МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

«Програмний модуль автоматичного транскрибування мовлення»

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма **«Комп'ютерні науки»**
Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи КН- 42

_____Гумен О.Р._____ 
(прізвище та ініціали)

Керівник _____Мінаєва Ю.І.___
(прізвище та ініціали)
_____доц. , к.т.н._____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *інтелектуальних технологій*
Протокол № 13 від __05.06.2023 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ - 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

_____Гумен Ольги Русланівни_____

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

«Програмний модуль автоматичного транскрибування мовлення»_____

затверджена протоколом засідання кафедри від « 11 » листопада 2022 р. № 4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2023 року

3. Вихідні дані до проекту (роботи)

_____Програмний модуль, що надає автоматичне транскрибування аудіо та відео файлів_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1. Аналітичний огляд області програмного модулю розпізнавання мовлення_____
_____
2. Розробка архітектури програмного модулю розпізнавання мовлення_____
3. Тестування програмного модулю_____

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

_____

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2023 року

Керівник _____/ Мінаєва Ю.І. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____/ Гумен О.Р. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Аналіз постановки задачі, формалізація задачі, аналіз літературних джерел та написання 1 розділу роботи	16.02.2023 – 01.03.2023	
2	Проектно-технологічна реалізація програмного модулю	02.03.2023 – 9.04.2023	
3	Розробка та тестування модулю автоматичного транскрибування мовлення	10.04.2023 – 7.05.2023	
4	Оформлення пояснювальної записки та презентації	8.05.2023 – 29.05.2023	

Студент-дипломник _____/ Гумен. О.Р. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____/ Мінаєва Ю.І. /
(підпис) (ПІБ)

Анотація

Гумен Ольга Русланівна виконала випускну кваліфікаційну роботу на тему «Програмний модуль автоматичного транскрибування мовлення» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз сучасних методів розпізнавання мовлення, розглянуто використання програмного модулю розпізнавання мовлення, розроблено інформаційне та програмне забезпечення, що виконує транскрипцію аудіо і відео файлів у текстові.

Ключові слова: розпізнавання мовлення, транскрипція файлів.

Summary

The degree project: «Software module of automatic transcription of speech» was completed by Olha Humen specialty 122 – «Computer Sciences».

In the final qualification work, an analysis of modern speech recognition methods was carried out, the use of the speech recognition software module was considered, information and software were developed that transcribe audio and video files into text.

Keywords: speech recognition, file transcription.

ЗМІСТ

ВСТУП

Технологія розпізнавання мовлення увійшла в суспільну свідомість і звичне життя відносно недавно проте, її революційний вплив на спосіб взаємодії людей з комп'ютерами та машинами залишиться назавжди. Амбіціям даної технології значно сприяє той факт, що вона справді корисна для виконання повсякденних завдань.

Розпізнавання мовлення (Speech Recognition), також відоме як автоматичне розпізнавання мовлення (Automatic Speech Recognition), комп'ютерне розпізнавання мовлення або перетворення мовлення в текст, – це здатність, яка дозволяє програмі обробляти людську мову в письмовий формат.

У даному дипломі буде проведено дослідження основи технології розпізнавання мовлення, включно з тим, як вона працює, її історію, а також різні програми та проблеми, пов'язані з нею. Як результат буде створено програмний застосунок із імплементацією програмного модулю розпізнавання мовлення.

Мета даного дипломного проекту полягає в реалізації програмного застосунку за допомогою якого буде можливе транскрибування локальних аудіо та відеофайлів у текстові за допомогою використання програмного модулю розпізнавання мовлення. Все ця робота буде відбуватись у зручному в використанні інтерфейсі веб-застосунку.

Об'єкт дослідження - засоби розпізнавання мовлення і формування текстового опису аудіо або відеофайлу.

Предмет дослідження - методи автоматичного транскрибування мовлення.

Протягом даного дипломного проекту буде потрібно вирішити такі завдання дослідження:

1. Проаналізувати проблеми предметної області розпізнавання мовлення;

2. Провести аналіз бізнес процесів даної предметної області;
3. Провести порівняльний аналіз вже існуючих програмних систем;
4. Здійснити постановку задачі на виконання у даному дипломному проекті;
5. Визначити основні вимоги до програмного забезпечення і провести функціональний аналіз;
6. Вибрати інструментальні засоби за допомогою яких в подальшому буде відбуватись програмна реалізація застосунку;
7. Провести структурний аналіз та специфікацію програмного забезпечення;
8. Перевірити працездатність застосунку;
9. Розробити веб-застосунок який дозволяє транскрибувати аудіо та відеофайли у текстові.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ПРОГРАМНОГО МОДУЛЮ РОЗПІЗНАВАННЯ МОВЛЕННЯ

1.1 Аналіз проблем предметної області розпізнавання мовлення

Після аналізу книг Gerardus Blokdyk «Speech Recognition A Complete Guide» [1] та Bernd T. Meyer «Speech Recognition by Man and Machine» [2] можна сформулювати наступне визначення про основні поняття програмного модулю розпізнавання мовлення.

Розпізнавання мовлення (Speech Recognition), також відоме як автоматичне розпізнавання мовлення (Automatic Speech Recognition), комп'ютерне розпізнавання мовлення або перетворення мовлення в текст, – це здатність, яка дозволяє програмі обробляти людську мову в письмовий формат. Хоча його зазвичай плутають із розпізнаванням голосу, розпізнавання мовлення зосереджується на перекладі мовлення з вербального формату в текстовий, тоді як розпізнавання голосу лише намагається ідентифікувати голос окремого користувача.

У сьогодення доступно багато програм і пристроїв для розпізнавання мовлення, але більш передові рішення використовують штучний інтелект та машинне навчання (machine learning). Вони інтегрують граматику, синтаксис, структуру та композицію звукових і голосових сигналів для розуміння й обробки людської мови. В ідеалі вони навчаються по ходу роботи, покращуючи відповіді з кожною взаємодією.

Найкращі види систем також дозволяють організаціям налаштовувати та адаптувати технологію до своїх конкретних вимог – усе від мови та нюансів мовлення до впізнавання бренду. Наприклад:

- Зважування мови: підвищує точність, зваживши конкретні слова, які часто вимовляються (наприклад, назви продуктів або галузевий жаргон), крім термінів, які вже є в базовому словнику;

- Позначення доповідача: виводить транскрипцію, яка цитує або позначає тегами внесок кожного доповідача в розмову з кількома учасниками;
- Навчання з акустики: звертає увагу на акустичну сторону бізнесу. А саме, навчає систему адаптуватися до акустичного середовища (наприклад, навколишнього шуму в театрі) і стилів мовця (наприклад, висоти голосу, гучності та темпу);
- Фільтрування нецензурної лексики: використовує фільтри, щоб ідентифікувати певні слова чи фрази та очистити мовлення.

Розпізнавання мови продовжує розвиватися зараз, як ніколи. Більшість значних компаній просуваються в кількох сферах, щоб покращити взаємодію між людиною та машиною.

1.2 Аналіз бізнес процесів предметної області розпізнавання мовлення

Відповідно до IIBA «A Guide to the Business Analysis Body pf Knowledge (BABOK Guide)» [3] можна сформулювати наступне визначення про аналіз бізнес процесів предметної області розпізнавання мовлення.

Визначення профілів зацікавлених сторін важливе оскільки воно:

- Може допомогти аналітику прийняти важливі рішення про те, як продати певній організації зміни в бізнесі та як найкраще повідомити про особливості та переваги рекомендацій;
- Може допомогти аналітику зрозуміти потреби зацікавлених сторін, пріоритети, вимоги та їхній відносний вплив/владу над проектом;
- Може надати важливу інформацію про кожну зацікавлену сторону, що може допомогти аналітику розробити стратегію прийняття системи. Наприклад, профілювання може виявити

занепокоєння зацікавлених сторін, проблеми та потреби, які можуть бути уточнені у вимогах;

- Профілювання зацікавлених сторін може допомогти аналітику виявити прогалини в їхніх знаннях про стейкхолдерів, які спонукатимуть його шукати та отримувати необхідну інформацію.

Зацікавленні сторони являють собою особу або ж члена групи, які напряму чи опосередковано беруть участь у проекті, бізнес операції чи організації.

Серед діючих лиць можна визначити наступних зацікавлених сторін:

- CEO;
- Інвестори;
- TechLead;
- Розробники;
- Тестувальники;
- Конкуренти;
- Користувачі;
- Потенційні клієнти;

Кожен з попередньо наведених стейкхолдерів має різний ступінь впливу та зацікавленості. Умовно категоризуємо їх на 4 групи, відповідно до матриці Менделоу, аби краще зрозуміти як правильно з ними взаємодіяти.

Візуально дану діаграму можна побачити на рисунку наведеному нижче, (рис. 1).

Рисунок 1 – Матриця Менделоу

1.3 Порівняльний аналіз вже існуючих програмних систем предметної області розпізнавання мовлення

Розглянемо більш детальніше використовування розпізнавання мовлення, а саме транскрибування аудіо або відео файлів, у галузі медіа та розваг. Як приклад використаємо автоматичні субтитри YouTube і Netflix.

Функція автоматичних субтитрів YouTube використовує технологію автоматичного розпізнавання мовлення (ASR) для транскрибування аудіо або відео та відображення тексту як субтитрів на екрані.

Даний процес працює наступним чином:

1. Програма YouTube для розпізнавання мовлення аналізує аудіо доріжку відео та створює текстову розшифровку вимовлених слів;

2. Стенограма потім синхронізується з відео, щоб субтитри з’являлися на екрані в потрібний час;
3. Технологія YouTube ASR намагається автоматично виправляти будь-які помилки в транскрипції, як-от орфографічні або неправильно прослухані слова;
4. Нарешті, YouTube дозволяє творцям вмісту редагувати транскрипцію, щоб виправити будь-які помилки, що залишилися, або підвищити загальну точність транскрипції.

Субтитри Netflix працюють так само, як субтитри YouTube, але з кількома ключовими відмінностями.

Даний процес працює наступним чином:

1. Як і YouTube, Netflix використовує технологію автоматичного розпізнавання мовлення (ASR) для транскрипції аудіо відео та створення текстової розшифровки вимовлених слів;
2. Стенограма потім синхронізується з відео, щоб субтитри з’являлися на екрані у відповідний час;
3. Однак, на відміну від YouTube, Netflix використовує професійних перекладачів для створення точних і культурних субтитрів для різних мов і регіонів;
4. Netflix також пропонує кілька варіантів субтитрів для кожного відео, включаючи різні мови та стилі;
5. Крім того, Netflix дозволяє глядачам налаштовувати вигляд субтитрів, наприклад змінювати розмір і колір шрифту або колір фону, щоб полегшити їх читання.

Переглянувши попередньо наведені приклади використання модулю розпізнавання мовлення у транскрибуванні аудіо і відео файлів створимо порівняльну таблицю для покращеного розуміння усіх переваг і недоліків кожної із систем, (табл. 1).

Таблиця 1. Результати порівняльного аналізу

Функціонал	Об’єкт	
	Автоматичні субтитри YouTube	Автоматичні субтитри Netflix
Створення текстової розшифровки вимовлених слів	+	+
Синхронізація з відео	+	+
Автоматичне виправлення помилок	+	-
Використовування професійних перекладачів	-	+
Дозволяє творцям вмісту редагувати транскрипцію	+	-
Включає декілька мов субтитрів	+	+
Дозволяє глядачам налаштовувати вигляд субтитрів	-	+
Дозволяє зберігати файл із усіма транскрибованими словами	-	-

Загалом транскрипція аудіо та відео є цінним інструментом, який можна застосувати в різних налаштуваннях, щоб допомогти покращити доступність, продуктивність і ефективність.

1.4 Постановка задачі розробки програмного модулю

Головна мета даного дипломного проекту буде полягати в створенні програмного застосунку за допомогою буде можлива транскрипція аудіо і відео файлів у текстові за допомогою використання програмного модулю розпізнавання мовлення.

По завершенню роботи над програмним застосунком користувач буде мати змогу працювати із веб-застосунком за допомогою якого буде можлива транскрипція аудіо і відео файлів у текстові локально.

