МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
Фізико-математичний факультет
Кафедра інформатики та прикладної математики

МЕТОДИКА ВИКОРИСТАННЯ ГЕНЕРАТИВНИХ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ СТВОРЕННЯ АДАПТИВНОГО НАВЧАЛЬНОГО КОНТЕНТУ З ІНФОРМАТИКИ

Кваліфікаційна робота студента групи Ім-24
ступінь вищої освіти «магістр»
спеціальності 014.09 Середня освіта (Інформатика)
Слободянюка Артема Валерійовича

Керівник: доктор педагогічних наук, професор,
старший дослідник
Семеріков Сергій Олексійович

Кривий Ріг – 2025

ЗАПЕВНЕННЯ

Я, Слободянюк Артем Валерійович, розумію і підтримую політику Криворізького державного педагогічного університету з академічної доброчесності. Запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають покликання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Криворізького державного педагогічного університету ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Стрімкий розвиток генеративних мовних моделей (ГММ) відкрив нові можливості для підтримки навчання інформатики та програмування. Доступні через публічні API та вбудовані в популярні сервіси, ГММ можуть пояснювати код і помилки, генерувати завдання й тести, надавати формувальний зворотний зв'язок, виступати у ролі діалогових тьюторів [1; 9; 22; 23]. Для вчителя інформатики це створює потенціал суттєвого зменшення рутинної роботи та розширення можливостей індивідуалізації навчання.

Традиційні засоби навчання інформатики (підручники, статичні збірники задач, однакові для всіх тести) лише частково враховують різний рівень підготовки й темп засвоєння матеріалу. У одному класі одночасно навчаються учні з різними стартовими знаннями, інтересами й типовими помилками, однак навчальний контент зазвичай має фіксовану структуру й складність. У результаті слабші учні не отримують достатньої цілеспрямованої підтримки, а сильніші – належних інтелектуальних викликів.

Дослідження в галузі інтелектуальних тьюторських систем і адаптивного навчання показують, що поєднання автоматизованого аналізу результатів з гнучким добором завдань сприяє індивідуалізації й покращенню навчальних результатів [8; 19; 20]. У роботах [9; 15; 17] продемонстровано, що ГММ можуть реалізовувати дидактичні стратегії *скаффолдингу* (поступового нарощування складності) та *послідовного тьюторства*, надаючи багаторівневі підказки, які допомагають учневі розібратися із суттю проблеми, а не лише отримати готовий розв'язок.

Паралельно розвиваються підходи, що поєднують ГММ із власними навчальними матеріалами (архітектура *retrieval-augmented generation, RAG*) [2; 12]. Це дозволяє моделі посилатися на конкретні конспекти, презентації та збірники задач, знижуючи ризик недостовірних відповідей, типових для генеративних моделей [1]. Інші роботи зосереджені на автоматизованій генерації тестів і аналізі рішень [7; 10; 24], гейміфікації та XR-сценаріях [16], а також підтримці викладача у створенні й курируванні контенту [6]. Бібліометричні дослідження [13; 14] фіксують різке зростання кількості публікацій у цій галузі після 2022 р.

Попри наявність численних окремих розробок, у практиці навчання інформатики бракує відтворюваних і технологічно доступних *методик*, які б пояснювали, як використовувати ГММ для побудови адаптивних завдань і тестів з урахуванням конкретного контексту (школа, університет) та обмежень (час, ресурси, політика захисту даних). Особливої уваги потребує питання того, як інтегрувати такі рішення в наявну хмарну інфраструктуру (Google Sheets, Google Forms, Google Classroom), яка вже використовується багатьма закладами освіти.

У наукових роботах, присвячених застосуванню ГММ в освіті, виокремлюються кілька основних напрямів: діалогові тьютори з багаторівневими підказками [9; 17; 20], системи автоматизованої генерації та оцінювання тестів [1; 10; 24], RAG-системи для навчання програмування [2; 12], гейміфіковані та XR-рішення [16], аналіз готовності викладачів і етичних аспектів [3; 21; 26].

Однак більшість таких рішень:

- або реалізовані у вигляді дослідницьких прототипів, що потребують значних технічних ресурсів;
- або інтегровані в закриті пропрієтарні платформи, малодоступні для адаптації вчителем у конкретній школі;
- або зосереджуються на окремому аспекті (генерація тестів, тьюторство, XR), не пропонуючи комплексної методики побудови адаптивного контенту, прив'язаної до реального навчального курсу.

Це створює розрив між накопиченим дослідницьким досвідом і потребами вчителя інформатики, який працює з конкретним класом й уже використовує Google Forms і Google Sheets як основні інструменти для організації контрольних робіт, опитувань та обліку результатів. Відсутність чітко описаного процесу, який би поєднував можливості ГММ із наявною інфраструктурою, ускладнює впровадження адаптивних підходів на практиці.

Обрана тема є актуальною, оскільки спрямована на розроблення **методики використання генеративних моделей штучного інтелекту для створення адаптивного навчального контенту з інформатики**, яка:

- спирається на аналіз сучасних досліджень (26 відкритих публікацій з наукометричної бази Scopus);
- технологічно реалізується на базі доступної інфраструктури (Google Sheets, Google Forms, Google Apps Script);
- ураховує педагогічні стратегії скаффолдингу, формувального зворотного зв'язку й персоналізації;
- долає ризики неточності, академічної недоброчесності та приватності даних.

Об'єктом дослідження є процес розроблення та використання адаптивного навчального контенту з інформатики.

Предметом дослідження є методика інтеграції генеративних моделей у процес розроблення та використання адаптивного навчального контенту з інформатики, яка поєднує:

- вибір архітектурних рішень (чат, RAG, інтеграція з автотестами, відслідковування знань);
- педагогічне проєктування (багаторівневі підказки, формувальний зворотний зв'язок, елементи гейміфікації);
- процедури забезпечення якості відповідей, академічної доброчесності та приватності даних.

Мета роботи полягає у розробленні методики автоматизованої генерації адаптивних тестів з інформатики із використанням генеративних мовних моделей та її практичній реалізації на базі Google Sheets, Google Forms і Google Apps Script.

Очікувані результати дослідження полягають у тому, що запропонована методика дозволить зменшити трудомісткість підготовки адаптивного тестового контенту для викладача та забезпечить індивідуалізацію навчання шляхом автоматичного врахування слабких тем кожного учня.

Для досягнення поставленої мети необхідно розв'язати такі **завдання**:

1. Здійснити огляд літератури (2023–2025 рр.) щодо застосування ГММ у навчанні інформатики та узагальнити основні напрями досліджень.

2. Проаналізувати архітектурні патерни та педагогічні стратегії використання ГММ в освітніх системах.

3. Визначити функціональні вимоги до системи генерації адаптивних тестів з інформатики.

4. Розробити методику автоматизованої генерації адаптивних тестів із використанням ГММ, інтегровану в інфраструктуру Google Sheets / Google Forms.

5. Реалізувати запропоновану методику у вигляді сценарію Google Apps Script та описати алгоритм її застосування.

Методологічною основою дослідження є компетентнісний, особистісно-орієнтований, соціально-конструктивістський та проєктний підходи. ГММ розглядаються не як самоціль, а як інструмент підтримки індивідуальних траєкторій навчання, формування предметних і надпредметних компетентностей (цифрових, комунікативних, саморегулятивних).

Структура роботи відображає логіку переходу від теоретичного аналізу до практичної реалізації методики:

- у **розділі 1** обґрунтовано актуальність теми, сформульовано проблему, мету, гіпотезу, завдання та теоретико-методологічні засади;

- у **розділі 2** описано методику дослідження: стратегію бібліографічного пошуку, процедуру PRISMA, карту кодування й аналітичну рамку "технології × педагогіка";

- у **розділі 3** подано докладний опис практичної реалізації методики генерації адаптивних тестів з інформатики на базі Google Sheets / Google Forms / Google Apps Script.

Робота містить вступ, 3 розділи, висновки, список із 26 використаних джерел, додатки, 14 рисунків. Загальний обсяг роботи – 67 сторінок.

РОЗДІЛ 1
ПОСТАНОВКА ПРОБЛЕМИ

1.1. Обґрунтування та актуальність теми

Упродовж останніх років спостерігається стрімке зростання інтересу до генеративних мовних моделей (далі – ГММ) як інструменту підтримки навчання інформатики та програмування. Ці моделі стають доступними через публічні API, інтегруються у навчальні середовища та дозволяють автоматизувати низку задач, що раніше вимагали значних зусиль з боку викладача: пояснення коду та помилок, формувальний зворотний зв'язок, генерація тестів і практичних завдань, побудова діалогових тьюторів [1; 22; 23].

У традиційній практиці викладання інформатики навчальний контент переважно має *статичний* характер: підручники, конспекти, збірники задач, тести зі заздалегідь визначеним набором запитань. Такий підхід складно адаптувати до індивідуальних особливостей учнів: темпу засвоєння, початкової підготовки, типових помилок. У результаті сильніші учні не завжди отримують належні виклики, а слабші – достатньої підтримки, зокрема в опрацюванні прогалин.

Роботи [9; 15; 17] демонструють, що інтеграція інтелектуальних тьюторів на основі ГММ із механізмами *адаптивного скафолдингу* (поступового нарощування складності й підтримки) позитивно впливає на результати навчання, особливо у початківців. Студенти отримують багаторівневі підказки, спрямовані на розуміння структури задачі та виправлення власних помилок, а не лише на отримання готового розв'язку. PyTutor [9] є показовим прикладом такої системи для вивчення Python, де ГММ не замінює викладача, а доповнює його, забезпечуючи гнучку підтримку під час розв'язування задач.

Паралельно розвиваються системи, що поєднують ГММ із *генерація з доповненням через пошук* (RAG), тобто використанням локального корпусу навчальних матеріалів для контекстуалізації відповідей [2; 12]. Це дозволяє моделі посилатися на конкретні конспекти, презентації чи збірники задач, знижуючи ризик недостовірних або «відірваних» від курсу пояснень,

що є типовою проблемою генеративних підходів [1]. Наприклад, у [12] описано CourseAssist – тьютора для курсів з інформатики, який комбінує ГММ із матеріалами курсу та дидактичними правилами.

Сучасні огляди підкреслюють, що застосування ГММ в освіті інформатики вже охоплює широкий спектр сценаріїв: автоматизоване оцінювання й аналіз коду [10; 22], генерація та перевірка тестових завдань [1; 7; 10], підтримка викладача у створенні та курирування контенту [6], гейміфіковані та XR-середовища [16], адаптивні середовища вивчення мов програмування [19], а також специфічні курси, наприклад, бази даних [13]. Бібліометричні дослідження [14] ілюструють різке зростання кількості публікацій після 2022 р., що свідчить про формування окремого напряму досліджень.

Ілюстрацію динаміки зростання кількості публікацій, присвячених використанню генеративних моделей в навчанні інформатики, подано на рис. 1.1.

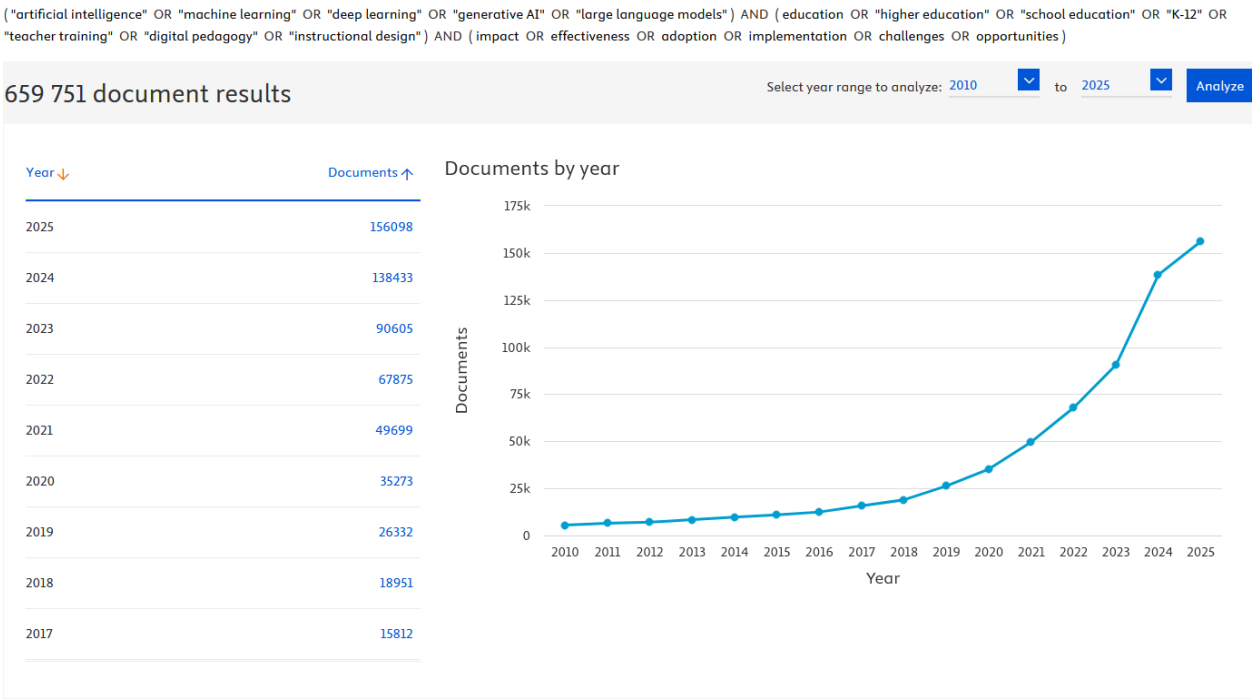


Рис. 1.1. Динаміка кількості публікацій про використання генеративних моделей у навчанні інформатики за даними бази Scopus.

Зростання академічного інтересу (рис. 1.1) корелює зі сплеском загального суспільного інтересу до генеративних моделей і електронних наставників на основі ШІ (AI-тьюторів), що підтверджується даними аналізу

соціальних медіа [11] та пошукових трендів (рис. 1.2). Це створює додатковий тиск на освітні установи, які змушені оперативно формувати політику використання ГММ, інтегрувати їх у навчальні плани й одночасно зберігати академічну доброчесність.

