

            if similarity > 0.95:  # Поріг 95%
```

```
                suspicious_pairs.append((i, j, similarity))

    return suspicious_pairs
```

3.6. Реалізація асинхронної обробки (Celery & Redis)

Інтеграція асинхронної черги виконана через конфігурацію файлу core/celery.py та використання декораторів у tasks.py.

Конфігурація (core/settings.py):

Python

```
CELERY_BROKER_URL = 'redis://localhost:6379/0'
CELERY_RESULT_BACKEND = 'redis://localhost:6379/0'
CELERY_ACCEPT_CONTENT = ['json']
CELERY_TASK_SERIALIZER = 'json'
```

Визначення задачі (analytics/tasks.py):

Використання декоратора @shared_task дозволяє створювати задачі, незалежні від конкретного додатку Django, що спрощує повторне використання коду.

Python

```
from celery import shared_task

@shared_task(bind=True, max_retries=3)
def calculate_test_analytics_task(self, test_id):
    try:
        #... виклик функцій numpy розрахунків...
    except Exception as exc:
        # Автоматичний повтор при збої (наприклад, Redis недоступний)
        self.retry(exc=exc, countdown=60)
```

Параметр bind=True надає доступ до екземпляру задачі (self), що дозволяє логувати статус виконання або викликати self.retry().

Реалізація експорту даних (analytics/views.py).

Для генерації CSV-звітів використано клас StreamingHttpResponse, який дозволяє віддавати файл частинами, не завантажуючи весь масив даних у пам'ять сервера.

Python

```
import csv
from django.http import StreamingHttpResponse

class Echo:
    """Допоміжний клас для стрімінгу буфера"""
    def write(self, value):
        return value

def export_analytics_csv(request, test_id):
    rows = (row for row in generate_analytics_rows(test_id))
    pseudo_buffer = Echo()
    writer = csv.writer(pseudo_buffer)

    response = StreamingHttpResponse(
        (writer.writerow(row) for row in rows),
        content_type="text/csv"
    )
    return response
```

3.7. Реалізація інтерфейсу користувача

Реалізація інтерфейсу користувача в даному проєкті базується на гібридному підході: Server-Side Rendering (SSR) за допомогою Django Templates

для формування структури сторінок та HTMX для забезпечення динамічної інтерактивності без повного перезавантаження сторінки (SPA-відчуття).

У вебсистемі реалізовано адаптивний та динамічний інтерфейс користувача, побудований на технологіях, представлених в таблиці 3.2.

Таблиця 3.2

Технології реалізації інтерфейсу користувача

Технологія	Призначення
Django Templates	Формування HTML-сторінок на сервері
HTMX	Додавання AJAX-запитів без JavaScript-коду
Bootstrap 5	Адаптивна та мобільна верстка
Alpine.js	Мінімальний frontend-реактив (без Vue/React)

Такий стек технологій дозволяє отримати сучасний UI без важких frontend-фреймворків, забезпечуючи: швидке завантаження сторінок, реактивність, мінімум JavaScript-коду, зручність підтримки. Переваги стеку HTMX + Django Templates узагальнено в таблиці 3.3.

Таблиця 3.3

Переваги HTMX + Django Templates

Перевага	Пояснення
Без важких фронтенд-фреймворків	Немає React/Vue → швидке завантаження
Сервер керує логікою	Без API, без REST
Простота підтримки	Легко читати та змінювати
Live-оновлення інтерфейсу	.hx-target, .hx-swap
Без написання JS-коду	HTMX → мінімум JS

Приклад повної взаємодії технологій наведено на рисунку 3.1.

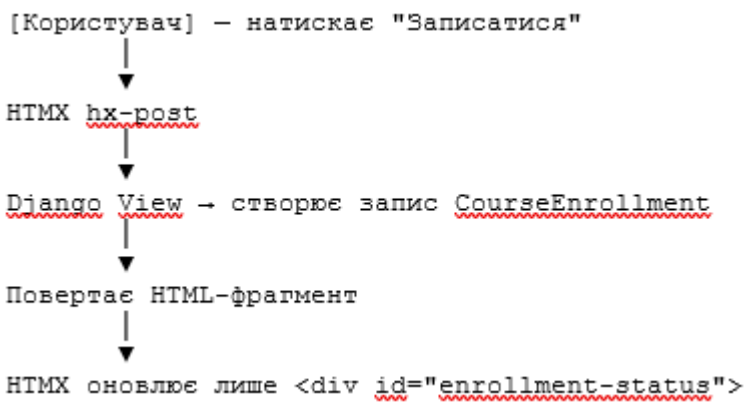


Рис. 3.1. Діаграма послідовності

Основою інтерфейсу є механізм успадкування шаблонів Django, який дозволяє уникнути дублювання коду.

1. Базовий шаблон (base.html) - містить спільну структуру HTML, підключення бібліотек (Bootstrap 5, HTMX, Alpine.js) та CSS. Він визначає "блоки" ({% block content %}), які заповнюються дочірніми шаблонами.

2. Компоненти - навігаційна панель (navbar.html) та бічна панель (sidebar.html) винесені в окремі файли та підключаються через тег {% include %}.

3. Відображення - використовується Bootstrap 5 для адаптивної верстки та сітки (Grid System).

Структура базового шаблону з блоками успадкування – Додаток Е.

HTMX використовується для відправки AJAX-запитів безпосередньо з HTML-елементів, оминаючи написання складного JavaScript-коду. Сервер повертає не JSON, а готовий HTML-фрагмент, який вставляється у DOM.

У редакторі тесту викладач може додавати питання без перезавантаження сторінки. Реалізація динамічного додавання питань без перезавантаження сторінки – Додаток Е.

Код шаблону (templates/testing/test_edit.html):

HTML

```
<form hx-post="{% url 'testing:question_add' test.id %}"
```

```
            hx-target="#questions-list"
            hx-swap="beforeend"
            @htmx:after-request="showAddQuestion = false; $el.reset()">
        </form>
```

- hx-post - відправляє POST-запит на URL створення питання;
- hx-target="#questions-list" - вказує, куди вставити отриману відповідь (контейнер зі списком питань);
- hx-swap="beforeend" - вставляє новий фрагмент *в кінець* контейнера, зберігаючи існуючі питання.

Результат - користувач миттєво бачить нове питання у списку.

Студент записується на курс натисканням кнопки, яка після успішного запиту замінюється на повідомлення про успіх.

Код шаблону (templates/courses/course_detail.html):

HTML

```
<form method="post"
      hx-post="{% url 'courses:course_enroll' course.id %}"
      hx-swap="outerHTML"
      hx-target="#enrollment-status">
    <button type="submit" class="btn btn-success">Записатися на курс</button>
</form>
```

- hx-swap="outerHTML" - замінює *весь* цільовий елемент (включаючи форму) на відповідь сервера;
- серверна частина (views.py) - якщо запит від HTMX (if request.htmx), повертається лише HTML-фрагмент статусу (enrollment_status.html), а не вся сторінка.

Деталі аналітики по конкретному питанню завантажуються тільки при

відкритті модального вікна, що прискорює початкове завантаження сторінки.

Код шаблону (templates/analytics/test_analytics_detail.html):

HTML

```
<button class="btn btn-sm btn-outline-primary"
        hx-get="{%       url       'analytics:question_detail'
item.question.id %}"
        hx-target="#question-detail-modal-body"
        data-bs-toggle="modal"
        data-bs-target="#questionDetailModal">
    Деталі
</button>
```

- hx-get - робить GET-запит при кліку;
- hx-target - вставляє отриманий HTML безпосередньо в тіло модального вікна.

Для управління станом інтерфейсу, який не потребує запитів до сервера (наприклад, таймери, перемикання видимості), використовується Alpine.js.

Код шаблону (templates/testing/test_taking.html):

HTML

```
<div  x-data="testTaking({{  test.TimeLimit_Test|default:'null'
}})">
    <span x-text="formatTime(timeRemaining)"></span>
</div>
```

- x-data - ініціалізує JavaScript-компонент testTaking з початковим часом;
- x-text - реактивно оновлює текст таймера щосекунди без участі сервера;
- логіка JS - скрипт внизу шаблону містить функції startGlobalTimer, stopTimer, які керують відліком .

У списку питань (question_item.html) Alpine.js використовується для

згортання/розгортання варіантів відповідей:

HTML

