Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного забезпечення для вивчення правильної вимови звуків»

Виконав: студент IV курсу
групи 3ПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кучерявий І.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Романюк О.В.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. КН Колодний В.Б
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____ О.Н. Романюк
«13» 06 2022 р.

ВНТУ – 2022

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
_____
25 березня 2022 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кучерявому Ігорю Володимировичу

1. Тема роботи – «Розробка програмного забезпечення для вивчення правильної вимови звуків».

Керівник роботи: Романюк Оксана Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 24 березня 2022 р. № 66.

2. Строк подання студентом роботи 13 червня 2022 р.

3. Вихідні дані до роботи: принцип розробки ПЗ – Inversion of Control; патерни проектування програмного забезпечення: Dependency Injection, Model View Controller; база даних – PostgreSQL; контейнер для інверсії контролю – Spring; ORM система – Hibernate; методи розпізнавання вимови – Google API; вхідні дані – база даних, яка містить матеріали для вивчення; аудіо файл з вимовленим текстом користувача; вихідні дані – розпізнаний текст, який вимовив користувач й збережені власноруч навчальні матеріали; середовище розробки – InteliJ IDEA; мова програмування – Java, Python.

4. Зміст розрахунково–пояснювальної записки: вступ; дослідження сфери розпізнавання мовлення та постановка задачі дослідження; розробка структури інтерфейсу та алгоритмів роботи програмного додатку; розробка програмних компонент для веб-інтерактивного додатку; тестування програмного додатку; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, предмет та об'єкт дослідження; задачі дослідження; новизна і практична цінність одержаних результатів; порівняльна характеристика програмних додатків; структура графічного інтерфейсу; блок-схема алгоритму роботи додатку для розпізнавання вимови з аудіо файлу; блок-схема алгоритму для визначення правильно вимовлених частин тексту; блок-схема алгоритму зберігання матеріалу

для вивчення в сховище; використані технології під час розробки додатку; модульне тестування додатку; тестування інтернаціоналізації й створення матеріалу для вивчення; тестування перевірки вимови; інструкція для роботи з програмним кодом; апробація та публікації результатів роботи; 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	25.03.22	10.06.22

7. Дата видачі завдання 25 березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження галузі розпізнавання мовлення і вибір методів для вирішення поставлених задач дослідження	26.03.2022 - 09.04.2022	Вик.
2	Розробка структури інтерфейсу та алгоритмів програмного додатку	10.04.2022 – 28.04.2022	Вик.
3	Обґрунтування вибору мови та середовища розробки ПЗ	29.04.2022 - 03.05.2022	Вик.
4	Програмна реалізація додатку	04.05.2022 - 17.05.2022	Вик.
5	Розробка бази даних із використанням Hibernate	18.05.2022 - 22.05.2022	Вик.
6	Тестування програмного додатку	23.05.2022 - 27.05.2022	Вик.
7	Оформлення матеріалів до захисту БДР	28.05.2022 - 10.06.2022	Вик.

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

Керівник бакалаврської дипломної роботи _____ (підпис) **Романюк О. В.** (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 106 сторінок формату А4, на яких є 49 рисунків, 3 таблиці, список використаних джерел містить 30 найменувань.

У бакалаврській дипломній роботі виконано аналіз області застосування перевірки мовлення й додатків для вивчення правильної вимови. Визначено об’єкт, предмет, завдання та методи дослідження. Сформульовано мету дослідження – підвищення ефективності вивчення іноземної мови шляхом розробки додатку для вивчення правильної вимови звуків. Для реалізації мети було розроблено алгоритми роботи додатку, графічний інтерфейс додатку та виконано програмну реалізацію додатку для вивчення правильної вимови.

Запопоновано алгоритми визначення правильно вимовлених частин тексту відповідно до навчального матеріалу та розпізнавання вимови з аудіо файлу користувача, які дозволили підвищити точність визначення правильно вимовлених частин тексту та загалом підвищити еефективність вивчення правильної вимови. Розроблено алгоритм зберігання навчальних матеріалів в сховище для даних.

Веб-інтерактивний додаток створено з використанням мов програмування Java, JavaScript, Python. Для реалізації Inversion of Control було використано Spring Framework. Для прискорення розробки архітектури додатку застосовано Spring Boot. Для роботи з базою даних було використано ORM систему Hibernate і Spring Data. Для розпізнавання вимови використано Google API speech-to-text. Результатом виконання бакалаврської дипломної роботи є розроблений програмний додаток, який перевірено і для якого підготовлені вимоги для використання.

Ключові слова: веб-інтерактивний додаток, база даних, ORM, Spring, шаблони проектування програмного забезпечення, розпізнавання мовлення.

ABSTRACT

The bachelor's thesis consists of 106 A4 pages, which contain 49 figures, 3 tables, a list of sources used contains 30 items.

In the bachelor's thesis the analysis of the field of application of check of speech and applications for studying of the correct pronunciation is executed. The object, subject, tasks and research methods are defined. The purpose of the research is formulated - to increase the efficiency of learning a foreign language by developing an application for studying the correct pronunciation of sounds. To achieve this goal, the algorithms of the application, the graphical interface of the application were developed and the software implementation of the application for learning the correct pronunciation was performed.

Algorithms for determining correctly pronounced parts of the text according to the training material and pronunciation recognition from the user's audio file are proposed, which allowed to increase the accuracy of determining correctly pronounced parts of the text and generally increase the efficiency of correct pronunciation. An algorithm for storing educational materials in the data warehouse has been developed.

Web-based interactive application created using programming languages Java, JavaScript, Python. The Spring Framework was used to implement Inversion of Control. Spring Boot is used to speed up the development of the application architecture. Hibernate and Spring Data ORM systems were used to work with the database. Google speech-to-text API was used for pronunciation recognition. The result of the bachelor's thesis is a developed software application, which is tested and for which the requirements for use are prepared.

Keywords: web interactive application, database, ORM, Spring, software design templates, speech recognition.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Створення та використання штучного середовища у процесі вивчення іноземних мов є одним із найважливіших питань сучасної методики. У житті сучасної людини багато що визначає знання іноземних мов, особливо англійської, і як наслідок, їх знання стало необхідністю майже для кожного. На сьогодні безліч варіантів вивчення спрямовано на запам’ятовування візуально та аудіально. Оскільки правильна вимова є показником освіченості – кожен повинен нею володіти на задовільному рівні. Приблизно 75% осіб має проблему з вимовою іноземних звуків і бажають цю проблему виправити. І ключовим фактором у вивченні будь-якої мови є практика говоріння та правильна вимова. Для того, щоб досягти цієї мети люди, які навчаються, змушені ходити до репетиторів або знаходити осіб, для яких ця мова рідна й практикуватись разом з ними.

Мовленнєвий розвиток людей і їхнє успішне навчання перебуває у певній залежності від правильної організації та змісту робіт. Знання з фонетики повинні стати фундаментом у навчанні. Вчитель і учні чітко повинні усвідомлювати різницю між сказаним і написаним словом. Це завдання успішно вирішується за умови ефективної організації звуко-літерного розбору, головною особливістю якого є вказівка на положення звуку в слові, як його визначальна умова при виборі літери на його позначення.

Сьогодні існує мала кількість хороших додатків для вивчення вимови, особливо веб. Для вивчення правильної вимови слів слугують два якісних мобільних додатки, за допомогою яких можна перевірити власне мовлення, а саме: Cake [1] й Lingvist [2], але користувачі прив’язані виключно до мобільної платформи, що може не підходити всім у зв’язку своєї незручності під навчання. Варто зазначити, що додаток Cake розпізнає вимову з відчутною затримкою, тому необхідно реалізувати алгоритм, щоб прискорити процес визначення правильно вимовлених слів відповідно до навчальних матеріалів. Крім того, щоб користувачі завжди мали змогу вивчати необхідний матеріал, потрібно створити можливість

власноруч зберігати матеріали для навчання в репозиторій й розробити адаптивну структуру для таких вікон, щоб зовнішній вигляд був привабливим і комфортним для користування. Готові рішення в області розпізнавання мовлення забезпечать можливість налаштувати процеси в залежності від потреб [3]. Застосування Spring Framework з використанням його анотацій спростить процес побудови архітектури додатку й дозволить приділити більше уваги на створення бізнес-логіки.

Тому, підвищення ефективності вивчення правильної вимови шляхом розробки адаптивного до різних платформ додатку для вивчення правильної вимови звуків є актуальною задачею.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета і завдання дослідження. Метою бакалаврської роботи є підвищення ефективності вивчення іноземної мови шляхом розробки додатку для вивчення правильної вимови звуків, у якому реалізовано удосконалені алгоритми розпізнавання вимови з аудіо-файлу користувача та виявлення правильних і неправильних фрагментів вимовленого тексту.

Відповідно до поставленої мети в бакалаврській дипломній роботі потрібно вирішити такі **завдання:**

– розробити структуру графічного інтерфейсу;

– створити адаптивний графічний інтерфейс;

– розробити алгоритм для розпізнавання вимови з аудіо файлу;

– розробити алгоритм для визначення правильно вимовлених слів відповідно до навчального матеріалу;

– розробити алгоритм для зберігання матеріалів для вивчення в сховище;

– виконати розробку бази даних з використанням ORM технологій;

– розробити програмні компоненти для програмного додатку;

– провести тестування програмного додатку;

– розробити інструкцію користувача.

Об’єкт дослідження – процеси розпізнавання правильної вимови з аудіо файлу.

Предмет дослідження – методи та засоби розробки програмного забезпечення для вивчення правильної вимови звуків.

Методи дослідження. У процесі виконання бакалаврської дипломної роботи було виконано дослідження з застосуванням наступних методів: методи теорії баз даних для проектування і розробки бази даних; теорія алгоритмів для розробки та вдосконалення алгоритмів ПЗ; методи розпізнавання мовлення; методи хешування; методи проектування програмного забезпечення; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень методи тестування для перевірки програмного додатку.

Новизна отриманих результатів:

– Вперше запропоновано алгоритм розпізнавання вимови з аудіо файлу користувача, що повертає користувачу не лише повністю розпізнаний текст, а й визначає правильно вимовлені фрагменти тексту, що дозволило підвищити ефективність вивчення іноземної мови.

– Удосконалено алгоритм визначення правильно вимовлених частин тексту відповідно до навчального матеріалу, у якому, на відміну від відомих алгоритмів, реалізований механізм усунення можливих колізій, що виникають при хешуванні рядків кільцевою хеш-функцією, що дозволило підвищити точність виявлення правильно вимовлених слів.

Практична цінність одержаних результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для вивчення правильної вимови звуків. Розроблений додаток може використовуватись особами, які вивчають іноземні мови для покращення розмовних навичок.

Особистий внесок. Наукові результати, які подано у дипломній роботі, отримано автором особисто. У наукових роботах, які опубліковані у співавторстві, автору належать: дослідження технологій автоматизованої перевірки завдань для

освітніх платформ [4], оптимізація запитів до баз даних [5], особливості застосування ORM технологій при роботі з реляційними базами даних [6], особливості використання фреймворку Spring для розробки веб-інтерактивних додатків [7].

Апробація результатів роботи. Матеріали, які використовувались у бакалаврській дипломній роботі, доповідались на:

– XLIX Науково-технічній конференції підрозділів ВНТУ (2020 р., Вінниця);

– Міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (9-10 листопада 2021 р., м. Вінниця);

– LI Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2022 р., Вінниця);

– XXI Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів «СТАН, ДОСЯГНЕННЯ ТА ПЕРСПЕКТИВИ ІНФОРМАЦІЙНИХ СИСТЕМ І ТЕХНОЛОГІЙ» (21-22 квітня 2022 р., м. Одеса).

Публікації. Результати проведених досліджень було опубліковано у 4 наукових працях у збірниках матеріалів конференцій.

1 ДОСЛІДЖЕННЯ СФЕРИ РОЗПІЗНАВАННЯ МОВЛЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз галузей застосування перевірки мовлення

На сьогодні, автоматичне розпізнавання вимови є широко поширеним терміном в багатьох сферах людської діяльності, оскільки після переведення людської мови в комп’ютерний текст, програміст у власному додатку зможе реалізувати будь-які бажання. Популярні системи надають змогу налаштувати технологію до конкретних вимог, починаючи від тону голосу і закінчуючи акцентом диктора [3].

Ідентифікація мовлення зазвичай оцінюється за рівнем точності, а саме: швидкістю, коефіцієнтом помилок в словах. Великої популярності ця технологія отримала в мобільних пристроях, особливо у використанні для обробки мовлення в текст, голосового пошуку та голосового набору повідомлень. Віртуальні помічники використовують розпізнавання мовлення для спілкування з користувачами та виконання різноманітних завдань, які запускаються голосовими командами.

В області освіти ключовою компонентною навчального процесу є перевірка, та аналіз інформації. Автоматична перевірка знань зможе впливово підвищити ефективність навчання [4]. У вивченні іноземної мови важливою складовою є говоріння. Так автоматична перевірка вимовляння звуків може спростити та підвищити якість навчання на будь-яких етапах. Саме в цьому випадку використовується програмне забезпечення для розпізнавання мовлення, що надає користувачам потужну допомогу з вивченням та застосуванням на практиці говоріння.

У сфері охорони здоров’я лікарі можуть використовувати програмне забезпечення із розпізнаванням мови, щоб зберегти нотатки у режимі реального часу в медичні записи [8]. В більшості випадків програмне забезпечення для розпізнавання мовлення в медичній сфері має форму пристроїв для перетворення мови в текст. Ці пристрої часто використовуються під час сеансу з клієнтом, коли

лікарі використовують розпізнавання мовлення, щоб застосовуючи ці додатки допомогти вести свої медичні конспекти. Також великий внесок у допомозі по інвалідності – програмний додаток зможе перекладати вимовлені слова в текст, використовуючи субтитри, щоб людина з втратою слуху могла зрозуміти, що говорять інші. До того ж технологія ідентифікації мовлення зможе допомогти людям з обмеженим використанням рук працювати з комп'ютером, використовуючи голосові команди замість введення тексту.

Важливою перевагою використання розпізнавання мови є спілкування між індивідом та машиною. Система зможе дозволити електронному пристрою спілкуватися з людиною природною мовою. Прикладом є віртуальні асистенти від компанії Apple Siri та Google Assistant, які виконують голосові запити людською мовою, щоб давати відповіді на питання користувачів, знаходити рекомендації.

Добре розроблений програмний додаток простий в експлуатації та часто працює у фоновому режимі. Постійне автоматичне вдосконалення системи розпізнавання мовлення включає AI, яка з часом стає більш ефективною та простішою у застосуванні. Через те, що технологія вирішує завдання розпізнавання мовлення, вона генерує велику кількість даних про людську мову і таким методом покращує свою роботу.

Основним недоліком розпізнавання мовлення є непослідовне виконання. Навіть сучасні системи можуть бути не в змозі точно визначити слова через різницю у вимові кількох осіб, які розмовляють різними мовами. Навколишній шум та швидкість говоріння можуть стати особливою складністю під час розпізнавання, оскільки операція розпізнавання іноді потребує часу, а ці фактори лише збільшують складність цієї функції. Досить часто можуть з'явитись проблеми з аудіо записом, оскільки успіх розпізнавання залежить від використовуваного обладнання, а не лише від програмного забезпечення.

Потужні організації використовують розпізнання вимови для мовного зважування – ця функція вказує алгоритмам надавати особливу увагу певним словам, які вимовляються часто, або які є унікальними для розмови чи теми. У випадку вивчення іноземних мов такий підхід можна використати для визначення

складних слів, щоб у майбутньому учні орієнтувались, що для вивчення певного набору слів необхідно виділити більше часу та уваги. Досить часто, застосовують фільтрацію нецензурної лексики, наприклад, алгоритми YouTube та потокової платформи Twich, що знаходять слова у вимові користувачів, які можуть образити певні верстви населення і в майбутньому блокують їх.

Отже, виходячи з проведеного аналізу відомо, що розпізнавання мови використовується в усіх сферах людського життя, які з часом лише розвиваються і використовуються для нових функцій особливо у вивченні іноземних мов, де необхідно проводити дослідження, оскільки це зможе прискорити та покращити якість навчання.

1.2 Порівняльний аналіз додатків для вивчення правильної вимови

Зовсім нещодавно англійська мова для українців та переважно більшої кількості населення вважалась іноземною, але з часом її роль лише зростала, і сьогодні вона сприймається для усіх як світова мова. Вивчення кожної нової або навіть англійської мови для кожного є великою метою. Кожен дорослий бажає та активно працює над тим, щоб опанувати англійську як мінімум на задовільному рівні й готовий постійно витрачати великі кошти, щоб отримати цю здібність.

Оскільки правильна вимова значимо підкреслює освіченість та статусність, для кожного хто вивчає мову це дуже важливо. Тому навіть для людини, яка розмовляє рідною мовою, є поширена проблема з вимовою певних слів та звуків. Саме тому нині існує велика кількість різного програмного забезпечення як для телефонних операційних систем, так і для комп’ютерних, яскравими прикладами можуть слугувати наступні додатки:

- Speechworld;
- Cake;
- Lingvist;
- Influent.

Cake – це простий у використанні додаток для вивчення англійської мови за допомогою унікального контенту у вигляді діалогів та практичних уроків [1].

Вивчення проводиться через діалоги від реальних носіїв мови щодня, де студент приділяючи цьому лише пару хвилин, зможе дуже швидко отримати бажаний результат. В ньому також доступні спеціально підібрані курси, які містять в собі: десятки найпопулярніших тем, побутові фрази та багато інших важливих речей. Для того, щоб переконатися, що розмовна мова чітка і правильна, перевірка вимови відбувається за допомогою штучного інтелекту. Тоді користувач знатиме, на що ж йому потрібно звернути увагу насамперед. Користувацький інтерфейс додатку, та його вікна представлені на рисунку 1.1.

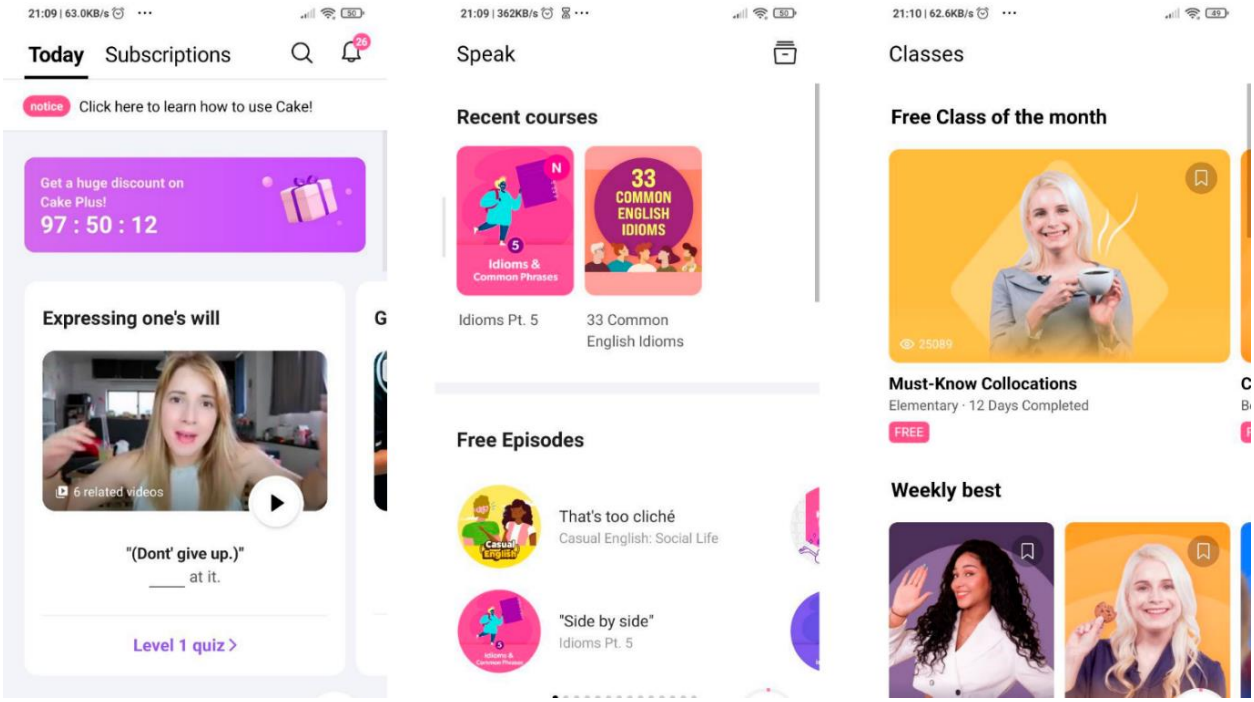

Рисунок 1.1 – Інтерфейс мобільного додатку Cake

Lingvist – це мобільний застосунок, який надає змогу своїм користувачам вивчати англійську, німецьку, іспанську та інші індоєвропейські мови. Дуже потужний додаток саме для покращення нового словникового запасу і за допомогою нього можна покращити свої мовні здібності набагато швидше. Lingvist створений інженерами та вченими з мови, які поєднують останні дослідження в області машинного навчання і персоналізованими мовними курсами. Для підтримки користувачів з усього світу Lingvist доступний 13 мовами. До того ж цей додаток має гарну особливість у вигляді автоматичного визначення рівня, а саме

рівень навичок студента аналізується та динамічно коригується в міру його знань [2]. Крім того, користувач може зберегти власний контент у вигляді слів, зображень або тексту і Lingvist міститиме в собі словниковий запас, пов’язаний з темою, яку вивчаєте. Для зображення слів можна використовувати голосове введення. Зовнішній вигляд програми представлений на рисунку 1.2.

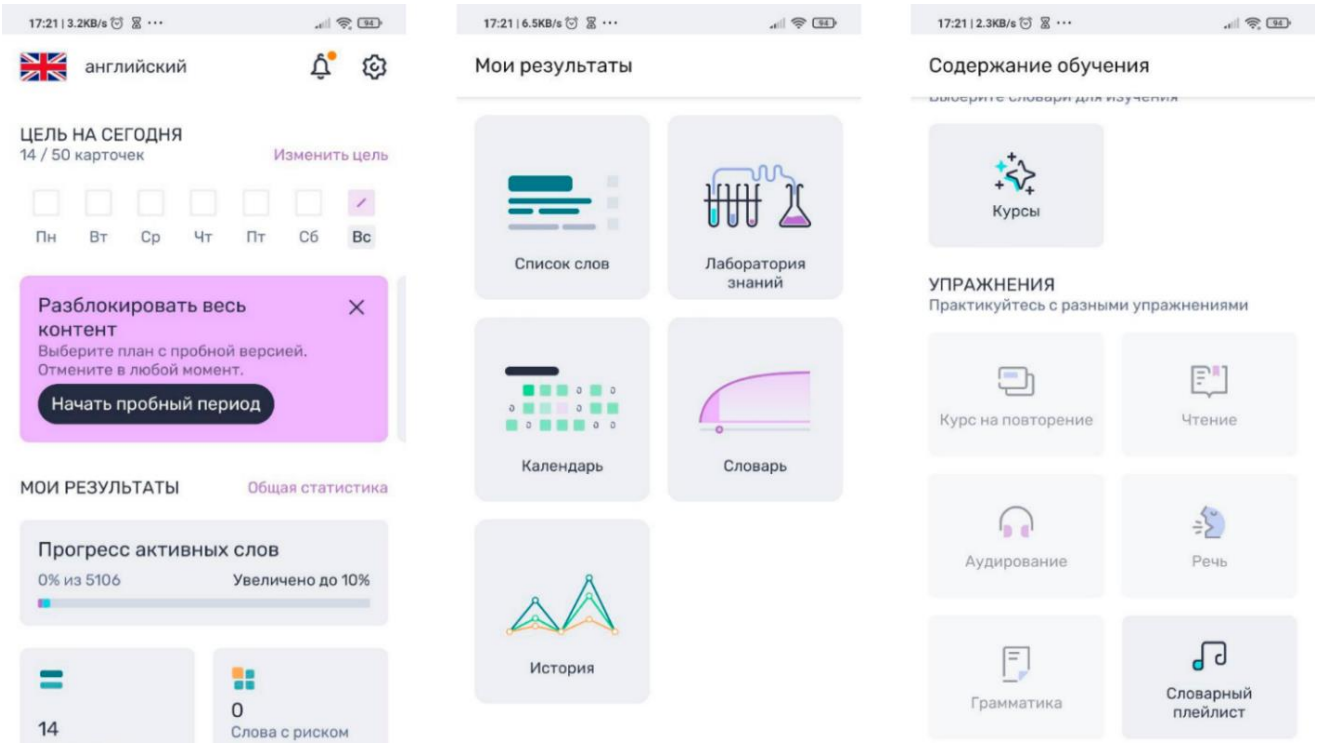

Рисунок 1.2 – Інтерфейс мобільного додатку Lingvist

Influent – це комп’ютерна відеогра, спрямована на те, щоб надихнути людей по всьому світу опанувати нову мову, зробивши засвоєння словникового запасу та правильну вимову веселим і корисним досвідом [9]. Гравці досліджують інтерактивне 3D-середовище, наповнене сотнями колекційних об’єктів, які можна вибрати. Кожен об’єкт має назву, а іноді навіть опис або слово дії, що дозволяє гравцям об’єднувати іменники, прикметники та дієслова у списки словникового запасу, які можна вправляти та опановувати в режимі пошуку та знищення високого рівня. Крім того, програмне забезпечення наповнене сотнями аудіо із правильною вимовою слів від справжніх носіїв мови. Гра представлена на рисунку 1.3.

Рисунок 1.3 – Інтерфейс відеогри Influent

SpeechWorld – веб додаток, який за допомогою науки про мовну терапію перетворює її в цікавий для дітей контент [10]. Використовуючи цей додаток батьки можуть впливати на мовлення своєї дитини, сидячи дома без втручання вчителів. Крім великої кількості карток та відео про артикуляцію, веб-додаток містить в собі навчальні посібники від досвідчених логопедів. Програма також має хмарне сховище даних, тому користувачам не доведеться турбуватися про втрату власного прогресу. Відео контент програмного додатку з правильною артикуляцією для вимови звуку представлений на рисунку 1.4.

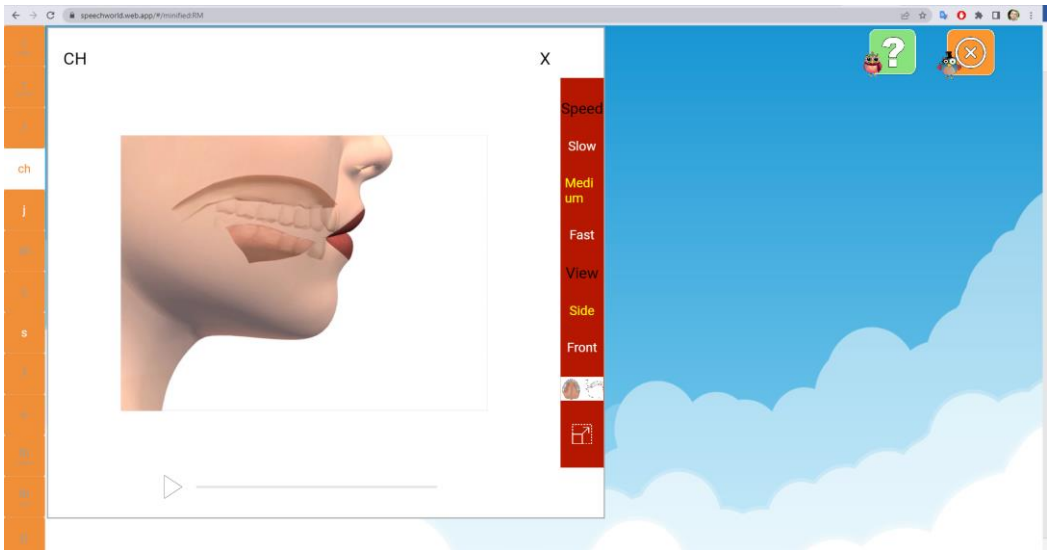

Рисунок 1.4 – Інтерфейс веб-додатку SpeechWorld

Після детального аналізу кожного додатку, створених під мобільні та комп’ютерні пристрої, можемо виділити переваги та недоліки кожного, оскільки вони є унікальними і мають свої особливості. Для цього створимо порівняльну характеристику програмних застосунків із розробленим під назвою «SpeechLearning» у вигляді таблиці представленої у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика програмних додатків

Критерій	Cake	Lingvist	Speech World	Influent	Speech Learning
Створення власного матеріалу для вивчення	0	1	0	1	1
Перевірка вимови користувача	1	1	0	0	1
Багатоплатформність	0	0.5	1	0	1
Безкоштовність повного функціоналу	0	0	0	0	1
Наявність інтернаціоналізації	1	1	0	1	1
Загальний результат	2	3.5	1	2	5

Програмний додаток Lingvist містить не всі функції у власному вебзастосунку, які є наявні в мобільному додатку. Наприклад, введення слів за допомогою голосу відсутнє, тому вирішено, що оскільки користувачі персональних комп’ютерів можуть скористатись основними функціями додатку, вирішено поставити 0.5 за багатоплатформність.

