ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтеграція Speech-to-Text API в багатокористувацькі системи для розпізнавання тексту»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми 126 Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

(підпис) *Віктор ЗОЛОЧЕВСЬКИЙ*
Ім'я, ПРІЗВИЩЕ здобувача

Виконав: здобувач вищої освіти гр. ІСДМ-63
Віктор ЗОЛОЧЕВСЬКИЙ
Ім'я, ПРІЗВИЩЕ

Керівник: Андрій БОНДАРЧУК
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент:
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
Каміла СТОРЧАК
(Ім'я, ПРІЗВИЩЕ)
«____» _____ 20__ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Золочевський Віктор Юрійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Інтеграція Speech-to-Text API в багатокористувацькі системи для розпізнавання тексту

керівник кваліфікаційної роботи Андрій БОНДАРЧУК, д.т.н., професор ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» 10.2024р. №320

2. Строк подання кваліфікаційної роботи «27» грудня 2024р.

3. Вихідні дані до кваліфікаційної роботи:

1. Науково-технічна література: Огляд сучасних публікацій, статей та книг
2. Хмарні сервіси та технології: Параметри та можливості сервісів Microsoft Azure Speech-to-Text, Google Speech-to-Text та OpenAI Whisper.
3. Вимоги до систем розпізнавання мовлення: Функціональні характеристики, точність розпізнавання (WER), підтримка мов та діалектів, інтеграція з іншими сервісами.
4. Інструменти розробки: Використання Node.js, Express.js, Mongoose, MongoDB для створення веб-додатків та управління даними.
5. Методи тестування та оцінки продуктивності: Оцінка точності розпізнавання мовлення, час обробки, вплив швидкості інтернет-з'єднання на ефективність системи.
6. Результати експериментів та тестування: Аналіз результатів тестування багатокористувацької системи, обробка декількох файлів одночасно, порівняння хмарних технологій.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз існуючих технологій розпізнавання мовлення
2. Дослідження хмарних технологій для розпізнавання мовлення
3. Розробка архітектури багатокористувацької системи
4. Розробка програмної платформи на основі Node.jsta MongoDB
5. Інтеграція з Microsoft Azure Speech-to-Text API
6. Тестування та оцінка ефективності системи

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «15» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури.	15.10-17.10.24	виконав
2	Дослідження предметної області. Огляд основних підходів та технологій розпізнавання мовлення	18.10-24.10.24	виконав
3	Обґрунтування вибору сучасних підходів і технологій розпізнавання мовлення.	25.10-29.10.24	виконав
4	Вибірковий аналіз API хмарних провайдерів для розпізнавання мовлення	30.10-05.11.24	виконав
5	Обґрунтування архітектурних рішень системи	06.11-08.11.24	виконав
6	Розробка алгоритму роботи програмного продукту. Об’рунтування вибору інструментальних засобів	09.11-11.11.24	виконав
8	Розробка програмного продукту. Опис компонентів і функціоналу	12.11-04.12.24	виконав
9	Тестування системи та аналіз результатів роботи	06.12-13.12.24	виконав
10	Підготовка доповіді, презентації, роздаткового матеріалу, реферату	20.12-26.12.24	виконав

Здобувач вищої освіти _____ Віктор ЗОЛОЧЕВСЬКИЙ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Андрій БОНДАРЧУК
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 79 стор., 14 рис., 13 табл., 30 джерел.

Мета роботи – розробка клієнтської програми для автоматичного перетворення мовлення в текст із використанням хмарних API, таких як Microsoft Azure, OpenAI та Google Cloud, для забезпечення високої точності, швидкості та зручності.

Об'єкт дослідження – технології автоматичного розпізнавання мовлення.

Предмет дослідження – методи інтеграції, порівняння та використання хмарних API для створення багатокористувацької клієнтської програми.

Короткий зміст роботи. У роботі проведено аналіз сучасних технологій розпізнавання мовлення [1] та обґрунтовано вибір технологій для створення програмного продукту. Розроблено клієнтську програму на основі Node.js, яка підтримує багатокористувацький доступ із різними рівнями прав. Виконано інтеграцію API Microsoft Azure для забезпечення точного та швидкого розпізнавання мовлення. Проведено порівняльний аналіз ефективності різних API [3] за параметрами точності, швидкості та вартості. Програмний продукт успішно протестовано на різних наборах даних.

Тестування продемонструвало високу ефективність розробленого рішення та його здатність адаптуватися до різних умов. Для подальшого розвитку пропонується інтеграція додаткових API, розширення функціоналу для аналізу якості вхідного аудіо та перехід на мікросервісну архітектуру для масштабованості.

КЛЮЧОВІ СЛОВА: РОЗПІЗНАВАННЯ МОВЛЕННЯ, ХМАРНІ API, SPEECH-TO-TEXT, NODE.JS, БАГАТОКОРИСТУВАЦЬКА СИСТЕМА.

ABSTRACT

The textual part of the qualification work for obtaining a master's degree: 77 pages, 14 figures, 5 tables, 30 sources.

The aim of the work – development of a client application for automatic speech-to-text conversion using cloud APIs such as Microsoft Azure, OpenAI, and Google Cloud to ensure high accuracy, speed, and convenience.

The object of the research – technologies for automatic speech recognition.

The subject of the research – methods of integration, comparison, and utilization of cloud APIs to create a multi-user client application.

Summary of the work. The study analyzed modern speech recognition technologies and justified the choice of technologies for developing the software product. A client application based on Node.js was developed, supporting multi-user access with different permission levels. Integration of Microsoft Azure, was implemented to ensure accurate and fast speech recognition. A comparative analysis of the effectiveness of various APIs was conducted based on accuracy, speed, and cost parameters. The software product was successfully tested on various datasets.

Testing demonstrated the high efficiency of the developed solution and its ability to adapt to different conditions. Future development suggestions include the integration of additional APIs, extended functionality for analyzing input audio quality, and transitioning to a microservices architecture for scalability.

KEYWORDS: SPEECH RECOGNITION, CLOUD APIs, SPEECH-TO-TEXT, NODE.JS, MULTI-USER SYSTEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI	Artificial Intelligence
WAV	Waveform Audio Format
ID	Identity Document
STT	Speech-to-Text
HTTP	HyperText Transfer Protocol
API	Application Programming Interface
WER	Word Error Rate
REST	Representational State Transfer
CRUD	Create Read Update Delete
NPM	Node Package Manager
JSON	JavaScript Object Notation

ВСТУП

Актуальність. У сучасному світі автоматичне розпізнавання мовлення стало важливою частиною багатьох галузей, таких як охорона здоров'я, юридичні послуги, освіта та бізнес. Попит на точні та швидкі системи обробки мовлення постійно зростає, адже вони допомагають організаціям оптимізувати процеси, підвищувати ефективність та продуктивність. Інтеграція хмарних API для розпізнавання мовлення забезпечує високі стандарти якості та доступність сервісів для багатокористувацьких систем [17].

Об’єктом дослідження є технології автоматичного розпізнавання мовлення, які включають методи перетворення аудіо сигналів у текст та їх інтеграцію в різні програмні системи.

Предметом дослідження є методологія та технічні рішення, що стосуються автоматизації процесів розпізнавання мовлення, зокрема інтеграція API Microsoft Azure Speech-to-Text у багатокористувацькі системи.

Мета дослідження: полягає у розробці та впровадженні програмної платформи для автоматичного розпізнавання мовлення на основі API Microsoft Azure Speech-to-Text, яка забезпечує ефективну обробку аудіофайлів та інтеграцію з іншими програмними продуктами. Це передбачає вивчення існуючих хмарних рішень, вибір оптимальних моделей та розробку архітектури платформи, що відповідає сучасним вимогам.

Методи дослідження: включають аналіз наукових праць та літератури у сфері автоматичного розпізнавання мовлення, експериментальні дослідження продуктивності різних моделей, а також розробку програмного забезпечення з використанням технологій NodeJS, ExpressJS та mongoose. Особлива увага приділена експериментальним дослідженням, які дозволили визначити оптимальні параметри роботи системи та її продуктивність у різних умовах.

Структура роботи. Дипломна робота складається зі вступу, трьох розділів, висновків, додатків та переліку використаних джерел. Перший розділ розглядає

актуальність теми, проблематику автоматичного розпізнавання мовлення та аналіз існуючих рішень. Другий розділ присвячено теоретичній підготовці та аналізу хмарних рішень, а також вибору оптимальних інструментів і технологій для розробки системи. Третій розділ детально описує процес розробки та тестування системи розпізнавання мовлення на основі API Microsoft Azure Speech-to-Text, включаючи інтеграцію з іншими програмними продуктами та оцінку ефективності роботи системи. Висновки підсумовують результати дослідження та містять рекомендації щодо подальшого розвитку системи. Додатки включають додаткові матеріали, які сприяють більш глибокому розумінню роботи системи.

1 РОЗПІЗНАВАННЯ МОВЛЕННЯ НА ОСНОВІ ХМАРНИХ API

1.1 Актуальність теми та проблематика

Сучасні технології автоматичного розпізнавання мовлення [14] стають все більш важливими у багатьох сферах життя, забезпечуючи нові можливості для підвищення ефективності та продуктивності. Інтеграція цих технологій у багатокористувацькі системи на підприємствах різних галузей дозволяє автоматизувати рутинні завдання, покращувати внутрішню комунікацію та забезпечувати швидкий доступ до необхідної інформації. Впровадження систем розпізнавання мовлення значно спрощує процеси документування та обробки даних, що є надзвичайно важливим для ефективної роботи сучасних організацій.

У глобальному масштабі автоматичне розпізнавання мовлення сприяє прискоренню інформаційного обміну та підвищенню якості надання послуг. Це стає можливим завдяки здатності таких систем миттєво перетворювати мовні повідомлення в текст, що спрощує їх подальший аналіз та обробку. Таким чином, інтеграція розпізнавання мовлення в інформаційні системи підприємств дозволяє забезпечити більш ефективну комунікацію, скоротити час на виконання завдань та зменшити кількість помилок, пов’язаних з ручним введенням даних.

Однак, як і будь-яка інноваційна технологія, системи розпізнавання мовлення мають свої переваги та недоліки. Важливо враховувати обидва аспекти для забезпечення максимальної ефективності впровадження та використання цих систем. Відповідні підходи до інтеграції та налаштування систем дозволяють максимально використовувати їх потенціал, забезпечуючи при цьому надійність та безпеку даних.

Не можна не згадати про економічну ефективність впровадження систем розпізнавання мовлення. Витрати на впровадження та утримання таких систем можуть бути значними, однак вони окупаються завдяки значному підвищенню продуктивності та ефективності роботи підприємства. Автоматизація рутинних

завдань дозволяє скоротити витрати часу та ресурсів, що в кінцевому підсумку призводить до зменшення операційних витрат та збільшення прибутковості.

Іншим важливим аспектом є сумісність систем розпізнавання мовлення з існуючими програмними продуктами, що використовуються на підприємствах. Для забезпечення безперебійної роботи необхідно, щоб системи розпізнавання мовлення могли інтегруватися з наявними інформаційними системами, забезпечуючи швидкий та надійний доступ до даних. Це дозволяє забезпечити більш ефективну роботу підприємства, підвищуючи загальну продуктивність та зменшуючи ризик виникнення помилок.

1.1.1 Опис важливості розпізнавання мовлення у сучасному світі

Автоматичне розпізнавання мовлення у сучасному світі має велике значення для багатьох секторів і сфер життя [22]. Однією з ключових переваг цієї технології є можливість значно підвищити ефективність роботи за рахунок автоматизації процесів, які раніше виконувалися вручну. Завдяки цьому підходу підприємства можуть швидко й безпомилково обробляти величезні обсяги інформації, що сприяє зменшенню навантаження на працівників та покращенню загальної продуктивності.

Одним з важливих прикладів застосування розпізнавання мовлення є інтеграція цієї технології у корпоративні комунікаційні системи. Використання розпізнавання мовлення у таких системах дозволяє автоматично створювати текстові версії розмов, конференцій та зустрічей, що значно полегшує процес документування та подальшого аналізу інформації. Це дозволяє швидко знайти потрібні дані, що сприяє ефективнішому прийняттю рішень та покращенню внутрішньої комунікації.

Ще одним прикладом є використання технологій розпізнавання мовлення у системах підтримки клієнтів. Інтеграція автоматичного розпізнавання мовлення у такі системи дозволяє швидко обробляти запити клієнтів, автоматизувати відповіді на типові питання та покращити якість обслуговування. Це особливо важливо для

компаній, що працюють з великою кількістю клієнтів, адже забезпечує швидке та точне оброблення запитів, зменшує час очікування та підвищує задоволеність клієнтів.

Розпізнавання мовлення також має велике значення в освітніх системах. Використання цієї технології в навчальних процесах дозволяє автоматично створювати текстові версії лекцій, семінарів та інших навчальних матеріалів, що робить їх доступнішими для студентів. Це особливо корисно для студентів з обмеженими можливостями, яким важливо мати доступ до текстових версій аудіо- та відеоматеріалів.

Важливість розпізнавання мовлення також полягає в його здатності покращити доступність інформації. За допомогою цієї технології можна автоматично створювати субтитри для відео, що робить контент доступнішим для людей з порушенням слуху. Також, розпізнавання мовлення може використовуватися для автоматичного перекладу текстів, що сприяє кращій міжнародній комунікації та розширенню аудиторії.

Інтеграція систем розпізнавання мовлення у багатокористувацькі системи на підприємствах різних галузей відкриває нові можливості для розвитку та впровадження інновацій. Вона забезпечує більш швидке та точне виконання завдань, покращує внутрішню комунікацію та дозволяє зменшити людські помилки. Це робить автоматичне розпізнавання мовлення невід'ємною частиною сучасних технологічних рішень, що сприяє підвищенню ефективності та продуктивності підприємств у різних сферах.

1.1.2 Недоліки та обмеження існуючих систем

Незважаючи на численні переваги та високий потенціал автоматичного розпізнавання мовлення, існуючі системи мають ряд недоліків та обмежень, які можуть ускладнювати їх інтеграцію у багатокористувацькі системи на підприємствах різних галузей. Ці проблеми варто враховувати при впровадженні

таких технологій, щоб забезпечити їх ефективну роботу та максимальну користь для організацій.

Одним з ключових недоліків існуючих систем є обмежена точність розпізнавання мовлення в умовах шуму або перешкод. У реальних умовах на підприємствах часто виникають ситуації, коли фоновий шум або голоси інших людей можуть впливати на якість розпізнавання мовлення. Це може призводити до помилок у розпізнаванні та, відповідно, до неправильного введення даних, що може суттєво вплинути на продуктивність та якість роботи системи.

Іншим важливим обмеженням є складність обробки різних мов та діалектів. Хоча сучасні системи розпізнавання мовлення можуть підтримувати кілька мов, точність розпізнавання може значно відрізнятися для різних мов та діалектів. Це особливо актуально для багатонаціональних підприємств, де співробітники можуть спілкуватися різними мовами. В таких випадках, недостатня точність розпізнавання може стати серйозною перешкодою для ефективної роботи системи.

