Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів»

Виконав: студент II курсу
групи 3ПІ-22м спеціальності
121 – Інженерія програмного забезпечення
Кучерявий Ігор Володимирович

Керівник: к.т.н., доц. каф. ПЗ Романюк О.В.
«08» грудня 2023 р.

Опонент: к.т.н., доц. каф. ЗІ Куперштейн Л.М.
«08» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«08» грудня 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О. Н.
«19» вересня 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Кучерявому Ігорю Володимировичу

1. Тема роботи – Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів.

Керівник роботи: Романюк Оксана Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. №247.

2. Строк подання студентом роботи: 5 грудня 2023 р.

3. Вихідні дані до роботи: кольоровий режим – True Color; максимальний розмір дискретного координатного простору – 4096*4096; шаблони проєктування програмного забезпечення – GOF, MVC; методи автоматизації персоналізованого помічника – Rule-based System; автоматизація збірки проєктів – Maven; система управління базами даних – PGAdmin; інструменти на базі штучного інтелекту – Google Bard, Google API Speech-to-text; мова запитів – SQL; тестування додатка – модульні тести; моделювання програмної системи – UML; вхідні дані – ключ для Google API, скрипт для заповнення навчальним матеріалом середовище; вихідні дані – персоналізований помічник у виправленні помилок вимови слів, слова які було неправильно вимовлено, що використовуються у грі для вивчення правильної вимови слів; середовище розробки – Intelij IDEA; мови програмування – Java, оцінка ефективності – коефіцієнт узгодженості думок експертів.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз і порівняльна характеристика сучасних методів для вивчення правильної вимови слів; розробка методів, алгоритмів та інтерфейсу користувача для програмного застосунку; розробка програмних засобів для підвищення ефективності вивчення правильної вимови слів; тестування програмного забезпечення; економічна частина; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний ... мета, предмет та об'єкт дослідження; задачі дослідження; новизна і практ... цінність отриманих результатів; порівняльна характеристика застосунків; ме... та блок-схема алгоритму динамічної генерації персоналізованої підказки ... студента; метод та блок-схема алгоритму гейміфікації вивчення правил... вимови слів; структура графічного інтерфейсу; проєктування програм... забезпечення; використані технології під час розробки застосунку; модул... тестування; тестування динамічної генерації персоналізованої підка... тестування гейміфікованого процесу навчання; аналіз ефективності розробле... методів; економічне обґрунтування; апробація та публікації результатів робо... фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О.В., к.т.н., доцент. кафедри ПЗ	19.09.2023	05.12.2023
5	Причепа І.В. к.е.н., доцент кафедри ЕПВМ	13.11.2023	01.12.2023

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітки
1	Аналіз і порівняльна характеристика сучасних методів для вивчення правильної вимови слів	20.09.2023 – 30.09. 2023	Вик
2	Розробка методів і алгоритмів програмного продукту	01.10. 2023 – 10.10. 2023	Вик
3	Розробка програмних засобів для підвищення ефективності вивчення правильної вимови слів	11.10. 2023 – 25.10. 2023	Вик
4	Аналіз ефективності запропонованих методів та тестування програмного забезпечення	26.10.2023 – 20.11.2023	Вик
5	Економічна частина	21.11.2023 – 01.12.2023	Вик

Студент _____ **Кучерявий І.В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Романюк О.В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Кучерявий І.В. Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023, 104 с.

На укр. мові. Бібліогр.: 44 назв; рис.: 41; табл.: 15.

Магістерська кваліфікаційна робота присвячена підвищенню ефективності вивчення правильної вимови слів. Проведено аналіз задачі вивчення правильної вимови слів та сучасних методів вивчення вимови. Виконано порівняння різних наявних програмних рішень, призначених для вивчення правильної вимови слів.

У магістерській кваліфікаційній роботі вперше запропоновано метод динамічної генерації підказки студенту, який з використанням штучного інтелекту враховує попередні помилки його вимови, та удосконалено метод гейміфікації вивчення правильної вимови слів, який формує список таких слів для вивчення, які викликали раніше труднощі при вивченні, але з урахуванням прогресу студента, що дозволило підвищити ефективність вивчення правильної вимови слів. Розроблено блок-схеми алгоритмів реалізації запропонованих методів.

Програмне забезпечення для вивчення правильної вимови слів розроблене з використанням UML діаграми. Для розробки програмного забезпечення використано мову програмування Java, середовище розробки IntelijIDEA, фреймворк Spring. Для створення інтерфейсу користувача використано шаблонізатор Thymeleaf та фреймворк Bootstrap. Отримані в магістерській кваліфікаційній роботі результати можна використати для побудови системи для вивчення правильної вимови слів.

Ключові слова: вивчення вимови, веб-застосунок, динамічна генерація, штучний інтелект.

ABSTRACT

UDC 004.42

Kucheryavy I.V. Development of methods and software tools to improve the effectiveness of learning the correct pronunciation of words. Master's qualification work on specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023, 104 p.

In Ukrainian speech Bibliography: 44 titles; Fig.: 41; tab.: 15.

The master's thesis is devoted to increasing the effectiveness of learning the correct pronunciation of words. An analysis of the task of learning the correct pronunciation of words and modern methods of learning pronunciation was carried out. A comparison of various existing software solutions designed for learning the correct pronunciation of words was performed.

In the master's qualification thesis, for the first time, a method of dynamically generating a hint for a student, which takes into account previous mistakes in his pronunciation using artificial intelligence, was proposed, and a method of gamification of learning the correct pronunciation of words was improved, which forms a list of such words for study that previously caused difficulties during study, but taking into account the progress of the student, which made it possible to increase the effectiveness of learning the correct pronunciation of words. Block diagrams of algorithms for implementing the proposed methods have been developed.

The software for learning the correct pronunciation of words is developed using the UML diagram. The Java programming language, the IntelijIDEA development environment, and the Spring framework were used to develop the software. The Thymeleaf templating tool and the Bootstrap framework were used to create the user interface. The results obtained in the master's qualification work can be used to build a system for learning the correct pronunciation of words.

Keywords: pronunciation learning, web application, dynamic generation, artificial intelligence.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. На сьогодні вимова є важливим аспектом вивчення мови, оскільки вона може вплинути на те, наскільки добре вас розуміють інші. Технології штучного інтелекту можна використовувати, щоб допомогти учням покращити свою вимову, забезпечуючи зворотний зв’язок у реальному часі. З розвитком технологій штучний інтелект (ШІ) поступово інтегрувався в різні сфери, включаючи освіту.

Штучний інтелект відіграє ключову роль у вивченні мови, особливо в покращенні вимови. Сучасні інструменти розпізнавання мовлення, як-от Google Speech-to-Text або Apple Siri, стають досконалими в розумінні акцентів і нюансів у мовах. Їх можна використовувати для визначення помилок та допомозі у їх виправлення. Порівнюючи вимову студента зі стандартною вимовою, студент може отримувати постійний зворотний зв’язок.

Обробка природної мови (NLP): Технології ОПМ можуть розуміти контекст, що є вирішальним при роботі з омонімами або словами, які вимовляються по-різному залежно від їх значення. AI може адаптувати уроки на основі індивідуальних потреб учнів. Наприклад, якщо учень постійно неправильно вимовляє певний звук або слово, ШІ може запропонувати цільові вправи, щоб допомогти виправити цю конкретну помилку. Використовуючи ШІ, можна забезпечити візуальний зворотний зв’язок щодо вимови. Наприклад, показати, як повинні рухатися язик і губи для вимови певних звуків.

Вимова залишається ключовим елементом опановування мовою, оскільки в загальному розумінні мова вивчається саме для спілкування. Не важливо яка ціль у студента: вивчення мови для подорожей, для просування по кар’єрних сходах, здача екзаменів чи звичайне бажання володіти великою кількістю мов. Без розуміння як вимовляти слова - потік слів може перерости в хаотичний набір звуків, які не зможе зрозуміти носій мови. Тому, правильна вимова необхідна для ефективного спілкування будь-якою мовою. Однак освоєння вимови незнайомих звуків або слів може бути складним для багатьох учнів.

В останні роки геймізація набула популярності в освітньому секторі як спосіб покращити традиційні методи навчання та зробити навчання більш захопливим і приємним для студентів. Статистика говорить сама за себе: гейміфікація може збільшити залученість на 85% [1], а запам’ятовування вивченого матеріалу може підвищитися на 90%. Крім того, згідно з опитуванням Kahoot приблизно 97% викладачів вважають, що ігровізація покращує залучення учнів [2]. Також 83% співробітників, які проходять гейміфіковане навчання стверджують, що вони є більш мотивованими на роботі [3]. Дослідження також виявили, що геймізація може покращити мотивацію та результати навчання з різних предметів, включаючи природничі науки, математику, програмування та вивчення мов. Включаючи ігрові елементи такі як: бали, значки та таблиці лідерів, у свої навчальні стратегії, викладачі можуть створити більш захопливий та інтерактивний досвід навчання, який підтримує мотивацію та залучення студентів.

У зв’язку з цим важливою задачею є розробка методів для підвищення ефективності вивчення правильної вимови слів. Використання штучного інтелекту для перевірки правильної вимови та створення персоналізованого помічника підвищить ефективність вивчення, а додавання елементів гри в неігрових контекстах підвищить зацікавленість й мотивацію студентів.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалась згідно з планом виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення ефективності вивчення правильної вимови слів шляхом запровадження динамічної генерації персоналізованої підказки для студентів, які мають проблеми із вимовою слів та створення гейміфікованого середовища для студентів, що дозволить підвищити їхню залученість в навчанні.

Основними задачами дослідження є:

– провести дослідження сучасних методів вивчення правильної вимови слів;

– розробити метод динамічної генерації персоналізованої підказки для студента при вивченні правильної вимови слів;

– розробити метод гейміфікації вивчення правильної вимови слів;

– створити блок-схеми алгоритмів для розроблених методів;

– розробити структуру графічного інтерфейсу;

– виконати проєктування програмного забезпечення;

– розробити програмні компоненти для запропонованих методів;

– провести тестування розробленого програмного засобу;

– провести дослідження ефективності розроблених методів.

Об’єкт дослідження – процес вивчення правильної вимови слів.

Предмет дослідження – методи та програмні засоби для вивчення правильної вимови слів.

Методи дослідження. У процесі дослідження використовувались: математична логіка, теорія множин для розробки методу динамічної генерації персоналізованої підказки для студента при вивченні правильної вимови слів; теорія навчання для розробки методу гейміфікації процесу вивчення правильної вимови; теорія алгоритмів для розробки та вдосконалення алгоритмів ПЗ; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень; методи обробки експертної інформації для дослідження ефективності розроблених методів.

Новизна отриманих результатів.

1. Уперше запропоновано метод динамічної генерації персоналізованої підказки для студента при вивченні правильної вимови слів, який за допомогою штучного інтелекту враховує та аналізує попередні помилки студента, що дозволить підвищити ефективність вивчення правильної вимови саме тих слів, які були складними для вивчення.

2. Подальшого розвитку отримав метод гейміфікації вивчення правильної вимови слів, який, на відміну від відомих, формує для вивчення список тих слів, які становили найбільшу складність при вивченні, таким чином враховується прогрес студента, що дає можливість підвищити зацікавленість студентів та

ефективність вивчення правильної вимови слів.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та програмні засоби для підвищення ефективності вивчення правильної вимови слів.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: гейміфікація в освіті на прикладі програмної платформи для вивчення мов Duolingo [4]; переваги й недоліки сучасної онлайн-освіти: Погляд здобувачів освіти [5]; дослідження архітектурного патерну MVC на прикладі мови Java [6]; аналіз використання технологій штучного інтелекту у вивченні правильної вимови [7]; розробка методу створення автоматизованого помічника для виправлення мовленнєвих помилок при вивченні іноземної мови [8].

Апробація матеріалів магістерської кваліфікаційної роботи. Результати роботи доповідалися на таких конференціях:

– LII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (Вінниця, 2023);

– LII Науково-технічній конференції Інституту Конфуція (Вінниця, 2023);

– Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2023)

– XVI Міжнародна науково-практична конференція Інформаційні технології і автоматизація (Одеса, 2023);

– Міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Суми/Вінниця, 2022).

Публікації. Основні результати досліджень опубліковано в 5 наукових працях у збірниках матеріалів конференцій.

1 АНАЛІЗ І ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА СУЧАСНИХ МЕТОДІВ ДЛЯ ВИВЧЕННЯ ПРАВИЛЬНОЇ ВИМОВИ СЛІВ

1.1 Аналіз задачі правильної вимови слів

На сьогодні вимова є важливим аспектом під час вивчення мови, оскільки вона може вплинути на те, наскільки добре вас розуміють інші. Студенти, які вивчають розмовну англійську або будь-яку іншу розмовну мову, стикаються з проблемами у вимові. Вимова – це одна важлива річ, яку повинні опанувати студенти, які вивчають іноземну мову.

Якщо людина вивчає іноземну мову в дитинстві, вона навчається говорити нею вільно і без акценту, але якщо розпочинає в дорослому віці, дуже малоймовірно, що зможе досягнути рівня рідної мови. Спостереження свідчать про те, що ті хто розпочинає вивчати іноземну мову після шкільних років [9], відчувають серйозні труднощі з набуттям правильної вимови, оскільки ступінь труднощів помітно зростає. Ці труднощі не мають нічого спільного з рівнем освіти чи рівнем інтелекту, чи навіть зі знаннями іноземної граматики та словникового запасу. Оскільки немає однозначної відповіді, в чому складність вивчення вимови.

Багато тих, хто вивчає англійську мову як другу, зустрічають труднощі з англійською вимовою слів після багатьох років вивчення. Це часто призводить до проблем під час пошуку роботи. Оскільки існує стандарт професіоналізму, який погіршує процес спілкування в колективі з людьми, які мають проблеми із вимовою [10].

Хоча кожна мова має тисячі слів, не всі вони часто вживаються. Зазвичай можна застосовувати принцип Парето, а саме правило 80/20, згідно з яким розуміння 20% найпоширеніших слів може допомогти учню зрозуміти приблизно 80% повсякденних розмов [11]. Тому проблема із вимовою може виникнути через слабкий словниковий запас. Отже, слова, які є важливими для щоденної розмови, можуть відрізнятися від тих, що застосовуються в академічних чи професійних умовах, що додає плутанину у вимові та розумінні

почутих слів студенту, якому необхідна для навчання іноземна мова, чи працівнику який використовує в роботі іноземну. Тому під час навчання варто створити списки популярних слів, щоб визначити їх пріоритетність, які найчастіше вживаються в цільовій мові.

У кожній мові є ідіоми чи приказки, які при буквальному перекладі інколи звучать абсурдно. Зловживання ними може призвести до плутанини або ненавмисного гумору. Слова, які посилаються на історичні події, можуть мати різні конотації в різних країнах, а неправильне їхнє вживання може свідчити про незнання. Також деякі теми можуть вважатися прийнятними в одній культурі, але небажаними в іншій, де їх варто уникати.

Деякі мови, а саме: мандарин, панджабі, в'єтнамська мають різні значення для одного слова залежно від його тону [12]. Тональні мови – це мови, де тон і вимова працюють разом, щоб передавати значення слів. Незначна неправильна вимова може повністю змінити значення слова. Звуки, присутні в одній мові, можуть не існувати в іншій, що ускладнює їх вимову для учнів, таким чином неправильний наголос на складі або неправильна інтонація можуть призвести до того, що правильно вимовлене слово звучить чужоземно.

Тому для вивчення правильної вимови слів потрібно слідувати наступним правилам:

1. Варто перевіряти та виправляти наголошення окремих слів: якщо студент зазвичай розмовляє з носіями англійської мови, це буде причина номер один, чому вони його неправильно розуміють.

2. Необхідно перевіряти наголошення слів у реченні: потрібно пам'ятати, що зміст речення може повністю змінитись, роблячи різний наголос в ньому на різні слова. Наприклад, речення можна трактувати різними способами: «Я не казав, що ми повинні їхати цим шляхом». Якщо наголос в реченні йде на «Я», значення речення – що йти цим шляхом не була ваша ідея [13]. З іншого боку, якщо наголос на «їхати», значення речення – що не варто було їхати на транспорті. Таким чином, якщо не звертати пильну увагу на слова, які наголошуються, можна надати іншу інформацію, протилежну від бажаної.

3. Необхідно групувати слова за темами, ситуаціями або частотою вживання, щоб зробити навчання більш практичним. Слова варто упорядковувати залежно від того, як часто вони з’являються в певному контексті, наприклад у повсякденних розмовах, літературі, газетах чи академічних текстах. Ці списки стануть корисними та ефективними для тих, хто розпочинає вчити мову, оскільки вони можуть зосередити свою енергію на засвоєнні слів, які їм, швидше за все, зустрінуться. Головною перевагою такого підходу є ефективність під час навчання та швидке розуміння. Замість того, щоб витрачати сили на тисячі слів, учні можуть зосередитися на певній множині слів, що забезпечить максимальну користь. Знаючи слова, які зустрічаються, учні зможуть розуміти більшість розмов та текстів, навіть якщо вони не зрозуміють частину слів.

4. Для покращення залучення та підвищення мотивації учнів необхідно інтегрувати ігрові елементи у неігровий навчальний контекст вивчення мови: Цей підхід використовує внутрішнє людське бажання конкуренції, досягнень і винагороди. Ігрові елементи, як бали чи значки, можуть мотивувати учнів продовжувати практику вивчення правильної вимови слів. Постійне залучення: виклики, рівні та місії, допоможуть залучати учнів на тривалий час. Ігри забезпечують миттєвий зворотний зв’язок, допомагаючи учням швидко розпізнавати та виправляти помилки, а безризикове середовище дозволить учням робити помилки без реальних наслідків, сприяючи більш спокійному навчальному середовищу.

5. Варто перевіряти, як вимовляти фонеми різних мов: Фонеми – це окремі звукові одиниці в мові, які відрізняють одне слово від іншого [14]. Не всі фонеми присутні в одній мові існують в іншій. Вимовляти незнайомі фонеми учням, може бути складно. Наприклад, звук «th» в англійській мові не існує в багатьох мовах, що призводить до заміни на «d» або «z». Спочатку учні можуть навіть не почути різниці між незнайомими фонемами. Гарним підходом для розв’язання цієї проблеми є детальний опис або зображення, де і як розташувати язик, губи та зуби, а також варто забезпечити миттєвий зворотний зв’язок учням, щодо

правильності їхньої вимови

Отже, варто дотримуватись усіх вище зазначених правил під час вивчення правильної вимови слів, щоб сприяти максимальній структуризації та ефективності під час навчання. Розв’язуючи проблему неправильної вимови на глибшому рівні, студенти зможуть не лише вивчити мову, але й ефективно й шанобливо по відношенню до інших культур користуватись нею в повсякденному житті. Крім того, розпізнавання голосу та інтерактивний зворотний зв’язок, допоможе значно покращити навички вимови учнів які вивчають іноземні мови.

1.2 Аналіз сучасних методів вивчення правильної вимови слів

Штучний інтелект вже застосовується в освіті, в інструментах, які допомагають розвивати різні навички, та системах їх тестування. Оскільки освітні рішення, які використовують штучний інтелект, продовжують розвиватися, ШІ зможе допомогти заповнити прогалини в навчанні та викладанні [15], що дозволить школам і вчителям навчати ефективніше. ШІ може підвищити ефективність, покращити персоналізацію та оптимізувати завдання, які дає вчитель учням, щоб дати їм більше часу і свободи для завдань, які потребують унікальних людських здібностей, з якими машинам було б важко.

ШІ може допомогти з вивченням вимови кількома способами [16], зокрема:

Забезпечення зворотного зв’язку в режимі реального часу: інструменти розпізнавання мовлення на основі штучного інтелекту можуть аналізувати мовлення учня та надавати відгуки щодо його вимови в реальному часі. Цей зворотний зв’язок може допомогти учням визначити та виправити свої помилки.

Створення персоналізованих інструкцій: ШІ можна використовувати для створення персоналізованих інструкцій для учнів на основі їхніх індивідуальних потреб. Наприклад, репетитор з вимови на основі штучного інтелекту може визначити конкретні звуки, з якими учень відчуває труднощі, і надати йому цілеспрямовані інструкції.

Створення захопливого навчального досвіду (engaging learning experiences): штучний інтелект можна використовувати для створення захопливого навчального досвіду, який робить вивчення вимови веселішим та ефективнішим. Наприклад, гра на основі штучного інтелекту може запропонувати учням правильно вимовляти слова, щоб прогресувати в грі. Можливість бачити свій результат зробить учнів більш вмотивованими, а змагальний дух навчальної гри допоможе учням створити атмосферу цікавого й одночасно ефективного навчання.

На основі проведеного дослідження 90% студентів віддали перевагу штучному інтелекту [17], а не справжнім вчителям. Крім того, більшість повністю замінили деякі уроки з вчителями на навчання разом із штучним інтелектом, а 95% стверджують, що подальше підвищення оцінок відбулося лише завдяки репетиторству штучного інтелекту. Математика та природничі науки були найпоширенішими предметами, які замінили репетиторів штучним інтелектом.

Гейміфікація в освіті – це техніка, яка використовує елементи гри в неігрових контекстах для підвищення зацікавленості, мотивації та результатів навчання учнів. Ігровізація або використання ігрового дизайну та механіки в неігрових контекстах останнім часом привертає все більше уваги в галузі освіти. Цей підхід до навчання використовує мотиваційну силу ігор, щоб залучити учнів покращити результати навчання. Доведено, що гейміфікація в освіті підвищує залученість студентів, мотивацію та запам’ятовування вивченого матеріалу. Необхідно дослідити переваги та недоліки гейміфікації в освіті, а також розглянути успішність проєктів, які викладачі можуть використовувати для покращення досвіду навчання.

Головними перевагами методу ігровізації навчання є:

1. Підвищена мотивація: ігровізація може зробити навчання більш приємним і мотивуючим, використовуючи вроджене прагнення людини до досягнень і конкуренції.

2. Персоналізоване навчання: ігрове навчання можна налаштувати

відповідно до індивідуального стилю та темпу навчання кожного учня, дозволяючи їм навчатися у комфортному для них темпі.

3. Миттєвий зворотний зв'язок: гейміфікація забезпечує миттєвий зворотний зв'язок щодо успішності студентів, дозволяючи їм визначити сфери, де їм потрібно покращити, і відповідно скоригувати свої навчальні стратегії. Ігровізація робить навчання веселим та захопливим, що може збільшити залученість студентів та інтерес до предмета.

4. Краще запам'ятовування: гейміфікація може допомогти учням запам'ятати інформацію, надаючи більш привабливий і незабутній досвід навчання. А симуляція контексту реального світу, де використовується мова, допомагає студентам застосовувати те, що вони навчилися, у практичних ситуаціях.

5. Доступність: геймізацію можна використовувати, щоб зробити навчання більш доступним для студентів з обмеженими можливостями або труднощами в навчанні.

6. Позитивна атмосфера: ігровізація забезпечує позитивну атмосферу за хорошу поведінку та продуктивність, що може підвищити самооцінку та впевненість учнів. Помилки в грі не мають реальних наслідків, що може зменшити тривогу та страх невдачі. Це може бути особливо корисним для тих, хто вивчає мову, але може не наважуватися говорити вголос через страх неправильної вимови слів.

Основними недоліками підходу ігровізації є:

1. Поверхневе навчання: геймізація може надавати перевагу отриманню балів або значків, що не є рівним отриманню глибокому розумінню і застосуванню предмета на практиці.

2. Короткострокова мотивація: гейміфікація може забезпечити лише короткострокову мотивацію, оскільки учні можуть втратити інтерес після досягнення певного рівня чи мети.

3. Надмірна залежність від технологій: гейміфікація може надмірно покладатися на технології, що призводить до відсутності особистої взаємодії та

особистого зв’язку між студентами та викладачами.

4. Відволікання: геймізація може створювати відволікання та спокуси, які заважають навчанню, наприклад витрачати більше часу на ігри, ніж на навчання.

Підсумовуючи, технології штучного інтелекту та гейміфікація мають величезний потенціал в сучасній освіті. Поєднання обох методів, дозволяє створити привабливий та ефективний інструмент для покращення вимови та зробити процес навчання цікавим.

1.3 Порівняльний аналіз сучасних програмних рішень для вивчення правильної вимови

Вимова є вирішальним аспектом вивчення мови та спілкування. Протягом багатьох років було розроблено багато інструментів і програм, які допомагають людям вивчити мову, але існує не багато додатків для вивчення її вимови. Тому варто провести аналіз найбільш популярних і ефективних програм і інструментів, доступних для навчання:

Forvo – це веб-застосунок, який дозволяє користувачам почути, як слова вимовляються носіями мови з усього світу [18]. Головними перевагами програмного засобу є:

Вимова слів: основою Forvo є записи з вимовою слів. Коли учень шукає слово, він отримає кілька варіантів вимови різними носіями мови.

Власна спільнота: платформа покладається на свою спільноту, щоб додавати та перевіряти вимову. Користувачі можуть запитувати вимову певних слів, а носії мови можуть потім завантажити відповідне аудіо.

Наявність фраз для подорожей: платформа надає спеціальний розділ, який зосереджується на основних фразах, корисних для мандрівників, які допомагають як у розумінні, так і у вимові. Поряд з окремими словами користувачі також можуть прослуховувати певні фрази різними мовами.

Різноманітність голосів: оскільки кілька користувачів можуть завантажувати вимову одного слова, користувач може почути варіації, що є особливо корисним для мов із різними діалектами.

Підсумовуючи, Forvo не є повним рішенням для вивчення мови, але це цінний інструмент для тих, хто потребує роз’яснення того, як правильно вимовити слово. Інтерфейс застосунку Forvo представлений на рисунку 1.1.

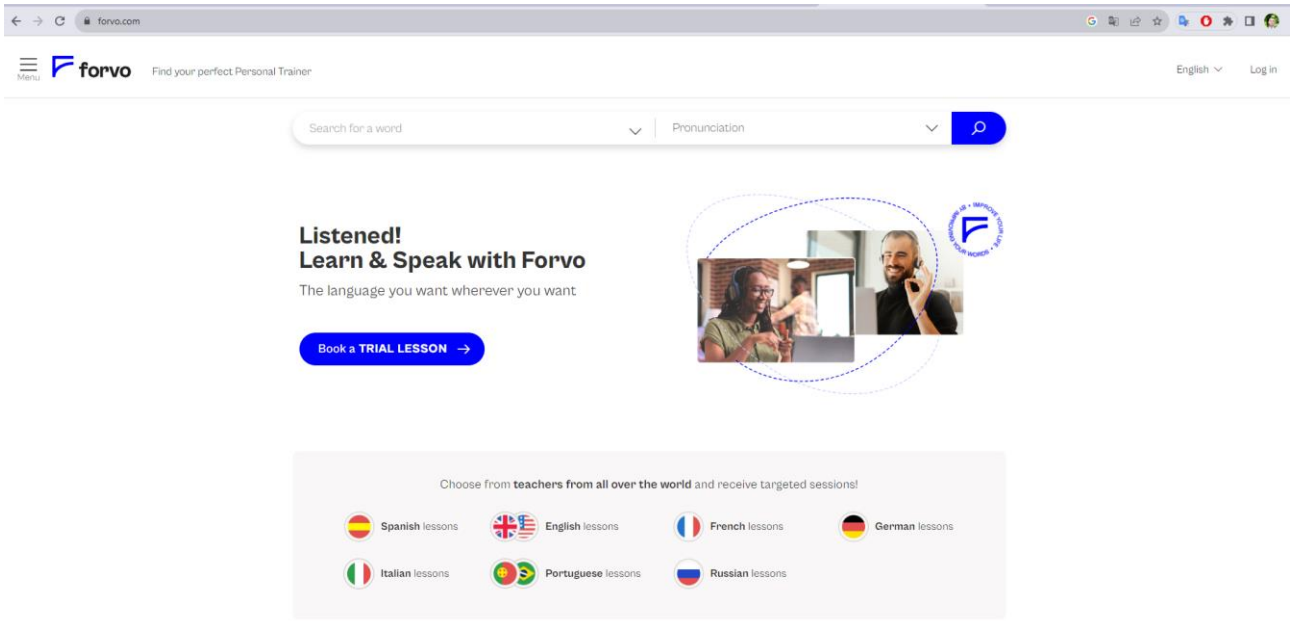


Рисунок 1.1 – Інтерфейс веб-застосунку Forvo

Anki – це універсальний застосунок для карток із можливістю їх налаштування. Застосунок використовує підхід повторення з інтервалами, щоб допомогти користувачам запам’ятати нову інформацію. Програмний засіб популярний серед тих, хто вивчає мови, студентів-медиків і всіх, кому потрібно запам’ятовувати величезні обсяги інформації протягом тривалого часу.

Головними особливостями застосунку Anki є:

Колоди, які можна налаштувати: користувачі можуть створювати свої колоди або завантажувати спільні колоди зі спільноти Anki. Ці колоди можуть містити текст, зображення, аудіо та навіть відео.

Система інтервального повторення (SRS): Anki використовує науково перевірений метод для покращення збереження пам’яті. Додаток обирає доступні для перегляду картки на основі того, наскільки добре користувач їх запам’ятав.

Теги та спеціальні поля: користувачі можуть позначати картки тегами для

кращої організації та використовувати спеціальні поля для структурування інформації різними способами.

Синхронізація: користувачі можуть синхронізувати свої колоди на кількох пристроях, гарантуючи, що їх прогрес оновлюється всюди.

Професійні терміни: багато професіоналів, наприклад студенти-медики, які готуються до іспитів, використовують Anki для вивчення професійних термінів.

Отже, Anki є потужним інструментом для тих, хто серйозно займається довгостроковим збереженням інформації. Хоча його інтерфейс і широкі можливості налаштування можуть лякати початківців, його ефективність і адаптивність роблять його фаворитом серед тих, хто вкладає час, щоб зрозуміти та використати його на повну. Інтерфейс застосунку Anki представлений на рисунку 1.2.

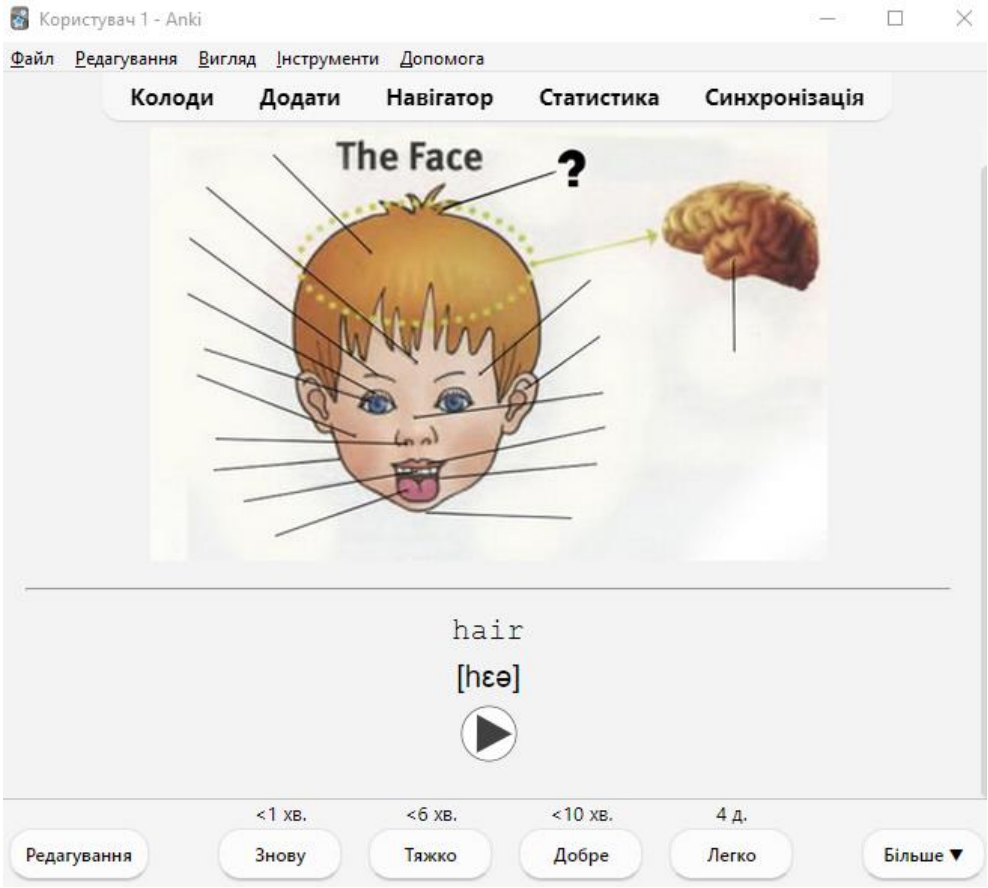


Рисунок 1.2 – Інтерфейс застосунку Anki

Одним із прикладів програми, яка використовує гейміфікацію в освіті, може слугувати Duolingo, програма для вивчення мови. Duolingo робить вивчення мови веселим і захопливим завдяки використанню елементів ігровізації, таких як бали, рівні та нагороди. Користувачі заробляють бали за проходження уроків і можуть підвищувати рівень у міру прогресу. Duolingo також містить систему винагород, як-от відкриття нових уроків і заробіток віртуальної валюти, яку можна використовувати для придбання бонусів [19]. Додаток також містить різноманітні ігрові функції, такі як режим тренування з часовим поясом, таблиця лідерів, де користувачі можуть змагатися з іншими, і функція, яка дозволяє користувачам «дуелювати» з іншими користувачами в змаганні один до одного. Завдяки використанню гейміфікації Duolingo зміг зробити вивчення мови більш привабливим і ефективним. Програма користується великою популярністю з понад 500 мільйонами завантажень по всьому світу і її високо оцінюють за її здатність зробити вивчення мови доступним для ширшої аудиторії. Інтерфейс застосунку Duolingo представлений на рисунку 1.3.