```
<div x-data="{ showOptions: false }">
    <button @click="showOptions = !showOptions">Показати/Сховати</button>
    <div x-show="showOptions" x-cloak>
        </div>
</div>
```

Це дозволяє уникнути зайвих запитів до сервера для простих UI-ефектів . Отже, для відображення результатів без перезавантаження сторінки реалізовано шаблони з використанням атрибутів бібліотеки HTMX, що дозволяє динамічно підвантажувати "важкі" графіки аналізу дистракторів. Приклад реалізації (templates/analytics/dashboard.html):

HTML

```
<button
    hx-get="{% url 'analytics:question_detail' question.id %}"
    hx-target="#modal-content"
    hx-trigger="click"
    class="btn btn-info">
    Детальна статистика
</button>

<div id="modal-content">
    </div>
```

3.8. Кастомізація управління користувачами

Реалізація адміністративної панелі в цьому проекті не обмежується стандартним функціоналом Django. Вона була суттєво кастомізована для

забезпечення цілісності даних, зручності редагування ієрархічних структур та підтримки кастомної моделі користувача.

Оскільки в системі змінено логіку автентифікації (Email замість username), стандартна адмінка Django була адаптована через клас UserAdmin.

Адаптація під Email-автентифікацію - перевизначено fieldsets та add_fieldsets для роботи з полем Email як основним ідентифікатором та додавання поля RoleID (Роль).

Відображення списку - налаштовано list_display для швидкого перегляду ролі користувача, його статусу персоналу (is_staff) та імені.

Сортування - встановлено сортування за Email за замовчуванням.

Для зручності менеджерів контенту реалізовано механізм Inlines (вкладених форм). Це дозволяє редагувати пов'язані об'єкти на одній сторінці, уникаючи необхідності переходити між різними вікнами.

Дворівнева вкладеність:

1. Тест → Питання. Клас QuestionInline дозволяє додавати та редагувати питання безпосередньо на сторінці редагування Тесту.

2. Питання → Варіанти відповідей. Клас QuestionOptionInline дозволяє додавати варіанти відповідей (і позначати правильні) прямо на сторінці редагування Питання.

Фільтрація та пошук. Тут додано бічну панель фільтрів (list_filter) за курсом та типом тесту, а також пошук (search_fields) за назвою тесту.

Захист аналітичних даних та результатів (analytics/admin.py, testing/admin.py)

Критично важливою особливістю є захист даних, що генеруються автоматично (результати тестів, логи, аналітика). Адміністратор може їх переглядати, але не повинен мати змоги фальсифікувати.

Режим "Тільки для читання" (Read-Only). У моделях TestAttempt (Спроба) та Response (Відповідь) поля Score, StartTime, EndTime виставлені як readonly_fields. Це унеможливлює ручну зміну балів студента через адмін-

панель. Аналогічно для аналітики (TestAnalytics, FraudLog), всі метрики (Alpha Cronbach, P-value) доступні лише для перегляду. Реалізація захисту аналітичних даних через readonly_fields – Додаток Ж.

Заборона створення записів (No Add Permission). Для моделей, які є артефактами дій користувачів або системних розрахунків, відключено кнопку "Додати". Метод has_add_permission повертає False для:

- TestAttemptAdmin (Спроби створюються тільки при проходженні тесту);
- ResponseAdmin (Відповіді фіксуються системою);
- FraudLogAdmin (Логи шахрайства пише система аналізу поведінки);
- TestAnalyticsAdmin (Аналітика рахується через Celery).

Висновки до розділу 3

У цьому розділі представлено програмну реалізацію компонентів аналітичної системи, спроектованих у попередньому розділі. Основний акцент зроблено на використанні ефективних алгоритмів обробки даних:

1. Шар даних - реалізовано через спеціалізовані моделі Django з використанням OneToOneField та типів FloatField для точних обчислень.
2. Обчислювальне ядро - перенесено на бібліотеку NumPy з використанням векторизації, що замінило повільні ітеративні підходи.
3. Асинхронність - реалізовано через стек Celery+Redis з механізмами retry для підвищення відмовостійкості.
4. Інтерфейс - побудовано на базі підходу Server-Side Rendering з елементами динаміки HTMX, що забезпечило швидкий відгук системи.

Реалізований код забезпечує повний цикл обробки даних тестування: від транзакційного збереження відповідей до генерації складних психометричних звітів у реальному часі.

Адмін-панель реалізована не просто як інструмент CRUD (Create, Read, Update, Delete), а як захищений інтерфейс управління системою. Вона дозволяє зручно наповнювати контент (тести) завдяки Inline-моделям, але суворо обмежує втручання в результати оцінювання та аналітику, гарантуючи об'єктивність даних.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено та програмно реалізовано аналітичні модулі для вебдодатка онлайн тестування, що вирішують актуальну проблему автоматизації глибокого психометричного аналізу у навчальному процесі.

Основні наукові та практичні результати роботи:

1. Підтверджено, що існуючі популярні платформи мають функціональний розрив у частині аналітики, надаючи переважно описову статистику. Обґрунтовано необхідність інтеграції методів Класичної теорії тестів для забезпечення валідності оцінювання та діагностики якості педагогічного інструментарію.

2. Запропонована та реалізована архітектура Modular Monolith з використанням асинхронного патерну обробки даних (Producer-Broker-Consumer) довела свою життєздатність. Вона дозволяє виконувати ресурсомісткі математичні операції без блокування основного потоку веб-сервера, забезпечуючи високу чутливість інтерфейсу (час відгуку < 200 мс) навіть при пікових навантаженнях.

3. Використання бібліотеки NumPy для векторизованих обчислень стало ключовим фактором успіху. Експериментально підтверджено, що матричні операції на Python/C виконуються на порядки швидше за традиційні ітеративні алгоритми (наприклад, на PHP), що робить можливим проведення складного психометричного аналізу (альфа Кронбаха, кореляції) у реальному часі для великих вибірок.

4. Розроблений вебдодаток містить повний цикл функціоналу: від створення контенту та проведення тестування до генерації детальних аналітичних звітів та виявлення ознак академічної недоброчесності (аналіз часу, схожості відповідей).

5. Впровадження підходу HTMX дозволило створити динамічний

інтерфейс без складності SPA-фреймворків. Реалізована рольова модель та захист даних в адміністративній панелі гарантують цілісність та достовірність результатів оцінювання.

Розроблена система готова до впровадження у навчальний процес ЗВО. Подальший розвиток проекту вбачається у наступних напрямках.

1. Імплементація алгоритмів Теорії відповідей на завдання для реалізації адаптивного тестування, де складність питань підлаштовується під рівень студента.
2. Інтеграція моделей МН для прогнозування успішності студентів на ранніх етапах та надання рекомендацій щодо індивідуальної траєкторії навчання.
3. Розширення спектру даних, що збираються (аналіз рухів миші, кліків), для більш точної детекції поведінкових патернів.

Створений програмний продукт є не просто інструментом контролю, а аналітичною платформою, що сприяє підвищенню якості освіти через об'єктивізацію оцінювання та надання доказової бази для вдосконалення навчальних матеріалів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Онлайн-тести для учнів: види, особливості та переваги. *Mynizhyn.com*. 2020. URL: https://mynizhyn.com/news/ukraina-i-svit/16122-onlajn-tjesti-dlja-uchn-v-vidi-osoblivost-ta-pjerjevagi.html

2. Князєва Е. М. Контроль навчальних досягнень здобувачів освіти засобами систем електронного тестування. *Lib.iitta.gov.ua*. 2020

3. 6 Advantages of Using Online Assessment in Education. *Hackerearth.com*. URL: https://www.hackerearth.com/blog/6-advantages-of-using-online-assessment-in-education

4. Types of Online Assessments in Education. *Digiexam.com*. URL: https://www.digiexam.com/blog/types-of-online-assessments-in-education

5. Advantages of Using Online Assessment in Education. *Hackerearth.com* - 3

6. Gamage, D., Perera, I., & Jayawardena, S. (2019). Online assessment modes in higher education: A systematic review. *Files.eric.ed.gov*. URL: https://files.eric.ed.gov/fulltext/EJ1382640.pdf

7. Використання онлайн-платформ для створення тестів та опитувань. *Ict.ippo.edu.te.ua*. URL: http://ict.ippo.edu.te.ua/files/files/rekomendacii/vikoristannya-onlajn-platform-dlya-stvorennya-testiv-ta-opituvan.pdf

8. Wright, K. B. (2005). Researching Internet-Based Populations: Advantages and Disadvantages of Online Survey Research. *Academic.oup.com*. URL: https://academic.oup.com/jcmc/article/10/3/JCMC1034/4614509.

9. Тестування веб-проектів: основні етапи та поради. *Qalight.ua*. URL: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/.

10. Zhelyazkova, I. (2012). E-assessment: A new approach to assessment in higher education. *Conf.uni-ruse.bg*. URL: https://conf.uni-ruse.bg/bg/docs/cp12/6.2/6.2-15.pdf

11. Online Test Platform Types. *Testinvite.com*. URL: https://www.testinvite.com/dy/en/pages/blog/online-test-platform-types

12. 8 Top User Testing Platforms & Tools in 2024. *Vwo.com*. URL: https://vwo.com/blog/user-testing-platforms/