Ще одним обмеженням є висока вартість впровадження та утримання систем розпізнавання мовлення. Використання хмарних API, таких як Microsoft Azure Speech-to-Text [15], може вимагати значних фінансових вкладень, особливо для великих підприємств з численними користувачами. Це може стати бар'єром для впровадження таких систем, особливо для малих та середніх підприємств, які не можуть дозволити собі високі витрати.

Крім того, інтеграція систем розпізнавання мовлення у багатокористувацькі системи може супроводжуватись технічними викликами, такими як сумісність з існуючими програмними продуктами та забезпеченням безпеки даних. Важливо, щоб системи розпізнавання мовлення могли безперешкодно інтегруватися з наявними інформаційними системами та забезпечувати захист конфіденційної інформації. Недоліки у сумісності та безпеці можуть призвести до витоків даних або збою в роботі системи, що негативно вплине на ефективність роботи підприємства.

Необхідність постійного оновлення та підтримки систем розпізнавання мовлення також є значним обмеженням. Сучасні технології швидко розвиваються,

і щоб залишатися ефективними, системи розпізнавання мовлення повинні регулярно оновлюватися. Це вимагає додаткових ресурсів і може бути складним для підприємств, які не мають достатнього технічного персоналу або можливостей для підтримки систем.

Узагальнюючи, існуючі системи розпізнавання мовлення мають ряд недоліків та обмежень, які можуть ускладнювати їх інтеграцію у багатокористувацькі системи на підприємствах різних галузей. Важливо враховувати ці проблеми при впровадженні технологій розпізнавання мовлення, щоб забезпечити їх ефективну роботу та максимальну користь для організацій. Подолання цих викликів вимагає комплексного підходу, включаючи вдосконалення технологій, оптимізацію витрат та забезпечення сумісності та безпеки систем.

1.2 Аналіз існуючих підходів і технологій

Автоматичне розпізнавання мовлення є однією з найбільш інноваційних технологій, що сприяють розвитку сучасного бізнесу. З розвитком технологій виникли два основних підходи до реалізації цієї задачі: локальні обчислювальні рішення та хмарні рішення. У цьому пункті ми детально розглянемо обидва підходи, зосереджуючи увагу на перевагах хмарних рішень у порівнянні з локальними, зокрема в контексті розпізнавання мовлення з аудіо для бізнесу.

Локальні обчислювальні рішення передбачають використання власних серверів та обладнання підприємства для обробки аудіо даних. Цей підхід надає певну автономію та можливість контролювати всі етапи обробки даних, що може бути важливо для деяких організацій. Однак, такі рішення мають ряд обмежень, які можуть негативно вплинути на ефективність та продуктивність.

По-перше, висока вартість початкового впровадження є суттєвим недоліком локальних обчислювальних рішень. Придбання необхідного обладнання та програмного забезпечення вимагає значних фінансових ресурсів. Додатково, необхідність технічної підтримки та обслуговування обладнання призводить до додаткових витрат.

По-друге, обмежені можливості масштабування локальних рішень є ще одним значним недоліком. Зростання обсягу даних або кількості користувачів може вимагати значних інвестицій у додаткове обладнання, що не завжди є економічно вигідним.

З іншого боку, хмарні рішення, такі як Microsoft Azure Speech-to-Text, пропонують значні переваги у порівнянні з локальними обчислювальними рішеннями. Використання хмарних сервісів дозволяє використовувати потужні обчислювальні ресурси [30], забезпечуючи високу продуктивність та надійність. Це особливо важливо для підприємств, що обробляють великі обсяги аудіо даних або мають велику кількість користувачів.

Однією з головних переваг хмарних рішень є їх гнучкість та можливість легкого масштабування [11]. У разі зростання потреб бізнесу, хмарні рішення дозволяють швидко та без значних витрат збільшити обчислювальні ресурси. Це дозволяє підприємствам ефективно реагувати на зміни ринкових умов та забезпечувати безперервність бізнес-процесів.

Крім того, хмарні рішення мають високу надійність та забезпечують захист даних. Використання передових технологій безпеки та регулярні оновлення забезпечують високий рівень захисту конфіденційної інформації. Це особливо важливо для підприємств, що працюють з чутливими даними та повинні дотримуватись високих стандартів безпеки.

Також важливо зазначити, що хмарні рішення зазвичай вимагають мінімальних витрат на технічну підтримку та обслуговування. Це дозволяє підприємствам зосередитися на основній діяльності, не витрачаючи ресурси на підтримку інфраструктури.

Нижче наведено порівняльну таблицю, яка відображає переваги хмарних рішень у порівнянні з локальними обчислювальними рішеннями:

Таблиця 1.1

Порівняння хмарних та локальних рішень

Локальні обчислювальні рішення	**Хмарні рішення**
Висока вартість початкового впровадження	Низька вартість початкового впровадження
Необхідність технічної підтримки та обслуговування	Мінімальні витрати на технічну підтримку
Обмежені можливості масштабування	Легке масштабування в залежності від потреб
Низька гнучкість та адаптивність	Висока гнучкість та адаптивність
Складність забезпечення безпеки даних	Високий рівень безпеки та захисту даних
Високі витрати на оновлення та модернізацію	Автоматичні оновлення та модернізація
Обмежена продуктивність	Висока продуктивність завдяки потужним хмарним ресурсам
Тривалі строки впровадження	Швидке впровадження та готовність до роботи
Необхідність постійної підтримки інфраструктури	Відсутність потреби у підтримці інфраструктури
Ризик збоїв через відмову обладнання	Висока надійність та мінімізація ризиків збоїв

Таким чином, аналіз існуючих підходів і технологій показує, що хмарні рішення мають значні переваги у порівнянні з локальними обчислювальними рішеннями. Вони забезпечують високу продуктивність, гнучкість, безпеку та економічну ефективність, що робить їх оптимальним вибором для бізнесу у контексті розпізнавання мовлення з аудіо. Хмарні рішення дозволяють підприємствам швидко адаптуватися до змін, забезпечуючи при цьому надійну та безперебійну роботу інформаційних систем.

1.3 Вплив якості аудіо на якість результату розпізнавання мовлення

Якість аудіо відіграє ключову роль у процесі розпізнавання мовлення. Низька якість записів може призвести до значного зниження точності розпізнавання, що, в свою чергу, може вплинути на ефективність використання автоматизованих систем.

Однією з основних проблем є шумові перешкоди. Фонові шуми, такі як звуки вулиці, музика або розмови інших людей, можуть перекривати мовлення користувача, роблячи його важко зрозумілим для системи. Щоб зменшити вплив шуму, можна використовувати шумоподавлюючі мікрофони та застосовувати алгоритми шумоподавлення на етапі попередньої обробки аудіо. Також важливо проводити записи у тихих середовищах, де рівень фонових шумів мінімальний.

Ехо та реверберація є ще однією проблемою, яка може виникати при записі в приміщеннях з поганою акустикою. Це призводить до багаторазового відбиття звуків, що ускладнює розпізнавання мовлення. Для вирішення цієї проблеми можна використовувати мікрофони з високою спрямованістю, які зменшують вплив відбитих звуків, а також акустичні панелі для поліпшення акустики приміщення. Застосування алгоритмів зниження ехо та реверберації під час обробки аудіо також є ефективним методом.

Низька якість запису може спричинити втрату важливої інформації через відсутність чіткості у записі, спотворення звуку та недостатню чутливість мікрофонів. Використання високоякісних мікрофонів та рекордерів, регулярна перевірка та калібрування обладнання для забезпечення оптимальної якості запису допоможуть уникнути цих проблем. Варто також використовувати формати з високою якістю аудіо, такі як WAV [13], для збереження максимальної деталізації звуку.

Різноманітність акцентів і діалектів також може спричинити помилки у розпізнаванні мовлення. Мовлення користувачів може суттєво відрізнятися в залежності від їх акцентів чи діалектів. Використання адаптивних алгоритмів розпізнавання, які можуть навчатися на даних з різними акцентами і діалектами, допоможе покращити точність розпізнавання. Збір і аналіз великої кількості зразків мовлення з різними акцентами також сприятиме вдосконаленню моделей розпізнавання.

Іншою важливою проблемою є формати аудіо файлів. Непідтримувані або нестандартні формати аудіо файлів можуть створювати додаткові труднощі для системи розпізнавання мовлення. Використання стандартних і поширених

форматів аудіо файлів, таких як WAV або MP3, та забезпечення можливості конвертації аудіо файлів у підтримувані формати перед розпізнаванням допоможуть уникнути цих труднощів.

Тривалість записів також може впливати на процес розпізнавання мовлення. Дуже довгі або дуже короткі аудіо файли можуть негативно впливати на якість розпізнавання. Оптимізація тривалості аудіо файлів для розпізнавання, розбиття довгих записів на менші фрагменти та налаштування параметрів розпізнавання для обробки коротких аудіо файлів є методами вирішення цієї проблеми.

Розв'язання цих проблем дозволить значно покращити якість розпізнавання мовлення та підвищити ефективність систем автоматичного розпізнавання мовлення. Враховуючи ці аспекти, можна створити систему, яка забезпечуватиме високу точність розпізнавання та задовольнятиме потреби різних категорій користувачів.

1.4 Постановка задачі

Оскільки основною метою є створення багатокористувацької системи, здатної забезпечити високу точність і продуктивність розпізнавання мовлення. Важливо, щоб ця система легко інтегрувалася з іншими програмними продуктами, була економічно ефективною та масштабованою.

Для досягнення цієї мети необхідно вирішити низку завдань. Перш за все, потрібно створити архітектуру системи, яка зможе ефективно обробляти великі обсяги аудіо даних у режимі реального часу. Це передбачає використання потужних обчислювальних ресурсів хмарних сервісів, здатних забезпечити високу продуктивність та надійність. Важливо також забезпечити можливість масштабування системи відповідно до потреб різних категорій підприємств, щоб вона могла безперебійно функціонувати навіть у разі значного збільшення обсягу даних або кількості користувачів.

Крім цього, необхідно розробити алгоритми та методи розпізнавання мовлення, які дозволять досягти високої точності обробки аудіо даних. Це включає

вибір оптимальних моделей і технологій, здатних ефективно розпізнавати мовлення навіть в умовах шуму або перешкод. Важливо також враховувати різні мови та діалекти для забезпечення максимальної універсальності та адаптивності системи.

Ще однією важливою задачею є інтеграція системи розпізнавання мовлення з іншими програмними продуктами. Це включає розробку універсальних API, які дозволять легко інтегрувати систему з іншими інформаційними системами, забезпечуючи швидкий і надійний доступ до даних. Важливо також забезпечити можливість інтеграції з існуючими програмними продуктами без значних змін у їх архітектурі, що дозволить зберегти цілісність і стабільність інформаційної інфраструктури.

Одним з ключових аспектів є забезпечення безпеки даних у системі розпізнавання мовлення. Використання передових технологій безпеки та регулярні оновлення повинні забезпечити високий рівень захисту конфіденційної інформації. Важливо також впровадити механізми управління доступом до даних, щоб контролювати, хто і як може використовувати інформацію, зберігаючи її конфіденційність та цілісність.

Забезпечення економічної ефективності системи також є надзвичайно важливим завданням. Це включає оптимізацію витрат на впровадження і обслуговування системи, що дозволить підприємствам використовувати ці технології без значних фінансових навантажень. Важливо враховувати довгострокові витрати на технічну підтримку і модернізацію системи, щоб забезпечити її стабільну і ефективну роботу протягом усього життєвого циклу.

Серед основних елементів системи можна виділити архітектуру, що забезпечує обробку великих обсягів аудіо даних у реальному часі з використанням хмарних обчислювальних ресурсів для забезпечення високої продуктивності і надійності. Розробка алгоритмів і методів розпізнавання мовлення передбачає вибір оптимальних моделей і технологій, врахування різних мов та діалектів для забезпечення високої точності. Інтеграція з іншими системами включає розробку універсальних API для легкого інтегрування з іншими інформаційними системами і забезпечення швидкого і надійного доступу до даних.

Особливої уваги потребує багатокористувацька складова системи. Універсальність розробленої системи повинна забезпечити можливість її використання багатьма користувачами одночасно, що є критично важливим для підприємств з великою кількістю співробітників. Багатокористувацька система дозволяє ефективно розподіляти ресурси, керувати доступом до інформації та забезпечувати безперервну роботу всіх користувачів. Це підвищує продуктивність і дозволяє підприємствам максимально використовувати можливості автоматичного розпізнавання мовлення.

Таким чином, задача дослідження полягає у створенні універсальної системи автоматичного розпізнавання мовлення, яка буде гнучкою, масштабованою, багатокористувацькою та економічно ефективною, забезпечуючи при цьому високу точність розпізнавання мовлення та безпеку даних для різних категорій підприємств.

1.4.1 Чітке формулювання задачі дослідження

Основною задачею дослідження є розробка та впровадження універсальної системи автоматичного розпізнавання мовлення на основі хмарних рішень, яка може бути адаптована для використання в різних типах підприємств та організацій. Ця система повинна забезпечити високу точність і продуктивність розпізнавання мовлення, легко інтегруватися з іншими програмними продуктами, а також бути економічно ефективною і масштабованою.

Перш за все, необхідно створити архітектуру системи, яка зможе ефективно обробляти великі обсяги аудіо даних у режимі реального часу. Це передбачає використання потужних обчислювальних ресурсів хмарних сервісів, що забезпечують високу продуктивність і надійність. Важливо також передбачити можливість масштабування системи відповідно до потреб різних категорій підприємств, щоб забезпечити безперебійну роботу навіть у разі значного збільшення обсягу даних або кількості користувачів.

Далі необхідно розробити алгоритми та методи розпізнавання мовлення, які забезпечать високу точність обробки аудіо даних. Це включає вибір оптимальних моделей і технологій, що здатні ефективно розпізнавати мовлення навіть в умовах шуму або перешкод. Важливо також враховувати різні мови та діалекти, щоб забезпечити максимальну універсальність і адаптивність системи.

Інтеграція системи розпізнавання мовлення з іншими програмними продуктами є ще однією важливою задачею. Це включає розробку універсальних API, які дозволять легко інтегрувати систему з іншими інформаційними системами, забезпечуючи швидкий і надійний доступ до даних. Важливо також передбачити можливість інтеграції з існуючими програмними продуктами без значних змін у їх архітектурі, що дозволить зберегти цілісність і стабільність інформаційної інфраструктури.

Одним із важливих аспектів є забезпечення економічної ефективності системи розпізнавання мовлення. Це включає оптимізацію витрат на впровадження і обслуговування системи, що дозволить підприємствам використовувати ці технології без значних фінансових навантажень. Важливо також враховувати довгострокові витрати на технічну підтримку і модернізацію системи, забезпечуючи її стабільну і ефективну роботу протягом усього життєвого циклу.

Ключовими елементами системи є:

- Архітектура системи: забезпечення обробки великих обсягів аудіо даних у реальному часі, використання хмарних обчислювальних ресурсів для забезпечення високої продуктивності і надійності.

- Алгоритми і методи розпізнавання мовлення: вибір оптимальних моделей і технологій, врахування різних мов та діалектів для забезпечення високої точності.

- Інтеграція з іншими системами: розробка універсальних API для легкого інтегрування з іншими інформаційними системами, забезпечення швидкого і надійного доступу до даних.