Коли користувач буде вперше відкривати даний застосунок його зустріне головне вікно у якому буде міститись поле у яке він зможе завантажити, перетягуючи бажаний аудіо або відео файл із папки або переглянувши і вибравши його із локального диску, для проведення транскрипції. На рисунку нижче наведено макет (рис. 2):

Рисунок 2 – Макет головної сторінка застосунку

Також, у лівому верхньому куті застосунку буде розміщена ікона меню. Натиснувши на неї у користувача відкриється бокова панель у якій буде можливо прослухати аудіо або відео файл який було вибрано для транскрипції і буде наведено перелік обмежень які користувач повинен буде враховувати сам перед завантаженням файлу, (рис. 3):

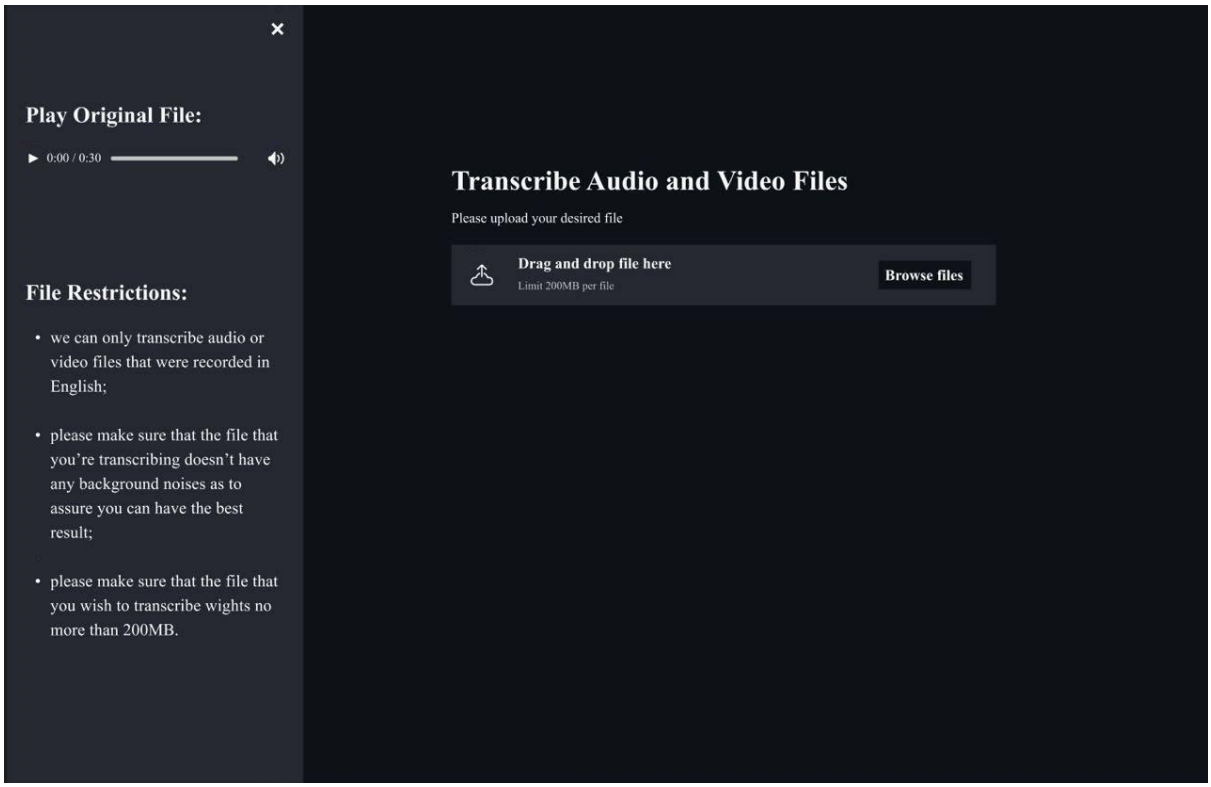

Рисунок 3 – Макет меню застосунку

Тепер, розглянемо дії програми коли користувач завантажить в неї аудіо або відео файл який буде транскрибуватись. При завантаженні обраного файлу його буде відображено на сторінці застосунку і, також, буде відбуватись відображення індикатора виконання транскрипції, (рис. 4):

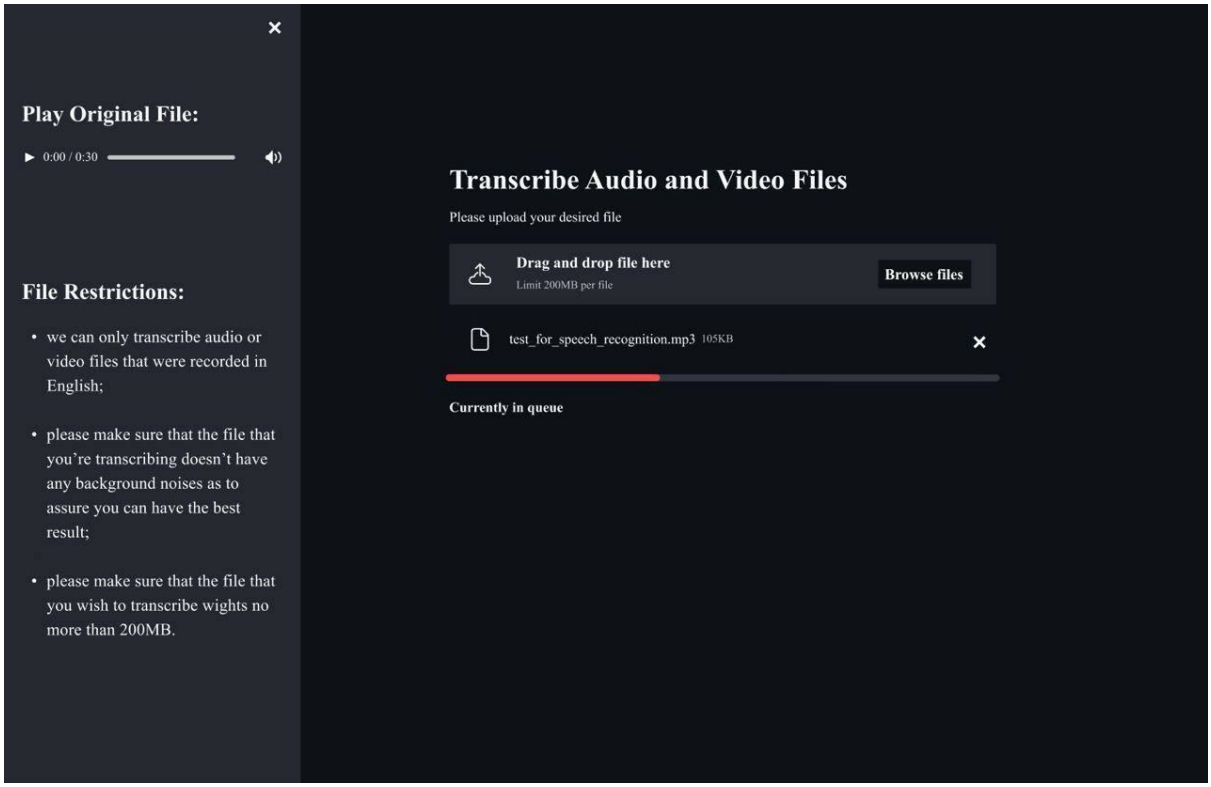

Рисунок 4 – Макет завантаження у застосунок бажаного файлу

По закінченню проведення транскрибування у користувача буде видаватись текст промовлених слів транскрибованого аудіо або відео файлу, (рис. 5):

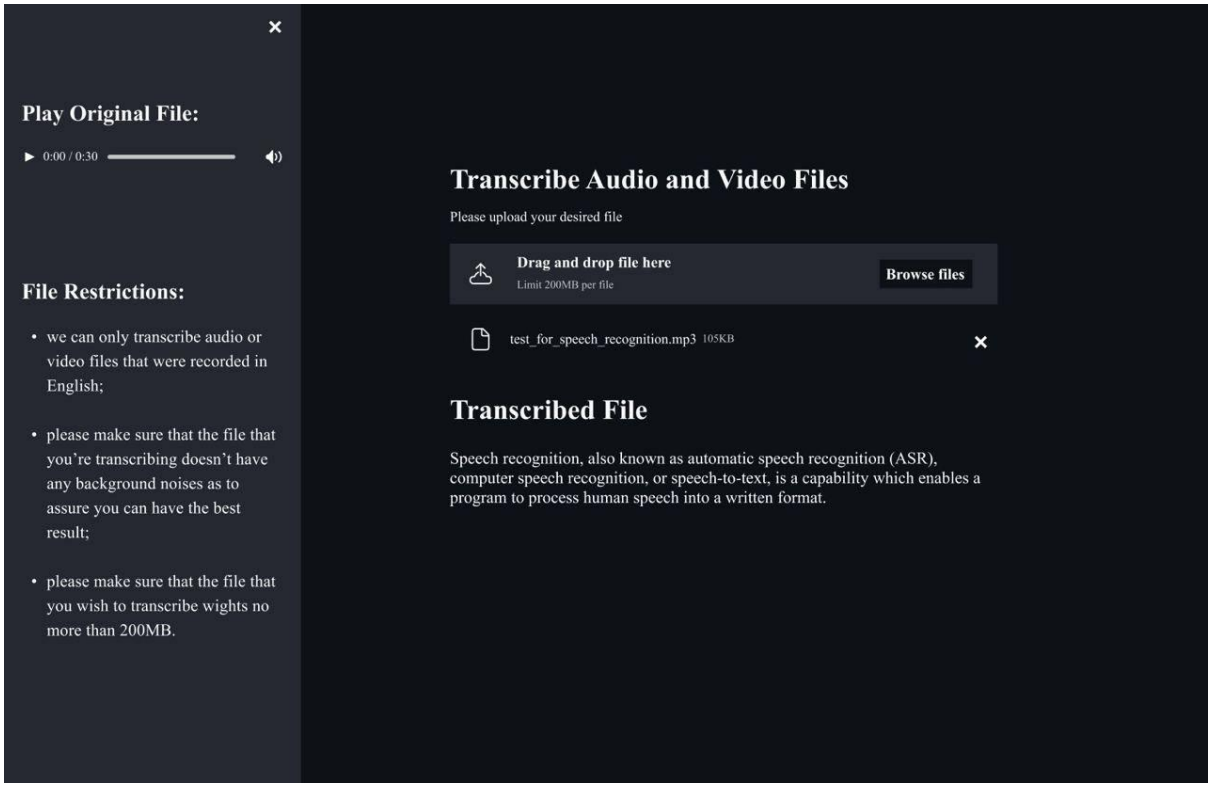

Рисунок 5 – Макет завершення транскрипції

Розглянемо із якими саме файлами повинна буде працювати система. Вона повинна буде мати змогу обробляти такі групи аудіо файлів, як:

- Файли нестисненого формату;
- Файли формату зі стисненням без втрат;
- Файли формату зі стисненням з втратами.

Окрім того як було наведено раніше система повинна буде мати змогу працювати із відео файлами і їхніми основними видами розширення.

Завантаження даних файлів має бути можливе як локально так і за допомогою посилання на один із них.

На жаль, система матиме певні обмеження при роботі із вище перерахованих файлів. Дані обмеження має передбачати користувач, але він буде попереджений про них як попередньо до завантаження файлів. Нижче наведені можливі обмеження:

- Обмеження стосовно мови: майбутня система буде мати змогу розпізнавати слова, записані лише англійською мовою. При спробі транскрипції файлу створеного на будь-яких інших мовах буде видаватись пустий текстовий файл;
- Обмеження стосовно якості файлу: якщо, наприклад, користувач обрав аудіо файл для транскрипції вміст якого включає в собі фоновий шум в результаті роботи програми текстовий файл буде мати не найкращу якість;
- Обмеження стосовно розмірів файлів: майбутній файл не має перевищувати 200MB. В випадку коли аудіо або відео файл буде мати формат більше зазначеного система буде припиняти роботу і видавати повідомлення про помилку.

Реалізації транскрипції аудіо і відео файлів у текстові буде можлива за допомогою використання наступних елементів:

- AssemblyAI API за допомогою якого буде можливе проведення розпізнавання мовлення;
- Streamlit за допомогою якого буде відбуватись створення веб-застосунку;
- Request library від Python за допомогою якого буде відбуватись надання запитів до AssemblyAI API;
- Python-dotenv, для зчитування змінних з файлів .env.

