Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра математичного моделювання та аналізу даних

До захисту допущено
Кафедрою ММ та АД протокол №___ від_____

завідувач кафедри _____ Володимир СТРУКОВ
(підпис) (ім'я, прізвище)

«___»_____ 2026 р.

Кваліфікаційна робота
здобувача першого (бакалаврського) рівня вищої освіти

«Проєктування та розробка програмного додатку для удосконалення банку тестових завдань навчального курсу "Бази даних"»
(назва роботи)

Спеціальність (спеціалізація) 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконавець _____ Єлизавета Шевченко
(підпис) *(ім'я, прізвище)*

Науковий керівник _____ Кирил Коробчинський
(підпис) *(ім'я, прізвище)*

Харків – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра математичного моделювання та аналізу даних

ЗАТВЕРДЖУЮ
Завідувач кафедри ММтаАД

_____ Володимир СТРУКОВ
підпис ім'я, прізвище

"____" _____2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувача першого (бакалаврського) рівня вищої освіти
(першого (бакалаврського) / другого (магістерського))

Шевченко Єлизавети Юріївни
(прізвище, ім'я, по батькові здобувача)

Спеціальність (спеціалізація) 122 Комп'ютерні науки; Інтелектуальні програмні системи і технології
(код та найменування спеціальності; спеціалізації спеціальності)

Освітня програма Комп'ютерні науки
(назва освітньої програми)

1. Тема роботи: «Проєктування та розробка програмного додатку для удосконалення банку тестових завдань навчального курсу "Бази даних"»

керівник роботи Коробчинський Кирил Петрович, к.т.н., доцент кафедри ММ та АД
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по Університету від 29.01.2026 року № 4101-5/225

2. Строк здобувачем роботи "____" _____2026 року

3. Вихідні дані до роботи: набір тестових завдань навчального курсу «Бази даних», результати тестування студентів, вимоги до програмного додатку та методи оцінювання якості тестових завдань.

4. Перелік питань, які потрібно розробити:

1) ознайомитись з особливостями організації банків тестових завдань, методами оцінювання якості тестів та психометричними підходами до аналізу результатів тестування;

2) провести огляд існуючих методів, математико-статистичних моделей та програмних засобів автоматизованого аналізу тестових завдань і результатів тестування;

3) познайомитися з алгоритмом роботи і методиками математико-статистичного та психометричного оцінювання якості тестових завдань, створити модель автоматизованої обробки, аналізу результатів тестування та виявлення некоректних або малоефективних завдань;

4) провести програмну реалізацію та експериментальне дослідження програмного додатку для удосконалення банку тестових завдань навчального курсу «Бази даних», реалізувати модулі імпорту даних, обчислення показників якості, візуалізації результатів і формування аналітичних звітів, а також оцінити ефективність розробленої системи.

5. План роботи

№ з/п	**Назви етапів роботи**	**Строк виконання етапів роботи**	**Примітка**
1	Формулювання мети та повного переліку завдань для виконання кваліфікаційної роботи (КР).	09.02.2026 – 14.02.2026	
2	Пошук та аналіз інформаційних джерел за темою КР.	15.02.2026 – 25.02.2026	
3	Дослідження математико-статистичних і психометричних методів аналізу тестових завдань та результатів тестування.	26.02.2026 – 02.03.2026	
4	Визначення особливостей організації банків тестових завдань, методів оцінювання їх якості та аналізу результатів тестування.	03.03.2026 – 08.03.2026	
5	Розробка математичної моделі та алгоритмів автоматизованого аналізу якості тестових завдань.	09.03.2026 – 25.03.2026	
6	Проєктування структури бази даних та архітектури програмного додатку.	26.03.2026 – 10.04.2026	
7	Програмна реалізація модулів обробки даних, обчислення показників якості та формування аналітичних звітів.	11.04.2026 – 10.05.2026	
8	Тестування програмного забезпечення та експериментальне дослідження на основі результатів тестування навчального курсу «Бази даних».	11.05.2026 – 16.05.2026	

9	Аналіз отриманих результатів та оцінювання ефективності реалізованих алгоритмів.	17.05.2026 – 25.05.2026	
10	Підсумок отриманих результатів та оформлення пакету документів для процедури захисту КР в екзаменаційній комісії ННІ КН та ШІ.	26.05.2026 – 31.05.2026	

6. Дата видачі завдання 09 лютого 2026 р.

Здобувач вищої освіти _____ Шевченко Є.Ю.
(підпис) (ініціали, прізвище)

Керівник роботи _____ Коробчинський К. П.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Основна частина роботи становить 93 сторінки. Робота містить 23 рисунки, 19 таблиць, 39 найменувань у списку використаних джерел та 5 додатків.

Метою роботи є проєктування та розробка програмного додатку для автоматизованого аналізу та оптимізації банку тестових завдань навчального курсу «Бази даних» шляхом формалізації критеріїв якості та застосування моделей класичної теорії тестів, Item Response Theory та теорії простору знань.

Методи дослідження: аналіз наукових джерел, математичне моделювання, статистичний аналіз, методи системного аналізу та програмної інженерії. Для реалізації використано Python, FastAPI, Microsoft SQL Server, React та спеціалізовані бібліотеки обробки даних і побудови звітів.

Результатом роботи є програмний додаток з трирівневою клієнт-серверною архітектурою, що автоматизує психометричний аналіз банку тестових завдань на основі показників CTT та параметрів 2PL-моделі IRT, моделює структуру знань студентів засобами KST та формує інтегральний показник якості Fi. Новизна полягає в інтеграції психометричного та когнітивного аналізу в єдиному інструменті з підтримкою AI-генерації замінних завдань із довідником верифікованих фактів предметної галузі.

Результати роботи рекомендовано використовувати у закладах вищої освіти, навчальних центрах та системах електронного тестування для об'єктивної оцінки якості банків тестових завдань, а також під час розробки та оптимізації тестів у професійній атестації та сертифікації.

КЛЮЧОВІ СЛОВА: ТЕСТУВАННЯ, БАНК ТЕСТОВИХ ЗАВДАНЬ, ITEM RESPONSE THEORY, IRT-МОДЕЛЬ, КЛАСИЧНА ТЕОРІЯ ТЕСТІВ, ТЕОРІЯ ПРОСТОРУ ЗНАНЬ, ОЦІНЮВАННЯ ЯКОСТІ, БАЗА ДАНИХ, ПРОГРАМНИЙ ДОДАТОК, АДАПТИВНЕ ТЕСТУВАННЯ.

ABSTRACT

The bachelor's qualification thesis report consists of an introduction, four chapters, conclusions, a list of references, and appendices. The main body of the thesis comprises 93 pages. The thesis contains 23 figures, 19 tables, 39 references, and 5 appendices.

The purpose of the thesis is to design and develop a software application for the automated analysis and optimization of a test item bank for the “Databases” course through the formalization of quality criteria and the application of Classical Test Theory (CTT), Item Response Theory (IRT), and Knowledge Space Theory (KST) models.

Research methods include the analysis of scientific literature, mathematical modeling, statistical analysis, systems analysis, and software engineering methods. The implementation is based on Python, FastAPI, Microsoft SQL Server, React, and specialized libraries for data processing and report generation.

The result of the thesis is a three-tier client-server software application that automates the psychometric analysis of a test item bank using CTT indicators and the parameters of the 2PL IRT model, models students’ knowledge structures using KST, and calculates an integrated quality index, Fi. The novelty of the work lies in the integration of psychometric and cognitive analysis within a single tool, supplemented by AI-based generation of replacement test items using a verified domain knowledge reference base.

The results of the thesis are recommended for use in higher education institutions, training centers, and electronic testing systems for the objective assessment of test item bank quality, as well as in the development and optimization of tests for professional assessment and certification.

KEYWORDS: TESTING, TEST ITEM BANK, ITEM RESPONSE THEORY, IRT MODEL, CLASSICAL TEST THEORY, KNOWLEDGE SPACE THEORY, QUALITY ASSESSMENT, DATABASE, SOFTWARE APPLICATION, ADAPTIVE TESTING.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ 11

ВСТУП 13

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ОЦІНЮВАННЯ ЯКОСТІ ТЕСТОВИХ ЗАВДАНЬ ТА АНАЛІЗ МЕТОДІВ МОДЕЛЮВАННЯ У КОНТРОЛІ ЗНАНЬ 16

1.1 Сутність тестування як методу контролю знань 16

1.2 Проблематика якості тестових завдань та критерії їх оцінювання 17

1.3 Класична теорія тестів (Classical Test Theory, CTT) та її обмеження 19

1.4 Моделі Item Response Theory (IRT) для оцінювання якості тестових завдань 22

1.5 Теорія простору знань (Knowledge Space Theory, KST) у моделюванні навчальних досягнень 26

1.6 Інтеграція психометричних (IRT) та когнітивних (KST) моделей у задачах удосконалення банку тестових завдань 29

1.7 Висновки за розділом 1 31

РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО АНАЛІЗУ БАНКУ ТЕСТОВИХ ЗАВДАНЬ 33

2.1 Аналіз вимог до програмного забезпечення 33

2.1.1 Мета та призначення системи 33

2.1.2 Функціональні та нефункціональні вимоги 34

2.1.3 Аналіз існуючих програмних рішень 39

2.2 Вибір архітектурного підходу та технологічного стеку 42

2.2.1 Вибір архітектурного підходу 42

2.2.2 Вибір технологічного стеку 45

2.3 Формалізація критеріїв оцінювання та відбору тестових завдань (на основі CTT, IRT та KST) 47

2.4 Розробка математичної моделі аналізу якості тестових завдань та простору знань студента (IRT + KST) 52

2.5 Проєктування моделі даних та структури бази даних 55

2.6 Висновки за розділом 2 61

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ДЛЯ АНАЛІЗУ ТА ОПТИМІЗАЦІЇ БАНКУ ТЕСТОВИХ ЗАВДАНЬ 62

3.1 Проєктування архітектури програмного додатку 62

3.2 Реалізація модулю збору та обробки результатів тестування 65

3.3 Реалізація математичного ядра для аналізу тестових завдань на основі моделей IRT та KST 69

3.4 Реалізація AI-модуля аналізу та генерації тестових завдань 74

3.5 Розробка інтерфейсу користувача та візуалізації результатів аналізу 78
3.6 Висновки за розділом 3 83
РОЗДІЛ 4 АПРОБАЦІЯ ПРОГРАМНОГО ДОДАТКУ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЙОГО РОБОТИ 85
4.1 Методологія тестування програмного забезпечення 85
4.2 Перевірка коректності реалізованих алгоритмів оцінювання 86
4.3 Експериментальна оцінка ефективності оптимізації банку тестових завдань 92
4.4 Оцінка зручності використання, інтерпретації результатів та якості AI-підтримки 98
4.5 Висновок до розділу 4 100
ВИСНОВКИ 102
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 105
ДОДАТОК А. Результати виконання автоматизованого тестового набору 110
ДОДАТОК Б. Програмний код математичного ядра 113
ДОДАТОК В. Програмний код AI-модуля та модуля даних 118
ДОДАТОК Г. Результати виконання автоматизованого тестового набору 119
ДОДАТОК Д. Скріншоти інтерфейсу системи 127

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

ACID – Atomicity, Consistency, Isolation, Durability

AI – Artificial Intelligence

API – Application Programming Interface

ASGI – Asynchronous Server Gateway Interface

CAT – Computerized Adaptive Testing

CORS – Cross-Origin Resource Sharing

CSV – Comma-Separated Values

CTT – Classical Test Theory

DAG – Directed Acyclic Graph

DCL – Data Control Language

DDL – Data Definition Language

DML – Data Manipulation Language

DOM – Document Object Model

EM – Expectation-Maximization

ER – Entity-Relationship

FK – Foreign Key

GEM – Generalized Expectation-Maximization

GIFT – General Import Format Template

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

ICC – Item Characteristic Curves

IEC – International Electrotechnical Commission

IEEE – Institute of Electrical and Electronics Engineers

IRT – Item Response Theory

ISO – International Organization for Standardization

JSON – JavaScript Object Notation

JWT – JSON Web Token

KST – Knowledge Space Theory

L-BFGS-B – Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Bounds

LMS – Learning Management System

MAP – Maximum A Posteriori

MCQ – Multiple Choice Questions

MLE – Maximum Likelihood Estimation

MSSQL – Microsoft SQL Server

ODBC – Open Database Connectivity

PDF – Portable Document Format

PK – Primary Key

REST – Representational State Transfer

SEM – Standard Error of Measurement

SPA – Single Page Application

XML – eXtensible Markup Language

НФ – Нефункціональні вимоги

НФБК – Нормальна форма Бойса-Кодда

СУБД – Система управління базами даних

Ф – Функціональні вимоги

ШІ – Штучний інтелект

ВСТУП

Розвиток сучасних освітніх технологій та широке впровадження електронного навчання зумовлюють зростання попиту на інструменти автоматизованого контролю знань. Тестування є одним із найпоширеніших методів оцінювання рівня засвоєння навчального матеріалу, однак його ефективність значною мірою залежить від якості тестових завдань. Некоректно складені або психометрично необґрунтовані завдання призводять до спотворення результатів оцінювання, зниження надійності вимірювань та унеможливлюють об'єктивну диференціацію студентів за рівнем підготовки.

У зв'язку з цим виникає потреба у програмних засобах, здатних не лише забезпечити проведення тестування, а й здійснювати глибокий аналіз банку тестових завдань із використанням сучасних математичних моделей.

Актуальність дослідження зумовлена обмеженістю традиційних підходів до оцінювання якості тестових завдань. Класична теорія тестів (CTT) залежить від конкретної вибірки студентів і не забезпечує коректного порівняння завдань між різними групами. Натомість моделі Item Response Theory (IRT) дають змогу отримувати більш об’єктивні оцінки складності та дискримінаційності завдань. Водночас існуючі засоби психометричного аналізу часто є складними у використанні та потребують спеціальної статистичної підготовки.

Особливої актуальності ця проблема набуває в умовах широкого використання інструментів штучного інтелекту, які можуть застосовуватися студентами для автоматизованого розв’язання тестових завдань без глибокого розуміння навчального матеріалу. Стандартні банки тестів, сформовані без психометричного обґрунтування, є вразливими до такого формального підходу до виконання завдань.

Аналіз існуючих програмних рішень, таких як Moodle, ALEKS, FastTest, Concerto та Xcalibre, свідчить про відсутність єдиного інструменту, який би поєднував психометричний аналіз на основі IRT, когнітивне моделювання

структури знань відповідно до Knowledge Space Theory (KST) та автоматичну генерацію замінних тестових завдань із використанням засобів штучного інтелекту. Наявні системи реалізують лише окремі з цих функцій і потребують значного доопрацювання для ефективного використання в навчальному процесі.

Це зумовлює необхідність розробки спеціалізованого програмного застосунку, який би автоматизував оцінювання якості тестових завдань, інтегрував сучасні психометричні та когнітивні моделі та забезпечував викладачам і методистам доступ до інтерпретованих результатів аналізу без потреби у спеціалізованій статистичній підготовці. Особливо актуальним є застосування такого підходу для навчального курсу «Бази даних», де якісна побудова тестового банку має суттєве значення для об’єктивного контролю знань студентів.

Об’єктом дослідження є процес оцінювання якості тестових завдань та побудова системи автоматизованого аналізу банку тестів.

Предметом дослідження є методи та моделі аналізу тестових завдань, зокрема моделі Item Response Theory та теорія простору знань, а також принципи проєктування та розробки програмного забезпечення для їх практичної реалізації.

Метою дослідження є проєктування та розробка програмного застосунку для удосконалення банку тестових завдань навчального курсу «Бази даних» шляхом формалізації критеріїв оцінювання та застосування інтегрованих моделей CTT, IRT та KST.

Для досягнення поставленої мети визначено такі завдання дослідження:

- проаналізувати стан проблеми управління банками тестових завдань та існуючі підходи до автоматизації контролю знань у курсі «Бази даних»;
- дослідити математичний апарат теорії відгуку на завдання (IRT) та теорії простору знань (KST);
- формалізувати критерії оцінювання та відбору тестових завдань;
- розробити архітектуру програмного застосунку та модель бази даних

на базі Microsoft SQL Server;

- реалізувати математичне ядро системи для розрахунку психометричних параметрів та побудови моделей знань;
- виконати експериментальну апробацію розробленого програмного забезпечення на реальному банку завдань.

Методи дослідження включають аналіз наукової літератури, методи математичного моделювання (CTT, IRT, KST), статистичні методи обробки даних, методи системного аналізу, методи програмної інженерії та експериментальну апробацію.

Наукова новизна роботи полягає у розробці інтегрованого підходу до оцінювання якості банку тестових завдань, що поєднує психометричний аналіз на основі двопараметричної моделі IRT з когнітивним моделюванням структури знань засобами KST та автоматичною генерацією замінних завдань із використанням великих мовних моделей із механізмом перевірки коректності.

Практична цінність роботи полягає у створенні програмного застосунку, який дозволяє автоматично діагностувати психометричні недоліки тестових завдань, формувати рекомендації щодо їх оптимізації та генерувати замінні завдання з заданими характеристиками. Результати експериментальної апробації на реальних даних підтвердили ефективність запропонованого підходу та доцільність його практичного використання для підвищення якості банків тестових завдань.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ОЦІНЮВАННЯ ЯКОСТІ ТЕСТОВИХ ЗАВДАНЬ ТА АНАЛІЗ МЕТОДІВ МОДЕЛЮВАННЯ У КОНТРОЛІ ЗНАНЬ

1.1 Сутність тестування як методу контролю знань

Тестування є науково обґрунтованим методом педагогічного вимірювання, що забезпечує кількісну та якісну оцінку рівня сформованості знань, умінь і навичок здобувачів освіти. На відміну від традиційних форм перевірки знань, тестовий контроль мінімізує суб'єктивний вплив оцінювача та забезпечує рівні умови для всіх учасників освітнього процесу [1]. Стандартизований характер тестування охоплює умови проведення контролю, регламентацію інструкцій, часові обмеження та процедури обробки результатів, що забезпечує відтворюваність і порівнюваність даних у різних групах і на різних етапах навчання [1,2].

Важливою характеристикою педагогічного тестування є критеріальна орієнтованість: результати інтерпретуються не лише шляхом порівняння студентів між собою, а й через їх відповідність наперед визначеним цілям навчального курсу та вимогам освітніх стандартів [2].Тестування виконує діагностичну, контролюючу та коригувальну функції: дозволяє визначати реальний рівень підготовки студентів, перевіряти відповідність результатів навчання вимогам програми та забезпечувати зворотний зв'язок для коригування навчального процесу. Ключовими показниками якості тесту при цьому є валідність — ступінь відповідності завдань поставленим навчальним цілям — та надійність, що характеризує стійкість результатів до впливу випадкових чинників [1].

Суттєвою перевагою тестування є його ефективність в умовах комп'ютеризованого та дистанційного навчання. Електронні системи тестування дозволяють автоматизувати оцінювання та накопичувати великі масиви даних про навчальні досягнення студентів, що створює аналітичну основу для подальшого вдосконалення як окремих тестових завдань, так і

навчального процесу загалом. Сучасні адаптивні підходи дозволяють динамічно змінювати складність завдань відповідно до поточного рівня підготовки студента, підвищуючи точність оцінювання в умовах масового тестування [3, 4].

Разом із тим тестування має обмеження: стандартизовані завдання не завжди дозволяють повною мірою оцінити творчі та комплексні аналітичні здібності, а формат вибору відповіді не виключає вгадування [6]. Ефективність тестування значною мірою визначається якістю окремих тестових завдань — некоректно сформульовані або методично необґрунтовані питання призводять до спотворення результатів і зниження надійності вимірювань [5, 7], що зумовлює необхідність систематичного аналізу психометричних характеристик завдань та автоматизації цього процесу. Саме розробка програмного інструменту, здатного діагностувати якість банку завдань і генерувати нові завдання на основі виявлених дефектів, є предметом цієї роботи; теоретичною основою такого інструменту слугують класична теорія тестів (CTT), моделі Item Response Theory (IRT) та теорія простору знань (KST), розгляду яких присвячені наступні підрозділи.

1.2 Проблематика якості тестових завдань та критерії їх оцінювання

Якість тестових завдань є одним із ключових чинників достовірності та об'єктивності педагогічного оцінювання. Використання низькоякісних завдань спотворює результати тестування та ускладнює диференціацію студентів за рівнем підготовки. Особливо важливим це є для адаптивного тестування, ефективність якого залежить від наявності банку завдань зі стабільними психометричними характеристиками [8].

Проблеми якості тестових завдань особливо проявляються у випадках, коли завдання є неоднозначними, не відповідають змісту навчальної програми або не узгоджені з цілями курсу. Такі завдання можуть оцінювати не рівень засвоєння знань, а випадкові чинники, зокрема вміння вгадувати відповіді або інтерпретувати нечіткі формулювання. Якщо завдання містить декілька задач

або проблем, студент може не дати повної відповіді, що призводить до зниження об'єктивності та валідності оцінювання [9].

Одним із основних підходів до оцінювання якості тестових завдань є аналіз їх психометричних характеристик, що дозволяє перетворити звичайний набір питань на науково верифікований інструмент вимірювання. До ключових показників належать складність, дискримінативність та надійність тестових завдань, які більш детально будуть розглянуті в межах класичної теорії тестів у підрозділі 1.3. Оцінювання зазначених характеристик може здійснюватися постфактум на основі результатів проходження тестів із використанням методів класичної теорії тестів або моделей Item Response Theory. Застосування таких підходів дозволяє виявляти завдання з надмірною або недостатньою складністю, низькою дискримінативністю, а також ті, що негативно впливають на загальну надійність тесту. Посттестовий психометричний аналіз є ефективним інструментом удосконалення банку тестових завдань без необхідності зміни процедури самого тестування.

Окрім психометричних характеристик, важливим аспектом якості тестових завдань є їх логічна та предметна структура, яка фіксується у специфікації тесту або матриці змісту. Тестові завдання мають відповідати ієрархії тем навчального курсу, забезпечувати послідовний перехід від базових понять до складніших і охоплювати ключові елементи змісту дисципліни. Для аналізу таких структурних зв’язків доцільним є використання теорії простору знань (Knowledge Space Theory, KST), відповідно до якої засвоєння складніших понять можливе лише за умови опанування необхідних передумов (prerequisites). Це дає змогу оптимізувати структуру банку тестових завдань, усунути надмірні або дублюючі питання та забезпечити логічну узгодженість між змістом навчального курсу і системою контролю знань [12].

Особливої актуальності проблематика якості тестових завдань набуває в умовах масового та дистанційного тестування, де через відсутність безпосереднього спілкування викладача та студента необхідно оперативно та в індивідуальному режимі контролювати навчальні досягнення. У таких

форматах підвищується ризик випадкового вибору правильної відповіді, загострюється проблема нерівномірної складності завдань та обмежується можливість оцінювання комплексних і творчих навичок здобувачів освіти. За відсутності глибокого аналізу завдань результати масового тестування можуть не відображати реальний рівень підготовки студентів.

Сучасні підходи до вдосконалення банків тестових завдань базуються на використанні технологій штучного інтелекту для генерації завдань, аналізу результатів тестування та формування рекомендацій щодо оптимізації складу банку тестів. Це дозволяє скоротити часові витрати на підтримку банку завдань без втрати якості оцінювання [12,13].

1.3 Класична теорія тестів (Classical Test Theory, CTT) та її обмеження

Щоб зрозуміти обмеження сучасних адаптивних та інтелектуальних систем контролю знань, доцільно звернутися до фундаментальних психометричних моделей, зокрема, до класичної теорії тестування (CTT). CTT – це фундаментальна психометрична модель, що використовується для оцінки якості тестових завдань та результатів тестування. Ця теорія описує зв'язок між спостережуваними балами та справжнім рівнем знань студента, трактуючи спостережуваний бал як суму справжнього рівня знань та випадкової похибки вимірювання. Це можна виразити формулою

$$X = T + E, \tag{1.1}$$

де X – спостережуваний бал;

T – істинний бал (істинний показник вимірюваної навчальної характеристики);

E – випадкова помилка [14].

Розмежування понять істинного та спостережуваного балів дозволяє оцінити, якою мірою результати тестування відображають фактичний рівень знань студента, а якою – спотворюються випадковими чинниками, зокрема неуважністю, вгадуванням відповідей або технічними обмеженнями систем

дистанційного навчання.

У межах CTT здійснюється аналіз ключових психометричних характеристик тестових завдань. Складність завдання (item difficulty, P-value) визначається як частка правильних відповідей у разі використання дихотомічних завдань або як відношення середнього набраного бала до максимально можливого у разі застосування політомічних шкал оцінювання. Завдання з надмірно високими або надмірно низькими значеннями цього показника втрачають здатність ефективно диференціювати рівень навчальних досягнень здобувачів освіти. Дискримінативність (item discrimination, D-index) характеризує здатність тестового завдання розрізняти студентів із різним рівнем знань; значення понад 0,40 зазвичай інтерпретуються як показник ефективного розділення «сильних» і «слабких» учасників. Надійність тесту (reliability) відображає внутрішню узгодженість завдань та стійкість результатів до випадкових факторів і оцінюється за допомогою коефіцієнтів Альфа Кронбаха або Омега Макдональда. Важливим є також чітке формулювання завдань, оскільки двозначність знижує достовірність оцінювання та стимулює вгадування [10].

Наочну інтерпретацію показників складності та дискримінаційної здатності тестових завдань подано на рисунку 1.3.1, де представлено графік аналізу елементів тесту за значеннями P-value та кореляцією з підсумковим балом. Візуалізація дозволяє швидко ідентифікувати надто легкі ($P \to 1$) і надто складні ($P \to 0$) завдання, які мають низьку інформативність, тоді як завдання із середніми значеннями (орієнтовно 0,6–0,8) забезпечують найкращу диференціацію рівнів підготовки студентів. Кольорове маркування додатково сигналізує про елементи, що відповідають вимогам якості або потребують перегляду та доопрацювання. Це забезпечує підвищення якості тестового інструментарію та обґрунтованість подальших рішень щодо його вдосконалення.

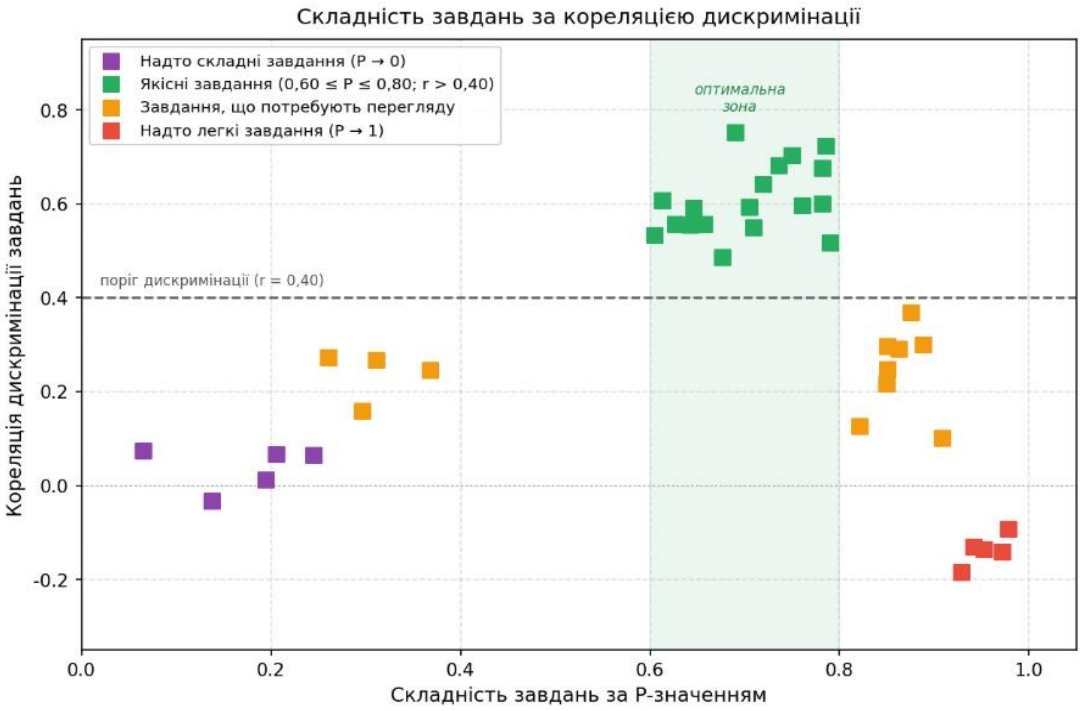


Рисунок 1.3.1 – Розподіл тестових завдань за показником складності P-value та кореляцією з підсумковим балом

У комп'ютерних системах контролю знань, зокрема в системах управління навчанням (LMS), CTT широко використовується для формування статистики проходження тестів, оцінювання ефективності завдань та підтримки аналітики банку тестів.

Однак, CTT має значні обмеження, які визначають межі її застосування у сучасних цифрових освітніх системах. Найголовнішим серед цих обмежень є той факт, що психометричні характеристики тестових завдань часто залежать від конкретних вибірок студентів — показники складності та дискримінації певного завдання, як правило, змінюються залежно від конкретної групи учасників тестування. Орієнтація CTT на тест у цілому, а не на окремі завдання, практично унеможливлює побудову адаптивних тестів, у яких складність динамічно підлаштовується під індивідуальний рівень підготовки студента. Крім того, модель не враховує додаткові параметри сучасних інтелектуальних систем — час відповіді, послідовність спроб, взаємозв'язки

між завданнями та динаміку засвоєння знань [14].

Водночас CTT зберігає практичну цінність як базова модель психометричного оцінювання і залишається необхідним фундаментом для аналізу якості тестових завдань. Її обмеження стимулюють інтеграцію сучасних методів: Item Response Theory (IRT), що оцінює завдання незалежно від вибірки студентів; Knowledge Space Theory (KST), яка застосовується для моделювання логічної структури навчального матеріалу та побудови індивідуальних траєкторій навчання; а також алгоритмів на основі штучного інтелекту та Data Mining, що забезпечують генерацію нових завдань і оптимізацію банку тестів. Розгляду цих підходів присвячені наступні підрозділи.

1.4 Моделі Item Response Theory (IRT) для оцінювання якості тестових завдань

Для подолання обмежень класичної теорії тестів (CTT) та підвищення точності й об’єктивності оцінювання навчальних досягнень у сучасних цифрових освітніх середовищах широко застосовуються моделі Item Response Theory (IRT). IRT є розвиненим математико-статистичним апаратом психометрії, який дозволяє аналізувати якість тестових завдань та результати тестування на основі формалізованого взаємозв’язку між латентним рівнем підготовки студента та параметрами конкретного завдання [24].

Ключовою ідеєю IRT є припущення, що ймовірність правильної відповіді на завдання визначається не лише кількістю набраних балів, а є функцією латентної змінної θ, яка інтерпретується як реальний рівень знань або здібностей студента, а також набору параметрів тестового завдання. Такий підхід дозволяє перейти від суто емпіричного оцінювання до формалізованого математичного моделювання навчальних результатів.

У межах класичних IRT-моделей кожне тестове завдання описується такими основними параметрами:

- складність завдання (b) — визначає рівень здібностей θ, за якого ймовірність правильної відповіді становить приблизно 50%; чим більше значення параметра b, тим складнішим є завдання;
- дискримінативність (a) — характеризує здатність завдання розрізняти студентів з різним рівнем підготовки; високі значення параметра свідчать про ефективне диференціювання знань;
- параметр вгадування (c) — враховує ймовірність випадкової правильної відповіді [16].

Залежно від кількості параметрів розрізняють кілька базових моделей IRT, наведених на рисунках 1.4.2-1.4.4: однопараметричну модель Раша (1PL), яка враховує лише складність завдання; двопараметричну модель (2PL), що додатково враховує дискримінацію; та трипараметричну модель (3PL), яка також включає параметр вгадування. 3PL-модель має високу гнучкість та придатна для реалізації в автоматизованих системах оцінювання знань [16].

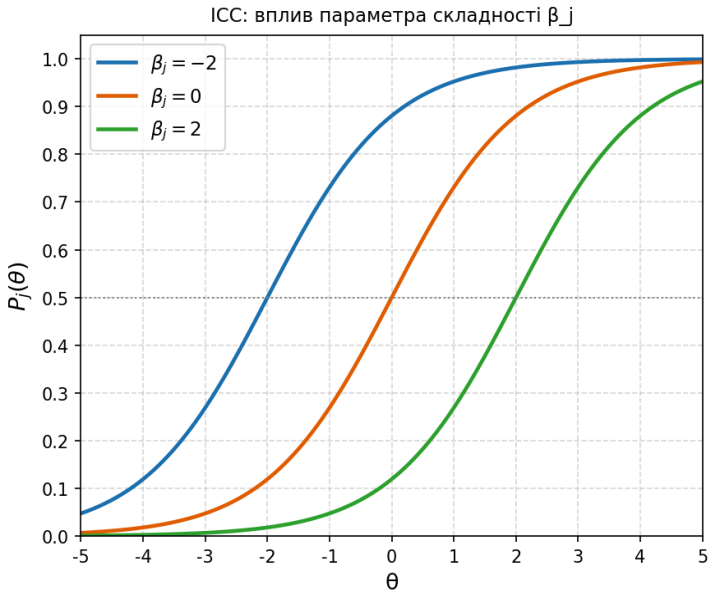


Рисунок 1.4.2 – Характеристичні криві завдань у однопараметричній моделі IRT (1PL)

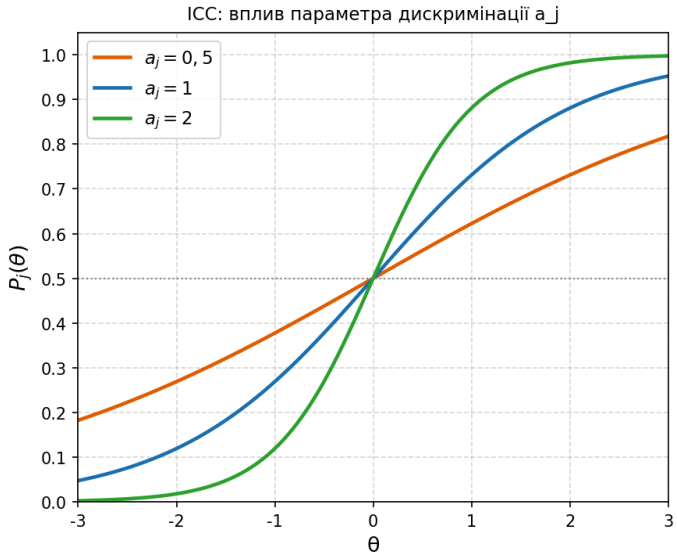


Рисунок 1.4.3 – Характеристичні криві завдань з різною дискримінаційною здатністю у моделі 2PL

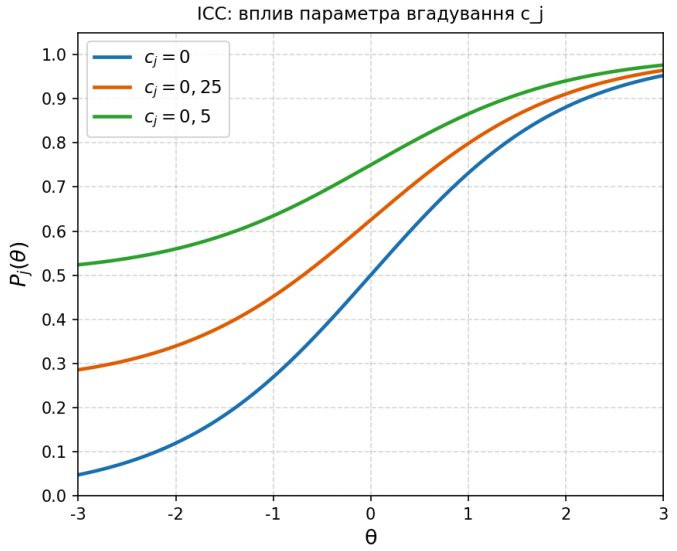


Рисунок 1.4.4 – Вплив параметра вгадування c на характеристичні криві завдань у моделі 3PL

Суттєвою перевагою IRT порівняно з CTT є інваріантність параметрів завдань, тобто їх незалежність від конкретної вибірки студентів. Це дозволяє порівнювати результати різних тестів, проводити повторне використання завдань, удосконалювати банк тестових одиниць і забезпечувати об’єктивне оцінювання в масштабних системах дистанційного навчання.

