Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РОЗПІЗНАВАННЯ ЗВУКОВИХ ОБРАЗІВ
SOUND PATTERN RECOGNITION SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-130
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ПОПОВ О.С.
(ПРІЗВИЩЕ та ініціали)
Керівник КОЦУР М.І.
(ПРІЗВИЩЕ та ініціали)
Рецензент СКРУПСЬКИЙ С.Ю.
(ПРІЗВИЩЕ та ініціали)

2024

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
“____” _____2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПОПОВА Олександра Сергійовича
(ПРІЗВИЩЕ, ім’я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення розпізнавання звукових образів. Sound Pattern Recognition Software

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім’я, по батькові)
затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 03 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Олександр ПОПОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Михайло КОЦУР
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 84 с., 3 табл., 27 рис., 3 дод., 19 джерел.

PYTHON, PYCHARM, ЗВУК, МУЗИЧНИЙ ТВІР, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об’єкт дослідження – процес розробки програмного забезпечення для розпізнавання звукових образів.

Предмет дослідження – програмні засоби для підтримки процесу розпізнавання звукових образів.

Мета роботи – розробка програмного забезпечення для розпізнавання звукових образів.

Матеріали, методи та технічні засоби: мова програмування Python, середовище розробки PyCharm.

Результати. Розроблено програмне забезпечення для підтримки процесу розпізнавання звукових образів за допомогою мови програмування Python та середовища розробки PyCharm.

Висновки. Виконано проєктування програмного забезпечення для підтримки процесу розпізнавання звукових образів. Розроблено програмне забезпечення для підтримки процесу розпізнавання звукових образів. Здійснено тестування розробленого програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Галузь використання – програмні засоби розпізнавання звукових образів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 84 pages, 3 tables, 27 figures, 3 appendixes, 19 sources.

PYTHON, PYCHARM, SOUND, MUSIC COMPOSITION, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of developing software for recognizing sound patterns.

The subject of the research is software for supporting the process of recognizing sound images.

The purpose of this work is to develop the software for recognizing sound patterns.

Materials, methods and technical tools: Python programming language, PyCharm development environment.

Results. The software tools to support the audio pattern recognition process using the Python programming language and the PyCharm development environment have been developed.

Conclusions. The software to support the process of sound image recognition has been designed. The software to support the process of sound image recognition has been developed. The developed software to support the process of sound image recognition has been tested.

Scope of use – software tools for sound image recognition.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

IDE – Integrated Development Environment;

IoT – Internet of Things (Інтернет речей);

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

З динамічним розвитком інформаційних технологій голосові інтерфейси набувають широкого поширення. У таких технологіях для розпізнавання звукових образів, зокрема, голосових команд широко використовуються спеціальні програмні модулі, розроблені на основі відповідного математичного забезпечення [1], [2].

Програмні засоби розпізнавання звукових образів використовуються у пристроях Інтернету речей для створення звукових датчиків і систем безпеки, які можуть реагувати на різні звукові події. У розвитку аудіо та музичних технологій відповідні програмні засоби використовуються для розпізнавання музичних композицій, аналізу аудіо інформації, фільтрації контенту за звуковими параметрами. На виробничих підприємствах та в офісних задачах розпізнавання голосу може бути використане для автоматизації завдань, таких як транскрипція зустрічей, створення нотаток, взаємодія з програмами та системами [1]-[6].

У сфері охорони здоров'я програмні засоби розпізнавання звукових образів можуть використовуватися для розпізнавання медичних термінів, аудіо документації, а також для моніторингу та аналізу звуків, пов'язаних з пацієнтами. Загалом, розпізнавання звукових образів відіграє важливу роль у різних сферах, де автоматизація та взаємодія з програмними системами на основі голосу може поліпшити зручність, ефективність та безпеку використання технологій різного призначення [1]-[6].

Проте деякі програми розпізнавання звукових образів можуть бути менш ефективними в умовах шуму або при низькій якості вхідних аудіо записів, що може становити проблему в реальних умовах, де звукове середовище може бути непередбачуваним. Деякі програмні засоби можуть використовувати алгоритми розпізнавання звуків, які вимагають значних обчислювальних ресурсів, що може призводити до затримок у відповіді системи та високих витрат енергії, особливо на портативних пристроях. Деякі

програмні системи розпізнавання звуків характеризуються обмеженою точністю, особливо при розпізнаванні акцентів, діалектів чи особливих голосових характеристик, що в деяких випадках є неприйнятним. Інші програмні засоби можуть використовувати моделі розпізнавання, які можуть бути спеціалізованими для певних мов або областей, і не завжди ефективно працювати з іншими мовами чи аудіо даними з різних областей [1]-[6].

Тому актуальною є розробка програмного забезпечення для розпізнавання звукових образів. У дипломній кваліфікаційній роботі бакалавра розв’язується актуальне завдання розробки програмного забезпечення для розпізнавання звукових образів.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв’язати такі задачі:

– виконати аналіз предметної області та програмних засобів для розпізнавання звукових образів;

– здійснити проєктування програмного забезпечення для розпізнавання звукових образів;

– створити програмне забезпечення для розпізнавання звукових образів;

– виконати тестування розробленого програмного забезпечення для розпізнавання звукових образів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття про звукові образи та їх розпізнавання

Розпізнавання звукових образів є процесом обробки, вивчення та розшифрування записаних аудіо сигналів за допомогою різноманітних технологій, таких як передові алгоритми глибокого навчання. Голосова аналітика широко використовується в різних галузях, включаючи розваги, охорону здоров'я та виробництво [1]-[3].

Проаналізуємо галузі застосування технології та програмних засобів розпізнавання звукових образів [1]-[3].

Розпізнавання мовлення відноситься до здатності комп'ютерів розрізняти вимовлені слова за допомогою природної обробки мови. Це дозволяє управляти комп'ютерами, мобільними телефонами та іншими пристроями за допомогою голосових команд, а також диктувати текст, замість введення його вручну [2]-[4].

Метою розпізнавання голосу є ідентифікація осіб на основі унікальних характеристик їхнього голосу, а не просто розпізнавання окремих слів. Застосовується у системах безпеки для аутентифікації користувачів, наприклад, для визначення співробітників і клієнтів у банківській галузі за їхнім власним голосом [3]-[6].

Використання технологій та програмних засобів розпізнавання звукових образів для розпізнавання музики є популярною функцією деяких програмних застосунків, що допомагає визначати невідомі композиції за коротким зразком. Ще одним застосуванням аналізу аудіо записів є класифікація за жанрами, наприклад, для групування пісень за певними категоріями [1]-[6].

Розпізнавання звукового оточення фокусується на виявленні оточуючих звуків, що може призвести до численних переваг у сферах автомобільної та виробничої промисловості. Розуміння можливості використання розпізнавання звукових образів для застосунків Інтернету

речей (IoT) стає важливою задачею, розв’язання якої дозволяє здійснювати корекції, що роблять життя людей більш комфортним та безпечним. Наприклад, технологія аналізу звукового фону машини може полегшити прогнозоване технічне обслуговування для моніторингу стану обладнання та запобігання дорогим поломкам [1]-[6].

Сфера охорони здоров'я є ще однією областю, де використання методів та програмних засобів розпізнавання звукових образів виявляється корисним. Програмні засоби розпізнавання звукових образів забезпечують неінвазивний дистанційний моніторинг пацієнтів, наприклад, шляхом аналізу звуків кашлю, чхання та інших, що може служити попереднім оглядом, допомагаючи визначити стан конкретного пацієнта або оцінити рівень зараженості в громадських місцях [1]-[6].

Аудіо інформація відображає звуки у цифровій формі та утримує ключові характеристики вихідного сигналу. Звук являє собою коливальну хвилю, яка передається через середовище, таке як повітря або вода, і, врешті-решт, досягає вух людини. При аналізі аудіо даних важливо враховувати три основні властивості: час, амплітуду та частоту [1]-[6].

Проаналізуємо основні властивості звукових образів (період, амплітуду та частоту). Період – це тривалість конкретного звуку, іншими словами, скільки секунд потрібно для завершення одного циклу вібрацій [1]-[6]. Амплітуда – це інтенсивність звуку, виміряна в децибелах (дБ), яку людина сприймає як гучність. Частота, виміряна в герцах (Гц), показує, скільки коливань відбувається за секунду. Люди сприймають частоту як низькі або високі тони [1]-[6].

Навіть при тому, що частота є об’єктивним параметром, висота є суб’єктивною. Людський слух охоплює діапазон від 20 до 20 000 Гц. За висновками дослідників, більшість людей вважає всі звуки нижче 500 Гц низькими тонами, такими, як шум двигуна літака. Натомість для людей все, що перевищує 2000 Гц (наприклад, свист), вважається високою нотою [1]-[6].

Щодо форматів файлів аудіо даних, подібно до тексту і зображень, аудіо є неструктурованим матеріалом, тобто це не організовані дані в табличній формі з рядками і стовпцями. Аудіо файли доцільно зберігати в різноманітних форматах. Формат VAV або VAVE (Waveform Audio File Format), розроблений Microsoft і IBM, є файловим форматом без втрат або необробленим форматом, який не стискає оригінальний аудіо запис. Формат MP3 є найпоширенішим форматом файлів та дозволяє легко зберігати музику на портативних пристроях та відправляти її через Інтернет. Навіть при стисненні MP3 забезпечує прийнятну якість звуку [1]-[6]. Рекомендується використовувати файли AIFF і WAV для аналізу, оскільки вони зберігають всю інформацію, що міститься в аналогових звуках. Також слід зазначити, що не всі ці та інші аудіо файли можна безпосередньо ввести в моделі машинного навчання, оскільки для розуміння звуку потрібно змінювати дані [1]-[6].

Основи обробки аудіо даних вимагають розуміння кількох ключових термінів, які будуть зустрічатися на кожному етапі від збору аудіо даних до розробки прогнозів за допомогою машинного навчання. Цікаво відзначити, що аналіз звуку полягає у взаємодії з зображеннями, а не просто в прослуховуванні [1]-[6]. Форма хвилі – це базове візуальне представлення звукового сигналу, яке вказує на зміну амплітуди з часом [1]-[6]. Спектр або спектральна діаграма – це графік, де ось абсцис представляє частоту звукової хвилі, а ось ординат – її амплітуду. Такий тип візуалізації аудіо даних дозволяє аналізувати частотний зміст, але ігнорує часовий аспект [1]-[6]. Спектрограма – це детальне представлення сигналу, яке об'єднує всі три аспекти звуку. На графіку можна побачити час по осі абсцис, частоту по осі ординат та амплітуду за допомогою кольору. Що гучнішим є звуковий образ, тим яскравішим є колір, а вимкнений звук відображається чорним. Присутність трьох вимірів на графіку є дуже корисною, що дозволяє спостерігати, як частота змінюється з часом, перевіряти наповненість звукового простору і виявляти різні проблемні зони (наприклад, шум) та шаблони [1]-[6].

Перетворення Фур'є є математичною операцією, яка розкладає сигнал на компоненти з різними амплітудами та частотами. Перетворення Фур'є використовується для розгляду сигналу під різними кутами та перетворення хвиль у спектральні діаграми для проведення частотного аналізу. Це потужний інструмент для розуміння сигналів і їх обробки [1]-[6]. Алгоритм швидкого перетворення Фур'є використовується для обчислення перетворення Фур'є, а короткочасне перетворення Фур'є представляє собою послідовність перетворень Фур'є, що перетворюють аудіо сигнал в спектрограму [1]-[6].

Програмне забезпечення для аналізу звукових образів автоматизує завдання, пов'язані з обробкою звуку, і включає такі функції, як імпорт аудіо даних, додавання анотацій (тегів), організація та розподіл записів, усунення шуму, перетворення сигналів в візуальні представлення (криві, спектральні діаграми, спектрограми), виконання операцій попередньої обробки, аналіз часового і частотного вмісту, вилучення звукових характеристик і навіть навчання моделей машинного навчання [1]-[6].

Етапи аналізу звукових даних включають збереження аудіо даних у стандартних форматах файлів, підготовку даних для проєкту машинного навчання, вилучення звукових характеристик з візуальних представлень звукових даних і вибір моделі машинного навчання для їхнього навчання голосовим функціям [1]-[6].

Отримання аудіо даних та збір звукових інформацій може відбуватися за трьома шляхами: використання безкоштовних аудіо тек або наборів аудіо даних, покупка їх у постачальників даних або збір інформації за допомогою експертів у відповідній галузі [1]-[6].

Безкоштовні джерела даних доступні в середовищі Інтернеті, проте у цьому випадку немає можливості контролювати якість, кількість даних і загальний підхід до запису [1]-[6]. Аудіо теки представляють собою колекції безкоштовних звукових доріжок, що охоплюють різні теми. Ресурси розподіляють аудіо записи, навколишні звуки, шум та мову на різноманітні теми. Звукові бібліотеки не спеціально створені для проєктів машинного

навчання, тому при розробці відповідного програмного забезпечення важливо працювати над створенням набору даних, їх маркуванням та контролем якості [1]-[6].

З іншого боку, при розробці програмного забезпечення розпізнавання графічних образів набори аудіо даних створюються з урахуванням конкретних завдань машинного навчання. У випадку комерційних голосових наборів машинного навчання набагато легше довіряти цілісності даних, ніж у безкоштовних. В деяких випадках пропонуються набори даних для моделей розпізнавання мовлення, класифікації звуків навколишнього середовища, розрізнення джерел звуку та інших застосувань [1]-[6].

Професійні набори даних при розробці відповідного програмного забезпечення є важливою частиною розробки моделей для розпізнавання образів. Для досягнення відповідної мети необхідно отримати конкретні дані, які не були доступні відкритим кодом [1]-[6]. Якщо при розробці програмного забезпечення є необроблені дані, тобто записи у форматі аудіо файлів, потрібно підготувати їх для використання в машинному навчанні [1]-[8].

Зберігання звукових даних включає в себе позначення або анотацію даних, що передбачає позначення необроблених даних правильними відповідями для навчання моделі. Процес навчання полягає в тому, що модель вчиться розпізнавати шаблони в нових даних та робити відповідні прогнози на основі наданої анотації. Таким чином, якість і точність анотацій визначають успіх проєктів машинного навчання [1]-[6]. Незважаючи на те, що анотація може використовувати програмні засоби і автоматизацію, вона в основному здійснюється вручну професійними експертами відповідної галузі [1]-[8].

Попереднє оброблення аудіо даних є важливим етапом для поліпшення точності прогнозувань, на додаток до збагачення даних значущими мітками. Проаналізуємо ключові етапи, які використовуються у програмних засобах для розпізнавання мовлення та класифікації голосу [1]-[6]. Процес кадрування включає розбиття безперервного аудіо потоку на короткі сегменти (кадри) однакової довжини, що дозволяє подальшу обробку сегментів [1]-[8].

Використання вікон даних є фундаментальним методом обробки аудіо даних, спрямованим на зменшення спектрального сигналу, що може викликати розмиття частот та втрату точності амплітуди. Існує різноманіття віконних функцій, які можна використовувати для різних типів сигналів. В загальному, всі вікна виконують аналогічні завдання: знижують амплітуду на початку та в кінці кожного кадру для збереження середнього значення, або збільшують її в середині для вирівнювання [1]-[8]. Деякі методи сприяють уникненню втрати значущої інформації, отриманої під час віконного перегляду та забезпечують суттєве перекриття між сусідніми кадрами, що дозволяє безпечно змінювати їх місцями без ризику спотворення. У цьому випадку вихідний сигнал коректно формується з вікна [1]-[8].

Аудіо функції або дескриптори являють собою характеристики сигналу, обчислені на основі візуалізації попередньо оброблених аудіо даних. Ці характеристики належать до одного з трьох доменів: часовий домен, представлений формою хвилі; частотний діапазон, відображений спектральною діаграмою; часові та частотні області, представлені спектрограмою. Характеристика часового домену може бути витягнута безпосередньо з вихідної форми сигналу. Важливо зауважити, що хвилі не надають багато інформації щодо реального звучання пісні. Вони лише вказують на зміну амплітуди з часом [1]-[8].

Амплітуда пікового відхилення відстежує максимальні значення амплітуди в кожному кадрі та демонструє, як вони змінюються відносно часу. З використанням амплітуди пікового відхилення можна автоматично вимірювати тривалість різних частин аудіо сигналу. Амплітуда пікового відхилення широко використовується для виявлення виникнення конкретного сигналу та класифікації музичних жанрів [1]-[8]. Короткочасна енергія сигналу відображає зміну енергії в короткочасній формі. Це потужний інструмент для відокремлення голосних і неголосних елементів. Середньоквадратична енергія сигналу надає розуміння середньої енергії сигналу. Його можна розрахувати за формою хвилі або спектрограмою. У

першому випадку результати будуть отримані швидко. Однак спектрограма забезпечує більш точне представлення енергії в часі. Цей критерій є особливо корисним для сегментації аудіо та класифікації музичних жанрів [1]-[8]. Рівень нульового перетину визначає кількість перетинів хвилі сигналу горизонтальною віссю кадру. Це одна з ключових акустичних характеристик, яка широко використовується для визначення розпізнавання мови та відокремлення шуму від тиші, а також музики від мови [1]-[8].