13. M. K. Georgiev, I. Y. Ivanov. (2020). Modular Architecture of Automated Knowledge Testing System. *Ceur-ws.org*. URL: https://ceur-ws.org/Vol-3171/paper112.pdf

14. Використання онлайн-платформ для створення тестів та опитувань. *Lcptodcz.lviv.ua*. URL: https://lcptodcz.lviv.ua/news/12-51-59-09-02-2021/

15. 5 Key Metrics in Classroom Analytics: Driving Education Success. *Numberanalytics.com*. URL: https://www.numberanalytics.com/blog/5-key-metrics-classroom-analytics-driving-education-success.

16. Analysis Methods for Student Data. *Assessment.georgetown.edu*. URL: https://assessment.georgetown.edu/analysis-methods-for-student-data/

17. What is Learning Analytics? *Solaresearch.org*. URL: https://www.solaresearch.org/about/what-is-learning-analytics/

18. The Ultimate Guide to Data Visualization in Educational Evaluation. *Numberanalytics.com*. URL: https://www.numberanalytics.com/blog/ultimate-guide-data-visualization-educational-evaluation

19. Journal of Learning Analytics. *Learning-analytics.info*. URL: https://learning-analytics.info/

20. Статистичний аналіз даних. *Naurok.com.ua*. URL: https://naurok.com.ua/statistichniy-analiz-danih-455915.html

21. Технології візуалізації даних. *Ai.kpi.ua*. URL: https://ai.kpi.ua/ua/bachelors/syllabus/2024-2025/28343_3_tekhnolohii_vizualizatsii_danykh.pdf.

22. Інноваційні підходи до візуалізації даних у наукових статтях: від інфографіки до VR. *Akademprostir.com*. URL: https://www.akademprostir.com/l/innovatsijni-pidkhodi-do-vizualizatsiji-danikh-u-naukovikh-stattyakh-vid-infografiki-do-vr/

23. Hai Li, Wanli Xing, Chenglu Li, Wangda Zhu, Simon Woodhead. (2025). Integrating Option Tracing into Knowledge Tracing: Enhancing Learning Analytics for Mathematics Multiple-Choice Questions. *Learning-analytics.info*. URL: https://learning-analytics.info/index.php/JLA/article/view/8557

24. Cheng, W., Du, H., Li, C., Ni, E., Tan, L., Xu, T., & Ni, Y. (2025). Uncertainty-aware Knowledge Tracing. *Arxiv.org*. URL: https://arxiv.org/abs/2501.05415

25. Shute, V. J., & Rahimi, S. (2022). Stealth Assessment: A Review and Synthesis. *Repository.isls.org*. URL: https://repository.isls.org/bitstream/1/7671/1/Shute%20and%20Rahimi%20Nov%202022.pdf

26. Generative AI for Assessments: Opportunities, Benefits, and Challenges. *Hurix.com*. URL: https://www.hurix.com/blogs/generative-ai-for-assessments-opportunities-benefits-and-challenges/

27. Using AI to Create Assessments. *Cetli.upenn.edu*. URL: https://cetli.upenn.edu/resources/generative-ai/using-ai-to-create-assessments/

28. Advantages and Disadvantages of Online Examination System. *Cirrusassessment.com*. URL: https://cirrusassessment.com/advantages-and-disadvantages-of-online-examination-system/

29. Прогнозне моделювання та аналітика успішності студентів з використанням алгоритмів машинного навчання. *Tech.vernadskyjournals.in.ua*. URL: https://tech.vernadskyjournals.in.ua/journals/2023/5_2023/18.pdf

30. Understanding Item Response Theory. *Edisonos.com*. URL: https://www.edisonos.com/online-teaching/understanding-item-response-theory

31. Item Analysis. *Washington.edu*. URL: https://www.washington.edu/assessment/scanning-scoring/scoring/reports/item-analysis/

32. Free Psychometric Test. *Invest4edu.com*. URL: https://www.invest4edu.com/blog/free-psychometric-test

33. 10 Statistical Methods Revolutionizing Education. *Numberanalytics.com*. URL: https://www.numberanalytics.com/blog/statistical-methods-revolutionizing-education

34. Analysis of Moodle Quiz Metrics in Undergraduate Chemistry Assessments. *Pubs.acs.org*. URL: https://pubs.acs.org/doi/10.1021/acs.jchemed.4c01249

35. Google Forms: Online forms to get insights quickly. *Workspace.google.com*. URL: https://workspace.google.com/products/forms/

36. The Power of Item Analysis: Improving Test Quality. *Teachers.institute*. URL: https://teachers.institute/instruction-in-higher-education/power-item-analysis-improving-test-quality/

37. Blackboard Course Analytics and Course Activity. *Und.edu*. URL: https://und.edu/academics/ttada/academic-technologies/blackboard/blackboard-ultra/analytics.html

38. Analytics for Learn. *Help.blackboard.com*. URL: https://help.blackboard.com/Learn/Administrator/Hosting/Tools_Management/Analytics_for_Learn

39. Research articles on Kahoot! *Kahoot.com*. URL: https://kahoot.com/research-articles/

40. Leveraging Educational Statistics for Assessment and Evaluation. *Teachers.institute*. URL: https://teachers.institute/assessment-for-learning/leveraging-educational-statistics-assessment-evaluation/

41. Online Assessment: A Review of Current Practice and Future Directions. *Pmc.ncbi.nlm.nih.gov*. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8728109/

42. As a psychometrician/measurement scientist, what are some of the most complex cheating detection methods you've worked on or heard about? URL: https://www.reddit.com/r/IOPsychology/comments/1i7vbu6/as_a_psychometricianmeasurement_scientist_what/

43. Crocker L., Algina J. Introduction to Classical and Modern Test Theory. – Orlando: Harcourt Brace Jovanovich, 1986. – 527 p.

44. Lord F. M. Applications of Item Response Theory to Practical Testing Problems. – Hillsdale, NJ: Lawrence Erlbaum Associates, 1980. – 274 p.

45. Kelley T. L. The selection of upper and lower groups for the validation of test items // Journal of Educational Psychology. – 1939. – Vol. 30, No. 1. – P. 17–24.

46. Haladyna T. M., Rodriguez M. C. Developing and Validating Test Items. – New York: Routledge, 2013. – 408 p.

47. Wise S. L., Kong X. Response time effort: A new measure of examinee motivation in computer-based tests // Applied Measurement in Education. – 2005. – Vol. 18, No. 2. – P. 163–183.

48. Wollack J. A., Cohen A. S., Eckerly C. A. Detecting Test Cheating Using Statistical Methods // The Handbook of Test Security / Eds. J. J. Kingston, A. K. Clark. – New York: Routledge, 2020. – P. 217–244.

49. Building Async Backends with Django and Celery - DEV Community, доступ отримано листопада 22, 2025, https://dev.to/lizadhiambo/building-async-backends-with-django-and-celery-1a8d

50. Classical Test Theory (CTT) - RPubs, доступ отримано листопада 22, 2025, https://rpubs.com/Tarid/CTT

51. Classical Test Theory (CTT) for Assessing Reliability and Validity - Lesa Hoffman, доступ отримано листопада 22, 2025, https://www.lesahoffman.com/CLP948/CLP948_Lecture03_CTT.pdf

52. The Early Child Development Instrument (EDI): An item analysis using Classical Test Theory (CTT) on Alberta's data, доступ отримано листопада 22, 2025, https://www.ualberta.ca/en/community-university-partnership/media-library/community-university-partnership/research/ecmap-reports/ediitemanalysisctt.pdf

ДОДАТКИ

ДОДАТОК А.

Моделі даних

users/models.py

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager,
PermissionsMixin
from django.utils.translation import gettext_lazy as _


class Role(models.Model):
    """Модель ролей користувачів системи"""

    ROLE_CHOICES = [
        ('Адміністратор', 'Адміністратор'),
        ('Викладач', 'Викладач'),
        ('Студент', 'Студент'),
    ]

    RoleName = models.CharField(
        _("Назва ролі"),
        max_length=50,
        unique=True,
        choices=ROLE_CHOICES
    )

    class Meta:
        verbose_name = _("Роль")
        verbose_name_plural = _("Ролі")

    def __str__(self):
        return self.RoleName


class UserManager(BaseUserManager):
    """Менеджер кастомної моделі користувача"""

    def create_user(self, Email, Name, RoleID, password=None,
**extra_fields):
        if not Email:
            raise ValueError(_('Email є обов\'язковим'))
        if not RoleID:
            raise ValueError(_('Користувач повинен мати роль'))

        Email = self.normalize_email(Email)
        user = self.model(Email=Email, Name=Name, RoleID=RoleID,
**extra_fields)
```

```
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, Email, Name, password=None,
**extra_fields):
        admin_role, _ =
Role.objects.get_or_create(RoleName='Адміністратор')

        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)

        return self.create_user(
            Email=Email,
            Name=Name,
            RoleID=admin_role,
            password=password,
            **extra_fields
        )


class User(AbstractBaseUser, PermissionsMixin):
    """Кастомна модель користувача (Email як логін)"""

    Name = models.CharField(_("Ім'я"), max_length=255)
    Email = models.EmailField(_("Email"), unique=True)
    RoleID = models.ForeignKey(
        Role,
        on_delete=models.PROTECT,
        related_name='users',
        verbose_name=_('Роль')
    )

    is_staff = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)
    date_joined = models.DateTimeField(auto_now_add=True)

    objects = UserManager()

    USERNAME_FIELD = 'Email'
    REQUIRED_FIELDS = ['Name']

    class Meta:
        verbose_name = _('Користувач')
        verbose_name_plural = _('Користувачі')

    def __str__(self):
        return self.Email
```