Розглянемо контекстну IDEF0 діаграму процесу діяльності даного програмного застосунку, (рис. 6):

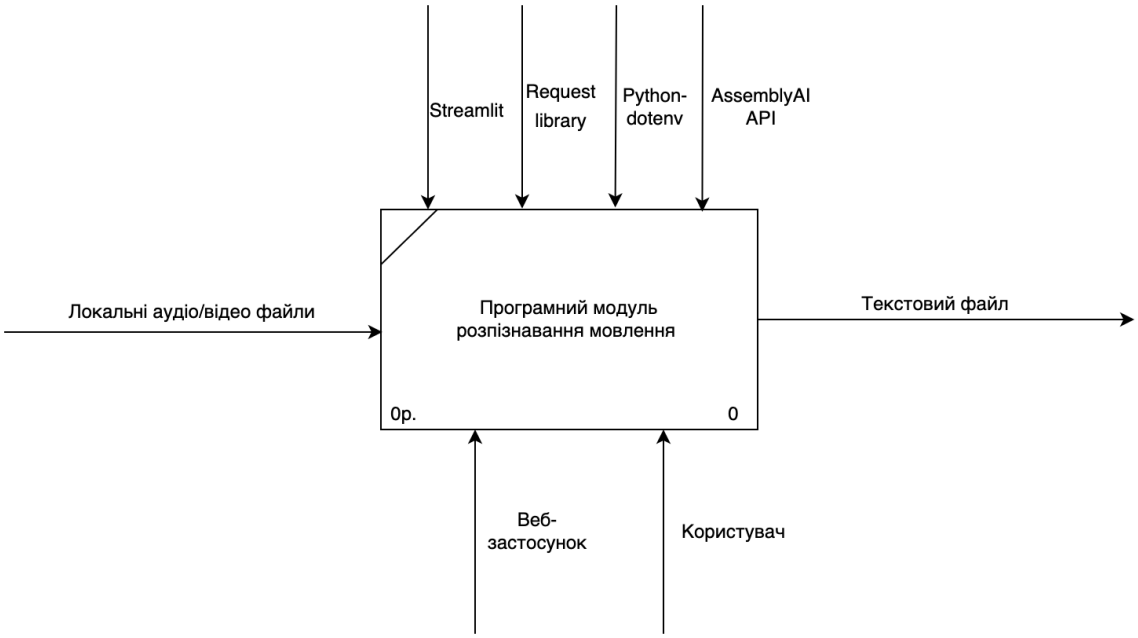

Рисунок 6 – Контекстна діаграма діяльності майбутнього застосунку

1.5 Функціональний аналіз та визначення основних вимог до програмного забезпечення

Важливим етапом є визначення проведення функціонального аналізу та визначення функціональних та функціональних вимог [5].

Після поставлення задачі майбутнього застосунку із використанням програмного модулю розпізнавання мовлення було визначено такі функціональні і нефункціональні вимоги:

- Функціональні вимоги:

- Перетворення аудіо і відео файлів у текстові;
- Проведення роботи із англомовними даними;
- Обробка файлів розміром не більше 200MB;

- Нефункціональні вимоги:

- Зручний та зрозумілий в використанні інтерфейс веб-застосунку;
- Можливість запуску на усіх видах і версіях операційних систем;
- Видача читабельного і відносно коректного текстового файлу;
- Можливість розпізнавання різних тембрів голосів і акцентів;
- Проведення роботи повинно здійснюватися не довше 30 секунд;
- При завантаженні будь-яких інші видів файлів окрім аудіо і відео буде видаватись помилка;
- Надійність та стійкість до збоїв в роботі.

Висновки до першого розділу:

Під час виконання роботи над першим розділом було проведено аналіз проблем предметної області розпізнавання мовлення. Було проаналізовано уже існуючі моделі роботи із даною предметною областю і були поставлені основні вимоги до майбутнього програмного забезпечення. Далі основна задача буде полягати у створенні проектних рішень області програмного застосунку для перетворення аудіо і відео файлів у текстові за допомогою програмного модулю розпізнавання мовлення.

РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО МОДУЛЮ АВТОМАТИЧНОГО ТРАНСКРИБУВАННЯ МОВЛЕННЯ

2.1 Алгоритми для здійснення розпізнавання мовлення

Вважається, що галузь роботи із розпізнаванням мовлення – це одна з найскладніших областей інформатики, яка включає лінгвістику, математику та статистику. Розпізнавання мовлення складається з кількох компонентів, таких як ввід мови, виділення ознак, вектори ознак, декодер і вихід слова. Декодер використовує акустичні моделі, словник вимови та моделі мови для визначення відповідного результату.

Технологія розпізнавання мовлення оцінюється за рівнем точності, тобто частотою помилок у словах (word error rate, WER), і швидкістю. На частоту помилок у словах може впливати ряд факторів, наприклад: вимова, акцент, тон, гучність і фоновий шум. Досягнення людської паритетності (рівності) – тобто рівня помилок, рівного рівню розмові двох людей – давно було метою систем розпізнавання мовлення. Переважно рівень помилок у словах оцінюється приблизно в 4%.

Для перетворення мовлення в текст та підвищення точності транскрипції використовуються різні алгоритми та методи обчислення. Нижче наведено короткі пояснення деяких із найпоширеніших методів:

- Natural language processing (NLP). Хоча NLP не обов’язково є спеціальним алгоритмом, який використовується для розпізнавання мовлення, це сфера штучного інтелекту, яка зосереджена на взаємодії між людьми та машинами через мову та текст. Багато мобільних пристроїв включають у свої системи розпізнавання мовлення для здійснення голосового пошуку, наприклад, Siri або надання більших можливостей для обміну текстовими повідомленнями;
- Приховані моделі Маркова. Побудовані на моделі ланцюга Маркова, яка передбачає, що ймовірність певного стану

залежить від поточного стану, а не від попередніх станів. У той час як модель ланцюга Маркова корисна для спостережуваних подій, таких як введення тексту, приховані моделі Маркова дозволяють включати приховані події, такі як теги частини мови, в імовірнісну модель. Вони використовуються як моделі послідовності в розпізнаванні мовлення, призначаючи мітки кожному блоку, тобто словам, складам, реченням у послідовності. Ці мітки створюють ймовірний план з наданими вхідними даними, що дозволяє визначити найбільш відповідну послідовність міток;

- N-грами. Це найпростіший тип моделі мови (language model), яка призначає ймовірності реченням або фразам. N-грама – це послідовність N-слів. Наприклад, «замовте цю піцу» – це триграми або 3-грами, а «будь-ласка, замовте піцу» – це 4-грами. Граматика та ймовірність певних послідовностей слів використовуються для покращення розпізнавання та точності;
- Нейронні мережі. Нейронні мережі, які переважно використовуються для алгоритмів глибокого навчання, обробляють навчальні дані, імітуючи взаємозв’язок людського мозку через рівні вузлів. Кожен вузол складається з входів, ваг, зміщення (або порогу) і виходу. Якщо це вихідне значення перевищує заданий поріг, воно «спрацьовує» або активує вузол, передаючи дані на наступний рівень у мережі. Нейронні мережі вивчають цю функцію через контрольоване навчання, коригуючи на основі функції втрат через процес градієнтного спуску. Хоча нейронні мережі, як правило, точніші та можуть приймати більше даних, це відбувається за рахунок ефективності продуктивності, оскільки вони, як правило,

повільніше навчаються порівняно з традиційними моделями мовлення;

- Speaker Diarization (SD). Алгоритми діаризації мовця ідентифікують і сегментують мову за особою мовця. Це допомагає програмам краще розрізняти людей у розмові та часто застосовується в коллцентрах, розрізняючи клієнтів і торгових агентів;
- Штучний інтелект. Штучний інтелект і методи машинного навчання, такі як глибоке навчання та нейронні мережі, поширені в розширеному програмному забезпеченні розпізнавання мовлення. Ці системи використовують граматику, структуру, синтаксис і композицію звукових і голосових сигналів для обробки мови. Системи машинного навчання отримують знання з кожним використанням, завдяки чому вони добре підходять для таких нюансів, як акценти.

Для виконання завдання, поставленого у даному дипломному проекті, буде проводитись робота і імплементація алгоритму розпізнавання мовлення за допомогою штучного інтелекту, а саме за допомогою імплементації документації AssemblyAI.

AssemblyAI – це документація транскрибування мови в текст, яка використовує штучний інтелект для транскрипції аудіо або відео файлів у текстові. Для впровадження ШІ у власний сервіс AssemblyAI використовує комбінацію алгоритмів машинного навчання та методів обробки природної мови.

Один із способів, яким AssemblyAI впроваджує штучний інтелект у свій сервіс, це використання алгоритмів глибокого навчання для навчання нейронної мережі розпізнаванню та транскрипції мови. Це передбачає подачу великих обсягів аудіо даних у нейронну мережу та налаштування вагових коефіцієнтів мережі на основі вихідних даних. Оскільки мережа

навчається на більшій кількості даних, вона стає краще розпізнавати шаблони мовлення та створювати точні транскрипції.

Крім того, AssemblyAI може використовувати методи обробки природної мови для підвищення точності своїх транскрипцій. Це передбачає аналіз структури та змісту транскрибованого тексту для виявлення та виправлення помилок. Наприклад, він може використовувати мовленнєві моделі та статистичні алгоритми для виявлення загальних моделей мовлення та прогнозування ймовірності появи певних слів або фраз у аудіо.

Основна її модель базується на архітектурі глибокого навчання, що називається згортковою нейронною мережею (CNN), яка навчається на великих обсягах позначених мовленнєвих даних.

2.2 Математична модель CNN і автоматичного розпізнавання мовлення (ASR)

Основною метою системи автоматичного розпізнавання мовлення (ASR) є перетворення вхідного звукового сигналу $x = (x_1, x_2, \dots x_T)$ з певною довжиною T у послідовність слів або символів (тобто мітки) $y = (y_1, y_2, \dots y_N), y_n \in V$, де V – це словниковий запас. Мітки можуть бути мітками рівня символів (тобто букв) або міток рівня слова (тобто слів).

Найімовірніша вихідна послідовність визначається так:

$$\hat{y} = \frac{argarg\, p(y|x)}{y \in V}$$

(2.1)

Типова система ASR має такі етапи обробки:

- Попередня обробка;
- Вилучення ознак;
- Класифікація;
- Моделювання мови.

Етап попередньої обробки спрямований на покращення аудіо сигналу шляхом зменшення співвідношення сигнал/шум, зменшення шуму та фільтрації сигналу.

Загалом, функції, які використовуються для ASR, витягуються за допомогою певної кількості значень або коефіцієнтів, які генеруються шляхом застосування різних методів на вхідних даних. Цей крок має бути надійним, що стосується різних факторів якості, таких як шум або ефект луни.

Загальний потік автоматичного розпізнавання мови (ASR) можна представити, як наведено нижче, (рис. 7):

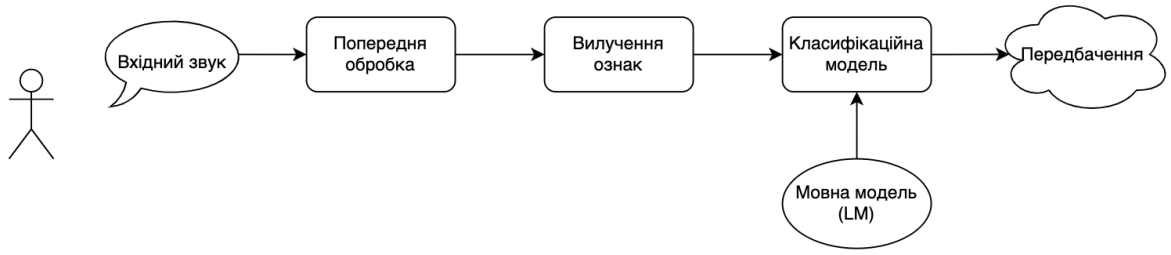

Рисунок 7 – Автоматичне розпізнавання мовлення

Модель класифікації спрямована на пошук усного тексту, який міститься у вхідному сигналі. Він бере витягнуті функції на етапі попередньої обробки та генерує вихідний текст.

Мовна модель (LM) є важливим модулем, оскільки вона фіксує граматичні правила або семантичну інформацію мови. Мовні моделі важливі для розпізнавання вихідного токена з моделі класифікації, а також для внесення виправлень у вихідний текст. Однією із них є математична модель згорткових нейронних мереж (CNN), (рис. 8):

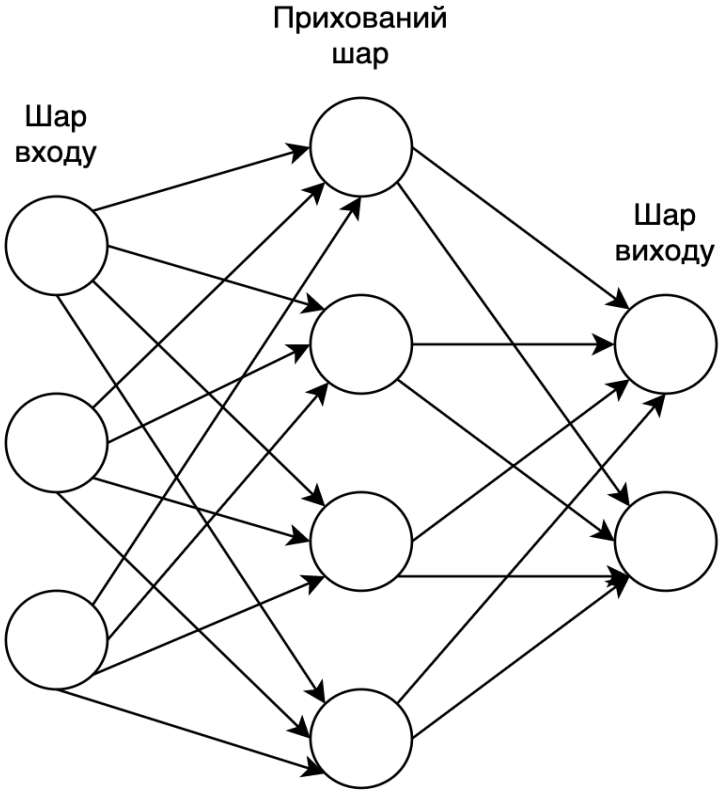

Рисунок 8 – Архітектура згорткової нейронної мережі

Згорткові нейронні мережі спочатку були реалізовані для завдань комп'ютерного зору (CV). Останніми роками CNN також широко застосовуються в області обробки природної мови (NLP) завдяки їх хорошій генерації та здатності до розрізнення.