Отже, в результаті проведення детального порівняльного аналізу додатків, було виділено їхні основні переваги та недоліки. Необхідно зазначити, що в результаті виконання програмний додаток повинен перекривати всі наявні мінуси наявних рішень. Варто зазначити, що веб-додаток Speech World не має адаптивної верстки для перегляду, тому під час відвідування даного ресурсу з мобільного

пристрою він не зможе показуватись привабливо для користувачів і ширина веб-сторінки перевищуватиме ширину області мобільного екрану, оскільки, користувачам доведеться прогортати сайт по ширині та висоті, що є дуже некомфортним, тому необхідно буде усунути цей недолік у розробленому програмному рішенні.

1.3 Аналіз методів розв’язання поставлених задач

Насамперед для того, щоб користувачі мали змогу зберігати створений ними контент, необхідно обрати підхід для збереження інформації. Найкращим розв’язанням цієї проблеми буде використання баз даних.

Ключовою частиною для збереження інформації є база даних. Важливою категорією програм сьогодні є системи обробки інформації, засновані на базах даних (БД). Під час створення бази даних користувач намагається впорядкувати інформацію за різними ознаками й швидко витягати вибірку з довільним поєднанням ознак. Зробити це можливо тоді, коли дані структуровані. При розробці БД важливо звертати увагу на оптимізацію коду SQL, оскільки від цього на пряму залежить швидкість і коректність обробки запитів. Для аналізу поточної ситуації проєкту спочатку аналізують план виконання запиту [5].

SQL – мова управління базами даних для реляційних БД [11]. За допомогою мови SQL розробки може створювати запити, для розширення функціональності. Запити зазвичай використовуються для CRUD операцій, особливе значення мають запити на зчитування (read), коли необхідно з кількох таблиць витягнути різні дані та відфільтрувати їх за потрібними параметрами.

Розроблена база даних повинна забезпечувати збереження та отримання інформації без істотних затримок часу. До того ж вона повинна бути незалежною і з легкістю мігрувати до інших баз даних. ORM технології абстрагують всі реляційні бази даних, тобто можна створити об’єкт лише один раз, після чого необхідна реляційна таблиця генерується для потрібної БД MySQL, PostgreSQL, MicrosoftSQL чи іншої [6].

Нині потужні компанії розробляють власні API для розпізнавання вимови

одразу як тільки користувач вимовить слово або з аудіо файлу. Тому для розпізнавання хорошим підходом буде використати готові наявні API. Велика кількість компаній такі як: IBM, Google, Bing, wit, sphinx надають свій широкий набір функцій для розпізнавання вимови. Усі вони мають широкий набір функцій. Хоча варто зазначити, що з усього переліку найсучасніші, найкраще розвинені методи, найлегшу інтеграцію в додатки надає саме Google API. До того ж безплатна підписка на Google Cloud надає можливість використовувати 1 мільйон запитів на місяць перетворення Speech-to-text.

Головними особливостями хмарного перетворення мовлення в текст є [12]:

1. Мовленнєва адаптація дозволяє налаштувати розпізнавання мовлення для того, щоб транскрибувати специфічні для домену терміни й найрідше вживані слова, що дає додаткові підказки, таким чином підвищуючи точність розпізнавання конкретних слів та фраз.

2. Доменні моделі надають змогу програмісту обрати із набору навчених моделей під які функції їх застосовуватиме, а саме: керування голосом, телефонних дзвінків або транскрипції для відео. Якщо модель для аудіо файлу, що надходить від телефонії, тоді вона матиме частоту дискретизації 8 кГц.

3. Легко порівнювати якість, тому розробник може з легкістю експериментувати зі звуком свого мовлення використовуючи простий у використанні інтерфейс користувача. Це надає змогу спробувати різні конфігурації, щоб в кінцевому результаті оптимізувати якість і точність.

Отже, під час аналізу методів розв’язання поставлених задач, було обрано підхід для збереження інформації, а саме: використання баз даних з інтеграцією ORM технологій для їхнього майбутнього абстрагування від конкретної БД. До того ж проаналізовано ефективні рішення для розпізнавання мови та обрано Google Cloud Speech-to-text.

1.4 Постановка задач розробки веб-інтерактивного додатку для вивчення правильної вимови звуків

Після проведеного аналізу сфери розпізнавання мовлення. Для ефективної

розробки веб-інтерактивного додатку для вивчення правильної вимови звуків було визначено наступні задачі, які необхідно здійснити під час створення програмного застосунку:

– розробити структуру графічного інтерфейсу;

– створити адаптивний графічний інтерфейс;

– розробити алгоритм для розпізнавання вимови з аудіо файлу;

– розробити алгоритм для визначення правильно вимовлених слів відповідно до навчального матеріалу;

– розробити алгоритм для зберігання матеріалів для вивчення в сховище;

– виконати розробку бази даних з використанням ORM технологій;

– розробити програмні компоненти для програмного додатку;

– провести тестування програмного додатку;

– розробити інструкцію користувача.

Отже, наведено основні задачі розробки застосунку для вивчення правильної вимови звуків. Технічне завдання до бакалаврської роботи подано у Додатку А.

1.5 Висновки

Отже, під час дослідження сфери розпізнавання мовлення. Було проведено аналіз галузей для яких застосовується перевірка мовлення та вирішено працювати саме над навчальною областю, оскільки вона може принести вагомий вклад у навчання та може спростити роботу викладачам. Після порівняльного аналізу додатків, які спеціалізуються на вивченні іноземних мов виявлено основні недоліки, для майбутнього усунення в створеній програмній реалізації. До того ж проаналізовано методи вирішення для поставлених задач і обрано використання Google API, для розпізнавання вимови з аудіо файлу, щоб виконувати збереження контенту створеного в цілях навчання вирішено використовувати бази даних побудовану з використанням ORM технологій. Крім того, поставлено задачі розробки веб-інтерактивного додатку для вивчення правильної вимови звуків.

2 РОЗРОБКА СТРУКТУРИ ІНТЕРФЕЙСУ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ДОДАТКУ

2.1 Розробка структури графічного інтерфейсу

Для розробки графічного інтерфейсу користувача передбачається веб-інтерфейс. Веб-інтерфейс – програмне забезпечення, яке працює на вебсервері. Користувач може отримати до нього доступ за допомогою підключення до інтернету [13]. Використовуючи технології JavaScript, CSS та Java, існує можливість додати спеціальні функції для програми, такі як малювання на екрані, відтворення звуку та доступ до клавіатури та миші. Методи загального призначення, такі як перетягування (drag and drop) також підтримуються цими технологіями. Веб-розробники досить часто використовують сценарії на стороні клієнта, щоб додати функціональність, після чого ці додатки мають назву веб-інтерактивний додаток. Розробимо структуру для вікон, які лише містять інформацію та посилання на інші сторінки (рисунок 2.1).

Рисунок 2.1 – Структура інтерфейсу для інформаційних сторінок

Інформаційні сторінки для додатку «SpeechLearning» повинні містити наступні елементи:

1. Логотип додатку;

2. Навігаційна панель;

3. Основна інформація про сторінку, її призначення та можливості;

4. Робоча область, на якій користувач може виконувати навігацію, між іншим, навчальним контентом;

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

Також потрібно розробити структуру інтерфейсу вікон, які належать до створення різних сутностей, наприклад для додавання звуків та контенту для навчання (рисунок 2.2).

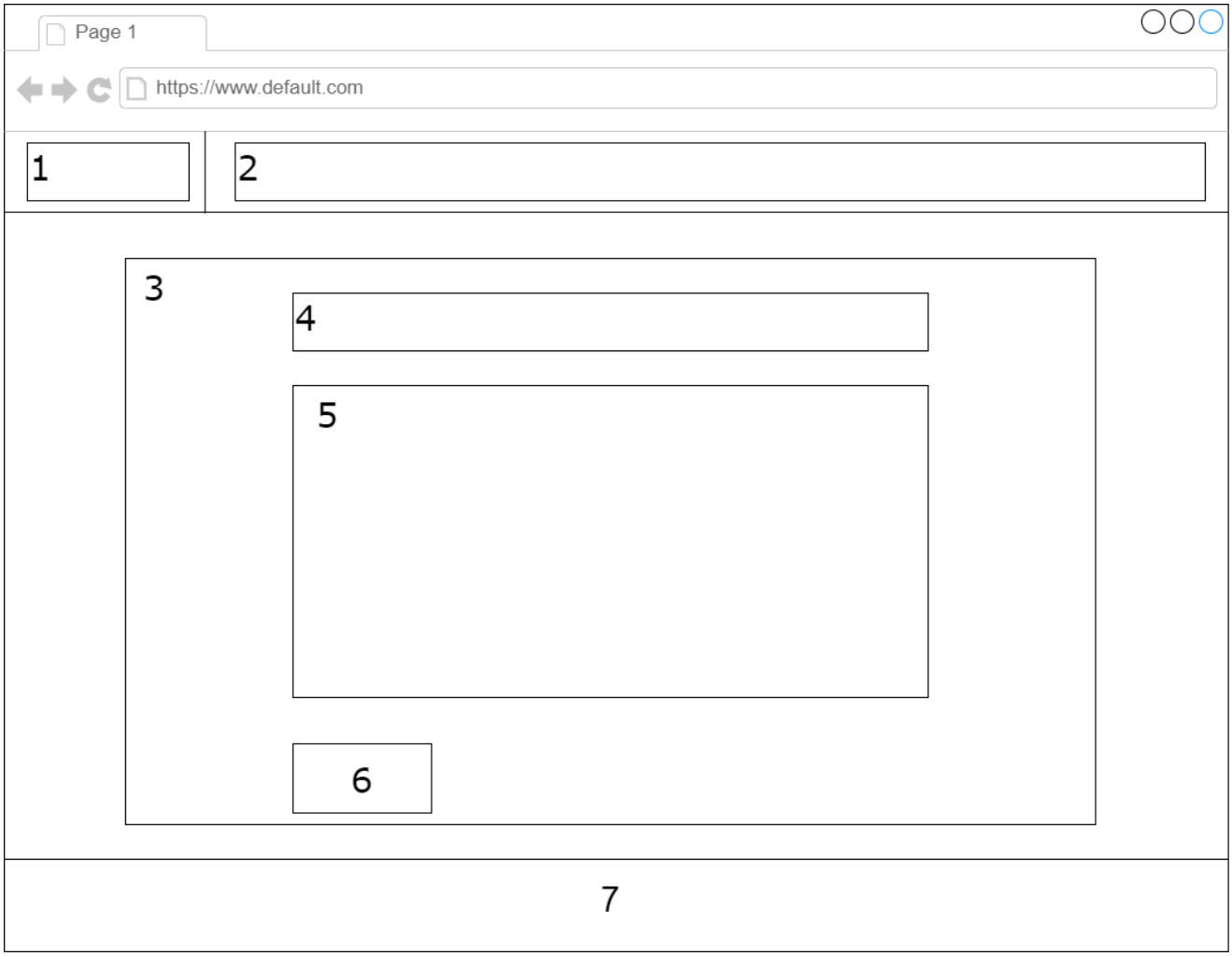

Рисунок 2.2 – Структура інтерфейсу вікна для додавання сутностей

Сторінки, які виконують функцію додавання різних сутностей повинні містити в собі такі елементи:

1. Логотип додатку;

2. Навігаційна панель;

3. Форма у вигляді картки;

4. Назва об’єкта, який доступний для створення на сторінці;

5. Перелік полів, які необхідно заповнити для успішного створення – це може бути текстове поле, аудіо чи файл у форматі зображення;

6. Кнопка підтвердження створення;

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

До того ж необхідно розробити структуру інтерфейсу для вікна, з яким користувач взаємодіятиме та зможе використовувати цю сторінку в навчальних цілях (рисунок 2.3).

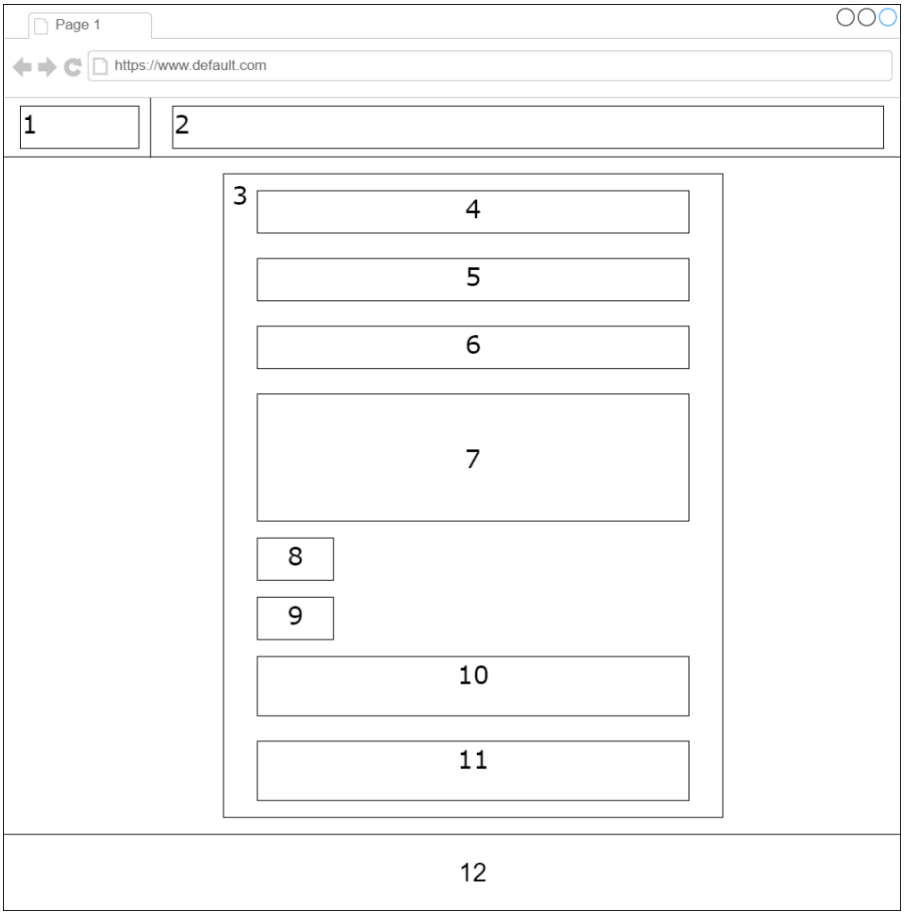

Рисунок 2.3 – Структура інтерфейсу сторінки для вивчення вимови

Вікно, в якому користувач проводитиме найбільшу частину свого часу, а саме: сторінка, на якій зображується контент для вивчення, повинно містити наступні елементи:

1. Логотип додатку;

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.

Отже, під час розробки структури графічного інтерфейсу визначено та створено основні структури вікон для різних цілей. Крім того, окремо було розроблено структуру для вікна, в якому користувач проводитиме найбільше часу, а саме вікна для вивчення вимови звуків, для нього додано детальний опис кожної клавіші та текстового поля.

2.2 Розробка алгоритму роботи додатку для розпізнавання вимови з аудіо файлу користувача

Оскільки алгоритм взаємодії між користувачем та сервісом для розпізнавання вимови з аудіо файлу є складним, необхідно створити для цього блок-схему з детальним описом кожного етапу (рисунок 2.4).

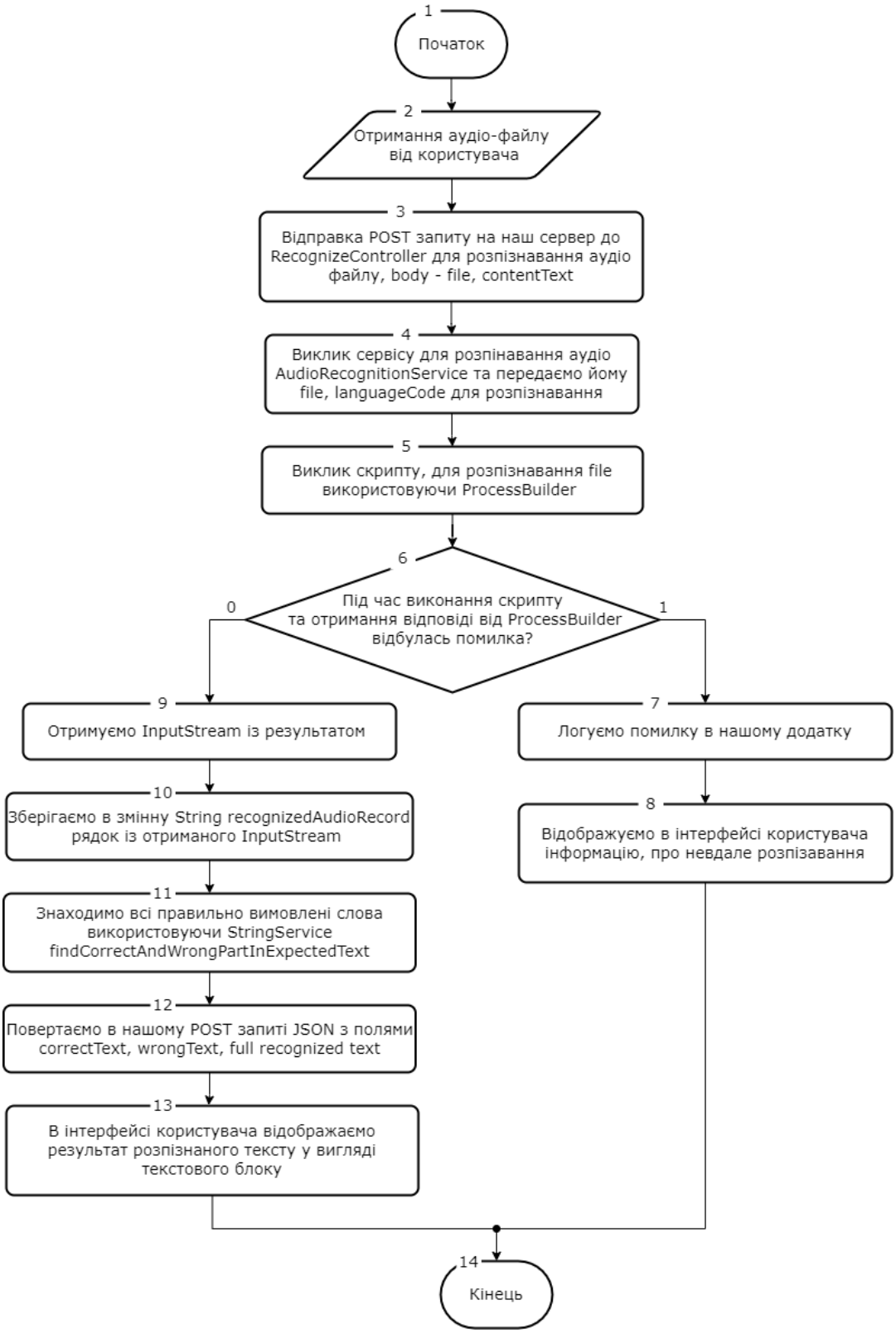

Рисунок 2.4 – Блок-схема алгоритму роботи додатку для розпізнавання вимови з аудіо файлу користувача

Крок 1. Початок.

Крок 2. Додаток отримує аудіо-файл від користувача.

Крок 3. Виконується POST запит для контролера RecognizeController. Тіло запиту повинно містити два поля. Перше з назвою file повинно бути типу MultipartFile для передачі самого файлу, а інше рядкове поле з назвою contentText для порівняння тексту.

Крок 4. В контролері викликається сервіс AudioRecognitionService, який потребує файл з аудіо та кодом мови для розпізнавання.

Крок 5. Сервіс використовує вже готовий написаний скрипт, який отримує лише шлях до файлу і застосувавши ProcessBuilder виконує команду для компіляції скрипта.

Крок 6. Виконується перевірка чи під час застосування скрипта та отримання результатів розпізнавання трапилась помилка.

Крок 7. За умови, що помилка відбулась, логується в додатку.

Крок 8. Відображується інформація для користувача про помилку.

Крок 9. За умови успішного виконання отримується результат розпізнавання у вигляді InputStream.

Крок 10. Зберігається текст, який повернув скрипт.

Крок 11. Використовуючи інший сервіс, знайдено текст, який було розпізнано відповідно до контенту, поділивши його на правильний та не правильний.

Крок 12. Повертається відповідь в POST запиті два рядки в форматі JSON, що містить слова які збігаються в межах контенту.

Крок 13. Відображено в інтерфейсі користувача результат розпізнавання.

Крок 14. Кінець виконання алгоритму.

Отже, створено блок-схему алгоритму роботи додатку впродовж розпізнавання вимови користувача, а також описано детально кожен крок.

2.3 Розробка алгоритму для визначення правильно вимовлених частин тексту

Після розпізнаної вимови необхідно буде визначити, які саме слова користувач вимовив правильно. Для цього необхідно розробити алгоритм, який порівнюватиме два вхідних тексти. Блок-схема представлена на рисунку 2.5.

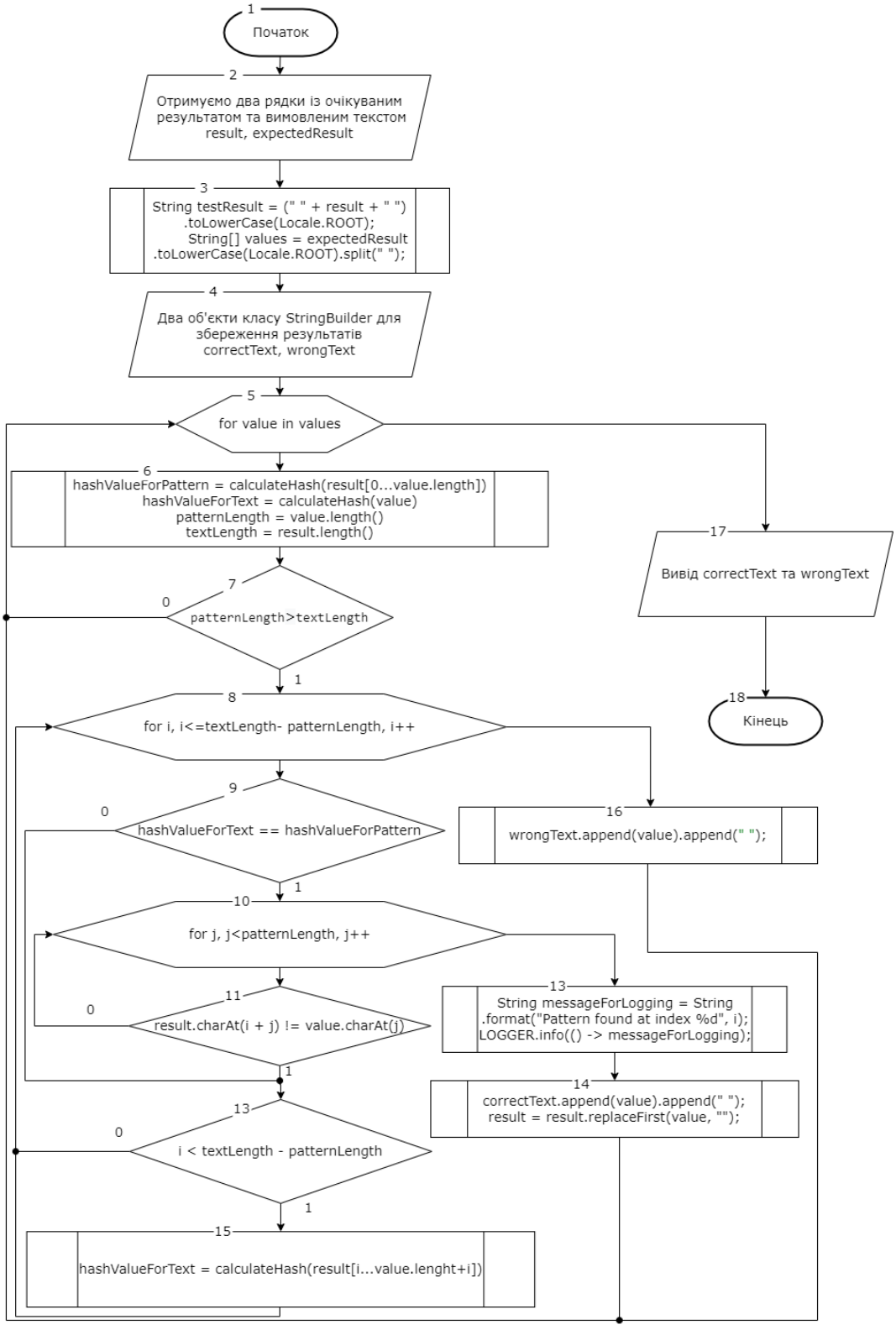

Рисунок 2.5 – Блок-схема алгоритму для визначення правильно вимовленого тексту

Розглянемо кроки алгоритму для визначення правильно вимовлених слів детально:

Крок 1. Початок.

Крок 2. Отримано два рядки типу String, які містять в собі текст очікуваного результату та результату користувача.

Крок 3. Розбивається текст очікуваного результату на окремі слова в масив рядків для їх майбутнього пошуку та управлінням.

Крок 4. Створення двох об’єктів класу StringBuilder для запису результату. Використовується саме StringBuilder оскільки він є Mutable.

Крок 5. Цикл типу foreach масиву рядків для пошуку окремого слова.

Крок 6. Розраховується hash функція для тексту, який вимовив користувач та для слова, яке перевіряється, для цього застосовано кільцеву хеш-функцію, також потрібно знайти довжину цих рядків.

Крок 7. Перевірка чи довжина очікуваного тексту, більша ніж текст умови.

Крок 8. Розпочинається цикл для пошуку слова в тексті.

Крок 9. Порівнюється хеш слова, яке необхідно знайти та частинки тексту.

Крок 10. За умови, що хеші однакові, пройти по кожному елементу частинки тексту. Це виконується для того, щоб однозначно бути впевненим, що це є необхідне слово, оскільки хеш може мати колізії.

Крок 11. Виконується перевірка кожного символу на відповідність.

Крок 12. За умови, що всі символи однакові, зберігається правильно вимовлене слово.

Крок 13. Зберігається повідомлення про знайдений текст.

Крок 14. Для останнього елементу перевірити чи не завершено перевірку всього тексту.

Крок 15. Порахувати хеш для нової частини тексту.

Крок 16. Після завершення циклу з кроку 7, якщо досі не знайдено жодного збігу, значить це слово було вимовлено не правильно і записується до відповідного об’єкту.

Крок 17. Виведення результат пошуку.

Крок 18. Кінець виконання алгоритму.