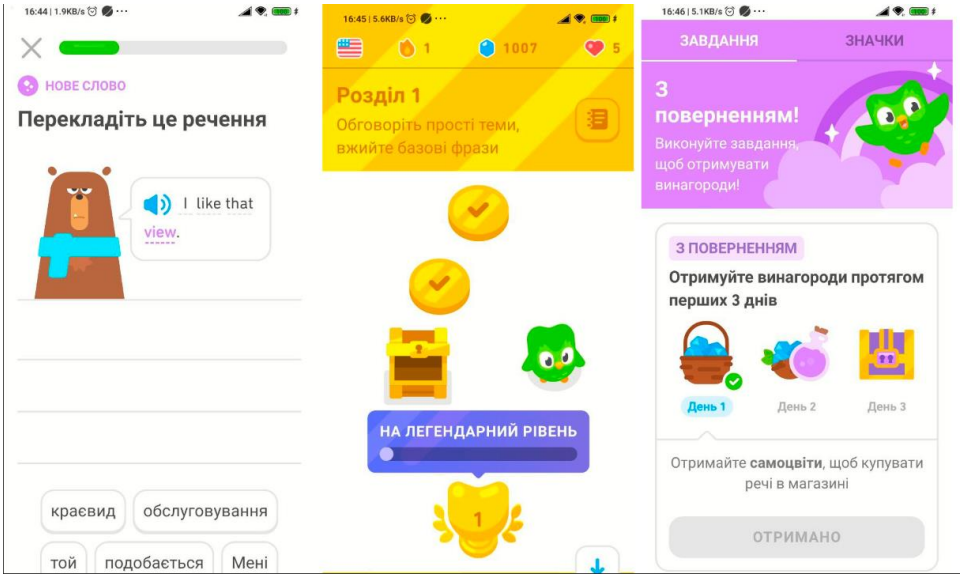


Рисунок 1.3 – Мобільний застосунку Duolingo

Одним з основних способів отримання прибутку для компанії є її підписка на користування Duolingo Plus, яка пропонує користувачам додаткові функції,

такі як навчання без реклами, офлайн-доступ і необмежену кількість помилок (життів), оскільки кожен користувач має їх в обмеженій кількості, але вони поновлюються з часом. Сервіс вийшов на IPO, а у перший день торгів оцінка склала 5 мільярдів доларів [20]. У 2020 році Duolingo отримала 161,7 мільйона доларів США доходу порівняно з 90,9 мільйона доларів у 2019 році. Зростання доходу компанії пов’язано зі збільшенням попиту на онлайн-освіту під час пандемії COVID-19, а також зростанням популярності програм для вивчення мов.

Створимо порівняльну характеристику для результатів аналізу додатків для вивчення правильної вимови. Визначено головні можливості та недоліки додатків, які враховуються при створенні власного веб-додатка з назвою «SpeechLearning» (табл 1.1).

Таблиця 1.1 – Порівняння характеристика застосунків

Критерій	Forvo	Anki	Duolingo	SpeechLearning
Наявність інтерактивного помічника	-	-	+	+
Простота використання	+	-	+	+
Елементи гри в не ігрових контекстах	-	-	+	+
Безкоштовність повного функціоналу	-	+	+	+
Можливість вибору матеріалу для вивчення	+	+	-	+
Багатоплатформність	+	+	-	+
Загальний результат	3	3	4	6

В результаті проведення детального аналізу аналогів, було виділено їхні основні переваги та недоліки, що доводить доцільність розробки програмного

продукту. Варто зазначити, що в результаті виконання застосунок повинен перекривати всі наявні недоліки існуючих рішень. Крім того, застосунок Anki має впливовий недолік у вигляді складності використання початківцями та застарілого інтерфейсу, на що вказують відгуки про нього, тому це необхідно усунути в розробленому рішенні, оскільки це може вплинути на залученість студентів.

1.4 Постановка задачі

На основі проведеного аналізу стану питання щодо вивчення правильної вимови слів, порівняння наявних програмних продуктів за рядом визначених критеріїв і з урахуванням існуючих методів і підходів було визначено такі задачі, які необхідно виконати для розробки програмного забезпечення для підвищення ефективності вивчення правильної вимови слів:

– провести дослідження сучасних методів вивчення правильної вимови слів;

– розробити метод динамічної генерації персоналізованої підказки для студента при вивченні правильної вимови слів;

– розробити метод гейміфікації вивчення правильної вимови слів;

– створити блок-схеми алгоритмів для розроблених методів;

– розробити структуру графічного інтерфейсу;

– виконати проєктування програмного забезпечення;

– розробити програмні компоненти для запропонованих методів;

– провести тестування розробленого програмного засобу;

– дослідити ефективність запропонованих методів.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі проаналізовано задачу правильної вимови слів та необхідність динамічної генерації персоналізованої підказки, яка може бути реалізований використовуючи штучний інтелект. Дослідження сучасних методів

навчання дозволяє обрати необхідний напрям для розв’язання задач, а саме: гейміфікація має потенціал зробити навчання більш захопливим та інтерактивним, але вона також має свої обмеження та недоліки. Важливо ретельно обміркувати переваги та недоліки ігровізації перед впровадженням в освіту та переконатися, що вона відповідає цілям і завданням навчальної програми. Інноваційний підхід до вивчення вимови на основі технологій штучного інтелекту пропонує перспективний набір інструментів як для студентів, так і для викладачів. Це не тільки сприяє ефективному навчанню, але й забезпечує індивідуальний підхід для кожного учня.

2 РОЗРОБКА МЕТОДІВ, АЛГОРИТМІВ ТА ІНТЕРФЕЙСУ КОРИСТУВАЧА ДЛЯ ПРОГРАМНОГО ЗАСТОСУНКУ

2.1 Розробка методу динамічної генерації персоналізованої підказки для допомоги виправлення помилок студентам

В ході виконання аналізу проблеми правильної вимови слів було доведено необхідність реалізації персоналізованого помічника для вимови слів. Тому прийнято рішення розробити власний метод, який буде враховувати пройдений матеріал та зроблені помилки студентом

Метод динамічної генерації персоналізованої підказки має складатись з трьох основних компонентів: модулю розпізнавання вимови, модулю збирання інформації про результати студента та модулю генерації інструкції.

Для імплементації розпізнавання вимови використано Google API Speech to Text – це RESTful API, яке дозволяє перетворювати аудіо файли в текст [21]. API використовує машинне навчання для розпізнавання мови та генерування тексту, який є точним і зручним для читання. У контексті навчання вимови транскрипція вимовлених слів є основним кроком для виявлення помилок вимови шляхом перекладу сказаних студентом слів у вигляд, який можна проаналізувати та використати для генерації підказки. Особливості конфігурації Speech to Text API дозволяють надати список ключових слів, які розробник очікує розпізнати, а також можна додати акценти користувачів. Поєднання цих двох конфігурацій дозволять підвищити точність розпізнавання вимови студента. Крім того, Google API пропонує кілька способів налаштувань параметрів для надійності та швидкості, таким чином активувавши обмеження по швидкості додаток може гарантовано отримувати швидкий результат, але це може вплинути на його якість.

Збирання інформації про студента необхідно реалізувати за допомогою окремої моделі. Створена модель повинна містити інформацію про навчальний матеріал, з яким студент займається, його результати та за необхідності додати текст, який допоможе йому навчатись.

Загальна модель, яка використовується для побудови персоналізованої підказки може бути представлено у вигляді кортежу множин:

$$<Rr, Ct, Nt, Pt, Tz , Uat, La, Mdm, Cnc>, \quad (2.1)$$

де *Rr* – результат розпізнавання вимови; *Ct* – правильно вимовлений текст; *Nt* – неправильно вимовлений текст; *Pt* – повний текст навчального матеріалу; *Tz* – текст запиту для побудови відповіді від помічника; *Uat* – автоматично згенерований текст від помічника; *La* – мова навчального матеріалу; *Mdm* – мінімальна допущена кількість помилок; *Cnc* – кількість зроблених помилок.

Результат розпізнавання вимови – забезпечується інтеграцією сервісу Google API Speech-to-text. Після правильної конфігурації доступними методами сервісу система отримує швидкий та якісно розпізнаний текст.

Правильно вимовлений текст – дотримується умов завдання, відповідно визначається за формулою:

$$Ct = Rr \cap Pt\ . \quad (2.2)$$

Неправильно вимовлений текст – є протилежним правильному тексту, для нього здійснюється умова 2.3, а також відповідає відмінностям умови 2.4.

$$Ct \cap Nt = \emptyset, \quad (2.3)$$

$$Nt = Ct \backslash Pt. \quad (2.4)$$

Текст запиту для побудови відповіді від помічника напряму залежить від кількості зроблених та мінімальної допущеної кількості помилок, а також він повинен містити в собі усі неправильно вимовлені слова, щоб включити їх в розроблену систему правил.

$$Tz = \begin{cases} Tz = Cnc > Mdm \\ Tz \subseteq \mathrm{N}t \end{cases}. \quad (2.5)$$

Отже, визначена множина допоможе представити інформацію з інтерфейсу користувача і передавати дані в функціях для персоналізованої підказки правильної вимови слів.

Модуль генерації інструкції має використовувати дані з представленої моделі, аналізувати слова, в яких зроблено помилки, і надавати коротку інструкцію для учнів. В основі він містить систему на основі правил (Rule-based Systems) – попередньо визначені правила для керування розмовою [22]. Система застосовує створені людиною правила для зберігання, сортування та обробки даних. Роблячи це, він імітує людський інтелект. Системи, засновані на правилах, вимагають набору фактів або джерел даних, а також набору правил для маніпулювання цими даними. Ці правила будемо визначати за допомогою операторів «If», після чого вони, виконуватимуться за наступним правилом «ЯКЩО X станеться, ТО виконайте Y». Отже, спершу необхідно буде визначити можливі варіанти запитів та поєднувати їх з фрагментами, які є унікальними для кожного студента.

Оскільки підказки мають бути персоналізованими та інформативними, варто застосувати штучний інтелект, а саме: розмовний генеративний чат-бот Bard від компанії Google [23]. Bard навчений на великій кількості текстових даних і здатний генерувати текст схожий на відповідь справжньої людини, у такий спосіб зможемо імітувати спілкування з реальним репетитором по вимові, який надаватиме індивідуальні підказки учням, коли вони робитимуть помилки. Хорошою перевагою є його інтернаціоналізація, оскільки додаток має підтримувати кілька мов, в якому необхідно також генерувати інструкції відповідними мовами.

Отже, в підрозділі запропоновано метод динамічної генерації персоналізованої підказки для вивчення правильної вимови слів. Попередньо створивши правила для системи заснованої на правилах й використовуючи інтеграцію двох сервісів від Google, а саме: Google Bard та Google Speech to text в поєднанні зі створеною моделлю для зберігання інформації – досягнуто миттєве отримання персоналізованої інструкції по правильній вимові слів.

Такий метод є перспективним підходом до вивчення правильної вимови. Він може стати цінним інструментом для тих, хто вивчає мову на всіх рівнях.

2.2 Розробка блок-схеми алгоритму динамічної генерації персоналізованої підказки для вивчення правильної вимови слів

На основі методу динамічної генерації персоналізованої підказки для вивчення правильної вимови слів, розроблено блок схему основного алгоритму програми. Оскільки він є ключовим для надання персоналізованої інструкції, необхідно розглянути його детально.

Розроблена блок-схема представлена на рисунку 2.1.

Крок 1. Початок.

Крок 2. Ввід даних – отримано файл вимовлений користувачем, навчальний матеріал, яким займався студент, та визначену кількість помилок, яку може допустити учень перед генерацією підказки.

Крок 3. Зберігання в зміну recognition результатів вимови у формат доступний для аналізу. Функція makeRecognitionAndSaveToResults використовує в собі Google API для транслітерації аудіо файлу в текст.

Крок 4. Перевірка чи дані було розпізнано і чи містить в собі будь-який текст. За умови, що аудіо файл було розпізнано переходимо до кроку 5, інакше до 11, щоб зберегти повідомлення про помилку.

Крок 5. Отримано з результатів неправильно вимовлений текст відповідно того, який вказаний в навчальному матеріалі.

Крок 6. Перевірка чи неправильно вимовлений текст є пустим, оскільки користувач міг вимовити все вірно. Якщо є елементи не правильно вимовленого тексту перехід до кроку 7. Інакше при успішній вимові всіх слів до кроку 8.

Крок 7. Збільшуємо лічильник неправильно вимовлених слів на 1.

Крок 8. Перевіряємо чи лічильник досяг значення, після якого відбувається генерація підказки. За умови, що користувач допустився достатньої кількості помилок переходимо до кроку 9. Інакше переходимо до кроку 13 для кінцевого відображення результату.

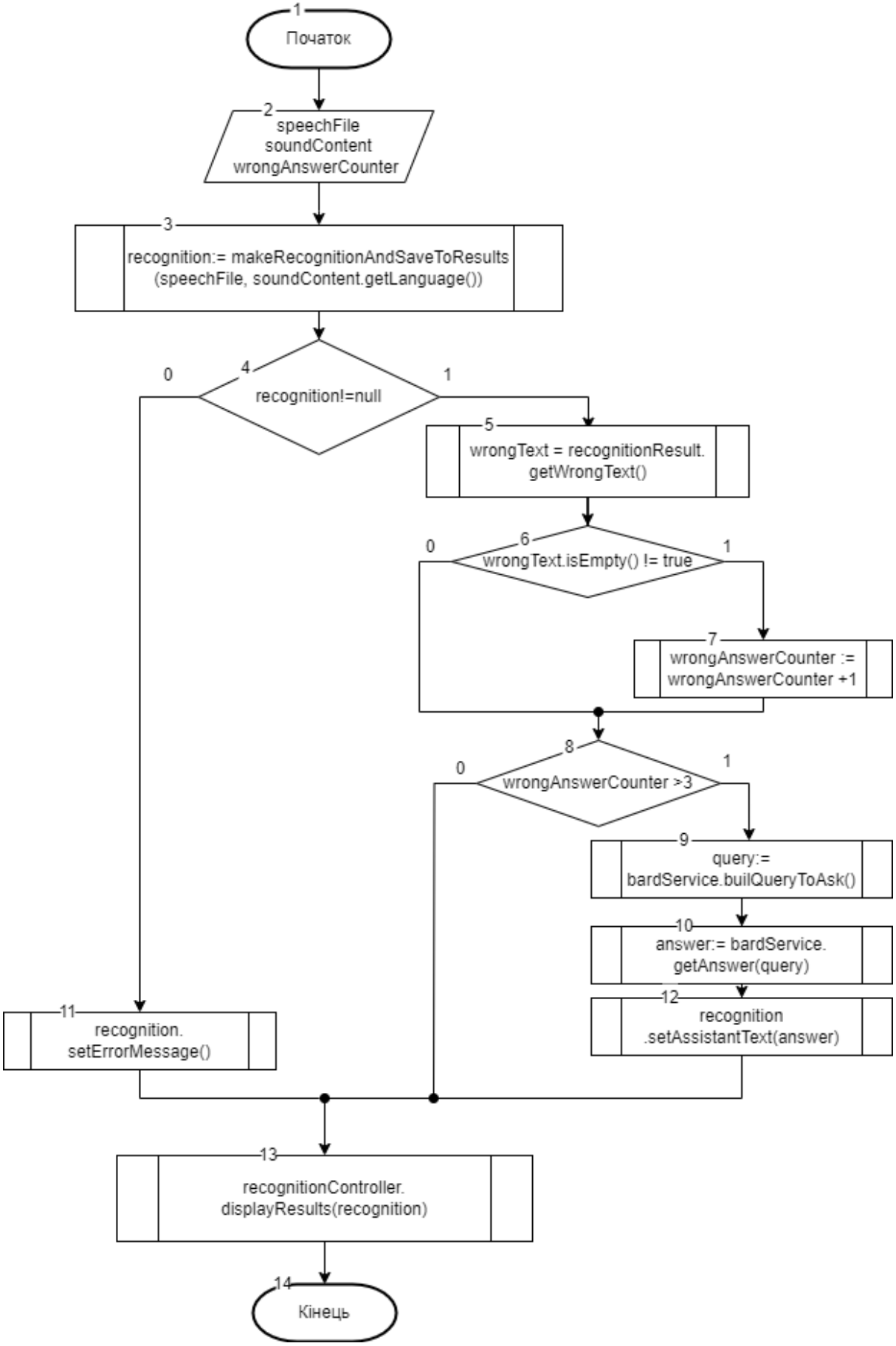


Рисунок 2.1 – Блок-схема алгоритму динамічної генерації персоналізованої підказки для вимови слів

Крок 9. Відбувається побудова запиту на основі визначених заранння правил нашої системи (Rule-based Systems) для того, щоб відправити запит сервісу Google Bard.

Крок 10. Відправка запиту до сервісу Bard та отримання від неї відповіді. Після чого цю відповідь зберігаємо в змінну answer для майбутнього збереження в модель для відображення.

Крок 11. Якщо розпізнавання було невдале, зберігаємо повідомлення про помилку.

Крок 12. Збережену відповідь answer передаємо в модель, яка відобразить її й інтерпретує в інструкцію для студента, який зробив потрібну мінімальну кількість помилок.

Крок 13. Відправка результатів кінцевому користувачу та за умови, що він робив помилки, додатково відображуємо інструкцію як правильно вимовити слова.

Крок 14. Кінець.

Отже, було розглянуто алгоритм динамічної генерації персоналізованої підказки для вивчення правильної вимови слів.

2.3 Розробка методу гейміфікації вивчення правильної вимови слів з урахуванням пройденого матеріалу студентом

Гейміфікація навчального процесу – це інтеграція ігрових елементів, принципів, техніки проєктування в навчальну діяльність і зміст. Щоб досягнути мети зробити навчання більш захопливим, мотивуючим і приємним для учнів. Необхідно використати такі елементи, як: змагання, винагорода, виклик та інтерактивність, щоб створити більш захопливий та інтерактивний досвід навчання.

Для того, щоб підвищити залученість студентів, зробити їхнє навчання веселим і водночас корисним, необхідно інтегрувати змагальний і винагороджувальний характер ігор, який зможе мотивувати учнів брати активну участь в навчанні та виконанні завдань.

Щоб запропонувати учням персоналізоване навчання в грі необхідно адаптувати завдання до індивідуальних потреб і вподобань студента. Адаптивні навчальні платформи можуть регулювати рівень складності завдань і надавати персоналізований зворотний зв’язок на основі результатів навчання. У такий спосіб розроблений метод має застосувати доступну інформацію про зроблені помилки студентом під час вивчення наявного навчального матеріалу та включити цей матеріал в гру.

Миттєвий зворотний зв’язок повинен не лише мати перевірку правильно вимовлених слів, а також включати функції відстеження прогресу, що дозволить учням побачити свій прогрес і поставити цілі для вдосконалення, що може підвищити самоконтроль у вивченні мови та відчуття досягнення цілей. Такий підхід приверне увагу студентів і підтримає їхню мотивацію в вивченні навчального матеріалу.

Використовуючи ігровізовану систему вчитель зможе зібрати дані про успішність студента, що допоможе йому проаналізувати області у яких студент відчуває труднощі. На основі властивостей, які має покривати розроблений метод створено модель гейміфікації вивчення правильної вимови слів у вигляді множини, що дозволить представити необхідну інформацію:

$$<Sw, Ig, Sc, Li, St, Rt, Cp, Wp> , \tag{2.6}$$

де *Sw* – слово або вираз, який необхідно вимовити студенту. Дані мають складатись з зроблених помилок студентом, які доступні під час користування навчальним контентом, у такий спосіб студент зможе акцентувати свою увагу на тренуванні складних для нього областей, що дозволить покращити ефективність вивчення слів;

Ig – відображення інформації про гру;

Sc – загальний рахунок успішно вимовлених слів студентом, показник повинен зміщуватись не на сталу величину, а залежати від складності самого слова для її обрахунку. Формула для визначення Sc:

$$Sc = Sw.lenght + 1; \quad (2.7)$$

Li – кількість життів, яку має ігровий персонаж. Величина яка повинна відображати доступну кількість помилок, що може допустити студент в межах однієї сесії гри. Варто зазначити умову (2.8), за якої сесія гри буде завершена, оскільки це залежить від поточного значення Li:

$$Li > 0 ; \quad (2.8)$$

St – стан персонажів, який залежить від результатів студента. Після розпізнавання вимови та отримання результатів повинна відображатись ігрова анімація, що додатково показуватиме фінальний результат завдання виконаного студентом;

Rt – текст, який є розпізнаний за допомогою сервісу Google Speech-to-Text API;

Cp – правильно вимовлений текст студентом відповідно поставленого ігрового завдання;

Wp – неправильно вимовлений текст студентом відповідно поставленого ігрового завдання.

Отже, необхідно відобразити залежність між множинами 2.6, щоб враховувати чи вимовлений текст є правильним. Залежність представлена формулою:

$$Rt \cap Sw = Cp. \quad (2.9)$$

Отже, удосконалено метод гейміфікації процесу вивчення правильної вимови слів, який на відміну від існуючих, враховує помилки студентів на основі пройденого навчального матеріалу, що дозволить покращити результати навчання та створити захопливе середовище для студентів сприяючи їхньому вдосконаленню у сфері вивчення мови.

2.4 Розробка блок-схеми алгоритму гейміфікації процесу вивчення правильної вимови слів

На основі розробленого методу гейміфікації процесу вивчення правильної вимови слів розроблено блок-схему для гри, за допомогою якої студенти будуть навчатись. Оскільки метод використовується для підвищення залучення студентів та покращення ефективності вивчення, було створено блок-схему, яка представлена на рисунку 2.2.

Крок 1. Початок.

Крок 2. Отримано набір неправильно вимовлених слів студентом під час навчання на основі наявних навчальних матеріалів додатка. Слова зберігаються в колекції для даних listWords.

Крок 3. Сесія гри яка буде продовжуватись, поки значення кількості життів ігрового персонажа (countLives) більше 0.

Крок 4. Відображення слова з колекції listWords для того, щоб студент міг вимовити під час гри. Слово функцією випадкового вибору.

Крок 5. Отримання та обробка аудіо файлу з вимовою від студента.

Крок 6. Визначення вимовленого тексту студентом з аудіо файлу та збереження в змінну text слово, яке необхідно було вимовити в грі.

Крок 7. Перевірка чи вимовлений текст відповідає умові.

Крок 8. За умови, що текст було неправильно вимовлено, в ігрового персонажа зменшується кількість життів (countLives).

Крок 9. Зміна стану ворога, а саме зміна анімації зі спокійного стану в атакуючий стан.

Крок 10. За умови, що текст було правильно вимовлено, рахунок гравця (score) збільшується.

Крок 11. Змінюється стан ігрового персонажа, а саме зміна анімації зі спокійного стану в атакуючий.

Крок 12. Після завершення усіх життів в межах сесії змінюється стан персонажа, який відображає завершення гри.

Крок 13. Відображується вигулькуюче вікно, що означає завершення гри

та містить в собі інформацію про загальний рахунок.

Крок 14. Кінець.

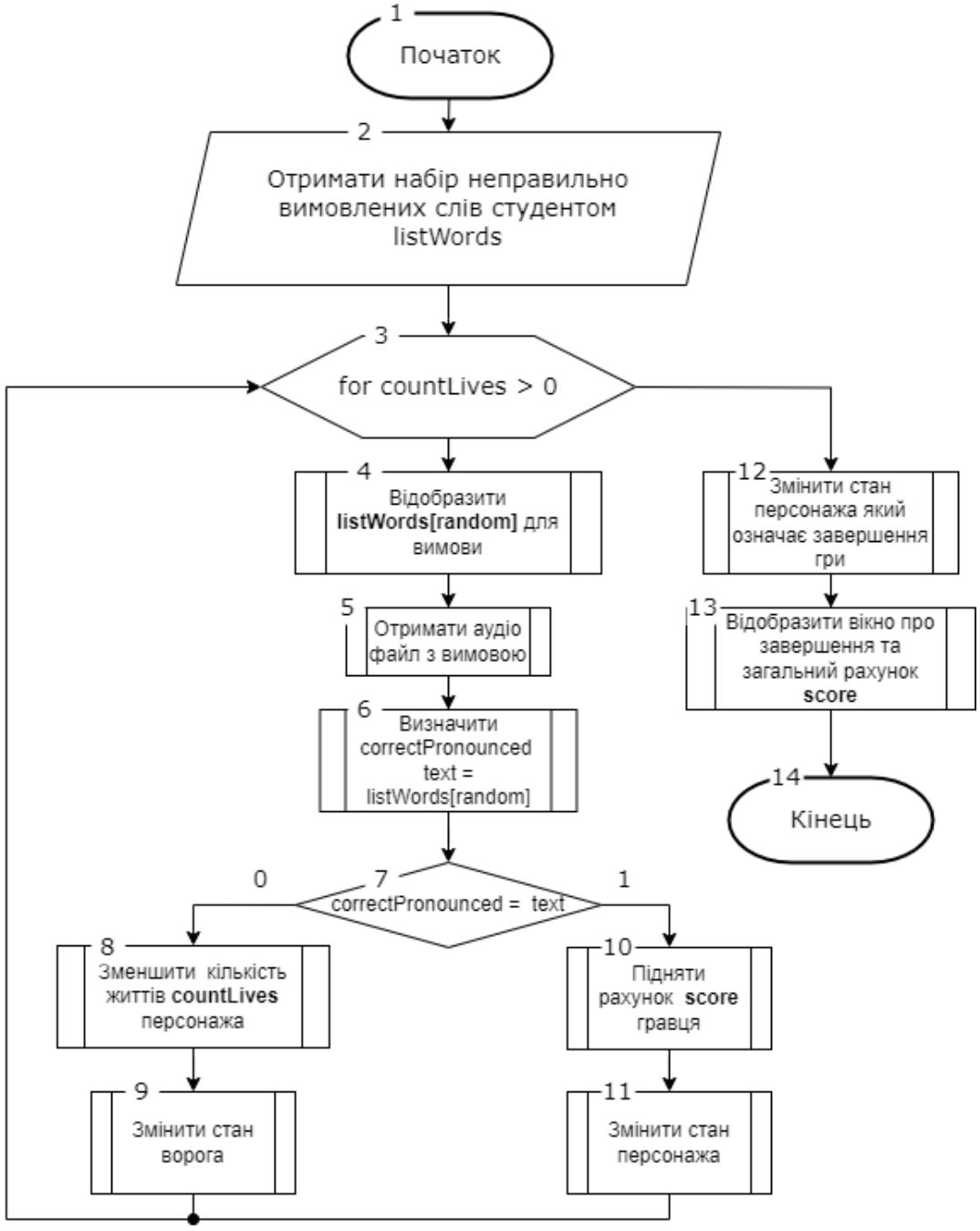


Рисунок 2.2 – Блок-схема алгоритму гри вивчення правильної вимови слів

Отже, створено блок-схему алгоритму гейміфікації процесу вивчення правильної вимови слів. Використовуючи потужність ігровізації, є можливість створити середовище, в якому студенти будуть мотивовано вдосконалювати свої навички вимови, що зрештою покращить рівень їхнього володіння мовою.

2.5 Розробка структури графічного інтерфейсу

Дизайн інтерфейсу користувача (UI) – це процес, який дизайнери використовують для створення інтерфейсів у програмному забезпеченні чи комп’ютеризованих пристроях, зосереджуючись на зовнішньому вигляді чи стилі [24]. Головна ціль дизайнера створювати інтерфейси, які користувачі вважають простими у використанні та приємними.

Графічний інтерфейс користувача (GUI) – це цифровий інтерфейс, у якому користувач взаємодіє з графічними компонентами [25]. У GUI візуальні елементи, що відображаються в інтерфейсі користувача, передають інформацію важливу для користувача, а також дії, які він може виконати.

Розробка структури графічного інтерфейсу передбачає проєктування та створення візуальних елементів та інтерактивних компонентів, які дозволяють користувачам взаємодіяти з програмним додатком або системою. Графічні інтерфейси є невід’ємною частиною сучасного програмного забезпечення, що полегшує користувачам взаємодію зі складними системами, надаючи візуальне представлення функцій і даних програми.

Під час взаємодії з вікном представленим на рисунку 2.3 студент вивчатиме навчальний матеріал, після чого зможе отримати персоналізовану підказку у вигляді вигулькуючого вікна над усім матеріалом.

Сторінка з помічником повинна містити в собі наступні елементи:

1. Логотип додатка;
2. Навігаційна панель;
3. Форма у вигляді картки;
4. Назва звуку, який користувач вивчає під час користування сторінкою;
5. Аудіо файл з правильною вимовою слів;
6. Текст, який користувач повинен вимовити для перевірки сервісом;
7. Зображення або гіф анімація з правильною артикуляційною постановкою для того, щоб вимовити звук, який вивчається;
8. Кнопка для запису аудіо файлу;
9. Клавіша для відправлення аудіо на розпізнавання вимови;

10. Частина форми, в якій зображується правильно або неправильно й повністю вимовлений текст;

11. Плеєри, в яких користувач може відтворити записані аудіо та прослухати власну вимову з можливістю завантажити на персональний комп’ютер;

12. Нижня частина сторінки, яка містить інформацію про додаток та корисні посилання;

13. Вікно з підказкою для студента.

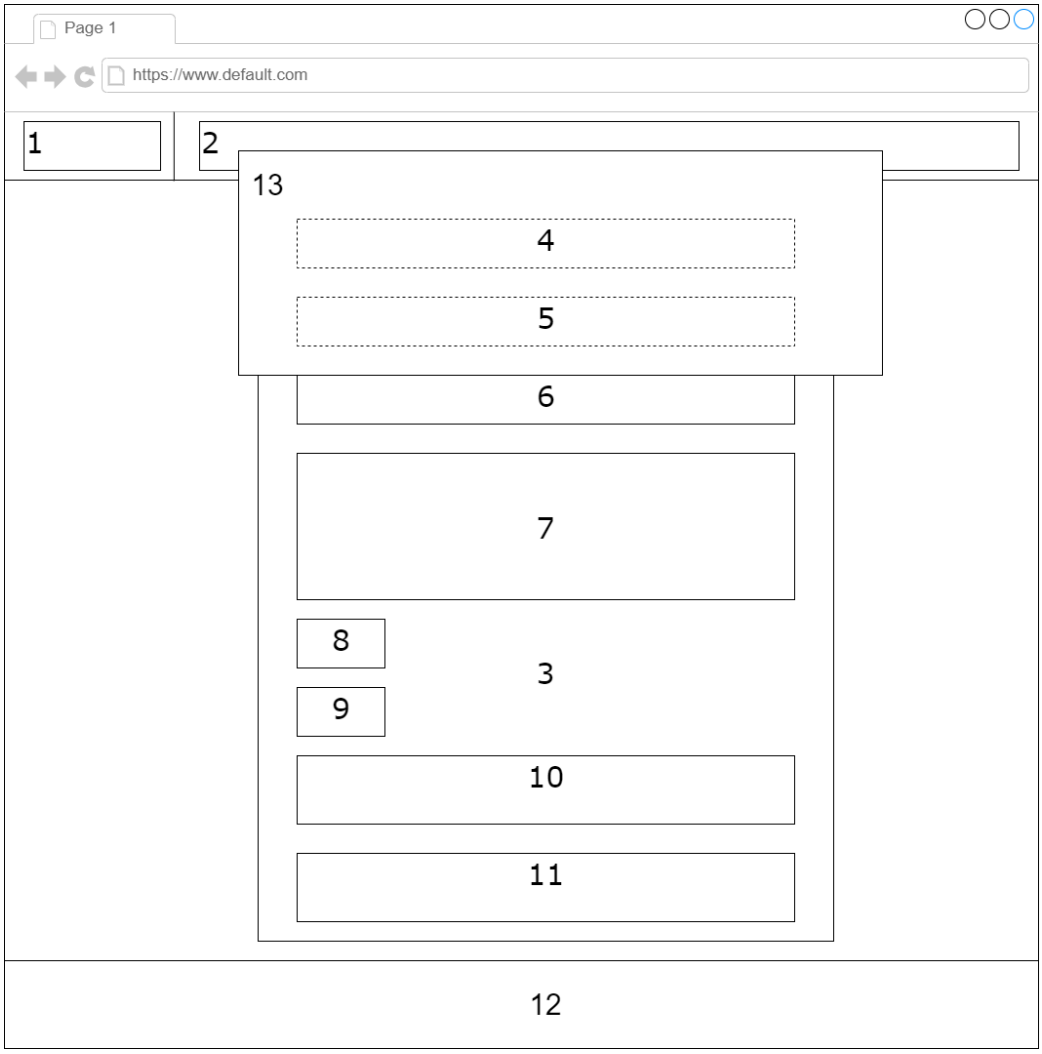


Рисунок 2.3 – Вікно вивчення з підказкою

Для ігровізації процесу вивчення було створено структуру вікна, в якому студент зможе вивчати правильну вимову в форматі гри (рис. 2.4). Це дозволить підвищити його залученість під час навчання.

