Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка

Фізико-математичний факультет
Кафедра інформатики та методики її навчання

Кваліфікаційна робота

РОЗРОБКА МЕТОДИКИ АДАПТИВНОГО ОЦІНЮВАННЯ НАВЧАЛЬНИХ ДОСЯГНЕНЬ УЧНІВ ЗСО

Спеціальність 014 Середня освіта (Інформатика)

Освітня програма
«Середня освіта (Інформатика, математика, STEM- освіта)»

Здобувача другого (магістерського)
рівня вищої освіти
Струка Олександра Сергійовича

НАУКОВИЙ КЕРІВНИК:
кандидат педагогічних наук, доцент
Габрусєв Валерій Юрійович

РЕЦЕНЗЕНТ:
В.О. завідувача центру інформатики, ІКТ та дистанційної освіти
Тернопільського обласного комунального інституту післядипломної педагогічної освіти
Кривокульський Любомир Євстахович

Тернопіль — 2025

АНОТАЦІЯ

Струк О. С. Розробка методики адаптивного оцінювання навчальних досягнень учнів ЗСО. Кваліфікаційна робота на здобуття освітнього ступеня «магістр» зі спеціальності 014 Середня освіта. Тернопіль, 2025.

У роботі здійснено розробку методики та програмної системи автоматизованого оцінювання з генерацією завдань за динамічно визначеними рівнями складності. Систематизовано психометричні моделі класичної тестової теорії та теорії відповіді на завдання, обґрунтовано їх застосування для оцінювання складності та добору завдань. Спроєктовано мікросервісну архітектуру з використанням Kafka для асинхронного обміну подіями між сервісами. Розроблено алгоритми апріорного визначення складності на основі таксономії Блума та апостеріорного оновлення параметрів на основі реальних результатів виконання. Реалізовано функціональний прототип на стеку Node.js, NestJS, React, PostgreSQL з REST API та JWT-автентифікацією. Система забезпечує автоматичну генерацію унікальних варіантів завдань, адаптивний добір відповідно до рівня учня та формування детальних звітів для коригування навчального процесу.

Ключові слова: автоматизоване оцінювання, адаптивне тестування, генерація завдань, рівні складності.

ABSTRACT

Struk O. S. Development of an adaptive methodology for assessing the learning achievements of secondary school students. Master's thesis submitted for the degree of MA in the speciality 014 Secondary Education.

This thesis presents the development of a methodology and software system for automated assessment with task generation based on dynamically determined difficulty levels. Psychometric models of Classical Test Theory and Item Response Theory were systematized, and their application for difficulty assessment and task selection was substantiated. A microservice architecture was designed using Kafka for asynchronous event exchange between services. Algorithms were developed for a priori difficulty determination based on Bloom's taxonomy and a posteriori parameter

updating based on actual performance results. A functional prototype was implemented using Node.js, NestJS, React, PostgreSQL stack with REST API and JWT authentication. The system provides automatic generation of unique task variants, adaptive selection according to student level, and formation of detailed reports for adjusting the learning process.

Keywords: automated assessment, adaptive testing, task generation, difficulty levels.

ЗМІСТ

ВСТУП

Українська освіта реформується, змінюючи підходи до оцінювання. Традиційні методи є суб'єктивними й трудомісткими, а стандартизовані тести не враховують індивідуальних відмінностей учнів.

Психометричні підходи забезпечують індивідуальний добір складності, але потребують великої бази завдань. Автоматична генерація варіантів вирішує цю проблему, забезпечуючи персоналізацію, запобігання списуванню та необмежений банк завдань. Цілісного рішення для української школи наразі немає.

Актуальність дослідження зумовлена потребою модернізації системи оцінювання навчальних досягнень відповідно до вимог Стратегії цифровізації освіти і науки України до 2027 року та Концепції реалізації державної політики у сфері реформування загальної середньої освіти «Нова українська школа». Концепція НУШ наголошує на необхідності переходу від репродуктивного до компетентнісного оцінювання, яке має бути об'єктивним, формувальним та орієнтованим на індивідуальний освітній поступ учня. Стратегія цифровізації визначає пріоритетом впровадження цифрових технологій для персоналізації навчання та автоматизації педагогічних процесів. Водночас у реальній шкільній практиці зберігаються системні проблеми: суб'єктивність оцінювання, обмеженість банків тестових завдань, неможливість адаптації складності до рівня конкретного учня, недостатність оперативного зворотного зв'язку. Розроблена методика адаптивного оцінювання з автоматичною генерацією завдань на основі динамічно оновлюваних рівнів складності забезпечує практичну реалізацію нормативних вимог щодо індивідуалізації навчання, об'єктивності контролю та інтеграції цифрових інструментів у освітній процес.

Мета дослідження є розробка методики адаптивного оцінювання навчальних досягнень учнів закладів загальної середньої освіти з автоматичною генерацією завдань на основі динамічного визначення рівнів складності та її програмна реалізація.

Завдання дослідження:

1. Проаналізувати систему оцінювання та обґрунтувати доцільність адаптивних підходів.
2. Опрацювати теоретичний фундамент: математичні моделі складності, механізми автоматичної генерації.
3. Побудувати концептуальну модель методики.
4. Розробити архітектуру, алгоритми системи та провести експериментальну апробацію прототипу.

Об'єктом дослідження є процес оцінювання навчальних досягнень учнів у закладах середньої освіти.

Предмет дослідження є методика адаптивного оцінювання з автоматичним створенням завдань відповідної складності.

Методи дослідження використані в роботі, включають аналіз літератури та моделювання, комп'ютерне моделювання та порівняльний аналіз, застосування психометричних моделей і проектування архітектури.

Наукова новизна роботи полягає в тому, що вперше запропоновано методику, яка поєднує математичні моделі, автоматичну генерацію та аналіз поведінки учнів під час тестування, а також удосконалено підхід до визначення складності через поєднання теоретичних припущень з реальною статистикою.

Практичне значення роботи полягає в автоматизації оцінювання, економії часу вчителів, запобіганні списуванню, а також у наданні детальної аналітики для коригування навчального процесу.

Структура роботи. Магістерська робота складається зі вступу, двох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗОВАНОГО ОЦІНЮВАННЯ ЗНАНЬ

1.1. Система оцінювання знань учнів: стан, проблеми та вимоги

Процес оцінювання в сучасній освіті виходить далеко за межі простої констатації результатів. Через нього реалізуються функції контролю, зворотного зв'язку та мотивації [8, с. 16]. Компетентнісний підхід змістив акценти: оцінка має стати засобом підтримки освітнього поступу, а не лише інструментом селекції [11]. Водночас реальна шкільна практика зберігає суперечливу картину — попри гуманістичні декларації, значна частина процедур досі фокусується на результаті, а не на процесі навчання.

У разі аналізу повсякденної практики української школи можна виокремити низку системних проблем у сфері оцінювання. Попри формальну орієнтацію освітнього процесу на компетентнісний підхід, за якого оцінювання має відображати не лише обсяг знань, а й здатність учнів застосовувати їх у практичних ситуаціях [12], у навчанні переважають традиційні форми контролю з домінуванням репродуктивних завдань. Унаслідок цього результати оцінювання нерідко фіксують насамперед рівень підготовленості до контрольної ситуації, а не реальний ступінь сформованості компетентностей.

Крім того, усне опитування обмежує можливості об’єктивного оцінювання через малу кількість учнів, яких можна залучити протягом одного уроку, тоді як письмові роботи потребують значних часових ресурсів на перевірку. Відтермінування між виконанням завдання та отриманням результатів знижує ефективність формувального оцінювання[9, с. 14], оскільки учні втрачають навчальний контекст і не завжди можуть співвіднести підсумковий бал із конкретними прогалинами у знаннях і вміннях.

Суб'єктивність залишається критичною проблемою. Оцінюючи роботу, педагог спирається не лише на формальні критерії, а й на власні уявлення про сильних та слабких учнів, попередній досвід взаємодії. Психологічні дослідження описують ефекти ореолу та контрасту, вплив ситуативних

чинників [10, с. 89]. Традиційна бальна оцінка має низьку діагностичну цінність — учень часто не розуміє, які саме знання недостатні і що потрібно змінити.

Широке впровадження тестових технологій мало зменшити суб'єктивність та стандартизувати контроль [16, с. 45]. Однак виникли нові виклики. Більшість тестів є статичними: єдиний набір завдань для всіх незалежно від рівня підготовки. Для частини учнів завдання занадто прості і не створюють інтелектуального виклику, для іншої — надмірно складні, що викликає безпорадність. З погляду психометрії така неузгодженість знижує точність вимірювання [23, с. 67; 37, с. 112]. Статичний тест найкраще працює для середнього рівня, тоді як результати слабших і сильніших учнів мають нижчу інформативність.

Окремою проблемою сучасної практики оцінювання є явище «натаскування». У межах такого підходу підготовка учнів часто редукується до багаторазового виконання однотипних тестових завдань і відпрацювання алгоритмів їх розв’язання. Це сприяє формуванню переважно процедурних стратегій замість глибокого концептуального розуміння навчального матеріалу. Як наслідок, високі результати тестування можуть не корелювати з реальним рівнем сформованості компетентностей, зокрема здатністю застосовувати знання в нових або проблемних ситуаціях.

Додатковим чинником є обмеженість і повторюваність банків завдань, що підвищує ризики порушення академічної доброчесності. Крім того, традиційні тести з вибором однієї правильної відповіді частіше зорієнтовані на перевірку нижчих рівнів когнітивної діяльності, тоді як оцінювання аналізу, синтезу та критичного оцінювання потребує складніших інструментів і форматів. У підсумку зміст контролю має тенденцію звужуватися до того, що простіше стандартизовано виміряти, а не до того, що є ключовим для розвитку компетентностей.

Цифровізація освіти змінює роль автоматизованих систем оцінювання [2, с. 22; 13]. Вони переходять від допоміжних засобів до ключових компонентів навчального середовища. Системи з автоматичною генерацією завдань

поєднують масовість навчання з індивідуалізацією контролю [1, с. 35; 29, с. 78], створюють передумови для неперервного моніторингу без додаткового навантаження на вчителя. Особливо значущим є потенціал в організації самостійної роботи [7, с. 156]. Традиційна модель домашніх завдань передбачає значний розрив між виконанням і зворотним зв'язком. Автоматизовані системи забезпечують миттєвий зворотний зв'язок: учень відразу бачить результат, правильні відповіді, коментарі до помилок. Можливість багаторазового виконання подібних за структурою, але параметрично змінюваних завдань сприяє формуванню стійких умінь [17, с. 80; 21, с. 134].

Автоматична генерація змінює підхід до контролю [22; 30, с. 45]. Генеративні тести формують унікальні варіанти для кожного учня при збереженні еквівалентності за складністю. Це знижує можливість списування і забезпечує справедливість. Зникає проблема старіння банку завдань, оскільки кожна спроба є новою комбінацією параметрів [31, с. 189]. Система оцінювання стає інструментом підтримки рішень: автоматизована обробка дає не лише бали, а й детальну аналітику — розподіл учнів за рівнями, типові помилки, проблемні теми [36, с. 234]. Педагог може оперативно коригувати планування, змінювати темп, формувати групи для диференційованої підтримки.

Надійне функціонування системи потребує дотримання комплексу вимог. Валідність означає, що згенеровані завдання адекватно відображають цілі навчання та відповідають програмам [8, с. 18; 16, с. 67]. Кожне завдання має перевіряти саме ті знання та вміння, які декларуються. У випадку автоматичної генерації важливо забезпечити семантичну коректність: завдання мають бути логічними, недвозначними, без фактичних помилок [29, с. 145]. Надійність пов'язана зі стабільністю результатів — різні екземпляри з одного шаблону повинні мати близький рівень складності [25, с. 298]. Значні коливання знижують справедливість, тому важливі механізми калібрування та використання статистики для коригування моделей генерації [10, с. 123; 32, с. 167].

Масштабованість архітектури набуває критичного значення в умовах масового використання [35, с. 423; 39, с. 245]. Велика кількість учнів може одночасно розпочинати виконання, що створює пікові навантаження. Оскільки автоматична генерація потребує більше обчислень, система має бути здатною до горизонтального масштабування [38, с. 178]. Інтероперабельність є необхідною для інтеграції в освітній простір. Система має підтримувати стандартизовані протоколи обміну даними, зокрема LTI (взаємодія навчальних інструментів) для інтеграції із системами управління навчанням, формати SCORM (еталонна модель об'єктів спільного доступу до контенту) або QTI (взаємодія питань і тестів) для імпорту та експорту матеріалів [18, с. 38].

Адаптивність інтерфейсу визначає ефективність з погляду користувацького досвіду. Оцінювання відбувається в різних контекстах: у класах, вдома, з мобільних пристроїв. Інтерфейс має коректно відображатися на екранах різного розміру, бути інтуїтивно зрозумілим, мінімізувати зайві дії. Важливим є дотримання принципів універсального дизайну та вимог доступності. Окремої уваги потребує підтримка коректного відображення математичних формул, графіків, діаграм [4, с. 145; 6, с. 98].

У сучасній педагогічній теорії оцінювання розглядається як цілісний процес збирання, інтерпретації та використання інформації про навчальні досягнення [8, с. 15]. До структури процесу входять визначення критеріїв, добір інструментів, проведення процедур, аналіз результатів і прийняття рішень. Важливо розрізняти оцінювання як процес та оцінку як його числовий результат.

Функції педагогічного оцінювання є багатовимірними [8, с. 17; 9, с. 13]. Контролююча функція полягає у фіксації досягнутого рівня. Діагностична — у виявленні характеру та причин помилок, визначенні слабких місць. Вона створює підґрунтя для корекції навчального процесу. Мотиваційна функція є критичною: одержуючи результати, учень формує уявлення про власні можливості, переживає емоції успіху чи невдачі. Оцінювання може бути

джерелом внутрішньої мотивації, якщо воно побудоване як справедливий процес, що демонструє зв'язок між зусиллями та результатами.

Реалізація функцій можлива за умови дотримання принципів. Систематичність передбачає регулярність процедур, їх зв'язок з логікою подання матеріалу. Автоматизовані системи створюють можливості для неперервного моніторингу без збільшення навантаження. Об'єктивність пов'язана з незалежністю результатів від особистісних особливостей чи упереджень. Для цього потрібні чіткі критерії, уніфіковані процедури, стандартизовані завдання. Автоматизовані системи зменшують суб'єктивний фактор, але висувають вимоги до якості конструкції завдань. Прозорість означає зрозумілість критеріїв та результатів. Учень повинен розуміти, за що він отримує бал, які помилки були критичними. У автоматизованих системах це досягається через чітку специфікацію тестів, опис рівнів складності, зрозумілу візуалізацію результатів.

Ключовим є розмежування між формальним та підсумковим оцінюванням. Підсумкове орієнтоване на констатацію результатів після завершення етапу. Його мета — фіксація досягнення стандартів, рішення про переведення. Формувальне принципово відрізняється тим, що інтегроване у щоденний процес і спрямоване на підтримку навчання [9, с. 15]. Воно передбачає систематичне збирання інформації про просування учня, своєчасне виявлення труднощів, гнучке коригування методів і темпу. Учень отримує не лише бал, а й пояснення, рекомендації. Помилка розглядається як природна частина процесу.