- Економічна ефективність: оптимізація витрат на впровадження і обслуговування системи, врахування довгострокових витрат на технічну підтримку і модернізацію.

Особливої уваги потребує багатокористувацька складова системи. Універсальність розробленої системи має забезпечити можливість її використання багатьма користувачами одночасно, що є критично важливим для підприємств з великою кількістю співробітників. Багатокористувацька система дозволяє ефективно розподіляти ресурси, керувати доступом до інформації і забезпечувати безперервну роботу всіх користувачів. Це підвищує продуктивність і дозволяє підприємствам максимально використовувати можливості автоматичного розпізнавання мовлення.

Таким чином, задача дослідження полягає у створенні універсальної системи автоматичного розпізнавання мовлення, яка буде гнучкою, масштабованою, багатокористувацькою та економічно ефективною, забезпечуючи при цьому високу точність розпізнавання мовлення та безпеку даних для різних категорій підприємств.

1.4.2 Очікувані результати

Очікувані результати є важливою частиною будь-якого дослідження, оскільки вони визначають, чого саме ми прагнемо досягти в результаті виконаної роботи. У контексті розробки універсальної системи автоматичного розпізнавання мовлення на основі хмарних рішень, основними очікуваними результатами є:

- *Висока точність розпізнавання мовлення*: Розробка алгоритмів та методів, які забезпечують високу точність розпізнавання мовлення в умовах різних рівнів шуму та перешкод. Очікується, що система буде здатна розпізнавати мовлення з точністю, достатньою для використання у різних галузях, включаючи медицину, юриспруденцію, освіту та бізнес.

- *Інтеграція з іншими системами*: Створення універсальних API, які дозволять легко інтегрувати систему з іншими інформаційними системами підприємств. Очікується, що система буде сумісною з існуючими програмними продуктами та зможе працювати без значних змін у їх архітектурі, що дозволить зберегти цілісність та стабільність інформаційної інфраструктури.

- *Ефективне керування користувачами та групами*: Розробка багатокористувацької системи, яка забезпечить ефективне керування доступом до ресурсів та інформації. Очікується, що система дозволить керувати великою кількістю користувачів, які можуть бути поділені на групи відповідно до проєктів, над якими вони працюють. Система повинна забезпечити можливість одночасного доступу багатьох користувачів до ресурсів без зниження продуктивності.

- *Економічна ефективність*: Оптимізація витрат на впровадження та обслуговування системи. Очікується, що система буде економічно ефективною, що дозволить підприємствам використовувати її без значних фінансових навантажень. Важливо також враховувати довгострокові витрати на технічну підтримку та модернізацію системи, забезпечуючи її стабільну і ефективну роботу протягом усього життєвого циклу.

- *Універсальність та адаптивність*: Розробка системи, яка буде універсальною і зможе адаптуватися до потреб різних категорій підприємств. Очікується, що система зможе працювати з різними мовами та діалектами, забезпечуючи високу точність розпізнавання мовлення для різних користувачів.

- *Простота використання*: Створення інтуїтивно зрозумілого інтерфейсу користувача, який дозволить легко використовувати систему без необхідності спеціальної підготовки. Очікується, що система буде зручною у використанні як для технічних, так і для нетехнічних користувачів, що сприятиме її широкому впровадженню.

Загалом, очікувані результати включають розробку універсальної багатокористувацької системи автоматичного розпізнавання мовлення, яка буде забезпечувати високу точність, ефективність, безпеку та економічну доцільність. Це дозволить підприємствам з різних галузей максимально використовувати можливості автоматичного розпізнавання мовлення для покращення своїх бізнес-процесів та підвищення продуктивності.

1.5 Висновки до розділу

У даному розділі були розглянуті ключові аспекти розпізнавання мовлення на основі хмарних API та вплив якості аудіо на якість результату розпізнавання. Досліджено актуальність цієї теми в сучасному світі, підкреслюючи важливість автоматичного розпізнавання мовлення у різних галузях. Також ознайомлено з основними недоліками та обмеженнями існуючих систем, які стимулюють розвиток нових підходів та технологій.

Було проведено детальний аналіз існуючих підходів і технологій, порівнюючи хмарні рішення з локальними обчисленнями. Виявлено, що хмарні рішення мають значні переваги, такі як масштабованість, економічна ефективність та висока продуктивність, у порівнянні з локальними системами, які можуть бути обмеженими через високі витрати на впровадження та технічну підтримку.

Особлива увага була приділена впливу якості аудіо на результати розпізнавання мовлення. Було висвітлено основні проблеми, пов'язані з підготовкою аудіо, такі як шумові перешкоди, ехо, низька якість запису, різноманітність акцентів і діалектів, а також тривалість записів. Розглянуто можливі методи вирішення цих проблем, включаючи використання шумоподавлюючих мікрофонів, алгоритмів шумоподавлення, високоякісного записуючого обладнання та адаптивних алгоритмів розпізнавання.

Чітко сформульовано задачі дослідження, які передбачають розробку універсальної багатокористувацької системи автоматичного розпізнавання мовлення. Така система має забезпечити високу точність і продуктивність, легку інтеграцію з іншими програмними продуктами, а також бути економічно ефективною і масштабованою. Важливою складовою цієї системи є її універсальність, яка дозволить використовувати її у різних типах підприємств та організацій, задовольняючи потреби великої кількості користувачів одночасно.

Також визначено очікувані результати розробки системи, серед яких висока точність розпізнавання мовлення, масштабованість і гнучкість, ефективне

керування користувачами та групами, безпека даних, економічна ефективність, універсальність та адаптивність, а також простота використання.

Очікується, що розробка на основі хмарних API для розпізнавання мовлення дозволить забезпечити ефективне використання цієї технології в різних галузях і підприємствах. Наступним кроком є перехід до детального планування розробки, визначення архітектури системи та вибору оптимальних технологій для її реалізації.

2 ТЕОРЕТИЧНА ПІДГОТОВКА ДО ВИРІШЕННЯ ЗАДАЧІ

2.1 Вибірковий аналіз існуючих хмарних рішень для розпізнавання мовлення

Завдяки хмарним рішенням, обробка мовлення стала більш доступною, точною та ефективною. Хмарні сервіси дозволяють підприємствам використовувати передові технології для підвищення продуктивності та зменшення витрат на інфраструктуру.

Серед різних хмарних рішень для розпізнавання мовлення особливо виділяються Microsoft Azure, Google Speech та OpenAI Whisper. Кожне з цих рішень має свої унікальні характеристики та можливості, що робить їх популярними серед користувачів. Розуміння їх особливостей допоможе визначити, яке з них найкраще підходить для конкретних потреб і завдань.

Детальний огляд цих сервісів включає аналіз їхніх основних характеристик, таких як точність розпізнавання, підтримка мов і діалектів, можливості адаптації моделей, інтеграція з іншими системами, вартість використання, а також забезпечення безпеки і конфіденційності даних.

Вибірковий аналіз існуючих хмарних рішень для розпізнавання мовлення дозволяє виявити їх сильні та слабкі сторони, що допоможе зробити обґрунтований вибір найбільш оптимального рішення для конкретних потреб підприємства.

2.1.1 Огляд існуючих хмарних рішень для розпізнавання мовлення

Автоматичне розпізнавання мовлення стало невід'ємною частиною багатьох сучасних систем і додатків. Завдяки хмарним рішенням, обробка мовлення стала більш доступною, точною та ефективною. Розглянемо три основні хмарні рішення для розпізнавання мовлення: Microsoft Azure, Google Speech та OpenAI.

Microsoft Azure Speech-to-Text

Microsoft Azure Speech-to-Text - це потужний сервіс для автоматичного розпізнавання мовлення, який пропонує високий рівень точності та масштабованості [21]. Основні характеристики цього сервісу включають підтримку понад 85 мов і діалектів, можливість обробки аудіо в режимі реального часу, а також роботу з попередньо записаними файлами. Azure Speech-to-Text використовує передові технології машинного навчання для забезпечення високої точності розпізнавання мовлення.

Сервіс надає можливість автоматичного розпізнавання знаків пунктуації, що дозволяє покращити читабельність розпізнаного тексту. Крім того, він підтримує різні формати аудіо файлів, що робить його універсальним інструментом для різних застосувань. Microsoft Azure також пропонує можливості адаптації моделей для специфічних потреб користувачів, що дозволяє досягти ще вищої точності в конкретних сценаріях використання.

Однією з переваг Azure Speech-to-Text є його інтеграція з іншими сервісами Microsoft, такими як Azure Cognitive Services, що забезпечує зручність використання у комплексних рішеннях. Цей сервіс також надає високий рівень безпеки та конфіденційності даних, що є важливим для підприємств, які працюють з чутливою інформацією.

Google Cloud Speech-to-Text

Google Cloud Speech-to-Text є ще одним провідним хмарним рішенням для розпізнавання мовлення. Сервіс підтримує більше 125 мов і діалектів, що робить його одним з найбільш універсальних інструментів на ринку. Google Cloud Speech-to-Text використовує передові моделі глибокого навчання для забезпечення високої точності розпізнавання мовлення.

Основними можливостями цього сервісу є обробка мовлення в режимі реального часу, підтримка попередньо записаних аудіо файлів, а також автоматичне розпізнавання знаків пунктуації. Google Cloud Speech-to-Text надає можливості адаптації моделей для специфічних завдань і галузей, що дозволяє значно підвищити точність розпізнавання в умовах, які відрізняються від загальних сценаріїв.

Додатково, сервіс інтегрується з іншими продуктами Google Cloud, що дозволяє створювати комплексні рішення, поєднуючи різні технології. Однією з ключових переваг є здатність працювати з великими обсягами даних, забезпечуючи високу продуктивність та надійність. Google також забезпечує високий рівень безпеки та захисту даних, що є критично важливим для багатьох підприємств.

OpenAI Whisper

OpenAI Whisper є ще одним потужним інструментом для розпізнавання мовлення, розробленим OpenAI. Whisper використовує найсучасніші моделі машинного навчання для забезпечення високої точності розпізнавання мовлення в різних умовах. Цей сервіс підтримує кілька десятків мов, що робить його зручним для міжнародного використання.

Однією з особливостей OpenAI Whisper є його здатність розпізнавати мовлення в складних умовах, таких як наявність фонових шумів або мовлення з різними акцентами. Whisper також може обробляти аудіо файли різних форматів, що забезпечує гнучкість у використанні. Як і інші провідні сервіси, Whisper надає можливості адаптації моделей для специфічних потреб, що дозволяє підвищити точність розпізнавання для конкретних завдань.

OpenAI також забезпечує високий рівень безпеки та конфіденційності даних, що є важливим для підприємств, які працюють з чутливою інформацією. Крім того, Whisper інтегрується з іншими продуктами OpenAI, що дозволяє створювати комплексні рішення для різних галузей.

2.1.2 Порівняння Microsoft Azure Speech-to-Text API, Google Speech-to-Text API, OpenAI Speech-to-Text API

Сучасні технології розпізнавання мовлення на основі хмарних рішень пропонують широкі можливості для різних підприємств та організацій. Серед найбільш популярних і ефективних сервісів можна виділити Microsoft Azure Speech-to-Text, Google Speech-to-Text та OpenAI Speech-to-Text. Кожен з цих сервісів має свої унікальні характеристики, які варто детально розглянути.

Точність розпізнавання (WER)

Точність розпізнавання мовлення, яка вимірюється за допомогою метрики Word Error Rate (WER) [28], є ключовим фактором при виборі відповідного сервісу. WER визначає частоту помилок у розпізнаному тексті, включаючи пропущені, додаткові або неправильні слова.

Microsoft Azure Speech-to-Text демонструє високу точність розпізнавання з низьким рівнем WER, який становить 14.7%. Особливо цей сервіс добре працює при обробці різних акцентів та діалектів. Це дозволяє покращити точність розпізнавання в складних умовах, таких як фоновий шум або різноманітність вимови.

Google Speech-to-Text також забезпечує високий рівень точності розпізнавання, проте іноді може стикатися з викликами при обробці менш поширених акцентів та діалектів. Його WER в середньому становить 52.90%, що трохи вищий у порівнянні з Azure у специфічних умовах.

OpenAI Speech-to-Text використовує передові моделі для розпізнавання мовлення, що дозволяє досягати високої точності. Його WER становить 7.6%, що є дуже ефективним результатом. Проте точність може варіюватися залежно від умов запису і якості аудіо, що впливає на кінцевий результат.

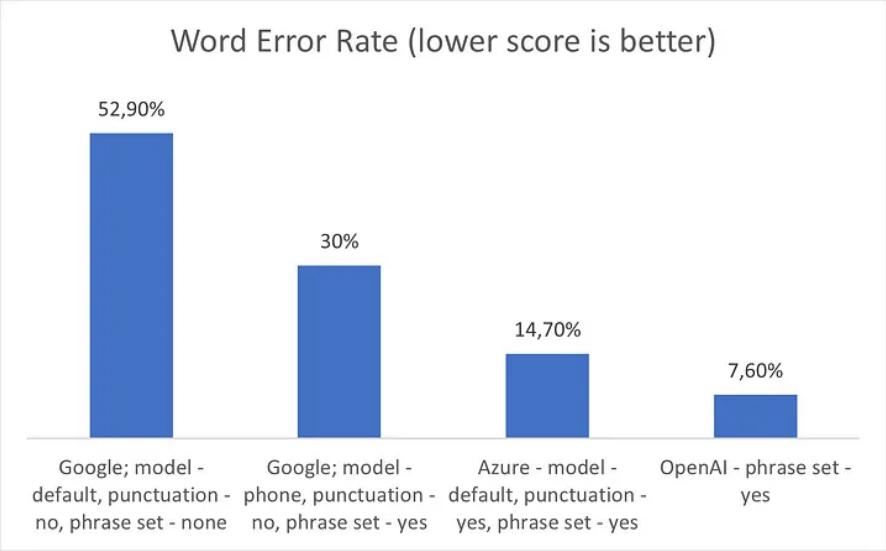


Рис. 2.1. Порівняння показників рівня помилок WER розпізнавання мовлення у хмарних системах: Microsoft Azure Speech-to-Text, Google Speech-to-Text та OpenAI Speech-to-Text

Підтримка мов і діалектів

Кількість підтримуваних мов і діалектів є важливим аспектом для міжнародних компаній та проектів. Google Speech-to-Text має явну перевагу завдяки підтримці понад 125 мов і діалектів, що робить його одним з найуніверсальніших рішень.

Microsoft Azure Speech-to-Text підтримує понад 85 мов і діалектів, що також є значним показником. Важливо зазначити, що якість розпізнавання не тільки залежить від кількості підтримуваних мов, але й від точності кожної з них. Це забезпечує високу ефективність Azure у цьому аспекті.

OpenAI Speech-to-Text підтримує кілька десятків мов, що робить його менш універсальним у порівнянні з Google та Azure. Проте цей сервіс все одно достатньо ефективний для більшості поширених мов.

Адаптація моделей

Можливість адаптації моделей для специфічних завдань є важливим фактором при виборі хмарного рішення. Усі три сервіси пропонують можливість адаптації моделей, проте Microsoft Azure Speech-to-Text виділяється більш гнучкими налаштуваннями та ширшими можливостями для адаптації. Це дозволяє підприємствам налаштовувати модель відповідно до своїх специфічних потреб, що забезпечує вищу точність розпізнавання у конкретних сценаріях використання.

