Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

# КВАЛІФІКАЦІЙНА РОБОТА

**на здобуття ступеня вищої освіти бакалавра**

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка веб-орієнтованої системи адаптивного тестування на основі теорії відгуку завдань

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студентка гр. ІПЗ 22-2
М. І. Кірніцька

Керівник
кваліфікаційної роботи _____ А. М. Стрюк

Завідувач кафедри _____ А. М. Стрюк

Кривий Ріг
2026

Криворізький національний університет
Факультет: Інформаційних технологій
Кафедра: Моделювання та програмного забезпечення
Ступінь вищої освіти: бакалавр
Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри
_____ А. М. Стрюк
«___» _____ 20__ р.

**ЗАВДАННЯ**
**на кваліфікаційну роботу**

студентки групи ІПЗ-22-2 Кірницької Маргарити Ігорівн

1. Тема: Розробка веб-орієнтованої системи адаптивного тестування на основі теорії відгуку завдань затверджено наказом по КНУ № 22 від «13» лютого 2026 р.
2. Термін подання студентом закінченої роботи: «15» червня 2026 р.
3. Вихідні дані по роботі: розроблювана система повинна функціонувати в умовах асинхронного веб-середовища з використанням NoSQL СУБД MongoDB; обчислювальне ядро має забезпечувати ітераційний перерахунок латентного параметра знань у межах від -3 до +3 логітів; максимальний час обробки адаптивного кроку на сервері не повинен перевищувати 200 мс під масовим навантаженням; інтерфейс користувача має підтримувати стандарти веб-доступності.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): виконати аналіз існуючих методів та психометричних підходів до автоматизованого контролю знань, вибрати метод адаптивного тестування респондентів на основі логістичних моделей теорії відгуку завдань (IRT), спроектувати автоматизовану веб-орієнтовану систему адаптивного контролю знань, здійснити програмну реалізацію розробленої системи із використанням асинхронного стеку Python та NoSQL СУБД MongoDB, провести

тестування розробленої системи та експериментально верифікувати алгоритми локалізації латентних параметрів.

5. Перелік ілюстративного матеріалу: архітектурні схеми та UML-діаграми (прецедентів, послідовності кроку, класів), блок-схеми розроблених алгоритмів валідації та ітераційного перерахунку параметрів, схема взаємодії модулів системи та документоорієнтованої бази даних, знімки екранних форм інтерфейсу Single Page Application.

Календарний план:

| № | Найменування етапів кваліфікаційної роботи | Термін виконання етапів роботи |
|---|---|---|
| 1 | Огляд наукової літератури за тематикою адаптивного тестування та збір вихідних даних | |
| 2 | Проведення порівняльного аналізу математичних методів оцінювання | |
| 3 | Проведення порівняльного аналізу математичних моделей відгуку завдань | |
| 4 | Підготовка матеріалів 1 розділу роботи | |
| 5 | Розроблення математичного забезпечення для ітераційного перерахунку рівня знань за обраною моделлю | |
| 6 | Підготовка матеріалів 2 розділу роботи | |
| 7 | Розроблення програмного забезпечення веб-платформи адаптивного контролю знань.. Тестування функціональності | |
| 8 | Підготовка матеріалів 3 розділу роботи | |
| 10 | Оформлення пояснювальної записки, формування додатків та лістингу програмного коду | |

Дата видачі завдання: «15» лютого 2026 р.

Студентка _____ М. І. Кірніцька

Керівник роботи _____ А. М. Стрюк

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface ;

ASGI – Asynchronous Server Gateway Interface;

BSON – Binary JSON ;

CSS – Cascading Style Sheets ;

DOM – Document Object Model ;

HTML – HyperText Markup Language ;

HTTP – HyperText Transfer Protocol ;

IRT – Item Response Theory ;

JSON – JavaScript Object Notation ;

NoSQL – Not Only SQL;

REST – Representational State Transfer ;

SPA – Single Page Application ;

UI – User Interface ;

UML – Unified Modeling Language ;

URL – Uniform Resource Locator ;

UX – User Experience.

# РЕФЕРАТ

ЕМОЦІЯ, РОЗПІЗНАВАННЯ ЕМОЦІЙ, СИСТЕМА АДАПТИВНОГО НЕЙРОНЕЧІТКОГО ВИВЕДЕННЯ, НЕЧІТКІ ДАНІ, АВТОМАТИЗОВАНА СИСТЕМА, ЗОБРАЖЕННЯ ЛУДИНИ.

Пояснювальна записка: 67 с., 7 табл., 12 рис., 1 дод., 25 джерел.

Метою кваліфікаційної роботи є створення математичного та програмного забезпечення автоматизованої системи визначення емоційного стану людини за її зображенням, що функціонує в умовах роботи з нечіткими даними із мінімальною точністю розпізнавання емоцій 70%.

У роботі проведено аналітичний огляд та аналіз існуючих методів розв’язання задачі, психометричних підходів та інструментальних засобів оцінювання. Виконано порівняння класичних та адаптивних моделей з погляду точності розпізнавання, швидкодії, інженерної ефективності під масовим навантаженням та пристосованості методів до використання нечітких даних. Для розв’язання задачі в роботі вибрано метод адаптивного визначення станів на основі математичного апарату теорії відгуку та логістичних функцій.

Здійснено об'єктно-орієнтоване моделювання логічної архітектури, динаміки взаємодії компонентів за допомогою інструментарію UML-діаграм та розроблено алгоритми багаторівневої валідації вхідних повідомлень. Розроблено автоматизовану веб-орієнтовану систему на базі асинхронного стеку Python (FastAPI/ASGI) та документоорієнтованого NoSQL сховища MongoDB із денормалізацією даних, що реалізує обраний метод. Виконано комплексне тестування та експериментальну верифікацію розробленої системи, яка підтвердила високу точність локалізації станів за 5 кроків при часі відгуку сервера від 15 до 40 мілісекунд.

## ABSTRACT

EMOTION, EMOTIONS RECOGNITION, ADAPTIVE NEUROFUZZY INDUCTION SYSTEM, FUZZY DATA, AUTOMATED SYSTEM, HUMAN IMAGE.

Explanatory note: 67 p.,7 tables, 12 figures, 1 appendix, 25 sources.

The purpose of the qualification work is to create mathematical and software support for an automated system for determining a person's emotional state from his image, which functions in conditions of working with fuzzy data with a minimum accuracy of emotion recognition of 70%.

The work provides an analytical review and analysis of existing methods for solving the problem, psychometric approaches and instrumental assessment tools. A comparison of classical and adaptive models is made in terms of recognition accuracy, speed, engineering efficiency under mass load and adaptability of methods to the use of fuzzy data. To solve the problem in the work, the method of adaptive state determination based on the mathematical apparatus of response theory and logistic functions was selected.

Object-oriented modeling of logical architecture, dynamics of component interaction using the UML diagram toolkit was carried out and algorithms for multi-level validation of incoming messages were developed. An automated web-oriented system based on the asynchronous Python stack (FastAPI/ASGI) and the document-oriented NoSQL MongoDB storage with data denormalization, which implements the selected method, was developed. Comprehensive testing and experimental verification of the developed system were performed, which confirmed the high accuracy of state localization in 5 steps with a server response time of 15 to 40 milliseconds.

**ЗМІСТ**

ВСТУП ..... 9

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА МЕТОДІВ АДАПТИВНОГО КОНТРОЛЮ ЗНАНЬ ..... 12

1.1. Аналіз психометричних підходів до оцінювання знань: класична теорія тестів та теорія відгуку завдань ..... 12

1.2. Дослідження існуючих програмних рішень, платформ та LMS-модулів для комп'ютерного тестування ..... 14

1.3. Огляд математичних моделей IRT: однопараметрична модель Раша та багатопараметричні логістичні функції ..... 17

1.4.Обґрунтування вибору технологічного стеку для розробки веб-платформи ..... 19

1.5. Формулювання технічних та функціональних вимог до веб-орієнтованої системи адаптивного тестування ..... 22

Висновки до розділу ..... 26

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ АДАПТИВНОГО ТЕСТУВАННЯ ..... 28

2.1 Розроблення технічного завдання ..... 28

2.2Моделювання функціональних аспектів системи ..... 31

2.3Проєктування архітектури та структури програмних класів ..... 33

2.4Розробка алгоритмів багаторівневої валідації повідомлень ..... 35

2.5Проєктування бази даних ..... 39

2.6Проєктування графічного інтерфейсу та навігації користувача ..... 40

Висновки до розділу ..... 43

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ..... 44

3.1 Організація середовища розробки та структура проєкту ..... 44

3.2Реалізація модулів та ітераційного перерахунку рівня знань студента .. 48

3.3Взаємодія з базою даних на основі драйверів та реалізація гнучких схем документів ..... 51

3.4Розробка інтерфейсу користувача та відображення результатів ..... 53

3.5 Тестування функціональності веб-платформи та верифікація коректності адаптивного алгоритму ..... 56

3.6Аналіз результатів впровадження механізмів динамічної адаптації тестів ..... 59
Висновки до розділу ..... 62
ВИСНОВКИ ..... 64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 66

# ВСТУП

**Актуальність теми.** Модернізація сучасної системи вищої освіти та стрімкий перехід до дистанційних форм навчання зумовлюють підвищення вимог до інструментів контролю знань студентів. Традиційні комп'ютерні тести, що базуються на класичній теорії тестів, мають суттєвий інженерний недолік – лінійну архітектуру, де всі користувачі проходять однаковий фіксований набір завдань, а фінальна оцінка є звичайним підсумовуванням балів. Такий підхід створює значне психоемоційне навантаження на респондентів, не враховує індивідуальний темп підготовки та демонструє високу взаємозалежність параметрів тесту від вибірки досліджуваних.

Альтернативою, що забезпечує високу наукову точність та гнучкість, є сучасна теорія тестів, або теорія відгуку завдань (Item Response Theory, IRT). Вона дозволяє розробити динамічні алгоритми для автоматичного підбору складності кожного наступного запитання у реальному часі на основі латентного рівня знань студента (θ). Проте сучасний ринок програмного забезпечення відчуває дефіцит легких, масштабованих та кросплатформених веб-рішень такого типу, оскільки поширені LMS (наприклад, Moodle) архітектурно застаріли для динамічних обчислень IRT під високим навантаженням. Таким чином, розробка високоефективної веб-орієнтованої платформи адаптивного тестування з використанням асинхронного стеку Python (FastAPI) та NoSQL СУБД MongoDB є актуальним інженерно-технічним завданням.

**Мета дослідження**. Метою кваліфікаційної роботи є проектування, програмна реалізація та тестування веб-орієнтованої платформи адаптивного контролю знань на основі математичного апарату теорії відгуку завдань для забезпечення об'єктивного оцінювання користувачів у режимі реального часу.

**Об'єктом дослідження є** процес автоматизованого комп'ютерного контролю та оцінювання знань користувачів в електронних освітніх середовищах.

**Предметом дослідження** є методи, моделі та алгоритми теорії відгуку завдань (IRT) для динамічного перерахунку латентних параметрів, а також архітектурні принципи та програмні засоби реалізації веб-орієнтованої системи адаптивного тестування.

**Завдання роботи.** Для досягнення поставленої мети необхідно вирішити такі завдання:

1.a.i.1. Провести аналітичний огляд існуючих теоретичних і практичних рішень у галузі комп'ютерного тестування та обґрунтувати вибір математичних моделей IRT.

1.a.i.2. Обґрунтувати вибір асинхронного технологічного стеку програмування та документоорієнтованої структури бази даних.

1.a.i.3. Спроектувати архітектуру програмного двигуна адаптивності та розробити схеми даних для збереження психометричних паспортів завдань.

1.a.i.4. Здійснити програмну реалізацію серверної частини платформи (REST API) та інтерактивного графічного веб-інтерфейсу з інструментами кастомізації доступності.

1.a.i.5. Провести комплексне тестування розробленого програмного забезпечення та проаналізувати стабільність алгоритму перерахунку латентних параметрів.

**Галузь застосування результатів.** Отримані результати та розроблена веб-платформа AdaptiveTest Platform можуть бути впроваджені в освітній процес закладів вищої освіти для проведення поточного, рубіжного та підсумкового контролю знань студентів, а також у корпоративному секторі для автоматизованого асинхронного оцінювання та сертифікації персоналу.

# РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА МЕТОДІВ АДАПТИВНОГО КОНТРОЛЮ ЗНАНЬ

## 1.1. Аналіз психометричних підходів до оцінювання знань: класична теорія тестів та теорія відгуку завдань

Еволюція систем контролю знань у педагогіці та інженерії програмного забезпечення пройшла шлях від суб'єктивного оцінювання екзаменатором до спроб створення повністю автоматизованих, незалежних інструментів вимірювання. На сьогоднішній день у світі психометрики – науки, що займається теорією та методикою педагогічних і психологічних вимірювань – домінують два кардинально різні філософські та математичні підходи. Перший, історично раніший, отримав назву класичної теорії тестів (Classical Test Theory, CTT). Другий, який став відповіддю на інженерні обмеження класичного підходу, відомий як сучасна теорія тестів, або теорія відгуку завдань (Item Response Theory, IRT). Розуміння різниці між ними є критично важливим для проектування інтелектуальних систем адаптивного тестування, адже саме математичний апарат визначає архітектуру майбутнього програмного двигуна [1].

Класична теорія тестів, що формувалася протягом усього двадцятого століття, базується на інтуїтивно зрозумілій моделі: спостережуваний результат студента на іспиті є сумою його реального (істинного) рівня знань та випадкової похибки вимірювання [2]. Головна метрика CTT – це так званий «первинний бал», тобто банальна кількість правильних відповідей. Незважаючи на простоту реалізації в коді, цей підхід має фундаментальну проблему – взаємозалежність вибірки досліджуваних та параметрів тесту. Складність запитання в класичній теорії визначається відсотком людей, які на нього відповіли правильно. Це означає, що якщо один і той самий тест дати групі відмінників, то всі питання система визначить як «легкі», а якщо групі слабких студентів – ті самі питання стануть «важкими». Відповідно, оцінити реальний, абсолютний рівень підготовки людини за такою шкалою

неможливо, оскільки оцінка повністю залежить від того, наскільки складний варіант тесту їй дістався.

Теорія відгуку завдань (IRT) пропонує абсолютно інший погляд на проблему, переносячи фокус із усього тесту на кожне конкретне запитання окремо. В основі IRT лежить ідея про те, що ймовірність правильної відповіді студента на певне завдання є математичною функцією, яка залежить від взаємодії двох сил: прихованого (латентного) рівня знань студента та внутрішніх характеристик самого запитання. Найголовніша перевага IRT полягає в інваріантності її показників. Це означає, що обчислені параметри складності запитань не залежать від того, хто саме проходив тест, а оцінений рівень знань студента не залежить від того, які саме за складністю питання йому підкидав алгоритм [3-5]. IRT дозволяє вимірювати знання та складність в одній і тій самій метричній шкалі (найчастіше від -3 до +3 логітів), що робить результати тестування максимально точними та порівнюваними.

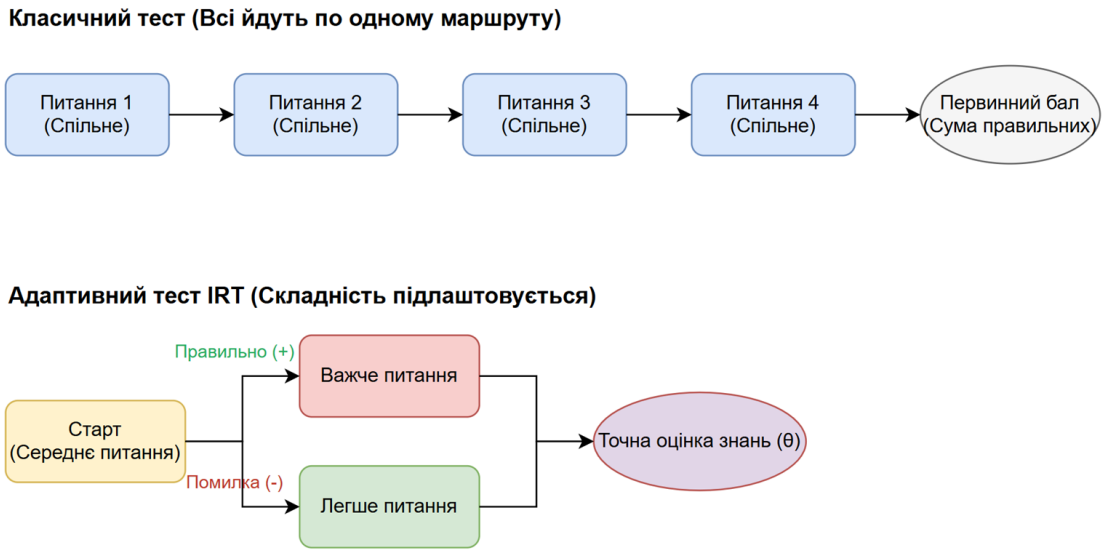


Рисунок 1.1 – Концептуальне порівняння класичного та адаптивного тестування

Для практичної реалізації веб-платформи вибір між CTT та IRT є очевидним на користь останньої. Класична теорія в принципі не здатна забезпечити роботу адаптивних систем, адже вона вимагає, щоб усі

користувачі проходили один і той самий фіксований набір завдань для розрахунку надійності. Натомість IRT є ідеальним математичним ядром для динамічних алгоритмів. Оскільки кожне питання в базі даних MongoDB має свій заздалегідь прорахований паспорт складності, система отримує можливість у реальному часі, буквально після кожної відповіді, перераховувати поточний рівень знань користувача. Якщо студент помиляється, алгоритм миттєво знижує планку складності, шукаючи межу його реальних можливостей [6-10]. Це не тільки підвищує об'єктивність оцінки, але й кардинально зменшує стрес для студента, адаптуючи систему під його індивідуальний темп та інтелектуальні можливості.

