НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„Київський політехнічний інститут імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Євгенія СУЛЕМА

“___” _____ 2024 р.

ПЕРСОНАЛЬНИЙ АСИСТЕНТ ДЛЯ ПІДБОРУ ФІЛЬМІВ З ВИКОРИСТАННЯМ МУЛЬТИМЕДІЙНОГО ПОМІЧНИКА ТА ГОЛОСОВОГО АСИСТЕНТА. РОЗРОБЛЕННЯ КОМПОНЕНТІВ ГОЛОСОВОГО ІНТЕРФЕЙСУ

Пояснювальна записка

Виконавець:

_____ Максим ПАВЛЮШИН

2024

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

REST (скор. від англ. Representational State Transfer) – стиль архітектури для розроблення вебсервісів, що дозволяє передавати дані між клієнтом і сервером.

Google API – набір інструментів і служб від Google, які дозволяють використовувати їхні функціональні можливості, наприклад, розпізнавання мовлення.

ПЗ – програмне забезпечення, що складається з сукупності програм для виконання комп’ютером.

Машинне навчання – алгоритми із галузі штучного інтелекту, для яких характерне не пряме рішення поставленої задачі, а прогнозування результату після проходження процесу навчання на зібраних даних, використовуючи при цьому математичну статистику, теорію ймовірностей тощо.

API (скор. від англ. application programming interface) – прикладний програмний інтерфейс, який надає набір чітко визначних правил для взаємодії одних програм з іншими.

3D (скор. від англ. 3 Dimensions) – розділ комп'ютерної графіки, сукупність прийомів та інструментів (як програмних, так і апаратних), призначених для зображення об'ємних об'єктів.

STT (скор. від англ. Speech-to-Text) – технологія перетворення голосу в текстову інформацію.

TTS (скор. від англ. Text-to-Speech) – технологія перетворення текстової інформації в аудіо файл, в якому озвучується людським голосом надані слова у вигляді тексту.

ВСТУП

На сьогоднішній день існує достатньо програмного забезпечення, яке застосовується в різних сферах життя. Очевидним є той факт, що розробники та дизайнери прагнуть створити такий застосунок, який би був найбільш зручним для використання кінцевими користувачами, тому що в світі загалом існує досить багато програмного забезпечення майже для всіх можливих задач, але користуються попитом лише ті програми, які виконують свої функціональні можливості за мінімальну кількість взаємодій, і причому коли ці взаємодії є інтуїтивними та зрозумілими. Тому в останні роки можна побачити стрімкий розвиток голосових асистентів, за допомогою яких можна швидко виконати будь-які дії з однієї «точки входу».

Даний проєкт присвячений розробленню персонального асистента для підбору фільмів, який об’єднує в собі мультимедійного помічника, функції голосового управління та систему персоналізованих рекомендацій. Однією з ключових складових цієї системи є модуль голосового інтерфейсу, який відповідає за розпізнавання голосових запитів, їх обробку та озвучування відповідей. Голосовий інтерфейс забезпечує користувачам можливість швидкого і зручного доступу до функцій пошуку та рекомендації фільмів, роблячи процес використання системи простішим та більш інтерактивним.

Метою даної роботи є розроблення ефективного та надійного модуля голосового інтерфейсу, що включає реалізацію механізмів розпізнавання мовлення, обробки текстових команд та синтезу мови. Основні задачі, які вирішуються в рамках цього завдання:

- вибір відповідних технологій та бібліотек для реалізації голосового інтерфейсу;
- створення алгоритмів обробки голосових запитів;

- забезпечення інтеграції голосового інтерфейсу з іншими компонентами системи.

Розроблений модуль голосового інтерфейсу має забезпечити високу точність розпізнавання голосу, швидкість роботи та комфортність взаємодії для кінцевого користувача.

1.ТЕХНОЛОГІЇ ДЛЯ РЕАЛІЗАЦІЇ ГОЛОСОВОГО ІНТЕРФЕЙСУ

1.1 . Аналіз Google API для розпізнавання мовлення

У розробці персонального асистента для підбору фільмів, голосовий інтерфейс відіграє ключову роль у забезпеченні інтерактивної взаємодії між користувачем і системою. Основним завданням є перетворення голосових команд у текст для їхньої подальшої обробки. Це потребує використання високоточних технологій розпізнавання мовлення, які здатні працювати з різними мовами, акцентами та умовами навколишнього середовища.

