Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматики
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Інформаційна технологія тестування знань при вивченні іноземних слів»

Виконала: студентка 2 курсу, групи 1КН-23м
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
_____ Тодощак А.М.
(прізвище та ініціали)
Керівник: к.ф.-м.н., доцент каф. КН
_____ Шевчук О. Ф.
(прізвище та ініціали)
«05» 12 2024 р.
Опонент: к.т.н., доцент каф. САІТ
_____ Горячев Г.В
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
«06» 12 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А. А.
11.10. 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Тодощак Анні Миколаївні
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія тестування знань при вивченні іноземних слів
   керівник роботи к.ф.-м.н., доцент кафедри КН Шевчук О. Ф.
   затверджені наказом ВНТУ від «17» 09 2024 року № 310
2. Строк подання студентом роботи 11.11. 2024 року
3. Вихідні дані до роботи: мова програмування – об'єктно-орієнтована; мінімальна кількість тестових завдань різної складності – 500; мінімальна кількість тестових питань до користувача – 10; мінімальна кількість параметрів налаштування інтегральної оцінки тестування – 4; тип інтерфейсу користувача інтуїтивно-зрозумілий.
4. Зміст текстової частини:
   Вступ, аналіз сучасних інформаційних технологій тестування знань при вивченні іноземних слів, моделювання інформаційної технології тестування знань при вивченні іноземних слів, програмна реалізація інформаційної технології тестування знань при вивченні іноземних слів, економічна частина, висновки, перелік використаних джерел, додатки
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): схема удосконаленого алгоритму тестування знань при вивченні іноземних слів, основні складові структури інформаційної технології, UML-діаграма послідовностей роботи та UML-діаграма класів інформаційної технології тестування знань при вивченні іноземних слів, результати тестування розробленої інформаційної технології тестування знань при вивченні іноземних слів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Шевчук О.Ф., к.ф.-м.н, доцент каф. КН		
4	Лесько О.Й., д.е.н., проф. каф. ЕПВМ		

7. Дата видачі завдання 02.09. 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасних інформаційних технологій тестування знань при вивченні іноземних слів	02.09.24-16.09.24	
2	Моделювання інформаційної технології тестування знань при вивченні іноземних слів	17.09.24 - 01.10.24	
3	Програмна реалізація інформаційної технології тестування знань при вивченні іноземних слів	02.10.24 - 15.10.24	
4	Підготовка економічної частини	16.10.24 - 28.10.24	
5	Апробація та/або впровадження результатів дослідження	29.10.24 - 04.11.24	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	05.11.24 - 11.11.24	

Студентка _____ Тодощак А.М.
(підпис)

Керівник роботи _____ Шевчук О.Ф.
(підпис)

АНОТАЦІЯ

УДК 004.9

Тодощак А. М. Інформаційна технологія тестування знань при вивченні іноземних слів. Магістерська кваліфікаційна робота зі спеціальності 122 «Комп'ютерні науки», освітня програма «Системи штучного інтелекту». Вінниця: ВНТУ, 2024. 123 с.

Укр. мовою. Бібліогр.: 38 назв; рис.: 25; табл. 14.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології тестування знань при вивченні іноземних слів. Проведено огляд сучасних технологій та моделей тестування знань для вивчення іноземних мов. Розроблено математичну модель та удосконалений алгоритм інтегральної оцінки тестування, що враховує час виконання та рівень складності завдання. Розроблено структуру інформаційної технології адаптивного тестування з автоматичним налаштуванням складності завдань. Реалізовано програмне забезпечення, обґрунтовано вибір мов та середовища розробки, створено UML-діаграми класів та послідовностей для ключових модулів системи. Тестування підтвердило коректність роботи модулів при встановлених порогових значеннях ефективності $\theta = 0{,}8$, вагових коефіцієнтах точності та часу $\omega_1 = 0{,}6$, $\omega_2 = 0{,}4$; коефіцієнтах складності завдань $\alpha = 1$; $\alpha = 0{,}85$ та $\alpha = 0{,}7$.

Графічна частина складається з 6 плакатів.

У економічному розділі оцінено та доведено ефективність виконання науково-дослідної роботи з високим науковим, технічним і економічним рівнями $(К_р = 1{,}085;\ СБ_с = 40{,}67)$. Визначено, що орієнтовна сума загальних витрат на проведення всіх етапів науково-дослідної роботи та оформлення її результатів складає 298478,3 грн.

Ключові слова: інформаційна технологія, тестування знань, адаптивне тестування, інтегральна оцінка, іноземні слова.

ABSTRACT

Todoschak A. M. Information technology for knowledge testing in foreign language vocabulary learning. Master's thesis in the specialty 122 «Computer science», educational program «Artificial intelligence systems». Vinnytsia: VNTU, 2024. 123 p.

In Ukrainian. Bibliography: 38 titles; illustrations: 25; tables: 14.

This master's qualification thesis is dedicated to the development of an information technology system for knowledge testing in the process of learning foreign words. A review of modern technologies and models for knowledge testing in foreign language learning was conducted. A mathematical model and an improved algorithm for integral assessment of testing were developed, incorporating task completion time and difficulty level. The structure of the adaptive testing information technology was designed, featuring automatic adjustment of task difficulty. Software was implemented, with a justified selection of programming languages and development environments. UML diagrams of classes and sequences were created for the system's key modules. Testing confirmed the correctness of module performance under the established efficiency threshold values of $\theta = 0.8$, accuracy and time weight coefficients of $\omega_1 = 0,6$, $\omega_2 = 0,4$, and task difficulty coefficients of $\alpha = 1$; $\alpha = 0,85$ та $\alpha = 0,7$.

The graphical part consists of six posters.

In the economic section, the effectiveness of the research and development project was evaluated and demonstrated to meet high scientific, technical, and economic standards $(К_р = 1,085;\ СБ_с = 40,67)$. The approximate total cost for all stages of the research and the preparation of its results was estimated at 298478,3 UAH.

Keywords: information technology, knowledge testing, adaptive testing, integral assessment, foreign language vocabulary.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. На сьогоднішній день відбувається швидкий розвиток науки і техніки у всіх областях людської діяльності. Особливої уваги заслуговує галузь освіти, де автоматизація та комп'ютеризація процесів навчання значно підвищують ефективність засвоєння знань. Одним із важливих аспектів освіти є вивчення іноземних мов, що вимагає постійного тестування та оцінки знань студентів. Відсутність загальнодоступних, точних та ефективних програмних засобів для тестування знань призводить до неповного засвоєння матеріалу та знижує мотивацію студентів до навчання. У цій ситуації доцільним є створення програмного засобу для тестування знань при вивченні іноземних слів, що зможе забезпечити високу точність оцінки знань та інтерактивність навчального процесу.

Тестування знань при вивченні іноземних мов на сьогоднішній день часто проводиться за допомогою традиційних методів, які не завжди забезпечують достатню ефективність та мотивацію для студентів. Зазвичай використовуються письмові тести або усні перевірки, що потребують значних зусиль з боку викладачів та студентів. Відсутність автоматизованих інструментів для тестування знань призводить до затримок у навчальному процесі та недооцінки рівня засвоєння матеріалу.

Сучасні інформаційні технології дозволяють створювати інтерактивні та адаптивні системи тестування знань, які можуть значно покращити процес вивчення іноземних мов. Використання таких систем дозволяє забезпечити індивідуальний підхід до кожного студента, швидко оцінювати рівень його знань та надавати зворотний зв'язок. Автоматизація процесу тестування знань сприяє підвищенню мотивації студентів до навчання та покращенню якості засвоєння матеріалу.

Тема магістерської кваліфікаційної роботи є актуальною, оскільки завжди існує потреба у точному та ефективному тестуванні знань студентів при вивченні іноземних мов. Розробка інформаційної технології тестування знань дозволить

забезпечити високу точність та інтерактивність навчального процесу, що буде корисним для освітніх установ різного рівня. Використання сучасних методів автоматизації та інтерактивних технологій для розв’язання поставленої задачі є актуальним через те, що надає великі можливості для модифікацій і налаштування під індивідуальні потреби користувачів, а отже, дозволяє досягати кращих результатів у порівнянні з традиційними методами.

Таким чином, застосування сучасних інформаційних технологій для тестування знань при вивченні іноземних слів забезпечує подальше впровадження інноваційних методів у галузь освіти, а також є актуальною темою дослідження нових підходів до автоматизації навчальних процесів.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напряму наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей тестування знань в процесі вивчення іноземних слів.

Для досягнення поставленої мети необхідно виконати наступні **завдання**:

- провести аналіз сучасних інформаційних технологій тестування знань при вивченні іноземних слів;
- розробити математичну модель інформаційної технології тестування знань при вивченні іноземних слів та удосконалити її згідно з метою роботи;
- розробити удосконалений алгоритм процедури тестування знань та структуру програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології тестування знань при вивчення іноземних слів;
- провести тестування й налаштування параметрів інформаційної технології та виконати аналіз отриманих результатів;

- провести оцінювання важливості та наукової значимості науково-технічної розробки за можливого її застосування.

Об’єкт дослідження – це процес проведення тестування знань при вивченні іноземних слів з використанням інформаційних технологій.

Предмет дослідження – методи та програмні засоби тестування знань при вивченні іноземних слів та їхні функціональні можливості.

Методи дослідження. У роботі використано такі методи наукових досліджень: метод системного аналізу для розробки структури інформаційної технології тестування знань; методи адаптивного тестування для персоналізації процесу оцінки знань; методи математичної статистики для обробки даних результатів тестування та налаштування параметрів інформаційної системи; методи об’єктно-орієнтованого програмування для реалізації функціональних модулів системи та автоматизації процесу розрахунків.

Наукова новизна одержаних результатів полягає в наступному: удосконалено інформаційну технологію тестування знань при вивченні іноземних слів, яка відрізняється від існуючих розширеним функціоналом, за рахунок використання адаптивного налаштування параметрів інтегральної оцінки та порогових значень складності завдань, що сприяє підвищенню точності та об'єктивності результатів оцінювання знань.

Практичне значення одержаних результатів полягає у такому:

1. Розроблено алгоритми адаптивного налаштування складності завдань та розрахунку інтегральної оцінки тестування.

2. Розроблено програмний засіб для тестування знань при вивченні іноземних слів, який інтегрує адаптивні механізми та систему автоматизованого аналізу результатів.

Представлені в роботі алгоритми та програмний засіб можуть бути впроваджені у навчальний процес для проведення тренувального та контрольного тестування знань у рамках курсів з вивчення іноземних мов.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки завдань, коректним

застосуванням методів об’єктно-орієнтованого програмування та алгоритмів адаптивного налаштування складності завдань, строгим обґрунтуванням структурних рішень та верифікацією отриманих результатів через порівняння з відомими аналогами.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У роботах, опублікованих у співавторстві, автору належать такі результати: [1] – розроблено структуру модулю тестування знань при вивченні іноземних слів, що пришвидшує процес засвоєння іншомовної лексики; [2] – розроблено удосконалений алгоритм тестування знань, що включає етап визначення складності завдань, [3] – розроблено адаптивну модель тестування за рахунок введення динамічних параметрів часу виконання завдань.

Апробація результатів роботи. Результати роботи були апробовані на LII науково-технічній конференції підрозділів Вінницького національного технічного університету НТКП ВНТУ-2023 (м. Вінниця, Україна, 2023 р.) [1], VI міжнародній науково-практичній конференції “Innovations and prospects in modern science» (м. Стокгольм, Швеція, 2023 р.) [2], II Міжнародній науково-практичній інтернет-конференції «Progressive Opportunities and Solutions of Advanced Society», (м. Дніпро, 2024 р.) [3] та опубліковані у збірниках даних конференцій.

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано 3 наукові праці: 1 стаття у збірнику праць конференції [2], 2 тези доповідей конференцій [1, 3].

Отримано свідоцтво про реєстрацію авторського права на комп’ютерну програму «Інтелектуальний модуль тестування знань при вивченні іноземних слів» [4].

1 АНАЛІЗ СУЧАСНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТЕСТУВАННЯ ЗНАНЬ ПРИ ВИВЧЕННІ ІНОЗЕМНИХ СЛІВ

Знання будь-якої іноземної мови не тільки покращує якість життя людини, а й відкриває для неї нові культури, абсолютно не притаманне їй мислення та світосприйняття.

За результатами статистичного наукового дослідження Лондонського університету, люди, які вивчають іноземні мови, мають більш гнучке та оригінальне мислення [5].

Сьогодні знання іноземних мов – це, перш за все, ключ до успіху та нових можливостей, які відкриваються перед людиною та роблять її фізичний та духовний світ набагато яскравішим, цікавим, та, в умовах глобалізаційних процесів, які відбуваються у сучасному світовому суспільстві, є просто необхідним компонентом професійної діяльності фахівця будь-якого рівня будь-якої сфери.

Отже, володіння іноземними мовами на сьогоднішній день – це просто суттєва життєва необхідність, розуміючи яку з’являється все більше і більше людей, охочих вивчати іноземні мови.

1.1 Сучасні технології тестування знань при вивченні іноземних слів

Стрімкий розвиток інформаційних технологій значно розширив інструментарій та підходи до тестування знань при вивченні іноземних мов, відкриваючи нові можливості для інтерактивних, адаптивних та персоналізованих методів оцінювання, що робить навчальний процес більш ефективним і гнучким для здобувачів освіти [6-9]. Наведемо основні технології, що визначають сучасні підходи до тестування знань у процесі вивчення іноземних слів:

1. Мобільні додатки [10, 11].

Мобільні додатки, такі як Duolingo, Memrise, Babbel, і Quizlet, стали

популярними інструментами для тестування знань у вивченні іноземних мов. До їхніх переваг слід віднести: можливість адаптації до рівня користувача (адаптивні тести); використання різноманітних типів завдань (вибір відповідей, складання речень, переклад тощо); миттєвий зворотний зв'язок та інтеграція гейміфікації для підвищення мотивації; можливість навчання у будь-якому місці завдяки мобільним пристроям. Їхніми недоліками є обмежена кількість вправ на глибокий аналіз та практичні мовні навички, вузьконаправленність та неможливість покриття всіх аспектів вивчення мови (наприклад, розмовну практику).

2. Адаптивні платформи для тестування [12].

Адаптивні системи (наприклад, Kahoot!, Socrative, Quizizz) використовують алгоритми, які підбирають рівень складності завдань на основі результатів попередніх тестів. Вони забезпечують персоналізацію навчального процесу, можливість швидкого визначення прогалин у знаннях, створення тестів у реальному часі. Проте, вимагають постійного доступу до Інтернету та є обмеженими в обсязі доступних мовних вправ.

3. Онлайн платформи з використанням штучного інтелекту [13].

Деякі сучасні платформи, наприклад, Lingvist або Busuu, використовують штучний інтелект для створення індивідуальних навчальних планів і адаптивного тестування на основі результатів користувача. Перевагами таких платформ є можливість адаптації тестування до унікальних потреб здобувача, використання великих баз даних для аналізу помилок і прогресу, висока інтерактивність і точність у вимірюванні знань. Але вони мають значну залежність від технологій ШІ, які можуть бути недоступними або недостатньо точними для окремих користувачів.

4. Гейміфікація та інтерактивні тренажери [14].

Багато сучасних систем включають елементи гейміфікації та інтерактивні тренажери для покращення вивчення іноземних мов, наприклад, Drops або Tinycards. Це дозволяє перетворювати процес тестування в захопливу гру. Отже такий підхід підвищує мотивацію та інтерес до вивчення завдяки елементам

змагань і нагород. При цьому додатково створюється відчуття прогресу через систему балів і досягнень. Проте можливе поверхневе засвоєння матеріалу через акцент на ігровий процес.

5. Системи дистанційного тестування [15].

Платформи, як Edmodo, Socrative дозволяють викладачам створювати і відправляти тести, а також оцінювати результати онлайн. Вони широко використовуються для формального тестування знань. Перевагами таких платформ є зручність для дистанційного навчання; підтримка різних типів завдань (відкриті питання, вибіркові завдання); можливість масштабованого тестування великої кількості студентів. З недоліків можна відзначити можливі труднощі з академічною чесністю та контролем за процесом тестування а також залежність від доступу до інтернету.

6. Віртуальні помічники та голосові технології [16].

Віртуальні помічники, такі як Google Assistant або Siri, надають можливість практикувати розмовні навички та миттєво отримувати зворотний зв’язок. Вони забезпечують достатньо якісну практику вимови та аудіювання через голосові команди, до того ж надають можливість тестування розмовних навичок у реальному часі. До недоліків можемо віднести обмежені можливості для глибокого тестування граматичних та лексичних знань.

Отже, сучасні методи тестування знань при вивченні іноземних слів надають широкі можливості для інтерактивного, адаптивного та персоналізованого підходу до оцінювання знань. Вони дозволяють підвищити ефективність тестування завдяки використанню технологій ШІ, мобільних додатків та гейміфікації. Однак варто враховувати обмеження кожного методу залежно від цілей навчання, зокрема, тестування розмовних навичок і глибокого аналізу знань.

1.2 Сучасні математичні моделі тестування знань при вивченні іноземних слів

Математична модель для тестування знань при вивченні іноземних мов може ґрунтуватися на оцінці різних аспектів знань, таких як лексика, граматика, сприйняття на слух, читання і розмовна мова. Однією з найефективніших моделей є адаптивне тестування на основі Байєсовської оцінки, яке дозволяє адаптувати складність завдань під рівень знань користувача і оцінювати прогрес на основі ймовірнісних розрахунків [17, 18].

Базовими елементами моделі є множина питань або завдань Q, які використовуються для тестування ($Q = \{q_1, q_2, ..., q_n\}$); множина навичок S, які підлягають тестуванню (лексика, граматика, сприйняття на слух тощо) $S = \{s_1, s_2, ..., s_m\}$ та рівень складності питань $d(q_i)$. Отже для кожного питання q_i визначається рівень його складності $d(q_i) \in [0, 1]$, де 0 визначає найнижчу складність, а 1 – найвищу.

Ймовірнісна оцінка знань. Нехай рівень знань користувача щодо певної навички s_j представлено через ймовірність $P(s_j|R)$, де R – це множина відповідей користувача на попередні питання. Тоді після кожного питання ймовірність правильного або неправильного виконання завдання змінюється на основі Байєсовської оцінки [18]:

$$P(s_j|R,\ q_i) = \frac{P(q_i|s_j) \cdot P(s_j|R)}{P(q_i)}, \qquad (1.1)$$

де $P(q_i|s_j)$ – ймовірність правильної відповіді на питання q_i для користувача з рівнем знань s_j;

$P(s_j|R)$ – початкова ймовірність того, що користувач має відповідну навичку s_j, на основі попередніх відповідей R;

$P(q_i)$ – ймовірність появи питання q_i у тесті.

Оцінка успішності. Для кожного типу завдань (лексика, граматика тощо) підраховується успішність виконання на основі ймовірнісної моделі. Після кожного завдання оновлюється ймовірність того, що користувач володіє тією чи іншою навичкою:

$$Z = \frac{1}{m}\sum_{j=1}^{m} P(s_j|R), \tag{1.2}$$

де Z – загальний рівень знань користувача;

m – кількість навичок, що тестуються.

Адаптивне тестування. Щоб зробити тест адаптивним, після кожної відповіді вибирається наступне питання (q_{i+1}) на основі ймовірнісного профілю користувача:

$$q_{i+1} = \arg\left(max\left|P(s_j|R) - P(q|s_j)\right|\right)\ ,\ \ q \in Q. \tag{1.3}$$

Тобто за такої умови буде обране те питання, яке найточніше може оцінити поточний рівень знань користувача для навички s_j.

Після завершення тесту користувач отримує підсумкову оцінку знань, яка включає як оцінку по кожній навичці (оцінюється через середнє значення ймовірностей $P(s_j|R)$ для відповідних питань), так і загальний рівень знань. При цьому підсумковий результат можна подавати як усереднений рівень знань по всіх навичках або як комплексну оцінку на основі вагомості кожної навички.

Розглянута математична модель оцінки результату тестування може бути розширена шляхом введення динамічних параметрів часу на виконання тесту, або ж з урахуванням помилок типу "майже правильна відповідь", що, зокрема, дозволить оцінити глибину знань користувача в реальних умовах спілкування мовою.

Введення динамічних параметрів часу на виконання тесту дозволяє враховувати не лише правильність відповідей, але й час, витрачений на їхнє

виконання. Це допоможе глибше оцінити рівень знань користувача, адже швидкість виконання завдань є важливим показником володіння мовою, особливо у реальних умовах комунікації [19, 20]. Розглянемо, як додаткові аспекти часу можуть бути інтегровані в попередню модель.

Моделювання часу відповіді. Позначимо через t_i час, що витрачений на надання відповіді на питання q_i. Тоді, для кожного завдання можна ввести функцію ймовірності часу відповіді $f(t_i)$, яка залежить від складності питання $d(q_i)$ та рівня знань користувача щодо відповідної навички s_j. При цьому модель може враховувати такі основні компоненти:

- швидкі відповіді на легкі питання. У випадку питання з низьким рівнем складності $d(q_i)$, користувач повинен відповідати швидше;
- більший час для складніших питань. Якщо питання є складним, то час відповіді, навпаки, зростає.

Цей підхід можна змоделювати наступною функцією:

$$f\left(t_i \middle| d(\,q_i),\, s_j\right) = exp\left(-\frac{\left(t_i - \mu\left(\,s_j,\; d(q_i)\right)\right)^2}{2\sigma^2}\right), \tag{1.4}$$

де $\mu\left(s_j,\; d(q_i)\right)$ – очікуваний час відповіді користувача з рівнем знань s_j на питання складності $d(q_i)$, σ – стандартне відхилення часу відповіді.

Корекція оцінки знань з урахуванням часу. Оцінка знань користувача може коригуватися на основі того, наскільки швидко він відповідає на запитання відносно очікуваного часу $\mu\left(\,s_j,\; d(q_i)\right)$ [18]. Якщо користувач відповідає швидше, ніж очікувалося, це вказує на глибше володіння матеріалом, і тоді ймовірність володіння навичкою може збільшуватися:

$$P\left(s_j|R,\; t_i\right) = P\left(s_j|R\right) \cdot\; f\left(t_i \middle| d(q_i),\; s_j\right). \tag{1.5}$$

Таким чином, швидка і правильна відповідь буде підвищувати оцінку знань користувача, а повільна відповідь або помилка, навпаки, можуть її знизити.

Показник ефективності відповіді. Для кожної відповіді можна вводити показник ефективності на основі точності та швидкості [21]:

$$E_i = \alpha \cdot \frac{1}{t_i} + \beta \cdot \delta_i \,, \tag{1.6}$$

де t_i – час, витрачений на відповідь, δ_i – індикатор правильної відповіді (1 – правильно, 0 – неправильно), α та β – вагові коефіцієнти, що визначають важливість часу та точності.

Цей показник дозволить врахувати не тільки правильність, а й швидкість відповіді при загальному оцінюванні рівня знань.

Адаптація рівня складності. На основі часу відповіді та показника ефективності система може адаптувати рівень складності наступних питань. Якщо користувач відповідає правильно і швидко, наступне питання може бути складнішим, тоді як тривалість або неправильна відповідь може призвести, навпаки, до зниження складності [22, 23]:

$$d(q_{i+1}) = d(q_i) + \gamma(E_i - \theta) \,, \tag{1.7}$$

де $d(q_{i+1})$ – рівень складності наступного питання, γ – коефіцієнт зміни складності, θ – порогове значення ефективності.

Аналіз прогресу з урахуванням часу. Остаточна оцінка користувача може включати часовий коефіцієнт, що враховує середній час виконання всіх завдань:

$$\bar{T} = \frac{1}{n} \sum_{i=1}^{n} t_i \,. \tag{1.8}$$

Інтегруючи цей показник із точністю відповідей, можна отримати загальну оцінку, яка дає повніше уявлення про реальний рівень володіння мовою.

Умови обмеженого часу. Для підвищення стресової стійкості та перевірки швидкості мислення можна також вводити обмеження за часом на виконання всього тесту або окремих його частин. Це дозволить моделі тестувати не лише знання, а й здатність користувача швидко застосовувати свої навички у стислі терміни.

Отже, введення динамічних параметрів часу у модель тестування знань дозволяє краще оцінювати рівень володіння мовою. Час відповіді стає додатковим показником, який може бути використаний для корекції загальної оцінки знань, адаптації складності тесту, а також для надання користувачам детальнішої зворотної інформації про їхній прогрес.

1.3 Порівняльний аналіз сучасних засобів тестування знань при вивченні іноземних слів

Насьогодні існує велика кількість застосунків для обробки даних отриманих в результаті проведення тестувань в електронному вигляді. Варто виділити декілька відомих та провести аналіз переваг та недоліків їхнього використання. Відзначимо наступні програми для електронного тестування знань:

Duolingo [24] – є однією з найпопулярніших платформ для вивчення іноземних мов, яка використовує гейміфіковані методи навчання (рис. 1.1). Вона поєднує інтерактивність, доступність та ефективність, що робить її привабливою як для початківців, так і для більш досвідчених користувачів.

Основні характеристики Duolingo:

- Гейміфікація. Duolingo використовує ігрові елементи, такі як набір очок, проходження рівнів, щоденні серії вправ, що підвищує мотивацію користувачів. Навчальний процес розділений на короткі завдання, які легко інтегрувати в повсякденне життя.
- Інтерактивні завдання. Платформа пропонує різні типи вправ – від вибору правильних варіантів до перекладу, побудови речень та аудіювання.

- Адаптивність. Duolingo адаптується до рівня користувача, пропонуючи нові завдання залежно від його успіхів і помилок.
- Мобільна доступність. Додаток доступний на смартфонах, планшетах та у вебверсії, що дозволяє навчатися будь-де та будь-коли.
- Широкий вибір мов. Платформа пропонує безліч мов для вивчення, включаючи рідкісні мови, як-от валлійська чи іврит.

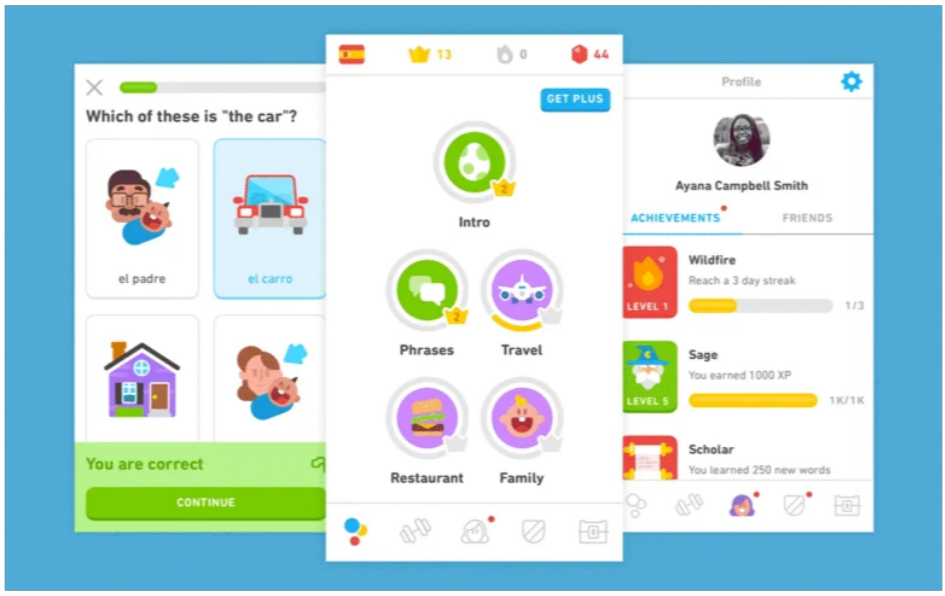


Рисунок 1.1 – Функціональні вікна мобільного додатку Duolingo

Перевагами Duolingo є безкоштовність; легкість у використанні, завдяки інтуїтивного інтерфейсу; ігрова мотиваційність, що стимулює регулярну практику та короткі уроки, що дозволяє легко інтегрувати навчання в повсякденну рутину.

До недоліків варто віднести обмежений контент для просунутих рівнів; значний фокус на перекладі (багато вправ базуються на перекладі слів і речень, що не завжди відображає реальні мовні ситуації); відсутність розмовної практики та невеликий акцент на граматиці (платформа, насамперед, зосереджується на запам'ятовуванні фраз).

Також Duolingo пропонує користувачам можливість перевірити свій прогрес через короткі тести та визначає рівень засвоєння матеріалу. Проте

глибина оцінки знань обмежена вправами платформи і не завжди відображає реальні навички спілкування.

Busuu [25] – це онлайн-платформа для вивчення іноземних мов, яка пропонує інтерактивний підхід до навчання з акцентом на комунікативні навички. Платформа отримала свою назву від мови бушу, яка є рідною для невеликої спільноти в Папуа-Нова Гвінея, і намагається поєднати інноваційні методи навчання з культурним обміном.

Основні характеристики Busuu:

- Курси для різних рівнів. Busuu пропонує курси для початківців, середнього та просунутого рівня з різних мов, таких як англійська, іспанська, французька, німецька, італійська, японська та інші.
- Інтерактивні завдання. Користувачі виконують вправи, що включають аудіювання, читання, говоріння та письмо. Завдання розроблені для розвитку всіх мовних навичок.
- Соціальна взаємодія. Busuu має функцію обміну мовами, де користувачі можуть перевіряти один одного та отримувати фідбек від носіїв мови, що робить процес навчання більш інтерактивним.
- Адаптивні технології. Платформа використовує адаптивні алгоритми для налаштування курсу в залежності від успішності учня, що забезпечує персоналізований підхід до навчання.
- Мобільний додаток (рис. 1.2). Busuu доступний на смартфонах і планшетах, що дозволяє навчатися на ходу.

Перевагами Busuu є висока якість контенту, оскільки курси створюються експертами з лінгвістики та методики викладання, що забезпечує ефективність навчання. Підтримується також можливість отримання фідбеку від носіїв мови, що значно покращує навички спілкування і розуміння мови. Гнучкість у навчанні забезпечується вільним вибором теми та рівня складності, що дозволяє навчатися відповідно до своїх потреб і часу. Відзначимо також, що завдання часто включають культурні аспекти та реальні життєві ситуації, що робить навчання більш практичним і релевантним.