Автоматизовані системи особливо доречні для формувального оцінювання. Їхні можливості миттєвого надання результатів, пропозиції додаткових завдань, відстеження динаміки дозволяють реалізувати модель оцінювання для навчання. Вчитель може налаштовувати режим так, щоб учні виконували серію тренувальних завдань із поступовим ускладненням, а система пропонувала підказки або посилання на матеріал.

Другим важливим розрізненням є відмінність між нормо-орієнтованим і критеріально-орієнтованим підходами [20; 23, с. 34]. Нормо-орієнтоване спрямоване на порівняння досягнень учня з групою. Результати інтерпретуються відносно норми [37, с. 89]. Це дає інформацію про позицію, але менш придатне для діагностики прогалин. Критеріально-орієнтоване орієнтується на заздалегідь визначені критерії — описані рівні володіння компетентністю [12]. Результат інтерпретується через співвіднесення з еталонним описом: чи досягнуто рівня, які компоненти сформовано. Такий підхід лежить в основі компетентнісної парадигми і є найбільш релевантним для автоматизованої системи.

Моделювання системи неможливе без урахування рівневої структури навчальних досягнень. Таксономія освітніх цілей виокремлює послідовні рівні: від запам'ятовування до аналізу, синтезу та оцінювання. Для системи з автоматичною генерацією це означає необхідність банку шаблонів, який відображає різні рівні складності не лише за формальними ознаками, а й за характером мисленнєвих операцій [29, с. 167; 31, с. 223].

На нижчих рівнях завдання спрямовані на перевірку того, чи здатен учень впізнавати або відтворювати основні факти, формули. Це завдання на розпізнавання відповіді, заповнення пропусків. Для автоматизованої системи вони відносно прості — достатньо структурованого банку базових фактів [1, с. 36]. Середні рівні, пов'язані з розумінням та застосуванням, вимагають іншого типу. Учень повинен не лише пригадати правило, а й пояснити зміст, інтерпретувати інформацію, використати модель для розв'язання задачі [29, с. 189]. Доцільно використовувати параметризовані задачі: змінюючи числові значення, контексти, система формує різні варіанти однакового когнітивного рівня.

На вищих рівнях завдання спрямовані на виявлення здатності розкладати ситуації на елементи, встановлювати зв'язки, порівнювати способи розв'язання, обґрунтовувати вибір. Автоматична генерація тут значно складніша, оскільки

вимагає не лише варіювання параметрів, а й конструювання цілісних ситуацій із збереженням логічної коректності [31, с. 267].

1.2. Психометричні моделі та рівні складності тестових завдань

Науково обґрунтована система оцінювання неможлива без психометричних моделей, які дозволяють перейти від інтуїтивних уявлень про легкі та важкі завдання до формалізованих параметрів [16, с. 23]. Психометрія забезпечує апарат для опису зв'язку між латентними навчальними досягненнями учня та результатами тесту, а також дає змогу кількісно оцінювати складність, дискримінантну здатність та інші характеристики тестових завдань [23, с. 12]. У сучасній освітній кваліметрії домінують дві парадигми: класична теорія тестування та теорія відповіді на завдання [20].

У межах педагогічної кваліметрії педагогічний тест визначають як стандартизовану систему завдань спеціально сконструйованої форми та змісту, призначених для кількісної та якісної оцінки рівня знань, умінь та навичок [8, с. 17]. На відміну від традиційної контрольної роботи, тест розглядається як вимірювальний інструмент, для якого можна оцінити надійність, валідність, чутливість і диференційну здатність. Стандартизованість означає, що для всіх учасників застосовуються однакові умови, інструкції, правила оцінювання та інтерпретації результатів [16, с. 34].

Структурно тест складається з набору тестових завдань, кожне з яких є елементарним вимірювальним актом. Сукупність таких пунктів формує інструмент із певною специфікацією: перелік тем, пропорцію завдань різних рівнів складності, типи перевірюваних когнітивних операцій. Для комп'ютеризованих систем надзвичайно важливим є вибір типів завдань, оскільки саме вони визначають можливість алгоритмізації перевірки та глибину діагностичної інформації [1, с. 35].

Найпоширенішими є завдання закритої форми з вибором однієї правильної відповіді. Учню пропонується ствердження або запитання і декілька варіантів, серед яких тільки один правильний. Такий формат особливо зручний

для автоматизованих систем, оскільки перевірка зводиться до порівняння обраного варіанта з еталонним кодом [17, с. 81]. До окремої групи належать завдання на встановлення відповідності та впорядкування. У завданнях на відповідність учень повинен утворити пари між елементами двох множин. У завданнях на впорядкування потрібно розмістити елементи в певній послідовності — за хронологією, логікою виконання алгоритму або зростанням значення. Такі завдання добре виявляють здатність бачити системні зв'язки та структурувати матеріал [19].

Завдання відкритої форми з короткою відповіддю передбачають, що учень самостійно вводить відповідь у вигляді числа, символічного виразу чи короткого слова. Автоматична перевірка таких завдань вимагає обробки введеного рядка, нормалізації форматів, врахування можливих еквівалентних записів. Перевагою є зменшення ймовірності випадкового вгадування. Окремо розглядаються задачі з розгорнутою відповіддю, де учень має описати хід міркувань, довести твердження. Такі завдання найкраще відбивають вищі рівні когнітивної діяльності, але їх автоматична перевірка є суттєво складнішою [29, с. 156].

Суттєве місце посідають параметризовані завдання. Вони будуються на основі шаблону, де окремі елементи замінено параметрами [31, с. 134]. Під час генерування конкретного екземпляра система підставляє випадкові або спеціально підібрані значення параметрів, дотримуючись заданих обмежень. Таким чином, з одного шаблону можна отримати велику кількість ізоформних варіантів, що мають однакову структурну та когнітивну складність, але різняться конкретними даними. Параметризовані завдання є природною основою для автоматичної генерації тестового контенту [29, с. 89].

Складність тестового завдання є ключовою психометричною характеристикою, яка визначає, наскільки завдання є легким і важким для респондентів певного рівня підготовки [16, с. 56]. У найзагальнішому розумінні складність можна трактувати як ймовірність правильного виконання: чим

нижча ця ймовірність, тим завдання складніше [23, с. 45]. У класичній тестовій теорії ця імовірність оцінюється емпірично через індекс складності

$$p = \frac{N_{\text{прав}}}{N_{\text{заг}}} \tag{1.1}$$

Чим ближче p до 1, тим завдання легше. У практиці стандартизованого тестування завдання з $p > 0{,}8$ відносять до дуже легких, а з $p < 0{,}2$ — до дуже складних, тоді як найбільш інформативними вважаються завдання із середніми значеннями у проміжку приблизно 0,4–0,8 [16, с. 67].

Сучасні психометричні підходи, зокрема теорія реагування на завдання, розглядають складність як параметр b у моделі залежності ймовірності правильної відповіді від рівня здібностей учня θ [37, с. 98]. У найпростішій одновимірній моделі Раша ймовірність правильної відповіді описується логістичною функцією [41, с. 56]:

$$P(\text{правильна відповідь} \mid \theta) = \frac{1}{1 + e^{-(\theta - b)}} \tag{1.2}$$

У цій схемі b інтерпретується як той рівень здібностей, за якого шанси відповісти правильно становлять 50%. Чим більше значення b, тим завдання зсувається у зону вищих здібностей і тим воно складніше. На відміну від простого індексу p, параметр b менш залежний від конкретної вибірки учнів і дає можливість будувати адаптивні тести [25, с. 187].

З погляду практики конструювання тестів важливо розрізняти апріорну та апостеріорну складність. Апріорна складність визначається на етапі розробки завдання на основі аналізу змісту, формату, очікуваних когнітивних операцій [29, с. 112]. Вона залежить від низки факторів. Змістові фактори пов'язані з тим, який саме навчальний матеріал перевіряється: кількістю понять, рівнем абстрактності, міжпредметними зв'язками. Завдання, яке вимагає лише пригадати означення одного базового поняття, апріорі легше, ніж завдання, де потрібно одночасно оперувати кількома теоремами.

Логічні фактори відбивають структурну складність міркувань, необхідних для розв'язання. Однокрокові завдання є легшими за багатокрокові задачі, де результат досягається послідовністю перетворень, і помилка на

одному з етапів призводить до хибного підсумку. Мовні фактори впливають на сприйняття завдання. Навіть відносно простий за змістом приклад може сприйматися як складний, якщо умова сформульована довгими реченнями з багатьма підрядними конструкціями, подвійними запереченнями. Навпаки, чіткі, лаконічні формулювання зменшують додаткове когнітивне навантаження [16, с. 78].

Формальні фактори пов'язані з типом завдання та організацією варіантів відповіді. У тестах з вибором відповіді суттєве значення має кількість і якість дистракторів. Якщо неправильні варіанти очевидно абсурдні, завдання виконується легко навіть при поверхневих знаннях; якщо ж дистрактори побудовано на типових помилках, завдання стає істотно складнішим [23, с. 89]. Часовий фактор пов'язаний з тим, який час відведено на виконання. У жорстко лімітованих за часом тестах навіть нескладні задачі можуть набувати високої суб'єктивної складності.

Фактична складність визначається за результатами реального виконання [10, с. 134]. Вона може характеризуватися індексом складності p, параметром b у моделях IRT (теорія відповіді на завдання), середнім часом виконання, частотою повернень до завдання. У комп'ютеризованих навчальних середовищах накопичені дані дозволяють побудувати інтегральний показник складності як функцію кількох змінних:

$$D = f(p, t_{\text{сер}}, k_{\text{спроб}}, k_{\text{змін}}) \qquad (1.3)$$

де p — частка правильних відповідей, $t_{сер}$ — середній час виконання, $k_{спроб}$ — середня кількість спроб, $k_{змін}$ — частота змін відповіді [3, с. 69].

Практичне використання результатів психометричного аналізу потребує не тільки числових оцінок, а й їх інтерпретації у вигляді якісних рівнів [8, с. 19]. У педагогічній практиці традиційно використовують трирівневі або чотирирівневі шкали, де виділяють низький, середній, високий рівні складності. Таке рівневе представлення є зрозумілим для вчителів та учнів і дозволяє планувати структуру тесту.

Якщо виходити з індексу складності p, можна умовно вважати, що завдання з p близько до 1 належать до низького рівня складності: з ними справляється переважна більшість учнів. Такі завдання доцільно використовувати для перевірки базових знань, мотивації слабших учнів. Завдання середнього рівня відповідають проміжним значенням p і є найбільш інформативними для розрізнення різних рівнів підготовки [16, с. 91]. Високий рівень складності асоціюється з низькими значеннями p, коли завдання стає доступним лише для добре підготовлених учнів.

У термінах теорії IRT рівні складності можуть задаватися інтервалами значень параметра b [20]. Наприклад, завдання з $b \leq -1$ можна вважати легкими, з b у проміжку $[-1; 1]$ — середніми, а з $b \geq 1$ — складними. У комп'ютеризованому адаптивному тестуванні рівні складності безпосередньо пов'язуються з поточною оцінкою здібностей θ учня: для максимізації інформативності обираються завдання з параметрами b, найближчими до поточного значення θ [37, с. 157].

Збалансований розподіл завдань за рівнями складності є однією з основних умов якості тесту [16, с. 102]. Якщо тест складено переважно з легких завдань, він не дозволяє розрізнити сильних і слабких учнів: більшість набирає високі бали, виникає стеля вимірювання. Якщо ж тест містить надмірну кількість складних завдань, слабкі учні опинилися в ситуації постійної невдачі. У навчальному процесі рівні складності завдань повинні корелювати з рівневими характеристиками навчальних досягнень [19]. Для початкового рівня достатньо завдань на впізнавання та відтворення базових фактів. Середній рівень передбачає включення завдань на розуміння та застосування матеріалу в типових ситуаціях. Високий рівень вимагає задач, що стимулюють аналіз, синтез, побудову нових способів розв'язання.

У контексті систем з автоматичною генерацією рівні складності набувають додаткового значення [31, с. 178]. Шаблони завдань повинні маркуватися за рівнем складності, що дає змогу генерувати тести, які відповідають певному профілю. В адаптивному режимі система може

використовувати поточну оцінку знань учня для добору завдань цільового рівня: коли учень успішно виконує декілька завдань середнього рівня, алгоритм підвищує складність наступних; у разі помилок — повертається до легших.

З точки зору навчальної аналітики рівні складності можуть уточнюватися у процесі накопичення даних [36, с. 267]. Завдання, яке апріорно було віднесено до високого рівня, але виявилося легким для більшості, може бути перекласифіковане у середній рівень. І навпаки, завдання, яке учні масово не виконують, може бути віднесено до більш високого рівня. Така динамічна класифікація дозволяє підтримувати актуальність рівнів складності у банку завдань.

Класична тестова теорія історично сформувала базові уявлення про вимірювання в освіті [23, с. 23]. Її вихідною моделлю є уявлення про спостережуваний тестовий бал як суму істинного балу та похибки вимірювання:

$$X = T + E \tag{1.4}$$

де X — спостережуваний результат тесту, T — істинний результат, який відображає реальний рівень знань учня, а E — випадкова похибка вимірювання.

Одним з ключових показників є індекс складності завдання [10, с. 98]. Якщо позначити через N загальну кількість учасників, а через $N_{прав,j}$ — число правильних відповідей на j-те завдання, то індекс складності p_j обчислюється так:

$$p_j = \frac{N_{\text{прав},j}}{N} \tag{1.5}$$

Чим більше значення p_j, тим легшим вважається завдання. Другим базовим параметром СТТ (класична тестова теорія**)** є індекс дискримінації завдання. Він характеризує здатність завдання розрізняти учнів з високою та низькою загальною успішністю [23, с. 78]. Вибірку поділяють на дві групи: сильну з високими сумарними балами та слабку з низькими, після чого обчислюють частку правильних відповідей у кожній групі. Індекс дискримінації D_j визначається як:

$$D_j = p_{\text{верх},j} - p_{\text{низ},j} \tag{1.6}$$

Якщо D_j наближається до 1, це означає, що завдання добре розрізняє сильних і слабких учнів. Значення, близькі до нуля, свідчать про низьку дискримінативну здатність, а негативні — про проблеми з коректністю формулювання. Для автоматизованої системи індекси складності та дискримінації є найбільш практично значущими: вони дозволяють виявляти завдання, які або надто легкі, або надто складні, або не розрізняють рівні підготовки [16, с. 112].

Фундаментальним обмеженням CTT є залежність параметрів від конкретної вибірки учнів [23, с. 34]. Індекс складності, обчислений у сильному класі, може значно відрізнятися від індексу в слабкому, а отже, одна й та сама задача виявлятиметься легкою чи важкою залежно від контексту. Попри це, CTT залишається де-факто стандартом для оцінювання якості традиційних контрольних робіт, а для автоматизованих систем надає прості та наочні метрики для первинного калібрування.

Теорія відповіді на завдання є сучасною психометричною парадигмою, спрямованою на подолання основних обмежень CTT [20; 32, с. 12]. Її ключова ідея полягає в тому, що ймовірність правильної відповіді на певне завдання залежить від латентної характеристики учня — рівня його знань θ та параметрів самого завдання. На відміну від CTT, де аналіз базується на сумарних балах, у IRT кожне завдання моделюється окремо, а результати різних тестових форм можна поставити на єдину шкалу [37, с. 45].