Інтеграція з іншими системами

Здатність інтегрувати хмарний сервіс з існуючими інформаційними системами та додатками є важливим аспектом для забезпечення безперебійної роботи. Microsoft Azure Speech-to-Text має явну перевагу завдяки своїй інтеграції з іншими сервісами Microsoft, такими як Azure Cognitive Services. Це забезпечує зручність використання у комплексних рішеннях та дозволяє підприємствам максимально використовувати можливості екосистеми Microsoft.

Google Speech-to-Text також інтегрується з іншими продуктами Google Cloud, що робить його зручним для користувачів, які вже використовують сервіси Google. OpenAI Speech-to-Text надає інтеграційні можливості, проте не має такої широкої інтеграції, як Azure та Google.

Вартість

Економічна ефективність є важливим критерієм при виборі хмарного рішення. Вартість використання Microsoft Azure Speech-to-Text, Google Speech-to-Text та OpenAI Speech-to-Text може значно варіюватися в залежності від обсягу використання та специфічних потреб підприємства.

Microsoft Azure пропонує гнучкі цінові моделі та варіанти підписки, що дозволяє підприємствам обирати найбільш оптимальний варіант залежно від їх бюджету та вимог. Azure часто пропонує різні знижки та спеціальні пропозиції для корпоративних клієнтів, що може значно знизити витрати на впровадження та використання сервісу.

Google Speech-to-Text має конкурентоспроможну цінову політику, але може бути дорожчим за Azure при використанні в великих обсягах. OpenAI Speech-to-Text пропонує різні цінові моделі, які можуть бути привабливими для стартапів та малих підприємств, проте вартість використання може зрости при збільшенні обсягу даних.

2.2 Вибір інструментів і технологій

При розробці нових систем, таких як автоматичне розпізнавання мовлення, важливо вибрати найбільш підходящі інструменти та технології, що забезпечать високу продуктивність, надійність і зручність використання. Правильний вибір технологій може значно вплинути на успіх проекту та ефективність його реалізації.

Одним з ключових рішень при виборі технологій є визначення, чи використовувати локальні обчислення або хмарні рішення. Кожен з підходів має свої переваги та недоліки, які варто враховувати під час прийняття рішення. Хмарні рішення набувають все більшої популярності завдяки своїй масштабованості, економічній ефективності та гнучкості. Також вони забезпечують високу доступність, надійність і можливість швидкого впровадження нових технологій без значних витрат на інфраструктуру.

При розробці веб-додатків важливо вибрати правильні технології, які забезпечать високу продуктивність та зручність розробки. Node.js [19], Express.js та Mongoose є потужними інструментами, які надають великі можливості для створення ефективних та масштабованих веб-додатків. Використання цих технологій дозволяє розробникам використовувати один стек технологій для серверної та клієнтської частин додатку, що спрощує процес розробки та забезпечує консистентність коду.

Далі ми детально розглянемо причини вибору хмарних рішень замість локальних обчислень та переваги використання Node.js, Express.js та Mongoose для розробки сучасних веб-додатків. Це допоможе зробити обґрунтований вибір інструментів та технологій для реалізації проекту автоматичного розпізнавання мовлення.

2.2.1 Причини вибору хмарного рішення замість локальних обчислень

У сучасному світі технології розпізнавання мовлення відіграють важливу роль у багатьох галузях, таких як медицина, освіта, бізнес та інші [24]. Одним з ключових питань при виборі технології розпізнавання мовлення є визначення, чи використовувати локальні обчислення або хмарні рішення. У цьому підпункті буде розглянуто причини вибору хмарного рішення замість локальних обчислень, а також переваги, які надають хмарні сервіси.

Масштабованість і гнучкість

Однією з основних переваг хмарних рішень є їх масштабованість. Хмарні платформи дозволяють підприємствам швидко і легко збільшувати або зменшувати ресурси відповідно до своїх потреб [18]. Це особливо важливо для підприємств, які мають змінні навантаження або планують рости. Локальні обчислення, навпаки, потребують значних інвестицій у серверне обладнання, яке може виявитися надмірним у разі зниження навантаження.

Економічна ефективність

Хмарні рішення зазвичай надають моделі оплати за використання (pay-as-you-go), що дозволяє підприємствам оплачувати лише ті ресурси, які вони фактично використовують. Це знижує початкові витрати на впровадження технології та забезпечує гнучкість у управлінні бюджетом. Локальні обчислення вимагають значних капітальних вкладень у серверне обладнання, ліцензії на програмне забезпечення, а також витрат на технічне обслуговування та оновлення.

Доступність і надійність

Хмарні сервіси надають високу доступність та надійність завдяки використанню розподілених обчислювальних ресурсів. Більшість провайдерів хмарних рішень пропонують рівень доступності не менше 99.9%, що забезпечує безперебійну роботу системи навіть у випадку відмови окремих серверів. Локальні обчислення можуть бути вразливі до апаратних збоїв та інших технічних проблем, що може призвести до простою системи.

Оновлення і підтримка

Хмарні рішення забезпечують автоматичні оновлення та технічну підтримку, що дозволяє підприємствам завжди мати доступ до найновіших технологій та функціональних можливостей. Це звільняє IT-персонал від необхідності вручну оновлювати програмне забезпечення та підтримувати його у належному стані. Локальні обчислення вимагають регулярних оновлень і обслуговування, що може бути трудомістким і затратним процесом.

Інтеграція з іншими сервісами

Хмарні платформи зазвичай пропонують широкий спектр інтеграцій з іншими сервісами та додатками, що дозволяє підприємствам створювати комплексні рішення для своїх потреб. Це включає інтеграцію з іншими хмарними сервісами, такими як бази даних, аналітичні інструменти, системи управління проектами тощо. Локальні системи можуть мати обмежені можливості інтеграції, що ускладнює створення ефективних та зручних у використанні рішень.

Глобальна доступність

Хмарні сервіси забезпечують глобальну доступність, що дозволяє підприємствам використовувати свої ресурси з будь-якого куточка світу. Це особливо важливо для міжнародних компаній, які мають офіси та співробітників у різних країнах. Локальні системи обмежені географічним розташуванням серверів, що може ускладнити доступ до ресурсів для віддалених користувачів.

Швидкий старт і впровадження

Хмарні рішення дозволяють швидко розпочати використання технології без необхідності витрачати час і ресурси на налаштування серверного обладнання та інфраструктури. Це дозволяє підприємствам швидко впроваджувати нові технології та зосередитися на їх основних бізнес-процесах. Локальні обчислення вимагають значного часу на налаштування та впровадження, що може затримати старт проекту.

2.2.2 Вибір технологій для розробки

При розробці сучасних веб-додатків, вибір правильних технологій є критично важливим для забезпечення високої продуктивності, масштабованості та зручності розробки. У цьому підпункті ми розглянемо, чому Node.js, Express.js і mongoose є оптимальними інструментами для розробки веб-додатків.

Node.js [16] є середовищем виконання JavaScript, яке працює на сервері. Його основні переваги включають:

- *Подіє-орієнтована архітектура*: Node.jsвикористовує подіє-орієнтовану [25], неблокуючу I/O модель, що дозволяє обробляти багато одночасних з'єднань без потреби створювати новий потік для кожного з них. Це робить Node.jsідеальним для розробки масштабованих мережевих додатків, де важливою є продуктивність і здатність обробляти великі обсяги запитів одночасно.

- *Висока продуктивність*: Node.js працює на V8 движку від Google, який компілює JavaScript у нативний код, забезпечуючи високу швидкість виконання. Це дозволяє створювати додатки з високою продуктивністю та низькою затримкою, що є критично важливим для реальних часів взаємодії, таких як чати, стрімінгові сервіси та онлайн-ігри.

- *Один стек технологій*: Використання JavaScript як на сервері, так і на клієнті дозволяє розробникам використовувати один і той самий стек технологій по всьому додатку. Це спрощує процес розробки, забезпечує консистентність коду, полегшує навчання нових розробників та зменшує кількість необхідних інструментів і середовищ.

- *Широка екосистема*: Node.js має величезну екосистему пакетів та модулів, доступних через npm (Node Package Manager), що дозволяє легко знайти та використовувати готові рішення для різних задач. Це скорочує час розробки, дозволяючи розробникам зосередитися на унікальних вимогах своїх проектів, а не на розробці базової функціональності з нуля.

- *Підтримка реального часу*: Node.js чудово підходить для розробки додатків реального часу, таких як чати, онлайн-ігри та системи відстеження, завдяки своїй подія-навіженій архітектурі. Це дозволяє створювати швидкі та ефективні рішення для взаємодії з користувачами в реальному часі.

Express.js

Express.js є мінімалістичним та гнучким веб-фреймворком для Node.js, який надає набір інструментів для розробки веб-додатків та API. Основні причини вибору Express.js включають:

- *Простота використання*: Express.js надає простий та інтуїтивно зрозумілий інтерфейс для створення веб-додатків, що робить його легким у освоєнні навіть для новачків. Це забезпечує швидке навчання та зменшує бар'єр входу для розробників.

- *Швидкий старт*: Express.js дозволяє швидко розпочати розробку завдяки простоті налаштування та великій кількості готових модулів та середовищ виконання. Це забезпечує швидкий початок роботи над проектом та зменшує час, необхідний для створення прототипів і MVP (мінімально життєздатних продуктів).

- *Модульність*: Express.js дозволяє використовувати модулі middleware для обробки запитів, що забезпечує гнучкість та можливість розширення функціональності додатка. Це дозволяє додавати нові функції та змінювати існуючі без порушення основної логіки додатку, що підвищує зручність та ефективність розробки.

- *Підтримка маршрутизації*: Express.js надає потужну та гнучку систему маршрутизації для управління різними маршрутами та запитами, що дозволяє легко створювати складні веб-додатки. Це забезпечує можливість налаштування різних маршрутів для обробки різних типів запитів, що є важливим для створення ефективних і зручних у використанні веб-додатків.

- *Активна спільнота*: Express.js має велику та активну спільноту розробників, що забезпечує наявність великої кількості ресурсів, документації та прикладів коду. Це сприяє швидкому вирішенню проблем, обміну знаннями та постійному вдосконаленню фреймворку.

Mongoose

Mongoose є об'єктно-документним мапером (ODM) для MongoDB [9] і надає потужні інструменти для роботи з документами MongoDB у Node.js. Основні переваги Mongoose включають:

- *Схемна модель даних*: Mongoose дозволяє визначати схеми для документів MongoDB, що забезпечує структуру та валідність даних. Це допомагає зберігати дані в структурованому вигляді та уникати помилок при обробці, що є важливим для підтримання цілісності даних.

- *Об'єктно-орієнтований підхід*: Mongoose надає об'єктно-орієнтовані методи для роботи з документами, що полегшує роботу з базою даних та забезпечує зручний API для маніпулювання даними. Це дозволяє розробникам працювати з даними у вигляді об'єктів, що сприяє підвищенню продуктивності та зручності розробки.

- *Можливості валідації*: Mongoose підтримує вбудовану валідацію даних, що дозволяє перевіряти дані перед їх збереженням у базу даних. Це забезпечує збереження коректних та консистентних даних, що є важливим для надійної роботи додатка.

- *Підтримка середовищ подій*: Mongoose дозволяє визначати події, які можна використовувати для виконання певних дій під час різних етапів життєвого циклу документа, таких як створення, оновлення або видалення. Це дозволяє створювати більш гнучкі та потужні рішення для обробки даних.

- *Плагіни та розширення*: Mongoose підтримує плагіни, що дозволяє легко розширювати функціональність і додавати нові можливості без необхідності модифікувати основний код. Це забезпечує гнучкість та можливість адаптації додатка до специфічних потреб.

Таким чином, використання Node.js, Express.js та Mongoose для розробки веб-додатків надає значні переваги, включаючи високу продуктивність, масштабованість, зручність розробки та підтримку роботи з базами даних. Ці інструменти дозволяють створювати ефективні, надійні та масштабовані веб-додатки, що відповідають сучасним вимогам та стандартам.

2.3 Проектування архітектури системи

Проектування системи розпізнавання мовлення визначає загальну структуру, взаємодію між компонентами та забезпечує високий рівень продуктивності, надійності й масштабованості [27]. Ретельне планування архітектури системи дозволяє створити ефективне рішення, яке відповідатиме вимогам сучасних технологій і забезпечуватиме якісне розпізнавання мовлення.

В рамках проектування системи необхідно розглянути основні компоненти архітектури, такі як управління користувачами, завдання розпізнавання мовлення, групи мовлення, мовні файли, медіафайли та транскрипції. Особливу увагу слід приділити інтеграції з хмарними сервісами, такими як Microsoft Azure Cognitive Services [23], для забезпечення високої точності розпізнавання мовлення.

Також важливим аспектом є розробка бази даних, яка забезпечить ефективне збереження, організацію та доступ до розпізнаних текстових даних. Використання NoSQL технологій, зокрема MongoDB, дозволить досягти необхідної продуктивності, гнучкості та масштабованості системи. Структура даних, схема бази даних та оптимізація запитів і продуктивності є ключовими елементами в процесі розробки бази даних.

Таким чином, проектування архітектури системи розпізнавання мовлення включає детальний аналіз і визначення компонентів системи [26], вибір технологій

для реалізації та забезпечення безпеки і захисту даних. Це дозволяє створити надійну та ефективну систему, яка відповідає сучасним вимогам та стандартам у галузі автоматичного розпізнавання мовлення.

2.3.1 Опис архітектури системи розпізнавання мовлення

Розробка системи автоматичного розпізнавання мовлення вимагає ретельного планування та визначення архітектури, яка забезпечить високу продуктивність, надійність і масштабованість. На основі наданої схеми можна розглянути компоненти архітектури, яка інтегрує користувачів, профілі Azure, завдання розпізнавання мовлення, групи мовлення, мовні файли, медіафайли та транскрипції мовлення.

Компоненти архітектури

Користувачі: Система підтримує управління користувачами, дозволяючи створювати нових користувачів, отримувати інформацію про всіх користувачів, оновлювати та видаляти користувачів. Це забезпечує контроль доступу та персоналізацію досвіду користувача.

Azure профілі: Azure профілі використовуються для зберігання інформації про конфігурацію та налаштування, необхідні для взаємодії з Microsoft Azure Cognitive Services API. Це включає створення, отримання, оновлення та видалення профілів.

Завдання розпізнавання (Transcriber Tasks): Завдання розпізнавання мовлення створюються для обробки аудіо файлів. Система дозволяє створювати нові завдання, отримувати інформацію про всі завдання, відстежувати стан конкретного завдання, скасовувати та видаляти завдання. Це забезпечує гнучке управління процесом розпізнавання.

Групи мовлення (Speech Groups): Групи мовлення використовуються для організації мовних файлів у логічні групи. Система підтримує створення нових груп, отримання інформації про всі групи, оновлення та видалення груп.

Мовні файли (Speeches): Мовні файли зберігаються у групах мовлення. Система дозволяє створювати нові мовні файли, отримувати інформацію про всі

мовні файли, перейменовувати та видаляти мовні файли. Це забезпечує структурування та організацію даних.