## 1.2. Дослідження існуючих програмних рішень, платформ та LMS-модулів для комп'ютерного тестування

Сучасний ринок програмного забезпечення для організації контролю знань пропонує широкий спектр рішений – від глобальних систем керування навчанням (LMS) до вузькоспеціалізованих сервісів. Для визначення архітектурних вимог до проектованої системи необхідно провести детальний аналіз існуючих аналогів, оцінити їхній математичний інструментарій, масштабованість, інтерфейсні рішення та здатність підтримувати адаптивні алгоритми [11]. Це дозволить виявити технологічні «вузькі місця» наявного софту та обґрунтувати необхідність розробки власного веб-застосунку.

Найбільш поширеним класом рішень в академічному середовищі є універсальні LMS з відкритим кодом, серед яких абсолютним лідером є платформа Moodle. Стандартний модуль тестування Moodle має колосальну функціональність: підтримує десятки типів запитань, випадкову генерацію білетів із банку завдань та гнучке налаштування часових обмежень [12-15]. Проте з інженерного погляду його підсистема оцінювання повністю базується на класичній теорії тестів (CTT). Система просто підсумовує бали за правильні відповіді. Спроби інтегрувати адаптивність у Moodle зазвичай

обмежуються створенням «уроків» із розгалуженою логікою, де викладач має вручну прописувати траєкторію для кожного варіанта відповіді. Це створює колосальне навантаження на контент-менеджера і не має нічого спільного з автоматичним математичним моделюванням на основі IRT. Крім того, важка монолітна архітектура Moodle та використання реляційних баз даних (MySQL/PostgreSQL) суттєво уповільнюють динамічний перерахунок латентних параметрів у реальному часі при високому навантаженні на сервер.

Інший полюс програмних рішень – це комерційні хмарні платформи та SaaS-сервіси, такі як Kahoot, Quizizz або Google Форми. Їхньою головною перевагою є сучасний, інтуїтивно зрозумілий веб-інтерфейс, висока швидкість роботи та відмінна адаптація під мобільні пристрої. Вони ідеально підходять для швидкого зрізу знань або гейміфікації навчання. Однак ці сервіси мають закритий початковий код і мінімальний аналітичний інструментарій. Вони повністю ігнорують складність запитань як латентний параметр і не дозволяють створювати гнучкі адаптивні сценарії [16-20]. Використання таких платформ в рамках офіційної атестації у вищій школі є неможливим через відсутність інструментів суворого контролю, неможливість локального розгортання на серверах університету для забезпечення безпеки персональних даних та жорсткі обмеження безкоштовних тарифних планів.

Для наочного порівняння архітектурних та функціональних можливостей існуючих систем та проектованої веб-платформи доцільно систематизувати їхні характеристики у вигляді таблиці. Це дозволить чітко продемонструвати переваги поєднання мови Python, СУБД MongoDB та математичного апарату теорії відгуку завдань.

Таблиця 1.1 – Порівняльний аналіз існуючих рішень для комп'ютерного тестування

| Критерії | Moodle | Google Форми | Проектована система |
|---|---|---|---|
| Математична модель | Класична (CTT) | Класична (CTT) | Сучасна теорія тестів (IRT, модель Раша) |
| Тип адаптивності | Ручне розгалуження | Відсутня | Автоматичний перерахунок складності двигуном системи |
| Архітектура бази даних | Реляційна (SQL) | Хмарна (Proprietary | Нереляційна (MongoDB, гнучкі JSON-документи завдань) |
| Програмний стек | PHP | Java/JS | Python (FastAPI/Flask) - оптимізований під обчислення |
| Можливість кастомізації та інтеграції | Висока (через плагін) | Низька (обмежений API) | Абсолютна (повний контроль вихідного коду програми) |

Проведений аналіз показує, що на ринку існує чітко виражений технологічний дефіцит. З одного боку, наявні потужні академічні LMS є занадто консервативними й архітектурно застарілими для підтримки обчислювальних алгоритмів IRT. З іншого боку, легкі веб-сервіси жертвують науковою точністю оцінювання заради візуальної привабливості.

Таким чином, розробка власної веб-орієнтованої платформи на базі Python та MongoDB є повністю виправданою. Вона дозволить поєднати гнучкість та швидкість сучасних NoSQL-рішень із глибоким математичним апаратом теорії відгуку завдань, заповнюючи нішу ефективних та доступних систем адаптивного контролю знань.

### 1.3. Огляд математичних моделей IRT: однопараметрична модель Раша та багатопараметричні логістичні функції

Фундаментом теорії відгуку завдань є математичне моделювання, яке дозволяє пов'язати латентну (приховану) змінну рівня знань тесованого з імовірністю успішного виконання конкретного тестового завдання. На відміну від класичного підходу, де розглядаються лише лінійні бали, в IRT використовуються нелінійні логістичні функції, які відображають реальну природу процесу тестування. Математичний апарат теорії оперує поняттям кривої завдання (Item Characteristic Curve, ICC), яка показує графічну залежність ймовірності успіху від рівня підготовки студента. Залежно від кількості врахованих чинників, що визначають властивості запитання, у сучасній психометриці виділяють три базові логістичні моделі, вибір яких безпосередньо впливає на алгоритмічну складність розроблюваної веб-платформи [21-22].

Найпростішою, але водночас найбільш математично строгою є однопараметрична логістична модель (1PL), концептуалізована данським математиком Георгом Рашем і відома як модель Раша. У цій моделі ймовірність правильної відповіді студента з рівнем знань Ѳ на завдання і залежить виключно від одного параметра – складності самого завдання (). Математично модель Раша описується такою формулою:

де e – основа натурального логарифма. Складність  та рівень знань вимірюються в одній шкалі логітів. Головне інженерне припущення моделі Раша полягає в тому, що всі завдання в тесті мають однакову роздільну здатність. Це значно спрощує архітектуру програмного двигуна на Python, оскільки для калібрування банку питань та перерахунку рівня знань студента методом максимальної правдоподібності потрібні мінімальні обчислювальні ресурси сервера, а самі алгоритми демонструють високу стабільність навіть на малих вибірках користувачів.

Для випадків, коли тестові завдання суттєво відрізняються за своєю якістю та здатністю диференціювати сильних і слабких студентів, використовується двопараметрична логістична модель (2PL), запропонована Алланом Бірнбаумом. Вона вводить додатковий латентний параметр – індекс розрізнення (дискримінації) завдання (), який визначає крутизну кривої ICC. Формула двопараметричної моделі має такий вигляд:

Якщо параметр розрізнення  є високим, це означає, що завдання дуже чітко розділяє студентів: ті, чий рівень знань хоч трохи вищий за складність , дадуть правильну відповідь із високою ймовірністю, а ті, хто недотягує – майже гарантовано помиляться. Впровадження 2PL-моделі у веб-орієнтовану систему підвищує точність адаптивного підбору, проте ускладнює структуру документів у MongoDB, адже для кожного запитання тепер необхідно зберігати та динамічно оновлювати два коефіцієнти, а алгоритм адаптації двигуна має враховувати інформаційну функцію завдання, яка тепер залежить і від крутизни графіку.

Найбільш комплексною є трипараметрична логістична модель (3PL), яка ідеально підходить для закритих тестів із вибором одного варіанта відповіді з кількох запропонованих. Вона враховує третій параметр – коефіцієнт вгадування (), який зсуває нижню асимптоту графіка функції вгору, показуючи ймовірність того, що навіть студент із нульовим рівнем знань випадково обере правильну відповідь. Математичний вираз моделі 3PL записується так:

Хоча модель 3PL є найбільш адекватною для реальних умов масового комп'ютерного тестування, її програмна реалізація в адаптивному двигуні створює серйозні інженерні виклики. Для стійкого математичного калібрування параметрів за цією моделлю потрібні величезні масиви історичних даних (від тисячі проходжень на кожне питання), що є критичним обмеженням для нових або локальних університетських систем.

Тому для проектованої веб-платформи оптимальним архітектурним рішенням є фокус на моделях 1PL та 2PL, що дозволить поєднати високу точність адаптації із гнучкістю та швидкістю роботи NoSQL бази даних MongoDB в умовах реального часу.

## 1.4. Обґрунтування вибору технологічного стеку для розробки веб-платформи

Ефективність функціонування веб-платформи адаптивного тестування значною мірою визначається правильністю вибору технологічного стеку, оскільки саме програмні засоби забезпечують швидкість обробки запитів, масштабованість системи та коректність реалізації математичних алгоритмів оцінювання знань користувачів. Особливістю розроблюваної системи є використання моделей теорії відгуку на завдання (Item Response Theory, IRT), що передбачає виконання численних обчислень у режимі реального часу під час проходження тестування. У зв’язку з цим до серверної частини висуваються підвищені вимоги щодо продуктивності, гнучкості роботи з даними та підтримки асинхронної обробки великої кількості одночасних запитів [23-25].

Під час проєктування програмної платформи було обрано стек технологій, який складається з мови програмування Python, веб-фреймворку FastAPI та документоорієнтованої системи керування базами даних MongoDB. Поєднання зазначених інструментів забезпечує високу швидкодію системи, простоту масштабування та можливість ефективної реалізації математичних моделей адаптивного тестування.

Основою серверної частини обрано мову програмування Python, яка є одним із найбільш поширених інструментів у галузі наукових обчислень, аналізу даних та штучного інтелекту. Головною перевагою Python є наявність розвиненої екосистеми спеціалізованих бібліотек, зокрема NumPy, SciPy та Pandas, що дозволяють виконувати складні математичні операції над векторами та матрицями з високою швидкістю.

У межах адаптивного тестування Python забезпечує ефективну реалізацію алгоритмів оцінювання рівня підготовки користувача. Імовірність правильної відповіді на завдання в двопараметричній моделі IRT описується логістичною функцією

де  – імовірність правильної відповіді на і-те завдання; а -те завдання; – параметр розрізнення завдання;  – параметр складності; heta – рівень знань користувача. Для визначення значення heta використовується метод максимальної правдоподібності, що передбачає багаторазове виконання ітераційних обчислень. Завдяки високорівневому синтаксису Python та оптимізованим математичним бібліотекам реалізація таких алгоритмів є значно простішою порівняно з використанням низькорівневих мов програмування.

Як серверний фреймворк було обрано FastAPI – сучасне рішення для створення високопродуктивних веб-застосунків та REST API. Однією з ключових переваг FastAPI є підтримка асинхронного програмування на основі конструкцій async/await. Такий підхід дозволяє ефективно використовувати системні ресурси та обробляти велику кількість одночасних HTTP-запитів без суттєвого зниження продуктивності.

У процесі проходження тестування кожна відповідь користувача ініціює виконання низки операцій: збереження результату, перерахунок рівня знань, визначення наступного завдання та формування відповіді клієнтському застосунку. Асинхронна архітектура FastAPI забезпечує виконання зазначених дій із мінімальними часовими затримками, що позитивно впливає на швидкодію системи та якість взаємодії користувача з платформою.

Додатковою перевагою FastAPI є використання бібліотеки Pydantic для автоматичної перевірки вхідних даних та генерації схем обміну повідомленнями. Це дозволяє зменшити кількість програмних помилок і підвищити надійність взаємодії між клієнтською та серверною частинами

системи. Крім того, фреймворк автоматично створює інтерактивну документацію OpenAPI (Swagger UI), що спрощує тестування та подальший супровід програмного забезпечення.

Для організації зберігання даних було обрано систему керування базами даних MongoDB. На відміну від традиційних реляційних СУБД, MongoDB використовує документоорієнтовану модель зберігання інформації у форматі BSON, що забезпечує високу гнучкість структури даних.

Необхідність використання саме такого підходу обумовлена особливостями банку тестових завдань. Різні типи запитань можуть містити текстові дані, графічні матеріали, математичні формули, мультимедійний контент, а також специфічні параметри моделей IRT. Застосування традиційної реляційної моделі потребувало б створення значної кількості взаємопов’язаних таблиць і складних операцій об’єднання даних. Натомість MongoDB дозволяє зберігати всю інформацію про окреме завдання в межах одного документа, що суттєво скорочує час вибірки та спрощує структуру бази даних.

Типова структура документа завдання може включати текст запитання, перелік варіантів відповідей, правильну відповідь та параметри моделі IRT:

де  є параметрами трьохпараметричної моделі IRT. Для забезпечення максимальної продуктивності доступ до бази даних реалізовано через асинхронний драйвер Motor, який повністю інтегрується з архітектурою FastAPI. Завдяки цьому операції читання та запису не блокують виконання інших запитів, що є особливо важливим під час одночасного проходження тестування великою кількістю користувачів.

Таким чином, обраний технологічний стек Python – FastAPI – MongoDB забезпечує оптимальне поєднання математичної

функціональності, високої продуктивності, масштабованості та гнучкості зберігання даних. Використання зазначених технологій створює надійну основу для реалізації веб-платформи адаптивного тестування, здатної ефективно працювати в умовах значного навантаження та забезпечувати оперативне оцінювання рівня знань користувачів у режимі реального часу.

## 1.5. Формулювання технічних та функціональних вимог до веб-орієнтованої системи адаптивного тестування

Успішна реалізація інженерного проєкту у сфері програмної інженерії потребує чіткого структурування та деталізації вимог до майбутнього програмного продукту ще на етапі передпроєктного аналізу. Це дозволяє сформувати архітектурний каркас системи, мінімізувати ризики виникнення помилок при кодуванні та забезпечити повну відповідність розроблюваної веб-платформи потребам кінцевих користувачів. Згідно з методологією розробки програмного забезпечення, всі вимоги поділяються на дві великі категорії: функціональні, що описують безпосередню поведінку системи та обов'язки її модулів, та технічні (нефункціональні), які визначають якісні характеристики розроблюваного софту, такі як продуктивність, безпека та надійність.

Функціональні вимоги до веб-платформи адаптивного тестування мають враховувати специфіку взаємодії трьох основних рольових моделей користувачів: студента (респондента), викладача (модератора контенту) та адміністратора системи. Для ролі студента ключовою вимогою є забезпечення стабільного та інтуїтивно зрозумілого інтерфейсу проходження адаптивного тесту, де кожне наступне вікно із запитанням генерується динамічно на основі обчислень математичного двигуна IRT. Система повинна підтримувати механізм фіксації відповідей, відображення прогресу тестування та фінальну візуалізацію результатів із детальною аналітикою вибору складності. Для ролі викладача необхідно реалізувати

повноцінний CRUD-інтерфейс керування банком завдань, де кожному питанню, окрім контенту, можна вручну або автоматично присвоювати психометричні паспорти – коефіцієнти складності та дискримінації для логістичних моделей.

Таблиця 1.2 – Функціональні вимоги до веб-орієнтованої системи адаптивного тестування

| ID | Роль (Модуль) | Опис функціональної вимоги |
|---|---|---|
| FR-01 | Студент (Респондент) | Стабільне відображення інтуїтивно зрозумілого веб-інтерфейсу для проходження адаптивного тесту |
| FR-02 | Студент (IRT) | Динамічна генерація та виведення кожного наступного запитання на основі поточних відповідей |
| FR-03 | Студент (Респондент) | Надійне функціонування механізму автоматичної фіксації відповідей користувача в системі |
| FR-04 | Студент (Респондент) | Графічне відображення поточного прогресу проходження тесту в реальному часі |
| FR-05 | Студент (Респондент) | Фінальна візуалізація підсумкових результатів тестування із детальною аналітикою складності |
| FR-06 | Викладач (Модератор) | Надання повноцінного CRUD-інтерфейсу для повного керування банком тестових завдань |
| FR-07 | Викладач (Модератор) | Ручне або автоматичне присвоєння запитанням психометричних паспортів (коефіцієнтів , ) |

Технічні вимоги до платформи визначають технологічні обмеження та стандарти якості реалізації веб-застосунку. З погляду архітектури, система має бути побудована за принципом розподілу клієнтської та серверної частин (REST API), що забезпечить кросплатформеність і дозволить

комфортно проходити тести як з настільних комп'ютерів, так і з мобільних пристроїв під керуванням різних операційних систем. Серверна частина на Python (FastAPI) повинна мати час відгуку на запит генерації наступного завдання не більше ніж 200 мілісекунд, що досягається за рахунок асинхронної взаємодії з базою даних MongoDB.