Найпростішою моделлю IRT є однопараметрична логістична модель, відома як модель Раша [41, с. 23]. Вона виходить із припущення, що кожне завдання характеризується єдиним параметром складності b_j, тоді як дискримінативна здатність усіх завдань вважається однаковою. Ймовірність того, що учень з рівнем знань θ_i правильно виконає завдання j, описується логістичною функцією [25, с. 89]:

$$P(X_{ij} = 1|\theta_i b_j) = \frac{1}{1 + e^{-(\theta i - bj)}} \quad (1.7)$$

де $X_{ij} = 1$ означає правильну відповідь. Якщо $\theta_i = b_j$, то імовірність успіху становить 0,5. Якщо $\theta_i > b_j$, імовірність перевищує 0,5 і зростає зі збільшенням різниці; якщо $\theta_i < b_j$, імовірність зменшується.

У більш загальних логістичних моделях вводяться додаткові параметри [32, с. 156]. Двопараметрична модель (2PL) враховує, окрім складності b_j, ще й дискримінативність завдання a_j; тоді формула набуває вигляду:

$$P(X_{ij} = 1|\theta_i a_j b_j) = \frac{1}{1 + e^{-aj(\theta i - bj)}} \quad (1.8)$$

Параметр a_j визначає крутизну кривої: завдання з великим a_j різко відтинає учнів із рівнем знань нижче порогу. У трипараметричній моделі додається ще й параметр c_j, який відображає ймовірність вгадування [37, с. 189]:

$$P(X_{ij} = 1|\theta_i a_j b_j c_j) = c_j + (1 - c_j) \cdot \left[\frac{1}{1 + e^{-aj(\theta i - bj)}}\right] \quad (1.9)$$

Попри наявність різних модифікацій, для шкільної практики та проектування автоматизованих систем особливо привабливою є модель Раша, оскільки вона зберігає відносну простоту, але водночас забезпечує низку важливих властивостей [25, с. 134].

Ключовою перевагою IRT є властивість інваріантності параметрів [32, с. 78]. Оцінка складності завдання b_j, отримана на основі результатів великої та різнорідної вибірки, не залежить від конкретного розподілу здібностей. Так само оцінки рівня знань θ_i окремих учнів не залежать від того, який саме набір завдань вони виконували, за умови, що всі завдання належать до попередньо каліброваного банку. Це створює можливість порівнювати результати, отримані в різний час, у різних класах [37, с. 212].

Саме на основі IRT будуються алгоритми комп'ютерного адаптивного тестування [20]. У таких системах після кожної відповіді оцінюється поточне значення θ_i, а наступне завдання добирається так, щоб його параметр b_j забезпечував максимальну інформацію про рівень знань. Таким чином, кожен респондент отримує індивідуальну послідовність завдань, адаптовану до його рівня, що підвищує ефективність вимірювання [37, с. 245].

Для автоматизованої системи з автоматичною генерацією IRT дає змогу не лише оцінювати складність уже існуючих завдань, а й прогнозувати складність нових екземплярів, згенерованих із певного шаблону [31, с. 234]. Якщо шаблон вже має оцінений параметр b, то всі його ізоформні варіанти можуть апріорно вважатися завданнями близької складності. У подальшому параметри можуть уточнюватися. Таким чином, IRT виступає математичним фундаментом як для калібрування банку завдань, так і для реалізації адаптивних алгоритмів [25, с. 298].

Питання вибору між CTT та IRT у шкільній практиці немає однозначної відповіді [23, с. 167]. З точку зору повсякденної роботи вчителя CTT є значно доступнішою: індекси обчислюються на основі простих підрахунків, результати легко інтерпретуються. Для базового аналізу достатньо звичайних електронних таблиць. IRT потребує значно більших обсягів даних для стабільної оцінки параметрів [32, с. 234]. Калібрування моделі на основі одного класу зазвичай недостатнє; потрібні агреговані дані з багатьох груп. Разом із тим, для автоматизованих систем, які функціонують у масштабі школи чи платформи, ці обмеження стають менш критичними. Якщо система накопичує результати багатьох учнів протягом тривалого часу, необхідний обсяг даних природно формується [36, с. 312]. У такому випадку IRT відкриває суттєві переваги: можливість об'єктивного порівняння результатів різних класів і років, точніший опис рівнів знань, реалізація адаптивних тестів. Реалістичною для практики є комбінована стратегія. Внутрішня логіка функціонування системи — адаптивний добір завдань, калібрування шаблонів, оцінка рівня знань — може спиратися на моделі IRT [37, с. 267]. Зовнішня ж складова — інтерфейс для вчителя, формування звітів — може бути реалізована в термінах, ближчих до CTT. Таким чином, складний психометричний апарат залишається за лаштунками, тоді як користувачі отримують зрозумілу інформацію [8, с. 21].

1.3. Навчальна аналітика та автоматична генерація завдань за рівнями складності

Швидке поширення електронних освітніх середовищ створило принципово новий контекст для оцінювання складності завдань [7, с. 169]. Якщо в традиційній школі вчитель спирається переважно на інтуїтивні судження та обмежені результати контрольних робіт, то в цифровому середовищі кожна дія учня залишає цифровий слід. Сукупність таких слідів формує великі масиви освітніх даних, аналіз яких дає змогу значно точніше оцінювати реальну складність завдань, виявляти аномалії у тестах та будувати персоналізовані траєкторії навчання [36, с. 45]. Навчальна аналітика виступає логічним продовженням психометричного підходу, доповнюючи індекси CTT та параметри IRT поведінковими характеристиками взаємодії учнів із завданнями [36, с. 78].

Навчальна аналітика є міждисциплінарним напрямом, що поєднує методи педагогіки, психології, статистики та інформатики для вимірювання, збору та аналізу даних про здобувачів освіти з метою оптимізації навчального процесу [36, с. 12]. На відміну від класичного психометричного аналізу, який фокусується на підсумкових результатах тестів, навчальна аналітика працює з даними, що генеруються неперервно: від входу користувача в систему до завершення курсу.

Джерелом даних виступають журнали подій у системах LMS [18, с. 39]. Вони містять інформацію про час і частоту входів, переходи між сторінками, перегляд матеріалів, спроби виконання тестів, звернення до підказок. У контексті оцінювання складності особливий інтерес становлять дані про результати тестування, часові показники, структурні характеристики взаємодії [36, с. 156].

Ключове завдання навчальної аналітики полягає в оптимізації навчання [36, с. 89]. На основі освітніх даних система може автоматично виявляти завдання, що ведуть себе аномально, що часто вказує на методичні або технічні помилки. Система формує аналітичні профілі учнів, які відображають їхній рівень підготовки, темп роботи, схильність до певних типів помилок. На основі

цих профілів персоналізується навчання, пропонуючи кожному учню завдання відповідної складності [18, с. 41].

Концептуально процес можна подати як послідовність етапів: дані → індикатори → інтерпретація → рішення [36, с. 134]. Спочатку збираються сирі дані з LMS, далі вони агрегуються у показники, які потім інтерпретуються з погляду дидактичних цілей, після чого вчитель або система приймають рішення щодо адаптації курсу, зміни банку завдань чи корекції методики.

Психометричні моделі CTT та IRT оперують переважно результативними показниками — правильністю відповідей, індексами складності та дискримінації [23, с. 56; 32, с. 89]. Навчальна аналітика розширює цю картину, залучаючи поведінкові показники [36, с. 178]. Разом ці дані дозволяють переходити від суто емпіричного індексу важко/легко до багатовимірного уявлення про складність.

Одним із базових індикаторів є час виконання завдання [36, с. 201]. Якщо позначити через t_{ij} час, який витрачає i-й учень на j-те завдання, можна розглядати середній час по групі:

$$\underline{t_j} = \frac{1}{N}\sum \qquad t_{\underline{y}} \tag{1.10}$$

де N — кількість учнів. Зазвичай спостерігається тенденція: чим більше значення $\bar{t_j}$, тим більшим є когнітивне навантаження [10, с. 112]. Водночас час потребує контекстної інтерпретації: надто короткі інтервали в поєднанні з високою успішністю можуть свідчити про тривіальність завдання, тоді як надто короткий час і випадкова правильність — про ймовірне вгадування.

Іншим важливим показником є кількість спроб до успішного виконання [36, с. 223]. У системах формувального оцінювання учень може мати кілька спроб відповісти на завдання. Високі значення середньої кількості спроб у поєднанні з відносно низьким відсотком правильних фінальних відповідей вказують на об'єктивно високу складність або на недостатню прозорість умови. Зміна відповіді перед остаточним підтвердженням є показником невпевненості та когнітивного навантаження [10, с. 145]. У логах фіксуються послідовності

дій учня: спочатку вибір одного варіанта, потім його скасування та вибір іншого. Масова наявність таких коливань по одному завданню може свідчити про неоднозначність формулювання.

Окремий пласт інформації надає аналіз типових помилок [16, с. 89]. Розкладаючи неправильні відповіді за категоріями, можна виявити, на якому саме етапі розв'язання учні найчастіше припускаються помилок: на етапі розуміння умови, вибору формули, виконання обчислень. У тестах з вибором відповіді дистрактори можуть бути спроектовані як репрезентації типових помилок, і тоді частота вибору кожного з них дає змогу реконструювати профіль труднощів [23, с. 134].

Узагальнено, інтегрований показник фактичної складності завдання можна подати як функцію кількох змінних:

$$D_j^* = f(p_j, \underline{t_j}, \underline{k_j}, c_j, e_j) \tag{1.11}$$

де p_j — частка правильних відповідей, $\bar{t_j}$ — середній час виконання, $\bar{k_j}$ — середня кількість спроб, c_j — частка учнів, які змінювали відповідь, e_j — показники помилок. У спрощеному випадку можна ввести нормовані показники [3, с. 270]:

$$D_j^* = a_1(1 - p_j) \; + \; a_2\underline{t_j} + a_3\underline{k_j} + a_4 c_j \tag{1.12}$$

де $\tilde{t_j}$, $\tilde{k_j}$, $\tilde{c_j}$ — стандартизовані змінні, а α_1, ..., α_4 — вагові коефіцієнти [10, с. 156]. Такий інтегральний показник може використовуватися для автоматичного ранжування завдань за фактичною складністю та виявлення аномальних пунктів.

Зростання обсягів освітніх даних зумовило перехід від ручного аналізу до використання методів аналізу даних і машинного навчання для класифікації складності завдань [36, с. 289]. На відміну від експертних оцінок, ці методи дозволяють виявляти закономірності у великих вибірках та будувати формалізовані моделі, які здатні прогнозувати складність нових завдань до їх масового використання.

Один з напрямів — кластеризація завдань на основі їхніх емпіричних і поведінкових характеристик [36, с. 312]. Завдання представляються у вигляді векторів ознак, після чого застосовуються алгоритми неконтрольованого навчання. У результаті формується кілька кластерів, які можуть інтерпретуватися як групи завдань легкого, середнього та високого рівнів складності. Перевага такого підходу полягає у відсутності потреби в попередньому ручному маркуванні: рівні складності виявляються з даних.

Другий напрям — регресійне моделювання [10, с. 167]. Наприклад, можна будувати модель, що прогнозує очікуваний середній час виконання завдання на основі його структурних характеристик. Перспективним є використання методів обробки природної мови для аналізу текстових характеристик умови завдання [42, с. 323]. На рівні класичних підходів застосовується аналіз довжини речень, частоти термінів, складності синтаксичних конструкцій. Більш сучасні підходи базуються на векторних поданнях тексту, які дозволяють порівнювати завдання за семантичною близькістю.

Численні практичні системи використовують також рейтинг подібні схеми на кшталт модифікованої системи Ело, адаптованої до контексту учень-завдання [36, с. 367]. Кожному учневі i приписується рейтинг $R^{(s)}_i$, а кожному завданню j — рейтинг складності $R^{(q)}_j$. Очікувана ймовірність правильної відповіді може бути задана логістичною функцією різниці рейтингів:

$$P(X_{ij} = 1) = \frac{1}{1 + 10^{\frac{-(R(s)i - R(q)j)}{400}}} \qquad (1.13)$$

Після кожної відповіді рейтинги оновлюються: у разі успіху учня рейтинг завдання дещо знижується, а рейтинг учня зростає, і навпаки. Така схема дозволяє поступово уточнювати оцінки складності на основі потоку нових даних без проведення окремої масової апробації.

На основі описаних методів будуються системи рекомендацій, які пропонують учням завдання оптимальної складності [18, с. 42]. Якщо система має оцінку рівня знань учня θ_i та банк завдань із відомими параметрами складності b_j, вона може динамічно добирати такі завдання, для яких

ймовірність успіху є помірною, забезпечуючи баланс між фрустрацією та нудьгою [37, с. 198]. Це безпосередньо пов'язує навчальну аналітику з адаптивним навчанням.

Автоматична генерація тестових завдань виступає ключовим механізмом подолання проблеми старіння тестового контенту, коли завдання з часом стають добре відомими учням і втрачають диференціюючу здатність [29, с. 45]. У контексті системи оцінювання з автоматичною генерацією (Automatic Item Generation, AIG) розглядається не лише як технічний інструмент для множення кількості запитань, а як складний педагогічно-психометричний процес, у якому кожен згенерований екземпляр має відповідати вимогам валідності, коректності та контрольованої складності [29, с. 67].

Під автоматичною генерацією розуміють процес створення нових тестових пунктів за допомогою комп'ютерних алгоритмів на основі певної моделі предметної області та формалізованих шаблонів [1, с. 36]. У найзагальнішому вигляді можна вважати, що кожне згенероване завдання є результатом застосування деякого генератора G до шаблону S та набору параметрів α:

$$item = G(S, a) \tag{1.14}$$

Найпоширенішим у школі є шаблонний підхід [29, с. 89]. У цьому випадку створюється скелет завдання — текстова та структурна схема, у якій деякі елементи позначаються як змінні параметри. Для кожного шаблону задається допустимий діапазон значень параметрів та обмеження на їхні поєднання. Алгоритм генерації підставляє конкретні значення, формуючи новий екземпляр. Наприклад, параметризована задача з алгебри може мати загальний вигляд: «Розв'яжіть рівняння $ax + b = 0$», де a і b — цілі числа з наперед визначеного діапазону, причому $a \neq 0$. Конкретний варіант отримує уточнення, наприклад $3x - 6 = 0$ [4, с. 167].

Розширенням шаблонного підходу є параметризована генерація, коли змінюються не лише числові значення, а й контекст умови, імена об'єктів, формулювання сюжетної ситуації [31, с. 145]. Наприклад, одна й та сама

математична модель може описувати рух автомобіля, потяга, пішохода. У цьому разі зберігається структура задачі та її когнітивний рівень, однак змінюється зовнішня оболонка, що зменшує ризик механічного впізнавання.

Іншим напрямом є онтологічні підходи, які базуються на формальному описі предметної області у вигляді семантичних мереж або онтологій [31, с. 189]. У такій моделі поняття предмета поєднані відношеннями типу є частиною, є різновидом, є прикладом. На основі цих зв'язків система може автоматично генерувати завдання, що перевіряють відношення між поняттями. Особливу групу становлять підходи до конструктивної генерації, коли завдання створюється через синтез візуальних або структурних об'єктів: графіків функцій, діаграм, геометричних фігур, фрагментів коду [6, с. 123].