Медіафайли (Media Files): Медіафайли містять аудіо контент, який потрібно розпізнати. Система підтримує завантаження медіафайлів, отримання інформації про всі медіафайли, перегляд статичних медіафайлів і видалення медіафайлів. Це дозволяє керувати аудіо контентом, який буде оброблятися.

Транскрипції мовлення (Speech Transcriptions): Транскрипції містять розпізнаний текст мовлення. Система дозволяє отримувати всі транскрипції, читати окремі транскрипції, завантажувати транскрипції, редагувати та видаляти транскрипції. Це забезпечує зберігання та управління результатами розпізнавання мовлення.

API Endpoints

API Endpoints (кінцеві точки API) - це специфічні URL-адреси, до яких можна надіслати запит для взаємодії з сервером або додатком через API (Application Programming Interface). Вони визначають, які дані або функції доступні через API, і дозволяють клієнтським програмам виконувати різні операції, такі як створення, отримання, оновлення та видалення даних.

Користувачі:

- *Створення нового користувача*: Метод POST, запит до /users
- *Отримання всіх користувачів*: Метод GET, запит до /users
- *Отримання одного користувача*: Метод GET, запит до /users/userId
- *Оновлення одного користувача*: Метод PATCH, запит до /users/userId
- *Видалення одного користувача*: Метод DELETE, запит до /users/userId

Azure профілі:

- *Створення нового профілю*: Метод POST, запит до /azure-profiles
- *Отримання всіх профілів*: Метод GET, запит до /azure-profiles
- *Отримання одного профілю*: Метод GET, запит до /azure-profiles/azureProfileId
- *Оновлення одного профілю*: Метод PATCH, запит до /azure-profiles/azureProfileId

- *Видалення одного профілю*: Метод DELETE, запит до /azure-profiles/azureProfileId

Завдання розпізнавання (Transcriber Tasks):

- *Створення нового завдання*: Метод POST, запит до /transcriber
- *Отримання всіх завдань*: Метод GET, запит до /transcriber
- *Отримання одного завдання*: Метод GET, запит до /transcriber/taskId
- *Скасування одного завдання*: Метод PATCH, запит до /transcriber/taskId
- *Видалення одного завдання*: Метод DELETE, запит до /transcriber/taskId

Групи мовлення (Speech Groups):

- *Створення нової групи мовлення*: Метод POST, запит до /speech-groups
- *Отримання всіх груп мовлення*: Метод GET, запит до /speech-groups
- *Отримання однієї групи мовлення*: Метод GET, запит до /speech-groups/speechGroupId
- *Оновлення однієї групи мовлення*: Метод PATCH, запит до /speech-groups/speechGroupId
- *Видалення однієї групи мовлення*: Метод DELETE, запит до /speech-groups/speechGroupId

Мовні файли (Speeches):

- *Створення нового мовного файлу*: Метод POST, запит до /speech-groups/speechGroupId/speeches
- *Отримання всіх мовних файлів*: Метод GET, запит до /speech-groups/speechGroupId/speeches
- *Отримання мовного файлу*: Метод GET, запит до /speech-groups/speechGroupId/speeches/speechName
- *Перейменування мовного файлу*: Метод PATCH, запит до /speech-groups/speechGroupId/speeches/speechName
- *Видалення мовного файлу*: Метод DELETE, запит до /speech-groups/speechGroupId/speeches/speechName

Медіафайли (Media Files):

- *Завантаження медіафайлу*: Метод POST, запит до /speech-groups/speechGroupId/speeches/speechName/speech-media-file
- *Отримання всіх медіафайлів*: Метод GET, запит до /speech-groups/speechGroupId/speeches/speechName/speech-media-file
- *Отримання інформації про медіафайл*: Метод GET, запит до /speech-groups/speechGroupId/speeches/speech-media-file/mediaFileName
- *Перегляд статичного медіафайлу*: Метод GET, запит до /speech-groups/speechGroupId/speeches/speechName/static/speech-media-file/mediaFileName
- *Видалення медіафайлу*: Метод DELETE, запит до /speech-groups/speechGroupId/speeches/speechName/speech-media-file/mediaFileName

Транскрипції мовлення (Speech Transcriptions):

- *Отримання всіх транскрипцій*: Метод GET, запит до /speech-groups/speechGroupId/speeches/speechName/vtt
- *Читання транскрипції*: Метод GET, запит до /speech-groups/speechGroupId/speeches/speechName/vtt/vttName
- *Завантаження транскрипції*: Метод GET, запит до /speech-groups/speechGroupId/speeches/speechName/vtt/vttName/download
- *Редагування транскрипції*: Метод PATCH, запит до /speech-groups/speechGroupId/speeches/speechName/vtt/vttName
- *Видалення транскрипції*: Метод DELETE, запит до /speech-groups/speechGroupId/speeches/speechName/vtt/vttName

Інтеграція з Microsoft Azure Cognitive Services API

Система інтегрується з Microsoft Azure Cognitive Services API для виконання функцій розпізнавання мовлення. Використання передових технологій машинного навчання від Microsoft дозволяє забезпечити високу точність розпізнавання мовлення навіть в умовах шуму та різних акцентів.

База даних і мережевий файловий сервер

Система використовує базу даних для зберігання інформації про користувачів, профілі Azure, завдання розпізнавання, групи мов

2.3.2 Розробка бази даних

База даних забезпечує збереження, організацію та доступ до розпізнаних текстових даних. У цьому підпункті буде розглянуто ключові аспекти розробки бази даних для системи розпізнавання мовлення, зосереджуючись на використанні NoSQL технологій, зокрема MongoDB [20], для забезпечення необхідної продуктивності, гнучкості та масштабованості.

Структура даних

Структура даних в базі даних повинна забезпечувати ефективне збереження і доступ до різних типів інформації, таких як аудіо файли, розпізнані текстові дані, метадані про записи та інші пов'язані дані. Основні типи даних, які зберігаються в базі даних, включають:

- *Аудіо файли*: зберігаються у вигляді посилань на файли, які знаходяться у зовнішньому сховищі.

- *Розпізнані текстові дані*: текст, отриманий в результаті обробки аудіо файлів.

- *Метадані*: інформація про записи, така як час і дата запису, тривалість аудіо файлу, мовні параметри, ідентифікатори користувачів та інші атрибути.

Схема бази даних повинна відображати структуру даних і забезпечувати ефективне збереження та доступ до інформації. Нижче представлена прикладова схема бази даних для системи розпізнавання мовлення:

Колекція users:

- *_id:* унікальний ідентифікатор користувача, автоматично генерується MongoDB
- *name:* ім'я користувача
- **email**: електронна пошта користувача
- **created_at**: дата і час створення запису

Колекція audio_files:

- *_id:* унікальний ідентифікатор аудіо файлу, автоматично генерується MongoDB
- *user_id:* ідентифікатор користувача, який створив запис, посилання на колекцію users
- *file_path*: посилання на місцезнаходження файлу
- *duration*: тривалість аудіо файлу
- *created_at*: дата і час створення запису

Колекція transcripts:

- *_id:* унікальний ідентифікатор розшифровки, автоматично генерується MongoDB
- *audio_id*: ідентифікатор аудіо файлу, посилання на колекцію audio_files
- *transcript_text*: розпізнаний текст
- *confidence_score*: рівень впевненості у точності розпізнавання
- *created_at*: дата і час створення запису

Колекція metadata:

- *_id:* унікальний ідентифікатор метаданих, автоматично генерується MongoDB
- *audio_id*: ідентифікатор аудіо файлу, посилання на колекцію audio_files
- *language*: мова аудіо файлу
- *accent:* акцент мовця
- *noise_level:* рівень шуму
- *created_at*: дата і час створення запису

Вибір технологій

Для реалізації бази даних було обрано MongoDB, яка є популярною NoSQL базою даних, що забезпечує високу продуктивність та масштабованість. Основні причини вибору MongoDB включають:

- *Гнучкість*: MongoDB дозволяє зберігати дані у вигляді документів, що забезпечує гнучкість у структурі даних і полегшує роботу з неструктурованими даними.

- *Масштабованість*: MongoDB підтримує горизонтальне масштабування, що дозволяє додавати нові сервери для обробки великих обсягів даних без втрати продуктивності.

- *Продуктивність*: MongoDB забезпечує високу швидкість читання і запису даних завдяки своїй архітектурі, що є важливим для системи розпізнавання мовлення, яка потребує обробки великих обсягів даних у реальному часі.

- *Інтеграція*: MongoDB легко інтегрується з Node.js через бібліотеку mongoose, що забезпечує зручний об'єктно-орієнтований інтерфейс для роботи з базою даних.

Оптимізація запитів і продуктивності

Для забезпечення ефективної роботи бази даних необхідно оптимізувати запити та продуктивність. Це може включати:

- Використання індексів для прискорення пошуку даних.

- Оптимізацію структур даних для зменшення зайвих операцій читання і запису.

- Моніторинг продуктивності бази даних для виявлення потенційних проблем та їх вирішення.

2.4 Висновки до розділу

Цей розділ надав аналіз та обґрунтування вибору технологій та архітектури для створення системи автоматичного розпізнавання мовлення. Проведене дослідження дозволило визначити оптимальні хмарні рішення, інструменти та методи розробки, що забезпечують високу продуктивність, надійність та масштабованість системи.

Було розглянуто різні хмарні сервіси для розпізнавання мовлення, їхні переваги та недоліки. Аналіз показав, що Microsoft Azure Speech-to-Text API є найбільш оптимальним рішенням завдяки своїй високій точності, гнучкості та інтеграційним можливостям.

Вибір технологій та інструментів для розробки був обґрунтований перевагами хмарних рішень над локальними обчисленнями, такими як масштабованість, економічна ефективність та швидкий старт. Використання Node.js, Express.js та Mongoose забезпечує високу продуктивність та зручність розробки, що є важливим для створення сучасних веб-додатків [29].

Проектування архітектури системи включало визначення основних компонентів, таких як управління користувачами, завдання розпізнавання мовлення, групи мовлення, мовні файли, медіафайли та транскрипції. Особливу увагу було приділено інтеграції з Microsoft Azure Cognitive Services для досягнення високої точності розпізнавання мовлення. Було розроблено базу даних на основі MongoDB, що забезпечує необхідну продуктивність та гнучкість.

Таким чином, цей розділ заклав основу для подальшої реалізації проекту, забезпечуючи розуміння вибору технологій та архітектури, необхідних для створення ефективної та надійної системи автоматичного розпізнавання мовлення.

3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Інтеграція з Microsoft Azure Speech-to-Text API

Для того, щоб розроблена система взаємодіяла з хмарними ресурсами обраного API, необхідно виконати підготовчі дії, такі як реєстрація облікового запису, а також при проектуванні своєї розробки необхідно враховувати те, що для будь-якого підприємства система повинна бути багатокористувацькою.

3.1.1 Організація багатокористувацької системи з використанням Azure-профілів

На підприємствах, які займаються різноманітними проектами, часто виникає необхідність у поділі ресурсів і фінансування для кожного окремого проекту. У таких випадках доцільно створити окремі групи користувачів, які відповідатимуть конкретним проектам або їх частинам. Ці групи можуть використовувати різні облікові записи Azure, що дозволяє налаштувати індивідуальні підписки та права доступу до хмарних сервісів.

У нашій системі така структура організації передбачає створення кількох Azure-профілів, кожен з яких може включати різну групу користувачів або навіть декілька груп одночасно. Кожен профіль має власну підписку на використання хмарного API, секретний ключ (або ключі, якщо використовуються основний і резервний), а також вказаний географічний регіон, найближчий до користувачів. Цей підхід забезпечує гнучкість у розподілі ресурсів, що особливо важливо для компаній, які реалізують одночасно кілька незалежних проектів або мають проекти з різним джерелом фінансування.

Наведемо основні аспекти такої системи:

1. Поділ на групи користувачів:

- Користувачі поділяються на групи, залежно від проекту, над яким вони працюють. Група може включати членів кількох команд або підрозділів, які працюють над спільними завданнями.

- Користувачі мають право належати до декількох груп одночасно, якщо їх діяльність охоплює кілька проектів.

2. Профілі Azure для окремих груп:

- Для кожної групи створюється окремий профіль Azure, який включає підписку, секретний ключ і регіон. Профіль не закріплюється за конкретним користувачем, а належить групі користувачів.

- Якщо два або більше проектів мають спільне фінансування, вони можуть користуватися одним і тим самим Azure-профілем. Це дозволяє компанії ефективно керувати бюджетом, уникаючи зайвих витрат на дублювання ресурсів.

3. Організація підписок і ключів доступу:

- Кожен профіль Azure має одну основну та одну резервну пару секретних ключів, які забезпечують доступ до хмарного API. Це дозволяє уникнути простоїв у роботі, якщо один із ключів стане недійсним або буде заблокований.

- Додатково в профілі зберігається інформація про регіон, що забезпечує мінімальні затримки при використанні хмарних сервісів.

4. База даних профілів:

- У базі даних системи зберігається структура всіх профілів Azure, включно з їх секретними ключами доступу та пов'язаними групами користувачів. Такий підхід спрощує адміністрування і дозволяє ефективно контролювати використання хмарних ресурсів.

Перевагами такого підходу є:

- Гнучке управління ресурсами: Завдяки поділу профілів і підписок компанія може точно відстежувати витрати кожного проекту, а також за потреби змінювати структуру груп користувачів і їх фінансування.

- Оптимізація витрат: Використання спільних профілів для груп, які працюють над взаємопов'язаними проектами, дозволяє скоротити витрати на обслуговування хмарних підписок.

- Гарантія безперервності роботи: Наявність резервних ключів і можливість швидкої зміни профілю в разі проблеми з підпискою гарантує стабільність доступу до сервісів.

- Висока масштабованість: Система дозволяє легко додавати нові профілі або групи користувачів у міру зростання потреб підприємства.

Така організація дозволяє створити прозору і зручну систему керування ресурсами для обробки мовлення, яка враховує специфіку хмарного сервісу Azure і потреби компанії. На рис. 3.1 проілюстровано основну структуру цієї системи, показуючи зв'язки між профілями Azure, групами користувачів і їхніми підписками.