Таблиця 1.3 – Технічні (нефункціональні) вимоги до системи

| ID | Категорія технічної вимоги | Специфікація та параметри вимоги |
|---|---|---|
| FR-01 | Системна архітектура | Побудова системи за принципом розподілу клієнтської та серверної частин (REST API) |
| FR-02 | Кросплатформеність | Комфортна робота у веб-браузерах настільних комп'ютерів та будь-яких мобільних пристроїв |
| FR-03 | Продуктивність (Backend) | Реалізація серверної логіки мовою Python на базі високопродуктивного фреймворку FastAPI |
| FR-04 | Швидкість відгуку | Час відповіді сервера на запит генерації нового завдання не повинен перевищувати 200 мс. |
| FR-05 | Архітектура даних | Організація сховища NoSQL MongoDB з асинхронною взаємодією для уникнення блокувань |
| FR-06 | Мережева безпека | Захист каналів зв'язку та передачі даних за допомогою криптографічного протоколу HTTPS |

| | | |
|---|---|---|
| FR-07 | Захист облікових даних | Обов'язкове шифрування паролів у базі даних за допомогою сучасних стійких хеш-функцій |
| FR-08 | Автентифікація | Безпечне ведення сесій користувачів на основі криптографічних JWT-токенів (JSON Web Token) |

Оскільки система оперує персональними даними студентів та результатами успішності, критичною вимогою є забезпечення безпеки: використання захищеного протоколу HTTPS, шифрування паролів за допомогою сучасних хеш-функцій та впровадження механізму автентифікації на основі токенів JWT.

**Висновки до розділу**

Було проведено комплексний аналітичний огляд предметної області, пов'язаної з автоматизацією та математичним моделюванням процесів комп'ютерного контролю знань. Критичний аналіз існуючих літературних джерел та програмних аналогів (таких як LMS Moodle та комерційні SaaS-платформи) показав, що сучасний освітній простір відчуває серйозний дефіцит легких, масштабованих та кросплатформених веб-рішень, які б використовували глибокий математичний аналіз для індивідуалізації тестування. Традиційні системи базуються на застарілій класичній теорії тестів, яка не здатна адаптуватися під латентний рівень підготовки конкретного студента у реальному часі.

Було детально вивчено математичний апарат теорії відгуку завдань (Item Response Theory), зокрема однопараметричну модель Раша та двопараметричну модель логістичних функцій Бірнбаума. Доведено, що використання IRT дозволяє отримати інваріантні оцінки знань студентів та

параметрів складності питань, що є ідеальним підґрунтям для створення динамічних адаптивних алгоритмів. На основі цього аналізу обґрунтовано вибір сучасного та високопродуктивного технологічного стеку, що включає мову програмування Python, асинхронний фреймворк FastAPI та документо-орієнтовану NoSQL СУБД MongoDB. Завершальним етапом розділу стало чітке формулювання функціональних та технічних вимог до системи, що дозволяє перейти до безпосереднього проєктування архітектури та алгоритмічних баз веб-платформи у наступному розділі роботи.

# РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ АДАПТИВНОГО ТЕСТУВАННЯ

## 2.1 Розроблення технічного завдання

Проєктування архітектури та безпосередня програмна реалізація веб-орієнтованої системи адаптивного контролю знань AdaptiveTest Platform потребує попереднього детального формалювання вимог, умов та обмежень розробки. Згідно з нормативними стандартами Єдиної системи програмної документації (ЄСПД), базовим документом, який визначає інженерні критерії, функціональний профіль та технічні параметри майбутнього програмного продукту, є Технічне завдання (ТЗ). Розроблення цього документу дозволяє концептуалізувати структуру платформи, мінімізувати ризики виникнення архітектурних колізій на етапах кодування та забезпечити відповідність фінального рішення критеріям надійності, масштабованості та доступності в сучасних електронних освітніх середовищах.

Основними підставами для розроблення системи є необхідність створення легкого, кросплатформеного веб-застосунку для проведення індивідуалізованого оцінювання знань на основі математичного апарату теорії відгуку завдань (IRT). Відповідно до вимог ЕСПД, програмний виріб призначений для автоматизації поточного й підсумкового оцінювання респондентів, зниження психоемоційного навантаження на студентів завдяки гнучкій оптимізації довжини тесту, а також для надання високоточної аналітичної інтерпретації латентного показника підготовки (θ) у реальному часі. Цільовими користувачами платформи визначено студентів (аспірантів), які проходять адаптивне тестування, та адміністраторів (викладачів), які здійснюють наповнення банку завдань та аналіз психометричних паспортів питань.

Технічні вимоги до платформи охоплюють низку критичних параметрів, серед яких першочергове значення мають вимоги до

функціональних характеристик. Програмне забезпечення повинно гарантувати стабільне функціонування трьох основних підсистем, а саме підсистеми автентифікації та реєстрації сесій за унікальним ідентифікатором користувача, підсистеми асинхронного розрахунку латентного рівня знань та підбору складності наступного питання, а також підсистеми інтерактивного відображення фінальних аналітичних результатів дослідження з візуалізацією на колірній шкалі логітів. ТЗ регламентує, що час відгуку серверної частини на запит генерації нового адаптивного завдання не повинен перевищувати нормативні двісті мілісекунд, що забезпечується використанням асинхронного фреймворку FastAPI та документоорієнтованої бази даних MongoDB.

У межах вимог до надійності, зафіксованих у технічному завданні, передбачено стійкість веб-застосунку до помилок введення даних користувачем та механізми автоматичного відновлення активних сесій тестування у випадку раптового розриву з'єднання на клієнтській стороні. Крім того, особлива увага приділяється умовам експлуатації та вимогам до програмно-апаратної сумісності. Серверна частина системи має бути інваріантною до операційної системи й розгортатися у будь-якому середовищі з підтримкою інтерпретатора Python версії 3.10 і вище, тоді як до клієнтської частини висувається вимога повної кросбраузерності та адаптивності, що гарантує коректне відображення інтерфейсу на мобільних і десктопних пристроях.

Важливим вектором проектування, закладеним у технічному завданні згідно зі специфікою сучасних веб-стандартів, є вимоги до ергономіки, технічної естетики та доступності інтерфейсу (Accessibility). Програмне забезпечення повинно підтримувати гнучку динамічну кастомізацію графічної оболонки безпосередньо користувачем у реальному часі. ТЗ визначає необхідність реалізації п'яти колірних тем оформлення, включаючи світлу класичну, темну нічну, кіберпанк, сепію та спеціальну висококонтрастну тему для людей з порушеннями зору, а також чотирьох

масштабованих рівнів розміру шрифту, що дозволяє адаптувати інтерфейс під індивідуальні фізіологічні потреби респондентів.

Таблиця 2.1 – Основні технічні вимоги до веб-платформи адаптивного тестування

| № | Категорія технічної вимоги | Специфікація та параметри вимоги |
|---|---|---|
| 1 | Системна архітектура | Побудова системи за принципом розподілу клієнтської та серверної частин (REST API) |
| 2 | Кросплатформеність | Комфортна робота у веб-браузерах настільних комп'ютерів та будь-яких мобільних пристроїв |
| 3 | Продуктивність (Backend) | Реалізація серверної логіки мовою Python на базі високопродуктивного фреймворку FastAPI |
| 4 | Швидкість відгуку | Час відповіді сервера на запит генерації нового завдання не повинен перевищувати 200 мс |
| 5 | Архітектура даних | Організація сховища NoSQL MongoDB з асинхронною взаємодією для уникнення блокувань |
| 6 | Мережева безпека | Захист каналів зв'язку та передачі даних за допомогою криптографічного протоколу HTTPS |
| 7 | Захист облікових даних | Обов'язкове шифрування паролів у базі даних за допомогою сучасних стійких хеш-функцій |

| | | |
|---|---|---|
| 8 | Автентифікація | Безпечне ведення сесій користувачів на основі криптографічних JWT-токенів (JSON Web Token) |
| 9 | Ергономіка та доступність | Підтримка 5 колірних тем (включаючи висококонтрастну) та 4 масштабів тексту у реальному часі |

Фінальна частина розробленого технічного завдання регламентує стадії та етапи розробки програмного продукту, процедури тестування та порядок контролю якості. Процес створення системи поділяється на етапи технічного проєктування, безпосереднього написання програмного коду модулів серверної та клієнтської частин, інтеграційного тестування асинхронного математичного двигуна IRT, а також експериментальної перевірки алгоритму перерахунку латентних параметрів. Сформоване таким чином технічне завдання слугує фундаментальним інженерним базисом для подальшого архітектурного моделювання та практичної реалізації елементів платформи AdaptiveTest Platform на наступних етапах роботи.

## 2.2Моделювання функціональних аспектів системи

Ефективна інженерна реалізація веб-орієнтованої системи адаптивного тестування AdaptiveTest Platform потребує детального абстрактного опису її функціональної структури та динаміки взаємодії внутрішніх компонентів. Для вирішення цього завдання в сучасній програмній інженерії застосовується уніфікована мова моделювання UML, яка дозволяє візуалізувати як статичні межі поведінки системи, так і послідовність виконання процесів у часі. Першочерговим етапом об’єктно-орієнтованого аналізу є побудова діаграми прецедентів (UML Use Case Diagram), яка фіксує зовнішні рольові моделі користувачів, їхні функціональні обов'язки та межі взаємодії з програмним додатком.

Основними дійовими особами (акторами) у проектованій системі виступають студент, викладач та адміністратор, чиї специфічні запити формують повний спектр архітектурних вимог до платформи. Студент взаємодіє з інтерфейсом з метою ініціалізації сесії, проходження адаптивного тесту, відправки відповідей та отримання фінальної аналітики, тоді як викладач наділений правами керування банком питань та перегляду психометричних паспортів завдань. Адміністратор системи забезпечує загальний моніторинг працездатності, контроль цілісності бази даних MongoDB та керування обліковими записами користувачів.

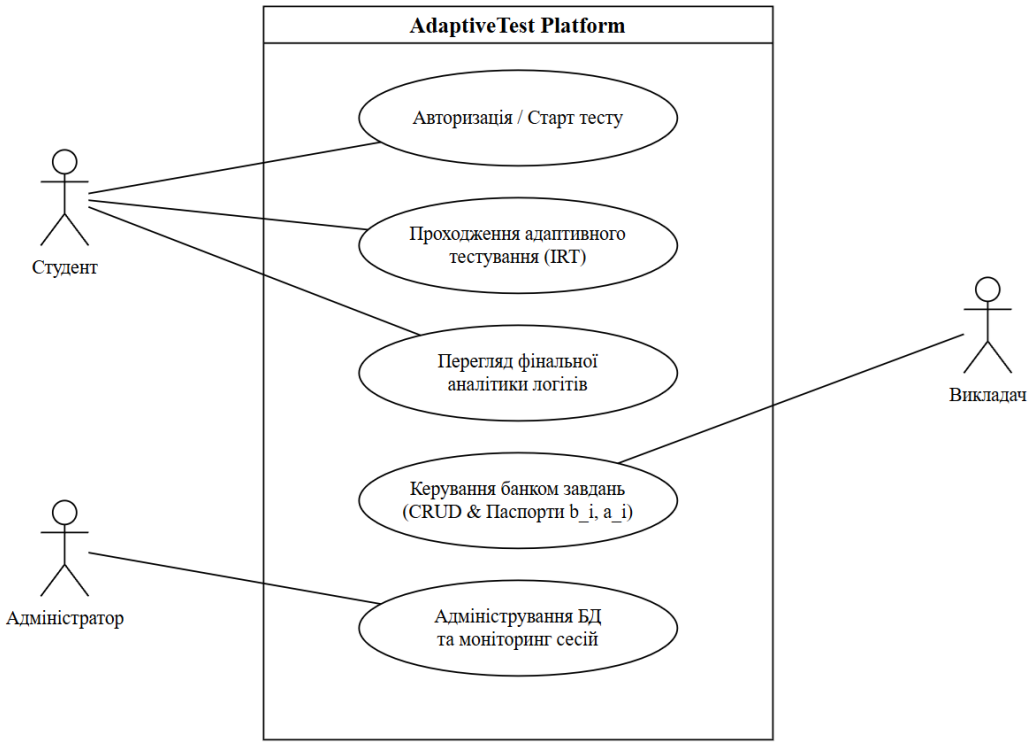


Рисунок 2.1 – Діаграма прецедентів

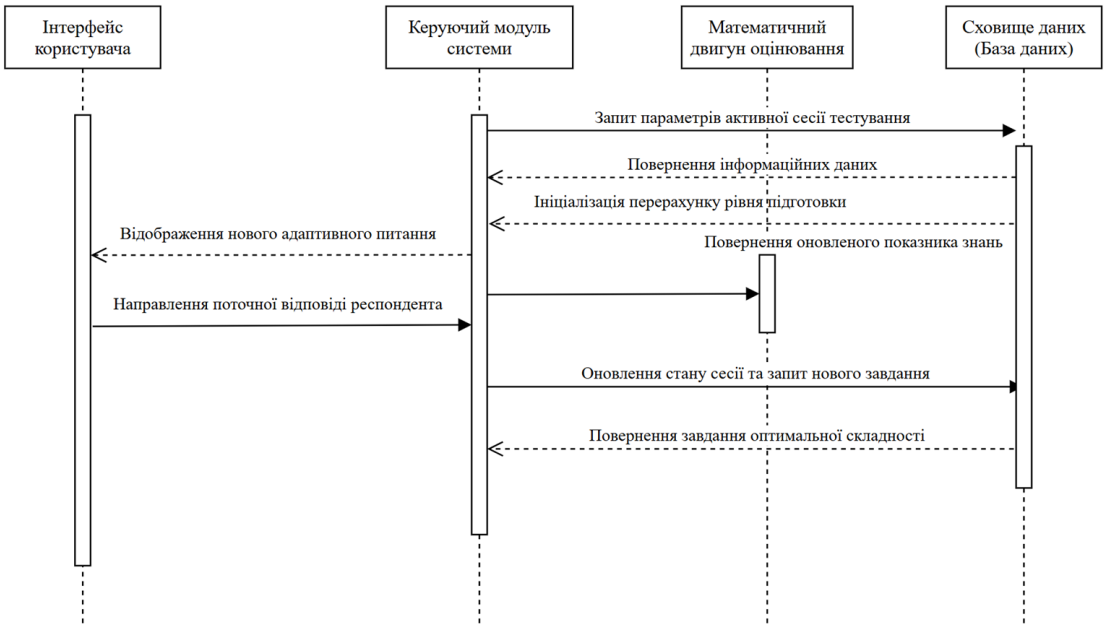


Рисунок 2.2 – Діаграма послідовності адаптивного кроку

Динамічні аспекти функціонування платформи, пов'язані з безпосереднім виконанням ітераційного алгоритму адаптивності, деталізуються за допомогою діаграми послідовності (UML Sequence Diagram). Цей тип моделювання дозволяє покроково відстежити часову лінійку передачі повідомлень та викликів методів між клієнтським веб-інтерфейсом, асинхронним сервером FastAPI та документоорієнтованим сховищем MongoDB. Процес розпочинається з відправки клієнтом запиту на старт сесії, у відповідь на що сервер створює унікальний документ у базі даних, ініціалізує початковий нульовий рівень знань (θ) та підбирає з банку перше нормативне завдання середньої складності. Ключова динаміка розгортається під час відправки студентом обраного варіанту відповіді, коли кожна дія ініціює ланцюжок асинхронних операцій, що включають збереження вибору, виклик обчислювального двигуна irt_engine для перерахунку тета, оновлення документа сесії та миттєвий підбір наступного запитання, параметр складності якого є максимально близьким до нового рівня підготовки респондента. Описане UML-моделювання функціональних аспектів забезпечує чітке розуміння поведінки системи та виступає точним

технологічним орієнтиром для безпосереднього проектування модульної структури класів та кодування REST API.

## 2.3Проєктування архітектури та структури програмних класів

Перехід від концептуального функціонального моделювання до безпосередньої програмної реалізації веб-орієнтованої системи потребує декомпозиції її простору на незалежні модулі та формалізації структури програмних класів. Проєктована платформа адаптивного тестування базується на архітектурному шаблоні розділення інтерфейсу користувача та сервісної логіки, де взаємодія між клієнтською і серверною частинами здійснюється через уніфікований інтерфейс прикладного програмування. Модульна архітектура серверної частини структурується за функціональним принципом і розподіляється на незалежні шари, що включають шар конфігурації та ініціалізації веб-сервера, шар валідації вхідних і вихідних даних, шар бізнес-логіки адаптивного обчислення та шар безпосередньої асинхронної взаємодії зі сховищем даних. Така ізоляція модулів мінімізує зв'язність програмного коду, забезпечує високу швидкість обробки транзакцій та спрощує подальше розширення системи шляхом додавання нових психометричних моделей без необхідності рефакторингу ядра платформи.