Google API (Google Cloud Speech-to-Text) є оптимальним вибором для реалізації цих завдань через такі причини:

- Висока точність розпізнавання мовлення: Google API забезпечує одну з найкращих точностей серед сервісів розпізнавання мовлення завдяки використанню сучасних моделей машинного навчання. Система підтримує контекстне розуміння: може коректно розпізнавати іменники власні, назви фільмів та жанрів.
- Підтримка багатьох мов: Google API розпізнає понад 120 мов, включаючи українську. Це дозволяє створити асистента, доступного для широкої аудиторії, зокрема україномовних користувачів.
- Обробка в реальному часі: API дозволяє розпізнавати мовлення в режимі реального часу, що критично для інтерактивних систем, а мінімальна затримка під час обробки команд забезпечує плавну взаємодію користувача з системою.
- Можливість налаштування специфічного словника: Google API дозволяє додавати спеціалізовані слова та фрази, наприклад назви фільмів, імена акторів, специфічні терміни. Це підвищує точність розпізнавання у вузькоспеціалізованих завданнях.

Технічні характеристики GoogleAPI:

- Алгоритм обробки: Google використовує моделі на базі нейронних мереж, зокрема трансформери.
- Параметри налаштування:
  a) Режим потокового розпізнавання (Streaming Recognition) для роботи в реальному часі.
  b) Режим пакетного розпізнавання (Batch Recognition) для завантаження записів і їхньої обробки.
- Обмеження:
  a) Довжина однієї сесії потокового розпізнавання — до 5 хвилин.
  b) Необхідність стабільного інтернет-з’єднання для хмарної обробки.

Таблиця 1.1

Порівняння існуючих програмних рішень

Параметр	**Google API**	**Microsoft Azure Speech**	**IBM Watson Speech-to-Text**
Точність розпізнавання	Висока	Висока	Середня
Підтримка мов	120	80	50
Доступність API	Доступна та легка в налаштуванні	Доступна та легка в налаштуванні	Складна у налаштуванні та інтегруванні
Вартість	$0.25 за годину	$0.5 за годину	$1.2 за годину

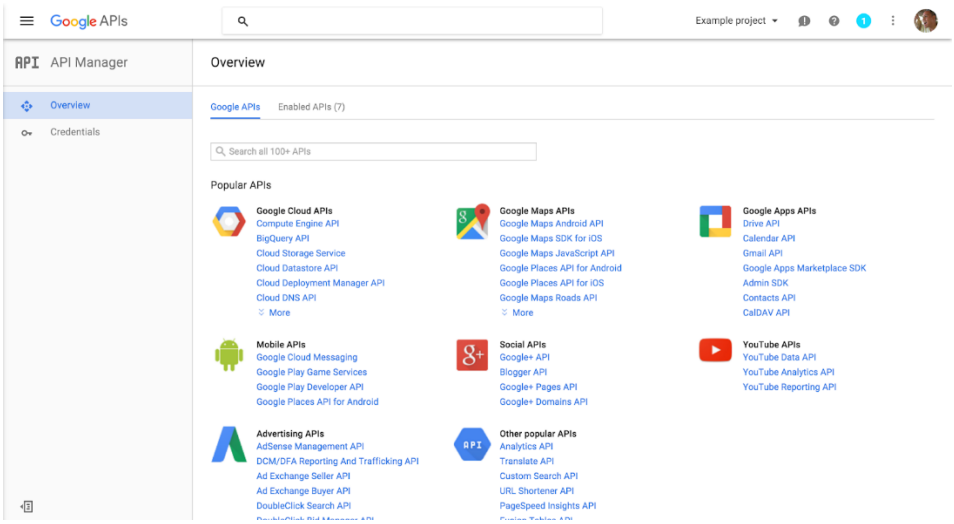

Рис. 1.1. Фрагмент отримання GoogleAPI

1.2 . Аналіз Llama для синтезу мовлення

Синтез мовлення (Text-to-Speech, TTS) є критично важливим компонентом голосового інтерфейсу, який дозволяє системі озвучувати відповіді користувачеві. У рамках проєкту персонального асистента для підбору фільмів цей функціонал забезпечує інтерактивність і зручність використання системи, роблячи взаємодію більш природною. Llama була обрана для реалізації цього компонента завдяки її сучасному підходу до обробки тексту та високій якості синтезованого мовлення.