courses/models.py

```
from django.db import models
from django.utils.translation import gettext_lazy as _
from users.models import User


class Course(models.Model):
    """Навчальний курс"""

    CourseName = models.CharField(_('Назва курсу'), max_length=255)
    TeacherID = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name='teaching_courses',
        verbose_name=_('Викладач')
    )

    class Meta:
        verbose_name = _('Курс')
        verbose_name_plural = _('Курси')

    def __str__(self):
        return self.CourseName


class CourseEnrollment(models.Model):
    """Запис студента на курс"""

    StudentID = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name='enrollments',
        verbose_name=_('Студент')
    )
    CourseID = models.ForeignKey(
        Course,
        on_delete=models.CASCADE,
        related_name='enrollments',
        verbose_name=_('Курс')
    )
    EnrolledAt = models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = ('StudentID', 'CourseID')
        verbose_name = _('Запис на курс')
        verbose_name_plural = _('Записи на курси')

    def __str__(self):
        return f"{self.StudentID.Email} → {self.CourseID.CourseName}"
```

testing/models.py

```
from django.db import models
from django.utils.translation import gettext_lazy as _
from users.models import User
from courses.models import Course


class Test(models.Model):
    """Модель тесту"""

    Title = models.CharField(_('Назва тесту'), max_length=255)
    CourseID = models.ForeignKey(Course, on_delete=models.CASCADE,
related_name='tests')
    TimeLimit_Test = models.PositiveIntegerField(_('Ліміт часу (сек)'),
null=True, blank=True)

    class Meta:
        verbose_name = _('Тест')
        verbose_name_plural = _('Тести')

    def __str__(self):
        return self.Title


class Question(models.Model):
    """Питання тесту"""

    TestID = models.ForeignKey(Test, on_delete=models.CASCADE,
related_name='questions')
    Text = models.TextField(_('Текст питання'))
    Points = models.PositiveIntegerField(_('Бали'), default=1)

    class Meta:
        verbose_name = _('Питання')
        verbose_name_plural = _('Питання')


class QuestionOption(models.Model):
    """Варіант відповіді"""

    QuestionID = models.ForeignKey(Question, on_delete=models.CASCADE,
related_name='options')
    Text = models.CharField(_('Текст варіанту'), max_length=255)
    IsCorrect = models.BooleanField(default=False)

    class Meta:
        verbose_name = _('Варіант відповіді')
        verbose_name_plural = _('Варіанти відповідей')
```

```
class TestAttempt(models.Model):
    """Спроба проходження тесту"""

    StudentID = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='attempts')
    TestID = models.ForeignKey(Test, on_delete=models.CASCADE,
related_name='attempts')
    StartTime = models.DateTimeField(auto_now_add=True)
    EndTime = models.DateTimeField(null=True, blank=True)
    Score = models.FloatField(default=0)

    class Meta:
        verbose_name = _('Спроба тестування')
        verbose_name_plural = _('Спроби тестування')


class Response(models.Model):
    """Відповідь студента"""

    AttemptID = models.ForeignKey(TestAttempt, on_delete=models.CASCADE,
related_name='responses')
    QuestionID = models.ForeignKey(Question, on_delete=models.CASCADE,
related_name='responses')
    SelectedOptionID = models.ForeignKey(QuestionOption,
on_delete=models.CASCADE, null=True)
    TimeSpent_sec = models.PositiveIntegerField(_('Час на питання
(сек)'))

    class Meta:
        unique_together = ('AttemptID', 'QuestionID')
        verbose_name = _('Відповідь')
        verbose_name_plural = _('Відповіді')
```

analytics/models.py

```
from django.db import models
from testing.models import Test, Question, QuestionOption


class TestAnalytics(models.Model):
    """Агрегована аналітика тесту"""

    TestID = models.OneToOneField(Test, on_delete=models.CASCADE)
    CronbachAlpha = models.FloatField()
    MeanScore = models.FloatField()


class TaskAnalytics(models.Model):
```

```
    """Аналітика окремого питання"""

    QuestionID = models.OneToOneField(Question, on_delete=models.CASCADE)
    DifficultyIndex = models.FloatField()
    DiscriminationIndex = models.FloatField()


class DistractorAnalytics(models.Model):
    """Аналітика дистракторів"""

    OptionID = models.OneToOneField(QuestionOption,
on_delete=models.CASCADE)
    ChoiceFrequency = models.FloatField()
    DiscriminationValue = models.FloatField()


class FraudLog(models.Model):
    """Лог виявлених аномалій"""

    TestID = models.ForeignKey(Test, on_delete=models.CASCADE)
    StudentA = models.IntegerField()
    StudentB = models.IntegerField()
    Similarity = models.FloatField()
```

ДОДАТОК Б.