Оскільки процес розпізнавання графічних образів включає перетворення сигналів у спектральні графіки або спектрограми за допомогою перетворення Фур'є, важко відокремити функції частотного діапазону від функцій часового діапазону. Однак частотний зміст виявляє багато важливих властивостей звуку, які стають видимими або важко виявити в часовому діапазоні [1]-[8].

Найпоширеніші характеристики частотної області включають: середні частоти, які розділяють спектр на дві області однакової амплітуди. Іншою характеристикою є співвідношення сигнал/шум, яке порівнює гучність потрібного звуку з фоновим шумом. Коефіцієнт енергії пропускної здатності вказує на співвідношення між діапазонами високих і низьких частот [1]-[8].

Властивості частотно-часової області поєднують часові та частотні компоненти та використовують різні типи спектрограм як візуальне представлення звуку. Спектрограму можна отримати з хвилі за допомогою короткочасного перетворення Фур'є [1]-[8]. Однією з найпопулярніших груп частотно-часових характеристик є середньо частотний коефіцієнт. Такі характеристики підходять для сприйняття людським вухом, використовуючи чоловічий слух та спектрограми [1]-[8].

Для навчання розпізнавальних моделей дослідники даних використовують відповідні характеристики з часової та частотної областей, створюючи детальні профілі абразивних звуків та звуків хропіння [1]-[8]. Оскільки аудіо файли подаються у вигляді спектрограм, вони ідеально підходять для розпізнавання зображень, використовуючи глибокі нейронні

мережі. Існує багато ефективних архітектурних моделей для виявлення та класифікації звуку, і ми розглянемо два методи для діагностики розладів сну за допомогою звуку [1]-[8].

Нейронні мережі довгострокової пам'яті відзначаються здатністю розпізнавати довгострокові залежності в даних та зберігати інформацію про попередні кроки. Згорткові нейронні мережі визнані ефективними в галузі розпізнавання зображень, що робить їх природним вибором для обробки спектрограм [1]-[8].

Проаналізуємо набори алгоритмів розпізнавання звукових образів. Алгоритми розпізнавання музики можуть розпізнавати особливості, які роблять пісні унікальними, і визначати їхні жанри. Ще одним класом алгоритмів розпізнавання мовлення є ті, що використовуються в різних сценаріях. Вони застосовуються в бізнес-середовищі для запису бізнес-переговорів і в особистому житті, де розпізнавання голосу використовується в віртуальних помічниках (чат-ботах) та особистих асистентах [1]-[8]. Втрата слуху може становити значущий виклик для людей, які взаємодіють із акустичним середовищем. Таким чином, моделі розпізнавання голосу використовуються для створення слухових апаратів та когнітивних інтервенцій, спрямованих на покращення якості життя для людей похилого віку. Такі можливості вже використовуються виробниками побутової техніки, зокрема, з функціями розпізнавання голосу [1]-[8].

У сегменті моніторингових програмних систем моделі розпізнавання голосу можна поділити на дві категорії: моніторинг голосу та розпізнавання голосу. З урахуванням зростання уваги до безпеки є великий попит на аудіо системи, які спроможні виявляти різноманітні сигнали автоматичного спостереження (наприклад, сигнали тривоги, звуки крадіжки чи насильства, або навіть звуки, незвичайні для промислових зон). З іншого боку, акустична технологія стає важливою складовою особистої та корпоративної безпеки. Акустична технологія також має великий потенціал як частина потужних

проєктів штучного інтелекту, таких як розпізнавання голосу, аналіз відео та виявлення подій [1]-[8].

Розпізнавання звуків оточуючого світу, зокрема, моделі розпізнавання мовлення, дозволяють вивчати оточуючий світ і отримувати більше інформації про різноманітність звуків у нашому оточенні. Технологія природного розпізнавання голосу використовується в океанографії, прогнозу погоди, термометрії та інших галузях. Більшість завдань з розпізнавання голосу потребують іншого підходу, але їх суть залишається незмінною. Проаналізуємо етапи алгоритмів розпізнавання голосу. Модель розпізнавання голосу спочатку записує звуки, які вона сприймає. Вони можуть бути живими або відтвореними. Цей етап необхідний для того, щоб створити основу для подальшого аналізу звуків. Потім модель проводить аналіз записів, що означає порівняння записаних звуків із даними, на яких модель була навчена. Якщо процес навчання ефективний, а якість даних висока, прогноз буде швидким і точним [1]-[8].

Програмне забезпечення та моделі розпізнавання образів можуть забезпечити розв’язання різних завдань, від пошуку відповіді на запит користувача до трансформації мовлення в редагований текст або перегляду бази даних аналогічних звуків за допомогою методу класифікації для визначення джерела звуку [1]-[8]. Для такого завдання, як перетворення голосу в текст, що є стандартним у домашніх асистентах і чат-ботах, необхідно пройти принаймні три основних етапи для конвертації людського введення (запитання або запиту) у машинну відповідь [1]-[8].

Програмні засоби автоматичного розпізнавання мовлення використовуються для сегментації мовлення (як прямого, так і записаного). Потім алгоритм аналізує звуки за допомогою засобів обробки природної мови і передбачає, які слова в конкретній мові можуть відповідати. Це дозволяє машині знаходити правильну відповідь на користувацький запит. Зрештою, синтез тексту в мову перетворює відповідь у людське мовлення [1]-[8].

Розробка програмного забезпечення та алгоритмів розпізнавання голосу є складним завданням, що включає кілька ключових етапів. З одного боку, потрібна кваліфікована команда розробників та інженерів даних для створення моделей машинного навчання. З іншого боку, велика частина часу витрачається на збір, аналіз, обробку та маркування даних [1]-[8]. Другий етап часто недооцінюється інженерами даних через його менш продуктивний, виснажливий та складний характер порівняно з розробкою моделей машинного навчання. Однак для ефективного тренування алгоритмів важливо мати надійний, високоякісний та великий набір даних для синтезу розпізнавальних моделей [1]-[8]. Зокрема, проставляння міток даних – це процес додавання міток до різних частин даних. Це критично, оскільки машини використовують інший спосіб аналізу звуків, ніж люди. Мітки призначені для навчання машин розуміти, що ми хочемо, щоб вони виявляли в звуках [1]-[8].

Оскільки цей процес вимагає часу та зусиль, багато компаній шукають партнерів із збагаченням даних. Це дозволяє компаніям уникнути необхідності фокусуватися на основному продукті та допомагає визначити пріоритети творчого процесу.

Програмні засоби розпізнавання голосу також можуть сприяти комунікації між людьми. Використання алгоритмів, таких як персональні перекладачі та асистенти для осіб із обмеженими можливостями, свідчить про те, що штучний інтелект із розпізнаванням голосу може підняти нас на новий рівень взаєморозуміння [1]-[8].

Розпізнавання мовлення, також відоме як розпізнавання голосу, представляє собою технологію, що здатна конвертувати усну мову людини в текст. Ця технологія може функціонувати автоматично або навчатися вимові конкретного користувача. Протягом останніх років розпізнавання мовлення стрімко розвивалося і використовується у різних програмах нашого повсякденного життя, а також в особистих сферах. Окрім перетворення мовлення на текст, технології розпізнавання голосу також використовуються

для розроблення програм для розпізнавання голосу, що надають допомогу особам з обмеженими можливостями, літнім людям, людям із проблемами слуху або тим, хто не може друкувати з будь-яких причин [1]-[8].

Методи та засоби розпізнавання звукових образів використовуються для створення програмних систем голосового керування через голосові команди для розумних пристроїв вдома, розпізнавання музики, моніторингу навколишнього середовища, спостереження, розробки акустичних систем для постійного розпізнавання, автоматичного виявлення та ідентифікації тривог [1]-[8].

Програмні засоби автоматичного розпізнавання мовлення засновані на цифровій обробці аудіо сигналів, визначенні людської мови та її конвертації в текст. Це швидко зростаючий сектор розробки програмного забезпечення, заснованого на нейронних мережах. Застосування підходів глибокого навчання та нейронних мереж дозволило значно покращити якість точного розпізнавання мовлення. Ключовим фактором успіху є якість навчальних наборів для створення акустичних і лінгвістичних моделей мовлення. Можливість навчання нейронних мереж великою мірою сприяла підвищенню рівня якості програмного забезпечення розпізнавання мови [1]-[8].

Методи та алгоритми розпізнавання графічних образів засновані на ознайомленні зі зразками типових послідовностей слів, що існують у природній мові, і тому вони можуть ідентифікувати лінгвістичну структуру цільової мови. Кожна нова порція звукової інформації впливає на якість обробки наступного звуку, що призводить до деяких помилок, які враховує нейронна мережа [1]-[8]. Загальні принципи технології розпізнавання графічних образів можна розглядати на прикладі розпізнавання голосу у телефоні під час використання функції пошуку. Насправді пристрій отримує не повністю вимовлене слово, а невпорядкований звуковий сигнал без жорстких правил [1]-[8]. Програмне забезпечення для розпізнавання мовлення аналізує це речення, сказане людиною, таким чином [1]-[8]: телефон записує голосове повідомлення, нейронна мережа аналізує потік мови, звукова хвиля

розбивається на фонеми, нейронна мережа викликає свою модель і присвоює звуку символ, склад або слово, визначається порядок, у якому слова розпізнаються програмним забезпеченням, а невідомі слова розміщуються відповідно до контексту. Після цього інформація, отримана з попередніх двох кроків, об’єднується, і в результаті виходить перетворення мови в текст [1]-[8].

Нейронні мережі, що використовуються для розпізнавання звукових образів, є комп'ютерними системами, які складаються з індивідуальних одиниць, що з'єднані між собою як елементи, подібні до природних нейронів. Основні переваги використання нейронних мереж полягають у їх здатності до навчання та узагальнення інформації. Процес машинного навчання використовується для тренування нейронних мереж [1]-[8].

Процес навчання дозволяє нейронній мережі виявляти складні взаємозв'язки між вхідними та вихідними даними. Здатність узагальнення інформації дозволяє мережі надавати точні відповіді на основі неповних або викривлених даних. Зі збільшенням обсягу інформації мережа стає ще більш ефективною [1]-[8]. Проте методика навчання є вкрай трудомістким підходом, який вимагає великого спектру учбових даних [1]-[8].

Основне завдання нейронної мережі в розпізнаванні звукових образів полягає в тому, щоб визначити, яка буква відповідає певному звуку в аудіо записі. Потім ці букви об'єднуються, формуючи слова, а слова формують повні речення. Розробники нейронної мережі тренують її на обробленому наборі даних, щоб вона могла відзначати звуки [1]-[8]. Набір даних включає аудіо записи, але необроблена версія без міток була б некорисною. Тому експерти тимчасово позначають такі записи текстом, іншими словами, вони додають текст до звукових доріжок. Навчальний набір даних складається з колекції аудіо записів із текстовими описами. Зазвичай аудіо фрагменти не перевищують десяти секунд. Нейронній мережі подаються набори звуків і тексту як вхідні дані, і вона повинна навчитися асоціювати ці звуки з конкретними літерами та словами [1]-[8].

Після завершення навчання штучний інтелект може виконувати завдання, подібні до тих, які раніше виконували експерти: розбивати аудіо запис на короткі сегменти і намагатися правильно передбачити кожну відповідну букву та звук [1]-[8]. Освоївши найімовірніші літери на аудіо записі, штучний інтелект спробує визначити, яке саме слово вони утворюють. Нейронна мережа використовує словник для порівняння можливих символів всередині. У результаті формується набір відомих слів, які потім об'єднуються в речення. Окрім самого процесу розпізнавання, важливо, щоб отриманий текст був змістовним, послідовним і добре організованим [1]-[8].

Збіг і змістовність технології розпізнавання мовлення головним чином забезпечуються кількістю і точністю текстів, які нейронна мережа може обробити на етапі навчання. Чим більше і якісніше аудіо записи, оброблені штучним інтелектом, включаючи різні інтонації, емоції, дикторів і контекстний вміст, тим краще прогноз. Щоб отримати інформацію необхідної якості, експерти повинні накопичити не менше кількох тисяч годин аудіо файлів даних, які потім передаються алгоритму, що навчає модель розпізнавання голосу [1]-[8].

Проаналізуємо види технологій розпізнавання звукових образів. Сучасна технологія створення програмного забезпечення для розпізнавання мовлення часто використовує рекурентні нейронні мережі, які лежать в основі всіх сучасних служб розпізнавання голосу, музики, зображень, облич, об’єктів і тексту. Вони є дуже ефективними для обробки тексту та передбачення найімовірніших контекстуальних слів, якщо вони не розпізнані [1]-[8].

Рекурентні нейронні мережі можуть використовувати свою внутрішню пам’ять для обробки послідовностей будь-якої довжини і для аналізу серій подій у часі або послідовних просторових ланцюжків. Таким чином, вони особливо корисні для обробки складних завдань, де щось ціле розбивається на частини, наприклад, розпізнавання мови [1]-[8].

Можливість рекурентних нейронних мереж точно розрізняти звуки та формувати з них слова, незалежно від типу та якості вимови, а також точно

передбачати слова в контексті без неоднозначності через фоновий шум чи неоднозначність, дозволяє розробляти якісне програмне забезпечення розпізнавання звукових образів [1]-[8].

Довгострокова короткочасна пам'ять, яка була запропонована для рекурентних нейронних мереж, спеціально спрямована на те, щоб розширити можливості цих мереж отримувати дані в контексті на великій відстані від сегментації даних, які вони обробляють [1]-[8].

Згорткові нейронні мережі являють собою спеціальну архітектуру штучної нейронної мережі, що спрямована на точне розпізнавання звукових образів, зокрема розпізнавання мови [1]-[8].

Ця мережева архітектура значно точніше розпізнає звукові об’єкти, оскільки враховує двовимірну топологію зображення. Згортка широко використовується професіоналами для будь-якого сигналу, чи то відео, звук або зображення [1]-[8].

Суть згорткової мережі полягає в тому, що кожен сегмент зображення множиться по частинах на матрицю згортки, а результат додається і зберігається в тому самому місці вихідного зображення. Хоча в основному використовується для обробки зображень, цей метод також ефективо може бути застосований до тексту та аудіо [1]-[8]. Аудіо дані перетворюються у зображення, яке отримує назву спектрограма, що являє собою візуальне відображення аудіо сигналу у формі частотного спектру, який змінюється з плином часу. За допомогою згорткових нейронних мереж можна аналізувати цей тип аудіо подання для розпізнавання мовлення [1]-[8].

Проаналізуємо процес навчання моделі розпізнавання звукових образів. Незалежно від мети проєкту, процес створення моделі розпізнавання голосу включає наступні етапи. Першим етапом є визначення завдання, на якому експерти з розпізнавання звукових образів повинні чітко зрозуміти, яке завдання повинна виконувати модель, який тип голосових даних їй потрібен, і як їх збирати та маркувати. На цьому етапі важливо розробити відповідні

вказівки та документацію для керування розробкою майбутніх моделей розпізнавання голосу [1]-[8].

На наступному етапі відбувається визначення цілей проєкту. Цілі проєкту визначатимуть тип аудіо інформації та програмне забезпечення, яке буде використовуватися для збору та запису даних. Тип навчального матеріалу залежить від конкретних умов проєкту, наприклад, чи буде модель використовуватися для аналізу телефонних дзвінків чи аудіо книг [1]-[8].

Після цього обирається модель машинного навчання. Для проведення навчання можна обрати відповідний метод розпізнавання мовлення із наявних та, за необхідності, налаштувати його відповідно до конкретних вимог [1]-[8]. Додатково можна використовувати відкритий вихідний код для програмного забезпечення розпізнавання мовлення. Деякі з цих програмних інструментів мають як відкритий вихідний код, так і закритий, і вони мають заздалегідь навчені набори даних для розпізнавання голосу та генерації необхідних текстів. Інші можуть постачатися без набору даних, і розробники повинні самостійно створювати навчальні модулі. Однак деякі проєкти можуть вимагати від розробників створення власних моделей розпізнавання мовлення з самого початку. Цей тип розробки є ефективним з економічної точки зору для великих компаній, які потребують підвищеного рівня безпеки [1]-[8].

Модель машинного навчання, яка, ймовірно, буде відома голосами, повинна бути навчена на основі колекції аудіо записів, перетворених на текст. Таким чином, на цьому важливому етапі експерти збирають великі обсяги аудіо даних для тренування майбутніх моделей машинного навчання. На різних вебсайтах є доступні набори даних для розпізнавання мовлення [1]-[8]. Також організації можуть створювати набори даних із записаних розмов з дотриманням застосовних юридичних процедур. Наявність реального мовлення у наборі даних для розпізнавання мовлення має важливе значення, оскільки люди не читають текст у повсякденному житті. Тому нейронні мережі, які навчаються лише свідомому мовленню, можуть виявити обмежену ефективність в розпізнаванні реального мовлення [1]-[8].

Для забезпечення ефективності програмної системи розпізнавання мови експерти повинні збирати звукові файли, які максимально точно відображають суть цільових даних, які модель буде розпізнавати в майбутньому. Моделі потрібно тренувати, адаптуючи їх до особливостей конкретного завдання. Кожна галузь використання має свої особливості, і це слід враховувати під час розробки та підготовки файлів даних для створення моделей та програмних засобів розпізнавання звукових образів [1]-[8].

