Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

«Розробка лінгвістичного тренажеру з розпізнаванням вимови» 122

Керівник курсової роботи
доцент Афонін А. О.
_____
(підпис)
“____”_____2024 р.

Виконала студентка 3 р. н.
Бернацька О. О.
“____”_____2024 р.

Київ 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
доцент, заступник декана
Афонін А. О.

(підпис)
„_____”_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентці Бернацькій Ользі Олександрівні факультету інформатики 3-го курсу

ТЕМА: Розробка лінгвістичного тренажеру з розпізнаванням вимови

Вихідні дані:

Зміст ТЧ до курсової роботи

Вступ

Анотація

1. Постановка задачі
2. Дослідження алгоритмів порівняння аудіо на схожість
3. Вибір засобів для розробки
4. Реалізація застосунку

Висновки

Список використаних джерел

Дата видачі „___”_____2024 р. Керівник _____
Завдання отримала _____

Календарний план виконання курсової роботи

Тема: Розробка лінгвістичного тренажеру з розпізнаванням вимови

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	Жовтень 2023 р.	
2.	Огляд технічної літератури за темою роботи.	Грудень 2023 р.	
3.	Аналіз сучасних методів порівняння схожості аудіозаписів.	Січень – лютий 2023 р.	
4.	Програмування обраного алгоритму.	Березень 2024 р.	
5.	Розробка додатку.	Березень 2024 р.	
6.	Написання текстової частини курсової.	Квітень 2024 р.	
7.	Створення презентації для захисту.	Травень 2024 р.	
8.	Здача роботи для перевірки на плагіат.	14 травня 2024 р.	
9.	Захист курсової роботи.	23 травня 2024 р.	

Студент: Бернацька О. О.

Керівник: Афонін А. О.

“_____”_____

ЗМІСТ

Анотація

Курсова робота присвячена дослідженню методів порівняння аудіозаписів на подібність та впровадження такого алгоритму в застосунок-тренажер з метою вивчення наголосів слів української мови. Було проведено аналіз трьох різних підходів для вирішення даного завдання та імплементація одного з них.

Було використано бібліотеки Meyda та dynamic-time-warping для реалізації методу обчислення ступеню схожості аудіо на основі аналізу Mel-Frequency Cepstral Coefficient. Розроблений сервіс надає можливість користувачу проводити тренування, записуючи свою вимову, та отримувати результат про її правильність. Також додатково імплементовано систему навчання по рівнях з прогресом.

Вступ

Сучасний світ з кожним роком все більше насичується новими технологіями та розвитком і ширшим використанням старих. Спроби перших досліджень методів аналізу голосу людини з допомогою комп'ютерів були ще більше 40 років тому. Змога аналізувати аудіозаписи стала настільки хорошою, що зараз людство може похизуватись доволі точними системами голосового розпізнавання. Іншими сферами застосування цієї технології є розпізнавання музики та пісень, пошук схожих аудіозаписів з метою надання персоналізованих рекомендацій для користувачів, виявлення порушення авторських прав тощо. І в основі реалізації цих можливостей лежить здатність порівнювати аудіозаписи на схожість.

Розробка методів повноцінного аналізу записів з можливістю подальшого порівняння не є легкою задачею. Аудіо формат має набагато складнішу структуру, аніж фото чи текст, і визначення дієвої моделі, яка буде здатна правильно ідентифікувати його зміст, є програмно складним. Різні алгоритми можуть давати різний результат щодо порівняння одних і тих самих вхідних

даних, але це не значить, що жоден з цих результатів не є правдивий. Це пояснюється тим фактом, що сприйняття аудіо є суб’єктивним навіть для людини, тому визначити ідеальний алгоритм постає майже неможливою задачею.

Головним питанням є те, як саме обрати критерії для оцінки схожості аудіо, оскільки характеристик у цього формату є безліч: темп, висота, тон, частота, гучність, наявність шуму. Тому робота присвячена дослідженню декількох різних підходів до порівняння аудіозаписів та реалізації одного з них.

Вона складається з чотирьох розділів.

Перший розділ містить інформацію про впровадження методів порівняння аудіозаписів у застосунок, який буде корисним реальному користувачу, та виокремлення тих функцій, які він буде виконувати.

Другий розділ включає в себе дослідження трьох різних методів отримання інформації про схожість аудіозаписів. Це методи порівняння аудіо за акустичним відбитком, способом прямого аналізу його ознак та за допомогою моделювання та зіставлення графу. В результаті було обрано другий за номером підхід та було впроваджено його у застосунок.

У третьому розділі було описано обрані інструменти для розробки, а у четвертому розділі – структуру проєкту та його веб-інтерфейс.

1. ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз предметної області

В результаті пошуку ідеї для застосування алгоритмів порівняння аудіозаписів на схожість, яка була б корисною у світі сучасних проблем та потенційно могла б мати широку базу користувачів, було обрано задачу написання сервісу для підготовки випускників до іспитів з української мови з можливістю автоматичного аналізу вимови користувача.

Один з пунктів матеріалу, який є обов’язковим для вивчення з метою успішно здати екзамени, є правильна постановка наголосу в словах. До переліку надають більше, аніж 200 слів, для запам’ятовування, а головною проблемою при підготовці є те, що в повсякденному житті люди звикли говорити слова неправильно. Через це підготовка стає складнішою і потребує окремих зусиль. В якості спрощення цього процесу на просторах інтернету вже існує певна кількість тренажерів для вивчення наголосів, однак після їхнього ретельного аналізу було виявлено великий список недоліків, які і зумовили остаточний вибір ідеї написання власного сервісу, а також було доведено теорію, що алгоритми автоматичної перевірки вимови користувача ще ніде не були застосовані, тому реалізація такого сервісу принесла б користь не лише розробнику у вивченні нових методів і підходів, а й стала б у нагоді реальній аудиторії.

1.2. Виокремлення важливих функцій

В процесі пошуку найкращого рішення у функціональності майбутнього застосунку для вивчення наголосів було необхідно виділити всі переваги та уникнути недоліків, які наявні в схожих сервісах. Після дослідження конкурентних додатків, було проаналізовано, що такі функції, як

1) наявність пояснення до кожного тесту;
2) місткість одного тренування не більше, аніж певна попередньо задана

кількість тестів, з метою забезпечення максимально комфортного часу, який користувач виділить на тренування;

3) показ статистики у вигляді відсотка правильних відповідей після завершення тесту,

мають бути реалізовані першочергово.

Оскільки головною метою було виокремити майбутній застосунок від уже наявних та застосувати методи аналізу аудіо, було відомо заздалегідь, що він повинен додатково виконувати наступні функції:

1) наявність особистого кабінету з можливістю відслідковувати загальний прогрес навчання;
2) можливість вивчення слів по рівнях;
3) можливість давати відповідь на тест з допомогою голосу.

Варто зупинитись більш детально на поясненні останніх двох функцій. Перша з них, вивчення по рівнях, була імплементована з метою покращення процесу навчання і зменшення кількості часу, потрібного для тренування. Її суть полягає у тому, що при правильній відповіді без підказки, слово переходить на наступний рівень. При попаданні слова на новий рівень, воно вилучається з переліку слів, які з’являються при тренуванні, на певну кількість днів. Це було додано з метою оптимізувати процес навчання, щоб фокусувати увагу лише на тих тестах, на які користувач відповідає неправильно, а інші, на яких він добре орієнтується, повертати в потік лише для повторення і кращого запам’ятовування. Кількість днів залежить від номеру рівня, яких є всього 3. При переході на перший рівень - слово знову з’явиться у переліку через 5 днів, на другий рівень – через 10 днів, на третій – через 15 днів. Коли слово потрапляє на третій рівень, це означає, що воно вважається вивченим. Варто згадати, що загальний прогрес зареєстрованого користувача вираховується таким чином, що 100% – це тоді, коли усі слова перебувають на останньому третьому рівні.

Друга функція, голосова відповідь, є головною функцією додатку, оскільки вона включає в себе застосування методів порівняння аудіо на схожість. Логіка їх імплементації саме в такому форматі є доволі простою –

коли користувач, що тільки почав навчання, думає над відповіддю, він згадує наголос слова, проговорюючи його подумки або вголос, визначаючи, який варіант звучить краще і більш природно. Можливість зчитати голос, програмно визначити, наскільки правильно було визначено наголос слова, і одразу повернути результат користувачу полегшить процес навчання, привчаючи користувача відразу говорити правильно.