Дуже типова архітектура CNN складається з кількох згорткових і об'єднуючих шарів із повністю зв'язаними шарами для класифікації. Згортковий шар складається з ядер, які згортаються разом із входом. Згорткове ядро ділить вхідний сигнал на менші частини, а саме сприйнятливе поле ядра. Крім того, операція згортки виконується шляхом множення ядра з відповідними частинами вхідних даних, які знаходяться в рецептивному полі. Згорткові методи можна згрупувати відповідно в 1-вимірні та 2-вимірні мережі.

2D-CNN будують 2D-карти характеристик на основі акустичного сигналу. Подібно до зображень, вони організовують акустичні характеристики, тобто особливості MFCC, у двовимірній карті функцій, де одна вісь представляє частотну область, а інша – часову область. Навпаки, 1D-CNN приймають акустичні характеристики безпосередньо як вхідні дані.

В 1D-CNN для розпізнавання мовлення, кожен вхідна карта функцій $X = (X_1, \dots, X_I)$ є пов'язаними до багатьох карт функцій $O = (O_1, \dots, O_J)$.

Операцію згортки можна записати так:

$$O_j = \sigma\left(\sum_{i=1}^{I} X_i \omega_{i,j}\right), j \in [1, J] \quad (2.2)$$

де w локальна вага.

- В 1D-CNN: w, O – вектори;
- В 2D-CNN вони матриці.

У випадку AssemblyAI CNN використовується для аналізу спектрограми вхідного аудіосигналу. Спектрограма — це візуальне представлення частотного спектру звукового сигналу в часі. Аналізуючи спектрограму, CNN можуть ідентифікувати шаблони та особливості звукового сигналу, які мають відношення до розпізнавання мовлення.

Моделі AssemblyAI, засновані на CNN, використовують техніку, яка називається частотно-часовим маскуванням, щоб підвищити свою точність. Частотно-часове маскування передбачає поділ спектрограми на невеликі сегменти та застосування маски до кожного сегменту. Маска вибірково відфільтровує шум і немовні елементи, залишаючи лише найбільш релевантні частини сигналу для аналізу CNN. Це дозволяє CNN зосередитися на найважливіших характеристиках мовного сигналу, що призводить до більш точної транскрипції.

2.3 Розробка архітектури системи із використанням модулю розпізнавання мовлення

Як було наведено попередньо мета даного дипломного проекту буде полягати в створенні програмного застосунку за допомогою буде можлива транскрипція аудіо і відео файлів у текстові за допомогою використання програмного модулю розпізнавання мовлення. Проаналізуємо як саме виконання даної задачі буде здійснюватися.

Спершу, побудуємо дерево функцій для даного програмного застосунку аби легше оцінювати, оптимізувати та можливо в подальшому змінювати певні з них, (рис. 9):

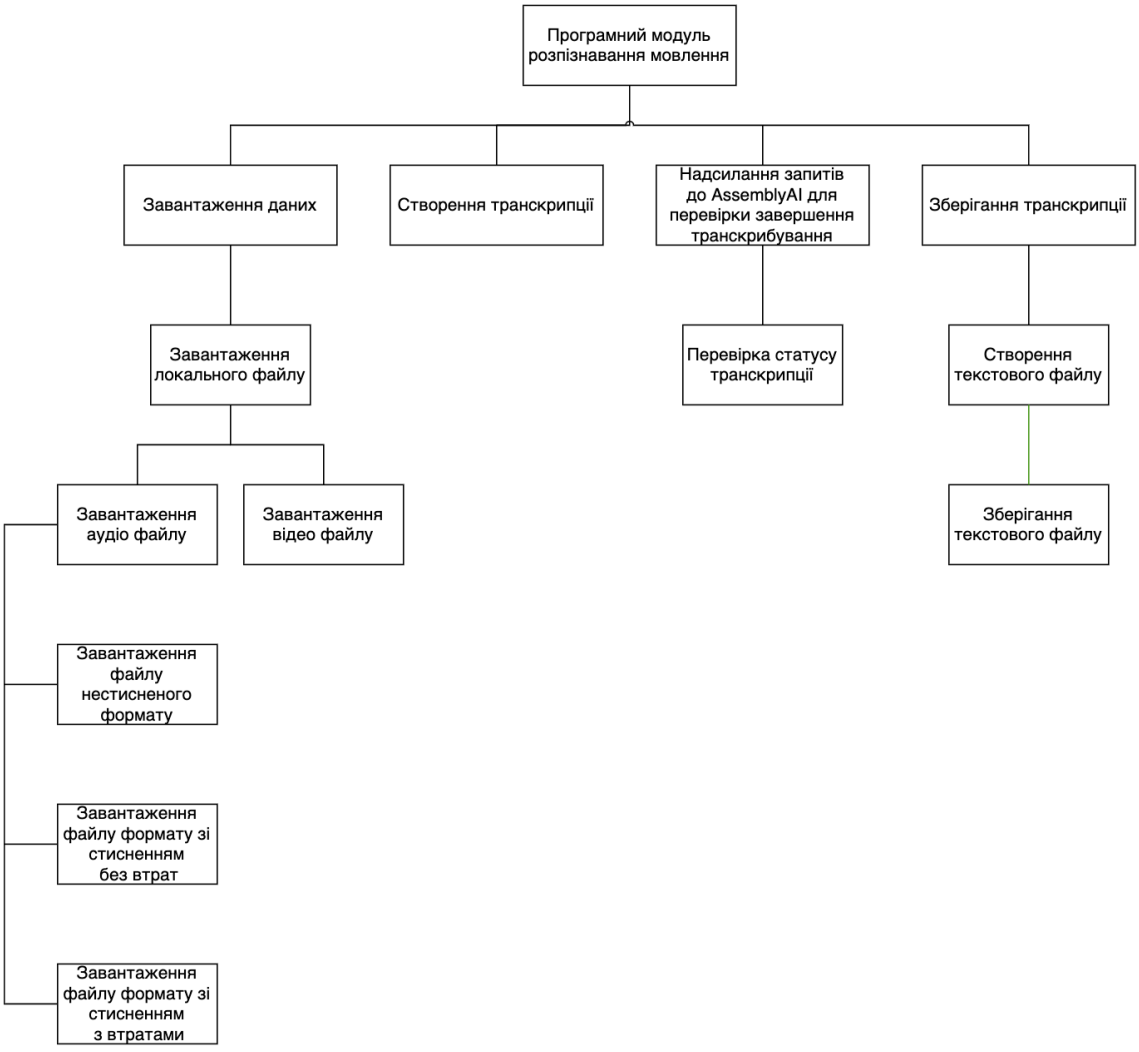

Рисунок 9 – Дерево функцій

У попередньому розділі було розглянуто контекстну діаграму діяльності майбутнього застосунку, тепер, побудуємо декомпозицію даної діаграми, (рис. 10):

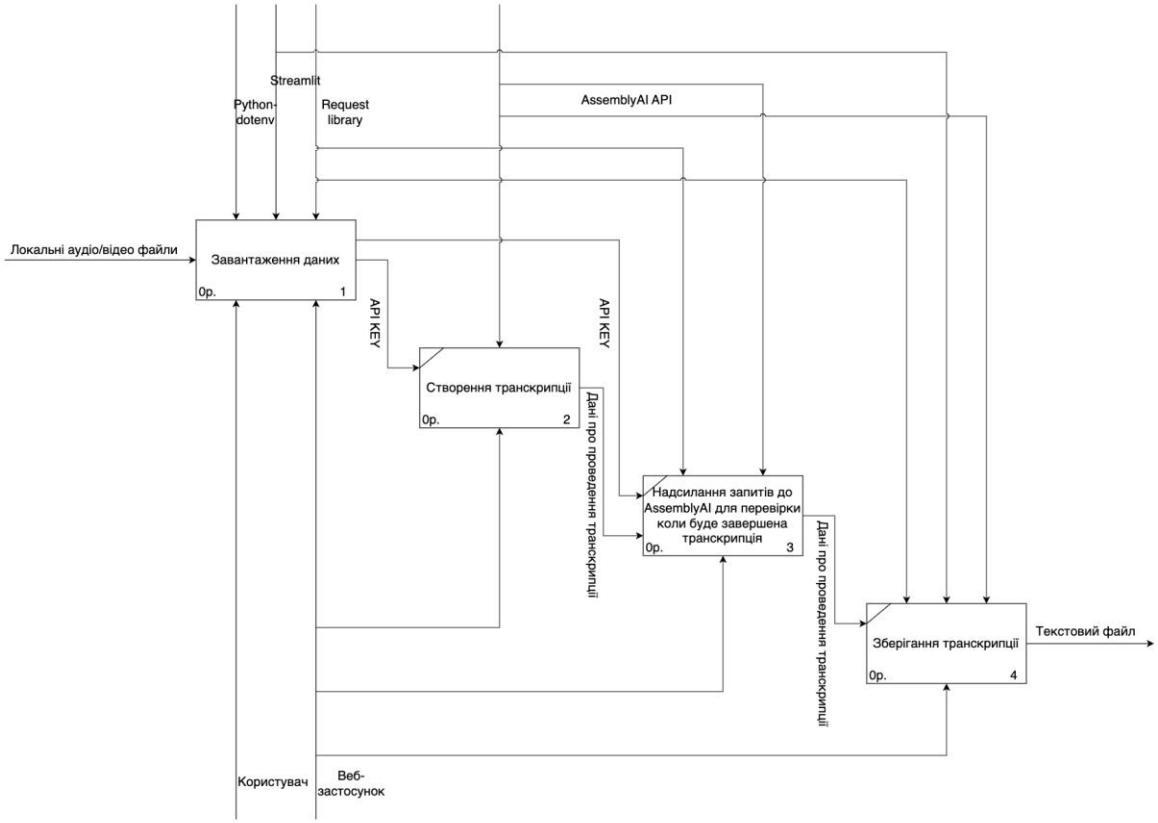

Рисунок 10 – Декомпозиція контекстної діаграми процесу діяльності застосунку

Головна робота системи буде відбуватись у двох класах: «main» і «transcribe». Клас «main» буде включати в собі увесь код який відноситься до Streamlit UI в той час як клас «transcribe» буде містити в собі помічні функції і код який взаємодіє із AssemblyAI API. Розглянемо поступово роботу даної системи, а саме, транскрипцію аудіо або відео файлів.

Спочатку розглянемо схему архітектури застосунку в який буде входити програмний модуль із використанням розпізнавання мовлення, (рис. 11):

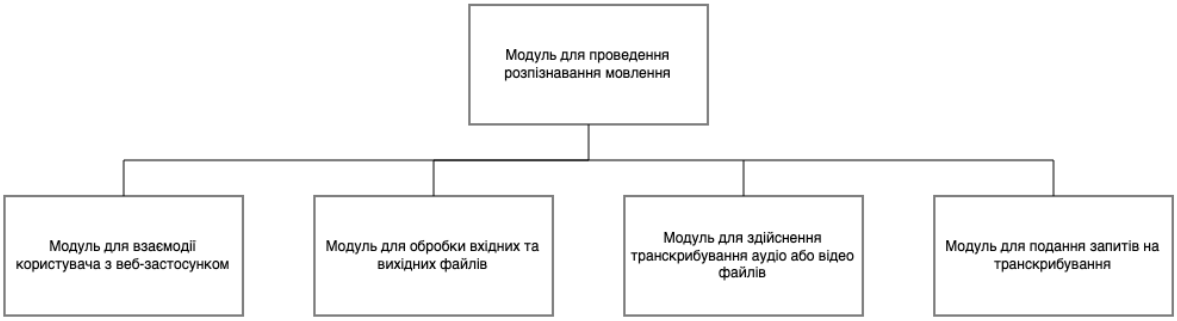

Рисунок 11 – Схема архітектури програмного модулю

Як було зазначено вище у даний застосунок будуть входити помічні функції за допомогою яких буде відбуватись завантаження вказаних файлів до серверів AssemblyAI аби моделі перекладали і повертали транскрибований текст. Код для даних функцій буде міститись у класі «transcribe». Тепер розглянемо кожну із майбутніх головних функцій:

- Перша головна функція. Завантаження локального аудіо або відео файлу до AssemblyAI.