Переваги використання Llama для синтезу мовлення

- Висока якість озвучення
    a) Llama використовує сучасні трансформерні моделі, які забезпечують природність і плавність мовлення. Це дозволяє створити голосовий інтерфейс, який звучить максимально "людяно" без механічних інтонацій.
- Гнучкість налаштувань

a) Система дозволяє змінювати швидкість, тональність та емоційне забарвлення голосу, що важливо для створення мультимедійного помічника, здатного виражати різні емоції.

- Підтримка багатомовності
    a) Llama підтримує синтез мовлення для багатьох мов, включаючи українську. Це робить її придатною для локалізованого голосового асистента.
- Ефективність обробки
    a) Завдяки оптимізованим моделям Llama забезпечує швидкий час генерації голосового виходу навіть при обробці складних текстів.
- Інтеграція із сучасними технологіями
    a) Llama легко інтегрується з інфраструктурою клієнт-серверної архітектури проєкту. Її API підтримує різні формати запитів, що спрощує взаємодію із серверною частиною, написаною на Node.js.
- Машинне навчання для покращення якості
    a) Llama дозволяє додатково тренувати моделі на специфічних даних (наприклад, текстах, пов'язаних із фільмами). Це забезпечує кращу адаптацію системи до конкретного домену.

Технічні характеристики Llama

- Модель трансформера
    a) Llama використовує GPT-подібну архітектуру, яка оптимізована для обробки тексту. Завдяки цьому, вона забезпечує як точний синтез, так і гнучкість у використанні.
- Можливості кастомізації

a) Інструмент дозволяє змінювати характеристики голосу (наприклад, робити його більш емоційним, офіційним чи неформальним).

- Режими роботи
  a) Режим пакетної обробки (Batch Mode): дозволяє обробляти текст і створювати звукові файли для подальшого використання.
  b) Режим потокової обробки (Streaming Mode): генерує мовлення у реальному часі, що ідеально підходить для інтерактивних систем.
- Мінімальні системні вимоги
  a) Llama оптимізована для роботи у хмарних середовищах і на локальних серверах із сучасними GPU.

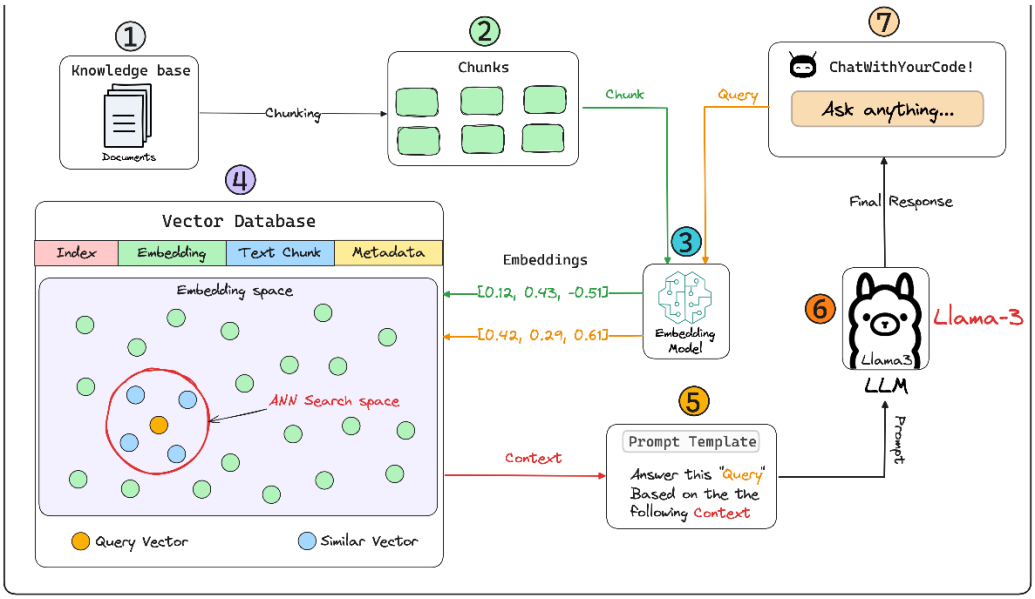

Рис. 1.2. Приклад як працює модель Llama-3

- База даних (Knowledge base): Вихідні документи діляться на фрагменти (chunks).