З практичної точки зору IRT є не лише теоретичною моделлю, а й ефективним інструментом для реалізації програмного забезпечення у складі систем управління навчанням (LMS) [16]. На основі IRT можливе впровадження комп’ютеризованого адаптивного тестування (Computerized Adaptive Testing, CAT), у межах якого складність наступного завдання автоматично коригується відповідно до поточної оцінки θ. Наприклад, після кожної відповіді студента система оновлює значення латентної змінної та обирає завдання з оптимальним рівнем складності, що дозволяє зменшити загальну кількість питань без втрати точності оцінювання.

IRT також забезпечує можливість побудови динамічних профілів знань студентів, прогнозування майбутніх результатів, виявлення зон ризику та рекомендацій щодо подальшого навчання. У контексті комп’ютерних наук це передбачає використання алгоритмів максимізації правдоподібності (MLE) або байєсівських методів для оцінювання параметра θ, а також ефективне масштабування обчислень у разі обробки великих вибірок [23].

Важливим аспектом практичного застосування IRT є аналіз банку тестових завдань, який дозволяє автоматично виявляти завдання з низькою дискримінативністю, надмірною або недостатньою складністю, а також оптимізувати структуру тестів [8]. У сучасних цифрових системах додатково реалізується візуалізація кривих характеристик завдань (Item Characteristic Curves, ICC), що підвищує інтерпретованість результатів для викладачів і адміністраторів курсів.

Разом із тим, використання IRT також має певні обмеження, зокрема потребу у достатньо великих вибірках для надійного оцінювання параметрів та відносну складність реалізації. Проте ці недоліки компенсуються високою точністю, масштабованістю та можливістю інтеграції з технологіями Big Data та штучного інтелекту. Параметри IRT можуть слугувати вхідними даними для інтелектуальних тьюторських систем, алгоритмів машинного навчання та систем рекомендацій, що забезпечує персоналізований підхід до навчання [16].

Отже, моделі Item Response Theory є теоретичною та алгоритмічною основою для створення сучасних автоматизованих систем оцінювання знань, що створює підґрунтя для подальшої інтеграції з когнітивними моделями, зокрема теорією простору знань (Knowledge Space Theory).

1.5 Теорія простору знань (Knowledge Space Theory, KST) у моделюванні навчальних досягнень

Розвиток сучасних систем автоматизованого оцінювання знань потребує переходу від суто кількісних показників до структурного аналізу навчальних досягнень студентів. Теорія простору знань (Knowledge Space Theory, KST) забезпечує формалізований опис структури навчальних досягнень, моделюючи процес навчання з урахуванням логічних залежностей між окремими темами, поняттями та навичками, що робить її ключовим інструментом для адаптивного навчання, інтелектуальних навчальних систем та побудови персоналізованих освітніх траєкторій у цифрових середовищах [17, 18].

Фундаментом KST є формалізація навчального матеріалу у вигляді скінченного набору елементарних одиниць знань, що дозволяє перейти від інтуїтивного уявлення про процес навчання до формалізованої моделі, придатної для використання в автоматизованих системах контролю та підтримки навчання. Множина знань (Knowledge Domain) визначається як скінченна множина:

$$Q=\{q_1,q_2,\ldots,q_n\}, \tag{1.2}$$

де Q — множина знань курсу;

q_i — окрема навчальна одиниця (поняття, тема, навичка або елемент навчального курсу);

n — загальна кількість елементів знань.

Рівень деталізації цієї множини визначається цілями аналізу та особливостями конкретної освітньої системи, що забезпечує гнучкість і масштабованість моделі [17].

Стан знань (Knowledge State) описується як підмножина: K⊆Q, яка відображає набір елементів знань, засвоєних студентом на певному етапі навчання. На відміну від традиційного оцінювання у вигляді одного числового показника, стан знань забезпечує структурний опис навчальних досягнень і дозволяє аналізувати характер засвоєння матеріалу.

Сукупність усіх допустимих станів знань формує простір знань (Knowledge Space), який є підмножиною булевої множини K ⊆ 2^Q. Принциповим положенням KST є те, що простір знань є обмеженим: не кожна комбінація елементів знань вважається когнітивно можливою або педагогічно доцільною [17]. Це пояснюється існуванням передумовних відношень (Prerequisite Relations) між окремими елементами навчального матеріалу. Якщо певне знання логічно або методично залежить від попереднього опанування іншого, то жоден допустимий стан знань не може містити залежний елемент без його передумови. Такі відношення формалізують логіку навчального процесу та відображають ієрархію засвоєння матеріалу [22].

З математичної точки зору, передумовні залежності задають частковий порядок на множині Q, який може бути представлений у вигляді орієнтованого ациклічного графа, який представлено на рисунку 1.5.5, де вершини $q_1 \ldots q_9$ відповідають елементам знань, а дуги відображають передумовні залежності між ними. Сині дуги відображають безпосередні передумовні залежності, тоді як оранжеві– транзитивні зв’язки, що випливають із властивостей часткового порядку. Це дозволяє застосовувати методи теорії графів для аналізу структури курсу, пошуку критичних шляхів навчання та оптимізації послідовності подання матеріалу [19].

Крім жорстких передумовних залежностей, простір знань може враховувати альтернативні шляхи засвоєння навчального матеріалу, за яких одна й та сама тема може бути досягнута через різні підмножини попередніх знань. Це дозволяє наблизити модель до реального процесу навчання та застосовувати KST у персоналізованих освітніх середовищах [24].

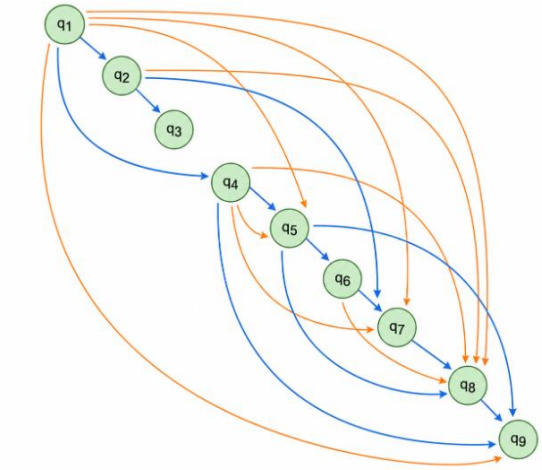


Рисунок 1.5.5 – Представлення передумовних залежностей між елементами знань у вигляді орієнтованого ациклічного графа (DAG)

Одним із ключових напрямів інтеграції KST з методами штучного інтелекту є автоматичне виявлення передумовних залежностей, формування рекомендацій на основі стану знань студента та адаптація освітніх маршрутів. Такий підхід зменшує залежність від ручної експертної розмітки та, у поєднанні з IRT, створює основу для інтелектуальних систем оцінювання [20].

Теорія простору знань забезпечує структурований профіль знань студента, що дозволяє не лише оцінювати рівень підготовки, а й виявляти прогалини, когнітивні розриви та помилкові уявлення. Водночас застосування KST має обмеження: формування передумовних відношень традиційно потребує експертної розмітки, а зі збільшенням обсягу матеріалу ускладнюється обчислення найбільш імовірного стану знань студента та актуалізація структури залежностей [21].

У сучасних освітніх системах KST застосовується для побудови структурованих моделей курсів у LMS, MOOC та корпоративному навчанні, що забезпечує автоматизацію формування індивідуальних освітніх маршрутів та адаптацію складності матеріалу до рівня знань користувача, підвищуючи ефективність навчання та оптимізуючи розвиток професійних компетентностей.

1.6 Інтеграція психометричних (IRT) та когнітивних (KST) моделей у задачах удосконалення банку тестових завдань

Банк тестових завдань є ключовим елементом сучасних систем оцінювання знань, оскільки його структура та якість безпосередньо впливають на точність діагностики рівня підготовки студентів, педагогічну цінність тестування та ефективність навчального процесу. Традиційні підходи до формування банків завдань, орієнтовані переважно на тематичне покриття курсу, часто призводять до дублювання завдань, нерівномірного представлення навчальних тем і відсутності перевірки передумовних знань [24]. Для подолання цих обмежень доцільним є інтегроване використання психометричних моделей Item Response Theory (IRT) та когнітивних моделей Knowledge Space Theory (KST).

Модель IRT забезпечує кількісну оцінку якості тестових завдань на основі психометричних параметрів — складності, дискримінаційної здатності та ймовірності вгадування [24]. Разом із тим психометричний аналіз не враховує логічну структуру навчального матеріалу, тому доповнюється когнітивною моделлю KST, яка перевіряє повноту покриття тем та дотримання передумовних залежностей [25]. Порівняльну характеристику двох моделей подано у таблиці 1.6.1.

Таблиця 1.6.1 – Порівняльна характеристика моделей IRT та KST у контексті оптимізації банку тестових завдань

Характеристика	**IRT**	**KST**
Тип аналізу	Психометричний (кількісний)	Когнітивний (структурний)
Одиниця аналізу	Окреме завдання	Елемент знань
Ключові параметри	Складність (b), дискримінація (a), вгадування (c)	Передумовні залежності, стан знань

Закінчення таблиці 1.6.1

Результат аналізу	Оцінка якості завдання	Профіль знань студента
Обмеження	Не враховує структуру матеріалу	Потребує експертної розмітки
Внесок в оптимізацію банку	Виявлення слабких завдань	Забезпечення логічної повноти покриття концептів та виявлення порушень передумовних залежностей

Практичним результатом інтеграції трьох підходів — CTT, IRT та KST — є інтегральний показник якості тестового завдання Fi, що обчислюється як зважена сума нормалізованих компонент:

$$Fi = 0{,}40 \cdot CTT + 0{,}30 \cdot IRT + 0{,}30 \cdot KST \quad (1.3)$$

де CTT-компонента відображає складність та дискримінаційну здатність завдання за класичною теорією тестів;

IRT-компонента — психометричну якість за двопараметричною моделлю;

KST-компонента — структурну коректність завдання у графі передумов.

Вагові коефіцієнти визначені з урахуванням відносної стабільності методів при малих вибірках: CTT отримує найбільшу вагу як найбільш стійкий при $n < 100$, IRT та KST — рівні ваги як взаємодоповнюючі психометричний і когнітивний виміри. Значення $Fi \in [0; 1]$ є основою для класифікації завдання за трьома статусами: залишити, переглянути або видалити. На рисунку 1.6.6 представлена концептуальна схема інтеграції CTT, IRT та KST в інтегральний показник якості Fi.

Інтеграція IRT та KST дозволяє одночасно усунути надлишковість завдань, виявити прогалини у викладі предметів та забезпечити всебічне охоплення простору знань курсу [23, 25, 26]. Практичне застосування у межах курсу «Бази даних» дає змогу формувати тестові набори з поступовим нарощуванням складності — від базових понять нормалізації до складних

SQL-запитів та управління транзакціями, що створює передумови для реалізації адаптивного тестування. Формалізація цього підходу у вигляді конкретних обчислюваних критеріїв та алгоритмів прийняття рішень розглядається у розділі 2.

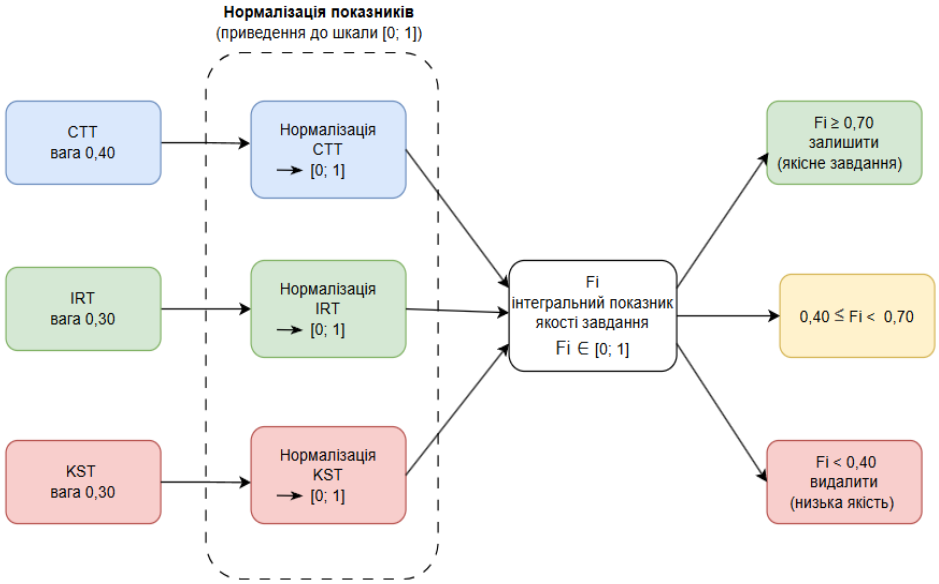


Рисунок 1.6.6 – Концептуальна схема інтеграції CTT, IRT та KST в інтегральний показник якості Fi

1.7 Висновки за розділом 1

Аналіз теоретичних основ підтвердив, що ефективність тестового контролю визначається не лише загальною структурою тесту, а й психометричною та методичною коректністю окремих завдань банку. Використання низькоякісних або логічно неузгоджених завдань спотворює результати оцінювання та знижує його педагогічну цінність.

Класична теорія тестів забезпечує базові інструменти оцінювання надійності та складності завдань, проте її залежність параметрів від вибірки обмежує застосування в адаптивних системах. Моделі IRT усувають цей недолік, дозволяючи оцінювати складність, дискримінацію та ймовірність вгадування кожного завдання незалежно від складу тестованих, що створює надійну основу для формування адаптивних тестів.

Теорія простору знань доповнює психометричний підхід когнітивним виміром: замість окремих статистичних показників KST описує структуру навчального матеріалу через передумовні залежності між темами, що дозволяє виявляти прогалини у покритті курсу та забезпечувати логічну послідовність перевірки знань.

Поєднання CTT, IRT та KST утворює комплексну основу для автоматизованого аналізу банку тестових завдань: CTT забезпечує базові статистичні показники якості, IRT — точну оцінку параметрів завдань незалежно від вибірки, KST — логічну узгодженість банку із когнітивною структурою курсу. Саме цей інтегрований підхід є теоретичним підґрунтям для проєктування системи, розглянутого у розділі 2.

РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО АНАЛІЗУ БАНКУ ТЕСТОВИХ ЗАВДАНЬ

2.1 Аналіз вимог до програмного забезпечення

2.1.1 Мета та призначення системи

З урахуванням обмежень традиційних підходів до формування банку тестових завдань, розглянутих у розділі 1, зростання обсягів тестових матеріалів та необхідності забезпечення об'єктивності й достовірності оцінювання результатів навчання виникає потреба у спеціалізованому програмному інструменті, здатному автоматизувати психометричний і когнітивний аналіз тестових елементів та підтримувати прийняття обґрунтованих рішень щодо їх оптимізації.

Метою розроблюваної системи є проєктування та реалізація програмного інструменту автоматизованого аналізу банку тестових завдань, який забезпечує комплексну оцінку їх якості, структури та педагогічної доцільності на основі сучасних психометричних і когнітивних підходів. Система автоматизує обчислення ключових статистичних показників, виявляє неефективні або надлишкові завдання, контролює логічну узгодженість навчального матеріалу та підтримує прийняття обґрунтованих рішень щодо оптимізації банку тестових завдань.

Для досягнення поставленої мети використовується поєднання психометричних методів (CTT, IRT) та когнітивних підходів (KST), детально розглянутих у розділі 1, що забезпечує одночасне врахування кількісних характеристик завдань і їх логічного місця у структурі курсу.

Окремої актуальності розробка такої системи набуває в умовах широкого поширення інструментів штучного інтелекту, які студенти можуть використовувати для автоматизованого розв'язання тестових завдань без реального засвоєння матеріалу. Подібні практики знижують достовірність оцінювання та спотворюють результати навчального контролю.

Використання психометричного та когнітивного аналізу дозволяє формувати завдання, що перевіряють глибоке розуміння змісту, логічні зв'язки між темами та застосування знань у нових ситуаціях, а не просте відтворення шаблонних відповідей. Так система сприятиме підвищенню стійкості тестів до формального або автоматизованого виконання та забезпечуватиме більш об'єктивну перевірку реальних навчальних досягнень студентів.

Узагальнюючи викладене, призначення розроблюваної системи полягає в автоматизації аналізу та оптимізації банку тестових завдань, підвищенні точності й надійності оцінювання, забезпеченні когнітивної узгодженості навчального матеріалу та створенні умов для більш ефективного освітнього процесу. Система також інтегрує модуль генерації тестових завдань на основі великих мовних моделей, що дозволяє оперативно поповнювати банк якісними завданнями різних типів і рівнів складності відповідно до таксономії Блума — від завдань на знання та розуміння до завдань на аналіз, синтез і оцінювання [27]. Визначені цілі слугують основою для формування вимог до функціональних можливостей і архітектури програмного забезпечення.

2.1.2 Функціональні та нефункціональні вимоги

Визначення вимог до програмного забезпечення є ключовим етапом проєктування системи аналізу банку тестових завдань, оскільки саме на цьому етапі формується формалізоване уявлення про функції, обмеження та якісні характеристики майбутнього програмного продукту. Коректна специфікація вимог забезпечує узгодженість між педагогічними потребами, психометричними моделями оцінювання та технічними можливостями системи, а також мінімізує ризики помилок під час реалізації та впровадження. Процес формування вимог здійснюється відповідно до принципів інженерії вимог, викладених у стандарті ISO/IEC/IEEE 29148 [28], який передбачає поділ вимог на функціональні, що описують поведінку системи, та нефункціональні, які характеризують її якість, обмеження й умови

експлуатації. Оцінювання якісних характеристик програмного забезпечення доцільно здійснювати з урахуванням моделі якості ISO/IEC 25010 [29].

Функціональні вимоги розроблюваної системи безпосередньо випливають зі специфіки предметної області — освітнього тестування та психометричного аналізу. Насамперед система повинна забезпечувати централізоване керування банком тестових завдань, оскільки ефективний аналіз неможливий без структурованого зберігання елементів тесту, їх метаданих і результатів використання. Тому виникає потреба в єдиному сховищі, яке підтримує імпорт, редагування, класифікацію та версіонування завдань, а також дозволяє відстежувати історію їх використання. Наявність такої функціональності забезпечує цілісність даних і створює основу для подальшого кількісного аналізу.

Другою принциповою вимогою є автоматизований збір і накопичення результатів тестування. Психометричні показники можуть бути розраховані лише на основі достатньо великих вибірок відповідей студентів, тому система повинна зберігати детальну інформацію про кожну спробу проходження тесту, включаючи обрані варіанти, кількість балів, час виконання та інші параметри поведінки користувача. Такий підхід забезпечує статистичну достовірність оцінювання, дозволяє проводити ретроспективний аналіз і формує базу для побудови моделей складності завдань.

Ключовою функціональною складовою є реалізація психометричного аналізу завдань із використанням методів класичної теорії тестів та моделей Item Response Theory. Автоматичне обчислення відповідних показників усуває суб'єктивність експертних оцінок і забезпечує об'єктивну інтерпретацію ефективності завдань. Крім того, можливість виявлення надто легких, надто складних або низькодискримінативних питань дозволяє своєчасно вилучати або коригувати неякісні елементи, що підвищує точність вимірювання навчальних досягнень.

Поряд із кількісним психометричним аналізом система повинна підтримувати когнітивне моделювання структури знань на основі підходів

Knowledge Space Theory. Без урахування передумовних залежностей між темами тест може перевіряти фрагментарні знання та не відображати реальний стан підготовки студента. Тому система повинна забезпечувати опис передумовних зв'язків між темами, аналіз покриття змісту курсу та виявлення прогалин або дублювання в банку завдань.

Генерація тестових завдань на основі великих мовних моделей є ще однією функціональною вимогою. Це дозволяє ефективно поповнювати банк завдань різними типами та рівнями складності, зокрема у випадках, коли аналіз виявляє прогалини у покритті навчального матеріалу або низьку якість наявних елементів.

Крім аналітичних функцій, система повинна забезпечувати зручні засоби візуалізації та формування звітів, оскільки результати аналізу мають бути зрозумілими для викладачів і адміністраторів, які не завжди володіють спеціальними статистичними знаннями. Подання інформації у вигляді графіків, діаграм і рекомендацій спрощує прийняття рішень щодо оптимізації банку завдань.

Поряд із функціональними можливостями важливу роль відіграють нефункціональні вимоги, які визначають якість і надійність програмного забезпечення. З огляду на використання системи в навчальних закладах із великою кількістю користувачів, вона повинна забезпечувати достатню продуктивність і масштабованість, що дозволить обробляти значні обсяги статистичних даних без затримок.

Не менш важливою є вимога надійності та збереження даних, оскільки результати тестування мають офіційний характер і можуть впливати на академічні рішення. Система повинна гарантувати цілісність інформації, підтримувати резервне копіювання та захист від збоїв. Враховуючи роботу з персональною інформацією студентів, необхідно також забезпечити високий рівень безпеки, включаючи автентифікацію користувачів, контроль доступу та захист від несанкціонованого використання.

Важливою характеристикою є зручність використання, адже складні або перевантажені інтерфейси можуть стати перешкодою для впровадження системи. Інтуїтивно зрозумілий інтерфейс і наочне представлення результатів знижують поріг входження для викладачів і підвищують ефективність їхньої роботи.

Нарешті, програмне забезпечення повинно бути підтримуваним і розширюваним, оскільки методи психометричного аналізу постійно розвиваються, а вимоги освітнього середовища змінюються. Модульна архітектура та можливість додавання нових алгоритмів аналізу, нових типів завдань і нових форматів звітів дозволять адаптувати систему до майбутніх потреб без суттєвих витрат на переробку. Сукупність визначених вимог наведено у таблиці 2.1.2.1

Таблиця 2.1.2.1 — Функціональні та нефункціональні вимоги до системи

№	Вимога	Тип	Пріоритет
Ф1	Керування банком завдань	Функціональна	Високий
Ф2	Збір результатів тестування	Функціональна	Високий
Ф3	Психометричний аналіз CTT/IRT	Функціональна	Високий
Ф4	Когнітивне моделювання KST	Функціональна	Високий
Ф5	Генерація завдань на основі LLM	Функціональна	Середній
Ф6	Візуалізація та звіти	Функціональна	Середній
НФ1	Продуктивність	Нефункціональна	Високий
НФ2	Надійність та збереження даних	Нефункціональна	Високий

Закінчення таблиці 2.1.2.1

НФ3	Безпека	Нефункціональна	Високий
НФ4	Зручність використання	Нефункціональна	Середній
НФ5	Розширюваність	Нефункціональна	Середній

Після таблиці вимог розміщується діаграма використання (рисунок 2.1.1.1), яка наочно відображає взаємодію між акторами системи та її функціональними можливостями.

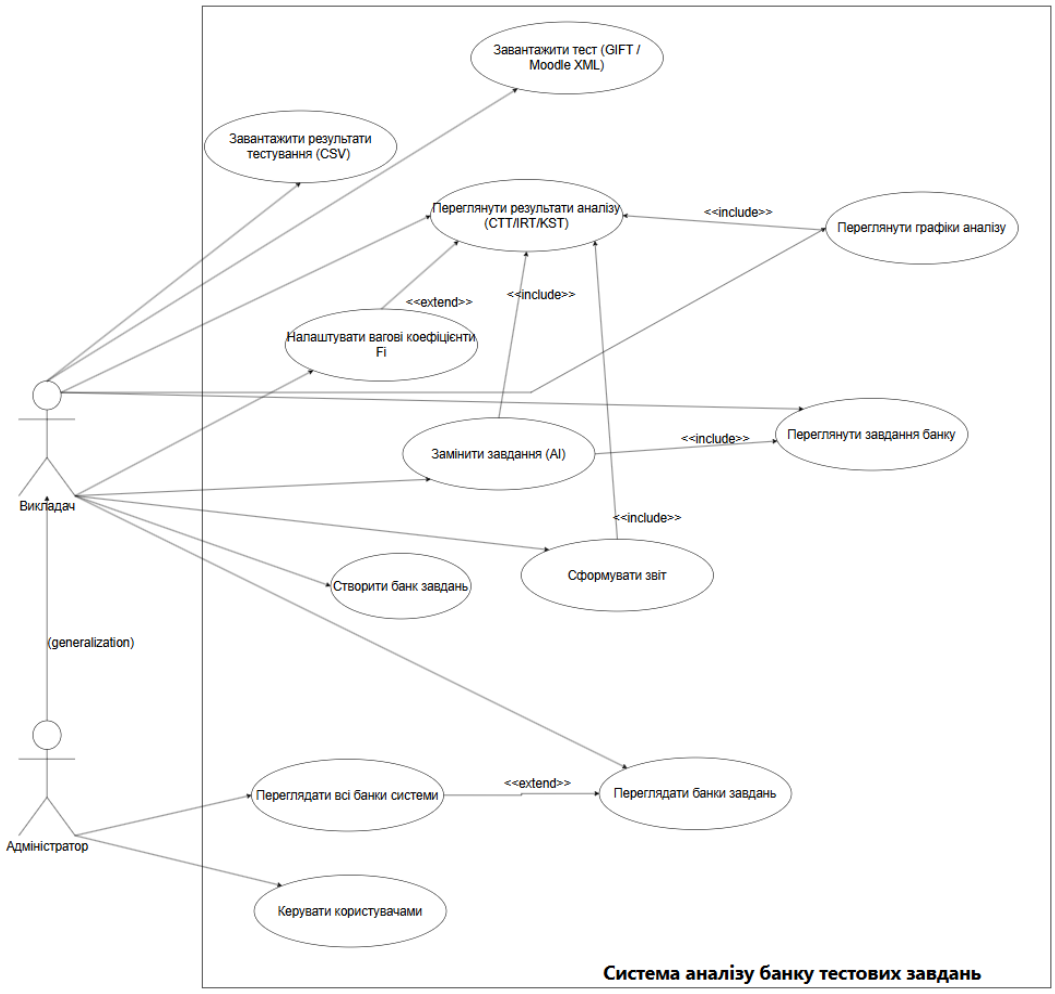


Рисунок 2.1.1.1 — Діаграма використання системи аналізу банку тестових завдань

Впровадження встановлених вимог є основою для створення надійного, масштабованого та науково обґрунтованого інструменту, призначеного для автоматизації аналізу банку тестових завдань та підвищення об'єктивності оцінювання результатів навчання.У межах першої версії реалізується базовий набір функцій психометричного та когнітивного аналізу, тоді як інтеграція з LMS та розширені механізми масштабування розглядаються як перспективи подальшого розвитку.

2.1.3 Аналіз існуючих програмних рішень

Аналіз наявних програмних продуктів для контролю знань, психометричного аналізу тестів та адаптивного тестування дає змогу визначити їхні переваги й недоліки, оцінити відповідність вимогам, виявити обмеження та обґрунтувати необхідність розробки нового рішення без дублювання функціоналу. Вибір для порівняння конкретних програмних продуктів здійснювався за критеріями: наявність аналітики якості тестових завдань, підтримка психометричного аналізу, можливість адаптивного тестування, інтеграція з LMS, наявність засобів звітності та візуалізації.

Одним із широко застосовуваних інструментів для організації тестування та часткового аналізу якості тестових завдань є система Moodle. Це відкрита платформа управління навчанням (LMS), яка дозволяє створювати інтерактивні тести, збирати відповіді користувачів та проводити базову статистичну обробку результатів. Перевагою Moodle є її широка адаптація у навчальних закладах, підтримка різних форматів тестових завдань та можливість розширення функціоналу за допомогою плагінів. Проте аналітичні можливості Moodle обмежуються стандартними звітами про проходження тесту (кількість правильних відповідей, середній бал, розподіл оцінок). Moodle не містить вбудованих засобів психометричного аналізу, таких як розрахунок параметрів IRT, кривих характеристик завдань або інтеграція когнітивних моделей знань [30].

До іншої категорії відноситься платформа ALEKS (Assessment and Learning in Knowledge Spaces), яка є прикладом програмного продукту, що реалізує підхід, близький до теорії простору знань (Knowledge Space Theory). ALEKS використовує формалізовані моделі знань для діагностики опанованих компетентностей і побудови персоналізованих траєкторій навчання. Система визначає «стан знань» студента, вимірює прогалини у розумінні ключових елементів предмета та адаптує навчальний матеріал відповідно. Такий підхід дозволяє не лише оцінювати якість відповідей, а й враховувати логічні передумови між темами. Водночас ALEKS не включає моделей IRT і не є універсальною LMS, що обмежує її інтеграцію з іншими освітніми системами та побудову загального банку тестів [31].

Серед програмних продуктів, які пропонують розширену психометричну аналітику, особливо виділяється платформа FastTest. Це комерційний програмний продукт, призначений для проведення стандартизованих тестувань і оцінювання результатів на основі психометричних підходів. FastTest дозволяє формувати тестові набори, зберігати банк запитань та збирати відповіді учасників у централізованому сховищі. Однією з ключових переваг FastTest є підтримка моделей Item Response Theory: система може оцінювати параметри завдань за допомогою 1PL, 2PL та 3PL моделей, будувати криві характеристик елементів та здійснювати адаптивний добір питань (Computerized Adaptive Testing, CAT). Це дозволяє більш точно диференціювати рівні підготовки учасників та оптимізувати довжину тестів. Незважаючи на це, система не має механізмів опису передумовних залежностей між темами та орієнтована на великі інституційні впровадження, що обмежує її доступність у вузькоспеціалізованих контекстах і не дозволяє повною мірою вирішувати задачі логічної узгодженості банку завдань [32].

Серед open-source рішень, спрямованих на адаптивне тестування і також використовують психометричні моделі, варто відзначити платформу Concerto. Concerto – це веб-орієнтована система для створення адаптивних CAT-тестів

на основі IRT з підтримкою моделювання латентної змінної та оцінювання параметрів завдань. Адже система не включає засобів аналізу структури знань і потребує значної додаткової розробки для інтеграції з LMS, що обмежує її практичну придатність для непрофільних користувачів [33].

Ще одним релевантним інструментом є Xcalibre, який представляє собою програму для психометричного аналізу на основі Item Response Theory. Xcalibre здатний оцінювати параметри різних моделей IRT, будувати графічні криві характеристик тестових елементів і видавати докладні звіти щодо якості кожного завдання. Такий функціонал робить його корисним для дослідників і методистів, які проводять глибокий аналіз банків тестів у науковому контексті. Проте Xcalibre є спеціалізованою аналітичною платформою і не надає інструментів для інтегрованого використання в рамках LMS або для реалізації повноцінних CAT-тестувань у реальному часі. Також він не включає когнітивного моделювання знань та не пропонує механізмів використання отриманих параметрів для побудови адаптивних освітніх траєкторій студентів без значної програмної доопрацювання [34].

Крім цього, варто зазначити, що пакетні рішення на базі R (ltm, mirt, psych) та Python (pyirt, scikit-learn) широко використовуються в академічній практиці для оцінювання параметрів IRT і CTT, проте не є самодостатніми продуктами для управління тестами і потребують експертних знань та інтеграції з іншими системами, що обмежує їх застосовність у повсякденній освітній діяльності [35].

Порівняльний аналіз розглянутих рішень свідчить, що наявні системи не забезпечують комплексної реалізації всіх необхідних функцій у межах єдиного інтегрованого середовища, поєднуючи лише частину можливостей — психометричний аналіз, когнітивне моделювання або засоби адаптивного тестування. Це зумовлює доцільність створення спеціалізованого програмного продукту, який поєднуватиме механізми психометричного аналізу на основі IRT, когнітивне моделювання знань відповідно до підходів KST, адаптивну

логіку добору завдань та зручні засоби візуалізації й підтримки прийняття педагогічних рішень.

2.2 Вибір архітектурного підходу та технологічного стеку

2.2.1 Вибір архітектурного підходу

Проєктування архітектури програмного забезпечення є визначальним етапом розроблення системи, оскільки саме архітектурні рішення встановлюють структуру застосунку, принципи взаємодії його компонентів, рівень продуктивності, зручність супроводу та можливості подальшого розширення функціональності [36]. Вибір архітектурного підходу здійснюється з урахуванням сформульованих функціональних і нефункціональних вимог, характеру предметної області, а також специфіки реалізації психометричних алгоритмів, зокрема моделей Item Response Theory та підходів когнітивного моделювання знань відповідно до Knowledge Space Theory.