Отже, в межах розділу було розроблено алгоритм для визначення правильно і неправильно вимовленого тексту й описано значення кожного кроку.

2.4 Розробка алгоритму зберігання в сховище матеріалу для вивчення

Необхідно розробити алгоритм для збереження контенту, з яким зможе взаємодіяти користувач. В ньому виконано детальний опис методів та репозиторіїв, які викликано під час виконання цього алгоритму. Блок-схема представлена на рисунку 2.6.

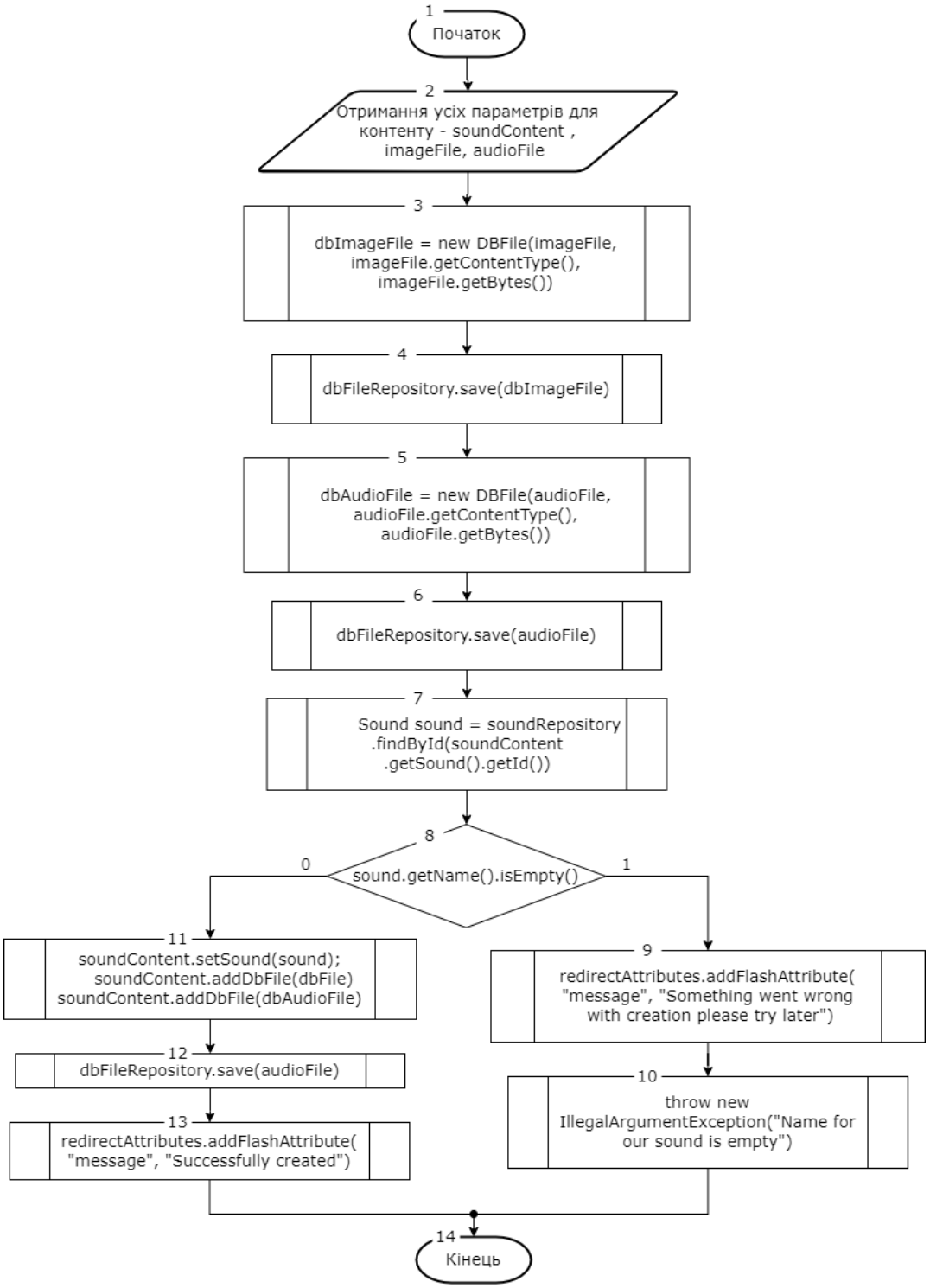

Рисунок 2.6 – Блок-схема алгоритму зберігання контенту для вивчення

Крок 1. Початок виконання алгоритму.

Крок 2. Отримується від користувача параметри навчального контенту та два файли формату зображення та аудіо.

Крок 3. Створюється об’єкт для зберігання зображення в сховищі.

Крок 4. Викликається dbFileRepository для зберігання картинки.

Крок 5. Створюється об’єкт для зберігання аудіо файлу в сховищі.

Крок 6. Викликається dbFileRepository для зберігання аудіо.

Крок 7. Знаходиться в сховищі потрібний для звук.

Крок 8. Перевіряється чи назва звуку містить якесь значення.

Крок 9. За умови, що сутність було отримано невірну, передається повідомлення для користувача про помилку.

Крок 10. Логується помилка в додатку.

Крок 11. Додається вибраний користувачем звук і попередньо збереженні файли в сховищі.

Крок 12. Зберігається контент для вивчення.

Крок 13. Передається повідомлення про успішне зберігання користувачу.

Крок 14. Завершується роботу алгоритму.

Отже, описано кожен крок роботи додатку під час збереження файлу в сховище та розроблено блок-схему алгоритму.

2.5 Висновки

Як наслідок, під час виконання розділу було розроблено структуру для графічного інтерфейсу з детальним описом сторінок і їх призначенням. Також було створено наступні алгоритми: визначення правильно вимовленого тексту, збереження контенту для вивчення й робота додатку під час розпізнавання вимови користувача з аудіо файлу. До того ж наведено блок-схеми та розгорнуто описано кожен крок.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ДЛЯ ВЕБ-ІНТЕРАКТИВНОГО ДОДАТКУ

3.1 Обґрунтування вибору мови та середовища розробки додатку

Сьогодні для прискорення та покращення процесу розробки програмного забезпечення необхідно правильно обрати мову програмування, оскільки кожна мова є унікальною, має власні переваги та недоліки й зазвичай вона є кращою ніж інші в певних областях. Тому потрібно обрати мову й обґрунтувати, для яких цілей вона буде застосовуватись.

Розпочнемо з оформлення інтерфейсу користувача. Для розробки структури та стилю веб-сторінок використовуватимемо мову гіпертекстової розмітки HTML та CSS. У випадку, коли користувач не буде перенаправлятись на іншу сторінку, а лише взаємодіятиме з клавішами, необхідно буде додавати нові текстові елементи та змінювати колір присутніх. У такому випадку використається мова програмування JavaScript, яку можна буде застосувати як на стороні клієнта, так і на стороні серверу, і дозволить створювати необхідні веб-сторінки інтерактивними [14].

Для використання Google API слід застосувати мову, яка є простою в засвоєнні й потужною в кількості власних бібліотек. За допомогою неї можемо реалізувати метод для розпізнавання вимови по аудіо файлу. Саме в цьому разі необхідно використати мову Python, яка є високо рівнева мова програмування та використовується як для розробки самостійних програм, так і для створення прикладних сценаріїв [15]. Це є потужна, легко переміщувана, проста у застосуванні мова, яка містить в собі стандартну бібліотеку, яка включає великий набір корисних функцій. До того ж для Python без будь-яких проблем можна встановити нові бібліотеки й з легкістю їх використовувати у власних програмах. Це надає значну перевагу розробниками під час розробки простих та складних рішень.

Отож, коли мови для розробки користувацького інтерфейсу та мови для розробки методу розпізнавання вимови обрано, далі необхідно обрати мову

програмування для об’єднання цих двох областей та на якій можна буде реалізувати розробку бази даних з використанням технології ORM та іншої логіки програми. Для цієї цілі слід застосувати мову програмування Java – це об’єктно орієнтована мова програмування, розроблена James Gosling в Sun Microsystems, якою з того часу почала займатись Oracle Corporation [16]. Випущена в 1995 році й нині є однією з найпопулярніших мов програмування. Вона може бути використана для розробки додатків на різноманітні середовища: настільні, веб і навіть на мобільні пристрої. Однією з головних особливостей Java є те, що вона не залежить від платформи. Слід зазначити, що програма, яка написана на Java, може виконуватися в будь-якій операційній системі (Windows, Mac або Linux).

Значною перевагою використання мови Java є можливість програмувати у такому середовищі розробки як IntelliJ IDEA – це інтегроване середовище розробки програмного забезпечення. В цій IDEA дуже просто використовувати «Debugging», адже достатньо просто поставити курсор в необхідне місце, та натиснути Alt+F8 [17]. До того ж має хорошу підтримку різних мов програмування.

Хорошим плюсом в IntelliJ IDEA є те, що вона з легкістю розуміє контекст коду, який пише програміст. В цьому середовищі розробки існує авто заповнення при написанні рядку коду, де середовище саме дасть підказку на використання можливих методів. Приклад автоматичного заповнення в середовищі IntelliJ IDEA представлений на рисунку 3.1.

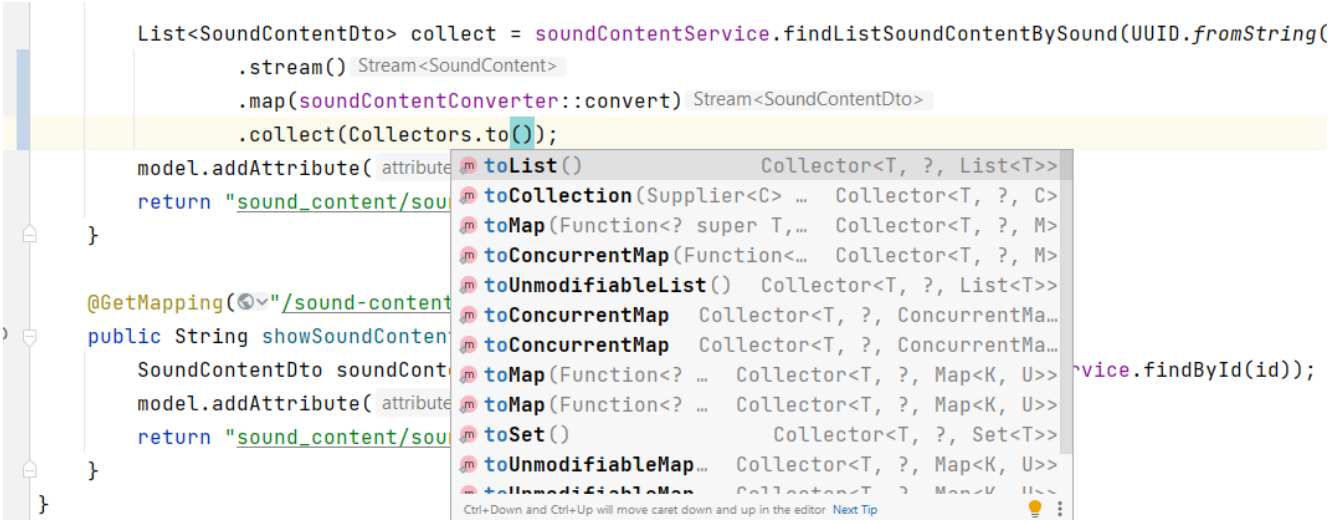

Рисунок 3.1 – Приклад авто заповнення в середовищі IntelliJ IDEA

Крім того, IntelliJ IDEA сильно спрощує процес рефакторингу. Рефакторингом є перероблення коду через процес зміни внутрішньої структури програми, яка не стосується її зовнішньої поведінки та має ціль полегшити розуміння її роботи [18]. В середовищі розробки цей процес є інтелектуальним, оскільки містить велику кількість готових функцій, які можна викликати через меню або гарячі клавіші. Приклад можливих функцій для рефакторингу представлений на рисунку 3.2.

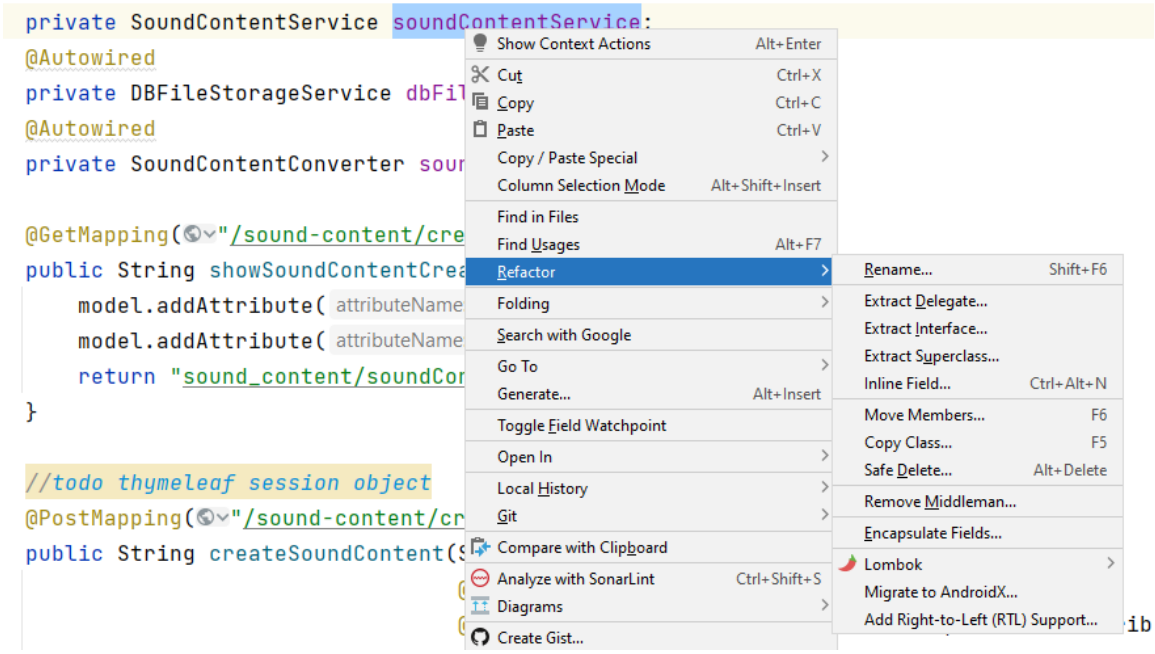

Рисунок 3.2 – Перелік функцій для рефакторингу в середовищі розробки

Отже, було обрано мови програмування для поставлених цілей та обґрунтовано їх використання. До того ж було обрано середовище програмування IntelliJ IDEA й наведено переваги, основною є наявність підтримки різних мов через те, що в залежності від потреб будуть використані різні мови.

3.2 Використання Spring технологій для розробки додатку

В загальному Spring Framework розглядається як множина менших фреймворків, які працюють незалежно один від одного, але надають велику кількість функцій [7]. Для швидкої розробки веб-інтерактивного використано Spring Boot, оскільки він автоматично знаходить classpath (шлях до класу)

налаштованих beans [19]. Отже, розробник зможе зосередитись більше на бізнес-функціях й менше на загальній архітектурі додатку. Слід зазначити, що Spring Boot не редагує створені файли і не генерує програмний код, а лише під час запуску додатку зв’язує усі компоненти програми та застосовує їх відповідно контексту програми.

Використавши просту анотацію @SpringBootApplication у класі для запуску програми, розробник одночасно зазначить для додатку наступні особливості:

@Configuration – позначає клас як джерело визначень bean для контексту програми.

@EnableAutoConfiguration – надає інформацію Spring Boot про додавання bean-компонентів, а саме: їхні налаштування шляху до класів, залежні компоненти і різні налаштування властивостей.

@ComponentScan – повідомляє, де необхідно шукати інші компоненти й конфігурації, дозволяючи йому знайти контролери та репозиторії.

У всій програмі використано анотацію @Component, щоб позначити beans як окремі компоненти Spring. А це значить, що Spring підбирає та реєструє bean лише за допомогою @Component.

Під час розробки додатку необхідно застосовувати принцип IoC (Inversion of Control). Інверсія керування – це принцип розробки програмного забезпечення, у якому управлінням об’єктами займається контейнер [20], а це означає, що виконання програми буде контролювати сама платформа. В майбутньому під час розширення функціоналу такий підхід полегшить перехід між різними реалізаціями.

Реалізувати цей принцип можна використовуючи Spring Framework, а саме: Dependency Injection – це шаблон проєктування, який можна використовувати для реалізації IoC застосувавши інвертований елемент керування, та встановити залежності для необхідного об’єкта [21]. Без використання ін’єкції залежностей модулі верхнього рівня залежатимуть від модулів нижнього рівня, що вимагає додаткових перетворень у коді після зміни реалізації. Тому модулі повинні залежати від абстракцій, а не реалізації. Spring дає змогу розробникам автоматично

впроваджувати залежності між спільними компонентами, використовуючи анотацію @Autowired. Приклад ін'єкції залежності між спільними компонентами сервісу та сховища для даних представлено на рисунку 3.3.

```
@Service
public class SoundContentService {

    private final SoundContentRepository soundContentRepository;

    @Autowired
    public SoundContentService(SoundContentRepository soundContentRepository) {
        this.soundContentRepository = soundContentRepository;
    }
```

Рисунок 3.3 – Dependency Injection в класі SoundContentService

Для спрощення розробки додатку по архітектурному шаблону MVC (Model View Controller) призначено створеним класам анотацію @Controller, оскільки лише тоді диспетчер зможе виявити всі методи анотовані анотаціями запитів. Приклад класу, який містить обробник запиту GET для завантаження файлу, представлений на рисунку 3.4.

```
@Controller
public class DBFileController {

    private DBFileStorageService dbFileStorageService;

    @Autowired
    public void setDbFileStorageService(DBFileStorageService dbFileStorageService) {
        this.dbFileStorageService = dbFileStorageService;
    }

    @GetMapping("/files/{fileId}")
    public ResponseEntity<Resource> downloadFile(@PathVariable UUID fileId) {
        // Load file from database
        DBFile dbFile = dbFileStorageService.findById(fileId);

        return ResponseEntity.ok()
                .contentType(MediaType.parseMediaType(dbFile.getFileType()))
                .header(HttpHeaders.CONTENT_DISPOSITION,
                        ...headerValues: "attachment; filename=\"" + dbFile.getFileName() + "\"")
                .body(new ByteArrayResource(dbFile.getData()));
    }
}
```

Рисунок 3.4 – Приклад використання анотації @Controller

Існує також анотація @Service, яка застосовується лише для відмітки, що клас містить в собі бізнес-логіку. Він є також підвидом анотації @Component. Приклад розробленого сервісу для зберігання файлу та знаходження його по ідентифікатору в сховищі представлений на рисунку 3.5.

```
@Service
public class DBFileStorageService {

    private final DBFileRepository dbFileRepository;

    @Autowired
    public DBFileStorageService(DBFileRepository dbFileRepository) {
        this.dbFileRepository = dbFileRepository;
    }

    @Transactional
    public DBFile storeFile(MultipartFile file) {
        // Normalize file name
        String fileName = StringUtils.cleanPath(Objects
                .requireNonNull(file.getOriginalFilename()));

        DBFile dbFile = null;
        try {
            dbFile = new DBFile(fileName, file.getContentType(), file.getBytes());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return dbFileRepository.save(dbFile);
    }

    public DBFile findById(UUID uuid) {
        return dbFileRepository.findById(uuid)
                .orElseThrow();
    }
}
```

Рисунок 3.5 – Сервіс для роботи з файлами

Під час розробки сховища для даних постійно використовується анотація @Repository, яка вказує програмі, що даний клас є репозиторієм і він призначений для зберігання, пошуку, оновлення та видалення об’єктів [22]. Це є особливий вид раніше зазначеної анотації @Component, що дозволяє автоматично визначати клас

реалізації. Тому за допомогою попередньо об’явленого @ComponentScan beans будуть автоматично знайдені та вбудовані. До того ж для прискорення розробки репозиторію використаємо технологію Spring Data [23]. Вона надає готові рішення для обробки даних (додавання, видалення, оновлення), таким чином спростивши реалізацію ключових запитів. Наведемо приклад використання цього інструменту для сховища файлів на рисунку 3.6.

```
@Repository
public interface DBFileRepository extends JpaRepository<DBFile, UUID> {

}
```

Рисунок 3.6 – Використання Spring Data для сховища файлів

Відтепер інтерфейс для зберігання файлів містить усі необхідні методи для збереження, знаходження та видалення, крім того, є методи, які працюють по ідентифікатору. Спробуємо відтворити цей самий функціонал лише для збереження і знаходження сутності. Результат представлений на рисунку 3.7.

```
@Repository
public class DBFileRepository {

    private final DBFileDao dbFileDao;

    @Autowired
    public DBFileRepository(DBFileDao dbFileDao) {
        this.dbFileDao = dbFileDao;
    }

    public DBFile save(DBFile dbFile) {
        dbFileDao.persist(dbFile);
        return dbFile;
    }

    public DBFile findById(UUID uuid) {
        return dbFileDao.findById(uuid);
    }
}
```

Рисунок 3.7 – Реалізація методів без використання Spring Data

Spring Framework також містить класи для створення інтернаціоналізації в додатку, щоб розроблена програма мала змогу визначити, побажання до обраної мови користувача, в конфігураційний клас необхідно додати bean LocaleResolver. Інтерфейс LocaleResolver має готові імплементації, за допомогою яких можна визначити поточний мовний стандарт використовуючи файли cookie, сеанс або застосувати фіксоване значення.

У програмного забезпеченні використано CookieLocaleResolver на coockie та встановили для нього локаль за замовчуванням US. Також встановимо шлях (classspath:messages) за яким програма знаходитиме файли з текстом, а отже усі файли з назвою messages будуть розпізнаватись як файли джерела для повідомлень (рисунок 3.8).

```
@Configuration
public class InternationalizationConfig implements WebMvcConfigurer {

    @Bean
    public LocaleResolver localeResolver() {
        CookieLocaleResolver cookieLocaleResolver = new CookieLocaleResolver();
        cookieLocaleResolver.setDefaultLocale(Locale.US);
        return cookieLocaleResolver;
    }

    @Bean
    public ReloadableResourceBundleMessageSource messageSource() {
        ReloadableResourceBundleMessageSource reloadableResourceBundleMessageSource
                = new ReloadableResourceBundleMessageSource();
        reloadableResourceBundleMessageSource.setDefaultEncoding("UTF-8");
        reloadableResourceBundleMessageSource.setBasename("classpath:messages");
        return reloadableResourceBundleMessageSource;
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        LocaleChangeInterceptor localeInterceptor = new LocaleChangeInterceptor();
        localeInterceptor.setParamName("lang");

        registry.addInterceptor(localeInterceptor).addPathPatterns("/*");
    }

}
```

Рисунок 3.8 – Конфігураційний клас для інтернаціоналізації додатку

Отож, в межах розділу наведено основні інструменти Spring Framework, які будуть застосовані для розробки додатку. Визначено основні анотації, з якими потрібно працювати. До того ж описано архітектурні принципи та шаблони проєктування Inversion of Control, Dependency Injection, Model View Controller й додано інформацію, як саме реалізовано їх в застосунку. Також, було створено конфігураційний клас для інтернаціоналізації додатку. Слід зазначити, що використання технології Spring Data значно спростить роботу з сховищем для даних і зменшить кількість написаного коду, як і було наведено в прикладах розділу.

3.3 Розробка інтерфейсу користувача за допомогою Bootstrap та Thymeleaf

Для розробки адаптивного веб-інтерфейсу для додатку застосовано безплатний фреймворк Bootstrap. Bootstrap використовується для швидкого створення та налаштування адаптивних сторінок, також цей популярний інструмент доступний з відкритим вихідним кодом, що включає адаптивну систему сіток та багато готових для використання компонентів та плагінів JavaScript [24]. Сторінка, яка має адаптивний дизайн, дозволяє користувачу визначати розмір та орієнтацію. Відповідно зображення буде автоматично налаштовуватись в залежності від пристрою, на якому запущений додаток, і його розмірів.

Bootstrap містить в собі велику кількість готових реалізацій для навігаційних панелей, анімацій, форм, клавіш. Також існують готові класи, які автоматично підберуть кольорову гаму, в залежності від ваших потреб.

Thymeleaf – це серверний механізм шаблонів для Java, який можна застосувати у представленні XML та HTML. Завдяки інструментам створеним для Spring Framework існує багато інтеграцій з цією технологією та є можливість підключати власні функції, тому він ідеально підходить для сучасної веб-розробки [25]. В додатку, який будуватиметься по шаблонові MVC, Thymeleaf використовуватимемо на рівні View (шар для відображення).

Щоб досягти простішої та якіснішої інтеграції з мовою Java, Thymeleaf надає власний діалект, який спеціально реалізує всі необхідні функції для правильної

роботи з Spring. Фрагмент коду, який використовується для зображення таблиці контенту для вивчення й котрий поєднує класи від Bootstrap, Thymeleaf і звичайний HTML представлений на рисунку 3.9.

```
<tbody class="bg-light text-dark">
<tr th:each="sound_content : ${soundContentsList}">
    <td th:text="${sound_content.getContentText()}"></td>
    <td th:text="${sound_content.getContentType()}"></td>
    <td>
        <audio controls>
            <source th:src="@{${'/files/'+sound_content.getAudioFile().getId()}}">
        </audio>
    </td>
    <td class="text-center">
        <a class="btn btn-info" th:href="@{${'/sound-content/'+ sound_content.getId()}}">Open</a>
    </td>
</tr>
</tbody>
```

Рисунок 3.9 – Фрагмент коду зображення контенту для вивчення

Використовуючи команди, які надає інструмент, програма спершу проходить по кожному елементу зі списку застосувавши команду th:each. Після чого призначає для тегу розмітки текст, який містить об’єкт, а саме: контент, його тип, аудіо файл й призначаємо посилання для відкриття контенту, як у звичайних тегах HTML. Результат виконання коду представлений на рисунку 3.10.

Content text	Type content	Audio	Link on content
fileUpload	type	0:00 / 0:00	Open
Listen the audio content for	test type	0:00 / 0:00	Open
listen content listen form	qwerqwer	0:00 / 0:00	Open
Chocolate champion chance	Enumeration wordsw	0:00 / 0:11	Open
testCreationViaUI	type	0:00 / 0:06	Open
qwerqwer	qwerqwer	0:00 / 0:06	Open
Champion	qwerqwer	0:00 / 0:02	Open

Рисунок 3.10 – Зображення таблиці контенту для вивчення

Отже, в даному розділі було обґрунтовано вибір інструментів для розробки веб-інтерфейсу користувача й наведено їхні основні переваги. До того ж описано фрагмент коду, який використовується для показу контенту для вивчення.

3.4 Розробка бази даних з використанням Hibernate ORM

На сьогодні бази даних набули широкого поширення для зберігання інформації й для того, щоб зв’язати базу даних з об’єктно орієнтованою мовою програмування програмісти у своїх рішеннях застосовують «Об’єктно реляційне зображення». ORM – техніка, яка дозволяє конвертувати об’єкти в зрозумілий для баз даних форму, та маніпулювати ними, використовуючи мову програмування, що дозволяє виконувати роботу з даними в межах класів. Ключова мета застосування ORM технологій – полегшити розробникам роботу з написанням коду для маніпулювання з базами даних.

Для розробки програмного забезпечення використано систему Hibernate – безплатна технологія об’єктно реляційного відображення для мови програмування Java, яка спрощує розробку програм які, взаємодіють з базами даних. Головною функцією є представлення класів у формі таблиць бази даних та керування ними за допомогою готових засобів для запитів create, read, update, delete (CRUD), що звільняє програміста від обробки та перетворення результатів в об’єктно орієнтований формат.