Дана функція буде відповідати за завантаження аудіо або відео файлу який зберігається локально на персональному комп’ютері. Модель AssemblyAI очікує, що доступ до файлу буде здійснюватися через URL у самому коді проте, у майбутнього застосунку буде наявний інтерфейс у вигляді веб-застосунку з яким зможе взаємодіяти користувач. Саме тому буде існувати потреба завантажувати файли до blob сховища щоб зробити їх доступними через URL.

Для того, щоб задовольнити вказаним вимогам буде потрібно зробити POST запит до кінцевої точки AssemblyAI API. Даний запит буде включати в собі тимчасовий URL до файлу і даний URL можна буде направити назад до AssemblyAI API. Даний URL є приватним і є доступним лише до AssemblyAI серверів.

Всі завантаження будуть одразу видалені після проведення транскрибування і ніколи не будуть зберігатись. Аби зробити даний POST запит буде використовуватись бібліотека Requests.

- Друга головна функція. Завантаження файлу для транскрибування.

Після того як ми отримали функцію для отримання URL для обраного аудіо або відео файлу даний URL буде використано і створено запит до кінцевої точки яка буде відповідати за транскрибування файлу. Ця функція повинна також бути у класі «transcribe».

Спочатку, коли надсилається запит на транскрипцію аудіо або відео файл має статус «queued» («в черзі»). Про те, як файл переходить із «queued» до «complete» буде детальніше описано в останній головній функції. Перед цим потрібно лише зробити запит до кінцевої точки транскрипції разом із URL до файлу.

- Третя головна функція. Завантаження аудіо або відео транскрипції

Коли отримується ID транскрипції аудіо або відео файлу ми можемо зробити запит GET до кінцевої точки AssemblyAI API аби перевірити статус транскрипції.

Статус транскрипції буде мінятись з «queued» до «processing» до «completed» поки не буде виявлено ніяких помилок. Поки це буде відбуватись застосунок буде надсилати запити до кінцевої точки поки не буде видано відповідь про те, що бажаний транскрибований файл достиг статусу «completed».

Тобто, дана головна функція буде отримувати наявний статус файлу.

- Четверта головввна функція. Надання запитів на транскрипцію від UI.

Попередня функція буде викликати перші дві помічні функції послідовно, в той час як дана функція буде підключена до кнопки «Upload» у веб-застосунку. В неї буде лише один параметр: об’єкт файл. Четверта помічна функція буде виконувати наступне:

- Завантаження токена API із .env файлу;

- Використовувати токен для попередньо визначених функцій;
- Повертати ID транскрипції.

По завершенню роботи усіх функцій і модулів майбутньої програми у користувача буде відобрвжвтись готовий транскрибований текст. Окрім цього буде відбуватись завантаження створеного текстового файлу на локальний комп’ютер.

2.4 Функціональний аналіз

Транскрибування аудіо або відео файлів в текст в подальшому застосунку, як було наведено в попередньому розділі, буде відбуватись з чотирьох кроків:

- Першим кроком є завантаження аудіо або відео файлу в програмне забезпечення. Це можна буде зробити через веб-інтерфейс яке буде мати певні вимоги до завантаженого файлу;
- Після завантаження аудіо або відео файлу програмне забезпечення буде відбуватись перетворення аудіо сигналу на текст за допомогою технології автоматичного розпізнавання мовлення. Залежно від якості звуку та мовлення цей процес може мати різний результат;
- Після завершення транскрипції програмне забезпечення буде надсилати сповіщення користувачеві, щоб повідомити, в якій стадії перебуває транскрипція. Чи готова вона, чи ні;
- Нарешті, коли транскрибування завершиться, користувач зможе побачити його результат і зберегти його для подальшої роботи із ним.

Нижче наведено VAD-діаграма процесу транскрибування аудіо і відео файлів які були описані попередньо для майбутнього програмного застосунку, (рис. 12):

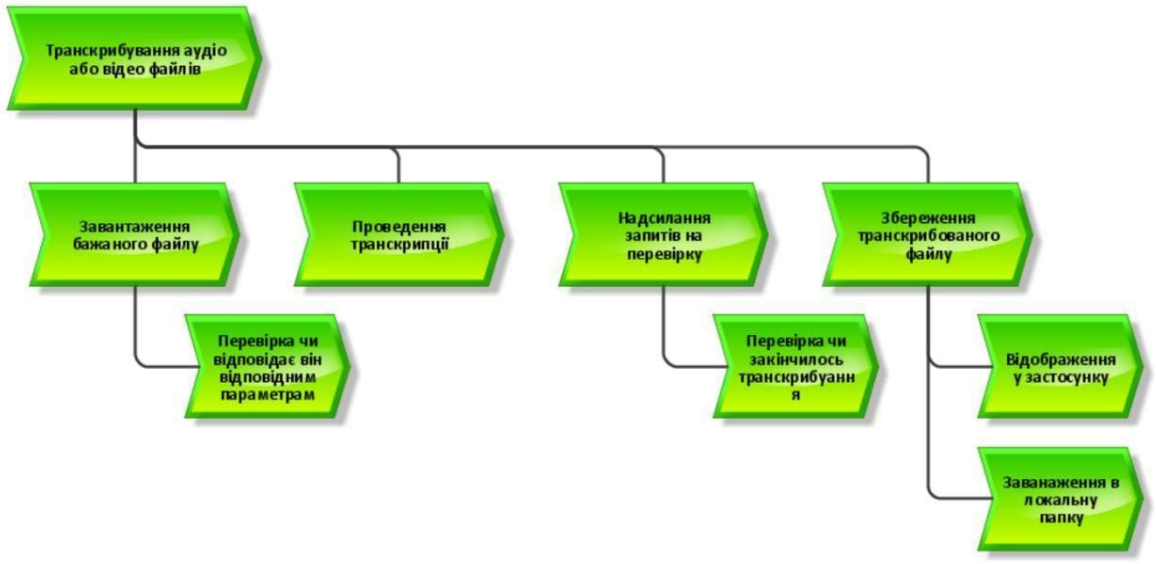

Рисунок 12 – VAD-діаграма процесу транскрибування аудіо і відео файлів

2.5 Визначення області застосування та перспективність вибраного алгоритмічного підходу

Практична цінність розробленого програмного модулю полягає в тому, що він може буде використаний в різних галузях, дозволить спростити та автоматизувати процес транскрибування аудіо чи відео файлів, що є трудомістким і часто виснажливим завданням, особливо вручну. Саме тут даний програмний застосунок зможе бути ефективним та корисними, оскільки він може автоматизувати процес і заощадити значну кількість часу та зусиль. Серед переваг розробленого програмного модулю є наступні:

- Доступність. Якщо користувач не можете вільно друкувати або маєте проблеми зі зором, розпізнавання мовлення і, як результат, даний застосунок може бути дуже корисним.
- Легкість в використанні. Завдяки зрозумілому і легкому в використанні інтерфейсі даний застосунок буде зручним у використанні для користувачів;

- Швидкість. Доволі часто існує потреба переробити аудіо або відео файл у текстовий і замість того, щоб робити це вручну буде набагато ефективніше і швидше використати модуль розпізнавання мовлення використаний в даному застосунку;
- Рентабельність. Наймання професіонала для даного завдання може бути дорогим і неможливим для певних осіб або малого бізнесу. Даний додаток може бути економічно ефективною альтернативою, що дозволяє швидко й точно транскрибувати аудіо чи відео файли за нижчою ціною;
- Можливість постійного автоматичного вдосконалення. Системи із розпізнаванням мовлення з часом стають ефективнішими та простішими у використанні. Коли системи виконують завдання розпізнавання мовлення, вони генерують більше даних про людську мову та стають кращими в тому, що вони роблять.

Проте, хоча даний застосунок є досить корисним, технології розпізнавання мовлення досі мають певні недоліки які не є до кінця виправленими. Дані недоліки включають в собі:

- Непослідовна продуктивність. Система може бути не в змозі точно фіксувати слова через варіації у вимові, відсутність підтримки інших мов, окрім англійської, і нездатності сортувати фоновий шум. Навколишній шум може бути особливо складним. Акустичні тренування можуть допомогти відфільтрувати це, але ці програми також не є ідеальні. Іноді неможливо виділити людський голос;
- Проблеми з вихідним файлом. Успішність роботи даного застосунку залежить також від використовуваного записуючого обладнання, а не лише від програмного забезпечення.

- Розміри бажаного файлу для транскрибування. Дана система буде мати змогу працювати з файлами які не перевищують розмір 200MB на файл;

В цілому, даний застосунок має потенціал автоматизувати процес транскрибування, його можна буде використовувати для різних галузей. Також, існує досить велика ймовірність того, що він продовжуватиме відігравати дедалі важливішу роль у повсякденному житті в наступні роки.

Висновки до другого розділу:

Під час виконання роботи над другим розділом було розроблено архітектуру програмного модулю автоматичного транскрибування мовлення для перетворення аудіо і відео файлів у текстові за допомогою програмного модулю розпізнавання мовлення. Були запропоновані проектні рішення для реалізації програмного модулю. Далі основна задача буде полягати у практичній реалізації області програмного застосунку для перетворення аудіо і відео файлів у текстові за допомогою програмного модулю розпізнавання мовлення.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ АВТОМАТИЧНОГО ТРАНСКРИБУВАННЯ МОВЛЕННЯ

3.1 Вибір інструментальних засобів для програмної реалізації застосунку

Для реалізації майбутнього застосунку, за допомогою якого буде являтись можлива транскрипція аудіо і відео файлів локально, будуть використовуватись наступні інструментальні засоби:

Мови програмування:

- Python, як основної мови розробки через її кросплатформеність, простоту та гнучкість у використанні. За допомогою даної мови буде відбуватись створення основних функцій які будуть відповідати за транскрибування файлів.

Бібліотеки:

- AssemblyAI, для транскрибування аудіо або відео в текст. Вона надає REST API, за допомогою якої можна якою будь-якою мовою, яка виконує виклик REST API, наприклад JavaScript, PHP, Python тощо. Як було наведено попередньо в даній роботі буде використовуватись Python для надсилання запитів до даної API;
- Streamlit, для створення веб-застосунку в якому користувач буде здійснювати транскрипцію. Даний фреймворк використовується для побудови UI для Machine Learning Models;
- Requests, дана бібліотека буде виконувати функцію подання запитів до AssemblyAI REST API;
- Python-dotenv, для зчитування змінних з файлів .env.

Середовища розробки:

- Visual Studio Code, як основного середовища для реалізації і роботи програмного застосунку, через його зручність при використанні.

Середовища для створення макету веб-застосунку:

- Figma за допомогою якого було створено макет майбутнього веб-застосунку для зручного користування користувачів.

3.2 Опис архітектури застосунку

Як було описано раніше, головна робота даного застосунку відбувається в двох класах, а саме: transcribe і main. Розглянемо детально кожен із них. Візуальна UML-діаграма даного веб-застосунку наведена нижче, (рис. 13):

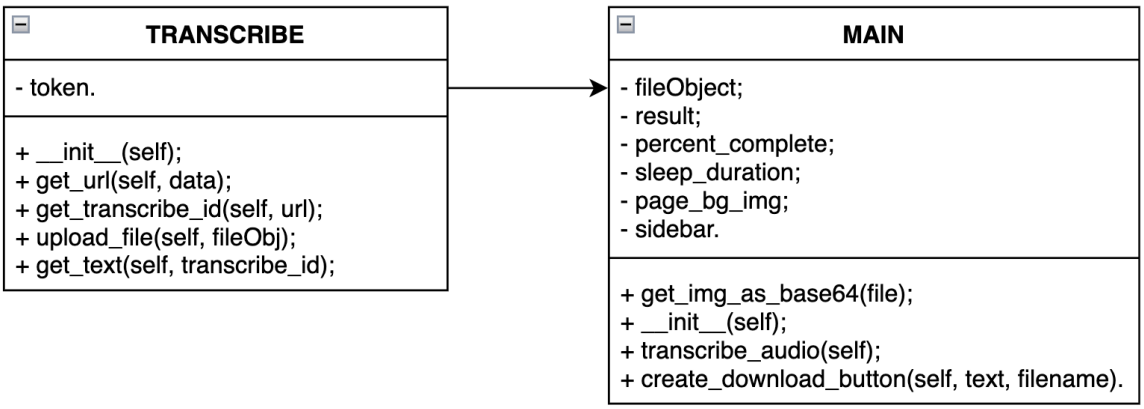

Рисунок 13 – UML-діаграма класів

КЛАС TRANSCRIBE:

Клас transcribe – це клас у якому зберігаються допоміжні функції і код який буде взаємодіяти із AssemblyAI API.