Розроблювана система передбачає виконання обчислювально складних алгоритмів, роботу з масивами числових даних і побудову аналітичних візуалізацій, що висуває підвищені вимоги до швидкодії, масштабованості та доступності системи. З огляду на необхідність забезпечення доступу кількох викладачів до єдиної бази тестових завдань, централізованого зберігання результатів тестування та зручності розгортання без інсталяції додаткового програмного забезпечення на робочих станціях користувачів, найбільш доцільним є застосування клієнт-серверної архітектури з розділенням на незалежні серверну та клієнтську частини. Для обґрунтування цього вибору розглянуто три альтернативні архітектурні підходи, порівняння яких наведено у таблиці 2.2.1.2

Монолітна десктопна архітектура забезпечує автономну роботу без мережевого підключення, проте є неприйнятною для системи орієнтованої на багатокористувацький режим роботи: вона ускладнює одночасний доступ

кількох викладачів до єдиної бази завдань та потребує окремого встановлення на кожній робочій станції.

Таблиця 2.2.1.2— Порівняльний аналіз архітектурних підходів

Критерій	Монолітна десктопна	Мікросервісна	Трирівнева клієнт-серверна
Багатокористувацький режим	Ні	Так	Так
Складність розгортання	Низька	Висока	Середня
Масштабованість	Низька	Висока	Середня
Доступ через браузер	Ні	Так	Так
Відповідність задачам проекту	Ні	Надлишково	Так

Мікросервісна архітектура, попри свою масштабованість та розподіленість, є надмірно складною для системи з обмеженою кількістю функціональних модулів та невеликою кількістю одночасних користувачів — вона збільшує складність розгортання, потребує додаткових механізмів оркестрації та не надає суттєвих практичних переваг у межах поставлених завдань.

З урахуванням наведених факторів обрано трирівневу клієнт-серверну архітектуру з чітким розділенням відповідальності між компонентами. Серверна частина реалізує бізнес-логіку, психометричні обчислення та доступ до бази даних; клієнтська частина забезпечує інтерфейс користувача та взаємодію з сервером через REST API. Така структура відповідає принципам слабкої зв'язаності та розділення відповідальності, що сприяє незалежному розвитку компонентів, спрощує тестування та підтримку системи [36].

У межах обраної архітектури система поділяється на три логічні рівні. Рівень представлення реалізований як SPA і відповідає за відображення даних та взаємодію з серверним API через браузер без необхідності встановлення додаткового програмного забезпечення. Рівень прикладної логіки реалізований на основі FastAPI і виконує психометричні обчислення, аналіз відповідей, оцінювання параметрів IRT та KST, генерацію звітів та інтеграцію з Groq API. Рівень доступу до даних забезпечує зберігання всіх даних системи у Microsoft SQL Server через модуль database/, що виступає шаром абстракції між бізнес-логікою та СУБД.

Схему трирівневої архітектури системи з позначенням технологій на кожному рівні наведено на рисунку 2.2.1.2

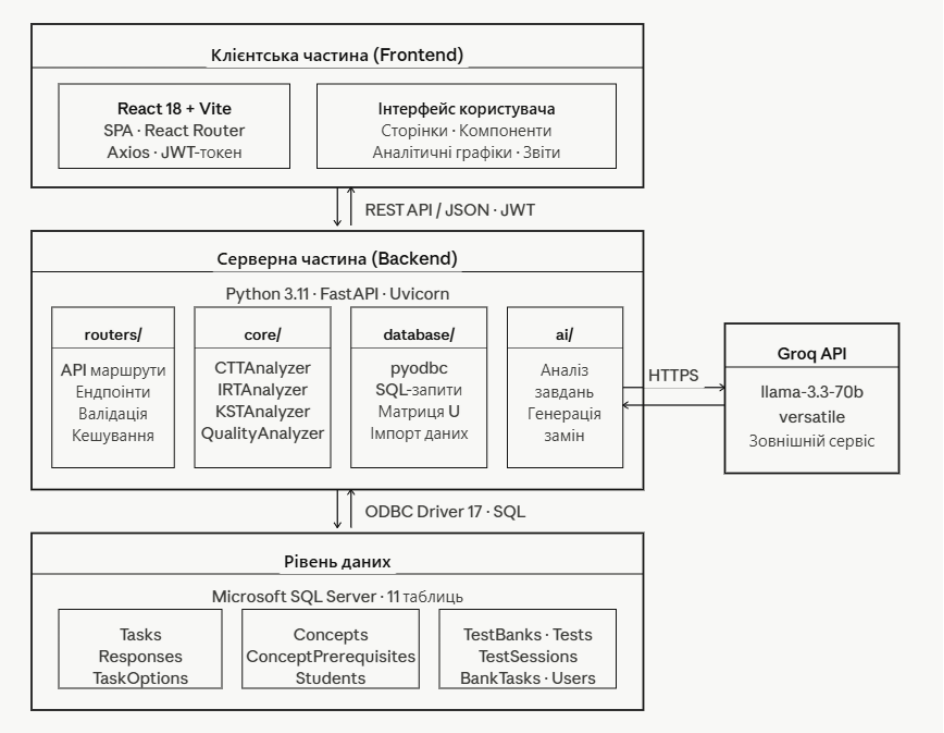


Рисунок 2.2.1.2 — Схема трирівневої клієнт-серверної архітектури системи аналізу банку тестових завдань

Такий багатошаровий поділ відповідає принципам модульності, слабкої зв'язаності та високої когезії компонентів, що позитивно впливає на підтримуваність і масштабованість програмного забезпечення. Крім того,

обрана архітектура спрощує подальший розвиток системи: за потреби окремі модулі можуть бути розширені або винесені в самостійні сервіси без повного перепроєктування застосунку. Перевагами клієнт-серверної архітектури з веб-інтерфейсом є платформна незалежність, централізоване зберігання даних, підтримка рольової моделі доступу та зручність розгортання і оновлення серверної частини без перевстановлення клієнтського програмного забезпечення.

2.2.2 Вибір технологічного стеку

Вибір технологічного стеку розроблюваної системи здійснюється з урахуванням визначених функціональних і нефункціональних вимог, а також обраного архітектурного підходу — трирівневої клієнт-серверної архітектури з розподілом на серверну частину, клієнтську частину та рівень зберігання даних.

Як основну мову серверної частини обрано Python 3.11 завдяки розвиненій екосистемі бібліотек для наукових обчислень, статистичного аналізу та машинного навчання, що безпосередньо відповідає завданням реалізації CTT, IRT та KST.

Для побудови серверного API використано фреймворк FastAPI. Вибір цього інструменту обумовлений його високою продуктивністю завдяки асинхронній обробці запитів, автоматичною генерацією OpenAPI-документації, підтримкою валідації даних через Pydantic-моделі та зручністю маршрутизації запитів. FastAPI запускається через ASGI-сервер Uvicorn, що забезпечує стабільну роботу при одночасних запитах від кількох користувачів [37].

Для реалізації обчислювального ядра використано бібліотеки NumPy та SciPy для роботи з масивами та методами чисельної оптимізації (EM-алгоритм, MAP-оцінювання), pandas — для формування матриці відповідей та агрегації результатів, NetworkX — для перевірки коректності графу передумов KST та виявлення циклів, matplotlib — для генерації аналітичних

графіків (ICC-криві, CTT-гістограми, граф знань, дашборд Fi).

Для підтримки AI-функціональності системи інтегровано Groq API з моделлю llama-3.3-70b-versatile. Вибір цього сервісу обумовлений низькою затримкою відповідей, що є критичним для інтерактивної роботи викладача з інтерфейсом. Модель використовується для семантичного визначення концепту завдання, класифікації за таксономією Блума, аналізу психометричних дефектів та генерації замінних завдань із заданими характеристиками якості.

Рівень представлення реалізований як SPA-застосунок на основі бібліотеки React 18 із використанням збірника Vite. Вибір React обумовлений компонентним підходом до побудови інтерфейсу, ефективним механізмом оновлення DOM через Virtual DOM та широкою екосистемою допоміжних інструментів. Маршрутизація клієнтської частини реалізована через React Router з розподілом на публічні та захищені маршрути. Для взаємодії фронтенду з серверним API використано бібліотеку Axios з двома перехоплювачами: перший автоматично додає JWT-токен до заголовка кожного запиту, другий обробляє відповіді зі статусом 401 для автоматичного виходу з системи [38].

Автентифікація та авторизація реалізовані з використанням механізму JSON Web Tokens (JWT) з алгоритмом підпису HS256 та терміном дії токена 8 годин. Серверна частина генерує підписаний токен при успішному вході користувача; клієнтська частина зберігає його в localStorage та передає у заголовку кожного запиту. Система підтримує рольову модель доступу з двома ролями — адміністратор та викладач, — що забезпечує розмежування функціональності відповідно до повноважень користувача. Перевірка ролей здійснюється через механізм FastAPI Dependency Injection[39].

Для зберігання даних використано Microsoft SQL Server як реляційну систему керування базами даних. Вибір цієї СУБД обумовлений підтримкою транзакційності, механізмами забезпечення цілісності даних на рівні зовнішніх ключів та обмежень CHECK, а також можливістю реалізації

складних аналітичних запитів із груповими операціями. Взаємодія Python-серверу з базою даних здійснюється через бібліотеку pyodbc з використанням ODBC Driver 17 for SQL Server та Windows Authentication, що виключає необхідність зберігання облікових даних у коді.

Для генерації звітів використано бібліотеки ReportLab для формування PDF-документів та openpyxl для створення Excel-файлів із результатами аналізу. Це забезпечує формування структурованих аналітичних матеріалів, що можуть бути використані для прийняття рішень щодо удосконалення тестового інструментарію.

2.3 Формалізація критеріїв оцінювання та відбору тестових завдань (на основі CTT, IRT та KST)

У процесі проєктування системи автоматизованого аналізу банку тестових завдань ключового значення набуває формалізація процедур кількісного оцінювання їх якості та визначення правил прийняття рішень щодо включення або виключення окремих елементів із банку. На відміну від теоретичних засад психометричних і когнітивних підходів, розглянутих у розділі 1, у цьому підрозділі здійснюється їх прикладна інтерпретація у вигляді конкретних обчислюваних показників, граничних значень та алгоритму прийняття рішень, що безпосередньо реалізується програмним забезпеченням.

Формалізація критеріїв дозволяє забезпечити об'єктивність і відтворюваність аналізу, мінімізувати суб'єктивний вплив експертів і надати процесу відбору завдань чітко визначеного алгоритмічного характеру. У межах розроблюваної системи оцінювання здійснюється одночасно з трьох позицій: психометричної на основі CTT, психометричної на основі IRT та когнітивної на основі KST. Результати всіх трьох компонентів інтегруються у єдиний показник якості завдання Fi.

Основою для всіх подальших обчислень є матриця відповідей $U=(u_{ij})$, де $u_{ij}=1$ якщо студент s_j правильно виконав завдання q_i, і $u_{ij}=0$ відповідно у протилежному випадку. За умови наявності менше 100 відповідей система не

блокує аналіз, проте автоматично адаптує ваги компонентів Fi, підвищуючи частку CTT як більш стабільного при малих вибірках методу та знижуючи частку IRT.

Частка правильних відповідей (P-value) для кожного завдання визначається формулою:

$$p_i = \frac{1}{m}\sum_{j=1}^{m} u_{ij} \tag{2.1}$$

де m — кількість студентів;

u_{ij} — бал студента j за завданням *i*.

Допустимий інтервал складності: $0.2 \le p\_i \le 0.9$. Завдання поза цими межами отримують знижену CTT-оцінку [14]. Дискримінаційна здатність завдання визначається як скоригована точково-бісеріальна кореляція:

$$r_i = \frac{\mathrm{cov}(u_i, T_{-i})}{\sqrt{D(u_i) \times D(T_{-i})}} \tag{2.2}$$

де r_i — дискримінаційний індекс *i*-го завдання;

u_i — вектор балів студентів за завданням i*;*

T_{-i} — загальний бал студента без урахування завдання i*;*

$D(u_i)$ — дисперсія балів за завданням i*;*

$D(T_{-i})$ — дисперсія загального балу без завдання i.

Мінімально прийнятним є значення $r\_i \ge 0.20$; від'ємні значення свідчать про серйозні проблеми із завданням [14].

Надійність тесту як цілісного інструменту оцінюється коефіцієнтом Кронбаха:

$$\alpha = \frac{k}{k-1}\left(1 - \frac{\sum_{i=1}^{k}\sigma_i^2}{\sigma_T^2}\right) \tag{2.3}$$

де k — кількість завдань;

σ_{i^2} — дисперсія результатів окремого завдання;

σ_{T^2} — дисперсія сумарного балу.

Від'ємне значення α свідчить про те, що завдання вимірюють різні конструкти — у такому разі CTT-компонента Fi для всіх завдань банку встановлюється у нуль.

У системі застосовується двопараметрична логістична модель (2PL). Ймовірність правильної відповіді студента з рівнем підготовки θ визначається формулою [24]:

$$P(u_{ij} = 1|\theta_j) = \frac{1}{1 + e^{-a_i(\theta_j - b_i)}} \quad (2.4)$$

де b_i — параметр складності;

a_i —параметр дискримінації.

Граничні значення параметрів: $-4.0 \leq b_i \leq 4.0$, $0.1 \leq a_i \leq 4.0$. Оцінювання параметрів здійснюється EM-алгоритмом із MAP-регуляризацією. Для оцінювання вимірювальної цінності додатково обчислюється інформаційна функція завдання:

$$I_i(\theta) = a_i^2 \cdot P_i(\theta) \cdot (1 - P_i(\theta)) \quad (2.5)$$

Визначається середня інформативність на інтервалі $\theta \in [-4; 4]$, що використовується як один із компонентів при нормалізації IRT-оцінки завдання.

Індивідуальний рівень підготовки студента θ_j оцінюється методом максимальної правдоподібності:

$$\hat{\theta}_j = arg\max_{\theta} \sum_{i=1}^{n} [u_{ij} \ln P_i(\theta) + (1 - u_{ij}) \ln (1 - P_i(\theta))] \quad (2.6)$$

Для моделювання передумов навчання формується орієнтований граф G = (C,E), де C — множина концептів курсу, а ребро (c_a, c_b) означає що засвоєння c_a є необхідною умовою для c_b. Для кожного концепту c перевіряється мінімальна вимога покриття:

$$|Q_c| \geq k_{min} \quad (2.7)$$

де k_{min} — мінімально допустима кількість завдань (зазвичай 2–3). Якщо для певного концепту кількість завдань є недостатньою, система автоматично фіксує прогалину в банку [25].

Інтегральний показник якості Fi розраховується як зважена сума трьох нормалізованих компонент:

$$F_i = w_{CTT} \cdot CTT_i + w_{IRT} \cdot IRT_i + w_{KST} \cdot KST_i \quad (2.8)$$

де:

Fi — інтегральний показник якості *i*-го тестового завдання;

CTTi — нормалізована оцінка якості завдання за показниками класичної теорії тестів (CTT);

IRTi — нормалізована оцінка якості завдання за моделлю Item Response Theory (IRT);

KSTi — нормалізована оцінка якості завдання за показниками Knowledge Space Theory (KST);

w_{CTT}, w_{IRT}, w_{KST} — вагові коефіцієнти відповідних компонентів

Причому всі компоненти нормалізовані до інтервалу [0;1]. Ваги адаптуються залежно від розміру вибірки: при $n \geq 100$ використовуються $w_{ctt} = 0.40$, $w_{irt} = 0.30$, $w_{kst} = 0.30$; при $n < 100$ — $w_{ctt} = 0.50$, $w_{irt} = 0.20$, $w_{kst} = 0.30$. Граничні значення всіх показників, що використовуються системою при прийнятті рішень, зведено у таблиці 2.3.3.

Таблиця 2.3.3 — Граничні значення показників якості тестових завдань

Показник	**Діапазон**	**Інтерпретація**
P-value (p_i)	< 0.2	Завдання занадто складне
P-value (p_i)	0.2 – 0.9	Прийнятна складність
P-value (p_i)	> 0.9	Завдання занадто легке
D-index (r_i)	< 0.20	Низька дискримінація
D-index (r_i)	≥ 0.20	Прийнятна дискримінація

Закінчення таблиці 2.3.3

α Кронбаха	< 0.7	Низька надійність тесту
α Кронбаха	0.7 – 0.89	Прийнятна надійність
α Кронбаха	≥ 0.9	Висока надійність
IRT a (a_i)	0.1 – 4.0	Допустимий діапазон
IRT b (b_i)	−4.0 – 4.0	Допустимий діапазон
F_i	≥ 0.70	Статус: active
F_i	0.40 – 0.69	Статус: review
F_i	< 0.40	Статус: remove

Алгоритм обчислення показника Fi реалізується у такій послідовності: збір результатів тестування та формування матриці відповідей; перевірка розміру вибірки та адаптація ваг; розрахунок показників CTT; оцінювання параметрів IRT за допомогою EM-алгоритму з MAP-регуляризацією; обчислення середньої інформативності завдань; перевірка когнітивних зв'язків та структурної коректності завдань у графі KST; нормалізація всіх компонент до інтервалу [0;1]; розрахунок Fi та присвоєння статусу; збереження всіх параметрів у базу даних. Блок-схему цього алгоритму наведено на рисунку 2.3.3.

Запропонована формалізація критеріїв поєднує кількісні психометричні показники з логічною структурою знань, що дозволяє оцінювати завдання одночасно за вимірювальною ефективністю та педагогічною доцільністю. Важливою особливістю алгоритму є механізм адаптивних ваг: при малій вибірці (n < 100) вага CTT-компоненти підвищується до 0,50, тоді як вага IRT знижується до 0,20, що забезпечує стабільність оцінювання за умов обмеженого обсягу даних. Тривагова формула Fi забезпечує гнучкість оцінювання та коректну поведінку системи за різних умов збору даних. Усі наведені процедури виконуються при кожному запуску аналізу банку завдань, що дозволяє системі уточнювати психометричні характеристики завдань зі зростанням кількості тестових сесій.

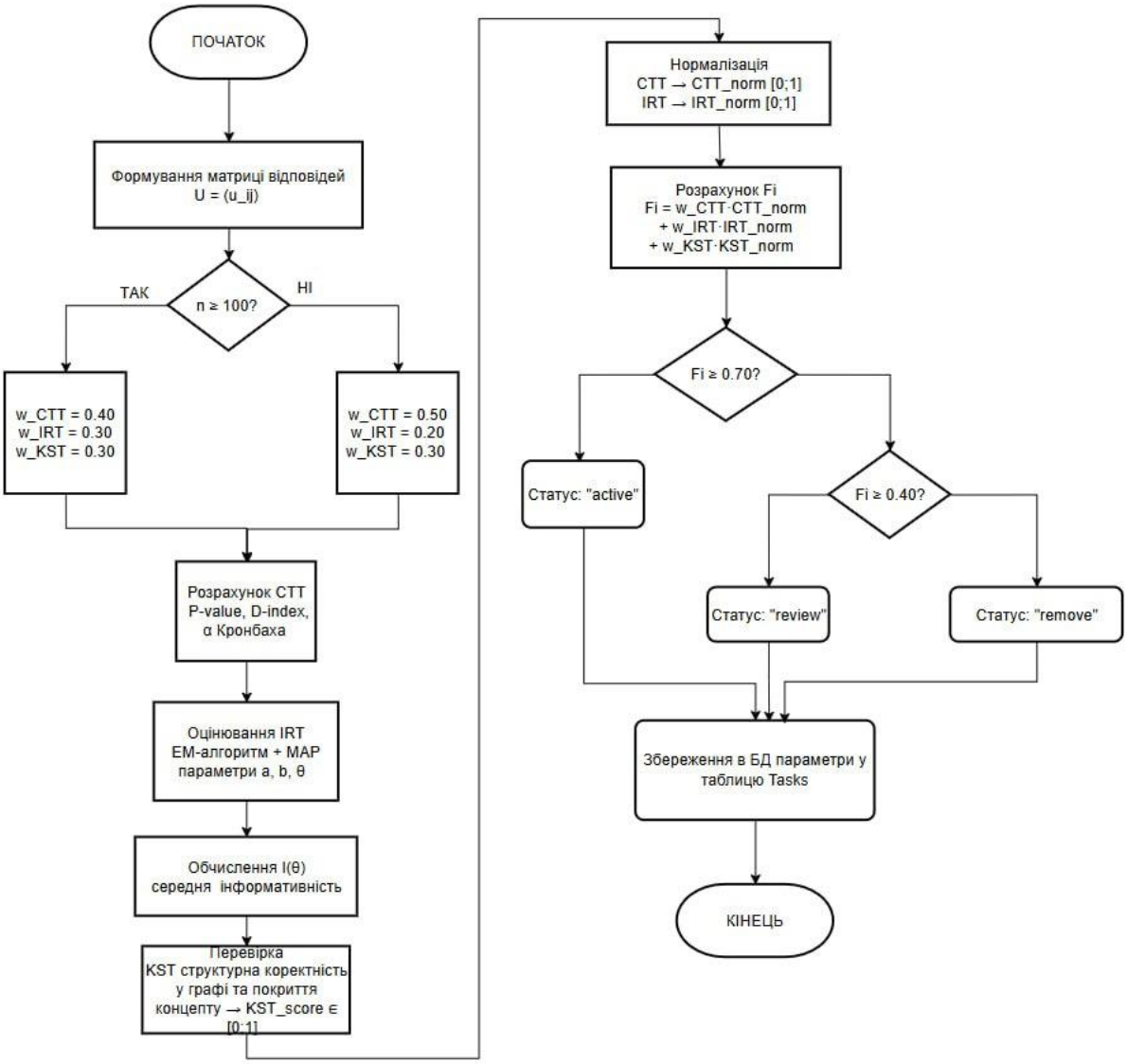


Рисунок 2.3.3 — Блок-схема алгоритму обчислення інтегрального показника якості тестового завдання Fi

2.4 Розробка математичної моделі аналізу якості тестових завдань та простору знань студента (IRT + KST)

Після формалізації критеріїв оцінювання та відбору тестових завдань наступним етапом є побудова цілісної математичної моделі, яка інтегрує психометричні показники якості завдань та структурний опис простору знань студента у межах єдиної формальної системи. Кортеж M є формальним описом усіх сутностей і зв'язків системи, що використовується надалі як основа

програмної реалізації. Метою побудови моделі є формалізація процесу оцінювання знань таким чином, щоб забезпечити максимальну точність визначення рівня підготовки студента при мінімальній кількості тестових завдань за умови збереження логічної послідовності засвоєння навчального матеріалу та контролю якості банку завдань.

Інтегрована математична модель задається кортежем:

$$M=(Q,S,U,A,B,\Theta,G,Q_c,F), \tag{2.9}$$

де: $Q=\{q_1,q_2,\ldots,q_n\}$ — множина тестових завдань;

$S=\{s_1,s_2,\ldots,s_m\}$ — множина студентів;

$U=(u_{ij})$ — матриця відповідей;

$A=\{a_i\}$ — параметри дискримінації завдань;

$B=\{b_i\}$ — параметри складності;

$\Theta=\{\theta_j\}$ — оцінені рівні здібностей студентів;

$G=(Q,E)$ — орієнтований граф передумов знань;

$Q_c \subseteq Q$ — підмножини завдань, що відповідають окремим концептам;

$F=\{F_i\}$ — інтегровані показники якості завдань.

Таке формальне представлення дозволяє чітко відокремити компоненти психометричного вимірювання від компонентів, що описують когнітивну структуру предметної області. Психометричні складові A, B, Θ та F обчислюються на основі критеріїв, формалізованих у підрозділі 2.3, тоді як когнітивна складова G і Qc описує логічну структуру навчального матеріалу відповідно до підходів KST.

Стан знань студента s_j описується множиною $K_j \subseteq Q$, яка задовольняє умову замкненості відносно передумов :

$$(q_a, q_b) \in E \ \wedge \ q_b \in K_j \Rightarrow q_a \in K_j \tag{2.10}$$

Це означає, що студент не може вважатися таким, що засвоїв складнішу тему, якщо він не опанував її передумови. Належність завдання до поточного стану знань визначається через порогову ймовірність успішного виконання:

$$q_i \in K_j \iff P_i(\theta_j) \geq P_{threshold}. \quad (2.11)$$

Додатково вводиться умова покриття концептів:

$$|Q_c \cap T_j| \geq k_{min}, \forall c, \quad (2.12)$$

де T_j — множина завдань, пред'явлених студенту;

k_{min} — мінімальна кількість завдань для перевірки кожного концепту. Це забезпечує повноту оцінювання навчального матеріалу.

Інтеграція IRT та KST реалізується через формулювання задачі адаптивного добору завдань як задачі умовної оптимізації. На кожному кроці тестування система розв'язує задачу

$$q_i \in K_j \iff P_i(\theta_j) \geq P_{threshold}. \quad (2.13)$$

$$q_i^* = \arg\max_{q_i \in Q} I_i(\theta_j) \quad (2.14)$$

за умов: $q_i \in Q_{allowed}$, $F_i \geq F_{min}$.

Множина Q_allowed формується на основі поточного стану знань K_j та структури графа передумов G: до неї включаються лише ті завдання, передумови яких вже перевірені. Таким чином, обирається завдання, яке є одночасно максимально інформативним для поточного рівня підготовки студента та структурно допустимим з погляду когнітивної логіки навчання.

Процес тестування має динамічний характер. Після кожної відповіді студента виконується оновлення оцінки здібностей:

$$\theta_j^{(t+1)} = f(\theta_j^{(t)}, u_{ij}), \quad (2.15)$$

де функція f реалізує процедуру максимальної правдоподібності або інший чисельний метод оцінювання.

Тестування завершується, якщо виконується умова досягнення заданої точності оцінювання

$$Var(\hat{\theta}_j) \leq \varepsilon, \quad (2.16)$$

або якщо кількість пред'явлених завдань досягає граничного значення. Таким чином, глобальна ціль моделі формалізується як задача мінімізації довжини

тесту $\min|T_j|$ за умови забезпечення необхідної точності оцінювання здібностей студента.

Математична модель поєднує психометричні та когнітивні аспекти оцінювання в межах єдиної формальної системи. Вона забезпечує автоматичний контроль якості завдань, динамічне оцінювання рівня підготовки студента, структурно коректний адаптивний добір завдань і мінімізацію тривалості тестування при збереженні заданої точності. Така модель є основою програмної реалізації адаптивної системи тестування та забезпечує узгодженість статистичних і педагогічних критеріїв оцінювання знань.

2.5 Проєктування моделі даних та структури бази даних

Проєктування моделі даних і структури бази даних є ключовим етапом розробки системи аналізу банку тестових завдань. Від правильного подання та організації даних залежить ефективність обчислень, швидкість доступу до інформації та надійність роботи програмного модуля. База даних виконує роль сховища для тестових завдань, результатів їх проходження, параметрів психометричних моделей, когнітивної структури знань та проміжних обчислюваних показників.

Основним принципом проєктування є нормалізація даних, яка запобігає дублюванню та забезпечує цілісність інформації. База даних реалізована засобами Microsoft SQL Server і містить одинадцять взаємопов'язаних таблиць: TestBanks, Tests, Tasks, BankTasks, TaskOptions, Concepts, ConceptPrerequisites, Students, TestSessions, Responses та Users. Повна структура всіх таблиць наведена в Додатку А. Нижче детально описано п'ять ключових таблиць, що безпосередньо реалізують математичну модель підрозділу 2.4.

Таблиця Concepts є довідником тем курсу «Бази даних», де кожен запис описує окремий навчальний концепт — наприклад, «SELECT запити», «JOIN операції» або «Нормалізація 3НФ» — і підготовлює граф передумов KST та аналізу покриття банку завдань.Її структуру наведено у таблиці 2.5.4.

Таблиця 2.5.4 — Структура таблиці Concepts

Поле	Тип даних	Опис	Обмеження
concept_id	INT	Унікальний ідентифікатор концепту	PRIMARY KEY, IDENTITY
name	VARCHAR	Назва теми курсу (унікальна)	NOT NULL, UNIQUE
description	TEXT	Розгорнутий опис теми	–
created_at	DATETIME	Дата та час додавання запису	NOT NULL, DEFAULT GETDATE()

Таблиця ConceptPrerequisites реалізує орієнтований ациклічний граф передумов між концептами — ключовий елемент Knowledge Space Theory. Кожен запис описує одне відношення: концепт prerequisite_id є необхідною умовою для засвоєння концепту concept_id. Обидва зовнішні ключі визначені з ON DELETE NO ACTION для захисту цілісності графу KST.Структуру таблиці наведено у таблиці 2.5.5

Таблиця 2.5.5 — Структура таблиці ConceptPrerequisites

Поле	Тип даних	Опис	Обмеження
concept_id	INT	Концепт, що перевіряється	FK → Concepts, ON DELETE NO ACTION
prerequisite_id	INT	Концепт-передумова	FK → Concepts, ON DELETE NO ACTION
–	–	Складений первинний ключ	PK (concept_id, prerequisite_id)

Таблиця Tasks є центральною таблицею системи, що зберігає тексти завдань та всі розраховані показники якості. Після запуску аналізу програма автоматично записує CTT-показники, IRT-параметри, результат KST-перевірки, інтегральний показник Fi, статус завдання та текстову рекомендацію від Groq API. Обмеження значень психометричних параметрів реалізуються на рівні прикладної логіки, а не через CHECK-constraints у СУБД, що забезпечує гнучкість при розширенні системи. Структуру таблиці наведено у таблиці 2.5.6.

Таблиця 2.5.6 — Структура таблиці Tasks

Поле	**Тип даних**	**Опис**	**Обмеження**
task_id	INT	Унікальний ідентифікатор завдання	PRIMARY KEY, IDENTITY
text	TEXT	Текст тестового завдання	NOT NULL
correct_answer	VARCHAR	Правильна відповідь (для відкритих питань та SQL-завдань; NULL для завдань з вибором відповіді)	NULL
concept_id	INT	Прив'язаний концепт	FK → Concepts, ON DELETE SET NULL
bloom_level	VARCHAR	Рівень таксономії Блума (визначається автоматично)	–
task_type	VARCHAR	Тип завдання (theoretical / practical)	–
difficulty_level	FLOAT	P-value: частка правильних відповідей (CTT)	–
discrimination	FLOAT	D-index: кореляція з підсумковим балом (CTT)	–
irt_a	FLOAT	Параметр дискримінації IRT, модель 2PL	–

Закінчення таблиці 2.5.6

irt_b	FLOAT	Параметр складності IRT, модель 2PL	–
information_avg	FLOAT	Середня інформативність I(θ) на діапазоні [–4; 4]	–
structural_ok	BIT	Результат KST-перевірки структурної коректності	NOT NULL, DEFAULT 0
quality_score	FLOAT	Інтегральний показник якості Fi	–
status	VARCHAR	Статус після аналізу: active / review / remove	DEFAULT 'active'
ai_recommendation	TEXT	Текстова рекомендація від Groq API	–
max_score	FLOAT	Максимальний бал за завдання (для нормалізації часткових балів)	DEFAULT 1.0
created_at	DATETIME	Дата додавання завдання	NOT NULL, DEFAULT GETDATE()

Таблиця TaskOptions зберігає варіанти відповідей для завдань із вибором. Кожен запис відповідає одному варіанту конкретного завдання; поле is_correct визначає, чи є варіант правильним. Таблиця використовується при імпорті завдань у форматах GIFT та Moodle XML, а також при збереженні AI-згенерованих завдань. Структуру таблиці наведено у таблиці 2.5.7.

Таблиця 2.5.7 — Структура таблиці TaskOptions

Поле	**Тип даних**	**Опис**	**Обмеження**
option_id	INT	Унікальний ідентифікатор варіанту	PRIMARY KEY, IDENTITY
task_id	INT	Завдання, до якого належить варіант	FK → Tasks, ON DELETE CASCADE

Закінчення таблиці 2.5.7

option_text	VARCHAR	Текст варіанту відповіді	NOT NULL
is_correct	BIT	Ознака правильності варіанту	NOT NULL

Таблиця Responses є основною аналітичною таблицею системи та зберігає відповіді студентів на завдання в межах тестових сесій. Поля score та is_correct використовуються для формування матриці відповідей $U=(u_{ij})$, яка є основою розрахунків CTT та IRT. Для поля task_id застосовано обмеження ON DELETE NO ACTION з метою збереження історичних даних. Структуру таблиці наведено в таблиці 2.5.8.

Таблиця 2.5.8 — Структура таблиці Responses

Поле	**Тип даних**	**Опис**	**Обмеження**
response_id	INT	Унікальний ідентифікатор відповіді	PRIMARY KEY, IDENTITY
student_id	INT	Студент, що виконував завдання	FK → Students, ON DELETE CASCADE
task_id	INT	Виконане тестове завдання	FK → Tasks, ON DELETE NO ACTION
session_id	INT	Сесія тестування	FK → TestSessions, ON DELETE SET NULL
is_correct	BIT	1 – правильна відповідь, 0 – неправильна	NOT NULL
score	FLOAT	Сирий бал студента за завдання до бінаризації	–
response_time	DECIMAL	Час виконання завдання у секундах	–
created_at	DATETIME	Дата та час фіксації відповіді	NOT NULL, DEFAULT GETDATE()

Для первинних і зовнішніх ключів використовується тип INT, для психометричних параметрів і рівня підготовки θ — FLOAT, для часу відповіді — DECIMAL, а для бінарних показників — BIT. Обмеження CASCADE, SET NULL та NO ACTION застосовуються залежно від необхідності видалення пов'язаних записів, збереження історії або захисту аналітичних даних. Контроль допустимих значень психометричних параметрів реалізовано на рівні прикладної логіки відповідно до таблиці 2.4.

Усі таблиці пов'язані зовнішніми ключами, що забезпечує цілісність даних на рівні СУБД. Центральною є таблиця Tasks, яка через concept_id пов'язана з Concepts, через BankTasks — з TestBanks та Tests, через Responses — зі Students та TestSessions, через TaskOptions — з варіантами відповідей. Для підвищення швидкодії створено індекси на полях concept_id, status, difficulty_level у таблиці Tasks та student_id, task_id, session_id у таблиці Responses. ER-діаграму бази даних наведено на рисунку 2.5.3.