На сучасному етапі AIG активно доповнюється генеративними підходами на основі моделей штучного інтелекту, насамперед великих мовних моделей [42, с. 325]. Вони здатні на основі вхідного опису теми генерувати нові формулювання, варіанти контекстів, альтернативні дистрактори. Втім, такі моделі потребують ретельної верифікації: необхідно контролювати фактичну правильність, уникати семантичних суперечностей [22].

Щоб автоматична генерація була не лише варіативною, а й педагогічно керованою, у моделі завдання необхідно явно представити його параметри складності [29, с. 134]. Йдеться як про апріорну задану наперед складність, так і про можливість динамічного уточнення на основі даних навчальної аналітики. Формально кожен шаблон можна описати як пару (S, Θ), де S — структурно-змістовий опис, а Θ — множина його параметрів [31, с. 56]. Доцільно ввести функцію складності $D : \Theta \to \mathbb{R}$, яка кожному набору параметрів $\alpha \in \Theta$ ставить у відповідність числову оцінку складності $D(\alpha)$. У найпростішому випадку це може бути лінійна модель:

$$D(a) = w_1 a_1 + w_2 a_2 + \ldots + w_k a_k \qquad (1.15)$$

де α_i — окремі характеристики згенерованого екземпляра, а w_i — вагові коефіцієнти, підібрані експертно або на основі даних [10, с. 178].

На практиці для регулювання складності в шаблонній генерації встановлюють безпосередні зв'язки між окремими параметрами та когнітивним навантаженням [29, с. 178]. Використання невеликих цілих чисел у межах від 1 до 10 зумовлює низьку обчислювальну складність і дозволяє зосередитися на структурі розв'язання. Збільшення діапазону, введення дробових чи від'ємних чисел істотно підвищує вимоги до обчислювальних навичок. Аналогічно, завдання з двома дистракторами є легшими, ніж завдання з п'ятьма варіантами, серед яких кілька дистракторів побудовано на типових помилках [16, с. 123].

Необхідним елементом моделі є система правил-обмежень, що накладаються на параметри α [31, с. 201]. Вони гарантують, що згенерований екземпляр залишиться у педагогічно та математично припустимій області. Наприклад, при генерації задачі на знаходження коренів квадратного рівняння можуть задаватися обмеження, щоб дискримінант був додатним і цілим, якщо планується робота в множині раціональних чисел [4, с. 189].

Крім суто числових характеристик, важливим є семантичне тегування шаблонів завдань [29, с. 234]. Кожен шаблон повинен бути пов'язаний із певними темами та підтемами навчальної програми, а також із рівнями цілей [19]. Це дозволяє контролювати не лише загальний рівень складності, а й змістову репрезентативність тесту.

У динамічній системі оцінювання складність конкретного екземпляра доцільно уточнювати безпосередньо перед його пред'явленням учню [3, с. 71]. Для цього може використовуватися комбінація апріорної оцінки $D(\alpha)$ та апостеріорних психометричних і аналітичних даних:

$$D_{\text{еф}} = \lambda \cdot D(a) \ + (1 - \lambda) \cdot \hat{b_j} \qquad (1.16)$$

де $\hat{b_j}$ — емпірично оцінена складність завдань даного шаблону, а $\lambda \in [0;1]$ — коефіцієнт довіри до теоретичної моделі [32, с. 201]. Таким чином, у процесі накопичення даних система поступово коригує свої уявлення про фактичну складність.

Автоматизація процесу створення тестового контенту не знімає, а посилює вимоги до його педагогічної якості [8, с. 20]. Кожне автоматично

згенероване завдання повинно відповідати низці критеріїв. Змістово-методичний вимір охоплює відповідність завдання навчальним програмам, цілям освітньої галузі та віковим особливостям учнів [12]. Система не повинна генерувати завдання, що виходять за межі передбаченого курсу або спираються на поняття, не вивчені на певному етапі.

Формально-лінгвістичні вимоги пов'язані з семантичною коректністю та однозначністю формулювань [29, с. 267]. Текст умови має бути логічно послідовним, граматично правильним і стилістично нейтральним. Особливо небезпечними є двозначні конструкції, подвійні заперечення, використання термінів без попереднього введення. Автоматична генерація тексту повинна супроводжуватися механізмами виявлення потенційних суперечностей.

Окремим пунктом є валідність ключів, тобто правильність автоматично обчислених відповідей для кожного згенерованого екземпляра [16, с. 134]. У шаблонній генерації це передбачає наявність коректної функції, яка для заданого набору параметрів обчислює правильну відповідь. Для задач із вибором відповіді ця функція також повинна генерувати дистрактори, які не збігаються з правильним результатом і відображають типові помилки [30, с. 51]. Некоректність ключа є однією з найсерйозніших помилок, оскільки призводить до систематичного неправильного оцінювання великої кількості учнів.

Висновки до І розділу

У першому розділі здійснено теоретичне обґрунтування системи оцінювання з автоматичною генерацією завдань на основі рівня складності. Показано, що традиційні форми контролю мають суттєві обмеження: суб'єктивність, низьку інформативність, значні часові витрати. Автоматизовані системи розглядаються як інструмент забезпечення систематичності та об'єктивності оцінювання.

Систематизовано психометричні підходи: класична тестова теорія є зрозумілою для вчителя, але залежить від вибірки; теорія відповіді на завдання забезпечує інваріантність параметрів і становить основу адаптивного тестування. Обґрунтовано комбіноване використання обох підходів.

Розглянуто складність завдання як інтегральну характеристику, що залежить від змістових, логічних, мовних та часових факторів.

Показано, що в електронних середовищах оцінювання складності базується не лише на результаті, а й на поведінкових показниках: часі виконання, кількості спроб, структурі помилок. Методи машинного навчання дають змогу автоматично класифікувати завдання за рівнями складності. Узагальнено підходи до автоматичної генерації: шаблонний, параметризований, онтологічний, генеративний. Кожен шаблон повинен містити метадані про складність, тематику та обмеження для запобігання некоректним варіантам.

Сформульовано вимоги до системи: підтримка формувального оцінювання з оперативним зворотним зв'язком; архітектура банку на основі шаблонів з параметрами складності; інтеграція психометричних моделей з навчальною аналітикою для динамічного калібрування та адаптивного добору завдань; прозора інтерпретація результатів.

РОЗДІЛ ІІ. ПРАКТИЧНИЙ АНАЛІЗ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

2.1. Характеристика та архітектура програмної системи

Загальна задача полягає у створенні програмної системи, яка формує динамічні тестові матеріали із завданнями заданого рівня складності, забезпечує автоматизований процес тестування та здійснює об'єктивне оцінювання на основі накопичених даних [1, с. 37]. Система автоматизує повний цикл контролю знань: від створення банку параметризованих шаблонів до генерації індивідуалізованих тестів, проведення оцінювання та подання аналітичних звітів [29, с. 98].

Функціональне призначення передбачає задоволення потреб трьох категорій користувачів. Учень використовує систему для проходження тестування і отримує оперативний зворотний зв'язок у вигляді кількісної оцінки, відсотка правильних відповідей та інформації про рівень засвоєння [9, с. 16]. Вчитель створює і редагує банк шаблонів завдань, налаштовує параметри генерації, визначає рівні складності та аналізує результати [31, с. 89]. Адміністратор керує обліковими записами, глобальними налаштуваннями та інтеграцією з зовнішніми системами.

Система покликана усунути ключові проблеми, визначені в теоретичному розділі [8, с. 18]. Вона мінімізує вплив суб'єктивного фактора завдяки формалізованим показникам складності та автоматизованим алгоритмам. Застосування параметризованих шаблонів і механізмів автоматичної генерації забезпечує масштабованість банку завдань і зменшує ймовірність запам'ятовування відповідей [29, с. 134]. Можливість динамічного добору завдань різного рівня складності створює передумови для адаптивного тестування [37, с. 178]. Накопичення даних та їх аналіз дозволяє підвищити обґрунтованість педагогічних рішень [29, с. 234].

До функціональних вимог належить підтримка реєстрації, аутентифікації та авторизації користувачів із розмежуванням прав доступу [33]. Ключовим елементом є банк шаблонів тестових завдань, який підтримує повний цикл

управління: створення, редагування, перегляд, деактивацію. Вчитель має можливість записувати умови задач у вигляді параметризованих шаблонів, що містять змінні параметри та правила формулювання відповідей [31, с. 112]. Для кожного шаблону задаються показники складності відповідно до критеріїв: рівнів таксономії навчальних цілей [19], кількості логічних кроків розв'язання, обсягу умови, типу відповіді.

Система підтримує різні режими проведення тестувань: формувальне, підсумкове та адаптивне [9, с. 17]. Формувальне орієнтоване на поточний контроль і надання зворотного зв'язку, підсумкове — на оцінювання досягнутих результатів, адаптивне — на динамічну зміну рівня складності залежно від відповідей учня [37, с. 201]. Обов'язковою є функція збору та обробки результатів тестувань, а також логів дій користувачів [36, с. 145]. На базі накопичених даних реалізується навчальна аналітика: побудова розподілу оцінок, аналіз середніх показників складності, виявлення проблемних тем і типових помилок.

До нефункціональних вимог належить продуктивність системи [35, с. 234]. Вона повинна забезпечувати одночасну коректну роботу заданої кількості користувачів без деградації часу відгуку, зокрема під час масового проходження тестувань. Важливою є вимога надійності та відмовостійкості — система повинна зберігати цілісність даних навіть у разі часткових відмов окремих компонентів [39, с. 156]. Використання черг повідомлень, ізольованих мікросервісів та механізмів резервного копіювання сприяє підвищенню стійкості [34].

Особлива увага приділяється безпеці. Система реалізує механізми захищеної автентифікації на основі токенів JWT (JSON Web Token) [33], розмежування прав доступу відповідно до ролей, захисту персональних даних та результатів. Передбачається застосування сучасних криптографічних протоколів для захищеної передачі інформації між клієнтською та серверною частинами [3, с. 267]. Суттєвою вимогою є масштабованість, яка досягається за рахунок мікросервісної архітектури [39, с. 78]. Це дозволяє незалежно

розширювати окремі функціональні модулі без зупинки всієї системи. Система має відкриті програмні інтерфейси API(Інтерфейс Програмування Застосунків), що забезпечують інтеграцію з існуючими системами управління навчанням, електронними журналами та іншими освітніми платформами [18, с. 40].

Розроблювана система орієнтована на впровадження в типовому закладі загальної середньої освіти [11]. Сучасний заклад має один або декілька комп'ютерних класів, оснащених персональними комп'ютерами з доступом до Інтернету. Поширення набуває підхід BYOD (Bring Your Own Device/Принеси свій власний пристрій), за якого учні використовують власні мобільні пристрої [5, с. 48]. У більшості закладів вже використовуються системи управління навчанням або електронні журнали [7, с. 134]. Водночас такі системи часто мають обмежені можливості щодо автоматичної генерації завдань, психометричного визначення складності та розвиненої навчальної аналітики. Це зумовлює потребу у впровадженні спеціалізованої системи, яка може бути інтегрована з наявною інфраструктурою [2, с. 24].

З технічної точки зору доцільним є використання веб-орієнтованої клієнт–серверної архітектури, реалізованої на основі стеку Node.js, NestJS, мікросервісного підходу, брокера повідомлень Kafka, протоколів REST(Representational State Transfer/Передача репрезентативного стану) та gRPC (Remote Procedure Call) для міжсервісної взаємодії та клієнтської частини на React із бібліотекою Ant Design [24; 26; 27]. Веб-орієнтована архітектура забезпечує кросплатформеність, можливість роботи на різних типах пристроїв. Клієнт–серверна модель дозволяє централізовано зберігати та обробляти дані [35, с. 89]. Мікросервісна архітектура забезпечує масштабованість і гнучкість, дає змогу незалежно розвивати окремі модулі [39, с. 118].

Архітектура системи ґрунтується на мікросервісному підході. На серверній стороні використовуються Node.js та NestJS для реалізації окремих мікросервісів, Kafka — для подіє орієнтованого обміну повідомленнями [34], gRPC — для високопродуктивної міжсервісної взаємодії [24], а на клієнтській стороні — React із Ant Design для побудови веб-інтерфейсу [27]. Доступ

клієнтських застосунків до внутрішніх сервісів здійснюється через API Gateway(Application Programming Interface Gateway/Шлюз Програмного Інтерфейсу Застосунків) як єдину точку входу [38, с. 123].

Загальну структуру системи формують такі основні елементи: API Gateway, Auth Service для автентифікації та управління користувачами, Item Bank Service для банку шаблонів завдань і параметрів складності, Test Generation Service для формування тестів, Testing Session Service для проведення сесій тестування, Analytics Service для аналітики та оновлення параметрів складності, Kafka Broker для обміну подіями між сервісами, клієнтські веб-застосунки для ролей учня та вчителя на React.

API Gateway відіграє роль фасаду для всіх клієнтських запитів [38, с. 156]. Він приймає HTTP (Hypertext Transfer Protocol/Протокол Передачі Гіпертексту) запити від браузерів, виконує попередню обробку, перевірку JWT-токена, маршрутизацію та перенаправляє їх до відповідних мікросервісів. Auth Service відповідає за автентифікацію та авторизацію користувачів, управління обліковими записами та ролями [33]. Після успішного входу він видає клієнтові JWT-токен для доступу до інших сервісів.

Item Bank Service реалізує функціональність банку шаблонів тестових завдань [29, с. 167]. У його сферу відповідальності входить зберігання текстів завдань, параметризованих шаблонів, наборів параметрів та психометричних параметрів складності, дискримінації та інших показників, які оновлюються за результатами тестувань [32, с. 134]. Test Generation Service забезпечує формування тестів на основі заданих критеріїв: тематики, рівнів складності, типів завдань [31, с. 178]. На основі даних з Item Bank Service він відбирає відповідний набір шаблонів та генерує конкретний набір завдань із підставленими параметрами.

Testing Session Service відповідає за управління життєвим циклом сесій тестування [35, с. 312]. Він реєструє початок сесії, відображає учневі згенерований тест, приймає й зберігає відповіді, фіксує час виконання. По завершенню сесії обчислює первинні результати та формує події для

аналітичної обробки. Analytics Service здійснює збір та обробку даних про результати тестувань, включно з логами дій. На основі накопичених даних він обчислює показники, узгоджені з моделями CTT та IRT: індекси складності, дискримінації, аналізує статистику виконання завдань та формує аналітичні звіти [23, с. 89; 32, с. 178]. За потреби він оновлює параметри складності окремих завдань у Item Bank Service.

Взаємодія між мікросервісами здійснюється за допомогою gRPC для синхронних викликів із низькими затримками та Kafka для асинхронного обміну подіями й забезпечення слабкого зв'язування між компонентами [24; 33]. Розглянемо типовий сценарій роботи з позиції учня. Учень відкриває веб-застосунок у браузері. Після введення облікових даних фронтенд надсилає запит на аутентифікацію до API Gateway, який перенаправляє запит до Auth Service [33]. У разі успішної перевірки видається JWT-токен, який зберігається на клієнті та додається до подальших запитів у заголовку авторизації.