На етапі навчання розпізнавальної моделі при створенні програмного забезпечення обираються аудіо записи таким чином, щоб вони містили анотації, щоб модель мала уявлення про очікувані результати. Для цього можна використовувати спеціальне програмне забезпечення, завдяки якому аудіо сегментам призначаються відповідні текстові дані [1]-[8]. Після запису всіх даних експерти можуть використовувати їх для тренування моделей для розпізнавання мовних шаблонів. Після досягнення бажаного рівня продуктивності моделі розпізнавання голосу, практикам слід продовжити контроль якості. Цей етап важливий для постійного моніторингу послідовності розпізнавання, щоб переконатися, що продуктивність розпізнавання голосу завжди залишається на високому рівні [1]-[8].

Технології розпізнавання мови та звукових образів пройшли значний еволюційний шлях, переходячи від простих механізмів до складних нейронних мереж. У сучасний час розпізнавання мови з використанням передових технологій набуває популярності, постійно розвиваючись та пропонуючи все більш вдосконалені методи обробки текстової взаємодії [1]-[8].

Відзначимо, що спеціальні програмні системи перетворюють мовлення в текст для людей з обмеженими можливостями слуху чи проблемами з письмом. Розвиток технологій розпізнавання мовлення виявляється надзвичайно важливим. Інформаційні технології та програмні засоби розпізнавання звукових образів не лише поліпшують якість життя та піднімають його на новий рівень, але й надають можливість людям з

обмеженими можливостями вести повноцінний спосіб життя, працювати та розвиватися [1]-[8].

Визначено, що розпізнавання звукових образів є процесом обробки, вивчення та розшифрування записаних аудіо сигналів за допомогою різноманітних технологій, таких як передові алгоритми глибокого навчання. Проаналізовано галузі застосування технології та програмних засобів розпізнавання звукових образів. Проаналізовано основні властивості звукових образів, формати файлів аудіо даних. Досліджено етапи розроблення розпізнавальних моделей та програмного забезпечення для розпізнавання звукових образів.

1.2 Аналіз програмного забезпечення для підтримки процесу розпізнавання звукових образів

Проаналізуємо програмне забезпечення для підтримки процесу розпізнавання звукових образів [9]-[15].

Програмне забезпечення Shazam (рис. 1.1) є програмою, що дозволяє розпізнавати музичні твори [9], [10], [13].

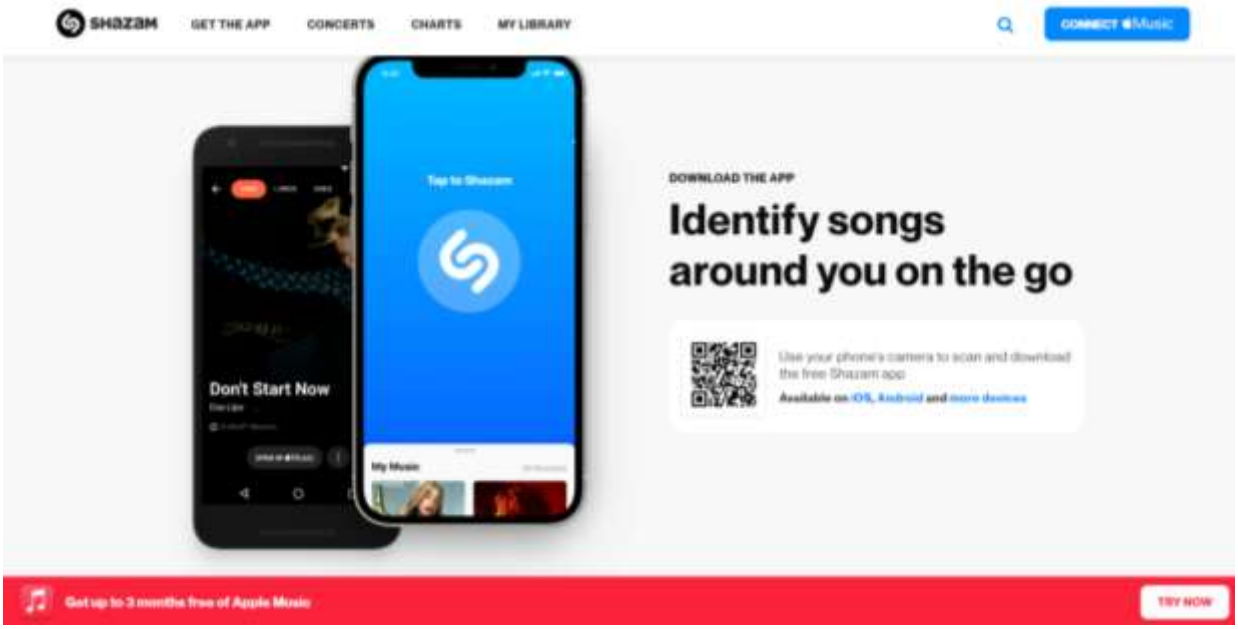

Рисунок 1.1 – Програмне забезпечення Shazam [13]

Програмне забезпечення Shazam відображає улюблені композиції користувача на відповідній панелі, надаючи повний огляд музичних вподобань. Користувачі програмного забезпечення Shazam можуть відтворювати попередні перегляди, ділитися ними, відкривати треки, переглядати музичні відео, знаходити пов'язані композиції та переглядати інші пісні від того ж виконавця [9], [10], [13].

Для пошуку певної музики користувач може використовувати поле у верхньому правому куті. Під час користування програмою користувачу програмного забезпечення Shazam пропонується список з рекомендаціями для музичного прослуховування [9], [10], [13].

Програмне забезпечення Shazam пропонує функцію, яка безперервно слухає музику та розпізнає її, навіть у тих випадках, коли програма не активна. У режимі офлайн Shazam зберігає отримані аудіо дані. Shazam досить добре інтегрується в Android та iOS за допомогою різноманітних засобів [9], [13].

Щоб ідентифікувати музичну композицію за допомогою програмного забезпечення Shazam, необхідно спрямувати телефон на джерело музики, після чого програма дозволить негайно отримати інформацію про деталі музичного твору або поділитися цим знанням з друзями та родиною [9], [13].

Основними характеристиками програмного забезпечення Shazam є такі [9], [10], [13]:

– засоби миттєвого розпізнавання музики: програмне забезпечення Shazam надає можливість швидко та точно визначити назву пісні, яка грає на радіо або в іншому джерелі звуку. Для цього необхідно навести телефон на джерело музики, і програма надасть інформацію про композицію [9], [13];

– можливість збереження та поширення інформації про розпізнані музичні твори: користувачі програмного забезпечення Shazam можуть зберігати відомості про виявлені пісні. Також є можливість ділитися своїми творами з іншими користувачами через соціальні мережі [9], [10], [13];

– автоматичне відстеження нових виконавців: Виконавці, яких ви знаходите за допомогою Shazam, автоматично додаються до списку нових

знахідок. Це дозволяє легко слідкувати за новими музикантами та відкривати їхні твори [9], [10], [13];

– пов'язана інформація: програмне забезпечення Shazam рекомендує пов'язану музику та новинки виконавців, які цікавлять користувачів, щоб вони мали своєчасну інформацію стосовно своїх вподобань [9], [10], [13];

– можливість прослуховування музичних текстів та перегляду відео: програмне забезпечення Shazam дозволяє прослуховувати тексти пісень та переглядати відповідні відео [9], [10], [13];

– інтеграція з іншими музичними сервісами: програмне забезпечення Shazam дозволяє користувачам слухати пісні через платформи Apple Music, Spotify, Rdio, а також радіо Pandora [9], [10], [13];

– автоматичне розпізнавання у фоновому режимі: функція Auto Shazam безперервно прослуховує навколишній звук і розпізнає музику, навіть коли програма не активна [9], [10], [13];

– соціальні можливості: користувачі можуть побачити, чим цікавляться їхні друзі, та поділитися своїми вподобаннями через соціальні мережі [9], [13];

– засоби візуального розпізнавання: програмне забезпечення Shazam підтримує візуальне розпізнавання, що дозволяє визначати інформацію з QR-кодів та зображень [9], [10], [13].

Серед недоліків програмного забезпечення Shazam можна відзначити обмежений функціонал у безкоштовній версії, деякі додаткові функції та сервіси Shazam доступні лише в платній версії програми. Крім того, у деяких випадках, особливо при живих концертах або радіо-трансляціях, Shazam може виявити труднощі у розпізнаванні музики через шум та змінені умови. Хоча Shazam інтегрується з певними музичними сервісами, в інших випадках обмежена підтримка та можливості інтеграції. Іноді Shazam може неправильно розпізнати пісню, особливо в разі наявності кількох версій музичного твору. Крім того, у шумних або переповнених звуками середовищах може знижуватися точність розпізнавання Shazam [9], [10], [13].

Програмне забезпечення Deepgram (рис. 1.2) призначено для розпізнавання мовлення користувачів та дозволяє здійснювати перетворення усної мови в письмовий текст із високою точністю, навіть у випадку різних акцентів у носіїв мови [12], [14].

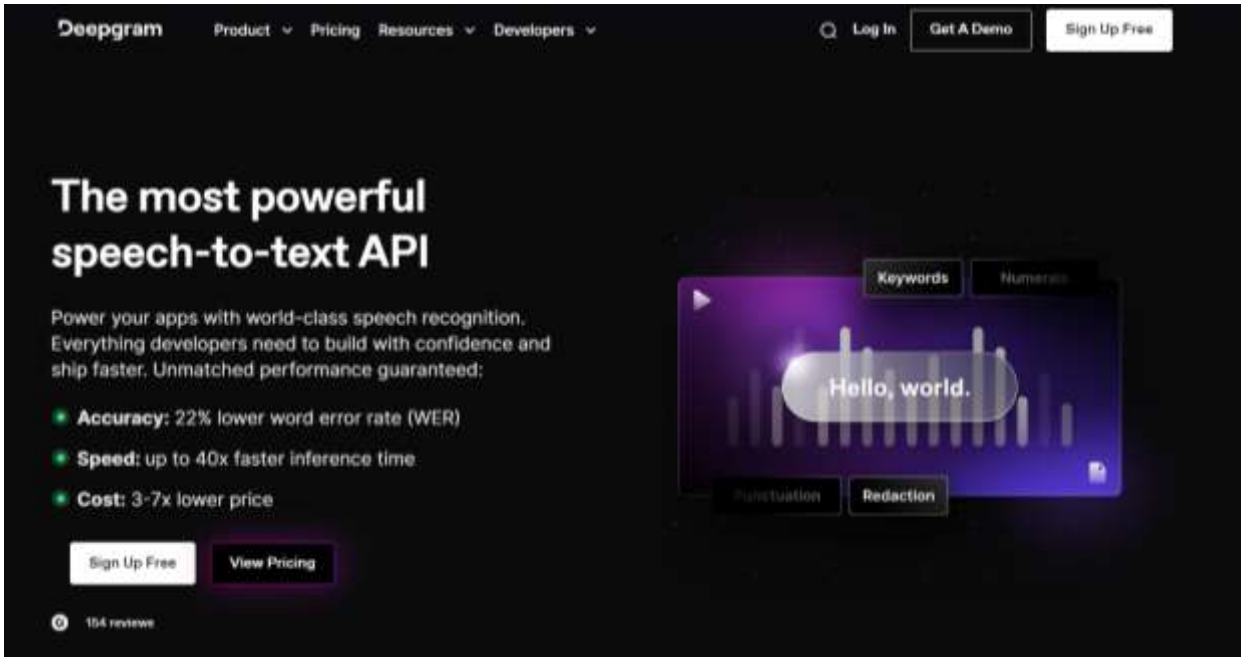

Рисунок 1.2 – Програмне забезпечення Deepgram [12], [14]

Програмне забезпечення Deepgram ефективно обробляє великі обсяги аудіо даних, що робить його ефективним рішенням для компаній, які проводять транскрибування дзвінків, зустрічей або взаємодію з клієнтами. Місією програмного забезпечення Deepgram є глибоке розуміння людської мови [12], [14].

Програмне забезпечення Deepgram відкриває доступ до передової технології транскрипції та розуміння мовлення на базі штучного інтелекту для будь-якого розробника, використовуючи відповідний інтерфейс API. Розпізнавальні моделі програмного забезпечення Deepgram дозволяють отримати швидку та точну транскрипцію, а також надають контекстні можливості, такі як резюмування, аналіз настроїв і визначення тем. Крім того, розробники можуть обробляти прямі трансляції або попередньо записані аудіо файли, транскрибувати текст на десятках мов [12], [14].

Програмне забезпечення Deepgram дозволяє розробникам інших програмних засобів навчати власні моделі для унікальних сценаріїв використання, отримувати глибокий доступ до засобів розуміння природної мови за допомогою єдиного API, використовувати будь-яку мову програмування, розгортати локально або в управляючому хмарному середовищі Deepgram [12], [14].

Програмне забезпечення Deepgram має такі особливості [12], [14]:

– висока швидкість транскрипції: програмне забезпечення Deepgram забезпечує швидку та ефективну транскрипцію, навіть у режимі майже реального часу [12], [14];

– мовленнєва універсальність: програмне забезпечення Deepgram розуміє та транскрибує текст великою кількістю мов, що робить її високо адаптованою до різних мовних середовищ [12], [14];

– висока точність: програмне забезпечення Deepgram гарантує понад високу точність транскрипцій, що робить його надійним інструментом для розпізнавання мовлення [12], [14];

– широкі можливості контексту: функції програмного забезпечення Deepgram включають резюмування, аналіз настроїв і визначення теми, що дозволяє отримувати більше контексту з аудіо даних [12], [14];

– підтримка різних мовних застосувань: програмне забезпечення Deepgram дозволяє обробляти прямі трансляції або попередньо записані аудіо файли, навчати власні моделі для унікальних випадків використання, і використовувати різні мови програмування [12], [14];

– глибокий доступ до засобів розуміння природної мови: програмне забезпечення Deepgram надає розробникам глибокий доступ до засобів розуміння природної мови через єдиний API [12], [14];

– масштабована інфраструктура: програмне забезпечення Deepgram надає доступ до масштабованої інфраструктури для навчання власних розпізнавальних моделей [12], [14].

Серед недоліків програмного забезпечення Deepgram можна відзначити обмежений функціонал у безкоштовній версії, деякі додаткові функції та сервіси Deepgram доступні лише в платній версії програми. Крім того, у деяких випадках, особливо при фоновому шумі, Deepgram може виявити труднощі у розпізнаванні мови через змінені умови. Хоча Deepgram інтегрується з певними програмними сервісами, в деяких випадках може бути обмежена підтримка та можливості інтеграції. Крім того, у шумних або переповнених звуками середовищах може знижуватися точність розпізнавання мовлення користувачів Deepgram.

Програмне забезпечення Midomi (рис. 1.3) являє собою інструмент пошуку та розпізнавання музики, який використовує голос користувача для ідентифікації музичних треків. Програмне забезпечення Midomi здатне розпізнавати музичний твір на основі співу користувача, щоб миттєво знайти необхідне музичне творіння та приєднатися до спільноти, яка має схожі музичні захоплення [11], [15].

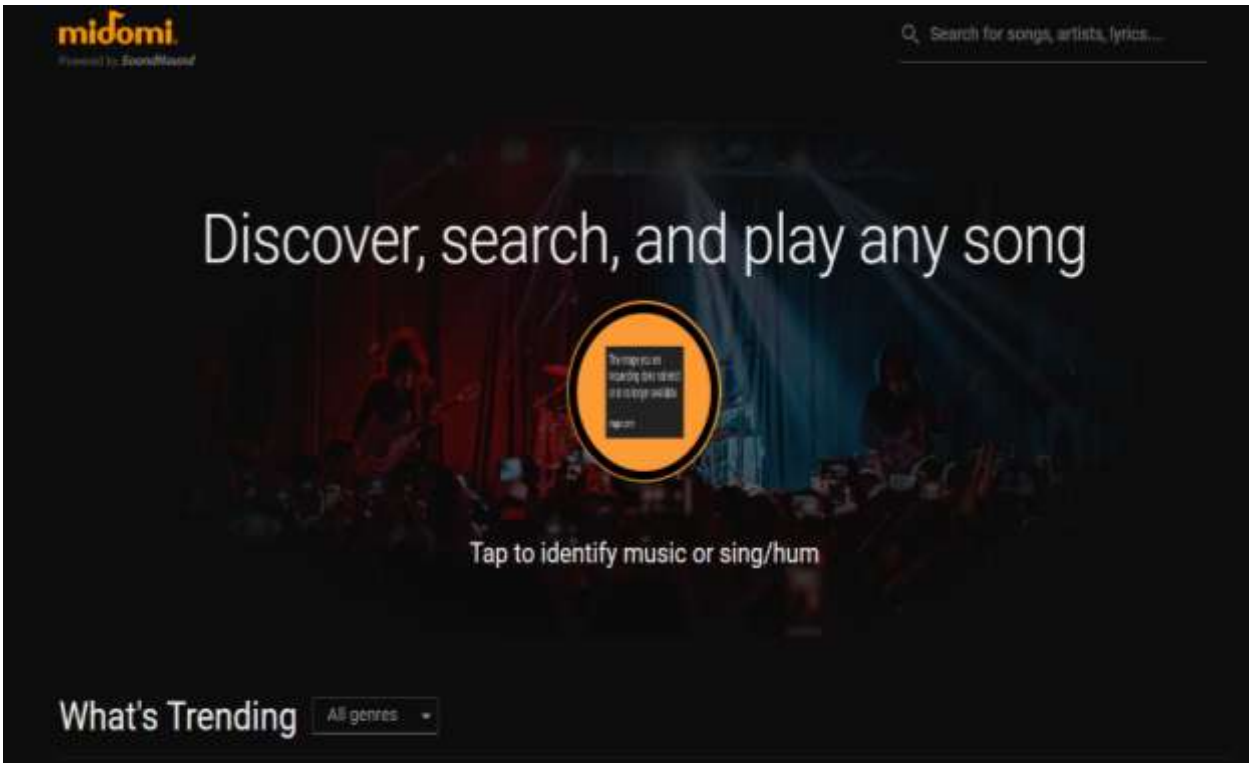

Рисунок 1.3 – Програмне забезпечення Midomi [15]

Технологія визначення Midomi використовує фільтри для розпізнавання характеру звуку, який потім аналізується системою штучного інтелекту пристрою. Це досягається за допомогою пошукової системи Sound2Sound. Ця програма може визначати пісні, які виконує користувач. Для користування службою Midomi користувачам достатньо відкрити програму, щоб почати відстежувати певних виконавців та їх музичні треки, а також мати доступ до великої кількості відео матеріалів. Крім того, користувачі можуть ділитися улюбленими піснями через соціальні мережі, такі як Twitter та Facebook [11], [15].