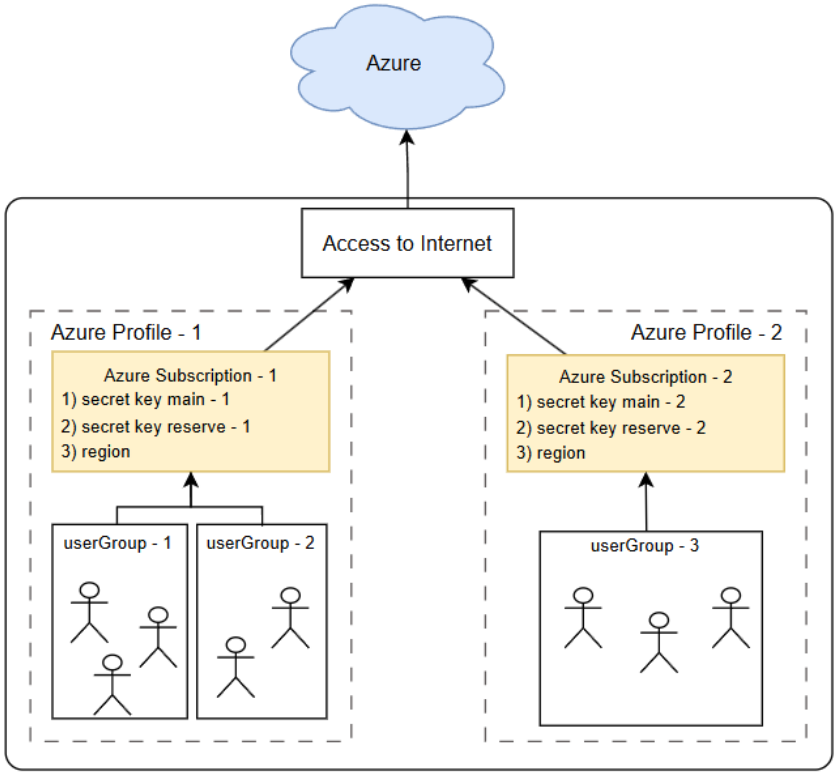


Рис. 3.1. Схема взаємодії користувачів, груп і Azure-профілів у багатокористувацькій системі

3.1.3 Azure-профілі у системі управління доступом

Azure-профілі у системі виконують важливу роль в організації доступу до Azure Speech-to-Text API. Вони являють собою збережені облікові дані та

налаштування, які дозволяють кожному користувачу мати унікальні параметри підключення. Основна мета впровадження Azure-профілів — спростити управління доступами для різних користувачів і забезпечити гнучкість при роботі з різними регіонами чи обліковими записами.

Призначення та структура.

Azure-профіль містить такі основні дані:

- Назва профілю (name): зручна для ідентифікації назва, яку задає користувач.

- Ключ доступу (accountKey): той самий секретний ключ для авторизації запитів до Azure, який був наданий при реєстрації профілю на порталі Azure.

- Регіон облікового запису (accountRegion): географічний регіон, де розташовані сервіси Azure.

- Групи користувачів (userGroupIds): список пов'язаних груп користувачів, які повинні мати доступ до підписки конкретного профілю на порталі Azure.

- Ідентифікатор адміністратора (creatorId): визначає, який користувач створив профіль.

Ця інформація зберігається у базі даних MongoDB, а її структуру визначає модель AzureProfile. Додатково підтримується автоматичне оновлення часу останньої модифікації (updatedAt) для забезпечення актуальності даних.

Схема моделі AzureProfile, що створена за допомогою ORM Mongoose, зображена на рис. 3.2.

```
const azureProfileSchema = new mongoose.Schema({
  name: { type: String,match: /^[a-zA-Z0-9\s_\-\.]{1,64}$/,required: true,},
  accountKey: {type: String,required: true,},
  accountRegion: { type: String, required: true,},
  userGroupIds: [{ type: String,}],
  creatorId: { type: String,},
  description: {type: String,match: /^.{0,1024}$/,default: "",},
  createdAt: {type: Date, default: Date.now, },
  updatedAt: {type: Date,default: Date.now,},
});
const AzureProfile = mongoose.model('AzureProfile', azureProfileSchema);
```

Рис. 3.2. Схема моделі AzureProfile

Azure-профілі дозволяють виконувати базові операції CRUD (створення, читання, оновлення та видалення), що забезпечує гнучке управління. Усі ці операції реалізовані у вигляді REST API [10] з маршрутизацією, яка базується на класі AzureProfilesRouter.

1. Створення профілю

Користувач може створити новий профіль, передаючи необхідні параметри, такі як назва, ключ доступу та регіон. Це дозволяє системі динамічно додавати нові конфігурації для роботи з Azure API.

2. Отримання списку профілів

API підтримує функцію перегляду всіх профілів з можливістю фільтрації за назвами та пагінації для ефективного відображення великих обсягів даних. Це корисно для адміністраторів, які управляють десятками чи сотнями профілів.

3. Оновлення профілю

Система дозволяє оновлювати вже існуючі профілі, наприклад, якщо змінилися ключі доступу або потрібні зміни у пов’язаних мовних групах. При цьому час останнього оновлення автоматично фіксується.

4. Видалення профілю

Для забезпечення безпеки та актуальності даних можна видаляти профілі, які більше не потрібні. Ця операція корисна для уникнення накопичення застарілих облікових даних.

Azure-профілі інтегровані у загальну систему через маршрути /azure-profiles, що описані у AzureProfilesRouter. Основна маршрутизація реалізована через клас IndexRouter, який дозволяє підключати Azure-профілі разом із іншими модулями, такими як маршрути для транскрипції мовлення. Усі операції профілів захищені середовищем автентифікації, яке перевіряє права доступу користувача до цих ресурсів.

Таким чином, Azure-профілі слугують ключовим компонентом системи для ефективного управління доступом до сервісів Azure, спрощуючи їх використання для кінцевих користувачів.

На рис. 3.3 зображено маршрутизацію Azure-профілів у AzureProfilesRouter.

```
const { Router } = require('express');
const AzureProfile = require('../models/AzureProfile.model.js');

class AzureProfilesRouter {
  constructor({middlewares}) {
    this.middlewares = middlewares;
    this.router = new Router();
    this._initRoutes();
  };

  _initRoutes = () => {
    const getRequestUser = this.middlewares.userServiceMiddlewares.getRequestUser;
    this.router.post('/transcriber', this.addNewProfile);
    this.router.get('/transcriber', this.getAllProfiles);
    this.router.get('/transcriber/:azureProfileId', this.getOneProfileById);
    this.router.patch('/transcriber/:azureProfileId', this.updateOneProfile);
    this.router.delete('/transcriber/:azureProfileId', this.removeOneProfile);
  };

  addNewProfile = async (req, res, next) => {/* Створення одного нового профілю */};
  getAllProfiles = async (req, res, next) => {/* Отримання всіх профілів */};
  getOneProfileById = async (req, res, next) => {/* отримання одного профілю */};
  updateOneProfile = async (req, res, next) => {/* оновлення одного профілю */};
  removeOneProfile = async (req, res, next) => {/* видалення одного профілю */};
};
module.exports = AzureProfilesRouter;
```

Рис. 3.3. Маршрутизація Azure-профілів у AzureProfilesRouter

Усі операції з Azure-профілями базуються на принципах безпеки та ролей доступу. Перед виконанням будь-якої дії, наприклад, створення чи видалення профілю, система перевіряє, чи авторизований запитувач (користувач) і чи має він необхідні права. Це реалізується за допомогою проміжного програмного забезпечення (middlewares), такого як getRequestUser. Цей проміжний обробник витягує інформацію про користувача із запиту (зазвичай з токена), перевіряє його достовірність та додає об'єкт користувача в req.user.

Цей механізм дозволяє уникнути несанкціонованих запитів і гарантує, що тільки авторизовані користувачі мають доступ до даних профілів. Наприклад, якщо запитувач не автентифікований, система повертає помилку Forbidden, а якщо ключові параметри, такі як azureProfileId, не надані — BadRequest.

Azure-профілі дозволяють кожному користувачу системи створювати власні облікові дані для підключення до Azure, забезпечуючи гнучкість і ізоляцію.

Наприклад, різні користувачі можуть мати доступ до різних регіонів Azure або працювати в рамках окремих проєктів.

Завдяки зв'язкам між профілями та групами користувачів, система може автоматично визначати, які профілі використовуються для яких сценаріїв. Це спрощує управління ресурсами і забезпечує масштабованість.

Адміністратор може створювати глобальні профілі, які доступні певним групам користувачів. Це зручно у випадках, коли одна група працює з конкретним проєктом або регіоном.

Переваги архітектури

- Гнучкість: підтримка масштабованості завдяки модульній реалізації.

- Безпека: кожен запит перевіряється на авторизацію, що мінімізує ризик витоку даних.

- Оптимізація: використання фільтрів і пагінації допомагає ефективно працювати з великими обсягами даних.

- Простота інтеграції: модулі взаємодіють через стандартизовані API, що дозволяє легко додавати нові функції.

Таким чином, Azure-профілі виконують не лише роль збереження конфігурацій, а й стають ключовим компонентом для інтеграції системи з Azure Speech-to-Text API. Вони дозволяють автоматизувати процеси, організовувати доступ і забезпечувати ефективне управління ресурсами, що особливо важливо у масштабованих проєктах.

3.1.3 Отримання ключів доступу для профілів на порталі Azure

Впровадження профілів Azure у систему дозволяє організувати ефективну взаємодію з хмарними сервісами Microsoft. Як було зазначено раніше, ці профілі слугують інструментом для зберігання параметрів доступу до сервісів, включаючи назву профілю, секретні ключі, регіон і додаткові параметри, необхідні для інтеграції. У базі даних кожен профіль зберігається у вигляді документа, який представляє собою об’єкт із структурованою інформацією. Така архітектура

забезпечує простоту доступу до профілів, можливість їх оновлення та масштабування системи в майбутньому.

Однак для того щоб створити профіль і повноцінно використовувати можливості хмарних сервісів Azure, необхідно виконати кілька підготовчих кроків, серед яких реєстрація в Azure, створення ресурсу для Speech-to-Text API і отримання ключів доступу. Ці дії є обов'язковими, оскільки саме ключі доступу забезпечують ідентифікацію користувача та надають йому можливість здійснювати запити до сервісів.

Реєстрація в Microsoft Azure є першим кроком цього процесу. Для її виконання потрібно перейти на офіційний сайт Azure (https://azure.microsoft.com) і створити обліковий запис. На етапі реєстрації пропонується вибрати тарифний план. Для розробки та тестування найдоцільніше скористатися безкоштовним тарифним планом Free (F0). Цей план надає базовий набір ресурсів, зокрема ліміт у 5000 транзакцій на місяць. Такий обсяг є достатнім для створення прототипів і перевірки функціоналу системи. Водночас слід враховувати, що в умовах масштабної експлуатації, наприклад, у межах організації, ресурси безкоштовного плану можуть бути обмеженими. У таких випадках рекомендується перейти на тариф «Pay-As-You-Go», який дозволяє оплату за фактичне використання ресурсів.

Далі, для можливості використання сервісу розпізнавання мовлення, на порталі Azure у своєму обліковому записі необхідно увімкнути ресурс «Speech-to-Text». При створенні необхідно вказати зрозумілу для себе назву ресурсу, наприклад «my resourse speech», обрати тарифний план, який ми вже підключили та регіон, у якому знаходяться API-сервери, з якими взаємодіє наша система.

Вибір регіону для ресурсу також є важливим моментом. Регіон визначає географічне розташування серверів, на яких буде розміщено ресурси. Це впливає на продуктивність та доступність сервісів. Наприклад, тариф Free (F0) доступний не у всіх регіонах: у регіоні West Europe він недоступний. Тому перед вибором регіону слід переконатися, що потрібний тарифний план підтримується. У подальшому регіон буде використовуватися як один із параметрів профілю Azure.

На рис. 3.4. відображено процес створення ресурсу розпізнавання мовлення.

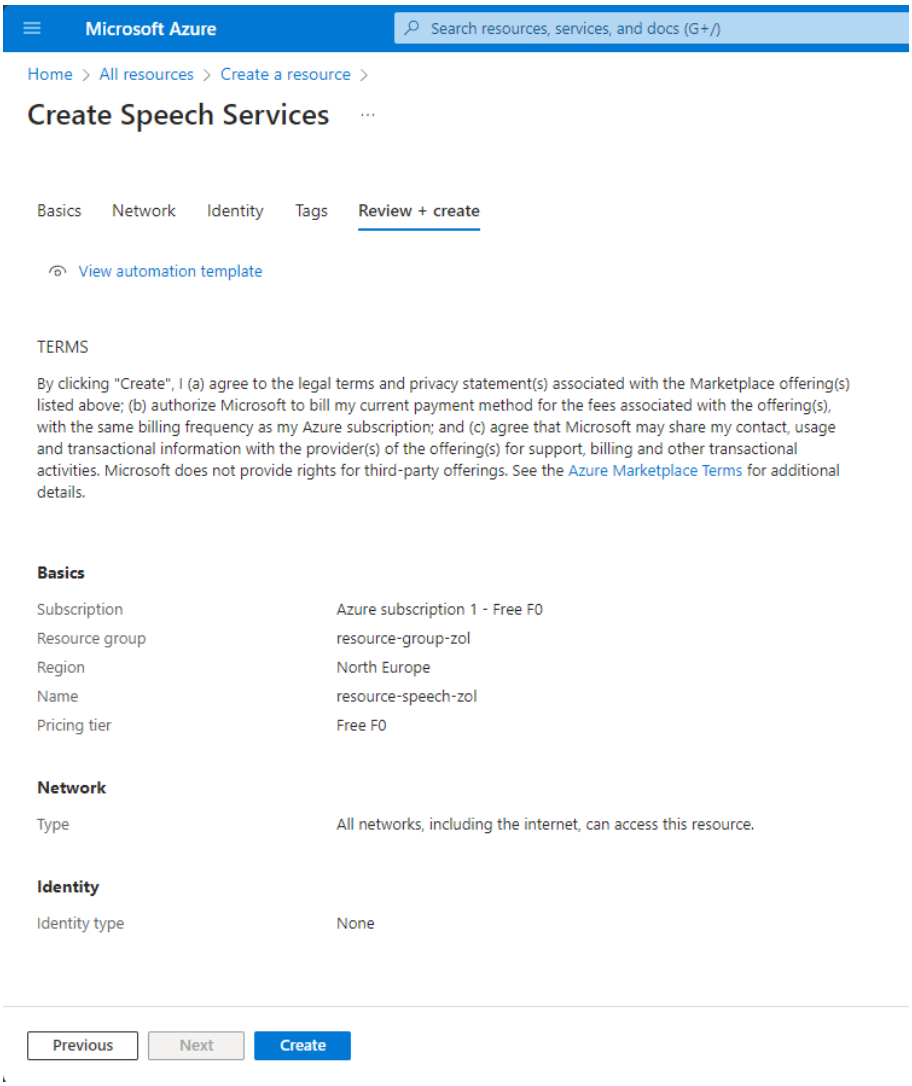


Рис. 3.4. Процес створення ресурсу розпізнавання мовлення на порталі Azure

Після того, як ресурс розпізнавання мовлення на порталі Azure було успішно створено, буде відображено: 1) детальну інформацію створеного ресурсу, що зображено на рис. 3.5; 2) доступні API ендпоінти, що зображені на рис. 3.6, які відносяться до серверів у тому регіоні, якого було обрано; 3) секретні ключі доступу (основний та резервний), що зображені на рис. 3.7, за допомогою яких наш сервіс буде взаємодіяти з хмарними обчисленнями під створеним і налаштованим на порталі обліковим записом.