Після автентифікації учень переходить до розділу тестування. Фронтенд надсилає запит до API Gateway для отримання переліку доступних тестів. При виборі конкретного тесту API Gateway надсилає запит до Test Generation Service [38, с. 189]. У запиті вказуються ідентифікатор курсу або теми, бажаний режим, цільовий рівень складності. Test Generation Service звертається до Item Bank Service через gRPC для отримання відповідних шаблонів завдань і їхніх параметрів складності [24]. Отримавши набір шаблонів, сервіс здійснює параметризацію та формує структурований тест [31, с. 201]. Згенерований тест передається у Testing Session Service та повертається через API Gateway на фронтенд.

Після початку тесту Testing Session Service реєструє створення нової сесії [35, с. 367]. Кожна відповідь, яку надсилає учень, передається через API Gateway до Testing Session Service, де зберігається разом із часовими мітками. У разі адаптивного режиму Testing Session Service може звертатися до Test Generation Service для динамічного добору наступних завдань на основі попередніх відповідей [37, с. 234].

Після завершення тесту Testing Session Service обчислює первинні результати: кількість правильних відповідей, відсоток виконання, можливі шкальні оцінки та формує подію TestSessionCompleted, яку публікує у Kafka [34]. Ця подія містить агреговану інформацію про сесію та деталізовані дані щодо кожного завдання. Analytics Service підписаний на відповідні теми Kafka [36, с. 234]. Отримуючи події про завершення сесій, він оновлює внутрішні статистичні моделі: обчислює показники складності та дискримінації завдань відповідно до методів CTT та IRT [23, с. 123; 32, с. 201], аналізує розподіл відповідей і час виконання. У результаті Analytics Service формує оновлені значення параметрів складності для окремих завдань [3, с. 70]. Ці оновлення надсилаються до Item Bank Service через gRPC або Kafka.

Модель даних є основою для коректного функціонування системи, оскільки вона визначає структуру зберігання, обробки та взаємозв'язку всієї інформації [28, с. 234; 35, с. 156]. Проектування моделі виконано з урахуванням логіки мікросервісної архітектури, вимог до масштабованості, підтримки психометричних моделей CTT та IRT [40].

Усі категорії користувачів реалізуються у вигляді єдиної сутності Users, із якою пов'язана окрема сутність Roles. Сутність Users містить: унікальний ідентифікатор, електронну пошту, хеш пароля, персональні дані, посилання на роль, статус активності, дати створення й оновлення. Базовою сутністю для генерації завдань є шаблон ItemTemplate. Основні атрибути: ідентифікатор, назва, текст умови з параметрами, тип завдання, навчальна тема, ідентифікатор вчителя-автора, посилання на параметри складності, статус доступності.

Сутність DifficultyProfile реалізує психометричні та дидактичні характеристики складності завдання [16, с. 98]. Основні атрибути: рівень складності за шкалою, когнітивний рівень за таксономією Блума [19], індекс складності за CTT [23, с. 56], індекс дискримінації [23, с. 78], параметр складності в IRT-моделі [37, с. 156], параметр дискримінації в IRT [32, с. 167], середній час виконання, середня кількість спроб. Оскільки система працює з параметризованими завданнями, необхідна окрема сутність для фіксації

конкретного згенерованого екземпляра GeneratedItem з атрибутами: посилання на шаблон, фактичні значення параметрів у форматі JSON, правильна відповідь [31, с. 234].

Сутність TestSession описує процес проходження тесту конкретним учнем: посилання на користувача-учня, тип тестування, час початку та завершення, статус активності, підсумковий бал [35, с. 401]. Сутність TestResult зберігає результати виконання окремих завдань: посилання на сесію, посилання на згенероване завдання, відповідь учня, ознака правильності, час виконання, кількість спроб [36, с. 267]. Для реалізації навчальної аналітики використовується сутність StudentActionLog: посилання на учня, пов'язана сесія, тип дії, додаткові дані у форматі JSON, час фіксації [36, с. 289].

Ефективна робота системи ґрунтується на чітко визначених інтерфейсах взаємодії. Застосовуються два основні підходи до обміну даними: синхронна взаємодія на основі REST та gRPC та асинхронна взаємодія через брокер подій Kafka. Взаємодія клієнтської частини на React з серверною частиною здійснюється через REST API, доступ до якого централізовано забезпечується через API Gateway [27]. Усі запити виконуються за протоколом HTTPS/HTTP із використанням JSON. Для захисту доступу використовується механізм JWT-автентифікації [33].

До основних REST-ендпоїнтів належать: /auth/login для аутентифікації, /auth/refresh для оновлення токена, /users/me для отримання інформації про користувача; /tests/available для списку доступних тестів, /tests/{id}/start для запуску тестової сесії, /tests/{sessionId}/answer для надсилання відповіді, /tests/{sessionId}/finish для завершення тестування; /results/my для перегляду власних результатів, /analytics/summary для зведених аналітичних показників; /items для перегляду списку шаблонів, створення, редагування та видалення.

Внутрішня взаємодія між мікросервісами реалізується двома механізмами. gRPC використовується для швидкого обміну даними у випадках, коли необхідна негайна відповідь [24]: Test Generation Service звертається до Item Bank Service для отримання шаблонів, Testing Session Service отримує

наступні завдання в адаптивному режимі, Analytics Service оновлює параметри складності після аналізу результатів. Kafka використовується для передавання подій, які не потребують негайної відповіді [34]: події завершення сесії тестування, події змін у банку завдань, події дій учнів. Такий підхід знижує зв'язність між сервісами та підвищує відмовостійкість.

Інтерфейси системи передбачають можливість інтеграції з наявними у закладах LMS (Learning Management System/Система управління навчанням) та електронними журналами [7, с. 178; 18, с. 41]. Основними напрямами інтеграції є експорт результатів тестування до зовнішніх електронних журналів, імпорт списків учнів і навчальних груп із LMS для автоматичного створення облікових записів, передавання інформації про тести як про електронні навчальні активності. Обмін даними здійснюється через REST API у стандартних форматах JSON або CSV [38, с. 289].

2.2. Алгоритмічне та технологічне забезпечення системи

Ключовим завданням розроблюваної системи є динамічне оновлення рівнів складності тестових завдань на основі реальних результатів їх виконання учнями [3, с. 68]. Це дозволяє усунути суб'єктивність апріорної оцінки, підвищити точність добору завдань та створити передумови для реалізації адаптивних алгоритмів [31, с. 189]. Такий підхід забезпечує постійне вдосконалення системи, оскільки кожна нова сесія тестування надає додаткову інформацію для уточнення параметрів складності [36, с. 201].

Оцінювання складності в системі ґрунтується на поєднанні класичних психометричних показників CTT, спрощених елементів IRT та поведінкових показників [23, с. 45; 32, с. 89]. Усі розрахунки виконуються в Analytics Service після завершення кожної сесії тестування на основі подій, що надходять через брокер повідомлень. Такий підхід дозволяє здійснювати обчислення асинхронно, не блокуючи основні процеси системи та забезпечуючи швидкий відгук для користувачів.

Основним показником складності є індекс складності, що визначається як частка правильних відповідей (1.1) [16, с. 67].

Значення p (індекс складності) належить діапазону [0;1], де $p \to 1$ означає дуже легке завдання, $p \to 0$ — дуже складне завдання [23, с. 56]. Наприклад, якщо завдання правильно виконали 75 учнів зі 100, індекс складності становить 0.75, що відповідає помірно легкому рівню. У системі цей показник накопичується для кожного шаблону як ковзне середнє, що забезпечує плавну адаптацію до нових даних [10, с. 134]:

$$p_{new} = \alpha \cdot p_{current} + (1 - \alpha) \cdot p_{old} \quad (2.1)$$

де α — коефіцієнт оновлення, $p_{current}$ — індекс за новими даними [3, с. 69]. Це дозволяє уникнути різких стрибків складності та стабілізувати показники при малій вибірці. Зазвичай значення α встановлюється в межах 0.2–0.3, що означає, що нові дані мають вагу 20–30%, а попередні накопичені дані — 70–80%. Така стратегія забезпечує баланс між чутливістю до змін та стабільністю оцінок [36, с. 245].

Індекс дискримінації характеризує здатність завдання розрізняти сильних і слабких учнів [23, с. 78] (Додаток Г). Це важлива характеристика якості тестового завдання, оскільки завдання, яке однаково виконують як сильні, так і слабкі учні, не має діагностичної цінності [16, с. 89]. Для його обчислення вибірка розбивається на дві підгрупи: група сильних учнів — верхні 27% за підсумковим балом, група слабких — нижні 27% [23, с. 82]. Вибір саме 27% є стандартною практикою в психометрії, що забезпечує достатню репрезентативність груп при збереженні контрастності між ними. Нехай p_{strong} — частка правильних відповідей у сильній групі, p_{weak} — частка у слабкій групі. Тоді індекс дискримінації D:

$$D = p_{strong} - p_{weak} \quad (2.2)$$

Інтерпретація: $D \approx 0$ означає, що завдання не диференціює учнів, $D > 0.3$ свідчить про хорошу дискримінаційну здатність, $D < 0$ вказує на ознаку помилки в завданні. Наприклад, якщо у сильній групі завдання правильно виконали 80% учнів, а у слабкій — 40%, індекс дискримінації становить 0.4, що є відмінним

показником [16, с. 102]. Від'ємне значення індексу може свідчити про те, що завдання сформульовано некоректно або ключ відповіді містить помилку.

Окрім класичних психометричних індексів, у системі враховуються поведінкові характеристики, які надають додаткову інформацію про когнітивну складність завдання [36, с. 223]. Середній час виконання обчислюється як $T_{avg} = (1/N)\ \Sigma\ t_i$, де t_i — час виконання завдання i. Чим більше середній час, тим більша когнітивна складність. Наприклад, завдання, на виконання якого учні витрачають у середньому 5 хвилин, є когнітивно складнішим, ніж завдання з середнім часом виконання 1 хвилина, навіть якщо їхні індекси правильності однакові. Середня кількість спроб обчислюється аналогічно $A_{avg} = (1/N)\ \Sigma\ a_i$. Чим більше спроб — тим більша ймовірність високої складності або некоректного формулювання [36, с. 234].

Фінальний інтегральний рівень складності обчислюється шляхом нормалізації всіх метрик та їх зваженого об'єднання [3, с. 70]:

$$L = w_1 \cdot (1 - p)\ +\ w_2 \cdot D + w_3 \cdot norm(T_{avg}) + w_4 \cdot norm(A_{avg}) \qquad (2.4)$$

де w_1, w_2, w_3, w_4 — вагові коефіцієнти, norm(x) — функція нормалізації до діапазону [0;1] [10, с. 167]. Вагові коефіцієнти визначають відносну важливість кожного показника. У системі за замовчуванням використовуються значення w_1=0.4, w_2=0.3, w_3=0.2, w_4=0.1, що надає найбільшу вагу класичному індексу складності, але враховує і додаткові характеристики. На основі L здійснюється переведення завдання до дискретного рівня складності за шкалою 1–10. Отримане значення оновлюється у полі difficultyLevel сутності DifficultyProfile.

Як доповнення до CTT у системі застосовується спрощена логістична IRT-модель:

$$P(\theta) = \frac{1}{1 + e^{-a(\theta - b)}} \qquad (2.5)$$

де θ — рівень підготовки учня, b — параметр складності завдання, a — параметр дискримінації. Ця модель дозволяє передбачити ймовірність правильної відповіді конкретного учня на конкретне завдання, що є основою для адаптивного тестування [25, с. 234]. Параметр b оновлюється ітеративно на

основі накопичених відповідей методом максимізації правдоподібності (Додаток Б). Після обробки пакетів результатів Analytics Service оновлює профіль складності в сутності DifficultyProfile, передає нові параметри до Item Bank Service через gRPC або Kafka [24; 34]. Ці параметри надалі використовуються Test Generation Service під час добору завдань. Таким чином, рівень складності є динамічним, самооновлюваним та емпірично обґрунтованим [3, с. 71].

Апріорна класифікація складності є необхідним етапом на початковому етапі життєвого циклу тестового завдання. На відміну від апостеріорної оцінки, що ґрунтується на емпіричних даних, апріорна класифікація дозволяє призначити початковий рівень складності новому завданню ще до його фактичного використання учнями. Це особливо важливо для новостворених шаблонів, які ще не мають статистики виконання. Алгоритм базується на формалізованому аналізі дидактичних, когнітивних та мовних характеристик: тип завдання, кількість логічних кроків розв'язання, лінгвістична складність умови, когнітивний рівень за таксономією Блума [19] (Додаток А).

Тип завдання визначає рівень когнітивного навантаження. Кожному типу відповідає базовий коефіцієнт складності: одна правильна відповідь — 0.2, кілька відповідей — 0.35, встановлення відповідностей — 0.5, коротка відповідь — 0.7, розгорнута відповідь — 0.9. Ці коефіцієнти відображають типову складність кожного формату завдання. Наприклад, завдання з однією правильною відповіддю є найпростішим, оскільки учень має 25% шансів вгадати правильну відповідь навіть без знань [16, с. 78]. Розгорнута відповідь, навпаки, вимагає глибокого розуміння та вміння формулювати думки, тому має найвищий коефіцієнт [29, с. 167].

Кількість кроків розв'язання s характеризує структурну складність [31, с. 112]. Для нормалізації використовується $k_{steps} = min(s, s_{max}) / s_{max}$. Наприклад, якщо завдання потребує 3 логічних кроки, а максимальне значення встановлено як 5, коефіцієнт становить 0.6. Лінгвістична складність оцінюється на основі кількості слів: $k_{text} = min(w, w_{max}) / w_{max}$. Довгі формулювання зазвичай важко

сприймаються учнями і потребують більше часу на читання та розуміння [16, с. 112].

Таксономія Блума виступає основним критерієм дидактичної складності [19]. Кожному рівню відповідає коефіцієнт: запам'ятовування — 0.1, розуміння — 0.25, застосування — 0.45, аналіз — 0.65, оцінювання — 0.85, створення — 1.0. Ця ієрархія відображає зростання когнітивної складності від простого відтворення інформації до творчого синтезу нових знань. Для формування узагальненої початкової оцінки використовується лінійна зважена модель:

$$L_{apriori} = w_1 \cdot k_{type} + w_2 \cdot k_{steps} + w_3 \cdot k_{test} + w_4 \cdot k_{bloom} \quad (2.6)$$

де w_1, w_2, w_3, w_4 — вагові коефіцієнти, які визначають відносний внесок кожного критерію. Отримане значення $L_{apriori} \in [0;1]$ перетворюється у дискретний рівень складності за шкалою 1–10:

$$Level = round(1 + 9 \cdot L_{apriori}) \quad (2.7)$$

Це значення зберігається у профілі складності як початковий рівень, що надалі коригується апостеріорними алгоритмами на основі реальних даних виконання.