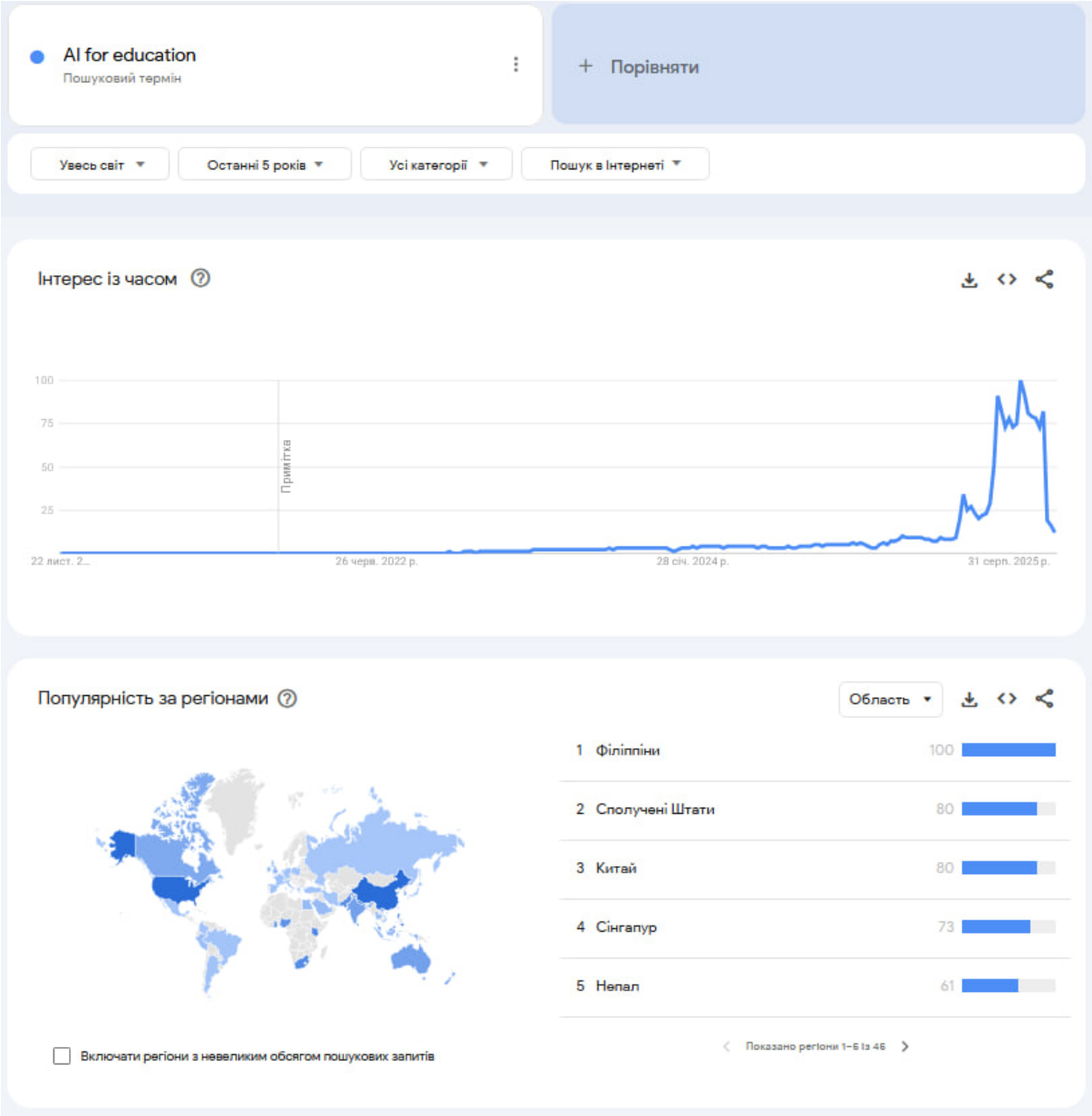


Рис. 1.2. Динаміка пошукового інтересу до генеративних моделей та AI-тьюторів за даними Google Trends.

Узагальнену структуру основних напрямів застосування генеративних мовних моделей у навчанні інформатики подано на рис. 1.3, де відображено найчастіше досліджувані сценарії за результатами оглядів [1; 5; 22; 23].

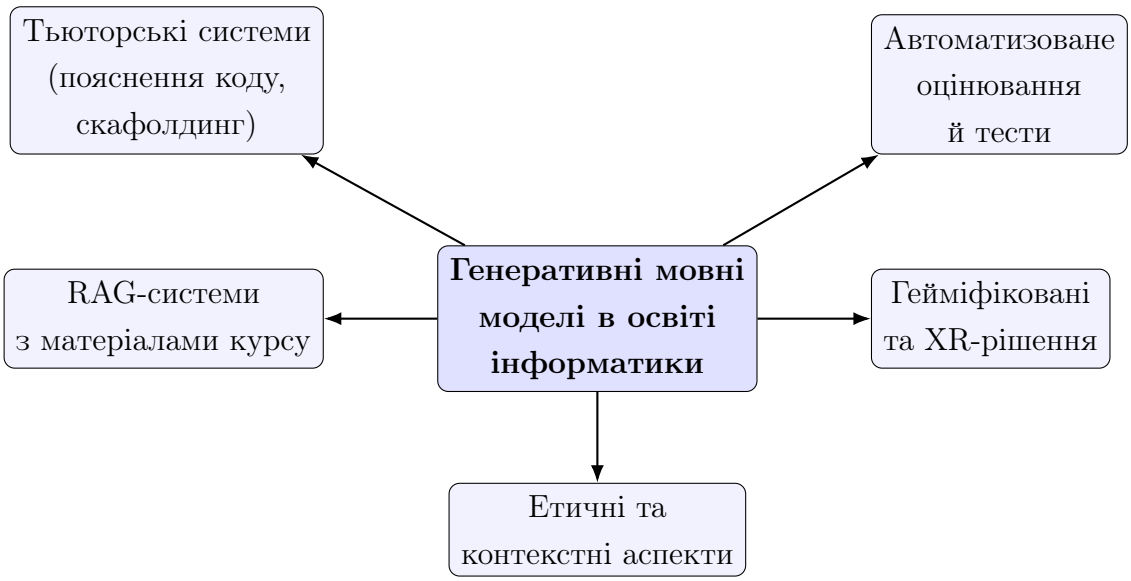


Рис. 1.3. Основні напрями застосування генеративних мовних моделей у навчанні інформатики (узагальнено за [1; 5; 22; 23]).

Попри наявність численних *окремих* рішень (*PyTutor* [9], *CourseAssist* [12], системи генерації тестів [7; 10], XR-тьютор [16], адаптивні платформи для мов програмування [19]), залишається невирішеною проблема розроблення **узагальненої методики** застосування ГММ саме для створення *адаптивного* навчального контенту з інформатики, орієнтованого на:

- різні рівні підготовки (школа, профосвіта, університет);
- реальні обмеження часу й навантаження викладача;
- вимоги до академічної доброчесності та захисту даних учнів [3; 21; 26].

Особливо це відчутно на шкільному та професійному рівнях, де роботи [3; 15; 21; 26] показують потенціал ГММ для розвитку цифрових компетентностей, але одночасно наголошують на ризиках цифрової нерівності, недостатній готовності викладачів та нормативних обмеженнях щодо використання хмарних сервісів. Дослідження [11] доповнює цю картину аналізом постів у соціальних медіа, демонструючи різноманіття очікувань та побоювань щодо генеративних систем у навчанні інформатики.

Таким чином, виникає потреба у методиці, яка не лише демонструє можливості ГММ, а й пропонує *відтворюваний процес* побудови адаптивних завдань і тестів, що інтегрується в існуючу інфраструктуру (напри-

клад, Google Classroom, Google Forms, Google Sheets) та узгоджується з дидактичними цілями курсу інформатики.

1.2. Стан розробки проблеми у вітчизняних та зарубіжних дослідженнях

Аналіз 26 відкритих робіт показує, що дослідники зосереджуються на кількох групах питань.

По-перше, **огляди та бібліометрія**. Роботи [1; 5; 13; 14; 22; 23] систематизують існуючі підходи до використання ГММ у навчанні інформатики, виділяють ключові напрями (тьюторство, оцінювання, генерація матеріалів, етика та політика) й окреслюють головні тренди. Наприклад, у [14] пропонується бібліометричне бачення розвитку теми, а Gaitantzi, Kazanidis [13] фокусується на викладанні баз даних і тому показує, як генеративні моделі «вписуються» у традиційні курси інформатики.

По-друге, **прикладні наставницькі системи**. Статті [2; 9; 12; 17—20] описують реалізацію діалогових наставників (тьюторів) з багаторівневими підказками, послідовним наставництвом та інтеграцією з ментальними картами, відео-лекціями чи завданнями. PyTutor [9] демонструє організацію автоматичного надання підказок у курсі Python; у [17] розглядається послідовне тьюторство для об'єктно-орієнтованого програмування; Lai, Lin [20] аналізує поведінку студентів у системі ITS-CAL; автори [18] поєднуть ментальні карти з чат-ботом на основі ГММ. У [2] описано «AI Tutor» як загальніший каркас інтелектуальної підтримки навчання.

По-третє, **оцінювання та тестування**. Серія робіт [4; 7; 10; 24] фокусується на автоматизованій генерації й аналізі тестів, проєктуванні завдань в умовах доступності ГММ та порівнянні продуктивності різних моделей (наприклад, ChatGPT і Claude) на завданнях з блоковим програмуванням [7]. У [10] аналізується можливість використання ChatGPT для персоналізованої генерації тестів з програмування, а у [4] зіставляється традиційний дизайн оцінювання й нові підходи в епоху генеративного ШІ.

По-четверте, **генерація з доповненням через пошук (RAG) і курс-специфічні системи**. У роботах [2; 6; 12] пропонуються архітектури, що поєднують ГММ із локальними матеріалами курсу та ме-

ханізмами маршрутизації запитів. Chaturvedi [6] демонструє використання LLM для персоналізованої роботи з відео-матеріалами та «кураторства» контенту.

По-п'яте, **етичні, правові та контекстні аспекти**. Роботи [3; 11; 21; 26] акцентують на питаннях доступу, цифрової нерівності, готовності викладачів і політик використання ГММ у формальній освіті. У [21] вивчаються уявлення студентів у країнах Субсахарської Африки, Annuš [3] аналізує впровадження ГММ в IT-орієнтованих закладах професійної освіти, а Wu, Zhang [26] розглядає вплив генеративних систем на інноваційні вміння та цифрову грамотність учнів середньої школи.

Окрему групу становлять роботи, що досліджують **адаптивність і відстеження знань**. У [8] пропонується підхід відстеження знань з урахуванням труднощів (difficulty-aware knowledge tracing) на основі ГММ, де модель враховує складність завдань та індивідуальну траєкторію учня. Стаття [15] аналізує, як підтримка ГММ впливає на навчальну ефективність і формування ключових компетентностей.

При цьому значна частина досліджень демонструє *локальні* результати (пілотні впровадження, окремі курси, специфічні набори інструментів) і лише окремі автори пропонують більш загальні рамки чи рекомендації. Це створює дослідницький розрив між накопиченим досвідом і практичними потребами вчителя інформатики, який потребує чіткої послідовності кроків і зрозумілих сценаріїв застосування ГММ у своєму курсі.

Схематично розрив між існуючими науковими підходами та практичними потребами вчителя інформатики, а також місце запропонованої в цій роботі методики показано на рис. 1.4.

Окрему увагу дослідники приділяють трансформації підходів до проєктування оцінювальних завдань у контексті появи генеративного ШІ. У роботі [4] аналізуються відмінності між традиційними формами оцінювання та дизайном завдань у середовищі, де учні можуть використовувати генеративні моделі, що вимагає переорієнтації з перевірки відтворення знань на оцінювання розуміння, застосування та рефлексії.

На основі цього огляду формулюється дослідницький розрив: відсутність узагальненої, відтворюваної методики, яка підкаже вчителю інформатики, як поєднати конкретні інструменти ГММ (API, чат-

Рис. 1.4. Дослідницький розрив та місце запропонованої методики серед наявних підходів (узагальнено за [1; 21; 22; 26]).

інтерфейси, RAG, автотести) з педагогічними стратегіями (скафолдинг, формувальний зворотний зв'язок, саморегульоване навчання) в умовах реальної школи чи університету.

1.3. Дослідницькі питання

Відповідно до завдань дослідження були визначені *дослідницькі питання* (ДП):

ДП1. Які архітектурні рішення (простий чат, RAG, інтеграція з автотестами) є найбільш доцільними для побудови системи генерації адаптивних тестів з інформатики?

ДП2. Які педагогічні стратегії (скафолдинг, формувальний зворотний зв'язок, персоналізація) доцільно реалізувати при генерації адаптивного тестового контенту?

ДП3. Як організувати профілювання учнів та виділення слабких тем на основі результатів тестування для забезпечення адаптивності?

ДП4. Яким чином інтегрувати ГММ із доступною хмарною інфраструктурою (Google Sheets, Google Forms) для практичного використання вчителем інформатики?

Висновки до 1 розділу

У першому розділі обґрунтовано актуальність дослідження, зумовлену стрімким розвитком генеративних мовних моделей та їхнім потенціалом для автоматизації створення навчального контенту з інформатики. На основі аналізу сучасних публікацій виявлено дослідницький розрив: попри наявність численних окремих рішень (тьюторські системи, генератори тестів, RAG-архітектури), бракує цілісних методик, які б поєднували можливості ГММ із доступною для вчителя інфраструктурою та враховували реальні обмеження освітнього контексту.

Сформульовано дослідницькі питання, спрямовані на визначення доцільних архітектурних рішень для генерації адаптивних тестів (ДП1), педагогічних стратегій персоналізації тестового контенту (ДП2), способів профілювання учнів за результатами тестування (ДП3) та підходів до інтеграції ГММ із хмарною інфраструктурою Google (ДП4). Ці питання визначають логіку подальшого дослідження та структуру практичної реалізації методики.

РОЗДІЛ 2
МЕТОДИКА ДОСЛІДЖЕННЯ

У цьому розділі представлено методичні засади дослідження, спрямованого на розроблення та обґрунтування методики використання генеративних моделей для створення адаптивного навчального контенту з інформатики. Описано логіку та етапи дослідження, джерела інформації й стратегію пошуку, процедуру відбору публікацій (PRISMA), підхід до кодування й аналізу 26 робіт, а також аналітичну рамку, що поєднує технологічний і педагогічний виміри.

2.1. Загальна логіка та етапи дослідження

Відповідно до мети та гіпотези, сформульованих у розділі 1, дослідження реалізується в кілька взаємопов'язаних етапів:

1. **Бібліографічний пошук і первинний скринінг.** Ідентифікація релевантних публікацій із бази даних Scopus та пов'язаних джерел за ключовими словами, що відображають перетин генеративних моделей, адаптивного навчання та інформатики [14].

2. **Відбір публікацій за критеріями PRISMA.** Послідовне усунення дублікатів, робіт без повного тексту, досліджень, що не стосуються навчання інформатики/програмування або не містять застосування ГММ.

3. **Побудова карти кодування.** Розроблення схеми категорій, яка дозволяє систематизувати інформацію про моделі, архітектури, педагогічні стратегії, метрики й ризики в кожній із відібраних робіт.

4. **Аналіз за аналітичною рамкою.** Узагальнення результатів за двома осями: *технологічною* (тип ГММ, архітектурний патерн, функції системи) та *педагогічною* (стратегії навчання, об'єкт підтримки, ефекти).

5. **Синтез вимог до методики.** Формулювання принципів і рекомендацій, які безпосередньо лягли в основу практичної реалізації

(розділ 3).

Такий підхід відповідає сучасним рекомендаціям щодо проведення оглядів у галузі освітніх технологій, де поєднуються кількісні бібліометричні індикатори з якісним тематичним аналізом [1; 22]. У межах цього дослідження огляд має не лише описовий, а й *проєктувальний* характер: завдання полягає у виокремленні таких поєднань «технологія + педагогіка», які можна використати як основу для побудови власної методики.

2.2. Джерела інформації, стратегія пошуку та PRISMA-процедура

Основним джерелом для формування корпусу стали публікації, індексовані в Scopus, доповнені джерелами з інших баз даних і списків літератури ключових оглядів [1; 13; 14; 22]. Пошук здійснювався англійською мовою за комбінаціями ключових слів, що відображають три тематичні виміри: (1) генеративні моделі/LLM, (2) адаптивне навчання й тьюторські системи, (3) освіта в галузі інформатики/програмування:

```
( "generative AI" OR "large language model*" OR "LLM" )
AND
( "adaptive learning" OR "personalized learning"
  OR "intelligent tutoring system*" )
AND
( "informatics" OR "computer science education"
  OR "programming education" OR "computer literacy" )
```

Часові рамки (2023–2025 рр.) визначаються тим, що саме у цей період генеративні моделі зазнали найбільшого розвитку та стали масово доступними для освітніх застосувань [14]. На першому етапі було ідентифіковано **60** публікацій, у назвах, резюме та ключових словах яких фігурували релевантні терміни.

До корпусу включалися роботи, які:

- описують використання генеративних моделей або LLM у контексті навчання інформатики, програмування чи суміжних дисциплін;

- містять або *опис системи* (архітектура, функції), або *емпіричне дослідження* (експеримент, квазіексперимент, опитування, кейс-стаді), або *систематичний/скопінг-огляд*;
- опубліковані у 2023–2025 рр. та мають доступний повний текст англійською.