- Chunking: Текст розбивається на невеликі частини для ефективної обробки.
- Embedding Model: Кожен фрагмент тексту перетворюється на векторне представлення (embedding) за допомогою моделі векторизації.
- Vector Database: Вектори фрагментів зберігаються у векторній базі даних. Під час запиту система знаходить схожі вектори (за допомогою пошуку найближчих сусідів — ANN Search).
- Prompt Template: Найбільш релевантний фрагмент (context) використовується для створення підказки (prompt), яка містить запит (query) і контекст.
- Llama-3: Модель Llama-3 обробляє запит разом із контекстом та генерує остаточну відповідь.
- Відповідь користувачу: Згенерована відповідь повертається користувачу у зручному форматі.

Таким чином, Llama-3 працює як генеративна мовна модель, що використовує релевантний контекст із бази знань для створення точних і осмислених відповідей на запитання користувача.

Приклад сценарію використання Llama у проєкті

1. Користувач вимовляє запит (наприклад, "Порекомендуй фільм жахів").
2. Система розпізнає запит за допомогою Google API та передає текстовий результат до логіки системи.
3. Відповідь (наприклад, "Рекомендую подивитися 'Сяйво'") передається у вигляді тексту до Llama.

4. Llama генерує синтезовану відповідь із врахуванням інтонації, яка відповідає дружньому стилю мультимедійного помічника.

2. РЕАЛІЗАЦІЯ ГОЛОСОВОГО ІНТЕРФЕЙСУ

2.1 Алгоритм обробки голосових запитів

У цьому розділі розглядається алгоритм обробки голосових запитів, який поєднує сучасні технології обробки природної мови та мовлення. Процес включає перетворення голосового запиту у текст, пошук відповідної інформації у базі знань та генерацію текстової та аудіо-відповіді. Ключовими компонентами є векторна база даних, моделі для генерації тексту (LLM) та синтезу мовлення. Перед тим як детально розглянути алгоритм, слід розібратися з поняттям бази знань.