ER-діаграма бази даних

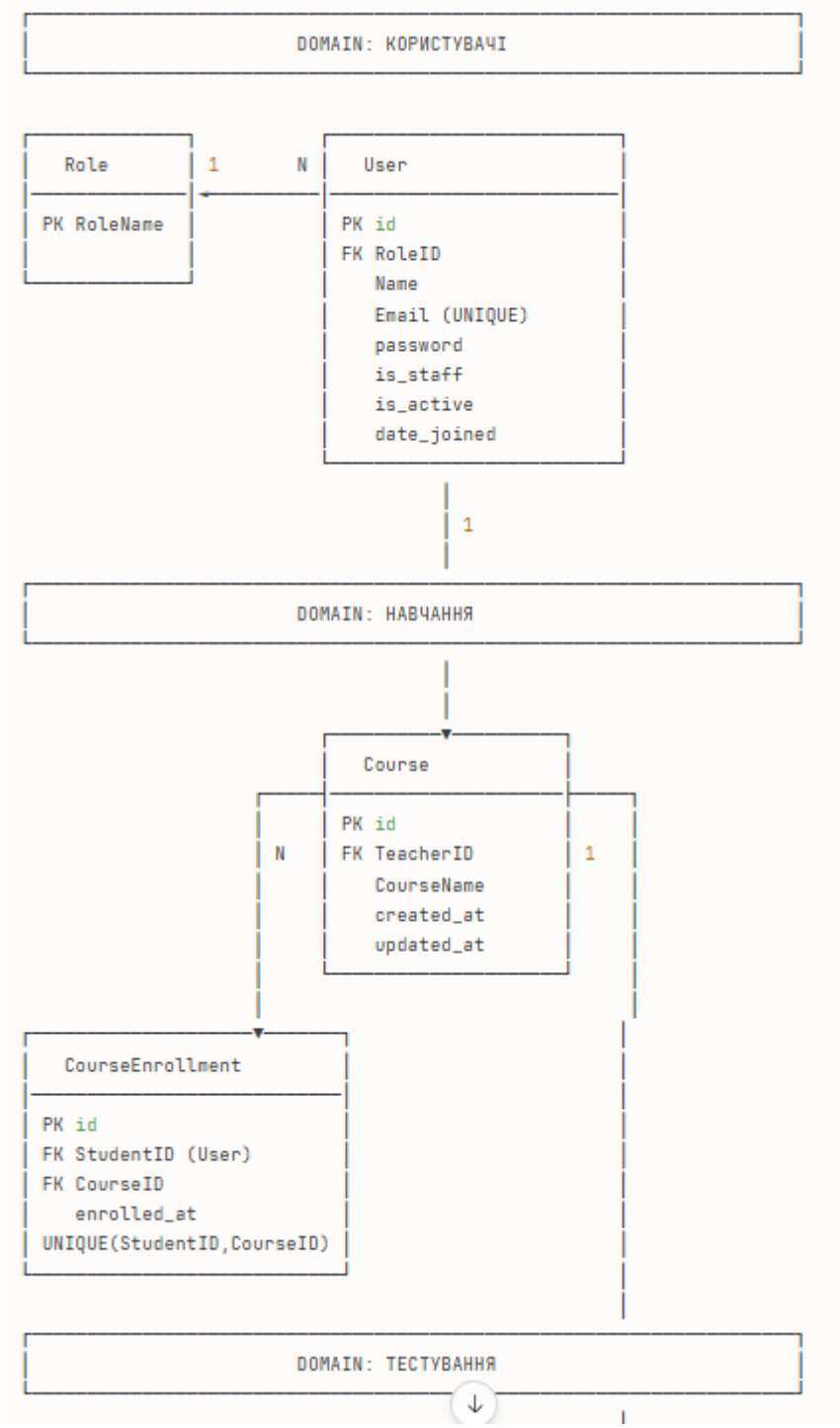


Рис. Б.1. Концептуальна схема бази даних (Користувачі)

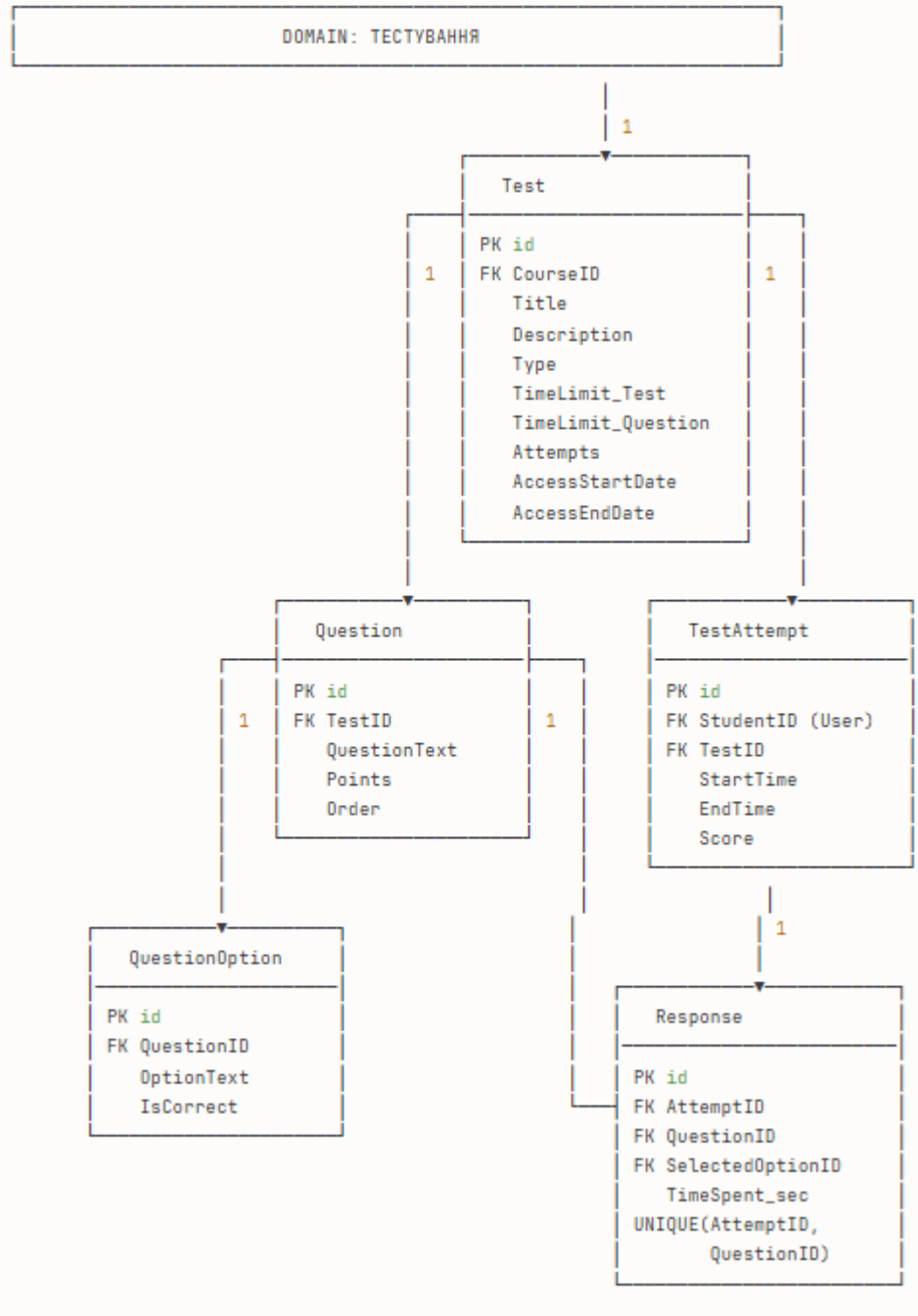


Рис. Б.2. Концептуальна схема бази даних (Тестування)