Не включалися роботи, які:

- стосуються суто технічних аспектів генеративних моделей без освітнього контексту;
- описують навчання поза сферою інформатики й програмування (наприклад, загальна освіта без фокусу на цифрових компетентностях);
- є короткими коментарями, редакційними статтями без опису методики.

Далі було застосовано процедуру PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses). На етапі скринінгу вилучалися:

- дублікати записів із різних джерел;
- роботи без доступного повного тексту;
- публікації, що описують суто технічні аспекти ГММ без освітнього застосування;
- дослідження, не пов'язані з навчанням інформатики/програмування.

У результаті було виключено **34** записи; до повнотекстового аналізу й кодування включено **26** робіт, бібліографічні описи яких сформовано у файл бібліографії. Детальна схема PRISMA-процедури подана на рис. 2.1.

2.3. Інструменти аналізу: VOSviewer, бібліометрія та трендові дані

Для ширшого контексту використано результати бібліометричного аналізу та візуалізації співзустрічі ключових слів, описані в [14]. Зокрема, за допомогою VOSviewer було побудовано карту, на якій:

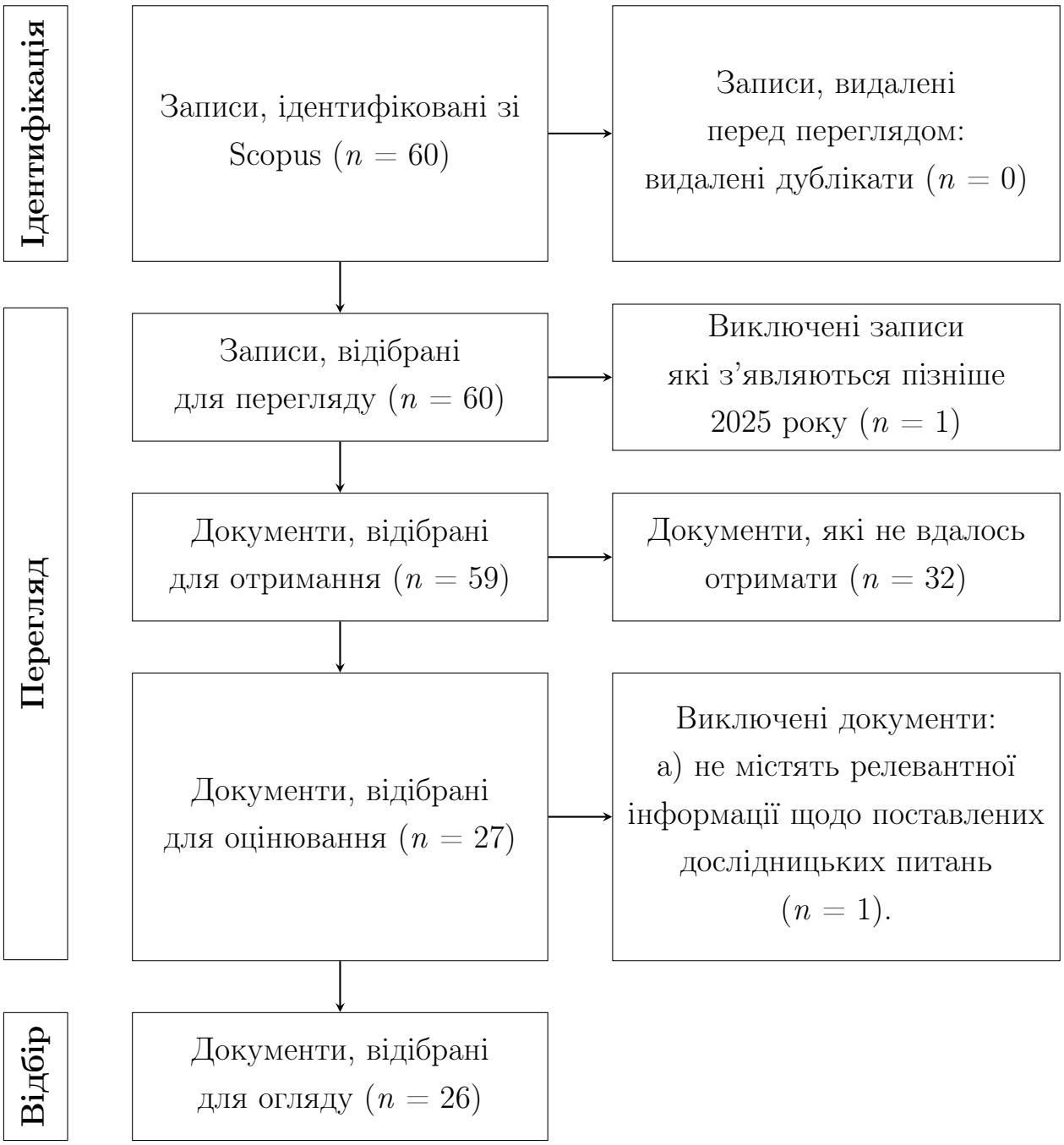


Рис. 2.1. Схема відбору даних для систематичного огляду (згідно методики PRISMA [25]).

- окремі вузли відповідають ключовим словам (*LLM*, *ChatGPT*, *programming education*, *assessment*, *RAG*, *XR*, *database instruction* тощо);
- розмір вузла відображає частоту використання терміна;
- товщина ребер – силу співзустрічі термінів у публікаціях;
- колірні кластери відповідають тематичним групам [5; 13].

На рис. 2.2 наведено карту співзустрічі ключових слів, побудовану в VOSviewer на основі бібліографічного файлу, сформованого за результатами пошуку в Scopus. Вона дозволяє виділити основні тематичні кластери і показує, як часто поєднуються між собою поняття, пов'язані з ГММ та освітою в галузі інформатики.

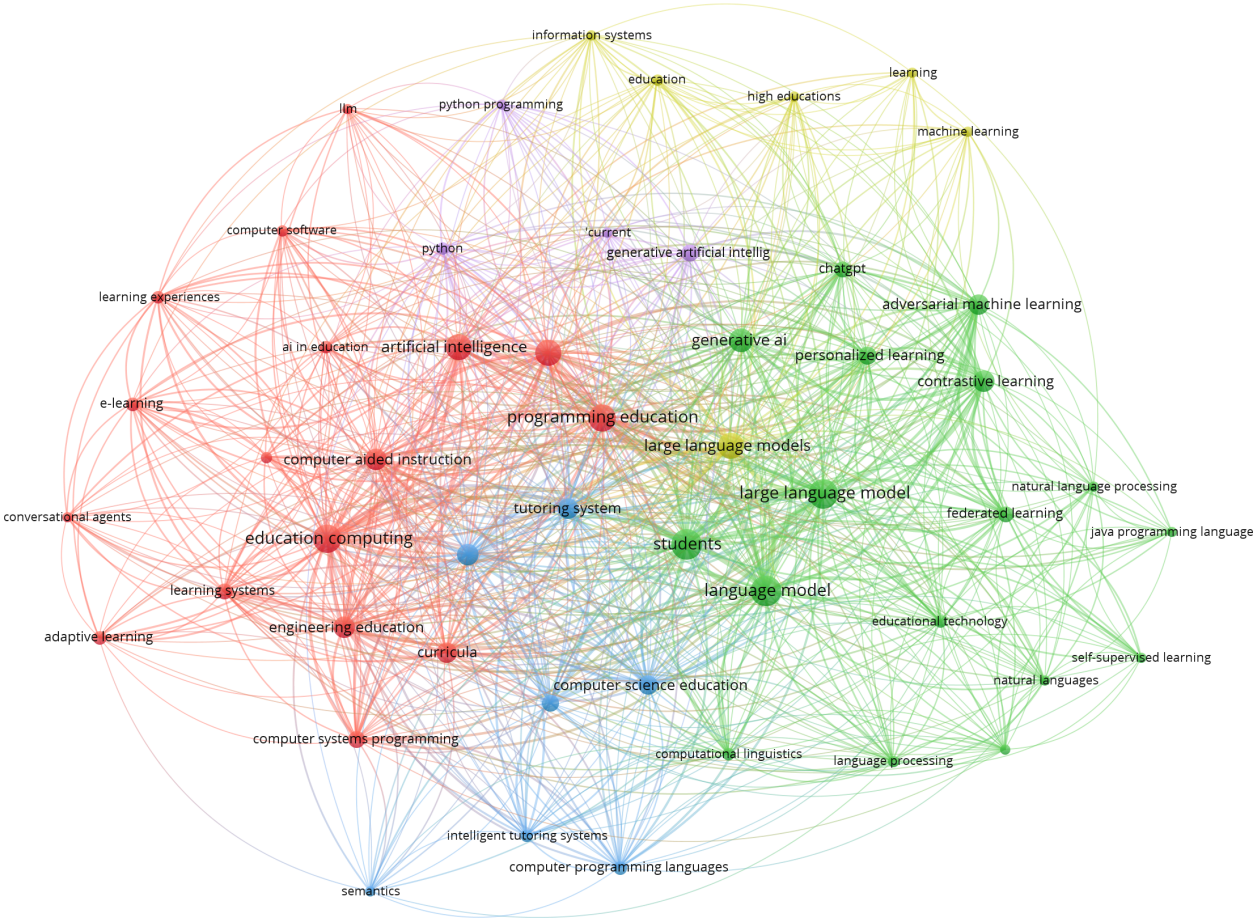


Рис. 2.2. Карта співзустрічі ключових слів, за бібліографічними даними.

Додатково може бути використана *щільнісна* мапа VOSviewer, яка візуалізує, які з тем утворюють найбільш «гарячі» зони наукової активності. На рис. 2.3 показано приклад такої щільнісної мапи: більш насичені ділянки відповідають комбінаціям ключових слів, що фігурують у більшості робіт (наприклад, *ChatGPT* + *programming education*, *assessment*, *tutoring*).

Статистика Scopus (кількість публікацій за роками, розподіл за типами видань) та динаміка пошукових запитів Google Trends щодо запитів "ChatGPT", "programming tutor" тощо використовуються як додаткові індикатори зростання інтересу до теми в науковій спільноті й суспільстві загалом. Відповідні графіки наведено в розділі 1 (рис. 1.1, 1.2) і логічно

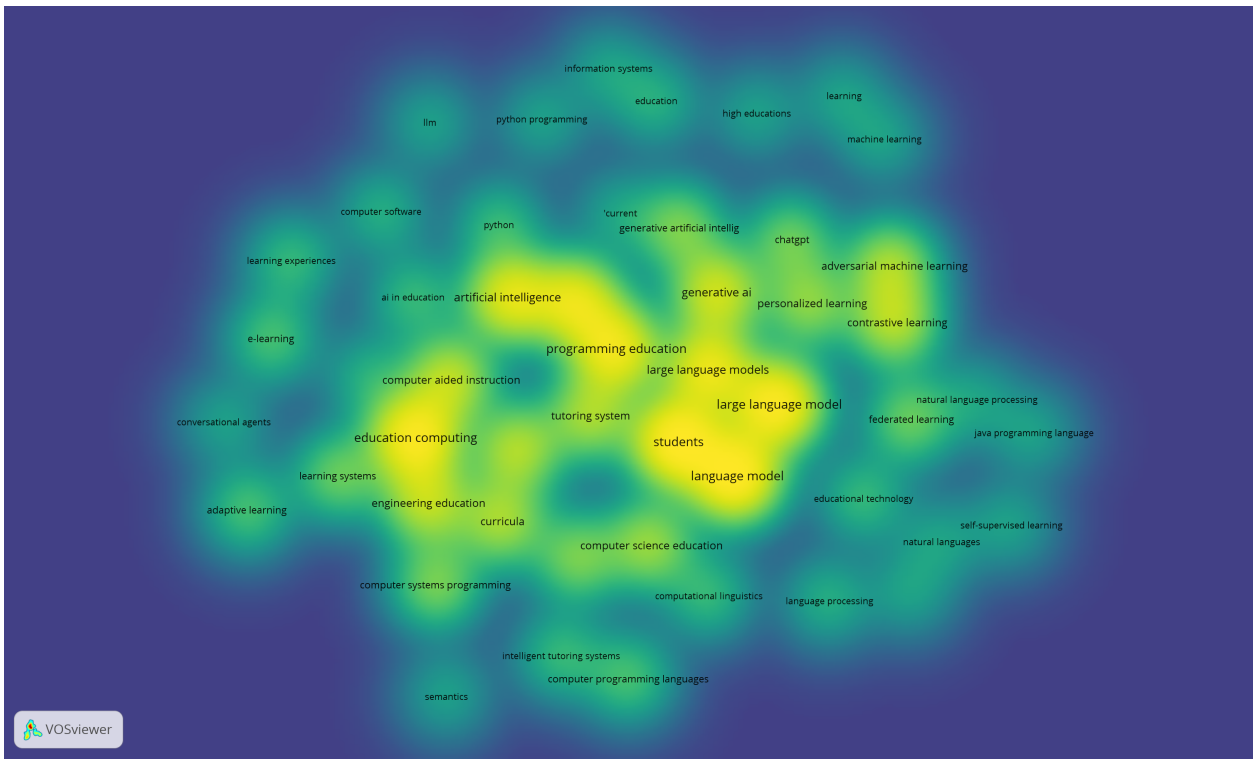


Рис. 2.3. Мапа щільнісна ключових слів, побудована в VOSviewer за бібліографічними даними Scopus (галузь: використання ГММ в освіті інформатики).

доповнюють карти VOSviewer.

2.4. Процедура кодування та карта категорій

Для систематизації відібраних 26 робіт було розроблено карту кодування, що поєднує *технологічні* та *педагогічні* параметри. Фактично кодування виконувалося у вигляді таблиці (Google Sheets), де для кожної публікації фіксувалися такі групи категорій:

- **Метадані:** рік, тип видання (журнал, конференція, огляд), рівень освіти (школа, університет, професвіта, змішане), географічний контекст (за інформацією про вибірку, якщо вона наявна).
- **Модель і її доступність:** тип ГММ (ChatGPT/GPT-4, інші LLM або комбіновані підходи), варіанти розгортання (публічний API, локальна інфраструктура) [11; 23].
- **Архітектурний патерн:** простий чат; RAG з підключенням матеріалів курсу [2; 12]; інтеграція з автотестами та код-ранером

[1; 10; 24]; відстеження засвоєння знань (knowledge tracing) [8]; XR/гейміфікація [16]; робота з відео та мультимедіа [6].

- **Функції системи:** пояснення коду/помилок, багаторівневі підказки, генерація завдань та тестів, формувальний зворотний зв'язок, підтримка викладача в конструюванні матеріалів [1; 6; 9; 19].
- **Педагогічні стратегії:** скафолдинг, оперативний зворотний зв'язок, послідовне тьюторство, гейміфікація/XR, підтримка саморегульованого навчання [9; 15—18].
- **Дизайн дослідження:** експеримент/квазіексперимент, кейс-стаді, опитування, систематичний чи тематичний огляд [1; 3; 5; 11; 13; 21; 22; 26].
- **Метрики й ефекти:** навчальні (до–після, оцінки, частота помилок, час розв'язання), поведінкові (кількість спроб, використання підказок), афективні (мотивація, самоефективність) [9; 15; 16; 20].
- **Ризики та способи їх пом'якшення:** галюцинації, академічна недоброчесність, приватність і безпека даних, цифрова нерівність, вартість і масштабованість [4; 14; 21; 22; 26].

Частина категорій мала *дедуктивний* характер (запозичені з попередніх оглядів [1; 5; 22]), частина формувалася *індуктивно* під час читання статей (наприклад, окреме виділення відстеження засвоєння знань [8] або інтеграції ментальних карт [18]). Це дозволило з одного боку забезпечити порівнюваність із наявними оглядами, а з іншого – не втратити специфічні інноваційні рішення.