1.3. Вибір підходу для реалізації аналізу голосу

На відміну від попередніх функцій, задача аналізу вимови користувача не є тривіальною та потребує додаткового пошуку методів її реалізації, тому головною метою написання даної роботи була імплементація такої можливості у реальний застосунок з попереднім аналізом алгоритмів, які теоретично можуть бути застосовані для вирішення цієї проблеми.

Першим припущенням, яке спадає на думку, є те, що реалізувати таку функцію можна з допомогою готових систем розпізнавання мовлення, проте це не є правдою. Внаслідок того, що темпи розвитку технологічних рішень зросли, розпізнавання вимови стало доволі вагомою складовою у різних сферах. Перетворення аудіозаписів у друковані тексти і навпаки, ідентифікація особи та її управління комп’ютером з допомогою голосу – приклад переліку можливостей застосувань цієї технології. Проте, для досягнення мети вирішення поставленої задачі, розробки тренажера з вимови, розпізнавання мовлення буде недостатньо. Існуючі бібліотеки успішно вирішують задачу ідентифікації тексту з аудіо, але проблема полягає в тому, що вимагається не просто розпізнавання тексту з відповіддю на питання, чи сказав користувач те чи інше слово, а саме виявлення способу, яким він це зробив, тобто наголосу слова. Пошук готових засобів, які можуть виокремити наголос слова українською мовою з аудіозапису, був неуспішним.

Внаслідок цього, рішення застосувати методи порівняння аудіозаписів на схожість було цілком зрозумілим. Було зроблено припущення, що висока

ступінь подібності окремих характеристик запису вимови користувача та еталонного запису з правильним наголосом слова буде означати, що вимова користувача теж є правильною. У результаті, ця ідея і лягла в основі реалізації головної функції майбутнього застосунку.

1.4. Визначення кінцевої цілі

Ретельно зваживши предметну область та ціль майбутнього сервісу, врахувавши виокремлені функції, які мають бути імплементовані, було сформовано задачу, яка постала на меті розробки – реалізувати удосконалений сервіс для тренування правильної вимови, в підгрунті якого лежить можливість аналізу голосового вводу користувача. Головним завданням став пошук та реалізація алгоритму, який забезпечить високу ефективність виявлення схожості між аудіозаписами.

2. ДОСЛІДЖЕННЯ АЛГОРИТМІВ ПОРІВНЯННЯ АУДІО НА СХОЖІСТЬ

2.1. Порівняння аудіо за акустичним відбитком

З метою пошуку алгоритмів порівняння аудіозаписів було розглянуто застосунки, які виконують подібне завдання, та вже існують на ринку. Мова йдеться про такі відомі програми, як Shazam та SoundHound, які були розроблені з метою надати можливість розпізнавати та ідентифікувати саундтреки за невеликим за довжиною відрізком або навіть наспівуванням. Алгоритми, які використовують ці застосунки, довели свою ефективність завдяки високому показнику якості результатів, тому варто розглянути ідеї, на яких вони базуються, та їх загальну структуру.

Головний напрямок роботи алгоритму, який використовує Shazam для порівняння аудіозаписів, включає в себе створення та аналіз акустичних відбитків. Акустичний відбиток – це представлення аудіозапису на основі його характеристик, які включають в себе такі аспекти, як частота, амплітуда, висота, темп, тембр, ритм тощо, у вигляді унікального хешу. Головна ідея такого підходу полягає в тому, що акустичні відбитки аудіозаписів, які людське вухо ідентифікує, як схожі, теж повинні мати схожий вигляд.

Наступним кроком для ідентифікації вхідного сигналу після його перетворення в унікальний відбиток йде порівняння цього аудіо відбитку з аудіо відбитками сигналів, які містяться в базі даних, та зіставлення з шаблонами.

При цьому, в якості додаткової попередньої обробки, з метою поліпшення якості звуку можуть застосовуватися алгоритми зменшення шуму, завдяки чому підтримується тенденція успішного аналізу навіть в умовах різного середовища і фонового звуку.

Отже, детальний опис усіх кроків даного алгоритму:

1) Отримання спектрограми вхідного сигналу. Спектрограма – це представлення сигналу в залежності від часу та частоти, яке не є оборотним. Це графік, де ось X позначає час, Y - частоту, і кожна точка має різну

насиченість кольору в залежності від амплітуди частоти, чим вона більша, тим темнішим є колір. Для обрахунку виокремлюються короткі вікна запису, до яких застосовуються перетворення Фур'є, яке розкладає їх на частоти. Приклад спектрограми зображено на рисунку 2.1.1 .

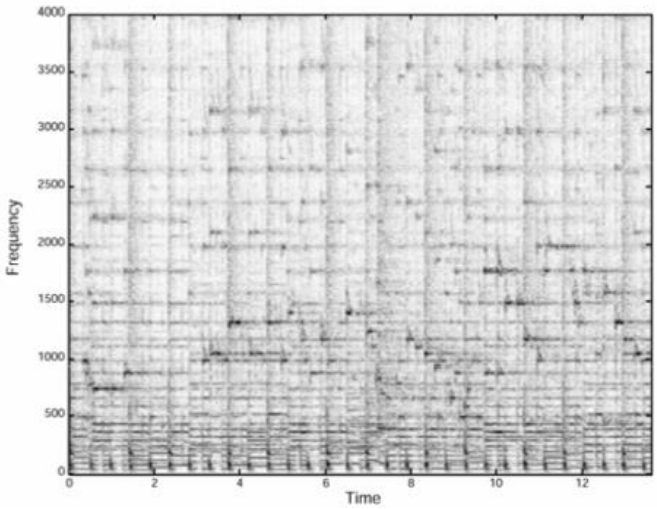

Рисунок 2.1.1 – Спектрограма

2) Фільтрація з метою залишити лише найтемніші точки на графіку спектрограми, з яких далі обиратимуться опорні точки та їх цільові зони, які беруть участь в обрахуванні хешу для кожної точки в цільовій зоні.

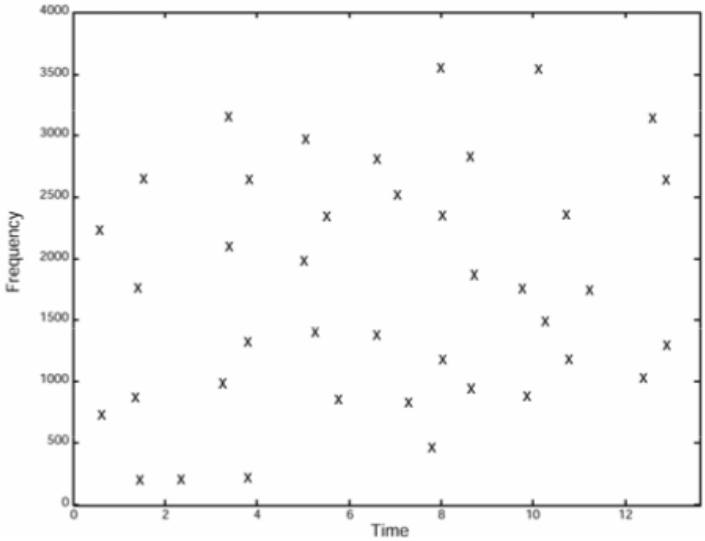

Рисунок 2.1.2 – Спектрограма після фільтрації

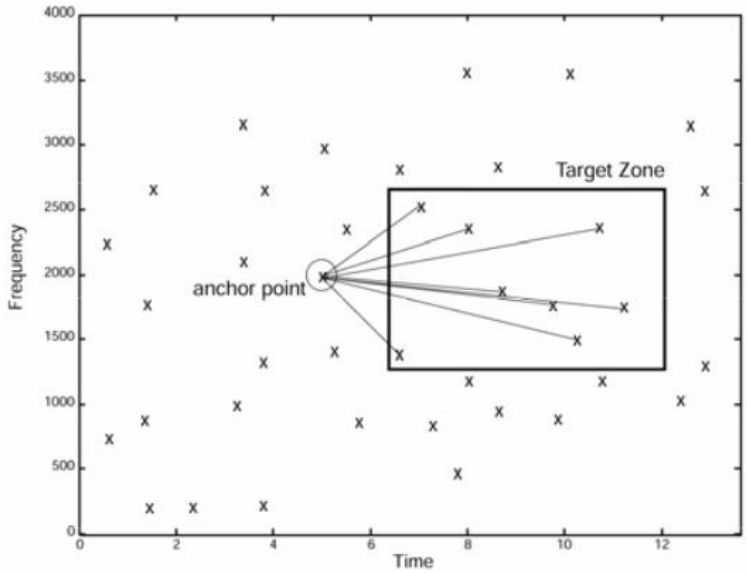

Рисунок 2.1.3 – Вибір опорних точок цільових зон для обрахунку хешу

3) Кожен такий обрахований хеш шукає собі відповідність в базі даних.