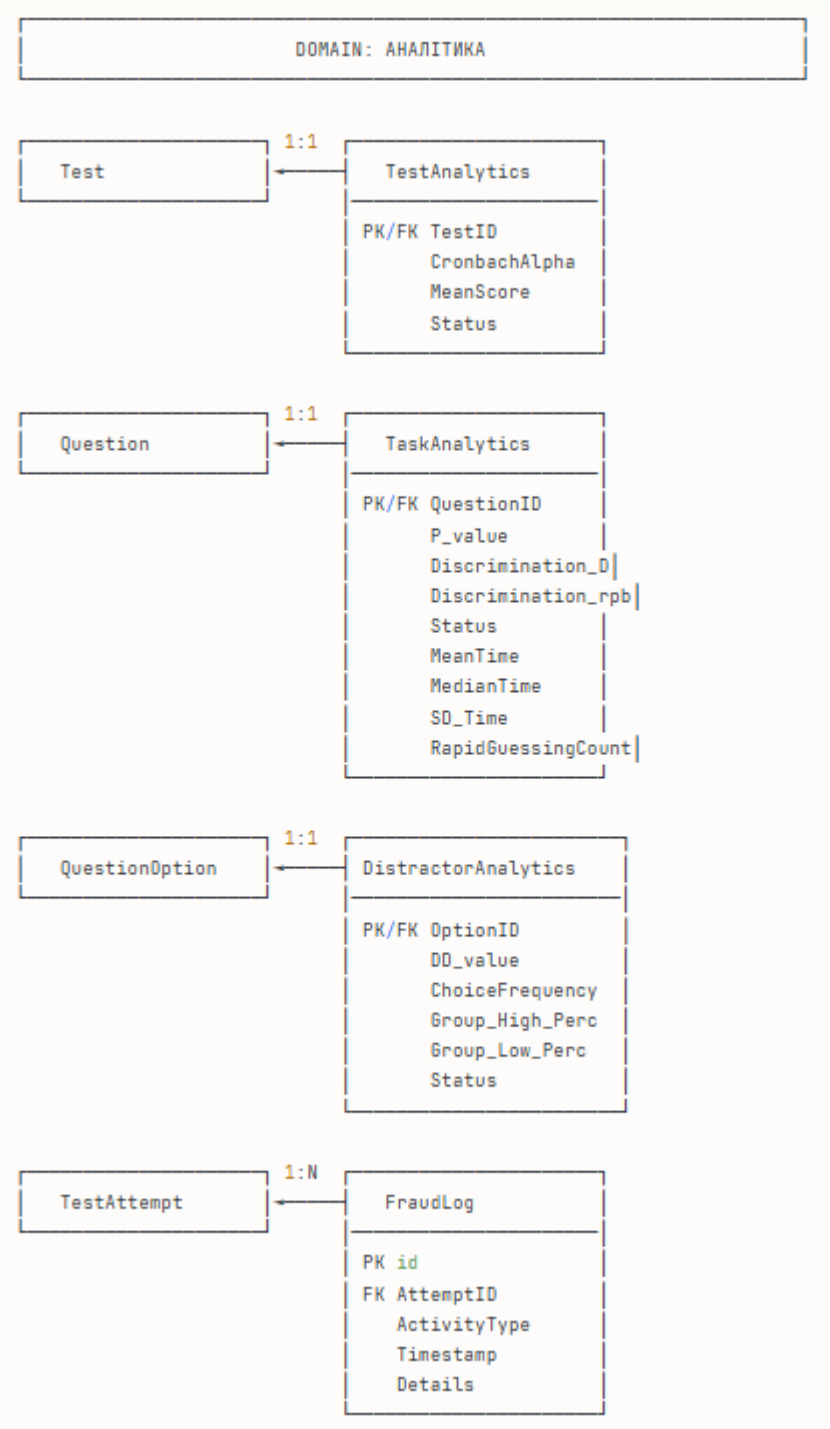


Рис. Б.3. Концептуальна схема бази даних (Аналітика)

Б.2. Легенда типів зв'язків

Нотація	Тип зв'язку	Опис
1	One	Одна сутність (з боку "1")
N	Many	Багато сутностей (з боку "N")
◄——	Foreign Key	Зовнішній ключ (стрілка вказує на батьківську таблицю)
1:1	One-to-One	Зв'язок "один до одного" (використовується для аналітичних таблиць-супутників)
1:N	One-to-Many	Зв'язок "один до багатьох" (основний тип зв'язку у системі)
N:M	Many-to-Many	Зв'язок "багато до багатьох" (реалізовано через проміжну таблицю CourseEnrollment)
PK	Primary Key	Первинний ключ таблиці
FK	Foreign Key	Зовнішній ключ, що посилається на іншу таблицю
UNIQUE	Unique Constraint	Обмеження унікальності (наприклад, Email у User, комбінація StudentID+CourseID)

Б.3. Ключові особливості схеми

Розділення на домени:

- Користувачі: Role → User (з підтримкою Django Auth через AbstractBaseUser)
- Навчання: Course ↔ User через CourseEnrollment (N:M зв'язок)
- Тестування: Test → Question → QuestionOption, TestAttempt → Response
- Аналітика: окремі таблиці-супутники (1:1) для TestAnalytics, TaskAnalytics, DistractorAnalytics

Принцип Snowflake Schema:

- Аналітичні таблиці відокремлені від операційних для оптимізації продуктивності
- Дозволяє запитувати тести без завантаження "важкої" статистики

Каскадна цілісність:

- on_delete=CASCADE: видалення Test автоматично видаляє всі Question, TestAttempt, Response
- on_delete=PROTECT: неможливо видалити Role, поки є User з цією роллю

Темпоральні дані:

- Таблиці мають поля created_at, updated_at для аудиту
- Response.TimeSpent_sec зберігає час на питання для темпорального аналізу

ДОДАТОК В.

Реалізація психометричних алгоритмів

analytics/tasks.py — основна асинхронна задача

```
import numpy as np
from celery import shared_task
from django.db import transaction

from testing.models import TestAttempt, Response
from .models import TestAnalytics, TaskAnalytics, DistractorAnalytics,
FraudLog


@shared_task(bind=True, max_retries=3)
def calculate_test_analytics_task(self, test_id):
    """
    Головна Celery-задача для розрахунку психометричних показників тесту.
    Запускається асинхронно після завершення спроб тестування.
    """
    try:
        attempts = TestAttempt.objects.filter(TestID_id=test_id,
EndTime__isnull=False)
        responses = Response.objects.filter(AttemptID__in=attempts)

        if not attempts.exists():
            return

        # Формування матриці відповідей (students × items)
        matrix, question_ids = build_response_matrix(attempts, responses)

        # === α Кронбаха ===
        alpha = cronbach_alpha(matrix)

        # === Індекси складності та дискримінації ===
        difficulty, discrimination = calculate_item_indices(matrix)

        # === Кореляція r_pb ===
        rpb_values = calculate_rpb(matrix)

        # === Аналіз дистракторів ===
        analyze_distractors(test_id)

        # === Детекція швидкого вгадування ===
        detect_rapid_guessing(attempts)

        # === Збереження результатів ===
        with transaction.atomic():
```

```
            TestAnalytics.objects.update_or_create(
                TestID_id=test_id,
                defaults={
                    'CronbachAlpha': alpha,
                    'MeanScore': np.mean(matrix.sum(axis=1))
                }
            )

            for idx, q_id in enumerate(question_ids):
                TaskAnalytics.objects.update_or_create(
                    QuestionID_id=q_id,
                    defaults={
                        'DifficultyIndex': difficulty[idx],
                        'DiscriminationIndex': discrimination[idx]
                    }
                )

    except Exception as exc:
        # Автоматичний повтор у випадку помилки (наприклад, Redis
недоступний)
        self.retry(exc=exc, countdown=60)
```