2.5. Аналітична рамка: технологічний вимір

Технологічний вимір описує, *як саме* реалізовано використання ГММ у розглянутих роботах. Умовно виокремлено кілька груп:

- **простий чат з продуманими підказками.** У [9; 17] ГММ виступає центральним компонентом діалогового тьютора, де особливу увагу приділено дизайну багаторівневих підказок, а не складній

інфраструктурі. Подібні сценарії фіксуються і в [15; 18], де чат-боти поєднуються з іншими інструментами (ментальні карти, вправи).

- **RAG та курс-специфічний контекст.** У роботах [2; 12] ГММ поєднується з локальним корпусом матеріалів курсу (конспекти, слайди, збірники задач), що дозволяє знижувати частоту недостовірних відповідей і контролювати зміст. Gaitantzi, Kazanidis [13] показує, що такий підхід є перспективним, зокрема для курсів баз даних.

- **Автотести та аналіз коду.** Публікації [1; 7; 10; 24] демонструють використання ГММ для генерації тестів, оцінювання рішень студентів, виправлення коду та надання формувального фідбеку. При цьому [10] більше фокусується на персоналізації, а [7] – на порівнянні різних моделей (Claude і ChatGPT) у завданнях із блоковим програмуванням.

- **Розширені компоненти.** Стаття [8] описує підхід до відстеження засвоєних знань на основі LLM, де система враховує складність завдань і історію відповідей; Gianni, Nikolakis, Antoniadis [16] розглядає поєднання ГММ із XR та гейміфікацією для підвищення мотивації; Chaturvedi [6] демонструє використання LLM для роботи з відео й побудови персоналізованих траєкторій за відеоконтентом.

Такий поділ дозволяє зрозуміти, які технологічні рішення вже апробовані, а які потребують подальших досліджень, і є відправною точкою для проєктування власної архітектури в розділі 3, де робиться ставка на поєднання відносно простих засобів (Google Sheets, Google Forms, Apps Script) з можливостями ГММ, доступних через API.

2.6. Аналітична рамка: педагогічний вимір

Педагогічний вимір описує *цілі* й *дидактичні ролі* ГММ у кожному з розглянутих сценаріїв:

- **Скафолдинг і послідовне тьюторство.** PyTutor [9] та система послідовного тьюторства [17] демонструють ефективність багаторівневих підказок і поетапного ускладнення завдань. В обох випад-

ках ГММ використовується не для видачі готового розв'язку, а для спонукання студента до рефлексії та виправлення власних помилок.

- **Формувальний зворотний зв'язок.** У [1; 24] ГММ використовується для надання зворотного зв'язку у реальному часі, дозволяючи студентам швидко коригувати помилки в коді чи відповідях. Це узгоджується з результатами [15], де фіксується підвищення самоефективності та мотивації студентів.

- **Гейміфікація та XR.** Gianni та співавт. [16] показують, що поєднання ГММ із умовними сценаріями та елементами гейміфікації підвищує мотивацію й знижує кількість повторних спроб, необхідних для досягнення правильного рішення.

- **Саморегульоване навчання.** У [15; 18] ГММ розглядаються як інструменти, що допомагають студентам планувати, моніторити й оцінювати власне навчання, розвиваючи навички саморефлексії й самоорганізації. Застосування ментальних карт у [18] демонструє, як можна структурувати навчальний процес навколо активної побудови знань.

Таке структурування дозволяє в подальшому цілеспрямовано комбінувати технологічні рішення з відповідними педагогічними підходами при розробленні методики адаптивної генерації контенту в розділі 3.

2.7. Оцінка якості досліджень та обмеження

Оцінюючи якість і надійність висновків розглянутих робіт, враховувалися: тип дизайну (експеримент, квазіексперимент, кейс-стаді, огляд), розмір вибірки, наявність контрольної групи, тривалість спостереження, прозорість опису методики та репортинг обмежень.

- Квазіексперименти [9; 15; 16; 20] із контрольними групами та кількісними метриками дають підстави говорити про позитивні ефекти використання ГММ (покращення успішності, мотивації, самоефективності).

- Менші за масштабом пілотні впровадження [2; 12; 17; 18] демонструють потенціал, але потребують підтвердження на ширших вибірках та в інших контекстах (зокрема, шкільному й професійному).

- Опитування [3; 11; 21; 26] надають цінну інформацію про сприйняття ГММ студентами й викладачами, але не завжди дозволяють робити висновки про причинно-наслідкові зв'язки.

- Систематичні й скопінг-огляди [1; 5; 13; 22; 23] забезпечують широку панораму підходів, але включають роботи різної якості та методології, що потрібно враховувати при інтерпретації узагальнених висновків.

Серед обмежень варто відзначити невеликі розміри вибірок, коротку тривалість більшості досліджень, переважання університетського контексту над шкільним, а також недостатню увагу до довгострокових ефектів і питань масштабованості. Ці обмеження враховано при формулюванні власної методики й плануванні її подальшої емпіричної перевірки: запропонований у розділі 3 підхід спочатку орієнтовано на пілотне застосування в окремих групах з подальшим розширенням.

Висновки до 2 розділу

У другому розділі описано методологію дослідження та подано результати аналізу 26 відібраних публікацій. Застосування PRISMA-процедури забезпечило прозорість і відтворюваність відбору джерел, а карта кодування дозволила систематизувати інформацію за технологічним і педагогічним вимірами.

Щодо **ДП1** (архітектурні рішення): виділено чотири основні патерни використання ГММ в освіті інформатики – простий чат із продуманими підказками, RAG-системи з підключенням матеріалів курсу, інтеграція з автотестами та аналізом коду, розширені компоненти (відстеження знань, XR/гейміфікація). Для побудови системи генерації адаптивних тестів найбільш доцільним є поєднання простого чату з інтеграцією в автотести, що забезпечує баланс між функціональністю та технологічною доступністю.

Щодо **ДП2** (педагогічні стратегії): проаналізовані роботи демонструють ефективність скафолдингу, формувального зворотного зв'язку та персоналізації за слабкими темами. Для адаптивного тестування ключовою є стратегія виділення індивідуальних прогалин у знаннях та цілеспрямованого добору завдань для їх опрацювання.

Отримані результати безпосередньо використано для обґрунтування архітектури та функцій практичної реалізації методики в розділі 3.

РОЗДІЛ 3
МЕТОДИКА ГЕНЕРАЦІЇ АДАПТИВНОГО КОНТЕНТУ

У цьому розділі описано практичну методику побудови системи генерації та використання адаптивного тестового контенту з інформатики на основі генеративних моделей. Методика спирається на результати огляду літератури (розділи 1–2) та реалізована у вигляді сценарію Google Apps Script для Google Sheets і Google Forms.

Розроблене рішення поєднує кілька ідей, які активно обговорюються у сучасних роботах:

- автоматизовану генерацію тестових завдань на основі ГММ (LLM), зокрема для програмування та інформатики [1; 10; 19];
- використання хмарних інструментів як простого інтерфейсу для вчителя (електронні таблиці, форми, пошта) [21; 26];
- підхід "teacher-in-the-loop", коли вчитель контролює і коригує роботу генеративних моделей [4; 6; 22];
- адаптивне навчання з урахуванням індивідуальних результатів учнів [9; 13; 21].

У результаті отримано приклад *повністю робочого* сценарію, який вчитель може запустити безпосередньо в Google Sheets: скрипт генерує тести через платформу OpenRouter, будує Google-форму, збирає результати, виділяє слабкі теми й формує адаптивні індивідуальні тести для "роботи над помилками".

3.1. Загальна архітектура рішення

3.1.1. Компоненти системи

З погляду користувача (вчителя) система складається з трьох основних компонентів:

1. **Google Sheets** – основний інтерфейс, де:

- розміщено службові аркуші `Settings`, `Results`, `AdaptiveLinks`;
- працює меню *"Інформатика AI"* для запуску всіх дій;
- зберігається агрегована статистика по учнях і темах.

2. **Google Forms** – автоматично створювані форми-тести (як діагностичні, так і адаптивні).
3. **Платформа OpenRouter** – "двигун" генерації запитань на основі великих мовних моделей (LLM), включно з безкоштовними моделями з позначкою `:free`.

Обмін даними відбувається наступним чином: вчитель задає запит у Google Sheets, скрипт звертається до OpenRouter API, отримує структурований масив об'єктів у форматі JSON з описами завдань, перетворює цей масив на Google Form, а потім аналізує результати виконання цих форм.

3.1.2. Структура коду

Реалізація зосереджена в одному сценарії Google Apps Script.
Основні логічні блоки коду:

- **Глобальні константи і налаштування** – `RESULTS_SHEET_NAME`, `ADAPTIVE_LINKS_SHEET_NAME`, порогові значення для адаптивності, список моделей за замовчуванням тощо.
- **Завантаження налаштувань** з аркуша `Settings` і зі `Script Properties` – функції `loadSettings_()`, `reloadSettings_()`, `getSetting_()`.
- **Формування меню** *"Інформатика AI"* у Google Sheets (функція `onOpen()`), де зібрані всі команди вчителя.
- **Взаємодія з OpenRouter** – модуль, який реалізує схему з кількома ключами й кількома моделями з повторними спробами: `callOpenRouterJson_()`, `makeSingleOpenRouterRequest_()`.
- **Генерація CSV-тестів** на основі майстер-промптів: `generateCsvForTopicViaAi_()`, `generateAdaptiveCsvForStudent_()`.

- **Створення форм** з CSV: `createQuizFormFromCsvText_()`.
- **Обробка результатів** та побудова профілів учнів: `importResultsFromDriveCsv_()`, `exportFromActiveResponseSheet_()`, `getStudentsStats_()`.
- **Адаптивні тести і розсилка**: `generateAdaptiveQuizForStudent_()`, `generateAdaptiveQuizzesForAll_()`, `sendAdaptiveLinksByEmail_()`.

3.2. Користувацький інтерфейс у Google Sheets

3.2.1. Меню "Інформатика AI"

Під час відкриття таблиці викликається функція `onOpen()`, яка створює меню *"Інформатика AI"* (рис. 3.1) з пунктами:

- "Створити тест (простий)..." – запуск діалогу створення нового діагностичного тесту;
- "Імпорт результатів учнів (CSV з Drive)..." – завантаження результатів у `Results`;
- "Експорт із активного аркуша → Results" – перенесення балів з аркуша відповідей форми в агрегований аркуш;
- "Профіль учня (теми та %)..." – перегляд зведеної статистики для конкретного учня за e-mail;
- "Генерувати адаптивний тест для учня..." – створення індивідуального адаптивного тесту;
- "Генерувати адаптивні тести для всіх..." – пакетне створення адаптивних тестів;
- "Розіслати адаптивні тести учням (email)..." – автоматична розсилка посилань на адаптивні тести.

Таким чином скрипт надає вчителю звичний "панельний" інтерфейс поверх складної логіки роботи з LLM та API. Це добре узгоджується з підходами, де вчитель виступає центральною фігурою, а ШІ використовується як інструмент для автоматизації рутинних операцій [4; 6; 22].

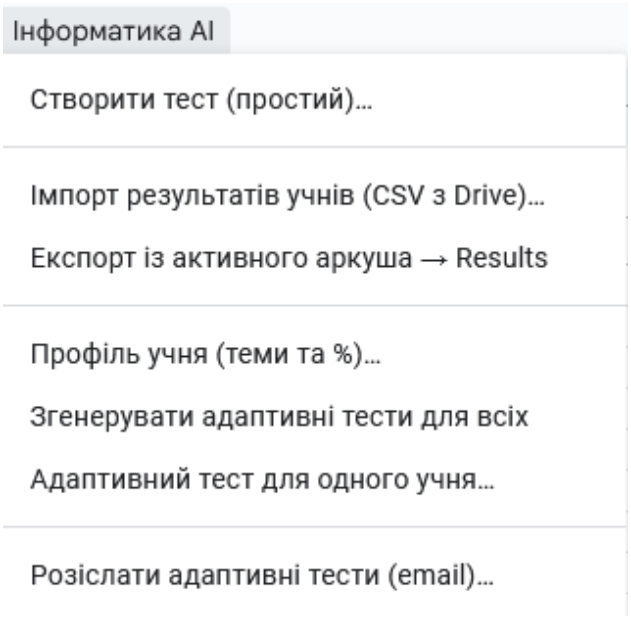


Рис. 3.1. Користувацьке меню скрипта в інтерфейсі Google Sheets.

3.2.2. Аркуші `Settings`, `Results`, `AdaptiveLinks`

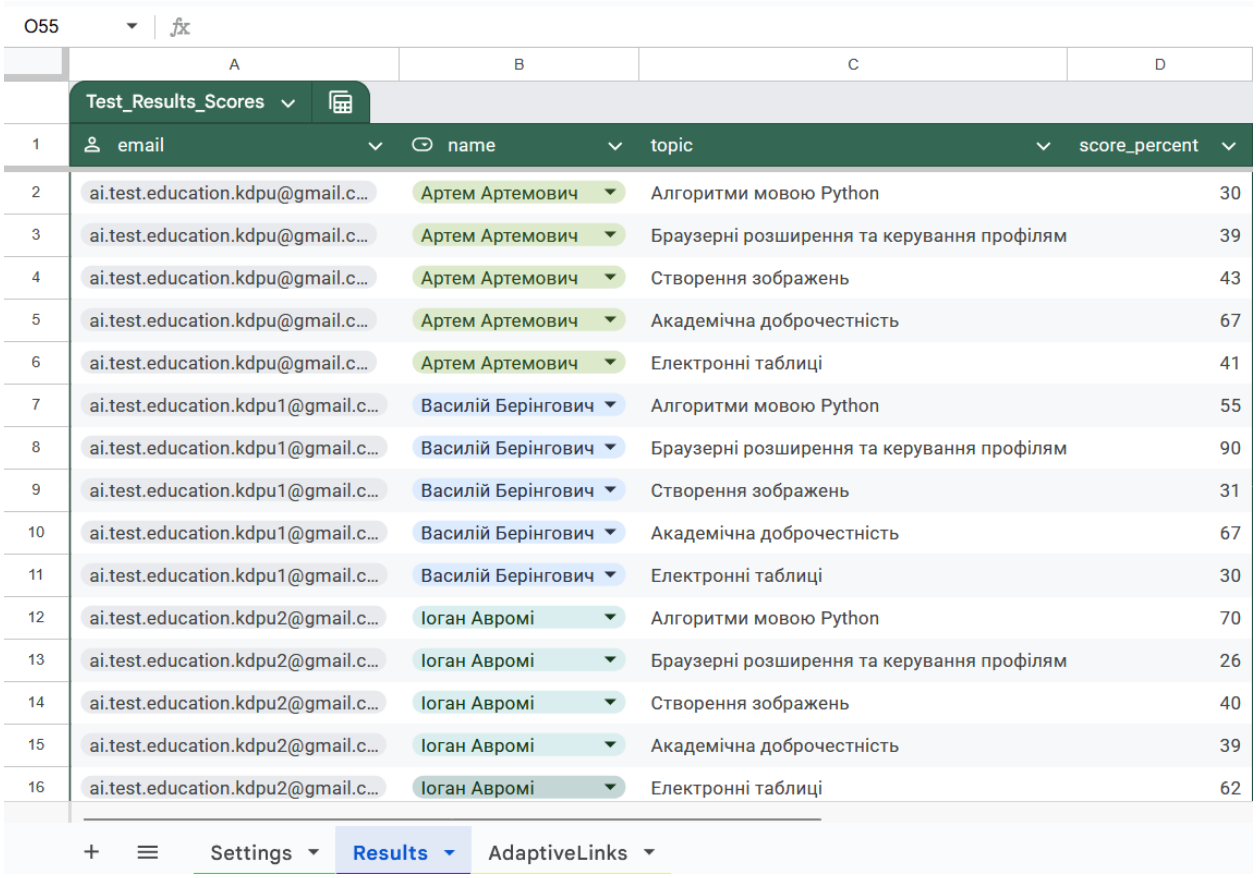
У системі використовуються три ключові службові аркуші:

- `Settings` – зберігає параметри генерації тестів, список тем, налаштування порогів складності, список моделей тощо;
- `Results` – агрегує результати проходження всіх тестів у форматі "учень – тема – відсоток";
- `AdaptiveLinks` – містить переліки адаптивних форм та посилань для учнів.