Слід зазначити, що розробнику не потрібно обирати конкретну базу даних. ORM системи абстрагують всі реляційні бази даних, отож завжди існує можливість, використовуючи мову програмування Java, створити об’єкт одного разу, а після чого необхідна таблиця автоматично генерується для потрібної БД. Це дозволяє розробнику керувати об’єктами і їх станом та поведінкою, а не реляційною моделлю даних. В іншому випадку потрібно буде створювати всі таблиці та будувати відношення між ними з самого початку, витративши зайвий час.

Складною частиною під час розробки бази даних з використанням Hibernate є конфігурація, оскільки окрім звичайних налаштувань, які використовуються для бази даних, необхідно буде зазначити інші особливості, як от діалект, підхід до

генерації таблиць, автоматичне логування запитів й багато інших функцій. Розпочнемо налаштування Hibernate використовуючи XML конфігурацію рисунок 3.11.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
        <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/SpeechRecognition</property>
        <property name="hibernate.connection.username">postgres</property>
        <property name="hibernate.connection.password">1234</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQL10Dialect</property>
        <property name="hibernate.hbm2ddl.auto">create-drop</property>
        <property name="hibernate.connection.autocommit" >false</property>

        <mapping class="ihorko.work.speech_recognition.db.entity.Sound"/>
        <mapping class="ihorko.work.speech_recognition.db.entity.SoundContent"/>
        <mapping class="ihorko.work.speech_recognition.db.entity.DBFile"/>
    </session-factory>
</hibernate-configuration>
```

Рисунок 3.11 – Конфігураційний XML файл для бази даних

Використання Spring Framework дозволяє виконати конфігурацію в вигляді Java класу лише зазначивши анотацією @Configuration, що цей клас є конфігураційним. Приклад такої конфігурації наведений на рисунку 3.12.

```java
@Bean
public DataSource dataSource() {
    BasicDataSource dataSource = new BasicDataSource();
    dataSource.setDriverClassName("org.postgresql.Driver");
    dataSource.setUrl("jdbc:postgresql://localhost:5432/SpeechRecognition");
    dataSource.setUsername("postgres");
    dataSource.setPassword("1234");

    return dataSource;
}

private Properties hibernateProperties() {
    Properties hibernateProperties = new Properties();
    hibernateProperties.setProperty(
            "hibernate.hbm2ddl.auto", "update");
    hibernateProperties.setProperty(
            "hibernate.dialect", "org.hibernate.dialect.PostgreSQL10Dialect");

    return hibernateProperties;
}
```

Рисунок 3.12 – Конфігурація Hibernate в Java класі

Необхідно створити сутності для бази даних, використовуючи мову Java й анотації, які надає ORM Hibernate. Варто розпочати з опису сутності для звуку, який містить ім’я, мову, унікальний ідентифікатор й список контенту для вивчення. Зображення з описом представлене на рисунку 3.13.

```
@Entity(name = "sound")
@Getter
public class Sound {

    @Id
    @GeneratedValue
    private UUID id;

    private String name;

    private String language;

    @OneToMany(mappedBy = "sound", cascade = CascadeType.ALL, targetEntity = SoundContent.class)
    private List<SoundContent> soundContents = new ArrayList<>();

    public void addSoundContent(SoundContent soundContent) {
        soundContents.add(soundContent);
        soundContent.setSound(this);
    }

    public void setId(UUID id) { this.id = id; }

    public void setName(String name) { this.name = name; }

    public void setLanguage(String language) { this.language = language; }
}
```

Рисунок 3.13 – Опис сутності Sound

@Entity – анотація використовується для того, щоб зазначити, що клас є сутністю й має назву sound, саме так буде називатись таблиця в БД.

@Id – анотація означає, що це поле є первинним ключем для таблиці.

@GeneratedValue – забезпечує певний підхід генерації для значень первинних ключів. Анотацію GeneratedValue зазвичай застосовують до атрибута первинного ключа. Використання цієї анотації автоматично генерує простий первинний ключ для поля під час збереження його в базу даних.

Додати зв’язок @OneToMany від Hibernate дуже легко, але потрібно знати правильний спосіб відображення такої асоціації, щоб вона генерувала необхідний

зв’язок між таблицями, тому це безумовно непроста річ. У реляційних БД асоціація «один до багатьох» зв’язує між собою дві таблиці на основі первинного ключа батьківської таблиці й стовпця зовнішнього ключа дочірньої [26]. Варто додати, що не можна використовувати звичайний setter для залежного поля, тому необхідно створити власний метод для додавання зв’язку «addSoundContent», який приймає поле як сетер, але додає до списку новий контент, а для контенту зазначає, що він залежить від цього звуку.

Якщо зв’язок не є одностороннім, необхідно вказати mappedBy, а саме: таблицю, яка є батьківською. Вказавши каскадний тип, необхідно обрати операції, які будуть впливати на залежну таблицю, а за допомогою targetEntity, визначається, який клас буде залежати від батьківського класу. Як наслідок, вказано, що для одного звуку, може бути багато контенту для вивчення.

Далі розробимо сутність контенту для вивчення. Програмний код даного класу представлений на рисунку 3.14.

```
@Entity(name = "sound_content")
@Getter
public class SoundContent {

    @Id
    @GeneratedValue
    private UUID id;

    private String contentText;

    private String typeContent;

    @ManyToOne
    @JoinColumn(name = "sound_id", nullable = false)
    private Sound sound;

    @OneToMany(mappedBy = "soundContent", targetEntity = DBFile.class)
    private List<DBFile> dbFiles = new ArrayList<>();

    public void addDbFile(DBFile dbFile) {
        dbFiles.add(dbFile);
        dbFile.setSoundContent(this);
    }
```

Рисунок 3.14 – Програмний код сутності SoundContent

Більшість анотацій є подібними, як і для попередньої сутності, але клас SoundContent залежить від Sound і повинен зберігати в собі на нього посилання. Тому, необхідно додати анотацію @ManyToOne й додати колонку за допомогою @JoinColumn для збереження ідентифікатора батьківської сутності, як заведено в реляційних базах даних. Також слід додати nullable = false, щоб не існувало можливості створити цю сутність без прив'язки до конкретного Sound. Усі атрибути в таблиці матимуть назви полів у класі, але існує можливість задати їхнє ім'я явно за допомогою анотації @Column.

Вміст для вивчення може містити, як аудіо файли, так і зображення, тому далі розробимо сутність для збереження файлів (рис 3.15).

```
@Entity(name = "file")
public class DBFile {

    @Id
    @GeneratedValue
    private UUID id;

    private String fileName;

    private String fileType;

    @Lob
    @Type(type = "org.hibernate.type.BinaryType")
    private byte[] data;

    @ManyToOne
    @JoinColumn(name = "sound_content_id")
    private SoundContent soundContent;
```

Рисунок 3.15 – Опис сутності для збереження файлу

Зберігання файлів представлене як масив байтів, тому необхідно зазначити, що це буде бінарний тип за допомогою анотації @Type. Також файли можуть бути великих розмірів й для цього використовується анотація @Lob, яка вказує базі даних, що це поле зберігатиметься як великий об'єкт. Після запуску програми

автоматично генеруються таблиці для БД. Через простоту використання й потужну СУБД PgAdmin, для зберігання інформації використано базу даних PostgreSQL, й також згенеровано готову ER-модель в додатку PgAdmin [27]. Модель представлена на рисунку 3.16.

Рисунок 3.16 – ER-модель бази даних для збереження навчального контенту

Отже, в даному розділі описано розробку бази даних з використанням системи ORM бібліотеки Hibernate, а саме: наведено основні переваги

використання об’єктно реляційного інструменту для веб-додатку. До того ж розроблено три сутності у вигляді Java класів для збереження навчальних матеріалів. Крім того, додано детальний опис анотацій, які використано під час розробки бази даних. Використовуючи pgAdmin, згенеровано ER-модель для бази даних.

3.5 Розробка програмного модуля для розпізнавання вимови користувача

Програмний модуль для розпізнавання вимови користувача складається з кількох етапів, першим етапом якого є визначення вимови з аудіо файлу. Для цього етапу розроблено скрипт на Python, який використовує Google API. Перед цим потрібно було встановити дві бібліотеки Python, а саме: SpeechRecognition та pyAudio. Лістинг коду скрипту представлений на рисунку 3.17.

```
import speech_recognition
import sys

fileNameForRecognition = sys.argv[1]
languageCode = sys.argv[2]
recognizer = speech_recognition.Recognizer()
test_file = speech_recognition.AudioFile(fileNameForRecognition)
with test_file as source:
    audio = recognizer.record(source)
    text = recognizer.recognize_google(audio, None, languageCode, False)
    print(text)
```

Рисунок 3.17 – Лістинг коду скрипту який виводить текст з аудіо файлу

Для скрипту слід передати місце знаходження файлу і код мови якою записаний аудіо файл. Як наслідок можна використовувати цей скрипт у будь-якому додатку як окремий сервіс.

Другим етапом розробки модулю буде створення сервісу (AudioRecognitionService), який викликатиме скрипт і зберігатиме розпізнаний текст в змінну типу String. Для цього використано ProcessBuilder, який може виконати команди для Command Line. Оскільки планується робота з українською мовою, необхідно буде використати кирилицю, тому потрібно вказати charset

(набір символів), а саме windows-1251. Лістинг коду розробленого сервісу представлений на рисунку 3.18.

```
@Service
public class AudioRecognitionService {

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

    @SneakyThrows
    public String recognizeAudioRecord(String filePathFromSourceRoot, Language language) {
        ProcessBuilder processBuilder = new ProcessBuilder( ...command: "python",
                "src/main/resources/python/audio_recognition.py", filePathFromSourceRoot, language.getCode());
        processBuilder.redirectErrorStream(true);
        Process process = processBuilder.start();

        try (InputStream inputStream = process.getInputStream();
             BufferedReader bufferedReader = new BufferedReader(
                     new InputStreamReader(inputStream, charsetName: "windows-1251"))) {

            String recognizedText = bufferedReader.readLine();
            if (recognizedText != null)
                LOGGER.warning(() -> "Recognized text: " + recognizedText);
            return recognizedText;
        } catch (IOException e) {
            LOGGER.severe(e.getMessage());
        }
        return null;
    }
}
```

Рисунок 3.18 – Сервіс розпізнавання мовлення

Наступним етапом буде створення Controller, що містить метод який прийматиме POST request для розпізнавання мовлення відповідно до навчального матеріалу. Параметрами якого буде файл й унікальний ідентифікатор контенту у форматі String. Після чого знаходить в сховищі матеріал для вивчення використовуючи soundContentService, перед цим відформатувавши ідентифікатор із String до типу UUID. Після отримання результатів від audioRecognitionService перевіряється чи взагалі було розпізнано хоч якийсь текст. За умови, що значення розпізнаного тексту null, повертається код 400 (Bad request). Коли отримується позитивна відповідь від сервісу, то сервіс знаходить правильно й неправильно вимовлений текст відповідно до вмісту матеріалу для вивчення. Після чого результати зберігаються в об’єкт класу RecognitionResult, який містить три поля: весь текст який було розпізнано (fullText), текст який збігається відповідно до умови навчального матеріалу (correctText), текст який відрізняється від умови

(wrongText). Збережений об’єкт перетворюємо у формат Json, застосувавши бібліотеку GSON, і відправляємо його у відповідь. Лістинг коду POST методу представлений на рисунку 3.19.

```
@SneakyThrows
@PostMapping(value = "/recognize", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<?> recognizeAudioFile(@RequestParam("file") MultipartFile file,
                                            @RequestParam("soundContentId") String soundContentId) {
    String fileName = file.getOriginalFilename();
    File fileDestination = new File( pathname: "D:\\Study\\VNTU\\Dyplom\\speechTherapy\\web_app" +
            "\\speech_recognition\\src\\main\\resources\\python\\" + fileName);
    file.transferTo(fileDestination);

    SoundContent soundContent = soundContentService.findById(UUID.fromString(soundContentId));

    String recognizedAudioRecord = audioRecognitionService.recognizeAudioRecord(
            fileDestination.getPath(),
            Language.valueOf(soundContent.getSound().getLanguage().toUpperCase()));
    if (recognizedAudioRecord == null) {
        return ResponseEntity.badRequest()
                .body(gson.toJson( src: ""));
    }
    RecognitionResult recognitionResult = stringService
            .findCorrectAndWrongPartInExpectedText(recognizedAudioRecord,
            soundContent.getContentText());
    return ResponseEntity.ok()
            .body(gson.toJson(recognitionResult));
}
```

Рисунок 3.19 – POST метод для розпізнавання мовлення з аудіо файлу

Отож, створено програмний скрипт на мові Python, який приймає шлях до аудіо файлу та код мови, і виводить виявлений текст. Також розроблено сервіс й контролер мовою Java для використання скрипту.

3.6 Висновки

Отже, в межах розділу було обґрунтовано вибір мови програмування в залежності від цілей та середовище розробки. Описано доцільність використання Spring Framework і наведено його переваги. Продемонстровано розробку інтерфейсу програмного користувача та бази даних з використанням потужних бібліотек. Також розроблено програмний модуль для розпізнавання вимови користувача.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення – це процес виконання технічних досліджень для створеного програмного продукту з метою перевірки його відповідності очікуваним вимогам [28]. Методи тестування включають в себе процеси пошуку дефектів і тестування програмних компонентів. Правильно перевірений застосунок забезпечує надійність, безпеку та хорошу продуктивність, що підвищить рівень задоволеності клієнтів і заощадить час у майбутньому.

Тестування чорної скриньки – це вид тестування, у якому тестувальник не знає, як система влаштована всередині. У нього немає доступу до коду або він не вміє його читати. Тому орієнтація тільки на зовнішню поведінку або ТЗ.

Тестування білої скриньки – це підхід до перевірки ПЗ, коли у тестувальника є доступ до коду. Є можливість читати сам програмний код, запустити додаток в режимі налагодження, створити юніт та автотести.

Набрало високої популярності позитивне та негативне тестування. Позитивне виконується виключно із валідними даними, під час дотримання цього підходу під час перевірки вводять лише дані, які очікує програма. Негативне тестування використовують для того, щоб оцінити стабільність та надійність програми під час використання даних, які програма не очікує отримати. Тестувальник програмного забезпечення у негативному підході повинен визначити методи або певний порядок дій за допомогою яких, можна отримати повідомлення про неправильну поведінку користувача. Прикладом негативного тестування може бути випадок коли у поле, яке очікує отримати файл зображення, передати аудіо або відео файл.

Існує модульний підхід до тестування ПЗ, в якому так звані unit (одиниці) або окремі частини програми, можна перевірити окремо на правильну роботу [29]. Ця методика тестування зазвичай виконуються самими розробниками під час процесу розробки програмного продукту, а іноді й самими тестувальниками. Його основною перевагою є швидкість, оскільки кожен модуль тестується незалежно.

Інтеграційним тестування є спеціальний вид тестування, де програмні модулі логічно інтегровані в додатку; це класи репозиторії, контролери та сервіси і вони тестуються, як група поєднуючи два та більше компоненти одночасно. Програмний проєкт повинен складатись з кількох програмних модулів. Головною ціллю цього рівня тестування – виявлення дефектів під час взаємодії між різними програмними модулями, коли вони інтегровані і працюють разом.

Отже, було описано методи тестування програмного забезпечення, які слід буде застосувати під час перевірки додатку. Варто зазначити, що оскільки наявний доступ до програмного коду, потрібно створити власні модульні тести.

4.2 Тестування розробленого програмного додатку

Оскільки під час тестування існує доступ до коду, потрібно створити модульні тести для перевірки основних функцій додатку. Розробники Spring провели дослідження і виявили, що правильне використання інверсії керування (IoC), полегшує модульне та інтеграційне тестування тому, що існування методів для встановлення відповідних конструкторів у класах полегшує їх використання у тесті [30].

Використання ін’єкції залежностей зробить код менш залежним від контейнера. Оскільки під час розробки дотримувались основ архітектури для Spring, отримане розшарування на сервіси, сховища та контролери кодової бази зможе полегшити модульне тестування. Наприклад можна тестувати об’єкти сервісного рівня шляхом заглушення або використовуючи Mock інтерфейси для DAO або репозиторію, не потребуючи доступу до постійних даних.

Використання анотації @SpringBootTest завантажує для програми повний контекст Spring. На відміну від звичайної анотації @SpringBootApplication, анотація тестового фрагмента завантажує лише необхідні для розробника тестові компоненти. І завдяки цьому можна уникнути використання непотрібних Mock елементів.

Необхідно виконати створення модульних тестів для додатку. Головним сервісом для додатку є клас для розпізнавання вимови, тому потрібно створити

тести для перевірки англійської та української вимови. Спершу створюється два аудіо записи для кожної мови й розміщуються в пакеті ресурсів. Файл запису англійської мови містить наступний текст «audio recording could not load content»; для українського запису використано популярні й притаманні лише для української мови слова «інтернаціональний паляниця філіжанка швидкісний». Тести для перевірки розпізнавання двома мовами представлені на рисунку 4.1.

```
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class TestRecognition {

    @Autowired
    AudioRecognitionService audioRecognitionService;

    @Test
    void testEnglishRecognition(){
        String translatedString = audioRecognitionService.recognizeAudioRecord(
                filePathFromSourceRoot: "src/test/resources/englishRecord.wav", Language.ENGLISH);

        //verify if text in audio was recognized
        Assertions.assertTrue(translatedString.contains("in relation"),
                message: "String doesn't contains expected text - in relation");
        Assertions.assertTrue(translatedString.contains("audio recording could not load content"),
                message: "String doesn't contains expected text -audio recording could not load content");
    }

    @Test
    void testUkraineRecognition(){
        String translatedString = audioRecognitionService.recognizeAudioRecord(
                filePathFromSourceRoot: "src/test/resources/ukraineRecord.wav", Language.UKRAINIAN);

        //verify if text in audio was recognized
        Assertions.assertTrue(translatedString
                .equalsIgnoreCase( anotherString: "інтернаціональний паляниця філіжанка швидкісний"));
    }
}
```

Рисунок 4.1 – Тестування сервісу для розпізнавання вимови з аудіо файлу

Створено тести, які перевіряють розроблений клас SoundRepository для збереження звуків в сховищі даних. Для цього використано анотації @TestMethodOrder та @Order, щоб визначити порядок виконання створених тестів. Спершу перевірено створення сутностей в розробленому репозиторії, а на другому етапі видалення очікуємо повідомлення про помилку для цього застосовано метод assertThrows (рис. 4.2).

```java
@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 :Sound = 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 :UUID = createdSound.getId();
        soundRepository.delete(soundId);
        Assertions.assertThrows(EmptyResultDataAccessException.class,
                () -> soundRepository.findById(soundId));
    }
}
```

Рисунок 4.2 – Тестування сховища для зберігання сутності звуку

Створено тести для перевірки сервісу, який виконує алгоритм для знаходження правильно й неправильного вимовленого тексту відповідно до навчального матеріалу (рис. 4.3).

```java
class StringTest {

    private final StringService stringService = new StringService();
    private final String firstTestString = "this is first text example";
    private final String secondTestString = "this is next text example";

    @Test
    void testCheckAlgorithmForFindingSubTextInText() {
        RecognitionResult result = stringService
                .findCorrectAndWrongPartInExpectedText(firstTestString, secondTestString);
        Assertions.assertTrue(result.getCorrectText().trim().equalsIgnoreCase( anotherString: "this is text example"));
    }

    @Test
    void testCheckAlgorithmForFindingWrongText() {
        RecognitionResult result = stringService
                .findCorrectAndWrongPartInExpectedText(firstTestString, secondTestString);
        Assertions.assertEquals( expected: "first", result.getWrongText().trim());
    }