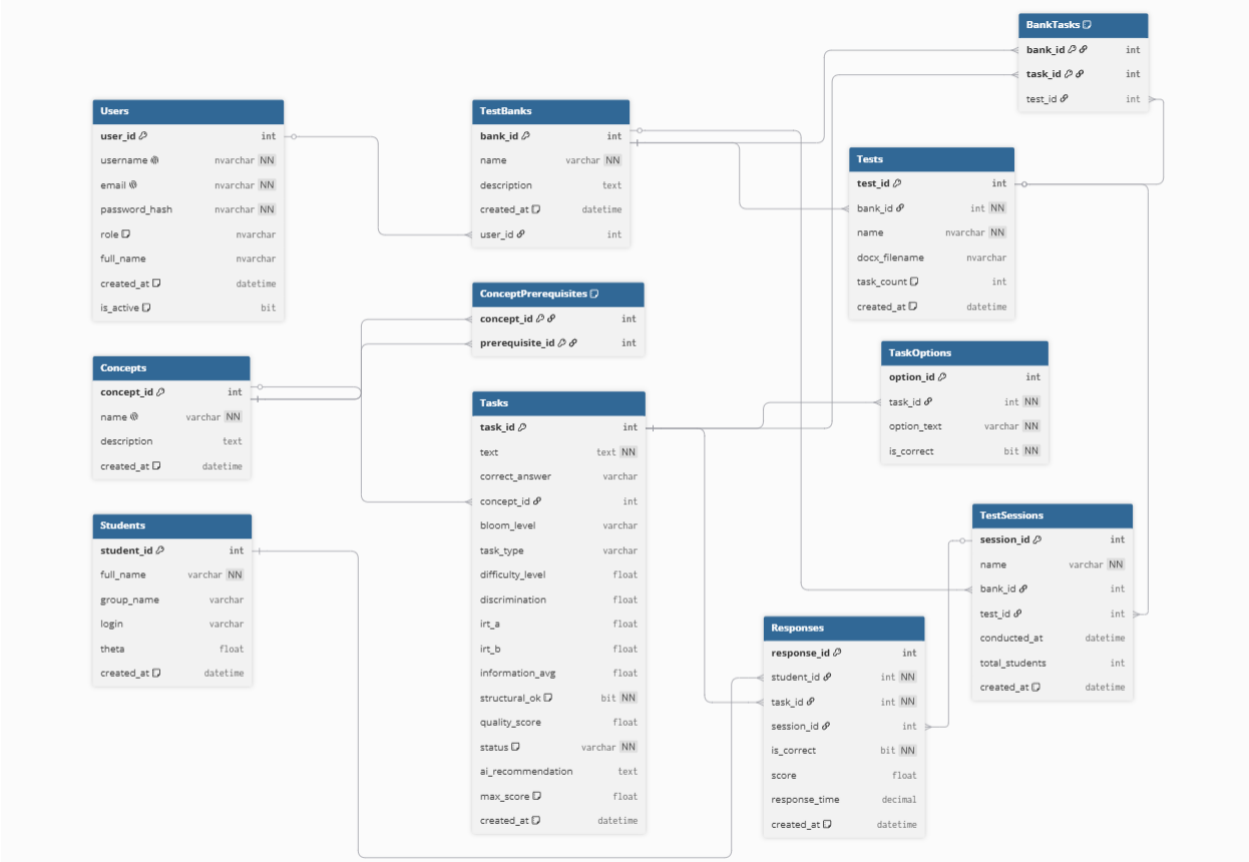


Рисунок 2.5.3 — ER-діаграма бази даних системи аналізу банку тестових завдань

2.6 Висновки за розділом 2

За результатами другого розділу сформовано цілісну методичну та технологічну основу проєктування системи автоматизованого аналізу банку тестових завдань. Відповідно до стандартів ISO/IEC/IEEE 29148 та ISO/IEC 25010 визначено одинадцять функціональних і нефункціональних вимог, що охоплюють психометричний аналіз CTT та IRT, когнітивне моделювання KST, генерацію завдань засобами великих мовних моделей, а також вимоги до продуктивності, надійності та безпеки системи. Діаграма використання відображає розподіл функціональності між двома акторами — викладачем та адміністратором.

Порівняльний аналіз існуючих рішень — Moodle, ALEKS, FastTest, Concerto, Xcalibre та бібліотек R і Python — підтвердив, що жодне з них не забезпечує комплексної інтеграції психометричного та когнітивного аналізу в єдиному інструменті, що обґрунтовує доцільність розроблення спеціалізованого програмного продукту.

Обґрунтовано вибір трирівневої клієнт-серверної архітектури та технологічного стеку на основі Python 3.11, FastAPI, React 18, Microsoft SQL Server і Groq API. Формалізовано критерії оцінювання завдань, побудовано інтегровану математичну модель IRT + KST у вигляді кортежу M (формула 2.9) та спроєктовано реляційну структуру бази даних з одинадцяти таблиць.

Таким чином, другий розділ сформував цілісну методичну та технологічну основу реалізації системи, забезпечивши логічний перехід від теоретичних положень до практичного програмного проєктування та створивши передумови для безпосередньої розробки програмного додатку, що розглядається у розділі 3.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ДЛЯ АНАЛІЗУ ТА ОПТИМІЗАЦІЇ БАНКУ ТЕСТОВИХ ЗАВДАНЬ

3.1 Проєктування архітектури програмного додатку

Архітектура розробленого програмного додатку реалізована за принципом трирівневої клієнт-серверної взаємодії, обґрунтованим у підрозділі 2.2.1, з розділенням функцій між рівнем представлення, рівнем прикладної логіки та рівнем даних. Математичне ядро системи ізольоване від транспортного рівня, що забезпечує можливість повторного використання аналітичних модулів незалежно від способу взаємодії з клієнтом.

Загальна структура системи складається з трьох основних рівнів: рівень представлення — React SPA; рівень прикладної логіки — FastAPI Backend; рівень даних — Microsoft SQL Server. Взаємодія між рівнями здійснюється через REST API у форматі JSON.

Серверна частина організована за модульним принципом та містить окремі компоненти для маршрутизації запитів, математичного аналізу, роботи з базою даних і AI-функціональності. Структуру серверної частини наведено нижче:

```
test_bank_analyzer/
├── main.py
├── auth.py
├── routers/
├── core/
├── database/
└── ai/
```

Директорія routers/ містить API-маршрутизатори для управління банками завдань, тестовими сесіями, імпортом даних, запуском аналізу та генерацією звітів. Директорія core/ реалізує математичне ядро системи, що включає модулі CTT, IRT, KST та інтегрального оцінювання якості тестових завдань — детальну реалізацію наведено у підрозділі 3.3. Директорія database/ відповідає за взаємодію з базою даних, формування матриці відповідей та імпорт зовнішніх форматів тестових даних. Директорія ai/ містить модуль

інтеграції з Groq API для аналізу та генерації тестових завдань — його реалізацію розглянуто у підрозділі 3.4.

Точкою входу серверної частини є файл main.py, у якому реєструються API-маршрутизатори та налаштовуються механізми взаємодії між компонентами системи. Кожен маршрутизатор інкапсулює логіку окремого функціонального модуля та взаємодіє з математичним ядром і рівнем даних через чітко визначені інтерфейси. Для забезпечення продуктивності при повторному зверненні до результатів аналізу у маршрутизаторі analysis.py реалізовано механізм кешування через словник `_analysis_cache`. Після завершення аналізу об'єкт `QualityAnalyzer` зберігається у кеші за ідентифікатором банку, що дозволяє генерувати звіти та отримувати графіки без повторного запуску обчислювально затратного EM-алгоритму IRT. Кеш діє в межах поточного серверного процесу і очищується при перезапуску сервера.

Взаємодія між клієнтською і серверною частинами налаштована через CORS middleware. У середовищі розробки дозволяються запити з адреси http://localhost:5173 — порту розробницького сервера Vite; у виробничому середовищі список дозволених джерел обмежується конкретною адресою розгорнутого клієнтського застосунку.

Система автентифікації реалізована на основі JWT-токенів із використанням рольової моделі доступу. Додаток підтримує дві ролі користувачів: admin та teacher. Захист API-маршрутів здійснюється через механізм Dependency Injection фреймворку FastAPI, що забезпечує централізовану перевірку автентичності при кожному запиті. Паролі користувачів зберігаються у хешованому вигляді з використанням алгоритму bcrypt.

Рівень доступу до даних забезпечує централізовану взаємодію з Microsoft SQL Server та формування матриці відповідей студентів для подальшого психометричного аналізу. Для представлення основних сутностей

предметної області використовуються dataclass-моделі завдання, студента, відповіді, тестової сесії та банку тестових завдань.

Для забезпечення цілісності даних між таблицями бази даних встановлено систему зовнішніх ключів та обмежень цілісності. Це дозволяє гарантувати коректність зв'язків між банками тестових завдань, тестовими сесіями, студентами та їх відповідями. Крім того, централізований рівень доступу до даних ізолює бізнес-логіку від конкретної реалізації сховища, що спрощує подальшу модифікацію або заміну системи управління базами даних без необхідності суттєвих змін у прикладній логіці.

Клієнтська частина організована за принципом сторінкових компонентів та підтримує розділення на публічні й захищені маршрути. HTTP-взаємодія із сервером централізована через окремий API-модуль, який автоматично додає JWT-токен до кожного запиту та обробляє помилки автентифікації через перенаправлення на сторінку входу. Така організація дозволяє уникнути дублювання коду при роботі з API та забезпечує єдиний механізм взаємодії між клієнтом і сервером.

Центральним процесом системи є запуск повного психометричного аналізу банку тестових завдань. При виклику ендпоінту POST /api/analysis/{bank_id}/run відбувається така послідовність операцій: функція `get_response_matrix()` формує матрицю відповідей студентів розміром n_students × n_items; екземпляр QualityAnalyzer послідовно виконує CTT, IRT та KST аналіз із нормалізацією результатів кожної моделі до інтервалу [0;1]; інтегральний показник Fi зберігається у таблиці Tasks разом із розрахованими параметрами завдань; клас Visualizer генерує п'ять аналітичних графіків; API повертає структурований JSON із підсумковою статистикою та детальними результатами по кожному завданню. Загальну схему потоку даних наведено на рисунку 3.1.1.

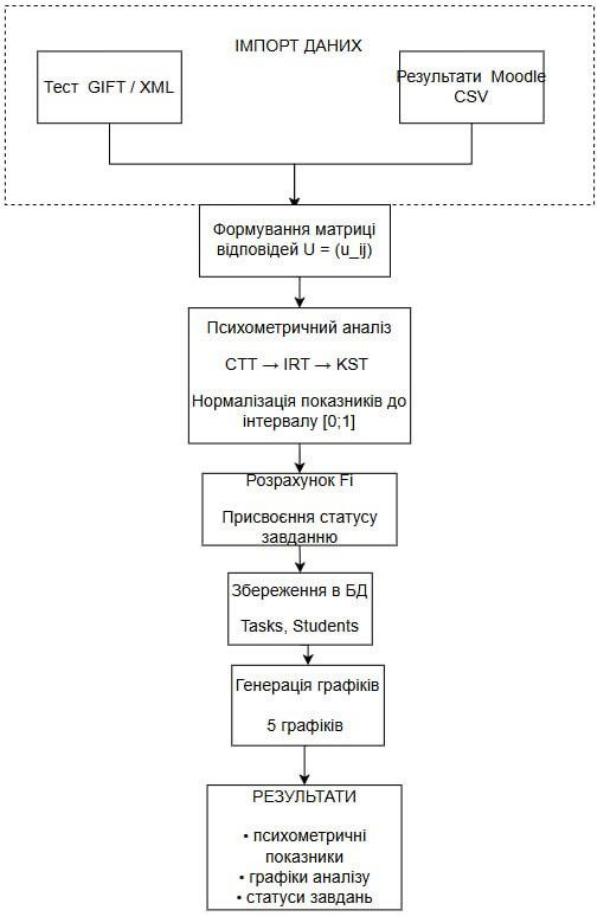


Рисунок 3.1.1 — Pipeline обробки даних від імпорту результатів тестування до оновлення банку тестових завдань

Запропонована архітектура забезпечує модульність системи, незалежність окремих рівнів застосунку та можливість подальшого розширення функціональності шляхом додавання нових аналітичних модулів або сервісів без зміни існуючої структури програмного забезпечення.

3.2 Реалізація модулю збору та обробки результатів тестування

Модуль збору та обробки результатів тестування є вхідним шаром системи, що забезпечує отримання, структурування та підготовку даних для подальшого психометричного аналізу. Основними функціями модуля є імпорт тестових завдань із зовнішніх форматів, імпорт результатів тестування студентів, нормалізація отриманих даних та формування матриці відповідей

для математичного ядра системи. Реалізація модуля зосереджена у директорії database/ та маршрутизаторах routers/imports.py і routers/tests.py.

Підсистема імпорту тестових завдань забезпечує сумісність із наявними освітніми платформами через підтримку двох форматів — GIFT та Moodle XML Quiz Export. Вибір форматів обумовлений їх широким застосуванням у системі управління навчанням Moodle, що використовується як основне середовище тестування у навчальному процесі.

Імпорт завдань у форматі GIFT реалізовано у файлі tasks_importer.py. Парсер виконує послідовну обробку текстового файлу, автоматично визначаючи структуру тестового завдання та його тип. Система підтримує завдання з одиничним і множинним вибором, короткою відповіддю, встановленням відповідності та частковим нарахуванням балів. Під час імпорту виконується очищення HTML-розмітки, декодування службових символів формату GIFT та уніфікація структури відповідей для подальшого збереження у базі даних. Окремо реалізовано механізм визначення теми завдання через аналіз службових HTML-елементів Moodle до етапу очищення розмітки, що дозволяє зберігати зв'язок між тестовим завданням і тематичною структурою навчального курсу навіть після видалення службового форматування.

Імпорт завдань у форматі Moodle XML реалізовано у файлі moodle_xml_importer.py з використанням стандартного XML-парсера Python. Підтримуються основні типи питань Moodle: multichoice, truefalse, shortanswer, numerical, essay та matching. Коректність варіантів відповідей визначається на основі вагових коефіцієнтів, що забезпечує підтримку як звичайних тестів, так і завдань із частковими балами. Під час імпорту система також виконує очищення HTML-розмітки та уніфікацію структури відповідей незалежно від вихідного формату. Обидва імпортери реалізують єдину логіку обробки даних і повертають уніфіковану структуру, що містить текст завдання, тип питання, варіанти відповідей, правильні відповіді, категорію та тему. Такий підхід ізолює логіку парсингу від механізмів збереження даних і спрощує подальше

розширення системи новими форматами імпорту.

Визначення концепту навчального курсу для кожного завдання є необхідною умовою подальшого застосування моделі Knowledge Space Theory і здійснюється у два етапи. На першому етапі система виконує автоматичне співставлення теми або категорії завдання з наявними концептами у базі даних — використовується як точне, так і часткове співставлення назв без урахування регістру, з пріоритетом теми зі структури Moodle над категорією. Якщо автоматичне співставлення не дає результату, система звертається до AI-модуля (підрозділ 3.4) для семантичного визначення концепту: модель аналізує текст завдання та перелік доступних концептів курсу і повертає найбільш імовірний результат. Додатково для кожного завдання визначається рівень таксономії Блума. Отримані значення `concept_id` та `bloom_level` зберігаються у базі даних і надалі використовуються компонентами KST-аналізу та AI-модуля рекомендацій.

Імпорт результатів тестування студентів реалізовано на основі стандартного CSV-звіту Moodle через функцію `import_moodle_csv()` у файлі moodle_importer.py. Система автоматично визначає кодування та структуру CSV-файлу, після чого виконує обробку результатів тестових сесій. На етапі попередньої обробки видаляються службові рядки статистики Moodle та визначаються колонки, що містять результати відповідей студентів; для кожного тестового завдання додатково визначається максимальний бал, необхідний для подальшої нормалізації результатів. Система підтримує три стратегії обробки повторних спроб проходження тесту: збереження першої спроби, збереження останньої спроби або збереження найкращого результату.

За замовчуванням використовується остання спроба, оскільки вона найбільш повно відображає актуальний рівень знань студента після завершення навчання.

Після імпорту результати проходять етап нормалізації та бінаризації. Поряд із сирими балами система формує бінарний показник правильності відповіді: відповідь вважається правильною у випадку отримання студентом

не менше 0.99 від максимального балу за завдання, що дозволяє компенсувати можливі похибки округлення у CSV-файлах Moodle. Одночасне збереження сирих балів і бінаризованих значень забезпечує сумісність із різними методами психометричного аналізу: модулі CTT та IRT використовують нормалізовані числові значення, тоді як окремі етапи KST-аналізу працюють із бінарною матрицею відповідей. Для кожної відповіді також зберігається орієнтовний час виконання завдання.

Ключовим компонентом підсистеми є функція `get_response_matrix()` у файлі queries.py, яка формує вхідні дані для математичного ядра системи. Результатом її роботи є матриця відповідей студентів розміром n_students × n_items, список ідентифікаторів студентів, список ідентифікаторів завдань та вектор максимальних балів — що відповідає матриці $U = (u_{ij})$, визначеній у математичній моделі підрозділу 2.4. Формування матриці включає об'єднання результатів тестових сесій, усунення дублікатів відповідей, нормалізацію значень та вирівнювання порядку завдань. Сформована матриця виступає єдиним джерелом вхідних даних для математичного ядра системи та забезпечує уніфікований формат представлення результатів незалежно від початкового формату імпортованих даних. Ключові фрагменти програмного коду функції `get_response_matrix()` наведено у додатку В (В.1).

Взаємодія користувача з модулем здійснюється через REST API. Система підтримує окремі ендпоінти для імпорту тестових завдань, імпорту результатів тестування, створення тестових сесій та додавання нових результатів до існуючих тестів. Перед обробкою виконується базова валідація структури файлів і перевірка коректності даних. Тимчасові файли, створені під час імпорту, автоматично видаляються після завершення обробки незалежно від її результату, що запобігає накопиченню службових даних на сервері.

Реалізований модуль забезпечує практичну придатність системи для використання у реальному освітньому середовищі завдяки підтримці кількох форматів тестових даних та гнучким стратегіям обробки результатів.

3.3 Реалізація математичного ядра для аналізу тестових завдань на основі моделей IRT та KST

Математичне ядро системи реалізовано у директорії core/ і є центральним обчислювальним компонентом програмного додатку. Його основним призначенням є психометричний аналіз тестових завдань, оцінювання якості банку тестів та моделювання структури знань студентів на основі поєднання класичної теорії тестів (CTT), двопараметричної моделі теорії латентних рис (IRT) та теорії простору знань (KST).

Архітектура ядра побудована за модульним принципом і включає чотири взаємопов'язані компоненти: `CTTAnalyzer`, `IRTAnalyzer`, `KSTAnalyzer` та `QualityAnalyzer`. Модель CTT використовується для статистичного оцінювання якості тестових завдань на основі емпіричних даних, модель IRT — для оцінювання латентних параметрів завдань і рівня підготовки студентів, а модель KST — для перевірки структурної узгодженості тестового банку з когнітивною моделлю навчального курсу. Інтеграція результатів усіх трьох моделей виконується компонентом `QualityAnalyzer`, який формує єдиний інтегральний показник якості завдання Fi.

Клас `CTTAnalyzer`, реалізований у файлі core/ctt.py, забезпечує розрахунок базових психометричних характеристик тестових завдань. Вхідними даними є матриця відповідей студентів та вектор максимальних балів для кожного завдання. Перед виконанням обчислень матриця нормалізується до інтервалу [0; 1] шляхом поділу кожного стовпця на відповідний максимальний бал, що дозволяє коректно аналізувати як бінарні завдання, так і завдання з частковими балами. Складність завдань оцінюється через показник P-value відповідно до формули (2.1), дискримінаційна здатність — через `corrected point-biserial correlation` відповідно до формули (2.2). На відміну від стандартної кореляції item-total, загальний бал тесту обчислюється без урахування поточного завдання, що усуває ефект штучного завищення кореляції `item-total bias`; розрахунок

виконується через функцію `scipy.stats.pearsonr`. Ключові фрагменти програмного коду класу `CTTAnalyzer` наведено у додатку Б (Б.1).

Внутрішня узгодженість тесту оцінюється через коефіцієнт Кронбаха α відповідно до формули (2.3). Реалізація коректно обробляє від'ємні значення α, які виникають коли завдання вимірюють різні конструкти, та нульову дисперсію загального балу. Додатково реалізовано показник `Cronbach's Alpha if Deleted`, який визначає зміну загальної надійності тесту при вилученні кожного окремого завдання: якщо видалення завдання підвищує α більш ніж на 0.02, це свідчить про його неконсистентність із рештою банку, що враховується при розрахунку показника Fi. Точність вимірювання оцінюється через стандартну похибку вимірювання:

$$SEM = SD\sqrt{\{1-\alpha\}} \tag{3.1}$$

При від'ємному значенні α значення SEM перевищує стандартне відхилення, що сигналізує про низьку якість тесту як вимірювального інструменту.

Клас `IRTAnalyzer`, реалізований у файлі core/irt.py, забезпечує оцінювання параметрів тестових завдань та латентного рівня підготовки студентів на основі моделі 2PL відповідно до формули (2.4). Модель вимагає бінарної матриці відповідей — з цієї причини у компоненті `QualityAnalyzer` перед передачею даних до `IRTAnalyzer` виконується попередня бінаризація матриці через функцію `binarize_matrix()`. Часткові бали нормалізуються до інтервалу [0; 1] і порівнюються з порогом 0.5: відповідь вважається правильною якщо студент набрав не менше 50% від максимального балу. Таке перетворення необхідне оскільки модель 2PL математично визначена лише для дихотомічних змінних — при передачі неперервних значень балів параметри a і b втратили б своє психометричне тлумачення. Ключові фрагменти програмного коду класу `IRTAnalyzer` наведено у додатку Б (Б.2).

Оцінювання параметрів завдань реалізовано методом MAP (Maximum A Posteriori), що поєднує максимізацію правдоподібності з байєсівською

регуляризацією через апріорний нормальний розподіл. Апріорні параметри встановлено як $\mu_a = 1.0$, $\sigma_a = 0.5$ для дискримінації та $\mu_b = 0.0$, $\sigma_b = 1.0$ для складності. Використання MAP замість класичного MLE обумовлено проблемою розбіжності оцінок у граничних випадках: при P = 1.0 метод максимальної правдоподібності дає $a \to \infty$ та $b \to -\infty$, що унеможливлює подальші обчислення. Для оптимізації використано алгоритм L-BFGS-B з обмеженнями $a \in [0.1; 4.0]$ та $b \in [-4.0; 4.0]$.

Оцінювання рівня підготовки студентів θ реалізовано методом максимальної правдоподібності відповідно до формули (2.6) при фіксованих параметрах завдань. Початкова оцінка θ обчислюється через logit-трансформацію середнього балу студента; для студентів із нульовим балом присвоюється $\theta = -3.0$, для студентів із повним балом — $\theta = 3.0$, що запобігає числовій нестійкості під час оптимізації.

Спільне оцінювання параметрів завдань і рівнів підготовки реалізовано у методі `fit()` на основі узагальненого EM-алгоритму (GEM). На E-кроці оцінюються параметри завдань при фіксованих значеннях θ методом MAP, на M-кроці уточнюються значення θ при фіксованих параметрах завдань методом MLE. Максимальна кількість ітерацій встановлена на рівні 20, критерієм завершення є досягнення порогу збіжності 10^{-4} за максимальним відхиленням параметрів між суміжними ітераціями. Константні завдання, для яких P-value дорівнює 0 або 1, виключаються з процедури оцінювання та отримують граничні значення параметрів, що позначається маскою `constant_mask`. Для оцінювання інформативності тестових завдань реалізовано інформаційну функцію відповідно до формули (2.5); метод `average_information()` обчислює середню інформативність кожного завдання шляхом усереднення значень по рівномірній сітці з 50 точок на інтервалі $\theta \in [-4; 4]$. Оцінені параметри a і b для кожного завдання зберігаються у базі даних і відображаються у таблиці результатів на сторінці аналізу, а їх вплив на поведінку завдання наочно простежується через ICC-криві (підрозділ 3.5).

Клас `KSTAnalyzer`, реалізований у файлі core/kst.py, забезпечує моделювання когнітивної структури навчального курсу та перевірку структурної коректності тестових завдань. Для представлення структури знань використовується орієнтований ациклічний граф `networkx.DiGraph`, у якому вершини відповідають концептам курсу, а орієнтовані ребра — передумовним залежностям між ними відповідно до моделі G = (C, E) з формули (2.9). Метод `load_from_db()` завантажує з бази даних концепти з таблиці Concepts, зв'язки передумов з таблиці ConceptPrerequisites та відповідність між завданнями і концептами; на основі цих даних формується граф знань і словник coverage, що відображає покриття концептів тестовими завданнями. Ключові фрагменти програмного коду класу `KSTAnalyzer` наведено у додатку Б (Б.3).

Коректність структури перевіряється методом `is_valid_dag()` через функцію `networkx.is_directed_acyclic_graph()`. Ациклічність є необхідною умовою коректності моделі KST: наявність циклів означає логічне протиріччя у визначенні залежностей між темами курсу. Метод `check_structural_correctness()` оцінює структурну коректність завдання за трьома послідовними умовами: завдання прив'язане до концепту, відповідний концепт присутній у графі, а всі предки концепту мають достатнє покриття завданнями в банку. Якщо хоча б одна умова не виконується, завдання отримує оцінку структурної коректності 0. Метод `get_kst_scores()` обчислює KST-оцінку кожного завдання як середнє арифметичне між показником структурної коректності (0 або 1) та коефіцієнтом покриття концепту — відношенням кількості завдань для концепту до мінімально достатнього значення 3, обмеженим зверху 1.0. Стан знань студента визначається методом `get_student_knowledge_state()` як множина концептів, для яких студент правильно виконав не менше 50% пов'язаних завдань, що відповідає формальному визначенню стану знань $K_j \subseteq Q$ з формули (2.10).

Клас `QualityAnalyzer`, реалізований у файлі core/quality.py, є головним оркестратором математичного ядра та інтегрує результати CTT, IRT і KST аналізу в єдиний показник Fi відповідно до формули (2.8). Метод compute_fi() реалізує послідовний конвеєр обчислень. На першому кроці визначаються адаптивні ваги залежно від розміру вибірки: при $n < 100$ вага CTT підвищується до 0.50, вага IRT знижується до 0.20, вага KST залишається на рівні 0.30; при $n \geq 100$ використовуються стандартні ваги $w_{CTT} = 0.40$, $w_{IRT} = 0.30$, $w_{KST} = 0.30$. При вибірці менше 100 студентів оцінки параметрів a і b стають нестійкими, тому вплив IRT знижується на користь більш стабільних CTT-характеристик. На другому кроці матриця відповідей бінаризується для передачі до `IRTAnalyzer`, на третьому — послідовно виконуються CTT, IRT та KST аналіз. Ключові фрагменти програмного коду класу `QualityAnalyzer` наведено у додатку Б (Б.4).

Нормалізація CTT-показників реалізована у методі `_normalize_ctt()` за кусково-лінійними функціями: P-value у діапазоні [0.50; 0.95] отримує максимальний бал 1.0, значення $P \geq 0.95$ штрафуються за надмірну легкість, $P \in [0.20; 0.50)$ отримують лінійно зростаючу оцінку, $P < 0.20$ — оцінку 0.0. D-index оцінюється за аналогічною схемою з явним штрафом 0.0 для від'ємних значень. Alpha-оцінка встановлюється 0.0 для завдань видалення яких підвищує загальний α більш ніж на 0.02, та 1.0 для завдань видалення яких знижує α. Зважена CTT-оцінка обчислюється як $0.35 \cdot P\_score + 0.45 \cdot D\_score + 0.20 \cdot Alpha\_score$. Нормалізація IRT-показників реалізована у методі `_normalize_irt()`: параметр a оцінюється максимально у діапазоні [1.0; 2.5], параметр b — у діапазоні [−1.0; 1.0]; за межами цих діапазонів оцінка лінійно знижується. Константні завдання отримують IRT-оцінку 0.0, середня інформативність нормалізується відносно максимального значення серед неконстантних завдань.

Класифікація завдань здійснюється методом `classify()` за трьома порогами: $Fi \geq 0.70$ — статус active, $0.40 \leq Fi < 0.70$ — статус review, $Fi < 0.40$ — статус remove. Метод `save_to_db()` зберігає всі розраховані параметри

у таблицю Tasks бази даних — P-value, D-index, параметри IRT (a, b), середню інформативність, результат KST-перевірки, інтегральний показник Fi та статус завдання — і одночасно оновлює значення θ для всіх студентів у таблиці Students. Використання кешу `_kst_scores_cache` запобігає повторному обчисленню KST-оцінок під час формування звіту після збереження результатів.

3.4 Реалізація AI-модуля аналізу та генерації тестових завдань

AI-модуль системи реалізований у двох взаємодоповнюючих компонентах: класі `AIAdvisor` у файлі ai/advisor.py, що забезпечує аналіз якості існуючих завдань та генерацію нових, та ендпоінтах маршрутизатора tasks.py, що реалізують взаємодію з великою мовною моделлю для покращення конкретних завдань і формування зведеного звіту по банку. Інтеграція з AI-сервісом здійснюється через Groq API з використанням моделі llama-3.3-70b-versatile, що забезпечує генерацію відповідей з низькою затримкою, необхідною для інтерактивної роботи викладача з інтерфейсом системи.

Клас `AIAdvisor` реалізує п'ять методів, кожен з яких вирішує окрему аналітичну задачу. Метод `analyze_task()` формує структурований промпт із психометричним контекстом завдання — значеннями Fi, P-value, D-index та рівнем таксономії Блума — і генерує рекомендацію у форматі ПРОБЛЕМА / РЕКОМЕНДАЦІЯ / ПРІОРИТЕТ. Метод `classify_bloom()` визначає рівень таксономії Блума шляхом класифікації на шість рівнів і повертає результат у форматі JSON. Метод `analyze_distractors()` оцінює якість неправильних варіантів відповідей у завданнях типу MCQ та надає рекомендації щодо їх покращення. Метод `generate_tasks()` генерує нові тестові завдання для заданого концепту та рівня таксономії Блума у форматі JSON з варіантами відповідей. Метод `detect_concept()` автоматично визначає тематичну належність завдання шляхом зіставлення тексту з переліком концептів із бази даних. Для

пакетного оновлення банку реалізовано метод `analyze_weak_tasks()`, який послідовно аналізує всі завдання з Fi < 0.70 та зберігає рекомендації у полі ai_recommendation таблиці Tasks, що забезпечує їх постійну доступність через API без повторних викликів AI-сервісу. Ключові фрагменти програмного коду класу AIAdvisor наведено у додатку В (В.2).

Центральним елементом AI-модуля є ендпоінт POST /api/tasks/{task_id}/improve, що реалізує адаптивну генерацію замінного завдання з урахуванням психометричного профілю існуючого. Логіка ендпоінту охоплює чотири послідовні етапи, схему яких наведено на рисунку 3.4.2. На першому етапі з бази даних витягується повний контекст завдання: текст, значення P-value та D-index, Fi-score, концепт та варіанти відповідей. На другому етапі автоматично визначаються психометричні проблеми на основі порогових значень: P-value > 0.70 інтерпретується як надмірна легкість, P-value < 0.30 — як надмірна складність, D-index < 0.20 — як недостатня дискримінативність. На третьому етапі формується промпт із зазначенням виявлених проблем та цільових характеристик нового завдання: P-value ∈ [0.40; 0.65] та D-index > 0.30. На четвертому етапі відповідь моделі валідується як коректний JSON та повертається клієнтській частині для підтвердження заміни викладачем.

Система підтримує сім типів завдань для генерації, кожен з яких має власний структурований промпт із точним JSON-форматом відповіді: MCQ (одна правильна відповідь з чотирьох), Multiple Choice (три правильних відповіді з шести), True/False, Matching (чотири пари відповідності), Fill Blank (заповнення пропусків у SQL-коді), Code Output (визначення результату SQL-запиту) та Find Error (знаходження помилки у SQL-коді). Вибір типу здійснюється викладачем в інтерфейсі перед запуском генерації.

Після підтвердження викладачем згенерованого завдання виконується ендпоінт POST /api/tasks/{task_id}/replace, який атомарно виконує три операції: оригінальне завдання отримує статус removed і залишається в базі даних як архівний запис, нове завдання вставляється в таблицю Tasks зі

статусом undefined, його варіанти відповідей записуються в TaskOptions, а зв'язок з банком фіксується в BankTasks зі збереженням тих самих bank_id та test_id. Збереження архівного запису дозволяє відстежувати еволюцію банку, порівнювати психометричні характеристики до і після заміни та за потреби відновлювати попередні версії завдань.