4) Коли збіг знайдено, будується графік, який співставляє хеші вхідного запису та запису у базі даних, де точками вказується перетин хешів. Якщо на такому графіку утворюється багато точок, які будуть розміщенні по діагоналі, то це означатиме, що висока ймовірність того, що дані записи збігаються.

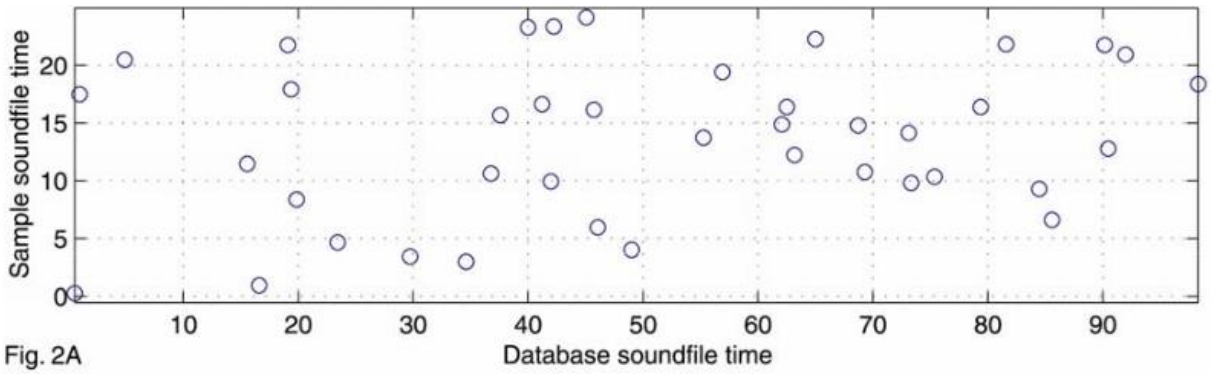

Рисунок 2.1.4 – Порівняння невідповідних записів

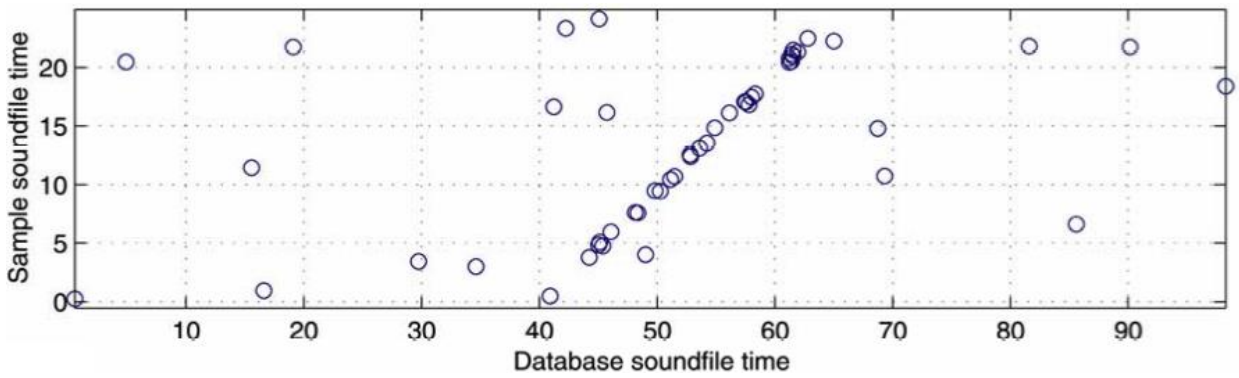

Рисунок 2.1.5 – Порівняння відповідних записів

5) Обраховується гістограма різниць, у випадку збігу на ній будуть “піки”, коли у протилежному випадку коливання амплітуди буде дуже малим.

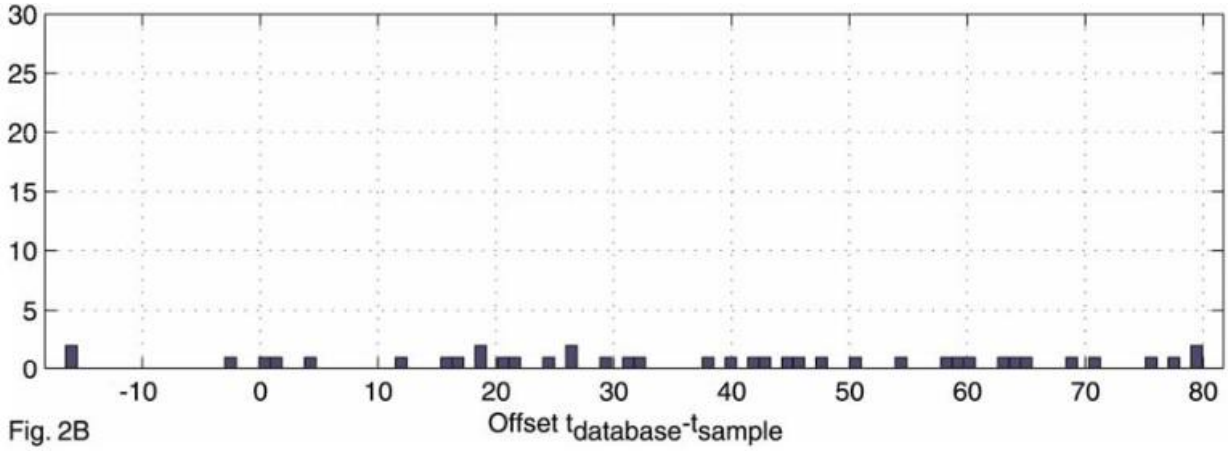

Рисунок 2.1.6 – Гістограма при відсутності збігу

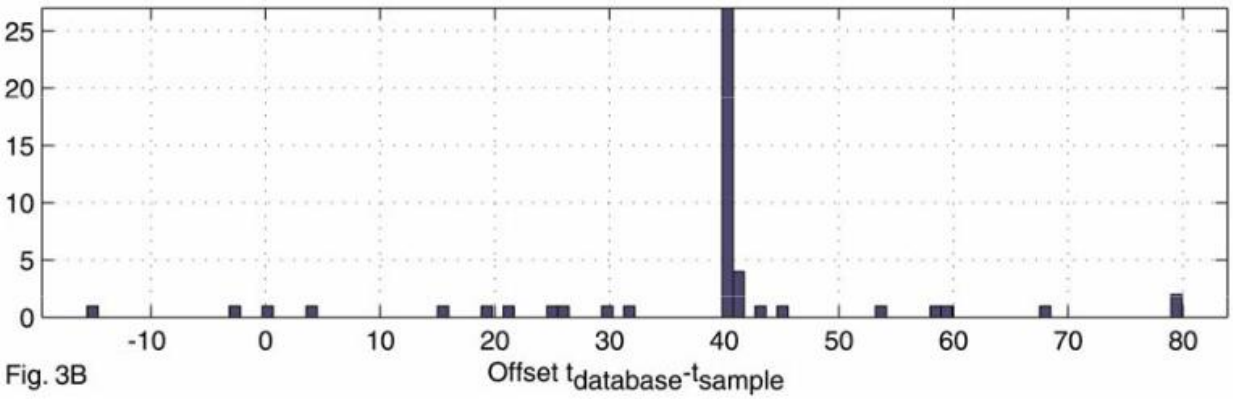

Рисунок 2.1.7 – Гістограма при виявленні збігу

Це головні кроки алгоритму пошуку відповідності сигналів з допомогою аудіо відбитків, але не можна виключати той факт, що він може бути більш ускладненим додатковими етапами з метою покращення результатів пошуку, зменшення часової складності та підвищення точності збігів.