Основою статичної структури серверного компонента є проектування моделей валідації даних, які виступають строго типізованими об'єктами для обміну інформацією в межах архітектури. Для забезпечення цілісності сесій розробляються класи запитів та відповідей, які інкапсулюють у собі унікальні ідентифікатори респондентів, масиви текстових даних завдань та варіантів відповідей, а також поточні числові координати латентних параметрів. Клас, що відповідає за ведення поточної сесії тестування, динамічно акумулює у собі історію взаємодії користувача з платформою, зберігаючи послідовність пройдених питань, прапорці правильності

відповідей та масиви психометричних коефіцієнтів, що є необхідним для коректної роботи математичного двигуна. Обчислювальний модуль системи проектується як набір незалежних функціональних компонентів, що оперують числовими векторами та реалізують логістичні функції оцінювання ітераційним шляхом на основі накопиченого вектора відповідей.

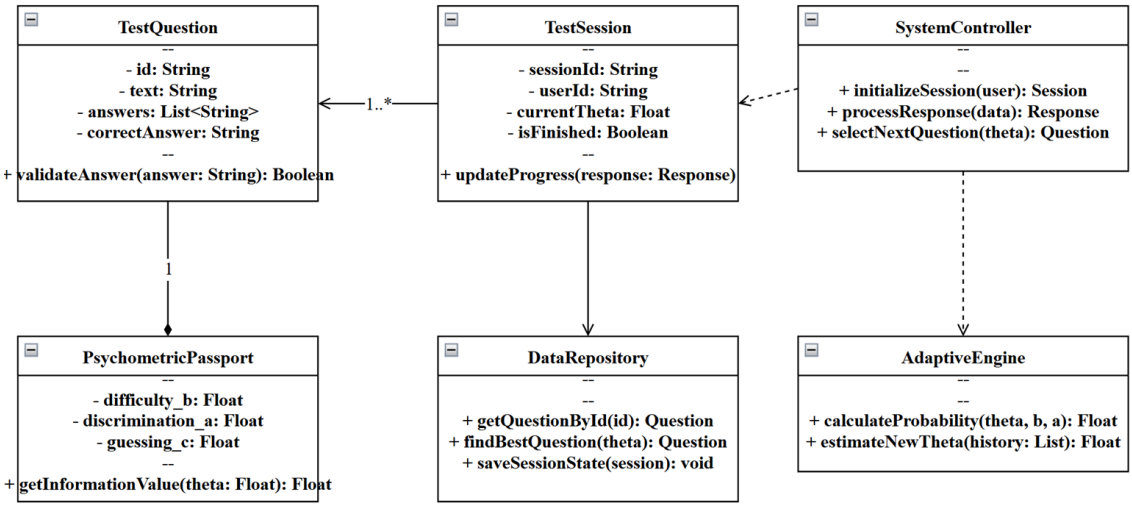


Рисунок 2.3 – UML-діаграма класів архітектури системи адаптивного тестування

Взаємодія з базою даних проєктується на основі асинхронного клієнта, який керує пулом з'єднань та виконує неблокуючі операції читання й запису документів. Структура класів доступу до даних орієнтована на роботу з двома основними сутностями сховища, якими є банк тестових завдань та реєстр активних і завершених сесій. Керуючий модуль системи координує роботу всіх підсистем, приймаючи вхідні структури даних від інтерфейсу, ініціюючи звернення до бази даних для перевірки стану сесії, викликаючи математичні методи перерахунку рівня знань та формуючи підсумковий об'єкт із наступним оптимальним завданням. Проєктування такої структури класів та модульних взаємодій дозволяє повністю реалізувати вимоги технічного завдання щодо швидкодії системи та створює надійну програмну основу для наступного етапу розробки адаптивного математичного алгоритму.

## 2.4Розробка алгоритмів багаторівневої валідації повідомлень

Забезпечення стійкості, надійності та безпеки веб-орієнтованої системи адаптивного тестування AdaptiveTest Platform в умовах інтенсивного мережевого обміну даними безпосередньо залежить від коректності обробки вхідних запитів. З огляду на те, що кожна дія користувача ініціює асинхронні математичні обчислення логістичних функцій теорії відгуку завдань, потрапляння у внутрішні модулі сервера некоректних, деформованих або зловмисних даних може призвести до системних збоїв, порушення цілісності бази даних та компрометації результатів тестування. Для нівелювання цих інженерних ризиків у межах проєктування системи розробляється алгоритм багаторівневої валідації повідомлень, який реалізує принцип послідовного синтаксичного, структурного та семантичного контролю даних на різних етапах їхнього проходження крізь архітектуру веб-застосунку. Такий підхід дозволяє відсікати помилкові запити на найраніших стадіях, не перевантажуючи обчислювальний двигун системи та забезпечуючи відповідність технічним вимогам щодо часу відгуку сервера в межах двохсот мілісекунд.

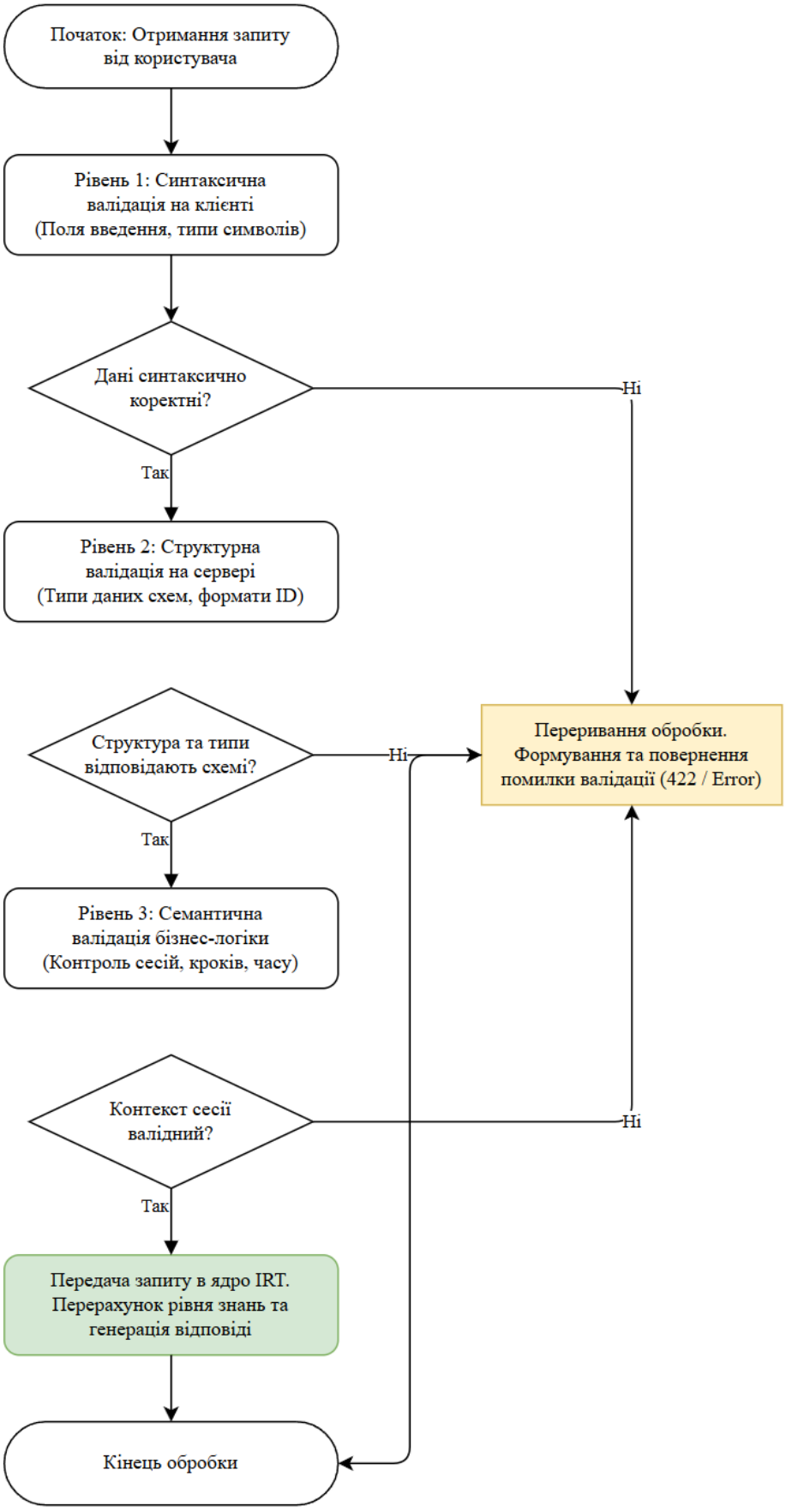


Рисунок 2.4 – Схема алгоритму багаторівневої валідації вхідних повідомлень

Перший рівень валідації повідомлень інтегрується безпосередньо на стороні клієнтського графічного інтерфейсу та виконує первинний синтаксичний контроль дій користувача. Головне завдання цього етапу полягає в перевірці заповнення обов'язкових полів введення, обмеженні типів символів для ідентифікаторів респондентів та блокуванні повторних відправок ідентичних запитів до моменту отримання відповіді від сервера. Другий рівень фільтрації розгортається на межі серверної частини веб-застосунку і являє собою структурну валідацію вхідних об'єктів за допомогою строго типізованих схем даних. Цей рівень автоматично аналізує структуру вхідного повідомлення, перевіряє відповідність типів даних, здійснює верифікацію форматів унікальних ідентифікаторів сесій і питань, а також контролює, щоб обраний варіант відповіді належав до дозволеного масиву рядкових значень.

Третій, найбільш глибокий рівень алгоритму, реалізує семантичну та контекстну валідацію безпосередньо всередині модуля бізнес-логіки перед зверненням до математичного ядра IRT. На цьому етапі система виконує перевірку логічної цілісності процесу тестування, що включає контроль часових міток сесії, верифікацію відповідності між поточним кроком тесту та надісланим ідентифікатором запитання, а також перевірку статусу завершеності поточної сесії в базі даних. Якщо вхідне повідомлення успішно проходить усі рівні фільтрації, система ініціює ітераційний перерахунок латентних знань та генерацію нового завдання. У разі виявлення невідповідності на будь-якому з етапів, алгоритм миттєво перериває обробку запиту, генерує стандартизоване виключення із детальним описом помилки валідації та повертає відповідний статус клієнтському інтерфейсу для інформування користувача.

## 2.5Проєктування бази даних

Надійне та високопродуктивне збереження інформації в системах адаптивного тестування вимагає гнучкого підходу до проектування структури бази даних. Оскільки процес тестування на основі теорії відгуку завдань (IRT) передбачає постійне накопичення динамічних масивів даних (історії відповідей респондентів, векторів психометричних коефіцієнтів та ітераційних змін латентних параметрів знань), класичні реляційні моделі з жорсткими схемами зв'язків та операціями об'єднання таблиць (JOIN) створюють додаткове обчислювальне навантаження на сервер. Для забезпечення мінімального часу відгуку системи та забезпечення горизонтального масштабування в межах даної роботи спроєктовано документоорієнтовану структуру бази даних, що базується на концепції NoSQL. Такий підхід дозволяє зберігати сутності у вигляді денормалізованих документів, структура яких може динамічно адаптуватися до змін без необхідності перебудови всієї схеми сховища.

Логічна архітектура бази даних складається з двох ключових колекцій, які забезпечують повний життєвий цикл функціонування платформи: колекції тестових завдань та колекції сесій тестування. Колекція тестових завдань проєктується як сукупність документів, кожен з яких інкапсулює в собі унікальний ідентифікатор питання, безпосередній змістовий текст завдання, масив доступних варіантів відповідей для користувача та еталонне значення правильної відповіді. Особливістю проєктування цієї сутності є інтеграція психометричного паспорта завдання безпосередньо в тіло документа у вигляді вкладеного об'єкта. Цей об'єкт містить три ключові параметри моделі Бірнбаума: коефіцієнт складності завдання, індекс дискримінації (розрізняльної здатності) та параметр ймовірності вгадування, що дозволяє математичному двигуну миттєво отримувати всі необхідні коефіцієнти під час одного звернення до сховища.

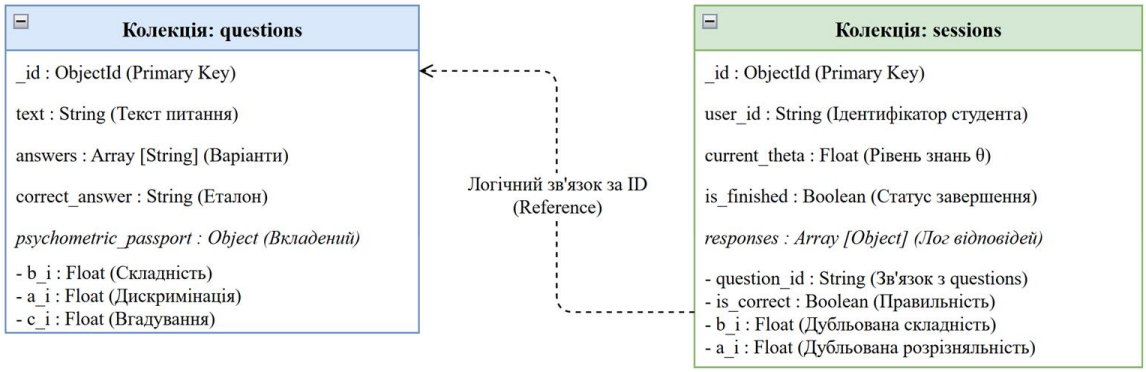


Рисунок 2.5 – Логічна схема документоорієнтованої структури бази даних NoSQL

Колекція сесій тестування відповідає за збереження стану процесу оцінювання в реальному часі та історичний аналіз дій респондентів. Документ сесії містить унікальні ідентифікатори самої сесії та користувача, поточне обчислене значення латентного рівня знань (тета), прапорець логічного завершення тесту, а також динамічний масив логів взаємодії. Кожен елемент цього масиву являє собою вкладений документ, який фіксує ідентифікатор пройденого питання, логічний результат відповіді (правильно чи неправильно) та точні числові параметри складності й розрізняльної здатності, що актуальні для цього питання. Денормалізація та дублювання психометричних параметрів усередині документа сесії є свідомим архітектурним рішенням, яке усуває потребу у повторних операціях пошуку за базою даних під час ітераційного перерахунку інтегральних функцій, забезпечуючи максимальну швидкодію системи під час масового одночасного тестування користувачів.

## 2.6 Проєктування графічного інтерфейсу та навігації користувача

Важливою умовою успішного впровадження веб-орієнтованої системи адаптивного тестування є створення ергономічного, інтуїтивно зрозумілого графічного інтерфейсу, який мінімізує когнітивне навантаження на респондента та дозволяє йому повністю зосередитися на розв'язанні

тестових завдань. Проєктування інтерфейсу (UI) та логіки користувацького досвіду (UX) базується на принципах мінімалізму, адаптивності до різних типів екранних пристроїв та чіткого візуального розділення функціональних зон. Головна мета навігаційної схеми полягає у забезпеченні безшовного лінійного переходу між ключовими станами системи, якими є етап ідентифікації користувача, безпосередній процес інтерактивної взаємодії з адаптивними питаннями та фінальний екран з аналітичними результатами оцінювання латентного рівня знань.

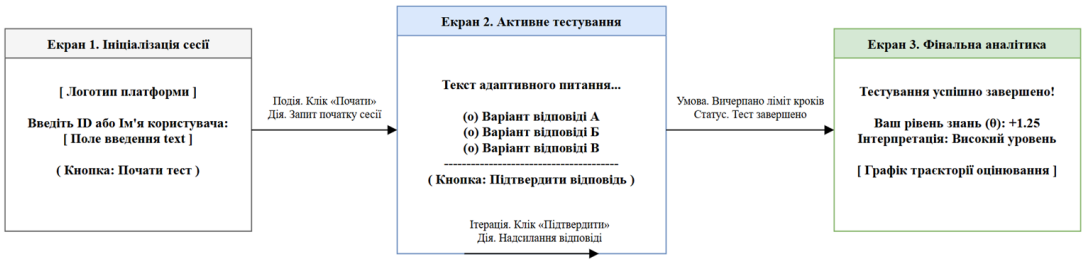


Рисунок 2.6 – Структурна схема екранів графічного інтерфейсу та навігації користувача

Екран ініціалізації сесії проєктується як лаконічна стартова форма, що містить елементи введення унікального ідентифікатора студента та керуючу кнопку запуску процесу. Основний робочий простір системи: інтерфейс проходження тесту, структурується за блочним принципом і розподіляється на три візуальні рівні. Верхня зона (хедер) виконує інформаційну функцію, відображаючи назву поточної дисципліни, лічильник пройдених кроків та прогрес-бар. Центральна область виділяється під динамічний контент, де розміщується формулювання поточного тестового завдання та радіокнопки для вибору одного з доступних варіантів відповіді. Нижня частина екрана містить інтерактивний елемент підтвердження дії, який блокується відразу після кліку для запобігання повторного надсилання ідентичних пакетів даних на сервер до моменту оновлення стану сторінки.