Автоматична генерація завдань є одним із ключових функціональних компонентів системи [1, с. 36]. Її призначення полягає у формуванні великої кількості унікальних тестових завдань на основі обмеженої множини параметризованих шаблонів із гарантованим дотриманням заданого рівня складності. Такий підхід дозволяє швидко створювати варіативні тести, уникаючи необхідності ручного створення сотень або тисяч індивідуальних завдань. У системі реалізовано шаблонний підхід, за якого кожне завдання описується у вигляді узагальненого шаблону з параметрами, а конкретні інстанси формуються шляхом автоматичної підстановки значень відповідно до заданого рівня.

Кожен шаблон описується як формалізована структура: текст умови з параметрами, опис параметрів із типом та допустимим діапазоном значень, обмеження для забезпечення коректності, правило обчислення правильної відповіді, тип завдання та спосіб перевірки, зв'язок із профілем складності.

Наприклад, шаблон завдання з лінійних рівнянь може містити параметри для коефіцієнтів та вільного члена, з обмеженнями на їхні значення для уникнення надто простих або надто складних випадків.

Рівень складності визначається не лише типом завдання, а й характером і діапазонами його параметрів. Легкий рівень характеризується невеликими числовими значеннями, простими залежностями, мінімальною кількістю кроків. Наприклад, рівняння 2x + 3 = 7 має легкий рівень складності через малі цілі коефіцієнти та одну операцію. Середній рівень має ширший діапазон чисел, появу дробових значень, 2–3 логічних кроки. Рівняння 3.5x - 2.8 = 4.9 відноситься до середнього рівня через наявність дробових чисел. Високий рівень характеризується складними числовими залежностями, використанням кількох формул, параметрами з кореляцією. Система рівнянь або рівняння з параметрами належать до високого рівня.

Алгоритм генерації конкретного інстансу виконується у Test Generation Service і включає кілька етапів: вибір шаблону із банку за заданим рівнем складності та темою, вибір діапазонів параметрів відповідно до цільового рівня, генерація випадкових значень параметрів із відповідних діапазонів із перевіркою обмежень, обчислення правильної відповіді на основі формули шаблону, формування фінального тексту завдання з підставленими параметрами [1, с. 37]. Кожен етап є критичним для забезпечення якості згенерованого завдання.

Для завдань закритого типу система автоматично формує дистрактори — правдоподібні, але неправильні варіанти відповідей [30, с. 48]. Використовуються методи арифметичних збурень, типових помилок учнів, зміни порядку виконання операцій. Наприклад, якщо правильна відповідь на рівняння становить 5, дистракторами можуть бути 3 (помилка в знаку), 10 (помилка при скороченні), -5 (помилка в знаку результату). Усі варіанти відповідей перевіряються на відсутність збігів між собою та з правильною відповіддю, що гарантує коректність тесту.

Для серверної частини обрано платформу Node.js у поєднанні з фреймворком NestJS та мовою TypeScript [26]. Node.js забезпечує асинхронну, неблокуючу модель обробки запитів, що є важливим для обробки великої кількості паралельних запитів від учнів і вчителів. Асинхронність особливо критична для операцій введення-виведення, таких як звернення до бази даних або зовнішніх сервісів. TypeScript дозволяє запровадити статичну типізацію, що зменшує ймовірність помилок, полегшує супровід коду та сприяє підвищенню надійності. Статична типізація виявляє потенційні помилки на етапі компіляції, а не під час виконання, що особливо важливо для великих проектів.

NestJS обрано як основний засіб розробки з огляду на його модульну архітектуру, чітку структуризацію коду та вбудовану підтримку патернів, орієнтованих на корпоративні застосунки [26]. Фреймворк забезпечує dependency injection, що полегшує тестування та підтримку коду. NestJS дозволяє організувати окремі модулі для аутентифікації, банку завдань, генерації тестів, проведення сесій, аналітики, що добре узгоджується з мікросервісним підходом. Кожен модуль може розроблятися та тестуватися незалежно. NestJS має вбудовані засоби інтеграції з gRPC та Kafka, що спрощує реалізацію міжсервісної взаємодії та не потребує додаткових бібліотек [24; 26].

Для клієнтської частини обрано React у поєднанні з Ant Design [27]. React забезпечує декларативний підхід до побудови користувацького інтерфейсу, що дозволяє ефективно реалізувати динамічні елементи: інтерактивні форми тестування, панелі аналітики, інтерфейси керування банком завдань [27]. Декларативний підхід означає, що розробник описує, яким має бути інтерфейс, а не як його побудувати, що значно спрощує розробку. Ant Design надає набір готових, стилістично узгоджених компонентів, що істотно зменшує витрати часу на розробку інтерфейсу та забезпечує сучасний професійний вигляд. Використання готової бібліотеки компонентів також гарантує консистентність інтерфейсу та дотримання стандартів accessibility.

Для зберігання структурованих даних обрано PostgreSQL [40]. Дана СУБД (Система Управління Базами Даних) вирізняється високою надійністю,

підтримкою транзакцій, розвиненими механізмами цілісності даних та широкими можливостями розширення. Підтримка ACID-властивостей забезпечує коректність даних навіть у випадку збоїв. PostgreSQL добре підходить для зберігання складних структур, включно з використанням полів типу JSONB для параметризованих шаблонів завдань та логів дій. JSONB дозволяє зберігати структуровані документи безпосередньо в реляційній базі даних, що зручно для параметрів, які можуть мати різну структуру.

Для реалізації механізмів авторизації обрано JWT. Застосування JWT-токенів дозволяє реалізувати безстанову авторизацію на рівні API Gateway: клієнт після автентифікації отримує підписаний токен, який додається до кожного запиту в заголовку Authorization. Сервер перевіряє підпис токена та витягує з нього інформацію про користувача і його права доступу, не звертаючись до бази даних на кожному запиті. Це зменшує навантаження на серверну частину та спрощує масштабування, оскільки запити можуть оброблятися будь-яким екземпляром сервера без необхідності спільного сховища сесій.

Серверна частина реалізована на NestJS і організована за модульним принципом. Кожен модуль відповідає за окрему предметну область: автентифікацію, роботу з користувачами, банк завдань, генерацію тестів, проведення тестувань, аналітику. Така організація забезпечує високу зв'язність всередині модуля та слабке зв'язування між модулями. Взаємодія із зовнішніми клієнтами здійснюється через REST API, внутрішня міжсервісна комунікація реалізована на gRPC та Kafka. Використання різних протоколів для різних типів комунікації оптимізує продуктивність системи.

Основу становить набір NestJS-модулів: AuthModule для аутентифікації та авторизації, UsersModule для керування обліковими записами, ItemsModule для банку шаблонів завдань, TestGenerationModule для генерації завдань, TestingModule для сесій тестування, AnalyticsModule для обчислення психометричних показників та формування звітів, IntegrationModule для інтеграції з зовнішніми системами, SharedModule для спільних сервісів. Кожен

модуль має чітко визначений інтерфейс і може бути замінений або розширений без впливу на інші частини системи.

Для клієнтських застосунків реалізовано REST-ендпоїнти, які забезпечують весь необхідний функціонал. Автентифікація: POST /auth/login приймає облікові дані (email та пароль) та повертає JWT-токен у разі успішної перевірки, GET /auth/me повертає інформацію про поточного авторизованого користувача на основі токена. Робота з банком завдань: GET /items для отримання списку шаблонів з можливістю фільтрації та пагінації, POST /items для створення нового шаблону, PUT /items/:id для оновлення існуючого шаблону, DELETE /items/:id для деактивації шаблону.

Генерація та тестування: POST /tests/generate для формування тесту за вказаними параметрами (тема, кількість завдань, рівень складності), POST /tests/sessions для створення нової сесії тестування, POST /tests/sessions/:id/answer для надсилання відповіді на конкретне завдання в рамках сесії, POST /tests/sessions/:id/finish для завершення сесії та отримання результатів. Аналітика: GET /analytics/my для перегляду власної статистики учнем [9, с. 20], GET /analytics/groups/:groupId для перегляду групової статистики вчителем.

Для внутрішньої взаємодії використовуються gRPC-сервіси, визначені у вигляді protobuf-контрактів, та Kafka-топіки для обробки подій. gRPC забезпечує ефективну бінарну серіалізацію даних та підтримку bidirectional streaming, що важливо для високонавантажених операцій. Після завершення тестової сесії TestingModule публікує подію test.session.completed у відповідний Kafka-топік, яку підписується AnalyticsModule для оновлення статистичних показників. Така асинхронна архітектура дозволяє обробляти аналітику без блокування основного процесу тестування.

Механізм JWT-авторизації інтегрований через AuthModule, що містить декілька ключових компонентів. AuthService відповідає за перевірку облікових даних користувача (порівняння хешу пароля) та генерацію підписаного JWT-токена з інформацією про користувача і час дії. JwtStrategy реалізує стратегію

валідації підпису токена при кожному запиті, витягуючи інформацію про користувача з токена. AuthGuard використовується як декоратор для захисту ендпоїнтів та обмеження доступу залежно від ролі користувача, забезпечуючи гранулярний контроль доступу.

Клієнтський інтерфейс реалізовано у вигляді Single Page Application на React із Ant Design. Інтерфейс побудовано за рольовим принципом, що передбачає різні набори екранів і функцій для вчителя, учня та адміністратора. Така сегментація забезпечує зручність використання та безпеку, показуючи кожному користувачеві лише релевантний функціонал. Для навігації використовується React Router, що забезпечує швидкі переходи між екранами без перезавантаження сторінки. Для обміну даними застосовується REST API з передаванням JWT-токена в заголовку Authorization кожного запиту.

Інтерфейс учителя орієнтований на підтримку діяльності з підготовки тестових матеріалів, керування процесом тестування та аналізу результатів. Основними екранами є: екран управління банком завдань з таблицею шаблонів із можливістю фільтрації за темою, рівнем складності, типом завдання та сортування за різними критеріями; конструктор шаблонів у вигляді багатокрокової форми для введення умови з параметрами, визначення діапазонів, встановлення обмежень та формули правильної відповіді; екран запуску тестування для вибору теми, кількості завдань, параметрів генерації та режиму проведення; панель аналітики з використанням інтерактивних таблиць, графіків розподілу результатів та діаграм складності завдань.

Інтерфейс учня орієнтований на максимально просту та інтуїтивну взаємодію в процесі тестування. Основні екрани включають: екран вибору тесту зі списку доступних тестів з інформацією про тему, кількість завдань та очікуваний час виконання; інтерфейс проходження тесту з послідовним відображенням завдань, варіантів відповідей, можливістю позначення питань для перегляду та таймером; екран перегляду результатів із зведеною інформацією про кількість правильних відповідей, відсоток виконання, рекомендації щодо подальшого навчання та можливість перегляду правильних

відповідей. Інтерфейс адміністратора призначений для глобального керування системою: управління користувачами з таблицею для створення, редагування та блокування облікових записів; налаштування системи з параметрами роботи, інтеграції та безпеки. Ant Design відіграє ключову роль у забезпеченні єдиного професійного стилю: компонент Table для списків з сортуванням і пагінацією, Form та Input для введення даних з валідацією, Modal для діалогових вікон, Button та Alert для керування діями та сповіщень, Progress та Statistic для візуалізації результатів і прогресу.

2.3. Експериментальна перевірка та порівняльний аналіз

Експериментальна перевірка працездатності розроблюваної системи має симулятивний характер, що зумовлено відсутністю можливості проведення повномасштабного педагогічного експерименту в часових межах магістерської роботи. Такий підхід застосовується на початкових етапах апробації інтелектуальних освітніх систем і дозволяє перевірити коректність алгоритмів, стабільність роботи та адекватність аналітичних показників.

У межах експерименту сформовано вибірку із 100 віртуальних учнів, рівень підготовки яких задається латентною змінною θ у діапазоні від –3 до +3. Розподіл значень θ є близьким до нормального із математичним сподіванням, близьким до нуля, що відповідає типовій гетерогенній навчальній групі. Окремо задається індивідуальна варіативність поведінкових характеристик: середній час виконання та кількість спроб моделюються як функції від θ із додаванням випадкової похибки.

Для експерименту сформовано банк із 200 тестових завдань, кожне з яких має заданий початковий рівень складності за апріорною шкалою 1–10, параметри складності b та дискримінації a за IRT-моделлю, тип завдання, очікувану кількість кроків розв'язання та орієнтовний середній час виконання. Рівні складності в банку розподілені рівномірно, що дозволяє перевірити коректність генератора для всіх підрівнів.

Симуляція процесу організована так, що кожному віртуальному учневі призначається від 3 до 7 тестових сесій, для кожної з яких система автоматично

генерує тест із 10–20 завдань різних рівнів складності. Ймовірність правильної відповіді на кожне завдання визначається за спрощеною логістичною IRT-моделлю:

$$P(correct) = \frac{1}{1 + e^{-a(\theta - b)}} \quad (2.8)$$

На основі згенерованих ймовірностей випадковим чином визначається факт правильної і неправильної відповіді. Для кожної відповіді додатково моделюються час виконання залежно від рівня складності та θ, а також кількість спроб. Сумарно в ході експерименту накопичується кілька десятків тисяч записів відповідей, що є достатнім для перевірки стабільності аналітичних модулів та оновлення параметрів складності.

Основною метою симулятивного експерименту є перевірка коректності алгоритмів генерації завдань заданого рівня складності, аналіз адекватності механізмів оцінювання складності за результатами тестування, перевірка узгодженості апріорних і апостеріорних оцінок складності та демонстрація можливостей навчальної аналітики. Результати симуляції не претендують на повну педагогічну валідність, оскільки не враховують психологічних, мотиваційних та соціальних чинників реального навчального процесу, проте є достатніми для техніко-алгоритмічної апробації системи.

Для демонстрації можливостей системи розглянуто три типові сценарії використання в освітньому процесі, що відображають різні педагогічні цілі: поточний контроль знань, підсумкове оцінювання та побудову індивідуальних освітніх траєкторій.

Сценарій формувального тестування передбачає оперативний контроль засвоєння навчального матеріалу з наданням учням миттєвого зворотного зв'язку. Вчитель обирає тему заняття, бажаний діапазон рівнів складності, кількість завдань та режим тестування. Модуль генерації відбирає відповідні шаблони, параметризує їх і формує індивідуальні варіанти тестів. Після завершення кожен учень отримує короткий звіт про результат, а вчитель бачить узагальнену статистику по класу. Система забезпечує швидке діагностування рівня засвоєння теми та надає основу для корекції подальшої роботи.

Сценарій підсумкового тестування орієнтований на об'єктивне оцінювання навчальних досягнень за завершеною темою або розділом. Вчитель задає структуру тесту із заданою пропорцією рівнів складності та обмеженням за часом. Система автоматично формує тест, добираючи шаблони з різних піддіапазонів складності. Під час проходження фіксуються не лише правильність відповідей, а й час виконання кожного завдання, кількість спроб, паузи між відповідями. Після завершення система обчислює підсумкові бали, формує рейтинг результатів та передає події до модуля аналітики. Вчитель отримує детальні аналітичні звіти, що відображають розподіли результатів, складність виконання кожного завдання та дискримінаційну здатність.

Сценарій адаптивного режиму реалізує персоналізований підхід шляхом динамічного підбору завдань відповідно до рівня підготовки учня. Вчитель активує режим адаптивного тестування, учень проходить стартовий блок завдань середнього рівня, на основі чого формується первинна оцінка підготовки. Якщо учень успішно справляється із завданнями, система автоматично підвищує рівень складності наступних завдань, у разі труднощів — знижує. У процесі безперервно накопичуються дані для уточнення параметрів складності. Після завершення система формує рекомендації щодо подальшого навчання. Такий підхід дозволяє максимально узгодити складність завдань з актуальним рівнем підготовки кожного учня (Додаток В).