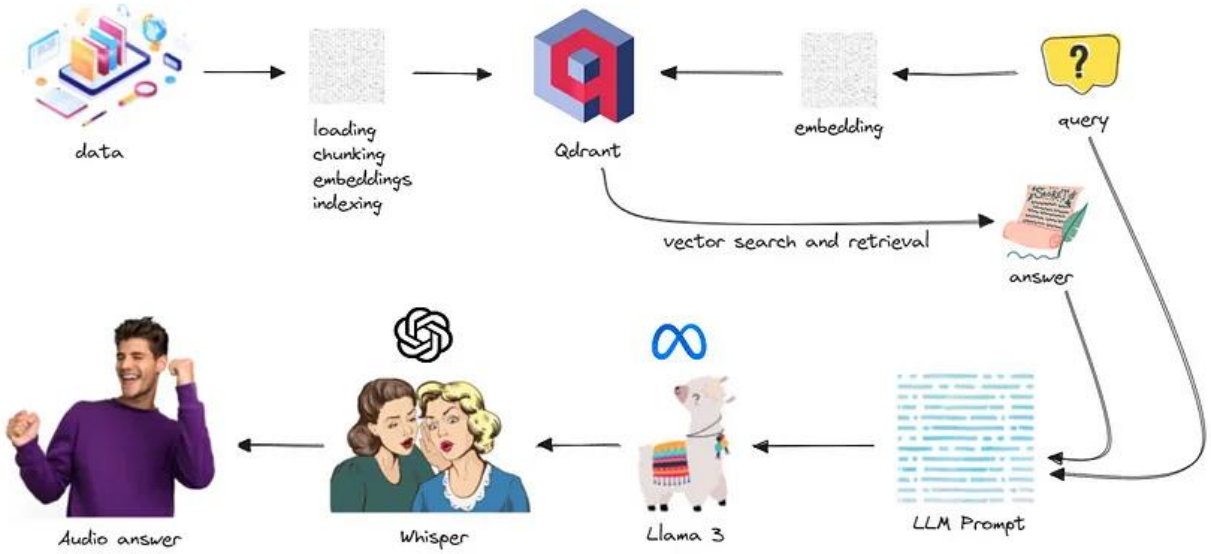

Рис. 2.1. Алгоритм обробки голосових запитів

Процес обробки голосових запитів є багатокроковим і включає інтеграцію декількох сучасних технологій. Він починається з отримання голосового запиту від користувача, його перетворення у текст, пошуку відповідної інформації у базі знань та завершується генерацією відповіді у текстовому та аудіоформаті.

На першому етапі система отримує голосовий запит від користувача. Для цього використовується технологія розпізнавання мовлення, зокрема Google API, яка конвертує аудіосигнал у текстовий формат. Текстовий запит стає основою для подальшої обробки.

Наступний крок — обробка тексту і перетворення його у векторну форму. Це відбувається за допомогою моделі для векторизації тексту, яка створює числове представлення (вектори) вихідного запиту. Ці вектори порівнюються з векторами, які вже збережені у векторній базі даних, наприклад, Qdrant. У базі знань інформація заздалегідь оброблена, розбита на фрагменти (чанкована), векторизована та проіндексована для зручного пошуку. Таким чином, коли система отримує запит, вона може ефективно виконати пошук найбільш схожих фрагментів на основі векторного представлення.

Після виявлення релевантної інформації відповідні фрагменти витягуються із векторної бази даних і передаються до моделі Llama 3. Llama 3 є великою мовною моделлю (LLM), яка відповідає за розуміння контексту запиту і на основі наданої інформації генерує відповідь. Llama 3 аналізує отримані дані, зіставляє їх з контекстом запиту та формує текстову відповідь, яка є логічно вивіреною та максимально точною.

Згенерований текстовий результат передається на наступний етап — синтез мовлення. Для цього використовується система Whisper, яка конвертує текстову відповідь у аудіоформат. Результатом цього процесу є озвучена відповідь, що повертається користувачу у вигляді аудіофайлу.

Таким чином, алгоритм обробки голосових запитів поєднує розпізнавання мовлення, векторний пошук у базі знань, генерацію відповідей за допомогою LLM та синтез аудіо. Кожен компонент відіграє важливу роль у забезпеченні точності, швидкодії та зручності користування системою.

2.2 . Інтеграція голосового інтерфейсу із серверною логікою системи

У цьому розділі розглянемо процес інтеграції голосового інтерфейсу із серверною логікою, реалізованою за допомогою Node.js та фреймворку Express. Система забезпечує обробку голосових запитів користувача, їх розпізнавання, обробку на сервері, а також генерацію текстових та аудіовідповідей.

Серверна частина реалізована як REST API із наступними ключовими компонентами:

1. Node.js: забезпечує асинхронне виконання операцій та високу продуктивність.
2. Express: використовується як веб-фреймворк для створення ендпоінтів, обробки HTTP-запитів і відповіді на них.
3. Groq SDK: інтеграція із зовнішнім API, що відповідає за трансформацію голосових запитів у текст (через Whisper) та генерацію відповідей (через Llama 3).
4. Multer: обробка завантаження аудіофайлів на сервер.
5. Файлова система (fs): збереження та обробка аудіофайлів на сервері.

На початковому етапі створюється сервер за допомогою Express. Сервер запускається на порту 3000 та обробляє вхідні HTTP-запити. Для зручності додано CORS, що дозволяє клієнтам з інших доменів звертатись до сервера, а також статичну обробку файлів з папки "public". Для взаємодії з великими мовними моделями (LLM) створюється ендпоінт POST /ask, який приймає текстовий запит користувача, передає його у модель Llama 3 через Groq SDK і повертає сформовану відповідь. Функція getGroqChatCompletion відправляє користувацький запит у Llama 3 і отримує результат. Для обробки аудіофайлів створено ендпоінт POST /transcript, що використовує Multer для завантаження

аудіо на сервер. Завантажений файл зберігається у тимчасовій директорії, а після цього обробляється для транскрипції. Після транскрипції голосовий запит перетворюється на текст. Отриманий текст передається до ендпоінту /ask для обробки Llama 3. Відповідь від моделі знову може бути синтезована у голосовий формат за допомогою Whisper (TTS — текст у мовлення).