Рисунок 1.2 – Функціональні вікна мобільного додатку Busuu

Недоліками є платний контент для повноцінного використання всіх функцій та ресурсів; обмежена кількість мов; можливість перевантаження (великі обсяги інформації можуть бути важкими для сприйняття, особливо для новачків).

Busuu надає можливість перевірити свої знання через тестування в кінці кожного курсу, а також регулярно оцінює прогрес учня за допомогою системи балів та досягнень. Однак, як і в інших платформах, важливо комбінувати навчання в Busuu з практикою мовлення та спілкування для досягнення максимальних результатів у вивченні мови.

Fun Easy Learn [26] – додаток, що містить шість тисяч слів. Такого словникового запасу цілком достатньо, щоб комфортно спілкуватися англійською мовою, читати тексти та дивитись фільми. Якщо вчити по п’ятдесят в тиждень, то з нуля цей об’єм можна подолати за два роки. Однак недоліком є те, що більша частина слів – іменники, які означають щось конкретне. Тому використання цього додатку є ефективним лише при наявності базових знань, які допоможуть будувати структуровані речення. Отже для ефективності цього застосунку необхідна наявність категоризації, яка буде включати в себе слова, які відносяться до різних частин мови.

Quizlet [27] – інструмент для запам’ятовування, що дозволяє користувачам створювати термінологію та набори визначень відповідно до їх потреб. Ці набори вміщують в себе флеш-карти. Цей режим схожий на режим паперових карток. Кожен термін представляє користувачеві «картку», яку можна перевернути, натиснувши або використовуючи клавіші зі стрілками чи пробіл. Користувачі можуть вибрати зображення, слово або обидва на лицьовій стороні картки.

Дана платформа не розміщує слова в завданнях по рівням складності. Тобто, наприклад, новачок у вивченні іноземної мови може захотіти вивчити тему «їжа», проте у наборі не буде стандартних, базових слів, а будуть ті, які використовують, наприклад, шеф-повари за профілем. Виходячи з цього зрозуміло, що є необхідність використання функціоналу, який спрямований на визначення складності слів, які знаходяться в тематиках. Таким чином ефективність та швидкість вивчення іноземних слів збільшиться.

Кожен з вище наведених додатків має свої переваги та недоліки при використанні (табл. 1.1). Для наглядності оцінки їх застосування, створимо таблицю.

Таблиця 1.1 – Порівняння основних характеристик і функції кожної платформ для тестування знань при вивченні іноземних слів

Характеристика	Duolingo	Busuu	Fun Easy Learn	Quizlet
1	2	3	4	5
Тип навчання	Гейміфікація	Інтерактивні курси	Ігрові вправи та флеш-картки	Флеш-картки, тести, ігри
Кількість мов	Більше 30 мов	Близько 12 мов	Понад 30 мов	Більше 20 мов
Рівні складності	Початковий, середній, просунутий	Початковий, середній, просунутий	Початковий, середній	Початковий, середній, просунутий
Адаптивність	Так (алгоритми адаптації)	Так (індивідуальні навчальні плани)	Обмежена	Так (можливість налаштування)
Соціальна взаємодія	Обмежена (коментарі)	Так (перевірка носіями мови)	Немає	Так (можливість обміну картками)

Продовження таблиці 1.1

1	2	3	4	5
Мобільний додаток	Так	Так	Так	Так
Платність	Безкоштовний з платними функціями	Безкоштовний з платними функціями	Безкоштовний з платними функціями	Безкоштовний з платними функціями
Основний фокус	Вивчення словникового запасу та граматики	Всі аспекти мови (читання, говоріння)	Вивчення словникового запасу	Запам'ятовування та повторення
Розмовна практика	Обмежена	Так (через перевірку носіями мови)	Немає	Обмежена (можливість створення ігор)
Оцінка прогресу	Так (тестування)	Так (оцінка прогресу)	Так (вимірювання успіху)	Так (аналітика результатів)

Також у таблиці 1.2 представлено порівняльну характеристику платформ для вивчення іноземних мов за критерієм оцінки прогресу.

Таблиця 1.2 – Порівняльна характеристика платформ для вивчення іноземних мов за критерієм оцінки прогресу

Duolingo	Busuu	Fun Easy Learn	Quizlet
Використовує систему балів та досягнень для оцінки успіху	Оцінка прогресу на основі виконаних завдань та тестів	Використовує прості тестування, які дозволяють користувачам перевіряти себе на кожному етапі навчання	Надає детальну статистику по користувачах, включаючи кількість створених флеш-карток та пройдених тестів.
Проводить регулярні тестування в кінці уроків, що дозволяє користувачам бачити свій прогрес у реальному часі	Надає статистику по вивченню, включаючи кількість завершених уроків і досягнення користувача	Результати можуть не бути такими детальними, як у інших платформах, але дають уявлення про засвоєння матеріалу	Можливість створювати власні набори карток, що дозволяє користувачам слідкувати за своїм прогресом у запам'ятовуванні
Інтерфейс відображає, які теми вже освоєні, а які потребують повторення	Вбудовані функції оцінки від носіїв мови, що дозволяє отримати зовнішню оцінку успішності	Не містить глибокої аналітики прогресу	Аналітика результатів доступна для користувачів, щоб оцінити свій прогрес

Узагальнюючи наведенні дані можна відзначити, що в усіх платформ відсутня комплексна оцінка мовних навичок. Більшість тестів фокусуються на окремих аспектах, таких як словниковий запас чи граматика, не охоплюючи інші важливі елементи. Це, в свою чергу, може призвести до суб'єктивних оцінок і неточностей в розумінні реального рівня володіння мовою.

1.4 Постановка задачі

Для розробки удосконаленої інформаційної технології тестування знань при вивченні іноземних слів застосуємо математичну модель у вигляді:

$$Y = f(X),$$

де: Y – рівень володіння іноземними словами;

$X(K, T, R, S, \theta)$ – вектор вхідних даних, де:

K – кількість слів для тестування;

T – час, відведений на виконання тесту;

R – точність відповідей (відсоток правильних відповідей);

S – ступінь складності тесту;

Θ – порогове значення ефективності.

Напрямки удосконалення:

1. Статистичний аналіз часу відведеного на виконання тесту та впровадження механізмів, які дозволяють автоматично його регулювати в залежності від складності запитань і швидкості відповідей здобувача. Це дозволить створити індивідуалізований підхід до навчання та покращити точність оцінки знань.

2. Розробка удосконаленого алгоритму для динамічного визначення порогового значення ефективності (θ) на основі аналізу результатів попередніх тестувань. Це допоможе адаптувати вимоги до здобувачів залежно від їхнього прогресу та забезпечить об’єктивнішу та справедливішу оцінку рівня знань.

1.5 Висновок до розділу 1

У першому розділі проведено детальний аналіз сучасних технологій та математичних моделей тестування знань при вивченні іноземних слів. Зокрема, було розглянуто основні платформи, такі як Duolingo, Busuu, Fun Easy Learn та Quizlet, які широко застосовуються для навчання мов і оцінювання прогресу користувачів. Відзначено, що ці ресурси пропонують різноманітні інструменти для інтерактивного навчання, однак мають певні обмеження в оцінці комплексних мовних навичок. Окремо розглянуто сучасні математичні моделі тестування знань, що враховують кількість та складність запитань, точність відповідей, а також час на виконання завдань. Порівняльний аналіз зазначених технологій і моделей показав, що, незважаючи на ефективність платформ для оцінювання знань, їм бракує гнучкості для адаптації до індивідуальних потреб користувачів та повної інтеграції математичних моделей для динамічного оцінювання. На основі проведеного аналізу сформульовано постановку задачі для подальшого дослідження.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ТЕСТУВАННЯ ЗНАНЬ ПРИ ВИВЧЕННІ ІНОЗЕМНИХ СЛІВ

2.1 Удосконалення математичної моделі тестування знань при вивченні іноземних слів

Удосконалення математичної моделі тестування знань при вивченні іноземних слів охоплюватиме два ключові напрямки: інтегрування точності відповідей з часом виконання тесту та введення адаптивних порогових значень для оцінки ефективності відповіді.

Насамперед, інтеграція показника часу з точністю відповідей дасть можливість побудови комбінованої оцінки, яка відображає не тільки те, наскільки правильно користувач відповідає на питання, але і те, наскільки швидко він це робить. Таким чином буде отримане повніше уявлення про реальний рівень володіння мовою, оскільки і точність, і швидкість є важливими аспектами мовної компетенції.

В загальному, основними перевагами такого поєднання показників часу виконання тесту та точності є:

- Баланс знань і швидкості. Точність показує, наскільки добре користувач знає матеріал, а час – наскільки автоматизовані ці знання. Якщо користувач відповідає правильно, але надто повільно, це може означати, що він ще не досяг достатньої впевненості у використанні мови.
- Оцінка реальних умов. У реальному спілкуванні важливо не лише знати правильну відповідь, а й встигнути її сформулювати швидко. Таким чином, інтеграція часу дозволяє оцінити ефективність застосування знань у практичних умовах.
- Мотивація для вдосконалення. Користувачі можуть бачити не лише свої помилки, але й наскільки швидко вони виконують завдання. Це може мотивувати їх працювати над тим, щоб відповідати швидше, одночасно покращуючи точність.

Крім того доцільно поєднати підхід інтеграції точності відповідей із часом виконання тесту як для кожного окремого питання, так і для всього тесту загалом. Тобто необхідно застосувати багаторівневу модель, яка враховує обидва аспекти. Це дозволить одночасно аналізувати продуктивність на рівні окремих питань та отримувати загальну оцінку для всього тесту [3].

Алгоритм об'єднання включатиме наступні два етапи:

1. Розрахунок для кожного питання. На першому етапі оцінка проводиться для кожного питання індивідуально. Окремо враховуються правильність відповіді (α_i) та фактично витрачений час відповіді (t_i) на питання. Після чого, розраховується оцінка кожного питання:

$$E_i = \omega_1 \cdot \alpha_i + \omega_2 \cdot \frac{t_{norm,\,i}}{t_i}\,, \tag{2.1}$$

Де $t_{norm,\,i}$ – нормативний час для виконання конкретного питання,

ω_1 та ω_2 – вагові коефіцієнти, що визначають важливість часу та точності

2. Узагальнення результатів для загального тесту. Після того, як оцінки для всіх питань розраховані, можна отримати загальну оцінку для всього тесту, використовуючи середнє значення E_i для кожного питання або підсумовуючи їх із відповідними вагами:

$$E_{\text{тест}} = \omega_1 \cdot \frac{n_{\text{прав}}}{n} + \omega_2 \cdot \frac{T_{norm}}{T_{\text{факт}}}\,, \tag{2.2}$$

де $n_{\text{прав}}$ – кількість правильних відповідей,

n – загальна кількість питань у тесті,

T_{norm} – загальний нормативний час на весь тест,

$T_{\text{факт}}$ – загальний час, витрачений на тест.

Таким чином, перший підхід надаватиме можливість детально аналізувати кожне окреме питання. Що у свою чергу дозволить визначати, на яких питаннях

учень зазнає труднощів або витрачає забагато часу, навіть якщо відповідає правильно. А другий рівень дозволятиме отримати загальну оцінку ефективності виконання тесту, яка враховує всі питання в цілому, а також час на виконання тесту.

До переваг запропонованого підходу можна віднести:

- Гнучкість. Викладач або система можуть адаптувати алгоритм залежно від конкретної мети тестування – чи необхідно глибоко аналізувати відповіді на кожне питання, чи швидко оцінити загальний прогрес.
- Повнота оцінки. Поєднання двох рівнів дозволяє отримати як локальну (по кожному питанню), так і глобальну (по всьому тесту) оцінку, що підвищує об’єктивність і точність оцінювання знань.

Таким чином, комбінуючи обидва підходи, ми отримуємо багатовимірну модель, яка враховує різні аспекти продуктивності учня і дозволяє більш детально аналізувати його знання.

Наступним напрямком удосконалення моделі є впровадження адаптивних порогових значень ефективності (θ), яке використовуватиметься для оцінки результативності відповіді користувача а також для прийняття рішення щодо зміни рівня складності наступних запитань. У контексті пропонованої математичної моделі тестування знань цей показник заложитиме як від правильності, так і від швидкості. Значення ефективності для кожної відповіді порівнюється з певним порогом θ, щоб оцінити, чи досягає користувач очікуваного рівня. Отже цей поріг виступає в якості певного стандарту.

Проте, вибір порогового значення ефективності θ залежить від декількох факторів, які пов’язані з цілями тестування, рівнем знань користувачів та складністю тестових завдань. Щоб правильно визначити це значення, потрібно врахувати наступні аспекти:

1. Цілі тестування (базовий чи високий рівень знань). Якщо тестування спрямоване на перевірку базового рівня знань, значення θ можна вибрати на нижчому рівні, щоб оцінка залишалася доступною для більшості користувачів. Якщо тест орієнтований на користувачів з високим рівнем знань, значення θ має

бути вищим, щоб відображати більш суворі вимоги до ефективності.

2. Ступінь складності питань (легкі та складні). Для тестів, що містять здебільшого легкі питання, θ можна встановити на більш високому рівні, оскільки користувачі, ймовірно, відповідатимуть швидко і правильно. Якщо тест містить важкі питання, значення θ слід знизити, щоб користувачі мали змогу набрати позитивний результат, навіть якщо час на відповідь триваліший.

3. Швидкість відповіді. Для тестів, де швидкість є ключовим показником, значення θ можна підвищити, щоб заохочувати швидкі відповіді. Наприклад, для експрес-тестів або екзаменів на час. Якщо швидкість відповіді не є критичною (наприклад, у випадках, коли головна мета – ґрунтовне розуміння матеріалу), θ можна знизити, щоб дозволити користувачам витрачати більше часу на відповіді.

4. Попередні дані тестувань. Використання результатів попередніх тестів може допомогти встановити оптимальне значення θ. Наприклад, на основі історичних даних можна обчислити середній час відповіді та точність для різних груп користувачів і налаштувати θ таким чином, щоб відповідати реальним можливостям користувачів.

5. Адаптивність тестування. Значення θ можна змінювати в процесі тестування залежно від того, як користувач виконує завдання. Наприклад, якщо користувач відповідає правильно і швидко, поріг можна поступово підвищувати для підвищення складності тесту.

У роботі для початкової оцінки порогового значення ефективності θ буде застосований наступний алгоритм:

1. Збір даних. Зібрати дані щодо часу відповіді та правильності для різних груп користувачів. Наприклад, середній час для досвідчених користувачів може становити 10 секунд, а для початківців – 20 секунд.

2. Аналіз розподілу. Побудувати розподіли для часу відповіді та точності відповідей для кожного рівня знань. Для кожного рівня визначити мінімальний прийнятний поріг ефективності.

3. Емпіричний вибір. На основі цих даних вибрати таке значення θ, яке є прийнятним для досягнення адекватного рівня оцінки. Наприклад, якщо 80%

досвідчених користувачів відповідають за 10 секунд з точністю 90%, θ може бути встановлене відповідно до цих критеріїв.

4. Тестування та корекція. Після початкового вибору θ, проводяться пробні тести для перевірки відповідності порогового значення реальним результатам. Якщо більшість користувачів не досягає порогу, його можна знизити, або навпаки, підвищити для складніших тестів.

Додатково в якості альтернативних підходів, будуть проаналізовані методи:

1. Аналіз перцентилів. Вибрати θ, засноване на перцентилі даних про ефективність попередніх користувачів. Наприклад, можна вибрати θ, яке відповідає 75-му перцентилю серед часу та точності відповідей.

2. Симуляція. Використання симуляцій для визначення оптимального значення θ, яке максимізує точність оцінки знань користувачів, мінімізуючи кількість неправильних рішень (занадто складні або легкі питання).

Підсумовуючи, можна зазначити, що порогове значення ефективності θ є ключовим показником, що визначає результати користувача в тесті, враховуючи як швидкість, так і правильність відповідей. Він виступає основою для адаптації рівня складності завдань, роблячи процес тестування динамічним і більш відповідним рівню знань користувача. Вибір значення θ має бути гнучким і базуватися на рівні складності тестів, знаннях користувачів та цілях тестування. Емпіричний підхід до вибору цього показника, що ґрунтується на аналізі даних і адаптації до отриманих результатів, дозволить підвищити точність оцінювання та адаптивність тестової системи.

2.2 Розробка удосконаленого алгоритму тестування знань при вивченні іноземних слів

Алгоритм тестування знань при вивченні іноземних слів орієнтований на швидкий процес отримання, обробки та повернення даних користувачеві з метою отримання результуючого значення після виконання тестування.

Класичний алгоритм тестування знань зазвичай розпочинається з етапу вибору завдання, де користувач може обрати один із запропонованих варіантів. Завдання можуть бути представлені у формі довгих запитань із кількома короткими відповідями. Однією з альтернативних форм тестування є завдання, що вимагають заповнення пропусків у наданих реченнях. У цьому випадку користувач має можливість обрати відповідь, яка, на його думку, є найбільш доречною, спираючись на вже наявні знання.

Завдання можуть також включати в себе візуальні елементи, такі як картинки, або можливість голосового введення відповіді. Для тестування знань при вивченні іноземних слів особливо актуальним є використання методу вибору однієї правильної відповіді. Проте цей підхід має обмеження, зокрема відсутність можливості визначення рівня складності завдання, що може уповільнювати процес вивчення іноземних слів. Тому доцільно запровадити механізм вибору завдання, який враховує складність.

Введення оцінки складності слів, що відповідатиме певним рівням, дозволить користувачам не лише обирати тематику завдання, а й рівень складності вивчених слів. Це забезпечить систематичне проходження кожної теми, що в свою чергу сприятиме формуванню показника, який дозволяє класифікувати слова за їх складністю. Завдяки цьому система автоматично генеруватиме завдання, що включають слова, відносно яких були визначені відповідні характеристики [2].

Основні етапи тестування знань за допомогою автоматизованого класичного алгоритму включають:

1. Вибір тематики. Користувач має можливість налаштувати тест, переглядаючи запропоновані варіанти. Залежно від своїх уподобань та потреб, користувач обирає групу слів, яку бажає вивчити.

2. Виведення завдання на екран. Питання з’являються послідовно і вимагають надання відповіді. Лише в цьому випадку користувач матиме шанс отримати наступне завдання. Цей цикл триватиме до завершення всіх завдань блоку тематики, яка була обрана.

3. Запам’ятовування та оцінка відповідей. При проходженні кожного з питань відбувається фіксація відповіді та її співставлення з правильною. У разі коректної відповіді нараховуються бали, які додаються до змінної "результат". Завдяки цьому циклу, по завершенні тестування формується підсумковий результат.

4. Виведення результатів. По завершенні тестування відображаються результати виконаних завдань.

Удосконалений алгоритм тестування знань при вивченні іноземних слів додатково включатиме етап визначення складності завдання, а також адаптивні порогові значення для оцінки ефективності відповіді, зокрема показник порогу ефективності θ, що визначає результативність користувача. При формуванні результатів тестування передбачатиметься виведення відсотка правильно виконаних завдань. Формування завдань за складністю буде відбуватися на основі попереднього проходження завдань користувачем. Це удосконалення надасть можливість користувачеві самостійно обирати рівень складності завдання, що сприятиме швидшому вдосконаленню навичок.

Додатково до врахування точності відповідей, важливим аспектом удосконаленого алгоритму є аналіз часу, витраченого на відповідь. Швидкість відповіді є важливим показником рівня автоматизації знань. Інтеграція часу відповіді у загальну модель тестування дозволяє створити більш точну оцінку навичок користувача, оскільки враховується як правильність, так і швидкість надання відповіді [3].

На основі отриманого показника ефективності, система зможе адаптивно регулювати складність завдань, а також коригувати зворотній зв'язок для користувача. Якщо користувач відповідає повільно, навіть за правильної відповіді, алгоритм враховуватиме цей фактор для підвищення мотивації до вдосконалення не тільки точності, але й швидкості відповіді.

Таким чином, час відповіді стає ключовим компонентом у загальній оцінці знань, оскільки забезпечує більш повне уявлення про рівень володіння мовою і здатність користувача застосовувати знання в умовах, наближених до реальних.

Проте на початковому етапі функціонування системи тестування знань необхідно визначити рівень складності кожного тесту та встановити нормативний час, потрібний для його проходження. Це дозволить створити стандартизовані умови тестування для всіх користувачів. Для реалізації цього процесу система повинна збирати та аналізувати дані, отримані під час виконання тренувальних завдань різними користувачами. Отримана інформація буде використана для навчання системи з метою вдосконалення алгоритмів оцінювання та для підготовки користувачів до проведення контрольних тестів.

Загалом початковий етап передбачає збирання даних (результати відповідей і час виконання), аналіз (визначення складності завдань і розрахунок часу) та навчання системи для подальшої адаптації тестів до рівня знань користувачів і підвищення точності оцінювання. На рис. 2.1 наведено UML-діаграму алгоритму початкового аналізу рівня складності тесту та часу його проходження. Відзначимо її основні рівні:

1. Аналіз складності тесту. Визначення рівня складності тесту ґрунтується на даних про успішність виконання завдань різними користувачами під час тренувальних тестів. Після збору інформації про кількість правильних відповідей система класифікує тестові завдання за рівнем складності відповідно до таких правил:

- Простий рівень. Якщо понад 90% користувачів дали правильну відповідь на завдання, тест вважається простим.
- Середній рівень. Тест визначається як середньої складності, якщо відсоток користувачів, що дали правильні відповіді, становить від 60% до 90%.
- Складний рівень. тест вважається складним, якщо правильні відповіді надали менше ніж 60% користувачів.

2. Аналіз часу проходження тесту. Окрім аналізу правильності відповідей, система також фіксує час, необхідний кожному користувачеві для виконання тесту. Цей час використовується для розрахунку основних числових характеристик, таких як:

- Середній час виконання тесту для всіх користувачів.

- Медіанний час – час, при якому 50% користувачів завершують тест.
- Дисперсія часу – для оцінки варіативності часу виконання завдань.

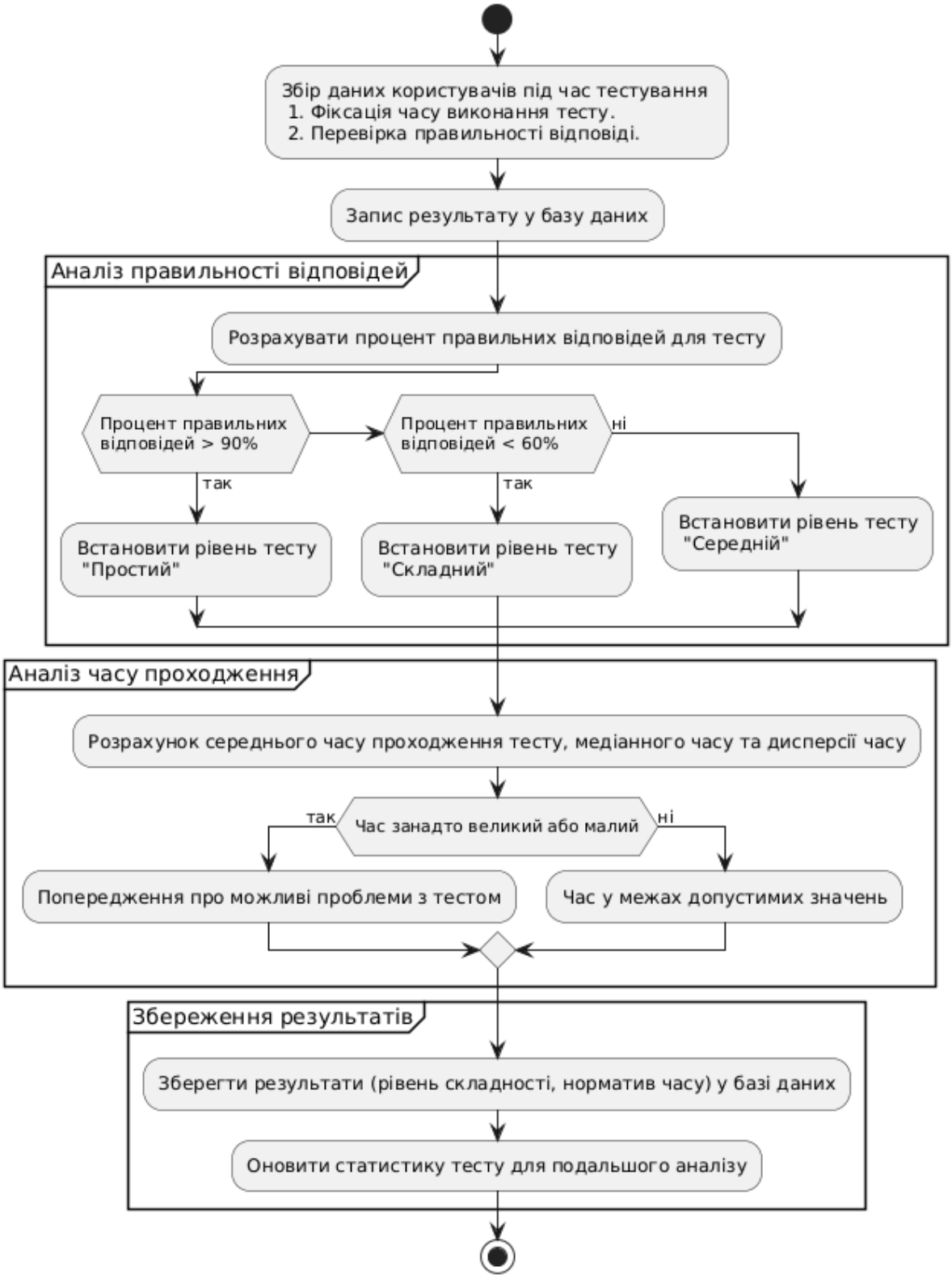


Рисунок 2.1 – UML-діаграма алгоритму аналізу рівня складності тесту та часу його проходження

Аналіз часу є важливим елементом для оцінки ефективності тестів. Наприклад, надмірно великий або, навпаки, дуже короткий час виконання може свідчити про невідповідність рівня складності тесту або неправильне структурування завдань. Ці показники також допомагають системі налаштовувати адаптацію складності для різних категорій користувачів та створювати рекомендації щодо оптимального часу для виконання тестів.

3. Використання отриманих даних. Отримані дані про рівень складності тестів та час їх проходження будуть використовуватись для подальшого удосконалення системи тестування. Зокрема, ці дані дозволяють:

- Налаштовувати вдосконалену тестову оцінку: з урахуванням рівня знань користувачів та їх продуктивності при виконанні тестів. Це може включати персоналізовану адаптацію складності або тривалість тестування, що дозволить покращити якість оцінювання і підвищити відповідність тестів рівню підготовки користувачів.

- Оцінювати ефективність тестів: за допомогою аналізу взаємозв'язку між рівнем складності та часом виконання завдань можна визначити, наскільки добре тест відповідає очікуваному рівню знань користувачів та чи потребує він коригування. Наприклад, якщо для тестів з однаковим рівнем складності різко відрізняється час виконання або кількість правильних відповідей, це може свідчити про необхідність перегляду тесту.

- Ідентифікувати потенційні проблеми з тестом: аналіз часу виконання та правильності відповідей також дозволить виявити аномалії, такі як надто великий або малий час проходження тесту, які можуть сигналізувати про некоректне формулювання завдань або про те, що рівень складності тесту не відповідає реальним знанням користувачів. Це дозволить вчасно коригувати тести, забезпечуючи їх точність і справедливість.

На наступному етапі функціонування системи тестування відбувається безпосереднє проведення тесту з адаптацією рівня складності на основі відповідей користувача в режимі реального часу. UML-діаграма удосконаленого алгоритму тестування знань представлена на рис. 2.2.

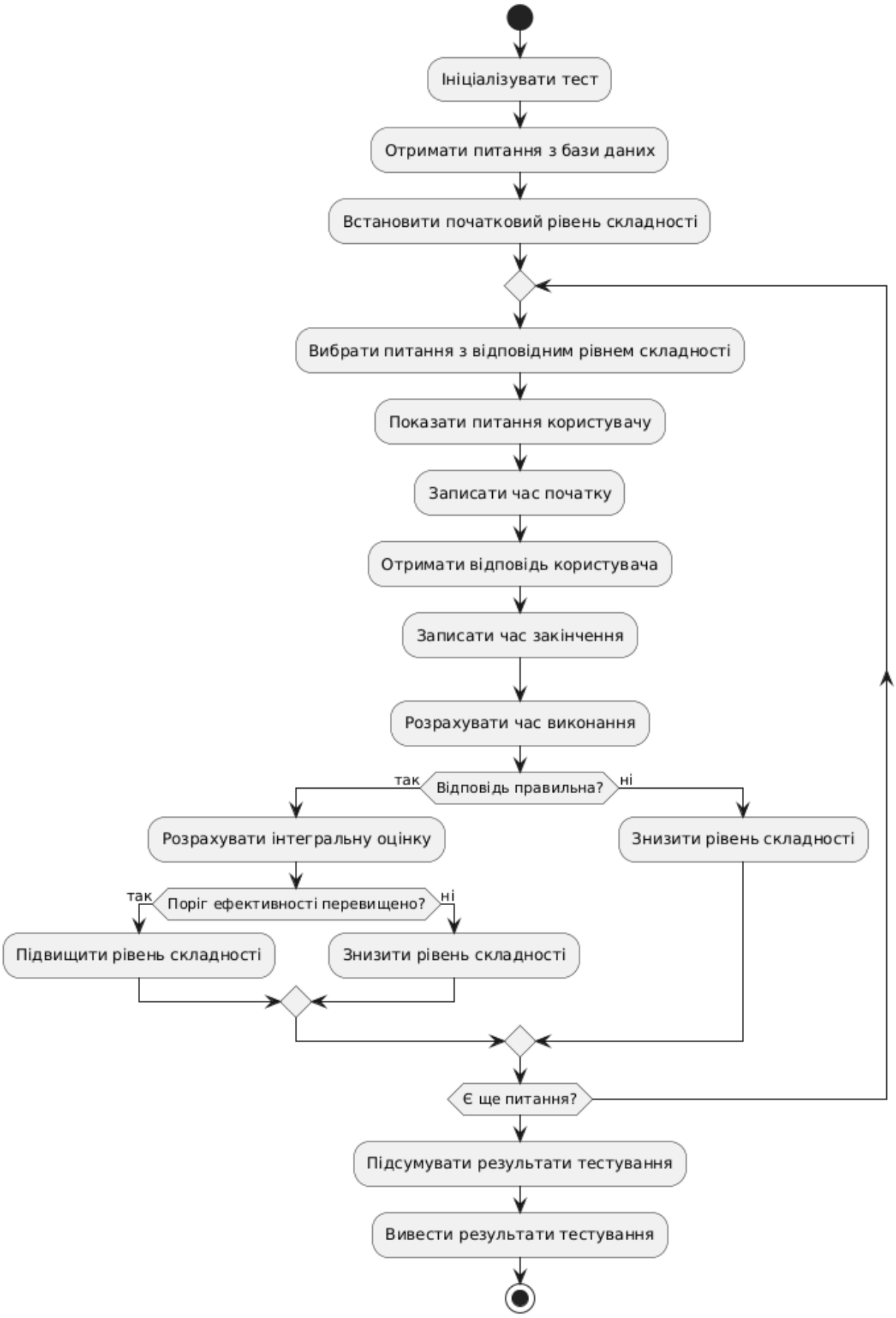


Рисунок 2.2 – UML-діаграма удосконаленого алгоритму тестування знань при вивченні іноземних слів

Його основними кроками є: ініціалізація тесту (на початку тесту система ініціалізує сесію та отримує питання з бази даних); вибір завдання відповідного рівня складності (початковий рівень складності визначається на основі попередньої інформації про користувача або загальних параметрів тесту); отримання відповіді та її аналіз (аналізується правильність відповіді та розраховується час, витрачений на відповідь); адаптація рівня складності (якщо відповідь правильна, бал користувача збільшується, а рівень складності наступного питання підвищується, і навпаки, якщо відповідь неправильна, система зменшує рівень складності наступного завдання; оцінка ефективності; розрахунок результатів та їхнє виведення.