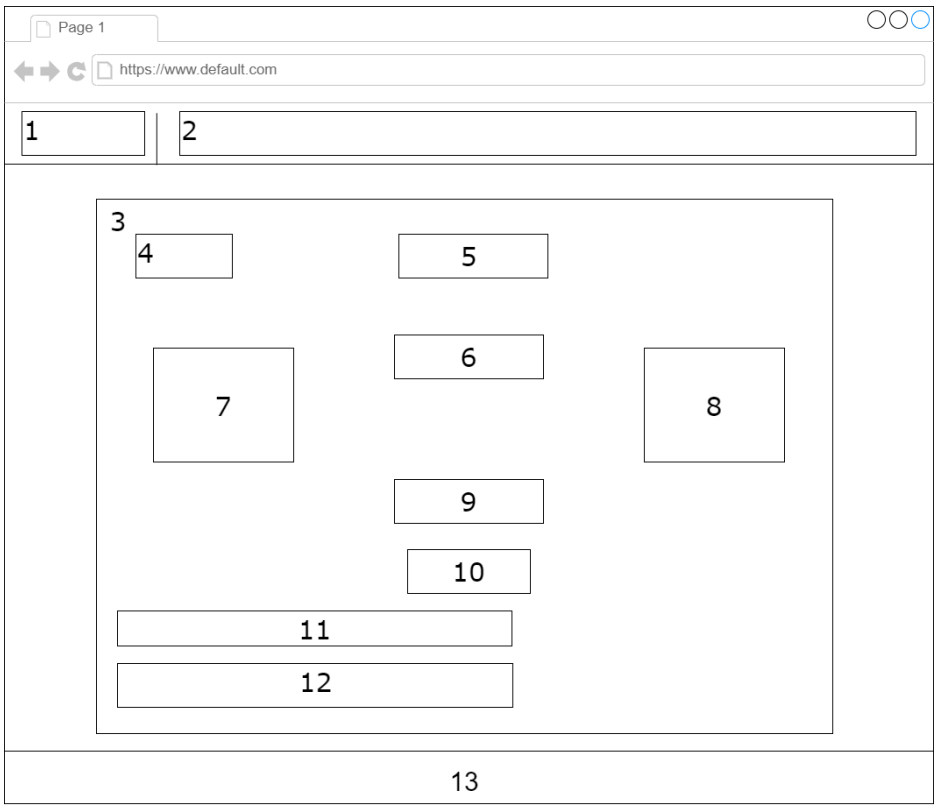


Рисунок 2.4 – Вікно для вивчення слів в форматі гри

Сторінка з грою повинна містити в собі наступні елементи:

1. Логотип додатка;

2. Навігаційна панель;

3. Робоча область взаємодії студента з навчальним матеріалом;

4. Кількість життів ігрового персонажа;

5. Кількість отриманих балів під час вивчення;

6. Слово, яке потрібно вимовити;

7. Персонаж студента, який за умови успішно вимовленого слова, виконує анімацію атаки;

8. Персонаж, який при неправильній вимові слова виконує анімацію атаки, та зменшує кількість життя студента;

9. Кнопка розпочати аудіо запис вимови;

10. Клавіша для перевірки вимови з аудіозапису;

11. Плеєри, в яких користувач може відтворити записані аудіо та прослухати власну вимову з можливістю завантажити на персональний комп’ютер;

12. Частина форми, в якій зображується правильно або неправильно й повністю вимовлений текст;

13. Нижня частина сторінки, яка містить інформацію про додаток та корисні посилання;

Отже, було створено структури графічного інтерфейсу для головних особливостей додатка, а саме процесу вивчення правильної вимови слів у форматі гри та відображення підказки студента під час його навчання користуючись наявними матеріалами. Це дозволить створити зручний та ефективний інтерфейс для програмного рішення.

2.6 Висновки

Отже, в межах другого розділу розроблено метод динамічної генерації персоналізованої підказки для допомоги виправлення неправильно вимовлених слів. Для його реалізації обрано два сервіси для інтеграції та створено модель для маніпулювання даними. Створений метод дозволить ефективніше навчатись студентам, оскільки його підказки є миттєвими та персоналізованими. Також подано вичерпний огляд розробки методу гейміфікації, спрямованого на покращення вивчення правильної вимови слів, які обираються на основі пройденого матеріалу. Крім того, визначено важливість інтеграції елементів гейміфікації в процес вивчення мови, таких як: ігрова механіка, механізми зворотного зв’язку та відстеження прогресу. Оскільки блок-схеми є незамінними інструментами для спрощення, візуалізації та передачі даних про системи та процеси, було створено дві блок-схеми алгоритмів реалізації запропонованих методів.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ВИВЧЕННЯ ПРАВИЛЬНОЇ ВИМОВИ СЛІВ

3.1 Проєктування програмного забезпечення для вивчення правильної вимови слів

Архітектурний дизайн програмної системи має вирішальне значення для її загального успіху. Він надає високорівневе уявлення про систему та її компоненти, що дозволяє розробникам виявляти потенційні проблеми проєктування на ранніх стадіях процесу розробки.

Model-View-Controller (MVC) – це широко використовуваний патерн проєктування програмного забезпечення, який розділяє додаток на три взаємопов’язані компоненти: модель, вигляд і контролер [26]. Модель представляє дані та бізнес-логіку додатка, представлення – користувацький інтерфейс, а контролер керує потоком даних між компонентами моделі та представлення. Цей патерн корисний при розробці великомасштабних додатків, оскільки він дозволяє модулювати та розділяти проблеми, що, зі свого боку, покращує ремонтопридатність і масштабованість. Приклад використання шаблону проєктування представлений на рисунку 3.1.

Рисунок 3.1 – Використання патерну MVC

Діаграма компонентів використовується для того, щоб показати взаємозв’язок між компонентами в системі й те, як вони взаємодіють один з

одним. Це тип діаграми UML, який зображує фізичні компоненти системи, такі як от програмні модулі, бібліотеки, виконувані файли й таблиці баз даних, а також їхні зв’язки та залежності.

Основна мета діаграми компонентів – допомогти зрозуміти архітектуру системи та ідентифікувати компоненти, з яких вона складається. Вона також допомагає визначити інтерфейси між компонентами, що корисно для проєктування і тестування кожного компонента окремо [27]. Крім того, діаграми компонентів можна використовувати для полегшення комунікації між командами розробників, зацікавленими сторонами та менеджерами проєктів, надаючи чітке і стисле уявлення про архітектуру системи.

Тому для того, щоб покращити розуміння архітектури системи та ідентифікувати компоненти було розроблено діаграму компонентів, яка представлена на рисунку 3.2.

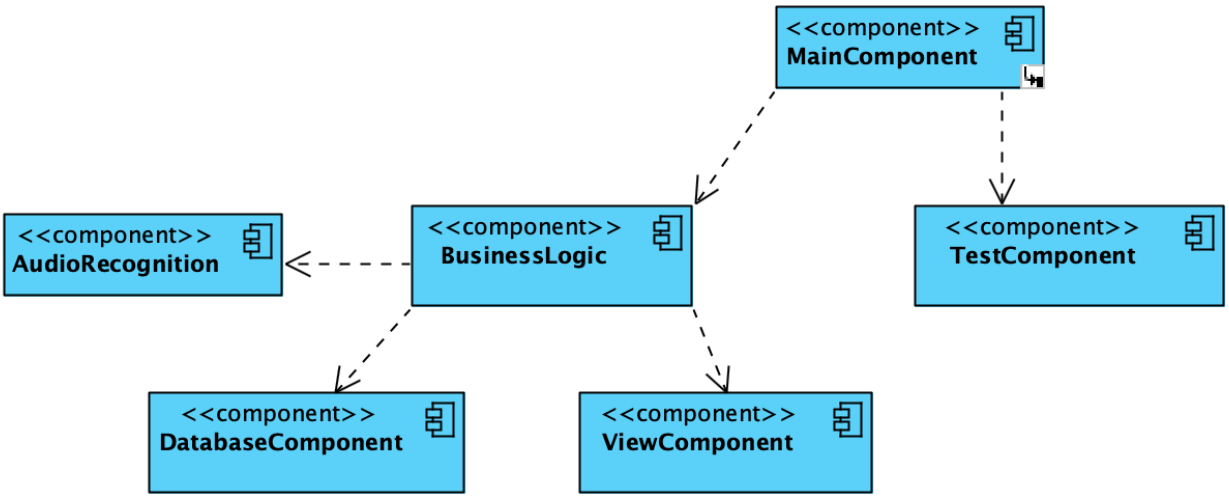


Рисунок 3.2 – Діаграма компонентів

У програмній системі на основі шаблону проєктування Model-View-Controller компонент моделі відповідає за управління даними та бізнес-логікою додатка. Це містить в собі перевірку даних, їх збереження та пошук. Модель взаємодіє з базою даних та іншими зовнішніми системами для отримання та зберігання даних. Компонент представлення, з іншого боку, відповідає за представлення даних користувачеві. Він обробляє дані, введені користувачем,

такі як от клацання мишею та введення з клавіатури, і відповідно оновлює інтерфейс користувача. Нарешті, компонент контролера діє як посередник між моделлю і представленням, керуючи потоком даних між ними. Він отримує користувацьке введення від компонента представлення, відповідно оновлює модель, після чого надсилає оновлені дані назад до компонента представлення для відображення.

Основна перевага використання патерну MVC полягає в тому, що він забезпечує модульність і поділ завдань. Розділяючи додаток на окремі компоненти, розробники можуть працювати над кожним компонентом незалежно, не впливаючи на інші. Це робить кодову базу більш зручною для підтримки та легшою для розуміння. Крім того, шаблон проєктування MVC забезпечує чітке відокремлення коду інтерфейсу користувача від бізнес-логіки, що полегшує оновлення або зміну будь-якого з компонентів, не впливаючи на інший.

При проєктуванні системи на основі шаблону MVC важливо ретельно продумати взаємодію між компонентами. Контролер повинен відповідати за управління потоком даних між представленням і моделлю й не повинен містити жодної бізнес-логіки або коду перевірки даних [28]. Модель повинна бути спроєктована так, щоб її можна було легко протестувати, з чіткими й лаконічними методами доступу до даних. Компонент представлення повинен бути спроєктований якомога легшим, з мінімальною бізнес-логікою або обробкою даних.

Загалом архітектура на основі MVC може бути ефективним способом проєктування великомасштабних програмних систем, які легко підтримувати та масштабувати. Розділяючи додаток на окремі компоненти, розробники можуть працювати над кожним компонентом незалежно, роблячи кодову базу більш модульною і легшою для розуміння. Однак важливо ретельно продумати взаємодію між компонентами, щоб система була ефективною та результативною. Тому спираючись на ефективність та переваги використання патерну MVC для програмного рішення застосовано шаблон проєктування Model-View-Controller.

Для розробки програмного забезпечення необхідно застосувати надійний набір інструментів, який є популярним серед розробників, а саме: шаблони проєктування GOF. Розуміючи та використовуючи ці шаблони, розробник зможе розробити програмні модулі ефективно, модульно та з урахуванням майбутніх модифікацій. Патерни Gang of Four (GoF) – це набір шаблонів проєктування, представлений Еріхом Гаммою, Річардом Хелмом, Ральфом Джонсоном і Джоном Влісайдсом. Ці шаблони забезпечують перевірені рішення повторюваних проблем у розробці програмного забезпечення, що робить їх незамінними для багатьох розробників програмного забезпечення. При розробці дотримання шаблонів проєктування GoF надає наступні переваги:

1. Зручність читання та обслуговування, оскільки ці шаблони широко відомі, розробники, знайомі з ними, які можуть легко зрозуміти структуру та призначення модуля, який відповідає шаблону GoF.

2. Повторне використання: модулі, розроблені за цими шаблонами, часто можна повторно використовувати в різних контекстах з мінімальними змінами.

3. Масштабованість і гнучкість: розробка з дотриманням шаблонів відокремлюють функції, полегшуючи масштабування або зміну окремих компонентів, не впливаючи на інші.

Для проєктування додатка використовується Visual Paradigm, який підтримує моделювання UML. Уніфікована мова моделювання (UML) – це візуальна мова, що використовується для проєктування та моделювання програмних систем [29]. Вона надає набір стандартизованих діаграм і нотацій, які допомагають розробникам спілкуватися і документувати різні аспекти програмної системи. Діаграми UML можуть представляти різні погляди на систему, включаючи структурні, поведінкові та архітектурні погляди. UML є широко використовуваною мовою у розробці програмного забезпечення і підтримується багатьма інструментами та фреймворками. Її основна перевага полягає в тому, що вона забезпечує спільну мову для розробників і зацікавлених сторін, полегшуючи комунікацію і розуміння програмних систем між різними командами й доменами.

Також необхідно розробити діаграму послідовності для актора студент, який буде взаємодіяти з системою, де відображено увесь порядок його дій. Діаграма представлена на рисунку 3.3.

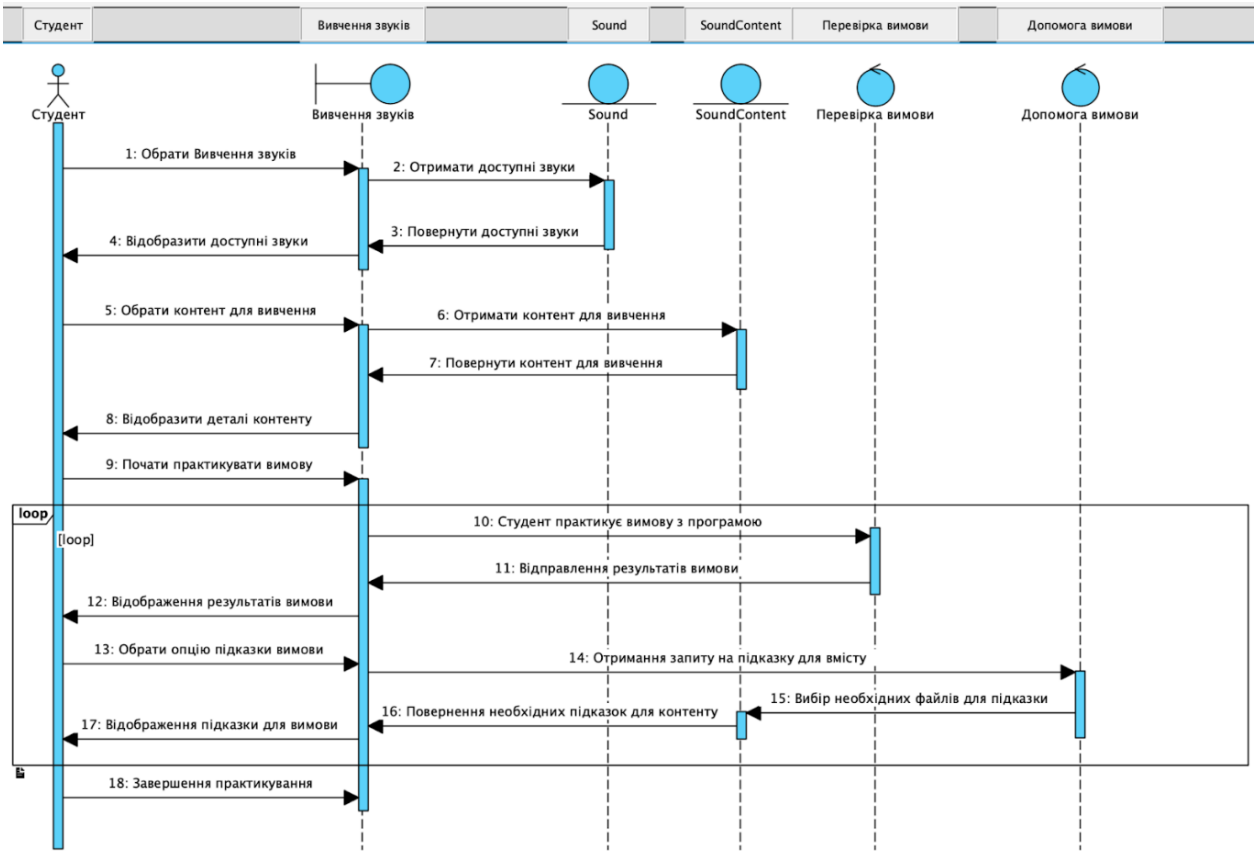


Рисунок 3.3 – Підсумкова діаграма послідовності

Студент: представляє студента-актора, який взаємодіє з системою.

Вивчення звуків: представляє компонент взаємодії користувача з інтерфейсом системи.

Sound: представляє звук окремої мови, який обирає студент.

Sound Content: представляє звуковий вміст, який вивчає студент.

Перевірка вимови: представляє компонент, який відповідальний за перевірку вимови студента.

Допомога вимови: представляє компонент, який відповідальний за опцію допомога користувачу.

Для предметної області виділено наступних акторів, які представлені на таблиці 3.1.

Таблиця 3.1 – Характеристика акторів

Актор	Короткий опис
Студент	Користувач, який користується додатком для покращення власної вимови слів
Вчитель	Користувач, який займається призначенням своїм учням завдання з вимови
Адміністратор	Керує програмним забезпеченням і його користувачами, включаючи студентів та викладачів

Для надання візуального представлення того, як бізнес-процес працює в програмному забезпеченні, створено діаграму діяльності бізнес-процесу. Діаграма діяльності для бізнес-процесу додатка представлена на рисунку 3.4.

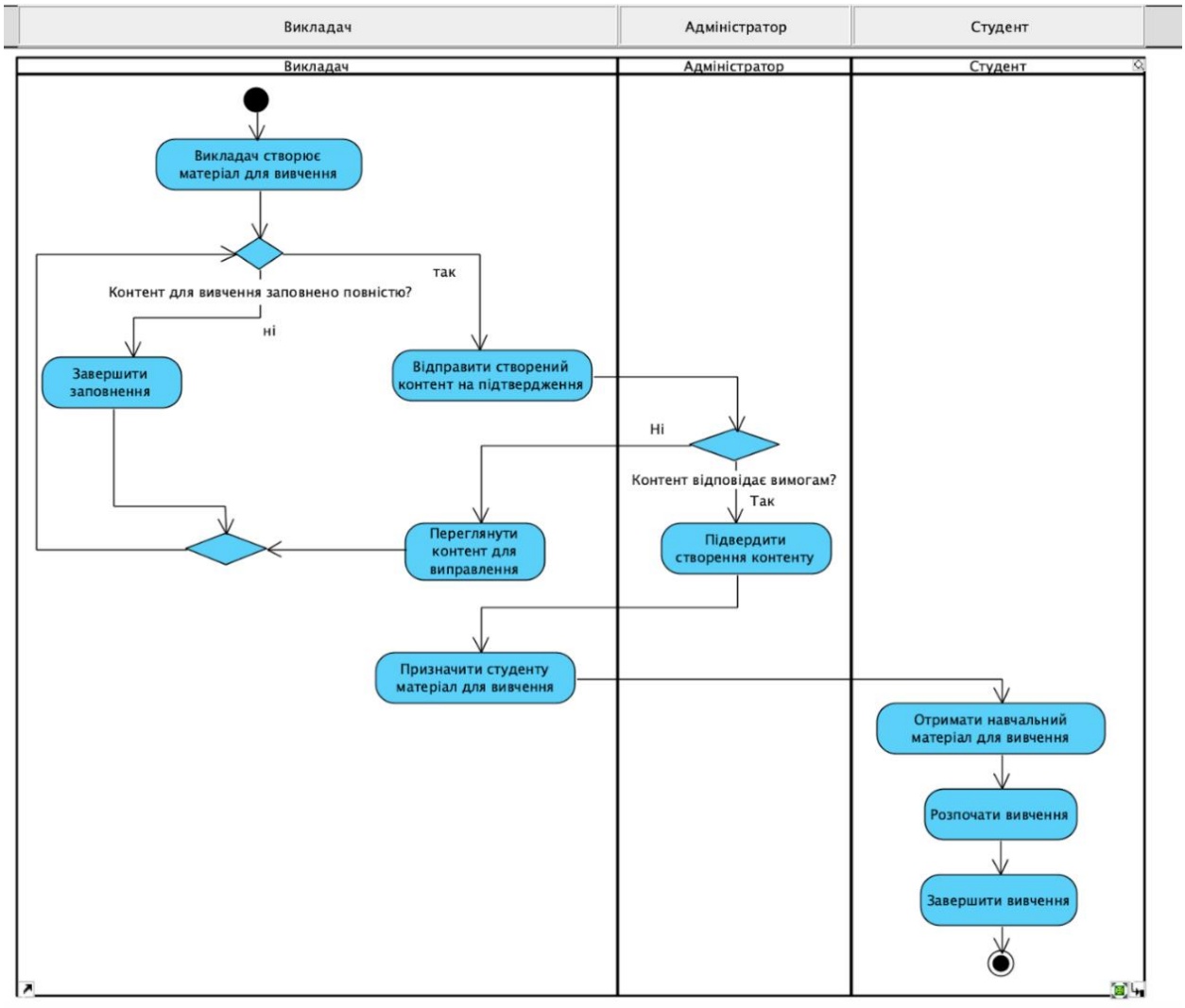


Рисунок 3.4 − Діаграма діяльності бізнес-процесу

Створена діаграма може бути особливо корисною для зацікавлених сторін, розробників і керівників проєктів, щоб зрозуміти головну функціональність програмного забезпечення.

Отже, результатом детального проєктування програмного забезпечення є створені UML діаграми: діаграма діяльності бізнес процесу та підсумкова діаграма послідовності. Також створено таблицю з описом акторів.

3.2 Варіантний аналіз і обґрунтування вибору мови програмування

Java – це об’єктно-орієнтована мова програмування на основі класів, розроблена з урахуванням принципу «напиши один раз, запусти будь-де». Програми Java, скомпільовані в байт-код, можуть працювати на будь-якій віртуальній машині Java (JVM), незалежно від архітектури комп’ютера [30]. Це робить Java дуже портативною. Завдяки величезній стандартній бібліотеці та величезній екосистемі Java стала домінуючою силою в корпоративних середовищах, особливо в розробці програм на стороні сервера. Java забезпечує високу продуктивність, надійність і безпеку. Фреймворки, такі як Spring Boot, також дозволяють розробникам ефективно створювати масштабовані веб-додатки.

C# – це сучасна об'єктно-орієнтована мова, розроблена Microsoft. C# був випущений у 2000 році та надає функціональність, подібну до Java, але тісніше інтегрований з платформою Windows. Він має конструкції, які підтримують компонентно-орієнтоване, декларативне, функціональне та паралельне програмування. Завдяки .NET Core C# став кросплатформним, дозволяючи розробку на macOS, Linux і Windows [31].

Ruby – динамічна, рефлексивна, об'єктно-орієнтована мова програмування. Ruby надає перевагу продуктивності розробника. Його елегантний синтаксис робить код читабельним і часто зменшує загальний обсяг коду. Ruby on Rails – це фреймворк веб-додатків, який популяризував Ruby, просуваючи конвенції над конфігурацією та принцип «Не повторюй себе» (DRY) [32]. Завдяки активній спільноті та великій кількості якісних бібліотек Ruby

забезпечує швидку розробку веб-додатків.

В таблиці 3.2 наведено результати порівняння розглянутих мов програмування.

Таблиця 3.2 – Порівняння розглянутих мов програмування

Критерій	Ruby	C#	Java
Фреймворки безпеки	0	1	1
Плагіни від спільноти	1	0.5	1
Платформо незалежність	0.5	0.5	1
Веб масштабованість	0.5	0.5	1
Загальний результат	2	2.5	4

У таблиці 1 означає повну підтримку або відповідність критерію, 0,5 вказує на часткову підтримку або відповідність, а 0 означає відсутність підтримки.

Масштабованість у веб-додатках означає здатність системи обробляти зростаючий обсяг роботи, трафіку або запитів шляхом додавання ресурсів до системи. Java, особливо з фреймворком Spring Boot, була головним вибором розробників для створення масштабованих веб-додатків корпоративного рівня. Продуктивність Java виграє внаслідок компіляції Just-In-Time (JIT), що робить її більш придатною для потреб високої продуктивності та дозволяє обробляти більшу кількість одночасних запитів із меншими витратами.

ASP.NET Core є потужним фреймворком і може використовуватися для створення масштабованих програм. Однак історично склалося, що .NET був орієнтований на Windows, що іноді призводило до проблем щодо масштабованості пов’язаних зі стратегіями масштабування Windows Server. Ruby on Rails відомий швидким розвитком. Хоча його можна зробити масштабованим, він не був розроблений з урахуванням високої масштабованості. Twitter, наприклад, спочатку почав роботу з Ruby on Rails, але

перейшов до складнішої системи через потреби в масштабованості.

Для реалізації розпізнавання тексту варто застосувати мову Python, оскільки його перевагою є безліч безплатних бібліотек для обробки звуку та машинного навчання, що робить завдання розпізнавання звуку простішим і ефективнішим. Простий і зрозумілий синтаксис Python дозволяє швидше розробляти та створювати прототипи, дозволяючи розробникам швидко тестувати ідеї. Відокремивши спеціалізовану функцію, розпізнавання аудіо в Python, від основної логіки програми в Java, це дозволить легше підтримувати, оновлювати або масштабувати кожен компонент створеного додатка незалежно.

Відповідно до результатів порівняння наведених в таблиці 3.2 можна зробити висновок, що мова програмування Java є хорошим рішенням для розробки додатка для вивчення правильної вимови слів, оскільки фреймворк Spring надає велику перевагу в створенні веб-додатків та значно пришвидшує їх розробку.

3.3 Вибір середовища розробки та СУБД

На початку існування мови Java розробники створювали програми, використовуючи лише блокнот і командний рядок DOS [33]. Завдяки появі IDE (Integrated Development Environment) ці часи минули назавжди. Завдяки використанню IDE різко підвищилась швидкість і ефективність розробки. Розробникам Java пощастило мати широкий вибір потужних IDE.

Суть програмування мовою Java полягає не тільки в написанні ефективного коду, але й в інструментах, які полегшують його розробку. Серед цих інструментів ключову роль відіграють інтегровані середовища розробки. Середовища розробки інтегрують кілька інструментів, які допомагають розробникам у написанні, налагодженні, тестуванні, рефакторингу та оптимізації коду, тим самим підвищуючи продуктивність і якість коду. Тому необхідно провести аналіз та обрати середовище розробки, яке найбільше підходить для поставлених задач, з’ясувавши їхні особливості, функції та значення в екосистемі Java.

IntelliJ IDEA – це багатофункціональна IDE, розроблена JetBrains [34]. Він пропонує як безплатне видання Community, так і платне видання Ultimate. IntelliJ IDEA підтримує широкий спектр мов, окрім Java, включаючи Kotlin, Groovy та Scala. Він також забезпечує глибоку інтеграцію з популярними фреймворками та технологіями. Головними особливостями є: інтелектуальне завершення коду, аналіз коду на етапі його написання, розширені інструменти рефакторингу, інструменти бази даних, вбудований термінал, інтеграція контролю версій Git і багато плагінів. Середовище розробки представлене на рисунку 3.5.

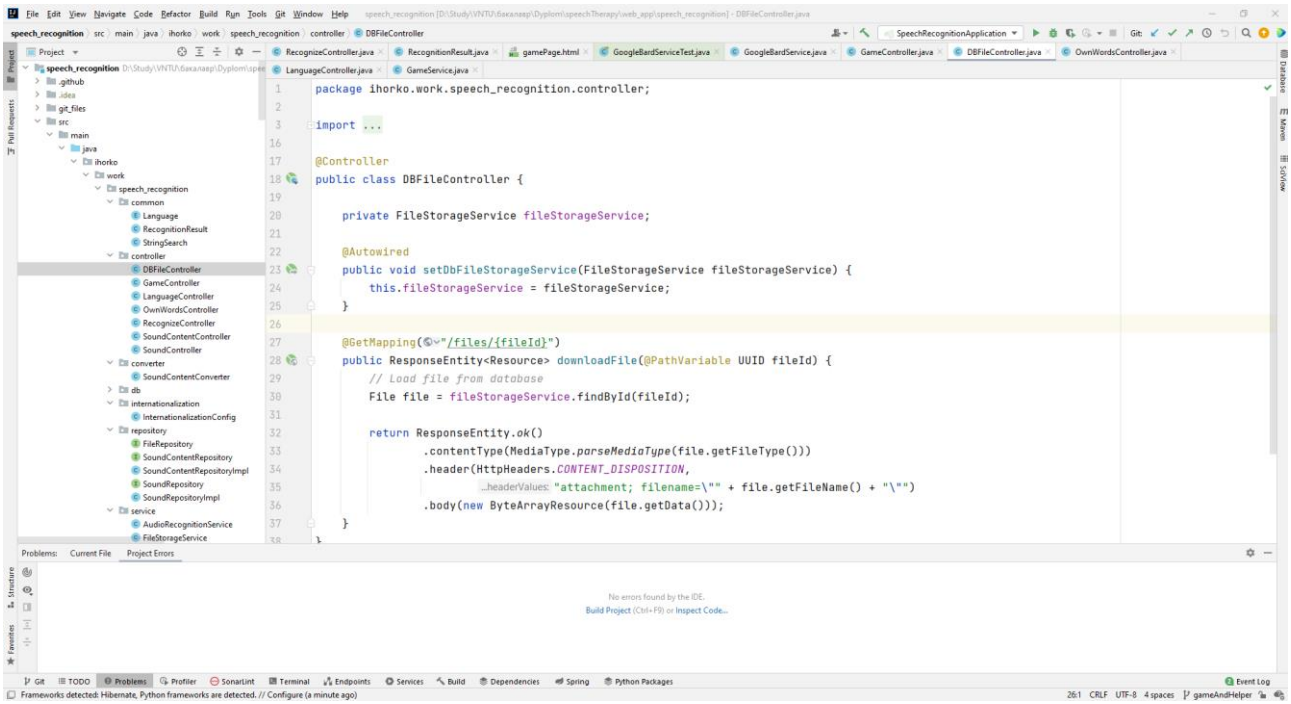


Рисунок 3.5 – Середовище розробки Intelij IDEA

Eclipse – це IDE з відкритим вихідним кодом, яка існує протягом тривалого часу та має велику базу користувачів [35]. Оскільки Eclipse підтримує багато мови, розробник може легко об’єднати підтримку кількох мов та інші функції в з пакетів за замовчуванням, а Eclipse Marketplace дозволяє практично необмежене налаштування та розширення (рис. 3.6). Головними особливостями є: доповнення коду, багата документація API, інтеграція Git, величезний ринок плагінів і PDE (середовище розробки плагінів) для розширення його можливостей.

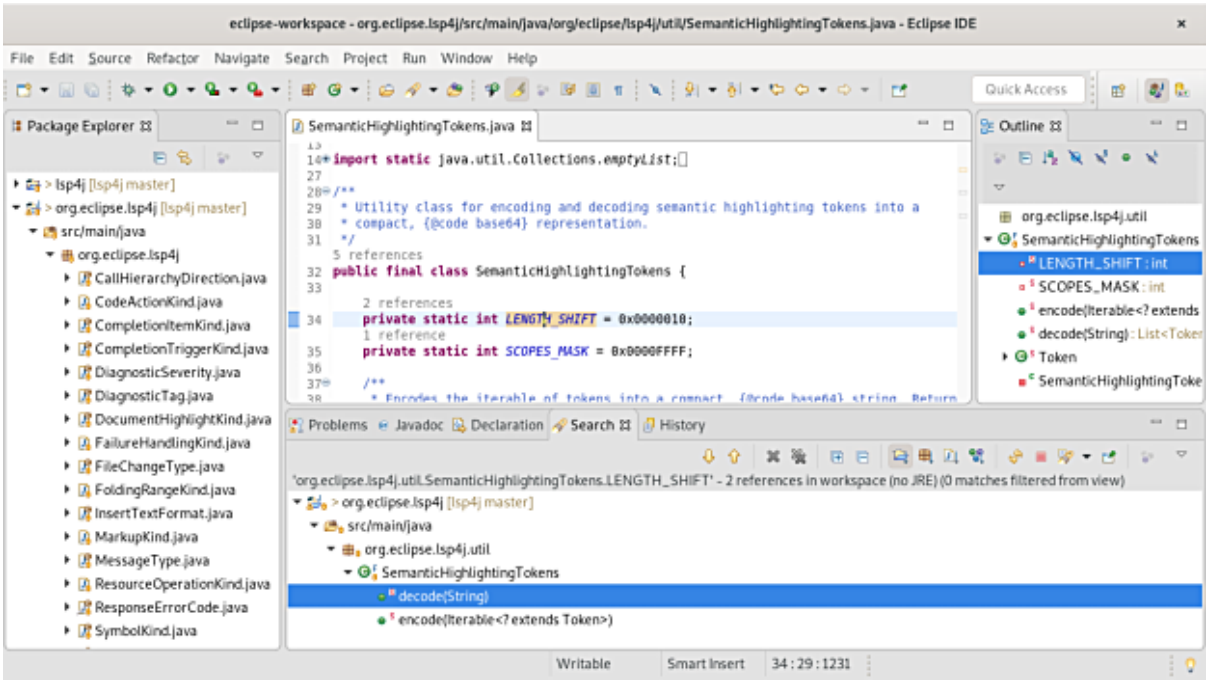


Рисунок 3.6 – Середовище розробки Eclipse

NetBeans – спочатку розроблений Sun Microsystems, а пізніше придбаний Oracle, є одним із популярних середовищ розробки із відкритим кодом IDE для розробників Java (рис. 3.7). Він підтримує кілька мов, але широко відомий своїми можливостями Java [36]. Головними особливостями є: шаблони коду, підказки щодо програмування, проста навігація, вбудована підтримка Maven, профайлер та інтегрований контроль версій.

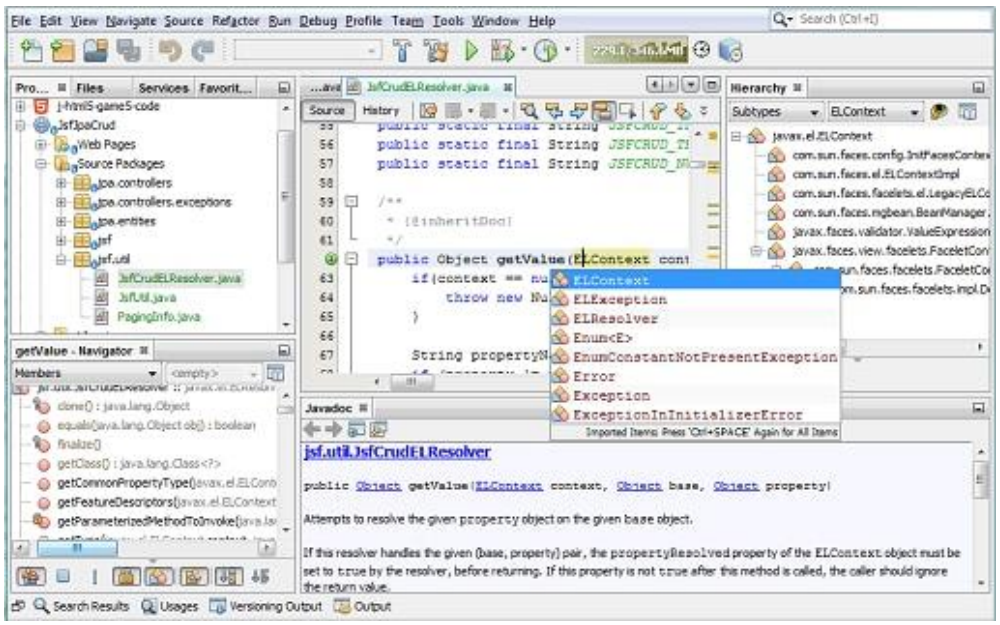


Рисунок 3.7 – Середовище розробки NetBeans

JDeveloper – це безплатна IDE, розроблена компанією Oracle яка зосереджена на наданні функцій для набору технологій розробки Oracle [37], хоча вона також підтримує стандартну розробку Java (рис. 3.8). Головними особливостями є: візуальна розробка, інтегроване налагодження, профілювання та підтримка Oracle ADF (Application Development Framework).