    @Test
    void testIfObjectContainsFullText(){
        RecognitionResult result = stringService
                .findCorrectAndWrongPartInExpectedText(firstTestString, secondTestString);
        Assertions.assertTrue(result.getFullText().equalsIgnoreCase( anotherString: "this is next text example"));
    }
}
```

Рисунок 4.3 – Тестування алгоритму для перевірки тексту

Необхідно перевірити всі розроблені тести. Успішний результат виконання модульних тестів представлений на рисунку 4.4.

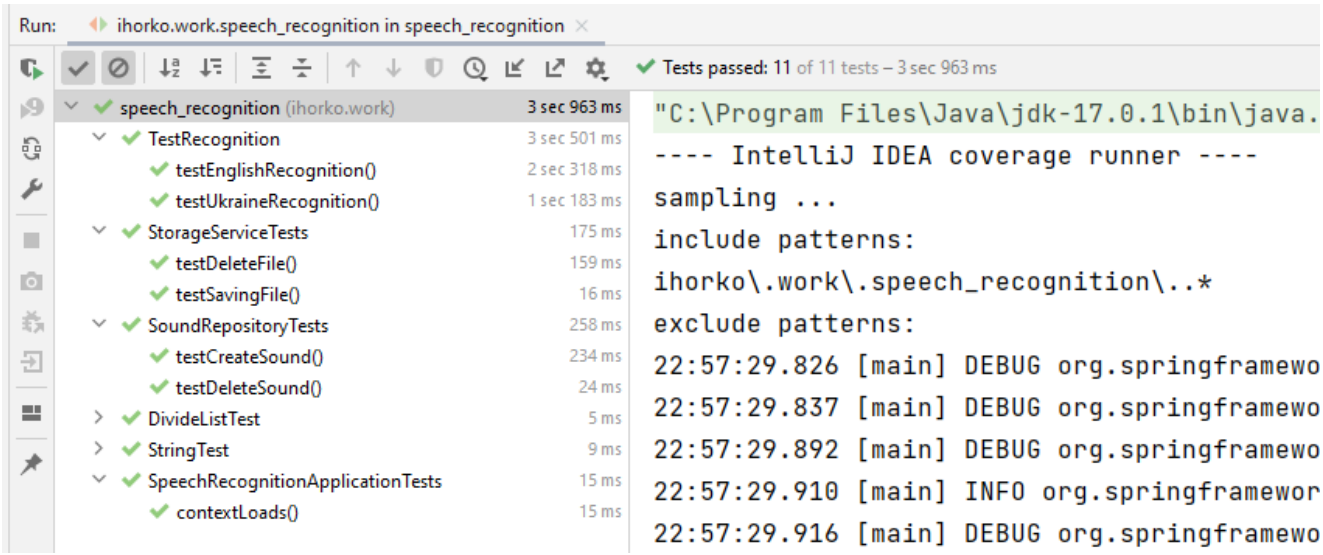

Рисунок 4.4 – Результат виконання тестів

Проаналізувавши даний тестовий запуск, варто зробити висновок, що основні методи виконуються як і очікувалось, а час виконання модульних тестів не займає багато часу. Всі тести виконались за 4 секунди, що свідчить про правильний підхід до їхньої розробки. Для перевірки основних функцій програмного додатку створено тест-кейси.

Тест-кейс №1 – Створення навчального матеріалу

1. Відкрити веб-додаток;

2. Перейти на сторінку з матеріалами для звуку;

3. Заповнити усі необхідні поля;

4. Натиснути клавішу зберегти;

5. Відкрити створений навчальний матеріал з таблиці.

Успішним результатом виконання тесту є відкрита сторінка з полями, що відповідають інформації, які було введено раніше, й відображення картинки та аудіо файлу.

Для створення навчального матеріалу використовується створений звук «CH» та вводяться слова, які можна використати для вивчення звуку «Chocolate

champion chance». Також завантажуються зображення з правильною артикуляцією для вимови цього звуку й додаємо аудіо файл із правильною вимовою. Результат виконання першого тест кейсу представлений на рисунку 4.5.

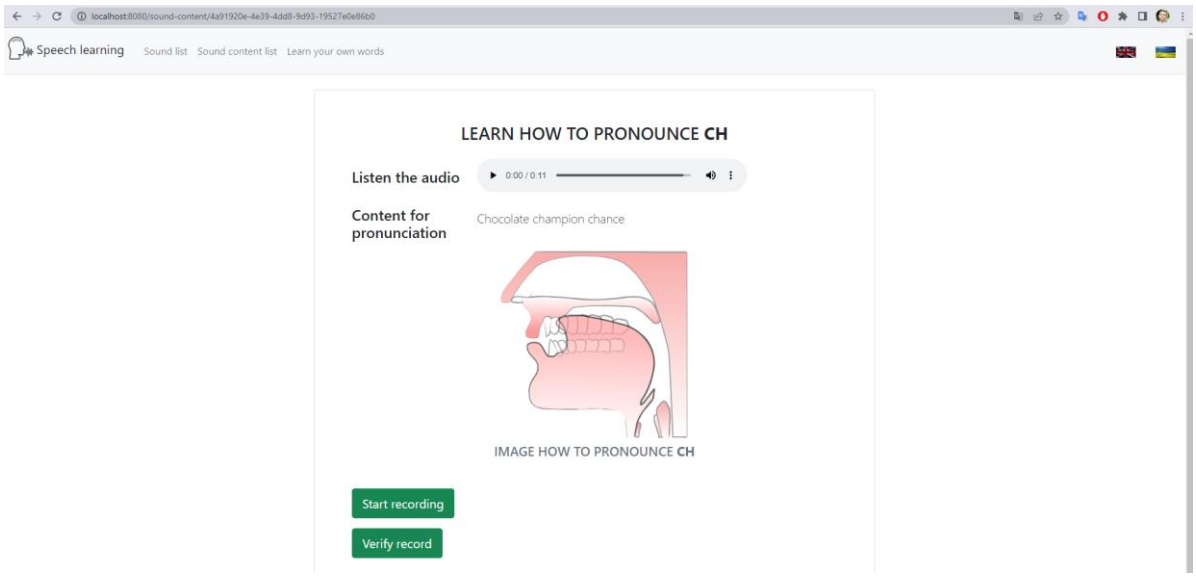

Рисунок 4.5 – Результат створення матеріалу для вивчення

Актуальний результат виконання тест-кейсу відповідає очікуваному, а це означає що він успішно пройдений.

Тест-кейс №2 – Перевірка інтернаціоналізації головної сторінки українською та англійською мовами.

1. Відкрити веб-додаток;

2. Перейти на головну сторінку;

3. Перекласти українською мовою натиснувши на український прапор на навігаційній панелі;

4. Перекласти додаток англійською мовою на англійський прапор на навігаційній панелі.

Очікуваним результатом для тест-кейсу №2 є відображення сторінки двома мовами, використовуючи клавіші, які зображуються на навігаційній панелі.

Для тест-кейсу було обрано головну сторінку, оскільки саме через неї будуть розпочинатись усі операції, і тому необхідно, щоб вона відображалась правильно. Результати перевірки інтернаціоналізації представлені на рисунках 4.6 та 4.7

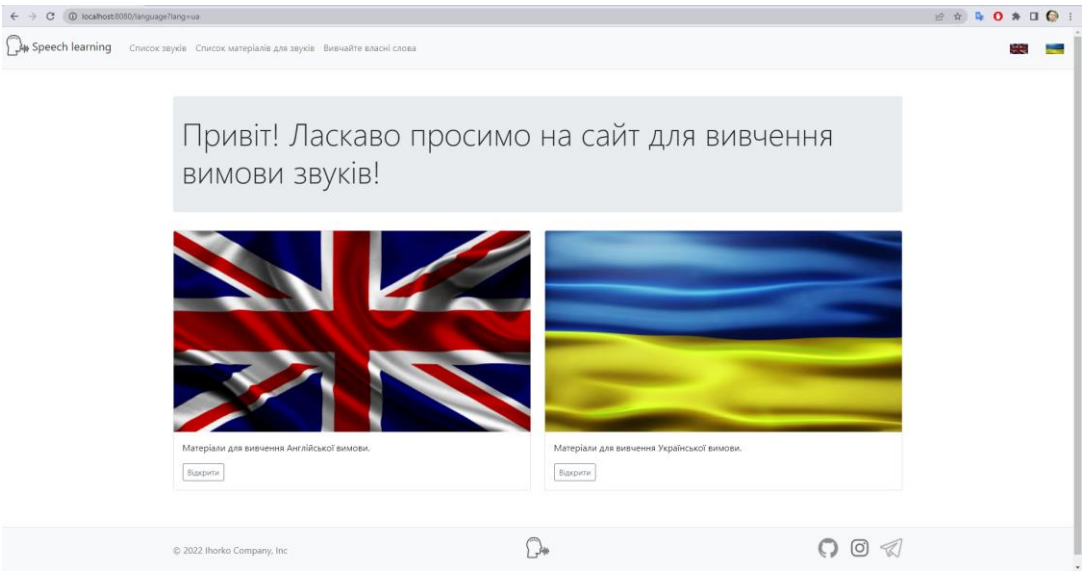

Рисунок 4.6 – Результат перевірки інтернаціоналізації українською мовою

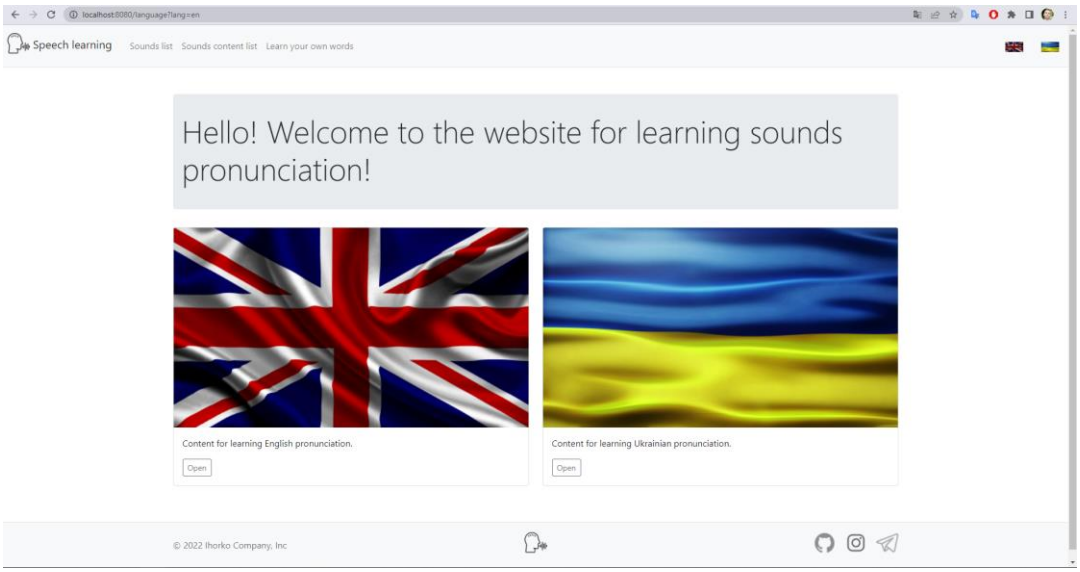

Рисунок 4.7 – Результат перевірки інтернаціоналізації англійською мовою

Під час виконання тест-кейсу №2 було отримано очікуваний результат. Головна сторінка є доступною двома мовами: англійською та українською, використовуючи клавіші на навігаційній панелі.

Тест-кейс №3 – Перевірка розпізнавання англійської вимови відповідно до навчального матеріалу.

1. Відкрити веб-додаток;
2. Перейти на навчальний матеріал для вивчення англійської вимови;
3. Вимовити навчальний контент та зробити помилку у вимові.

Успішним результатом виконання є отримання 3-ох повідомлень: правильно вимовленим текстом, який є в умові навчального матеріалу, повинен бути з зеленим ярликом; неправильно вимовлений текст повинен бути разом із червоним ярликом та весь розпізнаний текст для інформації з блакитним підписом. Також користувач може прослухати власний запис і завантажити його на власний комп'ютер.

Для виконання тестового випадку було обрано навчальний матеріал із першого тест-кейсу. Правильно вимовлено слова «chocolate champion» та додатково сказано «recognised text». Результат виконання представлений на рисунку 4.8.

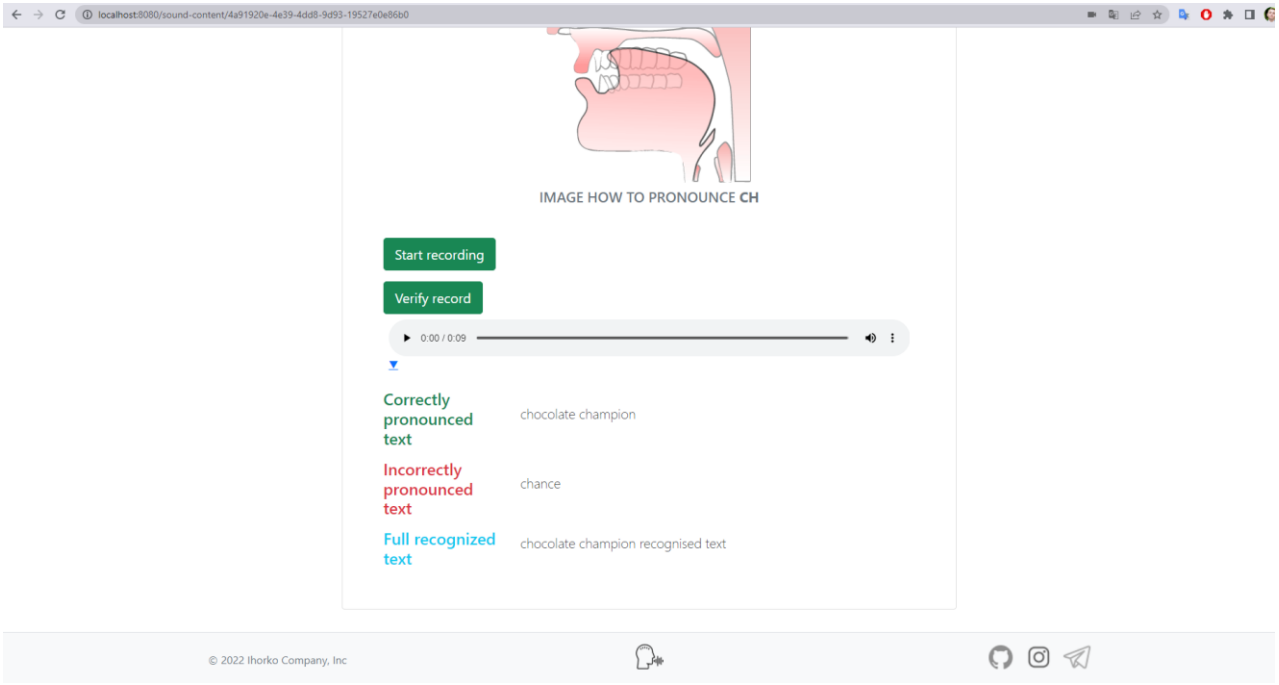

Рисунок 4.8 – Розпізнавання англійської вимови відповідно до навчального матеріалу

В результаті виконання тестового випадку було виведено три блоки тексту різними кольорами відповідно до аудіо файлу, який надіслав користувач. Текст розпізнано як і очікувалось. Також користувач має змогу прослухати власний аудіо запис та завантажити його.

Тест-кейс №4 – Перевірка розпізнавання української вимови відповідно до навчального матеріалу.

1. Відкрити веб-додаток;

2. Перейти на навчальний матеріал для вивчення української вимови;

3. Вимовити навчальний контент та зробити помилку у вимові.

Успішним результатом виконання даного тестового випадку є подібним до попереднього, а саме: отримання трьох різних повідомлень та можливість керувати власним аудіо записом користувачеві.

Для виконання тестового випадку було створено новий навчальний матеріал і новий звук «й» із набором слів, які необхідно вимовити «Їжачок йод їжачиха». Правильно вимовлено «їжачок» та додатково сказано притаманні для української мови слова «й коліжанка лежанка Інтернаціональний кава паляниця». Результат виконання представлений на рисунку 4.9.

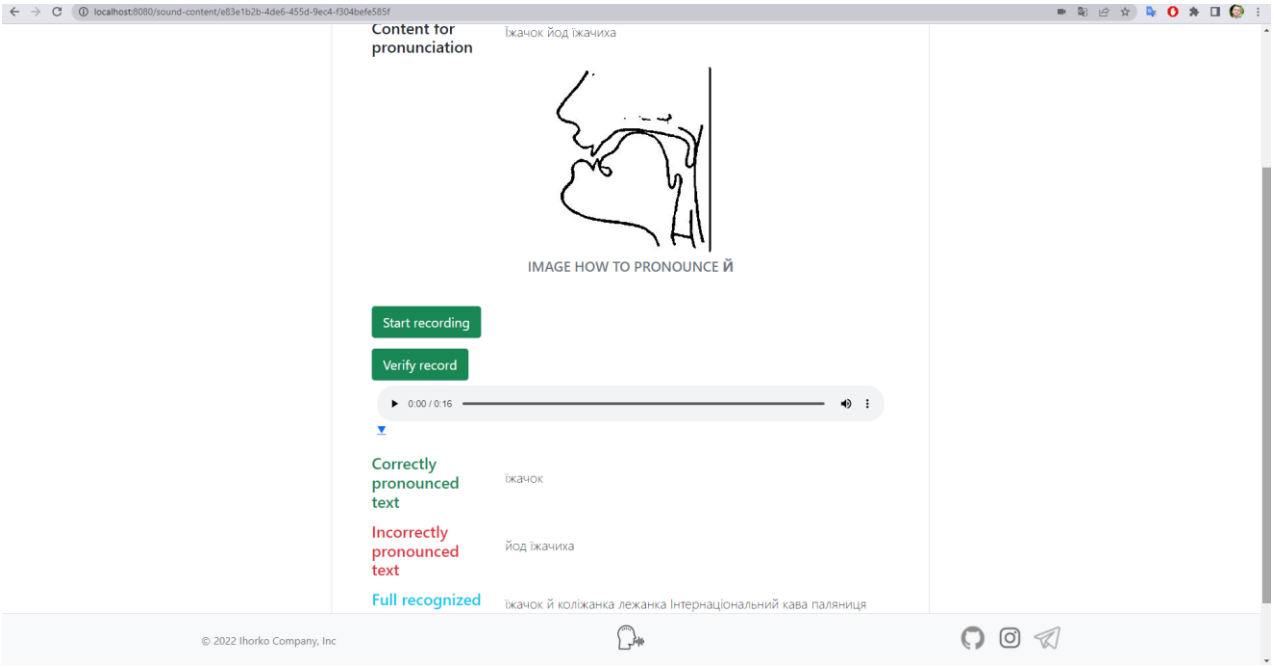

Рисунок 4.9 – Результат розпізнавання української вимови

В результаті виконання тестового випадку було виведено три тексти із підписами різними кольорами відповідно до аудіо файлу, який надіслав користувач. Також користувач має змогу керувати аудіо записом.

Отже, було розроблено модульні тести використовуючи Spring. Також для додатку було проведено тестовий запуск тестів і отримано успішні результати.

Створено тест-кейси для перевірки збереження навчального матеріалу, інтернаціоналізації та розпізнавання вимови.

4.3 Розробка інструкції користувача

Користуватись додатком можна за допомогою веб-браузера, тому спершу кожен користувач повинен його встановити на свій персональний комп’ютер чи інший пристрій. Після відкриття додатку одразу зустрічає головна сторінка. Її зображення представлене на рисунку 4.10.

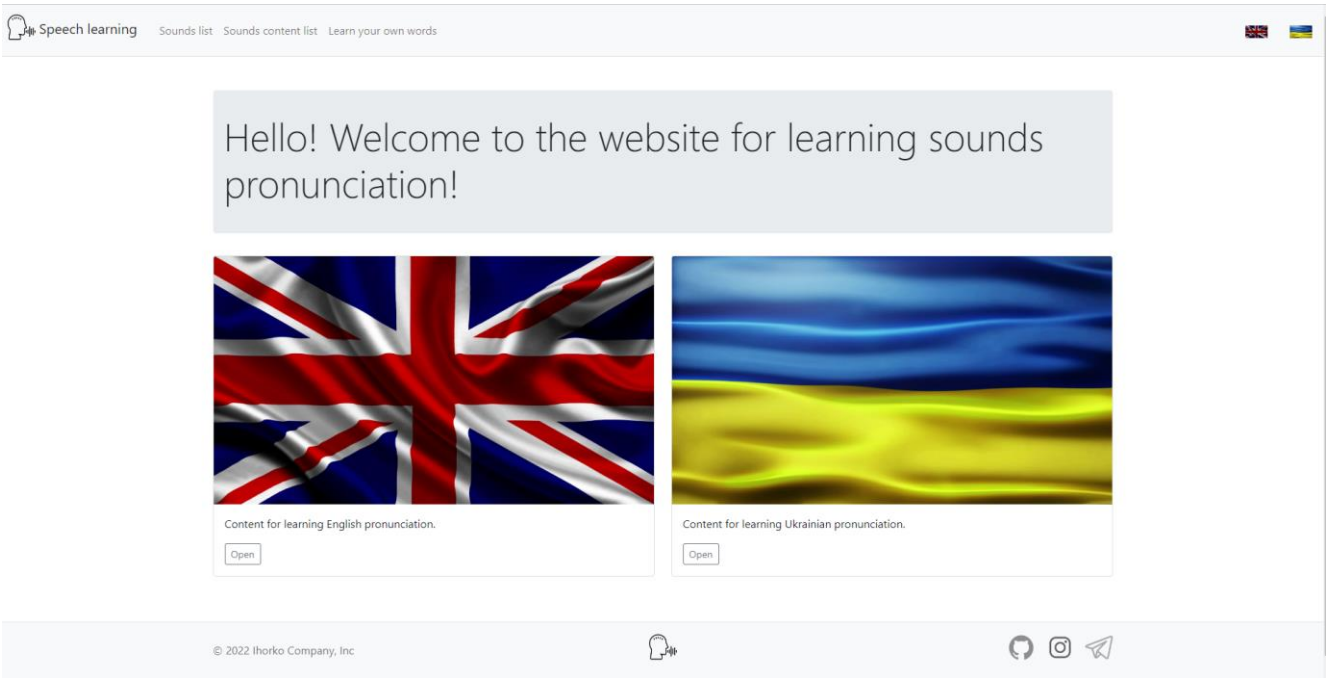

Рисунок 4.10 – Головна сторінка

Оскільки по замовчуванню додаток відображає інформацію англійською мовою, виконано переклад на українську за допомогою клавіші яка має вигляд українського прапору на навігаційній панелі (рисунок 4.11).

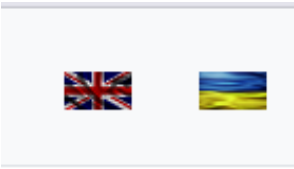

Рисунок 4.11 – Клавіші для зміни мови

Після чого отримано успішне відображення додатку українською мовою (рисунок 4.12).

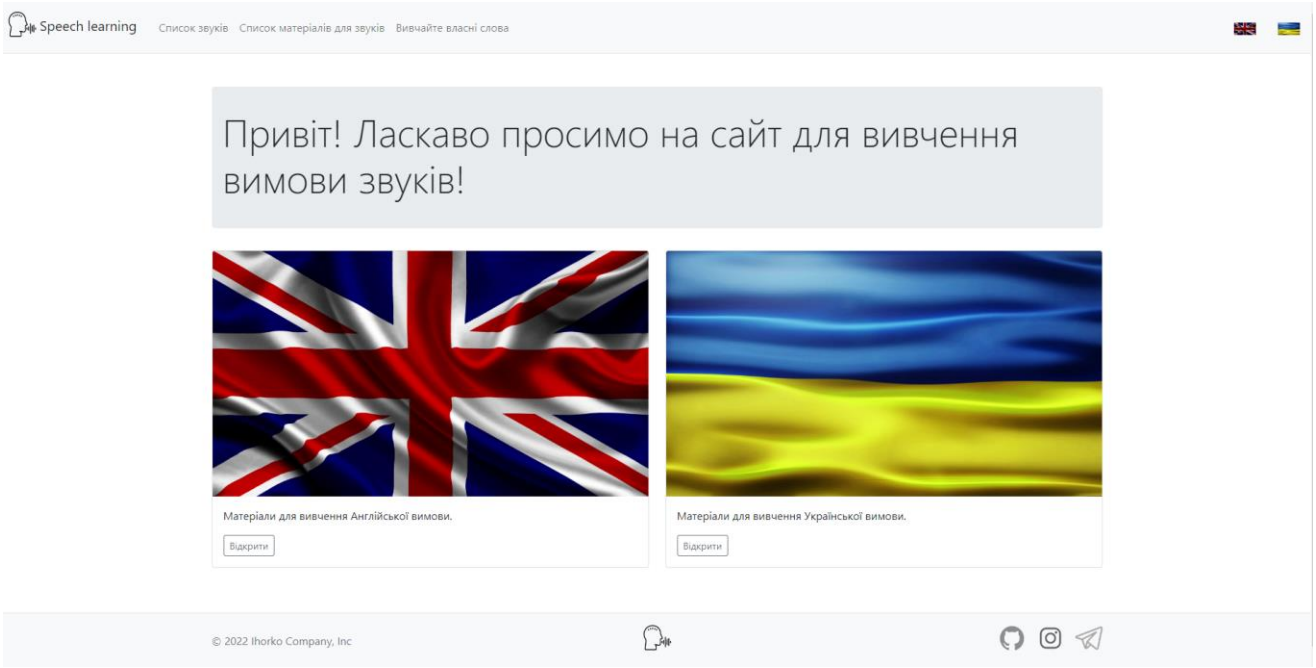

Рисунок 4.12 – Вигляд перекладеної сторінки

Далі стає доступним список звуків, виключно для української мови. Сторінка містить короткий опис й клавішу для створення нових звуків (рисунок 4.13).

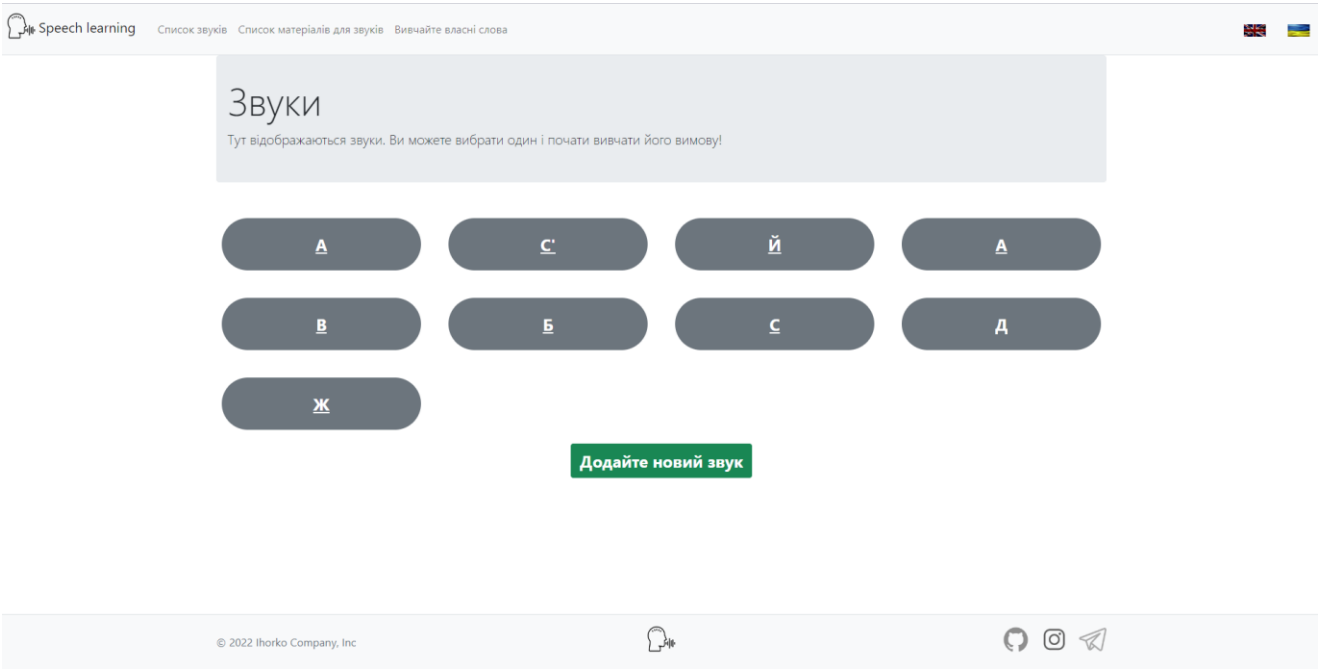

Рисунок 4.13 – Сторінка відображення звуків

Далі відкрито форму для додавання звуків (рисунок 4.14). На цій сторінці є можливість ввести саму назву звуку й обрати мову якій він належить.

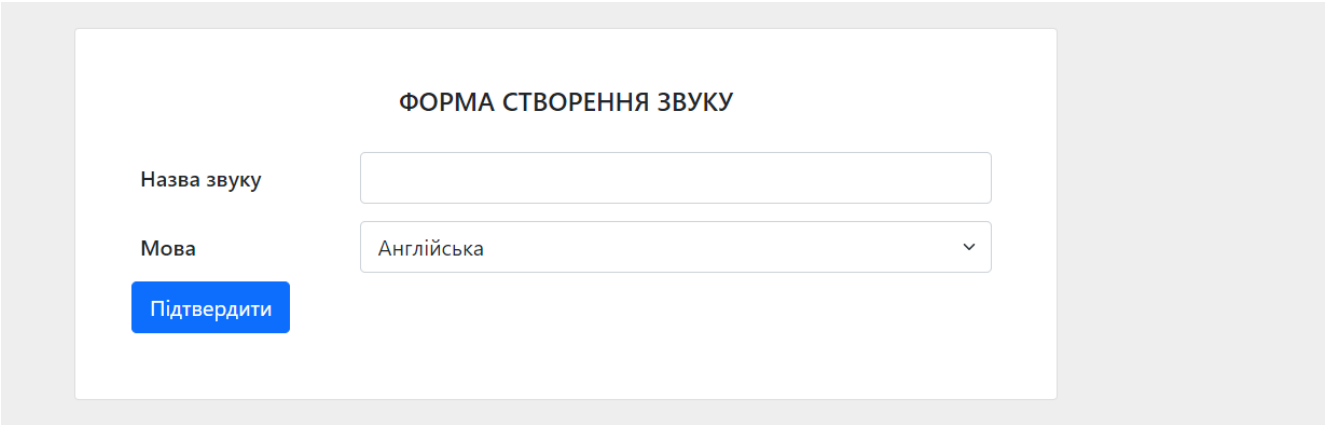

Рисунок 4.14 – Форма створення звуку

Після того, як користувач обрав необхідний йому звук він переходить на сторінку із матеріалами. На цій сторінці відображено таблицю з головною інформацією про матеріал, а саме: текст контенту, його тип, аудіо файл з вимовою й посилання на цей матеріал (рисунок 4.15).

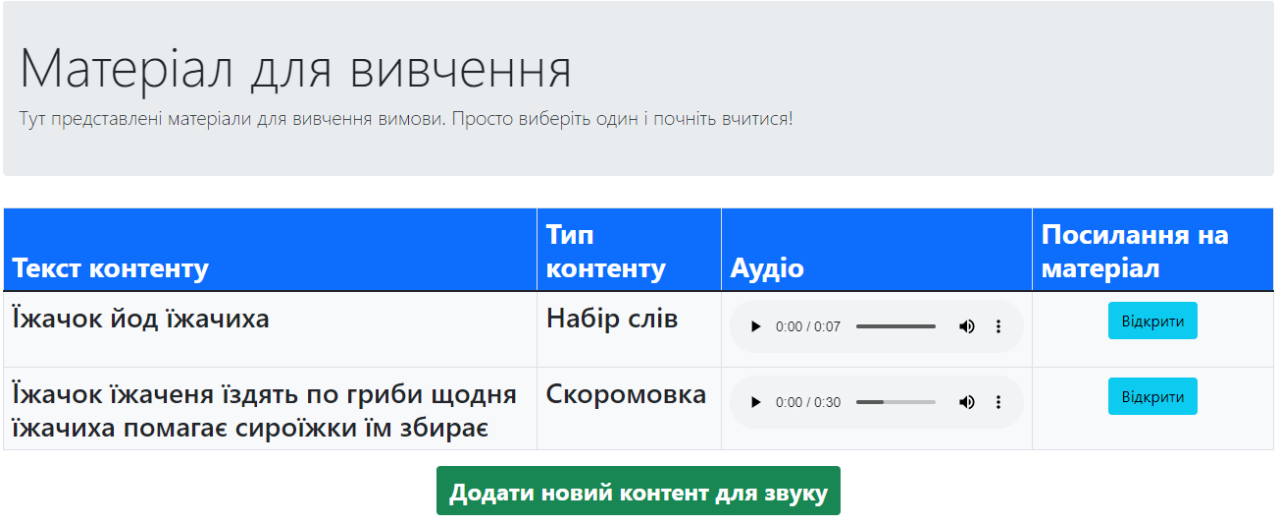

Рисунок 4.15 – Сторінка матеріалів для вивчення

Форма, за допомогою якої користувач може створити контенту для звуку, представлена на рисунку 4.16. На даній сторінці користувач обов’язково повинен ввести поле з текстом, який в майбутньому користувач буде вимовляти. Для

додаткової інформації потрібно заповнити поле з типом наприклад скоромовка, чи звичайний набір слів. Обрати звук, для якого необхідно додати матеріал й завантажити два файли аудіо і зображення, щоб користувачу було зрозуміло, як саме слід вимовляти ці звуки. Варто зазначити, що в додаткові встановлено обмеження на максимальний розмір файлу – 10 МБ.

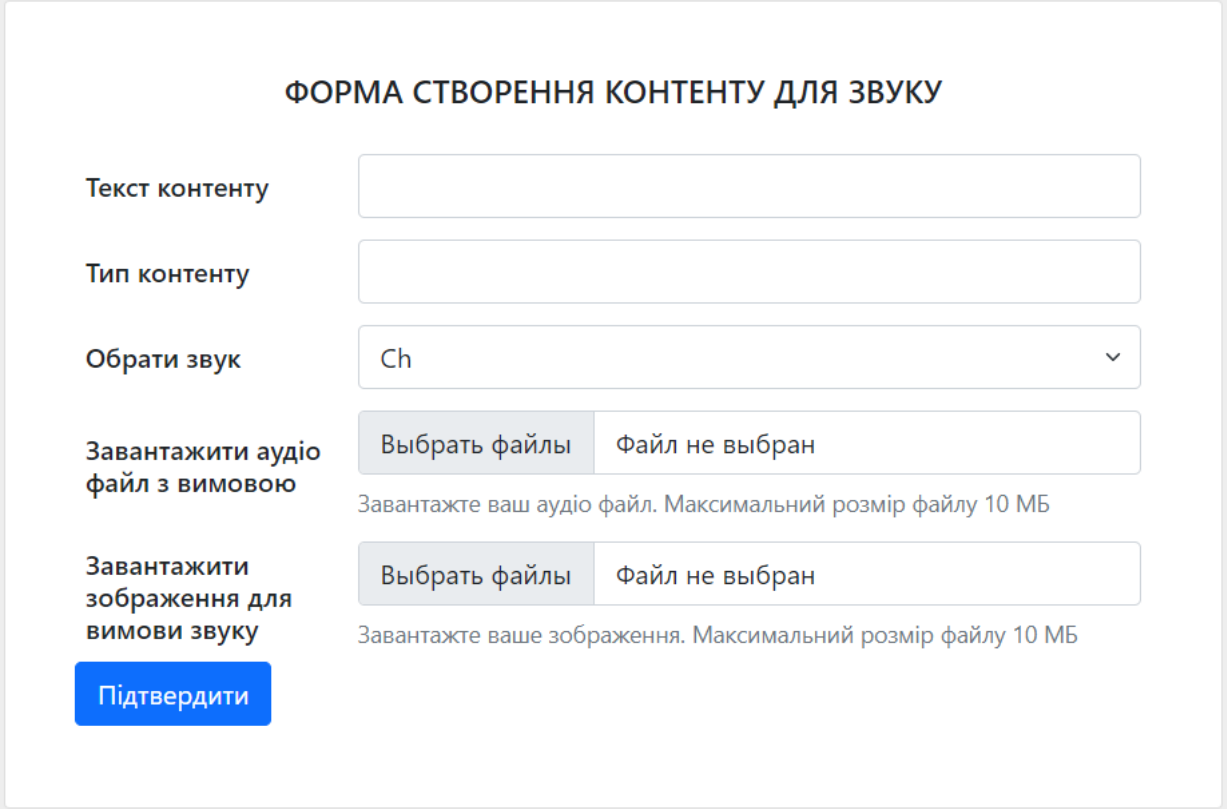

Рисунок 4.16 – Форма створення контенту для звуку

Після відкриття сторінки з матеріалом для вивчення, відображено сторінку з інформацією, яку заповнено, а саме: сам текст для контенту, який користувач має вимовити, зображення й аудіо для прикладу як вимовляти цей звук. Щоб розпочати запис необхідно натиснути кнопку «Почати запис» й її колір зміниться на червоний, щоб зупинити потрібно ще раз на неї натиснути. Після того як користувач зупинить запис аудіо його вимова відобразиться в аудіо плеєрі, який з’явиться нижче, біля якого буде синя клавіша для завантаження його на локальний комп’ютер. Після того як користувач натисне клавішу «Перевірити запис», відобразиться три текстових блоки з правильно й неправильно вимовленим текстом

й увесь розпізнаний текст, щоб користувачу було легше орієнтуватись, коли саме він дотримався помилки. Матеріал для вивчення й приклад результатів вимови тексту представлені на рисунку 4.17.

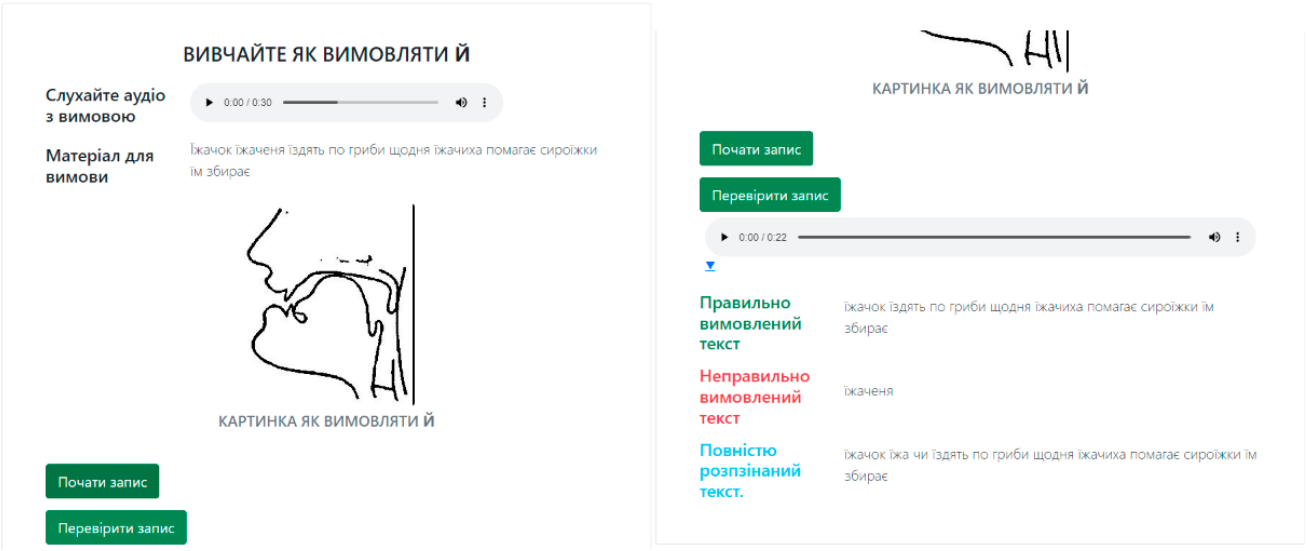

Рисунок 4.17 – Сторінка матеріалу для вивчення

Оскільки не завжди є можливість й час створити повноцінний матеріал для вивчення з зображенням і аудіо, було розроблено сторінку для якої користувач лише обирає потрібну мову і вводить власний текст (рисунок 4.18).

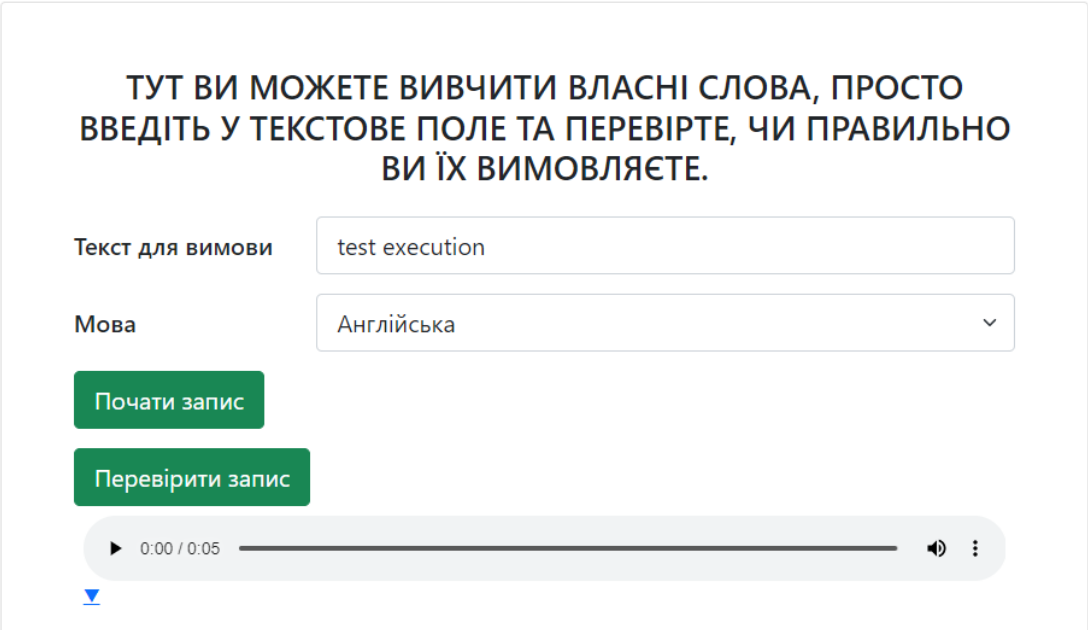

Рисунок 4.18 – Сторінка для вивчення власних слів

Варто відмітити, що було розроблено інструкцію людям, які мали б бажання детальніше розібратись з додатком і його програмним кодом. Інструкція розміщена в Readme.MD файлі й доступна для пошуку на сервісі GitHub. Інструкція містить: короткий опис, необхідні інсталяції перед початком роботи, безпосередні конфігурації в самому додатку й зображення програми. Приклад опису й результат зображені на рисунку 4.19 і 4.20.