Поля:

- token – це поле, яке відповідає за зберігання ключа AssemblyAI. У кожного користувача він особистий.

Методи:

- __init__(self) – це метод який дозволяє класу ініціалізувати об’єкти;
- get_url(self, data) – це метод, який приймає наступні параметри: API ключ і об’єкт аудіо файлу який буде завантажено. Здійснюється POST запит до попередньо розглянутої кінцевої точки завантаження AssemblyAI Upload API і включаємо маркер API та об’єкт файлу як частину тіла запиту. Об’єкт відповіді містить URL-адресу завантаженого файлу. Ця URL-адреса повертається функцією;
- get_transcribe_id(self, url) – це метод, який приймає наступні параметри: API ключ і URL аудіо файлу з попередньої функції. Здійснюється POST запит до кінцевої точки транскрипції в AssemblyAI API. Якщо аудіо файл зараз не транскрибується, то новий файл негайно обробляється. Якщо все ж таки транскрипція виконується, новий аудіо файл ставиться в чергу до завершення попереднього завдання. Аби виконувати кілька завдань одночасно, потрібно перейти на преміум-план:
    - Об’єкт міститиме ідентифікатор транскрипції. Цей ідентифікатор разом із окремою кінцевою точкою буде використано для отримання статусу транскрипції;
    - Згодом функція поверне цей ідентифікатор.
- get_text(self, transcribe_id) – це метод, який просто буде отримувати поточний статус транскрипції. В ньому буде використовуватись цикл while для безперервного надсилання запитів до кінцевої точки. Під час кожної ітерації циклу ми перевірятимемо статус транскрипції. Цикл продовжуватиме працювати, доки статус не буде завершено;
- upload_file(self, fileObj) – це метод, який буде пов’язаний з кнопкою «Upload» в інтерфейсі користувача Streamlit. Функція

має лише один параметр: об’єкт файлу. Даний метод зробить наступне:

- Завантажить маркер API з файлу .env;
- Використовуватиме маркер для виклику раніше визначених функцій;
- Поверне ID транскрипції.

КЛАС MAIN:

Клас main – це клас у якому міститиметься весь код, пов’язаний із інтерфейсом користувача Streamlit.

Поля:

- fileObject – це поле, яке містить об’єкт файлу, завантажений користувачем;
- result – це поле, яке містить результат транскрипції, повернутий сервером AssemblyAI;
- percent_complete – це поле, яке містить відсоток завершеного процесу транскрипції;
- sleep_duration – це поле, яке містить тривалість у секундах до сну між перевіркою статусу транскрипції;
- page_bg_img – це поле, яке містить рядок HTML для встановлення фонового зображення веб-сторінки;
- sidebar – це поле, яке містить об’єкт бічної панелі в інтерфейсі користувача.

Методи:

- get_img_as_base64(file) – це метод, який є визначений для читання файлу зображення та перетворення його на рядок у кодуванні base64, який використовується для встановлення фонового зображення у програмі Streamlit;
- __init__(self) – це метод, який дозволяє класу ініціалізувати об’єкти;

- transcribe_audio(self) – це метод, який коли викликається пропонує користувачеві завантажити файл і створює екземпляр Transcribe, далі він:
    - Викликає метод upload_file Transcribe, щоб завантажити файл і отримати тимчасову URL-адресу завантаженого файлу;
    - Викликає метод get_transcribe_id, щоб отримати ідентифікатор транскрипції.
    - Відображає індикатор виконання та відтворює оригінальний аудіофайл, очікуючи на обробку транскрипції.
    - Після завершення транскрипції метод відображає транскрибований текст за допомогою методу get_text Transcribe.
    - Нарешті створюється екземпляр класу AudioTranscriber і викликається його метод transcribe_audio, який запускає програму Streamlit для транскрипції аудіо.
- create_download_button(self, text, filename) – це метод, який приймає транскрибований текст і бажану назву файлу як параметри та повертає рядок HTML для кнопки завантаження txt файлу із транскрибованим текстом.

Розглянемо візуально схему структури програмного застосунку на рисунку наведеному нижче, (рис. 14):

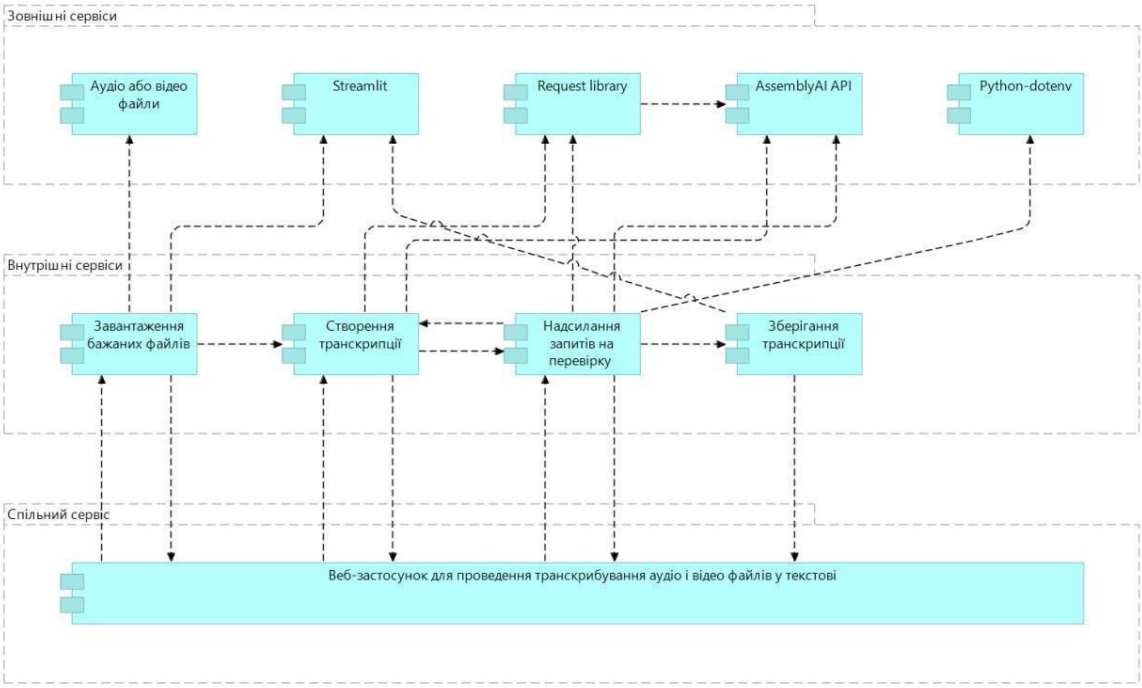

Рисунок 14 – Схема структури програмного застосунку

Розглянемо схему працездатності застосунку, (рис. 15):

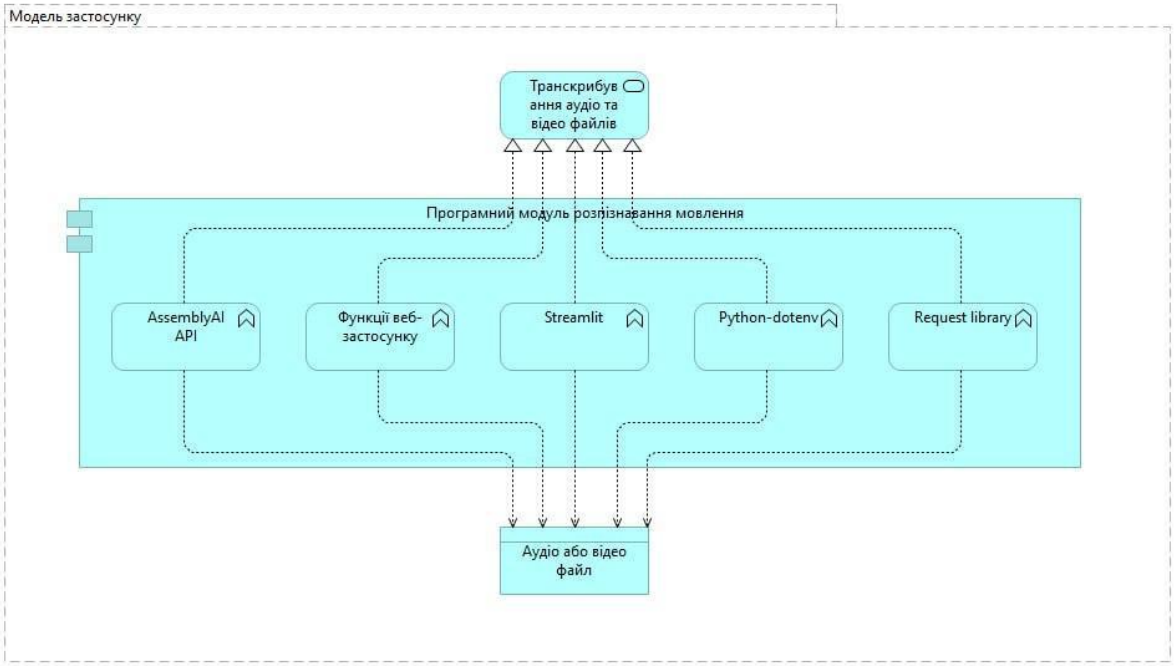

Рисунок 15 – Схема роботи застосунку

3.3 Розробка і реалізація інтерфейсу користувача

Для користувача робота майбутнього застосунку буде відбуватись на сторінці веб-застосунку. Його побудова буде відбуватись за допомогою Streamlit UI у класі «main» який буде відігравати роль точки входу для майбутнього застосунку. Більш детальніше про увесь функціонал буде розписано у подальшому підрозділі.

Під час роботи над першим розділом було створено макет веб-застосунку. У даному розділі його було реалізовано. Розглянемо його роботу, основні функції які він може виконувати і модифікації які були реалізовані у остаточній версії застосунку.

При першому запуску застосунку у користувача з’являється головне вікно у якому міститься поле у яке він може завантажити обраний аудіо файл для проведення транскрипції. У лівому верхньому куті застосунку була реалізована і розміщена іконка меню. Натиснувши на неї у користувача відкривається бокова панель у якій існує можливість прослухати аудіо або відео файл який було вибрано ним для проведення транскрибування. Також у ній наведено перелік обмежень які користувач повинен буде враховувати сам перед завантаженням бажаного аудіо або відео файлу, (рис. 16):

Рисунок 16 – Головна сторінка застосунку

Тепер, розглянемо дії веб-застосунку коли користувач завантажить в нього бажаний аудіо або відео файл на транскрибування.

При завантаженні обраного файлу його буде відображено на сторінці веб-застосунку і, також, буде відбуватись відображення індикатора виконання транскрибування. По закінченню проведення транскрибування у користувача буде видаватись текст промовлених слів транскрибованого аудіо або відео файлу, (рис. 17):

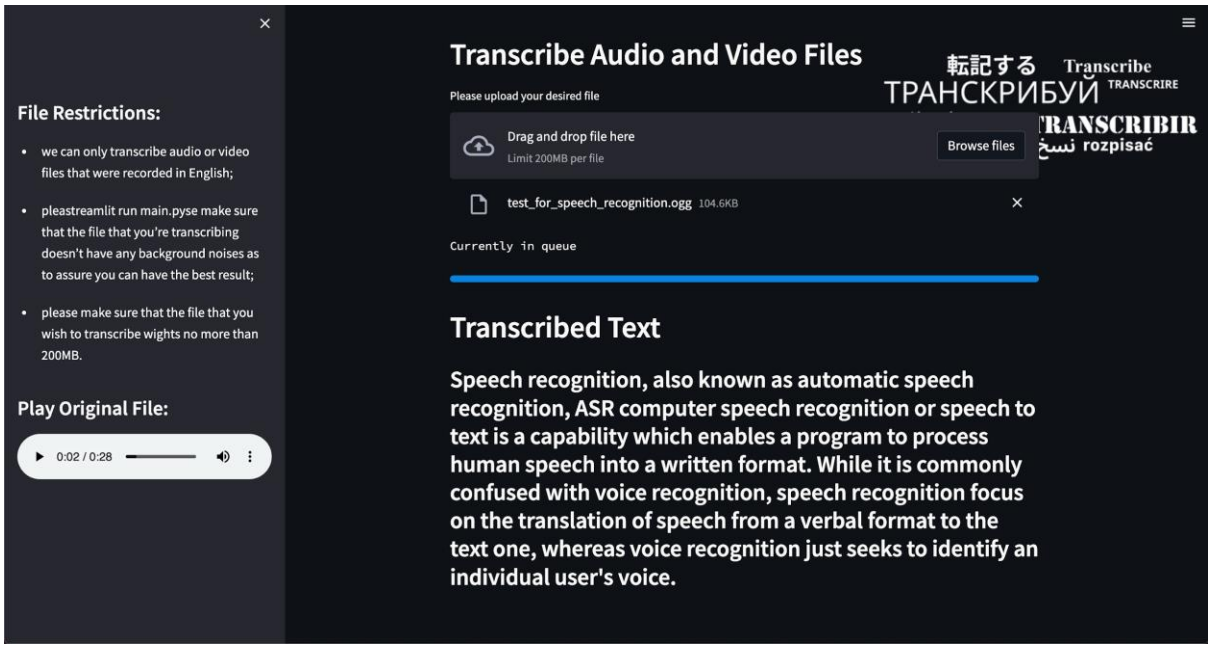

Рисунок 17 – Завершення транскрипції

Користувач зможе побачити результат транскрипції на сторінці застосунку і завантажити txt файл на свій локальний комп’ютер натиснувши кнопку «Download Transcribed Text», (рис. 18).