Аналіз симульованих даних проведено за кількома напрямами: порівняння розподілів результатів у традиційному та адаптивному тестуванні, узгодженість автоматично визначених рівнів складності із змодельованими рівнями знань, динаміка успішності слабших учнів та загальні переваги й обмеження підходу.

Аналіз розподілів результатів показує, що в традиційному режимі формується типовий профіль із концентрацією учнів у середньому інтервалі, тоді як крайні значення представлені меншою кількістю спостережень. Це зумовлено тим, що фіксована складність тесту не є оптимальною для всіх учнів. В адаптивному режимі розподіл стає більш диференційованим і рівномірним.

Сильні учні отримують можливість працювати з завданнями підвищеної складності, що дозволяє точніше оцінити їхній реальний рівень, тоді як слабші учні не потрапляють у ситуацію повної перевантаженості. Унаслідок зменшується кількість випадкових відповідей, зростає стабільність результатів, а самі оцінки краще відображають змодельований рівень знань.

Порівняльний аналіз узгодженості показав стійку позитивну кореляцію між змодельованим латентним рівнем підготовки учнів та автоматично визначеними рівнями складності завдань. Учні з вищими значеннями параметра підготовки успішніше виконували завдання підвищеної складності, тоді як учні з нижчими значеннями показували кращі результати на завданнях базового рівня. Це свідчить про коректність механізму апостеріорного оновлення складності на основі поєднання індексу складності, індексу дискримінації та поведінкових показників. На ранніх етапах симуляції спостерігалися окремі розбіжності між апріорною та апостеріорною оцінками, проте зі зростанням кількості відповідей ці розбіжності зменшувалися, а параметри стабілізувалися.

Аналіз динаміки успішності слабших учнів показав, що в традиційному режимі для цієї групи характерні високий відсоток невірних відповідей, значний середній час виконання та підвищена кількість спроб. Після переходу до адаптивного режиму система поступово знижує рівень складності завдань до зони найближчого розвитку, що зменшує кількість помилок, стабілізує час виконання та підвищує відсоток правильних відповідей на наступних сесіях. У симуляції фіксується поступове підтягування слабших учнів до стабільного базового або середнього рівня. Хоча така динаміка не означає автоматичного зростання реальних знань, вона демонструє потенціал системи як інструмента підтримки поступового входження учня в навчальний матеріал без когнітивних перевантажень.

На основі проведеного аналізу виокремлено потенційні переваги системи: підвищення точності оцінювання за рахунок адаптивного механізму, зменшення ефекту завдань не за рівнем, підтримку індивідуальних освітніх траєкторій, самооновлювану модель складності та аналітичну підтримку

рішень вчителя. Водночас виявлено обмеження симулятивного експерименту: відсутність реального мотиваційного та емоційного чинника, спрощені поведінкові моделі, відсутність навчального ефекту між сесіями та залежність точності моделей від обсягу даних на початковому етапі.

Для визначення місця розробленої системи в сучасному освітньому цифровому середовищі проведено порівняльний аналіз з репрезентативними системами: ALEKS, Moodle у поєднанні з плагіном STACK та Wolfram Problem Generator. Порівняння здійснювалося за критеріями підтримки рівнів складності, автоматичної генерації завдань, адаптивного тестування, навчальної аналітики та можливостей інтеграції.

Система ALEKS реалізує підхід на основі теорії просторів знань, відмовляючись від традиційних фіксованих рівнів складності на користь динамічного графа знань. Завдання формуються алгоритмічно, а складність визначається положенням завдання в структурі предметних знань. Перевагою є висока точність виявлення прогалин у знаннях, проте підхід є складним з точки зору впровадження та потребує значного обсягу експертно сформованих моделей предметної області.

Зв'язка Moodle + STACK орієнтована на параметризовану генерацію завдань у математичних та інженерних дисциплінах. STACK дозволяє будувати складні шаблони із використанням комп'ютерної алгебри та перевіряти математичну коректність відповідей. Рівні складності реалізуються опосередковано через складність параметрів, кількість операцій, тип чисел. Адаптивність у класичному розумінні реалізована обмежено та потребує додаткових налаштувань. Аналітика переважно носить статистичний характер без вбудованих психометричних моделей.

Wolfram Problem Generator спеціалізується на процедурній генерації завдань із чітким поділом на рівні складності. Ключовою перевагою є потужний обчислювальний апарат, що дозволяє генерувати та перевіряти завдання високої математичної складності з покроковими розв'язаннями. Водночас система орієнтована переважно на самостійну практику та не містить

розвинених механізмів управління навчальним процесом, аналітики класу чи адаптивного супроводу на рівні освітньої установи.

Розроблена система поєднує в єдиній архітектурі рівні складності, визначені апріорно та апостеріорно, шаблонну автоматичну генерацію завдань, адаптивний механізм добору завдань, модулі навчальної аналітики на основі CTT/IRT та поведінкових показників, а також мікросервісну архітектуру для інтеграції з LMS та іншими освітніми системами. На відміну від вузькоспеціалізованих генераторів або закритих комерційних платформ, система позиціонується як універсальне інтегроване рішення для закладів загальної середньої освіти.

Проведений аналіз дозволяє зробити висновки щодо унікальності підходу, технологічної конкурентоспроможності, педагогічної доцільності та практичного впровадження. Система займає проміжну нішу між вузькоспеціалізованими генераторами та комплексними комерційними платформами, поєднуючи наукову адаптивність, шаблонну параметризовану генерацію, керування рівнями складності та відкриту архітектуру.

До технічних обмежень належить залежність точності психометричних оцінок від обсягу накопичених даних. На початковому етапі експлуатації індекси складності, дискримінації та параметри IRT-моделей можуть бути нестійкими, що знижує точність автоматичної генерації. Обчислювальна складність аналітичних модулів може спричиняти збільшення затримок у побудові звітів при масштабуванні до рівня міста чи області. Складність забезпечення повної відмовостійкості мікросервісної архітектури ускладнює адміністрування та супровід системи.

Організаційні обмеження пов'язані з готовністю педагогічного колективу до використання системи. Повноцінна робота потребує оволодіння базовими цифровими компетентностями, розуміння принципів параметризованих завдань, логіки рівнів складності та адаптивного тестування, що вимагає додаткового навчання персоналу. Неоднорідність матеріально-технічної бази

закладів освіти може ускладнювати масове впровадження через відсутність стабільного доступу до Інтернету або достатньої кількості пристроїв.

Педагогічні обмеження полягають у тому, що адаптивне тестування не є тотожним адаптивному навчанню. Індекси складності та дискримінації залежать від конкретної вибірки учнів і не є абсолютними характеристиками завдань. Реальні освітні дані часто не повністю відповідають припущенням IRT щодо одновимірності латентної ознаки. Автоматично згенеровані завдання можуть поступатися авторським педагогічним задачам у плані методичної цінності та креативності.

Напрями вдосконалення включають перехід до повноцінних багатопараметричних IRT-моделей, застосування байєсівських підходів для оновлення параметрів в умовах обмежених вибірок, впровадження методів машинного навчання для автоматичної класифікації складності, інтеграцію моделей Knowledge Tracing для оцінювання ймовірності засвоєння конкретних елементів знань та розширення функціоналу AIG для генерації семантичних текстових завдань.

Перспективи інтеграції з зовнішніми системами включають інтеграцію з LMS для імпорту списків учнів та експорту результатів, автоматизоване передавання оцінок у електронні журнали, підтримку стандартів SCORM та LTI для універсальної сумісності. У довгостроковій перспективі система може стати частиною єдиної освітньої аналітичної платформи, де результати тестування поєднуються з даними про відвідуваність, виконання домашніх завдань та активність учнів в електронних курсах для побудови комплексних моделей прогнозування успішності.

Висновки до ІІ розділу

У другому розділі здійснено практичну реалізацію програмної системи автоматизованого оцінювання знань учнів із генерацією завдань на основі рівнів складності. Обґрунтовано вибір мікросервісної клієнт–серверної архітектури, що забезпечує масштабованість, надійність та інтеграційну сумісність. Визначено ключові ролі користувачів, функціональні модулі та

структуру даних. Побудована модель даних охоплює сутності користувачів, шаблонів завдань, профілів складності, сесій тестування та результатів.

Розроблено алгоритмічне забезпечення системи: алгоритм апостеріорного оцінювання складності на основі CTT, IRT та поведінкових показників; алгоритм апріорної класифікації за типом завдання, кількістю кроків, лінгвістичною складністю та таксономією Блума; алгоритм шаблонної генерації завдань із керуванням складності через діапазони параметрів. Обґрунтовано технологічну реалізацію на основі стеку Node.js, NestJS, gRPC, Kafka, React, Ant Design, PostgreSQL. Реалізовано JWT-авторизацію, REST API та окремі мікросервіси.

Експериментальна перевірка в умовах симулятивного експерименту показала переваги адаптивного тестування, узгодженість між автоматично визначеною складністю та змодельованими рівнями знань, потенціал системи щодо підтримки слабших учнів через індивідуальні траєкторії. Порівняльний аналіз із ALEKS, Moodle+STACK та Wolfram Problem Generator показав, що розроблена система займає проміжну нішу між вузькоспеціалізованими генераторами та комплексними платформами, поєднуючи автоматичну генерацію, формалізовані рівні складності, адаптивне тестування та навчальну аналітику.

Отримані результати підтверджують, що запропонована система відповідає загальній меті роботи та створює підґрунтя для подальших емпіричних досліджень і практичної апробації в умовах реального освітнього процесу.

ВИСНОВКИ

У магістерській роботі виконано усі поставлені завдання, зокрема здійснено теоретичне обґрунтування та практичну розробку системи автоматизованого оцінювання знань учнів з автоматичною генерацією завдань на основі рівня складності. Виконання дослідження дозволило розв'язати низку важливих теоретичних і практичних завдань щодо модернізації системи педагогічного контролю в умовах цифровізації загальної середньої освіти.

Проведений аналіз сучасного стану системи оцінювання засвідчує наявність суттєвих обмежень традиційних форм контролю: суб'єктивність ручного оцінювання, значна ресурсомісткість, низька діагностична цінність підсумкового бала. Проблемні аспекти тестового оцінювання включають статичність тестів, схильність до натаскування та ризики академічної недоброчесності. Це обґрунтовує необхідність переходу до автоматизованих систем, що поєднують адаптивність, масштабованість та об'єктивність.

Систематизовано психометричні моделі визначення складності. Класична теорія тестування забезпечує прості показники — індекси складності та дискримінації, однак параметри залежать від вибірки учнів. Теорія відповіді на завдання, зокрема модель Раша, забезпечує інваріантність параметрів та дозволяє будувати адаптивні тести. Обґрунтовано комбіноване використання обох підходів: IRT як математичного фундаменту внутрішніх алгоритмів та CTT для зовнішньої звітності. Показано, що поведінкові характеристики — середній час виконання, кількість спроб, структура помилок — надають додаткову діагностичну інформацію та дозволяють побудувати багатовимірний показник фактичної складності.

Розроблено концептуальну модель системи, що інтегрує психометричні показники, поведінкові характеристики та механізми динамічного калібрування складності. Модель передбачає апріорну класифікацію на основі дидактичних критеріїв, апостеріорне уточнення за результатами виконання, автоматичну генерацію унікальних екземплярів завдань та адаптивний добір залежно від рівня підготовки учня. Спроєктовано мікросервісну архітектуру з ключовими

модулями: Auth Service, Item Bank Service, Test Generation Service, Testing Session Service, Analytics Service. Взаємодія реалізована через gRPC для синхронних викликів та Kafka для асинхронного обміну подіями.

Розроблено алгоритмічне забезпечення системи. Алгоритм апостеріорного оцінювання поєднує індекси CTT, параметри IRT та поведінкові показники в інтегральний показник фактичної складності. Алгоритм апріорної класифікації використовує зважену модель на основі типу завдання, кількості кроків, лінгвістичної складності та таксономії Блума. Алгоритм шаблонної генерації забезпечує вибір шаблону, визначення діапазонів параметрів, обчислення правильної відповіді та формування дистракторів.

Реалізовано прототип на основі сучасного технологічного стеку: серверна частина на Node.js та NestJS, між сервісна взаємодія через gRPC та Kafka, клієнтський інтерфейс на React з Ant Design, збереження даних у PostgreSQL. Реалізовано JWT-авторизацію, REST API та окремі інтерфейси для різних ролей користувачів.

Симулятивна експериментальна перевірка на основі 100 віртуальних учнів та 200 завдань підтвердила коректність алгоритмів. Адаптивний режим забезпечує більш диференційований розподіл результатів, точніше оцінює реальний рівень підготовки та зменшує кількість випадкових відповідей. Виявлено стійку кореляцію між змодельованим рівнем підготовки та автоматично визначеними рівнями складності. Порівняльний аналіз з ALEKS, Moodle+STACK та Wolfram Problem Generator показав, що система займає проміжну нішу, поєднуючи наукову адаптивність, параметризовану генерацію, керування складністю та відкриту архітектуру.

Поставлену мету роботи досягнуто: розроблено та науково-методично обґрунтовано програмну систему автоматизованого оцінювання знань учнів з автоматичною генерацією завдань на основі рівня складності, яка забезпечує підвищення об'єктивності, адаптивності та аналітичної обґрунтованості педагогічного контролю.

Наукова новизна результатів полягає в тому, що вперше розроблено комплексну модель системи автоматизованого оцінювання, яка інтегрує психометричні підходи, алгоритми параметризованої генерації, механізми адаптивного добору та засоби навчальної аналітики на основі поведінкових показників; удосконалено методику оцінювання складності тестових завдань шляхом поєднання апріорних дидактичних критеріїв з апостеріорними психометричними показниками; набула подальшого розвитку мікросервісна архітектура освітніх систем.

Практичне значення результатів полягає в тому, що розроблена система може бути впроваджена в закладах загальної середньої освіти для автоматизації процесів оцінювання, зменшення часових витрат педагогів, підвищення рівня академічної доброчесності, реалізації адаптивного підходу та забезпечення педагогів аналітичною інформацією. Результати можуть бути використані вчителями математики, інформатики та інших дисциплін точного циклу.

Перспективи подальших досліджень пов’язані з інтеграцією сучасних методів штучного інтелекту для класифікації складності та генерації завдань, проведенням педагогічного експерименту й оцінюванням впливу системи на навчальні досягнення учнів, а також розробкою методичних рекомендацій для вчителів щодо використання автоматизованого оцінювання в навчальному процес.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматична генерація тестового контенту: методи та алгоритми. *Комп'ютер у школі та сім'ї*. 2023. № 4. С. 34–39.

2. Биков В. Ю. Цифрова трансформація суспільства і розвиток комп'ютерно-технологічної платформи освіти і науки. Інформаційно-цифровий освітній простір України: трансформаційні процеси і перспективи розвитку : *матеріали методологічного семінару НАПН України*. Київ, 2019. С. 20–26.