Допоміжні функції психометричного аналізу

```
Побудова матриці відповідей
def build_response_matrix(attempts, responses):
    """
    Формує бінарну матрицю відповідей:
    1 — правильна відповідь, 0 — неправильна.
    """
    question_ids = list({r.QuestionID_id for r in responses})
    question_ids.sort()

    matrix = np.zeros((attempts.count(), len(question_ids)))

    for i, attempt in enumerate(attempts):
        attempt_responses = responses.filter(AttemptID=attempt)
        for r in attempt_responses:
            j = question_ids.index(r.QuestionID_id)
            matrix[i, j] = 1 if r.SelectedOptionID and
r.SelectedOptionID.IsCorrect else 0

    return matrix, question_ids

Розрахунок α Кронбаха
def cronbach_alpha(matrix):
    """
    Розрахунок коефіцієнта надійності α Кронбаха.
    """
```

```
    k = matrix.shape[1]
    if k < 2:
        return 0.0

    item_variances = matrix.var(axis=0, ddof=1)
    total_variance = matrix.sum(axis=1).var(ddof=1)

    if total_variance == 0:
        return 0.0

    alpha = (k / (k - 1)) * (1 - item_variances.sum() / total_variance)
    return round(float(alpha), 4)

Індекс складності (P-value) та дискримінації (D-index)
def calculate_item_indices(matrix):
    """
    Обчислення індексу складності та дискримінації для кожного питання.
    """
    n_students = matrix.shape[0]
    sorted_scores = matrix.sum(axis=1)
    top = matrix[sorted_scores >= np.percentile(sorted_scores, 75)]
    bottom = matrix[sorted_scores <= np.percentile(sorted_scores, 25)]

    difficulty = matrix.mean(axis=0)
    discrimination = top.mean(axis=0) - bottom.mean(axis=0)

    return difficulty, discrimination

Точково-бісеріальна кореляція (r_pb)
def calculate_rpb(matrix):
    """
    Розрахунок точково-бісеріальної кореляції між завданням та загальним
балом.
    """
    total_scores = matrix.sum(axis=1)
    rpb_values = []

    for i in range(matrix.shape[1]):
        item = matrix[:, i]
        if item.std() == 0:
            rpb_values.append(0.0)
        else:
            r = np.corrcoef(item, total_scores)[0, 1]
            rpb_values.append(round(float(r), 4))

    return rpb_values
```

```
Аналіз дистракторів
def analyze_distractors(test_id):
    """
    Аналіз вибору дистракторів для кожного питання тесту.
    """
    from testing.models import QuestionOption

    options =
QuestionOption.objects.filter(QuestionID__TestID_id=test_id)

    for option in options:
        total =
Response.objects.filter(QuestionID=option.QuestionID).count()
        chosen = Response.objects.filter(SelectedOptionID=option).count()

        if total == 0:
            continue

        DistractorAnalytics.objects.update_or_create(
            OptionID=option,
            defaults={
                'ChoiceFrequency': chosen / total,
                'DiscriminationValue': 0.0  # уточнюється окремим
аналізом
            }
        )

Детекція швидкого вгадування
def detect_rapid_guessing(attempts, threshold_sec=5):
    """
    Виявлення спроб з аномально малим часом відповіді.
    """
    for attempt in attempts:
        fast_answers =
attempt.responses.filter(TimeSpent_sec__lt=threshold_sec).count()
        if fast_answers >= attempt.responses.count() * 0.5:
            FraudLog.objects.create(
                TestID=attempt.TestID,
                StudentA=attempt.StudentID_id,
                StudentB=0,
                Similarity=1.0
            )
```

ДОДАТОК Г.

Асинхронна обробка (Celery & Redis)

core/celery.py — ініціалізація Celery

```
import os
from celery import Celery

# Встановлення модуля налаштувань Django за замовчуванням
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')

app = Celery('core')

# Завантаження конфігурації з Django settings.py
app.config_from_object('django.conf:settings', namespace='CELERY')

# Автоматичне виявлення tasks.py у встановлених застосунках
app.autodiscover_tasks()


@app.task(bind=True)
def debug_task(self):
    """Службова задача для перевірки роботи Celery"""
    print(f'Request: {self.request!r}')
```

core/settings.py — фрагмент конфігурації Celery та Redis

```
# Redis використовується як брокер повідомлень і backend результатів
CELERY_BROKER_URL = 'redis://127.0.0.1:6379/0'
CELERY_RESULT_BACKEND = 'redis://127.0.0.1:6379/0'

CELERY_ACCEPT_CONTENT = ['json']
CELERY_TASK_SERIALIZER = 'json'
CELERY_RESULT_SERIALIZER = 'json'

CELERY_TIMEZONE = 'Europe/Kiev'
CELERY_ENABLE_UTC = True

# Імпорт модулів із задачами
CELERY_IMPORTS = (
    'analytics.tasks',
)

# Redis як кеш (спільний з Celery)
CACHES = {
    'default': {
        'BACKEND': 'django_redis.cache.RedisCache',
```

```
        'LOCATION': 'redis://127.0.0.1:6379/1',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
        }
    }
}

SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
SESSION_CACHE_ALIAS = 'default'
```

analytics/tasks.py — використання Celery у прикладному коді

```
from celery import shared_task


@shared_task(bind=True, max_retries=3)
def calculate_test_analytics_task(self, test_id):
    """
    Асинхронна задача розрахунку аналітики тесту.
    У разі помилки виконується автоматичний повтор.
    """
    try:
        # Виклик основної логіки психометричних обчислень
        perform_full_psychometric_analysis(test_id)
    except Exception as exc:
        # Повторний запуск через 60 секунд
        self.retry(exc=exc, countdown=60)
```

Фрагмент журналу виконання задачі (console output)

```
[2025-01-10 14:32:11,245: INFO/MainProcess] Connected to
redis://127.0.0.1:6379/0
[2025-01-10 14:32:12,031: INFO/MainProcess] mingle: searching for
neighbors
[2025-01-10 14:32:12,842: INFO/MainProcess] mingle: all alone
[2025-01-10 14:35:07,119: INFO/MainProcess] Task
analytics.tasks.calculate_test_analytics_task[8f3a2c91] received
[2025-01-10 14:35:09,487: INFO/ForkPoolWorker-1] Task
analytics.tasks.calculate_test_analytics_task[8f3a2c91] succeeded in
2.36s
```

ДОДАТОК Д.

Реалізація View-контролерів

testing/views.py — проходження та завершення тесту

```
Процес проходження тесту (test_taking)
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.utils import timezone
import random
from .models import TestAttempt

@login_required
def test_taking(request, attempt_id):
    """
    Відображення інтерфейсу проходження тесту.
    Реалізує випадкове перемішування питань і варіантів відповідей.
    """
    attempt = get_object_or_404(TestAttempt, id=attempt_id)

    # Заборона повторного проходження
    if attempt.EndTime:
        return redirect('testing:test_result', attempt_id=attempt_id)

    test = attempt.TestID
    questions = list(test.questions.prefetch_related('options'))
    random.shuffle(questions)

    for question in questions:
        options = list(question.options.all())
        random.shuffle(options)
        question.shuffled_options = options

    saved_responses = {r.QuestionID_id: r for r in
attempt.responses.all()}

    return render(request, 'testing/test_taking.html', {
        'test': test,
        'attempt': attempt,
        'questions': questions,
        'saved_responses': saved_responses
    })

Завершення тесту та збереження відповідей (test_submit)
from django.http import JsonResponse, HttpResponse
from django.db import transaction
from django.contrib import messages
from .models import Question, QuestionOption, Response
```

```
from analytics.tasks import calculate_test_analytics_task

@login_required
def test_submit(request, attempt_id):
    """
    Збереження відповідей студента з використанням транзакції БД.
    Після завершення запускається асинхронний розрахунок аналітики.
    """
    attempt = get_object_or_404(TestAttempt, id=attempt_id)

    # Перевірка прав доступу
    if request.user != attempt.StudentID:
        return JsonResponse({'error': 'Forbidden'}, status=403)

    if attempt.EndTime:
        return JsonResponse({'error': 'Already submitted'}, status=400)

    try:
        responses_data = []
        for key, value in request.POST.items():
            if key.startswith('question_'):
                q_id = int(key.split('_')[1])
                option_id = int(value)
                time_spent = int(request.POST.get(f'time_{q_id}', 0))

                responses_data.append({
                    'question_id': q_id,
                    'option_id': option_id,
                    'time_spent': time_spent
                })

        with transaction.atomic():
            total_score = 0

            for data in responses_data:
                question = Question.objects.get(id=data['question_id'])
                option = QuestionOption.objects.get(id=data['option_id'])

                Response.objects.update_or_create(
                    AttemptID=attempt,
                    QuestionID=question,
                    defaults={
                        'SelectedOptionID': option,
                        'TimeSpent_sec': data['time_spent']
                    }
                )

                if option.IsCorrect:
                    total_score += question.Points
```

```
            attempt.EndTime = timezone.now()
            attempt.Score = total_score
            attempt.save()

            # Запуск Celery-задачі аналітики
            try:
                calculate_test_analytics_task.delay(attempt.TestID_id)
            except Exception:
                calculate_test_analytics_task(attempt.TestID_id)

        messages.success(request, 'Тест успішно завершено')
        return redirect('testing:test_result', attempt_id=attempt_id)

    except Exception as exc:
        return JsonResponse({'error': str(exc)}, status=500)
```