Схема роботи інтеграції

- Користувач надсилає аудіозапит на сервер.
- Аудіофайл завантажується через POST /transcript та передається у Whisper API для транскрипції.
- Whisper повертає текстовий запит, який передається до Llama 3 через POST /ask.
- Llama 3 аналізує текстовий запит, знаходить відповідь і повертає її серверу.
- Сервер відповідає клієнту текстовим результатом.
- За необхідності відповідь синтезується у голосовий формат через Whisper TTS.

2.3 . Тестування роботи голосового інтерфейсу

Тестування роботи голосового інтерфейсу є критичним етапом для перевірки коректності функціонування системи, її надійності та продуктивності. Процес тестування охоплює різні аспекти: від перевірки коректності транскрипції голосових запитів до швидкості обробки запитів сервером та якості отриманих відповідей.

Основні цілі тестування:

- Перевірка коректності розпізнавання мовлення: Переконатися, що аудіозапит правильно перетворюється у текст за допомогою Whisper.
- Тестування обробки текстових запитів: Оцінити, наскільки коректно сервер реагує на текстові запити, отримані з транскрипції.
- Перевірка користувацького досвіду (UX):
  Оцінити зручність використання голосового інтерфейсу та зрозумілість відповідей.

2.4 . Рекомендації щодо подальшого вдосконалення

В процесі роботи над інтерактивною системою пошуку книг у бібліотеці з використанням мультимедійного помічника та голосового асистента було виявлено ряд напрямів та функцій, які можуть покращити розроблене ПЗ, а саме:

- Зменшення часу обробки запитів: Використати асинхронну обробку запитів та оптимізувати роботу з API Groq. Можливо, слід впровадити кешування результатів часто використовуваних запитів, щоб уникнути зайвих звернень до зовнішніх сервісів.
- Впровадження стиснення аудіофайлів: Для прискорення передачі даних на сервер можна впровадити стиснення аудіофайлів перед їх обробкою. Наприклад, використовувати формат OGG або інші ефективні формати з мінімальними втратами якості.
- Додаткове навчання моделі Whisper: Налаштувати модель Whisper для роботи з конкретною мовною доменністю (українська, російська, англійська) та специфічним словником, що включає технічні або тематичні терміни.

ВИСНОВКИ

Розроблений голосовий інтерфейс став ключовим елементом проєкту, що забезпечує інтерактивну взаємодію користувача із системою через голосові команди. У ході роботи реалізовано всі необхідні функціональні компоненти: розпізнавання мовлення, обробка запитів і генерація відповідей. Основними технологіями стали Whisper для розпізнавання голосу, Llama3 для аналізу текстових запитів і формування відповідей, а також синтез мовлення для створення аудіовідповідей.

Серверна логіка на базі Node.js дозволяє ефективно обробляти голосові запити, перетворювати їх у текст і генерувати релевантні відповіді. Інтеграція голосового інтерфейсу з сервером через REST API забезпечує стабільну та швидку взаємодію між компонентами. Крім того, вдалося реалізувати динамічну обробку голосових файлів, що додає зручності у використанні системи.

Під час тестування голосовий інтерфейс продемонстрував високу точність розпізнавання мовлення навіть у складних умовах.

Таким чином, розроблені компоненти голосового інтерфейсу повністю відповідають поставленим цілям, забезпечуючи інтерактивність, точність і зручність. Надалі можливі вдосконалення, спрямовані на підвищення продуктивності, інтеграцію додаткових функцій і покращення користувацького досвіду.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Millman Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., & Amodei, D. (2020). *Language models are few-shot learners*. Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020). https://arxiv.org/abs/2005.14165.
2. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT 2019. https://arxiv.org/abs/1810.04805.
3. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI Blog. https://openai.com/research/language-unsupervised
4. Groq. (2024). Groq SDK: Speech-to-text and NLP model API. Groq Documentation. https://docs.groq.ai/
5. How Does Speech to Text Software Work? [Електронний ресурс]. — Режим доступу: https://www.amberscript.com/en/blog/how-speech-to-textsoftware-works/
6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30. https://arxiv.org/abs/1706.03762