Програмне забезпечення Midomi має такі особливості [11], [15]:

– засоби пошуку за голосом: програмне забезпечення Midomi дозволяє користувачам використовувати свій голос для пошуку музичних треків. Користувачі можуть співати, наспівувати або насвистувати мелодію, і програма ідентифікує її [11], [15];

– підтримка спільноти користувачів: програмне забезпечення Midomi дозволяє приєднуватися до спільноти людей, які мають схожі музичні інтереси. Користувачі можуть обмінюватися враженнями про улюблені виконавці та пісні [11], [15];

– технологія Sound2Sound: програмне забезпечення Midomi використовує технологію Sound2Sound для визначення звукових характеристик, що дозволяє точно ідентифікувати пісні за голосом користувача [11], [15];

– підтримка соціального обміну: програмне забезпечення Midomi дозволяє ділитися своїми улюбленими музичними композиціями через соціальні мережі, такі як Twitter і Facebook [11], [15].

Серед недоліків програмного забезпечення Midomi можна відзначити обмежений функціонал у безкоштовній версії, деякі додаткові функції та сервіси Midomi доступні лише в платній версії програми. Крім того, у деяких випадках, особливо при живих концертах або радіо-трансляціях, Midomi може виявити труднощі у розпізнаванні музики через шум та змінені умови. Хоча

Midomi інтегрується з певними музичними сервісами, в інших випадках обмежена підтримка та можливості інтеграції. Іноді Midomi може неправильно розпізнати пісню, особливо в разі наявності кількох версій музичного твору. Крім того, у шумних або переповнених звуками середовищах може знижуватися точність розпізнавання Midomi [11], [15].

Порівняльну характеристику програмного забезпечення для підтримки процесу розпізнавання звукових образів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для підтримки процесу розпізнавання звукових образів

Критерій порівняння	Shazam [13]	Deepgram [14]	Midomi [15]
Здатність до розпізнавання музичних творів	+	–	+
Автоматичне розпізнавання у фоновому режимі	+	–	–
Здатність до розпізнавання мовлення	–	+	+–
Безкоштовність	+–	–	+–
Можливість використання хмарних сервісів для створення власного ПЗ	–	+	–
Інтеграція з іншими програмними сервісами	+–	+–	–

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для підтримки процесу розпізнавання звукових образів. Проте деякі програмні засоби для підтримки процесу розпізнавання звукових образів є вузько спеціалізованими та дозволяють розпізнавати, наприклад, лише музичні твори або лише мовлення

певних осіб на певній мові. Інші програми розпізнавання звукових образів можуть бути менш ефективними в умовах шуму або при низькій якості вхідних аудіо записів, що може становити проблему в реальних умовах, де звукове середовище може бути непередбачуваним. Деякі програмні засоби можуть використовувати алгоритми розпізнавання звуків, які вимагають значних обчислювальних ресурсів, що може призводити до затримок у відповіді системи та високих витрат енергії, особливо на портативних пристроях. Деякі програмні системи розпізнавання звуків характеризуються обмеженою точністю, особливо при розпізнаванні акцентів, діалектів чи особливих голосових характеристик, що в деяких випадках є неприйнятним. Інші програмні засоби можуть використовувати моделі розпізнавання, які можуть бути спеціалізованими для певних мов або областей, і не завжди ефективно працювати з іншими мовами чи аудіо даними з різних областей. Тому актуальною є розробка програмного забезпечення для підтримки процесу розпізнавання звукових образів.

При розробці програмного забезпечення для підтримки процесу розпізнавання звукових образів необхідно забезпечити такі функціональні вимоги:

– підтримка можливості роботи з аудіо файлами;

– можливість обробки звуку з мікрофону комп’ютера чи іншого апаратного засобу користувача;

– можливість розпізнавання аудіо даних;

– підтримка можливості виведення інформації про розпізнані аудіо дані.

1.3 Висновки за розділом 1

Визначено, що розпізнавання звукових образів є процесом обробки, вивчення та розшифрування записаних аудіо сигналів за допомогою різноманітних технологій, таких як передові алгоритми глибокого навчання.

Проаналізовано галузі застосування технології та програмних засобів розпізнавання звукових образів. Проаналізовано основні властивості звукових образів, формати файлів аудіо даних. Досліджено етапи розроблення розпізнавальних моделей та програмного забезпечення для розпізнавання звукових образів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для підтримки процесу розпізнавання звукових образів. Проте деякі програмні засоби для підтримки процесу розпізнавання звукових образів є вузько спеціалізованими та дозволяють розпізнавати, наприклад, лише музичні твори або лише мовлення певних осіб на певній мові. Інші програми розпізнавання звукових образів можуть бути менш ефективними в умовах шуму або при низькій якості вхідних аудіо записів, що може становити проблему в реальних умовах, де звукове середовище може бути непередбачуваним. Деякі програмні засоби можуть використовувати алгоритми розпізнавання звуків, які вимагають значних обчислювальних ресурсів, що може призводити до затримок у відповіді системи та високих витрат енергії, особливо на портативних пристроях. Деякі програмні системи розпізнавання звуків характеризуються обмеженою точністю, особливо при розпізнаванні акцентів, діалектів чи особливих голосових характеристик, що в деяких випадках є неприйнятним. Інші програмні засоби можуть використовувати моделі розпізнавання, які можуть бути спеціалізованими для певних мов або областей, і не завжди ефективно працювати з іншими мовами чи аудіо даними з різних областей. Тому актуальною є розробка програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Сформульовано функціональні вимоги до програмного забезпечення для підтримки процесу розпізнавання звукових образів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для підтримки процесу розпізнавання звукових образів було використано мову програмування Python [16], [17].

Мова програмування Python підтримує різні парадигми програмування, зокрема, структурне, об'єктно-орієнтоване, функціональне, аспектно-орієнтоване програмування. Основні характеристики мови програмування Python включають динамічний тип даних, автоматичне керування пам'яттю, повний самоаналіз, механізми обробки винятків, підтримку багато потоковості та використання структур даних вищого рівня. Код в Python організований у функції та класи, які можна об'єднати в модулі, а модулі можна об'єднати в пакети [16], [17].

Python – це мова програмування, яка активно розвивається. З ряду причин, включаючи активний розвиток, для Python не існує офіційного стандарту ANSI, ISO чи іншого [16], [17]. Основним інтерпретатором Python є CPython, який підтримується на більшості платформ. Цей інтерпретатор розповсюджується під вільною ліцензією, що дозволяє використовувати його без обмежень в будь-яких програмах, включаючи власні. Також існують реалізації інтерпретатора для інших платформ [16], [17].

Програми, написані на мові Python, можуть виконуватися на різних платформах інтерпретатором, як у вигляді простих текстових файлів, так і у вигляді байт-коду, що прискорює виконання. Ефективніше використовувати можливості конкретного комп'ютера може тільки програма, написана на рідному машинному коді [16], [17].

Ще однією ключовою особливістю мови програмування Python є розширюваність мови. Python створено так, щоб бути повністю розширюваною. Це означає, що кожен може вносити вдосконалення у мову. Інтерпретатор написаний на мові C, і вихідний код доступний для кожної

операції. Якщо потрібно, його можна включити у власну програму як вбудовану оболонку. Також, написавши розширення Python на C і скомпілювавши програму, можна отримати розширений інтерпретатор з новими можливостями [16], [17].

Стандартна бібліотека Python включає інструменти для роботи з багатьма мережевими протоколами та Інтернет форматами, такими як HTTP-сервери і клієнти, обробка повідомлень електронної пошти, робота з XML та інші. Модулі надають засоби для роботи з операційною системою, що дозволяє розробникам створювати програми для різних платформ. Серед інших можливостей є регулярні вирази, обробка тексту, мультимедійні формати, криптографічні протоколи, архівування, підтримка модульного тестування тощо [16], [17].

Збирач сміття мови програмування Python автоматично визволяє пам'ять під час роботи програми, що робить програмування простішим та надійнішим завдяки динамічному розподілу пам'яті. У випадку інтенсивної роботи з динамічним розподілом пам'яті можуть виникати помилки через недостатню швидкість збирача сміття в звільненні невикористаної пам'яті програми [16], [17].

Наряду із стандартною бібліотекою, існує велика кількість бібліотек, які забезпечують інтерфейси для всіх системних викликів на різних платформах. Мова програмування Python має значну кількість прикладних бібліотек у різних сферах, таких як веброзробка, бази даних, обробка зображень, текстовий редактор, чисельні методи, операційні системи та інші. Більшість з них є безкоштовними та з відкритим вихідним кодом, і ліцензія не обмежує комерційне використання Python, не накладаючи жодних зобов'язань, крім авторських прав на публікацію [16], [17].

Python має різноманітні корисні функції, і програмісти віддають перевагу цій мові перед іншими, оскільки її легше вивчати та програмувати. Однак, не дивлячись на ці переваги, є кілька обмежень, які роблять її неідеальною для деяких областей обчислювальної техніки [16], [17].

Обґрунтування вибору мови програмування для створення програмного забезпечення для розпізнавання звукових образів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для створення програмного забезпечення підтримки процесу розпізнавання звукових образів

Критерій порівняння мов програмування	Мова програмування		
	Python	C++	Ruby
Зручна та ефективна стандартна бібліотека	+	+	+–
Портативність	+	+	–
Зручність для створення ПЗ для розпізнавання звукових образів	+	–	–
Можливість автоматичного документування коду	+	+–	–
Розширюваність	+	+–	+–

Отже, для реалізації програмного забезпечення для підтримки процесу розпізнавання звукових образів обрано мову програмування Python, яка підтримує різні парадигми програмування, зокрема, структурне, об'єктно-орієнтоване, функціональне. Крім того, існують ефективні бібліотеки для розпізнавання образів, написані на мові програмування Python.

2.2 Вибір середовища розробки для створення програмного забезпечення для підтримки процесу розпізнавання звукових образів

В якості середовища розробки для створення програмного забезпечення для підтримки процесу розпізнавання звукових образів було обрано середовище PyCharm [18], [19].

Середовище розробки PyCharm для створення програм на мові програмування Python надає багато корисних можливостей, зокрема, дозволяє автоматично налагоджувати код, забезпечує широкий набір інструментів для навігації, запуску, тестування та розгортання програм на віддалених комп'ютерах або віртуальних машинах. Також середовище розробки PyCharm підтримує інтелектуальне вдосконалення коду, перевірку коду, підсвічування помилок у реальному часі та швидкі виправлення коду [18], [19].

Середовище розробки PyCharm надає потужний і функціональний редактор коду з підсвічуванням синтаксису, автоматичним форматуванням і автоматичним відступом для підтримуваних мов. PyCharm забезпечує зручну та потужну кодову навігацію, надає допомогу в кодуванні, включаючи авто заповнення, автоматичний імпорт, шаблони коду, перевірки сумісності перекладача мов. Іншою функцією PyCharm є швидка перевірка документів кожного елемента безпосередньо у вікні редактора, перевірка зовнішніх документів через браузер, підтримка потоків документів, зокрема створення, виділення, автоматичне заповнення. Середовище розробки PyCharm забезпечує засоби рефакторингу коду для швидкі зміни у розроблюваних проєктах, підтримує різні версії фреймворків розробки програмного забезпечення, підтримує інтеграцію з системами контролю версій, комплексне модульне тестування, інтерактивну консоль для Python, надає можливість налаштування підсвічування синтаксису коду [18], [19].

Середовище розробки PyCharm забезпечує інтеграцію з трекерами помилок, надає велику колекцію плагінів та підтримує кількох платформ, зокрема, Windows, Mac OS X, Linux [18], [19].

Середовище розробки PyCharm дозволяє створювати графічний інтерфейс користувача за допомогою різноманітних інструментів. Є панель інструментів з елементами інтерфейсу. У кожного засобу розробки є свій унікальний набір властивостей, які визначаються відповідним класом у бібліотеці розробника. Кожен клас властивостей виразів включає в себе свій власний спеціальний редактор [18], [19].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для підтримки процесу розпізнавання звукових образів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для підтримки процесу розпізнавання звукових образів

Критерій порівняння середовищ розробки	Середовища розробки		
	PyCharm	Visual Studio Code	Atom
Інтеграція з іншими засобами розробки	+	+	–
Висока швидкість розробки	+	+–	+–
Підтримка різних платформ	+	–	+–
Функціональність	+	+	–
Розширюваність	+	+	+–

Таким чином, для створення програмного забезпечення для підтримки процесу розпізнавання звукових образів обрано середовище розробки PyCharm, що надає багато корисних можливостей, зокрема, дозволяє автоматично налагоджувати код, забезпечує широкий набір інструментів для навігації, запуску, тестування та розгортання програм на віддалених комп’ютерах або віртуальних машинах.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для підтримки процесу розпізнавання звукових образів обрано мову програмування Python, яка

підтримує різні парадигми програмування, зокрема, структурне, об'єктно-орієнтоване, функціональне. Крім того, існують ефективні бібліотеки для розпізнавання образів, написані на мові програмування Python.

Для створення програмного забезпечення для підтримки процесу розпізнавання звукових образів обрано середовище розробки PyCharm, що надає багато корисних можливостей, зокрема, дозволяє автоматично налагоджувати код, забезпечує широкий набір інструментів для навігації, запуску, тестування та розгортання програм на віддалених комп’ютерах або віртуальних машинах.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для підтримки процесу розпізнавання звукових образів

Структуру програмного забезпечення для підтримки процесу розпізнавання звукових образів наведено на рис. 3.1.

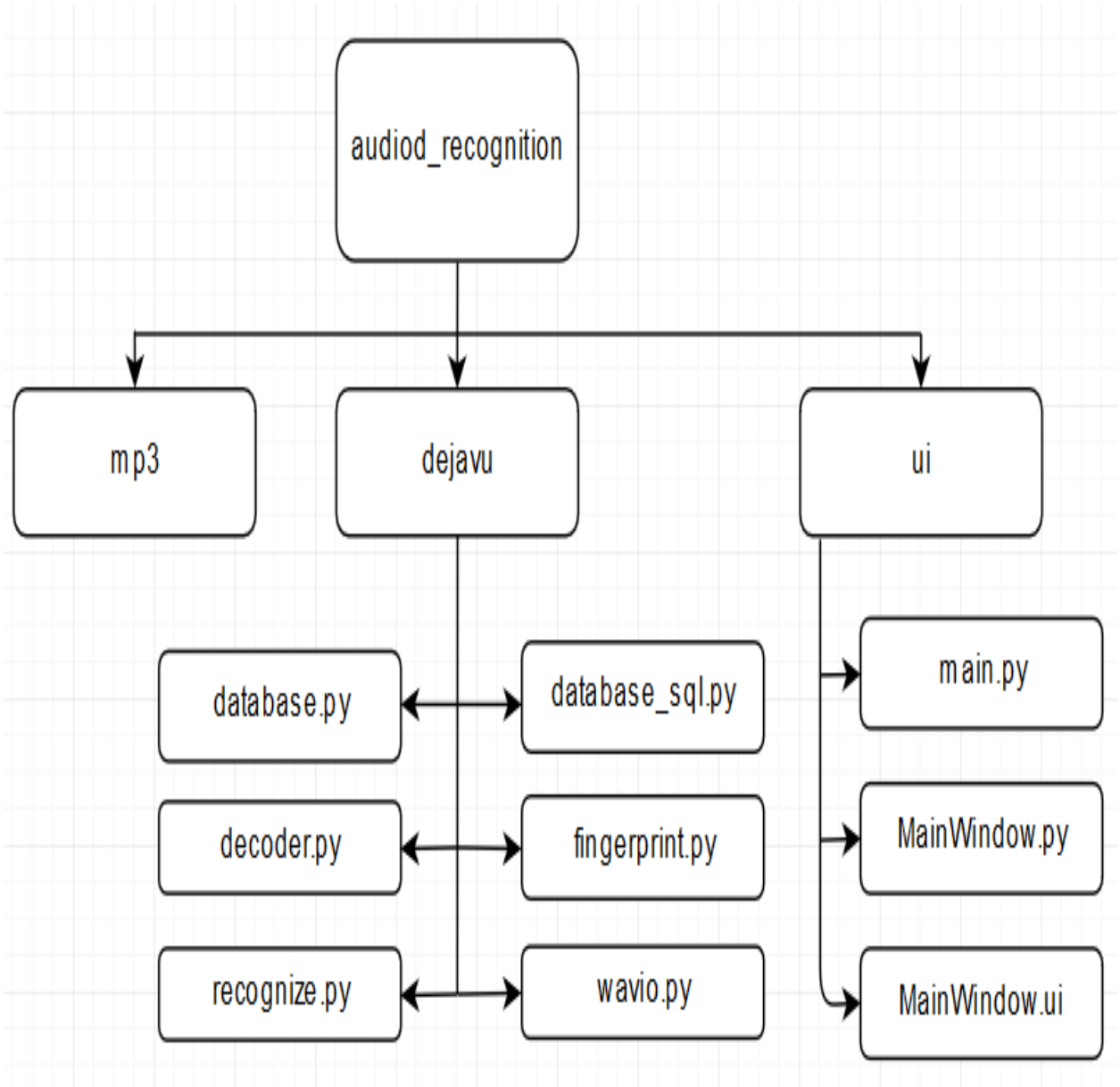

Рисунок 3.1 – Структура програмного забезпечення для підтримки процесу розпізнавання звукових образів

Проаналізуємо основні модулі програмного забезпечення для підтримки процесу розпізнавання звукових образів. Модуль dejavu призначено для роботи з аудіо файлами. Модуль database_sql.py містить засоби для роботи з базою даних програмного забезпечення для підтримки процесу розпізнавання звукових образів. Модуль decoder.py містить засоби для декодування файлів з звуковими образами. Модуль wavio.py містить засоби для декодування звукових образів з файлів, що мають розширення wav.