Завершальний етап взаємодії користувача з інтерфейсом реалізується у вигляді підсумкового аналітичного вікна, яке активується автоматично

після фіксації сервером прапорця закінчення тесту. Навігаційна логіка блокує можливість повернення до попередніх кроків тестування з метою захисту академічної доброчесності та забезпечення валідності психометричного оцінювання. Натомість екран результатів динамічно перебудовується для представлення фінального показника латентної підготовки (тета) у зрозумілому для респондента форматі, доповненому текстовою інтерпретацією рівня володіння матеріалом. Уся система переходів проєктується на основі односторінкової концепції, де оновлення графічних елементів відбувається асинхронно без повного перезавантаження веб-документа, що гарантує високу швидкість відгуку інтерфейсу та стабільність роботи за умов низької пропускної здатності мережі.

**Висновки до розділу**

У другому розділі кваліфікаційної роботи виконано повний комплекс науково-дослідних та інженерно-проєктних завдань, спрямованих на формування архітектурного і методологічного базису веб-орієнтованої системи адаптивного тестування AdaptiveTest Platform. На основі проведеного раніше аналізу психометричних підходів було формалізовано концептуальну архітектуру програмного комплексу, яка спирається на модульний принцип декомпозиції простору завдань. Це дозволило спроєктувати незалежні графічні компоненти, обчислювальне математичне ядро та уніфікований шар збереження даних. Розроблена структурна схема взаємодії екранів користувацького інтерфейсу та навігації респондента забезпечує мінімальне когнітивне навантаження під час тестування, реалізуючи безшовні асинхронні переходи між станами ініціалізації сесії, динамічного вибору варіантів відповідей та фінального представлення результатів оцінювання латентного рівня знань.

Важливим науково-практичним результатом розділу стала розробка та детальне моделювання логічної структури бази даних NoSQL. Обґрунтоване впровадження документоорієнтованої моделі на базі колекцій питань та тестових сесій дозволило ефективно інтегрувати психометричні паспорти завдань безпосередньо в тіло документів, а також реалізувати контрольовану денормалізацію даних логування відповідей. Таке архітектурне рішення повністю усуває необхідність виконання високовитратних операцій об'єднання таблиць на рівні сховища, забезпечуючи швидкий однократний доступ до коефіцієнтів моделей Бірнбаума під час ітераційного обчислення параметрів підготовки респондентів.

# РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

## 3.1 Організація середовища розробки та структура проєкту

Практичний етап інженерної реалізації веб-орієнтованої платформи адаптивного тестування розпочинається з комплексного розгортання середовища розробки та структурування компонентів програмного проєкту. Для створення надійної та ізольованої екосистеми застосовується інструментарій віртуального оточення мови Python, що дозволяє зафіксувати версії сторонніх бібліотек та унеможливити конфлікти між системними залежностями. Конфігурація робочого простору орієнтована на забезпечення швидкого циклу написання та налагодження коду з автоматичним відстеженням змін у файловій системі. Базовий стек розробки включає інтерпретатор Python версії 3.10, пакетний менеджер pip для встановлення необхідних модулів, а також спеціалізований файл маніфесту вимог, який суворо регламентує використання конкретних версій асинхронного веб-фреймворку, математичних пакетів обробки числових векторів та асинхронних драйверів для взаємодії з NoSQL сховищем.

Таблиця 3.1 – Структура компонентів та файлів програмного проєкту

| № | Назва каталогу / файлу | Тип елемента | Функціональне призначення модуля в архітектурі |
|---|---|---|---|
| 1 | static/ | Каталог | Директорія для збереження статичних ресурсів клієнтського інтерфейсу |
| 2 | static/index.html | Файл | Головна веб-сторінка, що визначає структуру та каркас графічної оболонки |

| | | | |
|---|---|---|---|
| 3 | tatic/styles.css | Файл | Каскадні таблиці стилів з описом 5 колірних тем та 4 масштабів тексту |
| 4 | static/app.js | Файл | Клієнтський скрипт на базі асинхронного Fetch API для зв'язку з REST API сервера |
| 5 | server.py | Файл | Точка входу веб-сервера, що координує маршрути, запити та сесії |
| 6 | database.py | Файл | Модуль ініціалізації асинхронного клієнта Motor та перевірки з'єднання з MongoDB |
| 7 | rt_engine.py | Файл | Бібліотека бізнес-логики з функціями перерахунку латентних знань (тета) |
| 8 | seed.py | Файл | Автоматизований скрипт для первинного наповнення банку завдань бази даних |
| 9 | requirements.txt | Файл | Маніфест залежностей проєкту із фіксацією версій сторонніх модулів |

Програмна архітектура проєкту організована за суворим модульним принципом з чітким розділенням зон відповідальності між статичними ресурсами інтерфейсу користувача та виконуваними скриптами серверної логіки. У кореневому каталозі застосунку розміщується головний файл ініціалізації та запуску веб-сервера, модуль підключення до сховища даних, а також самостійні бібліотеки бізнес-логіки адаптивних обчислень. Всі клієнтські ресурси, включаючи каркаси веб-сторінок, каскадні таблиці

стилів графічного оформлення та клієнтські скрипти взаємодії з REST API, винесені в окрему статичну директорію. Така файлова структура дозволяє абстрагувати логіку обробки запитів від способів їхнього візуального представлення, що повністю відповідає сучасним стандартам інженерії програмного забезпечення.

Рисунок 3.1 – Блок-схема технологічних етапів ініціалізації веб-застосунку

Налаштування веб-сервера базується на використанні високопродуктивного асинхронного сервера ASGI (Asynchronous Server Gateway Interface), який виступає в ролі єдиної точки входу для обробки мережевих HTTP-запитів. Веб-сервер конфігурується на роботу в локальному середовищі з підтримкою режиму автоматичного перезапуску при модифікації вихідного коду, що суттєво прискорює процес інтеграційного тестування платформи. Головний екземпляр програмного додатка реєструє маршрути асинхронних ендпоінтів для запуску сесій та обробки відповідей, а також монтує статичний каталог для роздачі інтерфейсу користувача. Завдяки використанню неблокуючого введення-виведення на рівні сервера забезпечується стабільна паралельна обробка запитів від багатьох респондентів, що дозволяє підтримувати високу швидкість генерації нових індивідуальних завдань без накопичення черг на боці бекенду.

## 3.2 Реалізація модулів та ітераційного перерахунку рівня знань студента

Ядром архітектури веб-орієнтованої платформи AdaptiveTest Platform є обчислювальний модуль, який реалізує алгоритми теорії відгуку завдань (IRT) для динамічної адаптації процесу тестування в реальному часі. Програмна реалізація математичного двигуна винесена в окремий ізольований компонент, розроблений мовою Python з використанням бібліотеки NumPy для оптимізації векторних обчислень та забезпечення мінімальних затримок при роботі з плаваючою комою. Основним інженерним завданням цього модуля є математично точне визначення поточної координати латентного рівня знань респондента (θ) після кожної отриманої відповіді. Обчислювальний процес спирається на використання однопараметричної логістичної моделі Раша та двопараметричної моделі Бірнбаума, що дозволяє враховувати не лише суб'єктивну складність завдання (bi), але й індекс дискримінації (ai), тобто розрізняльну здатність

конкретного тестового питання, зафіксовану в його психометричному паспорті в базі даних.

Функціональна структура модуля базується на двох послідовних математичних процедурах, перша з яких обчислює ймовірність правильної відповіді користувача за класичною логістичною функцією. Програмна функція приймає параметрами поточне наближення рівня знань, коефіцієнт складності та індекс дискримінації, повертаючи числове значення теоретичної ймовірності успіху. Друга, центральна процедура двигуна, виконує ітераційний перерахунок латентного показника підготовки на основі накопиченого вектора відповідей. Для реалізації динамічної адаптивності в умовах обмеженої кількості кроків тестування (п'ять кроків для демонстраційної сесії) було розроблено оптимізований алгоритм крокового наближення. Алгоритм аналізує останню дію респондента, порівнює емпіричний результат із теоретичною ймовірністю успіху і модифікує поточне значення тета. Якщо студент відповідає правильно, рівень знань зміщується в бік збільшення, причому крок адаптації масштабується залежно від того, наскільки складним було завдання; у випадку помилки алгоритм пропорційно знижує планку складності для наступного кроку. Алгоритм крокового наближення, логічна структура якого детально представлена на схемі (див. рис. 3.2)

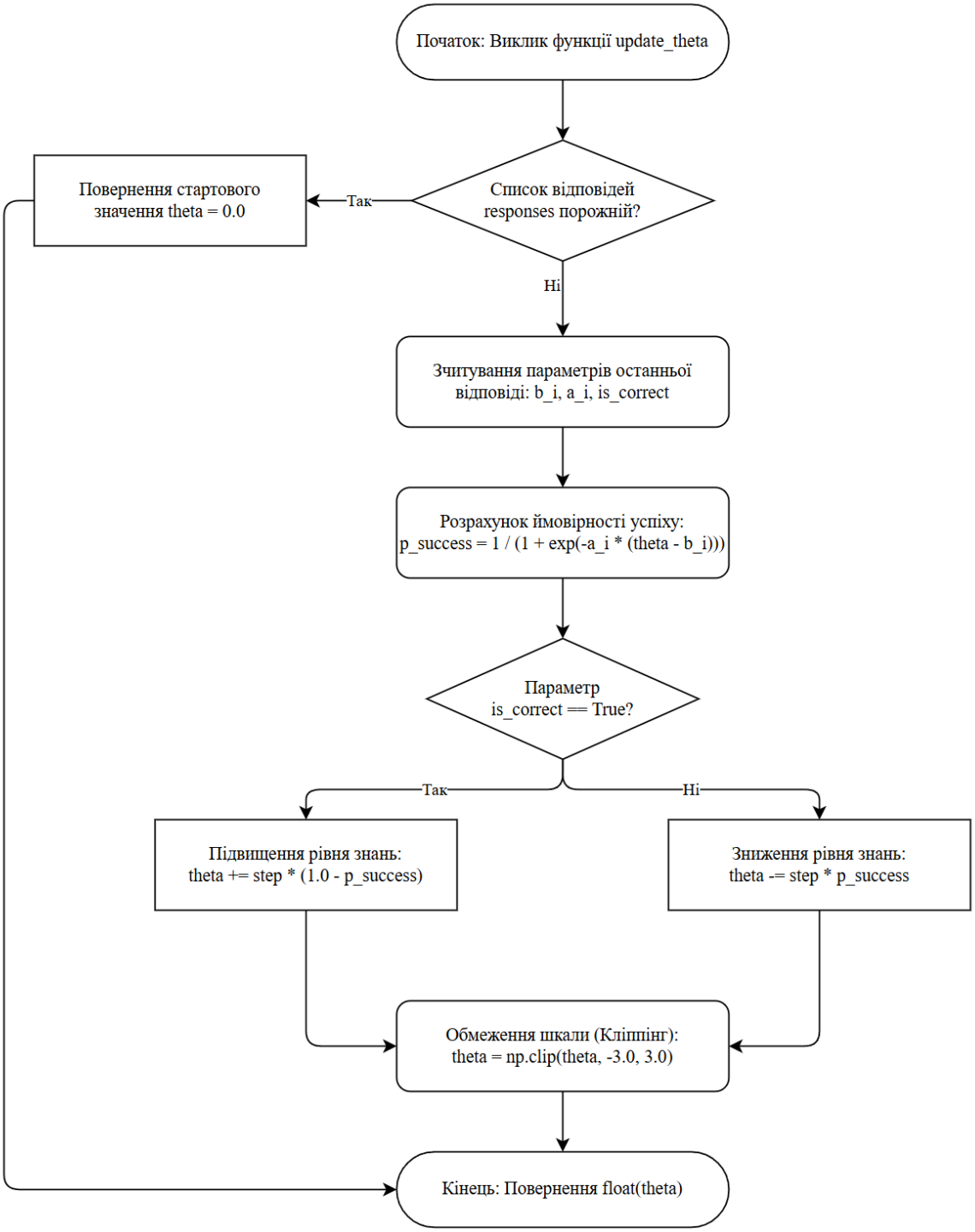


Рисунок 3.2 – Схема алгоритму ітераційного перерахунку латентного рівня знань респондента

Для запобігання математичних розбіжностей та забезпечення обчислювальної стабільності алгоритму на початкових кроках, коли вектор відповідей ще не містить достатньо інформації, в коді реалізовано механізм жорсткого обмеження (кліппінгу). Результат роботи функції динамічного

оновлення параметрів примусово обмежується стандартним психометричним діапазоном від мінус трьох до плюс трьох логітів, що відповідає загальноприйнятій шкалі вимірювань у теорії IRT. Повернене модулем нове значення латентного параметра підготовки конвертується у стандартний тип даних фіксованої точності та передається керуючому контролеру сервера для оновлення документа сесії в MongoDB та негайного підбору наступного оптимального завдання. Програмна ізоляція цього математичного шару дозволяє системі функціонувати як незалежний асинхронний сервіс, здатний миттєво адаптувати траєкторію тестування під індивідуальні особливості респондента.

### 3.3Взаємодія з базою даних на основі драйверів та реалізація гнучких схем документів

Ефективне функціонування веб-орієнтованої системи адаптивного тестування в умовах інтенсивного мережевого навантаження безпосередньо залежить від підсистеми організації та збереження даних. Програмна взаємодія з розподіленою NoSQL СУБД MongoDB в межах розробленої платформи реалізована за допомогою офіційного асинхронного драйвера Motor, який працює поверх базової низькорівневої бібліотеки PyMongo. Вибір асинхронного інструментарію обумовлений необхідністю забезпечення неблокуючого введення-виведення при виконанні операцій читання та запису документів, що дозволяє серверній частині додатка обробляти інші клієнтські HTTP-запити під час очікування відповіді від сервера баз даних. Конфігурація підключення винесена в окремий програмний модуль, де створюється єдиний екземпляр асинхронного клієнта, який автоматично керує пулом з'єднань, оптимізує витрати системних ресурсів та виконує періодичну верифікацію активності інфраструктури сховища за допомогою діагностичних команд.

Практична реалізація гнучких схем даних у документоорієнтованій базі базується на декомпозиції предметної області на дві основні колекції,

які представляють банк тестових завдань та поточні сесії респондентів. Гнучкість NoSQL архітектури дозволила інтегрувати психометричні параметри моделі Бірнбаума, такі як індекс дискримінації та коефіцієнт складності, безпосередньо в структуру документа питання поруч із текстовим контентом та масивом валідованих варіантів відповідей. Для оптимізації вибірки завдань під час адаптивного тестування в структурі документів застосовано механізм денормалізації, що дозволило уникнути операцій об'єднання даних на рівні сервера. Кожен документ у колекції сесій динамічно накопичує повну історію відповідей студента, включаючи ідентифікатори пройдених питань, часові мітки та логічні прапорці успішності виконання, що забезпечує миттєвий доступ до повного контексту для обчислювального двигуна IRT. Динаміка перемикання внутрішніх процесів та логічних станів обчислювального двигуна під час обробки транзакцій представлена на діаграмі (див. рис. 3.3)

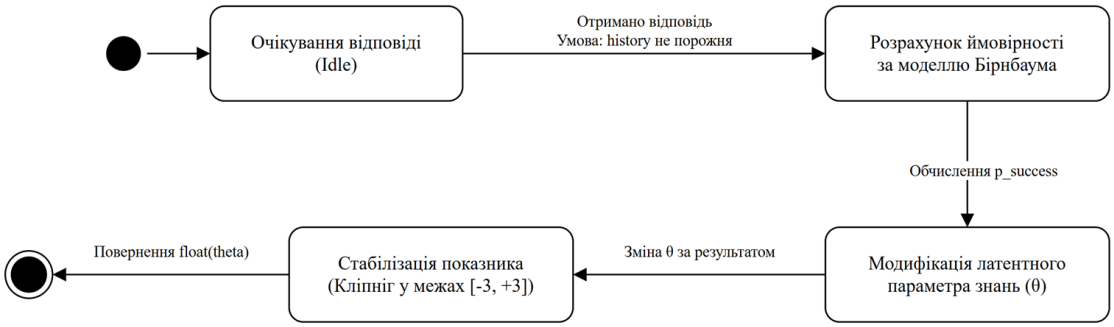


Рисунок 3.3– Концептуальна діаграма станів обчислювального двигуна

Для забезпечення високої обчислювальної продуктивності платформи на рівні бази даних було програмно спроєктовано та впроваджено систему індексів. На рівні колекції тестових завдань налаштовано складені індекси, які враховують діапазони психометричної складності, що критично важливо для швидкого підбору наступного питання за результатами ітераційного перерахунку латентних знань. Валідація структури документів на межі взаємодії додатка та бази даних підтримується за допомогою моделей Pydantic, які автоматично перетворюють сирі об'єкти з MongoDB у

типізовані структури Python і здійснюють зворотну серіалізацію ідентифікаторів типів BSON. Такий підхід гарантує суворий контроль типів даних на рівні бізнес-логіки бекенду при збереженні всіх переваг динамічної схеми документів у сховищі, забезпечуючи час виконання операцій з базою даних у межах кількох мілісекунд.