Аркуш `Settings` дозволяє вчителю безпосередньо керувати поведінкою сценарію: змінювати пороги слабких тем, кількість запитань, моделі за замовчуванням тощо, не торкаючись коду. Це відповідає рекомендаціям щодо створення "налаштовуваних" освітніх інструментів, де технічні деталі приховані від кінцевого користувача [15; 24].

Аркуш `Results` (рис. 3.2) служить центральним сховищем результатів, що спрощує побудову профілів та адаптивних маршрутів: усі подальші розрахунки виконуються саме з цього агрегованого набору даних.

Аркуш `AdaptiveLinks` використовується як таблиця відповідності "учень – адаптивний тест". Тут зберігаються: e-mail учня, ідентифікатор



	A	B	C	D
	Test_Results_Scores			
1	email	name	topic	score_percent
2	ai.test.education.kdpu@gmail.c...	Артем Артемович	Алгоритми мовою Python	30
3	ai.test.education.kdpu@gmail.c...	Артем Артемович	Браузерні розширення та керування профілям	39
4	ai.test.education.kdpu@gmail.c...	Артем Артемович	Створення зображень	43
5	ai.test.education.kdpu@gmail.c...	Артем Артемович	Академічна доброчестність	67
6	ai.test.education.kdpu@gmail.c...	Артем Артемович	Електронні таблиці	41
7	ai.test.education.kdpu1@gmail.c...	Василій Берінгович	Алгоритми мовою Python	55
8	ai.test.education.kdpu1@gmail.c...	Василій Берінгович	Браузерні розширення та керування профілям	90
9	ai.test.education.kdpu1@gmail.c...	Василій Берінгович	Створення зображень	31
10	ai.test.education.kdpu1@gmail.c...	Василій Берінгович	Академічна доброчестність	67
11	ai.test.education.kdpu1@gmail.c...	Василій Берінгович	Електронні таблиці	30
12	ai.test.education.kdpu2@gmail.c...	Іоган Авромі	Алгоритми мовою Python	70
13	ai.test.education.kdpu2@gmail.c...	Іоган Авромі	Браузерні розширення та керування профілям	26
14	ai.test.education.kdpu2@gmail.c...	Іоган Авромі	Створення зображень	40
15	ai.test.education.kdpu2@gmail.c...	Іоган Авромі	Академічна доброчестність	39
16	ai.test.education.kdpu2@gmail.c...	Іоган Авромі	Електронні таблиці	62

Рис. 3.2. Агреговані результати тестування учнів на аркуші `Results`.

форми, посилання на неї, дата створення та статус розсилки.

3.3. Генерація та обробка тестів

3.3.1. Генерація діагностичного тесту (JSON + CSV)

Базовий діагностичний тест генерується у два етапи:

1. ГММ через OpenRouter повертає масив об'єктів у форматі JSON, де кожен об'єкт описує одне запитання (тип, текст, варіанти, правильна відповідь, кількість балів тощо).

2. Цей масив перетворюється на CSV-рядки зі стандартною структурою колонок, після чого з них будується Google Form.

Такий дворівневий підхід (JSON як внутрішнє подання + CSV для сумісності) дозволяє, з одного боку, чітко контролювати структуру завдань на рівні коду, а з іншого – зберегти можливість експорту/імпорту за допомогою звичних інструментів (таблиці, текстові редактори).

3.3.2. Функція `callOpenRouterJson_()`

Функція `callOpenRouterJson_()` відповідає за універсальний виклик OpenRouter у JSON-режимі. Вона:

1) зчитує в налаштуваннях список моделей і ключів API;

2) по черзі намагається викликати кожну пару `(model, apiKey)`;

3) для кожної пари викликає `makeSingleOpenRouterRequest_()`;

4) у разі успіху одразу повертає результат у вигляді розібраного JSON;

5) у разі помилки (ліміти, тайм-аут, формат) переходить до наступної пари.

Сама функція `makeSingleOpenRouterRequest_()` формує HTTP-запит до `https://openrouter.ai/api/v1/chat/completions`, додає:

- заголовок `Authorization: Bearer <API_KEY>`;
- службові заголовки `HTTP-Referer` і `X-Title` для ідентифікації застосунку;
- тіло запиту з полями `model`, `messages`, `temperature`, `max_tokens` тощо.

На виході функція повертає `JSON.parse(response.getContentText())` або кидає помилку, яка обробляється на вищому рівні.

3.3.3. Функція `createQuizViaPrompts_()` і майстер-промпт

Функція `createQuizViaPrompts_()` виступає єдиною точкою входу для створення тесту. Вона поетапно запитує у вчителя:

1) клас (наприклад, 8 чи 9);

2) тему або підтему;

3) бажану кількість запитань;

4) бажаний рівень складності (наприклад, “базовий” чи “просунутий”);

5) кількість відкритих запитань.

На основі цих відповідей формується майстер-промпт українською мовою, у якому модель проситься:

- згенерувати N завдань із чітко визначеними полями (тип, текст, варіанти, правильна відповідь, бали);
- уникати надмірних пояснень поза структурою завдань;
- дотримуватися вікових і програмних вимог до теми.

Результат (JSON) далі перетворюється у CSV-текст з фіксованими колонками, який `createQuizFormFromCsvText_()` використовує для побудови Google Form.

3.3.4. Перетворення JSON на Google Form

В коді ключова функція `createFormFromJson_()` перетворює масив об'єктів JSON, отриманий від моделі, на повноцінну Google Form у режимі "Quiz". Вона:

1) створює нову форму `FormApp.create()`, вмикає режим "Quiz" і додає короткий опис (клас, тема, дата створення);
2) для кожного JSON-об'єкта визначає тип питання:
    - `MC` – питання з вибором однієї правильної відповіді;
    - `CHECKBOX` – питання з кількома правильними відповідями;
    - `SHORT` – коротка відкрита відповідь.
3) створює відповідні елементи форми (multiple choice, checkboxes, short answer), додає варіанти відповідей і позначає правильні відповіді;
4) встановлює кількість балів за кожне питання згідно з полем `points`;
5) додає додаткові поля (пояснення, підказки), якщо вони присутні в JSON.

Досвід з CSV-варіантом (рис. 3.3) показав, що формування форми безпосередньо з JSON більш стабільне: не потрібно піклуватися про роздільники, лапки та переноси рядків. Водночас зберігається можливість експортувати цю ж структуру в CSV для подальшої обробки або резервного копіювання.

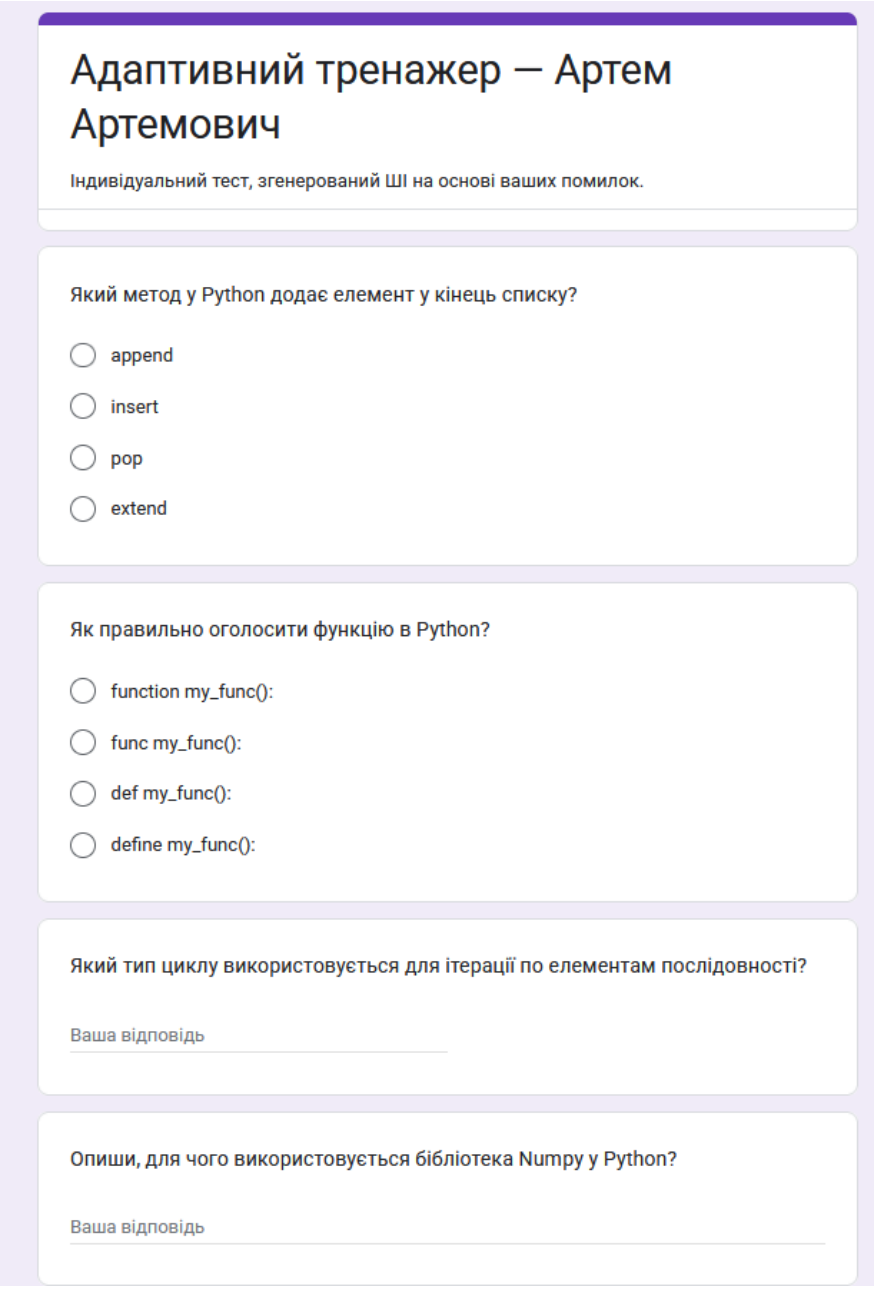


Рис. 3.3. Приклад Google-форми, згенерованої на основі CSV від OpenRouter.

3.4. Обробка результатів і побудова профілів учнів

3.4.1. Структура даних на аркуші `Results`

Аркуш `Results` містить записи у форматі:

```
[email, name, topic, score_percent]
```

де:

- `email` – адреса учня;
- `name` – ім'я учня (за наявності);
- `topic` – тема тесту;
- `score_percent` – відсоток правильних відповідей.

Імпорт у цей формат може відбуватися двома шляхами:

1) безпосередньо з аркуша відповідей Google Forms (через функцію `exportFromActiveResponseSheet_()`);
2) з CSV-файлу, завантаженого на Google Drive (через `importResultsFromDriveCsv_()`).

3.4.2. Обчислення профілів учнів і слабких тем

Функція `getStudentsStats_()` проходить по всіх записах у `Results` і будує структуру даних:

```
{
  email1: {
    name: "Ім'я Учня",
    topics: {
      "Алгоритми": { count: 3, avgPercent: 72 },
      "Мережі":    { count: 2, avgPercent: 65 },
      ...
    },
    globalAvgPercent: 68,
    weakTopics: ["Мережі", "Бази даних"]
```

```
  },
  email2: { ... },
  ...
}
```

Для кожного учня обчислюються:

- середні відсотки по кожній темі;
- загальний середній відсоток;
- список “слабких” тем (де середній відсоток нижчий за `WEAK_TOPIC_THRESHOLD`).

Отримана структура потім використовується в інших функціях: для перегляду профілю одного учня та для генерації адаптивних тестів. Ідея представлення учня через вектор успішності по темах відповідає підходам відстеження засвоєння знань (knowledge tracing) і персоналізованого наставництва [9; 13; 22].

Для ілюстрації можна використати приклад стовпчикової діаграми розподілу середніх результатів по темах (демонстраційні дані подані на рис. 3.4).

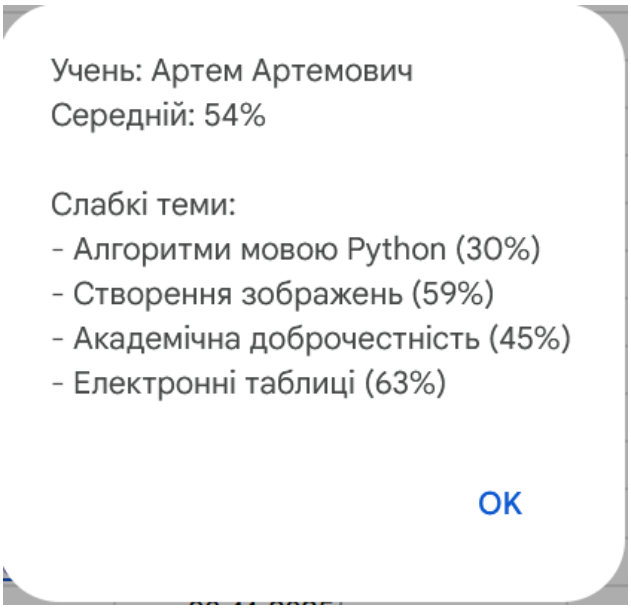


Рис. 3.4. Приклад розподілу середніх результатів учня за темами.

3.5. Генерація адаптивних тестів і розсилка

3.5.1. Адаптивний тест для одного учня (JSON)

Функція `generateAdaptiveQuizForStudent_()` будує окремий майстер-промпт для OpenRouter у JSON-режимі, у якому моделі передається:

- список усіх тем із середніми відсотками успішності цього учня;
- перелік слабких тем (нижче `WEAK_TOPIC_THRESHOLD`);
- загальна бажана кількість запитань (`ADAPTIVE_TOTAL_QUESTIONS`);
- бажане співвідношення типів питань (`MC`, `CHECKBOX`, `SHORT`) залежно від рівня учня.

У самому промпті модель отримує не лише "слабкі" теми, а й короткий контекст про сильні сторони учня, щоб не будувати тест лише на "прогалинах", а забезпечити збалансоване повторення. Це узгоджується з рекомендаціями щодо комбінування формувального й підсумкового оцінювання [13; 18].

Відповідь моделі – масив JSON-об'єктів, структура яких аналогічна до діагностичного тесту, але з акцентом на слабкі теми. Далі цей масив перетворюється на форму так само, як у випадку діагностичного тесту.

3.5.2. Створення адаптивної форми та запис у `AdaptiveLinks`

Функція `generateAdaptiveQuizForStudent_()` об'єднує кілька кроків:

1) знаходить або обчислює профіль учня (якщо він ще не був обчислений);
2) формує майстер-промпт для генерації адаптивного тесту;
3) викликає OpenRouter через `callOpenRouterJson_()`;
4) перетворює отриманий JSON на Google Form через `createFormFromJson_()`;

5) записує у `AdaptiveLinks` службові дані: e-mail, ім'я (якщо є), список слабких тем, посилання на форму, статус розсилки (рис. 3.5).