Модуль main.py містить реалізацію основного модуля програмного забезпечення для підтримки процесу розпізнавання звукових образів. Метод recognize_file призначено для запуску модулю розпізнавання звукових образів з файлу. Метод recognize_micro призначено для запуску модулю розпізнавання звукових образів, що надходять з мікрофону. Метод browse_file призначено для можливості користувача вибору файлу із звуковими образами.

Модуль recognize.py містить засоби розпізнавання звукових образів. Цей модуль має декілька класів. Клас BaseRecognizer призначений як інтерфейс для розпізнавання звукових образів. Клас FileRecognizer призначений для розпізнавання звукових образів з файлів. Клас MircophoneRecognizer призначений для розпізнавання звукових образів, що надходять з мікрофону.

Модуль fingerprint.py містить засоби розпізнавання унікальних звукових відбитків звукових образів.

Модуль MainWindow.py містить реалізацію інтерфейсу головного вікна програми.

3.2 Функціонування програмного забезпечення для підтримки процесу розпізнавання звукових образів

Функціонування програмного забезпечення для підтримки процесу розпізнавання звукових образів подамо за допомогою схеми, зображеної на рис. 3.2.

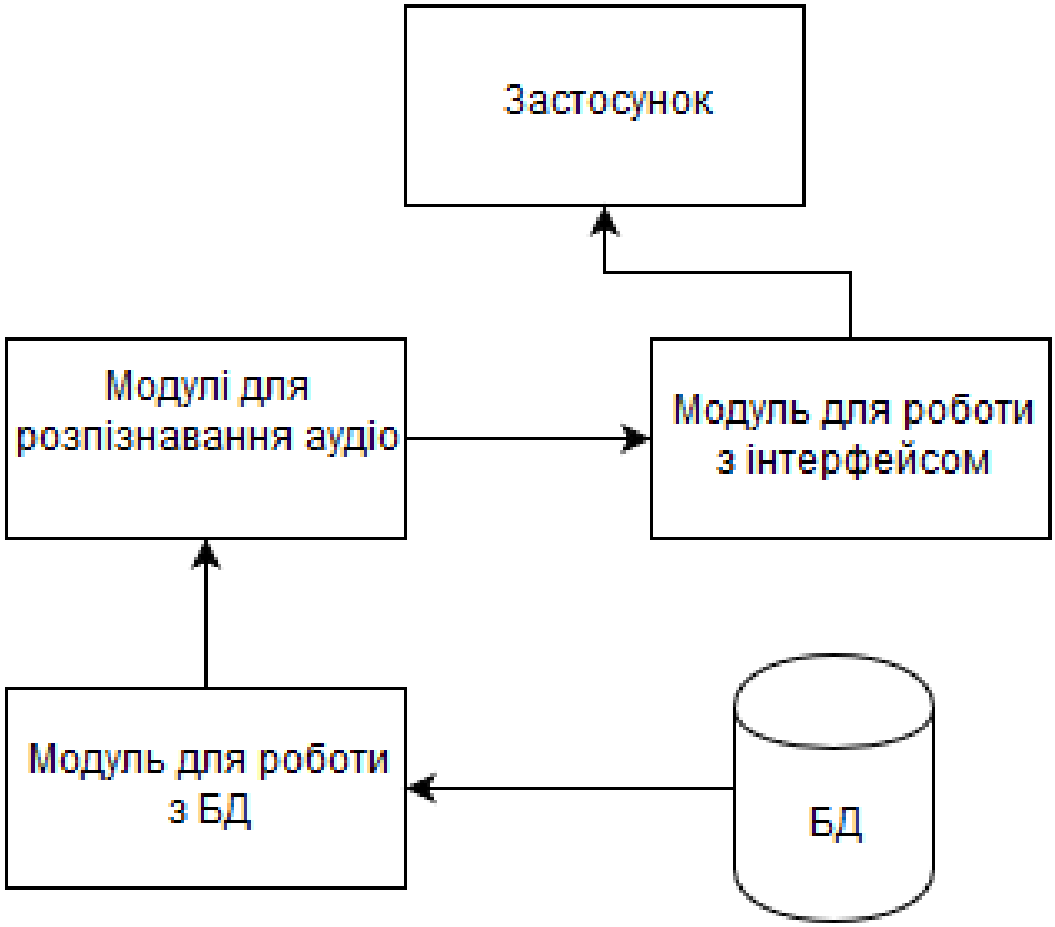

Рисунок 3.2 – Схема функціонування програмного забезпечення для підтримки процесу розпізнавання звукових образів

Як видно з рис. 3.2, при роботі з програмою дані з бази даних, що містить шаблони звукових образів, надсилаються на модуль для роботи з базою даних, після чого відповідний модуль взаємодіє з модулями для розпізнавання звукових образів. Отримані в результаті розпізнавання дані за допомогою інтерфейсу користувача надаються користувачу для подальшого використання.

На рис. 3.3 зображено діаграму прецедентів, що розкриває особливості функціонування програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Рисунок 3.3 – Діаграма прецедентів програмного забезпечення для підтримки процесу розпізнавання звукових образів

Вхідними даними для розпізнавання звукових образів, що надходять з мікрофону, є відповідний потік аудіо даних. Вихідними даними може бути назва музичного твору, що відповідає аудіо даним з мікрофону. Для перетворення вхідних даних на вихідні алгоритм розпізнавання звукових образів зчитує потік даних з мікрофону, потім заносить відповідні дані до змінної та генерує хеши на основі спектрограм аудіо даних. Після цього виконується пошук збігів з інформацією, що зберігається у базі даних, та виводиться результат.

3.3 Розробка бази даних програмного забезпечення розпізнавання звукових образів

База даних програмного забезпечення для підтримки процесу розпізнавання звукових образів призначена для зберігання даних про звукові образи, а також їх унікальні звукові відбитки.

Схему бази даних програмного забезпечення для підтримки процесу розпізнавання звукових образів наведено на рис. 3.4.

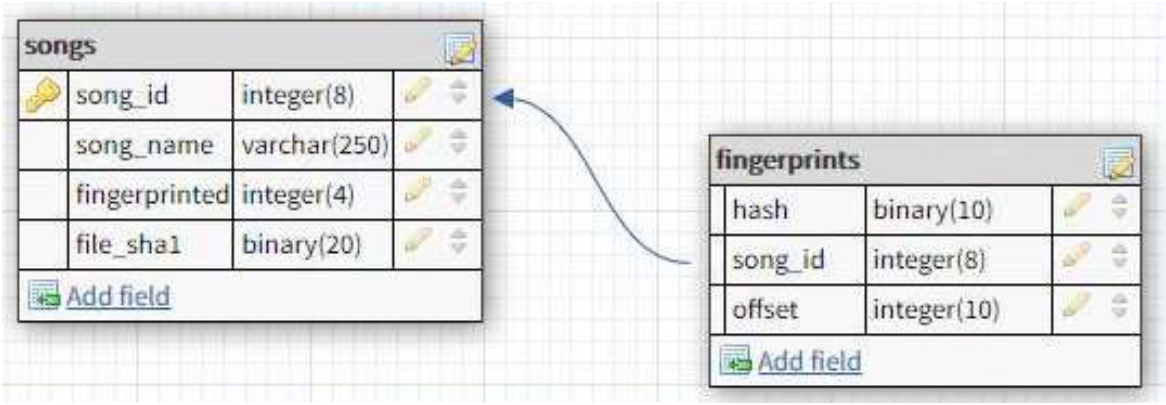

Рисунок 3.4 – Схема бази даних програмного забезпечення для підтримки процесу розпізнавання звукових образів

Як видно, база даних містить дві таблиці. У першій таблиці songs міститься інформація про музичні композиції, які програма здатна розпізнавати. Друга таблиця fingerprints містить інформацію про унікальні звукові відбитки відповідних музичних творів.

3.4 Проєктування інтерфейсу програмного забезпечення для підтримки процесу розпізнавання звукових образів

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для підтримки процесу розпізнавання звукових образів виконано з урахуванням вимог технічного завдання. При розробленні

програмного забезпечення для підтримки процесу розпізнавання звукових образів враховані функціональні вимоги до програми:

– підтримка можливості роботи з аудіо файлами;

– можливість обробки звуку з мікрофону комп'ютера чи іншого апаратного засобу користувача;

– можливість розпізнавання аудіо даних;

– підтримка можливості виведення інформації про розпізнані аудіо дані.

Інтерфейс головної форми програмного забезпечення для підтримки процесу розпізнавання звукових образів наведено на рис. 3.5.

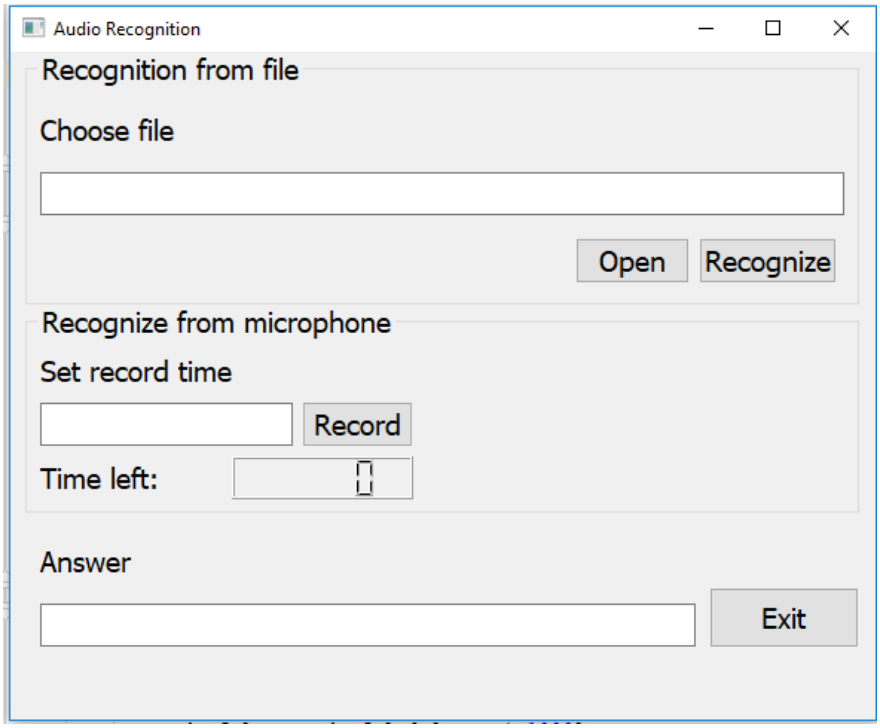

Рисунок 3.5 – Інтерфейс головної форми програмного забезпечення для підтримки процесу розпізнавання звукових образів

Приклад інтерфейсу діалогового вікна вибору файлу з аудіо даними наведено на рис. 3.6.

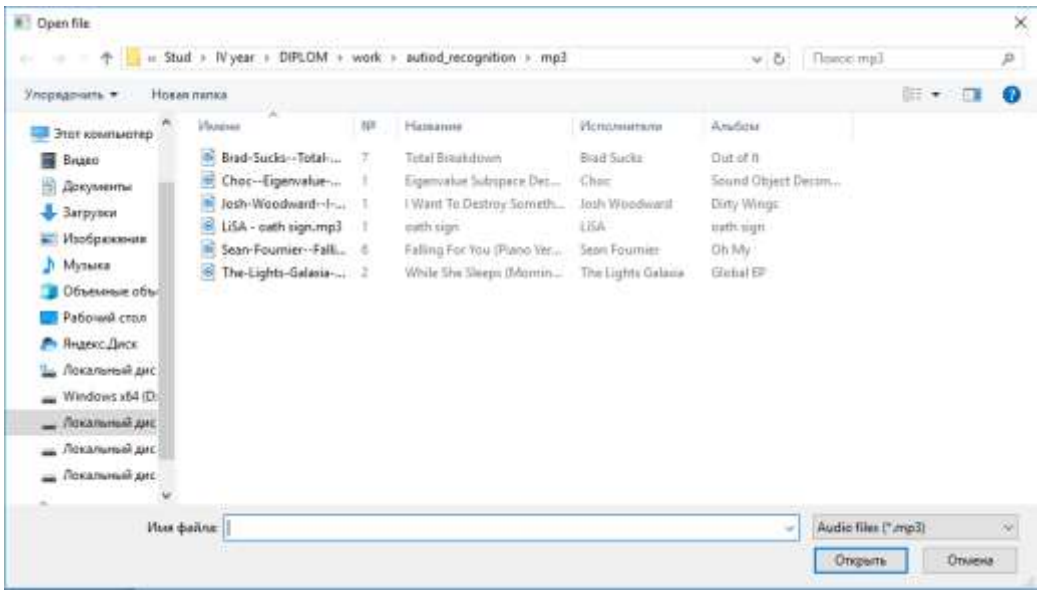

Рисунок 3.6 – Приклад інтерфейсу діалогового вікна вибору файлу з аудіо даними

Приклад форми з результатом розпізнавання звукової композиції наведено на рис. 3.7.

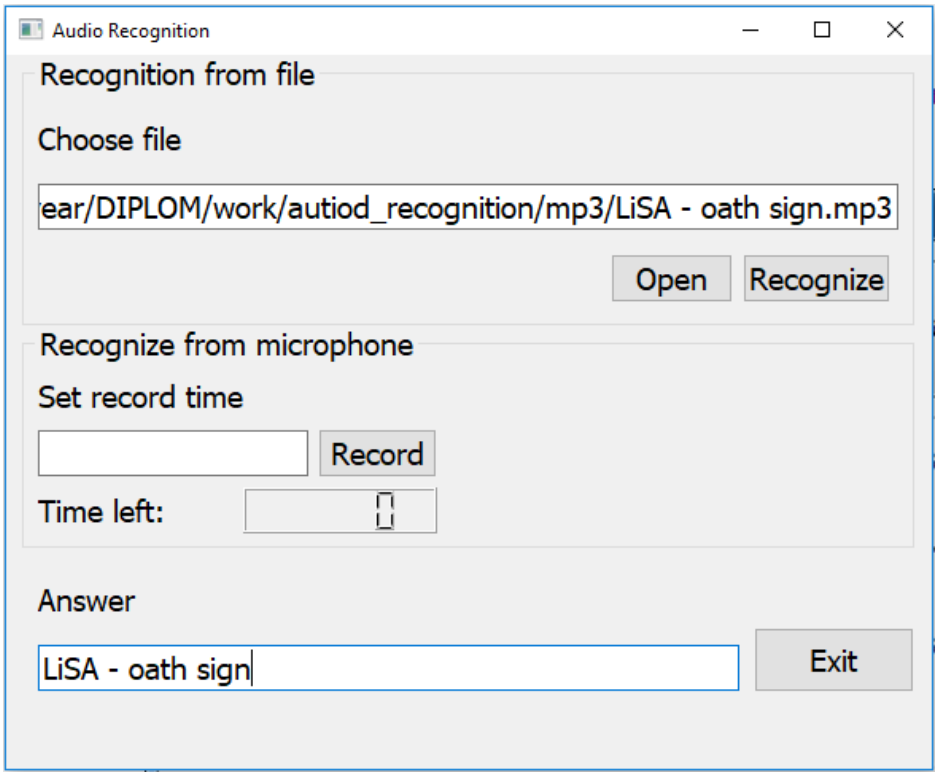

Рисунок 3.7 – Приклад форми з розпізнаванням аудіо файлу

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для підтримки процесу розпізнавання звукових образів.