Цей алгоритм було розроблено засновниками Shazam, які запатентували його з метою збереження авторських прав. Його побудова власними силами потребуватиме великої кількості часу та ресурсів. Проте, на його основі можна побудувати спрощений варіант алгоритму, для реалізації якого буде змога використати вже існуючі бібліотеки, наприклад, Dejavu та Chromaprint для Python, які дають змогу обрахувати акустичний відбиток для записів.

2.2. Порівняння аудіо способом прямого аналізу його ознак

2.2.1. *Поняття виокремлення ознак*

Виділення ознак — це процес обчислення певної послідовності ознак для кожного доволі короткого відрізку запису. Припускається, що цей невеликий сегмент аудіо буде достатньо інформативним, щоб мати змогу забезпечити повноцінний аналіз. Виокремлення ознак є ключовим кроком для точної подальшої обробки, який допомагає ідентифікувати вміст файлу, отримати поведінку сигналу та відкинути частини, які не несуть користі, наприклад, шуми. Необроблений сигнал рідко є підходящим для подальшої обробки, тому функції виокремлення ознак є необхідними для отримання коректних результатів аналізу аудіозаписів.

Існує великий перелік ознак, які можна витягнути з аудіо, наприклад Mel-Frequency Cepstral Coefficient (MFCC), який належить до короткочасного спектрального виду ознак, є одним з найбільш популярних способів аналізу звукових сигналів. MFCC є однією з найбільш широко використовуваних та домінуючих функцій для отримання характеристик із аудіо, у яких рівень шуму не є високим, та яка використовує нелінійну шкалу під назвою mel. MFCC - це набір коефіцієнтів, які підкреслюють особливості сигналу, відкидаючи

нерелевантну інформацію, яка несе мало користі. MFCC часто використовуються у задачах розпізнавання вимови та найкращим чином презентують звук таким способом, яким це робить людське вухо.

2.2.2. *Алгоритм обрахування MFCC*

Алгоритм отримання MFCC є таким:

1) Pre-Emphasis. Перш за все застосовується фільтр для виділення високих частот задля того, щоб виокремити їх на фоні низьких, оскільки порівняння відбувається саме за рахунок них. Підвищується амплітуда високочастотних сигналів, тим самим компенсуючи тендецію людського мовлення їх послаблювати. Низькі частоти зазвичай є другорядними і майже не грають ролі при пошуку характеристики звуків. Формула, яку застосовують на даному етапі має такий вигляд:

$$Y_n=X_n - a*X_{n-1}, \qquad (2.2.2.1)$$

де a - коефіцієнт, значення якого лежить в межах від 0,9 до 1.

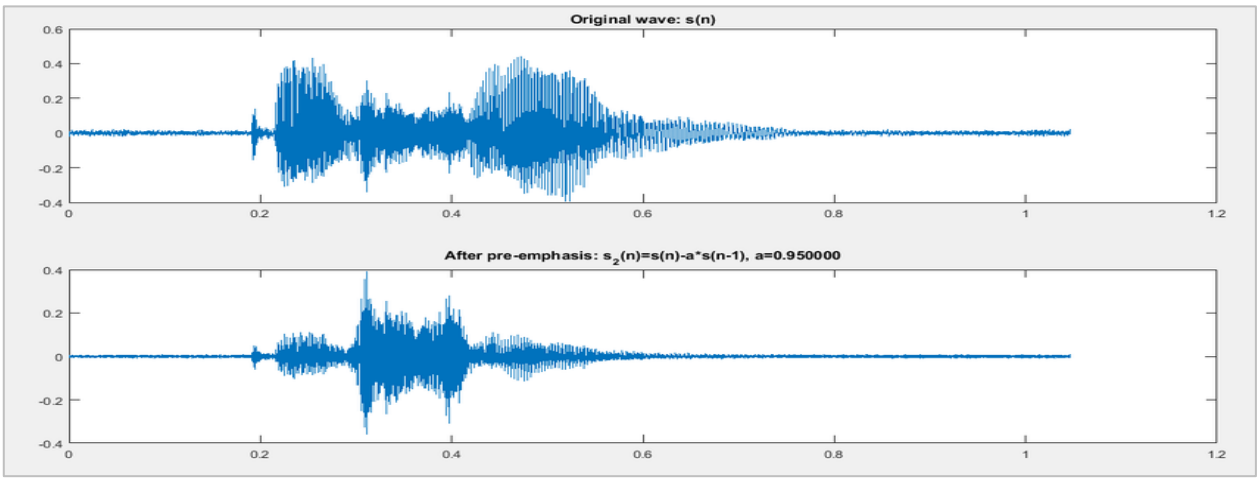

Рисунок 2.2.2.1 – Хвиля сигналу після застосування фільтру

2) Framing. Замість того, щоб аналізувати відразу весь вхідний сигнал, він розбивається на фрейми. Розмір такого кадру варіюється та може доходити до 500 мілісекунд, при цьому присутнє перекриття кадрів, оскільки вікно

Хеммінга, яке застосовується до кожного фрейма, стирає деякий відсоток інформації на початку та в кінці, а перекриття її повертає.

3) Windowing. Застосування методу накладання вікна на кадри має мету уникнути великих коливань на початку та в кінці кадрів та підготовлює їх до наступного кроку. Зазвичай для цього етапу обирається саме вікно Хеммінга.
4) Fast Fourier Transform. У результаті дискретного перетворення Фур'є, сигнал переходить із часової області в частотну, оскільки аналіз аудіо сигналів у частотній області легший, ніж у часовій. Застосовується швидке перетворення Фур'є, оскільки це більш вдосконалений алгоритм, який потребує менше часу на обчислення.
5) Mel-frequency warping. Сприйняття частоти звуку людиною не може бути представлене у лінійному масштабі, тому фактична частота кожного тону, яка обрахована в Гц, перетворюється у суб’єктивну висоту тону, що вимірюється за шкалою, яка називається «мел». Мел-шкала – це часто використовувана частотна шкала, яка є лінійною лише до 1000 Гц, а далі логарифмічною.

   Орієнтовна формула перетворення f у ГЦ в Mel –

$$Mel(f) = 1127*\log_{10}(1 + f/700) \qquad (2.2.2.2)$$

6) Discrete Cosine Transform. Оскільки спектральні коефіцієнти mel є дійсними числами, вони можуть бути перетворені назад у часову область за допомогою дискретного косинусного перетворення, у результаті якого отримуються Mel Frequency Cepstrum Coefficients.

2.2.3. *Оцінка відстані між MFCC різних сигналів*

Наступний крок полягає у правильному виборі алгоритму, з допомогою якого будуть порівнюватись ознаки двох аудіозаписів. Є доволі великий перелік алгоритмів, які підходять для даної задачі. Наприклад, це крос-

кореляція, яка дає змогу виявити подібні патерни, які можуть бути зміщеними у часі. Але більш дієвим у цій ситуації буде використання іншого алгоритму під назвою «динамічне викривлення часу» (Dynamic Time Warping).

Динамічне викривлення часу - це алгоритм, який застосовується при вирішенні завдання визначення подібності послідовностей, які можуть відрізнятись у швидкості чи часі. Він знайшов своє призначення у сфері розробки моделей автоматичного розпізнавання вимови, де суттєвою проблемою була мінливість мови. Цей алгоритм постав рішенням цієї проблеми, оскільки він дає змогу правильно порівняти промову людини, незалежно від того, у якому темпі вона говорить: можливо занадто повільно чи прискорюючись в певні моменти. За допомогою нього в процесі порівняння послідовності вирівнюються, завдяки чому результат відстані до еталону буде максимально правдивим.