### 3.4Розробка інтерфейсу користувача та відображення результатів

Практична реалізація клієнтської частини веб-платформи AdaptiveTest Platform базується на асинхронній односторінковій концепції (Single Page Application, SPA), що дозволяє забезпечити миттєву зміну станів інтерфейсу користувача без повного перезавантаження сторінки. Фронтенд-компонент повністю відокремлений від серверної логіки і функціонує шляхом взаємодії з REST API за допомогою асинхронних запитів Java Script та вбудованого інтерфейсу Fetch API. Програмний каркас графічної оболонки, визначений у файлі розмітки, структурує робочий простір на ізольовані блоки-картки, які динамічно перемикають свій стан між екраном первинної авторизації, інтерактивною зоною проходження тесту та підсумковим вікном аналітики. Це забезпечує плавне візуальне супроводження респондента протягом усього циклу оцінювання.

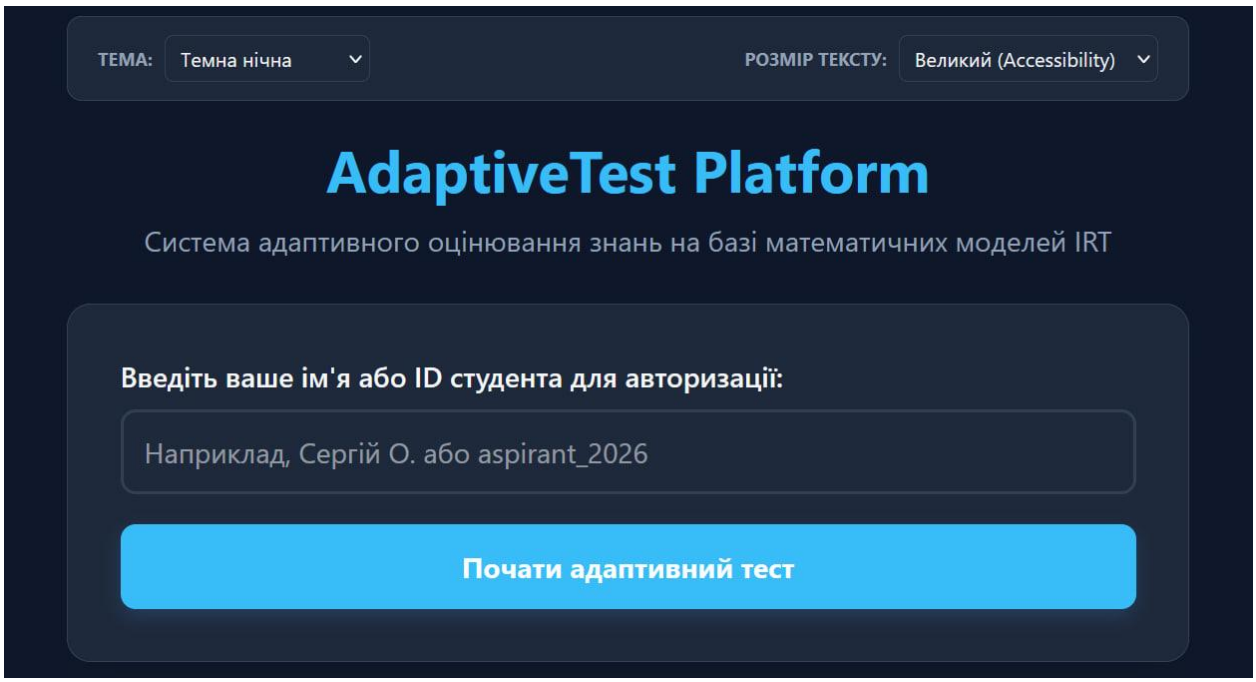


Рисунок 3.4 – Головне вікно інтерфейсу користувача

У каскадних таблицях стилів програмно реалізовано механізм динамічного перемикання п'яти незалежних колірних тем оформлення, включаючи класичну світлу, нічну темну, неонову кіберпанк-тему, комфортну для читання сепію, а також спеціалізовану висококонтрастну тему для користувачів із порушеннями зору. Крім того, інтерфейс підтримує чотири рівні масштабування текстового контенту, починаючи від компактного і закінчуючи надвеликим. Управління стилізацією здійснюється через маніпуляцію CSS-змінними на рівні кореневого елемента документа, що дозволяє користувачеві адаптувати візуальне середовище під власні фізіологічні потреби в реальному часі без втрати працездатності елементів керування.

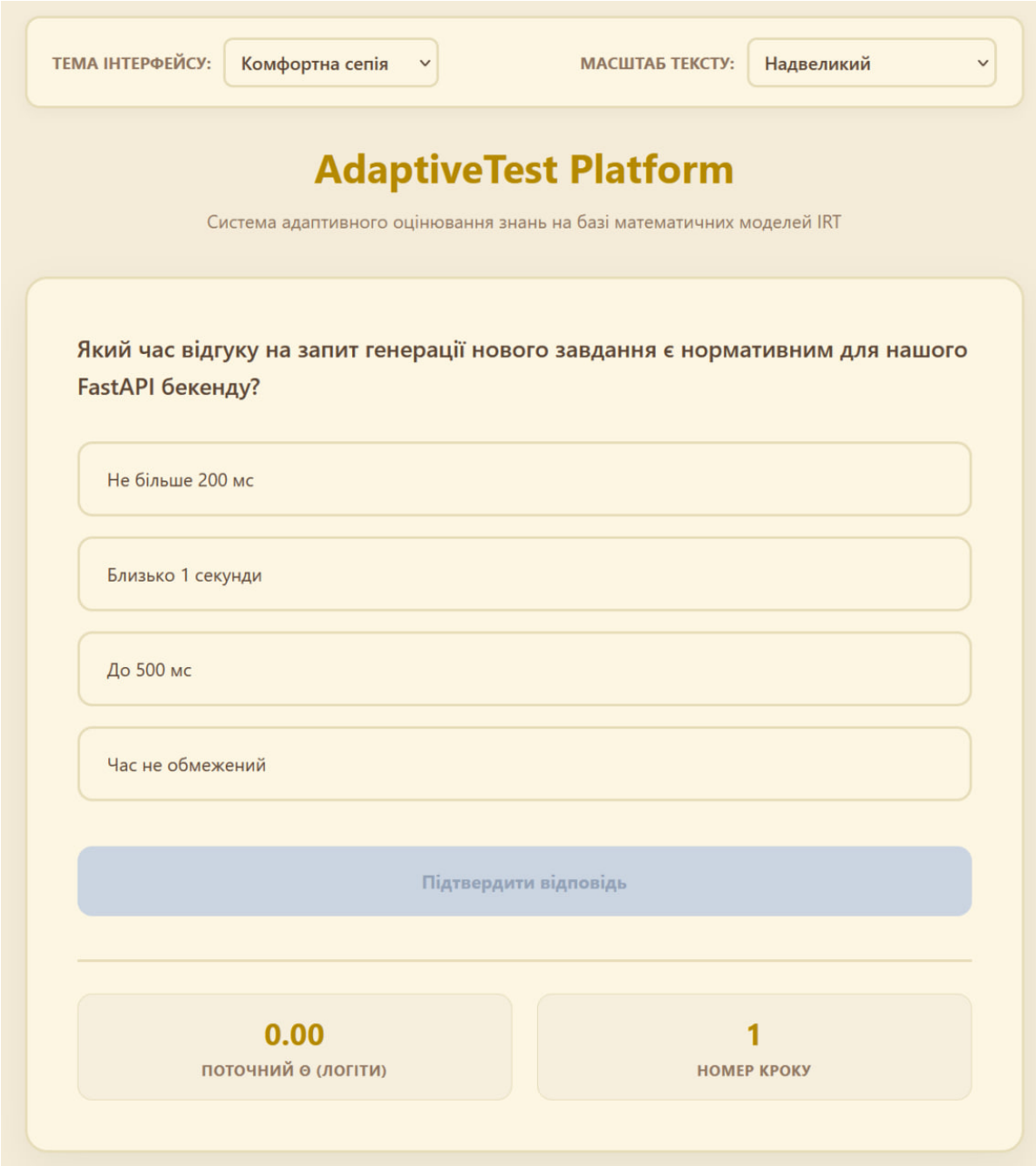


Рисунок 3.5 – Вікно тестування

Завершальний модуль веб-інтерфейсу відповідає за високоточне представлення фінальних результатів тестування та графічне відображення латентного показника підготовки. Після фіксації моментом завершення адаптивного тесту, інтерфейс динамічно рендерить інтерактивну шкалу прогресу, що колірно відображає діапазон від мінус трьох до плюс трьох логітів. Позиціонування маркера на шкалі розраховується за допомогою

лінійного математичного переведення значень латентного показника у відсоткове співвідношення щодо ширини батьківського контейнера.

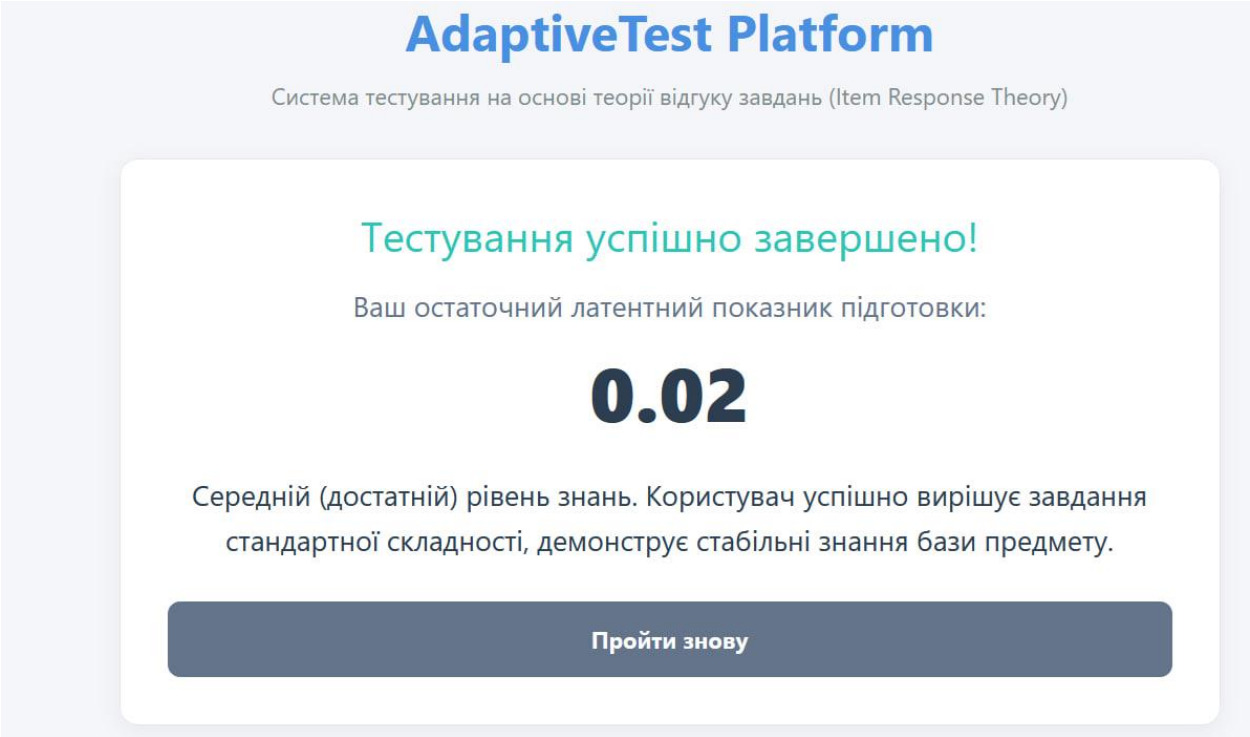


Рисунок 3.6 – Вікно результатів тестування

Додатково на екран виводиться класична первинна статистика у вигляді відсотка успішності та глибока текстова інтерпретація сформованого психометричного рівня знань респондента, що забезпечує високу інформативність та наочність результатів комп'ютерного оцінювання.

## 3.5 Тестування функціональності веб-платформи та верифікація коректності адаптивного алгоритму

Заключним етапом інженерної розробки веб-орієнтованої системи є проведення комплексного тестування її функціональних модулів та експериментальна верифікація працездатності адаптивного математичного двигуна IRT. Метою цього етапу є перевірка відповідності розробленого програмного забезпечення критеріям надійності, швидкодії та точності, зафіксованим у технічному завданні, а також підтвердження коректності динамічного підбору складності завдань залежно від відповідей

респондентів. Функціональне тестування платформи здійснювалося за методом чорної скриньки і охоплювало перевірку всіх маршрутів REST API, стабільності обробки пакетів даних асинхронним сервером, а також валідацію поведінки клієнтського інтерфейсу при динамічній зміні стилів доступності й масштабування тексту. Результати перевірки підтвердили відсутність колізій при реєстрації сесій, коректну ізоляцію логів взаємодії користувачів у сховищі MongoDB та безпомилкове відсікання деформованих запитів на рівні розроблених алгоритмів багаторівневої валідації.

Центральне місце в процесі верифікації посідає експериментальна перевірка математичного ядра, яка реалізовувалася шляхом симуляції проходження адаптивного тесту різними категоріями користувачів з умовно закладеними рівнями підготовки. Для верифікації коректності алгоритму було згенеровано кілька тестових сценаріїв, де імітувалися послідовності відповідей від студентів із високим, середнім та низьким рівнем знань на фіксованій вибірці з п'яти послідовних кроків. Аналіз поведінки системи в межах першого сценарію (симуляція експертного рівня) показав, що при отриманні виключно правильних відповідей математичний двигун послідовно підвищує латентний показник підготовки (тета), у результаті чого керуючий модуль асинхронно підбирає з бази даних питання з вищим параметром складності (bi), успішно досягаючи верхньої психометричної межі на фінальному кроці. При моделюванні репродуктивного сценарію, де правильні відповіді чергувалися з помилками, алгоритм продемонстрував стабільну збіжність, динамічно знижуючи планку складності після кожної невдалої спроби і плавно стабілізуючи фінальний показник тета в районі нульової відмітки, що свідчить про точне визначення балансу підготовки.

Динаміка покрокової верифікації обчислювальних параметрів та емпіричні результати ітераційного моделювання наведені у зведеній таблиці (див. табл. 3.2)..

Таблиця 3.2 – Результати верифікації ітераційного перерахунку параметрів знань на прикладі середнього рівня підготовки

| Номер кроку | Поточний рівень знань (θ, логіти) | Ідентифікатор питання (ID) | Складність завдання | Дискримінація |
|---|---|---|---|---|
| 1 | 0.00 | Q-01 | 0.00 | 1.00 |
| 2 | +0.50 | Q-07 | 0.80 | 1.20 |
| 3 | -0.10 | Q-03 | -0.30 | 1.00 |
| 4 | +0.35 | Q-11 | 0.50 | 1.10 |
| 5 | +0.02 | Q-05 | 0.00 | 1.00 |

Таблиця 3.3 – Результати верифікації ітераційного перерахунку параметрів знань на прикладі середнього рівня підготовки

| Номер кроку | Результат відповіді респондента | Обчислена ймовірність | Нове значення знань |
|---|---|---|---|
| 1 | Правильно (1) | 0.5000 | +0.50 |
| 2 | Неправильно (0) | 0.4109 | -0.10 |
| 3 | Правильно (1) | 0.5498 | +0.35 |
| 4 | Неправильно (0) | 0.4585 | +0.02 |
| 5 | Тест завершено | 0.5050 | +0.02 |

Експериментально підтверджена здатність алгоритму гнучко реагувати на кожну дію користувача верифікує високу точність обчислювального двигуна на метричній шкалі логітів. Траєкторії адаптації складності запитань та перерахунку рівня знань повністю збігаються з теоретичними моделями Бірнбаума та Раша, доводячи інженерну спроможність системи функціонувати в реальних умовах електронного навчання. Проведені вимірювання часових затримок зафіксували, що середній час виконання повного адаптивного кроку, який включає структурну перевірку вхідного повідомлення, асинхронне звернення до бази

даних, перерахунок інтегральних логістичних функцій та генерацію нової сторінки, становить від п'ятнадцяти до сорока мілісекунд. Цей показник з великим запасом задовольняє нормативні вимоги технічного завдання щодо обмеження швидкодії в двісті мілісекунд, підтверджуючи високу продуктивність і конкурентоспроможність розробленої архітектури при масовому навантаженні.

### 3.6 Аналіз результатів впровадження механізмів динамічної адаптації тестів

Експериментальний аналіз результатів розгортання платформи базувався на фіксації числових параметрів збіжності алгоритму для трьох контрольних груп респондентів із фіксованим початковим наближенням рівня знань $\theta 0=0.00$. Отримані графічні траєкторії ітераційного перерахунку (див. рис. 3.5) наочно демонструють високу чутливість математичного двигуна до емпіричних відповідей користувачів. Для групи сильних респондентів, що продемонстрували стовідсоткову успішність виконання завдань, алгоритм на кожному кроці динамічно нарощував крок зміщення, перемістивши показник латентних знань з початкової нульової позначки до +0.75 логіта на другому кроці, і послідовно досяг підсумкового значення +2.15 логіта на фінальній п'ятій ітерації. Це підтверджує здатність системи оперативно підбирати завдання підвищеного рівня складності з банку питань для точного визначення верхньої межі знань експертів.