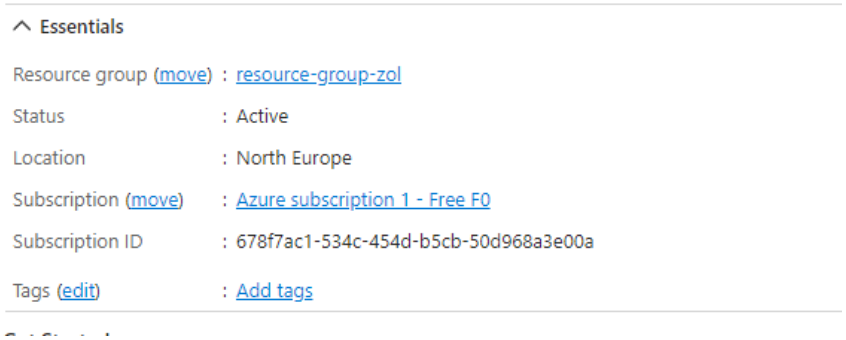


Рис. 3.5. Детальна інформація створеного ресурсу розпізнавання мовлення

Рис. 3.6. Доступні API ендпоінти до створеного ресурсу розпізнавання мовлення.

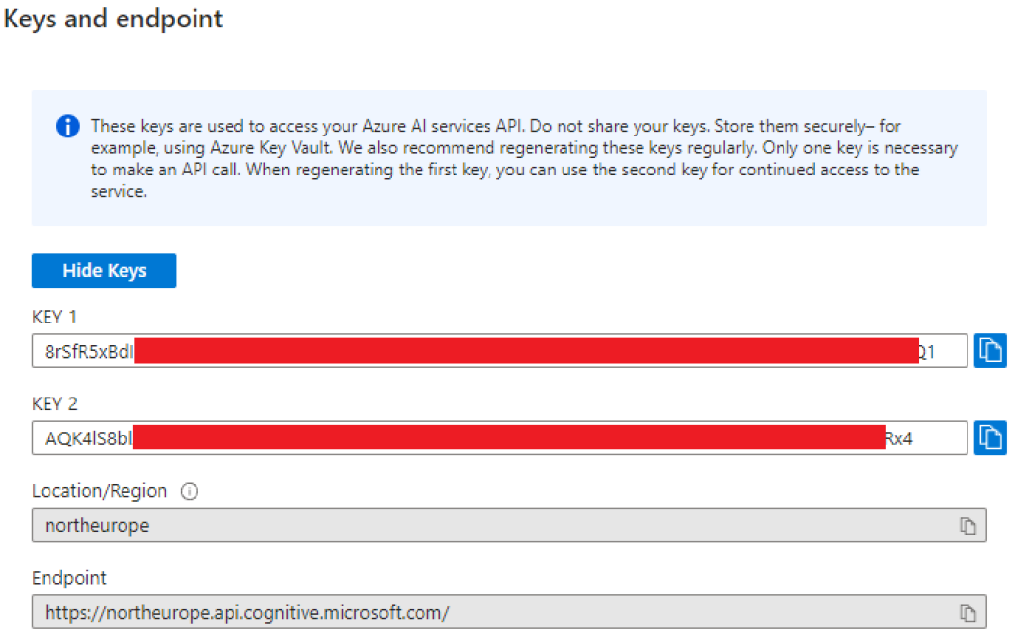


Рис. 3.7. Секретні ключі доступу до створеного ресурсу розпізнавання мовлення на порталі Azure.

Тепер, коли ми маємо секретні ключі, ми вже можемо повноцінно користуватися сервісами за допомогою HTTP-запитів. Тепер можна створити профіль, під яким ми будемо працювати. Для цього виконаємо HTTP запит з методом POST, в якому передаємо назву профілю, секретний ключ та регіон.

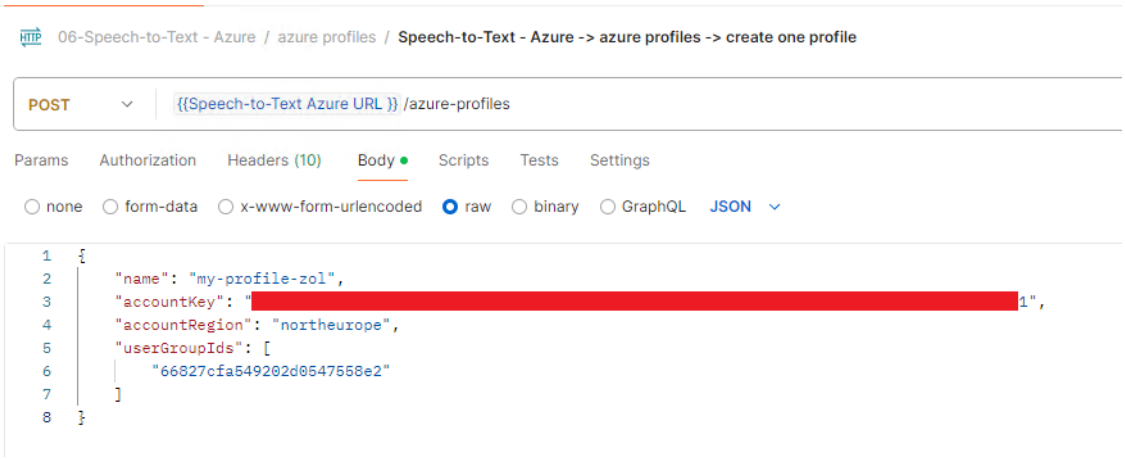


Рис. 3.8. HTTP-запит для створення нового Azure-профілю у розробленій системі

Щоб переконатися, що новий профіль створено, робимо HTTP запит методом GET, передаючи id профілю та отримуємо результат. У відповідь отримаємо ID створеного профілю. Тепер, за допомогою методу GET, можемо отримати повну інформацію створеного профілю, що проілюстровано на рис. 3.9.

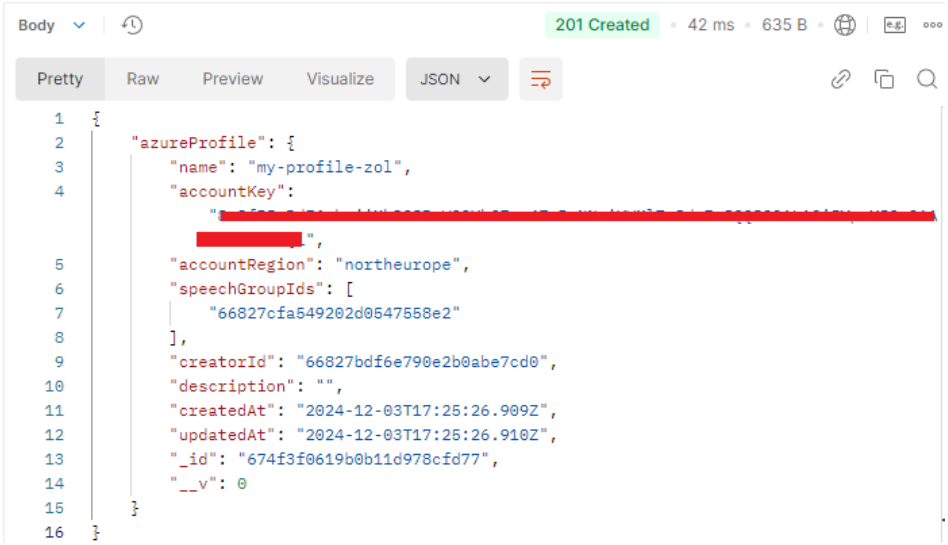


Рис.3.9. Повна інформація створеного Azure-профілю у системі

3.2 Блок розпізнавання мовлення – Transcriber

Блок розпізнавання мовлення Transcriber.js є важливою частиною системи, призначеною для перетворення аудіофайлів у текст [12] за допомогою хмарних сервісів Microsoft Azure. Для взаємодії з Azure використовується бібліотека, яка розроблена розробниками Azure, що має назву «microsoft-cognitiveservices-speech-sdk», яка дозволяє отримати доступ до інтерфейсу розпізнавання мовлення на серверах Azure. Цей блок не лише передає аудіофайли на сервер для обробки, а й отримує текстові результати у форматі, який потім можна використовувати для подальшої обробки, наприклад, для створення субтитрів чи аналізу мовлення.

Механіка розпізнавання мовлення у файлі Transcriber.js зосереджена у функції transcriber, що зображена на рисунку 3.10.

```
const SDK = require("microsoft-cognitiveservices-speech-SDK");

transcribe = async (azureProfile, fileBuffer, name, languages = []) => {
    const {accountKey, accountRegion} = azureProfile;
    let speechConfig = SDK.SpeechConfig.fromSubscription(accountKey, accountRegion);
    speechConfig.setProfanity(SDK.ProfanityOption.Raw);
    let audioConfig = SDK.AudioConfig.fromWavFileInput(fileBuffer, name);
    let speechRecognizer = new SDK.SpeechRecognizer(speechConfig, audioConfig);

    const autoDetectConfig = new SDK.AutoDetectSourceLanguageConfig.fromLanguages(languages);
    speechRecognizer = new SDK.SpeechRecognizer.FromConfig(this.speechConfig, autoDetectConfig, audioConfig);

    speechRecognizer.recognized = onRecognized;
    speechRecognizer.canceled = onCanceled;
    speechRecognizer.speechStartDetected = onSpeechStartDetected;
    speechRecognizer.speechEndDetected = speechEndDetected;
    speechRecognizer.sessionStarted = onSessionStarted;
    speechRecognizer.sessionStopped = onSessionStopped;

    const onStarted = () => console.log(`Continious transcribing by Azure named <${name}> was started.`);
    const onStartingError = (error) => console.error(`StartingError. Details: ${error}`);
    const onStopped = () => console.log(`Continious recognizing named <${name}> was stopped.`);
    const onStoppingError = (error) => console.error(`StoppingError. Details: ${error}`);

    const stop = () => speechRecognizer.stopContinuousRecognitionAsync(onStopped, onStoppingError);
    const start = () => speechRecognizer.startContinuousRecognitionAsync(onStarted, onStartingError);
}
```

Рис. 3.10. Функція transcribe у файлі Transcriber.js

Ключовим моментом у роботі з блоком є правильна ініціалізація з'єднання з хмарними сервісами. Transcriber ініціалізується з параметрами облікового запису користувача за допомогою Azure-профілю, до якого відноситься група користувачів, у якій присутній користувач, який виконує запит. При цьому важливо

зазначити, що для налаштування доступу використовується SDK.SpeechConfig.fromSubscription, що дозволяє налаштувати конфігурацію розпізнавання мовлення. Крім того, блок передбачає налаштування обробки ненормативної лексики через метод setProfanity(SDK.ProfanityOption.Raw), що дає змогу адаптувати систему до різних вимог щодо обробки мовлення.

Однією з головних особливостей Azure є здатність розпізнавати мовлення різними мовами. Для цього в системі передбачено налаштування мови розпізнавання за замовчуванням, а також можливість автовизначення мови, якщо конкретна мова не вказана. Це дозволяє системі працювати в багатомовному середовищі та точно розпізнавати мови, що особливо важливо в умовах глобалізації.

Аудіофайли передаються на сервер Azure у вигляді буферів. Цей процес реалізується через метод SDK.AudioConfig.fromWavFileInput, який приймає аудіофайл у форматі WAV та налаштовує його для подальшого використання. Важливою частиною цього етапу є обробка метаданих аудіофайлу, таких як частота дискретизації та тривалість. В системі використовується бібліотека node-wav, яка дозволяє витягнути необхідну інформацію з файлу WAV. Це дає змогу точно визначити тривалість мовлення, що важливо для обчислення прогресу транскрибування в реальному часі.

Коли аудіофайл налаштовано та готово для передачі на сервер Azure, система здійснює виклик до API, і починається процес розпізнавання мовлення. Сервер Azure проводить детальний аналіз аудіо та повертає текстові результати через спеціальні події. Однією з особливостей Azure є те, що результати розпізнавання надаються не лише у вигляді кінцевого тексту, а й в режимі реального часу, що дає змогу відслідковувати прогрес розпізнавання на різних етапах.

Отримані результати розпізнавання мовлення обробляються відповідно до формату JSON або VTT.

Цей етап включає також обробку подій розпізнавання, таких як recognized, canceled, speechStartDetected, speechEndDetected тощо. Подія recognized спрацьовує, коли система успішно розпізнає частину мовлення і готова передати

текстові дані. Результати можуть бути передані у різних форматах: VTT для субтитрів, SRT для відео або JSON для детальнішого аналізу. Якщо розпізнавання не вдалося, подія canceled обробляє помилки, які виникли під час процесу.

Однією з основних переваг Azure є можливість здійснювати розпізнавання мовлення в режимі реального часу. Блок Transcriber активно використовує події для того, щоб відстежувати прогрес процесу транскрибування. Кожного разу, коли сервер Azure повертає нову частину розпізнаного тексту, ця інформація передається користувачеві. Важливою частиною є обчислення прогресу розпізнавання, яке здійснюється на основі тривалості аудіофайлу та поточної частини тексту.

Це дає змогу забезпечити безперервне оновлення інформації для користувача, що дуже важливо в сценаріях, де необхідно постійно контролювати процес розпізнавання, таких як зйомка відео з субтитрами або інтерактивні системи, що використовують голосові команди.

Окрім підтримки конкретних мов, система також здатна автоматично визначати мову на основі вхідного аудіофайлу. Це забезпечує велику гнучкість у випадках, коли користувач не вказав мову заздалегідь або коли аудіофайл містить декілька мов. В Azure є можливість налаштування конфігурацій для автооприділення мови за допомогою методу SDK.AutoDetectSourceLanguageConfig.fromLanguages, який надає можливість вказати список мов, серед яких Azure може автоматично вибрати одну, що відповідає мовленню на аудіофайлі.

Система також підтримує обробку помилок на всіх етапах. Наприклад, якщо під час розпізнавання виникає помилка, то відразу спрацьовує подія canceled, і система може передати детальну інформацію про помилку через обробник, що виводить відповідні повідомлення для користувача. Важливою частиною є моніторинг подій, що дозволяє своєчасно виявляти проблеми, наприклад, неполадки з якістю аудіо або технічні збої.

У разі необхідності процес можна зупинити, використовуючи методи зупинки розпізнавання. Для цього передбачено спеціальний механізм зупинки процесу, через який користувач може завершити обробку аудіо, не чекаючи її завершення,

або при необхідності відновити транскрипцію з того місця, де вона була зупинена. Це дозволяє користувачам зручно працювати з великими аудіофайлами, зокрема в ситуаціях, коли розпізнавання займає тривалий час.

Загалом, блок розпізнавання мовлення у Transcriber.js є потужним інструментом для інтеграції з хмарними сервісами Microsoft Azure. Завдяки своїй гнучкості, підтримці різних мов і форматів, а також можливості працювати в реальному часі, цей блок є незамінним для створення ефективних систем, які використовують голосове управління, транскрипцію аудіо та інші додатки, що потребують високоточних результатів розпізнавання мовлення.