Отже, запропонований удосконалений алгоритм тестування знань підвищить ефективність процесу оцінювання та навчання за рахунок адаптації рівня складності завдань в режимі реального часу. Завдяки автоматичному формуванню питань, які змінюють свою складність залежно від правильності відповідей і часу виконання, система забезпечує індивідуальний підхід до кожного користувача, що сприяє кращому засвоєнню матеріалу та більш точній оцінці знань. Такий підхід також дозволяє вчасно ідентифікувати потенційні проблеми з тестовими завданнями і коригувати їх для підвищення якості тестування.

2.3 Розробка структури інформаційної технології тестування знань при вивченні іноземних слів

Основним завданням інформаційної технології тестування знань при вивченні іноземних слів є забезпечення ефективного навчання та точної оцінки рівня знань користувачів через автоматизовані алгоритми адаптивного тестування. Для досягнення цієї мети система повинна відповідати низці ключових вимог, зокрема забезпечувати надійність, масштабованість, гнучкість в адаптації під різні рівні знань, а також можливість легкої інтеграції з іншими навчальними платформами або сервісами [28].

Ефективність навчання в системі досягається завдяки використанню алгоритмів, які динамічно підлаштовують складність завдань на основі продуктивності користувачів. Таким чином, система дозволяє швидко оцінити знання користувачів і запропонувати тести відповідно до їхнього рівня.

Удосконалена оцінка знань реалізується через використання різних методів аналізу результатів тестування. Включення в систему таких функцій, як аналіз правильності відповідей, час виконання завдань, а також адаптація складності, дозволяє забезпечити об'єктивність і точність оцінювання.

На основі описаного вище алгоритму роботи інформаційної технології пропонується структура (рис. 2.3), яка включає такі основні складові:

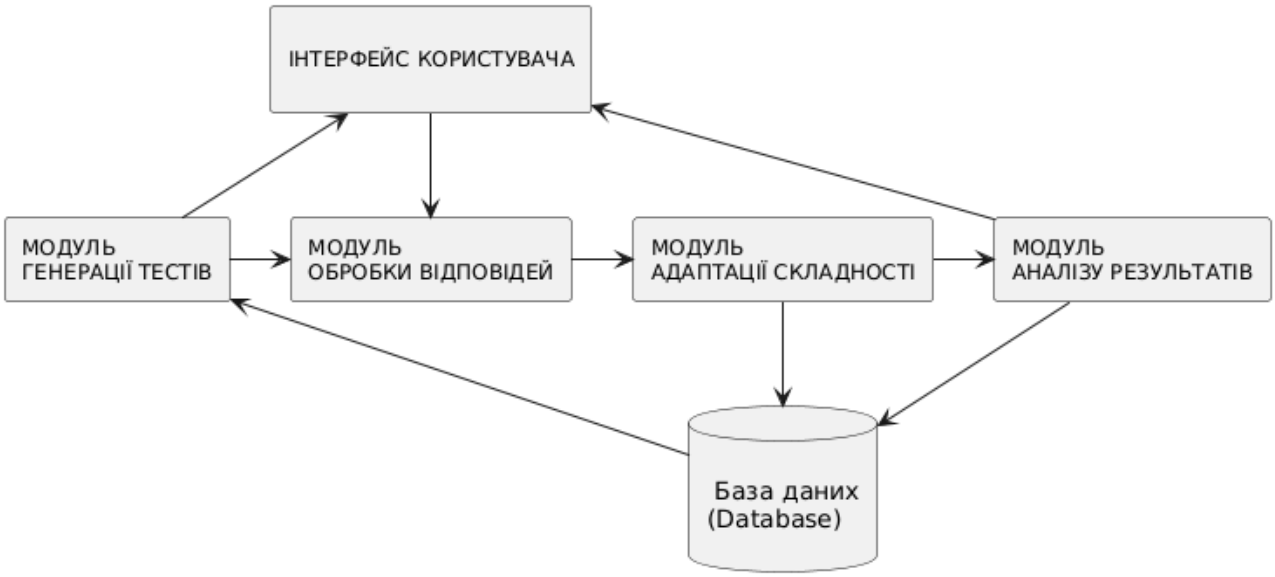


Рисунок 2.3 – Основні складові структури інформаційної технології тестування знань при вивченні іноземних слів

1. Інтерфейс користувача є ключовим компонентом системи, який забезпечує взаємодію між користувачем і системою тестування. Він повинен бути зручним, інтуїтивним і адаптованим під різні пристрої (десктоп, планшет, мобільний телефон). Основні функції, що реалізуються через інтерфейс:

- Відображення тестових завдань користувачу.
- Отримання відповідей і фіксація часу на кожне питання.

- Підсумкова візуалізація результатів тестування (результати, статистика, аналіз помилок).
- Можливість вибору рівня складності або режиму тестування (тренувальний або контрольний).

2. Модуль генерації тестів. Цей модуль відповідає за формування завдань для користувачів. Він використовує попередньо підготовлену базу даних зі словами, поділеними на рівні складності (простий, середній, складний). Функціонал модуля передбачає:

- Вибір відповідних питань на основі історії відповідей користувача та встановленого рівня складності.
- Динамічне формування тестів з урахуванням адаптивної складності (на основі відповідей користувача система може підвищувати або знижувати рівень складності завдань).
- Інтеграцію з базою даних для отримання інформації про кожне слово, його частоту використання та складність для різних користувачів.

3. Модуль обробки відповідей. Цей модуль виконує обробку даних про відповіді користувача. Кожна відповідь фіксується з інформацією про час виконання завдання, після чого система визначає правильність відповіді і проводить первинний аналіз: перевірка відповідей на правильність; фіксація часу на виконання завдань для подальшого аналізу продуктивності.

4. Модуль адаптації складності. Один із ключових компонентів системи, який забезпечує адаптивність тестування. На основі правил адаптації (якщо більше 90% користувачів відповідають правильно – рівень тесту простий, від 60% до 90% – складний, менше 60% – середній) система автоматично налаштовує складність наступних питань:

- Збільшення або зменшення рівня складності тесту залежно від продуктивності користувача.
- Використання даних з модулю аналізу правильності відповідей для гнучкої адаптації процесу навчання.

5. Модуль аналізу результатів. Цей модуль відповідає за збір та аналіз усіх результатів тестування для подальшого вдосконалення процесу. Основні функції:

- Формування статистики про кожен тест, включаючи правильність відповідей і час виконання.
- Визначення ключових числових характеристик часу проходження тестів (середнє значення, медіана, дисперсія), що дозволяє оцінити ефективність кожного користувача.
- Виявлення можливих проблем з тестовими завданнями (аномально довгий або короткий час виконання, занадто високий або низький відсоток правильних відповідей).
- Налаштування алгоритму адаптації складності на основі результатів.

6. База даних. База даних є центральним сховищем інформації для системи. Вона містить інформацію про слова, їхні переклади, рівень складності; дані про кожне тестове завдання та його характеристику (використовуване слово, рівень, середній час виконання, відсоток правильних відповідей); історію відповідей користувачів і дані про їхню продуктивність (результати попередніх тестів, адаптація рівня складності).

7. Зв'язки між модулями. Система функціонує за рахунок взаємодії між модулями. Кожен модуль має доступ до бази даних через чітко визначені інтерфейси API, що дозволяє ефективно керувати запитами на отримання і збереження даних. Наприклад, модуль генерації тестів звертається до бази даних для отримання питань; модуль обробки відповідей передає інформацію до модуля адаптації складності, який визначає наступний рівень питань; модуль аналізу результатів отримує усі зібрані дані та формує статистичні звіти.

Ключовою особливістю інформаційної технології тестування знань при вивченні іноземних слів є можливість її постійного вдосконалення на основі зворотного зв’язку від користувачів та даних про проходження тестів. Адаптивний підхід до формування тестів дозволяє забезпечити індивідуалізоване навчання, враховуючи індивідуальні темпи навчання та

поточний рівень знань кожного користувача.

Система також постійно аналізує результати тестів для виявлення можливих недоліків у формуванні питань, таких як надто висока складність чи неадекватний час на виконання завдання. Такий підхід дозволяє оперативно вносити корективи та підвищувати якість тестування.

На рис. 2.4 наведено UML-діаграму послідовностей, яка демонструє динамічну взаємодію між користувачем та різними модулями системи тестування знань при вивченні іноземних слів. Основні етапи процесу включають авторизацію користувача, генерацію тесту, обробку відповідей, адаптацію складності тестів і аналіз результатів.

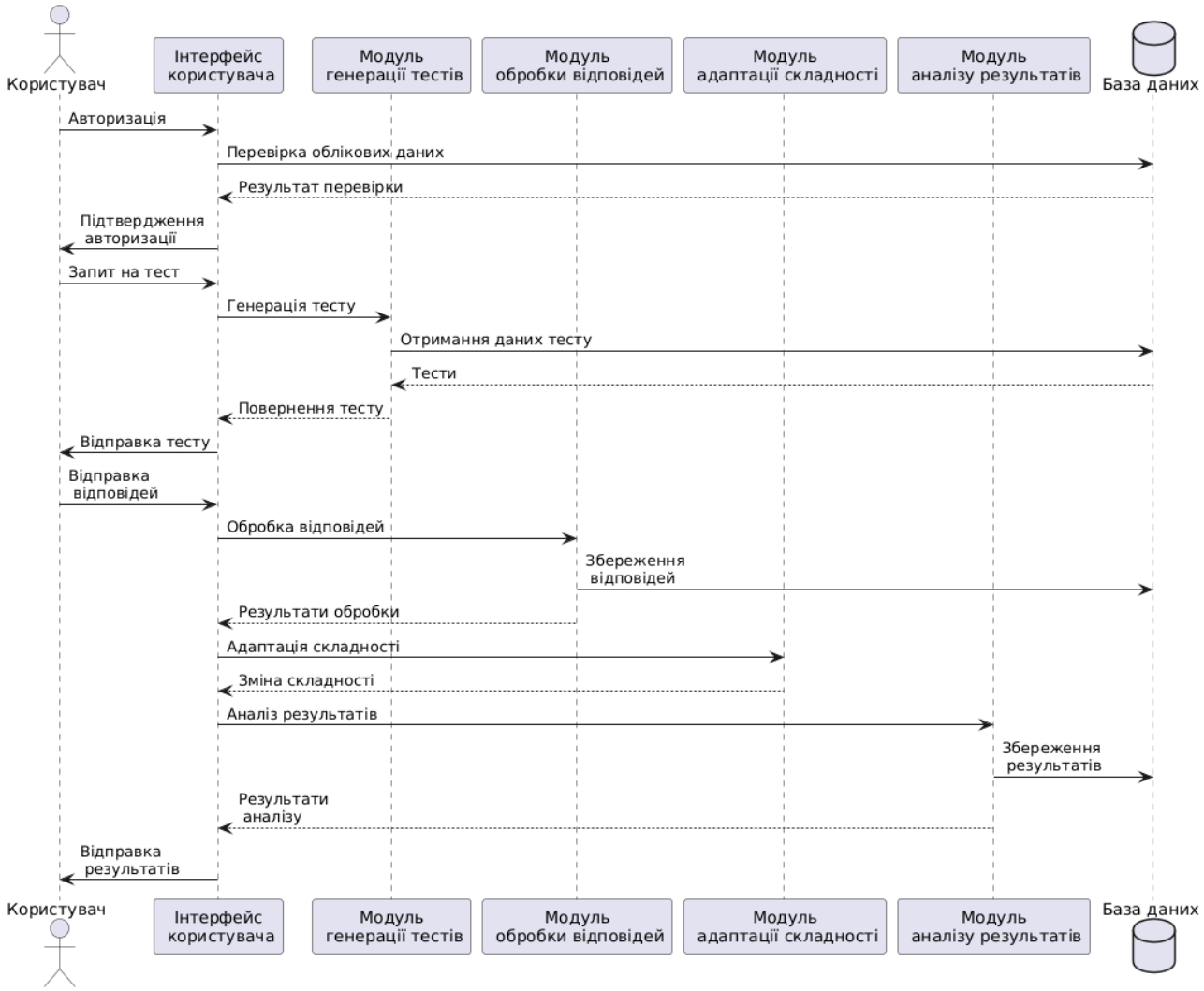


Рисунок 2.4 – UML-діаграма послідовностей роботи інформаційної системи тестування знань при вивченні іноземних слів

1. Авторизація користувача. Після запиту на авторизацію система звертається до бази даних для перевірки облікових даних користувача. У разі успішної перевірки користувач отримує підтвердження доступу.

2. Генерація тесту. Після успішної авторизації користувач запитує тест. Система звертається до модуля генерації тестів, який отримує необхідні дані з бази даних. Потім тест передається користувачу для виконання.

3. Обробка відповідей. Коли користувач надсилає свої відповіді, система передає їх до модуля обробки відповідей, який фіксує дані в базі даних і аналізує правильність відповідей.

4. Адаптація складності. За потреби система використовує модуль адаптації складності, щоб відповідно до попередніх результатів користувача коригувати рівень складності тесту.

5. Аналіз результатів. Модуль аналізу результатів здійснює оцінку ефективності тесту, фіксує дані про результативність у базі даних і формує звіт для користувача, який відправляється після завершення тесту.

2.4 Висновок до розділу 2

У другому розділі було розроблено математичну модель та удосконалений алгоритм тестування знань при вивченні іноземних слів, що включає механізм автоматичного визначення складності завдань на основі попередніх відповідей користувача, інтеграцію часу виконання для врахування як точності, так і швидкості відповідей, та адаптивні порогові значення ефективності відповіді θ. Це забезпечує персоналізацію процесу навчання та дозволяє більш точно оцінювати рівень володіння мовою. Розроблено також структуру інформаційної системи, яка надає зворотний зв'язок, адаптуючи завдання до поточного рівня знань користувача, що сприяє швидшому та ефективнішому засвоєнню матеріалу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ТЕСТУВАННЯ ЗНАНЬ ПРИ ВИВЧЕННІ ІНОЗЕМНИХ СЛІВ

3.1 Обґрунтування вибору мови програмування для розробки інформаційної технології тестування знань при вивченні іноземних слів

На початковому етапі розробки будь-якого програмного забезпечення, зокрема на етапі проектування, вибір мови програмування має ключове значення. Щоб обрати найбільш відповідну мову програмування, необхідно врахувати ряд вимог і провести порівняння між різними варіантами. На основі цього аналізу можна підбити підсумки, визначивши, чи дозволятиме обрана мова програмування реалізувати всі необхідні вимоги та визначені завдання [29].

Java [30] є однією з найпоширеніших мов програмування, що використовується для розробки великих та складних інформаційних систем. Її підтримка глобальною спільнотою забезпечує доступ до багатьох ресурсів, документації та готових рішень для різних проєктів. Стабільність та зрілість Java роблять її ідеальним вибором для тривалих проєктів.

Завдяки об'єктно-орієнтованому програмуванню (ООП) Java дозволяє ефективно структурувати систему через класи та об'єкти, що спрощує повторне використання коду. Для системи тестування знань, яка включає модулі генерації тестів, адаптації складності та аналізу результатів, можна створити, наприклад, клас Test для формування питань і клас User для відстеження прогресу користувачів.

Java відзначається високою продуктивністю і здатністю обробляти великі обсяги даних, що важливо для системи, яка адаптує рівень складності завдань та аналізує результати. Принцип "write once, run anywhere" дозволяє запускати програми на різних платформах без змін у коді, що забезпечує гнучкість у використанні на Windows, Linux та macOS.

Можливості Java для інтеграції з реляційними та нереляційними базами даних (JDBC, Hibernate) сприяють ефективному управлінню даними

користувачів, тестових питань та результатів тестування. Це забезпечує збереження та маніпуляцію даними в базі.

Крім того, Java дозволяє організовувати модульну структуру програми, що є важливим для системи тестування знань. Кожен модуль можна розробляти окремо, спрощуючи підтримку та оновлення. Вбудовані засоби безпеки, такі як криптографічні бібліотеки, захищають персональні дані користувачів.

Java також має розвинену екосистему для автоматизованого тестування (JUnit, TestNG), що дозволяє перевіряти функціональність модулів та системи загалом. За допомогою JavaFX або Swing можна створити сучасний графічний інтерфейс, що забезпечить зручність для користувачів під час проходження тестів.

C++ [31] є однією з найбільш поширених і впливових мов програмування, відомою своєю універсальністю та ефективністю. Вона активно використовується для розробки як системного, так і прикладного програмного забезпечення, зокрема операційних систем, драйверів, ігор та вбудованих систем. C++ вирізняється тим, що поєднує можливості низькорівневого програмування, характерні для мови C, з потужними інструментами об'єктно-орієнтованого підходу. Це робить її ідеальною для створення складних програмних рішень, де важлива продуктивність і контроль за ресурсами системи.

Серед ключових особливостей C++ – її система класів, яка дозволяє організовувати код навколо об'єктів та їх взаємодії. Класи можуть містити як дані, так і методи, що забезпечує гнучкість в управлінні логікою програми. Об'єкти класів створюються під час виконання, і це дозволяє програмі ефективно працювати з ресурсами. Крім того, C++ підтримує концепцію успадкування, що полегшує повторне використання коду та розширення функціональності за допомогою нових класів.

C++ також надає потужний набір операторів, зокрема арифметичних, порівняльних, бітових та логічних, що робить її гнучким інструментом для широкого спектра завдань. Особливістю мови є можливість перевантаження операторів, що дозволяє адаптувати їх поведінку для специфічних типів даних.

C++ є ідеальним вибором для тих проєктів, де продуктивність та оптимізація є критично важливими, оскільки вона дозволяє програмісту працювати з системними ресурсами на низькому рівні, при цьому забезпечуючи високу модульність і гнучкість розробки.

Python [32] є високорівневою мовою програмування, яка стала популярною завдяки своїй простоті та зрозумілості. Вона ідеально підходить для швидкої розробки та прототипування завдяки інтуїтивному синтаксису і структурованості коду. Python часто вибирають за його здатність забезпечити високу продуктивність розробників, особливо в проєктах, де важлива швидкість написання та підтримки коду. Його універсальність дозволяє використовувати мову в найрізноманітніших галузях, від веб-розробки до наукових досліджень та машинного навчання.

Однією з ключових характеристик Python є його динамічний тип даних, що дозволяє зосередитися на логіці програми, а не на її структурі. Python підтримує кілька парадигм програмування, включаючи об'єктно-орієнтоване, функціональне та процедурне, що дає розробникам велику свободу у виборі стилю коду. У Python реалізована потужна система модулів і пакетів, що дозволяє легко масштабувати проєкти та розширювати їх функціональність.

Крім того, Python має величезну кількість бібліотек та фреймворків, які спрощують виконання різних завдань. Наприклад, такі фреймворки як Django і Flask широко використовуються для веб-розробки, а бібліотеки NumPy, Pandas та TensorFlow – для наукових обчислень та аналізу даних. Інтерпретована природа Python дозволяє швидко тестувати та виправляти код без необхідності компіляції, що значно скорочує цикл розробки.

Завдяки простоті синтаксису та широким можливостям, Python є відмінним вибором для тих проєктів, де пріоритетом є зручність та швидкість розробки, гнучкість та легкість у підтримці коду, особливо у галузях, що вимагають обробки великих обсягів даних або інтенсивних обчислень.

C# (.NET) – це об'єктно-орієнтована мова програмування, створена компанією Microsoft, яка відзначається своєю інтеграцією в екосистему .NET.

Вона була розроблена для створення широкого спектра додатків, від настільних програм до вебсервісів і мобільних додатків. Мова C# поєднує в собі переваги високорівневої мови, такої як Java, з ефективністю та потужністю низькорівневого програмування, характерного для C++.

C# забезпечує сильну типізацію, що робить її кращою для великих і складних систем, де важлива передбачуваність поведінки та структурована організація коду. Завдяки вбудованій підтримці об'єктно-орієнтованих принципів, таких як успадкування, поліморфізм і інкапсуляція, C# забезпечує ефективне створення складних систем і повторно використовуваних компонентів. Крім того, мова підтримує асинхронне програмування, що дозволяє створювати високопродуктивні програми з низькою затримкою та багатозадачністю.

Однією з сильних сторін C# є тісна інтеграція з .NET Framework, який надає широкий набір бібліотек для вирішення різноманітних завдань, включаючи обробку даних, доступ до баз даних, роботу з мережею та інтерфейсами користувача. Завдяки цьому C# є чудовим вибором для розробки бізнес-додатків, де важлива стабільність, масштабованість та безпека. C# також добре підходить для розробки багатоплатформних програм завдяки .NET Core, що дозволяє легко створювати програми, які працюватимуть на Windows, macOS і Linux.

Завдяки високій продуктивності, підтримці сучасних парадигм програмування та широкому набору інструментів і фреймворків, C# є оптимальним вибором для розробників, які створюють надійні та масштабовані рішення у галузях веб-розробки, хмарних обчислень, ігор та корпоративних систем.

JavaScript (Node.js) – це мова програмування, яка спочатку була створена для роботи в браузері, але завдяки Node.js вона також стала популярною на серверній стороні. Node.js [33] – це середовище виконання, яке дозволяє запускати JavaScript поза браузером, що робить його чудовим інструментом для розробки повноцінних вебдодатків із єдиною мовою на обох кінцях –

клієнтському та серверному.

JavaScript є динамічно типізованою мовою, що надає велику гнучкість у написанні коду. Однією з головних її переваг є асинхронність та подієво орієнтована модель. Це дозволяє JavaScript ефективно обробляти великий потік запитів із мінімальною затримкою, що робить її надзвичайно придатною для розробки серверів, які працюють у реальному часі, наприклад, для чатів або систем обміну повідомленнями. Завдяки своєму неблокуючому вводу/виводу, Node.js є чудовим вибором для розробки високонавантажених додатків, які потребують швидкої обробки запитів і масштабованості.

Екосистема JavaScript є однією з найбільших у світі, з величезною кількістю бібліотек і модулів, доступних через менеджер пакетів npm (Node Package Manager). Це дає розробникам доступ до широкого спектра інструментів для створення додатків будь-якої складності, від вебдодатків до серверів, мобільних програм і навіть інструментів для штучного інтелекту.

Node.js підтримує як об'єктно-орієнтоване, так і функціональне програмування, що дозволяє розробникам використовувати різні підходи до вирішення завдань. Завдяки таким фреймворкам, як Express.js, можна швидко створювати вебсервери та API, а також інтегруватися з базами даних і зовнішніми сервісами.

JavaScript (Node.js) забезпечує високу швидкість розробки, кросплатформеність і масштабованість, що робить його відмінним вибором для створення сучасних вебдодатків, мікросервісів, серверів реального часу та хмарних додатків.

Отже, ознайомившись із характеристиками наведених мов програмування, у таблиці 3.1 узагальнено наведемо основні їхні переваги та недоліки.

Проаналізувавши наведені результати, було прийнято рішення використати поєднання Java та Python для реалізації інтелектуальної системи тестування знань при вивченні іноземних слів. Java виявилася лідером серед конкурентів для побудови надійної серверної архітектури та забезпечення ефективної роботи з базами даних, що є важливим для стабільного збереження

результатів тестування і безперебійної роботи системи. Python, у свою чергу, ідеально підходить для реалізації складних алгоритмів адаптації складності тестів та аналізу результатів, завдяки своїм потужним інструментам для обробки даних.

Таким чином, вибір цих двох мов дозволятиме забезпечити систему необхідною продуктивністю, надійністю та гнучкістю, що сприятиме ефективності навчання та адаптації під кожного користувача.

Таблиця 3.1 – Порівняльна характеристика мов програмування

Критерій	Java	JavaScript (Node.js)	Python	C# (.NET)	C++
Масштабованість	Висока	Висока	Середня	Висока	Висока
Продуктивність	Висока	Середня	Середня	Висока	Максимальна
Безпека	Висока	Середня	Середня	Висока	Низька
Об'єктно-орієнтованість	Висока	Середня	Висока	Висока	Висока
Підтримка баз даних	Висока (JDBC, Hibernate)	Висока (ORM, Sequelize)	Висока (SQL Alchemy)	Висока (Entity Framework)	Висока (SQL API)
Кросплатформність	Висока	Висока	Висока	Середня (.NET Core)	Висока
Підтримка GUI	Висока (JavaFX, Swing)	Висока (через Electron)	Середня (Tkinter, PyQt)	Висока (WPF, WinForms)	Висока (Qt, wxWidgets)
Легкість опанування	Середня	Висока	Висока	Середня	Низька
Ефективність інструментів	Висока (Spring, Maven)	Висока (npm, Express)	Висока (Django, Flask)	Висока (Visual Studio)	Середня
Підтримка та спільнота	Висока	Висока	Висока	Висока	Середня
Модульність	Висока	Висока	Висока	Висока	Низька

При розробці інформаційної технології тестування знань, Python та Java можуть бути використані для реалізації різних модулів, залежно від їх функціональних потреб та технічних характеристик.

Модулі, що можуть використовувати Python:

- Модуль аналізу результатів. Python добре підходить для аналізу даних,

особливо завдяки бібліотекам, як-от Pandas, NumPy, Scikit-learn. Цей модуль може аналізувати результати тестів, проводити статистичний аналіз та оцінювати прогрес користувача.

- Модуль адаптації складності. Адаптивні алгоритми, засновані на машинному навчанні чи інтелектуальних моделях, можуть бути реалізовані на Python за допомогою бібліотек, таких як TensorFlow або PyTorch. Python дуже популярний у сфері машинного навчання, що робить його ідеальним для адаптації складності тестів на основі результатів користувача.

- Модуль взаємодії з базою даних для аналітики. Python може використовуватися для вибірки даних з бази даних та проведення на їх основі аналізу, зокрема, з використанням ORM-інструментів, як SQLAlchemy.

- Модуль навчальних підказок. Python може бути корисним для розробки модуля, що надаватиме рекомендації користувачам щодо їх подальшого навчання на основі результатів тестів.

Модулі, що можуть використовувати Java:

- Модуль генерації тестів. Java добре підходить для створення структурованих тестів, включаючи генерацію запитань з бази даних та організацію їх в тести, з можливістю налаштування параметрів складності. Java дозволяє легко інтегруватися з базами даних через JDBC або інші інструменти.

- Модуль обробки відповідей. Java може обробляти введені користувачем відповіді, перевіряти їх на коректність та здійснювати відповідні обчислення (підрахунок балів). Також цей модуль може взаємодіяти з іншими модулями для визначення правильності відповідей.

- Модуль управління базами даних. Java може бути використана для прямої роботи з базою даних, включаючи збереження результатів тестування та управління тестовими матеріалами. Java добре інтегрується з SQL-базами даних, що спрощує розробку таких модулів.

- Модуль керування тестовими спробами. Java може також контролювати процес виконання тестів, зберігати прогрес та керувати кількістю спроб користувача. Цей модуль зможе також відслідковувати час, витрачений на тест.

Модулі, що можуть використовувати Python та Java спільно:

- Модуль зворотного зв'язку. Результати тестування можуть збиратися Java, але аналіз користувацьких відгуків може виконуватись на Python з використанням інструментів для обробки тексту або аналізу природної мови (наприклад, NLTK).

- API-інтерфейс для інтеграції модулів. Якщо різні модулі розробляються на різних мовах, можна створити API, через яке ці модулі будуть взаємодіяти. Java може використовуватися для основних функцій тестування, а Python – для аналізу результатів.

Отже, розподіл модулів між Python і Java дозволяє використовувати сильні сторони кожної мови: Java – для побудови основної архітектури та управління базою даних, а Python – для аналізу даних і адаптивних алгоритмів.

3.2 Обґрунтування вибору середовища програмування для розробки інформаційної технології тестування знань при вивченні іноземних слів

Для створення Java-додатків існує кілька варіантів середовищ розробки, таких як Eclipse, NetBeans та IntelliJ IDEA, які розповсюджуються з відкритим вихідним кодом [34]. Використання цих середовищ має певні переваги: по-перше, вони безкоштовні; по-друге, забезпечують можливість додавання нових функцій, що розширює їхні можливості; по-третє, завдяки спільноті розробників, яка може вносити поліпшення, ці IDE стають більш стабільними, оскільки зростає ймовірність виявлення та усунення помилок.