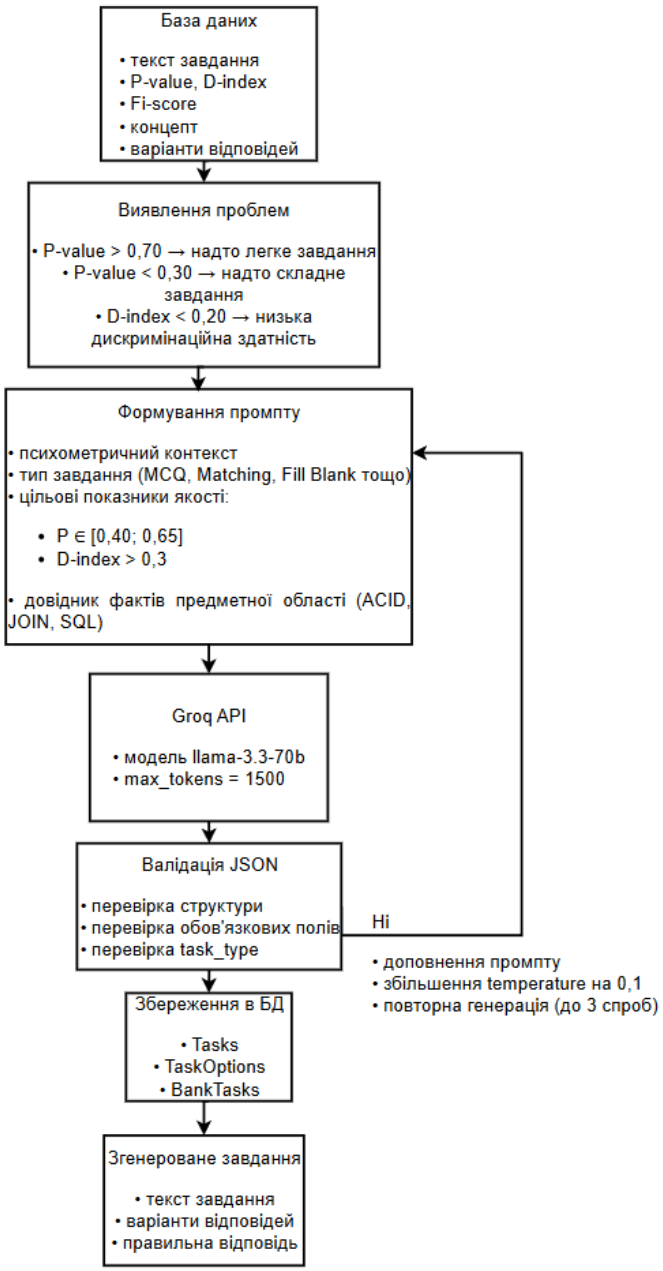


Рисунок 3.4.2 — Схема процесу генерації завдання для заміни

Завдання типів Fill Blank, Code Output, Find Error та Matching містять структуровані метадані, що виходять за межі простого тексту і переліку варіантів відповідей: SQL-код з пропусками, масив правильних відповідей,

пари відповідності або пояснення помилки. Для їх зберігання обрано підхід серіалізації у вигляді JSON-рядка, що додається до поля text таблиці Tasks через розмежувач META. Такий підхід гарантує атомарність операцій читання і запису завдання та не вимагає змін схеми бази даних при розширенні переліку підтримуваних типів.

Практика використання великих мовних моделей виявила схильність до фактичних неточностей. У контексті курсу «Бази даних» це проявляється у некоректному визначенні властивостей ACID-транзакцій, посиланнях на неіснуючі типи JOIN та помилках у синтаксисі SQL. Для вирішення цієї проблеми до системного промпту вбудовано структурований довідник верифікованих фактів (ACID, нормальні форми 1НФ–5НФ/НФБК, типи JOIN, класифікація DDL/DML/DCL), а для мінімізації варіативності встановлено temperature = 0.3 для генерації та temperature = 0.2 для AI-critique. Реалізовано цикл повторних спроб: у разі невдалої структурної валідації промпт доповнюється описом помилки і запит надсилається повторно — до трьох спроб з поступовим підвищенням temperature на 0.1.

Окрім генерації окремих завдань, AI-модуль реалізує ендпоінт POST /api/tasks/bank/{bank_id}/summary, що формує зведений аналітичний звіт по всьому банку. Якщо жодне завдання не має розрахованого Fi-score, повертається відповідь з ознакою `not_analyzed` та рекомендацією запустити аналіз, що запобігає генерації висновків на основі відсутніх даних. За наявності результатів ендпоінт агрегує розширену статистику: загальні показники банку, розподіл завдань за статусами, профіль трьох завдань з найнижчим Fi, статистику якості по кожній темі курсу та розподіл за рівнями таксономії Блума. Зібрана статистика передається моделі з вимогою включити конкретні числові значення у кожен пункт висновку: сильні сторони, проблеми, рекомендації та пріоритетна дія.

Таким чином, реалізований AI-модуль забезпечує два рівні підтримки прийняття рішень: операційний — генерація замінних завдань з верифікацією фактичної точності та структурною валідацією, та стратегічний —

формування аналітичного звіту з пріоритизованими рекомендаціями. Цикл оптимізації охоплює чотири етапи: виявлення проблемного завдання за показником Fi, генерацію замінного завдання з цільовими характеристиками, верифікацію на реальній студентській аудиторії та оновлення показників якості банку. Повторення цього циклу після кожної тестової сесії забезпечує поступове наближення банку завдань до психометричного оптимуму.

3.5 Розробка інтерфейсу користувача та візуалізації результатів аналізу

Інтерфейс користувача розроблено як Single Page Application на основі бібліотеки React 18 відповідно до архітектурних рішень, описаних у підрозділі 2.2.2. Архітектура інтерфейсу організована за принципом сторінкових компонентів у директорії src/pages/: кожна функціональна підсистема відповідає окремій сторінці з власним станом та логікою взаємодії з API. Єдина кольорова схема визначена у константі COLORS: основний колір #1e3a5f для заголовків та навігації, акцентні кольори для індикації статусів завдань — зелений #16a34a для active, жовтий #ca8a04 для review та червоний #dc2626 для remove.

Публічна частина системи реалізована у компоненті LandingPage і є точкою входу для незареєстрованих користувачів. Захищені маршрути організовані під префіксом /app/* через компонент ProtectedLayout з перевіркою JWT-токена. Система підтримує дві ролі: teacher забезпечує доступ виключно до власних банків завдань, admin надає повний доступ до всіх банків системи. Зовнішній вигляд лендінг-сторінки та сторінки автентифікації наведено у додатку Д (рисунки Д.1, Д.2).

Сторінка ImportPage є головним робочим простором для управління банками завдань. Сітка банків відображає кожен банк як картку з індикатором середнього Fi у вигляді прогрес-бару з кольоровим кодуванням. Майстер імпорту реалізований як триступеневий wizard: Крок 1 — вибір або створення банку завдань; Крок 2 — завантаження файлу завдань у форматі GIFT або

Moodle XML; Крок 3 — завантаження CSV-файлу з результатами тестування. Інтерфейс майстра імпорту наведено у додатку Д (рисунок Д.3). Вигляд сітки карток банків наведено на рисунку 3.5.3.

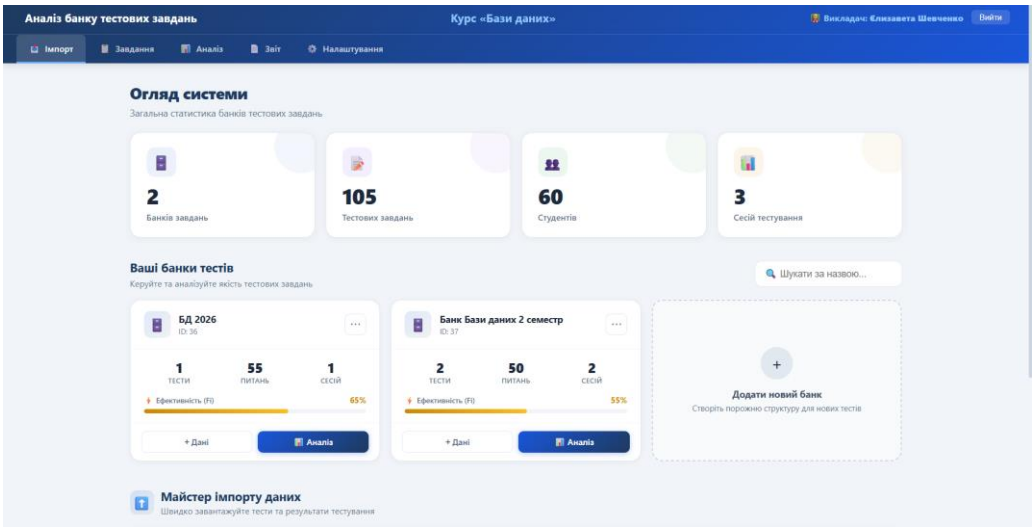


Рисунок 3.5.3 — Сторінка імпорту зі сіткою карток банків завдань та індикаторами Fi

Сторінка TasksPage реалізує повний цикл роботи з окремими завданнями банку. Верхня частина містить AI-звіт по банку із загальною оцінкою якості, ключовими метриками та пріоритетною рекомендацією (рисунок 3.5.4). Основний інтерфейс складається з двох колонок: таблиця завдань із показниками P-value, D-index, Fi-score та кольоровим маркуванням статусу (рисунок 3.5.5) та панель деталей вибраного завдання з психометричними показниками, AI-рекомендацією та блоком генерації замінного завдання (рисунок 3.5.6).

Сторінка `AnalysisPage` забезпечує запуск повного психометричного аналізу та перегляд його результатів. Блок візуалізації містить п'ять аналітичних графіків: Fi дашборд зі стовпчастим графіком та розподілом статусів (рисунок 3.5.7), ICC криві завдань відповідно до моделі 2PL (рисунок 3.5.8), CTT гістограми P-value та D-index, IRT інформаційні криві та граф знань KST з маркуванням покриття концептів (рисунок 3.5.9).

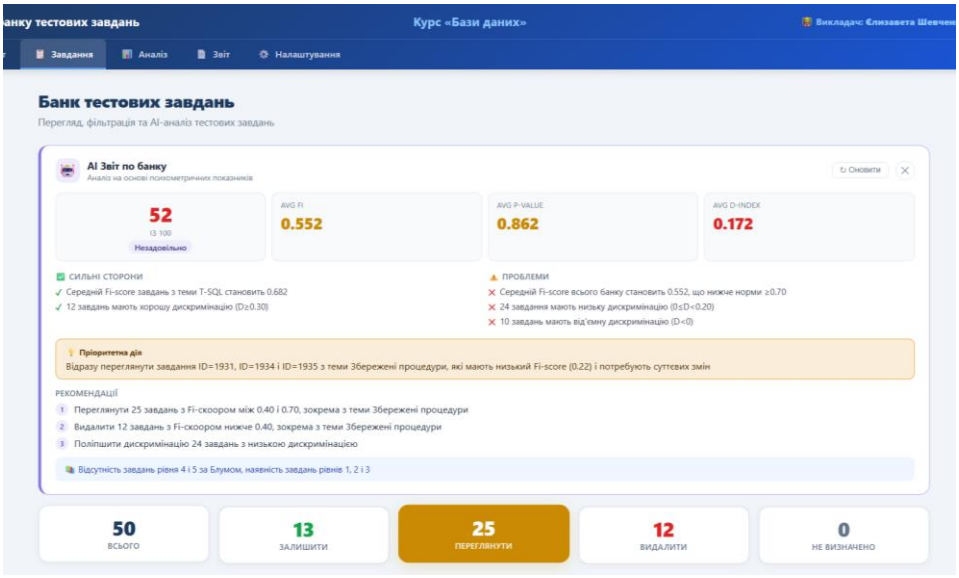


Рисунок 3.5.4 — Компонент BankSummaryBlock з AI-звітом по банку завдань

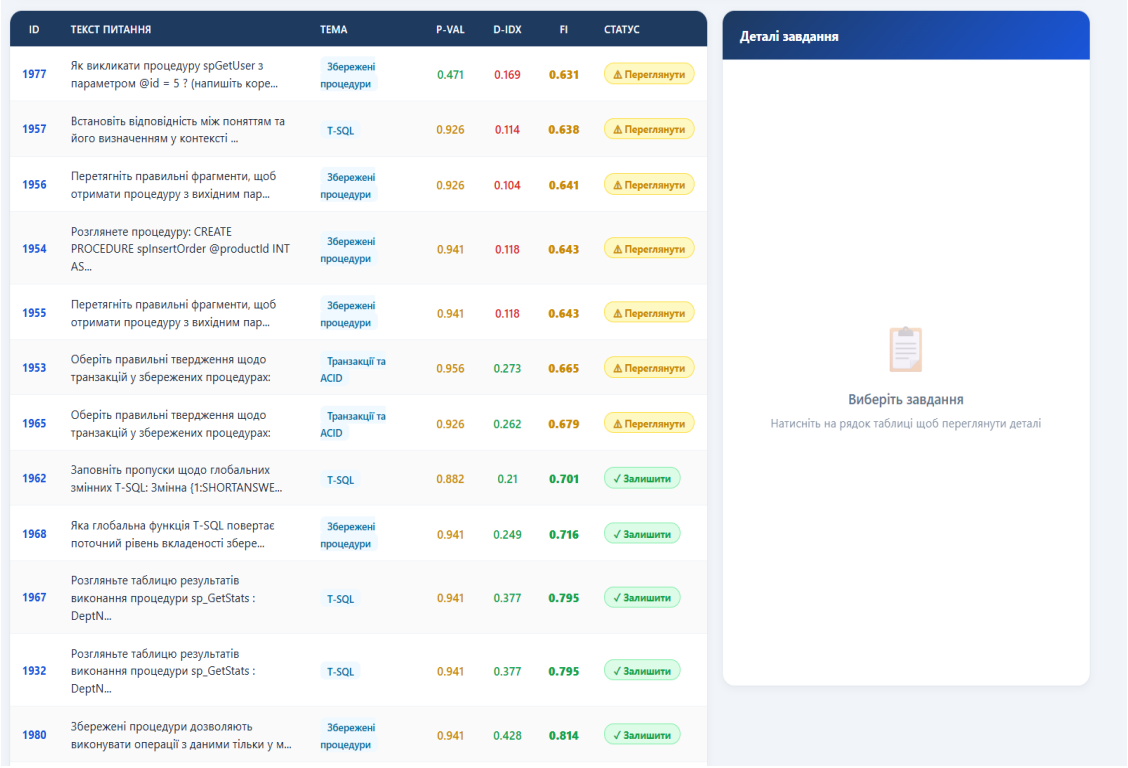


Рисунок 3.5.5 — Таблиця завдань з кольоровим маркуванням статусів

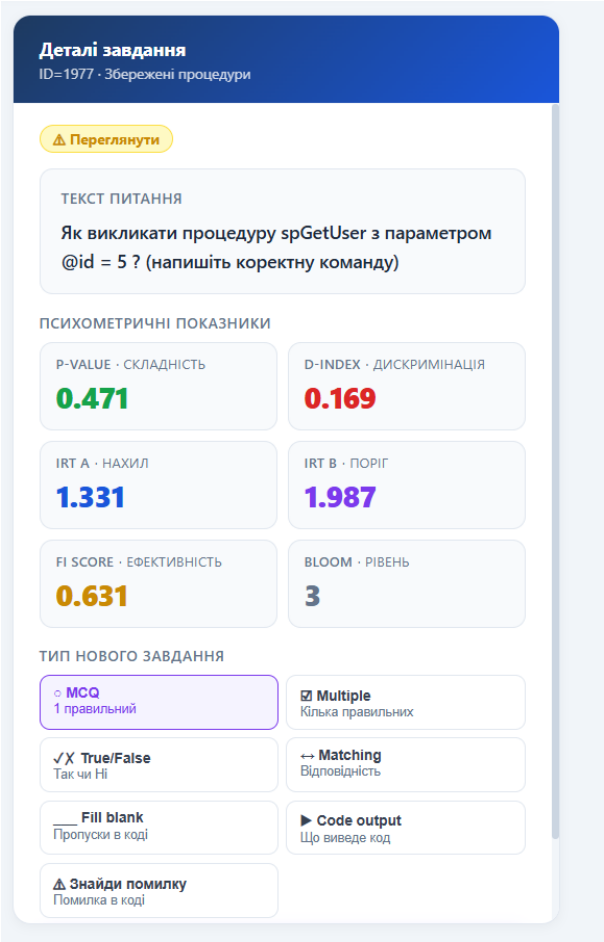


Рисунок 3.5.6 — Панель деталей завдання з психометричними показниками та результатом AI-генерації

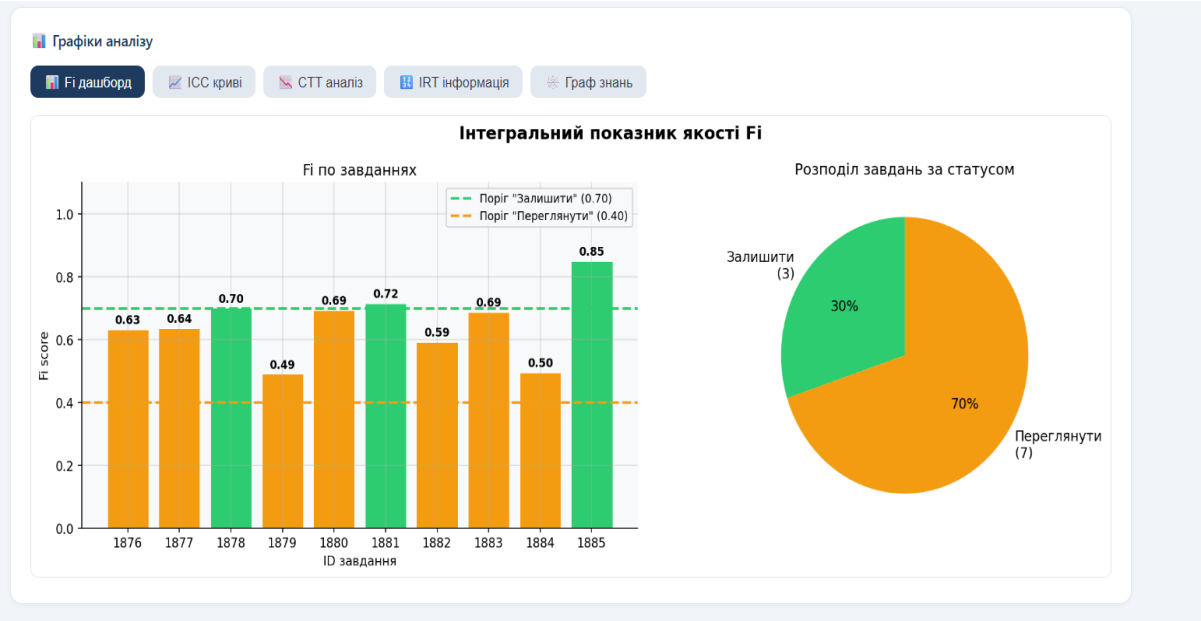


Рисунок 3.5.7 — Fi дашборд зі стовпчастим графіком та круговою діаграмою розподілу статусів

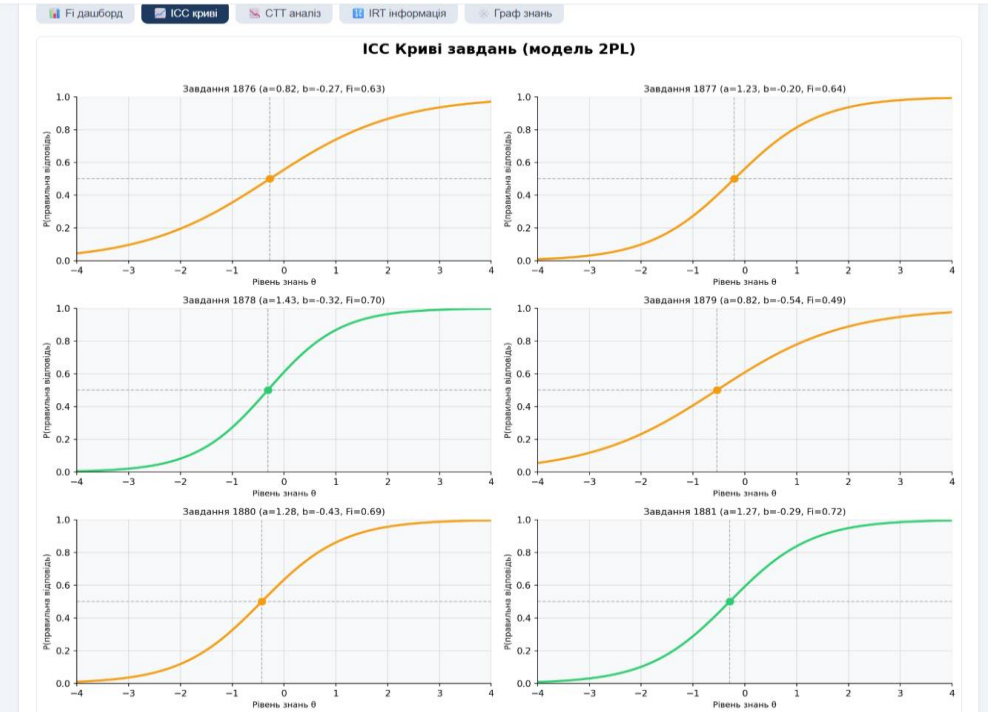


Рисунок 3.5.8 — ICC криві завдань банку відповідно до моделі 2PL

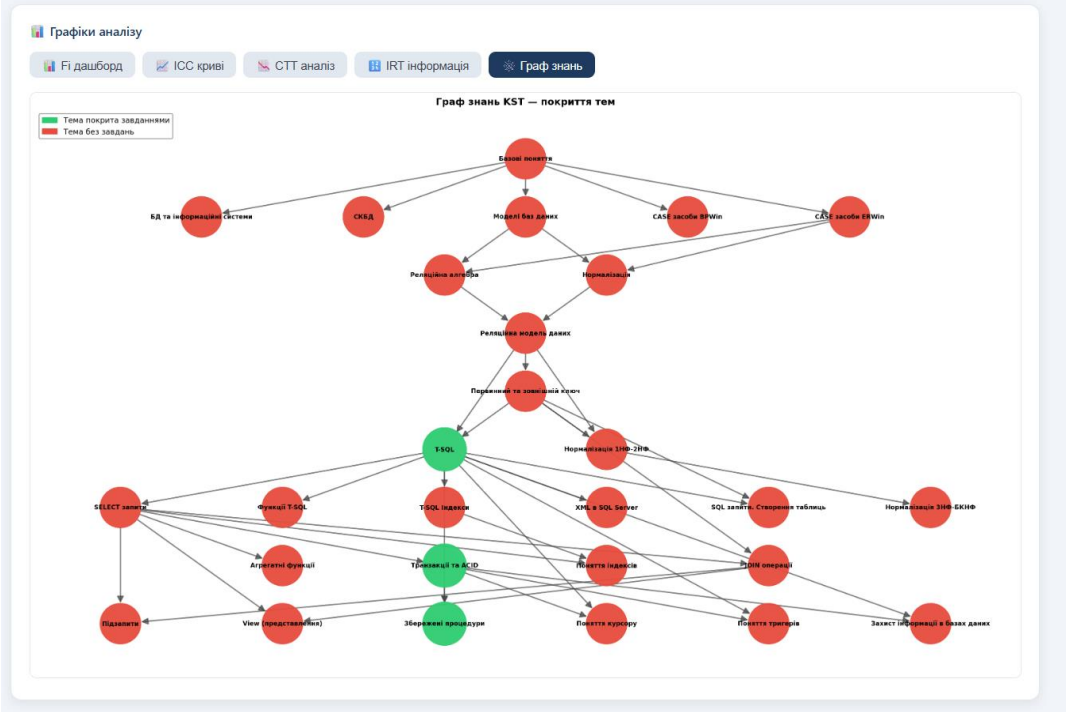


Рисунок 3.5.9 — Граф знань KST з маркуванням покриття концептів курсу

Сторінка ReportPage забезпечує генерацію звітності у форматах Excel та PDF. Сторінка SettingsPage дозволяє налаштовувати вагові коефіцієнти Fi

через слайдери з динамічною перевіркою суми — кнопка збереження блокується якщо w_CTT + w_IRT + w_KST ≠ 1.0. Інтерфейси цих сторінок наведено у додатку Д (рисунок Д4 та Д5).

3.6 Висновки за розділом 3

У третьому розділі описано практичну реалізацію програмного додатку для аналізу та оптимізації банку тестових завдань курсу «Бази даних», що охоплює всі компоненти системи — від архітектури та збору даних до математичного ядра, AI-модуля та інтерфейсу користувача.

Спроєктовано та реалізовано трирівневу клієнт-серверну архітектуру з модульним розділенням відповідальності між компонентами: директорія routers/ забезпечує маршрутизацію запитів, core/ — математичні обчислення, database/ — взаємодію з базою даних, ai/ — інтеграцію з Groq API. Реалізовано механізм кешування результатів аналізу через `_analysis_cache`, що виключає повторний запуск EM-алгоритму при формуванні звітів.

Модуль збору та обробки даних (підрозділ 3.2) забезпечує імпорт тестових завдань у форматах GIFT та Moodle XML, імпорт результатів тестування з CSV-звітів Moodle з підтримкою трьох стратегій обробки повторних спроб та автоматичне формування матриці відповідей $U = (u_{ij})$ відповідно до математичної моделі підрозділу 2.4.

Математичне ядро (підрозділ 3.3) реалізує чотири взаємопов'язані компоненти. `CTTAnalyzer` обчислює P-value, скориговану point-biserial кореляцію, коефіцієнт Кронбаха α та стандартну похибку вимірювання (формула 3.1). `IRTAnalyzer` реалізує модель 2PL з оцінюванням параметрів методом MAP та спільним калібруванням через GEM-алгоритм. `KSTAnalyzer` перевіряє ациклічність графу передумов та структурну коректність завдань. `QualityAnalyzer` інтегрує результати трьох моделей в показник Fi відповідно до формули (2.8) з адаптивними вагами залежно від розміру вибірки.

AI-модуль (підрозділ 3.4) реалізує два рівні підтримки прийняття

рішень: операційний — чотириетапна генерація замінних завдань семи типів через Groq API з верифікацією фактичної точності та структурною валідацією, та стратегічний — формування зведеного аналітичного звіту по банку. Для зниження частоти галюцинацій моделі вбудовано довідник верифікованих фактів з теорії реляційних баз даних та встановлено temperature = 0.3.

Інтерфейс користувача (підрозділ 3.5) реалізовано як SPA на основі React 18 з тринадцятьма рисунками що ілюструють ключові сторінки системи. Реалізовано триступеневий майстер імпорту, двоколонковий перегляд завдань, п'ять аналітичних графіків та генерацію звітів у форматах PDF та Excel.

Реалізований програмний додаток забезпечує повний цикл роботи з банком тестових завдань: від імпорту даних та психометричного аналізу до формування обґрунтованих рекомендацій щодо оптимізації та генерації замінних завдань. Отримані результати слугують основою для експериментальної апробації системи, що розглядається у розділі 4.

РОЗДІЛ 4 АПРОБАЦІЯ ПРОГРАМНОГО ДОДАТКУ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЙОГО РОБОТИ

4.1 Методологія тестування програмного забезпечення

Тестування програмного додатку здійснювалося на двох незалежних рівнях: автоматизоване модульне тестування математичного ядра системи та функціональне тестування повного циклу обробки даних на реальних результатах тестування студентів.

На першому рівні застосовувалося модульне тестування, метою якого є перевірка коректності ізольованих компонентів системи — модулів `CTTAnalyzer`, `IRTAnalyzer` та `KSTAnalyzer` — незалежно від зовнішніх залежностей. На другому рівні проводилося функціональне тестування, яке охоплювало повний цикл роботи додатку від імпорту даних до формування звітів та збереження результатів у базу даних. Таке розмежування дозволяє чітко локалізувати помилки: збої на рівні модульних тестів вказують на некоректність алгоритму, тоді як збої на рівні функціонального тестування — на проблеми інтеграції компонентів.

Систему тестування реалізовано без використання зовнішніх тестових фреймворків, оскільки це забезпечує повний контроль над логікою запуску тестів, форматом звітності та поведінкою при вироджених сценаріях. Власний тестовий набір розміщено у директорії tests/ проєкту. Запуск усіх тестів здійснюється скриптом run_tests.py, який послідовно виконує тести, виводить журнал виконання до консолі та формує машинозчитуваний звіт у файлі reports/test_report.txt.

Тест вважається пройденим за умови відповідності розрахованого значення очікуваному з точністю до 0,001 для числових показників або коректного завершення без аварійного збою для тестів крайніх випадків.

Модульні тести ґрунтуються на синтетичних матрицях із наперед відомими статистичними властивостями, що дозволяє однозначно перевіряти математичну коректність розрахунків. Для тестів крайніх випадків навмисно

використовуються вироджені вхідні дані — матриці з константними завданнями, одним студентом або від'ємним коефіцієнтом Кронбаха — з метою верифікації стабільності системи в нештатних ситуаціях.

Оскільки `KSTAnalyzer` у робочому режимі звертається до SQL Server для отримання графа передумов, для забезпечення ізольованого тестування логіки структурного аналізу без підключення до бази даних розроблено клас `MockKSTAnalyzer`, який імітує інтерфейс та поведінку реального модуля на основі заздалегідь визначеного тестового графа знань.

Автоматизований тестовий набір складається з трьох файлів і охоплює 36 тестів, розподілених за функціональними групами: test_ctt.py — 9 тестів модуля `CTTAnalyzer` (класична теорія тестів); test_irt.py — 10 тестів модуля `IRTAnalyzer` (двопараметрична IRT-модель та EM-алгоритм); test_kst_quality_edge.py — 17 тестів, що охоплюють логіку KST-аналізу на основі `MockKSTAnalyzer` (8 тестів), інтегральний показник якості Fi (4 тести) та крайні випадки системи (5 тестів). Детальний опис того, що саме перевіряється в кожній групі тестів, та результати їх виконання наведено у підрозділі 4.2. Результати функціонального тестування на реальних даних описано у підрозділі 4.3.

4.2 Перевірка коректності реалізованих алгоритмів оцінювання

Перевірка коректності алгоритмів здійснювалася шляхом виконання 36 автоматизованих модульних тестів, розподілених за чотирма функціональними групами: CTT, IRT, KST та інтегральний показник Fi. Додатково перевірялася стабільність системи у п'яти сценаріях крайніх випадків. За результатами запуску усі 36 тестів пройдено успішно (100%). Зведені результати виконання тестового набору наведено в таблиці 4.2.1.

Таблиця 4.2.1 — Зведені результати виконання автоматизованого тестового набору

Модуль	**Кількість тестів**	**Пройдено**	**Час виконання**
CTTAnalyzer	9	9 (100%)	< 12 мс
IRTAnalyzer	10	10 (100%)	~9 с
KSTAnalyzer + MockKSTAnalyzer	8	8 (100%)	< 2 мс
Інтегральний показник Fi	4	4 (100%)	< 50 мс
Крайні випадки	5	5 (100%)	< 50 мс
Всього	**36**	**36 (100%)**	**~9 с**

Усі 9 тестів модуля `CTTAnalyzer` виконалися менш ніж за 12 мс сумарно, що обумовлено прямим характером статистичних обчислень без ітеративної оптимізації. Точність розрахунку індексу складності перевірялася на синтетичній матриці 20×5, де кожен стовпець містить заздалегідь визначену частку правильних відповідей: 1,0; 0,75; 0,50; 0,25 та 0,0. Тест `test_p_values_known` підтвердив, що розраховані P-value відповідають очікуваним із похибкою, що не перевищує 0,001. Тест `test_p_values_range` верифікував належність усіх значень діапазону [0, 1] на випадковій матриці 50×10.

Коректність індексу дискримінації перевірялася трьома тестами. Тест `test_discrimination_positive_for_good_items` підтвердив, що завдання, які чітко розділяють сильних і слабких студентів, отримують додатний D-index. Тест `test_discrimination_constant_item_zero` верифікував, що константне завдання (P=1,0) отримує D-index, близький до нуля. Тест `test_discrimination_corrected_lower_than_naive`

підтвердив ключову властивість скоригованого point-biserial: середнє значення corrected D-index є меншим або рівним нескоригованому варіанту, що свідчить про усунення item-total bias.

Коефіцієнт Кронбаха перевірявся у трьох сценаріях. Для узгодженого тесту (матриця make_discriminating_matrix) отримано alpha=1,000, що відповідає повній внутрішній консистентності. Для неузгодженого тесту, де завдання вимірюють різні конструкти, система коректно повертає від'ємне значення без аварійного завершення. При одному завданні alpha повертає 0,0 відповідно до визначення коефіцієнта. Окремо верифіковано нормалізацію часткових балів: для матриці з max_scores=[2,0; 2,0] розрахований P-value першого завдання склав 0,625, що відповідає очікуваному результату.

Тести модуля IRTAnalyzer є найбільш обчислювально затратними через використання ітераційної EM-процедури оцінювання параметрів 2PL-моделі. Сумарний час виконання 10 тестів склав близько 9 секунд; окремі тести виконувалися від десятків мілісекунд до кількох секунд залежно від кількості ітерацій та розміру матриці. Основні тести використовують синтетичну матрицю 130×6, що відповідає рекомендованому мінімуму для стабільного оцінювання параметрів 2PL-моделі, із завданнями впорядкованими від дуже легких до дуже важких з P-value: 0,95; 0,75; 0,50; 0,50; 0,25; 0,05.

Тест test_item_params_bounds підтвердив, що після виконання fit() усі параметри дискримінації та складності залишаються в межах $a \in [0{,}1;\ 4{,}0]$ та $b \in [-4{,}0;\ 4{,}0]$. Отримані значення параметрів після 20 ітерацій наведено в таблиці 4.2.2.

Таблиця 4.2.2 — Оцінені параметри IRT-моделі для синтетичної матриці 130×6

Завдання	P-value	a	b
0	≈0,95	2,046	–3,437
1	≈0,75	2,460	–1,612
2	≈0,50	2,590	–0,003
3	≈0,50	0,100	+0,003
4	≈0,25	2,459	+1,577
5	≈0,05	2,070	+3,378

Тест `test_difficulty_ordering` підтвердив монотонний зв'язок між P-value та параметром b: b[0]=–3,437 < b[5]=3,378, що відповідає очікуваній властивості моделі. Коректність характеристичних кривих завдань перевірялася двома тестами: `test_icc_monotone` підтвердив монотонне зростання функції P(θ) при додатному параметрі дискримінації для всіх неконстантних завдань, а `test_icc_p_at_b_equals_05` верифікував фундаментальну математичну властивість 2PL-моделі — при θ=b ймовірність правильної відповіді дорівнює 0,5 із похибкою менше 0,001. Відповідно на рисунку 4.2.1 можемо побачити результати прооведення тестів.

Збіжність EM-процедури перевірялася тестом `test_em_convergence` шляхом порівняння параметрів після 10 та 30 ітерацій. Максимальна різниця склала `delta_a`=0,0179 та `delta_b`=0,0394, що підтверджує практичну стабільність оцінок. Динаміку зменшення `max_delta` протягом ітерацій та повний звіт з виконанням тестів наведено у додатку Г (лістинг Г.1).