```
## Починаємо

1. Потрібно мати встановлені Java 11+, PostgreSql та Python 3.9+
2. Налаштувати змінні середовища (якщо користувач Windows)
3. Встановити наступні бібліотеки Python: google-cloud-speech, PyAudio, pyttsx3, SpeechRecognition
4. Склонувати репозиторій
```sh
   git clone https://github.com/Ascomos21/speech_recognition
   ```
5. Для успішної локалізації мовою яка використовує кирилицю налаштувати файл properties на charset - UTF-8
6. В класі ihorko/work/speech_recognition/db/util/HibernateConfig.java в методі DataSource ввести назву своєї бази
   даних, пароль та ім'я користувача

```sh
dataSource.setUrl("jdbc:postgresql://localhost:5432/{db-name}");
      dataSource.setUsername("{username}");
      dataSource.setPassword("{password}");
```

Зображення

Головна сторінка

<img src="git_files/BasePage.png" alt="base page" width="633" height="322">
```

Рисунок 4.19 – Приклад опису інструкції

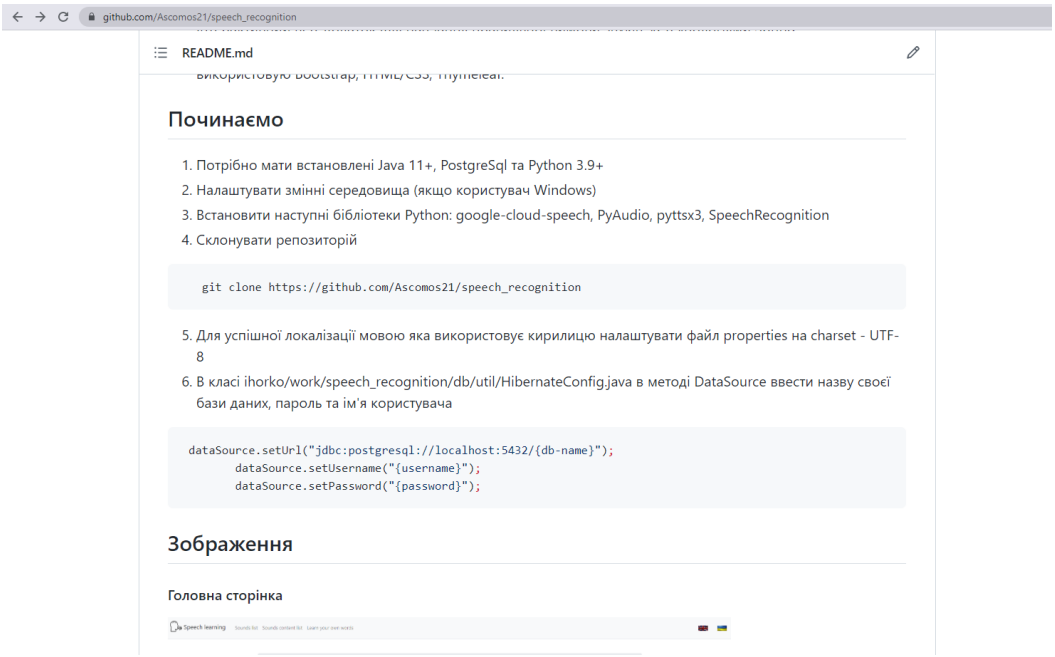

Рисунок 4.20 – Відображення інструкції на сервісі GitHub

Отже, під час виконання даного розділу було розроблено інструкцію користувача й описано основні функції додатку з доданими зображення. Крім того, було створену інструкцію на сервісі GitHub для людей, які хотіли б детальніше ознайомитись з програмним кодом

4.4 Вимоги до комп’ютера для використання додатку

Оскільки розробленим програмним додатком можна користуватись за допомогою веб-браузера, більшість вимог підлягають саме під них. Сьогодні можна встановити браузер на різні операційні системи та пристрої як мобільні так і комп’ютерні. Також для запису та прослуховування власного аудіо файлу необхідно мати мікрофон з пристроєм виведення звуку. Тому створено мінімальну та рекомендовану конфігурацію до ПК у таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1,2 ГГц
Об’єм оперативної пам’яті	1 ГБ для 32-розрядної системи і 2 ГБ для 64-розрядної системи
Місце на жорсткому диску	1 ГБ
Графічний пристрій	Інтегрований графічний пристрій, сумісний з DirectX9
Операційна система	Windows 10
Фізичний пристрій	Мікрофон та динаміки або навушники з мікрофоном

Таблиця 4.2 – Рекомендована конфігурація

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 2,4 ГГц
Об’єм оперативної пам’яті	2 ГБ для 32-розрядної системи і 4 ГБ для 64-розрядної системи
Місце на жорсткому диску	1 ГБ
Графічний пристрій	Інтегрований графічний пристрій, сумісний з DirectX10
Операційна система	Windows 10
Фізичний пристрій	Мікрофон та динаміки або навушники з мікрофоном

Отже, було описано вимоги: до персонального комп’ютера, які пристрої потрібно використовувати для успішного запуску додатку, й створено дві таблиці з мінімальною та рекомендованою конфігураціями.

4.5 Висновки

В межах четвертого розділу проаналізовано методи тестування програмного додатку та обрано методи для тестування додатку. Оскільки наявний доступ до коду, то було обрано метод білої скриньки та для перевірки головних відображення певних текстових елементів використано метод чорної скриньки.

Під час тестування застосунку було створено модульні тести, а використовуючи Spring Framework описано основні приклади модульних тестів, й наведено приклад тестового запуску. Використовуючи створені тестові випадки, було перевірено інтернаціоналізацію головної сторінки, збереження навчальних матеріалів й розпізнавання вимови для української та англійської мови. Усі тести було пройдено успішно.

Розроблено інструкцію користувача, визначено вимоги для персонального комп’ютера користувача та перелік інструментів, які він повинен мати для успішного використання додатку.

ВИСНОВКИ

Під час виконання дипломної роботи було розроблено веб інтерактивний додаток під назвою «SpeechLearning» для вивчення правильної вимови звуків, використовуючи Spring технології. Створений додаток призначений для підвищення ефективності вичення мови шляхом визначення правильності вимови звуків. Оскільки застосунок є інтерактивним, користувач може взаємодіяти з ним створюючи нові матеріали для вивчення, публікуючи текст, аудіо та зображення з можливістю переглядати цей вміст та маючи змогу записати власне аудіо й перевірити чи правильно було вимовлено слова відповідно до навчальних матеріалів.

Було проведено дослідження сфери розпізнавання мовлення, під час виконання якого визначено цілі використання, й виконано порівняльний аналіз додатків для вивчення мов. Обрано методи для розв’язання поставлених задач й поставлено основні задачі для розробки додатку.

Розроблено структуру графічного інтерфейсу для віко додатку. Розроблено блок-схеми основних алгоритмів, а саме: алгоритму визначення правильно вимовленого тексту відповідно до навчальних матеріалів, алгоритму роботи додатку для розпізнавання вимови з аудіо файлу користувача, алгоритму зберігання в сховище матеріалу для вивчення.

Під час виконання розробки програмних компонентів було обґрунтовано вибір мови й середовища для створення додатку. Наведено приклади використання Spring технологій в межах розробки. Створено адаптивний графічний інтерфейс користувача за допомогою Thymeleaf та Bootstrap. Розроблено базу за допомогою ORM системи Hibernate. Описано програмну реалізацію модуля для розпізнавання вимови з аудіо файлу.

Проведено дослідження основних методів тестування ПЗ й обрано методи тестування додатку. Виконано тестування на основі обраних методів й реалізовано модульні тести для перевірки методів програми. Розроблено інстуркцію користувача та визначено системні вимоги до обладнання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cake: Lesson updates every day [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=me.mycake&hl=en&gl=US.

2. Lingvist: Learn Languages Fast [Електронний ресурс] // Lingvist Technologies OÜ. – 2022. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=io.lingvist.android&hl=uk&gl=US.

3. Lutkevich B. Speech recognition [Електронний ресурс] / Ben Lutkevich // TechTarget. – 2021. – Режим доступу до ресурсу: https://www.techtarget.com/searchcustomerexperience/definition/speech-recognition.

4. Кучерявий І. В. Дослідження технологій автоматизованої перевірки завдань для освітніх платформ [Електронний ресурс] / І. В. Кучерявий, О. В. Романюк // Міжнародна науково-практична Інтернет-конференція 9-10 листопада 2021р. Вінницький національний технічний університет. – 2021. – Режим доступу до ресурсу: https://ir.lib.vntu.edu.ua/handle/123456789/34861.

5. Кучерявий І. В. Оптимізація запитів до баз даних [Електронний ресурс] / І. В. Кучерявий, А. І. Веренько, О. В. Романюк // XLIX науково-технічна конференція підрозділів ВНТУ, Вінниця, 27-28 квітня 2020 р. – 2020. – Режим доступу до ресурсу: https://ir.lib.vntu.edu.ua/handle/123456789/29420.

6. Кучерявий І. В. Особливості застосування ORM технологій при роботі з реляційними базами даних / І. В. Кучерявий, О. В. Романюк // XXI Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів «СТАН, ДОСЯГНЕННЯ ТА ПЕРСПЕКТИВИ ІНФОРМАЦІЙНИХ СИСТЕМ І ТЕХНОЛОГІЙ», Одеса. – 2022. – Режим доступу до ресурсу: https://www.onaft.edu.ua/download/konfi/2022/Conference_abstract-IT-21-22-04-22.pdf.

7. Кучерявий І. В. Особливості використання фреймворку Spring для розробки веб-інтерактивних додатків [Електронний ресурс] / І. В. Кучерявий, О. В. Романюк // LI Науково-технічна конференція факультету інформаційних технологій та комп’ютерної інженерії, Вінниця: ВНТУ, 31 травня 2022. – 2022. –

Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/16115.

8. Medical Voice Recognition Software: How Does It Work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.deepscribe.ai/resources/medical-voice-recognition-software-how-does-it-work.

9. Influent [Електронний ресурс] – Режим доступу до ресурсу: https://store.steampowered.com/app/274980/Influent/.

10. Speech Tutor [Електронний ресурс] – Режим доступу до ресурсу: https://speechworld.web.app/#/.

11. Грофф Д. SQL. Полное руководство / Джеймс Грофф. – Чикаго, 2015. – 957 с. – (Вильямс).

12. Speech-to-Text [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/speech-to-text.

13. Urmann D. What is Web Interface? [Електронний ресурс] / Daniel Urmann // diib. – 2020. – Режим доступу до ресурсу: https://diib.com/learn/web-interface/.

14. Learn JavaScript Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/javascript-tutorial.

15. Лутц М. Learning Python / Марк Лутц. – Кембридж: O'reilly, 2011. – 1280 с. – (4).

16. Java: Learn Java in One Day and Learn It Well. Java for Beginners with Hands-on Project. / Джемі Чан., 2016. – 225 с.

17. Ефимов А. А. IntelliJ IDEA. Профессиональное программирование на Java / А. А. Ефимов., 2005. – 800 с.

18. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship / Robert Cecil Martin., 2009. – 431 с.

19. Graig W. Spring in Action 5.0 / Walls Graig. – Shelter Island: Manning, 2019. – 521 с. – (5).

20. Crusovean L. Intro to Inversion of Control and Dependency Injection with Spring [Електронний ресурс] / Loredana Crusovean // Baeldug. – 2022. – Режим

доступу до ресурсу: https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring.

21. Spring Dependency Injection [Електронний ресурс] // Baeldug – Режим доступу до ресурсу: https://www.baeldung.com/spring-dependency-injection.

22. Bodnar J. Spring Boot @Repository [Електронний ресурс] / Jan Bodnar // zetcode. – 2022. – Режим доступу до ресурсу: https://cutt.ly/mHG6R2i.

23. Paraschiv E. Introduction to Spring Data JPA [Електронний ресурс] / Eugen Paraschiv // Baeldug. – 2022. – Режим доступу до ресурсу: https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa.

24. Build fast, responsive sites with Bootstrap [Електронний ресурс] // Bootstrap – Режим доступу до ресурсу: https://getbootstrap.com/.

25. Thymeleaf [Електронний ресурс] // The Thymeleaf Team. – 2022. – Режим доступу до ресурсу: https://www.thymeleaf.org/.

26. Романюк О.Н. Організація баз даних і знань / О.Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2003. – 123 с. – ISBN 966-641-081-8.

27. PgAdmin [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.pgadmin.org/.

28. Hamilton T. What is Software Testing? Definition, Basics & Types in Software Engineering [Електронний ресурс] / Thomas Hamilton // guru99. – 2022. – Режим доступу до ресурсу: https://www.guru99.com/software-testing-introduction-importance.html.

29. Hamilton T. Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE [Електронний ресурс] / Thomas Hamilton // guru99. – 2022. – Режим доступу до ресурсу: https://www.guru99.com/unit-testing-guide.html.

30. Testing [Електронний ресурс] // Spring 2022 – Режим доступу до ресурсу: https://docs.spring.io/spring-ramework/docs/current/reference/html/testing.html#testing-introduction.

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., проф. Романюк О.Н.
_____
31 березня 2022 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного забезпечення для вивчення правильної вимови звуків» за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доцент О.В.Романюк
31 березня 2022 р.
Виконав:
студент гр. ЗПІ-18б І.В. Кучерявий
31 березня 2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного забезпечення для вивчення правильної вимови звуків».

Галузь застосування – програмні додатки для вивчення правильної вимови.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №66 від 24 березня 2022 р. ректора по ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою бакалаврської роботи є підвищення ефективності вивчення іноземної мови шляхом розробки додатку для вивчення правильної вимови звуків, у якому реалізовано удосконалені алгоритми розпізнавання вимови з аудіо-файлу користувача та виявлення правильних і неправильних фрагментів вимовленого тексту.

Призначення роботи – розробка та програмна реалізація додатку для визначення правильної вимови звуків при вивченні іноземних мов.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких створюватиметься БДР.

1. Java: Learn Java in One Day and Learn It Well. Java for Beginners with Hands-on Project. / Джемі Чан., 2016. – 225 с.

2. Graig W. Spring in Action 5.0 / Walls Graig. – Shelter Island: Manning, 2019. – 521 с. – (5).

3. Speech-to-Text [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/speech-to-text.

4. Романюк О.Н. Організація баз даних і знань / О.Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2003. – 123 с. – ISBN 966-641-081-8.

5. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship / Robert Cecil Martin., 2009. – 431 с.

5. Технічні вимоги

Модель розробки – принцип розробки ПЗ – Inversion of Control; патерни проектування програмного забезпечення: Dependency Injection, Model View Controller; база даних – PostgreSQL; контейнер для інверсії контролю – Spring; ORM система – Hibernate; методи розпізнавання вимови – Google API; вхідні дані – база даних, яка містить матеріали для вивчення; аудіо файл з вимовленим текстом користувача; вихідні дані – розпізнаний текст, який вимовив користувач й збережені власноруч навчальні матеріали; середовище розробки – InteliJ IDEA; мова програмування – Java, Python.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

1. Пояснювальна записка до БДР;
2. Технічне завдання;
3. 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 бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження галузі розпізнавання мовлення і вибір методів для вирішення поставлених задач дослідження	26.03.2022 - 09.04.2022	Вик.
2	Розробка структури інтерфейсу та алгоритмів програмного додатку	10.04.2022 – 28.04.2022	Вик.
3	Обґрунтування вибору мови та середовища розробки ПЗ	29.04.2022 - 03.05.2022	Вик.
4	Програмна реалізація додатку	04.05.2022 - 17.05.2022	Вик.
5	Розробка бази даних із використанням Hibernate	18.05.2022 - 22.05.2022	Вик.
6	Тестування програмного додатку	23.05.2022 - 27.05.2022	Вик.
7	Оформлення матеріалів до захисту БДР	28.05.2022 - 10.06.2022	Вик.

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка програмного забезпечення для вивчення правильної вимови звуків»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Романюк О. В.

Оригінальність	97,5
Схожість	2,5

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.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Кучерявий І.В.

Керівник роботи _____ Романюк О. В.

Додаток В. Лістинг програми

SpeechRecognitionApplication.java

```
package ihorko.work.speech_recognition;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication
@EnableJpaRepositories
public class SpeechRecognitionApplication {

   public static void main(String[] args) {
      SpringApplication.run(SpeechRecognitionApplication.class, args);
   }
}
```

StringService.java

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

import ihorko.work.speech_recognition.common.RecognitionResult;
import ihorko.work.speech_recognition.common.StringSearch;
import org.springframework.stereotype.Service;

import java.util.Locale;

@Service
public class StringService implements IStringService {

    private static final StringSearch STRING_SEARCH = new StringSearch();

    public RecognitionResult findCorrectAndWrongPartInExpectedText(String
expectedResult, String result) {
        String testResult = (" " + result + " ").toLowerCase(Locale.ROOT);
        String[] values = expectedResult.toLowerCase(Locale.ROOT).split(" ");

        StringBuilder correctText = new StringBuilder();
        StringBuilder wrongText = new StringBuilder();

        for (String pattern : values) {
            if (STRING_SEARCH.search(" " + pattern + " ", testResult)) {
                correctText.append(pattern).append(" ");
                testResult = testResult.replaceFirst(pattern, "");
            } else {
                wrongText.append(pattern).append(" ");
            }
        }

        return new RecognitionResult(correctText.toString(), wrongText.toString(),
result);
    }
}
```

IStringService.java

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

import ihorko.work.speech_recognition.common.RecognitionResult;
import org.springframework.stereotype.Service;
```

```
@Service
public interface IStringService {

    RecognitionResult findCorrectAndWrongPartInExpectedText(String expectedResult,
String result);
}
```

AudioRecognitionService.java

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

import ihorko.work.speech_recognition.common.Language;
import lombok.SneakyThrows;
import org.springframework.stereotype.Service;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.logging.Logger;

@Service
public class AudioRecognitionService implements IAudioRecognitionService {

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

    @SneakyThrows
    public String recognizeAudioRecord(String filePathFromSourceRoot, Language
language) {
        ProcessBuilder processBuilder = new ProcessBuilder("python",
                "src/main/resources/python/audio_recognition.py",
filePathFromSourceRoot, language.getCode());
        processBuilder.redirectErrorStream(true);
        Process process = processBuilder.start();

        try (InputStream inputStream = process.getInputStream();
             BufferedReader bufferedReader = new BufferedReader(
                     new InputStreamReader(inputStream, "windows-1251"))) {

            String recognizedText = bufferedReader.readLine();
            if (recognizedText != null)
                LOGGER.warning(() -> "Recognized text: " + recognizedText);
            return recognizedText;
        } catch (IOException e) {
            LOGGER.severe(e.getMessage());
        }
        return null;
    }
}
```

IAudioRecognitionService.java

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

import ihorko.work.speech_recognition.common.Language;
import org.springframework.stereotype.Service;

@Service
public interface IAudioRecognitionService {
```

```
    String recognizeAudioRecord(String filePathFromSourceRoot, Language language);
}
```

FileRepository.java

```
package ihorko.work.speech_recognition.repository;

import ihorko.work.speech_recognition.db.entity.File;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.util.UUID;

@Repository
public interface FileRepository extends CrudRepository<File, UUID> {

}
```

```
package ihorko.work.speech_recognition.repository;

import ihorko.work.speech_recognition.db.entity.SoundContent;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository
public interface SoundContentRepository {

    void save(SoundContent soundContent);

    List<SoundContent> findAll();

    List<SoundContent> findListSoundContentBySound(UUID sound);

    SoundContent findById(UUID uuid);

    void delete(UUID uuid);
}
```

SoundContentRepositoryImpl.java

```
package ihorko.work.speech_recognition.repository;

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

import java.util.List;
import java.util.UUID;

@Repository
public class SoundContentRepositoryImpl implements SoundContentRepository {

    private final SoundContentDao soundContentDao;

    @Autowired
    public SoundContentRepositoryImpl(SoundContentDao soundContentDao) {
        this.soundContentDao = soundContentDao;
    }

    public void save(SoundContent soundContent) {
        soundContentDao.persist(soundContent);
```

```
    }

    public List<SoundContent> findAll() {
        return soundContentDao.listSoundsContent();
    }

    public List<SoundContent> findListSoundContentBySound(UUID sound) {
        return soundContentDao.listSoundsContentBySound(sound);
    }

    public SoundContent findById(UUID uuid) {
        return soundContentDao.findById(uuid);
    }

    public void delete(UUID id) {
        soundContentDao.delete(id);
    }
}
```

SoundRepository.java

```
package ihorko.work.speech_recognition.repository;

import ihorko.work.speech_recognition.db.entity.Sound;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository
public interface SoundRepository {

    void save(Sound sound);

    List<Sound> findAll();

    List<Sound> findByName(String name);

    List<Sound> findByLanguage(String language);