IntelliJ IDEA [35] – це потужне інтегроване середовище розробки (IDE), відоме своєю інтуїтивною підтримкою коду, зручним інтерфейсом і багатими можливостями для підвищення продуктивності розробників. Серед ключових переваг є глибока інтеграція з інструментами розробки та версійного контролю, що робить його ідеальним вибором для складних проєктів. IDEA також підтримує великий спектр плагінів і розширень, що дозволяє налаштовувати середовище під індивідуальні потреби. IDE автоматизує рутинні завдання,

зокрема рефакторинг коду, підказки та виправлення помилок у реальному часі.

Eclipse [36] – це класичне середовище для Java-розробки, яке надає стабільну й гнучку платформу з багатими функціями для великих проєктів. Eclipse вирізняється своєю відкритістю та потужною підтримкою плагінів, що робить його популярним серед розробників, які шукають високий ступінь кастомізації. Це середовище також є кросплатформним, що дозволяє розробляти програми на різних ОС. Хоча інтерфейс Eclipse менш інтуїтивний порівняно з IntelliJ IDEA, він залишається популярним завдяки потужній спільноті розробників.

NetBeans – це середовище розробки з відкритим вихідним кодом, яке часто використовується для розробки Java-додатків, особливо для веброзробки. Однією з головних особливостей є зручний інтерфейс і вбудовані інструменти для роботи з JavaFX, Swing, а також підтримка HTML5 і JavaScript. NetBeans забезпечує інтеграцію з серверними рішеннями, такими як Apache Tomcat і GlassFish, що полегшує розробку та деплой вебдодатків. Хоча воно дещо повільніше порівняно з іншими IDE, NetBeans пропонує надійні функції для інтегрованого розробника.

Таблиця 3.2 – Порівняльна таблиця середовищ розробки для Java

Характеристика	IntelliJ IDEA	Eclipse	NetBeans
Розробник	JetBrains	Eclipse Foundation	Oracle
Ліцензія	Community (безкоштовна), Ultimate (платна)	Відкритий код (EPL)	Відкритий код (GPL)
Підтримка плагінів	Широкий вибір	Велика бібліотека	Модульна структура
Інтерфейс	Інтуїтивний	Складний для новачків	Зручний для початківців
Підтримка мов	Java, Kotlin та інші	Java, C/C++, PHP та інші	Java, PHP, C/C++
Інтеграція з Git	Вбудована	Вбудована	Вбудована
Інтеграція з Maven/Gradle	Вбудована	Вбудована	Вбудована
Продуктивність	Висока	Висока	Середня
Цільова аудиторія	Професійні розробники	Різні користувачі	Студенти, початківці
Ціна	Безкоштовна та платна версія	Безкоштовна	Безкоштовна

Розглянемо також основні середовища розробки для Python (табл. 3.3).

PyCharm – це одне з найпопулярніших середовищ розробки для Python, що надає потужний набір інструментів для розробників. Його функціональність включає в себе автоматичне завершення коду, вбудовану підтримку тестування, інтеграцію з системами контролю версій, такими як Git, а також можливість роботи з віртуальними середовищами. PyCharm має розширені можливості для налагодження коду та аналізу якості, що робить його ідеальним вибором для середніх і великих проектів. Крім того, PyCharm підтримує численні плагіни, що дозволяє адаптувати середовище під специфічні потреби розробників.

Таблиця 3.3 – Порівняльна таблиця середовищ розробки для Python

Характеристика	PyCharm	Visual Studio Code (VS Code)	Jupyter Notebook	Spyder
Розробник	JetBrains	Microsoft	Project Jupyter	Anaconda
Ліцензія	Community (безкоштовна), Professional (платна)	Безкоштовна	Відкритий код	Відкритий код
Підтримка плагінів	Широкий вибір	Велика бібліотека	Обмежена	Обмежена
Інтерфейс	Інтуїтивний	Легкий і налаштовуваний	Інтерактивний	Зручний для наукових досліджень
Підтримка мов	Python та інші	Python та інші	Python	Python
Інтеграція з Git	Вбудована	Вбудована	Відсутня	Відсутня
Інтеграція з системами контролю версій	Вбудована	Вбудована	Відсутня	Відсутня
Продуктивність	Висока	Висока	Залежить від проекту	Висока
Цільова аудиторія	Професійні розробники	Різні користувачі	Науковці, студенти	Науковці, студенти
Ціна	Безкоштовна та платна версія	Безкоштовна	Безкоштовна	Безкоштовна

Visual Studio Code – це легке та універсальне середовище для кодування, яке підтримує Python через розширення. Воно надає базові можливості, такі як підсвічування синтаксису, автоматичне завершення та інтеграцію з Git. VS Code має велику бібліотеку плагінів, що дозволяє налаштувати середовище під конкретні потреби проекту. Це середовище підходить для розробки простих і середніх проектів, а також для інтеграції з веб-технологіями.

Jupyter Notebook – це інтерактивне середовище, яке особливо корисне для аналізу даних, наукових досліджень та візуалізації. Воно дозволяє розробникам писати код, виконувати його та документувати результати в одному місці. Jupyter підтримує різні мови програмування, але найбільше використовується для Python. Його можливості з візуалізації даних роблять його ідеальним для навчальних проектів, наукових досліджень і прототипування.

Spyder – це IDE, спеціально розроблене для наукових обчислень та аналізу даних. Воно включає в себе зручний інтерфейс, потужний редактор коду, інтегрований консольний інтерфейс та можливості для візуалізації даних. Spyder підходить для тих, хто займається науковими дослідженнями або обробкою даних, оскільки надає зручні інструменти для роботи з бібліотеками, такими як NumPy та pandas.

На основі проведеного аналізу вибір зупинився на використанні IntelliJ IDEA для Java та PyCharm для Python. Ці середовища розробки забезпечують оптимальні умови для ефективної реалізації інтелектуального модуля тестування знань, поєднуючи потужні функціональні можливості та зручний інтерфейс для розробників. Взаємодія між цими середовищами може відбуватися через API та модулі, які дозволяють обмінюватися даними і викликати функціонал одного середовища з іншого. Наприклад, можна розробити серверну частину на Java, яка оброблятиме запити, і клієнтську частину на Python, що виконуватиме аналітику та візуалізацію даних. Таким чином, використання IntelliJ IDEA та PyCharm у комбінації дозволяє реалізувати рішення, що поєднує переваги обох мов програмування, забезпечуючи ефективність і зручність в розробці програмних продуктів.

3.3 Обґрунтування вибору системи управління базою даних та її структури

Вибір системи управління базою даних (СУБД) є ключовим етапом у розробці інформаційної технології, оскільки він безпосередньо впливає на продуктивність, безпеку та масштабованість системи. Для нашої інформаційної технології, яка реалізує модулі для тестування знань при вивченні іноземних слів, обрано MySQL як основну СУБД.

MySQL є однією з найпопулярніших вільних та відкритих СУБД, яка забезпечує надійний та швидкий механізм для зберігання та обробки даних. Однією з основних причин вибору MySQL є її стабільність і продуктивність, що є критично важливими для обробки великих обсягів даних, які генеруються під час тестування. Крім того, MySQL підтримує транзакції, що забезпечує цілісність даних при виконанні складних операцій.

Важливою перевагою MySQL є її сумісність із двома обраними мовами програмування: Java і Python. MySQL має розширені драйвери та бібліотеки для обох мов, що забезпечує зручну інтеграцію та ефективну роботу з базою даних. Це дозволяє розробникам використовувати об'єктно-орієнтований підхід, спрощуючи управління даними та реалізацію бізнес-логіки.

Структура даних (табл. 3.4), яка була розроблена для підтримки функціональності інформаційної технології включає таблиці, що містять інформацію про користувачів, їхні відповіді, результати тестування та інші параметри, що є необхідними для подальшого аналізу.

Таблиця 3.4 – Структура даних training_test_results

Назва поля	Тип даних	Опис
1	2	3
result_id	INT (PK, AUTO_INCREMENT)	Унікальний ідентифікатор результату тестування
user_id	INT	Ідентифікатор користувача
test_id	INT	Ідентифікатор тесту
question_id	INT	Ідентифікатор запитання

Продовження таблиці 3.4

1	2	3
answer_given	VARCHAR(255)	Відповідь, яку дав користувач
is_correct	BOOLEAN	Прапорець, чи була відповідь правильною (1 = правильно, 0 = ні)
score	FLOAT	Набраний бал за питання (якщо є адаптивне тестування)
difficulty_level	VARCHAR(50)	Рівень складності запитання (легкий, середній, важкий)
time_taken	TIME	Час, витрачений на запитання
date_taken	DATE	Дата проходження тесту
total_score	FLOAT	Загальний бал за тест
test_duration	TIME	Час, витрачений на виконання всього тесту
completion_status	VARCHAR(20)	Статус завершення тесту (завершено/перервано)
test_attempt	INT	Номер спроби проходження тесту
user_feedback	TEXT	Коментарі або відгуки користувача про тест

3.4. Розробка UML-діаграми класів інформаційної технології тестування знань при вивченні іноземних слів

Створення UML-діаграм класів є важливим етапом у процесі розробки програмного забезпечення, оскільки вони забезпечують візуалізацію структури системи та взаємодії її компонентів.

UML-діаграми класів використовуються для визначення архітектури системи, її компонентів та їх взаємозв'язків, що є особливо важливим для інформаційних технологій, що реалізують адаптивне тестування. Діаграми можуть слугувати основою для автоматичної генерації початкового коду, що спрощує процес розробки та підвищує продуктивність.

Крім того, UML-діаграми класів є невід'ємною частиною технічної документації, що полегшує розуміння структури системи іншими розробниками. Вони дозволяють проводити аналіз системи на ранніх етапах розробки, виявляючи можливі проблеми в архітектурі, а також сприяють спільному розумінню структури системи серед усіх учасників проекту.

Такі діаграми вказують на можливості рефакторингу та оптимізації структури системи, а також дозволяють відобразити основні об'єктно-орієнтовані концепції, такі як спадкування, поліморфізм, інкапсуляція та інші, що є ключовими для забезпечення гнучкості та підтримки програмного забезпечення в майбутньому.

Для розробки інформаційної технології тестування знань при вивченні іноземних слів використовується низка класів, що відповідають за окремі функціональні модулі, такі як інтерфейс користувача, генерація тестів, обробка відповідей, адаптація складності та аналіз результатів (рис. 3.1):

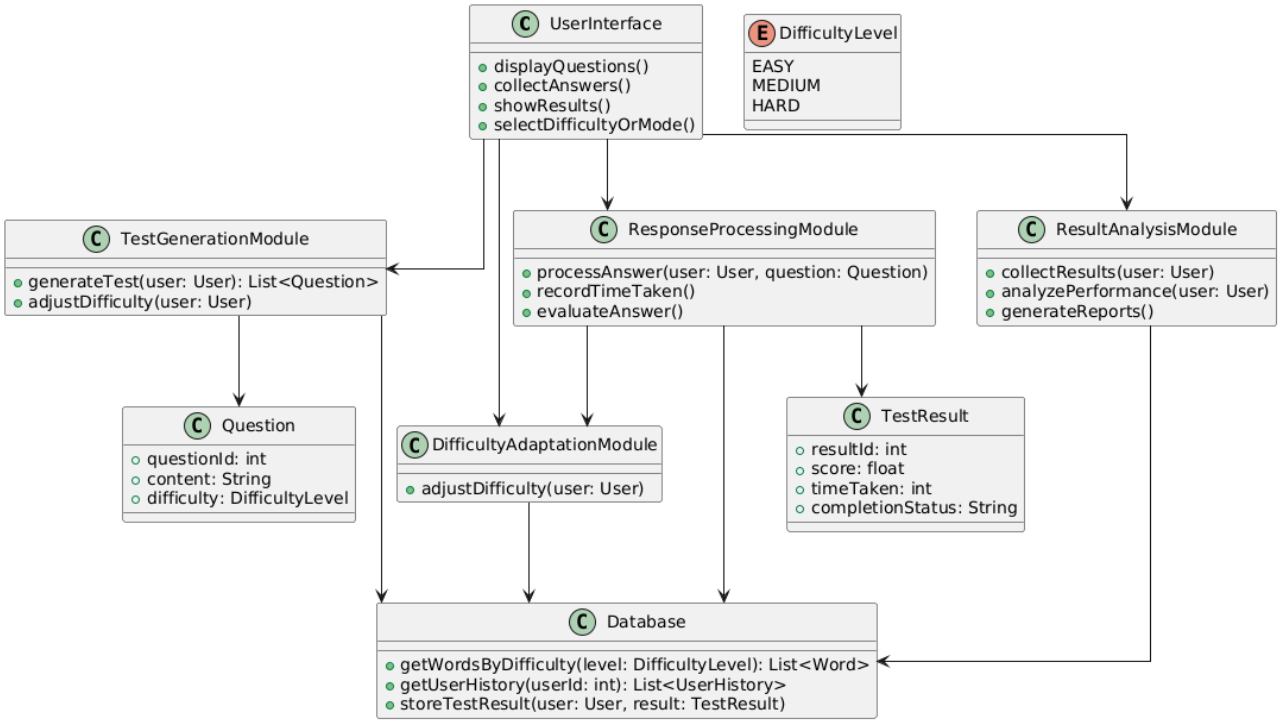


Рисунок 3.1 – UML-діаграма класів інформаційної технології тестування знань при вивченні іноземних слів

1. UserInterface – клас, що відповідає за взаємодію з користувачем, відображення тестових завдань, отримання відповідей, підсумкову візуалізацію результатів та вибір рівня складності або режиму тестування. Його основні функції включають:

- displayQuestions() – відображає питання тесту користувачу;

- collectAnswers() – збирає відповіді від користувача на запропоновані питання;

- showResults() – відображає результати тестування після його завершення

- selectDifficultyOrMode() – дозволяє користувачу вибрати рівень складності або режим тестування.

2. TestGenerationModule – клас, що відповідає за створення тестів. Основні функції включають:

- generateTest(user: User) – генерує тест для конкретного користувача на основі його історії та налаштувань;

- adjustDifficulty(user: User) – регулює складність тесту залежно від результатів попередніх тестувань користувача.

3. ResponseProcessingModule. Цей клас обробляє відповіді користувача. Його функції:

- processAnswer(user: User, question: Question) – обробляє відповідь користувача на конкретне питання, перевіряючи її на правильність;

- recordTimeTaken() – фіксує час, витрачений на виконання питання;

- evaluateAnswer() – оцінює правильність відповіді користувача, зберігаючи результати.

4. DifficultyAdaptationModule. Цей клас відповідає за динамічне налаштування рівня складності тестових завдань. Його функція:

- adjustDifficulty(user: User) – регулює складність завдань на основі результатів користувача, щоб забезпечити оптимальну адаптацію до його рівня знань.

5. ResultAnalysisModule. Цей клас аналізує результати тестування. Основні функції:

- collectResults(user: User) – збирає результати тестування користувача;

- analyzePerformance(user: User) – аналізує ефективність результату тестування користувача на основі отриманих результатів;

- generateReports() – створює звіти про результати тестування, які можуть бути використані для подальшого навчання.

6. Database – клас, що відповідає за управління даними. Основні функції:

- getWordsByDifficulty(level: DifficultyLevel) – отримує список слів відповідно до вибраного рівня складності;
- getUserHistory(userId: int) – отримує історію тестувань конкретного користувача;
- storeTestResult(user: User, result: TestResult) – зберігає результати тестування користувача в базі даних.

7. Question. Цей клас представляє питання тесту. Основні атрибути:

- questionId – ідентифікатор питання;
- content – текст питання;
- difficulty – рівень складності питання, що визначається через Difficulty Level.

8. TestResult представляє результати тестування. Основні атрибути:

- resultId – ідентифікатор результату тесту;
- score – оцінка, отримана користувачем за тест;
- timeTaken – час, витрачений на виконання тесту;
- completionStatus – статус завершення тесту (наприклад, "завершено", "незавершено").

9. DifficultyLevel – перелік, що визначає можливі рівні складності питань, які можуть бути:

- EASY – легкий рівень;
- MEDIUM – середній рівень;
- HARD – важкий рівень.

Класи взаємодіють один з одним через визначені методи та зв'язки:

- UserInterface взаємодіє з модулями генерації тестів, обробки відповідей, адаптації складності та аналізу результатів;
- TestGenerationModule, ResponseProcessingModule, DifficultyAdaptation Module та ResultAnalysisModuleотримують дані з класу Database.
- ResponseProcessingModule і TestGenerationModule працюють з класами Question та TestResult, обробляючи питання та результати тестування відповідно.

3.5 Розробка функціоналу інформаційної технології тестування знань при вивченні іноземних слів

Розробка бази користувачів та блоку ініціалізації. Першою дією, яку виконує користувач, є ініціалізація. Кожен користувач повинен пройти процес реєстрації, при цьому логін та пароль повинні бути коректними та унікальними. Зчитуються дані, які вводить користувач, і вони передаються до модуля, що обирає зв'язок із базою даних. Далі виконується пошук у базі даних для визначення, чи був користувач зареєстрований раніше. Якщо запис знайдено, користувача перенаправляють на сторінку реєстрації. У разі відсутності запису, нові дані записуються до бази даних. UML-діаграму послідовностей процесу ініціалізації наведено на рисунку 3.2.

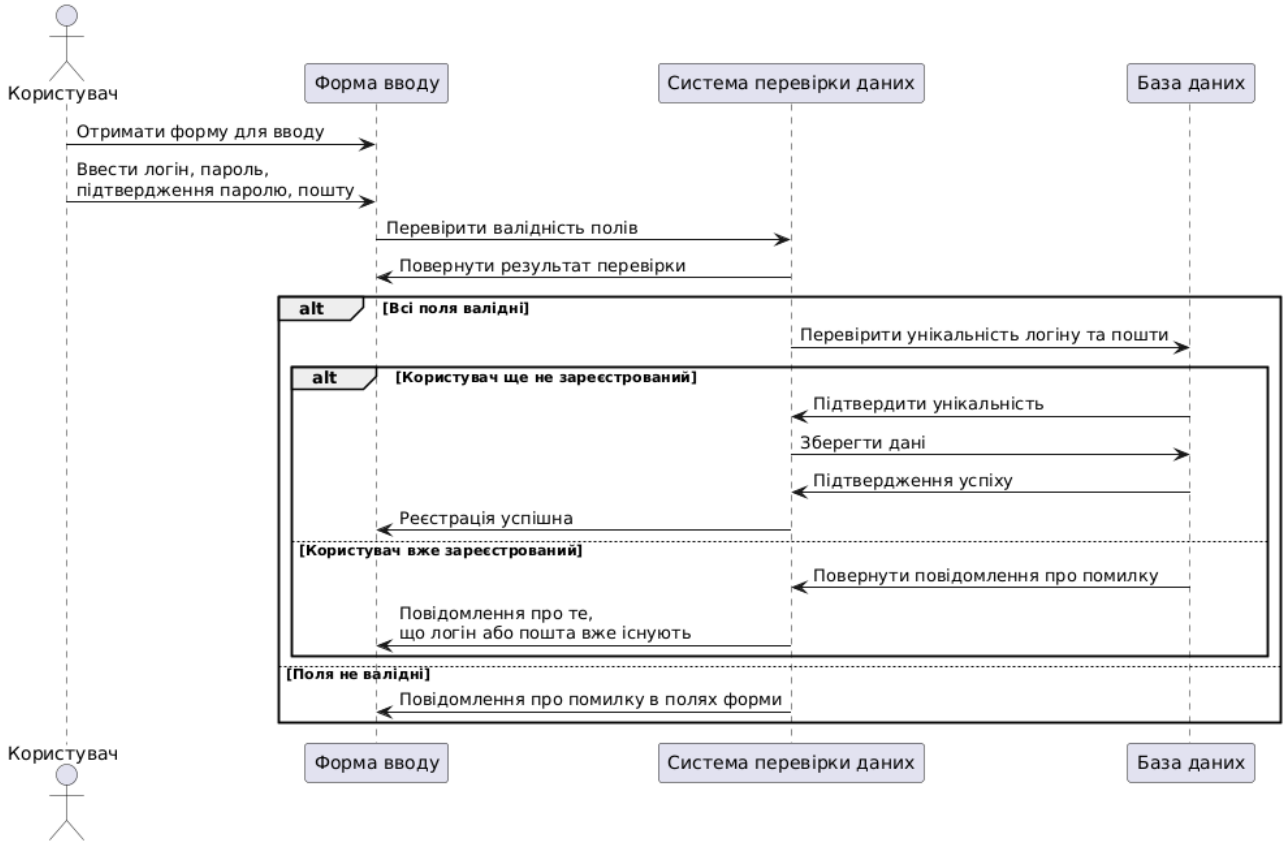


Рисунок 3.2 – UML-діаграма послідовностей процесу ініціалізації

Користувач отримає форму для вводу з полями для логіна, пароля, підтвердження паролю та пошти. Наступним іде перевірка полів форми на

валідність, якщо вона успішна, дані зберігаються в таблицю бази даних. Фрагмент лістингу функції запису користувача у базу даних наведено на рисунку 3.3.

```
public class RegistrListener implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        delete();
        User user = User.builder()
                .name(nameField.getText())
                .isAdmin(false)
                .username(usernameField.getText())
                .password(passwordField.getText())
                .build();
        try {
            User saved = userService.save(user);
            State.setUser(saved);
            menuFrame.create();
        }catch (IllegalArgumentException ex){
            JOptionPane.showMessageDialog(null, ex.getMessage());
            create();
        }

    }
}
```

Рисунок 3.3 – Фрагмент лістингу реєстрації нового користувача

Розробка блоку вибору тематики. На рисунку 3.4 зображено UML-діаграму функціонування системи по вибору завдання. Користувач визначається з темою та типом, який уособлює варіанти перевірки, це може бути або українська – англійська, або англійська – українська. Також користувачем визначається рівень складності слів у тесті. При умові коректного підбору характеристик, можливо дійти до кінцевої цілі, а саме ознайомлення, вивчення та перевірка знань, отриманих при проходженні тематики.

Налаштування тесту містять три поля з вибором одного значення. Після здійснення вибору, користувач натискає «Почати тест» та переходить до тестування. Лістинг коду зображений на рисунку 3.5.

Рисунок 3.4 – UML-діаграма функціонування системи по вибору завдання

```
@Override
public void create() {
    frame.setSize(400, 200);
    Set<String> topics = questionService.getTopics();

    topicBox = new JComboBox<>(topics.toArray(new String[]{}));
    topicBox.setFont(font);
    label.add(topicBox, new GridBagConstraints(0, 0, 1, 1, 0.9, 0.9, GridBagConstraints.NORTH,
            GridBagConstraints.HORIZONTAL, new Insets(8, 10, 8, 10), 0, 0));


    typeBox = new JComboBox<>(TYPES.toArray(new String[]{}));
    typeBox.setFont(font);
    label.add(typeBox, new GridBagConstraints(0, 1, 1, 1, 0.9, 0.9, GridBagConstraints.NORTH,
            GridBagConstraints.HORIZONTAL, new Insets(8, 10, 8, 10), 0, 0));

    levelBox = new JComboBox<>(LEVEL.toArray(new String[]{}));
    levelBox.setFont(font);
    label.add(levelBox, new GridBagConstraints(0, 2, 1, 1, 0.9, 0.9, GridBagConstraints.NORTH,
            GridBagConstraints.HORIZONTAL, new Insets(8, 10, 8, 10), 0, 0));


    JButton testButton = new JButton();

    addButton("Почати тест", 3, new StartListener(), testButton);
```

Рисунок 3.5 – Фрагмент лістингу коду здійснення налаштування тесту

Модуль генерації тестів. На рис. 3.6 наведено UML-діаграму послідовностей для модуля генерації тестів, що описує взаємодію між користувачем, модулем генератора тестів і базою даних, що містить тестові питання.

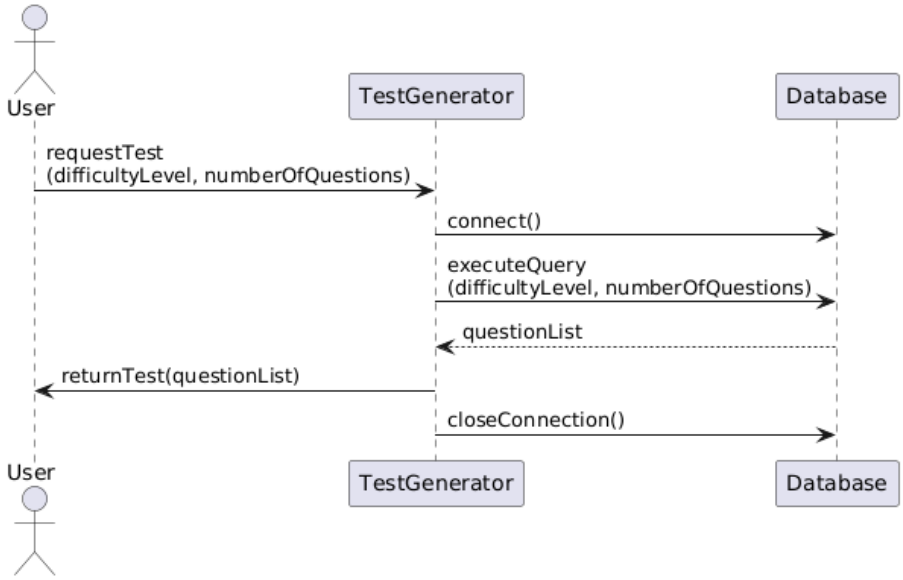


Рисунок 3.6 – UML-діаграма послідовностей для модуля генерації тестів

Процес починається з того, що Користувач (User) звертається до модуля генерації тестів (TestGenerator) із запитом на тест, вказуючи рівень складності завдань і бажану кількість питань (в програмному коді на рис. 3.7 це реалізується методом requestTest). Після цього модуль генерації тестів ініціює з'єднання з базою даних за допомогою connect() і виконує SQL-запит executeQuery, який здійснює вибірку запитуваних питань відповідного рівня складності. Цей запит випадковим чином обирає питання, що відповідають зазначеним параметрам (реалізовано у методі getQuestions). Після виконання запиту база даних (Database) повертає список вибраних питань до модуля генерації тестів (в коді questionList).

Модуль генерації тестів далі передає цей список питань користувачу у вигляді завершеного тесту та відразу ж закриває підключення до бази даних за допомогою методу closeConnection().

Підключення до бази даних: здійснюється за допомогою DriverManager, де "jdbc:mysql://localhost:3306/test_db" – це шлях до бази даних, а user та password– облікові дані.

Метод getQuestions вибирає випадкові питання (ORDER BY RAND()) із зазначеним рівнем складності (difficultyLevel). Кількість питань обмежена значенням numberOfQuestions. Метод close закриває з'єднання з базою даних.

```
public TestGenerator() {
    try {
        connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/test_db", "user", "password");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public List<String> getQuestions(int difficultyLevel, int numberOfQuestions) {
    List<String> questions = new ArrayList<>();
    try {
        String query = "SELECT question_text FROM questions WHERE difficulty = ? ORDER BY RAND() LIMIT ?";
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setInt(1, difficultyLevel);
        statement.setInt(2, numberOfQuestions);
        ResultSet resultSet = statement.executeQuery();

        while (resultSet.next()) {
            questions.add(resultSet.getString("question_text"));
        }

        resultSet.close();
        statement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return questions;
```

Рисунок 3.7 – Фрагмент лістингу модуля генерації тестів

Модуль обробки відповідей. У модулі обробки відповідей реалізується логіка перевірки відповідей користувача, обчислення балів та фіксації результатів у базі даних (рис. 3.8). Модуль виконує такі основні функції: приймає дані про відповідь користувача, правильну відповідь, рівень складності питання та час виконання.

Загальний бал тесту оновлюється шляхом додавання отриманого балу до поточного значення. Незалежно від правильності відповіді, результати фіксуються в базі даних. Для цього виконується запис даних у таблицю answer_log, де зберігаються інформація про отримані бали та час, витрачений на відповідь у мілісекундах. Це дозволяє здійснювати подальший аналіз результатів та використовувати дані для навчання моделей.

Модуль надає можливість користувачеві отримати загальний бал тестування через метод getTotalScore, який повертає підсумковий результат після виконання всіх питань.

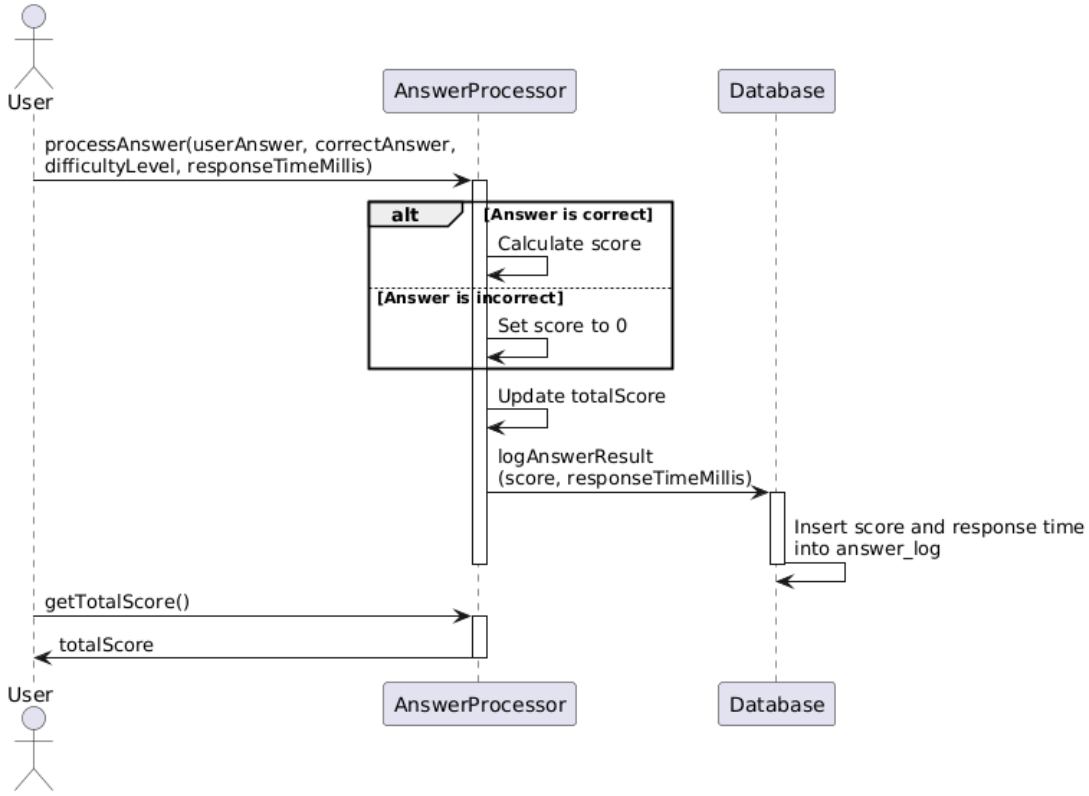


Рисунок 3.8 – UML-діаграма послідовностей для модуля обробки відповідей

```java
public class AnswerProcessor {
    private int totalScore = 0;

    public int processAnswer(String userAnswer, String correctAnswer, int difficultyLevel, long
responseTimeMillis) {
        int score = 0;
        if (userAnswer.equalsIgnoreCase(correctAnswer)) {
            switch (difficultyLevel) {
                case 1:
                    score = 0.7;
                    break;
                case 2:
                    score = 0.85;
                    break;
                case 3:
                    score = 1;
                    break;
                default:
                    score = 0;
                    break;
            }
        }
        totalScore += score;
        logAnswerResult(score, responseTimeMillis);
        return score;
    }
    private void logAnswerResult(int score, long responseTimeMillis) {
        String url = "jdbc:mysql://localhost:3306/test_db";
```

Рисунок 3.9 – Фрагмент лістингу модуля обробки відповідей

На рис. 3.10. наведено UML-діаграму послідовностей модуля адаптації складності, яка демонструє взаємодію між користувачем, адаптером складності та системою. Напочатку користувач надає свою оцінку, яка передається до системи. Після чого система викликає метод adapt_difficulty(score) в модулі DifficultyAdapter, щоб адаптувати рівень складності на основі отриманої оцінки.