F8 | 24.11.2025 15:28:13

	A	B	C	D	E	F
1	email	name	weak_topics	form_edit_url	form_url	sent
2	ai.test.education	Артем Артемови	Алгоритми мовс	https://docs.gooc	https://docs.gooc	23.11.2025
3	ai.test.education	Василій Берінго	Алгоритми мовс	https://docs.gooc	https://docs.gooc	23.11.2025
4	ai.test.education	Іоган Авромі	Браузерні розш	https://docs.gooc	https://docs.gooc	23.11.2025
5	ai.test.education	Василій Берінго	Алгоритми мовс	https://docs.gooc	https://docs.gooc	23.11.2025
6	ai.test.education	Іоган Авромі	Браузерні розш	https://docs.gooc	https://docs.gooc	23.11.2025
7	ai.test.education	Василій Берінго	Алгоритми мовс	https://docs.gooc	https://docs.gooc	23.11.2025
8	ai.test.education	Іоган Авромі	Академічна доб	https://docs.gooc	https://docs.gooc	24.11.2025
9	ai.test.education	Артем Артемови	Алгоритми мовс	https://docs.gooc	https://docs.gooc	30.11.2025
10	ai.test.education	Артем Артемови	Алгоритми мовс	https://docs.gooc	https://docs.gooc	30.11.2025
11	ai.test.education	Василій Берінго	Алгоритми мовс	https://docs.gooc	https://docs.gooc	30.11.2025
12	ai.test.education	Іоган Авромі	Браузерні розш	https://docs.gooc	https://docs.gooc	30.11.2025
13	ai.test.education	Артем Артемови	Алгоритми мовс	https://docs.gooc	https://docs.gooc	30.11.2025
14	ai.test.education	Василій Берінго	Алгоритми мовс	https://docs.gooc	https://docs.gooc	30.11.2025
15	ai.test.education	Іоган Авромі	Браузерні розш	https://docs.gooc	https://docs.gooc	30.11.2025
16	ai.test.education	Артем Артемови	Алгоритми мовс	https://docs.gooc	https://docs.gooc	30.11.2025
17	ai.test.education	Василій Берінго	Алгоритми мовс	https://docs.gooc	https://docs.gooc	30.11.2025

Settings | Results | AdaptiveLinks

Рис. 3.5. Аркуш `AdaptiveLinks` із переліком згенерованих адаптивних форм для учнів.

3.5.3. Розсилка адаптивних тестів електронною поштою

Функція `sendAdaptiveLinksByEmail_()` бере всі рядки з аркуша `AdaptiveLinks`, де статус відправки ще не позначено як "відправлено", і розсилає кожному учневі персоналізоване e-mail-повідомлення з посиланням на його адаптивний тест. Після успішної відправки статус у таблиці оновлюється, що запобігає дублюванню листів.

Для більш наочного представлення узагальненого послідовність етапів роботи сценарію `ai_test.gs` використаємо схему з трьома "доріжками" (рис. **??**): вчитель, скрипт і хмарні сервіси (OpenRouter + Google).

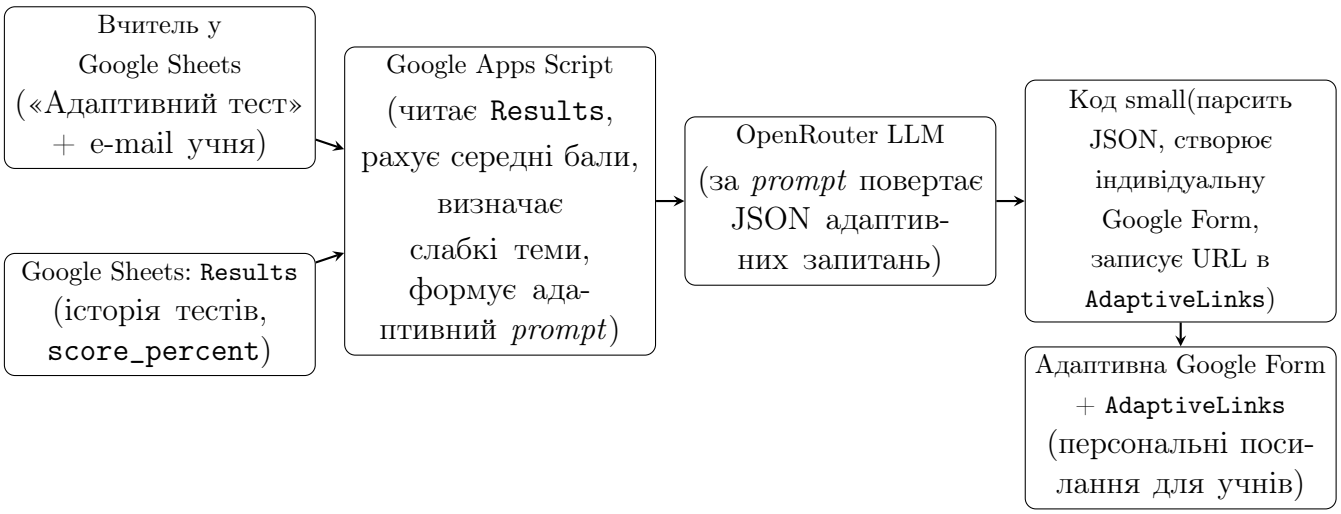


Рис. 3.6. Послідовність етапів генерації адаптивного тесту для окремого учня

3.6. Порівняння CSV та JSON у контексті створення навчального контенту

Під час роботи над кодом було реалізовано дві різні версії формату подання тестового контенту: початкову, орієнтовану на CSV, та оновлену, орієнтовану на структурований JSON. Обидва підходи мають спільну дидактичну мету – автоматизувати створення тестів і підтримати адаптивну логіку, але відрізняються низкою технічних і методичних аспектів.

3.6.1. Еволюція від прототипу до стабільної версії

На ранньому етапі розробки обрано формат CSV як "мінімальний спільний знаменник" між різними інструментами: табличними процесорами, текстовими редакторами, системами імпорту/експорту тестів. Це давало можливість:

- швидко налагоджувати прототип без складної обробки структурованих даних;
- вручну переглядати й редагувати згенеровані завдання в таблиці;
- легко переносити тести між різними системами.

Однак у процесі експериментів з LLM виявилося, що CSV має низку обмежень:

- чутливість до роздільників (кома, крапка з комою), лапок та переносу рядків;
- складність кодування вкладених структур (наприклад, кількох правильних відповідей, пояснень, тегів);
- більший ризик не коректних даних, форматів та в результаті відповідей як показано на рис. 3.7.

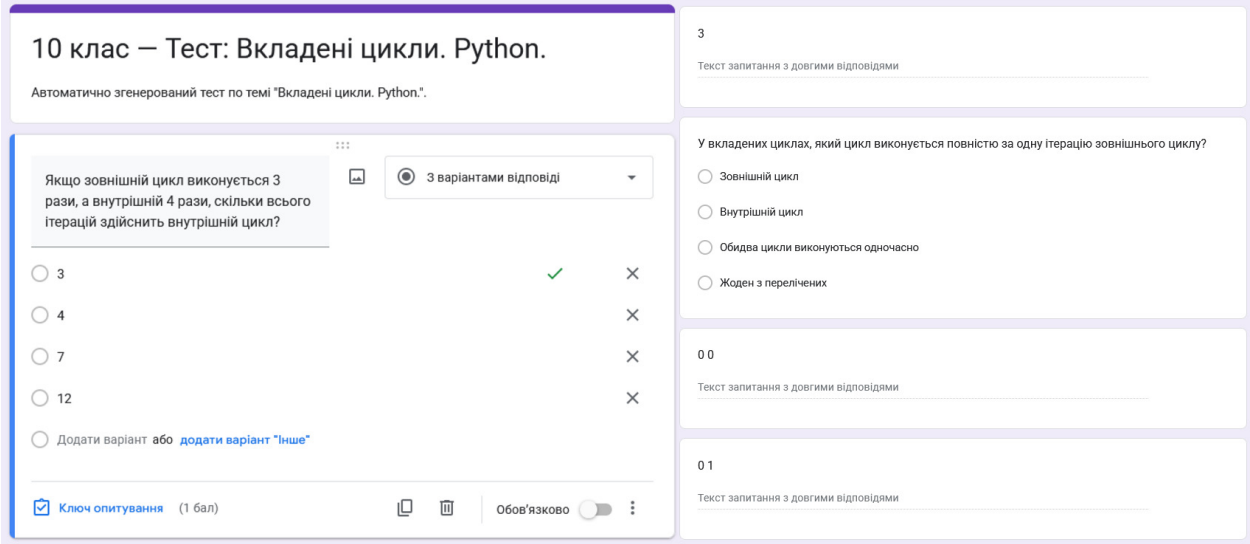


Рис. 3.7. Приклад Google-форми, згенерованої на основі CSV з наявними помилками.

Тому наступним кроком стала JSON-орієнтована версія, де модель одразу повертає масив об'єктів зі строго визначеними полями. CSV при цьому використовується як "зовнішній" формат обміну, а не як основне внутрішнє подання.

3.6.2. Приклад JSON-відповіді від моделі

У JSON-орієнтованій версії майстер-промпт містить чітку інструкцію: повертати масив об'єктів із полями `type`, `question`, `options`, `correct`, `points`, `tags` тощо. Приклад спрощеної відповіді (для однієї теми й кількох питань) може виглядати так:

```
[
  {
```

```
    "type": "MC",
    "question": "Що таке алгоритм?",
    "options": [
      "Точний опис послідовності дій",
      "Опис структури комп'ютера",
      "Графічне зображення даних",
      "Мова програмування"
    ],
    "correct": 0,
    "points": 2,
    "tags": ["Алгоритми", "5-9 класи"]
  },
  {
    "type": "SHORT",
    "question": "Поясніть, що таке змінна у програмуванні.",
    "points": 3,
    "tags": ["Програмування", "7-9 класи"]
  }
]
```

Цей формат значно зручніший для обробки в коді: можна безпосередньо звертатися до полів `question`, `options`, `correct` без попереднього розбору рядка CSV та турботи про лапки й коми.

3.6.3. Технічні переваги JSON над CSV у контексті LLM

Як показав експериментальний етап, перехід до JSON дає кілька практично важливих переваг:

- **стійкість до "балакучості" моделі** – навіть якщо LLM схильна додавати службові коментарі, їх легше відрізати по підпису (`"assistant_comment"` тощо), ніж виправляти зламаний CSV;
- **підтримка вкладених структур** – можна безпосередньо кодувати списки варіантів, кілька правильних відповідей, пояснення, теги, рівні складності;

- **діагностика помилок** – при парсингу JSON у коді легше виявити, яке саме поле відсутнє або має неправильний тип, порівняно з "немовчазними" помилками при роботі з CSV.

3.6.4. Використання CSV для вторинних функцій системи

Важливо підкреслити, що перехід до JSON-формату не означає повної відмови від CSV. У сценарії й надалі використовується CSV:

- для імпорту історичних результатів тестування з інших систем;
- для ручного експорту результатів у `Results` у випадках, коли вчитель працює з уже наявними Google Form;
- простий формат резервного копіювання.

Таким чином, JSON виступає внутрішнім форматом подання тестового контенту (для взаємодії з LLM та побудови форм), а CSV зберігає роль універсального мосту для обміну даними з оточенням. Такий поділ обов'язків між форматами дозволяє одночасно зберегти прозорість системи для вчителя і досягти потрібної стабільності при роботі з генеративними моделями.

Висновки до 3 розділу

У третьому розділі описано практичну реалізацію методики генерації адаптивних тестів з інформатики, що є відповіддю на **ДП3** та **ДП4**.

Щодо **ДП3** (профілювання учнів): реалізовано механізм збору результатів тестування в єдиний аркуш `Results` та автоматичного обчислення індивідуальних профілів. Для кожного учня визначаються середні бали по темах і формується список "слабких" тем на основі налаштованого порогу (`WEAK_TOPIC_THRESHOLD`). Ця інформація використовується для генерації персоналізованих адаптивних тестів, у яких переважають завдання саме зі слабких тем конкретного учня.

Щодо **ДП4** (інтеграція з хмарною інфраструктурою): запропоновано архітектуру, де Google Sheets слугує основним інтерфейсом для вчителя та сховищем даних, Google Forms – платформою для проведення тестів,

а платформа OpenRouter – точкою доступу до ГММ. Реалізація у вигляді сценарію Google Apps Script не потребує додаткової інфраструктури й може бути розгорнута безпосередньо в робочому середовищі вчителя.

Обґрунтовано вибір JSON як внутрішнього формату подання тестового контенту, що забезпечує кращу структурованість даних порівняно з CSV та спрощує обробку відповідей моделі. Система реалізує повний цикл: від генерації діагностичних тестів через профілювання учнів до автоматичного створення та розсилки індивідуальних адаптивних завдань.

ВИСНОВКИ

У процесі дослідження розв'язано поставлені завдання та отримано такі результати:

1. **Здійснено огляд літератури** (2023–2025 рр.) щодо застосування генеративних мовних моделей у навчанні інформатики. На основі PRISMA-процедури відібрано та проаналізовано 26 публікацій із бази Scopus. Виявлено основні напрями досліджень: діалогові тьютори з багаторівневими підказками, системи автоматизованої генерації тестів, RAG-архітектури для навчання програмування, гейміфіковані та XR-рішення. Показано, що попри наявність численних окремих розробок, бракує цілісних методик інтеграції ГММ у доступну для вчителя інфраструктуру.

2. **Проаналізовано архітектурні патерни та педагогічні стратегії** використання ГММ в освітніх системах. Виділено чотири основні архітектурні рішення: простий чат із продуманими підказками, RAG-системи з матеріалами курсу, інтеграція з автотестами та аналізом коду, розширені компоненти (відстеження знань, XR). У педагогічному вимірі узагальнено роль скафолдингу, формувального зворотного зв'язку, персоналізації за слабкими темами та підтримки саморегульованого навчання. Для генерації адаптивних тестів найдоцільнішим визначено поєднання простого чату з інтеграцією в автотести.

3. **Визначено функціональні вимоги** до системи генерації адаптивних тестів з інформатики:

   - автоматизована генерація тестових завдань різних типів (множинний вибір, множинна відповідь, коротка відповідь) на основі заданої теми та рівня складності;
   - збір результатів тестування та побудова індивідуальних профілів учнів із виділенням слабких тем;
   - генерація персоналізованих адаптивних тестів, орієнтованих на опрацювання індивідуальних прогалин;

- інтеграція з наявною хмарною інфраструктурою без потреби у спеціальних технічних ресурсах;
- збереження центральної ролі вчителя у контролі за якістю контенту та налаштуванні параметрів системи.

4. **Розроблено методику автоматизованої генерації адаптивних тестів** із використанням ГММ, інтегровану в інфраструктуру Google Sheets / Google Forms. Методика охоплює чотири етапи: (I) генерація діагностичних тестів через ГММ у форматі JSON із перетворенням у Google Forms; (II) збір результатів та обчислення індивідуальних профілів учнів із виділенням слабких тем; (III) генерація персоналізованих адаптивних тестів на основі профілю кожного учня; (IV) автоматична розсилка посилань на адаптивні тести електронною поштою.

5. **Реалізовано запропоновану методику** у вигляді сценарію Google Apps Script. Архітектура системи передбачає використання Google Sheets як інтерфейсу для вчителя та сховища даних, Google Forms – для проведення тестів, платформи OpenRouter – для доступу до ГММ. Реалізовано механізм стійкості до збоїв із підтримкою кількох API-ключів і моделей. Обґрунтовано вибір JSON як внутрішнього формату подання тестового контенту, що забезпечує кращу структурованість даних та спрощує обробку відповідей моделі порівняно з CSV.

Теоретичне значення роботи полягає в систематизації архітектурних і педагогічних патернів використання ГММ у навчанні інформатики та формулюванні функціональних вимог до системи генерації адаптивних тестів.

Практичне значення полягає у розробленні конкретного рішення, яке може бути впроваджене у школі чи закладі вищої освіти з використанням безкоштовних інструментів Google без потреби у спеціальній інфраструктурі. Запропоновану методику можна адаптувати до різних тем курсу інформатики завдяки гнучким налаштуванням.