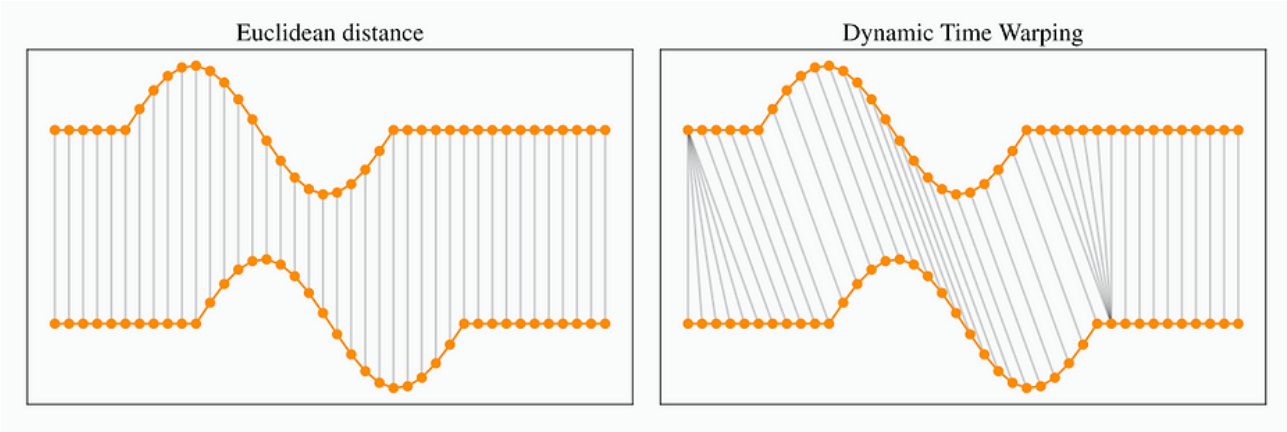

Рисунок 2.2.3.1 – Візуальне порівняння відстані Евкліда та динамічного викривлення часу

Кроки алгоритму:

1) Створення і заповнення матриці відстаней, які обраховуються за формулою, яка обирається заздалегідь. Наприклад, це може бути відстань Евкліда, якщо списки містять об’єкти з характеристиками. Для чисельних послідовностей формула заповнення матриці буде виглядати так

$$M_{i,j} = |A_i — B_j| + \min (M_{i-1,j-1}, M_{i,j-1}, M_{i-1,j}) \qquad (2.2.3.2)$$

2) Починаючи з верхнього правого кутка матриці знайти викривлений шлях до лівого нижнього з найменшими відстанями, які обираються на основі сусідніх значень.

4	13	10	9	8	8	7	9	7
1	10	8	8	9	7	8	6	10
2	10	7	6	6	5	7	6	7
5	9	7	5	4	5	5	8	4
4	5	4	3	3	4	4	6	4
1	2	2	3	4	4	5	3	7
3	2	1	1	2	2	3	5	7
1	0	1	3	6	8	11	11	15
	1	2	3	4	3	4	1	5

Рисунок 2.2.3.2 – Знайдений шлях у матриці відстаней

3) Обрахування кінцевого результату відстані між послідовностями наступним чином: сума усіх елементів шляху, отриманого на попередньому етапі, ділиться на довжину шляху.

   При цьому є перелік правил та обмежень, які повинні виконуватись:

- Кожен індекс з першої послідовності повинен бути зіставлений з одним або кількома індексами з іншої послідовності і навпаки
- Перші індекси обох послідовностей повинні відповідати один одному, однак це не означає, що це має бути їх єдиний збіг.
- Аналогічно, останній індекс з першої послідовності повинен відповідати останньому індексу з іншої послідовності.
- Відображення індексів з першої послідовності в індекси другої послідовності має бути монотонно зростаючим, і навпаки.

Отримане значення відстані між двома аудіозаписами можна аналізувати декількома способами:

1) нормалізувати в межах від нуля до одного;
2) визначити поріг максимальної відстані, перевищення якого буде означати відсутність подібності;

3) класифікувати подібність аудіозапису порівнюючи його відстані до інших аудіозаписів.

2.3. Вимірювання подібності аудіо за допомогою моделювання та зіставлення графу

Новий спосіб порівняння аудіосигналів було запропоновано вченими університетів Гонконгу та Пекіну. В цьому підході було змінено концепцію розбиття сигналу на частини та описано новий підхід, який включає в себе моделювання графу, при чому ефективність такого алгоритму залишається доволі високою.

Для порівняння аудіо, особливо великої довжини, доцільно використовувати не його повністю, а його розбиття на менші частинки, якими оперувати буде набагато зручніше. В загальновідомих підходах прийнято ділити аудіо на фрейми певної довжини. Даний же підхід передбачає розбиття не на кадри, а на сегменти, довжина яких не є попередньо-означена. Це обґрунтовується тим, що маленький за розміром кадр може не містити достатньої інформації для подальшого аналізу, а виділення характеристик з таких фреймів може спотворити і дати неточний аналіз про увесь запис. Тому в якості кращого підходу було запропоновано виділяти саме аудіо сегменти, кожен з яких є набором фреймів, які є акустично однорідними, та використовувати їх для виокремлення ознак.

Сегментація аудіозапису дозволяє охопити коливання змін його характеристик впродовж часу, що робить сегмент більш інформативною та семантично багатою одиницею, яка вдало підходить для задачі порівняння аудіо.

Після того, як аудіокліп розбитий на сегменти, залишається проблема найбільш ефективного пошуку подібності. Схожість двох записів базується на схожості його частин, тому це можна змоделювати у вигляді дводольного графу, де кожна вершина представляє сегмент кожного кліпу, а ребра

з’єднують сегменти з різних множин. Вага ребра - це числове значення подібності пари сегментів, які він з’єднує. Таким чином утворюється зв’язок багато-до-багатьох. Проте, такий граф потрібно перетворити, щоб був зв’язок один-до-одного, і щоб кожному сегменту з одного аудіозапису відповідав максимум один сегмент з іншого. Для того, щоб реалізувати це перетворення, використовується алгоритм, оптимального співставлення (optimal matching).

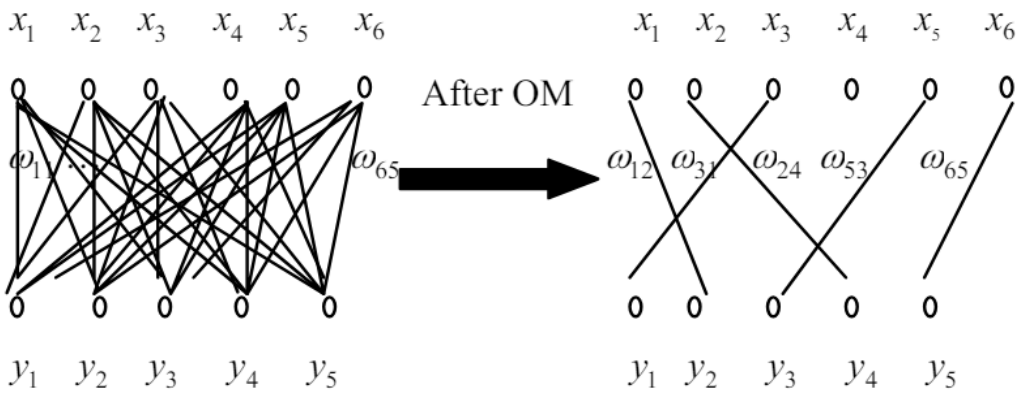

Рисунок 2.3.1 – Перетворення графу у граф зі зв’язком один-до-одного методом оптимального зіставлення

У підході беруться до уваги чотири фактори: акустичний, деталізація, часовий порядок та інтерференція. Деталізація означає ступінь однозначного збігу двох різних сегментів, порядок - подібність у часовій послідовності, а інтерференція показує відсоток невідповідних частин.

Отже, можна виділити конкретні кроки алгоритму:

1) Попередня обробка, яка включає в себе сегментацію аудіозапису та виокремлення окремих характеристик. При сегментації для виявлення змін використовується Байєсівський інформаційний критерій. Після цього кожен сегмент характеризується середнім значенням ознак усіх його кадрів, що можливо завдяки його однорідності. Ознака представляється вектором, який включає в себе логарифм енергії та MFCC.
2) Визначення функції подібності сегментів для подальшої побудови графу. Вона обраховується за формулою

$$Sim_{OM}(s_i, s_j) = \exp(-\,Distance(s_i, s_j)/2), \qquad (2.3.1)$$

де Distance - це функція знаходження відстані Евкліда між векторами;

s_n – сегмент з індексом n.

3) Утворення дводольного повного графу зі зв’язком багато-до-багатьох.
4) Перетворення графу у граф зі зв’язком один-до-одного методом оптимального співставлення. Цей алгоритм побудований на основі угорського алгоритму, також відомого, як алгоритм Куна-Мункреса, який використовується для пошуку ребер з максимальною вагою в дводольних графах.
5) Подібність аудіокліпів за деталізацією та акустичним фактором на основі утвореного графа обраховується за формулою

$$Sim_{OM}(X, Y_k) = \frac{\sum \omega_{ij}}{\max(p, q)}, \qquad (2.3.2)$$

де p і q – це кількість сегментів у кожному записі;

ω – це вага ребра.