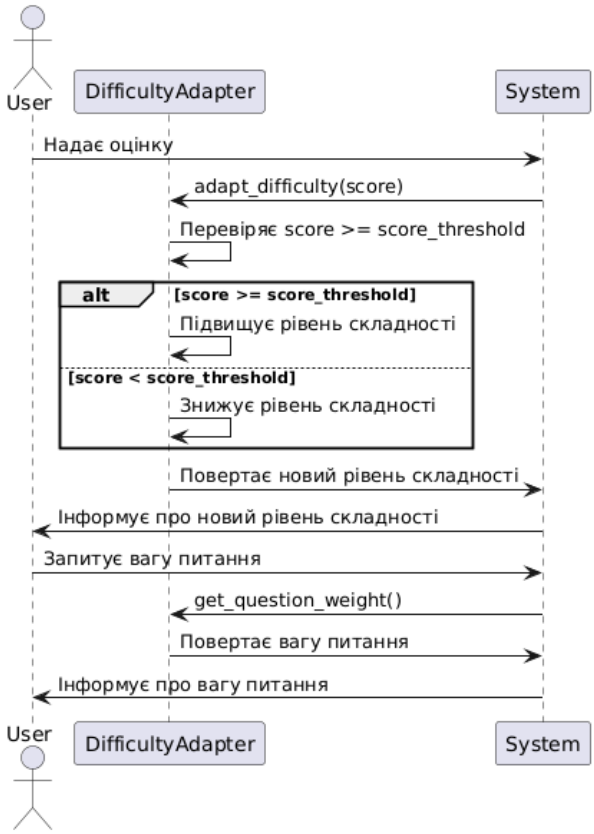


Рисунок 3.10 – UML-діаграма послідовностей модуля адаптації складності

У класі DifficultyAdapter (рис. 3.11) перевіряється, чи перевищує оцінка визначений поріг (змінна score_threshold). Якщо оцінка дорівнює або перевищує цей поріг, адаптер підвищує рівень складності; якщо ж оцінка нижча, рівень складності знижується. Після цього адаптер повертає новий рівень складності до системи, яка, в свою чергу, інформує користувача про зміни у рівні складності. Наступним кроком є запит користувача про вагу питання. Система знову взаємодіє з адаптером, викликаючи метод get_question_weight(), для отримання відповідної ваги. Адаптер передає цю вагу назад до системи, яка інформує користувача про результати.

```
class DifficultyAdapter:
    def __init__(self):
        self.current_difficulty = "medium"
        self.score_threshold = 0.6
        self.difficulty_levels = {
            "easy": 0.7,
            "medium": 0.85,
            "hard": 1.0
        }
    def adapt_difficulty(self, score):
        if score >= self.score_threshold:
            if self.current_difficulty == "easy":
                self.current_difficulty = "medium"
            elif self.current_difficulty == "medium":
                self.current_difficulty = "hard"
        else:
            if self.current_difficulty == "hard":
                self.current_difficulty = "medium"
            elif self.current_difficulty == "medium":
                self.current_difficulty = "easy"
        return self.current_difficulty
    def get_question_weight(self):
        return self.difficulty_levels[self.current_difficulty]
```

Рисунок 3.11 – Фрагмент лістингу модуля адаптації складності

Модуль аналізу результатів побудований навколо ключового класу ResultsAnalyzer, який відповідає за обробку відповідей користувача, обчислення інтегральної оцінки кожного питання, а також запис результатів у базу даних для подальшого аналізу та адаптації складності тесту.

Початково, метод processAnswer отримує відповідь користувача, правильну відповідь, рівень складності питання і час виконання. Цей метод визначає, чи є відповідь правильною, і на цій основі обчислює бал. Якщо відповідь неправильна, то інтегральний результат всього тесту (змінна totalScore) встановлюється на 0. У випадку правильної відповіді використовуються коефіцієнти coef1 і coef2 (зі значеннями 0.6 і 0.4 відповідно) для формули:

$$score = coef1 * settings['weight'] + coef2 * efficiency$$

де settings містить специфічні характеристики для кожного рівня складності (наприклад, weight – вага для відповідного рівня, яка коригує оцінку відповідно до складності), а efficiency базується на обчисленій швидкості відповіді.

Наступний метод, updateDifficulty, базуючись на підсумковому балі score (чи він перевищує поріг у 0.6), визначає, чи буде наступне питання легшим або важчим за попереднє. Якщо оцінка нижча за 0.6, система знижує складність, інакше підвищує її.

Метод logResults записує результат у базу даних. Він виконує збереження відповіді, часу виконання та обчисленого балу для кожного питання у таблицю answer_log. Таким чином, система накопичує історичні дані для кожного користувача, що дає можливість проводити детальний аналіз результатів. Ці дані також використовуються в інших модулях для налаштування майбутніх тестів і покращення адаптивності.

Діаграма послідовностей для цього модуля (рис. 3.12) відображає взаємодію між компонентами: починаючи з отримання відповіді, обчислення оцінки, прийняття рішення щодо зміни рівня складності, і закінчуючи записом результатів у базу даних. Фрагмент коду наведено на рис. 3.13.

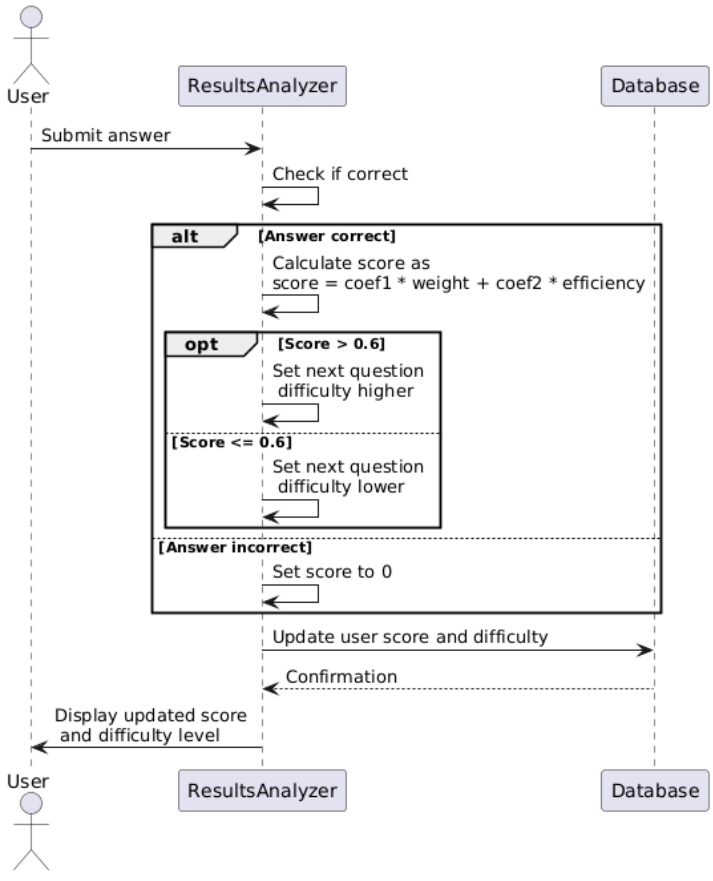


Рисунок – 3.12 UML-діаграма послідовностей модуля аналізу результатів

```
class ResultsAnalyzer:
        def calculate_score(self, difficulty, is_correct, efficiency):
        if not is_correct:
            return 0
        weight = self.settings[difficulty]['weight']
        score = self.coef1 * weight + self.coef2 * efficiency
        return score

        def determine_next_difficulty(self, score, current_difficulty):
        if score > self.threshold_score:
            if current_difficulty == 'simple':
                return 'medium'
            elif current_difficulty == 'medium':
                return 'hard'
        else:
            if current_difficulty == 'hard':
                return 'medium'
            elif current_difficulty == 'medium':
                return 'simple'
        return current_difficulty
```

Рисунок 3.13 – Фрагмент лістингу модуля аналізу результатів

Отже, в результаті розробки функціоналу інформаційної технології тестування знань при вивченні іноземних слів було створено основні функціональні модулі, які тісно взаємодіють між собою та формують цілісну адаптивну систему. Її впровадження дозволить ефективно здійснювати процес тестування знань, враховуючи рівень складності завдань, персональні результати користувачів та гнучко адаптуючи тестові матеріали відповідно до індивідуальних потреб. Розроблена технологія забезпечує зручність, точність оцінювання та високу продуктивність, що сприятиме загальній оптимізації навчального процесу.

3.6 Тестування та налаштування параметрів роботи інформаційної технології тестування знань при вивченні іноземних слів

Для перевірки основних функціональних можливостей інформаційної технології тестування знань при вивчення іноземних слів було проведено серію тестувань з подальшим налаштуванням параметрів інтегральної оцінки.

Початковим етапом є авторизація користувача. У разі введення некоректних даних або залишення полів порожніми, система генерує

повідомлення про невідповідність даних (рис. 3.14).

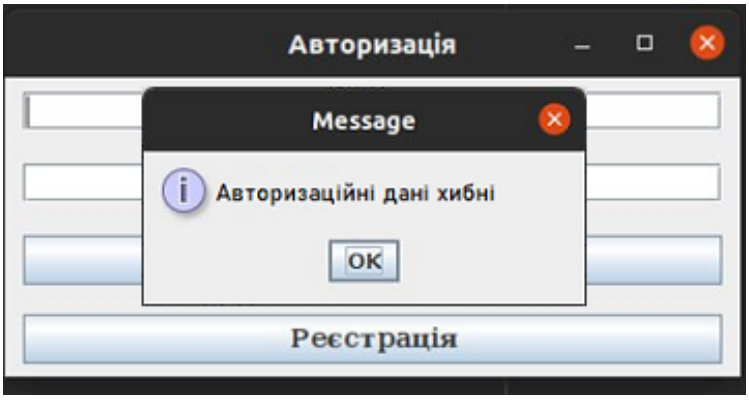


Рисунок 3.14 – Повідомлення про помилку при введенні хибних даних

Після успішної авторизації користувач переходить до головного меню, звідки можна обрати розділи для доступу до тестування, аналізу помилок, індивідуального словника та статистики (рис. 3.15). У розділі тестування передбачена можливість налаштування параметрів тесту, що є однією з додаткових функціональних можливостей. Це налаштування сприяє оптимізації процесу вивчення іноземних слів, дозволяючи адаптувати тестування під індивідуальні потреби користувача.

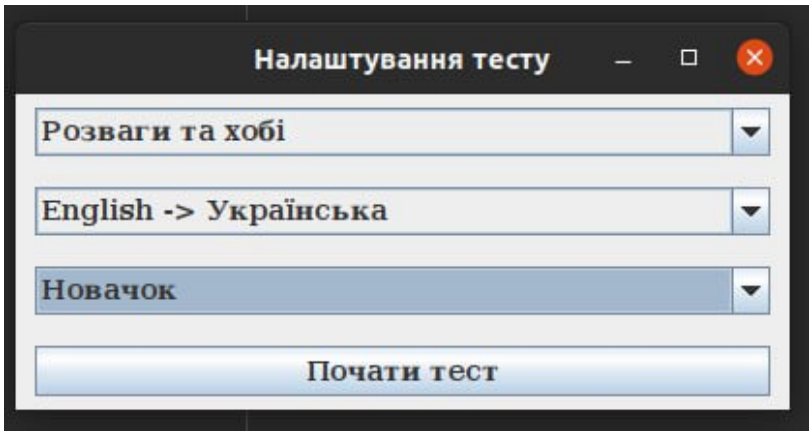


Рисунок 3.15 – Налаштування тесту зі складністю «Новачок»

Під час виконання тестових завдань проводиться як оцінка правильності відповідей, так і реєстрація часу, витраченого на кожне завдання, у базі даних. На наступному етапі отримані дані виступатимуть основою для статистичного

аналізу та налаштування параметрів інтегральної оцінки знань користувача.

Встановлення нормативного часу для проходження тестів різних рівнів складності здійснюється на основі аналізу відповідних гістограм (рис. 3.16) та коробкових графіків (рис. 3.17). Наведені графіки дають змогу оцінити розподіл часу для кожної категорії питань, а також виявити можливі аномальні відхилення у часі виконання завдань. Зокрема, виявлені аномалії слугуватимуть додатковими індикаторами для перевірки коректності розроблених тестових завдань.

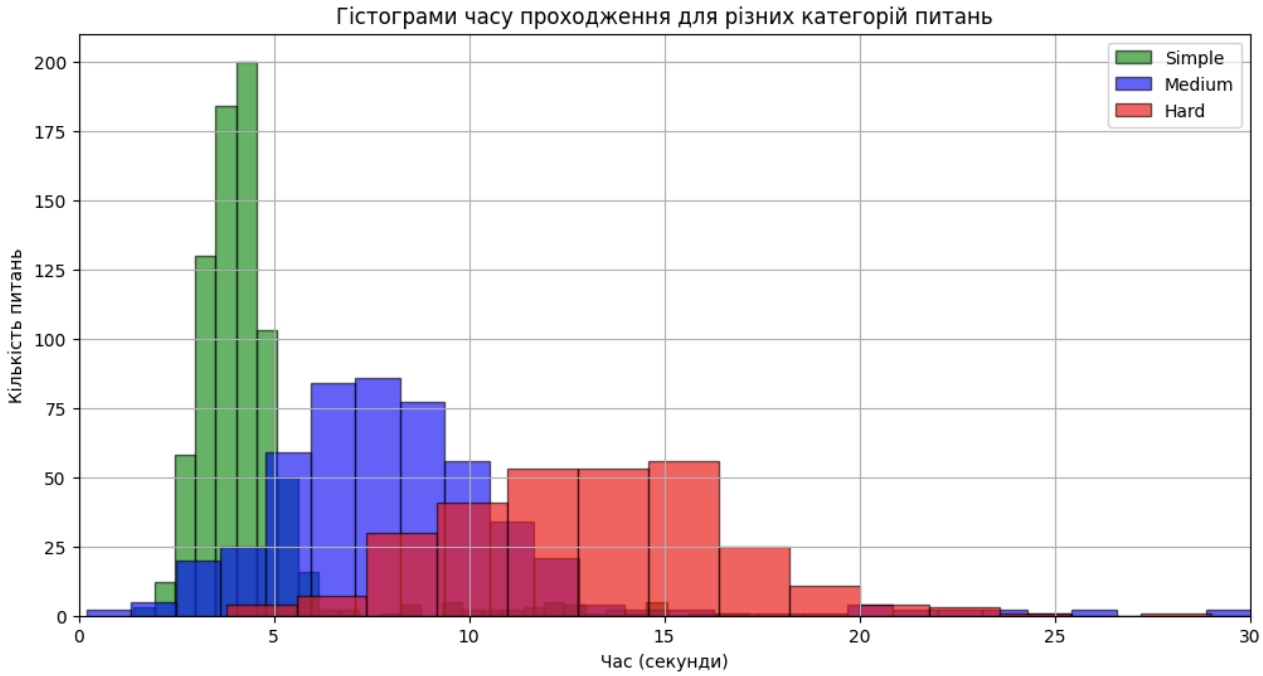


Рисунок 3.16 – Гістограми часу проходження тестів для різних категорій питань

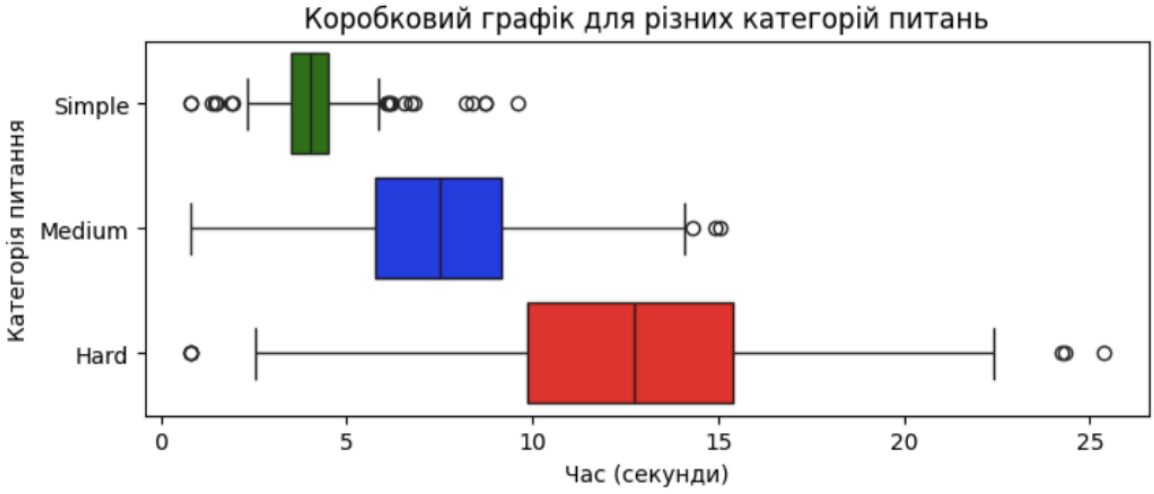


Рисунок 3.17 – Коробковий графік часу проходження тестів

За аналізом представлених графіків (рис. 3.16 та рис. 3.17) було прийнято рішення встановити нормативний час проходження тестів різної категорії питань на рівні 25%-го (для простих питань), 50%-го (для питань середньої складності) та 75%-го (для складних питань) процентилів. Для тестів, час виконання яких перевищував встановлений норматив, загальна ефективність проходження відповідно знижувалася, що впливало на підрахунок бальної оцінки (рис. 3.18). При цьому вагові коефіцієнти формули (2.1), що визначають важливість часу та точності обрано на рівні 0,6 та 0,4, а коефіцієнт що відповідає за складність питань дорівнює 1 для складного рівня; 0,85 – для середнього та 0,7 – для простого. Поріг ефективності проходження тесту, визначено на рівні 0,8 після проведення серії тестувань.

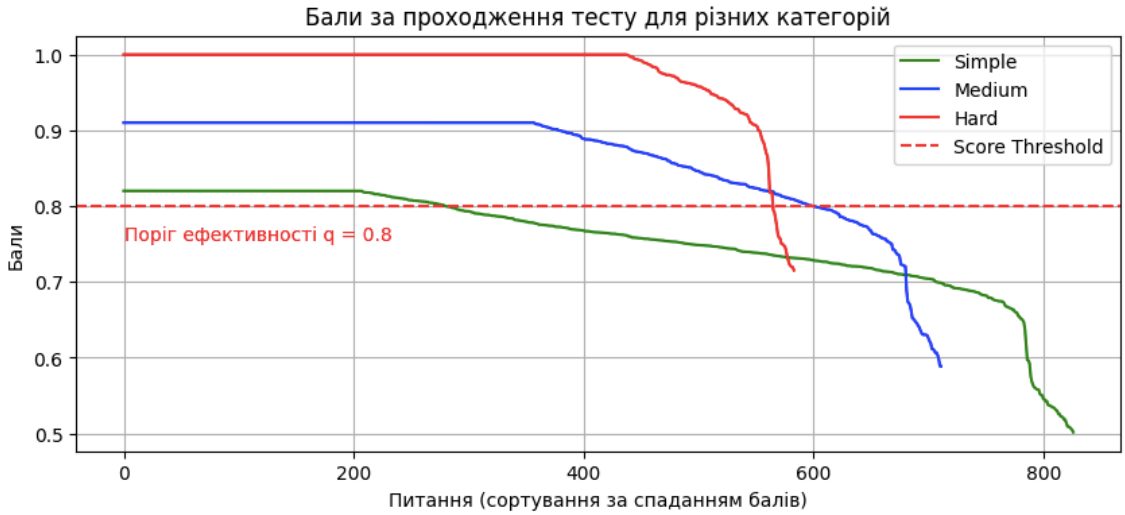


Рисунок 3.18 – Залежність інтегральної оцінки тесту від часу виконання для різних категорій питань

На рис. 3.19 наведено загальний результат тестування, проведений на основі проходження користувачем 50 тестів, для визначених вище параметрів розробленої інформаційної системи. Як бачимо, робота системи є коректною. Підвищення рівня складності наступного питання відбувається лише після перевищення порогу ефективності. І навпаки, складність питань зменшується при недосяжності порогу ефективності. Затримки у часі виконання тестового завдання впливають на розрахунок кількості балів.

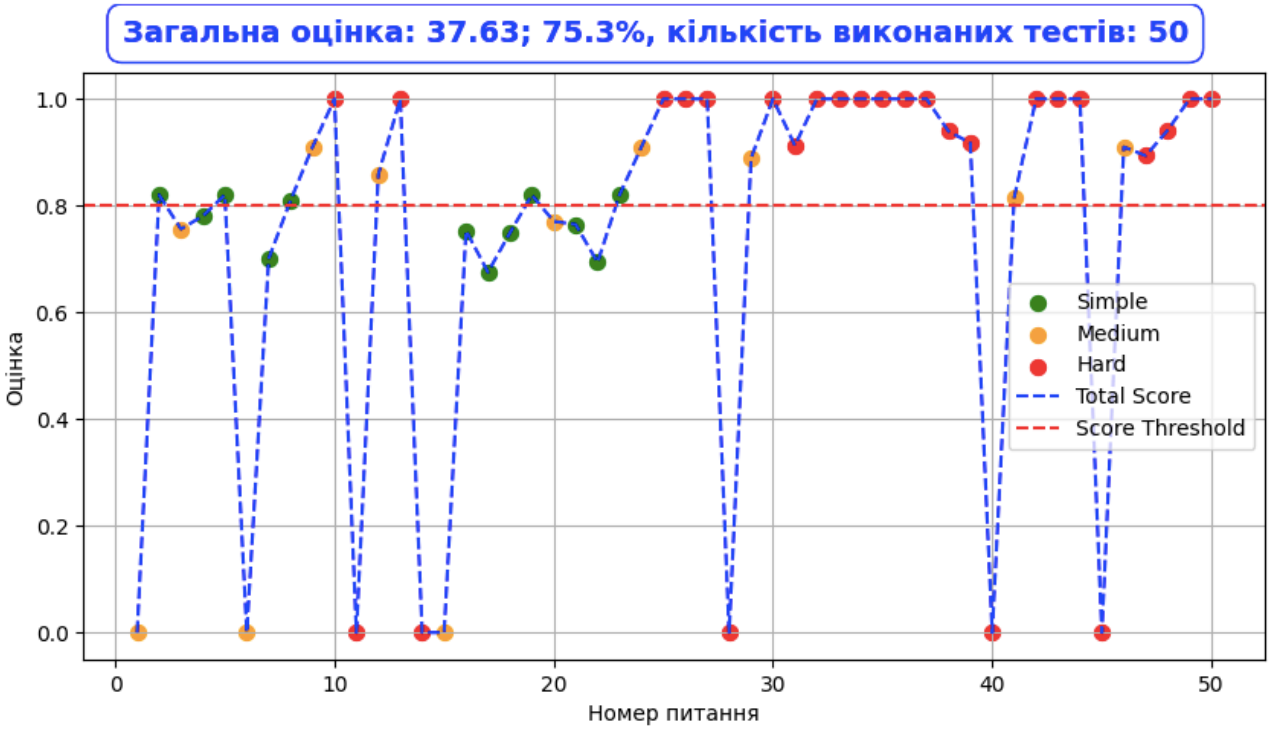


Рисунок 3.19 – Розрахунок інтегральної оцінки тестування

У таблиці 3.5. наведено результати порівняння функціоналу розробленої інформаційної технології тестування знань при вивченні іноземних слів з найпопулярнішими системами.

Таблиця 3.5 – Порівняння функціоналу розробленої інформаційної технології тестування знань з конкурентними системами

Функціонал	Розроблена IT	Duolingo	Busuu	Fun Easy Learn	Quizlet
Адаптація складності питань	+	+	+	–	–
Модуль аналізу результатів	+	–	+	–	+
Автоматичний вибір питань відповідно до рівня складності	+	+	+	–	–
Модуль налаштування коефіцієнтів інтегральної оцінки	+	–	–	–	–
Врахування часу виконання для кожного питання	+	–	–	–	+
Гнучкість у налаштуванні параметрів тесту	+	–	+	+	+
Можливість персоналізації на основі прогресу	+	+	+	–	+
Інтерфейс для аналізу даних результатів	+	–	+	–	+

Результати порівняння (табл. 3.5) підкреслюють розширені можливості розробленої інформаційної технології, досягнуті завдяки впровадженню модуля налаштування коефіцієнтів інтегральної оцінки та врахування часу виконання кожного завдання. Зазначені функціональні характеристики забезпечують системі вищий рівень гнучкості та адаптивності порівняно з іншими популярними аналогами. Зокрема, розроблена інформаційна система містить 8 параметрів для функціоналу тестування знань при вивченні іноземних слів, що на 33% перевищує аналогічні показники найближчого конкурента – платформи Busuu, яка має лише 6 параметрів.

3.7 Висновок до розділу 3

У даному розділі обґрунтовано вибір мов програмування, інтегрованого середовища розробки та засобів обробки даних для реалізації інформаційної технології тестування знань при вивченні іноземних слів. Завдяки поєднанню двох мов програмування (Java та Python) вдалося досягти оптимальної взаємодії між модулями, що забезпечило високий рівень продуктивності та можливість реалізації широкого функціоналу.

Створено UML-діаграму класів для програмного забезпечення системи тестування знань, яка описує взаємодію основних модулів, таких як генерація питань, обробка відповідей, адаптація складності та аналіз результатів тестування.

Здійснено програмну реалізацію всіх модулів системи та проведено тестування, яке підтвердило їхню коректність та ефективність у процесі роботи.

Порівняльний аналіз розробленої інформаційної технології з існуючими аналогами відзначає розширення її функціональних можливостей завдяки впровадженню модуля налаштування коефіцієнтів інтегральної оцінки та врахуванню часу виконання кожного завдання.

4 ЕКОНОМІЧНА ЧАСТИНА

Для успішного впровадження інформаційної технології тестування знань при вивченні іноземних слів важливо враховувати її відповідність сучасним вимогам науково-технічного прогресу. Однак слід зазначити, що дана розробка відноситься до науково-дослідної роботи пошукового характеру, а тому оцінювання її ефективності є складним процесом, який часто базується на експертних оцінках і має ймовірнісний характер. Щодо економічної ефективності впровадження таких розробок зазвичай мова взагалі не ведеться.

З огляду на специфіку цієї роботи, ключовими аспектами є науково-технічна значущість і потенціал впровадження розробки. Інформаційна технологія здатна забезпечити адаптацію завдань відповідно до рівня знань користувача, сприяти підвищенню мотивації до навчання та оптимізувати процес оцінювання результатів. Це робить її перспективною для використання як у закладах освіти, так і серед окремих користувачів.

Розглядаючи перспективи подальшого впровадження та розвитку інформаційної технології, необхідно провести оцінювання її важливості та наукової значимості, що включатиме технологічний аудит та розрахунок загальних витрат на здійснення науково-технічної розробки.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Оцінку науково-технічного рівня та комерційного потенціалу пропонованої інформаційної технології тестування знань при вивченні іноземних слів проведемо експертним методом з використанням п’ятибальної системи оцінювання за дванадцятьма критеріями, наведеними у [37, с.12].

Для опитування було залучено експертів з ВНТУ, кафедри комп’ютерних наук: к.т.н., доцента Арсенюка Ігоря Ростиславовича, к.т.н., доцентку Крилик

Людмилу Вікторівну, к.т.н., професора Колесницького Олега Костянтиновича. Результати проведеного оцінювання наведено у таблиці 4.1.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Арсенюк І. Р., доцент	Крилик Л.В., доцент	Колесницький О.К., професор
	Бали:		
1	2	3	4
1. Технічна здійсненність концепції	4	3	4
2. Ринкові переваги (наявність аналогів)	2	3	2
3. Ринкові переваги ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	2	2	3
5. Ринкові переваги (експлуатаційні витрати)	4	4	3
6. Ринкові перспективи (розмір ринку)	3	3	2
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	3
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	$СБ_1 = 41$	$СБ_2 = 42$	$СБ_3 = 39$
Середньоарифметична сума балів $СБ_с$	$СБ_с = 40,67$		

Згідно з проведеними дослідженнями, науково-технічний потенціал пропонованої інформаційної технології тестування знань при вивченні іноземних слів оцінюється у 40,67 бала. Відповідно до рекомендаційної таблиці визначення наукового рівня [37], це свідчить про її високу значущість. Такий високий рівень потенціалу обґрунтовує доцільність її практичного впровадження, що, зокрема, досягнуто завдяки розширенню функціональних можливостей нової інформаційної системи порівняно з аналогами, що існують на ринку на даний час.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи за темою «Інформаційна технологія тестування знань при вивченні іноземних слів» включатимуть витрати на оплату праці; відрахування на соціальні заходи; матеріали; паливо та енергія для науково-виробничих цілей; спецустаткування для наукових (експериментальних) робіт; програмне забезпечення для наукових (експериментальних) робіт; витрати на роботи, які виконують сторонні підприємства, установи і організації; інші витрати; накладні (загальновиробничі) витрати.