Обмеження та перспективи подальших досліджень. Робота має проєктувальний характер і не включає експериментальної перевірки

ефективності методики на реальних вибірках учнів. Перспективними напрямами є:

- проведення квазіекспериментів для оцінки впливу адаптивних тестів на навчальні результати та мотивацію учнів;
- розширення методики на інші формати контенту (практичні завдання з програмування, пояснювальні матеріали);
- інтеграція з RAG-підходами для прив'язки завдань до конкретних матеріалів курсу;
- дослідження питань академічної доброчесності та захисту даних учнів при використанні ГММ в освітньому процесі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Systematic Literature Review of the Practical Applications of Artificial Intelligence-Generated Content (AIGC) Using OpenAI ChatGPT, Copilot, and Codex in Programming Education / C.-I. Chang [та ін.] // Proceeding of the 2024 8th International Conference on Education and E-Learning. — New York, NY, USA : Association for Computing Machinery, 2025. — С. 13—19. — (ICEEL '24). — ISBN 9798400717413. — DOI: 10.1145/3719487.3719519.

2. AI Tutor: Transforming Education with Intelligent Learning / F. Keshtkar [та ін.] // Proceedings of the International Florida Artificial Intelligence Research Society Conference, FLAIRS. Т. 38. — 2025. — DOI: 10.32473/flairs.38.1.138666.

3. *Annuš N.* Investigation of Generative AI Adoption in IT-Focused Vocational Secondary School Programming Education // Education Sciences. — 2025. — Т. 15, № 9. — С. 1152. — DOI: 10.3390/educsci15091152.

4. Assessment Design Before and After the Emergence of Generative AI / Q. Ngoc Tran [та ін.] // 17th WCEAM Proceedings / за ред. G. Abdul-Nour [та ін.]. — Cham : Springer Nature Switzerland, 2024. — С. 145—152. — (Lecture Notes in Mechanical Engineering). — DOI: 10.1007/978-3-031-59042-9_12.

5. *Barzanji C., Loitsch C.* Exploring conversational agents for novice programmers: a scoping review // Discover Artificial Intelligence. — 2025. — Т. 5, № 1. — С. 271. — DOI: 10.1007/s44163-025-00521-4.

6. *Chaturvedi N.* LLMs and NLP for Generalized Learning in AI-Enhanced Educational Videos and Powering Curated Videos with Generative Intelligence // Proceedings of the 1st Workshop on NLP for Science (NLP4Science) / за ред. L. Peled-Cohen [та ін.]. — Miami, FL, USA : Association for Computational Linguistics, 11.2024. — С. 148—154. — DOI: 10.18653/v1/2024.nlp4science-1.12.

7. Comparison of Claude (Sonnet and Opus) and ChatGPT (GPT-4, GPT-4o, GPT-o1) in Analyzing Educational Image-Based Questions from Block-Based Programming Assessments / W. C. Choi [та ін.] // 2025

13th International Conference on Information and Education Technology (ICIET). — 2025. — С. 55—60. — DOI: 10.1109/ICIET66371.2025.11046263.

8. Difficulty aware programming knowledge tracing via large language models / L. Yang [та ін.] // Scientific Reports. — 2025. — Т. 15, № 1. — С. 11475. — DOI: 10.1038/s41598-025-96540-3.

9. Enhancing python learning with PyTutor: Efficacy of a ChatGPT-Based intelligent tutoring system in programming education / A. C. Yang [та ін.] // Computers and Education: Artificial Intelligence. — 2024. — Т. 7. — С. 100309. — DOI: 10.1016/j.caeai.2024.100309.

10. Evaluating ChatGPT-driven Automated Test Generation for Personalized Programming Education / C. Troussas [та ін.] // 2024 2nd International Conference on Foundation and Large Language Models (FLLM). — 2024. — С. 194—200. — DOI: 10.1109/FLLM63129.2024.10852510.

11. Explore Public's Perspectives on Generative AI in Computer Science (CS) Education: A Social Media Data Analysis / S. Oh [та ін.] // Proceedings - Frontiers in Education Conference, FIE. — 2024. — DOI: 10.1109/FIE61694.2024.10893102.

12. *Feng T.*, *Liu S.*, *Ghosal D.* CourseAssist: Pedagogically Appropriate AI Tutor for Computer Science Education // Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 2. — Virtual Event, NC, USA : Association for Computing Machinery, 2024. — С. 310—311. — (SIGCSE Virtual 2024). — ISBN 9798400706042. — DOI: 10.1145/3649409.3691094. — URL: https://doi.org/10.1145/3649409.3691094.

13. *Gaitantzi A.*, *Kazanidis I. K.* The Role of Artificial Intelligence in Computer Science Education: A Systematic Review with a Focus on Database Instruction // Applied Sciences. — 2025. — Т. 15, № 7. — С. 3960. — DOI: 10.3390/app15073960.

14. Generative AI in Education: Perspectives Through an Academic Lens / I. Întorsureanu [та ін.] // Electronics. — 2025. — Т. 14, № 5. — С. 1053. — DOI: 10.3390/electronics14051053.

15. Generative Artificial Intelligence Supported Programming Learning: Learning Effectiveness and Core Competence / H. Li [та ін.] // SAGE Open. — 2025. — Т. 15, № 3. — DOI: 10.1177/21582440251377986.

16. *Gianni A. M.*, *Nikolakis N.*, *Antoniadis N. A.* An LLM based learning framework for adaptive feedback mechanisms in gamified XR // Computers and Education: X Reality. — 2025. — Т. 7. — С. 100116. — DOI: 10.1016/j.cexr.2025.100116.

17. Guided Learning with AI: A Didactic Strategy Using Sequential Tutoring via ChatGPT for Object-Oriented Programming / S. Gomez-Jaramillo [та ін.] // International Journal of Learning, Teaching and Educational Research. — 2025. — Т. 24, № 7. — С. 717—736. — DOI: 10.26803/ijlter.24.7.35.

18. Improving students' programming performance: an integrated mind mapping and generative AI chatbot learning approach / X. Ye [та ін.] // Humanities and Social Sciences Communications. — 2025. — Т. 12, № 1. — С. 558. — DOI: 10.1057/s41599-025-04846-4.

19. *Kwak M.*, *Jenkins J.*, *Kim J.* Adaptive programming language learning system based on generative AI // Issues in Information Systems. — 2023. — Т. 24, № 3. — С. 222—231. — DOI: 10.48009/3_iis_2023_119.

20. *Lai C.*, *Lin C.* Analysis of Learning Behaviors and Outcomes for Students with Different Knowledge Levels: A Case Study of Intelligent Tutoring System for Coding and Learning (ITS-CAL) // Applied Sciences. — 2025. — Т. 15, № 4. — С. 1922. — DOI: 10.3390/app15041922.

21. *Oyelere S. S.*, *Aruleba K. D.* A comparative study of student perceptions on generative AI in programming education across Sub-Saharan Africa // Computers and Education Open. — 2025. — Т. 8. — С. 100245. — DOI: 10.1016/j.caeo.2025.100245.

22. *Pereira A. F.*, *Ferreira Mello R.* A Systematic Literature Review on Large Language Models Applications in Computer Programming Teaching Evaluation Process // IEEE Access. — 2025. — Т. 13. — С. 113449—113460. — DOI: 10.1109/ACCESS.2025.3584060.

23. *Senanayake S.*, *Karunanayaka K.*, *Ekanayake K. V. J. P.* Review on AI Assistant Systems for Programming Language Learning in Learning Environments // 2024 8th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI). — 2024. — С. 1—6. — DOI: 10.1109/SLAAI-ICAI63667.2024.10844969.

24. Software Engineering Educational Experience in Building an Intelligent Tutoring System / Z. Fan [та ін.] // 2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T). — 2025. — С. 75—86. — DOI: 10.1109/CSEET66350.2025.00015.

25. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews / M. J. Page [та ін.] // BMJ. — 2021. — Т. 372. — n71. — DOI: 10.1136/bmj.n71.

26. *Wu D.*, *Zhang J.* Generative artificial intelligence in secondary education: Applications and effects on students' innovation skills and digital literacy // PLOS ONE. — 2025. — Т. 20. — e0323349. — DOI: 10.1371/journal.pone.0323349.

Додаток А
Вигляд JSON файлу

```
[
  {
    "type": "MC",
    "text": "Який тип даних в Python використовується
    для зберігання цілого числа?",
    "options": ["int", "float", "str", "bool"],
    "correctIndex": 1,
    "points": 1
  },
  {
    "type": "CHECKBOX",
    "text": "Які з наведених операторів використовуються
    для організації циклів у Python?",
    "options": ["for", "while", "loop", "repeat"],
    "correctIndices": [1, 2],
    "points": 2
  },
  {
    "type": "SHORT",
    "text": "Як називається вбудована функція Python
    для виведення тексту на екран?",
    "points": 1
  }
]
```