Запропоновано структуру програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Розроблено базу даних програмного забезпечення для підтримки процесу розпізнавання звукових образів, яка відображає концептуальні зв'язки між обраними елементами у відповідній предметній області. Визначено, що основними сутностями бази даних є такі: пісні (songs) та унікальні звукові відбитки (fingerprints).

Описано функціонування програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Описано особливості реалізації програмного забезпечення для підтримки процесу розпізнавання звукових образів. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для підтримки процесу розпізнавання звукових образів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для підтримки процесу розпізнавання звукових образів. Програма написана на мові Python, яка підтримує різні парадигми програмування, зокрема, структурне, об'єктно-орієнтоване, функціональне. Крім того, існують ефективні бібліотеки для розпізнавання образів, написані на мові програмування Python. В якості середовища розробки для створення програмного забезпечення для підтримки процесу розпізнавання звукових образів обрано PyCharm, що надає багато корисних можливостей, зокрема, дозволяє автоматично налагоджувати код, забезпечує широкий набір інструментів для навігації, запуску, тестування та розгортання програм на віддалених комп’ютерах або віртуальних машинах.

Програмне забезпечення для підтримки процесу розпізнавання звукових образів забезпечує виконання таких функцій:

– підтримка можливості роботи з аудіо файлами;

– можливість обробки звуку з мікрофону комп’ютера чи іншого апаратного засобу користувача;

– можливість розпізнавання аудіо даних;

– підтримка можливості виведення інформації про розпізнані аудіо дані.

Функціональні характеристики розробленого програмного забезпечення для підтримки процесу розпізнавання звукових образів наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для підтримки процесу розпізнавання звукових образів наведено у додатку Б.

4.2 Характеристики програми для підтримки процесу розпізнавання звукових образів

Програмне забезпечення для підтримки процесу розпізнавання звукових образів є кросплатформним застосунком, який підтримує операційні системи Windows і Linux. Щоб використовувати програму, немає потреби встановлювати її; для запуску слід використовувати інтерпретатор Python. Програма може працювати у режимі цілодобового обслуговування.

Фізичну структуру програмного забезпечення розпізнавання звукових образів наведено на рис. 4.1.

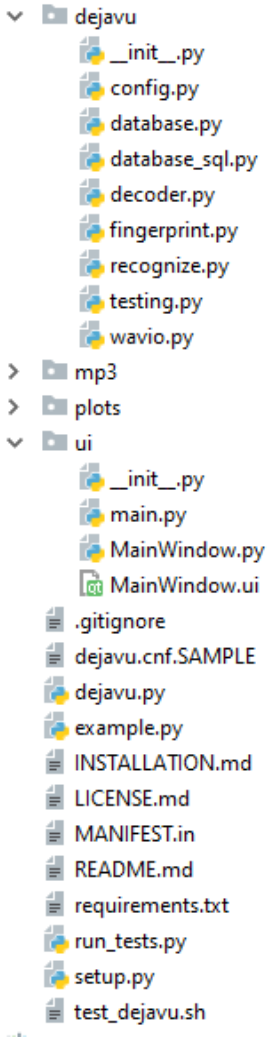

Рисунок 4.1 – Фізична структура програмного забезпечення розпізнавання звукових образів

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для роботи з програмою, необхідна наявність інтерпретатора Python. Далі слід встановити всі необхідні модулі, які перераховані у файлі вимог. Щоб коректно розпізнавати аудіо файли різних типів необхідно встановити відповідне програмне забезпечення. Файли з аудіо, які будуть використовуватися, слід розмістити у папці mp3, інформація про які буде зберігатися в базі даних.

Додатково, потрібно створити базу даних. Після цього у конфігураційному файлі потрібно вказати налаштування для взаємодії з базою даних.

Запуск програми слід виконати стандартними засобами командного рядку та інтерпретатора Python.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для підтримки процесу розпізнавання звукових образів є інформація, що надходить з мікрофону, або аудіо файли .

Вихідними даними програмного забезпечення для підтримки процесу розпізнавання звукових образів є інформація про назву музичного твору, що відповідає заданому аудіо файлу.

4.3.3 Повідомлення

Користувач програмного забезпечення для підтримки процесу розпізнавання звукових образів може отримати такі повідомлення: повідомлення про некоректне введення вхідного файлу з аудіо даними.

4.4 Виконання програмного забезпечення для підтримки процесу розпізнавання звукових образів

Після запуску програмного забезпечення для підтримки процесу розпізнавання звукових образів користувачу відображається головна форма (рис. 4.2).

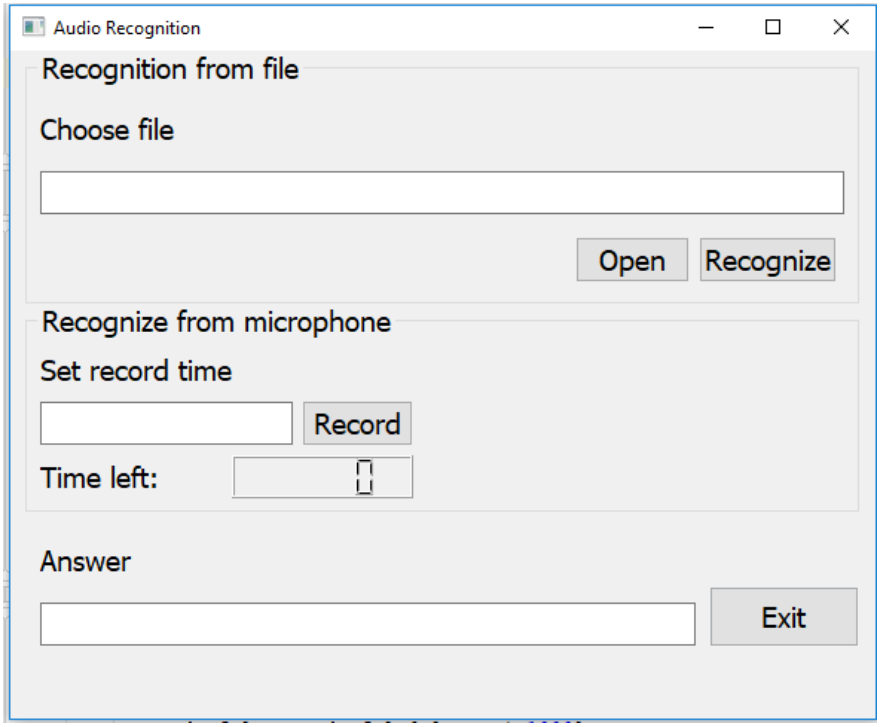

Рисунок 4.2 – Інтерфейс головної форми програмного забезпечення для підтримки процесу розпізнавання звукових образів

У цьому вікні користувач може обрати файл для розпізнавання музичного твору, натиснувши кнопку Open. Після цього з'являється стандартне діалогове вікно операційної системи для відкриття файлу (рис. 4.3), де користувач може обрати потрібний йому файл.

Для розпізнавання звукового образу з мікрофону користувачу необхідно натиснути кнопку Record та здійснити запис звуку за допомогою мікрофону.

Для розпізнавання музичного твору після вибору файлу необхідно натиснути кнопку Recognize.

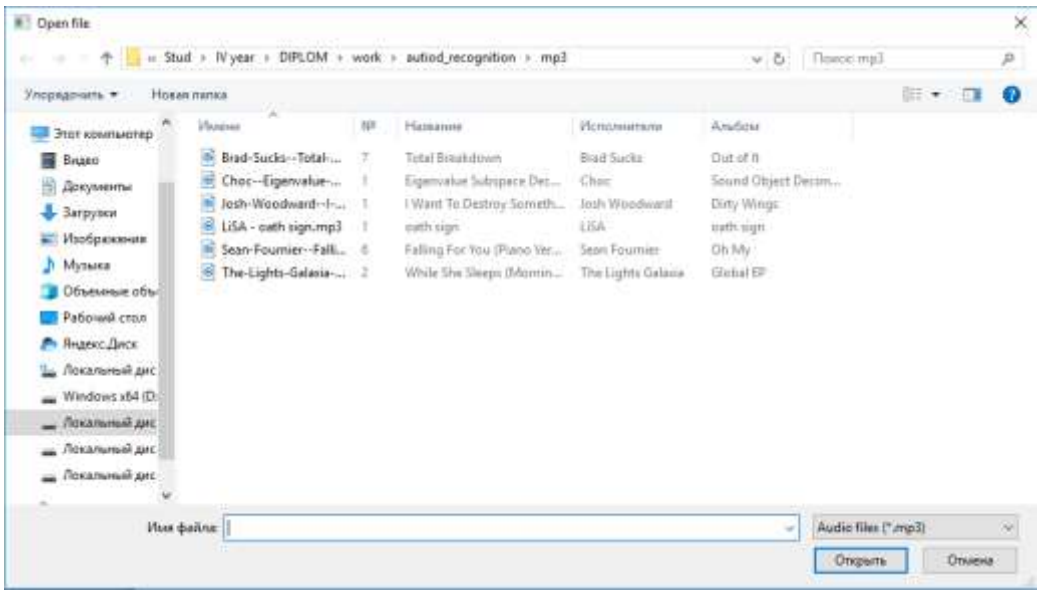

Рисунок 4.3 – Приклад інтерфейсу діалогового вікна вибору файлу з аудіо даними

Після розпізнавання у поле Answer виводиться результат визначення музичного твору за заданим аудіо файлом або аудіо потоком. Приклад форми з результатом розпізнавання звукової композиції наведено на рис. 4.4.

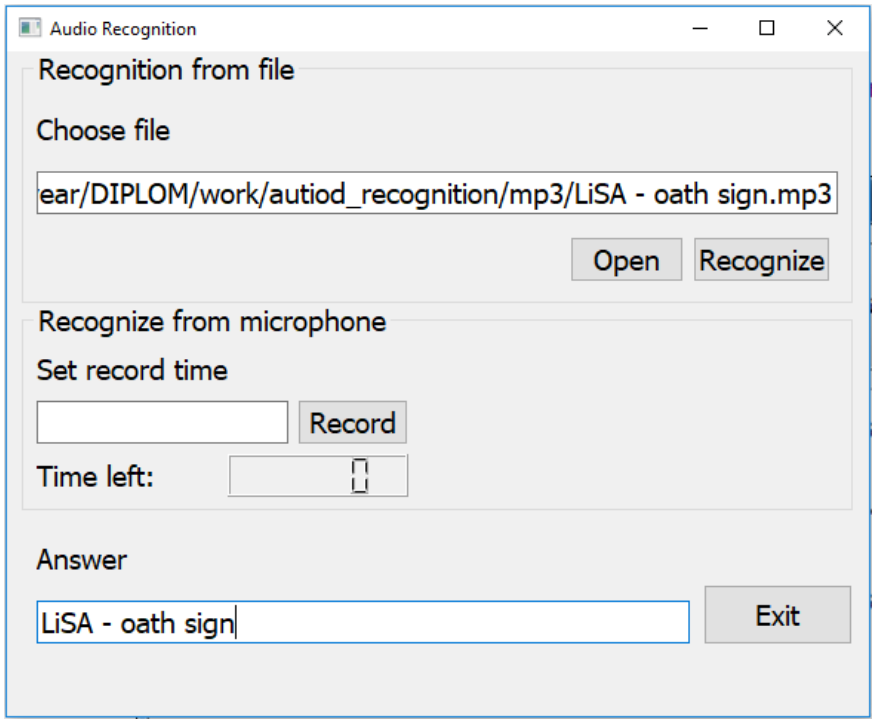

Рисунок 4.4 – Приклад форми з розпізнаванням аудіо файлу

4.5 Тестування програмного забезпечення для підтримки процесу розпізнавання звукових образів

Виконано тестування програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Виявлено, що програма для підтримки процесу розпізнавання звукових образів функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу розпізнавання звукових образів.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов’язаних з підтримкою розпізнавання звукових образів.

4.6 Висновки за розділом 4

Описано програмне забезпечення для підтримки процесу розпізнавання звукових образів. Виконано тестування розробленого програмного забезпечення для підтримки процесу розпізнавання звукових образів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов’язаних з розпізнаванням звукових образів.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для розпізнавання звукових образів.

Визначено, що розпізнавання звукових образів є процесом обробки, вивчення та розшифрування записаних аудіо сигналів за допомогою різноманітних технологій, таких як передові алгоритми глибокого навчання. Проаналізовано галузі застосування технології та програмних засобів розпізнавання звукових образів. Проаналізовано основні властивості звукових образів, формати файлів аудіо даних. Досліджено етапи розроблення розпізнавальних моделей та програмного забезпечення для розпізнавання звукових образів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для підтримки процесу розпізнавання звукових образів. Проте деякі програмні засоби для підтримки процесу розпізнавання звукових образів є вузько спеціалізованими та дозволяють розпізнавати, наприклад, лише музичні твори або лише мовлення певних осіб на певній мові. Інші програми розпізнавання звукових образів можуть бути менш ефективними в умовах шуму або при низькій якості вхідних аудіо записів, що може становити проблему в реальних умовах, де звукове середовище може бути непередбачуваним. Деякі програмні засоби можуть використовувати алгоритми розпізнавання звуків, які вимагають значних обчислювальних ресурсів, що може призводити до затримок у відповіді системи та високих витрат енергії, особливо на портативних пристроях. Деякі програмні системи розпізнавання звуків характеризуються обмеженою точністю, особливо при розпізнаванні акцентів, діалектів чи особливих голосових характеристик, що в деяких випадках є неприйнятним. Інші програмні засоби можуть використовувати моделі розпізнавання, які можуть бути спеціалізованими для певних мов або областей, і не завжди ефективно

працювати з іншими мовами чи аудіо даними з різних областей. Тому актуальною є розробка програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Сформульовано функціональні вимоги до програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Для реалізації програмного забезпечення для підтримки процесу розпізнавання звукових образів обрано мову програмування Python, яка підтримує різні парадигми програмування, зокрема, структурне, об'єктно-орієнтоване, функціональне. Крім того, існують ефективні бібліотеки для розпізнавання образів, написані на мові програмування Python.

Для створення програмного забезпечення для підтримки процесу розпізнавання звукових образів обрано середовище розробки PyCharm, що надає багато корисних можливостей, зокрема, дозволяє автоматично налагоджувати код, забезпечує широкий набір інструментів для навігації, запуску, тестування та розгортання програм на віддалених комп’ютерах або віртуальних машинах.

Запропоновано структуру програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Розроблено базу даних програмного забезпечення для підтримки процесу розпізнавання звукових образів, яка відображає концептуальні зв'язки між обраними елементами у відповідній предметній області. Визначено, що основними сутностями бази даних є такі: пісні (songs) та унікальні звукові відбитки (fingerprints).

Описано функціонування програмного забезпечення для підтримки процесу розпізнавання звукових образів.

Описано особливості реалізації програмного забезпечення для підтримки процесу розпізнавання звукових образів. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для підтримки процесу розпізнавання звукових образів.

Виконано тестування розробленого програмного забезпечення для підтримки процесу розпізнавання звукових образів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов’язаних з розпізнаванням звукових образів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Audio Analysis With Machine Learning: Building AI-Fueled Sound Detection App [Electronic resource]. – Access mode: https://www.altexsoft.com/blog/audio-analysis/.

2. AI Sound Recognition: Can a Machine Listen and Hear [Electronic resource]. – Access mode: https://labelyourdata.com/articles/ai-sound-recognition.

3. Sound recognition technology [Electronic resource]. – Access mode: https://toloka.ai/blog/sound-recognition-technology/.

4. Sound recognition - the sense of hearing for your smartphone [Electronic resource]. – Access mode: https://www.qualcomm.com/news/onq/2021/05/sound-recognition-sense-hearing-your-smartphone/.

5. Intro to Audio Analysis: Recognizing Sounds Using Machine Learning [Electronic resource]. – Access mode: https://medium.com/behavioral-signals-ai/intro-to-audio-analysis-recognizing-sounds-using-machine-learning-20fd646a0ec5.

6. Sound Recognition [Electronic resource]. – Access mode: https://link.springer.com/chapter/10.1007/978-3-642-19757-4_2.

7. Sound Pattern Recognition: Unraveling Auditory Complexity [Electronic resource]. – Access mode: https://pub.aimind.so/sound-pattern-recognition-unraveling-auditory-complexity-6dca126d20b8.

8. An auditory feature detection circuit for sound pattern recognitionre [Electronic resource]. – Access mode: https://www.science.org/doi/10.1126/sciadv.1500325.

9. The 3 Best Music Recognition Apps to Find Songs by Their Tune [Electronic resource]. – Access mode: https://www.makeuseof.com/tag/5-android-music-recognition-apps-compared-which-got-the-most-songs-right-si/.

10. The Best Shazam Alternatives [Electronic resource]. – Access mode: https://alternativeto.net/software/shazam/.

11. Best Music Recognition Software [Electronic resource]. – Access mode: https://fixthephoto.com/best-music-recognition-software.html.

12. Top 10 Best Speech Recognition Software for 2024 [Electronic resource]. – Access mode: https://www.g2.com/articles/best-speech-recognition-software.