    Sound findById(UUID id);

    void delete(UUID id);
}
```

SoundRepositoryImpl.java

```
package ihorko.work.speech_recognition.repository;

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

import java.util.List;
import java.util.UUID;

@Repository
public class SoundRepositoryImpl  implements SoundRepository{

    private final SoundDao soundDao;

    @Autowired
```

```
    public SoundRepositoryImpl(SoundDao soundDao) {
        this.soundDao = soundDao;
    }

    public void save(Sound sound) {
        soundDao.persist(sound);
    }

    public List<Sound> findAll() {
        return soundDao.listSounds();
    }

    public List<Sound> findByName(String name) {
        return soundDao.findByName(name);
    }

    public List<Sound> findByLanguage(String language) {
        return soundDao.findByLanguage(language);
    }

    public Sound findById(UUID id) {
        return soundDao.findById(id);
    }

    public void delete(UUID id){
        soundDao.deleteSound(id);
    }
}
```

IntenationalizationConfig.java

```
package ihorko.work.speech_recognition.internationalization;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ReloadableResourceBundleMessageSource;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.i18n.CookieLocaleResolver;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import java.util.Locale;
@Configuration
public class InternationalizationConfig implements WebMvcConfigurer {

    @Bean
    public LocaleResolver localeResolver() {
        CookieLocaleResolver cookieLocaleResolver = new CookieLocaleResolver();
        cookieLocaleResolver.setDefaultLocale(Locale.US);
        cookieLocaleResolver.setCookieMaxAge(60*100);
        return cookieLocaleResolver;
    }
    @Bean
    public ReloadableResourceBundleMessageSource messageSource() {
        ReloadableResourceBundleMessageSource
reloadableResourceBundleMessageSource
                = new ReloadableResourceBundleMessageSource();
        reloadableResourceBundleMessageSource.setDefaultEncoding("UTF-8");
        reloadableResourceBundleMessageSource.setBasename("classpath:messages");
        return reloadableResourceBundleMessageSource;
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
```

```
        LocaleChangeInterceptor localeInterceptor = new LocaleChangeInterceptor();
        localeInterceptor.setParamName("lang");
        registry.addInterceptor(localeInterceptor).addPathPatterns("/*");
    }
}
```

FileStorageService.java

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

import ihorko.work.speech_recognition.db.entity.File;
import ihorko.work.speech_recognition.repository.FileRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import org.springframework.web.multipart.MultipartFile;

import javax.transaction.Transactional;
import java.io.IOException;
import java.util.Objects;
import java.util.UUID;

@Service
public class FileStorageService {

    private final FileRepository fileRepository;

    @Autowired
    public FileStorageService(FileRepository fileRepository) {
        this.fileRepository = fileRepository;
    }

    @Transactional
    public File storeFile(MultipartFile file) {
        // Normalize file name
        String fileName = StringUtils.cleanPath(Objects
                .requireNonNull(file.getOriginalFilename()));

        File dbFile = null;
        try {
            dbFile = new File(fileName, file.getContentType(), file.getBytes());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return fileRepository.save(Objects.requireNonNull(dbFile));
    }

    public File findById(UUID uuid) {
        return fileRepository.findById(uuid)
                .orElseThrow();
    }

    public void delete(File file){
        fileRepository.delete(file);
    }
}
```

SoundRepositoryImpl.java

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

import ihorko.work.speech_recognition.db.entity.Sound;
import ihorko.work.speech_recognition.repository.SoundRepositoryImpl;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.UUID;

@Service
public class SoundService {

    private final SoundRepositoryImpl soundRepository;

    @Autowired
    public SoundService(SoundRepositoryImpl soundRepository) {
        this.soundRepository = soundRepository;
    }

    public void save(Sound sound) {
        soundRepository.save(sound);
    }

    public List<Sound> findAll() {
        return soundRepository.findAll();
    }

    public List<Sound> findByLanguage(String language) {
        return soundRepository.findByLanguage(language);
    }

    public List<Sound> findByName(String name) {
        return soundRepository.findByName(name);
    }

    public Sound findById(UUID id) {
        return soundRepository.findById(id);
    }
}
```

SoundContentDto.java

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

import ihorko.work.speech_recognition.db.entity.File;
import ihorko.work.speech_recognition.db.entity.Sound;
import lombok.Data;
import lombok.Getter;

import java.util.UUID;

@Data
@Getter
public class SoundContentDto {

    private UUID id;

    private String contentText;

    private String contentType;

    private Sound sound;

    private File audioFile;

    private File gifFile;
}
```

File.java

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

import org.hibernate.annotations.Type;

import javax.persistence.*;
import java.util.UUID;

@Entity(name = "file")
public class File {

    @Id
    @GeneratedValue
    private UUID id;

    private String fileName;

    private String fileType;

    @Lob
    @Type(type = "org.hibernate.type.BinaryType")
    private byte[] data;

    @ManyToOne
    @JoinColumn(name = "sound_content_id")
    private SoundContent soundContent;

    public File() {
    }

    public File(String fileName, String fileType, byte[] data) {
        this.fileName = fileName;
        this.fileType = fileType;
        this.data = data;
    }

    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
        this.id = id;
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

    public String getFileType() {
        return fileType;
    }

    public void setFileType(String fileType) {
        this.fileType = fileType;
    }

    public byte[] getData() {
        return data;
    }
```

```
    public void setData(byte[] data) {
        this.data = data;
    }

    public SoundContent getSoundContent() {
        return soundContent;
    }

    public void setSoundContent(SoundContent soundContent) {
        this.soundContent = soundContent;
    }
}
```

Sound.java

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

import lombok.Getter;
import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@Entity(name = "sound")
@Getter
public class Sound {
    @Id
    @GeneratedValue
    private UUID id;
    private String name;
    private String language;
    @OneToMany(mappedBy = "sound", cascade = CascadeType.ALL, targetEntity =
SoundContent.class)
    private List<SoundContent> soundContents = new ArrayList<>();
    public void addSoundContent(SoundContent soundContent) {
        soundContents.add(soundContent);
        soundContent.setSound(this);
    }
    public void setId(UUID id) {
        this.id = id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setLanguage(String language) {
        this.language = language;
    }
}
```

HibernateConfig.java

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

import org.apache.tomcat.dbcp.dbcp2.BasicDataSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.annotation.EnableTransactionManagement;

import javax.sql.DataSource;
```

```
import java.util.Properties;

@Configuration
@EnableTransactionManagement
public class HibernateConfig {

    @Bean(name="entityManagerFactory")
    public LocalSessionFactoryBean sessionFactory() {
        LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
        sessionFactory.setDataSource(dataSource());

sessionFactory.setPackagesToScan("ihorko/work/speech_recognition/db/entity");
        sessionFactory.setDataSource(dataSource());
        sessionFactory.setHibernateProperties(hibernateProperties());
        return sessionFactory;
    }

    @Bean
    public DataSource dataSource() {
        BasicDataSource dataSource = new BasicDataSource();
        dataSource.setDriverClassName("org.postgresql.Driver");
        dataSource.setUrl("jdbc:postgresql://localhost:5432/SpeechRecognition");
        dataSource.setUsername("postgres");
        dataSource.setPassword("1234");

        return dataSource;
    }

    private Properties hibernateProperties() {
        Properties hibernateProperties = new Properties();
        hibernateProperties.setProperty(
                "hibernate.hbm2ddl.auto", "update");
        hibernateProperties.setProperty(
                "hibernate.dialect", "org.hibernate.dialect.PostgreSQL10Dialect");

        return hibernateProperties;
    }
}
```

SoundContentConverter.java

```
package ihorko.work.speech_recognition.converter;

import ihorko.work.speech_recognition.db.dto.SoundContentDto;
import ihorko.work.speech_recognition.db.entity.SoundContent;
import org.springframework.stereotype.Component;

@Component
public class SoundContentConverter {

    public SoundContentDto convert(SoundContent soundContent) {
        SoundContentDto soundContentDto = new SoundContentDto();
        soundContentDto.setContentText(soundContent.getContentText());
        soundContentDto.setSound(soundContent.getSound());
        soundContentDto.setId(soundContent.getId());
        soundContentDto.setContentType(soundContent.getTypeContent());
        soundContentDto.setAudioFile(soundContent.getFiles()
                .stream()
                .filter(content -> content.getFileType().contains("audio"))
                .findFirst().orElse(null));
        soundContentDto.setGifFile(soundContent.getFiles()
                .stream()
                .filter(content -> content.getFileType().contains("image"))
```

```
                .findFirst().orElse(null));
        return soundContentDto;
    }

}
```

SoundController.java

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

import ihorko.work.speech_recognition.db.entity.Sound;
import ihorko.work.speech_recognition.service.SoundService;
import org.apache.commons.collections4.ListUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class SoundController {

    private SoundService soundService;

    @Autowired
    public void setSoundService(SoundService soundService) {
        this.soundService = soundService;
    }

    @GetMapping("/sound/create/page")
    public String showCreateSoundForm(Model model) {
        model.addAttribute("sound", new Sound());
        return "soundCreate";
    }

    @PostMapping("/sound/create")
    public String createSound(Sound sound) {
        soundService.save(sound);
        return "redirect:/sound/create/page";
    }

    @GetMapping("/sounds/list")
    public String showListSounds(Model model) {
        model.addAttribute("soundsLists",
ListUtils.partition(soundService.findAll(), 4));
        return "soundsList";
    }

    @GetMapping("/sounds/list/{language}")
    public String showListSoundsByLanguage(@PathVariable String language, Model
model) {
        model.addAttribute("soundsLists",
ListUtils.partition(soundService.findByLanguage(language), 4));
        return "soundsList";
    }
}
```

SoundContentController.java

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

import ihorko.work.speech_recognition.converter.SoundContentConverter;
import ihorko.work.speech_recognition.db.dto.SoundContentDto;
import ihorko.work.speech_recognition.db.entity.File;
```

```
import ihorko.work.speech_recognition.db.entity.Sound;
import ihorko.work.speech_recognition.db.entity.SoundContent;
import ihorko.work.speech_recognition.service.FileStorageService;
import ihorko.work.speech_recognition.service.SoundContentService;
import ihorko.work.speech_recognition.service.SoundService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

@Controller
public class SoundContentController {

    private final SoundService soundService;
    private final SoundContentService soundContentService;
    private final FileStorageService fileStorageService;
    private final SoundContentConverter soundContentConverter;

    @Autowired
    public SoundContentController(SoundService soundService, SoundContentService
soundContentService,
                                  FileStorageService fileStorageService,
SoundContentConverter soundContentConverter) {
        this.soundService = soundService;
        this.soundContentService = soundContentService;
        this.fileStorageService = fileStorageService;
        this.soundContentConverter = soundContentConverter;
    }

    @GetMapping("/sound-content/create/page")
    public String showSoundContentCreatePage(Model model) {
        model.addAttribute("soundContent", new SoundContent());
        model.addAttribute("sounds", soundService.findAll());
        return "sound_content/soundContentCreate";
    }

    @PostMapping("/sound-content/create")
    public String createSoundContent(SoundContent soundContent,
                                     @RequestParam MultipartFile imageFile,
                                     @RequestParam MultipartFile audioFile,
RedirectAttributes redirectAttributes) {
        File file = fileStorageService.storeFile(imageFile);
        File dbAudioFile = fileStorageService.storeFile(audioFile);

        Sound sound = soundService.findById(soundContent.getSound().getId());
        if (sound.getName().isEmpty()) {
            redirectAttributes.addFlashAttribute("message", "Failed");
            redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
            throw new IllegalArgumentException("Name for our sound is empty");
        }
        sound.addSoundContent(soundContent);
        soundContent.setSound(sound);
        soundContent.addDbFile(file);
        soundContent.addDbFile(dbAudioFile);
```

```
        soundContentService.save(soundContent);
        redirectAttributes.addFlashAttribute("message", "Success");
        redirectAttributes.addFlashAttribute("alertClass", "alert-success");

        return "redirect:/sound-contents/list";
    }

    @GetMapping("/sound-contents/list")
    public String showSoundContentsList(Model model) {
        List<SoundContentDto> collect =
soundContentService.findAll().stream().map(soundContentConverter::convert).collect
(Collectors.toList());
        model.addAttribute("soundContentsList", collect);
        return "sound_content/soundContentsList";
    }

    @GetMapping("/sound-contents/{soundId}")
    public String showSoundContentsList(@PathVariable String soundId, Model model)
{

        List<SoundContentDto> collect =
soundContentService.findListSoundContentBySound(UUID.fromString(soundId))
                .stream()
                .map(soundContentConverter::convert)
                .collect(Collectors.toList());
        model.addAttribute("soundContentsList", collect);
        return "sound_content/soundContentsList";
    }

    @GetMapping("/sound-content/{id}")
    public String showSoundContentPage(@PathVariable UUID id, Model model) {
        SoundContentDto soundContent =
soundContentConverter.convert(soundContentService.findById(id));
        model.addAttribute("soundContent", soundContent);
        return "sound_content/soundContent";
    }
}
```

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.AudioRecognitionService;
import ihorko.work.speech_recognition.service.SoundContentService;
import ihorko.work.speech_recognition.service.StringService;
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 static final Gson gson = new Gson();

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

    @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));

        RecognitionResult recognitionResult =
makeRecognitionAndSaveToResults(savedFile.getPath(),

Language.valueOf(soundContent.getSound().getLanguage().toUpperCase()),
                soundContent.getContentText());

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

        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));
    }

    @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);
    }
}
```

LanguageController.java

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

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class OwnWordsController {

    @GetMapping("/own-words")
    public String learnOwnWords() {
        return "learnOwnWords";
    }
}

package ihorko.work.speech_recognition.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class LanguageController {

    @GetMapping("/language")
    public String showLanguagePage() {
        return "/language";
    }
}
```

DBFileController.java

```
package ihorko.work.speech_recognition.controller;
import ihorko.work.speech_recognition.db.entity.File;
import ihorko.work.speech_recognition.service.FileStorageService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ByteArrayResource;
import org.springframework.core.io.Resource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import java.util.UUID;
@Controller
public class DBFileController {
    private FileStorageService fileStorageService;
    @Autowired
    public void setDbFileStorageService(FileStorageService fileStorageService) {
        this.fileStorageService = fileStorageService;    }

    @GetMapping("/files/{fileId}")
    public ResponseEntity<Resource> downloadFile(@PathVariable UUID fileId) {
        // Load file from database
        File file = fileStorageService.findById(fileId);
        return ResponseEntity.ok()
                .contentType(MediaType.parseMediaType(file.getFileType()))
                .header(HttpHeaders.CONTENT_DISPOSITION,
                        "attachment; filename=\"" + file.getFileName() + "\"")
                .body(new ByteArrayResource(file.getData()));
    }
}
```

StringSearch.java

```
package ihorko.work.speech_recognition.common;
import java.util.logging.Logger;
public class StringSearch {
    public static final int D = 256;
    private static final Logger LOGGER =
Logger.getLogger(StringSearch.class.getName());
    public boolean search(String pat, String txt) {
        int numberForCalculatingHash = 10;
        int patternLength = pat.length();
        int textLength = txt.length();
        if (patternLength > textLength)
            return false;
        int hashValueForPattern = 0;
        int hashValueForText = 0;
        int numberForChangingHash = 1;
        // The value of numberForChangingHash would be "pow(d, patternLength-
1)%numberForCalculatingHash"
        for (int i = 0; i < patternLength - 1; i++)
            numberForChangingHash = (numberForChangingHash * D) %
numberForCalculatingHash;
        // hash for pattern and our first part of text
        for (int i = 0; i < patternLength; i++) {
            hashValueForPattern = (D * hashValueForPattern + pat.charAt(i)) %
numberForCalculatingHash;
            hashValueForText = (D * hashValueForText + txt.charAt(i)) %
numberForCalculatingHash;
```

```
        }
        int j;
        //find our pattern in text one by one
        for (int i = 0; i <= textLength - patternLength; i++) {
            if (hashValueForPattern == hashValueForText) {
                /* Check for characters one by one */
                for (j = 0; j < patternLength; j++) {
                    if (txt.charAt(i + j) != pat.charAt(j))
                        break;
                }
                if (j == patternLength) {
                    String messageForLogging = String.format("Pattern found at
index %d", i);
                    LOGGER.info(() -> messageForLogging);
                    return true;
                }
            }
            //our calculating of polynomial hash
            if (i < textLength - patternLength) {
                hashValueForText = (D * (hashValueForText - txt.charAt(i) *
numberForChangingHash) + txt.charAt(i + patternLength)) %
numberForCalculatingHash;
                if (hashValueForText < 0)
                    hashValueForText = (hashValueForText +
numberForCalculatingHash);
            }
        }
        return false;
    }
}
```

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;

}
```

Language.java

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

import lombok.Getter;

@Getter
public enum Language {
    ENGLISH("en-US"),
    UKRAINIAN("uk-UA");
    private final String code;
    Language(String code) {
        this.code = code;
    }
}
```

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.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>
        </plugins>
    </build>
</project>
```

audio_recognition.py

```
import speech_recognition
import sys

fileNameForRecognition = sys.argv[1]
languageCode = sys.argv[2]
recognizer = speech_recognition.Recognizer()
test_file = speech_recognition.AudioFile(fileNameForRecognition)
with test_file as source:
    audio = recognizer.record(source)
    text = recognizer.recognize_google(audio, None, languageCode, False)
    print(text)
```

Додаток Г. Графічна частина

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ПРАВИЛЬНОЇ ВИМОВИ ЗВУКІВ

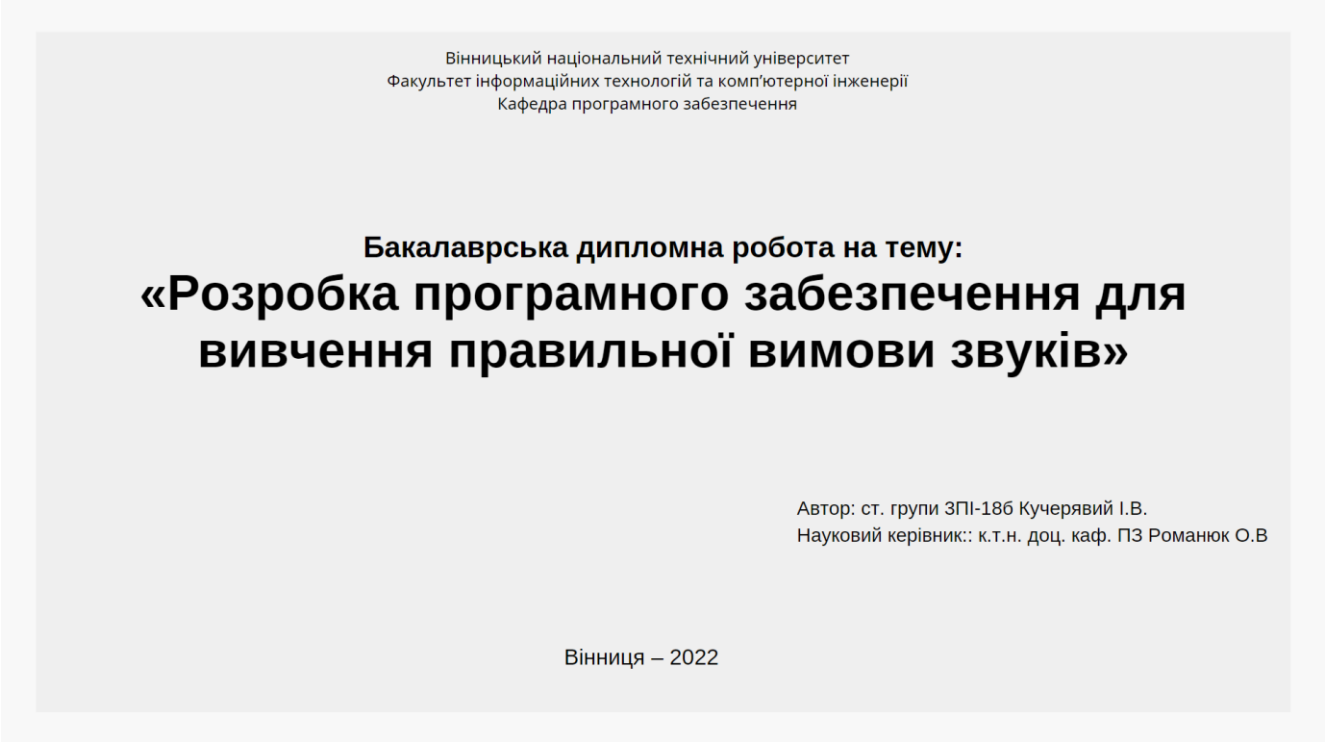

Рисунок Г.1 – Титульний слайд

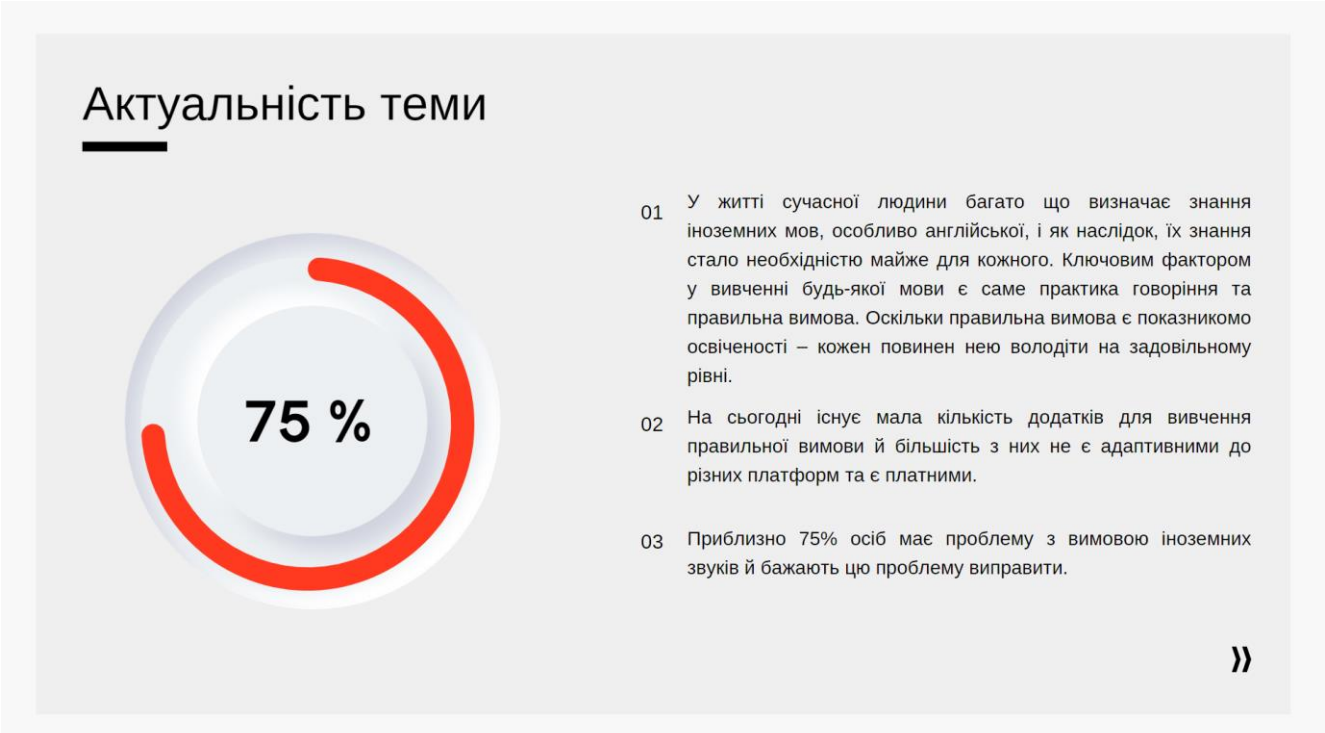

Рисунок Г.2 – Актуальність теми

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

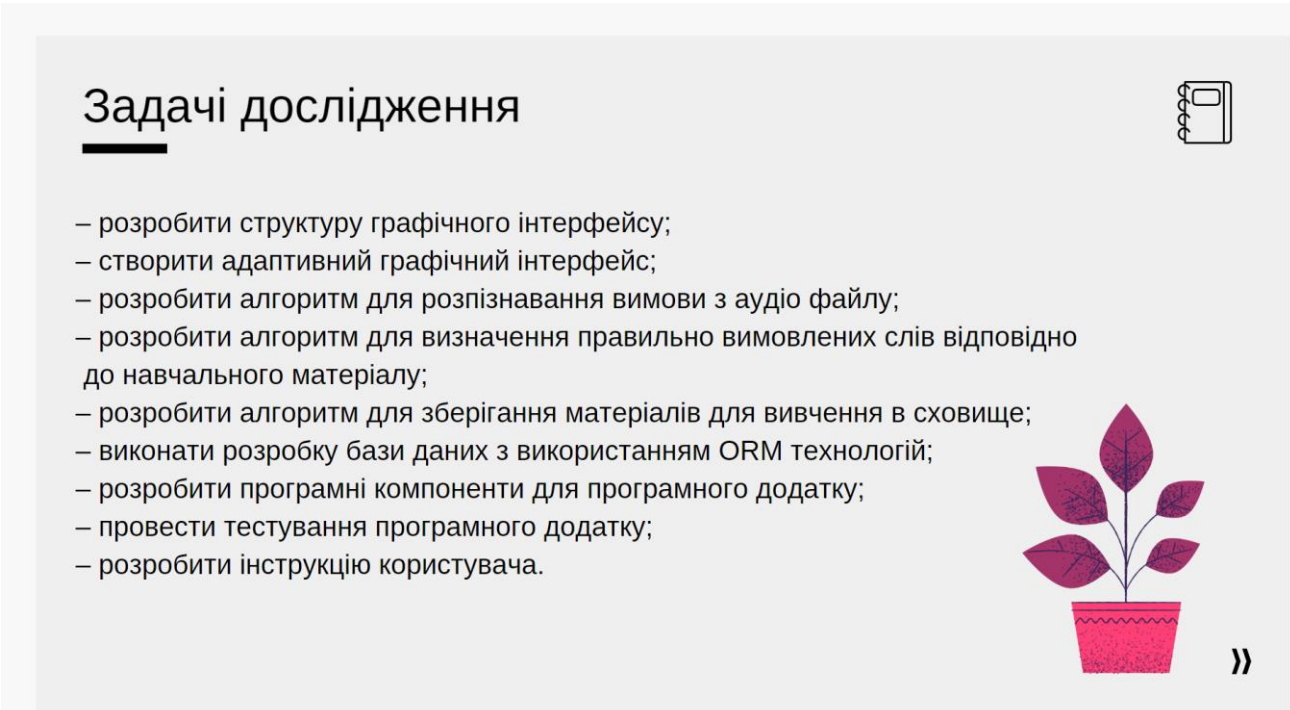

Рисунок Г.4 – Задачі дослідження

Рисунок Г.5 – Новизна і практична цінність отриманих результатів

Порівняльна характеристика програмних додатків

Критерій	Cake	Lingvist	SpeechWorld	Influent	Наш додаток
Створення власного матеріалу для вивчення	0	1	0	1	1
Перевірка вимови користувача	1	1	0	0	1
Багатоплатформність	0	0.5	1	0	1
Безкоштовність повного функціоналу	0	0	0	0	1
Наявність інтернаціоналізації	1	1	0	1	1
Загальний результат	2	3.5	1	2	5