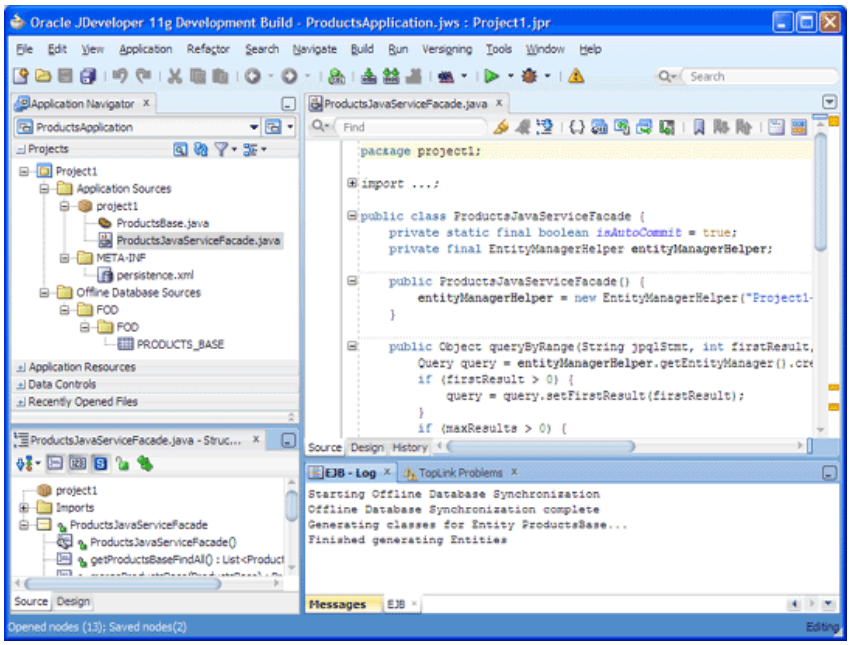


Рисунок 3.8 – Середовище розробки JDeveloper

Порівняльна характеристика середовищ розробки для мови Java представлена на таблиці 3.3.

Таблиця 3.3 – Порівняння розглянутих мов програмування

	Intelij IDEA	Eclipse	NetBeans	JDeveloper
Простота інтерфейсу	+	+-	+-	+-
Продуктивність	+	-	+-	+-
Розширення (Плагіни)	+	+	+	+-
Вартість (Безоплатність)	+-	+	+	+
Вбудований термінал	+	-	-	-
Інтеграція з фреймворками	+	+-	+-	+-
Результат	5.5	3	3.5	3

Значення в таблиці 3.3: + підтримується/позитивний аспект, - не підтримується/Негативний аспект, +- частково підтримується/змішаний аспект.

PostgreSQL – це потужна система управління реляційними базами даних (RDBMS) із відкритим кодом. PostgreSQL розроблявся понад 30 років і заслужив репутацію надійності, цілісності даних і можливості розширення [38]. Підтримує більшість стандартів SQL і пропонує багато розширених функцій, таких як от успадкування таблиць, зовнішні ключі, тригери, збережені процедури та багато іншого. Однією з головних особливостей PostgreSQL є її розширюваність. Розробник може визначати власні типи даних, створювати власні функції та навіть писати код різними мовами без потреби змін у зовнішньому програмному забезпеченні.

Найпопулярнішим інструментом адміністрування та керуванням для PostgreSQL є PGAdmin [39]. Головними причинами, вибору розробниками PGAdmin є:

1. Доступність між платформами:

Незалежно від операційної системи розробника, система PGAdmin готова відмінно працювати. Оскільки система створена на Python і використовує веб-платформу Flask, це робить її адаптованою до різних платформ. Незалежно від того, чи працює розробник у Windows, macOS чи на різних дистрибутивах Linux, PGAdmin пропонує стабільне рішення, забезпечуючи ефективну співпрацю команд.

2. Комплексний графічний інтерфейс:

PGAdmin надає повноцінний графічний інтерфейс користувача (GUI), який спрощує виконання багатьох завдань, пов’язаних із керуванням базою даних PostgreSQL (рис. 3.9). Для розробників, які віддають перевагу візуальному представленню своїх даних і об’єктів бази даних, а не інтерфейсу командного рядка, PGAdmin є хорошим рішенням. Графічний інтерфейс має інтуїтивно зрозумілу структуру, що дозволяє користувачам легко переміщатися по базах даних, схемах, таблицях, представленнях, функціях та інших об’єктах бази даних.

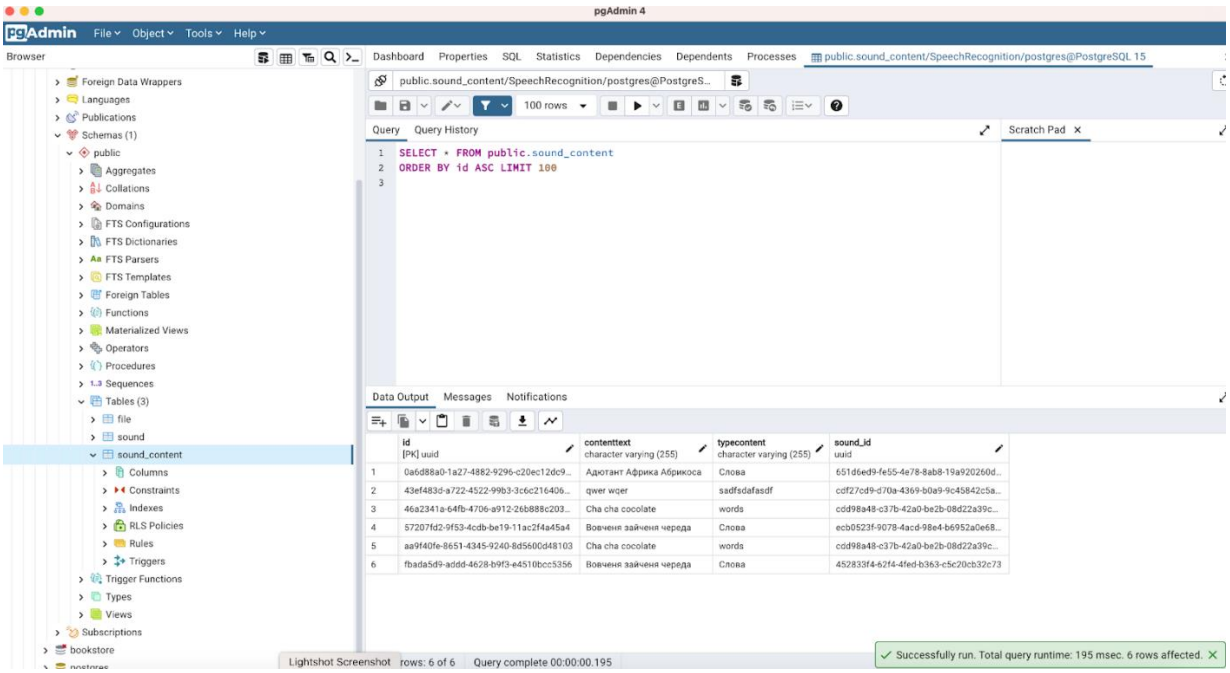


Рисунок 3.9 – Інтерфейс СУБД PGAdmin

3. Розширений редактор SQL:

Вбудований SQL-редактор PGAdmin є відмінним рішенням, оскільки це не просто написання та виконання запитів SQL. Редактор забезпечує підсвічування синтаксису, автозавершення та форматування SQL, що робить кодування ефективнішим і зменшує кількість помилок (рис. 3.10). Крім того, розробники можуть переглядати плани запитів, щоб зрозуміти характеристики продуктивності своїх команд SQL, сприяючи оптимізації запитів.

```
INSERT INTO public.sound_content(
    id, contenttext, typecontent, sound_id)
    VALUES (?, ?, ?, ?);
```

Рисунок 3.10 – Підсвічування командних слів

4. Моніторинг і діагностика:

Підтримка працездатності та продуктивності бази даних має вирішальне значення. PGAdmin постачається з інформаційною панеллю, яка надає швидкий огляд показників продуктивності сервера. Розробники можуть відстежувати активні сеанси, переглядати запущені запити та стежити за використанням

ЦП/пам’яті. У разі низьких місць продуктивності або проблем цей моніторинг у реальному часі допомагає швидко виявити та розв’язати проблеми (рис. 3.11).

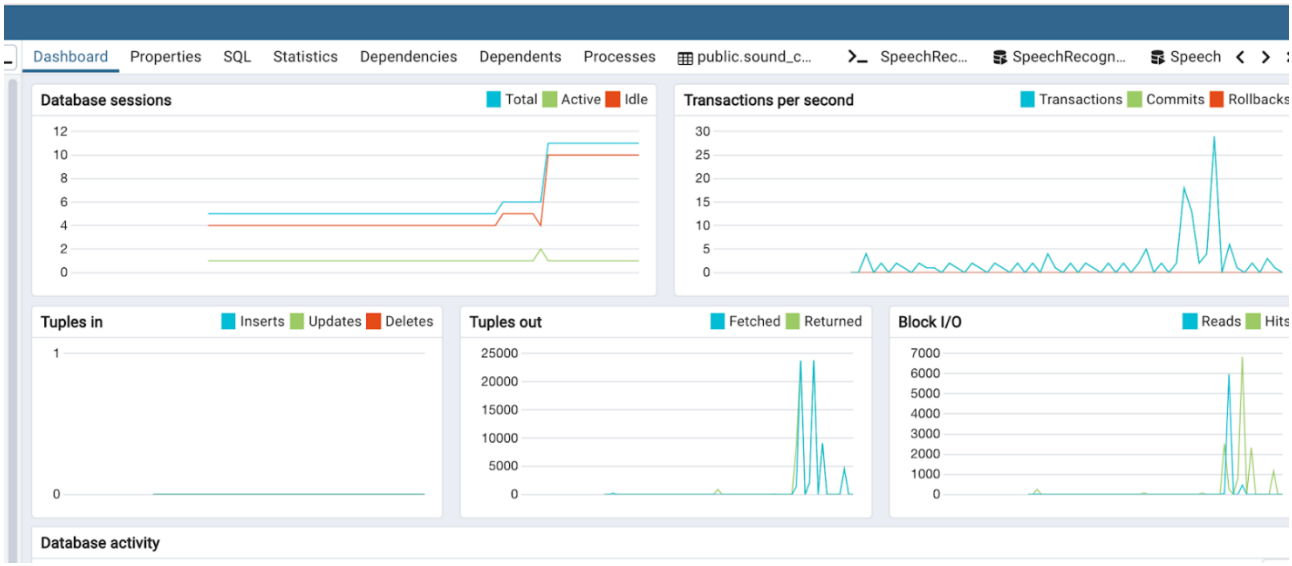


Рисунок 3.11 – Дошка для діагностики роботи бази даних

5. Універсальні функції керування:

Крім запитів PGAdmin містить інструменти, які спрощують керування базами даних. Такі функції, як резервне копіювання та відновлення, імпорту/експорту даних, гарантують, що розробники мають усе необхідне в одному місці. Цей комплексний набір інструментів означає, що розробники витрачають менше часу на зміну інструментів або використання командного рядка для конкретних завдань.

Підсумовуючи, PGAdmin виділяється як комплексний інструмент, який задовольняє не лише основні потреби запитів до бази даних і керування ними, але також заглиблюється в передові сфери моніторингу, діагностики та налаштування. Для розробників, які шукають комплексне рішення для PostgreSQL, PGAdmin є надійним вибором. Його поєднання зручного графічного інтерфейсу, потужних функцій і сильної підтримки спільноти робить його найкращим вибором як для новачків, так і для досвідчених користувачів PostgreSQL. Вибір IDE залежить від потреб розробника. Якщо необхідне потужне комплексне рішення, IntelliJ IDEA може стати найкращим вибором.

Eclipse і NetBeans є надійним вибором, причому перший має більше плагінів, а другий є більш оптимізованим. JDeveloper – це нішевий вибір, який найкраще підходить для розробників, які працюють переважно з технологіями Oracle.

3.4 Розробка графічного інтерфейсу користувача

Для створення адаптивного інтерфейсу для веб-додатка використано Bootstrap, оскільки це є відкритий та безплатний фреймворк. Він є ідеальним для створення адаптивних веб-сайтів. Використовуючи готові CSS класи, нові додані елементи легко адаптуються під інші наявні елементи, автоматично коригуючи їхній розмір.

Bootstrap підтримує доступ до величезної кількості готових рішень для навігації, анімації, форм та інших елементів. Це дозволяє легко підібрати кольорову схему або стиль, які відповідають конкретним потребам проєкту.

Для відображення інструкції по вимові використано алерти від JavaScript, які є однією з базових функцій для інтерактивного спілкування з користувачем. Їхня основна ідея полягає в тому, щоб надсилати користувачеві сповіщення. Приклад сповіщення продемонстровано на рисунку 3.12.

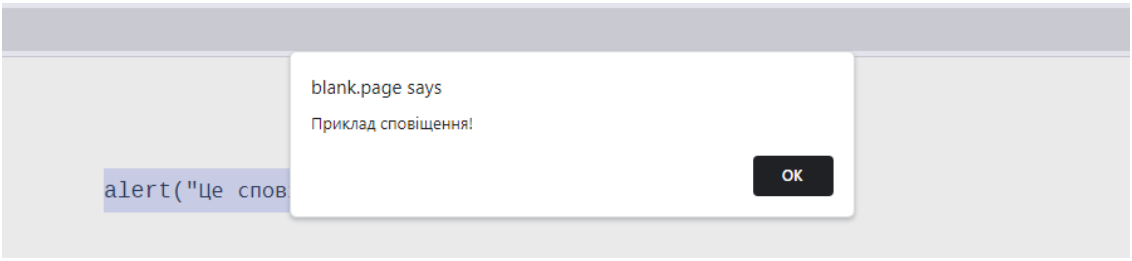


Рисунок 3.12 – Приклад сповіщення для користувача

Однією з переваг алертів є їх простота. Розробник може швидко додати алерт у будь-яке місце коду, щоб надати користувачеві важливу інформацію або допомогти собі в налагодженні коду. Коли алерт з’являється на екрані, він блокує інші дії на сторінці до тих пір, поки користувач не натисне «OK». Це може бути корисним, щоб впевнитись, що студент побачить автоматично згенеровану підказку про вимову слів, з якими він має проблеми перед тим, як продовжити.

Thymeleaf – це сучасна серверна система шаблонів Java, яка дозволяє розробникам створювати динамічно генерований HTML, JavaScript, CSS або навіть звичайний текст [40]. Він використовується у веб-додатках у поєднанні з Spring Framework для відтворення відображення кінцевого інтерфейсу (view). На відміну від деяких інших механізмів шаблонів, шаблони Thymeleaf присутні у файлах HTML, що робить їх доступними для перегляду безпосередньо в браузерах. Ця можливість «природного шаблонування» дозволяє розробникам використовувати той самий шаблон як для прототипування, так і для виробництва, оптимізуючи процес розробки. Синтаксис механізму виразний, надає директиви (так звані атрибути) для вставки динамічних значень, циклічного перегляду колекцій, умовного показу вмісту. Thymeleaf пропонує надійний та інтуїтивно зрозумілий спосіб інтеграції серверної логіки Java із зовнішньою презентацією. Інтеграція тегів Thymeleaf в фрагмент навігаційної панелі представлена на рисунку 3.13.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="en">
<head>
    <meta charset="UTF-8">
    <title>Navigation Bar</title>
</head>
<body>
<nav class=" navbar navbar-expand-lg navbar-light bg-light navbar-default" th:fragment="navbar">
    <a class="navbar-brand px-2" th:href="@{/language}">
        <img class="figure-img img-fluid rounded"
             th:src="@{/image/nav-icon.png}"
             alt="Logo" width="40" height="40"/>
        <span>
            Speech learning
        </span>
    </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item">
                <a class="nav-link" th:href="@{/sounds/list}" th:text="#{sound_list}">
                </a>
```

Рисунок 3.13 – Інтеграція коду Thymeleaf

Однією з головних особливостей Thymeleaf є його здатність використовувати природні шаблони. Це означає, що шаблони є наявними файлами HTML, які можна переглядати в браузерах, а також працювати з ними як зі статичними прототипами. Thymeleaf пропонує модуль інтеграції з Spring

Framework, що робить його природним вибором для багатьох розробників при роботі з Spring MVC. Thymeleaf використовує чіткий і виразний синтаксис, який дозволяє розробникам визначати все, починаючи від текстових значень, атрибутів, циклів, умовних умов тощо, за допомогою власного набору тегів і виразів. Можна визначати власні діалекти, дозволяючи створювати нові атрибути, теги, процесори тощо. Thymeleaf підтримує інтернаціоналізацію (i18n) для багатомовних програм. За замовчуванням Thymeleaf кешує шаблони для підвищення продуктивності. Однак на етапі розробки зазвичай вимикають цю функцію, щоб побачити зміни в реальному часі. Spring Security Integration: Thymeleaf пропонує додатковий модуль для інтеграції з Spring Security, що забезпечує пряму підтримку автентифікації та авторизації в шаблонах. Режими розробки та виробництва: Thymeleaf забезпечує два режими: «розробка» та «виробництво». Режим розробки більш легший, дозволяючи вносити зміни в реальному часі, тоді як режим виробництва оптимізований для продуктивності.

Таким чином, Thymeleaf забезпечує бездоганний міст між програмами Java і динамічними веб-інтерфейсами. Як механізм шаблонів на стороні сервера, він полегшує інтеграцію серверної логіки з зовнішнім представленням, перетворюючи статичний HTML на динамічний вміст. Його «природне шаблонування», спрощує перехід від дизайну до розробки. Дозволяючи розробникам створювати інтерактивні та візуально привабливі інтерфейси користувача, безпосередньо пов’язані з їхньою кодовою базою Java, Thymeleaf стає надійним вибором для розробки веб-додатків.

3.5 Розробка програмних компонент

У сфері розробки програмного забезпечення, що швидко розвивається, модульний дизайн програмного забезпечення став ключовим для забезпечення масштабованості, зручності обслуговування та ефективності розробки системи. Кожен модуль потрібно створити розрізнено, щоб відповідав лише за власні функції, залежності та механізми інтеграції. У такий спосіб кожен компонент є основою для загальної функціональності системи.

В межах розробки необхідно виконати наступні задачі:

1. Реалізувати автогенерацію відповідей за допомогою ШІ Google Bard;

2. Інтегрувати розпізнавання вимови з Google API Speech-to-text в систему правил створення запитів для персоналізованої підказки;

3. Створити скрипт для заповнення навчальним матеріалом середовище;

4. Додати збереження неправильно вимовлених слів під час користування навчальним матеріалом в майбутній список слів для гри;

5. Розробити функції для роботи зі словами, щоб інтегрувати їх ігрове навчальне середовище для вивчення правильної вимови звуків.

Перш за все для використання сторонніх сервісів необхідно додати залежності в створений проєкт. При розробці мовою Java для керування залежностями зазвичай використовують Maven. Maven – це потужний інструмент в екосистемі Java, відомий насамперед своїми можливостями керування проєктами та автоматизації збірки [41]. Maven налаштовується за допомогою pom.xml файлу, який містить інформацію про програмний додаток і деталі конфігурації, які Maven використовує для створення проєкту. Всередині нього міститься опис залежностей, плагіни та інші конфігурації. Однією з переваг є об’єднання бібліотек із вихідним кодом у Maven після вибору залежностей в POM, Maven автоматично завантажує залежності та зв’язує їх під час процесу збирання.

Maven має визначений життєвий цикл (compile, install, test) і коли користувач запускає команду Maven, виконується фаза конкретного життєвого циклу. Для великих багатомодульних проєктів, розробник може мати батьківський POM, який керує загальними конфігураціями для всіх підпроєктів або модулів.

Для розробки сервісу отримання відповідей від ШІ Google Bard додано залежність в POM файл, який знаходиться в кореневій папці. Це дозволить використати класи, що використовуються для зв’язку створеного коду на Java з Google сервісом. Додавання залежності google-bard 0.3.5 в проєкт продемонстровано на рисунку 3.14.

```xml
<dependency>
    <groupId>com.pkslow</groupId>
    <artifactId>google-bard</artifactId>
    <version>0.3.5</version>
</dependency>
```

Рисунок 3.14 – Maven залежність для сервісу Google Bard

На рисунку 3.15 використано створений клас GoogleBardService, який позначено Spring анотацією Service. В середині містить наявний клас GoogleBardClient, що дозволяє за допомогою правильного токена отримати відповідь на будь-який рядок, що є в змінній text. Використовуючи доступний метод ask отримується відповідь у вигляді класу Answer, що забезпечується використаною бібліотекою для роботи з клієнтом. Також, клас Answer містить метод getChosenAnswer за допомогою якого розробник отримує відповідь в форматі рядка з яким легко працювати.

```java
@Service
public class GoogleBardService {

    public String getAnswerFromBard(String text) {
        //__Secure-1PSID;__Secure-1PSIDTS
        GoogleBardClient client = new GoogleBardClient( token: "TOKEN_API");
        Answer ask = client.ask(text);
        return ask.getChosenAnswer();
    }
```

Рисунок 3.15 – Використання сервісу Google Bard

На рисунку 3.16 наведено приклад методу побудови запиту для обраної Rule-Based System, коли студент робить кілька помилок під час з вимови певних. Для побудови запиту використовується мова навчального матеріалу для цього реалізовано клас Language, а також самі слова з якими у студента була проблема. Після побудови запиту викликається метод getAnswerFromBard для якого передається змінна query та повертається відповідь від сервісу в форматі рядка.

```
public String buildQueryAboutPronunciationAndAskBard(String text, Language language) {
    String query;
    if (language == Language.ENGLISH) {
        query = String.format("I have a problem with the pronunciation of words %s." +
                " Please teach me how to pronounce" +
                " that within one paragraph and don't use more than 100 words", text);
    } else {
        query = String.format("У мене проблема з вимовою слів %s. Будь ласка, навчіть мене
    }
    return getAnswerFromBard(query);
}
```

Рисунок 3.16 – Побудова запиту на основі Rule-Based System

Для інтеграції сервісу розпізнавання за допомогою Google API Speech-to-text в систему правил створення запитів для персоналізованої підказки. Використано створений метод makeRecognitionAndSaveToResults, що реалізує всередині зв'язок з сервісом розпізнавання, та після чого отримано готовий розпізнаний текст, який можна використати в програмі. Сервіс розпізнавання викликає скрипт написаний на Python, який дозволяє швидко отримати відповідь у вигляді розпізнаного тексту від сервісу Google. Весь результат зберігаємо в створений клас RecognitionResult (рис. 3.18).

```
Language language = Language.valueOf(soundContent.getSound().getLanguage().toUpperCase());
RecognitionResult recognitionResult = makeRecognitionAndSaveToResults(savedFile.getPath(),
        language, soundContent.getContentText());

if (recognitionResult == null) {
    return ResponseEntity.badRequest().body(gson.toJson( src: ""));
}
```

Рисунок 3.17 – Збереження результатів розпізнавання в клас RecognitionResult

Після отримання результату та збереження в recognitionResult виконується перевірка чи містить результат не правильно вимовлений текст. І якщо так, тоді одразу зберігається нове слово в майбутній список слів для гри (GameService) для вивчення неправильно вимовлених слів у ігровому форматі, а також збільшується лічильник неправильно вимовлених слів. Лістинг методу для збереження слова та збереження результатів персоналізованої підказки представлені на рисунку 3.18.

```
if (!recognitionResult.getWrongText().isEmpty()) {
    gameService.addNewWord(recognitionResult.getWrongText());
    wrongAnswerCounter++;
}

int numberMistakes = 2;
if (wrongAnswerCounter == numberMistakes) {
    recognitionResult.setUserAssistantText(googleBardService
            .buildQueryAboutPronunciationAndAskBard(
                    recognitionResult.getWrongText(), language));
    wrongAnswerCounter = 0;
}

return ResponseEntity.ok().body(gson.toJson(recognitionResult));
```

Рисунок 3.18 – Обробка неправильно вимовлених слів

Коли лічильник доходить до визначеного значення мінімальної кількості помилок, тоді виконується функція для побудови запиту до сервісу GoogleBard, а також збереження отриманого тексту в результат розпізнавання для передачі відповіді у вигляді персоналізованої допомоги студентові.

Для гри необхідно створити сервіс, який зможе обробляти інформацію про слова, а також додавати їх в список для слів. Створений сервіс додає нові слова, коли студент робить помилки, а також при потребі відобразити їх в ігровій платформі використовує функції для випадкового вибору слів зі списку. Список слів містить популярні слова на випадок, якщо студент не матиме помилок у своїй вимові або матиме їх мало. У такий спосіб він зможе вивчати вимову слів у ігровій формі. Лістинг коду створеного сервісу представлено на рисунку 3.19.

```
@Service
public class GameService {

    private final List<String> listElements = new ArrayList<>(List.of("apple",
            "bow", "probably", "meal", "suddenly", "thick", "represent",
            "lesson", "bring", "quietly", "correct", "throat", "interior",
            "spite", "charge", "refer", "personal", "dust", "pride"));

    private final Random random = new Random();

    public String getRandomWord() {
        return listElements.get(random.nextInt(listElements.size()));
    }

    public void addNewWord(String word) { listElements.add(word); }
}
```

Рисунок 3.19 – Лістинг коду класу GameService

Коли студент навчається за допомогою гри і натискає клавішу «Перевірити запис» відбувається POST запит «/recognize-for-game» на розроблений контролер «RecognizeController» рисунок 3.20. Створений клас містить всередині метод позначений анотацією @PostMapping, що свідчить саме про виконання запиту типу POST, а також значення анотації із посиланням на яке звертається кінцевий користувач. Для успішного виконання запиту необхідно передати: файл, текст зображений на екрані гри, а також мову якою навчається студент як параметри запиту, що позначаються анотацією @RequestParam. Анотація RequestParam вказує, що параметр методу має бути прив'язаний до параметру веб-запиту.

```
@PostMapping(value = "/recognize-for-game", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<String> recognizeForGame(@RequestParam("file") MultipartFile file,
                                                @RequestParam("contentText") String textContent,
                                                @RequestParam("language") String language) {
    File savedFile = saveFileIntoResources(file);

    RecognitionResult recognitionResult = makeRecognitionAndSaveToResults(savedFile.getPath(),
            Language.valueOf(language.toUpperCase()), textContent);


    if (recognitionResult == null) {
        return ResponseEntity.badRequest().body(gson.toJson( src: ""));
    }

    return ResponseEntity.ok().body(gson.toJson(recognitionResult));
}
```

Рисунок 3.20 – Метод класу RecognizeController

Під час виконання методу відбувається збереження файлу в ресурси додатку для того щоб, керувати ним в майбутньому. Після отримання файлу, відбувається розпізнавання мови з файлу за допомогою методу makeRecognitionAndSaveToResults, якому необхідно також передати текст з гри й мову гри для збереження результатів використано створений клас RecognitionResult. Оскільки для майбутнього розширення додатку мови зберігаються у класі перечислення «enum», а веб-формат підтримує формат рядка необхідно визначити мову за допомогою методу valueOf, що дозволяє маючи лише рядок визначити чи існує в enum потрібна мова. Збережений

результат повинен містити правильно та неправильно вимовлений текст, а також за виконання умови текст персоналізованої підказки, щоб допомогти виправити студентам помилки. За умови що результату немає і він прирівнюється до значення null, надсилається помилка із статус кодом 400. Помилка на стороні клієнта обробляється та відображає повідомлення з невдалим результатом.

Після виконання усіх методів та перевірок результатом методу recognizeForGame є відправка результату клієнту із успішним статус кодом 200. Варто зазначити, що для відправки даних у веб форматі необхідно серіалізувати дані об’єкту результату recognitionResult у формат JSON. Оскільки саме формат JSON застосовується для інтеграції з сервісами та спілкуванні з різними модулями додатку. Для виконання серіалізації використано бібліотеку GSON. Google Gson – це безкоштовна бібліотека створена компанією Google для серіалізації та десеріалізації об’єктів Java. Робота із контролером на стороні клієнта представлена на рисунку 3.21. В даному випадку виконується функція gameRecognitionAndChangeState.

```
async function gameRecognitionAndChangeState() {
    let formData = new FormData();

    formData.append('file', savedAudio, 'testRecorder.wav');
    var contentText = document.getElementById("speech_target")
    formData.append("contentText", contentText.innerText);
    formData.append("language", getCookieByName('language'));

    await fetch('/recognize-for-game', {
        method: "POST",
        body: formData
    }).then(response =>
        response.json().then(data => ({
                data: data,
                status: response.status
            })
        ).then(result => {
                  if (!response.ok) {
                    successResult.textContent = 'The text was not recognized try again';
                    return;
                }
```

Рисунок 3.21 – Взаємодія із контролером Recognize Controller

Представлена функція спершу зберігає дані об’єкт FormData, щоб надіслати їх створеному контролері. Значення файлу, тексту та мови, повинні відповідати назві які були позначенні у методі створеного класу. Після збереження даних відправляється POST запит і очікується відповідь у форматі JSON та зберігаються параметри data, яка відповідає обєкту recognitionResult та status, що використовується як статус код. За умови що відповідь не є успішною відобразиться помилка, в іншому випадку відбувається виконання зміни стану гри.

Для заповнення даними нове середовище, щоб не створювати їх вручну, вирішено створити скрипт, який буде додавати об’єкти в базу даних, щоб був матеріал з яким можна працювати. Перш за все необхідно створити скрипт з назвою data.sql та розмістити його в директорію з ресурсами «src/main/resources». Після чого налаштовувати Spring Boot, щоб автоматично запускати створений скрипт після ініціалізації схеми бази даних, щоб заповнити середовище даними. Цей підхід є корисним для розробки та дозволить зекономити час при майбутньому тестуванню. Приклад створення даних в скрипті представлено на рисунку 3.22.

```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

INSERT INTO public.sound(
    id, language, name)
    VALUES (uuid_generate_v4(), 'English', 'Ch');

INSERT INTO public.sound(
    id, language, name)
    VALUES (uuid_generate_v4(), 'English', 'A');

INSERT INTO public.sound(
    id, language, name)
    VALUES (uuid_generate_v4(), 'English', 'B');
```

Рисунок 3.22 – Скрипт створення даних

Отже, створено компоненти, які визначено основними для додатка, виконано детальний опис функцій та продемонстровано лістинг коду. Під час

розробки компонент дотримано шаблони проєктування MVC та GOF. Повний лістинг наведено у додатку В.

3.6 Висновки

Під час розробки програмного додатка обрано шаблони проєктування GOF та MVC (Model-View-Controller) через їхні модульні та масштабовані атрибути. MVC дозволить створити додаток модульним, зручнішим в обслуговувані. Розроблена діаграма послідовності демонструє розуміння потоку дій студента під час користування програмним забезпеченням, а детальна діаграма бізнес-процесу забезпечить чітке уявлення про робочий процес додатка. спрямоване на підвищення ефективності вивчення правильної вимови слів. Обрано мову програмування та середовище розробки, а також інструменти, які пропонують гнучкість і покращують ефективність розробки, що дозволяє створювати адаптивну та інтуїтивно зрозумілу програму. Обрано СУБД для зберігання даних користувача та навчальних матеріалів. Під час порівняння мов програмування: Java, C# і Ruby, Java виявляється кращим вибором завдяки своїй незалежності від платформи, широкій підтримці бібліотек і потужній підтримці спільноти, а також майбутній веб-масштабованості.

Розроблено графічний інтерфейс користувача за допомогою Bootstrap, Thymeleaf. Розроблений інтерфейс, є не тільки візуально привабливим, але й інтуїтивно зрозумілим, що дозволяє користувачам легко орієнтуватися в програмному забезпеченні. Кожен створений компонент програмного забезпечення було розроблено базуючись на шаблонах проєктування, це дозволяє забезпечити ефективне виконання поставленої мети.

Розроблені програмні засоби дозволять покращити вивчення мови студентів. Поєднавши головні принципи розробки програмного забезпечення створено інструмент, який не тільки робить вивчення вимови ефективнішим, але й більш привабливим.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз методів тестування програмного забезпечення

Тестування ПЗ має ключове значення для забезпечення того, що програмний продукт відповідає очікуваним стандартам і виконує призначені функції без помилок. Це гарантує, що програма є високоякісною, безпечною, надійною та зручною для користувача. У процесі тестування програмного забезпечення використовуються різні методи, кожен з яких має певну мету та підхід [42].

Тестування чорної скриньки зосереджено на дослідженні зовнішньої поведінки програмного забезпечення без уявлення про його внутрішні механізми чи код [43]. Головною перевагою є те, що такий метод підходить для тестувальників без технічних знань внутрішньої системи, а тестові випадки можуть бути розроблені, як тільки специфікації будуть завершені. Але також існують недоліки, а саме: тестувальник без знань внутрішньої системи може пропустити деякі помилки, тестові приклади можуть бути зайвими, якщо розробник і тестувальник створюють тестові приклади для однакових функцій.

Тестування білого ящика стосується внутрішніх структур програми. Тут тестувальники мають доступ до кодової бази та можуть запускати програму в режимі налагодження. Основною перевагою є ретельність перевірки, оскільки код доступний особі, яка перевіряє ПЗ в такому випадку легше виявити приховані помилки. Недоліками підходу є необхідність кваліфікованих тестувальників з глибоким розумінням мови програмування та логіки програми, а також витрата часу, оскільки робота з кодовою базою потребує багато ресурсів.

Позитивне тестування – перевірка програмного забезпечення шляхом надання йому очікуваних вхідних даних. Під час проведення позитивного тестування відбувається перевірка, що програмне забезпечення працює належним чином у звичайних сценаріях.

Негативне тестування – тестування стійкості та стабільності програмного забезпечення проти несподіваного введення даних й поведінки користувача.

Такий підхід допомагає визначити можливі збої, системи при неочікувані поведінці користувача.

Модульне (Unit) тестування – фокусується на окремих блоках або компонентах програмного забезпечення, щоб забезпечити їхнє правильне функціонування [44]. Мета полягає в тому, щоб підтвердити, що кожна одиниця програмного забезпечення працює відповідно очікуваних результатів. Блоком або компонентом є найменша тестована частина будь-якого програмного забезпечення, яка може бути функцією, методом або класом. Помилки можна виявити на ранній стадії розробки. Кожен модуль можна тестувати одночасно, що прискорює процес перевірки роботи програмного забезпечення. Недоліком підходу є відсутня перевірка інтеграції між модулями. Ключовими характеристиками є:

Ізоляція: Unit тест має перевіряти компонент ізольовано, що означає відсутність залежності від зовнішніх систем або компонентів. Це гарантує, що тест перевіряє лише функціональність конкретного компонента, а не будь-які зовнішні фактори.

Автоматизація: Unit тести зазвичай автоматизовані. Це забезпечує швидкий зворотний зв’язок із розробниками та гарантує повторюваність тестів і можливість їх постійного виконання.

Швидке виконання: через їхню ізольованість Unit тести мають виконуватися швидко, що дозволяє розробникам запускати їх часто, не сповільнюючи процес розробки.

Простота рефакторингу: якщо модульні тести було побудовано по правилах розробники можуть впевнено вносити нові зміни в наявну кодову базу, знаючи, що отримають сповіщення, якщо зміни зламали наявні методи.

Тестування інтеграції включає тестування інтегрованих модулів або компонентів, щоб переконатися, що вони працюють безперебійно в поєднанні між собою. Такий метод допомагає виявити недоліки роботи між модулями, а також забезпечить плавний потік даних між модулями. Недоліком є необхідність «заглушки» (Mocks), у випадку, якщо один модуль ще не розроблено або реальне

відтворення результату займає багато часу, а сам функціонал модуля вже покрито тестами.

Отже, вибір методу тестування часто залежить від вимог проєкту, стадії розробки та наявних ресурсів. Зазвичай в одному життєвому циклі розробки ПЗ застосовуються кілька методів тестування. Використовуючи комбінацію цих методів, розробники програмного забезпечення та тестувальники можуть переконатися, що додаток є надійним, зручним для користувача та позбавленим критичних помилок.

4.2 Тестування розробленого програмного застосунку

Під час проведення тестування створеного застосунку, необхідно глибоко зануритись у сферу тестування програмного забезпечення. Хоча існує безліч доступних методів тестування, варто зосередити увагу на «модульних тестах». Оскільки покриття тестами фундаментальних компонентів додатка, забезпечать належне функціонування кожного окремого компонента програмного забезпечення. Завдяки правильно створеним модульним тестам буде виявлено розбіжності в роботі додатка на ранній стадії, а також це закладе міцну основу для наступних етапів тестування.

Для економії часу в майбутній розробці вирішено розробити юніт тести для головних компонентів розробленого ПЗ, оскільки такі тести можуть зекономити години, а то й дні дебагінгу. Це також дозволить зменшити кількість помилок після модифікації коду та підвищити якість коду. Створені тести спростять рефакторинг і внесення усіх змін, оскільки перевірити чи програма зможе працювати належним чином можна буде за кілька хвилин.

Для створення юніт тестів для розробленого сервісу отримання відповідей від Google Bard перш за все додамо залежність в тестовий клас використовуючи анотацію @Autowired. Після чого сторено тест, що перевіряє звичайну відповідь на будь-яке питання, а також тест в межах якого будується запит на основі слова та обраної мови. Для збереження і спрощення операцій із мовами використано enum клас Language, в якому збережено назву мови та її унікальний код. Приклад

побудови юніт тестів для сервісу Google Bard продемонстровано на рисунку 4.1.

Створені тести для перевірки справності роботи сервісу для отримання відповідей по правильній вимові слів повинні покрити наступні перевірки: складних виразів, простих слів, отримання відповідей на слова як українською мовою, так і англійською.