Стаття «Витрати на оплату праці» включає основну та додаткову заробітну плату керівників відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми. ЇЇ розмір обчислюється за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо, за формулою [37]:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = 22 дні; t_i – число робочих днів роботи розробника (дослідника). Результат розрахунків подано в таблиці 4.2.

Таблиця 4.2 – Витрати на основну заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проєкту	35000,00	1590,90	10	15909,00
Інженер-програміст	25000,00	1136,36	30	34090,9
Консультант-аналітик	20000,00	909,09	10	9090,9
Всього:	59090,80			

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховуємо за формулою [37]:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; t_i – час роботи робітника на виконання певної роботи, год.

Розрахуємо погодинну тарифну ставку робітника за формулою [37]:

$$C_i = \frac{М_м \cdot К_i \cdot К_с}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ = 8000 грн – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати; $К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ = 1,65 – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 22 дні; $t_{зм}$ – тривалість зміни, год.

Результат проведених розрахунків подано в таблиці 4.3.

Таблиця 4.3 – Витрати на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
1. Підготовчі	10	2	82,50	825,00
2. Налагоджувальні	10	4	112,50	1125,0
3. Випробувальні	3	3	101,25	303,75
Всього				2253,75

Додаткову заробітну плату дослідників та робітників розраховуємо в межах 10% від суми основної заробітної плати за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%} = (59090{,}80 + 2253{,}75) \cdot \frac{10\%}{100\%} = 6134{,}5 \text{ грн},$$

де $Н_{дод} = 10\%$ – норма нарахувань додаткової заробітної плати.

Розрахуємо також розмір відрахувань на соціальні заходи, до яких належать відрахування внеску на загальнообов'язкове державне соціальне страхування та внеску для здійснення заходів щодо соціального захисту населення [37]:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} = 14845{,}40 \text{ грн},$$

де $Н_{зп} = 22\%$ – норма нарахування на заробітну плату.

Розрахуємо витрати на сировину та матеріали. До цієї статті витрат належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їхнього витрачання.

У вартісному вираженні витрати на матеріали (*М*) розраховуємо окремо для кожного виду матеріалів за формулою [37]:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}\ , \qquad (4.4)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг.; n – кількість видів матеріалів; $Ц_j$ – вартість матеріалу j-го найменування, грн/кг; $К_j$ – коефіцієнт транспортних витрат, $К_j$ = (1,1…1,15); $В_j$ – маса відходів *j*-го найменування, кг; $Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

Результат розрахунків наведено в таблиці 4.4.

Таблиця 4.4 – Матеріали, що використані на розробку

Найменування матеріалу, марка, тип, сорт	Ціна за 1 од	Норма витрат, од	Вартість витраченого матеріалу, грн.
Офісний папір А4	205,00	1	205,00
Кулькова ручка	11,00	3	33,00
Флеш пам'ять Goodram Twister 64 GB (0,02 гр.)	198,2	2	396,40
Всього			634,40
З врахуванням коефіцієнта транспортування			697,84

Спецустаткування для наукових (експериментальних) робіт. До цієї статті належать витрати на виготовлення та придбання спецустаткування, необхідного для проведення досліджень, також витрати на їхнє проектування, виготовлення, транспортування, монтаж та встановлення.

Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10...12% від вартості устаткування.

Балансову вартість спецустаткування розрахуємо за формулою:

$$В_{спец} = \sum_{i=1}^{k} Ц_i \cdot С_{пр.i} \cdot К_i, \qquad (4.5)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн; $С_{пр.i}$ – кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.; $К_i$ – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($К_i$ = 1,10...1,12); k – кількість найменувань устаткування.

Результати розрахунків наведено у таблиці 4.5.

Таблиця 4.5 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, од	Ціна за одиницю, грн	Вартість, грн.
Сервер ARTLINE Business T17v23	1	29049,00	31953,90
Ноутбук HP 250-G10 (85A10EA)	1	27799,00	30578,90
Всього			62532,80

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, а також витрати на їх проектування, формування та встановлення в розмірі 10…12% від вартості програмного забезпечення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{прг.i} \cdot С_{прг.i} \cdot К_i, \quad (4.6)$$

де $Ц_{прг.i}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.i}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_i$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_i$ = (1,1…1,12); k – кількість видів програмного забезпечення.

Результат розрахунків наводимо у таблиці 4.6.

Таблиця 4.6 – Витрати на придбання програмних засобів по кожному виду

Найменування устаткування	Кількість, од	Ціна за одиницю, грн	Вартість, грн.
Програмне забезпечення SQLite server	1	9870,00	10857,00
Програмне забезпечення Vite server	1	9150,00	10065,00
Всього			20922,00

Амортизацію обладнання, комп’ютерів та приміщень (А), що використовувалися під час (чи для) виконання даного етапу роботи розраховуємо за спрощеною за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12}, \quad (4.7)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_в$ – термін корисного використання основного фонду, роки. Відповідні розрахунки наводимо у таблиці 4.7.

Таблиця 4.7 – Амортизаційні відрахування за видами основних фондів

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Сервер ARTLINE Business T17v23	31953,90	10	1	266,30
Ноутбук HP 250-G10 (85A10EA)	30578,90	6	1	424,70
Всього	691,00			

Розрахуємо витрати на паливо та енергію ($В_е$) для науково-виробничих цілей за формулою:

$$В_е = \sum_{i=1}^{k} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot К_{впі}}{\eta_i} \cdot, \quad (4.8)$$

де $В_е$ – встановлена потужність обладнання на визначеному етапі розробки, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год; $Ц_е$ – вартість 1 кВт-години електроенергії, грн; (за даними енергопостачальної компанії $Ц_е$ = 10,32 грн); $К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$; η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$$В_е = \frac{0,1 \cdot 1100 \cdot 10,32 \cdot 0,75 + 0,25 \cdot 1100 \cdot 10,32 \cdot 0,75}{0,9} = 3311,00 \text{ грн.}$$

До статті «Інші витрати» ($І_в$) належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Розрахуємо цю статтю витрат в межах 50% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} = (59090,80 + 2253,75) \cdot \frac{50}{100} = 30672,28 \text{ грн,}$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

До статті «Накладні (загальновиробничі) витрати» при впровадженні інформаційної технології тестування знань при вивченні іноземних слів відносяться такі витрати: витрати на управління організацією, витрати на винахідницьку та раціоналізаторську діяльність, витрати на підготовку, перепідготовку та навчання персоналу, витрати на наймання робочої сили, оплату послуг банків, освоєння нового виробництва, а також витрати на науково-технічну інформацію та рекламу.

Розмір витрат за статтею «Накладні (загальновиробничі) витрати» ($В_{нзв}$) розраховується як 100-150% від суми основної заробітної плати дослідників і працівників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} = (59090,80 + 2253,75) \cdot \frac{110}{100} = 67479,00 \text{ грн,}$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», прийнято на рівні 110%.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + В_{спец} + В_{прг} + А_{обл} + В_е + І_в + В_{нзв}, \quad (4.9)$$

$$В_{заг} = 59090{,}80 + 2253{,}75 + 6134{,}5 + 14845{,}40 + 697{,}84 + 62532{,}80 +$$
$$+20922{,}00 + 691{,}00 + 3311{,}00 + 30672{,}28 + 67479{,}00 = 268630{,}37 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{268630{,}37}{0.9} = 298478{,}2 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Враховуючи, що науково-технічна розробка знаходиться лише на стадії впровадження, прийнято η = 0,9.

4.3 Оцінювання важливості та наукової значимості інформаційної технології тестування знань при вивченні іноземних слів

Розроблена інформаційна технологія тестування знань при вивченні іноземних слів відноситься до категорії науково-дослідних робіт фундаментального або пошукового характеру. Її впровадження не передбачає отримання будь-якої економічної ефективності, оскільки основна мета полягає у підвищенні якості та доступності освітніх послуг, а також у створенні умов для індивідуалізації навчального процесу.

При цьому вона має науково-технічне та соціальне значення. З точки зору науково-технічного аспекту, запропонована технологія впроваджує адаптивний механізм тестування, який враховує індивідуальний рівень знань користувача, забезпечуючи персоналізований підхід до навчання. Це сприяє вдосконаленню методів оцінювання знань і впровадженню інновацій у галузі інформаційних технологій для освіти.

Соціальне значення результатів дослідження полягає у підвищенні ефективності процесу вивчення іноземних мов, що сприятиме зростанню мовної компетенції користувачів. Такий підхід відкриває нові можливості для професійного та освітнього розвитку в умовах глобалізації. Запропонована технологія також може бути інтегрована у навчальні процеси закладів освіти різних рівнів, сприяючи доступності якісної мовної освіти для широкого кола осіб.

Для обґрунтування доцільності виконання науково-дослідної роботи використаємо спеціальний комплексний показник ($К_р$) рівня науково-дослідної роботи, що враховує важливість, результативність роботи, можливість впровадження її результатів у виробництво, величину витрат на роботу.

Комплексний показник розрахуємо за формулою:

$$К_р = \frac{I^n \cdot T_c \cdot R}{ЗВ \cdot t}, \qquad (4.10)$$

де I – коефіцієнт важливості роботи, I = 2...5; n – коефіцієнт використання результатів роботи; n = 0, коли результати роботи не будуть використовуватись; n = 1, коли результати роботи будуть використовуватись частково; n = 2, коли результати роботи будуть використовуватись в дослідно-конструкторських розробках; n = 3, коли результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок; T_c – коефіцієнт складності роботи, T_c = 1...3; R – коефіцієнт результативності роботи; якщо результати роботи плануються вище відомих, то R = 4; якщо результати роботи відповідають відомому рівню, то R = 3 ; якщо нижче відомих результатів, то R = 1; ЗВ – вартість науково-дослідної роботи, тис. грн; t – час проведення дослідження, років.

Значення наведених показників визначено експертним методом: I = 3, n =3, T_c = 3, R = 4, ЗВ = 298,5 тис. грн, t = 1.

Отже маємо:

$$К_р = \frac{3^3 \cdot 3 \cdot 4}{296{,}4 \cdot 1} = \frac{324}{298{,}5} = 1{,}085.$$

Оскільки $К_р > 1$, то науково-дослідну роботу на тему «Інформаційна технологія тестування знань при вивченні іноземних слів» можна вважати ефективною з високим науковим, технічним і економічним рівнями.

Варто також зазначити, що комплексний показник $К_р$ рівня науково-дослідної роботи може досягати вищого рівня за рахунок оптимізації витрат на придбання спеціалізованого обладнання та програмних засобів. Це можливо завдяки використанню наявного технічного забезпечення, яке вже є у розпорядженні закладів освіти, де планується впровадження даної інформаційної технології. Такий підхід сприятиме зниженню фінансового навантаження на етапі впровадження, та позитивно вплине на економічну складову проєкту.

4.4 Висновок до розділу 4

Результати проведеного експертного оцінювання інформаційної технології тестування знань при вивченні іноземних слів відзначають її високий науково-технічний рівень з результатом у 40,67 бала, що, зокрема, досягнуто завдяки розширенню функціональних можливостей нової інформаційної системи порівняно з аналогами.

Сума загальних витрат на проведення усіх етапів науково-дослідної роботи та оформлення її результатів прогнозована на рівні 298478,2 грн. При цьому розрахунок комплексного показника $(К_р = 1{,}085)$ рівня науково-дослідної роботи підтверджує її ефективність з високим науковим, технічним і економічним рівнями. Значення даного показника, зокрема, може бути й вищим завдяки використання наявного у закладів освіти спецустаткування, необхідного для впровадження розробленої інформаційної технології.

Загалом проведені розрахунки дозволяють зробити висновок про доцільність проведення науково-дослідної роботи на тему «Інформаційна технологія тестування знань при вивченні іноземних слів»

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію тестування знань при вивченні іноземних слів, зокрема:

1. Проведено комплексний аналіз сучасних інформаційних технологій та математичних моделей тестування знань, орієнтованих на вивчення іноземних слів. Розглянуто найпопулярніші освітні платформи та їхні підходи до навчання та оцінювання прогресу користувачів. Виявлено, що ці платформи пропонують різноманітні інструменти для інтерактивного навчання та є досить ефективними для загального розвитку мовних навичок, проте їм бракує індивідуалізації та гнучкості в оцінюванні прогресу користувачів.

2. Запропоновано удосконалену математичну модель тестування знань при вивченні іноземних слів на основі інтегральної оцінки, що враховує рівень складності завдання, час виконання завдання та поріг ефективності відповіді.

3. Розроблено удосконалений алгоритм та структуру інформаційної технології, що забезпечує динамічне налаштування параметрів інтегральної оцінки рівня складності запитань відповідно до індивідуальних показників користувача.

4. Здійснено програмну реалізацію інформаційної технології тестування знань при вивченні іноземних слів. Зокрема, обґрунтовано вибір мов програмування та інтегрованого середовища розробки. Створено UML-діаграму класів та UML-діаграми послідовностей взаємодії основних модулів, таких як генерація питань, обробка відповідей, адаптація складності та аналіз результатів тестування.

5. Проведено тестування та налаштування параметрів інтегральної оцінки тестового контролю знань. Зокрема, встановлено порогові значення ефективності відповіді $\theta = 0{,}8$; вагові коефіцієнти, що визначають важливість точності та часу відповіді $\omega_1 = 0{,}6$ та $\omega_2 = 0{,}4$; коефіцієнти складності відповіді: $\alpha = 1$ для складних, $\alpha = 0{,}85$ для середніх та $\alpha = 0{,}7$ для простих

завдань. Тестування роботи інформаційної системи підтвердило коректність і надійність розроблених модулів, а співставлення з існуючими аналогами показало її перевагу та розширення функціональних можливостей на 33% у порівнянні з найближчим конкурентом – платформи Busuu.

6. Проведене комплексне оцінювання розробленої інформаційної технології підтверджує її ефективність з високим науковим, технічним і економічним рівнями $\left(К_р = 1{,}085;\ СБ_с = 40{,}67\right)$. При цьому орієнтовна сума загальних витрат на проведення всіх етапів науково-дослідної роботи та оформлення її результатів складає 298478,2 грн.

Таким чином, усі поставлені завдання успішно виконано та мету магістерської кваліфікаційної роботи досягнуто.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Савчук Т. О., Тодощак А. М. Структурні особливості модулю тестування знань при вивченні іноземних слів. *Тези LII науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ-2023).* URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023

2. Савчук Т. О., Ольшанська О.В., Тодощак А. М. Удосконалений алгоритм тестування знань при вивченні іноземних слів. *Innovations and prospects in modern science. Proceedings of the 6th International scientific and practical conference*. SSPG Publish. Stockholm, Sweden. 2023. P. 264-268. URL: https://sci-conf.com.ua/wp-content/uploads/2023/06/INNOVATIONS-AND-PROSPECTS-IN-MODERN-SCIENCE-5-7.06.2023.pdf

3. Шевчук О.Ф., Тодощак А.М. Підвищення ефективності тестування знань при вивченні іноземних слів. *Progressive Opportunities and Solutions of Advanced Society: Proceedings of the 2nd International Scientific and Practical Internet Conference*, November 7-8, 2024. FOP Marenichenko V.V., Dnipro, Ukraine, C. 281-282.

4. Савчук Т. О., Тодощак А.М. Свідоцтво про реєстрацію авторського права на твір вх. №. С202304033. Комп'ютерна програма «Модуль тестування знань при вивченні іноземних слів» / Т. О. Савчук, А.М. Тодощак (Україна), НАЦІОНАЛЬНИЙ ОРГАН ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ ДЕРЖАВНЕ ПІДПРИЄМСТВО «УКРАЇНСЬКИЙ ІНСТИТУТ ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ» (УКРПАТЕНТ) – Дата реєстрації 31-05-2023 р.

5. The Cognitive Benefits of Language Learning: Broadening Our Perspectives. *Cognitive Benefits of Language Learning*. URL: https://www.ucl.ac.uk/cognitive-benefits-language-learning/ (date of access: 01.09.2024).

6. Задорожна І. Використання інформаційно-комунікаційних технологій у навчанні іноземних мов: можливості, проблеми та шляхи їх вирішення. *Педагогічні науки: теорія, історія, інноваційні технології*, 2021. № 2 (106). С. 232-244.

7. Белзецький Р.С., Карабун В.С. Інтелектуальний модуль для розширення словникового запасу іноземної мови у користувача. Матеріали LI Науково-технічної конференції підрозділів Вінницького національного технічного університету (2022). URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/15469 (дата звернення: 03.09.2024)

8. Moussa M. An Evaluation of the Effectiveness of Mobile Applications in Language Learning: A Case Study of Duolingo. *International Journal of Language and Linguistics*. 2019. № 6(3). P. 114-121. URL: https://ijllnet.com/journals/ijll/Vol_6_No_3_September_2019/11.pdf

9. Aziz S. K., Raheel M. The Impact of Mobile Learning on Students' Achievement in Learning English Vocabulary. *Journal of Educational Technology Systems*. 2020. № 49(2). P. 245-263. DOI: 10.1177/0047239520906358

10. Zhou M., Lee Y. The Effectiveness of Mobile Apps for Vocabulary Learning: A Meta-Analysis. *Educational Technology Research and Development*, 2021. № 69(3). P. 1205-1223. DOI: 10.1007/s11423-021-09916-5

11. Comparing the Top 5 Language Learning Apps of the Year: 2024. *Ling Holic*. URL: https://lingholic.com/top-language-learning-apps/ (date of access: 03.09.2024).

12. Team J. E. Quizizz vs Kahoot!: Choosing the right quiz platform | The Jotform Blog. *Jotform*. URL: https://www.jotform.com/blog/quizizz-vs-kahoot/ (date of access: 03. 09.2024).

13. Language Learning with AI Tools. *International Conference of TESOL & Education (ICTE) – ICTE Conference Proceedings, TESOL & Education, AI in Language Education and Language Instruction*. URL: https://i-cte.org/academy/language-learning-with-ai-tools/ (date of access: 03.09.2024).

14. Gamification in Language Learning – Games2Teach. *Games2Teach*. URL: https://games2teach.uoregon.edu/2016/10/05/gamification-language-learning/ (date of access: 03.09.2024).

15. EDMODO – Distance Learning Toolkit. *HundrED*. URL: https://hundred.org/en/innovations/edmodo-distance-learning-toolkit (date of access: 03.09.2024).

16. 25 Engaging Ways to Practice English with Voice Assistants. *Ellii (formerly ESL Library) – ESL Lessons & Resources for English teachers*. URL: https://ellii.com/blog/english-with-voice-assistants(date of access: 03.09.2024).

17. Bulut O., Shin J., Yildirim-Erbasli S.N., Gorgun G., Pardos Z.A. An Introduction to Bayesian Knowledge Tracing with pyBKT. *Psych*. 2023; 5(3):770-786. https://doi.org/10.3390/psych5030050

18. van der Linden W. J., Guo F. Bayesian procedures for identifying aberrant response-time patterns in adaptive testing. *Psychometrika*. 2008. № 73, P. 365-384.

19. van der Linden W. J. Test design and speededness. *Journal of Educational Measurement*. 2011. № 48. P. 43-59.

20. van der Linden W. J. Setting time limits on tests. *Applied Psychological Measurement*. 2011. № 35. P. 183-199.

21. Wim J., van der Linden. Modeling response times with latent variables: Principles and applications. *Psychological Test and Assessment Modeling*. 2011. Vol. 53 (3). P. 334-358.

22. 34 Software Testing Metrics And KPIs: Complete Guide 2024 | ThinkSys Inc. *ThinkSys Inc.* URL: https://thinksys.com/qa-testing/software-testing-metrics-kpis/ (date of access: 03.10.2024).

23. Young N. What is Adaptive Testing? A Beginner's Guide to Tailored Assessments – Teachfloor. *Teachfloor: The leading social learning platform*. URL: https://www.teachfloor.com/elearning-glossary/adaptive-testing (date of access: 03.10.2024).

24. Test your English online. *Duolingo English Test*. URL: https://englishtest.duolingo.com/applicants (date of access: 3.10.2024).

25. Language Certification: Prove Your Level With Busuu – Busuu. Busuu. URL: https://www.busuu.com/en/languages/certification(date of access: 03.10.2024).

26. FunEasyLearn – Learn Languages Fast & for Free. *FunEasyLearn*. URL: https://www.funeasylearn.com(date of access: 03.11.2024).

27. Quizlet як сучасний інструмент опанування іншомовної лексики. URL: https://naurok.com.ua/prezentaciya-quizlet-suchasniy-instrument-dlya-opanuvannya-novo-leksiki-243442.html

28. Сучасні інформаційні технології у сфері штучного інтелекту : електронний навчальний посібник комбінованого (локального та мережного)

використання / Яровий А. А., Крилик Л. В. , Козловський А. В. Вінниця : ВНТУ, 2023. 145 с.

29. Мови програмування що для чого. Вибір мови програмування. *Портал про операційні системи.* URL: https://crashbox.ua/tools-to-help/programming languages-for-what-choice-of-programming-language/

30. What is Java technology and why do I need it? URL: https://java.com/en/download/faq/whatis_java.xml.

31. C++ Programming Language URL: https://www.techopedia.com/ definition/26184/c-programming-language.

32. Welcome to Python.org. *Python.org*. URL: https://www.python.org (date of access: 06.11.2024)

33. Node.js – Про Node.js®. Node*.js* – *Run JavaScript Everywhere*. URL: https://nodejs.org/uk/about (дата звернення: 06.11.2024).

34. Порівняння інтегрованих середовищ розробки додатків JAVA із відкритим кодом: ECLIPSE та INTELLIJ IDE. URL: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://elartu.tntu.edu.ua/bitstream/123456789/6592/2/FOSSLviv_2013_Kalinichenko_A_V-Comparison_integrated_73-75.pdf

35. Визначення та опис середовища розробки IntelliJ IDEA. URL: https://uk.wikipedia.org/wiki/IntelliJ_IDEA

36. Визначення та опис середовища розробки Eclipse. URL: https://uk.wikipedia.org/wiki/Eclipse

37. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В.О. Козловський, О.Й. Лесько, В.В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

38. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» / уклад.: А.А. Яровий, О. К. Колесницький. Вінниця : ВНТУ, 2023. 58 с.

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія тестування знань при вивченні іноземних слів

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 94,00% Схожість 6,00%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи Тодощак А.М.

Керівник роботи Шевчук О.Ф.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
@Service
@AllArgsConstructor
public class TestService {

    private static final int TEST_SIZE = 10;

    @Autowired
    private final QuestionRepository questionRepository;

    @Autowired
    private final AnswerRepository answerRepository;

    public void createTest(String topic, String type, String
level) {
        int typeInt = getType(type);
        List<Question> all = questionRepository.findAll();
        List<Question> questions = all
                .stream()
                .filter(question -> question.getType() == typeInt)
                .filter(question ->
question.getTopic().equals(topic))
                .limit(TEST_SIZE)
                .collect(Collectors.toList());
        State.setQuestions(questions);
    }


    private int getType(String type) {
        if (TYPE_1.equals(type)) {
            return 1;
        }
        return 2;
    }

    public void deleteCurrentQuestion() {
        State.getQuestions().remove(0);
    }

    public void setQuestion() {
        State.setQuestion(State.getQuestions().get(0));
    }

    public String getFinishButtonText() {
        long count = State.getAnswers().size();
        long genCount = TEST_SIZE;
        if (State.getIsMistakeTest()!= null &&
State.getIsMistakeTest().equals(true)){
            genCount = State.getAnswers().size() +
State.getQuestions().size();
```

```
            return String.format("Завершити роботу над помилками
(%s/%s)", count, genCount);
        }
        return String.format("Завершити тест (%s/%s)", count,
genCount);
    }

    public boolean processAnswer(String answerText) {
        Answer answer = new Answer();
        answer.setDate(getDate());
        answer.setQuestion(State.getQuestion());
        answer.setUser(State.getUser());
        Boolean isCorrect = isCorrect(answerText);
        answer.setIsCorrect(isCorrect);
        Answer savedAnswer = answerRepository.save(answer);
        State.addAnswer(savedAnswer);
        return isCorrect;
    }

    private Boolean isCorrect(String answerText) {
        return
State.getQuestion().getRightAnswer().equalsIgnoreCase(answerText);
    }

    private String getDate() {
        DateFormat formatter = new SimpleDateFormat("dd-MM-yyyy");
        String date = formatter.format(new Date());
        return date;
    }

    public String getFinishMessage() {
        int answerCount = State.getAnswers().size();
        long correctCount = State.getAnswers()
                .stream()
                .filter(Answer::getIsCorrect)
                .count();
        return String.format("Тест завершено: %s/%s",
correctCount, answerCount);
    }

    public boolean isLastQuestion() {
        return State.getQuestions().isEmpty();
    }

    public void createMistakeTest() {
        State.setIsMistakeTest(true);
        List<Answer> all = answerRepository.findAll()
                .stream().filter(answer ->
answer.getUser().getId() == State.getUser().getId())
                .collect(Collectors.toList());
        List<Question> questionList = all.stream()
                .filter(answer -> !answer.getIsCorrect())
                .map(Answer::getQuestion)
```

```
                .collect(Collectors.toSet())
                .stream()
                .collect(Collectors.toList());

        State.setQuestions(questionList);
    }
}
```

TestFrame.java

```
@Component
public class TestFrame extends AbstractFrame {

    private JLabel questionLabel = new JLabel();

    @Autowired
    private final TestService testService;
    @Autowired
    private final DictionaryService dictionaryService;

    public TestFrame(TestService testService, DictionaryService
dictionaryService) {
        super("Тестування");
        this.testService = testService;
        this.dictionaryService = dictionaryService;
    }

    @Override
    public void create() {
        label = new JLabel();
        label.setLayout(new GridBagLayout());
        label.setVisible(true);
        label.setBackground(Color.BLUE);
        label.setIcon(image);

        frame.setSize(400, 250);
        Question question = State.getQuestion();

        JLabel questionLabel = new JLabel();
        questionLabel.setFont(font);
        questionLabel.setForeground(Color.BLACK);
        questionLabel.setText("<html><p align=center>" +
question.getQuery() + "</p></html>");

        label.add(questionLabel, new GridBagConstraints(0, 0, 2,
1, 1, 1, GridBagConstraints.NORTH,
                GridBagConstraints.HORIZONTAL, new Insets(20, 200
- (question.getQuery().length() * 5), 8, 10), 0, 0));

        addButton(question.getAnswer1(), 0, 1, new
AnswerListener(), new JButton());
        addButton(question.getAnswer2(), 1, 1, new
AnswerListener(), new JButton());
        addButton(question.getAnswer3(), 0, 2, new
AnswerListener(), new JButton());
```

```
        addButton(question.getAnswer4(), 1, 2, new
AnswerListener(), new JButton());

        JButton dictionaryButton = new JButton();
        dictionaryButton.setFont(font);
        dictionaryButton.setText("Додати в мій словник");

        label.add(dictionaryButton, new GridBagConstraints(0, 4,
2, 1, 0.9, 0.9, GridBagConstraints.NORTH,
                GridBagConstraints.HORIZONTAL, new Insets(8, 10,
8, 10), 0, 0));
        dictionaryButton.addActionListener(new
DictionaryListener());

        JButton finishButton = new JButton();
        finishButton.setFont(font);
        finishButton.setText(testService.getFinishButtonText());

        label.add(finishButton, new GridBagConstraints(0, 5, 2, 1,
0.9, 0.9, GridBagConstraints.NORTH,
                GridBagConstraints.HORIZONTAL, new Insets(8, 10,
8, 10), 0, 0));

        finishButton.addActionListener(new FinishListener());

        frame.add(label);
        frame.setVisible(true);
        frame.revalidate();
        frame.repaint();
    }

    public class AnswerListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            String answer = ((JButton) e.getSource()).getText();
            boolean isCorrect = testService.processAnswer(answer);
            if (isCorrect){
                JOptionPane.showMessageDialog(null, "Вірно!");
            }else {
                JOptionPane.showMessageDialog(null,
String.format("На жаль Ви помилились. Правильна відповідь %s",
State.getQuestion().getRightAnswer()));
            }
            testService.deleteCurrentQuestion();
            if (testService.isLastQuestion()){
                JOptionPane.showMessageDialog(null,
testService.getFinishMessage());
                delete();
            }else{
                testService.setQuestion();
                frame.remove(label);
                create();
```

```
            }
        }
    }

    public class FinishListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            String answer = ((JButton) e.getSource()).getText();
            testService.processAnswer(answer);
            JOptionPane.showMessageDialog(null,
testService.getFinishMessage());

            delete();
        }
    }

    public class DictionaryListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            dictionaryService.addToDictionary();
        }
    }
}
```

TestSetUpFrame.java

```
@Component
public class TestSetUpFrame extends AbstractFrame {

    @Autowired
    private final QuestionService questionService;
    @Autowired
    private final TestFrame testFrame;
    @Autowired
    private final TestService testService;

    private JComboBox<String> topicBox, typeBox, levelBox;
    private static final List<String> TYPES = List.of("Тип",
TYPE_1, TYPE_2);
    private static final List<String> LEVEL =
List.of("Складність","Новачок", "Аматор", "Професіонал");


    public TestSetUpFrame(QuestionService questionService,
TestFrame testFrame, TestService testService) {
        super("Налаштування тесту");
        this.questionService = questionService;
        this.testFrame = testFrame;
        this.testService = testService;
    }
```

```
 @Override
    public void create() {
        frame.setSize(400, 200);
        Set<String> topics = questionService.getTopics();

        topicBox = new JComboBox<>(topics.toArray(new
String[]{}));
        topicBox.setFont(font);
        label.add(topicBox, new GridBagConstraints(0, 0, 1, 1,
0.9, 0.9, GridBagConstraints.NORTH,
                GridBagConstraints.HORIZONTAL, new Insets(8, 10,
8, 10), 0, 0));

        typeBox = new JComboBox<>(TYPES.toArray(new String[]{}));
        typeBox.setFont(font);
        label.add(typeBox, new GridBagConstraints(0, 1, 1, 1, 0.9,
0.9, GridBagConstraints.NORTH,
                GridBagConstraints.HORIZONTAL, new Insets(8, 10,
8, 10), 0, 0));

        levelBox = new JComboBox<>(LEVEL.toArray(new String[]{}));
        levelBox.setFont(font);
        label.add(levelBox, new GridBagConstraints(0, 2, 1, 1,
0.9, 0.9, GridBagConstraints.NORTH,
                GridBagConstraints.HORIZONTAL, new Insets(8, 10,
8, 10), 0, 0));