Рисунок Г.6 – Порівняльна характеристика програмних додатків

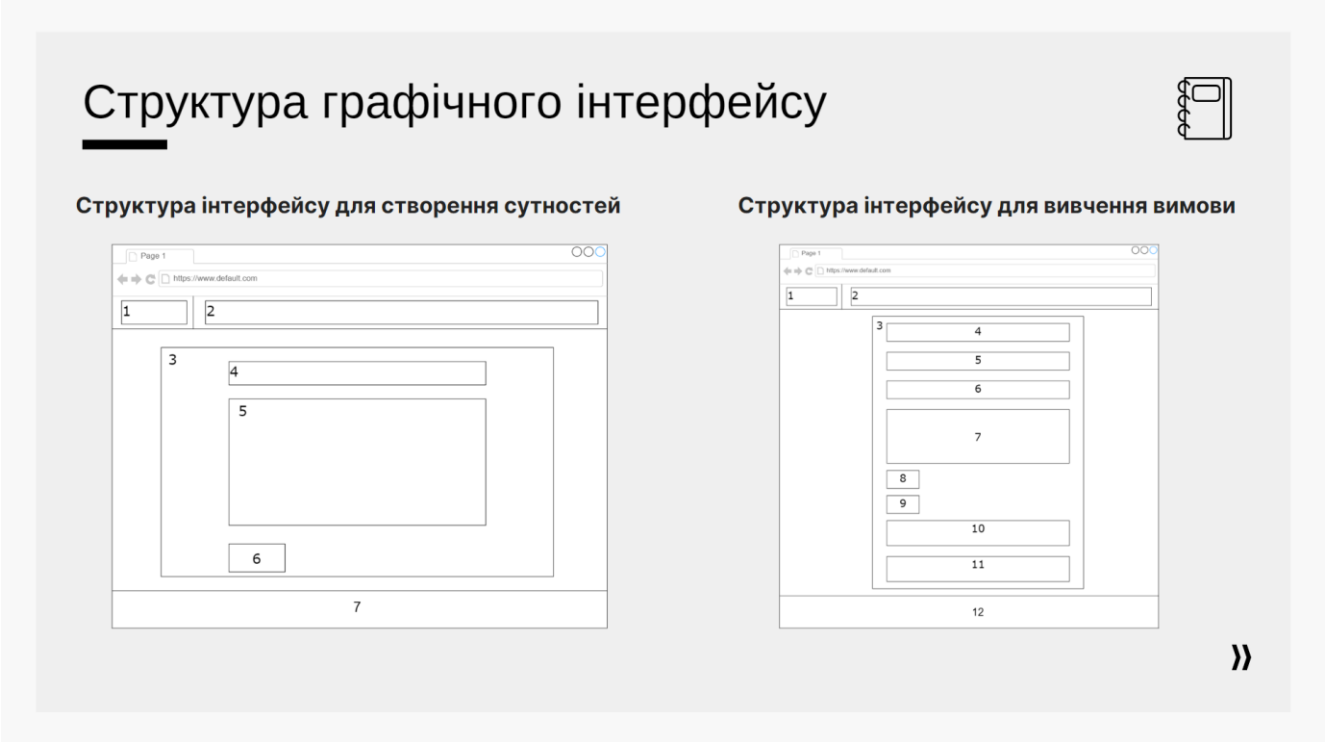

Рисунок Г.7 – Структура графічного інтерфейсу

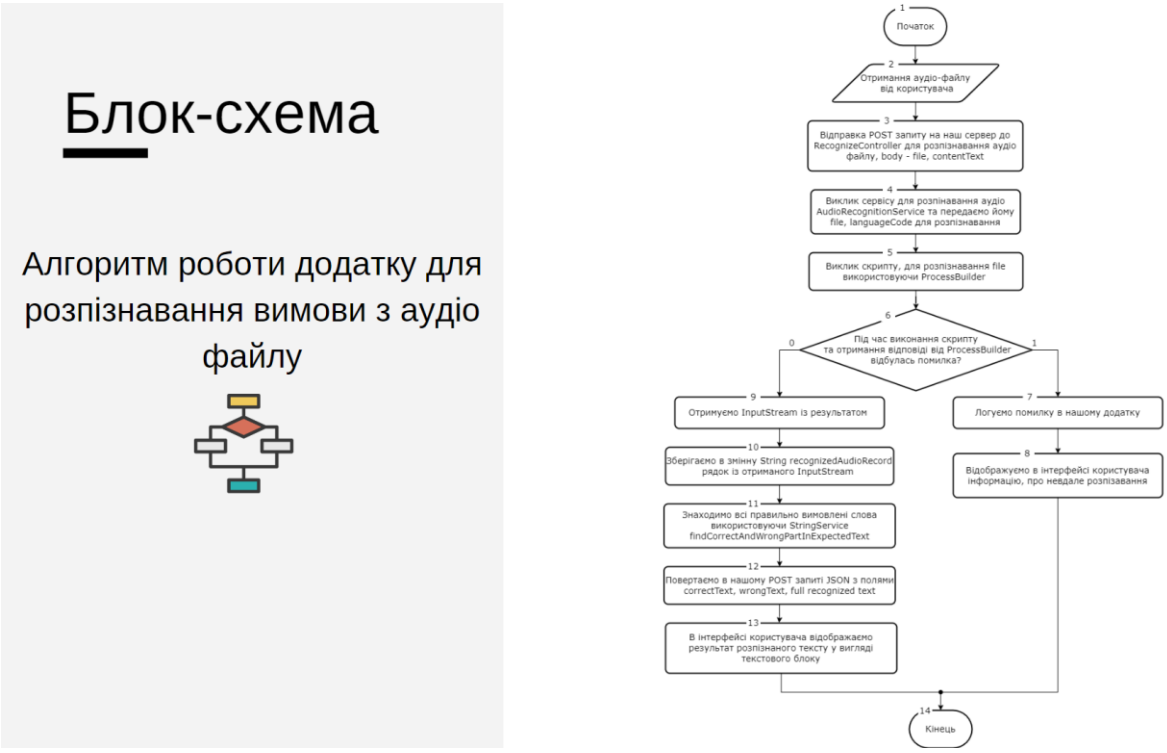

Рисунок Г.8 – Блок-схема алгоритму роботи додатку для розпізнавання вимови з аудіо файлу

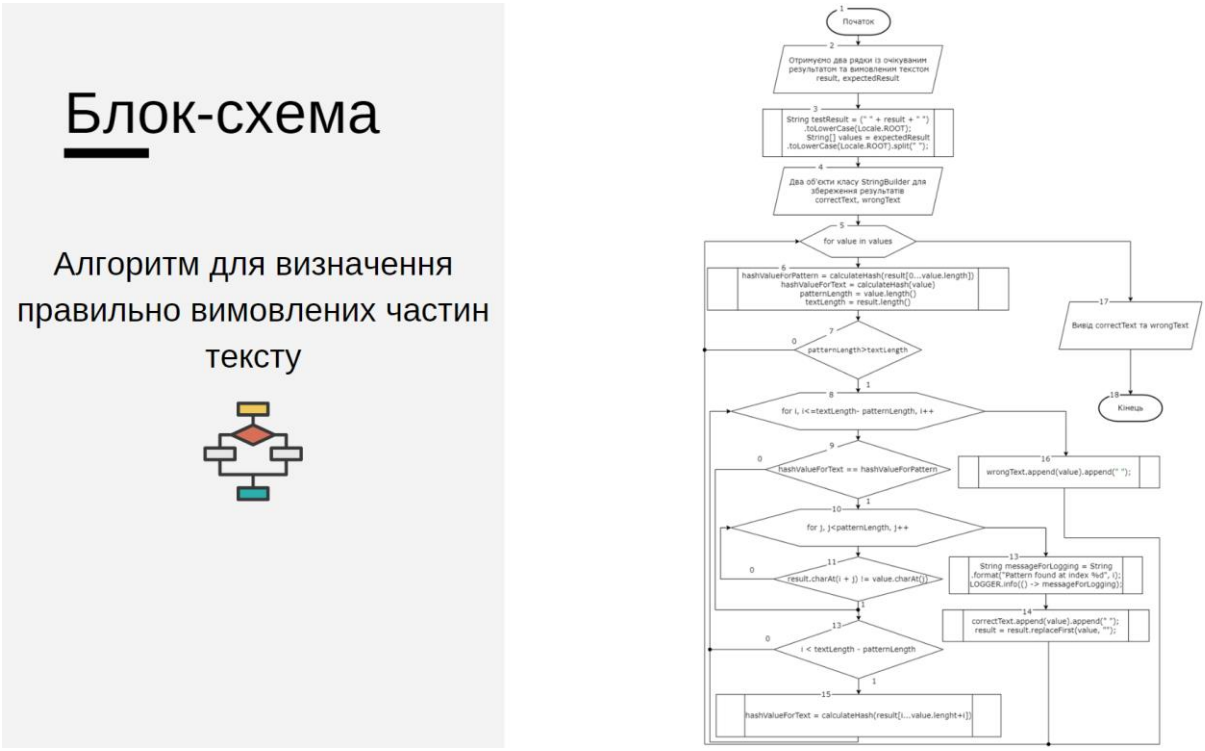

Рисунок Г.9 – Блок-схема алгоритму для визначення правильно вимовлених частин тексту

Рисунок Г.10 – Блок-схема алгоритму зберігання матеріалу для вивчення в сховище

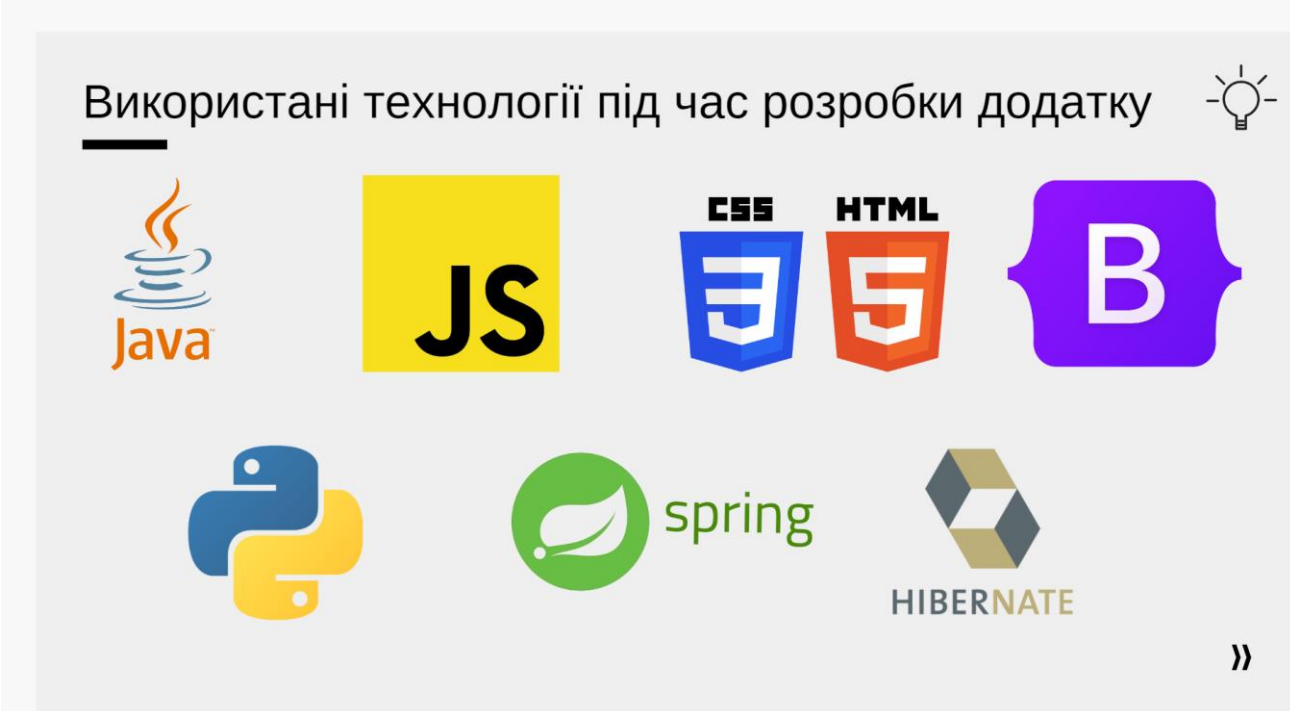

Рисунок Г.11 – Використані технології під час розробки додатку

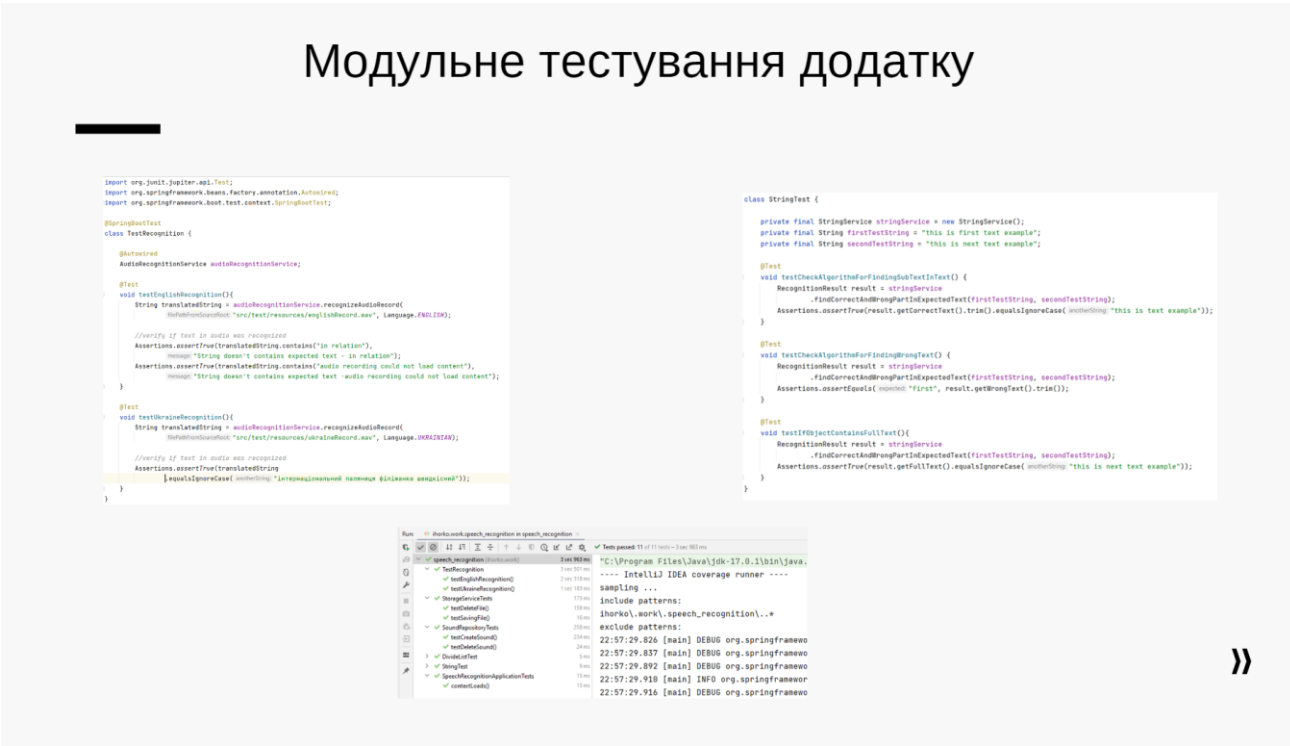

Рисунок Г.12 – Модульне тестування додатку

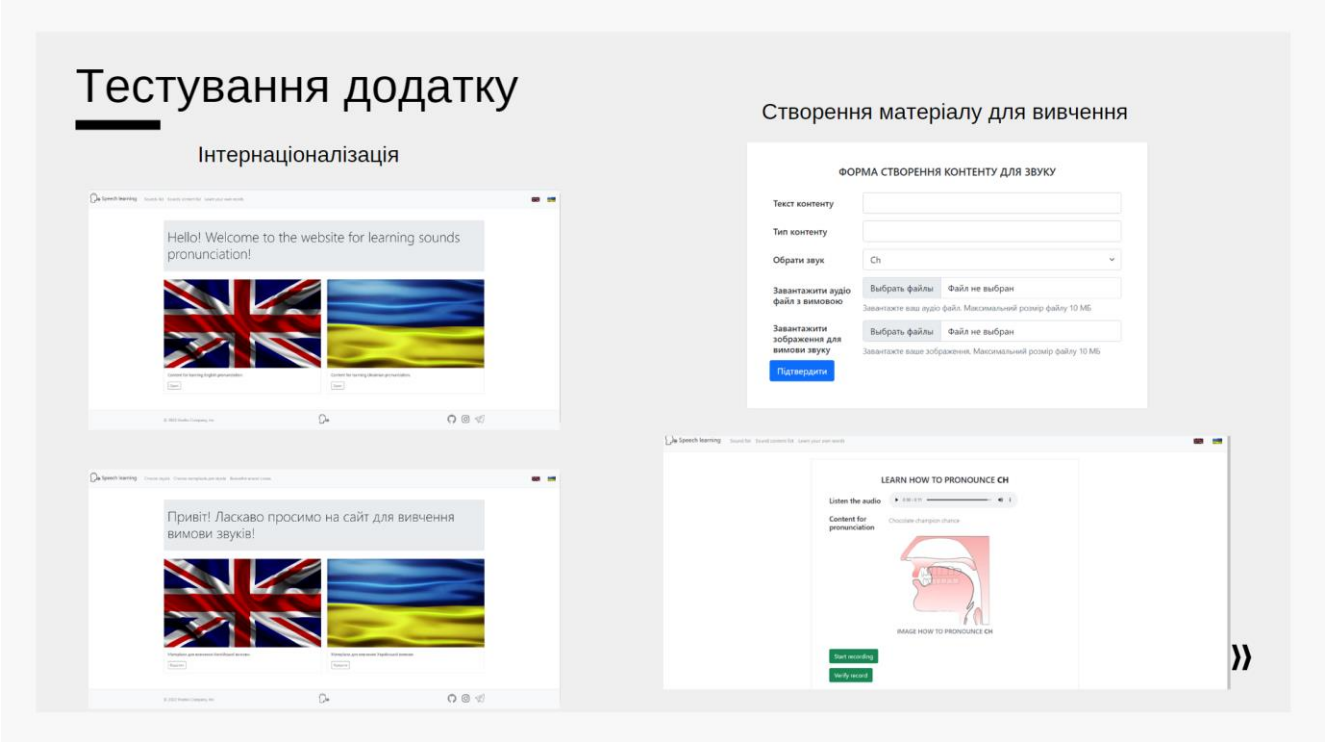

Рисунок Г.13 – Тестування інтернаціоналізації й створення матеріалу для вивчення

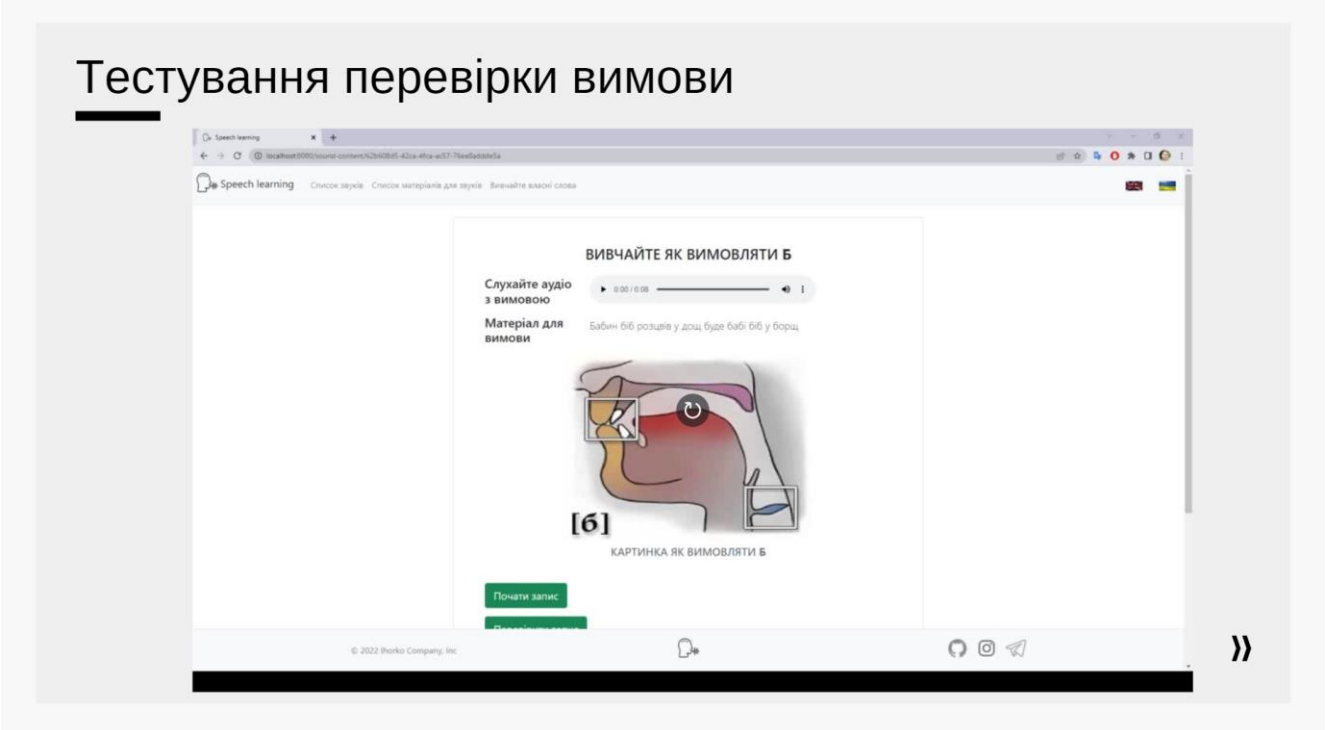

Рисунок Г.14 – Тестування перевірки вимови

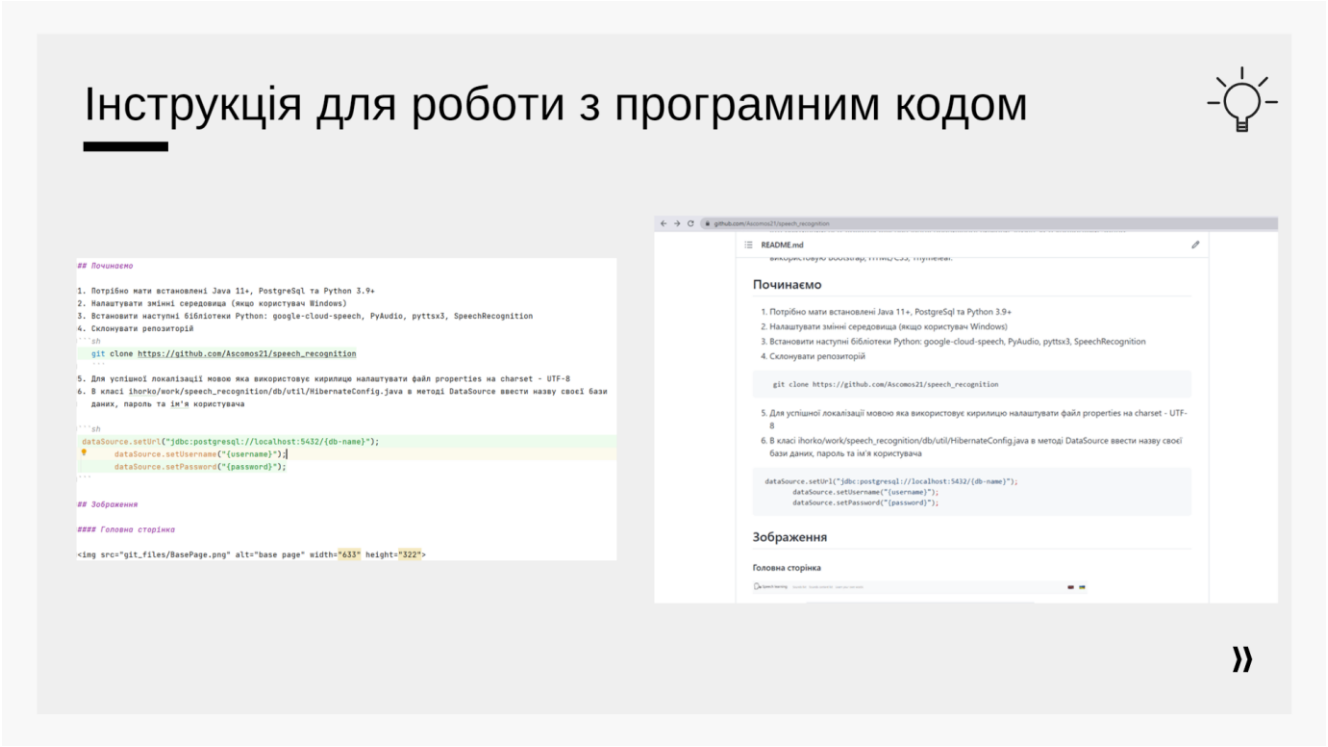

Рисунок Г.15 – Інструкція для роботи з програмним кодом

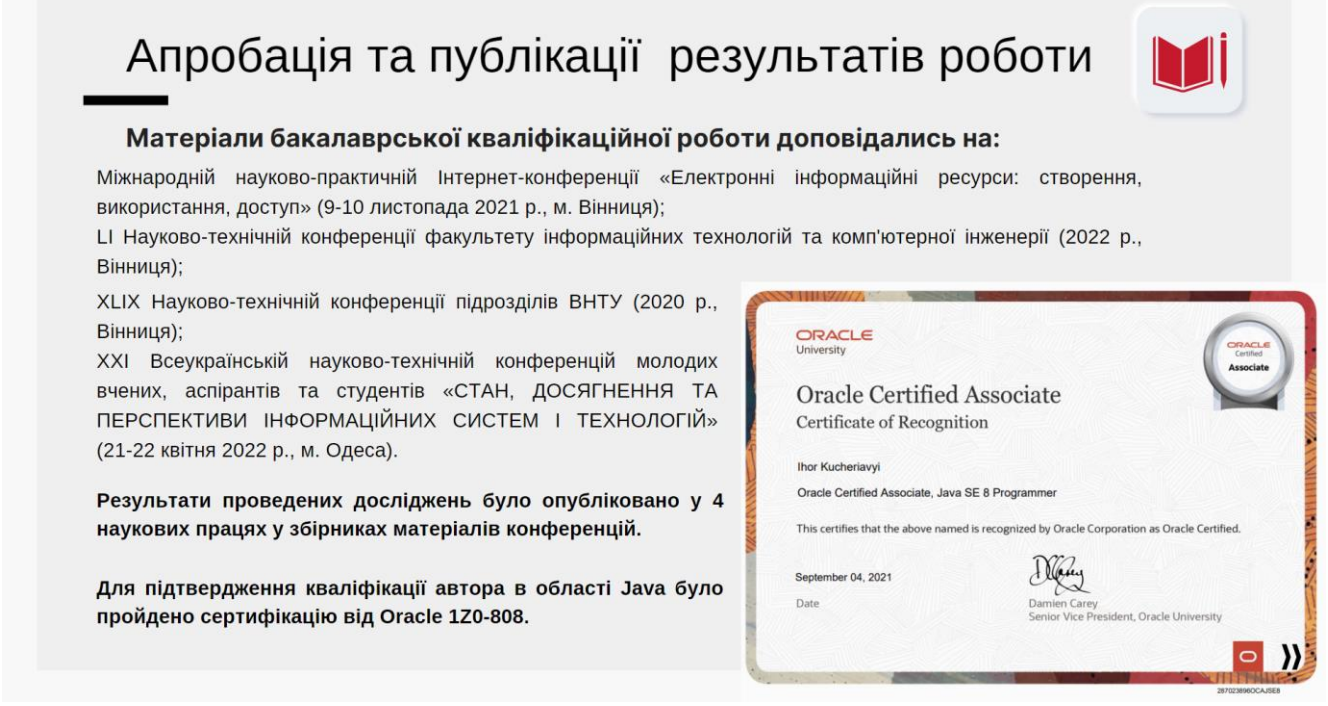

Рисунок Г.16 – Апробація та публікації результатів роботи

Рисунок Г.17 – Фінальний слайд