```
private static final Logger LOGGER = Logger.getLogger(GoogleBardServiceTest.class.getName());

@Autowired
GoogleBardService googleBardService;

@Test
void testSimpleAnswerFromGoogleBardService() {
    String answerFromBard = googleBardService.getAnswerFromBard( text: "What is it today?");
    LOGGER.info( msg: "Answer from BARD: " + answerFromBard);
    Assertions.assertFalse(answerFromBard.isEmpty(),  message: " Answer from BARD shouldn't be empty");
}

@Test
void testQueryFromGoogleBardService() {
    String wordForTesting = "schedule";
    String answerFromBard = googleBardService.buildQueryAboutPronunciationAndAskBard(wordForTesting, Language.ENGLISH);
    LOGGER.info( msg: "Answer from BARD: " + answerFromBard);
    Assertions.assertTrue(answerFromBard.contains(wordForTesting),  message: " Answer from BARD should contains word schedule");
}
```

Рисунок 4.1 – Тестування сервісу Google Bard

Результат виконання створених 18 тестів для основних компонент програми представлено на рисунку 4.2.

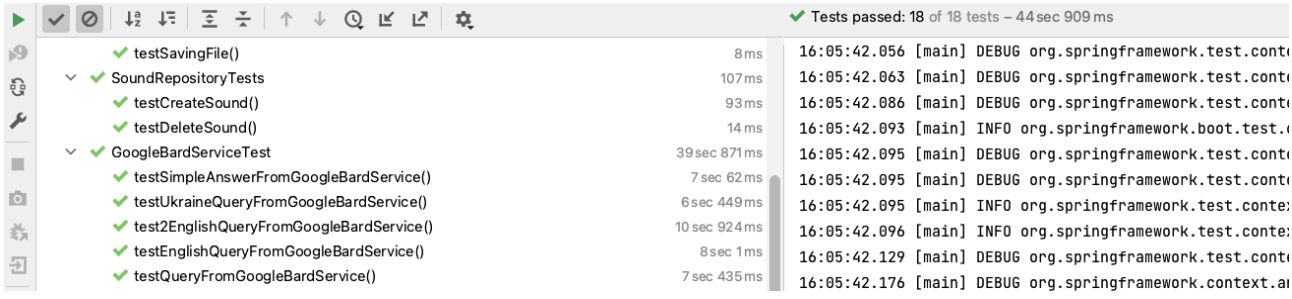


Рисунок 4.2 – Виконання тестів для головних компонент розробленої програми

Аналіз запуску тестів свідчить, що побудовані тести є успішними, робота усіх методів виконується як і очікувалось та відповідає вимогам. Крім того, потрібно провести ручне тестування та перевірити справність роботи помічника. Перш за все варто створити тест-кейси.

Тест-кейс №1 Перевірка роботи помічника англійською мовою.

1. Відкрити веб-додаток;
2. Обрати англійську мову;
3. Перейти до доступного звуку;
4. Обрати матеріал для вивчення;
5. Натиснути на клавішу Start recording;
6. Неправильно вимовити заданий текст двічі;
7. Відправити результат на перевірку.

Успішним результатом виконання тесту є поява вигулькуючого вікна у вигляді Alert. Що блокує взаємодію з додатком, поки студент не закриє його. Вікно повинно містити текст з описом як правильно вимовити слово, в якому було допущено помилку.

Для виконання тесту було обрано 2 наступних слова: Schedule, sceptically. Під час виконання 6 кроку слово Schedule вимовлено правильно, а в слові sceptically допущено помилку. Результат виконання першого тестового випадку продемонстровано на рисунку 4.3.

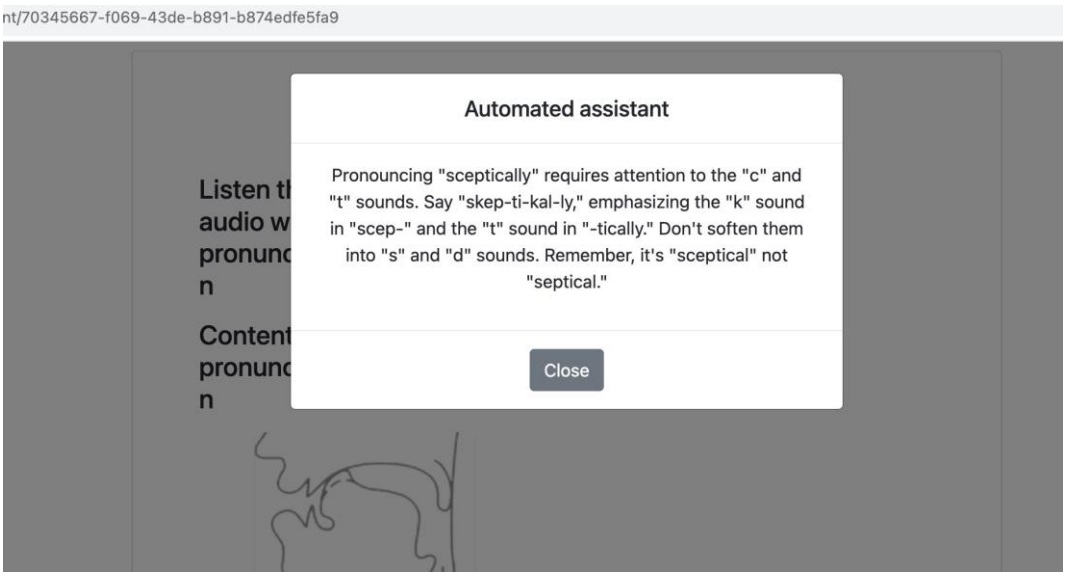


Рисунок 4.3 – Результат виконання тест-кейсу №1

Другий випадок повинен містити перевірку подібного функціоналу, але українською мовою. Окрім самого тексту, який містить опис з допомогою по вимові слів, усі елементи сторінки та вигулькуючого вікна повинні бути українською.

Тест-кейс №2 Перевірка роботи помічника українською мовою.

1. Відкрити веб-додаток;
2. Обрати українську мову;
3. Перейти до доступного звуку;
4. Обрати матеріал для вивчення;
5. Натиснути на кнопку «Почати запис»;
6. Неправильно вимовити заданий текст двічі;
7. Відправити результат на перевірку.

Успішним результатом виконання тесту є поява вигулькуючого вікна. Вікно повинно містити текст з описом українською мовою як правильно вимовити слово, в якому було допущено помилку.

Для виконання тесту було обрано 3 наступних слова: Їжачок, їхати, їсти. Під час виконання 6 кроку слова їхати та їсти вимовлено правильно, а в слові їжачок допущено помилку. Результат виконання тестового випадку продемонстровано на рисунку 4.4.

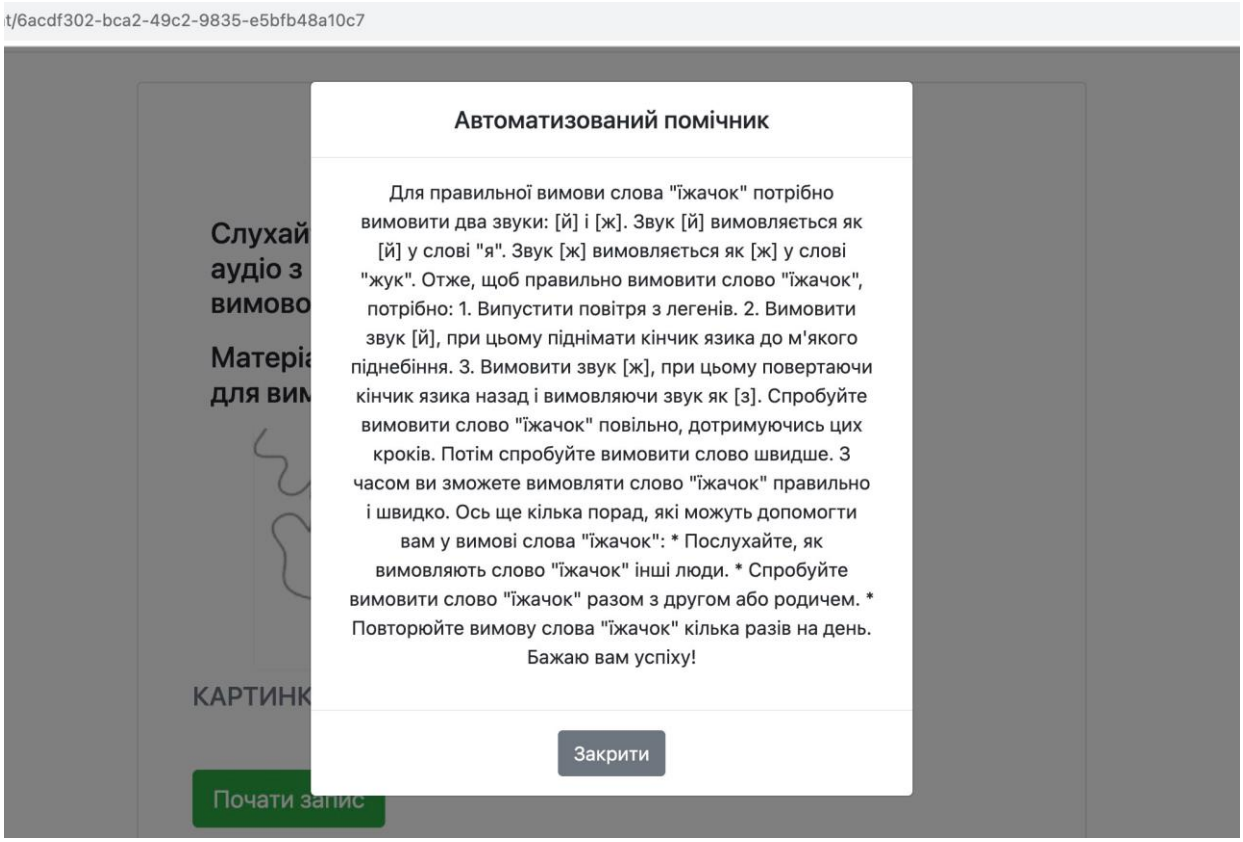


Рисунок 4.4 – Результат виконання тест-кейсу №2

Тест-кейс №3 Перевірка блокування екрана під час відображення вікна з допомогою вимови.

1. Відкрити веб-додаток;
2. Перейти до доступного звуку;
3. Обрати матеріал для вивчення;
4. Натиснути на кнопку «Почати запис»;
5. Неправильно вимовити заданий текст двічі;
6. Відправити результат на перевірку;
7. Закрити вигулькуюче вікно.

Успішним результатом виконання тесту є можливість взаємодії з навчальним матеріалом після закриття вікна. Результат розблокованого екрана представлено на рисунку 4.5.

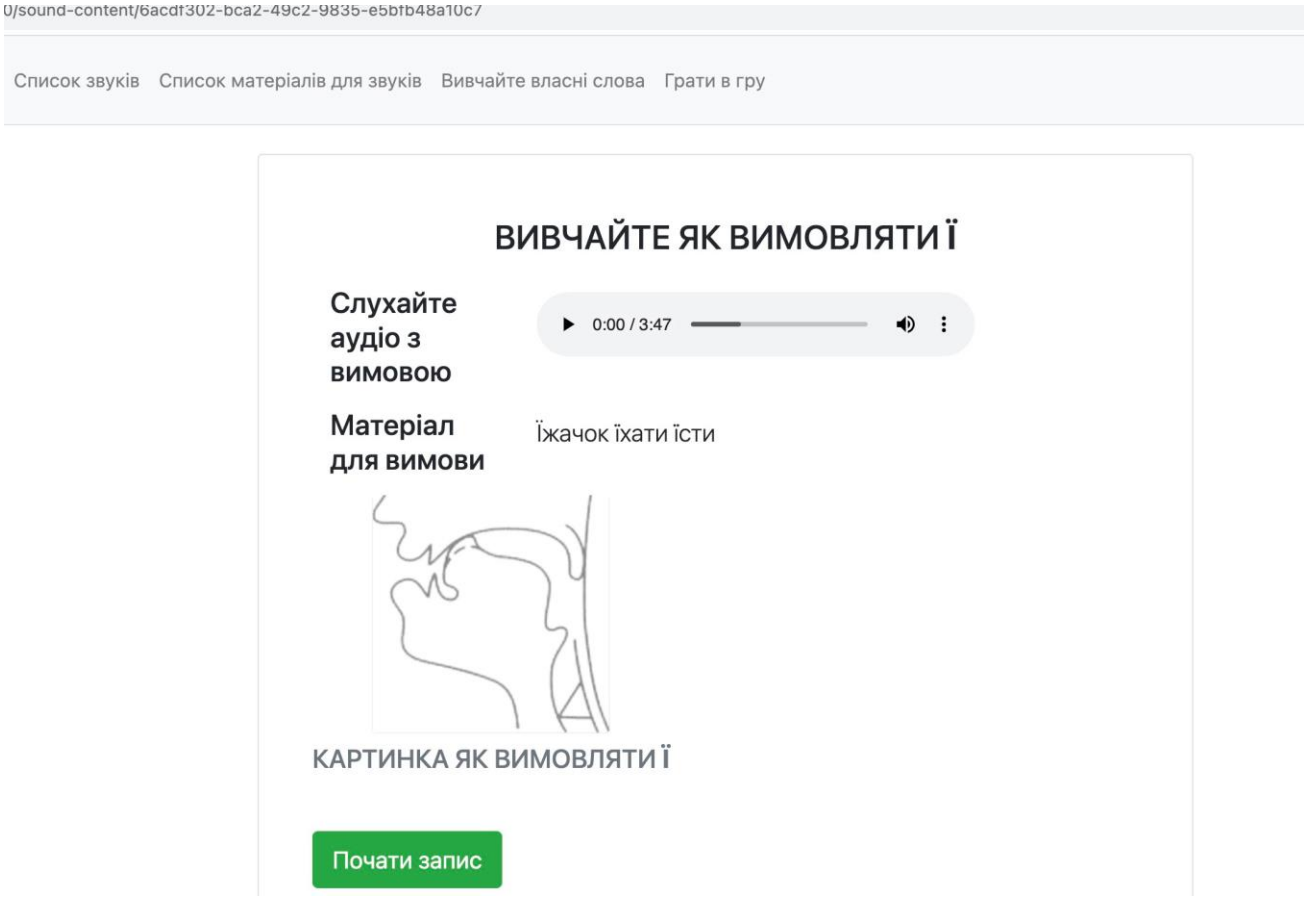


Рисунок 4.5 – Результат виконання тест-кейсу №3

Тест-кейс №4 Перевірка нарахування балів за правильно вимовлене слово.

1. Відкрити веб-додаток;

2. Перейти до сторінки з грою;

3. Натиснути на кнопку «Почати запис» та правильно вимовити слово на екрані;

4. Відправити результат на перевірку.

Успішним результатом виконання тестового випадку є атакуюча анімація героя, збільшення рахунку гравця, незмінна кількість життів, поява наступного слова. Результат виконання тесту представлено на рисунку 4.6.

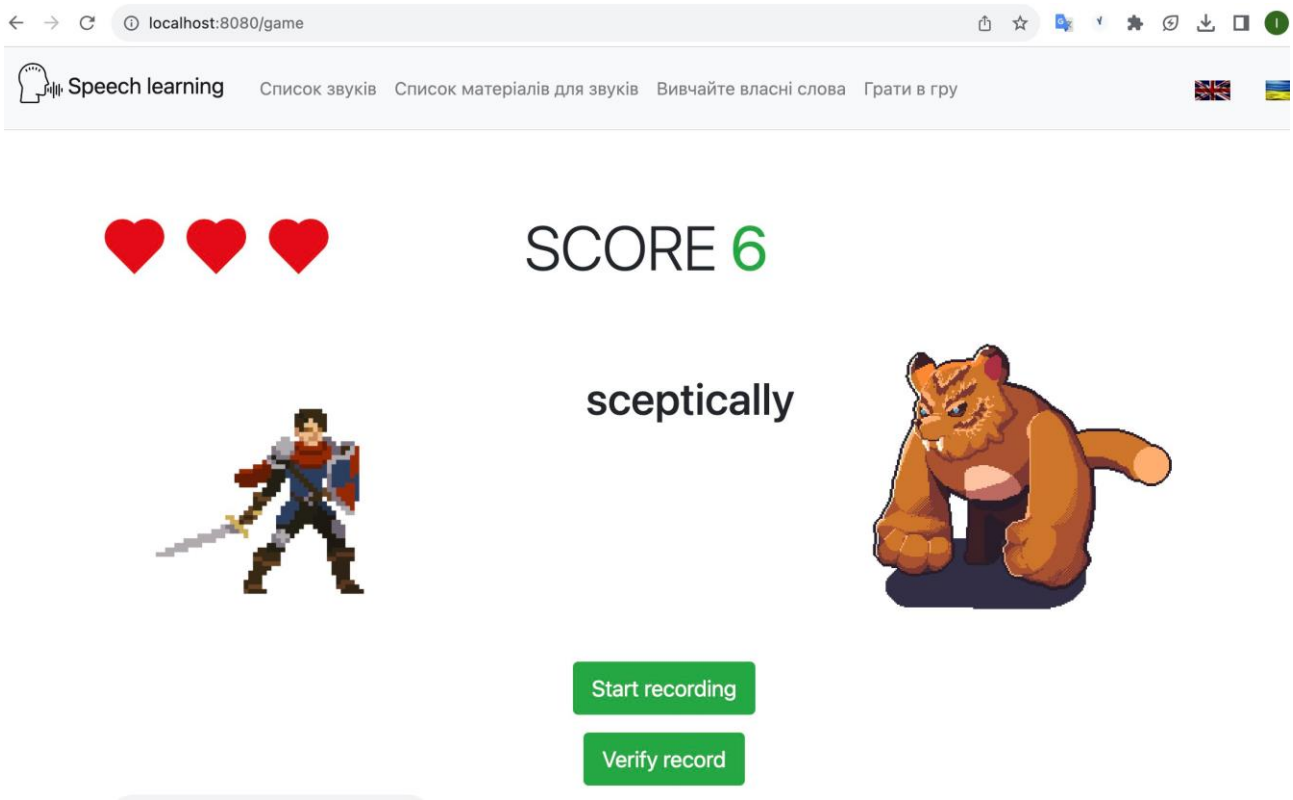


Рисунок 4.6 – Результат виконання тест-кейсу №4

Тест-кейс №5 Перевірка зменшення життів за неправильно вимовлене слово. Результат виконання тесту представлено на рисунку 4.7.

1. Відкрити веб-додаток;

2. Перейти до сторінки з грою;

3. Натиснути на кнопку «Почати запис» та не правильно вимовити слово яке доступне на екрані;

4. Відправити результат на перевірку.

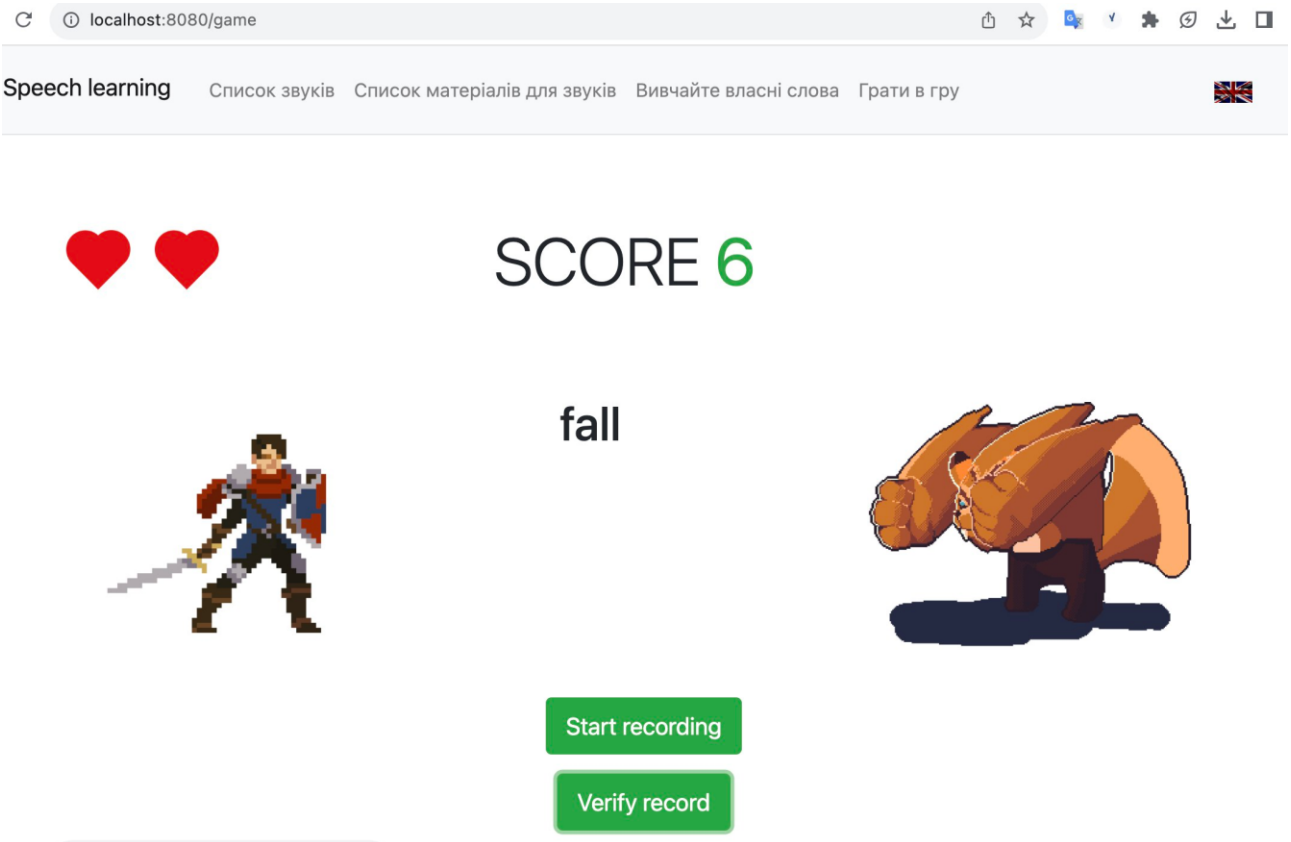


Рисунок 4.7 – Результат виконання тест-кейсу №5

Успішним результатом виконання тестового випадку №5 є атакуюча анімація монстра, зменшення кількості життів на 1, поява наступного слова для вивчення.

Тест-кейс №6 Перевірка завершення гри коли.

1. Відкрити веб-додаток;
2. Перейти до сторінки з грою;
3. Натиснути на кнопку «Почати запис»
4. Не правильно вимовити слово яке доступне на екрані;
5. Відправити результат на перевірку;
6. Отримати результат неправильно вимовленого слова;
7. Повторити кроки 3-6 двічі.

Успішним результатом виконання тестового випадку є атакуюча анімація монстра, анімація героя має бути відображати падіння на землю, що свідчить про завершення гри, кількість життів героя має бути 0, а також має відобразитись вигулькуюче вікно з відображенням результатів.

Результат виконання тесту представлено на рисунку 4.8.

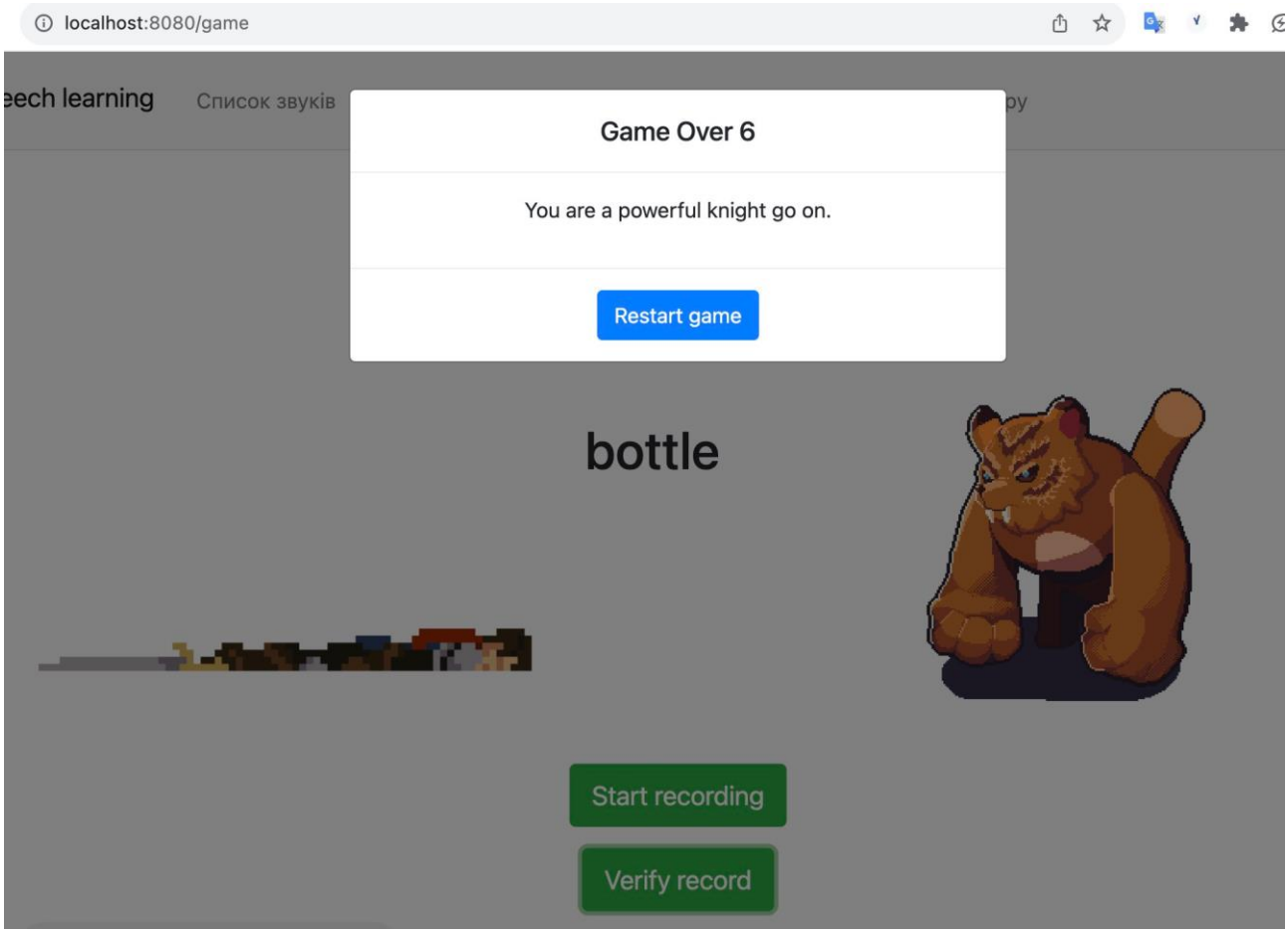


Рисунок 4.8 – Результат виконання тест-кейсу №6

Тест-кейс №7 Перевірка перезапуску гри.

1. Відкрити веб-додаток;

2. Перейти до сторінки з грою;

3. Натиснути на кнопку «Почати запис» та не правильно вимовити слово яке доступне на екрані;

4. Відправити результат на перевірку;

5. Повторити кроки 3-4 двічі;

6. Натиснути клавішу Restart game.

Успішним результатом виконання тестового випадку є спокійний стан обох персонажів на екрані, кількість життів повертається до стартової кількості, рахунок гравця стає 0. Результат виконання тест-кейсу №7 представлено на рисунку 4.9.

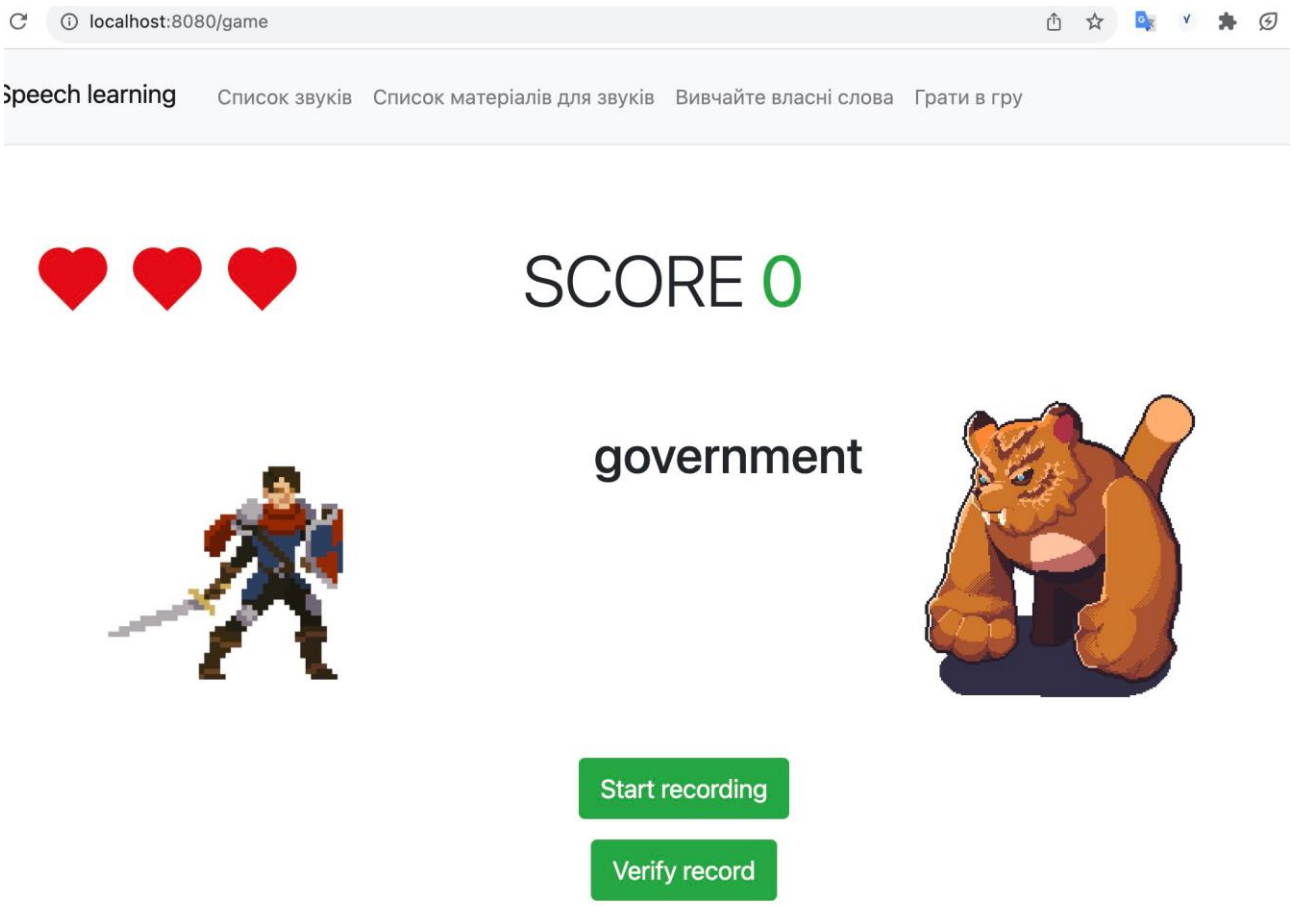


Рисунок 4.9 – Результат виконання тестового випадку №7

Тест-кейс №8 Перевірка відображення неправильно вимовлених слів у грі.

1. Відкрити веб-додаток;
2. Перейти до доступного звуку;
3. Обрати матеріал для вивчення;
4. Натиснути на кнопку «Почати запис»;
5. Неправильно вимовити заданий текст;
6. Відправити результат на перевірку;
7. Повторити кроки 4-6 двічі;
8. Перейти до сторінки з грою.

Успішним результатом виконання тестового випадку є відображення неправильно вимовленого слова у навчальному матеріалі. Якщо не правильно вимовлених слів було декілька необхідно пограти в гру, поки не з’явиться очікуване слово. Результат виконання тесту-кейсу №8 представлено на рисунку 4.10.

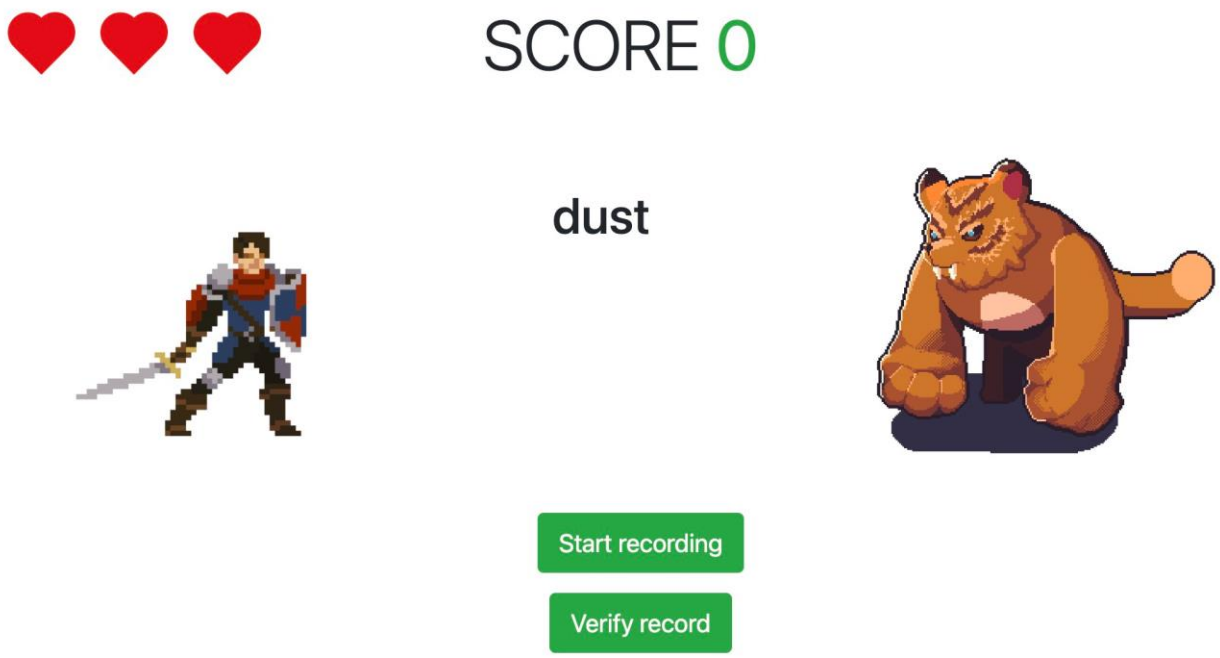


Рисунок 4.10 – Результат виконання тестового випадку №8

Отже, було створено модульні тести, які перевіряють головні компоненти додатка. Наведено приклад тестів, які перевіряють роботу сервісу Google Bard, який використовується для отримання відповідей персоналізованої підказки, що дозволяє перевірити чи розроблений метод динамічної генерації персоналізованої підказки працює належно і надає очікуваний для користувача результат. Створено тестові випадки для виконання ручного тестування та перевірки головних функцій додатка, які підвищують ефективність вивчення правильної вимови слів. Загалом створено 18 тестів, які перевіряють розроблений програмний код та 8 тест-кейсів для ручного тестування.

4.3 Аналіз ефективності розроблених методів

Оскільки метою роботи є підвищення ефективності вивчення правильної вимови слів, що базовано на використанні динамічної генерації персоналізованої підказки при допущених помилах та гейміфікації навчального процесу вивчення вимови слів, необхідно провести дослідження, яке дозволить визначити чи було досягнуто поставлену мету.

Перш за все необхідно оцінити ступінь узгодженості думок експертів. Для

оцінки думок за напрямком підвищення ефективності використано коефіцієнт конкордації [45].

Для оцінки узгодженості чи узгодженості думок серед респондентів одним із поширених методів є використання коефіцієнта конкордації Кендалла (W). Коефіцієнт конкордації Кендалла (W) – це непараметрична статистика, яка використовується для вимірювання згоди між оцінювачами. Значення W коливається від 0 (немає згоди) до 1 (повна згода) [46].

В якості експертів для оцінювання обрано людей, які займаються вивченням мови. Для цих осіб поширено додаток та на основі їхнього досвіду отримано відгуки в Google Forms тому необхідно їх проаналізувати, щоб визначити чи розроблені методи збільшують ефективність вивчення (рис. 4.11).

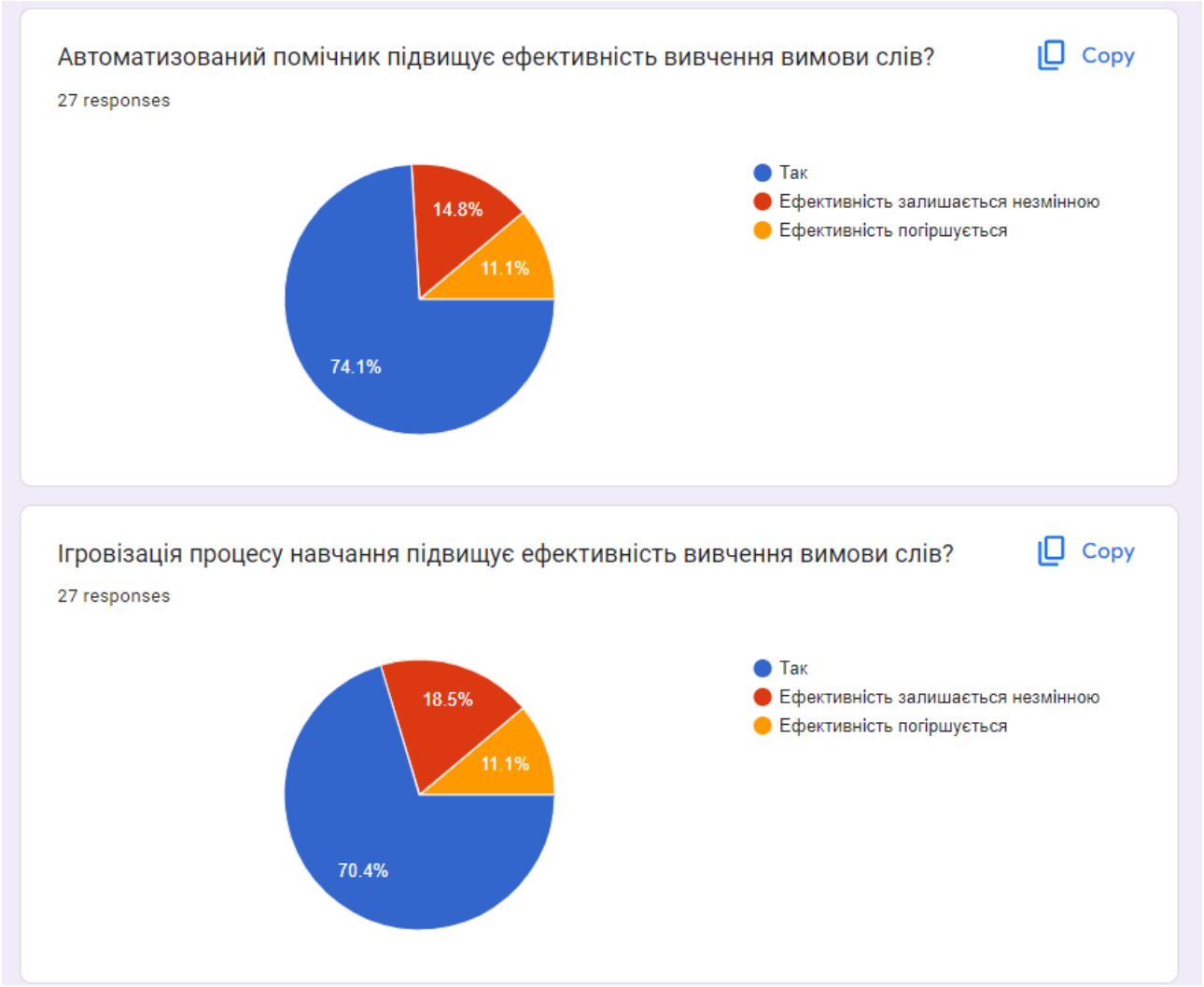


Рисунок 4.11 – Відгуки про розроблені методи на основі створеного додатка

Всього зібрано 27 відгуків на два запитання, що стосуються розроблених методів, а саме: «Автоматизований помічник підвищує ефективність вивчення вимови слів?» та «Ігровізація процесу навчання підвищує ефективність вивчення вимови слів?». Кожне питання має відповідь: «Так», «Ефективність залишається незмінною» та «Ефективність погіршується».

На перше запитання отримано наступні відповіді:

1. «Так» – 74.1%;
2. «Ефективність залишається незмінною» – 14.8%;
3. «Ефективність погіршується» – 11.1%.

На друге запитання отримано наступні відповіді:

1. «Так» – 70.4%;
2. «Ефективність залишається незмінною» – 18.5%;
3. «Ефективність погіршується» – 11.1%.

Враховуючи отримані дані з відгуків, необхідно обчислити коефіцієнт конкордації W за формулою:

$$W = \frac{12\ \sum S^2}{k^2(n^3 - n)}, \tag{4.1}$$

де $\sum S^2$ – є сумою квадратів рангових сум для кожного питання; k – кількість категорій; n – кількість оцінювачів або респондентів.

Було отримано такі значення рангів (для кожної категорії: більше значення означає вищий ранг):

1. «Так»: 3;
2. «Ефективність залишається незмінною»: 2;
3. «Ефективність знижується»: 1.

Порахуємо рангову суму для першого запитання, використовуючи такі дані:

1. «Так»: 74,1% з 27 = 20 осіб;
2. «Ефективність залишається незмінною»: 14,8% з 27 = 4 людини;

3. «Ефективність погіршується»: 11,1% з 27 = 3 людини.

Тоді

$$S_1 = 20 \times 3 + 4 \times 2 + 3 \times 1 = 71.$$

Порахуємо рангову суму для другого запитання, використовуючи такі дані:

1. «Так»: 70,4% з 27 = 19 осіб;
2. «Ефективність залишається незмінною»: 18,5% з 27 = 5 осіб;
3. «Ефективність знижується»: 11,1% з 27 = 3 людини.

Тоді

$$S_2 = 19 \times 3 + 5 \times 2 + 3 \times 1 = 70.$$

Визначимо за раніше вказаною формулою (4.1) коєфіцієнт Кендалла:

$$W = \frac{12 * (5041 + 4900)}{9(19656 - 27)} = 0.674. \quad (4.2)$$

На основі аналізу відповідей 27 осіб про ефективність автоматизованого помічника та гейміфікації для вивчення вимови виявлено, що більшість (74,1% і 70,4% відповідно) вважали, що й автоматизований помічник, і гейміфікація підвищують ефективність. Невеликий відсоток відчував відсутність змін або зниження ефективності. Конкордація серед респондентів, що розрахована за допомогою коефіцієнта Кендалла, виявилася 0.674, що вказує на помірну згоду серед експертів. Чим ближче значення *W* до 1, тим більша згода експертів, а тому можна вважати, що розроблені методи, на думку більшості експертів, підвищують ефективність вивчення правильної вимови слів.

4.4 Вимоги до персонального комп’ютера

Для забезпечення безперебійної та ефективної роботи веб-додатка призначеного для вивчення вимови слів, персональний комп’ютер (ПК)

користувача повинен відповідати певним вимогам. Щоб допомогти користувачам переконатися, що комп’ютер добре обладнаний для оптимальної роботи необхідно створити вимоги для ПК. Вимоги до обладнання представлені на таблиці 4.1

Таблиця 4.1 – Вимоги до обладнання

Процесор (CPU)	Двох ядерний процесор із мінімальною тактовою частотою 2,0 ГГц. Рекомендовано сучасний чотирьох ядерний процесор для бездоганної багатозадачності.
Пам’ять (RAM)	Мінімум 4 ГБ RAM. Для покращення продуктивності, особливо з кількома відкритими вкладками браузера або іншими програмами, рекомендується 8 ГБ оперативної пам’яті або більше.
Зберігання	Принаймні 2 ГБ вільного місця на диску для тимчасових файлів і кешу браузера. Для швидшого завантаження рекомендується SSD.
Звукова карта	Вбудована або спеціальна звукова карта для забезпечення чіткого відтворення аудіо файлів, які наявні у навчальних матеріалів для вивчення вимови.
Мікрофон	Є обов’язковою вимогою для вивчення правильної вимови слів, щоб у студента була можливість записати власну вимову та відправити файл на перевірку.
Динаміки або навушники	необхідні для прослуховування прикладів вимови. Для чіткого звуку рекомендується використовувати динамік або навушники хорошої якості.

Крім вимог до обладнання необхідно визначити основні вимоги до програмного забезпечення, щоб користувач міг успішно користуватись розробленим програмним додатком. Розроблені вимоги представлені на таблиці 4.2.

Таблиця 4.2 – Вимоги до програмного забезпечення

Операційна система	Програма є веб-додатком, тому є незалежною від платформи. Однак рекомендується мати: Windows 10 або новіша версія, MacOS 10.13 (High Sierra) або новішої версії, дистрибутиви Linux, випущені за останні п’ять років.
Веб-браузер	Для найкращої роботи потрібна остання версія одного з наступних браузерів: Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge.
JavaScript	Має бути ввімкнено в браузері, оскільки більшість веб-додатків використовує його для запису аудіо файлу та для відправки на перевірку.

Дозволи, які необхідно надати для користування функціями розробленого додатка на прикладі браузера Google Chrome наведені на рисунку 4.12.

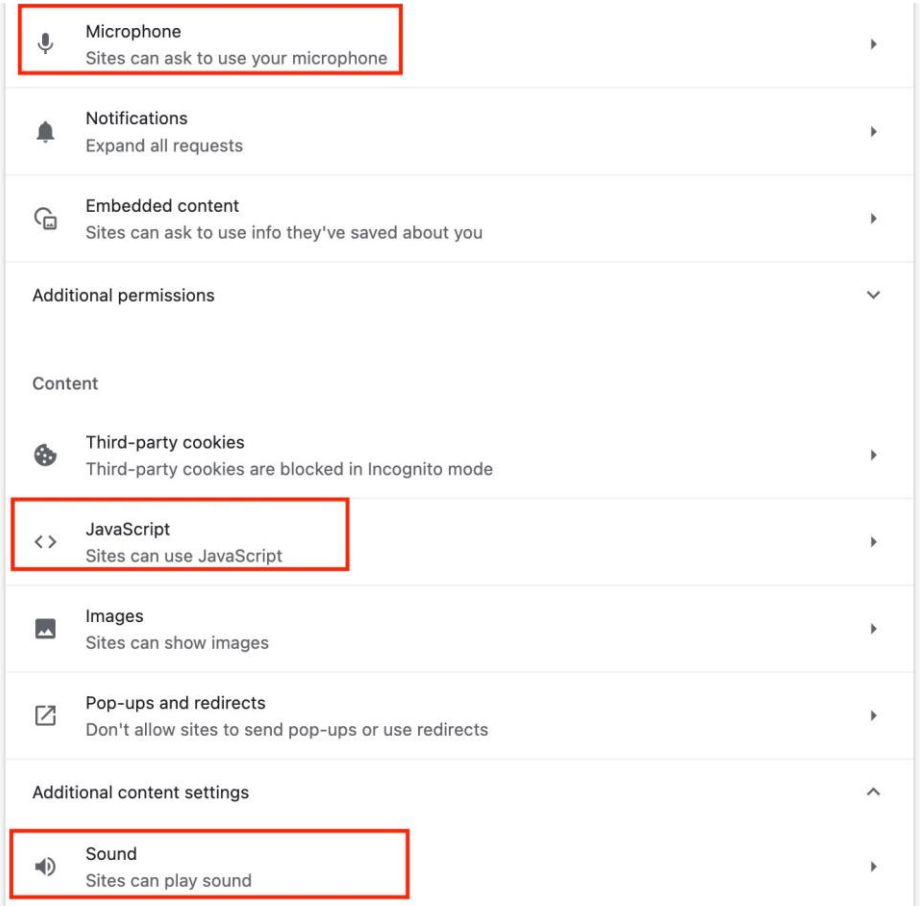


Рисунок 4.12 – Набір дозволів для користування додатком

Отже, щоб максимально використати створений додаток для вивчення правильної вимови слів, користувачі повинні переконатися, що їхні персональні комп’ютери відповідають або перевищують вищезазначені вимоги. Завдяки цьому вони можуть мати бездоганний і збагачувальний досвід навчання без технічних збоїв.

5.5 Висновки

У створеному розділі поглиблено оглянуто тонкощі тестування програмного забезпечення, зосереджуючись як на популярних методологіях, так і на спеціалізованих тестах, створених спеціально для розробленого програмного забезпечення. Під час аналізу популярних методів, що використовуються для тестування ПЗ, виділено їх актуальність і ефективність.

Для розробленого програмного забезпечення було розроблено спеціальні тести, які складалися з 18 автоматизованих тестів і 8 ручних тестів. Представлені результати цих тестів, які відображають продуктивність і надійність створеного програмного рішення. Для оцінки впливу розроблених методів на ефективність вивчення правильної вимови слів було проведено опитування серед користувачів. Для подальшого аналізу застосовано коефіцієнт узгодженості Кендалла, що дозволив зрозуміти узгодженість відповідей користувачів щодо ефективності програми.

Для забезпечення оптимальної взаємодії з користувачем описано як мінімальні, так і рекомендовані вимоги до обладнання, необхідні для безперебійної роботи зі створеним додатком на персональному комп’ютері. Завдяки всебічному тестуванню та аналізу гарантовано, що розроблене програмне забезпечення не тільки відповідає встановленим стандартам, але й резонує з цільовими користувачами.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів» є оцінювання науково-технічного рівня та

рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [47].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					

Продовження таблиці 5.1

6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 5.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає коштів	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	3	3
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	37	39
Середньоарифметична сума балів $СБ_c$	38		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [47].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів» становить 38 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам 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, обчисленої за посадовими окладами, відрядними розцінками, тарифними

ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [47]:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

$М_{ni}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

$Т_р$ – середнє число робочих днів в місяці, $Т_р$=22 дні.

$$З_о = 25000{,}00 \cdot 65 / 22 = 73863{,}64 \text{грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	25000	1136,36	65	73863,64
Інженер-розробник програмного забезпечення	23000	1045,45	60	62727,27
Консультант	18000	818,18	5	4090,91
Всього				140681,82

Основна заробітна плата робітників

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

$$З_р = \sum_{i=1}^{n} С_i \cdot t_i, \quad (5.2)$$

де $С_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [47].

K_c – 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.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (22 \cdot 8) = 69{,}09$ грн.

$З_{р1} = 69{,}09 \cdot 10{,}00 = 690{,}94$грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочого місця дослідника	10	2	1,1	69,09	690,94
Інсталяція програмного забезпечення	2	5	1,7	69,09	138,19
Тестування системи	10	5	1,7	106,78	1067,81
Всього					1896,94

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (140681,82+ 1896,94) · 11 / 100% = 15683,66грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

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

Зн = (140681,82+ 1896,94+ 15683,66) · 22 / 100% 34817,73214грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.6)$$

де $Н_j$ – 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*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1 = 2 \cdot 200,00 \cdot 1,1 - 0,000 \cdot 0,00 = 440,0$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір А4-500-80	200	2	0	0	440
Канцелярське приладдя (набір)	150	2	0	0	330
					770

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

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

Витрати на спецустаткування, які використовують при проведенні НДР на тему «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів» відсутні.

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

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, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 6200 \cdot 1 \cdot 1,1 = 6944$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.7– Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
IntelliJ IDEA Ultimate	1	6200	6944
Графічний редактор Adobe Photoshop	1	650	728
Всього			7672

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (20000,00 \cdot 3) / (5 \cdot 12) = 1000$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
ПК	20 000	5	3	1000,00
Монітор	5 000	3	3	416,67
Оргтехніка	6000	4	2	250,00
IntelliJ IDEA Ultimate	6944	3	3	578,67
Графічний редактор Adobe Photoshop	728	3	3	60,67
Всього				2306,00