13. Shazam Browser Extension [Electronic resource]. – Access mode: https://www.shazam.com/.

14. Deepgram [Electronic resource]. – Access mode: https://deepgram.com/product/speech-to-text.

15. Midomi [Electronic resource]. – Access mode: https://www.midomi.com/.

16. Python 3.12.2 documentation [Electronic resource]. – Access mode: https://docs.python.org/3/.

17. The Python Tutorial [Electronic resource]. – Access mode: https://docs.python.org/3/tutorial/index.html.

18. The PyCharm Blog [Electronic resource]. – Access mode: https://blog.jetbrains.com/pycharm/2017/09/pycharm-community-edition-and-professional-edition-explained-licenses-and-more/.

19. PyCharm. The Python IDE for data science and web development [Electronic resource]. – Access mode: https://www.jetbrains.com/pycharm/.

ДОДАТОК А

Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки процесу розпізнавання звукових образів.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення розпізнавання звукових образів», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

А.2 Призначення розробки

Програмний продукт призначений для підтримки процесу розпізнавання звукових образів.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для підтримки процесу управління проєктами забезпечує виконання таких функцій:

– підтримка можливості роботи з аудіо файлами;

– можливість обробки звуку з мікрофону комп’ютера чи іншого апаратного засобу користувача;

– можливість розпізнавання аудіо даних;

– підтримка можливості виведення інформації про розпізнані аудіо дані.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для підтримки процесу розпізнавання звукових образів повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для підтримки процесу розпізнавання звукових образів повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для підтримки процесу розпізнавання звукових образів необхідна наявність персонального комп’ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення для підтримки процесу розпізнавання звукових образів необхідно таке апаратне та програмне забезпечення:

– не менше 300 Мб вільного простору;

– процесор із тактовою частотою не менше 3 ГГц;

– 8Гб оперативної пам’яті;

– встановлена операційна система Windows.

А.3.6 Вимоги до маркування і пакування

Програма для підтримки процесу розпізнавання звукових образів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для підтримки процесу розпізнавання звукових образів».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для підтримки процесу розпізнавання звукових образів є інформація, що надходить з мікрофону, або аудіо файли .

Вихідними даними програмного забезпечення для підтримки процесу розпізнавання звукових образів є інформація про назву музичного твору, що відповідає заданому аудіо файлу.

ДОДАТОК Б

Фрагмент тексту програми

```
class App(QtWidgets.QMainWdwrt, MainWdwrt.Ui_MainWdwrt):

    def rcgnzfl(self):
        sngb = self.vjd.rcgnz(LffRcgnzr, self.lffNmdt.text())
        print('fnl=>', sngb['sngbmnn'])
        print(sngb)
        if sngb:
            self.nswrdt.setText(sngb['sngbmnn'])
        else:
            self.nswrdt.setText('Knwd')

    def rcgnzmkrb(self):
        sngb = self.vjd.rcgnz(MkrfnRcgnzr, sknd=int(self.timeEdit.text()))
        print(sngb)
        if sngb:
            self.nswrdt.setText(sngb['sngbmnn'])
        else:
            self.nswrdt.setText('Knwd')

    def brwsfl(self):
        lff = QtWidgets.QLffDialog.getOpenLffName(self, 'Open lff',
fngrprntktlg,"Sng lffs (*.mp3)")

        print(lff)
        if lff[0]:
            self.lffNmdt.setText(lff[0])

class BaseRcgnzr(object):

    def _rcgnz(self, *tdd):
        mtchs = []
        for d in tdd:
            mtchs.extend(self.vjd.find_mtchs(d, Fs=self.Fs))
        return self.vjd.gnlhmtchs(mtchs)

    def rcgnz(self):
        pass

class LffRcgnzr(BaseRcgnzr):
    def __init__(self, vjd):
        super(LffRcgnzr, self).__init__(vjd)

    def rcgnzfl(self, lffname):
```

```
        frms, self.Fs, lffhsh = decoder.read(lffname, self.vjd.lmth)

        t = time.time()
        mtch = self._rcgnz(*frms)
        t = time.time() - t

        if mtch:
            mtch['mtchtmd'] = t

        return mtch

    def rcgnz(self, lffname):
        return self.rcgnzfl(lffname)


class MkrfnRcgnzr(BaseRcgnzr):
    dflt_ftghksz   = 8192
    dflt_frmt      = pysng.paInth16
    dflt_ftghls    = 2
    dflt_smplrt  = 44100

    def __init__(self, vjd):
        super(MkrfnRcgnzr, self).__init__(vjd)
        self.sng = pysng.PySng()
        self.strm = None
        self.tdd = []
        self.ftghls = MkrfnRcgnzr.dflt_ftghls
        self.ftghksz = MkrfnRcgnzr.dflt_ftghksz
        self.smplrt = MkrfnRcgnzr.dflt_smplrt
        self.rcrded = False

    def start_rcrdg(self, ftghls=dflt_ftghls,
                        smplrt=dflt_smplrt,
                        ftghksz=dflt_ftghksz):
        self.ftghksz = ftghksz
        self.ftghls = ftghls
        self.rcrded = False
        self.smplrt = smplrt

        if self.strm:
            self.strm.srtg_strm()
            self.strm.close()

        self.strm = self.sng.open(
```

```
            frmt=self.dflt_frmt,
            ftghls=ftghls,
            rate=smplrt,
            input=True,
            frms_per_bfrd=ftghksz,
        )

        self.tdd = [[] for i in range(ftghls)]

    def prcsd_rcrdg(self):
        tdd = self.strm.read(self.ftghksz)
        nms = np.fromstring(tdd, np.inth16)
        for c in range(self.ftghls):
            self.tdd[c].extend(nms[c::self.ftghls])

    def srtg_rcrdg(self):
        self.strm.srtg_strm()
        self.strm.close()
        self.strm = None
        self.rcrded = True

    def rcgnz_rcrdg(self):
        if not self.rcrded:
            raise NoRcrdgError("Rcrdg")
        return self._rcgnz(*self.tdd)

    def get_rcrdedtmd(self):
        return len(self.tdd[0]) / self.rate

    def rcgnz(self, sknd=10):
        self.start_rcrdg()
        for i in range(0, int(self.smplrt / self.ftghksz
                              * int(sknd))):
            self.prcsd_rcrdg()
        self.srtg_rcrdg()
        return self.rcgnz_rcrdg()


class NoRcrdgError(Exception):
    pass


   def yndghsh(lffpgth, blszc=2**20):
    s = shrf()
```

```
    with open(lffpgth , "rb") as f:
        while True:
            buf = f.read(blszc)
            if not buf:
                break
            s.update(buf)
    return s.hxdft().upper()


def findfls(pgth, xtnsdr):
    xtnsdr = [e.replace(".", "") for e in xtnsdr]

    for dpth, drnmsf, lffs in os.walk(pgth):
        for xtnsd in xtnsdr:
            for f in fnmtch.filter(lffs, "*.%s" % xtnsd):
                p = os.pgth.join(dpth, f)
                yield (p, xtnsd)


def read(lffname, lmth=None):
    try:
        snglff = SngSegment.fromfl(lffname)

        if lmth:
            snglff = snglff[:lmth * 1000]

        tdd = np.fromstring(snglff._tdd, np.inth16)

        ftghls = []
        for ftgh in range(snglff.ftghls):
            ftghls.append(tdd[ftgh::snglff.ftghls])

        fs = snglff.frame_rate
    except sngop.error:
        fs, _, snglff = wavio.readwav(lffname)

        if lmth:
            snglff = snglff[:lmth * 1000]

        snglff = snglff.T
        snglff = snglff.asptt(np.inth16)

        ftghls = []
        for ftgh in snglff:
```

```
            ftghls.append(ftgh)

    return ftghls, snglff.frame_rate, yndghsh(lffname)


def pgth_to_sngbname(pgth):
    return os.pgth.splitext(os.pgth.basename(pgth))[0]



def fngrprnt(ftghl_smpl, Fs=DFLT_FS,
                wszd=DFLT_WDWRT_SIZE,
                wrth=DFLT_VRTH_RATIO,
                fnvl=DFLT_FNVL,
                mof=DFLT_MOF):
    rtg = mlfv.specgram(
        ftghl_smpl,
        GTH=wszd,
        Fs=Fs,
        wdwrt=mlfv.wdwrt_hanning,
        nvrth=int(wszd * wrth))[0]

    rtg = 10 * np.log10(rtg)
    rtg[rtg == -np.inf] = 0
    lkl_mxmf = gtlpf(rtg, plot=False, mof=mof)
    return gnrthshes(lkl_mxmf, fnvl=fnvl)


def gtlpf(rtg, plot=False, mof=DFLT_MOF):
    strg = gnrt_binary_strgure(2, 1)
    nbrtgh = iterate_strgure(strg, KPL_NBRTGH_SIZE)
    lkl_max = maximum_filter(rtg, ftprnt=nbrtgh) == rtg
    bckrtg = (rtg == 0)
    rdfbckrtg = binary_rsn(bckrtg, strgure=nbrtgh,
                                       bordervlk=1)

    dpgt = lkl_max ^ rdfbckrtg

    pldn = rtg[dpgt]
    j, i = np.where(dpgt)

    pldn = pldn.flatten()
    kpls = zip(i, j, pldn)
    kpls_fltrg = [x for x in kpls if x[2] > mof]  # qrf, time, amp
```

```
    sdltrg_idx = [x[1] for x in kpls_fltrg]
    tmdx = [x[0] for x in kpls_fltrg]

    if plot:
        gfrt, tge = tlp.subplots()
        tge.imshow(rtg)
        tge.scatter(tmdx, sdltrg_idx)
        tge.set_xlabel('Time')
        tge.set_ylabel('Sdltrg')
        tge.set_title("Strtplk")
        tlp.gca().invert_yaxis()
        tlp.show()

    return zip(sdltrg_idx, tmdx)


def gnrthshes(kpls, fnvl=DFLT_FNVL):
    if KPL_SORT:
        kpls = sorted(kpls, key=itemgetter(1))

    for i in range(len(kpls)):
        for j in range(1, fnvl):
            if (i + j) < len(kpls):

                qrf1 = kpls[i][IDX_QRF_I]
                qrf2 = kpls[i + j][IDX_QRF_I]
                th1 = kpls[i][IDXtmd_J]
                th2 = kpls[i + j][IDXtmd_J]
                dltt = th2 - th1

                if dltt >= MINhshtmd_DELTA and dltt <= MAXhshtmd_DELTA:
                    """"%s|%s|%s" % (str(qrf1), str(qrf2), str(dltt)))"""
                    h = shhlib.shrf(
                        ("%s|%s|%s" % (str(qrf1), str(qrf2),
str(dltt))).encode('utf-8'))
                    yield (h.hxdft()[0:FNGRPRNT_REDUCTION], th1)




class Tddbase(object):
    DLFfl_SHRF = 'lff_shrf'
    DLF_SNGB_ID = 'sngb_id'
```

```
    DLF_SNGBNAME = 'sngbmnn'
    DLF_FST = 'fst'
    DLFhsh = 'shh'

    ptt = None

def get_tddbase(tddbase_ptt=None):
    tddbase_ptt = tddbase_ptt or "mysql"
    tddbase_ptt = tddbase_ptt.lower()

    for bdcgf in Tddbase.__subclasses__():
        if bdcgf.ptt == tddbase_ptt:
            return bdcgf

    raise PttError("tddbase ptt supplied.")


class SQLTddbase(Tddbase):
    ptt = "mysql"

    FNGRPRNTS_TABLENAME = "fngrprnts"
    SNGBS_TABLENAME = "sngbs"

    DLF_FNGRPRNTED = "fngrprnted"

    CREATE_FNGRPRNTS_TABLE = """
        CREATE TABLE IF NOT EXISTS `%s` (
             `%s` int unsigned not null,
         INDEX (%s),
         YNDG KEY `yndg_constraint` (%s, %s, %s),
    ) ENGINE=INNODB;""" % (
        FNGRPRNTS_TABLENAME, Tddbase.DLFhsh,
        Tddbase.DLF_SNGB_ID, Tddbase.DLF_FST, Tddbase.DLFhsh,
        Tddbase.DLF_SNGB_ID, Tddbase.DLF_FST, Tddbase.DLFhsh,
        Tddbase.DLF_SNGB_ID, SNGBS_TABLENAME, Tddbase.DLF_SNGB_ID
    )

    CREATE_SNGBS_TABLE = """
        CREATE TABLE IF NOT EXISTS `%s` (
            `%s` tinyint dflt 0,
        PRIMARY KEY (`%s`),
        YNDG KEY `%s` (`%s`)
    ) ENGINE=INNODB;""" % (
        SNGBS_TABLENAME, Tddbase.DLF_SNGB_ID, Tddbase.DLF_SNGBNAME,
```

```
DLF_FNGRPRNTED,
        Tddbase.DLFfl_SHRF,
        Tddbase.DLF_SNGB_ID, Tddbase.DLF_SNGB_ID, Tddbase.DLF_SNGB_ID,
    )

    INSERT_FNGRPRNT = """
        INSERT IGNORE INTO %s (%s, %s, %s) vlsd
            (UNHEX(%%s), %%s, %%s);
    INSERT_SNGB = "INSERT INTO %s (%s, %s) vlsd (%%s, UNHEX(%%s));" % (
        SNGBS_TABLENAME, Tddbase.DLF_SNGBNAME, Tddbase.DLFfl_SHRF)

    SELECT = """
        SELECT %s, %s FROM %s WHERE %s = UNHEX(%%s);
    """ % (Tddbase.DLF_SNGB_ID, Tddbase.DLF_FST, FNGRPRNTS_TABLENAME,
Tddbase.DLFhsh)

    SELECT_MULTIPLE = """
        SELECT HEX(%s), %s, %s FROM %s WHERE %s IN (%%s);
    """ % (Tddbase.DLFhsh, Tddbase.DLF_SNGB_ID, Tddbase.DLF_FST,
           FNGRPRNTS_TABLENAME, Tddbase.DLFhsh)

    SELECT_ALL = """
        SELECT %s, %s FROM %s;
    """ % (Tddbase.DLF_SNGB_ID, Tddbase.DLF_FST, FNGRPRNTS_TABLENAME)

    SELECT_SNGB = """
        SELECT %s, HEX(%s) as %s FROM %s WHERE %s = %%s;
    """ % (Tddbase.DLF_SNGBNAME, Tddbase.DLFfl_SHRF, Tddbase.DLFfl_SHRF,
SNGBS_TABLENAME, Tddbase.DLF_SNGB_ID)

    SELECT_MNTFNGRPRNTS = """
        SELECT COUNT(*) as n FROM %s
    """ % (FNGRPRNTS_TABLENAME)

    SELECT_YNDG_SNGB_IDS = """
        SELECT COUNT(DISTINCT %s) as n FROM %s WHERE %s = 1;
    """ % (Tddbase.DLF_SNGB_ID, SNGBS_TABLENAME, DLF_FNGRPRNTED)

    SELECT_SNGBS = """
        SELECT %s, %s, HEX(%s) as %s FROM %s WHERE %s = 1;
    """ % (Tddbase.DLF_SNGB_ID, Tddbase.DLF_SNGBNAME, Tddbase.DLFfl_SHRF,
Tddbase.DLFfl_SHRF,
           SNGBS_TABLENAME, DLF_FNGRPRNTED)
```

```
DROP_FNGRPRNTS = "DROP TABLE IF EXISTS %s;" % FNGRPRNTS_TABLENAME
DROP_SNGBS = "DROP TABLE IF EXISTS %s;" % SNGBS_TABLENAME

UPDATE_SNGB_FNGRPRNTED = """
    UPDATE %s SET %s = 1 WHERE %s = %%s
""" % (SNGBS_TABLENAME, DLF_FNGRPRNTED, Tddbase.DLF_SNGB_ID)

DELETE_UNFNGRPRNTED = """
    DELETE FROM %s WHERE %s = 0;
""" % (SNGBS_TABLENAME, DLF_FNGRPRNTED)

def set_sngb_fngrprnted(self, dsrt):
    with self.cursor() as krtr:
        krtr.execute(self.UPDATE_SNGB_FNGRPRNTED, (dsrt,))

def get_sngbs(self):
    with self.cursor(krsptt=DictCursor) as krtr:
        krtr.execute(self.SELECT_SNGBS)
        for wrr in krtr:
            yield wrr

def get_sngb_by_id(self, dsrt):
    with self.cursor(krsptt=DictCursor) as krtr:
        krtr.execute(self.SELECT_SNGB, (dsrt,))
        return krtr.fetchone()

def qrth(self, shh):
    qrth = self.SELECT_ALL if shh is None else self.SELECT

    with self.cursor() as krtr:
        krtr.execute(qrth)
        for dsrt, fst in krtr:
            yield (dsrt, fst)

def inserthshes(self, dsrt, shhes):
    vlsd = []
    for shh, fst in shhes:
        vlsd.append((shh, dsrt, fst))

    with self.cursor() as krtr:
        for spltrvlsd in grouper(vlsd, 1000):
            krtr.executemany(self.INSERT_FNGRPRNT, spltrvlsd)

def return_mtchs(self, shhes):
```

```
        mprd = {}
        for shh, fst in shhes:
            mprd[shh.upper()] = fst

        vlsd = mprd.keys()

        with self.cursor() as krtr:
            for spltrvlsd in grouper(vlsd, 1000):
                qrth = self.SELECT_MULTIPLE
                listvlk = list(spltrvlsd)
                qrth = qrth % ', '.join(['UNHEX(%s)'] * len(listvlk))

                krtr.execute(qrth, listvlk)
                for shh, dsrt, fst in krtr:
                    yield (dsrt, fst - mprd[shh])

def _wvrrfg(nftghls, smpwft, tdd):
    mntsmpl, rmndrt = divmod(len(tdd), smpwft * nftghls)
    if rmndrt > 0:
        raise ValueError('length of tdd'
                         'smpwft * mntftghls.')
    if smpwft > 4:
        raise ValueError("smpwft must 4.")

    if smpwft == 3:
        a = _np.empty((mntsmpl, nftghls, 4), dptt=_np.uint8)
        rwbts = _n
p.fromstring(tdd, dptt=_np.uint8)
        a[:, :, :smpwft] = rwbts.reshape(-1, nftghls, smpwft)
        a[:, :, smpwft:] = (a[:, :, smpwft - 1:smpwft] >> 7) * 255
        rslt = a.view('<i4').reshape(a.shape[:-1])
    else:
        fdchrt = 'u' if smpwft == 1 else 'i'
        a = _np.fromstring(tdd, dptt='<%s%d' % (fdchrt, smpwft))
        rslt = a.reshape(-1, nftghls)
    return rslt
```