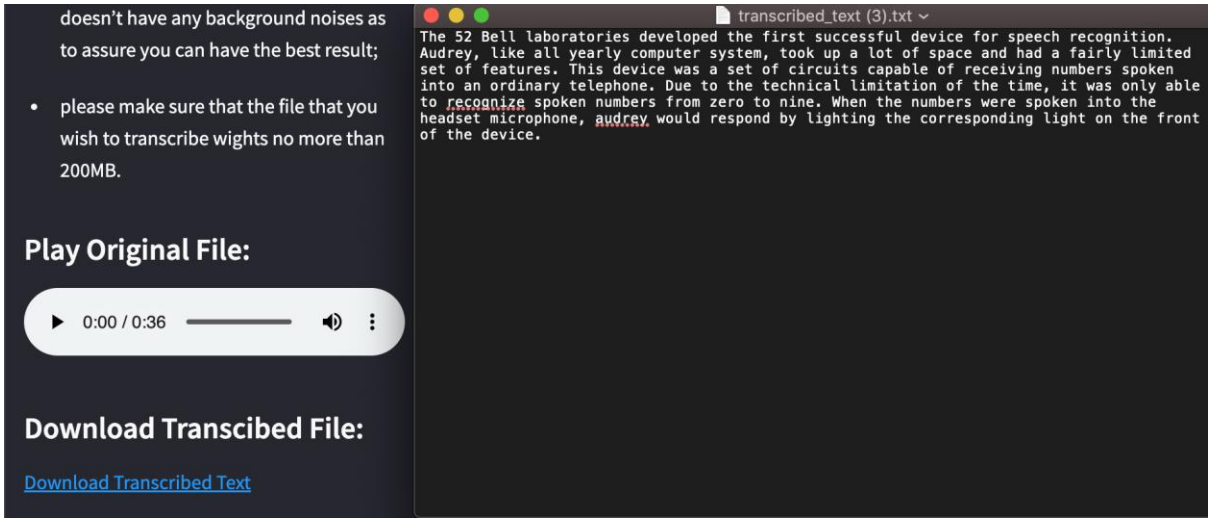

Рисунок 18 – Кнопка і результат її виконання

3.4 Опис інструктивних матеріалів користувача

Під час роботи із даним веб-застосунком користувач буде повинен враховувати певні вимоги при роботі із програмним модулем. Спочатку розглянемо обмеження до застосунку.

До обмежень, як було розписано попередньо, будуть входити:

- Обмеження стосовно мови. Застосунок зможе працювати лише з аудіо або відео файлами які були записані англійською мовою;
- Обмеження стосовно якості. Користувач повинен буде враховувати, що при наявності фонового шуму якість файлу може бути погіршена;
- Обмеження стосовно розміру. Програма зможе працювати лише з файлами які не перевищують розмір в 200MB.

Їх користувач повинен буде враховувати сам проте, він буде попередньо попереджений на головні сторінці веб-застосунку у боковій панелі.

Також, у користувача будуть існувати вимоги щодо самого запуску застосунку. Оскільки даний веб-застосунок у даній стадії його розробки потребує відкриття його, як запуску локально із Visual Studio Code, або

іншого бажаного застосунку, користувач повинен буде впевнитись у наступному:

- Користувач має бути зареєстрованим до AssemblyAI аби отримати API ключ, який у кожного із ним є особливим;
- Мати встановленим у своєму робочому середовищі такі бібліотеки як:
  - AssemblyAI;
  - Streamlit;
  - Requests;
  - Python-dotenv;
  - Base64.

Включаючи всі вищенаведені вимоги у користувача не повинні будуть виникати ніякі проблеми у роботі із застосунком.

3.5 Опис тест-кейсів для перевірки працездатності застосунку

Для перевірки працездатності і коректності роботи даного програмного застосунку для транскрибування аудіо файлів було здійснено наступні тести-кейси:

Таблиця 2. Таблиця тест-кейсів для перевірки працездатності застосунку

Тест-кейс	Кроки	Результати
1	Відкрити застосунок.	Застосунок відкрито.
	НЕ вибрати бажаний аудіо або відео файл записаний на англійській мові і розміром не більше 200MB.	Файл НЕ вибрано.

	Закрити застосунок.	Застосунок закритий.
2	Відкрити застосунок.	Застосунок відкрито.
	Вибрати бажаний аудіо або відео файл записаний на англійській мові і розміром не більше 200MB.	Файл вибрано.
	Закрити застосунок.	Застосунок закритий.
3	Відкрити застосунок.	Застосунок відкрито.
	Вибрати бажаний аудіо або відео файл записаний на англійській мові і розміром не більше 200MB.	Файл вибрано.
	Запустити застосунок.	Застосунок спрацював.
	Перевірити результат роботи застосунку. Результат відображено.	Результат задовільний.
	Закрити застосунок.	Застосунок закритий.
4	Відкрити застосунок.	Застосунок відкрито.

	Вибрати бажаний аудіо або відео файл записаний на англійській мові і розміром не більше 200MB.	Файл вибрано.
	Запустити застосунок.	Застосунок спрацював.
	Перевірити результат роботи застосунку. Результат відображено.	Результат задовільний.
	Розпочати перевірку нового аудіо або відео файлу записаного на англійській мові і розміром не більше 200MB.	Нову перевірку перервано.
	Запустити застосунок.	Застосунок НЕ спрацював. Викинуло повідомлення про помилку.
	Закрити застосунок.	Застосунок закритий.
5	Відкрити застосунок.	Застосунок відкрито.
	Вибрати бажаний аудіо або відео файл записаний НЕ на англійській мові і	Файл вибрано.

	розміром не більше 200MB.	
	Запустити застосунок.	Застосунок спрацював.
	Перевірити результат роботи застосунку.	Результат НЕ відображено.
	Закрити застосунок.	Застосунок закритий.
6	Відкрити застосунок.	Застосунок відкрито.
	Вибрати бажаний аудіо або відео файл записаний на англійській мові і розміром БІЛЬШЕ 200MB.	Файл вибрано.
	Запустити застосунок.	Застосунок НЕ спрацював. Викинуло повідомлення про помилку.
	Закрити застосунок.	Застосунок закритий.
7	Відкрити застосунок.	Застосунок відкрито.
	Вибрати НЕ аудіо або відео файл.	Файл вибрано.
	Запустити застосунок.	Застосунок НЕ спрацював. Викинуло повідомлення про помилку.

	Закрити застосунок.	Застосунок закритий.
8	Відкрити застосунок.	Застосунок відкрито.
	Вибрати бажаний аудіо або відео файл записаний на англійській мові, розміром не більше 200МВ і з ПОГАНОЮ ЯКІСТЮ.	Файл вибрано.
	Запустити застосунок.	Застосунок спрацював.
	Перевірити результат роботи застосунку.	Результат відображено. Результат НЕ задовільний.
	Закрити застосунок.	Застосунок закритий.

Тепер, розглянемо більш детальніше третій тест-кейс:

Спочатку відкриємо застосунок в якому користувач буде здійснювати транскрипцію бажаного аудіо файлу, (рис. 19):

Рисунок 19 – Відкриття застосунку

Вибираємо бажаний аудіо файл записаний на англійській мові і розміром не більше 200MB. Вміст даного файлу буде включати в себе наступний текст:

«In 1952 Bell Laboratories developed the first successful device for speech recognition - "Audrey". Like all early computers, system took up a lot of space and had a fairly limited set of features. This device was a set of circuits capable of receiving numbers spoken into an ordinary telephone. Due to the technical limitations of the time, it was only able to recognize spoken numbers from "0" to "9". When numbers were spoken into the handset's microphone, "Audrey" would respond by lighting the corresponding light on the front of the device.»

Запустимо застосунок і через певний має вивестись повідомлення про проведення транскрипції разом із індикатором виконання, (рис. 20):

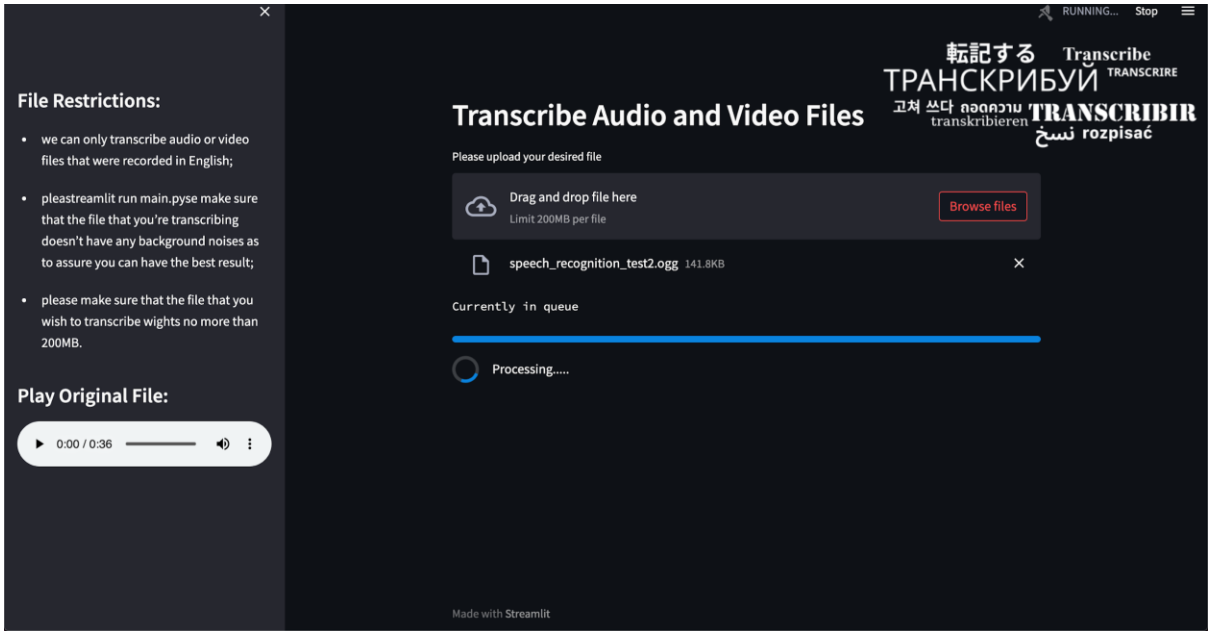

Рисунок 20 – Робота застосунку

Перевіримо результат роботи застосунку, який було відображено на його головній сторінці. Результат можна вважати задовільним, (рис. 21-22):