У процесі тестування для більшості запусків алгоритм досягав встановленого обмеження `max_iter` = 20 раніше, ніж формально виконувався критерій зупинки `tolerance`=10^{-4}, і система виводила відповідне попередження. Водночас аналіз динаміки max_delta показує стале зменшення від 1,717 на першій ітерації до 0,000543 на двадцятій, а тести

математичних властивостей — монотонність ICC, P(θ=b)=0,5, впорядкованість b — пройшли успішно. Це свідчить про практичну стабільність оцінених параметрів навіть за відсутності формального досягнення критерію зупинки.

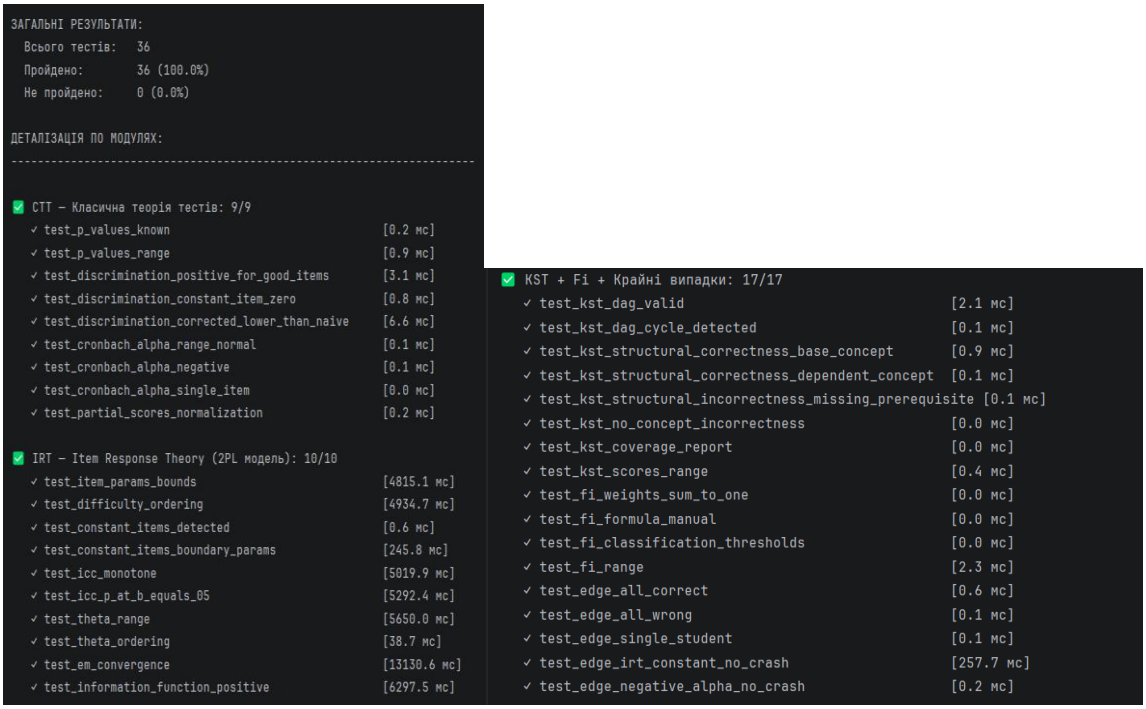


Рисунок 4.2.1 — Результати виконання автоматизованого тестового набору (36/36 тестів пройдено успішно)

Оцінки латентних здібностей θ студентів верифіковані двома тестами. Тест `test_theta_range` підтвердив належність усіх значень діапазону [–4,0; 4,0]. Тест `test_theta_ordering` підтвердив коректність впорядкування: студент із усіма правильними відповідями отримує theta = 3,000, студент без жодної правильної відповідно theta = –3,000. Невід'ємність інформаційної функції $I(\theta)=a^2 \cdot P(\theta) \cdot (1-P(\theta))$ верифіковано тестом `test_information_function_positive` для 50 рівномірно розподілених точок $\theta \in [-4; 4]$.

Оскільки реальний `KSTAnalyzer` залежить від підключення до SQL Server, тестування виконувалося на класі `MockKSTAnalyzer` із тестовим графом знань: концепт SELECT є передумовою для JOIN, а JOIN — для

Subquery. Усі 8 тестів виконалися менш ніж за 2 мс кожен. Тест `test_kst_dag_valid` підтвердив коректне визначення DAG-властивості для графа без циклів. Тест `test_kst_dag_cycle_detected` верифікував виявлення циклу при додаванні зворотного ребра 3→1 . Структурна коректність завдань перевірялася трьома тестами: завдання до базового концепту (без передумов) є коректним; завдання до залежного концепту є коректним лише за умови покриття його передумови; при видаленні покриття передумови завдання стає некоректним. Тест `test_kst_no_concept_incorrectness` підтвердив, що завдання без прив'язаного концепту отримує статус некоректного. Тест `test_kst_coverage_report` верифікував коректність звіту покриття: при двох покритих концептах із трьох `coverage_ratio`=0,667. Тест `test_kst_scores_range` підтвердив належність усіх KST-оцінок діапазону [0, 1]: отримані значення склали [0,667; 0,667; 0,667; 0,000] для завдань 101–104 відповідно.

Чотири тести верифікували коректність розрахунку Fi = 0,40·CTT + 0,30·IRT + 0,30·KST. Тест `test_fi_weights_sum_to_one` підтвердив, що суми вагових коефіцієнтів дорівнюють 1,0 як у стандартному режимі (0,40+0,30+0,30), так і в адаптивному для малих вибірок (0,50+0,20+0,30). Тест `test_fi_formula_manual` верифікував ручний розрахунок: при CTT=0,80, IRT=0,70, KST=0,60 отримано Fi=0,710, що відповідає очікуваному. Тест `test_fi_classification_thresholds` підтвердив коректність класифікації за порогами: F i≥ 0,70 — статус «active», 0,40 ≤ Fi < 0,70 — «review», Fi < 0,40 — «remove», включаючи граничні значення 0,70 та 0,40. Тест `test_fi_range` верифікував належність результату діапазону [0, 1] для 100 випадкових комбінацій нормалізованих компонент.

П'ять тестів перевіряли стабільність системи у вироджених ситуаціях. У всіх сценаріях система завершувала роботу без аварійного збою та повертала коректні значення. При матриці з усіма правильними відповідями (P=1,0)

розраховано alpha=0,000, що є математично очікуваним результатом через відсутність дисперсії. При матриці з усіма неправильними відповідями P-value=0,0 для всіх завдань. При матриці з одним студентом система коректно повертає вектор P-value без помилки ділення на нуль. IRT-аналіз із константними завданнями (P=0 та P=1) завершився без збою завдяки MAP-регуляризації. Від'ємний коефіцієнт Кронбаха повертається як коректне від'ємне значення.

4.3 Експериментальна оцінка ефективності оптимізації банку тестових завдань

Функціональне тестування системи проводилося на реальних даних курсу «Бази даних» Харківського національного університету імені В. Н. Каразіна. Вхідними даними слугував банк із 25 тестових завдань із теми «Збережені процедури та T-SQL» та матриця відповідей 34 студентів, імпортована у форматі Moodle CSV. Завдання охоплювали три підтеми: збережені процедури (22 завдання), T-SQL (2 завдання) та транзакції й ACID (1 завдання). Типи завдань включали питання з одиночним та множинним вибором, завдання на встановлення відповідності, заповнення пропусків і впорядкування фрагментів коду.

Експеримент складався з двох етапів. На першому етапі система виконала психометричний аналіз вихідного банку завдань та сформувала рекомендації щодо його оптимізації. На другому етапі на основі цих рекомендацій було замінено 12 із 25 завдань на нові, згенеровані за допомогою інтегрованого ШІ-модуля, після чого оновлений тест був апробований на іншій групі студентів (34 особи). Метою експерименту була верифікація двох гіпотез: здатності системи коректно діагностувати якість реального банку завдань та здатності ШІ-генерації продукувати психометрично якісні завдання, придатні для реального використання.

Після імпорту даних система автоматично виконала повний психометричний аналіз за моделями CTT та IRT і розрахувала інтегральний

показник якості Fi для кожного завдання. Зведені результати первинного аналізу наведено в таблиці 4.3.3.

Таблиця 4.3.3 — Результати психометричного аналізу вихідного банку тестових завдань

Показник	**Значення**
Кількість завдань	25
Кількість студентів	34
Середній Fi	0,327
Середній P-value	0,875
Середній D-index	0,023
Завдань «Видалити» (Fi < 0,40)	16 (64%)
Завдань «Переглянути» (0,40 ≤ Fi < 0,70)	9 (36%)
Завдань «Залишити» (Fi ≥ 0,70)	0 (0%)

Загальна оцінка якості банку системою склала 33 із 100 балів, що відповідає статусу «Незадовільно». Жодне завдання не досягло порогу якості Fi ≥ 0,70, необхідного для статусу «Залишити». Результати аналізу по кожному завданню вихідного банку наведено на рисунку 4.3.2.

Найбільш поширеною проблемою виявилася критично висока легкість завдань: 10 із 25 завдань отримали P-value = 1,0, тобто всі 34 студенти відповіли на них правильно. З психометричної точки зору такі завдання не несуть вимірювальної інформації — вони не диференціюють студентів за рівнем знань і не впливають на оцінку латентної здібності θ. Для цих завдань IRT-модель встановила граничні параметри a = 4,0, b = –4,0, що відповідає «нескінченно легкому» завданню в моделі 2PL. Інтегральний показник Fi для всіх десяти склав 0,15 — мінімально можливе значення в системі.

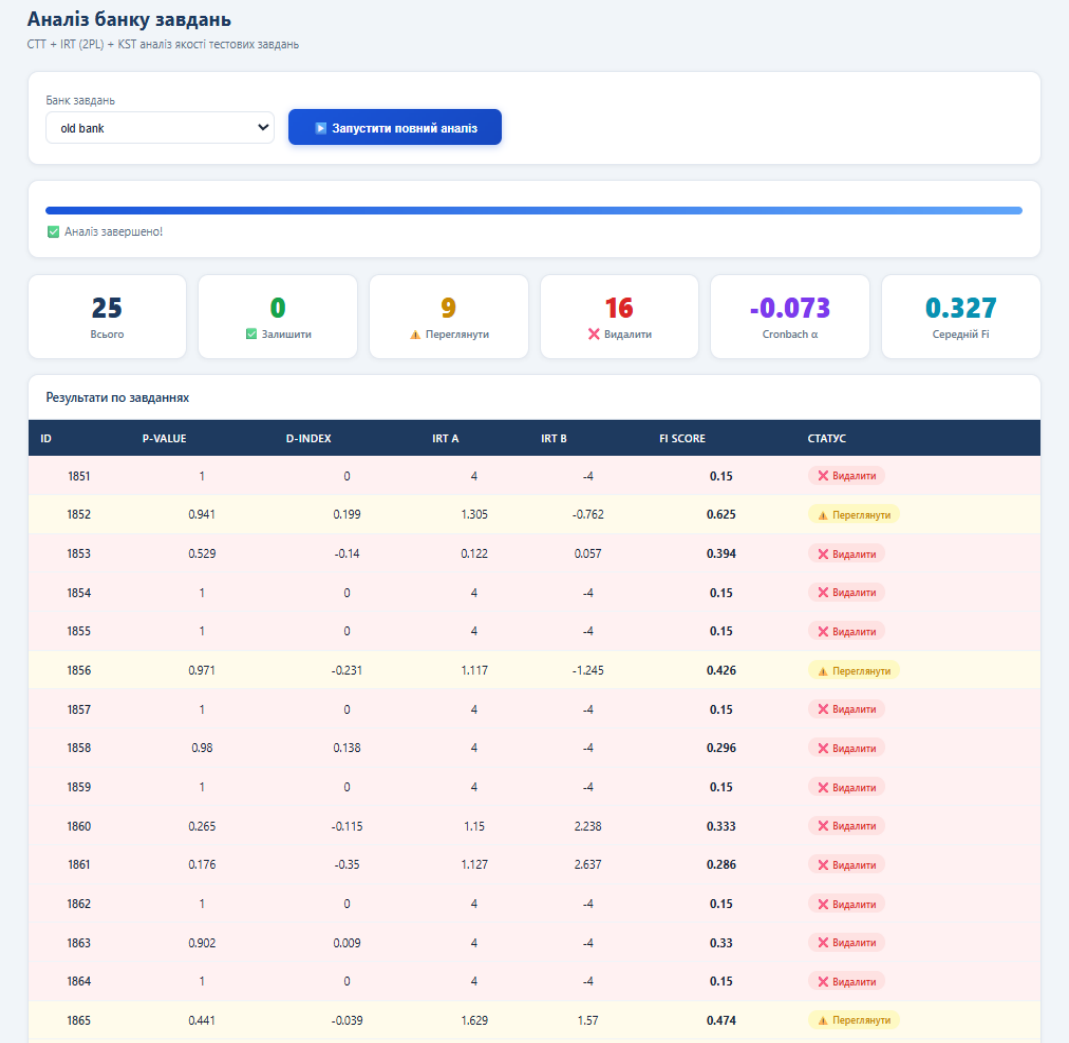


Рисунок 4.3.2 — Результати психометричного аналізу вихідного банку завдань у інтерфейсі системи

Сім завдань отримали від'ємний D-index, що є серйозним психометричним дефектом: сильніші студенти відповідають на ці завдання гірше, ніж слабкі. Найбільш виражений випадок — завдання з D-index = –0,35 та P-value = 0,176, яке одночасно є складним і антидискримінуючим, що вказує на некоректне формулювання або неоднозначність правильної відповіді. Середній D-index по банку склав 0,023 — значення, що практично не відрізняється від нуля, що означає відсутність надійної диференціації студентів за рівнем знань.

На основі рекомендацій системи було замінено 12 із 25 завдань зі статусом «Видалити», тоді як 13 завдань з найвищими значеннями Fi

залишилися без змін. Нові завдання були згенеровані ШІ-модулем з урахуванням виявлених психометричних дефектів і цільових показників P-value 0,40–0,85 та D-index > 0,20. Для покращення структури тесту також було змінено порядок розташування завдань.

Оновлений тест був апробований на іншій групі студентів того самого курсу (34 особи). Через використання незалежної вибірки пряме порівняння показників окремих завдань між версіями тесту не проводилося. Основним критерієм оцінки ефективності оптимізації стала психометрична якість нових згенерованих завдань, підтверджена на реальній вибірці. Зведені результати аналізу наведено в таблиці 4.4.4.

Таблиця 4.4.4 — Порівняльні результати психометричного аналізу банків завдань

Показник	**Вихідний банк**	**Оптимізований банк**
Кількість завдань	25	25
Кількість студентів	34	34
Середній Fi	0,327	0,670
Коефіцієнт Кронбаха α	-0,073	0,744
Завдань «Видалити» (Fi < 0,40)	16 (64%)	2 (8%)
Завдань «Переглянути» (0,40 ≤ Fi < 0,70)	9 (36%)	12 (48%)
Завдань «Залишити» (Fi ≥ 0,70)	0 (0%)	11 (44%)

Результати аналізу оптимізованого банку по кожному завданню наведено на рисунку 4.3.3.

Рисунок 4.3.3 — Результати психометричного аналізу оптимізованого банку завдань у інтерфейсі системи

Середній Fi зріс з 0,327 до 0,670, тобто на 105%. Частка завдань зі статусом «Видалити» скоротилася з 64% до 8%, а завдання зі статусом «Залишити», відсутні у вихідному банку, склали 44% оновленого. Коефіцієнт Кронбаха α = 0,744 свідчить про прийнятну внутрішню узгодженість тесту.

Ключовим результатом експерименту є психометрична якість 12 нових завдань, згенерованих ШІ-модулем. За результатами реального тестування 10 із 12 нових завдань отримали статус «Залишити» з Fi у діапазоні 0,73–0,85, D-index у діапазоні 0,27–0,80 та P-value у діапазоні 0,79–0,94. Два завдання отримали статус «Переглянути», жодне не потрапило до категорії «Видалити». Показники нових згенерованих завдань наведено на рисунку 4.3.4.

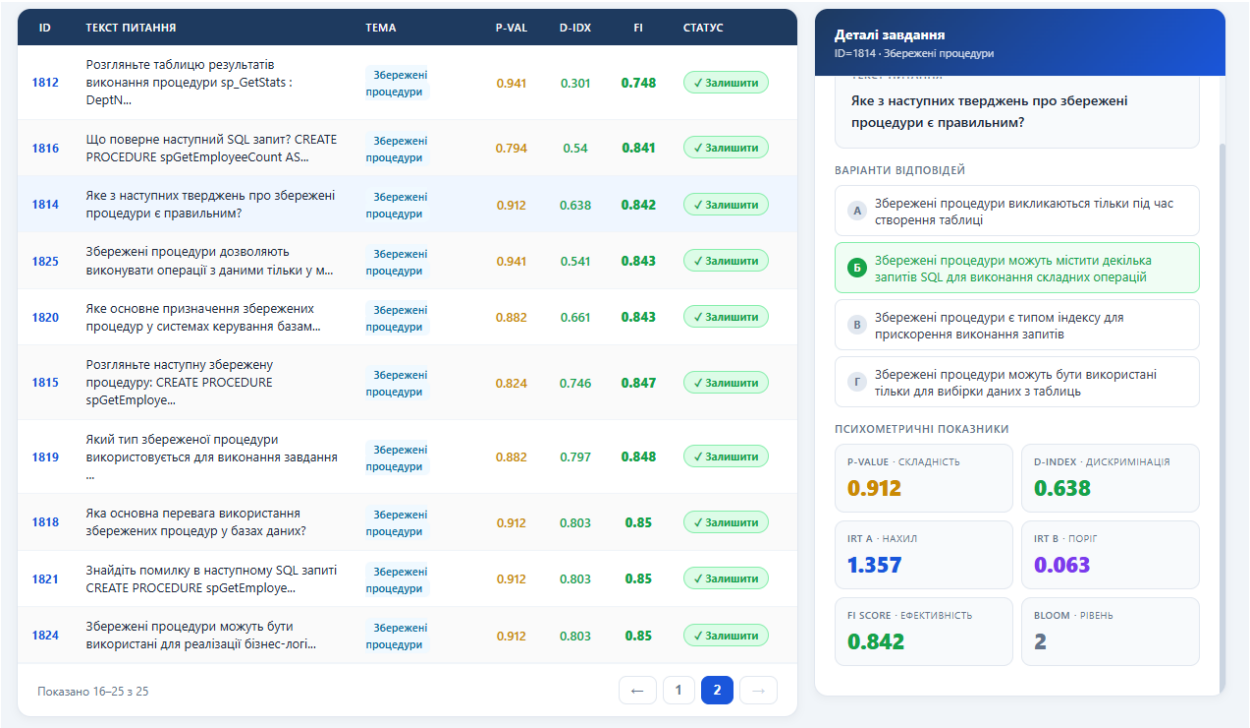


Рисунок 4.3.4 — Психометричні показники нових завдань, згенерованих ШІ-модулем системи

Для порівняння: у вихідному банку жодне завдання не досягало Fi = 0,70, тоді як нові згенеровані завдання стабільно перевищують цей поріг. Можемо зробити висновок, що ШІ-модуль здатний генерувати завдання, які відповідають психометричним вимогам якісного тесту, без ручного налаштування параметрів складності. Два завдання, що отримали статус «Переглянути» (Fi = 0,375 та Fi = 0,382), мають низький P-value (0,353 та 0,471 відповідно), що вказує на надмірну складність, а не на структурний дефект формулювання. Це є прийнятним результатом і може бути скоригованим шляхом повторної генерації з уточненими параметрами складності.

Результати двоетапної апробації підтверджують працездатність системи в обох її функціях. По-перше, діагностична функція реалізована коректно: система автоматично виявила реальні психометричні дефекти вихідного банку — тривіальні завдання, від'ємну дискримінацію та загальну відсутність диференціювальної здатності, — які залишаються непоміченими при ручній

перевірці. По-друге, функція оптимізації підтверджена експериментально: завдання, згенеровані ШІ-модулем системи, отримали статус «Залишити» за результатами реального тестування у 83% випадків (10 із 12), що демонструє практичну придатність автоматичної генерації як інструменту покращення якості банку тестових завдань.

4.4 Оцінка зручності використання, інтерпретації результатів та якості AI-підтримки

Інтерфейс системи організовано за принципом послідовного робочого процесу: імпорт даних — аналіз — перегляд результатів — оптимізація — формування звіту. Кожен етап реалізовано як окрему вкладку застосунку, що унеможливлює пропуск кроків і забезпечує логічну навігацію для користувача без спеціальної психометричної підготовки. У процесі апробації імпорт файлу з 34 студентами та 25 завданнями виконався без помилок і не вимагав ручного коригування даних — система автоматично розпізнала формат Moodle CSV, визначила кількість студентів і завдань та побудувала матрицю відповідей. Інтегральний показник Fi згортає три групи показників (CTT, IRT, KST) в одне число від 0 до 1 з чітким пороговим значенням: Fi ≥ 0,70 означає якісне завдання, Fi < 0,40 — завдання що підлягає видаленню, що усуває необхідність самостійної інтерпретації окремих показників. Кольорове кодування статусів на вкладці аналізу дозволяє миттєво ідентифікувати проблемні завдання без читання числових значень. AI-звіт формулює висновки природною мовою: замість «середній D-index=0,023» система повідомляє «64% завдань мають Fi менше 0,40» та «пріоритетна дія — видалити 16 завдань із теми Збережені процедури», що знижує поріг входження для викладачів, які не знайомі з психометричною термінологією. Також у процесі апробації було виявлено якісну відмінність між генерацією завдань без довідника фактів та з ним. При генерації завдань із теми «Збережені процедури T-SQL» без контекстного обмеження модель продукувала завдання з фактично некоректними варіантами відповіді — зокрема, хибними формулюваннями причин

синтаксичних помилок SQL. Приклад такої генерації наведено на рисунку 4.4.5.

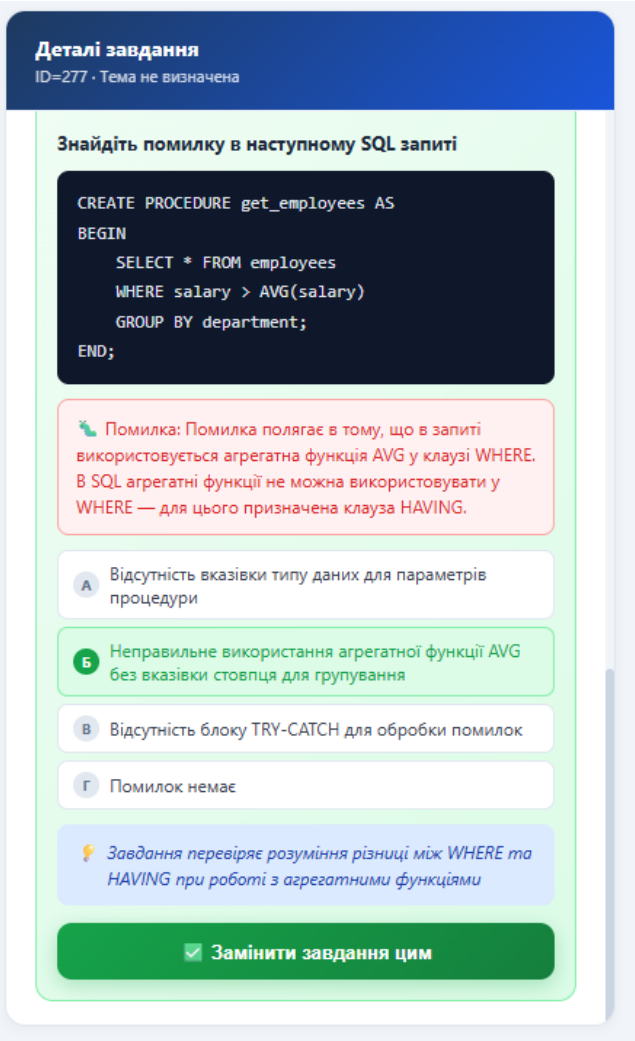


Рисунок 4.4.5 — Приклад завдання згенерованого без довідника фактів із зафіксованими помилками

Після активації механізму довідника фактів, якість генерацій суттєво покращилася: модель отримала фактологічне обмеження, що унеможливило генерацію завдань із некоректними еталонними відповідями. Таким чином, довідник фактів виконує функцію фактологічного заземлення мовної моделі в предметній області курсу, що є необхідною умовою практичного застосування AI-генерації в освітньому контексті. Приклади коректно згенерованих завдань із довідником наведено на рисунку 4.4.6.

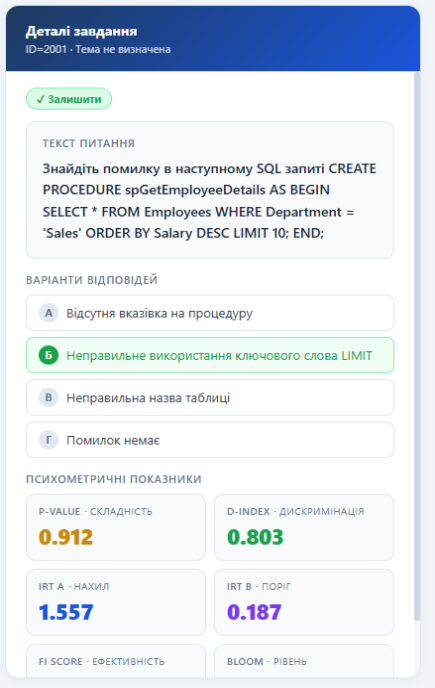


Рисунок 4.4.6 — Приклади завдань згенерованих з довідником фактів

Система формує звіти у форматах Excel та PDF, що відповідає потребам практичного використання: викладач отримує готовий документ із конкретними рекомендаціями, не потребуючи додаткової обробки даних. Приклад згенерованого PDF-звіту наведено у додатку Д на рисунку Д.6.

Апробація системи підтвердила її придатність до практичного використання в освітньому процесі. Інтерфейс забезпечує доступність інструментів психометричного аналізу для викладачів без спеціалізованої підготовки. Інтегральний показник Fi та AI-звітність знижують поріг інтерпретації результатів. Механізм довідника фактів забезпечує фактологічну коректність AI-генерованих завдань, а система звітності дозволяє фіксувати та передавати результати аналізу у стандартних форматах.

4.5 Висновок до розділу 4

У четвертому розділі проведено комплексне тестування та апробацію розробленої системи на двох незалежних рівнях: автоматизованому

модульному тестуванні математичного ядра та функціональному тестуванні на реальних даних курсу «Бази даних».

За результатами виконання 36 автоматизованих модульних тестів підтверджено математичну коректність усіх реалізованих алгоритмів — CTTAnalyzer, IRTAnalyzer та KSTAnalyzer, включаючи п'ять сценаріїв крайніх випадків із виродженими вхідними даними. Усі 36 тестів пройдено успішно (100%).

Функціональне тестування на реальному банку з 25 завдань та матриці відповідей 34 студентів підтвердило діагностичну здатність системи: автоматично виявлено критичні психометричні дефекти вихідного банку, зокрема 10 тривіальних завдань із P-value = 1,0 та 7 завдань із від'ємним *D-index*, які залишаються непоміченими при ручній перевірці. За результатами двоетапної апробації середній показник Fi зріс з 0,327 до 0,670 (приріст 105%), частка завдань зі статусом «Залишити» збільшилася з 0% до 44%, а коефіцієнт Кронбаха α оптимізованого банку склав 0,744, що свідчить про прийнятну внутрішню узгодженість тесту.

Підтверджено ефективність ШІ-модуля генерації замінних завдань: 10 із 12 згенерованих завдань отримали статус «Залишити» за результатами тестування (83%), що демонструє практичну придатність автоматичної генерації як інструменту оптимізації банку. Виявлено та підтверджено роль механізму довідника фактів як необхідної умови фактологічної коректності AI-генерованих завдань у предметній галузі баз даних. Апробація інтерфейсу підтвердила його придатність для використання викладачами без спеціалізованої психометричної підготовки.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було проаналізовано предметну область психометричного оцінювання тестових завдань, спроєктовано, розроблено і протестовано програмний додаток для автоматизованого аналізу та оптимізації банку тестових завдань на основі моделей класичної теорії тестів, двопараметричної моделі IRT та теорії простору знань. Поставлена мета — створення інструменту підтримки прийняття рішень щодо якості тестових завдань, доступного для викладачів без спеціалізованої психометричної підготовки, — була досягнута у повному обсязі.

У першому розділі проведено аналіз предметної області та існуючих підходів до оцінювання якості тестових завдань. Виконано огляд методів класичної теорії тестів, Item Response Theory та Knowledge Space Theory, визначено їх переваги та обмеження, що обґрунтувало доцільність інтеграції трьох моделей в єдиний інтегральний показник якості Fi. Проведено аналіз існуючих програмних рішень і виявлено відсутність інструментів що поєднують психометричний аналіз із AI-генерацією замінних завдань.

У другому розділі сформовано математичну модель системи. Формалізовано матрицю відповідей студентів, визначено показники CTT (P-value, D-index, коефіцієнт Кронбаха α), параметри 2PL-моделі IRT (дискримінація a, складність b, рівень підготовки θ) та структуру простору знань KST у вигляді орієнтованого ациклічного графа передумов. Розроблено формулу інтегрального показника якості Fi як зваженої суми нормалізованих складових трьох моделей з адаптивними ваговими коефіцієнтами залежно від розміру вибірки.

У третьому розділі виконано повну реалізацію програмного додатку. Серверна частина реалізована мовою Python з використанням фреймворку FastAPI та бази даних Microsoft SQL Server. Математичне ядро системи включає модулі CTTAnalyzer, IRTAnalyzer, KSTAnalyzer та QualityAnalyzer,

що реалізують відповідно розрахунок CTT-показників, оцінювання параметрів 2PL-моделі методом MAP з узагальненим EM-алгоритмом, перевірку структурної коректності завдань у графі знань та інтегральне оцінювання якості. Клієнтська частина реалізована як SPA на основі React 18 з інтерактивними аналітичними графіками та триступеневим майстром імпорту даних. AI-модуль інтегровано через Groq API з механізмом довідника верифікованих фактів для зниження фактичних помилок при генерації завдань.

У четвертому розділі проведено комплексне тестування та апробацію системи. За результатами виконання 36 автоматизованих модульних тестів підтверджено математичну коректність усіх реалізованих алгоритмів — усі тести пройдено успішно (100%), включаючи п'ять сценаріїв крайніх випадків. Функціональна апробація на реальному банку з 25 завдань та матриці відповідей 34 студентів підтвердила діагностичну ефективність системи: автоматично виявлено 10 тривіальних завдань із P-value = 1,0 та 7 завдань із від'ємним D-index. За результатами двоетапного експерименту середній показник Fi зріс з 0,327 до 0,670 (приріст 105%), частка завдань зі статусом «Залишити» збільшилася з 0% до 44%, а коефіцієнт Кронбаха α оптимізованого банку склав 0,744. Ефективність ШІ-модуля генерації підтверджена експериментально: 83% згенерованих завдань отримали статус «Залишити» за результатами реального тестування.

Практична цінність розробленої системи полягає у вирішенні реальної проблеми оцінювання якості тестових банків, яка у більшості освітніх закладів вирішується виключно на основі суб'єктивного досвіду викладача без кількісного підґрунтя. Розроблена система автоматично діагностує психометричні дефекти завдань — надмірну легкість, від'ємну дискримінацію, структурну невідповідність графу знань — формує пріоритизовані рекомендації щодо оптимізації та генерує замінні завдання з цільовими психометричними характеристиками. Інтегральний показник Fi та AI-звітність природною мовою забезпечують доступність результатів аналізу для

викладачів без спеціалізованої психометричної підготовки, що є ключовою умовою практичного впровадження системи у реальний освітній процес. Система може бути застосована у вищих навчальних закладах для контролю якості тестових банків дисциплін будь-якого профілю, а також як основа для побудови корпоративних систем оцінювання компетентностей персоналу.

Подальший розвиток системи може здійснюватись у кількох напрямках. По-перше, розширення підтримки психометричних моделей — зокрема трипараметричної моделі IRT з параметром вгадування c та полихорічної кореляції для політомічних завдань — підвищило б точність оцінювання для тестів із частковими балами та завдань типу «множинний вибір», де ймовірність випадкової правильної відповіді є суттєвою. Це дозволило б коректніше оцінювати параметр складності b для завдань з малою кількістю варіантів відповіді, де ефект вгадування систематично занижує оцінку реальної складності. По-друге, реалізація режиму комп'ютеризованого адаптивного тестування в реальному часі на основі відкаліброваних параметрів банку дозволила б використовувати систему безпосередньо під час проведення тестування: алгоритм послідовного вибору завдань на основі максимуму інформаційної функції $I(\theta)$ забезпечив би точнішу оцінку рівня підготовки студента за меншої кількості завдань порівняно з фіксованим тестом. По-третє, інтеграція з платформою Moodle через офіційний REST API замість імпорту CSV-файлів забезпечила б безперервну синхронізацію результатів тестування та автоматичне оновлення психометричних параметрів після кожної тестової сесії без ручного експорту даних. По-четверте, розширення AI-модуля підтримкою мультимодальної генерації — завдань із графічними елементами, діаграмами та схемами — відкрило б можливість оптимізації банків для дисциплін з високим рівнем візуальної складової, зокрема інженерних та природничих курсів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Теличко Н., Моргун І. Тестування як один з науково обґрунтованих дослідницьких методів педагогічного вимірювання та оцінювання // Науковий вісник Ужгородського університету. – 2022. URL: http://visnyk-ped.uzhnu.edu.ua/article/view/267714/263548 (дата звернення: 25.03.2026).

2. Основи педагогічного оцінювання. URL: https://www.testcentr.org.ua/docs/TB/ids/Навчально_методичні_та_інформаційно_довідкові_матеріали_для_педагогічних.pdf (дата звернення: 25.03.2026).

3. Ляшенко О. І. Assessment of student achievements by adaptive testing // Освіта для миру. – 2019. URL: https://lib.iitta.gov.ua/id/eprint/718779/1/volume_1_2019-178-189.pdf (дата звернення: 25.03.2026).

4. Standards for Educational and Psychological Testing / American Educational Research Association, American Psychological Association, National Council on Measurement in Education. – Washington, 2014. – 324 с.