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e = 7{,}50$ грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0{,}2 \cdot 540{,}0 \cdot 7{,}50 \cdot 0{,}95 / 0{,}97 = 793{,}30$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп’ютер	0,2	540	793,30
Монітор	0,08	540	317,32
Робоче місце дослідника	0,15	540	594,97
Оргтехніка	0,45	20	66,11
Всього			1771,7

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.10)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (140681,82+ 1896,94) · 20 / 100% = 28515,75грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.11)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 40%.

$В_{сп}$ = (140681,82+ 1896,94) · 40 / 100% = 57031,50грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.12)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 60%.

$I_в$ = (140681,82+ 1896,94) · 60 / 100% = 85547,25.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.13)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$В_{нзв}$ = (140681,82+ 1896,94) · 100 / 120% = 171 094,51 грн.

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

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.14)$$

$В_{заг}$= 547788,87грн.

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

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (5.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 547788,87/ 0,9 = 608654,30 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 5000 користувачів;

2-й рік –7000 користувачів;

3-й рік – 6000 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 25000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 1500 грн як платна підписка для одного користувача на рік;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів

від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [47].

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ = (500·25000,00+2000·5000)·0,83·0,4·(1-0,18/100%)= 6147540 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ = (500·25000,00+2000·(5000+7000))·0,83·0,4(1-0,18/100%)= 9972676грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ = (500·25000,00+2000·(5000+7000+6000))·0,83·0,4·(1-0,18/100%)= 13251364грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.17)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 6147540/(1+0,15)^1 + 9972676/(1+0,15)^2 + 13251364/(1+0,15)^3 = 21599449,69 \text{грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.18}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 608654,30 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 608654,30 = 1217308,59$грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.19}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 21599449,69грн;

PV – теперішня вартість початкових інвестицій, 1217308,59 грн.

$E_{абс} = ПП - PV = 21599449,69 - 1217308,59 = 20382141,10$грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 20382141,10грн;

PV – теперішня вартість початкових інвестицій, 1217308,59грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+ 20382141,10/ 1217308,59)^{1/3} = 1,61.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,18.

$\tau_{мін}$ = 0,1+0,18 = 0,28 < 1,61 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (5.22)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,61 = 0,62 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів» становить 38 балів, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 0,62 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів».

ВИСНОВКИ

У магістерській кваліфікаційній роботі було розроблено методи та програмні засоби для підвищення ефективності вивчення правильної вимови слів.

Виконано аналіз сучасних методів для вивчення правильної вимови слів. Розглянуто відомі програмні рішення виявлено їхні переваги та недоліки, а також виконано порівняння із власним додатком. Результати порівняння вказали на те, що розробка власного додатку є доцільною для вирішення наявних проблем. Проведено постановку задачі.

Уперше запропоновано метод динамічної генерації персоналізованої підказки для студента при вивченні правильної вимови слів, який за допомогою штучного інтелекту враховує та аналізує попередні помилки студента, що дозволить підвищити ефективність вивчення правильної вимови саме тих слів, які були складними для вивчення.

Подальшого розвитку отримав метод гейміфікації вивчення правильної вимови слів, який, на відміну від відомих, формує для вивчення список тих слів, які становили найбільшу складність при вивченні, таким чином враховується прогрес студента, що дає можливість підвищити зацікавленість студентів та ефективність вивчення правильної вимови слів.

Для створених методів розроблено блок-схеми алгоритмів для покращення візуалізації та розуміння процесів. Виконано розробку структури графічного інтерфейсу з детальним описом ключових вікон додатка.

Проведено розробку програмних засобів для підвищення ефективності вивчення правильної вимови слів. Виконано детальне проєктування програмного забезпечення, використовуючи діаграми UML для кращого розуміння головних функцій програми. Крім того, обрано шаблони проєктування MVC та GOF, яких варто дотримуватись під час розробки додатка для організації та ефективної розробки програмного забезпечення. Виконано варіантний аналіз і обґрунтування вибору мови програмування. Вирішено, що для реалізації

розроблених методів варто застосувати мову Java. Обрано середовище розробки Intelij IDEA, оскільки доступний перелік функцій спрощує розробку, а підтримка інтеграції з системою контролю версій Git допоможе надійно зберігати написаний код. Розроблено графічний інтерфейс користувача за допомогою фреймворку Bootstrap та Thymeleaf. Виконано розробку головних програмних компонент, використовуючи інтеграцію сервісів Google Bard для генерації персоналізованої підказки студента та Google API Speech-to-text. Для управління проектом використано інструмент збірки автоматизації Maven. Створено SQL скрипт для заповнення даними середовище.

Проаналізовано методи тестування ПЗ та, використовуючи комбінацію методів, виконано тестування розробленого програмного застосунку. Створено автоматизовані тести для перевірки роботи програми. Проаналізовано ефективність розроблених методів та визначено вимоги до персонального комп’ютера.

Отримані в магістерській кваліфікаційній роботі наукові та практичні результати можна використати для вивчення правильної вимови слів або покращення власної вимови.

Результати роботи опубліковані в 5 наукових публікаціях.

Згідно з проведеними дослідженнями рівень комерційного потенціалу розробки становить 38 балів, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього). Це доводить, що програмний продукт є перспективною розробкою.

Задачі магістерської кваліфікаційної роботи виконано в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aviles C. How to Engage the Students Who Need It Most: Gamification [Електронний ресурс] / Chris Aviles // edsurge. – 2014. – Режим доступу до ресурсу: https://www.edsurge.com/news/2014-12-20-how-to-engage-the-students-who-need-it-most-gamification.

2. Kahoot! as a Gamification Tool in Vocational Education: More Positive Attitude, Motivation and Less Anxiety in EFL [Електронний ресурс] // researchgate. – 2021. – Режим доступу до ресурсу: https://www.researchgate.net/publication/353646098_Kahoot_as_a_Gamification_Tool_in_Vocational_Education_More_Positive_Attitude_Motivation_and_Less_Anxiety_in_EFL.

3. Chang J. 54 Gamification Statistics You Must Know: 2023 Market Share Analysis & Data [Електронний ресурс] / Jenny Chang // financesonline. – 2023. – Режим доступу до ресурсу: https://financesonline.com/gamification-statistics/.

4. Кучерявий І. В. Гейміфікація в освіті на прикладі програмної платформи для вивчення мов Duolingo [Електронний ресурс] / І. В. Кучерявий, О. В. Романюк // LII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, м. Вінниця – 2023. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki2023/paper/view/17478/14581.

5. Переваги й недоліки сучасної онлайн-освіти: Погляд здобувачів освіти [Електронний ресурс] / І. В.Кучерявий, О. Б. Залюбівська, Б. В. Ковтун, А. В. Миргородський // LII Науково-технічна конференція Інституту Конфуція. м. Вінниця – 2023. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-inkonf/all-inkonf-2023/paper/view/17132.

6. Кучерявий І. В. Дослідження архітектурного патерну MVC на прикладі мови [Електронний ресурс] / І. В. Кучерявий, Л. Б. Ліщинська // Молодь в науці: дослідження, проблеми, перспективи. м. Вінниця – 2023. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/ mn/mn2023/paper/view/18129.

7. Кучерявий І. В. Аналіз використання технологій штучного інтелекту у вивченні правильної вимови / І. В. Кучерявий, О. В. Романюк. // XVI Міжнародна науково-практична конференція Інформаційні технології і автоматизація., – Одеса. – 2023.

8. Кучерявий І. В., Романюк О. В. Розробка методу створення автоматизованого помічника для виправлення мовленнєвих помилок при вивченні іноземної мови / І. В. Кучерявий, О. В. Романюк // Електронні інформаційні ресурси: створення, використання, доступ : Міжнар. науково-практ. Інтернет-конф., Суми/Вінниця, 28–29 листоп. 2022 р. – Суми/Вінниця, 2022. – С. 152–156.

9. Why is Pronunciation So Difficult to Learn? / Gilakjani A., Ahmadi S., Ahmadi M. // English Language Teaching. – 2011. – Режим доступу до ресурсу: https://doi.org/10.5539/elt.v4n3p74.

10. Talking about English pronunciation and accent in the workplace [Електронний ресурс] // speechactive. – 2018. – Режим доступу до ресурсу: https://www.speechactive.com/raising-the-topic-of-english-pronunciation-in-the-workplace.

11. Laoyan S. Understanding the Pareto principle (The 80/20 rule) [Електронний ресурс] / Sarah Laoyan // Asana. – 2022. – Режим доступу до ресурсу: https://asana.com/resources/pareto-principle-80-20-rule.

12. What Are Tonal Languages? - Rosetta Stone. Rosetta Stone. URL: https://shorturl.at/aegKZ (дата звернення: 11.10.2023).

13. Serrano M. What Are Tonal Languages? [Електронний ресурс] / Marisa Serrano // Rosetta Stone. – 2022. – Режим доступу до ресурсу: https://shorturl.at/aegKZ.

14. Що таке фонема [Електронний ресурс] // Словник іншомовних слів – Режим доступу до ресурсу: https://shorturl.at/fjvGO.

15. Marr B. How Is AI Used In Education — Real World Examples Of Today And A Peek Into The Future [Електронний ресурс] / Bernard Marr. – 2021. – Режим

доступу до ресурсу: https://bernardmarr.com/how-is-ai-used-in-education-real-world-examples-of-today-and-a-peek-into-the-future/.

16. 43 Examples of Artificial Intelligence in Education [Електронний ресурс] – Режим доступу до ресурсу: https://onlinedegrees.sandiego.edu/artificial-intelligence-education/.

17. Artificial intelligence goes to school [Електронний ресурс] // The Week – Режим доступу до ресурсу: https://theweek.com/education/1025698/artificial-intelligence-goes-to-school.

18. Forvo: the pronunciation dictionary. All the words in the world pronounced by native speakers. [Електронний ресурс] // Forvo – Режим доступу до ресурсу: https://forvo.com/.

19. Duolingo [Електронний ресурс] – Режим доступу до ресурсу: https://www.duolingo.com/.

20. Сервіс вивчення мови Duolingo вийшов на IPO. У перший день торгів його оцінка склала близько $5 млрд [Електронний ресурс] // Forbes. – 2021. – Режим доступу до ресурсу: https://forbes.ua/news/servis-vivchennya-movi-duolingo-viyshov-na-ipo-u-pershiy-den-torgiv-yogo-otsinka-sklala-blizko-5-mlrd-29072021-2158.

21. Speech-to-Text: Automatic Speech Recognition [Електронний ресурс] // Google Cloud – Режим доступу до ресурсу: https://cloud.google.com/speech-to-text.

22. Rule-Based System [Електронний ресурс] // Engati. – Режим доступу до ресурсу: https://www.engati.com/glossary/rule-based-system.

23. Bard – Chat-based AI tool from Google, powered by PaLM 2. [Електронний ресурс] // Google – Режим доступу до ресурсу: https://bard.google.com/chat?hl=en-GB.

24. User Interface (UI) Design [Електронний ресурс] // Interaction Design Foundation – Режим доступу до ресурсу: https://www.interaction-design.org/literature/topics/ui-design.

25. Juviler J. What Is GUI? Graphical User Interfaces, Explained [Електронний ресурс] / Jamie Juviler // HubSpot. – 2023. – Режим доступу до ресурсу: https://shorturl.at/jpuJS.

26. MDN Web Docs Glossary: Definitions of Web-related terms [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/MVC.

27. Ambler S. W. Elements of UML 2. 0 Style. Cambridge University Press, 2005.

28. MVC: Model, View, Controller [Електронний ресурс] // Codecademy – Режим доступу до ресурсу: https://www.codecademy.com/article/mvc.

29. What is Unified Modeling Language (UML)?. Ideal Modeling & Diagramming Tool for Agile Team Collaboration. [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/.

30. Chan J. Learn Java in One Day and Learn It Well (Learn Coding Fast) (Volume 4). CreateSpace Independent Publishing Platform, 2016.

31. Greene J., Stellman A. Head First C# (Head First). O'Reilly Media, Inc., 2007. 672 c.

32. David T. Programming Ruby: The pragmatic programmer's guide. Boston : Addison-Wesley, 2001. 564 c.

33. 10 Best Java IDE for Developers [2023] [Електронний ресурс] // InterviewBit – Режим доступу до ресурсу: https://www.interviewbit.com/blog/best-java-ide/.

34. IntelliJ IDEA – the Leading Java and Kotlin IDE [Електронний ресурс] // JetBrains – Режим доступу до ресурсу: https://www.jetbrains.com/idea/.

35. About the Eclipse Foundation [Електронний ресурс] // The Eclipse Foundation – Режим доступу до ресурсу: https://www.eclipse.org/org/.

36. Apache NetBeans Fits the Pieces Together [Електронний ресурс] // Apache NetBeans 20 – Режим доступу до ресурсу: https://netbeans.apache.org/front/main/.

37. Productive Java-based Application Development Oracle JDeveloper [Електронний ресурс] // Oracle – Режим доступу до ресурсу: https://www.oracle.com/application-development/technologies/jdeveloper.html .

38. PostgreSQL [Електронний ресурс] // PostgreSQL. – Режим доступу до ресурсу: https://www.postgresql.org/.

39. pgAdmin - PostgreSQL Tools. [Електронний ресурс] // PgAdmin – Режим доступу до ресурсу: https://www.pgadmin.org/.

40. Thymeleaf [Електронний ресурс] – Режим доступу до ресурсу: https://www.thymeleaf.org/.

41. Welcome to Apache Maven. [Електронний ресурс] // Maven. – Режим доступу до ресурсу: https://maven.apache.org/.

42. Що таке тестування програмного забезпечення? [Електронний ресурс] // QALight – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya/.

43. Black Box Vs White Box Testing [Електронний ресурс] // PractiTest – Режим доступу до ресурсу: https://shorturl.at/oqLOX.

44. Hamilton T. What is Module Testing? Definition, Examples [Електронний ресурс] / Thomas Hamilton. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/module-testing.html.

45. Оцінка ступеня узгодженості думок експертів [Електронний ресурс] // Бібліотека BukLib. – Режим доступу до ресурсу: https://buklib.net/books/32686/.

46. Коефіцієнти конкордації: приклад розрахунку і формула. Що таке коефіцієнт конкордації? [Електронний ресурс] // Урок – Режим доступу до ресурсу: https://yrok.pp.ua/serednya-osvta/10773-koefcyent-konkordacyi-priklad-rozrahunku-formula-scho-take-koefcyent-konkordacyi.html.

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

ДОДАТКИ

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О. Н.
«19» вересня 2023 року

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доц. каф. ПЗ Романюк О.В.
«19» вересня 2023 р.

Виконав:
студент гр.3ПІ-22м Кучерявий І.В.
«19» вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів».

Галузь застосування – вивчення іноземних мов.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ №247 від 18 вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності вивчення правильної вимови слів шляхом запровадження динамічної генерації персоналізованої підказки для студентів, які мають проблеми із вимовою слів та створення гейміфікованого середовища для студентів, що дозволить підвищити їхню залученість в навчанні.

Призначення роботи – розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Aviles C. How to Engage the Students Who Need It Most: Gamification [Електронний ресурс] / Chris Aviles // edsurge. – 2014. – Режим доступу до ресурсу: https://www.edsurge.com/news/2014-12-20-how-to-engage-the-students-who-need-it-most-gamification

2. Chang J. 54 Gamification Statistics You Must Know: 2023 Market Share Analysis & Data [Електронний ресурс] / Jenny Chang // financesonline. – 2023. – Режим доступу до ресурсу: https://financesonline.com/gamification-statistics/

3. Marr B. How Is AI Used In Education — Real World Examples Of Today And A Peek Into The Future [Електронний ресурс] / Bernard Marr. – 2021. – Режим

доступу до ресурсу: https://bernardmarr.com/how-is-ai-used-in-education-real-world-examples-of-today-and-a-peek-into-the-future/

4. MDN Web Docs Glossary: Definitions of Web-related terms [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/MVC

5. Оцінка ступеня узгодженості думок експертів [Електронний ресурс] // Бібліотека BukLib. – Режим доступу до ресурсу: https://buklib.net/books/32686/.

5. Технічні вимоги

Вихідні дані до роботи: кольоровий режим – True Color; максимальний розмір дискретного координатного простору – 4096*4096; шаблони проєктування програмного забезпечення – GOF, MVC; методи автоматизації персоналізованого помічника – Rule-based System; автоматизація збірки проєктів – Maven; система управління базами даних – PGAdmin; інструменти на базі штучного інтелекту – Google Bard, Google API Speech-to-text; мова запитів – SQL; тестування додатка – модульні тести; моделювання програмної системи – UML; вхідні дані – ключ для Google API, скрипт для заповнення навчальним матеріалом середовище; вихідні дані – персоналізований помічник у виправленні помилок вимови слів, слова які було неправильно вимовлено, що використовуються у грі для вивчення правильної вимови слів; середовище розробки – Intelij IDEA; мови програмування – Java, оцінка ефективності – коефіцієнт узгодженості думок експертів.

6. Конструктивні вимоги.

Розроблене програмне забезпечення повинно відповідати естетичним та ергономічним вимогам, бути зручним в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз і порівняльна характеристика сучасних методів для вивчення правильної вимови слів	20.09.2023 – 30.09. 2023
2	Розробка методів і алгоритмів програмного продукту	01.10. 2023 – 10.10. 2023
3	Розробка програмних засобів для підвищення ефективності вивчення правильної вимови слів	11.10. 2023 – 25.10. 2023
4	Аналіз ефективності запропонованих методів та тестування програмного забезпечення	26.10.2023 – 20.11.2023
5	Економічна частина	21.11.2023 – 01.12.2023

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Розробка методів і програмних засобів для підвищення ефективності вивчення правильної вимови слів

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ЗПІ – 22м

Науковий керівник: Романюк О.В.

Unicheck	
Оригінальність	**91,8%**
Схожість	8,2%

Аналіз звіту подібності

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 автора. Роботу направити на доопрацювання.

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.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Кучерявий І.В.

Керівник роботи _____ Романюк О.В.

ДОДАТОК В

Лістинг коду

GameController.java