3. Динамічне оновлення параметрів складності в адаптивних системах. *Матеріали конференції STEM-2024* (м. Тернопіль, 2024 рік). Тернопіль: ТНПУ імені В. Гнатюка, 2024. С. 67–72.

4. Жалдак М. І. Комп'ютер на уроках математики. Київ: НПУ імені М. П. Драгоманова, 2020. 230 с.

5. Жук Ю. О. Засоби навчання в умовах цифровізації загальної середньої освіти. *Проблеми сучасного підручника*. 2022. Вип. 28. С. 45–56.

6. Клочко В. І. Математичне моделювання в педагогічних дослідженнях. Вінниця: ВНТУ, 2021. 186 с.

7. Литвинова С. Г. Методика використання хмарних сервісів у навчальному процесі закладу загальної середньої освіти. Київ: Компринт, 2022. 240 с.

8. Ляшенко О. І. Методологічні засади оцінювання навчальних досягнень учнів. *Педагогіка і психологія*. 2021. № 3. С. 15–24.

9. Морзе Н. В., Барна О. В., Вембер В. П. Формувальне оцінювання: від теорії до практики. *Інформатика та інформаційні технології в закладах освіти*. 2021. № 1. С. 12–25.

10. Москальов І. О., Лисенко Д. П. Застосування методів математичної статистики у психолого-педагогічних дослідженнях: навч. посіб. Київ: НУОУ, 2023. 187 с.

11. Про затвердження Концепції реалізації державної політики у сфері реформування загальної середньої освіти «Нова українська школа» на період

до 2029 року : Розпорядження Кабінету Міністрів України від 14.12.2016 № 988-р.

12. Про затвердження критеріїв оцінювання результатів навчання учнів ЗСО : Наказ Міністерства освіти і науки України від 02.08.2024 № 1093.

13. Про затвердження Стратегії цифровізації освіти і науки України до 2027 року : Розпорядження Кабінету Міністрів України від 19.07.2024 № 1209-р.

14. Про освіту : Закон України від 05.09.2017 № 2145-VIII.

15. Про повну загальну середню освіту : Закон України від 16.01.2020 № 463-IX.

16. Психометрія та якість тестів: посібник для освітян. Київ: Центр тестування МОН України, 2023. 156 с.

17. Ракитіна Н. В. Автоматизація контролю знань в умовах дистанційного навчання. *Вісник Житомирського державного університету*. 2023. № 2. С. 78–85.

18. Співаковський О. В., Петухова Л. Є., Коткова В. В. Побудова індивідуальних освітніх траєкторій студентів у системі керування навчанням / О. В. Співаковський та ін. *Науковий часопис НПУ імені М. П. Драгоманова.* 2020. Вип. 12. С. 34–45.

19. Таксономія Блума: від відтворення до синтезу. На Урок. 2021.

20. Теорія відповіді на завдання (IRT). Вікіпедія.

21. Триус Ю. В. Комп'ютерно-орієнтовані системи навчання математичних дисциплін. Черкаси: Брама-Україна, 2020. 205 с.

22. Attali Y., LaFlair G. T. The Interactive Reading Task: Transformer-Based Automatic Item Generation. *Frontiers in Artificial Intelligence*. 2022. Vol. 5. Article 903077.

23. Baker F. B. The Basics of Item Response Theory. ERIC Clearinghouse on Assessment and Evaluation, 2001. 180 p.

24. Beighton B. A Microservice Architecture with NestJS & gRPC. JavaScript in Plain English. 2023.

25. Bond T. G., Fox C. M. Applying the Rasch Model: Fundamental Measurement in the Human Sciences. 4th ed. Routledge, 2020. 410 p.

26. Documentation: NestJS - A progressive Node.js framework. URL: https://docs.nestjs.com

27. Documentation: React - A JavaScript library for building user interfaces. URL: https://react.dev

28. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2003. 560 p.

29. Gierl M. J., Haladyna T. M. Automatic Item Generation: Theory and Practice. Routledge, 2012. 320 p.

30. Gierl M. J., Lai H. Using Automatic Item Generation to Create Solutions and Rationales for Computerized Formative Testing. *Applied Psychological Measurement*. 2018. Vol. 42, No 1. P. 42–57.

31. Gierl M. J., Lai H., Tanygin V. Advanced Methods in Automatic Item Generation. Routledge, 2021. 298 p.

32. Hambleton R. K., Swaminathan H. Item Response Theory: Principles and Applications. Springer, 1985. 318 p.

33. JSON Web Token (JWT) Authentication. RFC 7519, 2015. URL: https://datatracker.ietf.org/doc/html/rfc7519

34. Kafka Documentation: Apache Kafka. URL: https://kafka.apache.org/documentation/

35. Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, 2017. 562 p.

36. Lang C., Siemens G., Wise A., Gašević D. Handbook of Learning Analytics. 2nd ed. SOLAR, 2022. 520 p.

37. Lord F. M. Applications of Item Response Theory to Practical Testing Problems. Lawrence Erlbaum Associates, 1980. 274 p.

38. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.

39. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'Reilly Media, 2021. 614 p.

40. PostgreSQL 16 Documentation. The PostgreSQL Global Development Group, 2024. URL: https://www.postgresql.org/docs/16/

41. Rasch G. Probabilistic Models for Some Intelligence and Attainment Tests. University of Chicago Press, 1960. 199 p.

42. Tan B., Armoush N., Mazzullo E., Bulut O., Gierl M. A review of automatic item generation techniques leveraging large language models. *International Journal of Assessment Tools in Education.* 2025. Vol. 12, Issue 2. P. 317–340.

ДОДАТКИ

ДОДАТОК А

Модуль апріорного визначення складності

Файл calculate-apriori-level.ts з модуля Item Bank Service. Реалізує алгоритм апріорної класифікації складності на основі типу завдання, кількості кроків, лінгвістичної складності та рівня таксономії Блума .

Шлях: apps/item-bank/src/apriori/calculate-apriori-level.ts

```
export type AprioriTaskType =
  | 'single_choice'
  | 'multiple_choice'
  | 'matching'
  | 'short_answer'
  | 'essay';

export type AprioriBloomLevel =
  | 'Remembering'
  | 'Understanding'
  | 'Application'
  | 'Analysis'
  | 'Evaluation'
  | 'Creation';

export type CalculateAprioriArgs = {
  type: AprioriTaskType;
  stepsCount: number;
  conditionTemplate: string;
  bloomLevel: AprioriBloomLevel;
  sMax?: number;
  wMax?: number;
};

export type CalculateAprioriResult = {
  lApriori: number;
  difficultyLevel: number;
  parts: {
    kType: number;
    kSteps: number;
```

```
    kText: number;
    kBloom: number;
  };
};

export function calculateAprioriLevel(args: CalculateAprioriArgs):
CalculateAprioriResult {
  const sMax = args.sMax ?? 10;
  const wMax = args.wMax ?? 200;

  if (!Number.isFinite(args.stepsCount) || args.stepsCount < 1) {
    throw new Error(`stepsCount must be >= 1 (got
${args.stepsCount})`);
  }
  if (sMax <= 0) throw new Error(`sMax must be > 0 (got ${sMax})`);
  if (wMax <= 0) throw new Error(`wMax must be > 0 (got ${wMax})`);

  const kType = mapTypeToKType(args.type);
  const kSteps = Math.min(args.stepsCount, sMax) / sMax;
  const kText =
Math.min(countWordsWithoutPlaceholders(args.conditionTemplate), wMax) /
wMax;
  const kBloom = mapBloomToKBloom(args.bloomLevel);

  const lApriori = clamp01(0.4 * kType + 0.3 * kSteps + 0.15 * kText +
0.15 * kBloom);
  const difficultyLevel = clampInt(1, 10, Math.round(1 + 9 *
lApriori));

  return {
    lApriori,
    difficultyLevel,
    parts: { kType, kSteps, kText, kBloom },
  };
}

export function countWordsWithoutPlaceholders(conditionTemplate:
string): number {
  const text = (conditionTemplate ?? '').replace(/\{\{.*?\}\}/g, ' ');
```

```
  const words = text.split(/\s+/).filter((w) => w.length > 0);
  return words.length;
}

function mapTypeToKType(type: AprioriTaskType): number {
  switch (type) {
    case 'single_choice':
      return 0.2;
    case 'multiple_choice':
      return 0.35;
    case 'matching':
      return 0.5;
    case 'short_answer':
      return 0.7;
    case 'essay':
      return 0.9;
    default:
      throw new Error(`Unknown task type: ${type}`);
  }
}

function mapBloomToKBloom(level: AprioriBloomLevel): number {
  switch (level) {
    case 'Remembering':
      return 0.1;
    case 'Understanding':
      return 0.25;
    case 'Application':
      return 0.45;
    case 'Analysis':
      return 0.65;
    case 'Evaluation':
      return 0.85;
    case 'Creation':
      return 1.0;
    default:
      throw new Error(`Unknown bloom level: ${level}`);
  }
}
```

```
function clamp01(x: number): number {
  if (x < 0) return 0;
  if (x > 1) return 1;
  return x;
}

function clampInt(min: number, max: number, x: number): number {
  if (x < min) return min;
  if (x > max) return max;
  return x;
}
```

ДОДАТОК Б

Модуль перетворення CTT/IRT параметрів

Файл irt.utils.ts з модуля Analytics Service. Реалізує логіт-трансформацію емпіричного індексу складності p до IRT параметра b та мапінг b до дискретного рівня складності.

Шлях: apps/analytics/src/services/irt.utils.ts

```
export function clamp(n: number, min: number, max: number): number {
 return Math.max(min, Math.min(max, n));
}

/**
 * MVP IRT b estimate.
 *
 * In MVP we map the empirical CTT index p to a Rasch-style b parameter assuming θ≈0 and a≈1:
 *   b ≈ ln((1-p)/p)  (logit transform)
 *
 * This yields b in (-∞,+∞); we clamp to [-3,+3] to match DB constraints.
 */
export function estimateBFromP(p: number): number {
 const eps = 1e-6;
 const pClamped = clamp(p, eps, 1 - eps);
 const b = Math.log((1 - pClamped) / pClamped);
```

```
  return clamp(b, -3, 3);
}

/**
 * Map b ∈ [-3;+3] to difficultyLevel ∈ [1;10].
 */
export function mapBToLevel(b: number): number {
  const normalized = (b + 3) / 6;
  const level = Math.round(1 + 9 * normalized);
  return clamp(level, 1, 10);
}
```

ДОДАТОК В

Клас адаптивного тестування

Файл adaptive.engine.ts з модуля Test Generation Service. Реалізує адаптивний добір завдань за принципом b≈θ, оновлення параметра θ після кожної відповіді та критерії зупинки тестування.

Шлях: apps/test-gen/src/adaptive/adaptive.engine.ts

```
import { Injectable } from '@nestjs/common';

import type { Template } from '../proto/item_bank_v1.types';

export type TerminationDecision =
  | { shouldTerminate: false }
  | { shouldTerminate: true; reason: 'theta_converged' |
'item_limit_reached' };

@Injectable()
export class AdaptiveEngine {
  private readonly learningRate = 0.5;
  private readonly thetaMin = -3;
  private readonly thetaMax = 3;
  private readonly minItemsForConvergence = 5;
  private readonly varianceThreshold = 0.3;
  private readonly itemLimit = 15;
```

```
  mapLevelToB(level: number): number {
    return (level - 5.5) / 2;
  }

  mapThetaToLevel(theta: number): number {
    const clamped = Math.max(this.thetaMin, Math.min(this.thetaMax,
theta));
    const normalized = (clamped + 3) / 6;
    const level = 1 + 9 * normalized;
    return Math.max(1, Math.min(10, Math.round(level)));
  }

  updateTheta(currentTheta: number, itemDifficultyLevel: number,
isCorrect: boolean): number {
    const b = this.mapLevelToB(itemDifficultyLevel);
    const pExpected = 1 / (1 + Math.exp(-(currentTheta - b)));
    const error = (isCorrect ? 1 : 0) - pExpected;
    const delta = this.learningRate * error;
    const next = currentTheta + delta;
    return Math.max(this.thetaMin, Math.min(this.thetaMax, next));
  }

  shouldTerminate(thetaHistory: number[], itemCount: number):
TerminationDecision {
    if (itemCount >= this.itemLimit) {
      return { shouldTerminate: true, reason: 'item_limit_reached' };
    }
    if (!this.checkConvergence(thetaHistory)) {
      return { shouldTerminate: false };
    }
    return { shouldTerminate: true, reason: 'theta_converged' };
  }

  checkConvergence(thetaHistory: number[]): boolean {
    if (thetaHistory.length < this.minItemsForConvergence) return
false;
    const window = thetaHistory.slice(-this.minItemsForConvergence);
    const mean = window.reduce((s, v) => s + v, 0) / window.length;
    const variance = window.reduce((s, v) => s + (v - mean) ** 2, 0) /
```

```
window.length;
    return variance < this.varianceThreshold;
  }

  scoreCandidate(template: Template, theta: number): number {
    const calibratedB = template.difficulty_profile?.irt_b ?? 0;
    const b = calibratedB !== 0 ? calibratedB :
this.mapLevelToB(template.difficulty_profile?.level ?? 5);
    return Math.abs(b - theta);
  }
}
```

ДОДАТОК Г

Модуль статистичних функцій для апостеріорного оцінювання

Фрагмент файлу analytics-aggregates.service.ts з модуля Analytics Service. Реалізує обчислення стандартного відхилення та point-biserial correlation для оцінки дискримінаційної здатності завдань.

Шлях: apps/analytics/src/services/analytics-aggregates.service.ts

```
import { clamp } from './irt.utils';

/**
 * Standard deviation calculation.
 */
function stddev(values: number[]): number {
  if (values.length < 2) return 0;
  const mean = values.reduce((s, v) => s + v, 0) / values.length;
  const varSum = values.reduce((s, v) => s + (v - mean) * (v - mean), 0);
  return Math.sqrt(varSum / (values.length - 1));
}

/**
 * Point-biserial correlation proxy for discrimination.
 *
 * Measures how well an item discriminates between high and low performers.
 *
 * Input:
 * - y: correctness array (0/1) for each student response
 * - score: overall session score array [0;1] for each student
```

```
 *
 * Output:
 * - r_pb ∈ [-1;1]: correlation coefficient
 *   r_pb > 0.3: good discrimination
 *   r_pb ≈ 0: item does not differentiate
 *   r_pb < 0: possible error in item
 */
function pointBiserial(y: number[], score: number[]): number | null {
  if (y.length !== score.length || y.length < 10) return null;

  const p = y.reduce((s, v) => s + v, 0) / y.length;
  const q = 1 - p;

  const sAll = stddev(score);
  if (sAll === 0) return null;

  const correctScores: number[] = [];
  const wrongScores: number[] = [];
  for (let i = 0; i < y.length; i++) {
    (y[i] === 1 ? correctScores : wrongScores).push(score[i]);
  }
  if (correctScores.length === 0 || wrongScores.length === 0) return null;

  const m1 = correctScores.reduce((s, v) => s + v, 0) / correctScores.length;
  const m0 = wrongScores.reduce((s, v) => s + v, 0) / wrongScores.length;

  const r = ((m1 - m0) / sAll) * Math.sqrt(p * q);
  return clamp(r, -1, 1);
}

export { stddev, pointBiserial };
```