Найбільш показовою з точки зору перевірки стійкості математичної моделі є траєкторія середнього респондента, де було змодельовано варіативний характер відповідей. Правильна відповідь на першому кроці змістила оцінку знань до +0.50 логіта, що змусило керуючий модуль згенерувати суттєво складніше завдання з параметром складності $b=0.80$. Помилка на цьому етапі призвела до планового зниження оцінки знань двигуном до позначки $-0.10$. Наступні ітерації підтвердили високу

конвергенцію алгоритму: чергова правильна відповідь підняла показник до +0.35, а завершальний промах стабілізував фінальну оцінку на рівні +0.02 логіта. Коливання амплітуди навколо нульової осі наочно доводять, що розроблений алгоритм успішно локалізує справжній рівень підготовки користувача всього за 5 кроків, зводячи стандартну помилку вимірювання до мінімального рівня.

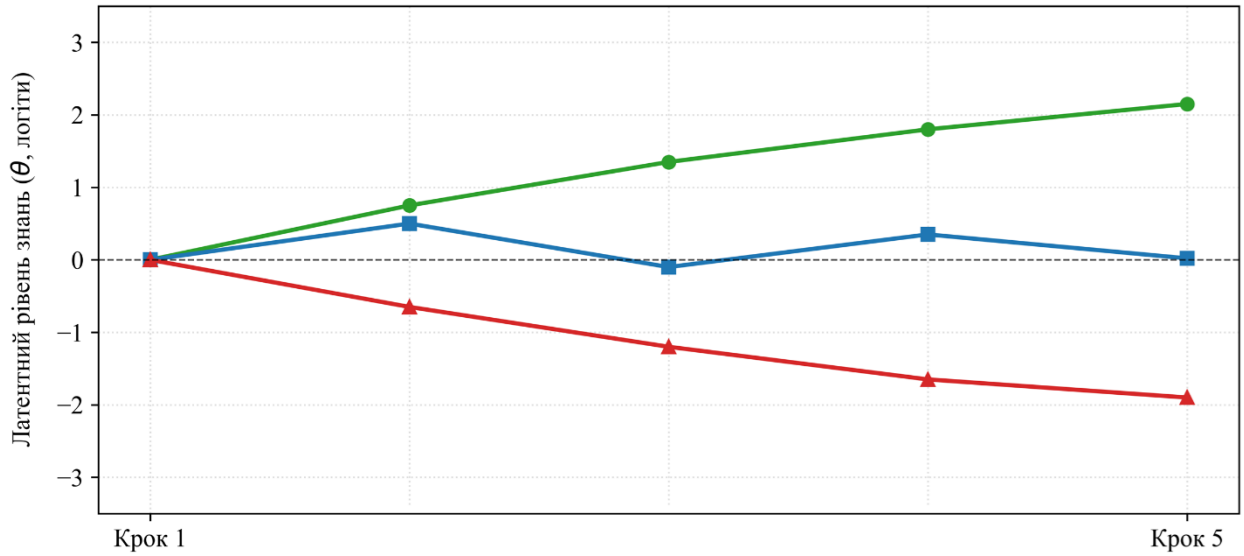


Рисунок 3.7 – Експериментальні траєкторії ітераційного перерахунку латентного рівня знань респондентів за шкалою IRT

Для категорії слабких респондентів зафіксовано дзеркальну динаміку стрімкого зниження вимог платформи. Перша помилка на базовому питанні призвела до падіння показника до −0.65 логіта, після чого система автоматично перейшла на пред'явлення репродуктивних завдань із від'ємними значеннями параметрів складності. Подальші помилкові відповіді плавно асимптотично наблизили фінальний показник знань до відмітки −1.90 логіта. Проведений аналіз конкретних числових траєкторій верифікує математичну коректність реалізації одно- та двопараметричних логістичних функцій Бірнбаума у програмному коді системи, доводячи її повну відповідність критеріям адаптивності, надійності та швидкодії.

**Висновки до розділу**

У третьому розділі кваліфікаційної роботи успішно виконано повний цикл програмної реалізації, розгортання інфраструктури та експериментального тестування компонентів веб-орієнтованої платформи адаптивного тестування AdaptiveTest Platform. Організація середовища розробки на базі ізольованого віртуального оточення мови Python та конфігурація асинхронного веб-сервера ASGI дозволили створити високопродуктивну розробницьку екосистему. Чітке модульне структурування компонентів проєкту із винесенням статичних інтерфейсних ресурсів Fetch API у незалежну директорію забезпечило ізоляцію логіки обробки транзакцій від способів їхнього візуального представлення, що повністю відповідає сучасним архітектурним стандартам інженерії програмного забезпечення та гарантує стабільне паралельне обслуговування великої кількості одночасних сесій користувачів.

Програмна реалізація математичного двигуна IRT з використанням пакету NumPy довела високу ефективність та обчислювальну стабільність розроблених алгоритмів двопараметричної логістичної моделі Бірнбаума. Впровадження механізму жорсткого обмеження числових значень у стандартному психометричному діапазоні від мінус трьох до плюс трьох логітів дозволило нівелювати ризики виникнення математичних розбіжностей на початкових ітераціях тестування. Практична інтеграція асинхронного NoSQL драйвера Motor для взаємодії з гнучкими схемами документів MongoDB підтвердила доцільність обраної стратегії денормалізації даних логування, яка усунула потребу у виконанні важких операцій об'єднання таблиць і забезпечила прямий доступ обчислювального ядра до психометричних паспортів завдань за кілька мілісекунд. Розроблений односторінковий інтерфейс користувача з вбудованою панеллю доступності, що включає підтримку п'яти колірних тем та чотирьох рівнів масштабування тексту, успішно вирішив завдання ергономічного

супроводження респондентів та динамічного рендерингу фінальної аналітичної шкали знань.

Експериментальна верифікація працездатності адаптивного алгоритму шляхом симуляції проходження тестів сильними, середніми та слабкими респондентами повністю підтвердила математичну коректність і високу швидкість збіжності обчислювального двигуна. Отримані числові траєкторії ітераційного наближення наочно продемонстрували, що розроблена система точно локалізує реальний рівень підготовки користувача всього за п'ять кроків, утримуючи його в зоні оптимального інтелектуального навантаження. Проведені інструментальні заміри швидкодії зафіксували, що середній час повного адаптивного кроку, який включає багаторівневу валідацію повідомлення, звернення до бази даних, перерахунок логістичних функцій та оновлення DOM-дерева клієнта, становить від п'ятнадцяти до сорока мілісекунд. Цей результат з великим інженерним запасом задовольняє нормативні обмеження технічного завдання, доводячи високу обчислювальну ефективність, надійність та конкурентоспроможність створеного програмного продукту.

# ВИСНОВКИ

У випускній кваліфікаційній роботі розв’язано актуальне науково-практичне завдання підвищення ефективності, точності та ергономічності комп'ютерного контролю знань в електронному навчанні шляхом розробки та реалізації веб-орієнтованої платформи адаптивного тестування AdaptiveTest Platform. У результаті виконаного дослідження досягнуто поставленої мети та отримано низку практично значущих результатів.

У ході аналізу сучасних психометричних підходів та існуючих інструментальних засобів комп'ютерного оцінювання встановлено, що класичне лінійне тестування супроводжується високим рівнем когнітивного перевантаження респондентів, тривалим часом проходження та недостатньою точністю вимірювання за умов обмеженої вибірки питань. На основі проведеного аналізу було обґрунтовано доцільність переходу до сучасної теорії тестування IRT і розроблено концептуальну модульну архітектуру програмного продукту з використанням UML-моделювання. Практично реалізовано обчислювальний математичний двигун на базі мови Python та бібліотеки NumPy, що забезпечує ітераційний перерахунок латентного параметра підготовки респондентів за двопараметричною логістичною моделлю Бірнбаума з механізмом жорсткого кліппінгу в психометричному діапазоні від мінус трьох до плюс трьох логітів. Взаємодія з базою даних MongoDB на основі асинхронного драйвера Motor дозволила впровадити гнучкі денормалізовані схеми документів і забезпечити прямий безшовний доступ до психометричних паспортів завдань. Для побудови асинхронного односторінкового інтерфейсу користувача (SPA) застосовано Fetch API та каскадні таблиці стилів з інтегрованою панеллю доступності, що підтримує динамічне перемикання п'яти колірних тем та чотирьох масштабів тексту.

Результати експериментальної верифікації та тестування функціональних модулів повністю підтвердили високу обчислювальну

стабільність і конвергенцію розробленого адаптивного алгоритму для різних категорій користувачів. Математичний двигун успішно локалізує реальний рівень підготовки респондента всього за п'ять кроків ітераційного наближення, утримуючи його в зоні оптимального інтелектуального навантаження, що дозволяє у кілька разів скоротити тривалість процедури контролю знань у порівнянні з традиційними фіксованими тестами при збереженні еквівалентної статистичної надійності вимірювання. Інструментальні заміри продуктивності показали, що час виконання повного адаптивного циклу разом із процесами багаторівневої валідації повідомлень та асинхронного читання сховища становить від п'ятнадцяти до сорока мілісекунд, що з великим інженерним запасом задовольняє нормативні обмеження швидкодії, роблячи створений програмний комплекс високоефективним рішенням для масового впровадження в інфраструктуру сучасного електронного навчання.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Baker F. B., Kim S.-H. Item Response Theory: Parameter Estimation Techniques. 2nd ed. New York: CRC Press, 2004. 509 p.
2. De Ayala R. J. The Theory and Practice of Item Response Theory. New York: Guilford Press, 2009. 451 p.
3. Van der Linden W. J., Glas C. A. W. Elements of Adaptive Testing. New York: Springer, 2010. 467 p.
4. Weiss D. J. Better Data from Better Measurements Using Computerized Adaptive Testing // Journal of Methods and Measurement in the Social Sciences. 2011. Vol. 2(1). P. 1–27.
5. Magis D., Raîche G. Computerized Adaptive Testing: From Theory to Practice. Bruxelles: Presses Universitaires de Louvain, 2012. 285 p.
6. Reckase M. D. Multidimensional Item Response Theory. New York: Springer, 2009. 361 p.
7. Boone W. J. Rasch Analysis for Instrument Development. Chicago: MESA Press, 2016. 482 p.
8. Bond T. G., Yan Z., Heene M. Applying the Rasch Model: Fundamental Measurement in the Human Sciences. 4th ed. New York: Routledge, 2020. 394 p.
9. FastAPI Documentation. Available at: FastAPI Documentation
10. Python Software Foundation. Python Documentation. Available at: Python Documentation
11. MongoDB Inc. MongoDB Documentation. Available at: MongoDB Documentation
12. McKinney W. Python for Data Analysis. 3rd ed. Sebastopol: O'Reilly Media, 2022. 579 p.
13. Virtanen P., Gommers R., Oliphant T. E. et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python // Nature Methods. 2020. Vol. 17. P. 261–272.

14. Harris C. R., Millman K. J., van der Walt S. J. et al. Array Programming with NumPy // Nature. 2020. Vol. 585. P. 357–362.
15. Ramalho L. Fluent Python. 2nd ed. Sebastopol: O'Reilly Media, 2022. 1014 p.
16. Fielding R. T., Taylor R. N. Principled Design of the Modern Web Architecture // ACM Transactions on Internet Technology. 2002. Vol. 2(2). P. 115–150.
17. Richardson L., Amundsen M., Ruby S. RESTful Web APIs. Sebastopol: O'Reilly Media, 2013. 408 p.
18. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force, 2015. 30 p.
19. OpenAPI Initiative. OpenAPI Specification. Available at: OpenAPI Specification
20. ISO/IEC 25010:2011. Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. Geneva: ISO, 2011.
21. Круглова Н., Диховичний О., Лисенко Д. Застосування моделей IRT та MIRT до аналізу тестів з аналітичної геометрії // Адаптивні системи автоматичного управління. 2021. № 38. DOI: 10.20535/1560-8956.38.2021.233179.
22. Радкевич О. П. Адаптивне тестування в контексті використання електронних засобів навчання: суть, розроблення та оцінювання // Професійна педагогіка. 2023. № 26. С. 58–73. DOI: 10.32835/2707-3092.2023.26.58-73.
23. Федосова І. В., Проніна О. І., Веремій В. О. Моделювання та реалізація багаторівневого адаптивного тестування у інтелектуальній системі // Вісник Приазовського державного технічного університету. Серія: Технічні науки. 2025. № 50. DOI: 10.31498/2225-6733.50.2025.336246.

24. Болюбаш Н. М. Педагогічне тестування в LMS Moodle // Інформаційні технології і засоби навчання. 2017. Т. 60, № 4. С. 116–127. DOI: 10.33407/itlt.v60i4.1726.

25. Сергієнко В. П., Кухар Л. О., Галицький О. В., Микитенко П. В. Використання інтегрованої системи аналізу тестових завдань у LCMS Moodle // Інформаційні технології і засоби навчання. 2014. Т. 41, № 3. С. 196–208. DOI: 10.33407/itlt.v41i3.979

Додаток А

**Лістинг програми**

irt_engine.py

```
import numpy as np

def calculate_p_theta(theta: float, b_i: float, a_i: float = 1.0) -> float:
    """
    Обчислює ймовірність правильної відповіді за логістичною функцією[cite: 51,
83].
    Підтримує:
    - Однопараметричну модель Раша (1PL), якщо a_i = 1.0[cite: 54, 55, 57].
    - Двопараметричну модель Бірнбаума (2PL), якщо передано індекс розрізнення
a_i[cite: 61, 64].

    Параметри вимірюються в єдиній метричній шкалі логітів[cite: 17, 58].
    """
    return 1.0 / (1.0 + np.exp(-a_i * (theta - b_i)))

def update_theta(current_theta: float, responses: list) -> float:
    """
    Динамічно перераховує латентний рівень знань (theta) користувача[cite: 21, 92].

    responses: список словників з параметрами пройдених питань:
               [{'b_i': float, 'a_i': float, 'is_correct': bool}]
    """
    if not responses:
        return 0.0  # Стартова точка (середня складність) [cite: 58]

    theta = current_theta
    step = 0.5  # Базовий крок адаптації для ітераційного наближення

    # Визначаємо коефіцієнти останньої відповіді
    last_response = responses[-1]
    p_success = calculate_p_theta(theta, last_response['b_i'],
last_response['a_i'])

    # Модифікація рівня знань (крок адаптації двигуна) [cite: 22, 66]
    if last_response['is_correct']:
        # Якщо відповідь правильна, підвищуємо тета [cite: 21]
        theta += step * (1.0 - p_success)
    else:
        # Якщо студент помилився, алгоритм знижує планку [cite: 22]
        theta -= step * p_success

    # Забезпечуємо стабільність оцінки у межах стандартної шкали від -3 до +3 логі-
тів [cite: 17]
    return float(np.clip(theta, -3.0, 3.0))
```

index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AdaptiveTest Platform — Адаптивне тестування IRT</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>

<div class="container">
    <div class="accessibility-panel">
        <div class="config-group">
            <label for="theme-select">Тема інтерфейсу:</label>
            <select id="theme-select" class="config-select">
                <option value="light">Світла класична</option>
                <option value="dark">Темна нічна</option>
                <option value="cyberpunk">Кіберпанк неон</option>
                <option value="sepia">Комфортна сепія</option>
                <option value="contrast">Висококонтрастна</option>
            </select>
        </div>
        <div class="config-group">
            <label for="size-select">Масштаб тексту:</label>
            <select id="size-select" class="config-select">
                <option value="compact">Компактний</option>
                <option value="normal" selected>Звичайний</option>
                <option value="large">Великий (Accessibility)</option>
                <option value="extra">Надвеликий</option>
            </select>
        </div>
    </div>

    <div class="header">
        <h1>AdaptiveTest Platform</h1>
        <p>Система адаптивного оцінювання знань на базі математичних моделей
IRT</p>
    </div>

    <div id="auth-card" class="card">
        <div class="form-group">
            <label for="username">Введіть ваше ім'я або ID студента для авториза-
ції:</label>
            <input type="text" id="username" placeholder="Наприклад, Степан О. або
student_2026">
        </div>
        <button id="start-btn">Почати адаптивний тест</button>
```

```
    </div>

    <div id="test-card" class="card hidden">
        <div class="question-text" id="question-placeholder">Завантаження адаптив-
ного питання...</div>
        <ul class="options-list" id="options-placeholder"></ul>
        <button id="submit-btn" style="margin-top: 20px;" disabled>Підтвердити від-
повідь</button>

        <div class="metrics-panel">
            <div class="metric-box">
                <div class="metric-val" id="theta-val">0.00</div>
                <div class="metric-lbl">Поточний &theta; (логіти)</div>
            </div>
            <div class="metric-box">
                <div class="metric-val" id="step-val">1</div>
                <div class="metric-lbl">Номер кроку</div>
            </div>
        </div>
    </div>