```
package ihorko.work.speech_recognition.controller;

import ihorko.work.speech_recognition.service.GameService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class GameController {

    @Autowired
    private GameService gameService;

    @GetMapping("/game")
    public String showSoundContentPage(Model model) {
        model.addAttribute("gameWord", gameService.getRandomWord());
        return "game/gamePage";
    }

    @GetMapping("/get-word")
    public ResponseEntity<String> handleFileUpload() {
        return ResponseEntity.ok()
                .body(gameService.getRandomWord());
    }

}
```

RecognizeController.java

```
package ihorko.work.speech_recognition.controller;

import com.google.gson.Gson;
import ihorko.work.speech_recognition.common.Language;
import ihorko.work.speech_recognition.common.RecognitionResult;
import ihorko.work.speech_recognition.db.entity.SoundContent;
import ihorko.work.speech_recognition.service.*;
import lombok.SneakyThrows;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ClassPathResource;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.util.UUID;

@Controller
```

```
public class RecognizeController {

    private final AudioRecognitionService audioRecognitionService;

    private final StringService stringService;

    private final SoundContentService soundContentService;

    private final GoogleBardService googleBardService;

    private final GameService gameService;

    private int wrongAnswerCounter = 0;

    private static final Gson gson = new Gson();

    @Autowired
    public RecognizeController(AudioRecognitionService audioRecognitionService,
StringService stringService, SoundContentService soundContentService,
                               GoogleBardService googleBardService, GameService
gameService) {
        this.audioRecognitionService = audioRecognitionService;
        this.stringService = stringService;
        this.soundContentService = soundContentService;
        this.googleBardService = googleBardService;
        this.gameService = gameService;
    }

    @PostMapping(value = "/recognize-from-content", produces =
MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<String> recognizeFromContent(@RequestParam("file")
MultipartFile file, @RequestParam("soundContentId") String soundContentId) {

        File savedFile = saveFileIntoResources(file);
        SoundContent soundContent =
soundContentService.findById(UUID.fromString(soundContentId));

        Language language =
Language.valueOf(soundContent.getSound().getLanguage().toUpperCase());
        RecognitionResult recognitionResult =
makeRecognitionAndSaveToResults(savedFile.getPath(),
                language, soundContent.getContentText());

        if (recognitionResult == null) {
            return ResponseEntity.badRequest().body(gson.toJson(""));
        }

        if (!recognitionResult.getWrongText().isEmpty()) {
            gameService.addNewWord(recognitionResult.getWrongText());
            wrongAnswerCounter++;
        }

        int numberMistakes = 2;
        if (wrongAnswerCounter == numberMistakes) {
            recognitionResult.setUserAssistantText(googleBardService
                    .buildQueryAboutPronunciationAndAskBard(
                            recognitionResult.getWrongText(), language));
            wrongAnswerCounter = 0;
        }

        return ResponseEntity.ok().body(gson.toJson(recognitionResult));
    }

    @PostMapping(value = "/recognize-from-text", produces =
```

```
MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<String> recognizeFromText(@RequestParam("file")
MultipartFile file, @RequestParam("textToRecognize") String textContent,
@RequestParam("language") String language) {

        File savedFile = saveFileIntoResources(file);

        RecognitionResult recognitionResult =
makeRecognitionAndSaveToResults(savedFile.getPath(),
Language.valueOf(language.toUpperCase()), textContent);

        if (recognitionResult == null) {
            return ResponseEntity.badRequest().body(gson.toJson(""));
        }

        return ResponseEntity.ok().body(gson.toJson(recognitionResult));
    }

    @PostMapping(value = "/recognize-for-game", produces =
MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<String> recognizeForGame(@RequestParam("file")
MultipartFile file, @RequestParam("contentText") String textContent) {

        File savedFile = saveFileIntoResources(file);

        RecognitionResult recognitionResult =
makeRecognitionAndSaveToResults(savedFile.getPath(),
                Language.ENGLISH, textContent);

        if (recognitionResult == null) {
            return ResponseEntity.badRequest().body(gson.toJson(""));
        }

        return ResponseEntity.ok().body(gson.toJson(recognitionResult));
    }

    @SneakyThrows
    private File saveFileIntoResources(MultipartFile file) {
        File fileDestination = new
ClassPathResource("/python/recorderDestination.wav").getFile();

        try (OutputStream os = new FileOutputStream(fileDestination)) {
            os.write(file.getBytes());
        }
        return fileDestination;
    }


    private RecognitionResult makeRecognitionAndSaveToResults(String filePath,
Language language, String textToRecognize) {
        String recognizedAudioRecord =
audioRecognitionService.recognizeAudioRecord(filePath, language);
        if (recognizedAudioRecord == null) return null;

        return
stringService.findCorrectAndWrongPartInExpectedText(textToRecognize,
recognizedAudioRecord);
    }
}
```

GameService.java

```
package ihorko.work.speech_recognition.service;

import org.springframework.stereotype.Service;

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

@Service
public class GameService {

    private final List<String> listElements = new ArrayList<>(List.of("apple",
"house", "smoke", "jungle", "lost", "window",
            "bow", "probably", "meal", "suddenly", "thick", "represent",
            "lesson", "bring", "quietly", "correct", "throat", "interior",
            "spite", "charge", "refer", "personal", "dust", "pride"));

    private final Random random = new Random();

    public String getRandomWord() {
        return listElements.get(random.nextInt(listElements.size()));
    }

    public void addNewWord(String word) {
        listElements.add(word);
    }
}
```

GoogleBardService.java

```
package ihorko.work.speech_recognition.service;

import com.pkslow.ai.GoogleBardClient;
import com.pkslow.ai.domain.Answer;
import ihorko.work.speech_recognition.common.Language;
import org.springframework.stereotype.Service;

@Service
public class GoogleBardService {

    public String getAnswerFromBard(String text) {
        //__Secure-1PSID;__Secure-1PSIDTS
        GoogleBardClient client = new GoogleBardClient("TOKEN_API");
        Answer ask = client.ask(text);
        return ask.getChosenAnswer();
    }

    public String buildQueryAboutPronunciationAndAskBard(String text, Language
language) {
        String query;
        if (language == Language.ENGLISH) {
            query = String.format("I have a problem with the pronunciation of
words %s." +
                    " Please teach me how to pronounce" +
                    " that within one paragraph and don't use more than 100
words", text);
        } else {
            query = String.format("У мене проблема з вимовою слів %s. Будь
ласка, навчіть мене вимовляти це в одному абзаці та не використовуйте більше 100
слів", text);
        }
        return getAnswerFromBard(query);
    }
}
```

GamePage.html

```
<body>
<div th:insert="fragments/navbar :: navbar">
</div>
<section class="p-4">
    <div class="container">
        <div class="card-body p-md-5">
            <div class="row">
                <div class="col-4 p-2 ">
                    <figure class="figure px-2">
                        <img alt="Image does not exist" class="figure-img img-
fluid rounded"
                             height="46"
                             id="heart1" th:src="@{image/heart.png}" width="54">
                    </figure>
                    <figure class="figure px-2">
                        <img alt="Image does not exist" class="figure-img img-
fluid rounded"
                             height="46"
                             id="heart2" th:src="@{image/heart.png}" width="54">
                    </figure>
                    <figure class="figure px-2">
                        <img alt="Image does not exist" class="figure-img img-
fluid rounded"
                             height="46"
                             id="heart3" th:src="@{image/heart.png}" width="54">
                    </figure>
                </div>
                <div class="col-7 px-5">
                    <h1 class="display-4 text-uppercase text-lg">
                        Score <strong class="text-success" id="score">0</strong>
                    </h1>
                </div>
            </div>
            <div class="row">
                <div class="col">
                    <div class="row p-2">
                        <div class="col-5 pt-4 ">
                            <figure class="figure px-2">
                                <img alt="Image does not exist" class="figure-
img img-fluid rounded"
                                     height="285" id="knight"
                                     th:src="@{image/knight-stay.gif}"
width="470" }>
                            </figure>
                        </div>
                        <div class="col-2 pt-5">
                            <h1 id="speech_target" th:text="${gameWord}"></h1>
                        </div>
                        <div class="col-5">
                            <figure class="figure px-2">
                                <img alt="Image does not exist" class="figure-
img img-fluid rounded"
                                     height="720" id="monster"
                                     th:src="@{image/monster-stay.gif}"
width="720" }>
                            </figure>
                        </div>
                    </div>

                    <div class="row p-2">
```

```
                        <div class="col text-center">
                            <div data-role="controls">
                                <button class="btn btn-success btn-lg"
id="audio-recorder">
                                    Start recording
                                </button>
                            </div>
                        </div>
                    </div>
                    <div class="row p-2">
                        <div class="col text-center">
                            <div data-role="recordings">
                                <button class="btn btn-success btn-lg"
id="upload-button" onclick="uploadFile()">
                                    Verify record
                                </button>
                            </div>
                        </div>
                    </div>
                    <div class="row align-items-center p-2">
                        <div class="col-3 ">
                            <h4 class="mb-0 text-success" id="success-label">
                            </h4>
                        </div>
                        <div class="col-9">
                            <p class="lead" id="recognized-success-result"></p>
                        </div>
                    </div>
                    <div class="row align-items-center p-2">
                        <div class="col-3 ">
                            <h4 class="mb-0 text-danger" id="failed-label">
                            </h4>
                        </div>
                        <div class="col-9">
                            <p class="lead" id="recognized-failed-result"></p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
<div th:insert="fragments/footer :: footer"></div>
<div class="modal" id="exampleModal" role="dialog">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <div class=" col text-center">
                    <h5 class="modal-title">
                        Game Over <span id="finish-result">0</span>
                    </h5>
                </div>
            </div>
            <div class="modal-body">
                <div class=" col text-center">
                    <p>
                        You are a powerful knight go on.
                    </p>
                </div>
            </div>
            <div class="modal-footer">
                <div class="col text-center">
                    <a class="btn btn-primary" th:href="@{/game}" type="button">
                        Restart game
```

```
                    </a>
                </div>
            </div>
        </div>
    </div>
</div>
</body>

<script src="https://markjivko.com/dist/recorder.js"></script>
<script>

    let savedAudio;
    let countFailed = 0;
    let countScore = 0;
    let successResult = document.getElementById("recognized-success-result");
    let successLabel = document.getElementById("success-label");

    /**
     * @param data.wrongText -  text that our user said wrong
     *  @param data.correctText - text that matching with our content
     *  @param data.fullText - full recognized text
     * @returns {Promise<void>}
     */
    async function uploadFile() {
        let formData = new FormData();

        formData.append('file', savedAudio, 'testRecorder.wav');
        var contentText = document.getElementById("speech_target")
        formData.append("contentText", contentText.innerText);

        await fetch('/recognize-for-game', {
            method: "POST",
            body: formData
        }).then(response =>
            response.json().then(data => ({
                    data: data,
                    status: response.status
                })
            ).then(result => {
                      if (!response.ok) {
                        successResult.textContent = 'The text was not recognized
try again';
                        return;
                    }
                    let knightImage = document.getElementById("knight");

                    if (result.data.correctText.length > 0) {
                        successResult = document.getElementById("recognized-
success-result");
                        successLabel = document.getElementById("success-label");

                        successResult.textContent = result.data.correctText;
                        successLabel.textContent = "Correctly pronounced text";
                        knightImage.src = "/image/knight-attack.gif"
                        setTimeout(function () {
                            knightImage.src = "/image/knight-stay.gif"
                        }, 2500);
                        countScore = countScore + 5;
                        let score = document.getElementById("score");
                        score.textContent = countScore;
                    } else {
                        let failedResult = document.getElementById("recognized-
failed-result");
                        let failedLabel = document.getElementById("failed-
```

```
label");
                        let monsterImage = document.getElementById("monster");

                        failedResult.textContent = result.data.wrongText;
                        failedLabel.textContent = "Incorrectly pronounced text";
                        monsterImage.src = "/image/monster-attack.gif"
                        setTimeout(function () {
                            monsterImage.src = "/image/monster-stay.gif"
                        }, 2500);
                        countFailed++;
                        let heart =
document.getElementById("heart".concat(countFailed));
                        heart.parentNode.removeChild(heart);
                    }

                    if (countFailed === 3) {
                        knightImage.src = "/image/knight-death1.gif";
                        document.getElementById("finish-result").textContent =
countScore;
                        $('#exampleModal').modal('toggle');
                    }
                    fetch('get-word', {
                        method: "GET"
                    }).then(response => response.text())
                        .then(response => {
                            let textForGamer =
document.getElementById("speech_target");
                            textForGamer.textContent = response;
                        })
                }
            ));
    }

    jQuery(document).ready(function () {
        var $ = jQuery;
        var myRecorder = {
            objects: {
                context: null,
                stream: null,
                recorder: null
            },
            init: function () {
                if (null === myRecorder.objects.context) {
                    myRecorder.objects.context = new (
                        window.AudioContext || window.webkitAudioContext
                    );
                }
            },
            start: function () {

                var options = {audio: true, video: false};
                navigator.mediaDevices.getUserMedia(options).then(function
(stream) {
                    myRecorder.objects.stream = stream;
                    myRecorder.objects.recorder = new Recorder(

myRecorder.objects.context.createMediaStreamSource(stream),
                        {numChannels: 1}
                    );
                    myRecorder.objects.recorder.record();
                }).catch(function (err) {
                });
            },
            stop: function (listObject) {
```

```
                if (null !== myRecorder.objects.stream) {
                    myRecorder.objects.stream.getAudioTracks()[0].stop();
                }
                if (null !== myRecorder.objects.recorder) {
                    myRecorder.objects.recorder.stop();

                    // Validate object
                    if (null !== listObject
                        && 'object' === typeof listObject
                        && listObject.length > 0) {
                        // Export the WAV file
                        myRecorder.objects.recorder.exportWAV(function (blob) {
                            var url = (window.URL || window.webkitURL)
                                .createObjectURL(blob);
                            savedAudio = blob

                            // Prepare the playback
                            var audioObject = $('<audio controls></audio>')
                                .attr('src', url);

                            // Prepare the download link
                            var downloadObject = $('<a>&#9660;</a>')
                                .attr('href', url)
                                .attr('download', new Date().toUTCString() +
'.wav');

                            // Wrap everything in a row
                            var holderObject = $('<div class="row p-2"></div>')
                                .append(audioObject)
                                .append(downloadObject);

                            // Append to the list
                            listObject.append(holderObject);
                        });
                    }
                }
            }
        };
        // Prepare the recordings list
        var listObject = $('[data-role="recordings"]');

        // Prepare the record button
        $('[data-role="controls"] > button').click(function () {
            // Initialize the recorder
            myRecorder.init();

            // Get the button state
            var buttonState = !!$(this).attr('data-recording');

            // Toggle
            if (!buttonState) {
                $(this).attr('data-recording', 'true');
                $(this).removeClass('btn-success');
                $(this).addClass('btn-danger')
                $(this).text('Stop recording')
                myRecorder.start();
            } else {
                $(this).attr('data-recording', '');
                $(this).removeClass('btn-danger');
                $(this).addClass('btn-success')
                $(this).text('Start recording')
                myRecorder.stop(listObject);
            }
        });
```

```
    });

</script>


async function uploadFile() {
    let formData = new FormData();

    formData.append('file', savedAudio, 'testRecorder.wav');
    const soundContentId = window.location.pathname.split("/").pop()

    formData.append("soundContentId", soundContentId);

    await fetch('/recognize-from-content', {
        method: "POST",
        body: formData
    }).then(response => response.json()
        .then(data => ({
                data: data,
                status: response.status
            })
        ).then(result => {
                if (!response.ok) {
                    successResult.textContent = 'The text was not recognized try
again';
                    return;
                }
                if(result.data.userAssistantText.length > 0){
                alert(result.data.userAssistantText);
                }

                if (result.data.correctText.length > 0) {
                    successResult.textContent = result.data.correctText;
                    successLabel.style.visibility = "visible";
                } else {
                    successResult.textContent = "";
                    successLabel.style.visibility = "hidden";
                }

                let failedResult = document.getElementById("recognized-failed-
result");
                let failedLabel = document.getElementById("failed-label");
                if (result.data.wrongText.length > 0) {
                    failedResult.textContent = result.data.wrongText;
                    failedLabel.style.visibility = "visible";
                } else {
                    failedResult.textContent = "";
                    failedLabel.style.visibility = "hidden";
                }

                let fullResult = document.getElementById("recognized-full-
result");
                let fullLabel = document.getElementById("full-label");
                if (result.data.fullText.length > 0) {
                    fullResult.textContent = result.data.fullText;
                    fullLabel.style.visibility = "visible";
                } else {
                    fullResult.textContent = "";
                    fullLabel.style.visibility = "hidden";
                }

            }
        ))
}
```

/fragments/navbar.html

```
<nav class=" navbar navbar-expand-lg navbar-light bg-light navbar-default"
th:fragment="navbar">
    <a class="navbar-brand px-2" th:href="@{/language}">
        <img class="figure-img img-fluid rounded"
             th:src="@{/image/nav-icon.png}"
             alt="Logo" width="40" height="40"/>
        <span>
            Speech learning
        </span>
    </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item">
                <a class="nav-link" th:href="@{/sounds/list}"
th:text="#{sound_list}">
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link" th:href="@{/sound-contents/list}"
th:text="#{sound_content_list}">
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link" th:href="@{/own-words}"
th:text="#{learn_your_own_words}">
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link" th:href="@{/game}" th:text="#{game}">
                </a>
            </li>
        </ul>

    </div>

    <a class="nav-link" th:href="@{/language?lang=en}">
        <img type="image" th:src="@{/image/england-flag.jpg}"
             alt="English language" width="32" height="18"/>
    </a>
    <a class="nav-link" th:href="@{/language?lang=ua}">
        <img type="image" th:src="@{/image/ukraine-flag.jpg}"
             alt="Ukrainian language" width="32" height="18"/>
    </a>
</nav>
```

GoogleBardServiceTest.java

```
package ihorko.work.speech_recognition.service;

import ihorko.work.speech_recognition.common.Language;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.boot.test.context.SpringBootTest;

import java.util.logging.Logger;

@SpringBootTest
class GoogleBardServiceTest {

    private static final Logger LOGGER =
Logger.getLogger(GoogleBardServiceTest.class.getName());

    @Autowired
    GoogleBardService googleBardService;

    @Test
    void testSimpleAnswerFromGoogleBardService() {
        String answerFromBard = googleBardService.getAnswerFromBard("What is it
today?");
        LOGGER.info("Answer from BARD: " + answerFromBard);
        Assertions.assertFalse(answerFromBard.isEmpty(), " Answer from BARD
shouldn't be empty");
    }


    @Test
    void testEnglishQueryFromGoogleBardService() {
        String wordForTesting = "schedule";
        String answerFromBard =
googleBardService.buildQueryAboutPronunciationAndAskBard(wordForTesting,
Language.ENGLISH);
        LOGGER.info("Answer from BARD: " + answerFromBard);
        Assertions.assertTrue(answerFromBard.contains(wordForTesting), " Answer
from BARD should contains word schedule");
    }

    @Test
    void testUkraineQueryFromGoogleBardService() {
        String wordForTesting = "Графік";
        String answerFromBard =
googleBardService.buildQueryAboutPronunciationAndAskBard(wordForTesting,
Language.UKRAINIAN);
        LOGGER.info("Answer from BARD: " + answerFromBard);

        Assertions.assertTrue(answerFromBard.contains(wordForTesting), " Answer
from BARD should contains word %s".formatted(wordForTesting));
    }
}
```

SoundRepositoryTests.java

```
package ihorko.work.speech_recognition.db;

import ihorko.work.speech_recognition.db.entity.Sound;
import ihorko.work.speech_recognition.repository.SoundRepository;
import org.junit.jupiter.api.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.dao.EmptyResultDataAccessException;

@SpringBootTest
@TestMethodOrder(MethodOrderer.MethodName.class)
class SoundRepositoryTests {

    @Autowired
```

```
    private SoundRepository soundRepository;
    private static final String TEST_NAME = "TestName1234";

    @Test
    @Order(1)
    void testCreateSound() {
        Sound sound = new Sound();
        sound.setName(TEST_NAME);
        sound.setLanguage("English");
        soundRepository.save(sound);
        var createdSound = soundRepository.findByName(TEST_NAME).get(0);
        Assertions.assertNotNull(createdSound);
        Assertions.assertEquals(TEST_NAME, createdSound.getName());
    }

    @Test
    @Order(2)
    void testDeleteSound() {
        Sound createdSound = soundRepository.findByName(TEST_NAME).get(0);
        var soundId = createdSound.getId();
        soundRepository.delete(soundId);
        Assertions.assertThrows(EmptyResultDataAccessException.class,
                () -> soundRepository.findById(soundId));
    }
}
```

messages.properties

```
greeting=Hello! Welcome to the website for learning sounds pronunciation!
lang.change=Change the language
content_for_learning_english=Content for learning English pronunciation.
content_for_learning_ukrainian=Content for learning Ukrainian pronunciation.
open=Open
sound_list=Sounds list
sound_content_list=Sounds content list
learn_your_own_words=Learn your own words
game=Play game
sound_creation_form=Sound creation form
sound_name=Sound name
language=Language
english=English
ukrainian=Ukrainian
submit=Submit
no_sounds_yet=No sounds yet!
sounds=Sounds
displayed_sounds_you_can_choose_and_learn=There are displayed sounds. You can
choose one and start to learn pronunciation!
add_sound=Add a new sound
success=Success!
operation_performed_successfully=The operation was performed successfully.
failure=Failure!
operation_failed=Operation failed. Please try again.
material_for_study=Material for study
displayed_sound_content=Materials for learning pronunciation are displayed here.
Just choose one and start learning!
content_text=Content text
type_content=Type content
audio=Audio
link_on_content=Link on content
add_new_sound_content=Add a new sound content
sound_content_creation_form=Sound content creation form
select_sound=Select sound
upload_audio_file_with_pronunciation=Upload audio file with pronunciation
upload_your_audio_file_max_size_10_mb=Upload your audio file. Max file size 10
```

```
MB
upload_image_or_gif_with_pronunciation=Upload image or gif with pronunciation
upload_your_image_file_max_size_10_mb=Upload your image file. Max file size 10
MB
learn_how_pronounce=Learn how to pronounce
listen_the_audio_with_pronunciation=Listen the audio with pronunciation
content_for_pronunciation=Content for pronunciation
image_how_to_pronounce=Image how to pronounce
start_recording=Start recording
verify_record=Verify record
correctly_pronounced_text=Correctly pronounced text
incorrectly_pronounced_text=Incorrectly pronounced text
full_recognized_text=Full recognized text
text_to_pronounce=Text to pronounce
here_you_can_learn_own_words_just_type_into_text_box=Here you can learn own
words just type into text box and verify if you pronounce it correctly.
```

messages_ua.properties

```
greeting=Привіт! Ласкаво просимо на сайт для вивчення вимови звуків!
lang.change=Змінити мову
content_for_learning_english = Матеріали для вивчення Англійської вимови.
content_for_learning_ukrainian = Матеріали для вивчення Української вимови.
open = Відкрити
sound_list = Список звуків
sound_content_list = Список матеріалів для звуків
learn_your_own_words = Вивчайте власні слова
sound_creation_form = Форма створення звуку
sound_name = Назва звуку
language = Мова
game=Грати в гру
english = Англійська
ukrainian = Українська
submit = Підтвердити
no_sounds_yet=Ще немає звуків!
sounds=Звуки
displayed_sounds_you_can_choose_and_learn=Тут відображаються звуки. Ви можете
вибрати один і почати вивчати його вимову!
add_sound=Додайте новий звук
success=Успіх!
operation_performed_successfully=Операцію виконано успішно.
failure=Невдача!
operation_failed=Операція не вдалася. Будь ласка спробуйте ще раз.
material_for_study=Матеріал для вивчення
displayed_sound_content=Тут представлені матеріали для вивчення вимови. Просто
виберіть один і почніть вчитися!
content_text=Текст контенту
type_content=Тип контенту
audio=Аудіо
link_on_content=Посилання на матеріал
add_new_sound_content=Додати новий контент для звуку
sound_content_creation_form=Форма створення контенту для звуку
select_sound=Обрати звук
upload_audio_file_with_pronunciation=Завантажити аудіо файл з вимовою
upload_your_audio_file_max_size_10_mb=Завантажте ваш аудіо файл. Максимальний
розмір файлу 10 МБ
upload_image_or_gif_with_pronunciation=Завантажити зображення для вимови звуку
upload_your_image_file_max_size_10_mb=Завантажте ваше зображення. Максимальний
розмір файлу 10 МБ
learn_how_pronounce=Вивчайте як вимовляти
listen_the_audio_with_pronunciation=Слухайте аудіо з вимовою
content_for_pronunciation=Матеріал для вимови
image_how_to_pronounce=Картинка як вимовляти
```

```
start_recording=Почати запис
verify_record=Перевірити запис
correctly_pronounced_text=Правильно вимовлений текст
incorrectly_pronounced_text=Неправильно вимовлений текст
full_recognized_text=Повністю розпізнаний текст
text_to_pronounce = Текст для вимови
here_you_can_learn_own_words_just_type_into_text_box = Тут ви можете вивчити
власні слова, просто введіть у текстове поле та перевірте, чи правильно ви їх
вимовляєте.
```

GenerateSounds.sql

```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

INSERT INTO public.sound(
   id, language, name)
   VALUES (uuid_generate_v4(), 'English', 'Ch');

INSERT INTO public.sound(
   id, language, name)
   VALUES (uuid_generate_v4(), 'English', 'A');

INSERT INTO public.sound(
   id, language, name)
   VALUES (uuid_generate_v4(), 'English', 'B');

INSERT INTO public.sound(
   id, language, name)
   VALUES (uuid_generate_v4(), 'English', 'C');

INSERT INTO public.sound(
   id, language, name)
   VALUES (uuid_generate_v4(), 'English', 'Q');

INSERT INTO public.sound(
    id, language, name)
VALUES (uuid_generate_v4(), 'Ukrainian', 'A');

INSERT INTO public.sound(
    id, language, name)
VALUES (uuid_generate_v4(), 'Ukrainian', 'B');

INSERT INTO public.sound(
    id, language, name)
VALUES (uuid_generate_v4(), 'Ukrainian', 'Б');

INSERT INTO public.sound(
    id, language, name)
VALUES (uuid_generate_v4(), 'Ukrainian', 'C');

INSERT INTO public.sound(
    id, language, name)
VALUES (uuid_generate_v4(), 'Ukrainian', 'Д');

INSERT INTO public.sound(
    id, language, name)
VALUES (uuid_generate_v4(), 'Ukrainian', 'Ж');
```

application.properties

```
spring.servlet.multipart.max-file-size=15MB
spring.servlet.multipart.max-request-size=15MB
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.2</version>
        <relativePath/>
    </parent>
    <groupId>ihorko.work</groupId>
    <artifactId>speech_recognition</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>speech_recognition</name>
    <description>Speech recognition web project</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
            <version>2.6.2</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.6.5.Final</version>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.3.3</version>
        </dependency>
        <dependency>
            <groupId>com.google.guava</groupId>
            <artifactId>guava</artifactId>
            <version>31.1-jre</version>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
            <version>5.3.16</version>
        </dependency>
        <dependency>
            <groupId>org.apache.tomcat</groupId>
            <artifactId>tomcat-dbcp</artifactId>
            <version>10.0.14</version>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-collections4</artifactId>
            <version>4.4</version>
        </dependency>
        <dependency>
            <groupId>com.pkslow</groupId>
            <artifactId>google-bard</artifactId>
            <version>0.3.5</version>
        </dependency>
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.9.0</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>${project.parent.version}</version>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>15</source>
                    <target>15</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

DBFileDao.java

```
package ihorko.work.speech_recognition.db.dao;

import ihorko.work.speech_recognition.db.entity.File;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import javax.transaction.Transactional;
```

```
import java.util.UUID;

@Repository
@Transactional
public class DBFileDao {

    private final SessionFactory sessionFactory;

    @Autowired
    public DBFileDao(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    public void persist(File file) {
        sessionFactory.getCurrentSession().saveOrUpdate(file);
    }

    public File findById(UUID uuid) {
        return sessionFactory.getCurrentSession().get(File.class, uuid);
    }
}
```

SoundContentDao.java

```
package ihorko.work.speech_recognition.db.dao;

import ihorko.work.speech_recognition.db.entity.SoundContent;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import javax.persistence.TypedQuery;
import javax.transaction.Transactional;
import java.util.List;
import java.util.UUID;

@Repository
@Transactional
public class SoundContentDao {

    private final SessionFactory sessionFactory;

    @Autowired
    public SoundContentDao(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    public void persist(SoundContent soundContent) {
        sessionFactory.getCurrentSession().saveOrUpdate(soundContent);
    }

    public List<SoundContent> listSoundsContent() {
        TypedQuery<SoundContent> query = sessionFactory.getCurrentSession()
                .createQuery("From sound_content", SoundContent.class);
        return query.getResultList();
    }

    public List<SoundContent> listSoundsContentBySound(UUID sound) {
        TypedQuery<SoundContent> query = sessionFactory.getCurrentSession()
                .createQuery("from sound_content s where s.sound.id= :sound_id",
SoundContent.class);
        query.setParameter("sound_id", sound);
```

```
        return query.getResultList();
    }

    public SoundContent findById(UUID uuid) {
        return sessionFactory.getCurrentSession()
                .get(SoundContent.class, uuid);
    }

    public void delete(UUID uuid) {
        sessionFactory.getCurrentSession().delete(findById(uuid));
    }
}
```