6) Подібність за фактором часового порядку обраховується з допомогою методів динамічного програмування. Нехай C – матриця, яка означає кількість пар сегментів, які зійшлись у часовому порядку. Елементи матриці обраховуються за формулою

$$c[i,j] = \begin{cases} 0 & i = 0 \text{, or } j = 0 \\ c[i-1, j-1] + 1 & i, j > 0 \text{, } (x_i, y_j) \in M \\ \max(c[i, j-1], c[i-1, j]) & i, j > 0 \text{, } (x_i, y_j) \notin M \end{cases}, \qquad (2.3.3)$$

де M – множина пар сегментів, які з’єднані ребром у графі після застосування алгоритму оптимального зіставлення.

Значення подібності за фактором часового порядку можна визначити за формулою

$$Sim_{DP}(X, Y) = C[p, q] / p \quad (2.3.4)$$

7) Подібність за фактором інтерференції, яка означає невідповідність сегментів, обраховується за формулою

$$Sim_{IF}(X, Y) = 2 * |M| / (p + q) \quad (2.3.5)$$

8) Визначення кінцевого результату подібності аудіозаписів на основі проміжних значень подібності за кожним з факторів відбувається наступним чином, представленим формулою (2.3.6).

$$Sim_{clip}(X, Y_k) = \sum_{i \in \{OM, DP, IF\}} \alpha_i Sim_i(X, Y_k) \quad (2.3.6)$$

2.4. Обґрунтування вибору алгоритму

В результаті аналізу вищеперелічених алгоритмів, було обрано метод пошуку схожості аудіозаписів способом обрахування Mel-Frequency Cepstral Coefficient та подальшому обрахуванню відстані методом динамічного викривлення часу.

Цей алгоритм було обрано завдяки наступним перевагам:

- достатньо точні результати;
- не вимагається об’ємних обчислень та складних структур даних, результат обраховується лише математичними формулами і перетвореннями;
- алгоритм динамічного викривлення часу забезпечує можливість розпізнавати схожі властивості навіть при різній швидкості та темпу вимови;
- висока точність при відсутності зайвих шумів.

Висновок про визначення відповіді користувача, як правильної чи неправильної, ґрунтується на визначені того, до якого з еталонних варіантів відповідей відстань запису буде найменша.

3. ВИБІР ЗАСОБІВ ДЛЯ РОЗРОБКИ

3.1. JavaScript у основі розробки клієнтської та серверної сторін застосунку

Для реалізації поставленої задачі, було обрано розробляти веб-сайт. Javascript у таких випадках є незмінною та універсальною основою, на якій будується переважна частина веб-додатків. Це можна пояснити переліком причин.

1) Динамічна типізація та відсутність звичних обмежень, які є в інших мовах, робить процес розробки більш гнучким та легким.
2) Її екосистемна складова охоплює велику спільноту розробників, які забезпечують постійну підтримку бібліотек та мови в цілому. Це означає, що у загальному доступі існують великі об’єми ресурсів, що полегшують процес розробки, оскільки завдяки цьому шанс вирішення будь-якої проблеми, яка може зустрітися на шляху, наближається до 100%.
3) Універсальність використання JavaScript як на клієнтському, так і на серверному боці за допомогою NodeJs дає можливість спростити розробку проєкту завдяки відсутності необхідності постійного перемикання з однієї мови програмування на іншу.
4) З другого пункту наслідується той факт, що існує великий перелік новітніх бібліотек та фреймворків, які варто використовувати для оптимізації написання застосунків. Саме тому в якості додаткових ресурсів для розробки було обрано бібліотеку Express для серверної сторони та фреймворк Vue JS для клієнтської.

Обрання Vue js в якості фреймворку пояснюється тим, що реактивність, яка притаманна для нього, дала змогу значно зменшити об’єм коду, який був би необхідний для реалізації тих же функцій. Також присутній фактор модульності, який дає можливість будувати компоненти, які можна перевикористовувати.

Express js – це бібліотека, яка надає можливість будувати серверну частину додатків з використанням різноманітних middleware, які полегшують

реалізацію таких головних функцій, як авторизація, коректна обробка HTTP запитів тощо.

3.2. Meyda як бібліотека для обчислення MFCC

Meyda - це бібліотека, яка дозволяє обробляти аудіодані та отримувати з них широкий перелік ознак, які можна використовувати для подальшого машинного навчання, різного виду аналізу, візуалізації звукових сигналів тощо. Meyda реалізовує набір стандартних аудіо функцій, які широко використовуються в різних сценаріях, що потребують перетворення звуку в числову інформацію.

Бібліотека може використовуватись як в браузері у режимі реального часу завдяки підтримки Web Audio Api, так і на серверній частині. Meyda підтримує вилучення трьох різних видів ознак:

1. Time-Domain (наприклад, RMS (Root Mean Square), energy, Zero Crossing Rate);
2. Frequency-Domain (наприклад, Mel Frequency Cepstral Coefficients та інші);
3. Perceptual (Perceptual Loudness, Perceptual Spread, Perceptual Sharpness);

Конкретно для реалізації обраного алгоритму пошуку міри подібності записів з великого переліку можливих ознак треба обраховувати лише MFCC.

Алгоритм, за яким бібліотека обраховує коефіцієнти, базується на реалізації іншої бібліотеки, написаної для Python, librosa. На вхід функції подається зразок довжиною степені двійки, а на виході за замовчуванням отримується 13 коефіцієнтів.

3.3. Бібліотека «dynamic-time-warping»

Це бібліотека для Javascript, яка імплементує пошук шляху та найменшу відстань між двома послідовностями об’єктів. При створенні об’єкту DynamicTimeWarping потрібно передати в конструктор два масиви та функцію,

яка приймає на вхід два об’єкти такого ж типу, як в масиві, та повертає відстань між ними. З допомогою функцій getDistance() та getPath() отримуються результати обрахунків, а саме відстань між вхідними послідовностями та шлях у вигляді списку масивів з двох елементів, які позначають індекси об’єктів.

Переваги саме цього пакету полягають в тому, що на вхід можуть подаватись списки з будь-яких об’єктів, а не лише з чисел, як у деяких інших бібліотеках, та є можливість самому визначити функцію, яку потрібно використати для обрахування відстані.

3.4. MongoDb у ролі бази даних

Обов’язковою потребою при розробці було використання сховища даних, необхідного для коректної роботи сервісу. Для досягнення цієї мети було обрано таку базу даних, як MongoDb. Головною причиною цього стало те, що формат зберігання даних у ній близький до формату JSON, який чудово підтримується JavaScript, легко перетворюється в об’єкти та використовується для обміну даних між клієнтом і сервером. Також ця база даних є доволі швидкою та високопродуктивною, що теж є вагомою перевагою.

4. РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1. Вміст веб-сайту

Як вже згадувалось – проєкт представляє собою веб-сайт з наступним переліком сторінок:

- головна сторінка, де зібрана вся необхідна користувачу інформація, що потрібна перед початком використання сервісу;
- сторінка з тренуванням, де можна обрати спосіб тестування
    a) з варіантами відповідей у вигляді номеру обраного складу;
    b) з можливістю перевірити вимову на правильність;
- сторінка з правилами, де зібраний перелік існуючих правил, які застосовуються до слів;
- сторінка для кожного слова окремо, на якій є правило, що пояснює принцип наголосу даного слова та перелік подібних слів;
- сторінка логіну та реєстрації;
- сторінка профілю користувача, на якій зображено особистий прогрес навчання та вивчені слова по рівнях.