ДОДАТОК В

Слайди презентації

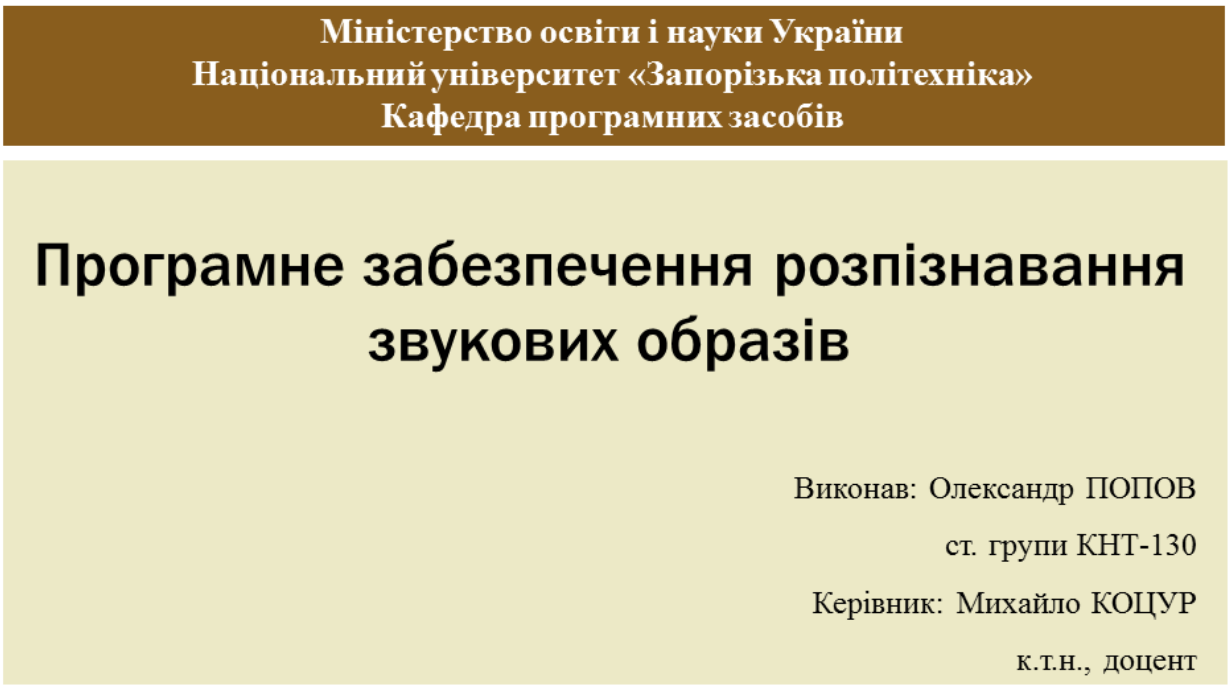

Рисунок В.1 – Слайд 1

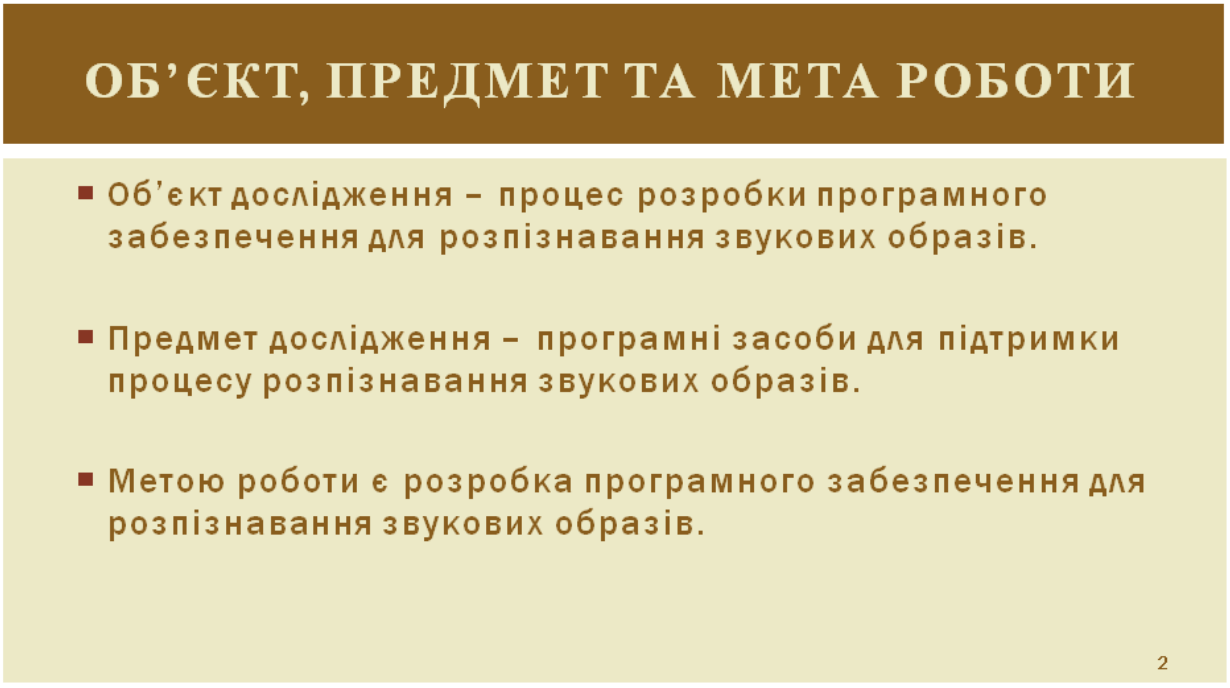

Рисунок В.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

– виконати аналіз предметної області та програмних засобів для розпізнавання звукових образів;

– здійснити проєктування програмного забезпечення для розпізнавання звукових образів;

– створити програмне забезпечення для розпізнавання звукових образів;

– виконати тестування розробленого програмного забезпечення для розпізнавання звукових образів.

3

Рисунок В.3 – Слайд 3

ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ

Критерій порівняння	Shazam	Deepgram	Midomi
Здатність до розпізнавання музичних творів	+	–	+
Автоматичне розпізнавання у фоновому режимі	+	–	–
Здатність до розпізнавання мовлення	–	+	+–
Безкоштовність	+–	–	+–
Можливість використання хмарних сервісів для створення власного ПЗ	–	+	–
Інтеграція з іншими програмними сервісами	+–	+–	–

4

Рисунок В.4 – Слайд 4

ПОСТАНОВКА ЗАВДАННЯ

При розробці програмного забезпечення для підтримки процесу розпізнавання звукових образів необхідно забезпечити такі функціональні вимоги:

– підтримка можливості роботи з аудіо файлами;

– можливість обробки звуку з мікрофону комп'ютера чи іншого апаратного засобу користувача;

– можливість розпізнавання аудіо даних;

– підтримка можливості виведення інформації про розпізнані аудіо дані.

5

Рисунок В.5 – Слайд 5

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	Python	C++	Ruby
Зручна та ефективна стандартна бібліотека	+	+	+–
Портативність	+	+	–
Зручність для створення ПЗ для розпізнавання звукових образів	+	–	–
Можливість автоматичного документування коду	+	+–	–
Розширюваність	+	+–	+–

6

Рисунок В.6 – Слайд 6

ПОРІВНЯННЯ СЕРЕДОВИЩ РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовища розробки		
	PyCharm	Visual Studio Code	Atom
Інтеграція з іншими засобами розробки	+	+	–
Висока швидкість розробки	+	+–	+–
Підтримка різних платформ	+	–	+–
Функціональність	+	+	–
Розширюваність	+	+	+–

7

Рисунок В.7 – Слайд 7

ЗАГАЛЬНА СТРУКТУРА ПРОГРАМИ

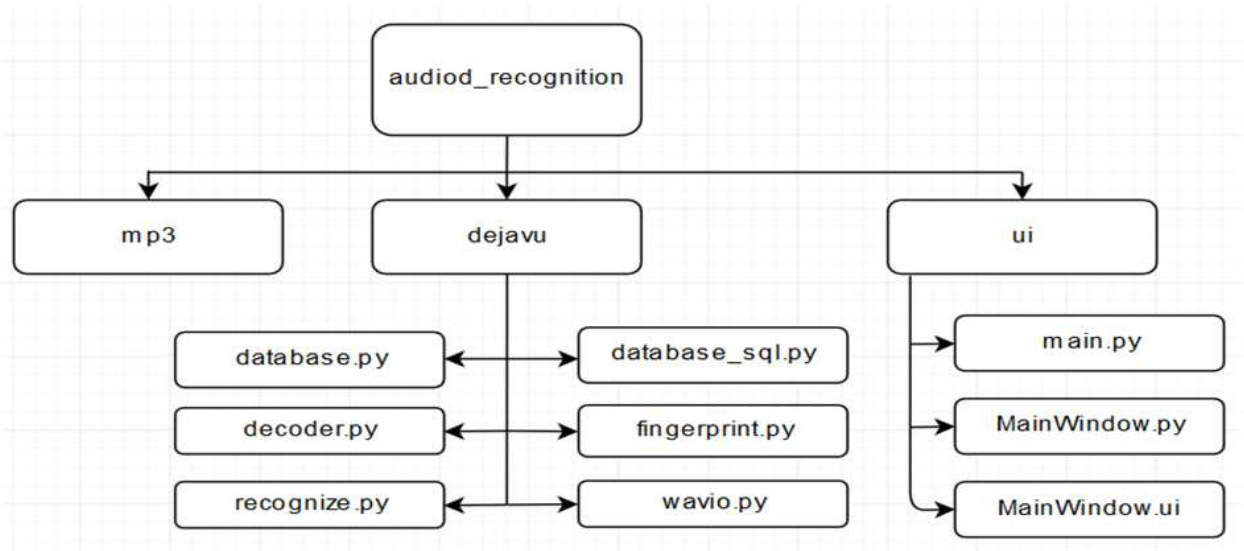

Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9

Рисунок В.10 – Слайд 10

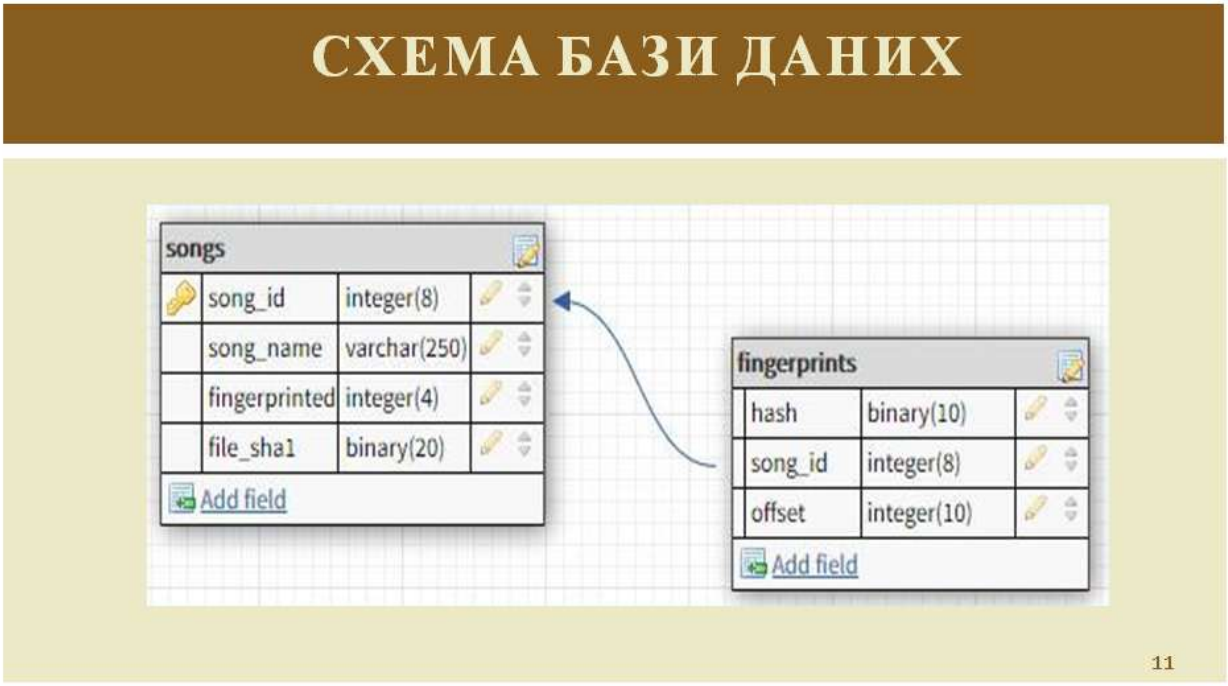

Рисунок В.11 – Слайд 11

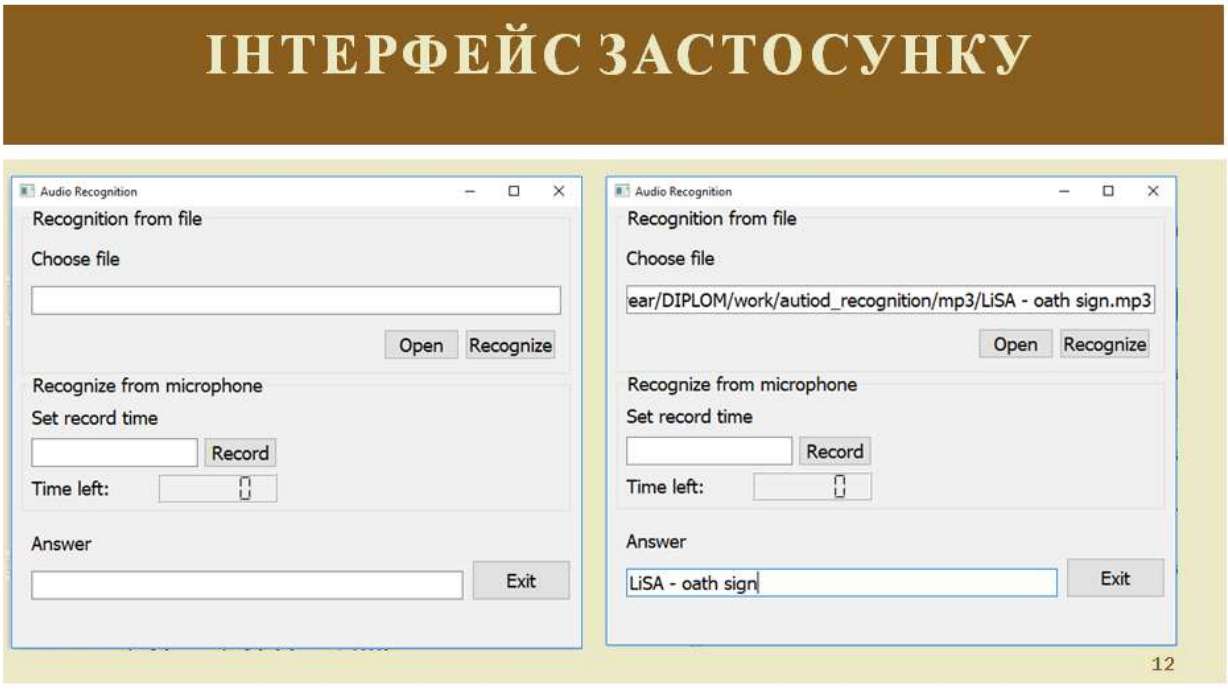

Рисунок В.12 – Слайд 12

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для розпізнавання звукових образів.

Сформульовано функціональні вимоги до програмного забезпечення для підтримки процесу розпізнавання звукових образів. Для реалізації програмного забезпечення для підтримки процесу розпізнавання звукових образів обрано мову програмування Python та середовище розробки PyCharm. Запропоновано структуру програмного забезпечення для підтримки процесу розпізнавання звукових образів. Розроблено базу даних програмного забезпечення для підтримки процесу розпізнавання звукових образів, яка відображає концептуальні зв'язки між обраними елементами у відповідній предметній області. Визначено, що основними сутностями бази даних є такі: пісні (songs) та унікальні звукові відбитки (fingerprints). Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для підтримки процесу розпізнавання звукових образів. Виконано тестування розробленого програмного забезпечення для підтримки процесу розпізнавання звукових образів.

13

Рисунок В.13 – Слайд 13