SoundDao.java

```
package ihorko.work.speech_recognition.db.dao;

import ihorko.work.speech_recognition.db.entity.Sound;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import javax.persistence.TypedQuery;
import javax.transaction.Transactional;
import java.util.List;
import java.util.UUID;

@Repository
@Transactional
public class SoundDao {

    private final SessionFactory sessionFactory;

    @Autowired
    public SoundDao(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    public void persist(Sound sound) {
        sessionFactory.getCurrentSession().saveOrUpdate(sound);
    }

    public List<Sound> findByName(String name) {
        TypedQuery<Sound> query = sessionFactory.getCurrentSession()
                .createQuery("From sound s where s.name = :name", Sound.class);
        query.setParameter("name", name);
        return query.getResultList();
    }

    public List<Sound> findByLanguage(String language) {
        TypedQuery<Sound> query = sessionFactory.getCurrentSession()
                .createQuery("From sound s where s.language = :language",
Sound.class);
        query.setParameter("language", language);
        return query.getResultList();
    }

    public Sound findById(UUID id) {
        TypedQuery<Sound> query = sessionFactory.getCurrentSession()
                .createQuery("From sound s where s.id = :id", Sound.class);
        query.setParameter("id", id);
        return query.getSingleResult();
```

```
    }

    public List<Sound> listSounds() {
        TypedQuery<Sound> query =
sessionFactory.getCurrentSession().createQuery("From sound", Sound.class);
        return query.getResultList();
    }

    public void deleteSound(UUID id) {
        sessionFactory.getCurrentSession().delete(findById(id));
    }
}
```

RecognitionResult.java

```
package ihorko.work.speech_recognition.common;


import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.Getter;

@Data
@Getter
@AllArgsConstructor
public class RecognitionResult {
    private String correctText;
    private String wrongText;
    private String fullText;
    private String userAssistantText;

    public RecognitionResult(String correctText, String wrongText, String
fullText) {
        this.correctText = correctText;
        this.wrongText = wrongText;
        this.fullText = fullText;
        this.userAssistantText = "";
    }
}
```

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ВИВЧЕННЯ ПРАВИЛЬНОЇ ВИМОВИ СЛІВ

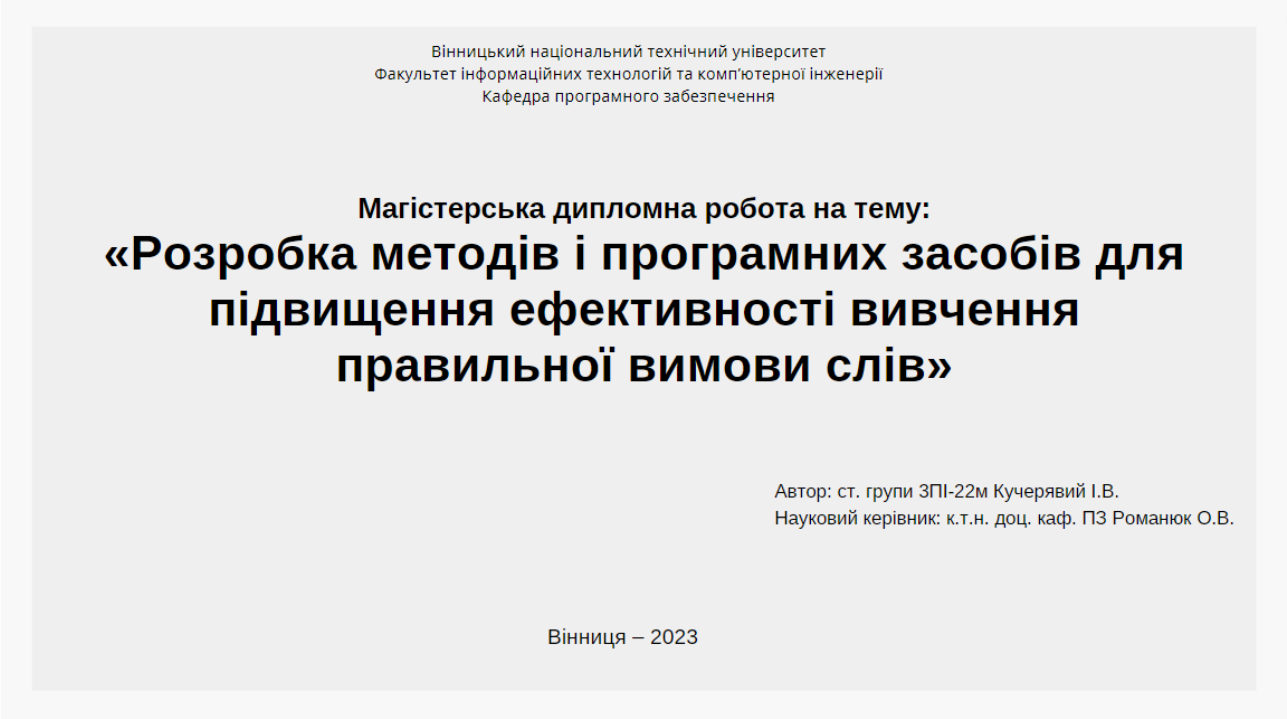


Рисунок Г.1 – Титульний слайд

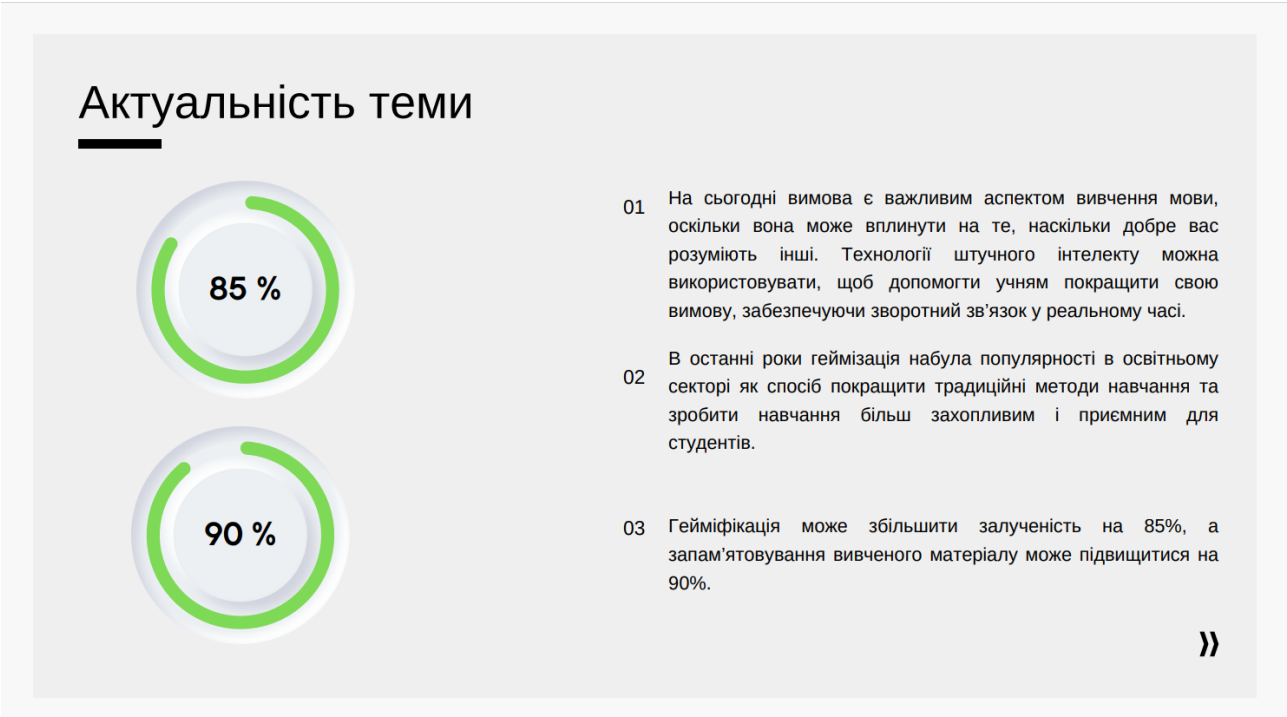


Рисунок Г.2 – Актуальність теми

Рисунок Г.3 – Мета, предмет та об’єкт дослідження

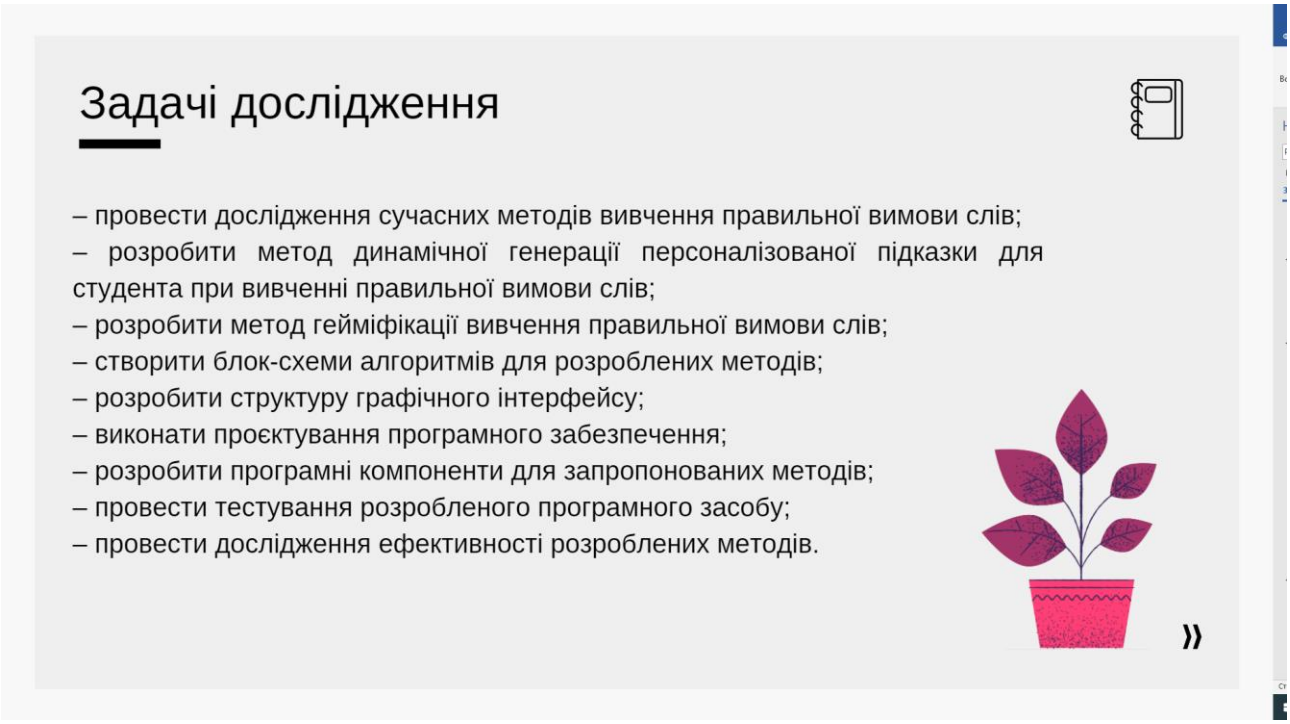


Рисунок Г.4 – Задачі дослідження

Рисунок Г.5 – Новизна і практична цінність отриманих результатів

Критерій	Forvo	Anki	Duolingo	SpeechLearning
Наявність інтерактивного помічника	-	-	+	+
Простота використання	+	-	+	+
Елементи гри в не ігрових контекстах	-	-	+	+
Безкоштовність повного функціоналу	-	+	+	+
Можливість вибору матеріалу для вивчення	+	+	-	+
Багатоплатформність	+	+	-	+
Загальний результат	3	3	4	6

Рисунок Г.6 – Порівняльна характеристика застосунків

Метод та блок-схема алгоритму динамічної генерації персоналізованої підказки для студента

Загальна модель, яка використовується для побудови динамічної генерації персоналізованої підказки може бути представлено у вигляді кортежу множин:

<Rr, Ct, Nt, Pt, Tz , Uat, La, Mdm, Cnc>

Правильно вимовлений текст – дотримується умов завдання, відповідно визначається за формулою:

$$Ct = Rr \cap Pt.$$

Неправильно вимовлений текст – є протилежним правильному тексту, для нього здійснюється умова наступні дві формули:

$$Ct \cap Nt = \emptyset,$$

$$Nt = Ct \setminus Pt$$

Текст запиту для побудови відповіді від помічника напряму залежить від кількості зроблених та мінімальної допущеної кількості помилок, а також він повинен містити в собі усі неправильно вимовлені слова, щоб включити їх в розроблену систему правил.

$$Tz = \{^{Tz = Cnc > Mdm}_{Tz \subseteq Nt}.$$

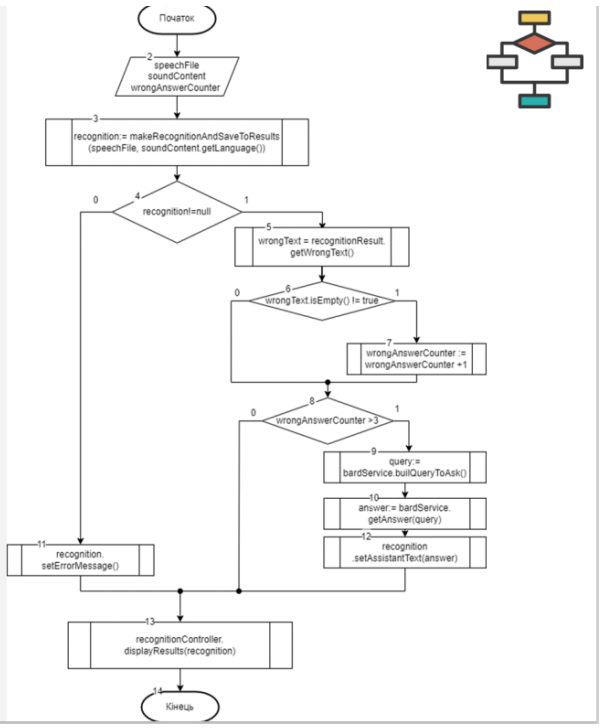


Рисунок Г.7 – Метод та блок-схема алгоритму динамічної генерації персоналізованої підказки для студента

Метод та блок-схема алгоритму гейміфікації вивчення правильної вимови слів

Множина, що дозволить представити необхідну інформацію для взаємодії із студентом

<Sw, Ig, Sc, Li, St, Rt, Cp, Wp>

Sw – слово або вираз, який необхідно вимовити студенту. Дані мають складатись з зроблених помилок студентом, які доступні під час користування навчальним контентом;

Ig – відображення інформації про гру;

Sc – загальний рахунок успішно вимовлених слів студентом, показник повинен зміщуватись не на сталу величину, а залежати від складності самого слова для її обрахунку. Формула для визначення Sc:

$$Sc = Sw.lenght + 1;$$

Li – кількість життів, яку має ігровий персонаж. Величина яка повинна відображати доступну кількість помилок, що може допустити студент в межах однієї сесії гри. Варто зазначити умову за якої сесія гри буде завершена, оскільки це залежить від поточного значення Li:

$$Li > 0;$$

Необхідно відобразити залежність між множинами, щоб враховувати чи вимовлений текст є правильним. Залежність представлена формулою:

$$Rt \cap Sw = Cp.$$

Рисунок Г.8 – Метод та блок-схема алгоритму гейміфікації вивчення правильної вимови слів

Рисунок Г.9 – Структура графічного інтерфейсу

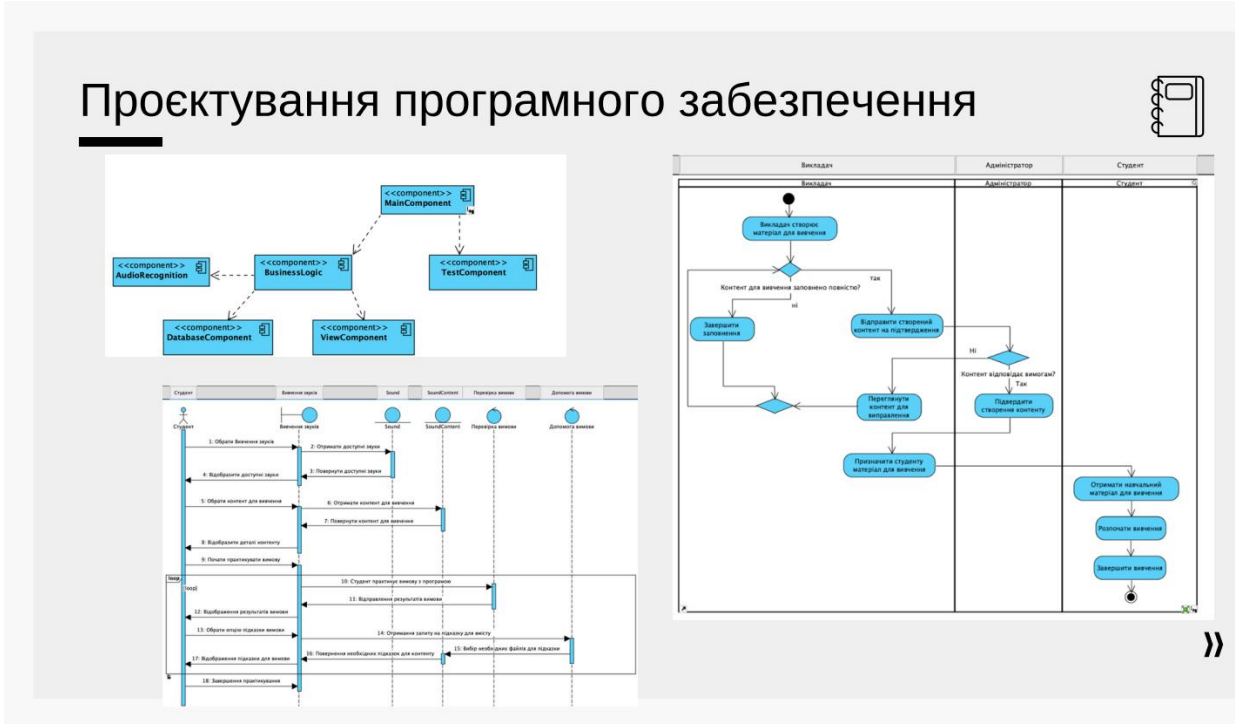


Рисунок Г.10 – Проєктування програмного забезпечення

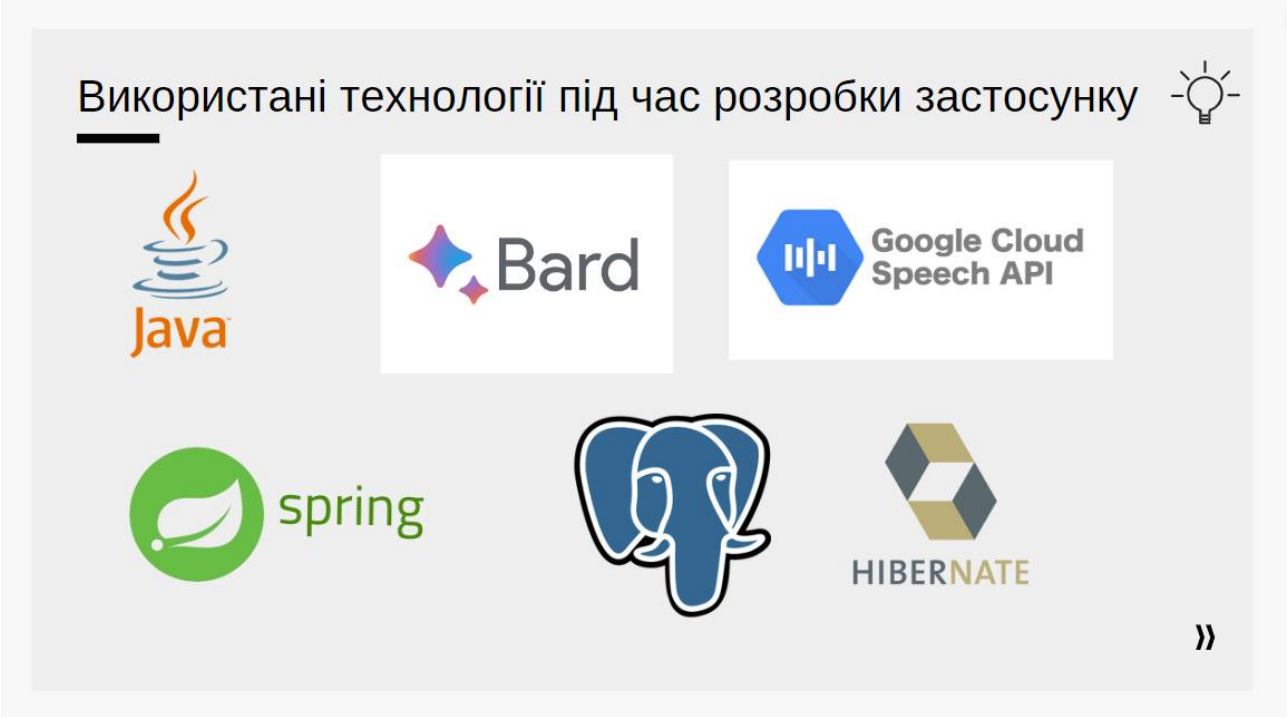


Рисунок Г.11 – Використані технології під час розробки застосунку

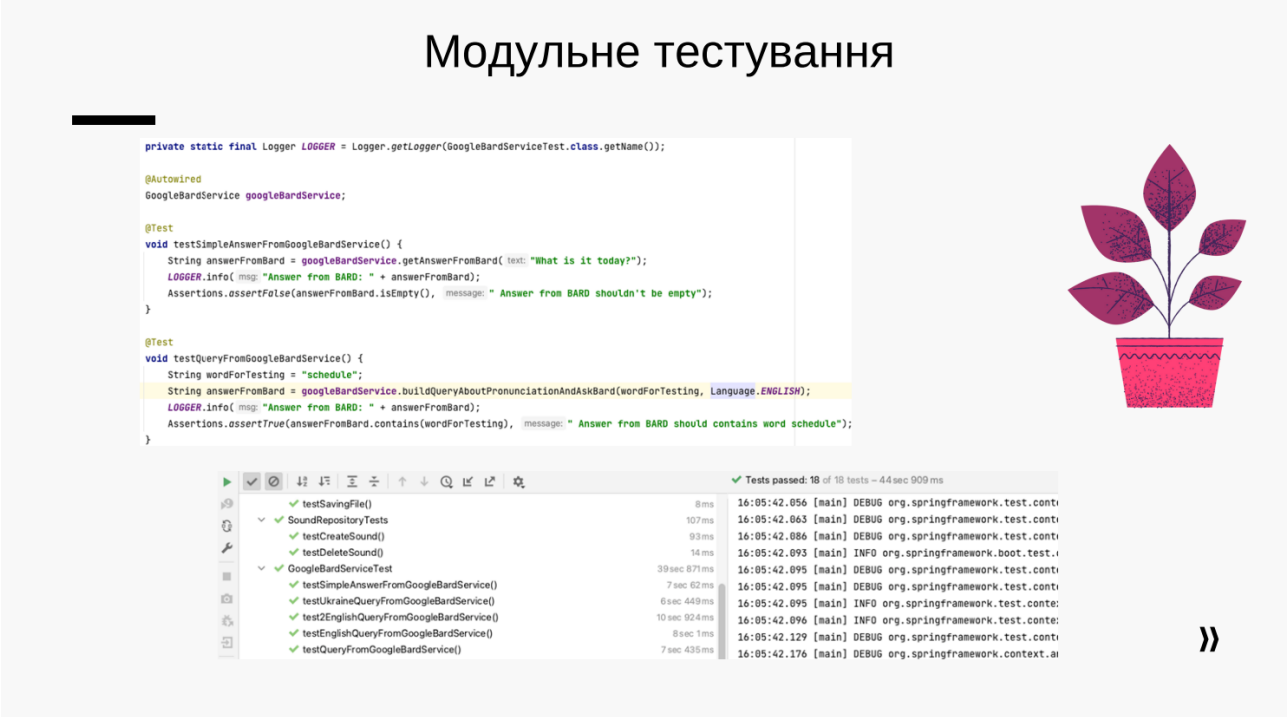


Рисунок Г.12 – Модульне тестування

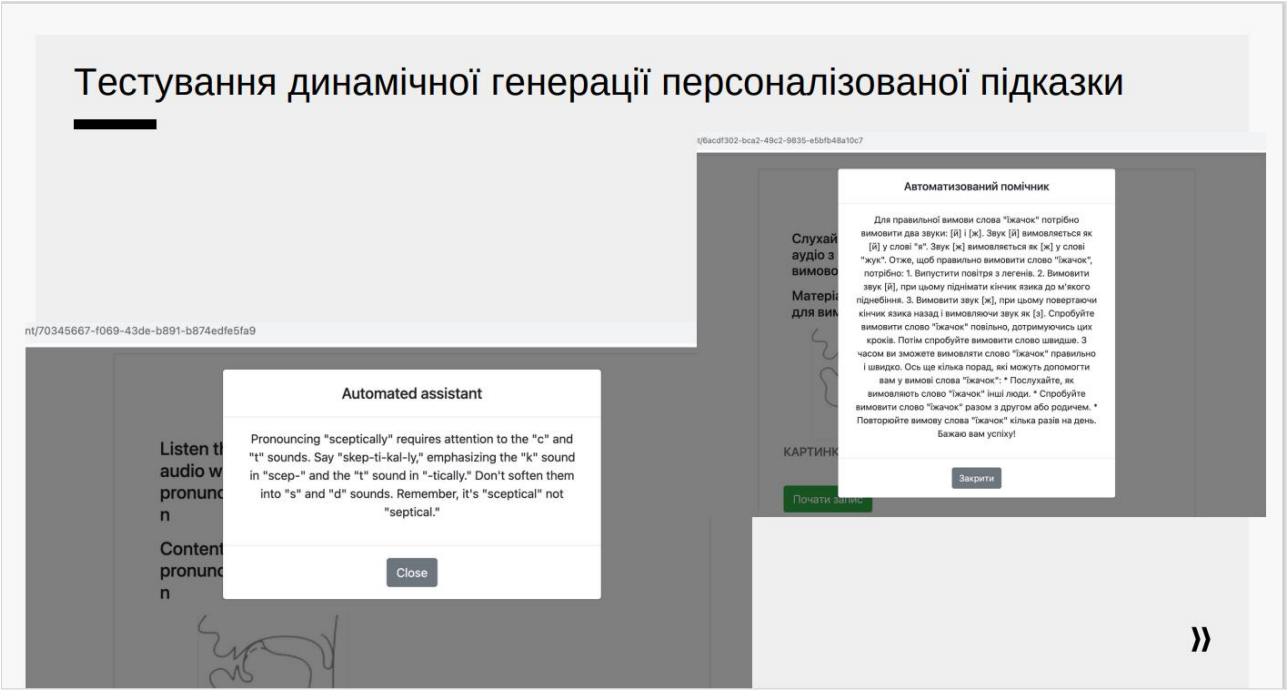


Рисунок Г.13 – Тестування динамічної генерації персоналізованої підказки

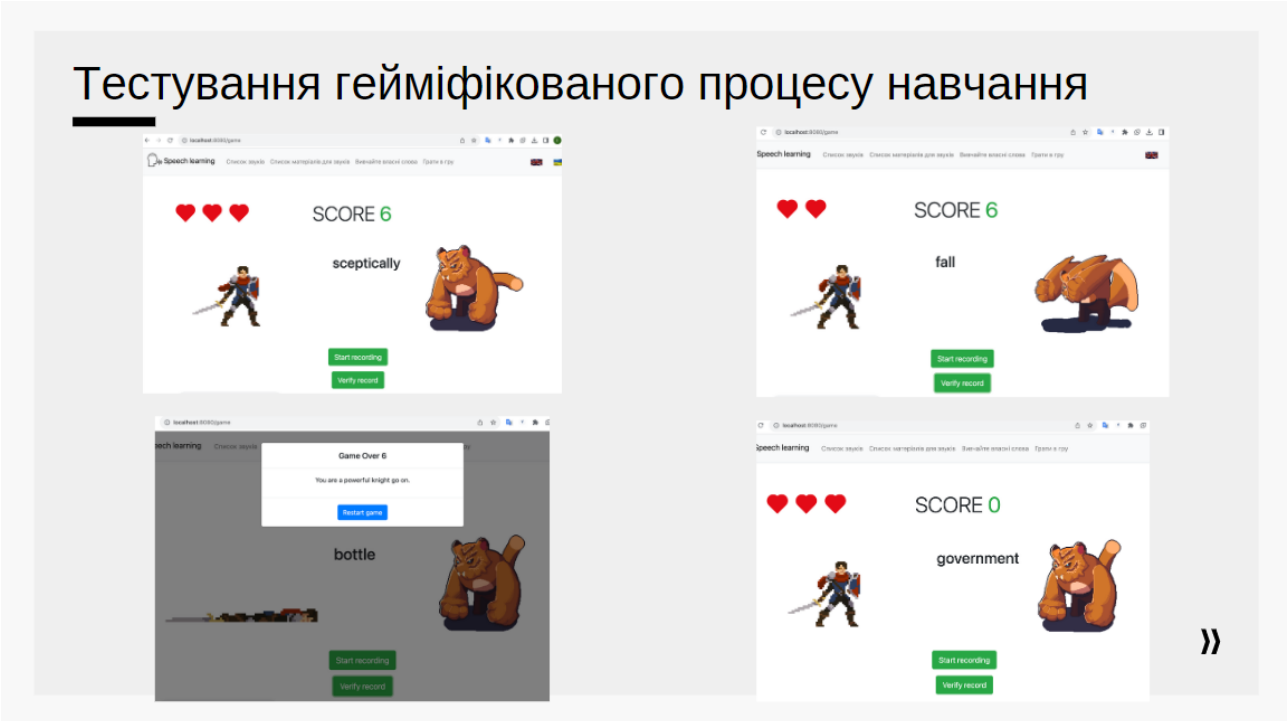
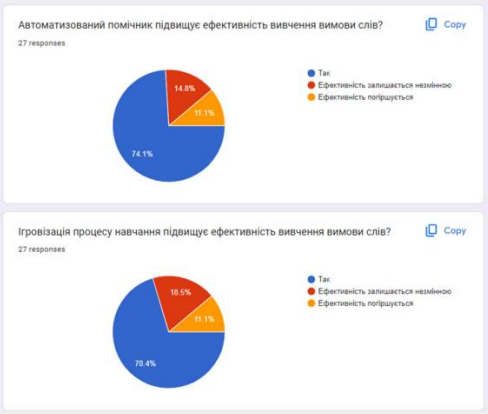


Рисунок Г.14 – Тестування гейміфікованого процесу навчання

Аналіз ефективності розроблених методів



Коефіцієнт конкордації Кендалла (W)

∑S^2 є сумою квадратів рангових сум для кожного питання.

$$W = \frac{12\sum S^2}{k^2(n^3 - n)}$$

k – кількість категорій.

n – кількість оцінювачів або респондентів.

S1=20×3+4×2+3×1=71

S2=19×3+5×2+3×1=70

$$W = \frac{12 * (5041 + 4900)}{9(19656 - 27)} = 0.674$$

»

Рисунок Г.15 – Аналіз ефективності розроблених методів

Економічне обґрунтування

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів та програмних засобів для підвищення ефективності вивчення правильної вимови слів» становить 38 балів, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 0,62 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

»

Рисунок Г.16 – Економічне обґрунтування

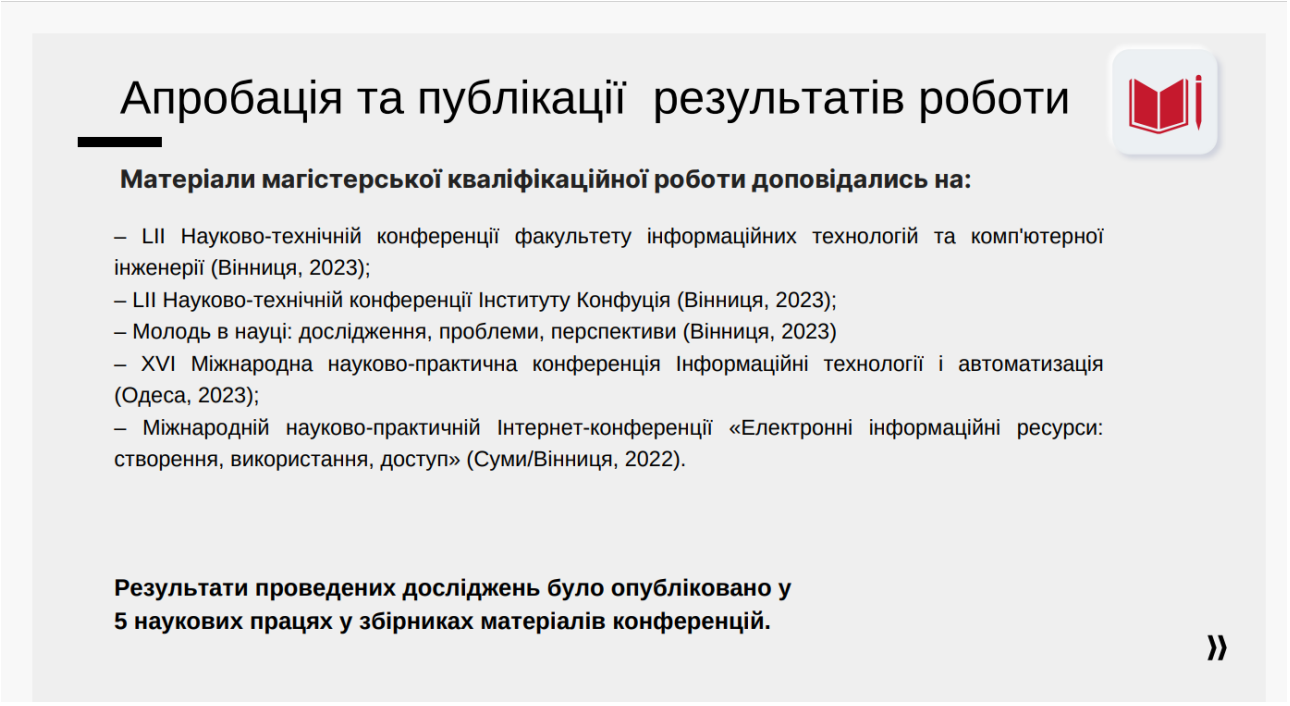


Рисунок Г.17 – Апробація та публікації результатів роботи

Рисунок Г.18 – Фінальний слайд