5. Мазорчук М. С., Коробчинський К. П., Григор'єва Т. А. Формування банку тестових завдань на основі аналізу психометричних властивостей тестів в умовах малих популяцій випробуваних // Радіоелектронні і комп'ютерні системи. – 2015. – № 4(74). – С. 151–155. URL: https://korobchinskiy.com/wp-content/uploads/2018/05/2015_recs_6.pdf (дата звернення: 25.03.2026).

6. Вплив форми тестових завдань і профільної спеціалізації класів на результати виконання здобувачами середньої освіти тесту з біології // Ukrainian Educational Journal. – 2020. URL: https://uej.undip.org.ua/index.php/journal/article/view/428 (дата звернення: 25.03.2026).

7. Орестівна О. Тест як ефективний засіб оцінювання якості знань студентів // Наукові записки. Серія: Педагогічні науки. – 2022. URL:

https://pednauk.cusu.edu.ua/index.php/pednauk/article/view/1193/1121 (дата звернення: 25.03.2026).

8. Теоретичні основи тестування у контролі навчальних досягнень студентів при дистанційному навчанні в ЗВО. – 2023. URL: http://www.innovpedagogy.od.ua/archives/2023/65/part_1/26.pdf (дата звернення: 25.03.2026).

9. Трегубова Г. М. Методологічні основи організації тестового контролю знань студентів. URL: https://ela.kpi.ua/server/api/core/bitstreams/c8422d64-ec1f-41b7-bbf4-a4706d28b42f/content (дата звернення: 25.03.2026).

10. Микитенко. Магістерська дисертація – Кафедра математичного аналізу та теорії ймовірностей. URL: https://matan.kpi.ua/media/2025/dis12/Mykytenko.pdf (дата звернення: 25.03.2026).

11. НМТ 2025 року – Звіт. Український центр оцінювання якості освіти. URL: https://testportal.gov.ua/wp-content/uploads/2025/09/Zvit-NMT_2025-Tom_2-11.09.2025.pdf (дата звернення: 25.03.2026);

12. Sabourin J., Rowe J., Mott B. Students' patterns of interaction with a mathematics intelligent tutor: learning analytics application. – 2016. URL: https://arxiv.org/pdf/1607.07284 (дата звернення: 25.03.2026).

13. Інформаційна технологія автоматичної генерації тестових завдань з керованою складністю. URL: https://api.dspace.wunu.edu.ua/api/core/bitstreams/d75be79a-cded-45e5-bc25-814cf6c63076/content (дата звернення: 25.03.2026).

14. Classical Test Theory // Cogn-IQ. URL: https://www.cogn-iq.org/learn/theory/classical-test-theory/ (дата звернення: 10.05.2026).

15. Isley C., Gilbert J., Kassos E. та ін. Assessing the Quality of AI-Generated Exams: A Large-Scale Field Study. – arXiv, 2025. URL: https://arxiv.org/pdf/2508.08314 (дата звернення: 25.03.2026).

16. Chen Y., Li X., Liu J., Ying Z. Item Response Theory – A Statistical Framework for Educational and Psychological Measurement. – arXiv, 2021. URL: https://arxiv.org/pdf/2108.08604 (дата звернення: 26.03.2026).

17. Doignon J.-P., Falmagne J.-C. Knowledge Spaces and Learning Spaces. – arXiv, 2015. URL: https://arxiv.org/pdf/1511.06757 (дата звернення: 10.05.2026).

18. Дем'яненко В. Б. Використання адаптивних систем тестування в освітньому процесі // Інститут інформаційних технологій і засобів навчання НАПН України. – 2020. URL: https://lib.iitta.gov.ua/id/eprint/721216/1/Demyanenko.pdf (дата звернення: 15.04.2026).

19. Knowledge Space Theory // TquanT Online Learning Contents. URL: https://tquant.eu/online-learning-contents/r-shiny-apps/knowledge-space-theory/ (дата звернення: 15.04.2026).

20. Deonovic B., Yudelson M., Bolsinova M., Attali M., Maris G. Learning meets Assessment: On the relation between Item Response Theory and Bayesian Knowledge Tracing. – arXiv, 2018. URL: https://arxiv.org/pdf/1901.10268 (дата звернення: 15.04.2026).

21. Pavlech J., Martinková P. Bridging Item Response Theory and Factor Analysis: A Four-Parameter Mixture-Dichotomized Model with Bayesian Estimation. – arXiv, 2024. URL: https://arxiv.org/pdf/2411.11520 (дата звернення: 14.05.2026).

22. A Systematic Review on Assessment in Adaptive Learning // International Journal of Advanced Computer Science and Applications (IJACSA). – 2024. – Vol. 15, No. 7. URL: https://thesai.org/Downloads/Volume15No7/Paper_85-A_Systematic_Review_on_Assessment_in_Adaptive_Learning.pdf (дата звернення: 26.03.2026).

23. Huebner A. An Overview of Recent Developments in Cognitive Diagnostic Computer Adaptive Assessments // Practical Assessment, Research &

Evaluation. – 2010. – Vol. 15, No. 3. URL: https://openpublishing.library.umass.edu/pare/article/1553/galley/1504/view/ (дата звернення: 16.04.2026).

24. Heller J., Doignon J.-P., Falmagne J.-C., Hockemeyer C. A Gentle Introduction to Knowledge Space Theory. – arXiv, 2018. URL: https://arxiv.org/pdf/1803.05926 (дата звернення: 7.05.2026).

25. Brancaccio A., Stefanutti L. Knowledge Assessment App (KAA) // TquanT Project Report. URL: https://tquant.eu/documents/KAA-Report.pdf (дата звернення: 10.04.2026).

26. Bridging item response theory and factor analysis: a four-parameter mixture-dichotomized model with bayesian estimation. – arXiv, 2024. URL: https://arxiv.org/pdf/2407.04071 (дата звернення: 10.05.2026).

27. Таксономія освітніх цілей Блума // Інститут цифровізації освіти НАПН України. Електронна енциклопедія освіти. URL: https://eduglos.iitta.gov.ua/index.php/Таксономія_освітніх_цілей_Блума (дата звернення: 10.05.2026).

28. ISO/IEC/IEEE 29148:2018 Systems and software engineering — Life cycle processes — Requirements engineering. URL: https://drkasbokar.com/wp-content/uploads/2024/09/29148-2018-ISOIECIEEE.pdf (дата звернення: 10.05.2026).

29. ISO/IEC 25010:2023 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE). URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-2:v1:en (дата звернення: 8.05.2026).

30. Moodle Documentation. URL: https://docs.moodle.org/501/en/Main_page (дата звернення: 10.05.2026).

31. ALEKS: Cosyn E., Uzun H., Doble C., Matayoshi J. A practical perspective on knowledge space theory: ALEKS and its data // Journal of Mathematical Psychology. – 2021. – Vol. 101. – P. 102512. URL: https://doi.org/10.1016/j.jmp.2021.102512 (дата звернення: 10.05.2026).

32. FastTest: Thompson N. A. Review of FastTest: A Platform for Adaptive Testing // Measurement: Interdisciplinary Research and Perspectives. – 2018. – Vol. 16, No. 4. – P. 226–229. URL: https://doi.org/10.1080/15366367.2018.1492867 (дата звернення: 10.05.2026).

33. Concerto**:** Scalise K., Allen D. D. Use of open-source software for adaptive measurement: Concerto as an R-based computer adaptive development and delivery platform // British Journal of Mathematical and Statistical Psychology. – 2015. – Vol. 68, No. 3. – P. 478–496. URL: https://doi.org/10.1111/bmsp.12057 (дата звернення: 10.05.2026).

34. Xcalibre**:** Thompson N. A., Lee J. Xcalibre Item Parameter Calibration Software for Item Response Theory and Rasch Models // Measurement: Interdisciplinary Research and Perspectives. – 2022. – Vol. 20, No. 4. – P. 218–228. URL: https://doi.org/10.1080/15366367.2022.2026736 (дата звернення: 10.05.2026).

35. R/mirt: Chalmers R. P. mirt: A Multidimensional Item Response Theory Package for the R Environment // Journal of Statistical Software. – 2012. – Vol. 48, No. 6. – P. 1–29. URL: https://doi.org/10.18637/jss.v048.i06 (дата звернення: 10.05.2026).

36. Марченко А. В. Проектування інформаційних систем : посібник. – Київ, 2016. – 89 с. URL: https://duikt.edu.ua/ua/lib/1/category/739/view/144 (дата звернення: 10.05.2026).

37. FastAPI. Recap // FastAPI Documentation. URL: https://fastapi.tiangolo.com/#recap (дата звернення: 10.05.2026).

38. React Documentation // React. URL: https://react.dev (дата звернення: 10.05.2026).

39. JSON Web Tokens (JWT) // JWT.IO. URL: https://jwt.io (дата звернення: 10.05.2026).

ДОДАТОК А. Результати виконання автоматизованого тестового набору

Таблиця А.1 — Структура таблиці TestBanks

Поле	Тип даних	Опис	Обмеження
bank_id	INT	Унікальний ідентифікатор банку	PRIMARY KEY, IDENTITY
name	VARCHAR	Назва банку тестових завдань	NOT NULL
description	TEXT	Опис банку	–
created_at	DATETIME	Дата створення	NOT NULL, DEFAULT GETDATE()
user_id	INT	Власник банку	FK → Users

Таблиця А.2 — Структура таблиці Tests

Поле	Тип даних	Опис	Обмеження
test_id	INT	Унікальний ідентифікатор тесту	PRIMARY KEY, IDENTITY
bank_id	INT	Банк якому належить тест	FK → TestBanks, NOT NULL
name	NVARCHAR	Назва тесту	NOT NULL
docx_filename	NVARCHAR	Ім'я файлу завдань	–
task_count	INT	Кількість завдань	DEFAULT 0
created_at	DATETIME	Дата створення	DEFAULT GETDATE()

Таблиця А.3 — Структура таблиці TestSessions

Поле	Тип даних	Опис	Обмеження
session_id	INT	Унікальний ідентифікатор сесії	PRIMARY KEY, IDENTITY
name	VARCHAR	Назва сесії тестування	NOT NULL
bank_id	INT	Прив'язаний банк	FK → TestBanks
test_id	INT	Прив'язаний тест	FK → Tests
conducted_at	DATETIME	Дата проведення	–
total_students	INT	Кількість студентів	–
created_at	DATETIME	Дата створення запису	NOT NULL, DEFAULT GETDATE()

Таблиця А.4 — Структура таблиці BankTasks

Поле	Тип даних	Опис	Обмеження
bank_id	INT	Ідентифікатор банку	FK → TestBanks
task_id	INT	Ідентифікатор завдання	FK → Tasks
test_id	INT	Опціональна прив'язка до тесту	FK → Tests

Складений первинний ключ: PK (bank_id, task_id)

Таблиця А.5 — Структура таблиці Students

Поле	Тип даних	Опис	Обмеження
student_id	INT	Унікальний ідентифікатор студента	PRIMARY KEY, IDENTITY
full_name	VARCHAR	Повне ім'я студента	NOT NULL

Закінчення таблиці А.5

group_name	VARCHAR	Назва групи	–
login	VARCHAR	Логін у системі Moodle	–
theta	FLOAT	Оцінений рівень підготовки θ	–
created_at	DATETIME	Дата додавання	NOT NULL, DEFAULT GETDATE()

Таблиця А.6 — Структура таблиці Users

Поле	**Тип даних**	**Опис**	**Обмеження**
user_id	INT	Унікальний ідентифікатор користувача	PRIMARY KEY, IDENTITY
username	NVARCHAR	Логін користувача	NOT NULL, UNIQUE
email	NVARCHAR	Електронна пошта	NOT NULL, UNIQUE
password_hash	NVARCHAR	Хеш пароля (bcrypt)	NOT NULL
role	NVARCHAR	Роль: admin або teacher	DEFAULT 'teacher'
full_name	NVARCHAR	Повне ім'я	–
created_at	DATETIME	Дата реєстрації	DEFAULT GETDATE()
is_active	BIT	Активність облікового запису	DEFAULT 1

ДОДАТОК Б. Програмний код математичного ядра

Лістинг Б.1 — Ключові методи модуля CTTAnalyzer (core/ctt.py)

```
class CTTAnalyzer:
    def __init__(self, response_matrix: np.ndarray,
                 max_scores: np.ndarray = None):
        self.raw = response_matrix.astype(float)
        self.n_students, self.n_items = self.raw.shape
        if max_scores is not None:
            self.max_scores = np.array(max_scores, dtype=float)
        else:
            col_max = self.raw.max(axis=0)
            self.max_scores = np.where(col_max > 0, col_max, 1.0)
        # Нормалізована матриця [0..1]
        self.U = self.raw / self.max_scores
        self.total_scores = self.U.sum(axis=1)

    def p_values(self) -> np.ndarray:
        # Формула (2.1): частка правильних відповідей
        return self.U.mean(axis=0)

    def discrimination_index(self) -> np.ndarray:
        # Формула (2.2): corrected point-biserial кореляція
        d_indices = []
        for i in range(self.n_items):
            item_scores = self.U[:, i]
            corrected_total = self.total_scores - item_scores
            if item_scores.std() == 0 or corrected_total.std() == 0:
                d_indices.append(0.0)
                continue
            r, _ = stats.pearsonr(item_scores, corrected_total)
            d_indices.append(round(float(r), 4))
        return np.array(d_indices)

    def cronbach_alpha(self) -> float:
        # Формула (2.3): коефіцієнт внутрішньої узгодженості
        k = self.n_items
        if k < 2:
            return 0.0
        item_variances = self.U.var(axis=0, ddof=1).sum()
        total_variance = self.total_scores.var(ddof=1)
        if total_variance == 0:
            return 0.0
        alpha = (k / (k - 1)) * (1 - item_variances / total_variance)
        return round(float(alpha), 4)

    def standard_error_of_measurement(self) -> float:
        # Формула (3.1): стандартна похибка вимірювання
        alpha = self.cronbach_alpha()
        sd = self.total_scores.std(ddof=1)
        return round(float(sd * np.sqrt(1 - alpha)), 4)
```

Лістинг Б.2 — Ключові методи модуля IRTAnalyzer (core/irt.py)

```
class IRTAnalyzer:
    PRIOR_A_MEAN = 1.0
    PRIOR_A_STD  = 0.5
    PRIOR_B_MEAN = 0.0
    PRIOR_B_STD  = 1.0

    @staticmethod
    def prob_vector(theta: np.ndarray,
                    a: float, b: float) -> np.ndarray:
        # Формула (2.4): модель 2PL
        z = np.clip(a * (theta - b), -500, 500)
        return 1.0 / (1.0 + np.exp(-z))

    def _item_map_objective(self, params, responses, theta):
        # MAP objective = -log_likelihood - log_prior
        a, b = params
        p = self.prob_vector(theta, a, b)
        p = np.clip(p, 1e-9, 1 - 1e-9)
        ll = responses * np.log(p) + (1 - responses) * np.log(1 - p)
        neg_ll = -np.sum(ll)
        # Байєсівська регуляризація (апріорний нормальний розподіл)
        log_prior_a = ((a - self.PRIOR_A_MEAN) ** 2 /
                       (2 * self.PRIOR_A_STD ** 2))
        log_prior_b = ((b - self.PRIOR_B_MEAN) ** 2 /
                       (2 * self.PRIOR_B_STD ** 2))
        return neg_ll + log_prior_a + log_prior_b

    def fit(self, max_iter: int = 20, tol: float = 1e-4) -> dict:
        # EM-алгоритм: спільна оцінка параметрів (a, b, θ)
        p_students = np.clip(self.U.mean(axis=1), 0.01, 0.99)
        theta_current = np.log(p_students / (1 - p_students))

        for iteration in range(max_iter):
            # E-step: оцінюємо a, b при фіксованому θ (MAP)
            a_new, b_new = self._estimate_item_params_step(
                theta_current)
            self.a = a_new
            self.b = b_new
            # M-step: оцінюємо θ при фіксованих a, b (MLE)
            theta_new = self._estimate_theta_step()

            max_delta = max(
                np.max(np.abs(a_new - self.a)),
                np.max(np.abs(b_new - self.b)),
                np.max(np.abs(theta_new - theta_current))
            )
            theta_current = theta_new
            if max_delta < tol and iteration > 0:
                break

        self.theta = theta_current
        return {'a': self.a, 'b': self.b, 'theta': self.theta}

    def item_information(self, theta: float, j: int) -> float:
        # Формула (2.5): інформаційна функція завдання
        p = self.prob(theta, self.a[j], self.b[j])
```

```
        return self.a[j] ** 2 * p * (1 - p)

    def average_information(self) -> np.ndarray:
        # Середня інформативність по θ ∈ [-4, 4]
        theta_range = np.linspace(-4, 4, 50)
        avg_info = np.zeros(self.n_items)
        for j in range(self.n_items):
            info_j = np.array([self.item_information(t, j)
                               for t in theta_range])
            avg_info[j] = info_j.mean()
        return avg_info
```

Лістинг Б.3 — Ключові методи модуля KSTAnalyzer (core/kst.py)

```
class KSTAnalyzer:
    def load_from_db(self, bank_id: int = None):
        # Завантажує граф концептів та передумов з БД
        conn = get_connection()
        cursor = conn.cursor()
        cursor.execute(
            "SELECT concept_id, name FROM Concepts ORDER BY
concept_id")
        for row in cursor.fetchall():
            cid, name = row[0], row[1]
            self.concepts[cid] = name
            self.graph.add_node(cid, name=name)
        cursor.execute(
            "SELECT concept_id, prerequisite_id "
            "FROM ConceptPrerequisites")
        for row in cursor.fetchall():
            # Ребро: prereq → concept (передумовна залежність)
            self.graph.add_edge(row[1], row[0])
        conn.close()

    def is_valid_dag(self) -> bool:
        # Перевірка ациклічності графу передумов
        return nx.is_directed_acyclic_graph(self.graph)

    def check_structural_correctness(self, task_id: int) -> bool:
        #  перевірка структурної коректності завдання
        concept_id = self.task_concepts.get(task_id)
        if concept_id is None:
            return False
        if concept_id not in self.graph.nodes:
            return False
        # Всі предки концепту мають бути покриті завданнями
        prerequisites = list(nx.ancestors(self.graph, concept_id))
        for prereq_id in prerequisites:
            if prereq_id not in self.coverage or \
               len(self.coverage[prereq_id]) == 0:
                return False
        return True

    def get_kst_scores(self, task_ids: list) -> np.ndarray:
        # KST оцінка = середнє між structural_ok і coverage_score
        scores = np.zeros(len(task_ids))
```

```
        for i, task_id in enumerate(task_ids):
            structural = self.check_structural_correctness(task_id)
            cid = self.task_concepts.get(task_id)
            if cid and cid in self.coverage:
                coverage_score = min(
                    len(self.coverage[cid]) / 3.0, 1.0)
            else:
                coverage_score = 0.0
            scores[i] = (float(structural) + coverage_score) / 2.0
        return scores

    def get_student_knowledge_state(self, student_responses,
                                    task_ids) -> set:
        # Формула (2.10): стан знань студента
        concept_scores = {}
        concept_counts = {}
        for i, task_id in enumerate(task_ids):
            cid = self.task_concepts.get(task_id)
            if cid is None:
                continue
            concept_scores.setdefault(cid, 0)
            concept_counts.setdefault(cid, 0)
            concept_scores[cid] += student_responses[i]
            concept_counts[cid] += 1
        # Концепт "знає" якщо >= 50% правильних відповідей
        return {cid for cid, score in concept_scores.items()
                if concept_counts[cid] > 0
                and score / concept_counts[cid] >= 0.5}
```

Лістинг Б.4 — Ключові методи модуля QualityAnalyzer (core/quality.py)

```
# Ваги інтегрального показника Fi
W_CTT, W_IRT, W_KST = 0.40, 0.30, 0.30          # n >= 100
W_CTT_SMALL, W_IRT_SMALL, W_KST_SMALL = 0.50, 0.20, 0.30  # n < 100
THRESHOLD_KEEP, THRESHOLD_REVIEW = 0.70, 0.40

class QualityAnalyzer:
    def compute_fi(self, response_matrix, student_ids,
                   task_ids, max_scores=None, bank_id=1):
        n_students = response_matrix.shape[0]
        # Адаптивні ваги залежно від розміру вибірки
        if n_students < 100:
            self.w_ctt, self.w_irt, self.w_kst = (
                W_CTT_SMALL, W_IRT_SMALL, W_KST_SMALL)
        else:
            self.w_ctt, self.w_irt, self.w_kst = W_CTT, W_IRT, W_KST

        # CTT аналіз
        self.ctt = CTTAnalyzer(response_matrix, max_scores)
        ctt_scores = self._normalize_ctt()

        # IRT аналіз (бінаризована матриця)
        binary_matrix = binarize_matrix(response_matrix, max_scores)
        self.irt = IRTAnalyzer(binary_matrix)
        self.irt.fit()
        irt_scores = self._normalize_irt()
```

```
        # KST аналіз
        self.kst = KSTAnalyzer()
        self.kst.load_from_db(bank_id=bank_id)
        kst_scores = self.kst.get_kst_scores(task_ids)
        self._kst_scores_cache = kst_scores

        # Формула (2.9): інтегральний показник Fi
        fi = (self.w_ctt * ctt_scores +
              self.w_irt * irt_scores +
              self.w_kst * kst_scores)
        self.fi_scores = np.clip(fi, 0.0, 1.0)
        return self.fi_scores

    def _normalize_ctt(self) -> np.ndarray:
        p = self.ctt.p_values()
        d = self.ctt.discrimination_index()
        alpha = self.ctt.cronbach_alpha()
        alpha_if_del = self.ctt.cronbach_if_deleted()
        # Нормалізація P-value за кусково-лінійною функцією
        p_score = np.where(p >= 0.95, 1.0 - (p - 0.95) / 0.05,
                  np.where(p >= 0.50, 1.0,
                  np.where(p >= 0.20, (p - 0.20) / 0.30, 0.0)))
        # Нормалізація D-index
        d_score = np.where(d < 0, 0.0,
                  np.where(d < 0.2, d / 0.2 * 0.5,
                  np.where(d < 0.4, 0.5 + (d-0.2)/0.2*0.5, 1.0)))
        # Alpha оцінка
        if alpha < 0:
            alpha_score = np.zeros(len(p))
        else:
            alpha_score = np.where(
                alpha_if_del > alpha + 0.02, 0.0,
                np.where(alpha_if_del < alpha - 0.02, 1.0, 0.7))
        return np.clip(0.35*p_score + 0.45*d_score +
                       0.20*alpha_score, 0.0, 1.0)

    def classify(self, fi: float) -> str:
        # Класифікація за порогами відповідно до таблиці 2.4
        if fi >= THRESHOLD_KEEP:
            return 'active'
        elif fi >= THRESHOLD_REVIEW:
            return 'review'
        return 'remove'
```

ДОДАТОК В. Програмний код AI-модуля та модуля даних

Лістинг В.1 — Програмний код функції get_response_matrix() (database/queries.py)

```
def get_response_matrix(bank_id: int):
    # Формує матрицю відповідей U = (u_ij) відповідно до моделі (2.9)
    conn = get_connection()
    cursor = conn.cursor()

    # Крок 1: отримуємо max_score для кожного завдання
    cursor.execute("""
        SELECT bt.task_id, COALESCE(t.max_score, 1.0) AS max_score
        FROM BankTasks bt
        JOIN Tasks t ON t.task_id = bt.task_id
        WHERE bt.bank_id = ?
        ORDER BY bt.task_id
    """, (bank_id,))
    task_rows = cursor.fetchall()
    task_max = {int(r[0]): float(r[1]) if float(r[1]) > 0 else 1.0
                for r in task_rows}

    # Крок 2: отримуємо відповіді студентів (score або is_correct)
    cursor.execute("""
        SELECT DISTINCT r.student_id, r.task_id,
               COALESCE(r.score, CAST(r.is_correct AS FLOAT)) as score
        FROM Responses r
        JOIN BankTasks bt ON bt.task_id = r.task_id
        WHERE bt.bank_id = ?
    """, (bank_id,))
    rows = cursor.fetchall()
    conn.close()

    data = [(int(r[0]), int(r[1]), float(r[2]))
            for r in rows if int(r[1]) in task_max]

    df = pd.DataFrame(data, columns=['student_id', 'task_id',
'score'])
    matrix = df.pivot_table(
        index='student_id', columns='task_id',
        values='score', aggfunc='max', fill_value=0)

    # Крок 3: вирівнюємо порядок стовпців
    task_ids_ordered = [tid for tid in task_max.keys()
                        if tid in matrix.columns]
    matrix = matrix[task_ids_ordered]
    max_scores = np.array([task_max[tid] for tid in task_ids_ordered])

    return (matrix.values, list(matrix.index),
            task_ids_ordered, max_scores)
```

ДОДАТОК Г. Результати виконання автоматизованого тестового набору

Лістинг Г.1 – Вивід консолі за результатами виконання автоматизованого тестового набору

```
C:\Users\eliza\test_bank_analyzer\venv\Scripts\python.exe
C:\Users\eliza\test_bank_analyzer\run_tests.py

============================================================
  ЗАПУСК ТЕСТІВ СИСТЕМИ АНАЛІЗУ БАНКУ ТЕСТОВИХ ЗАВДАНЬ
============================================================

============================================================
  CTT — Класична теорія тестів
============================================================
  ✓ test_p_values_known: PASSED
  ✓ test_p_values_range: PASSED
  ✓ test_discrimination_positive_for_good_items: PASSED
  ✓ test_discrimination_constant_item_zero: PASSED
  ✓ test_discrimination_corrected_lower_than_naive: PASSED
  ✓ test_cronbach_alpha_range_normal: PASSED (alpha=1.000)
  ✓ test_cronbach_alpha_negative: PASSED (alpha=0.000)
  ✓ test_cronbach_alpha_single_item: PASSED
  ✓ test_partial_scores_normalization: PASSED (P=[0.625 0.5  ])

  Результат: 9/9 пройдено

============================================================
  IRT — Item Response Theory (2PL модель)
============================================================
  ⏳ EM-алгоритм IRT (max 20 ітерацій)...
    Ітерація 1: max_delta = 1.717091
    Ітерація 2: max_delta = 0.760894
    Ітерація 3: max_delta = 0.300991
    Ітерація 4: max_delta = 0.186853
    Ітерація 5: max_delta = 0.124018
    Ітерація 6: max_delta = 0.083077
    Ітерація 7: max_delta = 0.055862
    Ітерація 8: max_delta = 0.037653
    Ітерація 9: max_delta = 0.025432
    Ітерація 10: max_delta = 0.017217
    Ітерація 11: max_delta = 0.011689
    Ітерація 12: max_delta = 0.007967
    Ітерація 13: max_delta = 0.005458
    Ітерація 14: max_delta = 0.003765
    Ітерація 15: max_delta = 0.002621
    Ітерація 16: max_delta = 0.001846
    Ітерація 17: max_delta = 0.001319
    Ітерація 18: max_delta = 0.000960
```

```
  Ітерація 19: max_delta = 0.000713
  Ітерація 20: max_delta = 0.000543
⚠ Досягнуто max_iter=20, збіжність не гарантована
⌛ Розрахунок інформаційної функції...
✓ test_item_params_bounds: PASSED
  a: [2.046 2.46  2.59  0.1   2.459 2.07 ]
  b: [-3.437e+00 -1.612e+00 -3.000e-03  3.000e-03  1.577e+00  3.378e+00]
⌛ EM-алгоритм IRT (max 20 ітерацій)...
  Ітерація 1: max_delta = 1.717091
  Ітерація 2: max_delta = 0.760894
  Ітерація 3: max_delta = 0.300991
  Ітерація 4: max_delta = 0.186853
  Ітерація 5: max_delta = 0.124018
  Ітерація 6: max_delta = 0.083077
  Ітерація 7: max_delta = 0.055862
  Ітерація 8: max_delta = 0.037653
  Ітерація 9: max_delta = 0.025432
  Ітерація 10: max_delta = 0.017217
  Ітерація 11: max_delta = 0.011689
  Ітерація 12: max_delta = 0.007967
  Ітерація 13: max_delta = 0.005458
  Ітерація 14: max_delta = 0.003765
  Ітерація 15: max_delta = 0.002621
  Ітерація 16: max_delta = 0.001846
  Ітерація 17: max_delta = 0.001319
  Ітерація 18: max_delta = 0.000960
  Ітерація 19: max_delta = 0.000713
  Ітерація 20: max_delta = 0.000543
⚠ Досягнуто max_iter=20, збіжність не гарантована
⌛ Розрахунок інформаційної функції...
✓ test_difficulty_ordering: PASSED (b[0]=-3.437 < b[5]=3.378)
⚠ Увага: вибірка 20 студентів замала для надійного IRT (рекомендовано ≥ 100).
  Параметри IRT будуть наближеними.
⚠ 2 завдань мають P=0 або P=1 — IRT для них встановить граничні значення.
✓ test_constant_items_detected: PASSED
⚠ Увага: вибірка 20 студентів замала для надійного IRT (рекомендовано ≥ 100).
  Параметри IRT будуть наближеними.
⚠ 2 завдань мають P=0 або P=1 — IRT для них встановить граничні значення.
⌛ EM-алгоритм IRT (max 5 ітерацій)...
  Ітерація 1: max_delta = 1.786520
  Ітерація 2: max_delta = 0.168656
  Ітерація 3: max_delta = 0.015750
  Ітерація 4: max_delta = 0.008002
  Ітерація 5: max_delta = 0.004431
⚠ Досягнуто max_iter=5, збіжність не гарантована
⌛ Розрахунок інформаційної функції...
✓ test_constant_items_boundary_params: PASSED
⌛ EM-алгоритм IRT (max 20 ітерацій)...
  Ітерація 1: max_delta = 1.717091
  Ітерація 2: max_delta = 0.760894
  Ітерація 3: max_delta = 0.300991
  Ітерація 4: max_delta = 0.186853
  Ітерація 5: max_delta = 0.124018
  Ітерація 6: max_delta = 0.083077
```

Ітерація 7: max_delta = 0.055862
Ітерація 8: max_delta = 0.037653
Ітерація 9: max_delta = 0.025432
Ітерація 10: max_delta = 0.017217
Ітерація 11: max_delta = 0.011689
Ітерація 12: max_delta = 0.007967
Ітерація 13: max_delta = 0.005458
Ітерація 14: max_delta = 0.003765
Ітерація 15: max_delta = 0.002621
Ітерація 16: max_delta = 0.001846
Ітерація 17: max_delta = 0.001319
Ітерація 18: max_delta = 0.000960
Ітерація 19: max_delta = 0.000713
Ітерація 20: max_delta = 0.000543
⚠ Досягнуто max_iter=20, збіжність не гарантована
⌛ Розрахунок інформаційної функції...
✓ test_icc_monotone: PASSED
⌛ ЕМ-алгоритм IRT (max 20 ітерацій)...
Ітерація 1: max_delta = 1.717091
Ітерація 2: max_delta = 0.760894
Ітерація 3: max_delta = 0.300991
Ітерація 4: max_delta = 0.186853
Ітерація 5: max_delta = 0.124018
Ітерація 6: max_delta = 0.083077
Ітерація 7: max_delta = 0.055862
Ітерація 8: max_delta = 0.037653
Ітерація 9: max_delta = 0.025432
Ітерація 10: max_delta = 0.017217
Ітерація 11: max_delta = 0.011689
Ітерація 12: max_delta = 0.007967
Ітерація 13: max_delta = 0.005458
Ітерація 14: max_delta = 0.003765
Ітерація 15: max_delta = 0.002621
Ітерація 16: max_delta = 0.001846
Ітерація 17: max_delta = 0.001319
Ітерація 18: max_delta = 0.000960
Ітерація 19: max_delta = 0.000713
Ітерація 20: max_delta = 0.000543
⚠ Досягнуто max_iter=20, збіжність не гарантована
⌛ Розрахунок інформаційної функції...
✓ test_icc_p_at_b_equals_05: PASSED
⌛ ЕМ-алгоритм IRT (max 20 ітерацій)...
Ітерація 1: max_delta = 1.717091
Ітерація 2: max_delta = 0.760894
Ітерація 3: max_delta = 0.300991
Ітерація 4: max_delta = 0.186853
Ітерація 5: max_delta = 0.124018
Ітерація 6: max_delta = 0.083077
Ітерація 7: max_delta = 0.055862
Ітерація 8: max_delta = 0.037653
Ітерація 9: max_delta = 0.025432
Ітерація 10: max_delta = 0.017217
Ітерація 11: max_delta = 0.011689
Ітерація 12: max_delta = 0.007967
Ітерація 13: max_delta = 0.005458
Ітерація 14: max_delta = 0.003765

```
  Ітерація 15: max_delta = 0.002621
  Ітерація 16: max_delta = 0.001846
  Ітерація 17: max_delta = 0.001319
  Ітерація 18: max_delta = 0.000960
  Ітерація 19: max_delta = 0.000713
  Ітерація 20: max_delta = 0.000543
⚠ Досягнуто max_iter=20, збіжність не гарантована
⌛ Розрахунок інформаційної функції...
✓ test_theta_range: PASSED (theta: [-4.000, 4.000])
⚠ Увага: вибірка 10 студентів замала для надійного IRT (рекомендовано ≥ 100).
  Параметри IRT будуть наближеними.
⌛ EM-алгоритм IRT (max 10 ітерацій)...
  Ітерація 1: max_delta = 1.595120
  Ітерація 2: max_delta = 0.000000
✓ Збіжність досягнута на ітерації 2
⌛ Розрахунок інформаційної функції...
✓ test_theta_ordering: PASSED (theta[all_correct]=3.000 > theta[none_correct]=-3.000)
⌛ EM-алгоритм IRT (max 10 ітерацій)...
  Ітерація 1: max_delta = 1.717091
  Ітерація 2: max_delta = 0.760894
  Ітерація 3: max_delta = 0.300991
  Ітерація 4: max_delta = 0.186853
  Ітерація 5: max_delta = 0.124018
  Ітерація 6: max_delta = 0.083077
  Ітерація 7: max_delta = 0.055862
  Ітерація 8: max_delta = 0.037653
  Ітерація 9: max_delta = 0.025432
  Ітерація 10: max_delta = 0.017217
⚠ Досягнуто max_iter=10, збіжність не гарантована
⌛ Розрахунок інформаційної функції...
⌛ EM-алгоритм IRT (max 30 ітерацій)...
  Ітерація 1: max_delta = 1.717091
  Ітерація 2: max_delta = 0.760894
  Ітерація 3: max_delta = 0.300991
  Ітерація 4: max_delta = 0.186853
  Ітерація 5: max_delta = 0.124018
  Ітерація 6: max_delta = 0.083077
  Ітерація 7: max_delta = 0.055862
  Ітерація 8: max_delta = 0.037653
  Ітерація 9: max_delta = 0.025432
  Ітерація 10: max_delta = 0.017217
  Ітерація 11: max_delta = 0.011689
  Ітерація 12: max_delta = 0.007967
  Ітерація 13: max_delta = 0.005458
  Ітерація 14: max_delta = 0.003765
  Ітерація 15: max_delta = 0.002621
  Ітерація 16: max_delta = 0.001846
  Ітерація 17: max_delta = 0.001319
  Ітерація 18: max_delta = 0.000960
  Ітерація 19: max_delta = 0.000713
  Ітерація 20: max_delta = 0.000543
  Ітерація 21: max_delta = 0.000424
  Ітерація 22: max_delta = 0.000341
  Ітерація 23: max_delta = 0.000280
  Ітерація 24: max_delta = 0.000246
```

```
  Ітерація 25: max_delta = 0.000222
  Ітерація 26: max_delta = 0.000202
  Ітерація 27: max_delta = 0.000185
  Ітерація 28: max_delta = 0.000170
  Ітерація 29: max_delta = 0.000157
  Ітерація 30: max_delta = 0.000145
⚠ Досягнуто max_iter=30, збіжність не гарантована
⏳ Розрахунок інформаційної функції...
✓ test_em_convergence: PASSED (delta_a=0.0179, delta_b=0.0394)
⏳ EM-алгоритм IRT (max 20 ітерацій)...
  Ітерація 1: max_delta = 1.717091
  Ітерація 2: max_delta = 0.760894
  Ітерація 3: max_delta = 0.300991
  Ітерація 4: max_delta = 0.186853
  Ітерація 5: max_delta = 0.124018
  Ітерація 6: max_delta = 0.083077
  Ітерація 7: max_delta = 0.055862
  Ітерація 8: max_delta = 0.037653
  Ітерація 9: max_delta = 0.025432
  Ітерація 10: max_delta = 0.017217
  Ітерація 11: max_delta = 0.011689
  Ітерація 12: max_delta = 0.007967
  Ітерація 13: max_delta = 0.005458
  Ітерація 14: max_delta = 0.003765
  Ітерація 15: max_delta = 0.002621
  Ітерація 16: max_delta = 0.001846
  Ітерація 17: max_delta = 0.001319
  Ітерація 18: max_delta = 0.000960
  Ітерація 19: max_delta = 0.000713
  Ітерація 20: max_delta = 0.000543
⚠ Досягнуто max_iter=20, збіжність не гарантована
⏳ Розрахунок інформаційної функції...
✓ test_information_function_positive: PASSED

Результат: 10/10 пройдено

============================================================
KST + Fi + Крайні випадки
============================================================
✓ test_kst_dag_valid: PASSED
✓ test_kst_dag_cycle_detected: PASSED
✓ test_kst_structural_correctness_base_concept: PASSED
✓ test_kst_structural_correctness_dependent_concept: PASSED
✓ test_kst_structural_incorrectness_missing_prerequisite: PASSED
✓ test_kst_no_concept_incorrectness: PASSED
✓ test_kst_coverage_report: PASSED (покрито 2/3)
✓ test_kst_scores_range: PASSED (scores=[0.667 0.667 0.667 0.   ])
✓ test_fi_weights_sum_to_one: PASSED
✓ test_fi_formula_manual: PASSED (Fi=0.710)
✓ test_fi_classification_thresholds: PASSED
✓ test_fi_range: PASSED (100 випадкових комбінацій)
✓ test_edge_all_correct: PASSED (alpha=0.000)
✓ test_edge_all_wrong: PASSED
✓ test_edge_single_student: PASSED
```