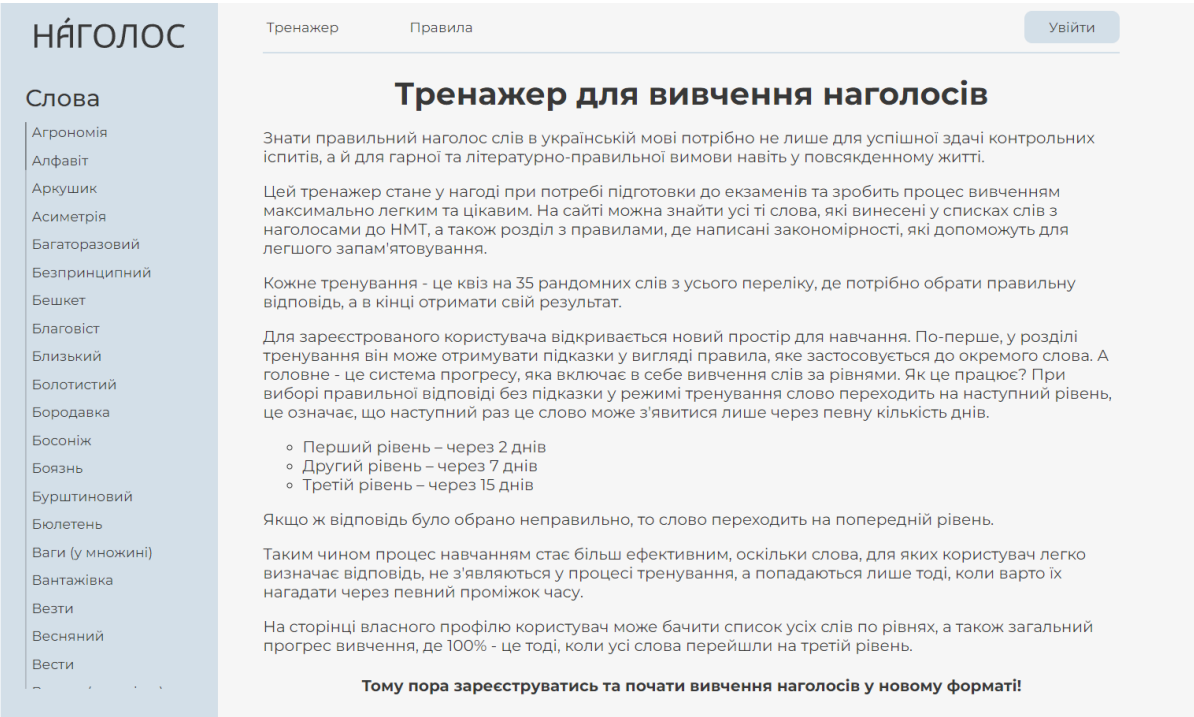

Рисунок 4.1.1 – Головна сторінка

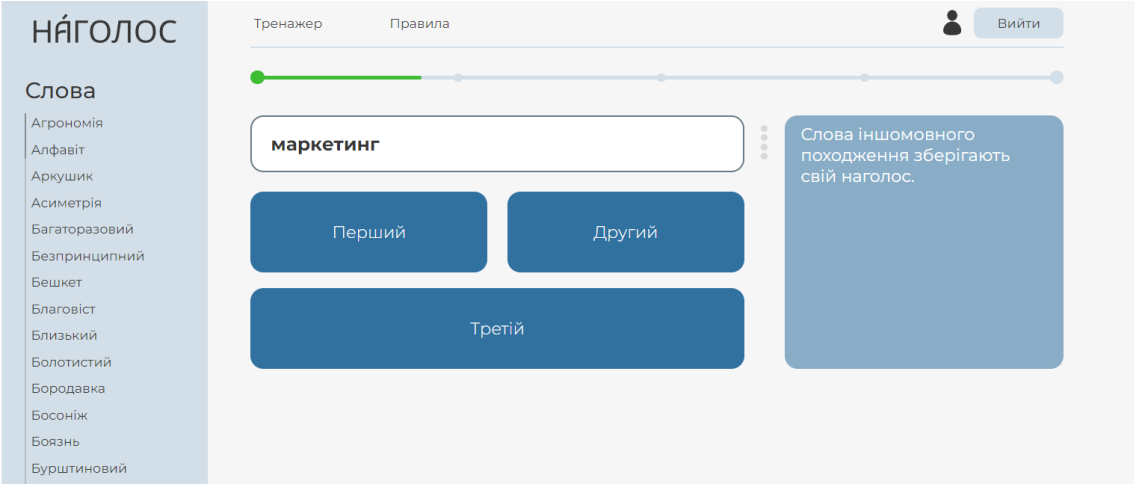

Рисунок 4.1.2 – Тренування з варіантами відповідей

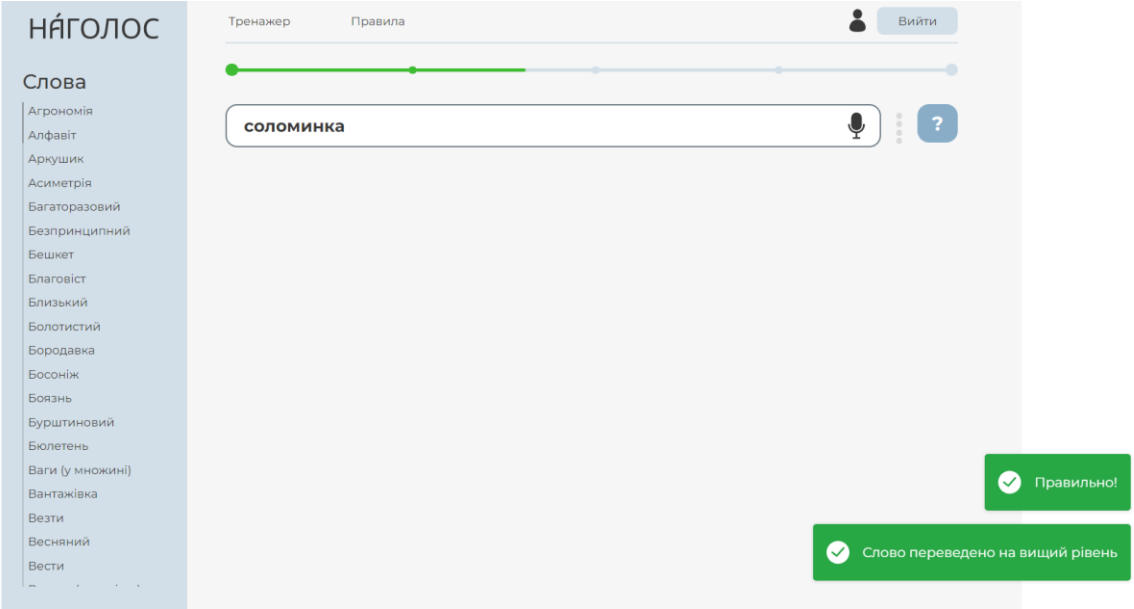

Рисунок 4.1.3 – Тренування вимови, сценарій правильної відповіді

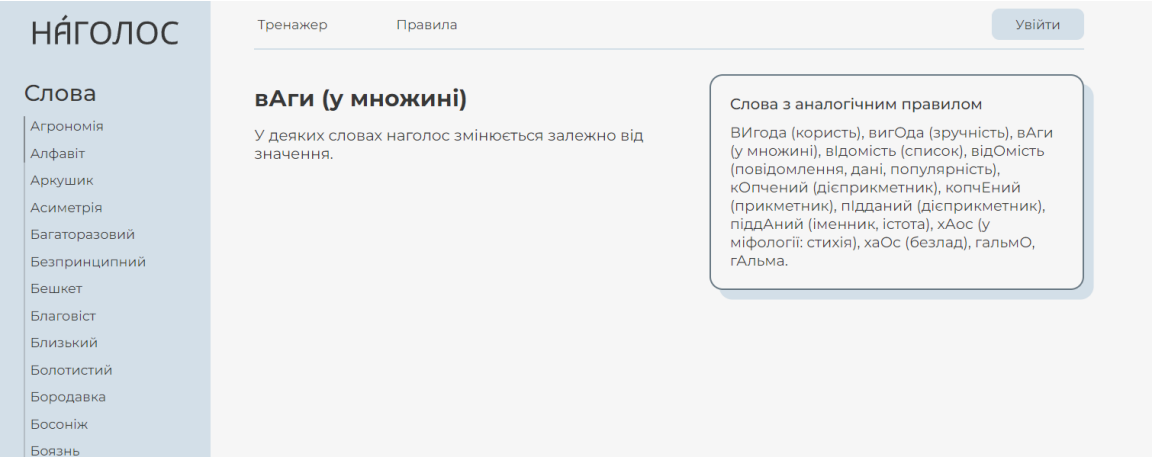

Рисунок 4.1.4 – Сторінка з довідкою про слово

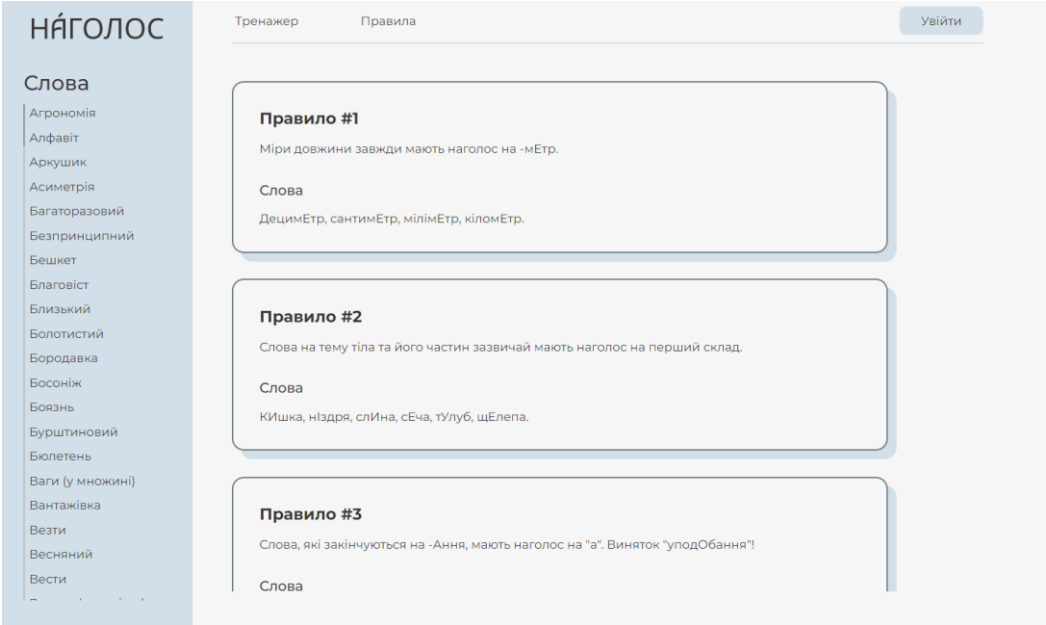

Рисунок 4.1.5 – Сторінка з правилами

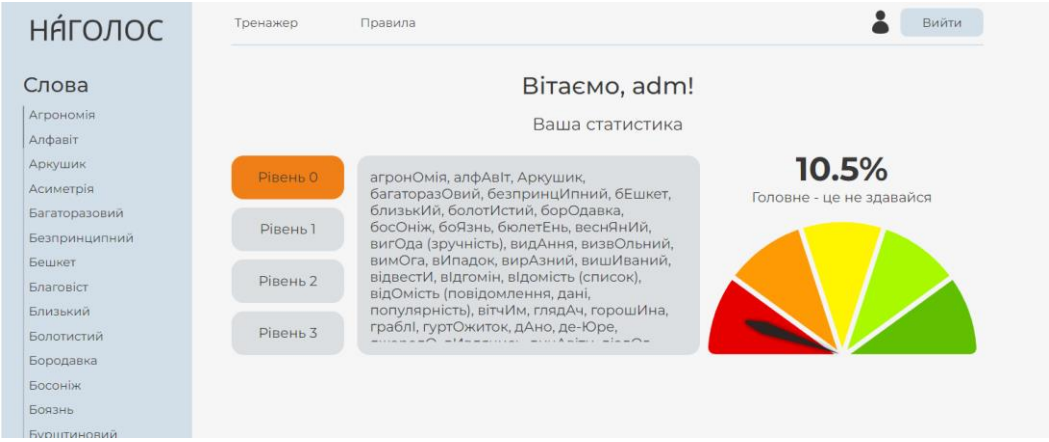

Рисунок 4.1.6 – Сторінка профілю користувача

4.2. Структура проєкту

Проєкт складається з серверної та клієнтської частин, взаємодія між якими реалізована з допомогою HTTP запитів.

Задачі серверу включають в себе:

- авторизацію та аутентифікацію користувача;
- взаємодія з базою даних;

- класифікація вхідного запису з вимовою користувача, як правильного чи неправильного, методом порівняння значень його схожості з еталонним аудіо;

База даних включає в себе такі колекції, як:

- колекція з усіма правилами;
- колекція з усіма словами, варіантами відповідей та правильними відповідями до тестів;
- колекція з усіма правилами;
- колекція зареєстрованих користувачів;
- колекція відповідності правил та слів;
- колекція відповідності слів з рівнями кожного користувача.

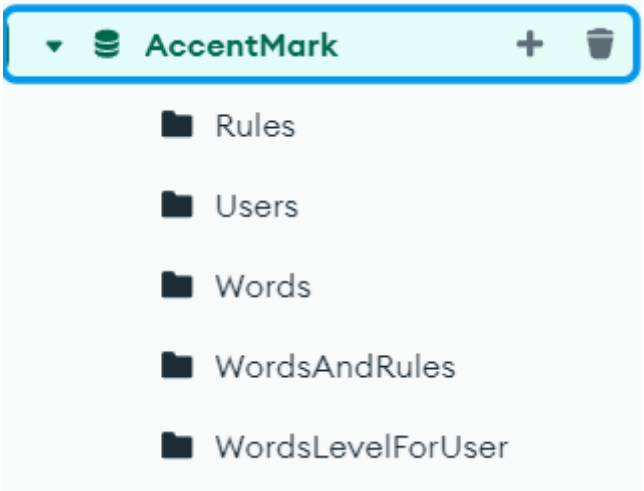

Рисунок 4.2.1 – Сторінка профілю користувача

Клієнтська частина охоплює усі сторінки та компоненти, з яких вони складаються, включає можливість зчитування інформації з мікрофона користувача та відправки її на сервер, а також усю логіку взаємодії користувача з сервісом.

ВИСНОВКИ

Результатом виконання цієї роботи стало дослідження методів порівняння аудіозаписів на схожість з подальшою розробкою сервісу для тренування правильної вимови, який забезпечує удосконалений процес вивчення наголосів слів в українській мові.