        JButton testButton = new JButton();

        addButton("Почати тест", 3, new StartListener(),
testButton);

        frame.add(label);
        frame.setVisible(true);

    }

    public class StartListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            delete();
            String topic = (String) topicBox.getSelectedItem();
            String type = (String) typeBox.getSelectedItem();
            String level = (String) levelBox.getSelectedItem();
            testService.createTest(topic, type, level);
            testService.setQuestion();
            testFrame.create();
        }
    }

}
```

State.java

```
public class State {

    public static final String TYPE_1 = "English -> Українська";
    public static final String TYPE_2 = "Українська -> English";

    private static User user;
    private static Question question;
    private static List<Question> questions = new ArrayList<>();
    private static List<Answer> answers = new ArrayList<>();
    private static Boolean isMistakeTest;
    public static User getUser() {
        return user;
    }

    public static void setUser(User user) {
        State.user = user;
    }
    public static Question getQuestion() {
        return question;
    }

    public static void setQuestion(Question question) {
        State.question = question;
    }

    public static List<Answer> getAnswers() {
        return answers;
    }

    public static void addAnswer(Answer answer) {
        State.answers.add(answer);
    }

    public static List<Question> getQuestions() {
        return questions;
    }

    public static void setQuestions(List<Question> questions) {
        State.questions = questions;
    }

    public static void setAnswers(List<Answer> answers) {
        State.answers = answers;
    }

    public static Boolean getIsMistakeTest() {
        return isMistakeTest;
    }

    public static void setIsMistakeTest(Boolean isMistakeTest) {
        State.isMistakeTest = isMistakeTest;
    }
```

```
}
```

QuestionService.java

```
@Service
@AllArgsConstructor
public class QuestionService {

    @Autowired
    private final QuestionRepository questionRepository;
    public Set<String> getTopics() {
        return questionRepository.findAll()
                .stream()
                .map(Question::getTopic)
                .collect(Collectors.toSet());
    }
}
```

Question.java

```
@Entity
@Table(name = "question")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Question {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "query")
    private String query;

    @Column(name = "answer1")
    private String answer1;

    @Column(name = "answer2")
    private String answer2;

    @Column(name = "answer3")
    private String answer3;

    @Column(name = "answer4")
    private String answer4;

    @Column(name = "right_answer")
    private String rightAnswer;

    @Column(name = "type")
    private int type; // 1 -> English to Ukrainian, 2 -> Ukrainian
to English

    @Column(name = "topic")
    private String topic;
```

```
}
```

Answer.java

```
@Entity
@Table(name = "answer")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Answer {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    //    @OneToOne(cascade = CascadeType.ALL)
//    @JoinColumn(name = "user_id", referencedColumnName = "id",
    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

    //    @OneToOne(cascade = CascadeType.ALL)
//    @JoinColumn(name = "question_id", referencedColumnName =
"id")
    @ManyToOne
    @JoinColumn(name = "question_id", nullable = false)
    private Question question;

    @Column(name = "is_correct")
    private Boolean isCorrect;

    @Column(name = "answer_date")
    private String date;

}
```

MenuFrame.java

```
@Component
public class MenuFrame extends AbstractFrame {

    @Autowired
    private final TestSetUpFrame testSetUpFrame;
    @Autowired
    private final TestService testService;
    @Autowired
    private final TestFrame testFrame;
    @Autowired
    private final DictionaryFrame dictionaryFrame;
    @Autowired
    private final StatisticsFrame statisticsFrame;

    public MenuFrame(TestSetUpFrame testSetUpFrame,
                     TestService testService,
                     TestFrame testFrame,
                     DictionaryFrame dictionaryFrame,
                     StatisticsFrame statisticsFrame) {
```

```
        super("Меню");
        this.testSetUpFrame = testSetUpFrame;
        this.testService = testService;
        this.testFrame = testFrame;
        this.dictionaryFrame = dictionaryFrame;
        this.statisticsFrame = statisticsFrame;
    }
    @Override
    public void create() {
        frame.setSize(400, 200);

        JButton testButton = new JButton();
        addButton("Тестування", 0, new TestListener(),
testButton);
        JButton mistakeButton = new JButton();
        addButton("Робота над помилками", 1, new
MistakeListener(), mistakeButton);
        JButton dictionaryButton = new JButton();
        addButton("Мій словник", 2, new DictionaryListener(),
dictionaryButton);
        JButton statisticsButton = new JButton();
        addButton("Статистика", 3, new StatisticsListener(),
statisticsButton);

        frame.add(label);
        frame.setVisible(true);
    }
    public class TestListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            delete();
            testSetUpFrame.create();
        }
    }
    public class MistakeListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            delete();
            testService.createMistakeTest();
            testService.setQuestion();
            testFrame.create();
        }
    }
    public class DictionaryListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            delete();
            dictionaryFrame.create();
        }
    }
```

```
    public class StatisticsListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            delete();
            statisticsFrame.create();
        }
                                }

public class Test {
    private String name;
    private List<User> users;
    private int correctAnswersCount;
    private int totalAnswersCount;
    private List<Long> completionTimes;

    public Test(String name) {
        this.name = name;
        this.users = new ArrayList<>();
        this.completionTimes = new ArrayList<>();
    }

    public void addUser(User user) {
        users.add(user);
        totalAnswersCount += user.getTotalAnswers();
        correctAnswersCount += user.getCorrectAnswers();
        completionTimes.add(user.getCompletionTime());
    }

    public String determineDifficultyLevel() {
        double successRate = (double) correctAnswersCount /
totalAnswersCount * 100;

        if (successRate > 90) {
            return "Простий рівень";
        } else if (successRate >= 60) {
            return "Середній рівень";
        } else {
            return "Складний рівень";
        }
    }

    public double calculateAverageCompletionTime() {
        return
completionTimes.stream().mapToLong(Long::longValue).average().orEl
se(0);
    }

    public double calculateMedianCompletionTime() {
        completionTimes.sort(Long::compareTo);
        int size = completionTimes.size();
        if (size % 2 == 0) {
```

```
            return (completionTimes.get(size / 2 - 1) +
completionTimes.get(size / 2)) / 2.0;
        } else {
            return completionTimes.get(size / 2);
        }
    }

    public double calculateTimeVariance() {
        double mean = calculateAverageCompletionTime();
        return completionTimes.stream()
                .mapToDouble(time -> Math.pow(time - mean, 2))
                .average().orElse(0);
    }

    public String getName() {
        return name;
    }
}

public class User {
    private String username;
    private int correctAnswers;
    private int totalAnswers;
    private long completionTime;

    public User(String username, int correctAnswers, int
totalAnswers, long completionTime) {
        this.username = username;
        this.correctAnswers = correctAnswers;
        this.totalAnswers = totalAnswers;
        this.completionTime = completionTime;
    }

    public int getCorrectAnswers() {
        return correctAnswers;
    }

    public int getTotalAnswers() {
        return totalAnswers;
    }

    public long getCompletionTime() {
        return completionTime;
    }

    public String getUsername() {
        return username;
    }
}


import java.util.Arrays;
```

```
public class Main {
    public static void main(String[] args) {

        String difficultyLevel =
mathTest.determineDifficultyLevel();
        System.out.println("Рівень складності тесту: " +
difficultyLevel);

        double averageTime =
mathTest.calculateAverageCompletionTime();
        double medianTime =
mathTest.calculateMedianCompletionTime();
        double varianceTime = mathTest.calculateTimeVariance();

        System.out.printf("Середній час виконання: %.2f секунд\n",
averageTime / 1000);
        System.out.printf("Медіанний час виконання: %.2f
секунд\n", medianTime / 1000);
        System.out.printf("Дисперсія часу виконання: %.2f
секунд\n", Math.sqrt(varianceTime) / 1000);
    }
}

public class TestGenerationModule {
    private Database database;
    private User currentUser;

    public TestGenerationModule(Database database) {
        this.database = database;
    }

    public void initializeTest() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Введіть ваш логін: ");
        String username = scanner.nextLine();

        System.out.print("Введіть ваш пароль: ");
        String password = scanner.nextLine();

        if (authenticateUser(username, password)) {
            System.out.println("Авторизація успішна.");
            currentUser = database.getUserByUsername(username);

            String knowledgeLevel =
currentUser.getKnowledgeLevel();
            System.out.println("Ваш рівень знань: " +
knowledgeLevel);

            String difficultyLevel =
determineDifficultyLevel(knowledgeLevel);
            System.out.println("Рівень складності тесту: " +
difficultyLevel);
```

```
            List<Question> questions =
database.getQuestions(difficultyLevel);

            if (!questions.isEmpty()) {
                Test test = new Test(questions);
                System.out.println("Тест сформовано.");
            } else {
                System.out.println("Відсутні питання для заданого
рівня складності.");
            }
        } else {
            System.out.println("Помилка авторизації. Перевірте
ваші облікові дані.");
        }

        scanner.close();
    }

    private boolean authenticateUser(String username, String
password) {
        return database.verifyCredentials(username, password);
    }

    private String determineDifficultyLevel(String knowledgeLevel)
{
        switch (knowledgeLevel) {
            case "Початковий":
                return "Легкий";
            case "Середній":
                return "Середній";
            case "Просунутий":
                return "Складний";
            default:
                return "Середній";
        }
    }

    public static void main(String[] args) {
        Database database = new Database();
        TestGenerationModule testGenModule = new
TestGenerationModule(database);
        testGenModule.initializeTest();
    }
}


import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class TestGenerator {
```

```
    private Connection connection;

    public TestGenerator() {
        try {
            connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/test_db",
"user", "password");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public List<String> getQuestions(int difficultyLevel, int
numberOfQuestions) {
        List<String> questions = new ArrayList<>();
        try {
            String query = "SELECT question_text FROM questions
WHERE difficulty = ? ORDER BY RAND() LIMIT ?";
            PreparedStatement statement =
connection.prepareStatement(query);
            statement.setInt(1, difficultyLevel);
            statement.setInt(2, numberOfQuestions);
            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()) {

questions.add(resultSet.getString("question_text"));
            }

            resultSet.close();
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return questions;
    }

    public void close() {
        try {
            if (connection != null) connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        TestGenerator generator = new TestGenerator();

        List<String> questions = generator.getQuestions(2, 50);

        System.out.println("Список питань:");
        for (String question : questions) {
            System.out.println(question);
```

```
        }

        generator.close();
    }
}

public class AnswerProcessor {
    private int totalScore = 0;

    public int processAnswer(String userAnswer, String
correctAnswer, int difficultyLevel, long responseTimeMillis) {
        int score = 0;
        if (userAnswer.equalsIgnoreCase(correctAnswer)) {
            switch (difficultyLevel) {
                case 1:
                    score = 7;
                    break;
                case 2:
                    score = 10;
                    break;
                case 3:
                    score = 15;
                    break;
                default:
                    score = 0;
                    break;
            }
        }
        totalScore += score;

        logAnswerResult(score, responseTimeMillis);

        return score;
    }

    private void logAnswerResult(int score, long
responseTimeMillis) {
        String url = "jdbc:mysql://localhost:3306/testsystem";
        String user = "your_username";
        String password = "your_password";

        String query = "INSERT INTO answer_log (score,
response_time) VALUES (?, ?)";

        try (Connection connection =
DriverManager.getConnection(url, user, password);
             PreparedStatement statement =
connection.prepareStatement(query)) {
            statement.setInt(1, score);
            statement.setLong(2, responseTimeMillis);
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
```

```
        }
    }

    public int getTotalScore() {
        return totalScore;
    }

    public static void main(String[] args) {
        AnswerProcessor processor = new AnswerProcessor();

        processor.processAnswer("answer1", "answer1", 1, 3000);
        processor.processAnswer("wrongAnswer", "answer2", 2,
5000);
        processor.processAnswer("answer3", "answer3", 3, 8000);

        System.out.println("Загальний бал: " +
processor.getTotalScore());
    }
}

class AnswerProcessor:
    def __init__(self):
        self.total_score = 0

    def process_answer(self, user_answer, correct_answer,
difficulty_level, response_time):
        score = 0
        if user_answer.lower() == correct_answer.lower():
            if difficulty_level == 1:
                score = 7
            elif difficulty_level == 2:
                score = 10
            elif difficulty_level == 3:
                score = 15

        self.total_score += score
        self.log_answer_result(score, response_time)
        return score

    def log_answer_result(self, score, response_time):
        conn = sqlite3.connect('test_system.db')
        cursor = conn.cursor()
        cursor.execute("CREATE TABLE IF NOT EXISTS answer_log
(score INTEGER, response_time REAL)")
        cursor.execute("INSERT INTO answer_log (score,
response_time) VALUES (?, ?)", (score, response_time))
        conn.commit()
        conn.close()

    def get_total_score(self):
        return self.total_score

if __name__ == "__main__":
```

```
    processor = AnswerProcessor()

    start_time = time.time()
    processor.process_answer("answer1", "answer1", 1, time.time()
- start_time)
    processor.process_answer("wrongAnswer", "answer2", 2,
time.time() - start_time)
    processor.process_answer("answer3", "answer3", 3, time.time()
- start_time)

    print("Загальний бал:", processor.get_total_score())

class DifficultyAdapter:
    def __init__(self):
        self.current_difficulty = "medium"
        self.score_threshold = 0.6
        self.difficulty_levels = {
            "easy": 0.7,
            "medium": 0.85,
            "hard": 1.0
        }

    def adapt_difficulty(self, score):
        if score >= self.score_threshold:
            if self.current_difficulty == "easy":
                self.current_difficulty = "medium"
            elif self.current_difficulty == "medium":
                self.current_difficulty = "hard"
        else:
            if self.current_difficulty == "hard":
                self.current_difficulty = "medium"
            elif self.current_difficulty == "medium":
                self.current_difficulty = "easy"

        return self.current_difficulty

    def get_question_weight(self):
        return self.difficulty_levels[self.current_difficulty]

adapter = DifficultyAdapter()
user_score = 0.75

new_difficulty = adapter.adapt_difficulty(user_score)
question_weight = adapter.get_question_weight()

print(f"Новий рівень складності: {new_difficulty}")
print(f"Вага питання для {new_difficulty} рівня:
{question_weight}")

class ResultsAnalyzer:
    def __init__(self):
        self.coef1 = 0.6
        self.coef2 = 0.4
```

```
        self.threshold_score = 0.6
        self.settings = {
            'simple': {'mean_time': 4, 'std_dev': 0.8, 'weight':
0.7, 'threshold_percentile': 25},
            'medium': {'mean_time': 7.5, 'std_dev': 2.5, 'weight':
0.85, 'threshold_percentile': 50},
            'hard': {'mean_time': 13, 'std_dev': 4, 'weight': 1.0,
'threshold_percentile': 75}
        }

    def calculate_score(self, difficulty, is_correct, efficiency):
        if not is_correct:
            return 0

        weight = self.settings[difficulty]['weight']
        score = self.coef1 * weight + self.coef2 * efficiency
        return score

    def determine_next_difficulty(self, score,
current_difficulty):
        if score > self.threshold_score:
            if current_difficulty == 'simple':
                return 'medium'
            elif current_difficulty == 'medium':
                return 'hard'
        else:
            if current_difficulty == 'hard':
                return 'medium'
            elif current_difficulty == 'medium':
                return 'simple'
        return current_difficulty

analyzer = ResultsAnalyzer()
difficulty = 'medium'
is_correct = True
efficiency = 0.7

score = analyzer.calculate_score(difficulty, is_correct,
efficiency)
next_difficulty = analyzer.determine_next_difficulty(score,
difficulty)

print("Score:", score)
print("Next Difficulty Level:", next_difficulty)


#TEST

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

difficulty_settings = {
```

```
    'simple': {'mean_time': 4, 'std_dev': 0.8, 'weight': 0.7,
'threshold_percentile': 25},
    'medium': {'mean_time': 7.5, 'std_dev': 2.5, 'weight': 0.85,
'threshold_percentile': 50},
    'hard': {'mean_time': 13, 'std_dev': 4, 'weight': 1.0,
'threshold_percentile': 75}
}

coef1, coef2 = 0.6, 0.4
score_threshold = 0.6

def generate_time_data(mean, std_dev):
    return np.random.normal(mean, std_dev)

def calculate_efficiency(time, threshold):
    return 1 if time <= threshold else threshold / time

def generate_test_results(username, num_tests=10,
initial_difficulty='medium'):
    results = []
    current_difficulty = initial_difficulty

    for question_num in range(1, num_tests + 1):
        settings = difficulty_settings[current_difficulty]

        response_time = generate_time_data(settings['mean_time'],
settings['std_dev'])

        threshold = np.percentile(
            np.random.normal(settings['mean_time'],
settings['std_dev'], 1000),
            settings['threshold_percentile']
        )

        efficiency = calculate_efficiency(response_time,
threshold)

        correctness_prob = {'simple': 0.95, 'medium': 0.9, 'hard':
0.8}[current_difficulty]
        correct = np.random.rand() < correctness_prob

        score = 0
        if correct:
            score = coef1 * settings['weight'] + coef2 *
efficiency
        else:
            score = 0

        results.append({
            'Username': username,
            'Question': question_num,
            'Difficulty': current_difficulty,
            'Correct': correct,
```

```
            'Response Time': response_time,
            'Efficiency': efficiency,
            'Score': score
        })

        if score > score_threshold:
            if current_difficulty == 'simple':
                current_difficulty = 'medium'
            elif current_difficulty == 'medium':
                current_difficulty = 'hard'
        else:
            if current_difficulty == 'hard':
                current_difficulty = 'medium'
            elif current_difficulty == 'medium':
                current_difficulty = 'simple'

    results_df = pd.DataFrame(results)
    total_score = results_df['Score'].sum()
    return results_df, total_score

def main():
    username = "user_123"
    results_df, total_score = generate_test_results(username)

    print(results_df)
    print(f"\nЗагальна оцінка: {total_score}")

    color_map = {'simple': 'green', 'medium': 'orange', 'hard':
'red'}
    plt.figure(figsize=(10, 5))

    for difficulty, color in color_map.items():
        subset = results_df[results_df['Difficulty'] ==
difficulty]
        plt.scatter(subset['Question'], subset['Score'],
color=color, label=difficulty.capitalize(), s=50)

    plt.plot(results_df['Question'], results_df['Score'],
color='b', linestyle='--', label='Total Score')
    plt.title('Оцінки за питання з різними рівнями складності')
    plt.xlabel('Номер питання')
    plt.ylabel('Оцінка')
    plt.legend()
    plt.grid(True)

    plt.figtext(
        0.5, -0.1, f'Загальна оцінка: {total_score:.2f}',
        fontsize=12, color='blue', weight='bold',
        ha='center', va='center', bbox=dict(facecolor='white',
edgecolor='blue', boxstyle='round,pad=0.5')
    )
    plt.show()
main()
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ТЕСТУВАННЯ ЗНАНЬ ПРИ ВИВЧЕННІ ІНОЗЕМНИХ СЛІВ

Виконала: студентка 2 курсу, групи 1КН-23м
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Тодощак А.М.
(прізвище та ініціали)
Керівник: к.ф.-м.н., доцент каф. КН
Шевчук О. Ф.
(прізвище та ініціали)
«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

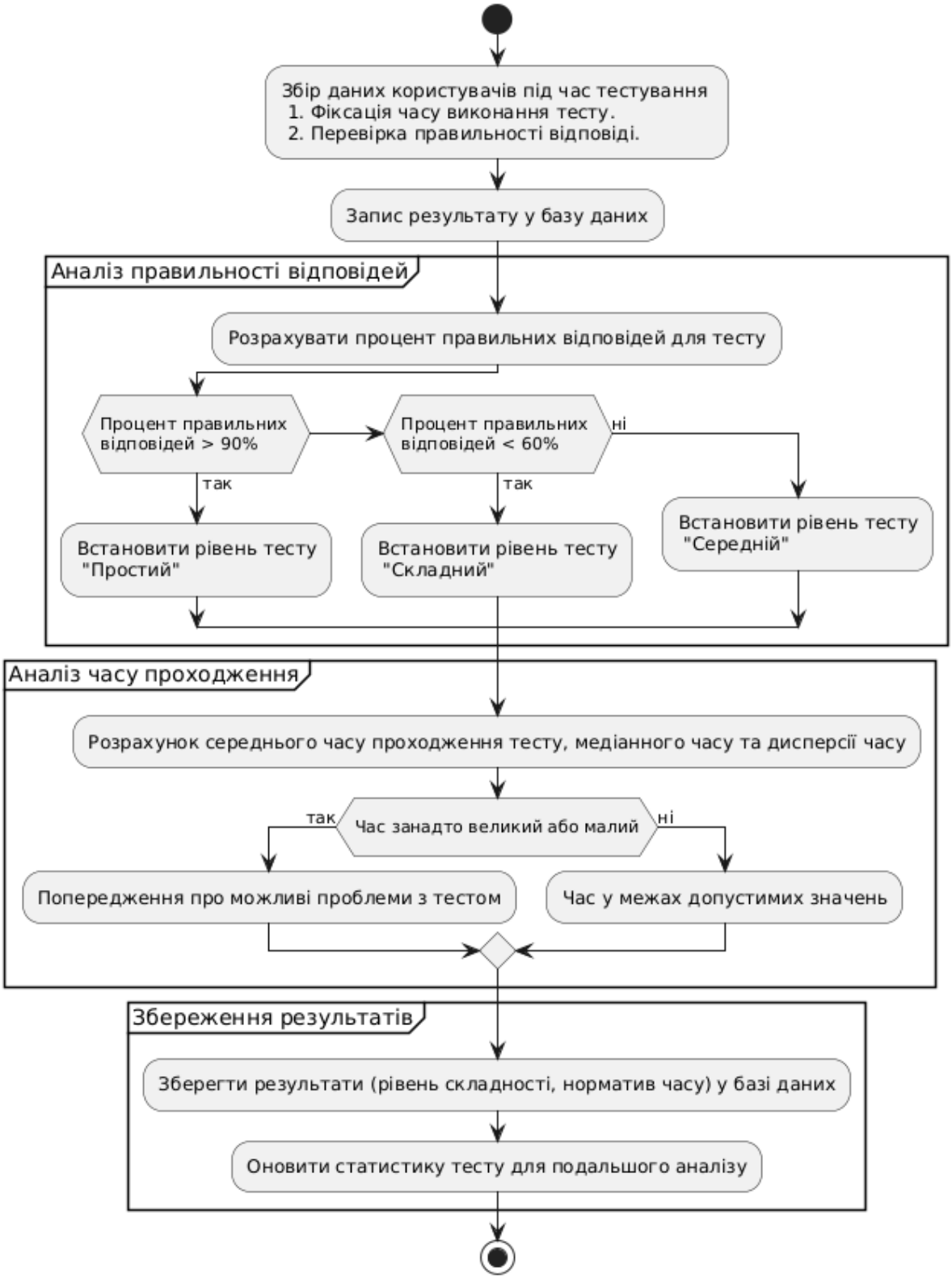


Рисунок В.1 – UML-діаграма алгоритму аналізу рівня складності тесту та часу його проходження

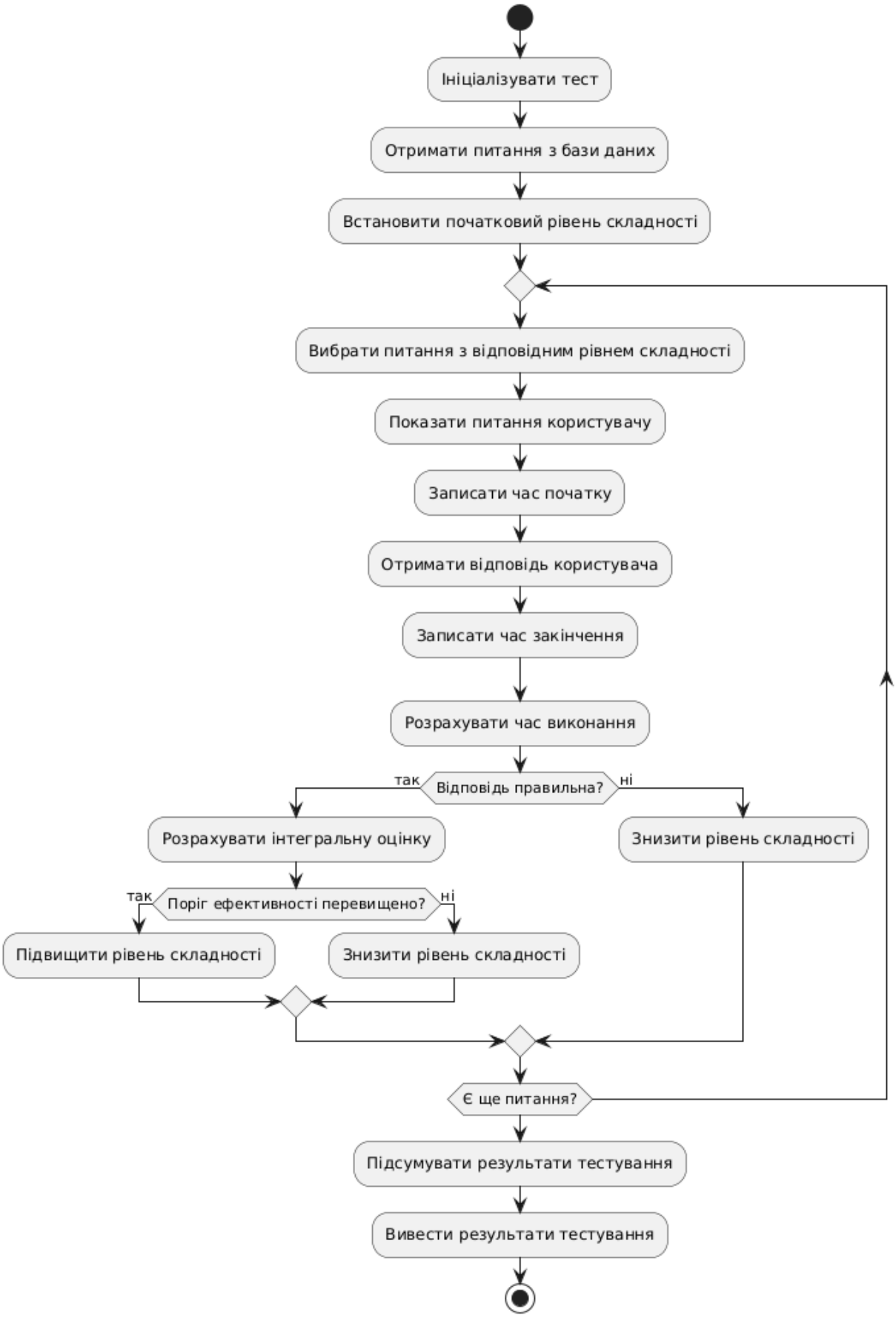


Рисунок В.2 – UML-діаграма удосконаленого алгоритму тестування знань при вивченні іноземних слів

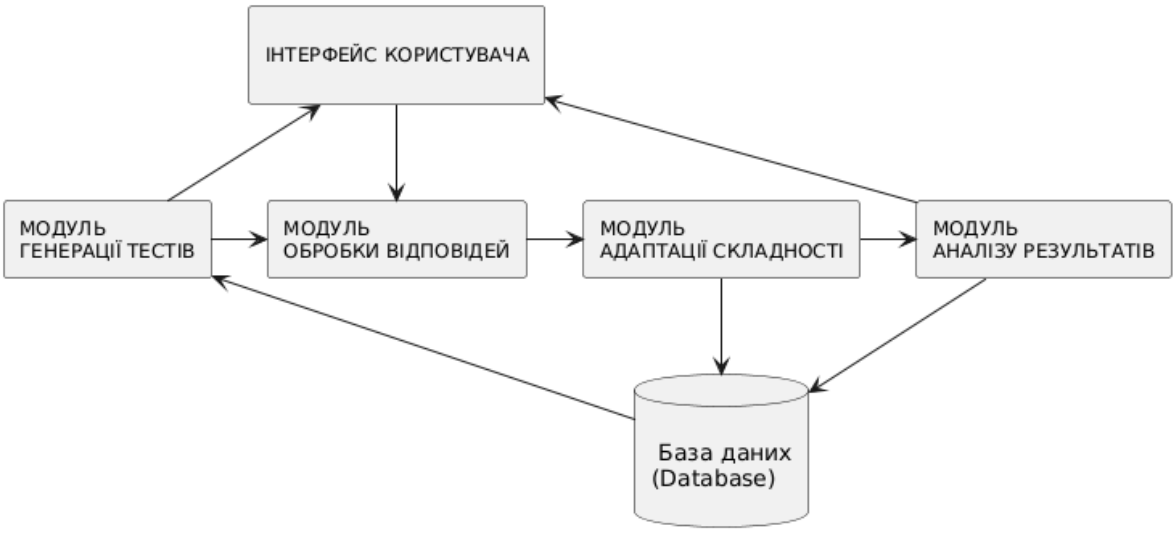


Рисунок В.3 – Основні складові структури інформаційної технології тестування знань при вивченні іноземних слів