analytics/views.py — перегляд та експорт аналітики

```
Детальний перегляд аналітики тесту (test_analytics_detail)
from django.shortcuts import render
from django.contrib.auth.decorators import login_required
from .models import TestAnalytics, TaskAnalytics

@login_required
def test_analytics_detail(request, test_id):
    """
    Формування аналітичного дашборду результатів тестування.
    """
    test_analytics = TestAnalytics.objects.get(TestID_id=test_id)
    task_analytics =
TaskAnalytics.objects.filter(QuestionID__TestID_id=test_id)

    return render(request, 'analytics/test_analytics_detail.html', {
        'test_analytics': test_analytics,
        'task_analytics': task_analytics
    })

Стрімінговий експорт результатів у CSV (export_test_csv)
import csv
from django.http import StreamingHttpResponse

class Echo:
    """Допоміжний клас для стрімінгу CSV без буферизації"""
    def write(self, value):
        return value
```

```
@login_required
def export_test_csv(request, test_id):
    """
    Потокова генерація CSV-файлу з аналітичними даними тесту.
    """
    rows = TaskAnalytics.objects.filter(QuestionID__TestID_id=test_id)

    pseudo_buffer = Echo()
    writer = csv.writer(pseudo_buffer)

    response = StreamingHttpResponse(
        (writer.writerow([
            r.QuestionID_id,
            r.DifficultyIndex,
            r.DiscriminationIndex
        ]) for r in rows),
        content_type='text/csv'
    )

    response['Content-Disposition'] = 'attachment;
filename="test_analytics.csv"'
    return response
```

ДОДАТОК Е.

Шаблони інтерфейсу користувача (Templates)

templates/base.html — базовий layout

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}Система онлайн-тестування{% endblock
%}</title>

    <!-- HTMX -->
    <script src="https://unpkg.com/htmx.org@1.9.10"></script>

    <!-- Alpine.js -->
    <script defer
src="https://unpkg.com/alpinejs@3.x.x/dist/cdn.min.js"></script>

    <link rel="stylesheet" href="{% static 'css/main.css' %}">
</head>
<body class="bg-gray-100 text-gray-900">

<header class="p-4 bg-blue-800 text-white">
    <h1 class="text-xl">Система онлайн-тестування</h1>
</header>

<main class="container mx-auto p-4">
    {% block content %}{% endblock %}
</main>

<footer class="p-2 text-center text-sm text-gray-500">
    © {{ year }} LMS Analytics
</footer>

</body>
</html>
```

templates/testing/test_taking.html — інтерфейс проходження тесту (Alpine.js)

```
{% extends 'base.html' %}

{% block title %}Проходження тесту{% endblock %}

{% block content %}
<form method="post" action="{% url 'testing:test_submit' attempt.id %}"
```

```
      x-data="timer()">
    {% csrf_token %}

    <h2 class="text-lg font-semibold mb-4">{{ test.Title }}</h2>

    {% for question in questions %}
        {% include 'testing/partials/question_item.html' %}
    {% endfor %}

    <button type="submit" class="mt-4 px-4 py-2 bg-green-600 text-white">
        Завершити тест
    </button>
</form>

<script>
function timer() {
    return {
        start: Date.now(),
        timeSpent(qid) {
            return Math.floor((Date.now() - this.start) / 1000);
        }
    }
}
</script>
{% endblock %}
```

templates/testing/partials/question_item.html — partial шаблон (HTMX-ready)

```
<div class="mb-6 p-4 border rounded">
    <p class="font-medium">{{ question.Text }}</p>

    {% for option in question.shuffled_options %}
        <label class="block mt-2">
            <input type="radio"
                   name="question_{{ question.id }}"
                   value="{{ option.id }}"
                   required>
            {{ option.Text }}
        </label>
    {% endfor %}

    <input type="hidden"
           name="time_{{ question.id }}"
           :value="timeSpent({{ question.id }})">
</div>
```

templates/analytics/test_analytics_detail.html — дашборд аналітики (HTMX)

```
{% extends 'base.html' %}

{% block title %}Аналітика тесту{% endblock %}

{% block content %}
<h2 class="text-lg font-semibold mb-4">Аналітичний звіт</h2>

<div class="mb-4">
    <p><strong>α Кронбаха:</strong> {{ test_analytics.CronbachAlpha
}}</p>
    <p><strong>Кількість спроб:</strong> {{ test_analytics.AttemptsCount
}}</p>
</div>

<table class="table-auto w-full border">
    <thead>
        <tr class="bg-gray-200">
            <th>Питання</th>
            <th>Складність</th>
            <th>Дискримінація</th>
        </tr>
    </thead>
    <tbody>
        {% for item in task_analytics %}
        <tr>
            <td>{{ item.QuestionID_id }}</td>
            <td>{{ item.DifficultyIndex|floatformat:2 }}</td>
            <td>{{ item.DiscriminationIndex|floatformat:2 }}</td>
        </tr>
        {% endfor %}
    </tbody>
</table>

<a href="{% url 'analytics:export_test_csv' test_analytics.TestID_id %}"
   class="inline-block mt-4 px-4 py-2 bg-blue-700 text-white">
    Експорт CSV
</a>
{% endblock %}
```

ДОДАТОК Ж.

Налаштування адміністративної панелі Django

users/admin.py — кастомізація UserAdmin

```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .models import User, Role

@admin.register(User)
class CustomUserAdmin(UserAdmin):
    """Кастомна адміністративна панель для користувачів системи"""

    model = User
    list_display = ('username', 'email', 'is_active', 'is_staff')
    list_filter = ('is_active', 'is_staff', 'roles')
    search_fields = ('username', 'email')

    fieldsets = (
        (None, {'fields': ('username', 'password')}),
        ('Персональні дані', {'fields': ('email',)}),
        ('Ролі користувача', {'fields': ('roles',)}),
        ('Права доступу', {'fields': ('is_active', 'is_staff',
'is_superuser')}),
    )

@admin.register(Role)
class RoleAdmin(admin.ModelAdmin):
    list_display = ('name',)
```

testing/admin.py — Inline-редагування тестів

```
from django.contrib import admin
from .models import Test, Question, QuestionOption


class QuestionOptionInline(admin.TabularInline):
    model = QuestionOption
    extra = 2

class QuestionInline(admin.TabularInline):
    model = Question
    extra = 1

@admin.register(Test)
class TestAdmin(admin.ModelAdmin):
    """Адміністрування тестів із вкладеними питаннями та дистракторами"""
```

```
    inlines = [QuestionInline]
    list_display = ('Title', 'IsActive')

@admin.register(Question)
class QuestionAdmin(admin.ModelAdmin):
    inlines = [QuestionOptionInline]
    list_display = ('Text', 'Points')
```

analytics/admin.py — перегляд аналітики (read-only)

```
from django.contrib import admin
from .models import TestAnalytics, TaskAnalytics, DistractorAnalytics

@admin.register(TestAnalytics)
class TestAnalyticsAdmin(admin.ModelAdmin):
    """Аналітика тестів доступна лише для перегляду"""

    readonly_fields = ('CronbachAlpha', 'AttemptsCount')
    list_display = ('TestID', 'CronbachAlpha', 'AttemptsCount')

@admin.register(TaskAnalytics)
class TaskAnalyticsAdmin(admin.ModelAdmin):
    readonly_fields = (
        'DifficultyIndex', 'DiscriminationIndex', 'PointBiserial'
    )
    list_display = ('QuestionID', 'DifficultyIndex',
'DiscriminationIndex')

@admin.register(DistractorAnalytics)
class DistractorAnalyticsAdmin(admin.ModelAdmin):
    readonly_fields = ('SelectionRate',)
    list_display = ('OptionID', 'SelectionRate')
```