В ході написання роботи було досліджено певну кількість алгоритмів та методів пошуку схожості аудіозаписів, які є успішно застосованими в реальності для вирішення проблем та полегшення життя. Ідея сервісу, який був розроблений, має практичну мету, яка може стати у нагоді великому переліку аудиторії, яка включає в себе школярів і випускників, для яких вивчення такого аспекту мови, як наголос, є необхідною для успішної здачі екзаменів, а також просто людей, яким цікава українська мова з її правилами.

Було реалізовано один з досліджених алгоритмів порівняння аудіо та інтегровано його в реальний проєкт.

Процес розробки дав змогу дослідити такі нові бібліотеки для javascript, як Meyda та dynamic-time-warping, а також методи побудови веб-сервісів, які включають в себе використання популярних фреймворків, як Vue js та Express.

Загальна ідея даного сервісу не є унікальною, але реалізація з включенням можливості автоматичної перевірки вимови додає нові перспективи та переваги.

Список використаної літератури

1. Dynamic-time-warping for Javascript github [Електронний ресурс]: https://github.com/GordonLesti/dynamic-time-warping
2. SparseDTW: A Novel Approach to Speed up Dynamic Time Warping / Ghazi Al-Naymat, Sanjay Chawla, Javid Taheri, 2 – 4 с.
3. Dynamic Time Warping. In: Information Retrieval for Music and Motion / Springer, Berlin, Heidelberg (2007), 2 с.
4. Meyda: an audio feature extraction library for the Web Audio API / Hugh Rawlinson, Nevo Segal, Jakub Fiala, London
5. Meyda. Audio feature extraction for JavaScript [Електронний ресурс]: https://meyda.js.org/
6. Study of MFCC and IHC Feature Extraction Methods With Probabilistic Acoustic Models for Speaker Biometric Applications / Sithara A , Abraham Thomas, Dominic Mathew, Kochi, 1– 4 с.
7. The Implementation of Speech Recognition using Mel-Frequency Cepstrum Coefficients (MFCC) and Support Vector Machine (SVM) method based on Python to Control Robot Arm / D Anggraeni , W S M Sanjaya, M Y S Nurasyidiek, M Munawwaroh, Bandung, 1 – 5 с.
8. Audio similarity measure by graph modeling and matching / Yuxin PENG, Chong-wah NGO, Cuihua FANG, Xiaoou CHEN, Jianguo XIAO, Singapore (2006).
9. An Industrial-Strength Audio Search Algorithm / Avery Li-Chun Wang, London.