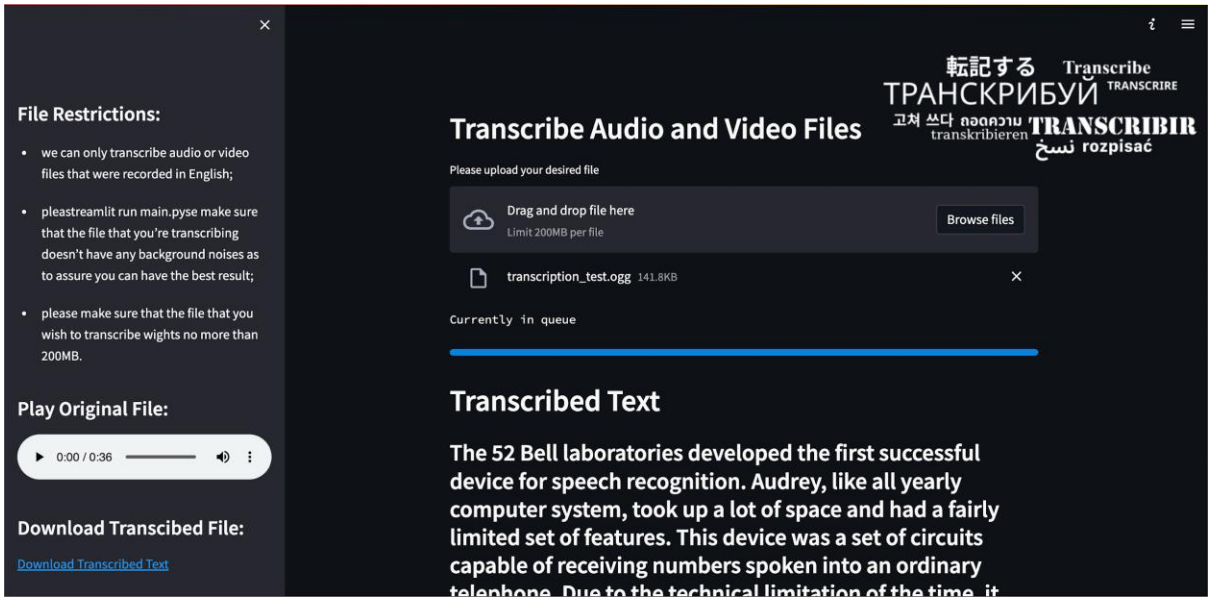

Рисунок 21 – Результат роботи

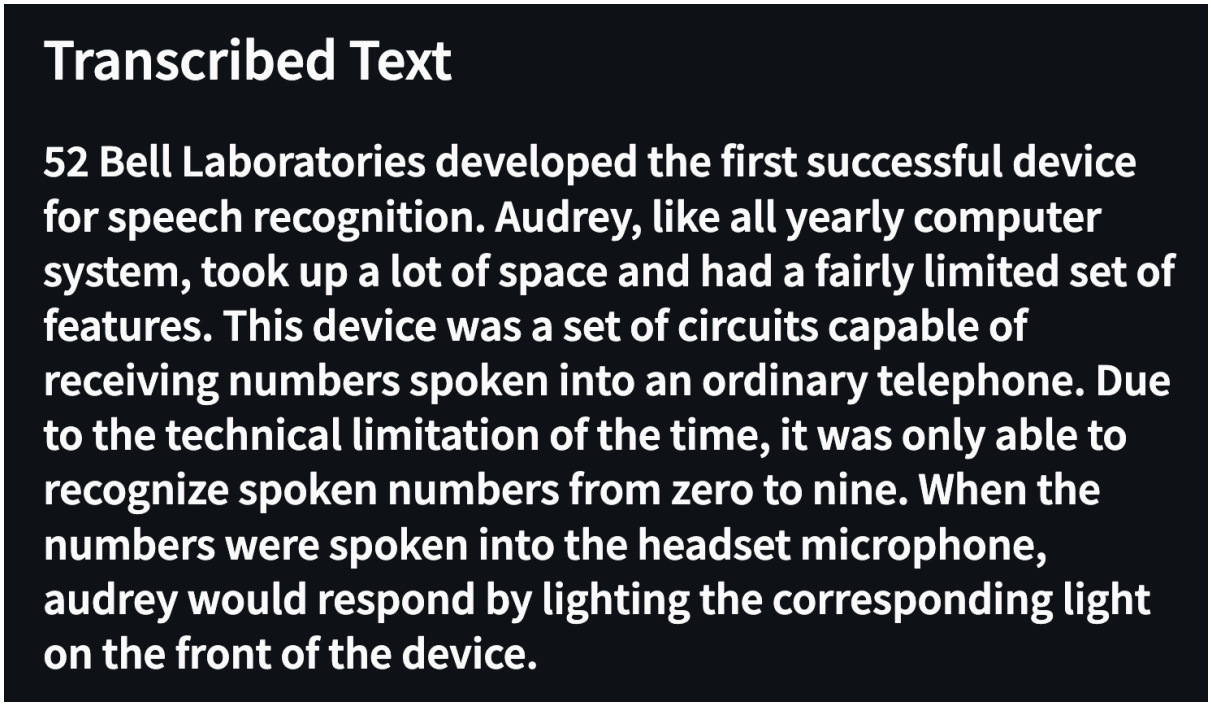

Рисунок 22 – Результат роботи

3.6 Опис та аналіз результатів тестових прикладів роботи застосунку

За допомогою тест-кейсів було проведена перевірка працездатності і коректності реалізації програмного застосунку для транскрибування аудіо файлів. Результати перевірки даних тестових прикладів можна побачити в (табл. 2).

Таблиця 3. Результати тестових прикладів

Мета тесту	Перевірити коректність роботи програмного застосунку
Початковий стан системи	Застосунок почав свою роботу в результаті чого було переведено транскрипцію обраного користувачем аудіо файлу
Вхідні дані	Аудіо файли
Схема проведення тесту	Користувач вибирає бажаний аудіо файл для транскрибування

Очікуваний результат	Застосунок виводить перероблений файл
Стан системи після проведення випробувань	Результат відображено

Висновки до третього розділу:

Були описані інструментальні засоби та середовище розробки, обгрунтовано їх вибір. Було наведено опис класів готової програми разом із полями і методами в ній. Також була проведена перевірка працездатності даного програмного застосунку.

ВИСНОВКИ

Задача транскрибування аудіо та відео файлів у текстовий формат є надзвичайно актуальним завданням у різних сферах, включаючи журналістику, право, медицину та освіту. Процес транскрипції передбачає прослуховування або перегляд відео чи аудіо файлу, а потім введення вимовлених слів у письмовий документ. Однією з технологій, яка надає можливість здійснювати даний процес є розпізнавання мовлення.

Головною темою даного дипломного проекту був програмний модуль автоматичного транскрибування мовлення, а саме: реалізація програмного веб-застосунку за допомогою якого можливе перетворення аудіо і відео файлів у текстові, використовуючи технології розпізнавання мовлення.

Після проведення аналізу предметної області і постановки задачі, яка буде розв’язуватись, було обрано основні інструментальні засоби, математичні моделі і реалізовано головні функції, які відповідали за транскрибування аудіо і відео файлів у текстові. За допомогою даних інструментів розроблений застосунок реалізує наступні функції використовуючи модуль розпізнавання мовлення:

- Завантаження файлу, який ми маємо локально, до AssemblyAI;
- Проведення транскрипції;
- Надсилання запитів до AssemblyAI для перевірки коли буде завершена транскрипція від UI;
- Отримання і зберігання отриманої транскрипції у веб-застосунку.

Були розроблені текстові приклади роботи із веб-застосунком і наведено вимоги користувачам для роботи із ним.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Gerardus Blokdyk. Speech Recognition A Complete Guide» – Emereo Publishing, 2020. – 304с.
2. Bernd T. Meyer. Speech Recognition by Man and Machine – KS OmniScriptum Publishing, 2010. – 140с.
3. Kevin Brennan. A Guide to the Business Analysis Body pf Knowledge (BABOK Guide) – IIBA, 2009. – 272 с.
4. Antonio J. Rubio Ayuso. Speech Recognition and Coding: New Advances and Trends / Juan M. Lopez Soler – Springer Berlin Heidelberg, 2012. – 505 с.
5. Thomas Hathaway. Functional and Non-Functional Requirements – Simply Put! / Angela Hathaway – CreateSpace Publishing, 2016. – 70 с.
6. Alexander Waibel. Readings in Speech Recognition / Kai-Fu Lee – Elsevier Science, 1990. – 680 с.
7. Automatic Speech Recognition Using Limited Vocabulary: A Survey / Jean Louis K. E Fendji, Diane C. M. Tala, Blaise O. Yenke, Marcellin Atemkeng Applied artificial intelligence. 2022. Том 36б No 1. с. 2–36.
8. Li Deng Deep learning: from speech recognition to language and multimodal processing. 2016. Vol 5. с. 1-15.
9. Luciano Romalho. Fluent Phyton; Clear, Concise, and Effective Programming – O'Reilly Media, 2022. – 1012 с.
10. Mark Liu. Make Python Talk: Build Apps with Voice Control and Speech Recognition – No Starch Press, 2021. – 384 с.
11. Yu D., Deng L. Deep Neural Networks. Automatic Speech Recognition. London, 2014. – с. 57–77.
12. Lee K.-F. Automatic Speech Recognition. Boston, MA : Springer US, 1989.

13. Automatic Speech Analysis and Recognition / ed. by J.-P. Haton. Dordrecht : Springer Netherlands, 1982.
14. Milosevic N. Introduction to Convolutional Neural Networks. Berkeley, CA : Apress, 2020.
15. Habibi Aghdam H., Jahani Heravi E. Guide to Convolutional Neural Networks. Cham : Springer International Publishing, 2017.
16. Richards T. Getting Started with Streamlit for Data Science: Create Streamlit Applications from Scratch. Packt Publishing, Limited, 2021. – 282 c.
17. Zadka M. Requests. DevOps in Python. Berkeley, CA, 2019. – c. 85–94.
18. O'Neil T. Beginners Guide to Figma: The Step by Step Figma Manual with Illustrations and Tips. Independently Published, 2022.
19. Clow M. Visual Studio Code. Angular 5 Projects. Berkeley, CA, 2018. – c. 57–68.
20. Khorasani M., Abdou M., Hernández Fernández J. Streamlit Basics. Web Application Development with Streamlit. Berkeley, CA, 2022. – c. 31–62.

ДОДАТОК А. Кінцевий лістинг програми

```
import os
from dotenv import load_dotenv
import requests
class Transcribe:
    def __init__(self):
        load_dotenv()
        self.token = os.getenv("API_TOKEN")

    def get_url(self, data):
        headers = {'authorization': self.token}
        response = requests.post('https://api.assemblyai.com/v2/upload',
                                 headers=headers,
                                 data=data)
        url = response.json()["upload_url"]
        print("Uploaded File and got temporary URL to file")
        return url

    def get_transcribe_id(self, url):
        endpoint = "https://api.assemblyai.com/v2/transcript"
        json = {
            "audio_url": url
        }
        headers = {
            "authorization": self.token,
            "content-type": "application/json"
        }
        response = requests.post(endpoint, json=json, headers=headers)
        transcribe_id = response.json()['id']
        print("Made request and file is currently queued")
        return transcribe_id

    def upload_file(self, fileObj):
        file_url = self.get_url(fileObj)
        transcribe_id = self.get_transcribe_id(file_url)
        return self.token, transcribe_id

    def get_text(self, transcribe_id):
        endpoint = f"https://api.assemblyai.com/v2/transcript/{transcribe_id}"
        headers = {
            "authorization": self.token
        }
        result = requests.get(endpoint, headers=headers).json()
        return result


            import streamlit as st

            from transcribe import Transcribe

            import time

            import base64


            @st.cache_data

            def get_img_as_base64(file):
```

```
    with open(file, "rb") as f:
        data = f.read()
        return base64.b64encode(data).decode()

img = get_img_as_base64("transcribe.png")
class Main:
    def __init__(self):
        self.fileObject = None
        self.result = {}
        self.percent_complete = 0
        self.sleep_duration = 1

        self.sidebar = st.sidebar

        self.sidebar.title("File Restrictions:")
        self.sidebar.markdown("- we can only transcribe audio or video files that were recorded in English;")
        self.sidebar.markdown("- pleastreamlit run main.pyse make sure that the file that you're transcribing doesn't have any background noises as to assure you can have the best result;")
        self.sidebar.markdown("- please make sure that the file that you wish to transcribe wights no more than 200MB.")

        page_bg_img = f"'
        <style>
        [data-testid="stAppViewContainer"] {{
            background-image: url("{img}");
            background-size: cover;
        }}
        </style>
        "'
        st.markdown(page_bg_img, unsafe_allow_html=True)


    def transcribe_audio(self):
        st.header("Transcribe Audio and Video Files")

        self.fileObject = st.file_uploader(label="Please upload your desired file")
        if self.fileObject:
            transcriber = Transcribe()
```

```
        token, t_id = transcriber.upload_file(self.fileObject)

        self.result = {}
        self.percent_complete = 0

        st.text("Currently in queue")
        self.progress_bar = st.progress(self.percent_complete)

        self.sidebar.title("Play Original File:")
        self.sidebar.audio(self.fileObject)

        while self.result.get("status") != "processing":
            self.percent_complete += self.sleep_duration
            time.sleep(self.sleep_duration)
            self.progress_bar.progress(self.percent_complete / 10)
            self.result = transcriber.get_text(t_id)

        self.sleep_duration = 0.01

        for percent in range(self.percent_complete, 101):
            time.sleep(self.sleep_duration)
            self.progress_bar.progress(percent)

        with st.spinner("Processing....."):
            while self.result.get("status") != 'completed':
                self.result = transcriber.get_text(t_id)

        st.balloons()
        st.header("Transcribed Text")
        st.subheader(self.result['text'])

        self.sidebar.title("Download Transcribed File:")
        download_text = self.result['text']
        download_filename = "transcribed_text.txt"
        download_button = self.create_download_button(download_text, download_filename)
        self.sidebar.markdown(download_button, unsafe_allow_html=True)

    def create_download_button(self, text, filename):
```

```
            b64_text = base64.b64encode(text.encode()).decode()
            button = f'<a href="data:text/plain;base64,{b64_text}" download="{filename}">Download Transcribed
Text</a>'
            return button

        audio_transcriber = Main()
        audio_transcriber.transcribe_audio()
```