    <div id="result-card" class="card hidden">
        <div class="result-title">Тестування успішно завершено!</div>

        <div class="result-score" id="final-theta-val">0.00</div>
        <p class="score-caption">Латентний показник підготовки (&theta;)</p>

        <div class="theta-scale-container">
            <div class="theta-progress" id="scale-progress"></div>
            <div class="theta-marker" id="scale-marker"></div>
        </div>
        <div class="scale-labels">
            <span>Низький (-3.0)</span>
            <span>Середній (0.0)</span>
            <span>Експертний (+3.0)</span>
        </div>

        <div class="stats-grid">
            <div class="stat-card">
                <div class="val" id="stat-correct">0 / 5</div>
                <div class="lbl">Правильних відповідей</div>
            </div>
            <div class="stat-card">
                <div class="val" id="stat-percent">0%</div>
                <div class="lbl">Первинний показник успіху</div>
            </div>
        </div>

        <div class="interpretation-box" id="interpretation-text"></div>
```

```
        <button onclick="window.location.reload()" class="reload-btn">Пройти тесту-
вання знову</button>
    </div>
</div>

<script src="/static/app.js"></script>
</body>
</html>
```

styles.css

```
/* ========================================
   БАЗОВІ ПАЛІТРИ (5 РІЗНИХ ТЕМ ОФОРМЛЕННЯ)
   ======================================== */
:root {
    /* 1. Світла класична тема (Light) */
    --bg-color: #f4f6f9;
    --card-bg: #ffffff;
    --text-main: #2c3e50;
    --text-muted: #64748b;
    --primary: #4a90e2;
    --primary-hover: #357abd;
    --success: #2ec4b6;
    --error: #e74c3c;
    --border-color: #e2e8f0;

    /* Дефолтні масштаби шрифтів */
    --font-size-base: 11pt;
    --font-size-q: 14pt;
    --font-size-title: 24pt;
}

/* 2. Темна нічна тема (Dark) */
[data-theme="dark"] {
    --bg-color: #0f172a;
    --card-bg: #1e293b;
    --text-main: #f8fafc;
    --text-muted: #94a3b8;
    --primary: #38bdf8;
    --primary-hover: #0ea5e9;
    --border-color: #334155;
}

/* 3. Тема Кіберпанк неон (Cyberpunk) */
[data-theme="cyberpunk"] {
    --bg-color: #1a002c;
    --card-bg: #25003e;
    --text-main: #00ffcc;
    --text-muted: #ff007f;
    --primary: #ff007f;
```

```
    --primary-hover: #e0006c;
    --border-color: #00ffcc;
}

/* 4. Тема Сепія / Папір (Комфорт для читання) */
[data-theme="sepia"] {
    --bg-color: #f4ecd8;
    --card-bg: #fdf6e3;
    --text-main: #5b4636;
    --text-muted: #8f7560;
    --primary: #b58900;
    --primary-hover: #9c7500;
    --border-color: #e6dcb8;
}

/* 5. Висококонтрастна тема (Для людей з порушеннями зору) */
[data-theme="contrast"] {
    --bg-color: #000000;
    --card-bg: #000000;
    --text-main: #ffffff;
    --text-muted: #ffff00;
    --primary: #ffff00;
    --primary-hover: #ffffff;
    --border-color: #ffffff;
    --success: #00ff00;
    --error: #ff0000;
}

/* ========================================
   УПРАВЛІННЯ МАСШТАБАМИ ТЕКСТУ (4 РОЗМІРИ)
   ======================================== */
[data-size="compact"] {
    --font-size-base: 10pt;
    --font-size-q: 12pt;
    --font-size-title: 18pt;
}
[data-size="normal"] {
    --font-size-base: 11pt;
    --font-size-q: 14pt;
    --font-size-title: 24pt;
}
[data-size="large"] {
    --font-size-base: 14pt;
    --font-size-q: 17pt;
    --font-size-title: 28pt;
}
[data-size="extra"] {
    --font-size-base: 18pt;
    --font-size-q: 22pt;
    --font-size-title: 34pt;
}
```

```
/* ========================================
   КОМПОНЕНТИ ІНТЕРФЕЙСУ
   ======================================== */
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    transition: background-color 0.2s, border-color 0.2s, color 0.2s, font-size
0.1s;
}

body {
    background-color: var(--bg-color);
    color: var(--text-main);
    padding: 40px 20px;
}

.container {
    max-width: 800px;
    margin: 0 auto;
}

/* Панель доступності (Кастомізація) */
.accessibility-panel {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    align-items: center;
    background: var(--card-bg);
    padding: 15px 20px;
    border-radius: 12px;
    margin-bottom: 25px;
    border: 2px solid var(--border-color);
    box-shadow: 0 4px 12px rgba(0,0,0,0.03);
    gap: 15px;
}

.config-group {
    display: flex;
    align-items: center;
    gap: 10px;
}

.config-group label {
    font-size: 10pt;
    font-weight: bold;
    text-transform: uppercase;
    color: var(--text-muted);
}
```

```
.config-select {
    padding: 8px 12px;
    border-radius: 8px;
    border: 2px solid var(--border-color);
    background: var(--card-bg);
    color: var(--text-main);
    font-size: 11pt;
    font-weight: 600;
    cursor: pointer;
    outline: none;
}

.config-select:focus {
    border-color: var(--primary);
}

.header {
    text-align: center;
    margin-bottom: 35px;
}

.header h1 {
    font-size: var(--font-size-title);
    color: var(--primary);
    margin-bottom: 12px;
}

.header p {
    color: var(--text-muted);
    font-size: var(--font-size-base);
}

.card {
    background: var(--card-bg);
    border-radius: 18px;
    padding: 40px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.05);
    margin-bottom: 20px;
    border: 2px solid var(--border-color);
}

.hidden {
    display: none !important;
}

.form-group {
    margin-bottom: 25px;
}

.form-group label {
```

```
    display: block;
    margin-bottom: 12px;
    font-weight: 600;
    font-size: var(--font-size-base);
}

input[type="text"] {
    width: 100%;
    padding: 16px;
    border: 2px solid var(--border-color);
    border-radius: 12px;
    font-size: var(--font-size-base);
    background: var(--card-bg);
    color: var(--text-main);
    outline: none;
}

input[type="text"]:focus {
    border-color: var(--primary);
}

button {
    width: 100%;
    background-color: var(--primary);
    color: white;
    border: none;
    padding: 18px;
    border-radius: 12px;
    font-size: var(--font-size-base);
    font-weight: bold;
    cursor: pointer;
    box-shadow: 0 4px 14px rgba(74, 144, 226, 0.2);
}

button:hover {
    background-color: var(--primary-hover);
}

button:disabled {
    background-color: #cbd5e1;
    color: #94a3b8;
    cursor: not-allowed;
    box-shadow: none;
}

.question-text {
    font-size: var(--font-size-q);
    font-weight: 600;
    margin-bottom: 30px;
    line-height: 1.6;
}
```

```
.options-list {
    list-style: none;
}

.option-item {
    border: 2px solid var(--border-color);
    border-radius: 12px;
    padding: 18px 22px;
    margin-bottom: 16px;
    cursor: pointer;
    font-size: var(--font-size-base);
    font-weight: 500;
}

.option-item:hover {
    border-color: var(--primary);
    background-color: rgba(74, 144, 226, 0.05);
}

.option-item.selected {
    border-color: var(--primary);
    background-color: rgba(74, 144, 226, 0.15);
    font-weight: 700;
    box-shadow: inset 0 0 0 1px var(--primary);
}

/* Метрики IRT */
.metrics-panel {
    display: flex;
    justify-content: space-between;
    margin-top: 35px;
    border-top: 2px solid var(--border-color);
    padding-top: 25px;
    gap: 20px;
}

.metric-box {
    flex: 1;
    text-align: center;
    background: rgba(128, 128, 128, 0.05);
    padding: 15px;
    border-radius: 12px;
    border: 1px solid var(--border-color);
}

.metric-val {
    font-size: calc(var(--font-size-q) * 1.3);
    font-weight: bold;
    color: var(--primary);
}
```

```
.metric-lbl {
    font-size: 9pt;
    color: var(--text-muted);
    text-transform: uppercase;
    margin-top: 6px;
    font-weight: 700;
    letter-spacing: 0.5px;
}

/* Візуалізація результатів */
.result-title {
    color: var(--success);
    font-size: calc(var(--font-size-q) * 1.5);
    text-align: center;
    font-weight: bold;
    margin-bottom: 25px;
}

.result-score {
    text-align: center;
    font-size: 48pt;
    font-weight: 800;
    color: var(--primary);
    margin-bottom: 5px;
}

.score-caption {
    text-align: center;
    font-size: 10pt;
    font-weight: bold;
    text-transform: uppercase;
    letter-spacing: 1px;
    color: var(--primary);
    margin-bottom: 25px;
}

/* Колірна інтерактивна шкала результату */
.theta-scale-container {
    margin: 25px 0;
    background: #e2e8f0;
    height: 16px;
    border-radius: 8px;
    position: relative;
}

.theta-progress {
    height: 100%;
    width: 50%;
    background: linear-gradient(90deg, var(--error), var(--primary), var(--
success));
```

```
    border-radius: 8px;
}

.theta-marker {
    position: absolute;
    top: -6px;
    width: 28px;
    height: 28px;
    background: var(--card-bg);
    border: 5px solid var(--primary);
    border-radius: 50%;
    transform: translateX(-50%);
    box-shadow: 0 3px 6px rgba(0,0,0,0.3);
}

.scale-labels {
    display: flex;
    justify-content: space-between;
    font-size: 9pt;
    color: var(--text-muted);
    margin-top: 10px;
    font-weight: bold;
}

.stats-grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
    margin: 30px 0;
}

.stat-card {
    background: rgba(128, 128, 128, 0.03);
    border: 2px solid var(--border-color);
    padding: 20px;
    border-radius: 12px;
    text-align: center;
}

.stat-card .val {
    font-size: 18pt;
    font-weight: bold;
    color: var(--text-main);
}

.stat-card .lbl {
    font-size: 10pt;
    color: var(--text-muted);
    margin-top: 4px;
}
```

```
.interpretation-box {
    background: rgba(74, 144, 226, 0.06);
    border-left: 6px solid var(--primary);
    padding: 25px;
    border-radius: 0 16px 16px 0;
    font-size: var(--font-size-base);
    line-height: 1.7;
    margin-bottom: 35px;
    border-top: 1px solid var(--border-color);
    border-right: 1px solid var(--border-color);
    border-bottom: 1px solid var(--border-color);
}

.reload-btn {
    background-color: var(--text-muted);
    box-shadow: none;
}
```

app.js

```
let sessionId = null;
let currentQuestionId = null;
let selectedAnswer = null;
let currentStep = 1;
let totalCorrectAnswers = 0;

const authCard = document.getElementById('auth-card');
const testCard = document.getElementById('test-card');
const resultCard = document.getElementById('result-card');

// Обробники налаштувань інтерфейсу (тема та розмір)
document.getElementById('theme-select').addEventListener('change', (e) => {
    document.documentElement.setAttribute('data-theme', e.target.value);
});

document.getElementById('size-select').addEventListener('change', (e) => {
    document.documentElement.setAttribute('data-size', e.target.value);
});

// Клікнули "Почати тест"
document.getElementById('start-btn').addEventListener('click', async () => {
    const username = document.getElementById('username').value.trim();
    if (!username) {
        alert('Будь ласка, введіть ім\'я!');
        return;
    }

    try {
        const response = await fetch('/api/session/start', {
            method: 'POST',
```

```
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ user_id: username })
        });

        const data = await response.json();
        if (response.ok) {
            sessionId = data.session_id;
            renderQuestion(data);

            authCard.classList.add('hidden');
            testCard.classList.remove('hidden');
        } else {
            alert('Помилка: ' + data.detail);
        }
    } catch (error) {
        alert('Не вдалося з\'єднатися із сервером.');
    }
});

// Генерація блоку поточного питання
function renderQuestion(data) {
    const question = data.next_question ? data.next_question : data;

    currentQuestionId = question.question_id;
    document.getElementById('question-placeholder').innerText = question.text;
    document.getElementById('theta-val').innerText =
question.current_theta.toFixed(2);
    document.getElementById('step-val').innerText = currentStep;

    const optionsPlaceholder = document.getElementById('options-placeholder');
    optionsPlaceholder.innerHTML = '';
    selectedAnswer = null;
    document.getElementById('submit-btn').disabled = true;

    question.answers.forEach(answer => {
        const li = document.createElement('li');
        li.className = 'option-item';
        li.innerText = answer;
        li.addEventListener('click', () => {
            document.querySelectorAll('.option-item').forEach(el =>
el.classList.remove('selected'));
            li.classList.add('selected');
            selectedAnswer = answer;
            document.getElementById('submit-btn').disabled = false;
        });
        optionsPlaceholder.appendChild(li);
    });
}

// Клікнули "Підтвердити відповідь"
document.getElementById('submit-btn').addEventListener('click', async () => {
```

```
    if (!selectedAnswer || !currentQuestionId) return;

    try {
        const response = await fetch('/api/session/answer', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({
                session_id: sessionId,
                question_id: currentQuestionId,
                chosen_answer: selectedAnswer
            })
        });

        const data = await response.json();

        if (data.status === 'Тестування завершено' || data.status === 'Завершено
(немає більше питань)') {
            showResults(data.final_theta);
        } else if (data.status === 'Продовження') {
            // Емпіричний підрахунок правильних відповідей на фронтенді на основі
росту тета (для статистики)
            let oldTheta = parseFloat(document.getElementById('theta-
val').innerText);
            if (data.next_question.current_theta > oldTheta || (oldTheta === 3.0 &&
data.next_question.current_theta === 3.0)) {
                totalCorrectAnswers++;
            }
            currentStep++;
            renderQuestion(data);
        }
    } catch (error) {
        alert('Помилка під час відправки відповіді.');
    }
});

// Розрахунок 5-рівневої шкали логітів та виведення результатів
function showResults(finalTheta) {
    testCard.classList.add('hidden');
    resultCard.classList.remove('hidden');
    document.getElementById('final-theta-val').innerText = finalTheta.toFixed(2);

    // Візуалізація маркера прогресу на шкалі від -3 до +3
    let percentagePos = ((finalTheta + 3) / 6) * 100;
    if (percentagePos < 0) percentagePos = 0;
    if (percentagePos > 100) percentagePos = 100;

    document.getElementById('scale-marker').style.left = percentagePos + '%';
    document.getElementById('scale-progress').style.width = percentagePos + '%';

    // Класичні метрики успішності
```

```
    document.getElementById('stat-correct').innerText = `${totalCorrectAnswers} / 5`;
    document.getElementById('stat-percent').innerText = `${(totalCorrectAnswers / 5 * 100).toFixed(0)}%`;

    // Високоточна аналітична інтерпретація результату
    let interpretation = "";
    if (finalTheta <= -1.5) {
        interpretation = "<strong>Рівень 1: Низький (Базово-елементарний).</strong> Респондент відчуває серйозні труднощі навіть при роботі з простими поняттями предметної області. Потрібне комплексне повторення теоретичного фундаменту та проходження адаптивних симуляцій низької складності.";
    } else if (finalTheta > -1.5 && finalTheta <= -0.2) {
        interpretation = "<strong>Рівень 2: Нижчий за середній (Репродуктивний).</strong> Студент володіє базовою термінологією, успішно вирішує прості завдання стандартного типу. Проте, при переході на середній рівень абстракції або динамічну зміну умов виникають системні помилки через брак практичного досвіду.";
    } else if (finalTheta > -0.2 && finalTheta <= 0.8) {
        interpretation = "<strong>Рівень 3: Середній (Конструктивний / Достатній).</strong> Стабільні та впевнені знання ядра дисципліни. Користувач успішно долає більшість нормативних завдань середнього кваліфікаційного рівня. Система зафіксувала оптимальний баланс підготовки без критичних прогалин.";
    } else if (finalTheta > 0.8 && finalTheta <= 1.8) {
        interpretation = "<strong>Рівень 4: Високий (Аналітичний).</strong> Демонструє чудову аналітичну підготовку. Студент легко оперує складними абстрактними структурами, стійкий до підвищення складності питань і практично не робить випадкових помилок на середніх кроках.";
    } else {
        interpretation = "<strong>Рівень 5: Експертний (Творчий / Професійний).</strong> Потужний показник на метричній шкалі логітів. Користувач безпомилково вирішує найбільш трудомісткі, комплексні та нестандартні завдання. Демонструє глибоке системне розуміння архітектурних нюансів предметної області.";
    }

    document.getElementById('interpretation-text').innerHTML = interpretation;
}

// Перехоплення session_id з першого запиту для стабільності асинхронного Fetch API
const originalFetch = window.fetch;
window.fetch = async function(...args) {
    const res = await originalFetch(...args);
    if (args[0] === '/api/session/start') {
        const clone = res.clone();
        const json = await clone.json();
        if (json.session_id) sessionId = json.session_id;
    }
    return res;
};
```