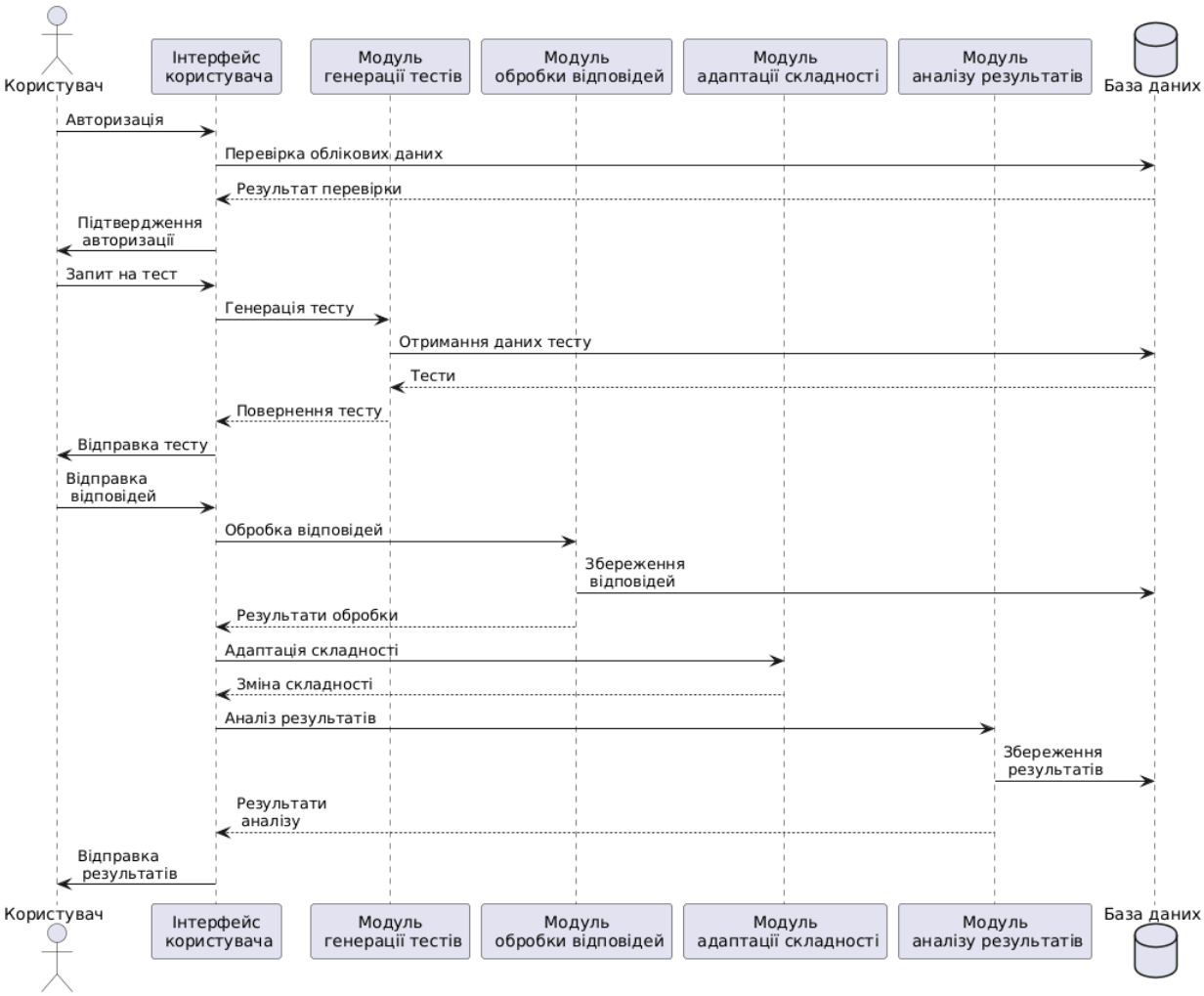


Рисунок В.4 – UML-діаграма послідовностей роботи інформаційної системи тестування знань при вивченні іноземних слів

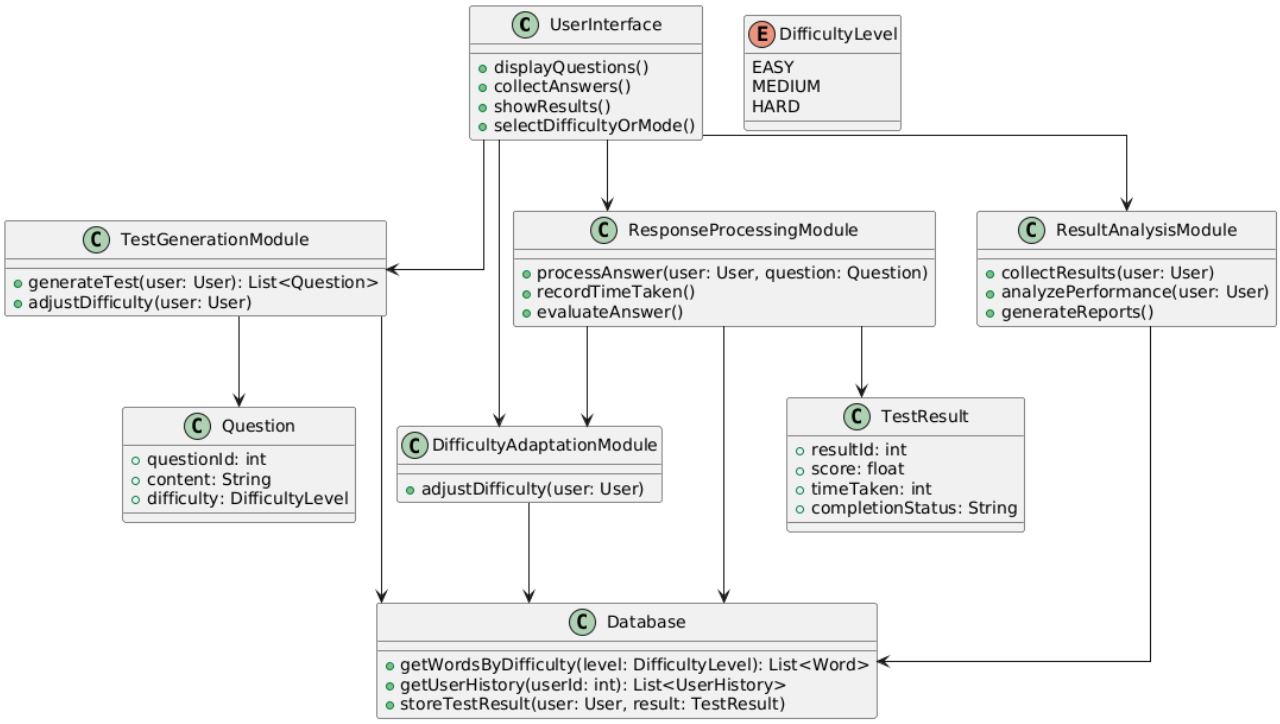


Рисунок В.5 – UML-діаграма класів інформаційної технології тестування знань при вивченні іноземних слів

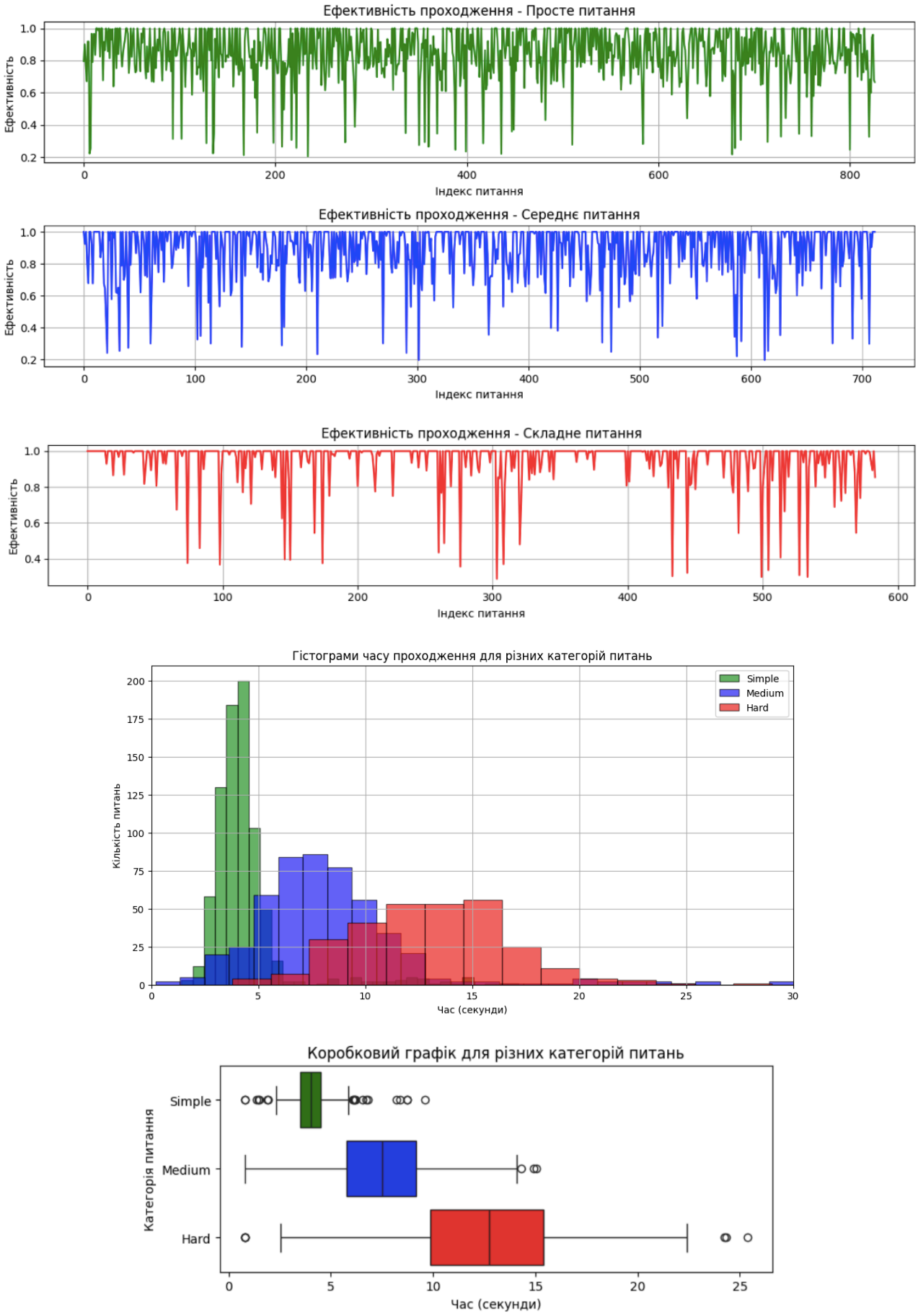
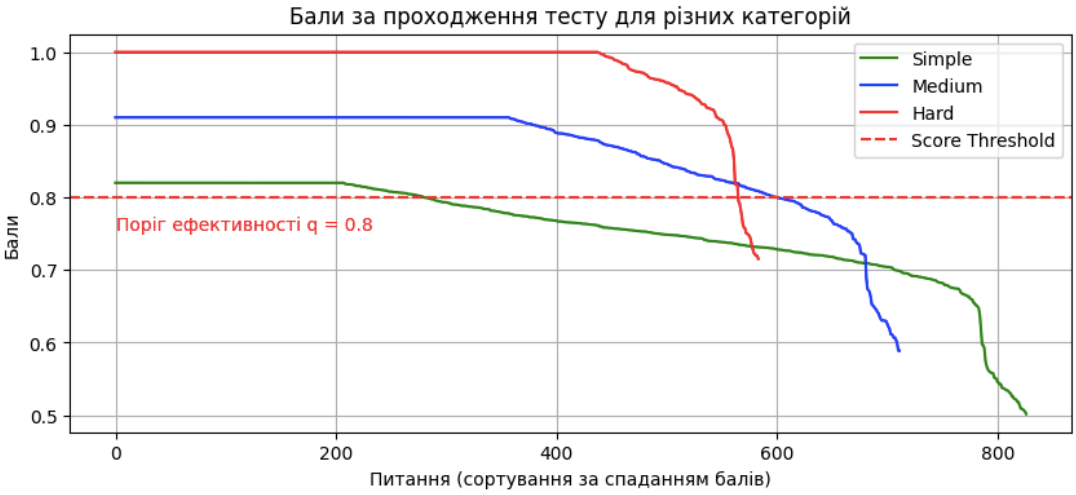
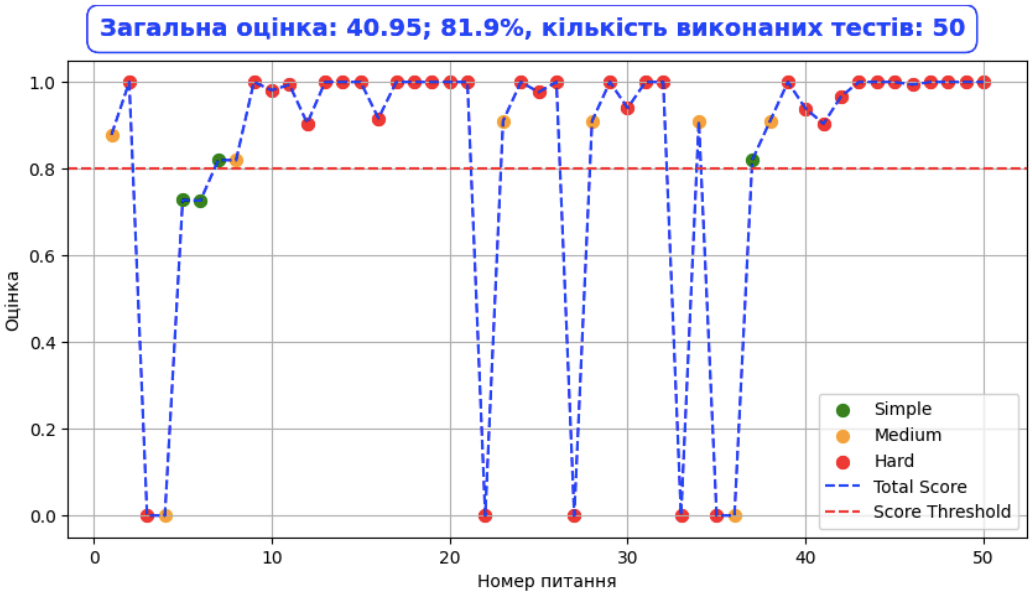


Рисунок В.6 – Тестова вибірка питань





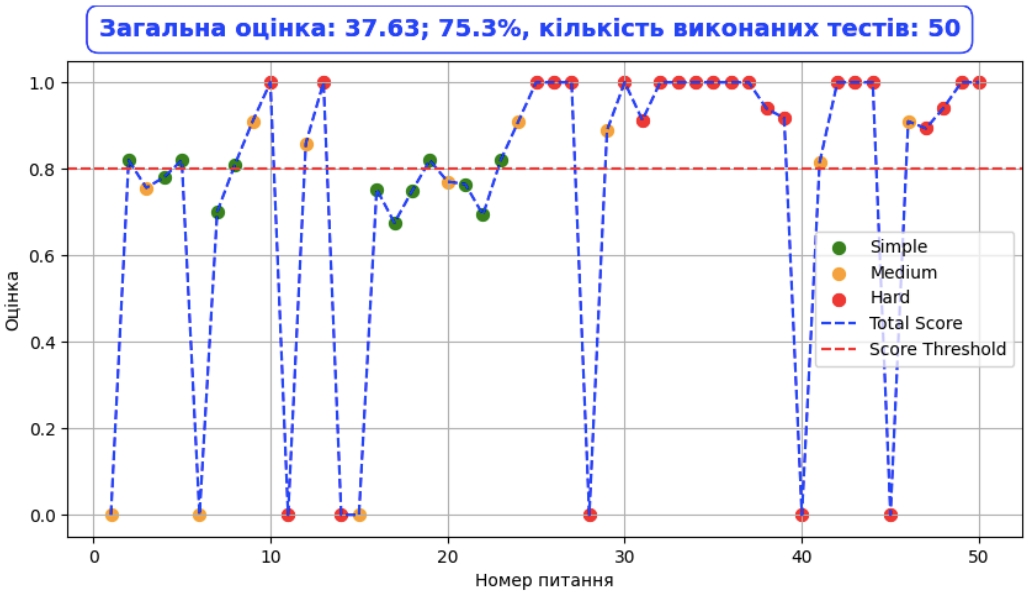


Рисунок В.7 – Результати тестування розробленої програми

Додаток Г (довідниковий)

Інструкція користувача

Робота програми розпочинається з авторизації користувача, що зображено на рисунку Г.1.

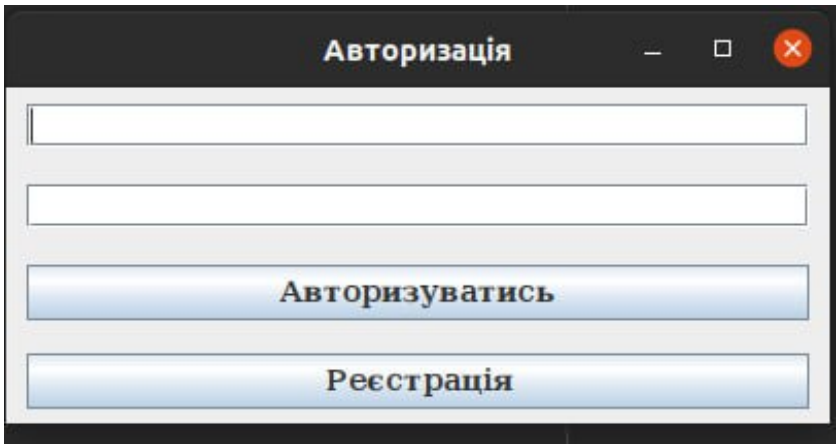


Рисунок Г.1 – Вікно авторизації

1. Користувач вводить своє ім'я та пароль та намагається авторизуватися.

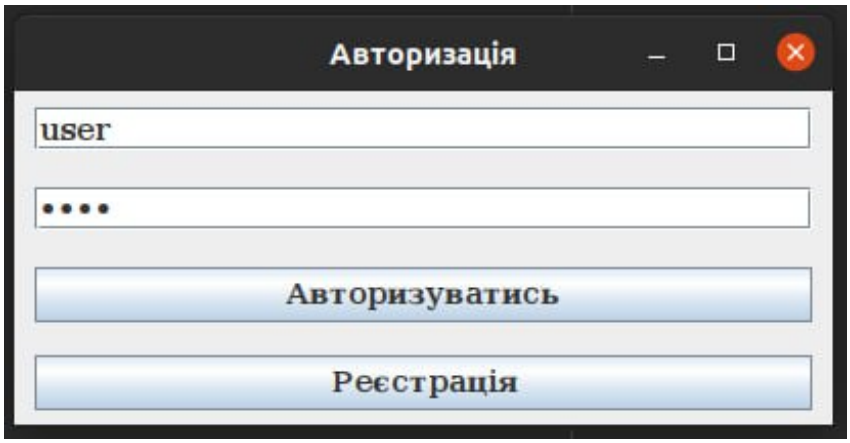


Рисунок Г.2 – Введення даних для авторизації користувача

2. Якщо користувача з таким ім'ям та паролем не існує, то відбувається перехід до вікна реєстрації шляхом натиснення на поле «Реєстрація». Тут користувач має можливість придумати ім'я та пароль, який необхідно підтвердити, прописавши його в третьому полі для вводу.

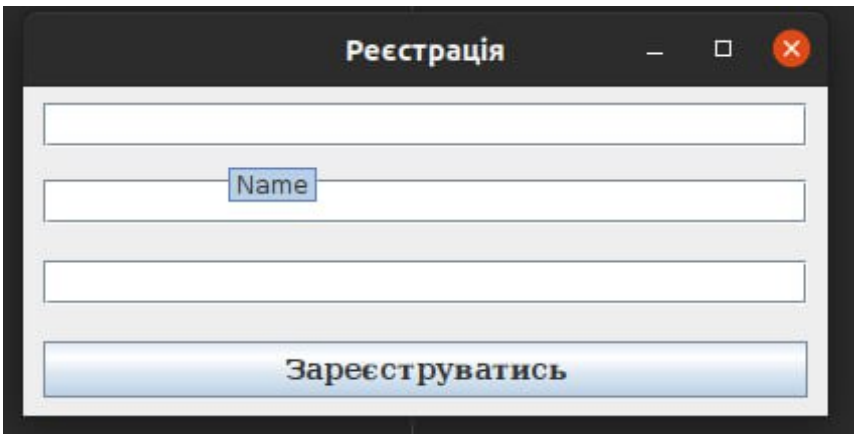


Рисунок Г.3 – Вікно для реєстрації користувача

3. Після авторизації користувач переходить до Меню, де запропоновано пройти тестування, зробити роботу над помилками, переглянути словник або ж статистику.

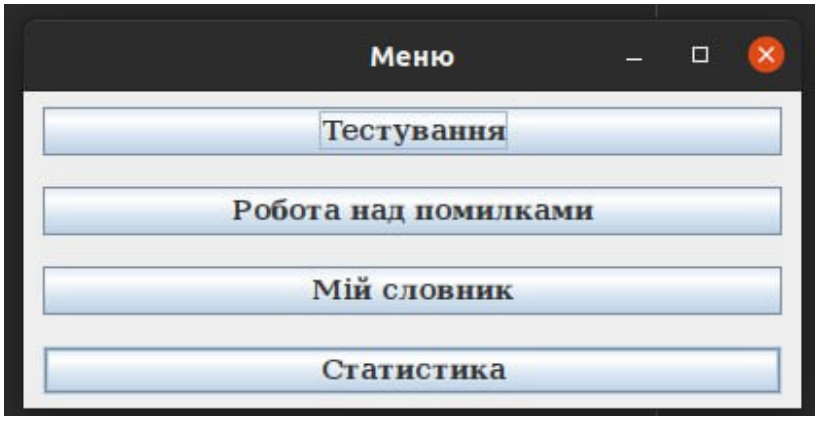


Рисунок Г.4 – Меню можливих задач

4. Якщо обрати тестування відкриється меню в якому можна обрати налаштування тесту, тобто назву тематики, тип, а також складність.

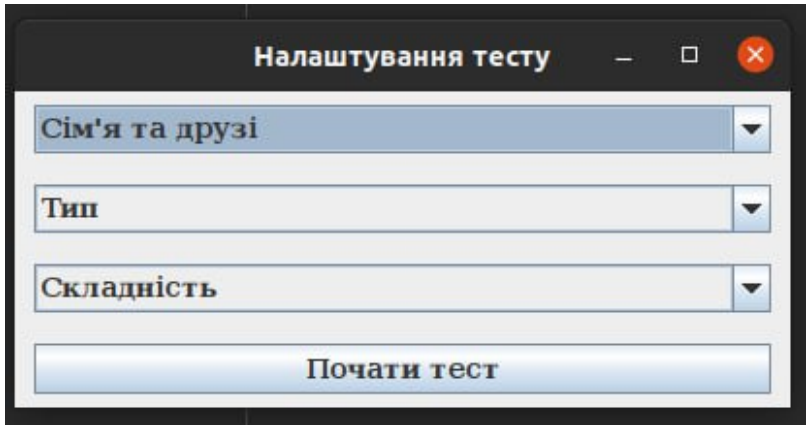


Рисунок Г.5 – Налаштування тесту

5. Тип тесту відповідає за вибір мов, якими будуть представлені слова в тесті. Складність включає в себе три варіанти: новачок, мідел та профі.

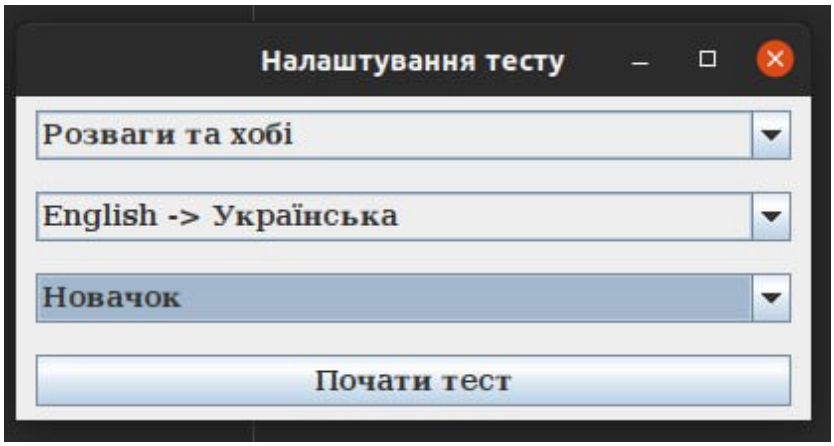


Рисунок Г.6 – Вибір типу тесту

6. Далі відображається вікно тестування, де відбувається вибір правильної відповіді. Також є можливість додати слово до словника за допомогою клавіші «Додати в мій словник». Тест можна завершити в будь-який момент натиснувши «Завершити тест».

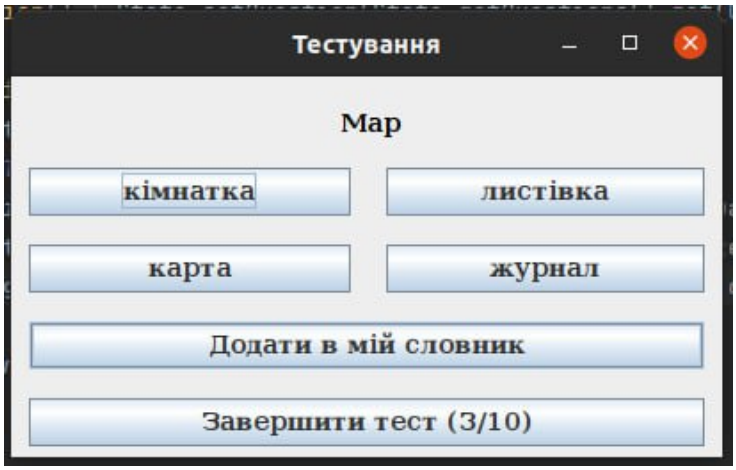


Рисунок Г.7 – Вікно тестування

7. Якщо обраний варіант відповіді виявився правильним, користувач отримує повідомлення зображене на рисунку Г.8.

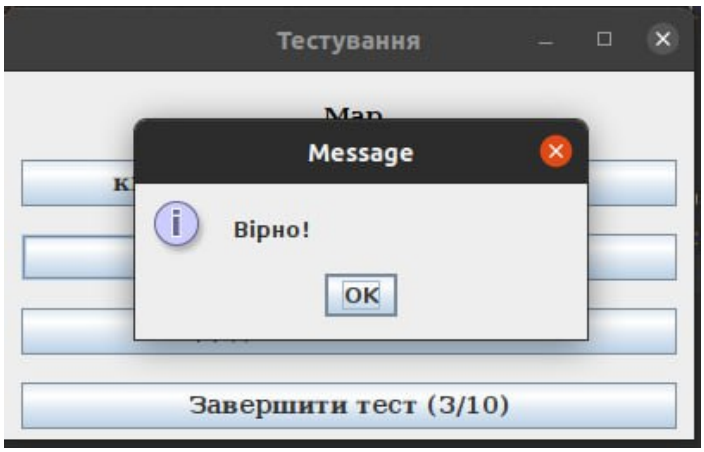


Рисунок Г.8 – Повідомлення про те, що відповідь була правильною

8. У тому випадку, якщо відповідь, відмічена користувачем виявилася хибною отримується інше повідомлення, яке зображено на рисунку Г.9

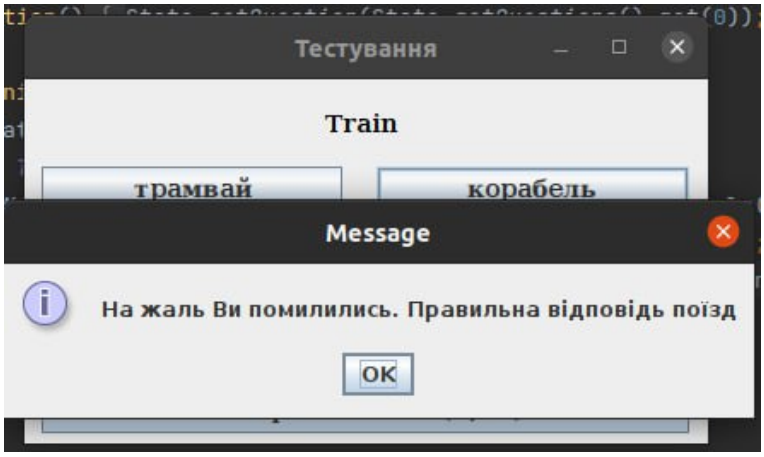


Рисунок Г.9 – Повідомлення про те, що відповідь хибна

9. При виході з тесту отримується повідомлення про результат проходження

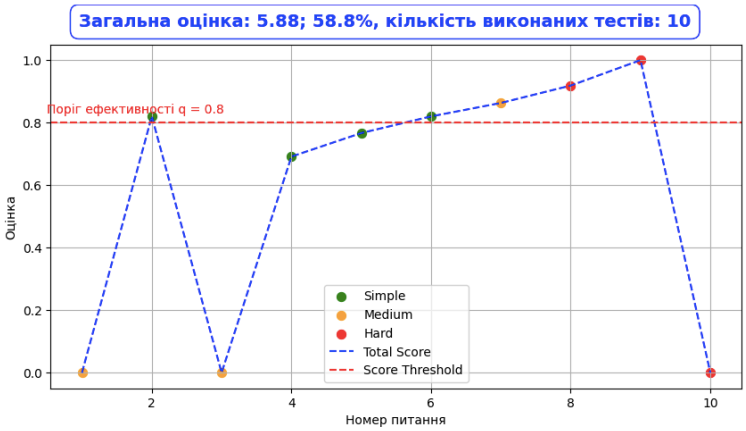


Рисунок Г.10 – Повідомлення про результат тестування

10. Результат проходження тесту також формується у вигляді розширеної таблиці (рисунок Г.11)

Username	Question	Difficulty	Correct	Response Time	Efficiency Time	Score
user_123	1	medium	False	11.225377	0.672875	0.000000
user_123	2	simple	True	2.625550	1.000000	0.820000
user_123	3	medium	False	7.270839	1.000000	0.000000
user_123	4	simple	True	5.123345	0.678672	0.691469
user_123	5	simple	True	3.976080	0.866739	0.766696
user_123	6	simple	True	3.197377	1.000000	0.820000
user_123	7	medium	True	8.527737	0.882557	0.863023
user_123	8	hard	True	19.959483	0.796538	0.918615
user_123	9	hard	True	8.715644	1.000000	1.000000
user_123	10	hard	False	18.777089	0.821122	0.000000

Рисунок Г.11 – Статистика користувача по проходженню тесту

11. Додатково користувач може переглядати слова у своєму індивідуальному словнику, які він додав до списку для повторення

Рисунок Г.12 – Словник користувача

Протестувавши програмне забезпечення, робимо висновок про коректність його роботи у відповідності до поставлених вимог.