⚠ Увага: вибірка 15 студентів замала для надійного IRT (рекомендовано ≥ 100).
Параметри IRT будуть наближеними.
⚠ 2 завдань мають P=0 або P=1 — IRT для них встановить граничні значення.
⏳ EM-алгоритм IRT (max 5 ітерацій)...
Ітерація 1: max_delta = 1.816065
Ітерація 2: max_delta = 0.138373
Ітерація 3: max_delta = 0.014670
Ітерація 4: max_delta = 0.007764
Ітерація 5: max_delta = 0.003972
⚠ Досягнуто max_iter=5, збіжність не гарантована
⏳ Розрахунок інформаційної функції...
✓ test_edge_irt_constant_no_crash: PASSED
✓ test_edge_negative_alpha_no_crash: PASSED (alpha=0.000)

Результат: 17/17 пройдено

==
ЗАГАЛЬНИЙ РЕЗУЛЬТАТ: 36/36 тестів пройдено
✓ Всі тести пройдено успішно
==

📄 Звіт збережено: C:\Users\eliza\reports\test_report.txt
==
ЗВІТ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
Система аналізу банку тестових завдань курсу «Бази даних»
Дата: 31.05.2026 09:48
==

ЗАГАЛЬНІ РЕЗУЛЬТАТИ:
Всього тестів: 36
Пройдено: 36 (100.0%)
Не пройдено: 0 (0.0%)

ДЕТАЛІЗАЦІЯ ПО МОДУЛЯХ:

✓ CTT — Класична теорія тестів: 9/9
✓ test_p_values_known [0.2 мс]
✓ test_p_values_range [0.9 мс]
✓ test_discrimination_positive_for_good_items [3.1 мс]
✓ test_discrimination_constant_item_zero [0.8 мс]
✓ test_discrimination_corrected_lower_than_naive [6.6 мс]
✓ test_cronbach_alpha_range_normal [0.1 мс]
✓ test_cronbach_alpha_negative [0.1 мс]
✓ test_cronbach_alpha_single_item [0.0 мс]
✓ test_partial_scores_normalization [0.2 мс]

✓ IRT — Item Response Theory (2PL модель): 10/10
✓ test_item_params_bounds [4815.1 мс]
✓ test_difficulty_ordering [4934.7 мс]
✓ test_constant_items_detected [0.6 мс]
✓ test_constant_items_boundary_params [245.8 мс]
✓ test_icc_monotone [5019.9 мс]

```
  ✓ test_icc_p_at_b_equals_05                    [5292.4 мс]
  ✓ test_theta_range                             [5650.0 мс]
  ✓ test_theta_ordering                          [38.7 мс]
  ✓ test_em_convergence                          [13130.6 мс]
  ✓ test_information_function_positive           [6297.5 мс]

✓ KST + Fi + Крайні випадки: 17/17
  ✓ test_kst_dag_valid                           [2.1 мс]
  ✓ test_kst_dag_cycle_detected                  [0.1 мс]
  ✓ test_kst_structural_correctness_base_concept       [0.9 мс]
  ✓ test_kst_structural_correctness_dependent_concept  [0.1 мс]
  ✓ test_kst_structural_incorrectness_missing_prerequisite [0.1 мс]
  ✓ test_kst_no_concept_incorrectness            [0.0 мс]
  ✓ test_kst_coverage_report                     [0.0 мс]
  ✓ test_kst_scores_range                        [0.4 мс]
  ✓ test_fi_weights_sum_to_one                   [0.0 мс]
  ✓ test_fi_formula_manual                       [0.0 мс]
  ✓ test_fi_classification_thresholds            [0.0 мс]
  ✓ test_fi_range                                [2.3 мс]
  ✓ test_edge_all_correct                        [0.6 мс]
  ✓ test_edge_all_wrong                          [0.1 мс]
  ✓ test_edge_single_student                     [0.1 мс]
  ✓ test_edge_irt_constant_no_crash              [257.7 мс]
  ✓ test_edge_negative_alpha_no_crash            [0.2 мс]

======================================================================

ЩО ПЕРЕВІРЯЛОСЬ:

CTT (Класична теорія тестів):
  - Розрахунок P-value на матриці з відомими значеннями
  - Corrected point-biserial D-index (усунення item-total bias)
  - Коефіцієнт Кронбаха alpha, включаючи від'ємні значення
  - Нормалізація часткових балів через max_scores

IRT (Item Response Theory, модель 2PL):
  - Параметри a та b у межах заданих bounds
  - Впорядкованість параметра b відповідно до складності
  - Виявлення та обробка константних завдань (P=0, P=1)
  - MAP-регуляризація: граничні параметри для P=0/P=1
  - Монотонність ICC кривих
  - P(θ=b) = 0.5 (математична властивість 2PL)
  - Збіжність EM-алгоритму
  - Невід'ємність інформаційної функції

KST (Knowledge Space Theory):
  - Перевірка DAG-властивості графа передумов
  - Виявлення циклів у графі
  - Структурна коректність завдань
  - Виявлення непокритих передумов
  - Звіт покриття концептів

Інтегральний показник Fi:
```

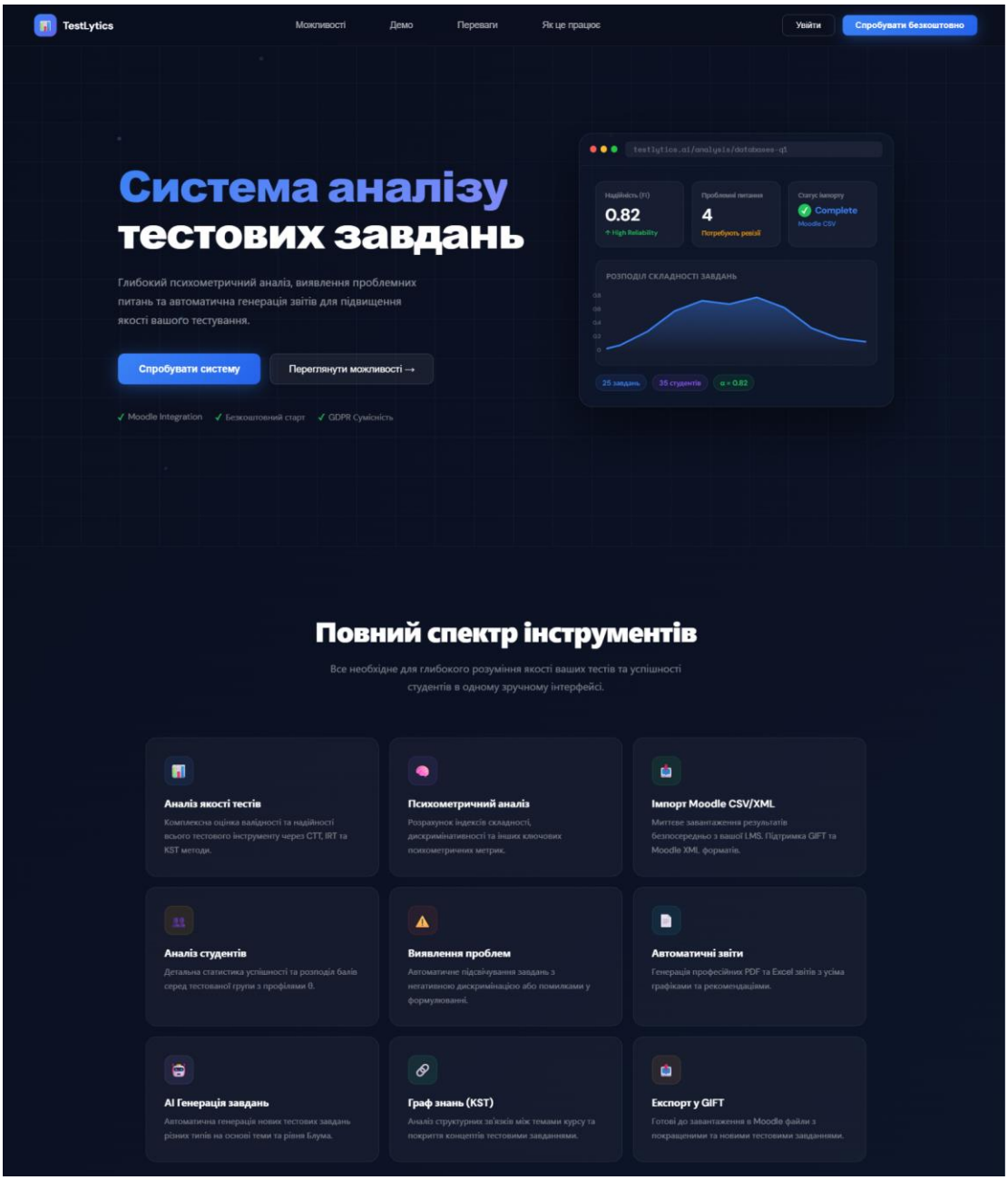
- Сума ваг = 1.0 (стандартний і адаптивний режими)
- Ручна перевірка формули Fi на відомих значеннях
- Коректність класифікації за порогами 0.40/0.70
- Діапазон Fi ∈ [0, 1] для 100 випадкових комбінацій

Крайні випадки:
- Матриця де всі відповіді правильні (P=1.0 для всіх)
- Матриця де всі відповіді неправильні (P=0.0 для всіх)
- Матриця з одним студентом
- IRT з константними завданнями без аварійного завершення
- Від'ємний Cronbach alpha без аварійного завершення

==========

ДОДАТОК Д. Скріншоти інтерфейсу системи

Рисунок Д.1 — Лендінг-сторінка системи з Hero-секцією та демонстраційною карткою



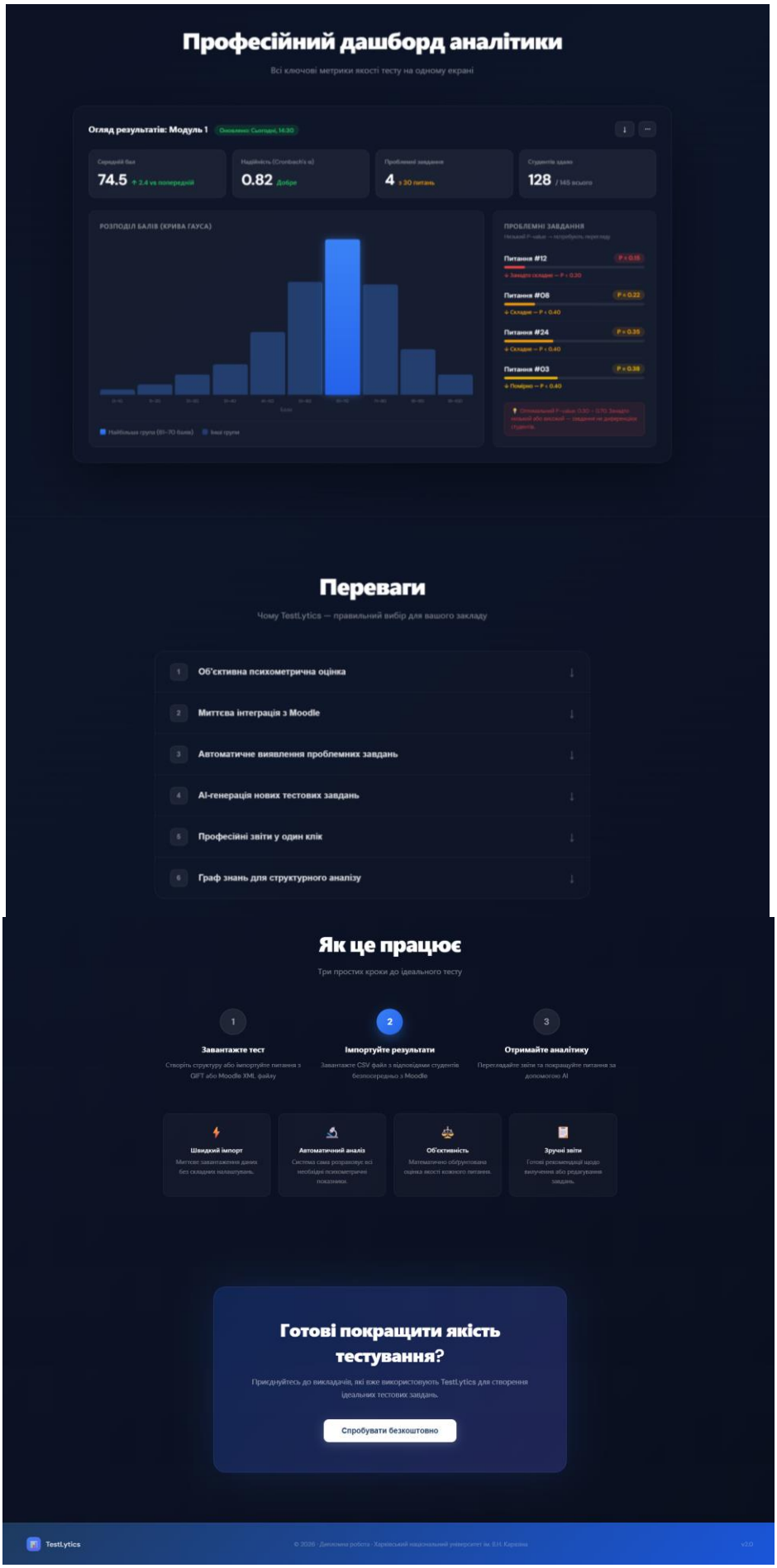

Професійний дашборд аналітики
Всі ключові метрики якості тесту на одному екрані
Огляд результатів: Модуль 1
Оновлено: Сьогодні, 14:30
Середній бал
74.5
↑ 2.4 vs попередній
Надійність (Cronbach's α)
0.82
Добре
Проблемні завдання
4
з 30 питань
Студентів здало
128
/ 145 всього
РОЗПОДІЛ БАЛІВ (КРИВА ГАУСА)
Бали
Найбільша група (61–70 балів)
Інші групи
ПРОБЛЕМНІ ЗАВДАННЯ
Низький P-value — потребують перегляду
Питання #12
P = 0.15
Занадто складне — P < 0.20
Питання #08
P = 0.22
Складне — P < 0.40
Питання #24
P = 0.35
Складне — P < 0.40
Питання #03
P = 0.38
Помірно — P < 0.40
Оптимальний P-value: 0.30 – 0.70. Занадто низький або високий — завдання не диференціює студентів.
Переваги
Чому TestLytics — правильний вибір для вашого закладу
1 Об'єктивна психометрична оцінка
2 Миттєва інтеграція з Moodle
3 Автоматичне виявлення проблемних завдань
4 AI-генерація нових тестових завдань
5 Професійні звіти у один клік
6 Граф знань для структурного аналізу
Як це працює
Три простих кроки до ідеального тесту
1
Завантажте тест
Створіть структуру або імпортуйте питання з GIFT або Moodle XML файлу
2
Імпортуйте результати
Завантажте CSV файл з відповідями студентів безпосередньо з Moodle
3
Отримайте аналітику
Перегладайте звіти та покращуйте питання за допомогою AI
Швидкий імпорт
Миттєве завантаження даних без складних налаштувань.
Автоматичний аналіз
Система сама розрахує всі необхідні психометричні показники.
Об'єктивність
Математично обґрунтована оцінка якості кожного питання.
Зручні звіти
Готові рекомендації щодо вилучення або редагування завдань.
Готові покращити якість тестування?
Приєднуйтесь до викладачів, які вже використовують TestLytics для створення ідеальних тестових завдань.
Спробувати безкоштовно
TestLytics
© 2026 · Дипломна робота · Харківський національний університет ім. В.Н. Каразіна
v2.0

Рисунок Д.2 — Сторінка автентифікації з вкладками входу та реєстрації

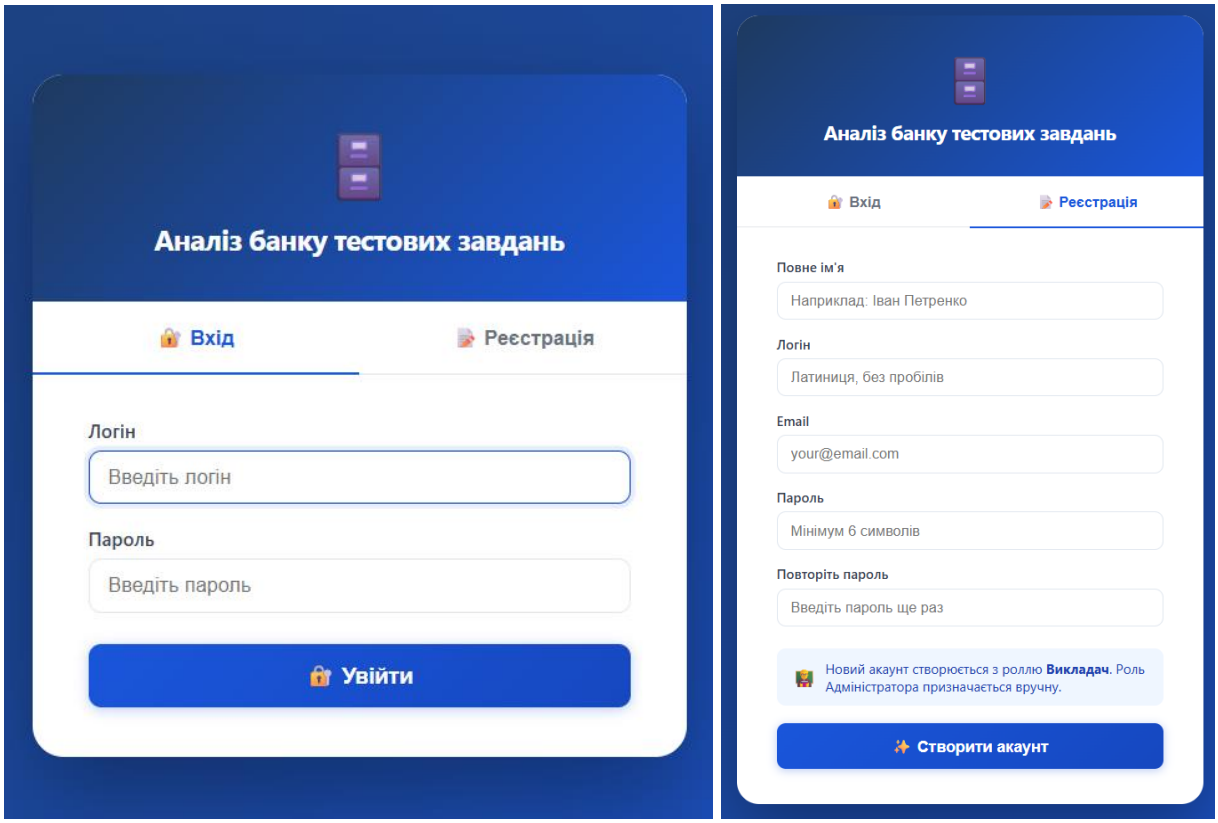
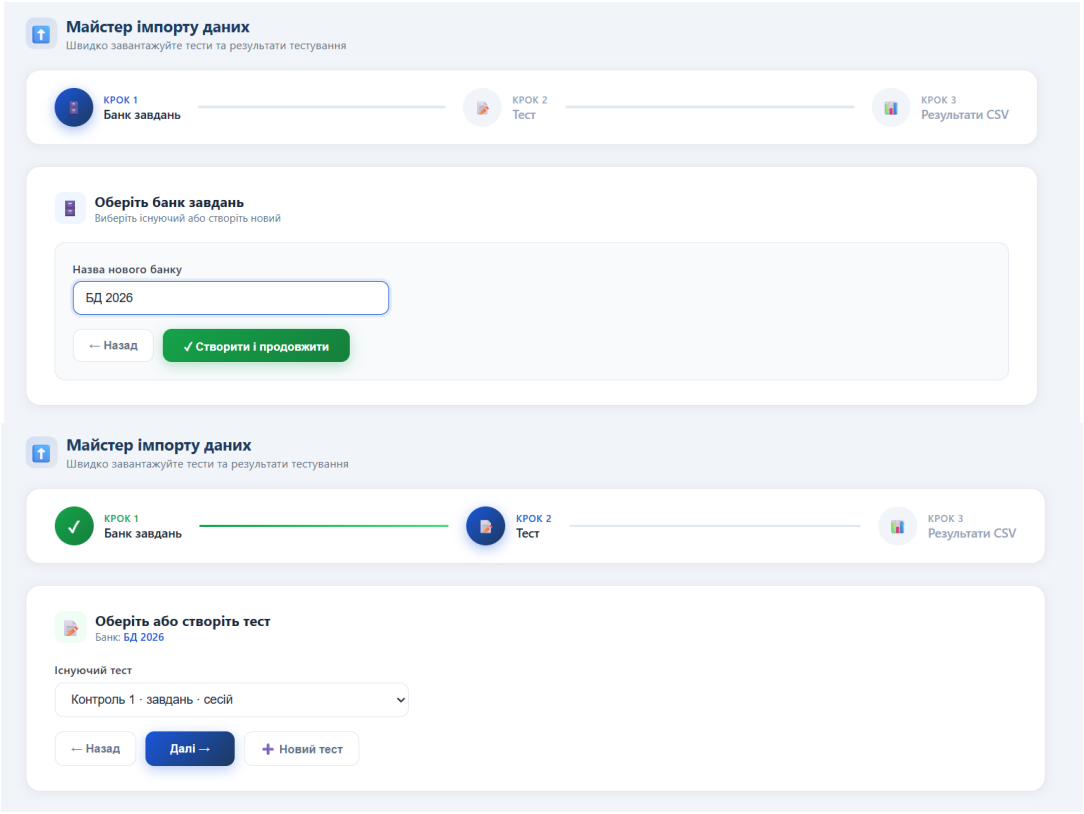


Рисунок Д.3 — Триступеневий майстер імпорту даних



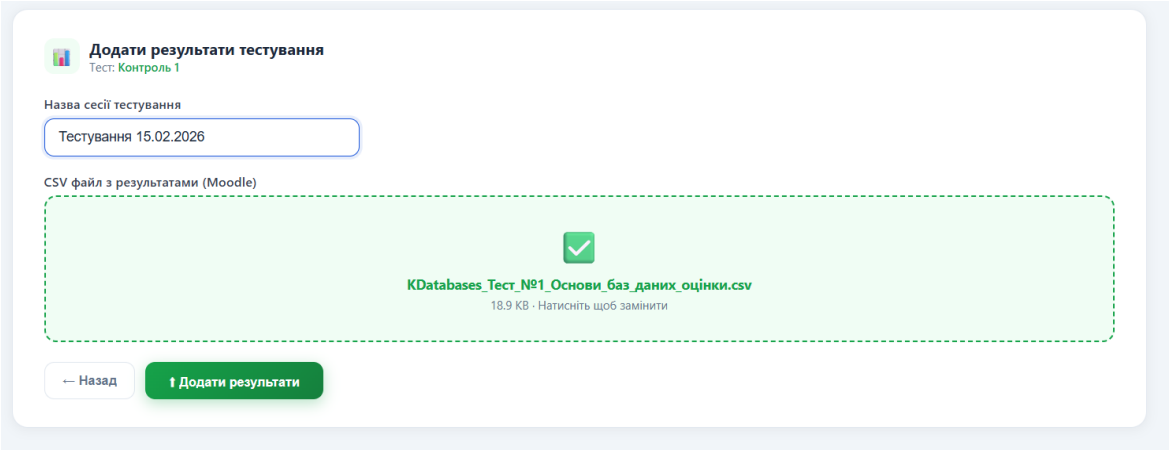


Рисунок Д.4 — Сторінка налаштувань вагових коефіцієнтів показника Fi

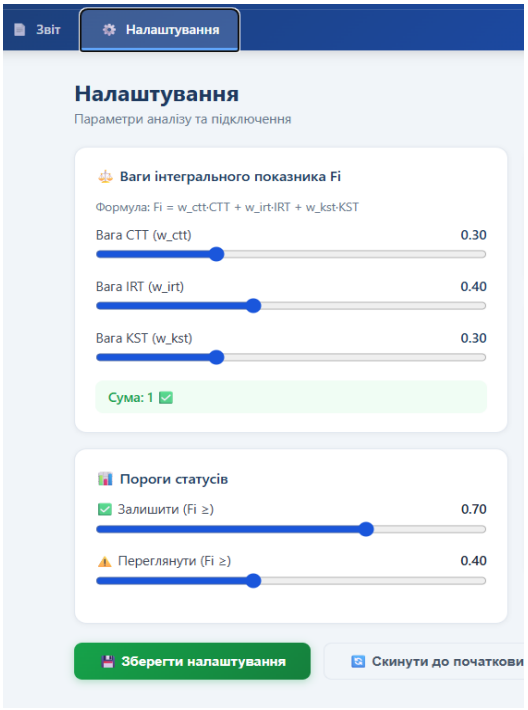


Рисунок Д.5 — Сторінка ReportPage забезпечує генерацію звітності у форматах Excel та PDF.

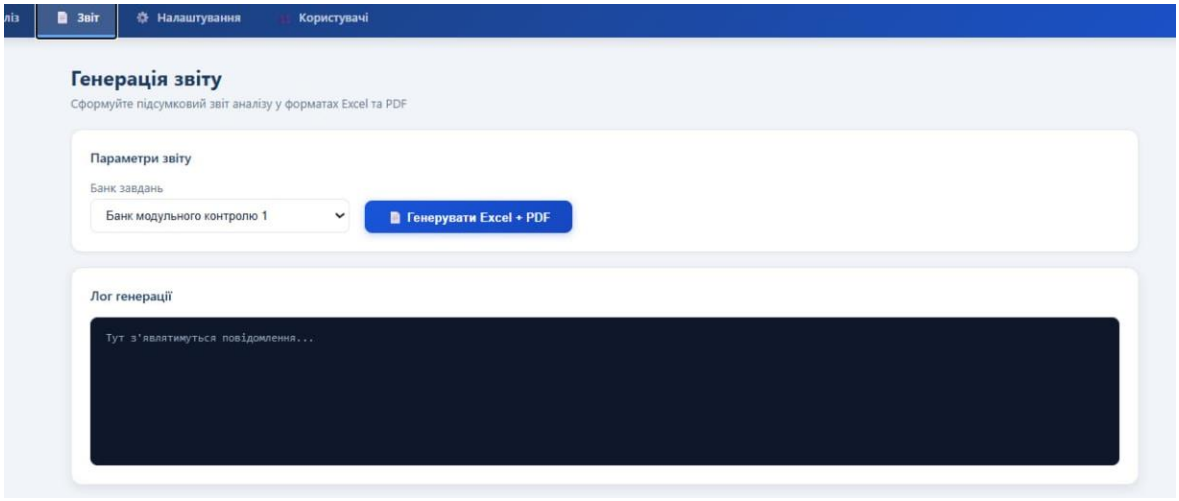


Рисунок Д.6 — Згенерований PDF-звіт по банку тестових завдань.

Звіт аналізу банку тестових завдань

Банк: New Bank

Дата: 31.05.2026 11:28

Зведена статистика

Показник	Значення
Всього завдань	25
Залишити (Fi≥0.70)	11
Переглянути (0.40-0.70)	12
Видалити (Fi<0.40)	2
Середній Fi	0.666
Cronbach α	0.744
Покриття тем	11%

Детальний аналіз завдань

ID	Текст	Fi	Статус
1808	Розгляньте таблицю результатів виконання процедури sp_GetSta...	0.375	Видалити
1822	Як викликати процедуру spGetUser з параметром @id = 5 ? (нап...	0.382	Видалити
1809	Що поверне виконання цієї процедури? CREATE PROCEDURE MyProc...	0.503	Переглянути
1806	Заповніть пропуски в синтаксисі створення процедури: {1:SHOR...	0.538	Переглянути
1802	Встановіть відповідність між поняттям та його визначенням у ...	0.557	Переглянути
1811	Розглянете процедуру: CREATE PROCEDURE spInsertOrder @produc...	0.560	Переглянути
1817	Оберіть правильні твердження щодо властивостей збережених пр...	0.561	Переглянути
1810	Оберіть правильні твердження щодо транзакцій у збережених пр...	0.562	Переглянути
1803	Встановіть відповідність між опціями збережених процедур та ...	0.567	Переглянути
1805	Заповніть пропуски в описі параметрів процедури: Ім'я параме...	0.569	Переглянути
1804	Встановіть відповідність між типом збереженої процедури та ї...	0.580	Переглянути
1823	Яку функцію використовують для перевірки існування процедури...	0.584	Переглянути
1801	Перетягніть правильні фрагменти, щоб отримати процедуру з ви...	0.597	Переглянути
1807	Заповніть пропуски щодо глобальних змінних T-SQL: Змінна {1:...	0.618	Переглянути
1813	Яка глобальна функція T-SQL повертає поточний рівень вкладен...	0.730	Залишити
1812	Розгляньте таблицю результатів виконання процедури sp_GetSta...	0.748	Залишити
1816	Що поверне наступний SQL запит? CREATE PROCEDURE spGetEmploy...	0.841	Залишити
1814	Яке з наступних тверджень про збережені процедури є правильн...	0.842	Залишити
1825	Збережені процедури дозволяють виконувати операції з даними ...	0.843	Залишити
1820	Яке основне призначення збережених процедур у системах керув...	0.843	Залишити
1815	Розгляньте наступну збережену процедуру: CREATE PROCEDURE sp...	0.847	Залишити
1819	Який тип збереженої процедури використовується для виконання...	0.848	Залишити

Продовження рисунку Д.6

1818	Яка основна перевага використання збережених процедур у база...	0.850	Залишити
1821	Знайдіть помилку в наступному SQL запиті CREATE PROCEDURE sp...	0.850	Залишити
1824	Збережені процедури можуть бути використані для реалізації б...	0.850	Залишити

Теми без завдань

- Базові поняття
- БД та інформаційні системи
- СКБД
- Моделі баз даних
- CASE засоби BPWin
- CASE засоби ERWin
- Реляційна алгебра
- Нормалізація
- Реляційна модель даних
- Первинний та зовнішній ключ
- Нормалізація 1НФ-2НФ
- Нормалізація 3НФ-БКНФ
- SELECT запити
- SQL запити. Створення таблиць
- Функції T-SQL
- T-SQL Індекси
- JOIN операції
- Агрегатні функції
- Підзапити
- View (представлення)
- Поняття курсору
- Поняття тригерів
- XML в SQL Server
- Поняття індексів
- Захист інформації в базах даних