Додаток Б
Повний код скрипту csvtoforms.gs

```
const RESULTS_SHEET_NAME = 'Results';
const ADAPTIVE_LINKS_SHEET_NAME = 'AdaptiveLinks';

const WEAK_TOPIC_THRESHOLD = 70;
const ADAPTIVE_TOTAL_QUESTIONS = 6;
const ADAPTIVE_FORM_TITLE_PREFIX = 'Адаптивний тренажер - ';

let SETTINGS_CACHE = null;

const DEFAULT_FREE_MODELS = [
  'tngtech/deepseek-r1t2-chimera:free',
  'qwen/qwen3-235b-a22b:free'
];


/************ ЧИТАННЯ НАЛАШТУВАНЬ ************/

function reloadSettings_() { SETTINGS_CACHE = null; }

function loadSettings_() {
  SETTINGS_CACHE = {};
  const ss = SpreadsheetApp.getActive();
  let sheet = ss.getSheetByName('Settings');
  if (!sheet) return;

  const values = sheet.getDataRange().getValues();
  // Проходимо по всіх рядках Settings
  for (let i = 1; i < values.length; i++) {
    const key = String(values[i][0] || '').trim();
    if (key) {
      SETTINGS_CACHE[key] = values[i][1];
    }
  }
}

/**
 * Отримує налаштування з пріоритетом:
```

```
 * 1. Script Properties (для API ключів)
 * 2. Аркуш "Settings"
 * 3. Значення defaultValue
 */
function getSetting_(key, defaultValue) {
  // 1. Спочатку пробуємо знайти в ScriptProperties (безпечне сховище для
  ↪  ключів)
  const scriptVal =
  ↪  PropertiesService.getScriptProperties().getProperty(key);
  if (scriptVal) return scriptVal;

  // 2. Якщо немає, завантажуємо з таблиці
  if (!SETTINGS_CACHE) loadSettings_();

  if (SETTINGS_CACHE && (key in SETTINGS_CACHE) && SETTINGS_CACHE[key] !==
  ↪  '') {
    const val = SETTINGS_CACHE[key];

    // Якщо очікуємо число (defaultValue є числом), пробуємо конвертувати
    if (typeof defaultValue === 'number') {
      const num = parseFloat(String(val).replace(',', '.'));
      return isNaN(num) ? defaultValue : num;
    }

    // Якщо очікуємо булеве значення
    if (typeof defaultValue === 'boolean') {
      const str = String(val).toLowerCase();
      return str === 'true' || str === '1' || str === 'yes' || str ===
      ↪  'так';
    }

    return val;
  }

  // 3. Якщо ніде немає - повертаємо дефолт
  return defaultValue;
}

/************ МЕНЮ ************/

function onOpen() {
  SpreadsheetApp.getUi().createMenu('Інформатика AI (Pro)')
```

```
    .addItem('Створити тест (клас/тема)...', 'createQuizViaPrompts_')
    .addSeparator()
    .addItem('Імпорт результатів (CSV)...', 'importResultsFromDriveCsv_')
    .addItem('Експорт результатів (Аркуш -> Results)',
    ↪  'exportFromActiveResponseSheet_')
    .addSeparator()
    .addItem('Згенерувати адаптивні тести (всім)',
    ↪  'generateAdaptiveQuizzesForAll_')
    .addItem('Адаптивний тест (одному)...',
    ↪  'generateAdaptiveQuizForOnePrompt_')
    .addSeparator()
    .addItem('Розіслати листи', 'sendAdaptiveLinksByEmail_')
    .addToUi();
}

/************ API OPENROUTER (JSON MODE) ************/

function callOpenRouterJson_(prompt) {
  const keysString = getSetting_('OPENROUTER_API_KEYS', '');
  if (!keysString) throw new Error('Додайте OPENROUTER_API_KEYS в аркуш
  ↪  Settings');

  const apiKeys = keysString.split(',').map(k => k.trim()).filter(Boolean);
  const modelsString = getSetting_('OPENROUTER_MODELS', '');
  const models = modelsString ? modelsString.split(',').map(m => m.trim())
  ↪  : DEFAULT_FREE_MODELS;

  // Перемішуємо ключі для розподілу навантаження
  for (let i = apiKeys.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [apiKeys[i], apiKeys[j]] = [apiKeys[j], apiKeys[i]];
  }

  Logger.log(`Старт генерації JSON. Ключів: ${apiKeys.length}`);
  let lastError = null;

  for (const apiKey of apiKeys) {
    for (const model of models) {
      try {
        Logger.log(`Спроба: ${model}`);
        const json = makeJsonRequest_(apiKey, model, prompt);
        return json; // Успіх
```

```
      } catch (e) {
        Logger.log(`FAIL ${model}: ${e.message}`);
        lastError = e;
      }
    }
  }
  throw new Error(`Всі моделі відмовили. Остання помилка:
  ↪ ${lastError?.message}`);
}

function makeJsonRequest_(apiKey, model, prompt) {
  const url = 'https://openrouter.ai/api/v1/chat/completions';

  const payload = {
    model: model,
    messages: [
      {
        role: 'system',
        content: 'You are a professional educational content creator for
        ↪ Ukrainian schools. ' +
                 'Strictly follow Ukrainian language rules. Avoid any
                 ↪ Chinese, Russian, or English artifacts in the text. ' +
                 'Ensure all Multiple Choice options are unique. ' +
                 'Output ONLY valid JSON array.'
      },
      { role: 'user', content: prompt }
    ]
  };

  const options = {
    method: 'post',
    contentType: 'application/json',
    muteHttpExceptions: true,
    headers: {
      'Authorization': 'Bearer ' + apiKey,
      'HTTP-Referer': 'https://script.google.com/',
      'X-Title': 'School Quiz JSON Pro'
    },
    payload: JSON.stringify(payload)
  };

  const res = UrlFetchApp.fetch(url, options);
```

```
  const code = res.getResponseCode();
  const body = res.getContentText();

  if (code >= 300) throw new Error(`API ${code}: ${body}`);

  const data = JSON.parse(body);
  const content = data.choices?.[0]?.message?.content;
  if (!content) throw new Error('Empty content from AI');

  // Очистка JSON від markdown
  const cleanJson = content.replace(/^```json/i, '').replace(/^```/i,
  ↪  '').replace(/```$/i, '').trim();

  try {
    const parsed = JSON.parse(cleanJson);
    if (!Array.isArray(parsed) && parsed.questions) return parsed.questions;
    if (Array.isArray(parsed)) return parsed;
    if (typeof parsed === 'object') return [parsed];
    throw new Error('Response is not an array');
  } catch (e) {
    Logger.log('Bad JSON received: ' + cleanJson);
    throw new Error('JSON Parse Error: ' + e.message);
  }
}


/************ ГЕНЕРАЦІЯ (ПРОМПТИ) ************/

function createQuizViaPrompts_() {
  const ui = SpreadsheetApp.getUi();

  const respGrade = ui.prompt('Клас', 'Наприклад: 9 клас',
  ↪  ui.ButtonSet.OK_CANCEL);
  if (respGrade.getSelectedButton() !== ui.Button.OK) return;
  const grade = respGrade.getResponseText();

  const respTopic = ui.prompt('Тема', 'Наприклад: Цикли в Python',
  ↪  ui.ButtonSet.OK_CANCEL);
  if (respTopic.getSelectedButton() !== ui.Button.OK) return;
  const topic = respTopic.getResponseText();
```

```
  const respCount = ui.prompt('Кількість питань', 'Рекомендовано: 10-12',
  ↪  ui.ButtonSet.OK_CANCEL);
  if (respCount.getSelectedButton() !== ui.Button.OK) return;
  const count = parseInt(respCount.getResponseText()) || 10;

  ui.alert('Генерація тесту... Це може зайняти до 60 секунд. Будь ласка,
  ↪  зачекайте.');

  try {
    const questions = generateJsonQuestions_(grade, topic, count);
    const title = `${grade} - Тест: ${topic}`;
    const form = createFormFromJson_(questions, title, `Тема: ${topic}.
    ↪  Згенеровано автоматично.`);

    ui.alert(
      'Тест успішно створено!',
      `Редагувати (для вчителя):\n${form.getEditUrl()}\n\n` +
      `Проходити (для учнів):\n${form.getPublishedUrl()}`,
      ui.ButtonSet.OK
    );
  } catch (e) {
    ui.alert('Помилка генерації: ' + e.message);
    Logger.log(e);
  }
}

function generateJsonQuestions_(grade, topic, count) {
  const prompt = `
    Ти - професійний методист з інформатики в Україні. Створи якісний тест
    ↪  українською мовою.
    Клас: ${grade}. Тема: "${topic}". Кількість питань: ${count}.

    ВИМОГИ:
    1. Мова: ЛІТЕРАТУРНА УКРАЇНСЬКА.
    2. Варіанти відповідей мають бути УНІКАЛЬНИМИ.
    3. Правильна відповідь має бути ОДНОЗНАЧНОЮ.
    4. Структура:
       - 70% MC (одна правильна).
       - 20% CHECKBOX (кілька правильних).
       - 10% SHORT (коротка текстова).

    ФОРМАТ (JSON Array):
```

```
    [
      {
        "type": "MC",
        "text": "Питання?",
        "options": ["А", "Б", "В", "Г"],
        "correctIndex": 1,
        "points": 1
      },
      {
        "type": "CHECKBOX",
        "text": "Питання з кількома відповідями...",
        "options": ["А", "Б", "В", "Г"],
        "correctIndices": [1, 3],
        "points": 2
      }
    ]
  `;
  return callOpenRouterJson_(prompt);
}

function generateAdaptiveQuizForStudent_(email) {
  // 1. Отримуємо поріг слабкої теми з налаштувань (або дефолт 70)
  const weakThreshold = getSetting_('WEAK_TOPIC_THRESHOLD',
  ↪  DEFAULT_WEAK_TOPIC_THRESHOLD);

  // 2. Отримуємо кількість питань з налаштувань (або дефолт 6)
  const totalQuestions = getSetting_('ADAPTIVE_TOTAL_QUESTIONS',
  ↪  DEFAULT_ADAPTIVE_TOTAL_QUESTIONS);

  // Передаємо threshold у функцію розрахунку статистики
  const stats = getStudentsStats_(weakThreshold);
  const st = stats[email];

  if (!st) throw new Error(`Учня ${email} немає в Results`);
  if (!st.weakTopics.length) throw new Error('Немає слабких тем (результат
  ↪  високий).');

  const weakStr = st.weakTopics.map(t => `${t.topic}
  ↪  (${Math.round(t.avg)}%)`).join(', ');

  const prompt = `
    Створи адаптивний тест з інформатики (JSON). Мова: Українська.
```

```
  Слабкі теми учня: ${weakStr}.
  Кількість питань: ${totalQuestions}.
  Варіанти відповідей НЕ ПОВИННІ ДУБЛЮВАТИСЯ.

  JSON формат:
  [
    {"type":"MC","text":"...","options":["A","Б","B","Г"],"correctIndex":
    ↪  1,"points":1},
    {"type":"CHECKBOX","text":"...","options":["A","B","C"],"correctIndic
    ↪  es":[1,2],"points":2}
  ]
`;

const questions = callOpenRouterJson_(prompt);

const prefix = getSetting_('ADAPTIVE_FORM_TITLE_PREFIX',
↪  ADAPTIVE_FORM_TITLE_PREFIX);
const title = prefix + (st.name || email);
const form = createFormFromJson_(questions, title, 'Індивідуальний
↪  адаптивний тест.');

// Save Link
const ss = SpreadsheetApp.getActive();
// Динамічна назва аркуша
const linksSheetName = getSetting_('ADAPTIVE_LINKS_SHEET_NAME',
↪  DEFAULT_ADAPTIVE_LINKS_SHEET_NAME);
let sheet = ss.getSheetByName(linksSheetName);
if (!sheet) {
  sheet = ss.insertSheet(linksSheetName);
  sheet.appendRow(['email', 'name', 'weak_topics', 'form_edit_url',
  ↪  'form_url', 'sent']);
}

sheet.appendRow([
  st.email, st.name, weakStr, form.getEditUrl(), form.getPublishedUrl(),
  ↪  ''
]);

if (getSetting_('SEND_EMAILS_AUTOMATICALLY', false)) {
  GmailApp.sendEmail(st.email, 'Тренувальний тест', `Ваш персональний
  ↪  тест: ${form.getPublishedUrl()}`);
  sheet.getRange(sheet.getLastRow(), 6).setValue(new Date());
```

```
  }

  return form;
}


/************ СТВОРЕННЯ ФОРМИ З JSON ************/

function createFormFromJson_(questions, title, desc) {
  if (!questions || !questions.length) throw new Error('Отримано порожній
  ↪  список питань від AI');

  const form = FormApp.create(title);
  form.setIsQuiz(true);
  form.setCollectEmail(false);
  form.setDescription(desc);

  // Функція для унікалізації варіантів
  const uniquifyOptions = (opts) => {
    const seen = new Set();
    return opts.map(opt => {
      let val = String(opt).trim();
      while (seen.has(val)) {
        val += ' ';
      }
      seen.add(val);
      return val;
    });
  };

  questions.forEach(q => {
    let type = (q.type || 'MC').toUpperCase();
    if (q.correctIndices && Array.isArray(q.correctIndices)) type =
    ↪  'CHECKBOX';

    // === MULTIPLE CHOICE ===
    if (type.includes('MC')) {
      const item = form.addMultipleChoiceItem();
      item.setTitle(q.text || 'Питання');
      item.setPoints(q.points || 1);

      let rawOpts = (q.options || []).filter(Boolean);
```

```
    if (rawOpts.length < 2) {
      form.addTextItem().setTitle(q.text + ' (Введіть
      ↪  відповідь)').setPoints(q.points || 1);
      return;
    }

    const opts = uniquifyOptions(rawOpts);

    let correctIdx = (q.correctIndex || 1) - 1;
    if (correctIdx < 0 || correctIdx >= opts.length) correctIdx = 0;

    const choices = opts.map((optText, i) => {
      return item.createChoice(optText, i === correctIdx);
    });
    item.setChoices(choices);
  }
  // === CHECKBOX ===
  else if (type.includes('CHECKBOX') || type.includes('MULTI')) {
    const item = form.addCheckboxItem();
    item.setTitle(q.text || 'Оберіть правильні варіанти');
    item.setPoints(q.points || 2);

    let rawOpts = (q.options || []).filter(Boolean);
    if (rawOpts.length < 2) {
       form.addParagraphTextItem().setTitle(q.text).setPoints(q.points ||
       ↪  1);
       return;
    }

    const opts = uniquifyOptions(rawOpts);

    const correctIndices = (q.correctIndices || []).map(idx => idx - 1);

    const choices = opts.map((optText, i) => {
      const isCorrect = correctIndices.includes(i);
      return item.createChoice(optText, isCorrect);
    });
    item.setChoices(choices);
  }
  // === SHORT/LONG ANSWER ===
  else if (type.includes('SHORT')) {
```

```
      form.addTextItem().setTitle(q.text || 'Питання').setPoints(q.points
      ↪  || 1);
    } else {
      form.addParagraphTextItem().setTitle(q.text ||
      ↪  'Питання').setPoints(q.points || 1);
    }
  });

  return form;
}


/************ ІНШІ ФУНКЦІЇ ************/

function generateAdaptiveQuizzesForAll_() {
  const ui = SpreadsheetApp.getUi();
  ui.alert('Починаю генерацію... Це займе час.');

  // Отримуємо поріг слабкої теми динамічно
  const weakThreshold = getSetting_('WEAK_TOPIC_THRESHOLD',
  ↪  DEFAULT_WEAK_TOPIC_THRESHOLD);
  const stats = getStudentsStats_(weakThreshold);

  let count = 0;

  for (const email in stats) {
    try {
      if (stats[email].weakTopics.length) {
        generateAdaptiveQuizForStudent_(email);
        count++;
      }
    } catch(e) { Logger.log(`Skipped ${email}: ${e.message}`); }
  }
  ui.alert(`Створено тестів: ${count}`);
}

function generateAdaptiveQuizForOnePrompt_() {
  const ui = SpreadsheetApp.getUi();
  const resp = ui.prompt('Email учня', ui.ButtonSet.OK_CANCEL);
  if (resp.getSelectedButton() !== ui.Button.OK) return;
  const email = resp.getResponseText().trim();
```

```
  if (email) {
    try {
      const form = generateAdaptiveQuizForStudent_(email);
      ui.alert(
        'Готово!',
        `Редагувати: ${form.getEditUrl()}\nУчню: ${form.getPublishedUrl()}`,
        ui.ButtonSet.OK
      );
    } catch(e) { ui.alert(e.message); }
  }
}

function importResultsFromDriveCsv_() {
  const ui = SpreadsheetApp.getUi();
  const resp = ui.prompt('Посилання на CSV або ID', ui.ButtonSet.OK_CANCEL);
  if (resp.getSelectedButton() !== ui.Button.OK) return;
  const url = resp.getResponseText().trim();

  if (!url) return;
  try {
    let id = url;
    const match = url.match(/[-\w]{25,}/);
    if (match) id = match[0];

    const blob = DriveApp.getFileById(id).getBlob();
    const data = Utilities.parseCsv(blob.getDataAsString());

    const ss = SpreadsheetApp.getActive();
    // Динамічна назва Results
    const resSheetName = getSetting_('RESULTS_SHEET_NAME',
    ↪  DEFAULT_RESULTS_SHEET_NAME);
    let sh = ss.getSheetByName(resSheetName);
    if (!sh) sh = ss.insertSheet(resSheetName);
    else sh.clearContents();

    sh.getRange(1, 1, data.length, data[0].length).setValues(data);
    ui.alert('Імпортовано!');
  } catch(e) { ui.alert('Помилка: ' + e.message); }
}

// Приймає поріг як аргумент (щоб можна було передати динамічне значення)
function getStudentsStats_(customThreshold) {
```

```
const ss = SpreadsheetApp.getActive();
const resSheetName = getSetting_('RESULTS_SHEET_NAME',
↪  DEFAULT_RESULTS_SHEET_NAME);
const sheet = ss.getSheetByName(resSheetName);

if (!sheet) return {};
const vals = sheet.getDataRange().getValues();
if (vals.length < 2) return {};

const h = vals[0].map(x => String(x).toLowerCase());

const iEmail = h.findIndex(x => x.includes('email'));
const iTopic = h.findIndex(x => x.includes('topic'));
const iScore = h.findIndex(x => x.includes('score') ||
↪  x.includes('percent'));
const iName = h.findIndex(x => x.includes('name'));

if (iEmail < 0) return {};

const students = {};
for (let r = 1; r < vals.length; r++) {
  const row = vals[r];
  const email = String(row[iEmail]).trim();
  if (!email) continue;

  if (!students[email]) students[email] = {
    email,
    name: iName >= 0 ? row[iName] : '',
    topics: {},
    avgTopics:[],
    weakTopics:[]
  };
  const topic = iTopic >= 0 ? String(row[iTopic]) : 'General';
  const score = iScore >= 0 ? parseFloat(row[iScore]) : 0;

  if (!students[email].topics[topic]) students[email].topics[topic] = [];
  students[email].topics[topic].push(score);
}

// Використовуємо переданий поріг або дефолтний
const weakTh = (customThreshold !== undefined) ? customThreshold :
↪  getSetting_('WEAK_TOPIC_THRESHOLD', DEFAULT_WEAK_TOPIC_THRESHOLD);
```

```
  Object.values(students).forEach(st => {
    for (const [t, arr] of Object.entries(st.topics)) {
      const avg = arr.reduce((a,b)=>a+b,0)/arr.length;
      st.avgTopics.push({topic:t, avg});
      if (avg < weakTh) st.weakTopics.push({topic:t, avg});
    }
  });
  return students;
}

function exportFromActiveResponseSheet_() {
  const ss = SpreadsheetApp.getActive();
  const sheet = ss.getActiveSheet();
  const ui = SpreadsheetApp.getUi();

  const rT = ui.prompt('Тема', ui.ButtonSet.OK_CANCEL);
  if (rT.getSelectedButton() !== ui.Button.OK) return;
  const topic = rT.getResponseText();

  const rM = ui.prompt('Макс. бал', ui.ButtonSet.OK_CANCEL);
  if (rM.getSelectedButton() !== ui.Button.OK) return;
  const max = parseFloat(rM.getResponseText());

  const data = sheet.getDataRange().getValues();
  if (data.length < 2) return;

  const h = data[0].map(x => String(x).toLowerCase());
  const iEmail = h.findIndex(x => x.includes('email') ||
  ↪  x.includes('адреса'));
  const iScore = h.findIndex(x => x.includes('score') || x.includes('бал')
  ↪  || x.includes('результат'));
  const iName = h.findIndex(x => x.includes("ім'я") || x.includes('name'));

  if (iEmail < 0 || iScore < 0) { ui.alert('Не знайдено Email або Бал');
  ↪  return; }

  const resSheetName = getSetting_('RESULTS_SHEET_NAME',
  ↪  DEFAULT_RESULTS_SHEET_NAME);
  let resSheet = ss.getSheetByName(resSheetName);
  if (!resSheet) {
    resSheet = ss.insertSheet(resSheetName);
```

```
    resSheet.appendRow(['email', 'name', 'topic', 'score_percent']);
  }

  let added = 0;
  data.slice(1).forEach(row => {
    if (row[iEmail]) {
      const sc = parseFloat(row[iScore]);
      if (!isNaN(sc)) {
        resSheet.appendRow([row[iEmail], iName>=0?row[iName]:'', topic,
        ↪  (sc/max)*100]);
        added++;
      }
    }
  });
  ui.alert(`Експортовано ${added} рядків.`);
}

function sendAdaptiveLinksByEmail_() {
  const ss = SpreadsheetApp.getActive();
  const linksSheetName = getSetting_('ADAPTIVE_LINKS_SHEET_NAME',
  ↪  DEFAULT_ADAPTIVE_LINKS_SHEET_NAME);
  const sheet = ss.getSheetByName(linksSheetName);

  if (!sheet) return;
  const data = sheet.getDataRange().getValues();

  let sent = 0;
  for (let i=1; i<data.length; i++) {
    if (!data[i][5] && data[i][0] && data[i][4]) {
      GmailApp.sendEmail(data[i][0], 'Тренувальний тест', `Ваш тест:
      ↪  ${data[i][4]}`);
      sheet.getRange(i+1, 6).setValue(new Date());
      sent++;
    }
  }
  SpreadsheetApp.getUi().alert(`Надіслано: ${sent}`);
}

function showStudentProfilePrompt_() {
  const ui = SpreadsheetApp.getUi();
  const resp = ui.prompt('Email учня:', ui.ButtonSet.OK_CANCEL);
  if (resp.getSelectedButton() === ui.Button.OK) {
```

```
    try { showStudentProfile_(resp.getResponseText().trim()); }
    catch(e) { ui.alert(e.message); }
  }
}

function showStudentProfile_(email) {
  const stats = getStudentsStats_();
  const st = stats[email];
  if (!st) throw new Error('Не знайдено.');

  let msg = `Учень: ${st.name || email}\nСлабкі теми:\n`;
  if (st.weakTopics.length) st.weakTopics.forEach(t => msg += `- ${t.topic}
  ↪  (${Math.round(t.avg)}%)\n`);
  else msg += "(відсутні)";

  SpreadsheetApp.getUi().alert(msg);
}
```