3.3 Взаємодія між користувачами та Transcriber

Процес взаємодії користувача з розпізнаванням мовлення відбувається за допомогою API, маршрути якого зображені на рис. 3.11.

```
class TranscriberRouter {
  constructor({initData, middlewares}) {
    this.initData = initData;
    this.middlewares = middlewares;
    this.sharedData = this.initData.config.sharedData;
    const SPEECHES_SERVICE_URL = this.initData.config.expressConfig.apiServices.SPEECHES_SERVICE_URL;
    this.speeechesService = new SpeechesService(SPEECHES_SERVICE_URL);
    this.storeTasks = new Map();
    this.router = new Router();
    this._initRoutes();
  };

  _initRoutes = () => {
    const getRequestUser = this.middlewares.userServiceMiddlewares.getRequestUser;

    this.router.post('/', [getRequestUser], this.transcribe);
    this.router.get('/', [getRequestUser], this.getAllTasks);
    this.router.get('/:taskId', [getRequestUser], this.getTaskProcessing);
    this.router.patch('/:taskId/cancel', [getRequestUser], this.cancelTask);
    this.router.delete('/:taskId', [getRequestUser], this.removeTask);
  };
  transcribe = async (req, res, next) => {};
  getAllTasks = async (req, res, next) => {};
  getTaskProcessing = async (req, res, next) => {};
  cancelTask = async (req, res, next) => {};
  removeTask = async (req, res, next) => {};
};
module.exports = TranscriberRouter;
```

Рис. 3.11. Маршрутизація HTTP-запитів між користувачами та Transcriber

Цей процес включає кілька етапів, на кожному з яких перевіряються дані, здійснюються необхідні операції та оновлюється статус завдання транскрибування. У результаті користувач отримує готовий текстовий файл, що містить транскрибований текст.

1. Запит на транскрибування

Все починається з того, що користувач надсилає запит на транскрибування медіафайлу. Цей запит містить важливі параметри, такі як userId, і mediaFile. За допомогою id користувача система знає до якої групи він належить, а отже і профілю Azure. Параметр mediaFile – це аудіо файл від користувача. Якщо ці параметри не надані або передані некоректно, система негайно повертає помилку 400, інформуючи користувача про необхідність заповнення всіх полів запиту.

2. Перевірка авторизації

Наступним етапом є перевірка авторизації користувача. Система аналізує, чи є користувач авторизованим для виконання даної операції. Якщо користувач не авторизований або не має прав доступу, система автоматично повертає помилку 403. Це важливий крок для забезпечення безпеки і недопущення несанкціонованого доступу до ресурсів транскрибування.

3. Перевірка параметрів запиту

Якщо користувач авторизований, система переходить до перевірки наданих параметрів запиту. У цьому кроці система перевіряє наявність всіх необхідних полів. Якщо хоча б одне з цих значень полів відсутнє, система негайно повертає помилку 400, що означає, що запит сформовано неправильно. Це гарантує, що всі необхідні дані передаються перед тим, як процес транскрибування почнеться.

4. Пошук Azure профілю та права на транскрибування

Далі система перевіряє, чи існує відповідний профіль Azure для груп, у яких знаходиться користувач. Цей профіль є необхідним для підключення до сервісу транскрибування, який буде обробляти медіафайл. Якщо профіль не знайдено, система видає помилку 404, що означає, що для вказаної групи мовлення не знайдений відповідний профіль.

Після успішного знаходження Azure профілю система перевіряє, чи має користувач необхідні права для виконання транскрибування на цей медіафайл. Це важливий етап, оскільки не кожен користувач може мати доступ до всіх медіафайлів або груп мовлення. Якщо у користувача немає прав на транскрибування для вказаного файлу, система повертає помилку 403, що означає відмову в доступі до ресурсу.

6. Отримання медіафайлу та створення завдання

Якщо всі попередні перевірки пройдено успішно, система отримує інформацію про медіафайл. Це включає перевірку його наявності, розмір, формат та інші метадані. Після цього система генерує унікальну назву для файлу результату транскрибування та створює запис у базі даних, що містить всі деталі завдання: ідентифікатор користувача, групи мовлення, назву медіафайлу, мовні налаштування та інші дані.

7. Запуск процесу транскрибування

Процес транскрибування запускається за допомогою функції transcribe з файлу Transcriber.js, що використовує профіль Azure для транскрибування медіафайлу. Для цього передається сам медіафайл, а також параметри, що визначають, які мови потрібно обробити. На цьому етапі система починає обробляти файл і генерувати транскрипцію. Статус завдання змінюється на "PROCESS", і користувач отримує повідомлення про початок обробки.

8. Оновлення результатів та прогресу

У процесі транскрибування система постійно оновлює прогрес завдання та часткові результати. Кожного разу, коли з'являються нові частини транскрипту, вони зберігаються в тимчасовому файлі і оновлюється прогрес. Користувач може спостерігати за цим процесом через систему, де отримує оновлення про поточний стан завдання. Це дозволяє відслідковувати хід роботи в реальному часі.

9. Завершення процесу транскрибування

Коли транскрибування завершено, система зберігає результат у фінальний файл у форматі .vtt. Якщо ж під час транскрибування виникла помилка, система оновлює статус завдання на "ERROR", і користувач отримує повідомлення про

помилку. Після цього завдання завершується, і користувач отримує доступ до файлу з результатами, якщо транскрибування було успішним.

Наведені етапи розпізнавання мовлення візуалізовано через блок-схему на рис. 3.12.

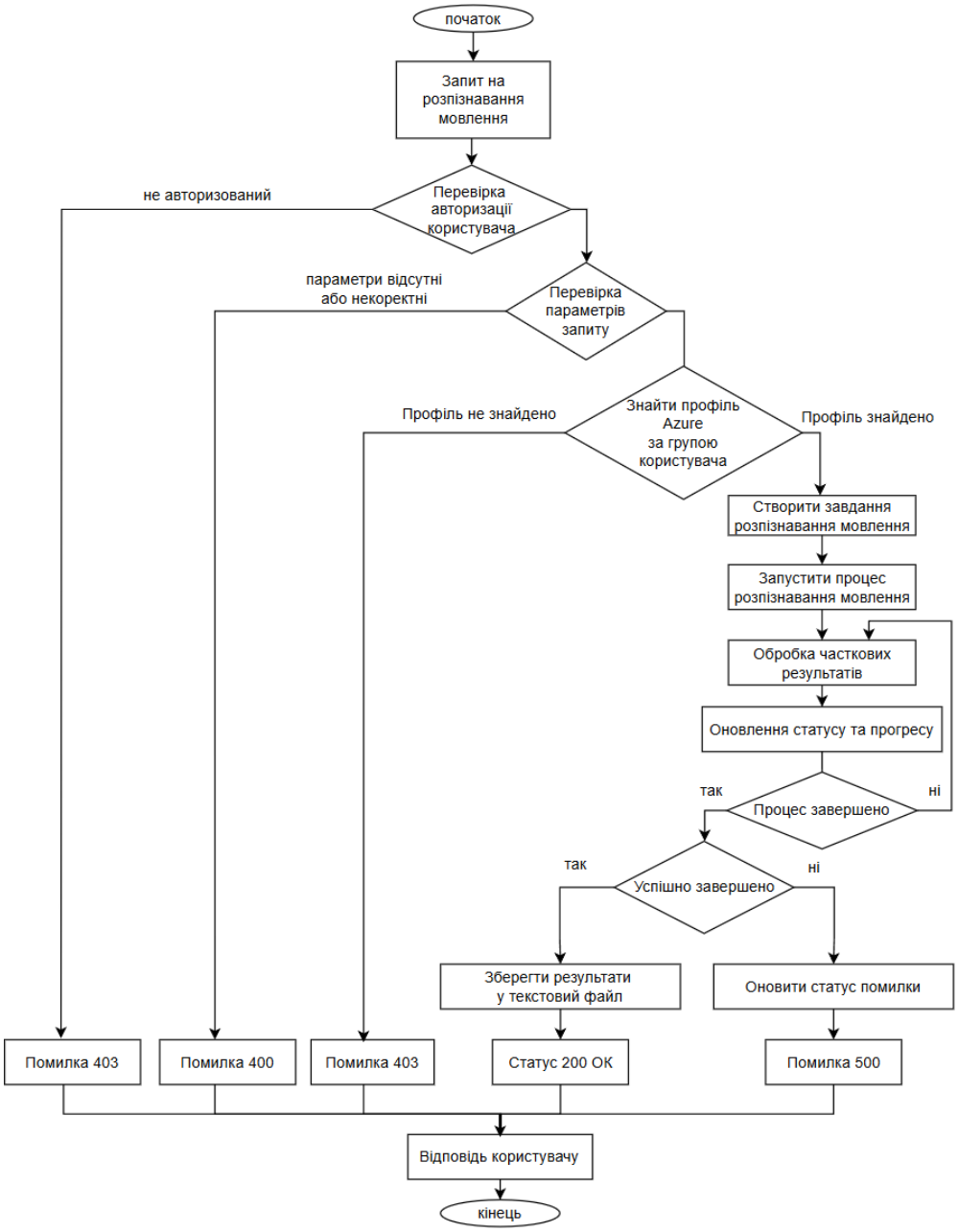


Рис. 3.12. Блок схема етапів розпізнавання при взаємодії миж користувачем та Transcriber

Загальну схему розробки зображено на рис. 3.13.

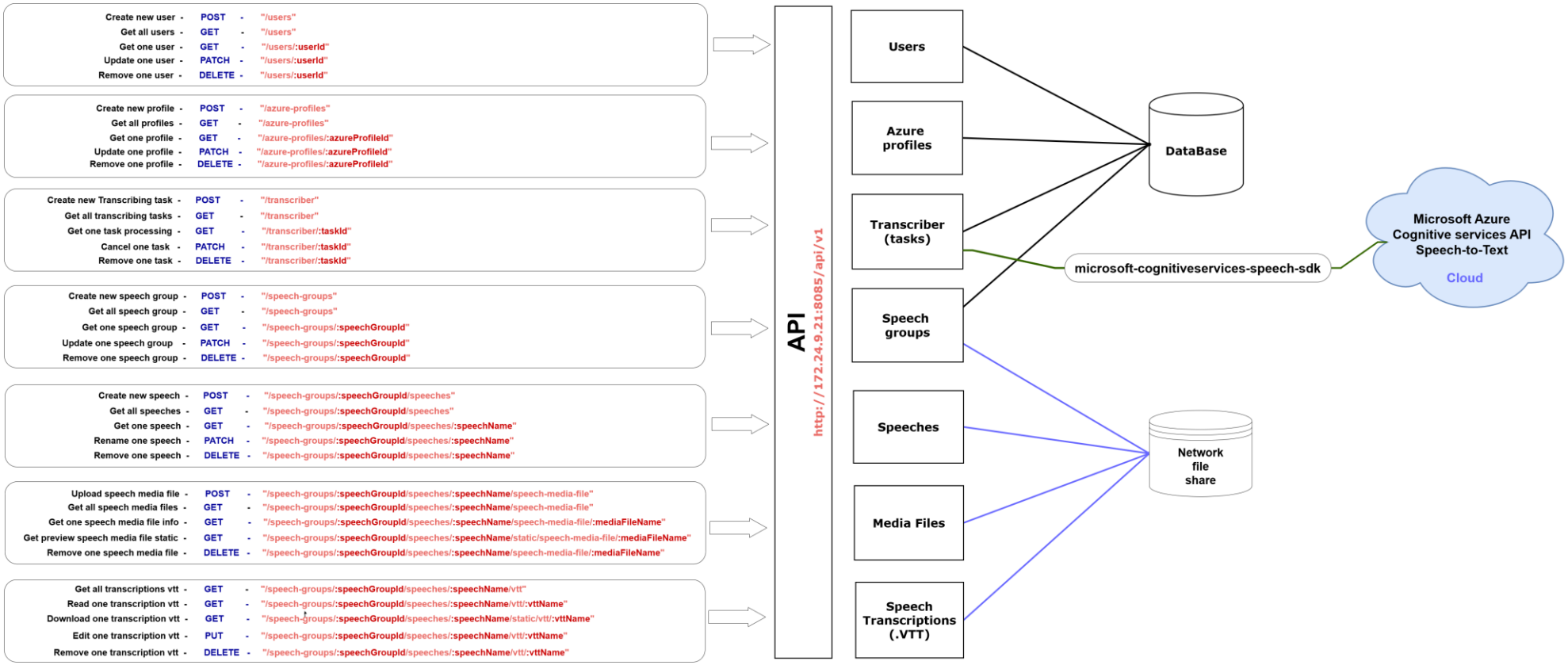


Рис. 3.14. Загальна схема розробленого багатокористувацького продукту для розпізнавання мовлення з використанням хмарних ресурсів за допомою API

3.4.1 Методологія тестування

Для оцінки ефективності розробленої багатокористувацької системи для розпізнавання мовлення було розроблено детальну методологію тестування, що охоплює кілька ключових аспектів [5]:

Мета тестування

Мета полягає у перевірці стабільності системи, її продуктивності при різній кількості одночасно оброблюваних файлів, а також у визначенні впливу зовнішніх факторів, таких як швидкість інтернет-з’єднання, на якість і швидкість роботи системи.

Параметри тестування

1. Обсяг тестових даних: використовувалися аудіофайли тривалістю по 5 хвилин, які містять чітко сформульовану мову з мінімальним фоновим шумом.

2. Сценарії тестування:

Обробка одного файлу.

Одночасна обробка трьох файлів.

Одночасна обробка п'яти файлів.

Одночасна обробка десяти файлів.

3. Метрики продуктивності:

- *Час обробки* — тривалість, необхідна для розпізнавання кожного файлу.

- *Точність розпізнавання* — оцінюється через метрику Word Error Rate (WER).

- *Стабільність системи* — визначається відсутністю збоїв або значного падіння продуктивності при збільшенні навантаження.

Інструменти та умови тестування

1. Сервер: Microsoft Azure Speech-to-Text API.

2. Клієнтська частина: багатокористувацька система, розроблена на основі Node.js.

3. Інтернет-з'єднання: швидкість 1 Гбіт/с.

4. Клієнтські запити: надсилалися через REST API із заздалегідь підготовлених профілів користувачів.

Процес тестування

1. Для кожного сценарію тестування запускалися аудіофайли одночасно.

2.Вимірювався час початку обробки ("Старт") і час завершення ("Фініш") кожного файлу.

3.Розраховувався час обробки файлів шляхом віднімання часу старту від часу фінішу.

4.Проводився аналіз отриманих транскриптів для оцінки точності розпізнавання за допомогою WER.

5. Результати кожного сценарію документувалися у вигляді таблиць

3.4.2 Результати тестування одночасного розпізнавання декількох файлів

Результати тестування показали стабільність роботи системи при обробці різної кількості файлів одночасно, результати зафіксовані у таблицях 3.1, 3.2, 3.3 та 3.4.

Таблиця 3.1

Результати обробки одного файлу

Обробка одного файлу					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	20:00:47	20:03:46	0:02:59	0,02

Таблиця 3.2

Результати обробки трьох файлів одночасно

Обробка двох файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	20:05:00	20:07:58	0:02:58	0,02
Name_2	5 хв	20:05:02	20:08:00	0:02:58	0,02
Name_3	5 хв	20:05:03	20:08:02	0:02:59	0,02

Таблиця 3.3

Результати обробки п’яти файлів одночасно

Обробка п'яти файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	20:09:48	20:12:46	0:02:58	0,02
Name_2	5 хв	20:09:50	20:12:48	0:02:58	0,02
Name_3	5 хв	20:09:52	20:12:49	0:02:57	0,02
Name_4	5 хв	20:09:54	20:12:51	0:02:57	0,02
Name_5	5 хв	20:09:56	20:12:53	0:02:57	0,02

Таблиця 3.4

Результати обробки п’яти файлів одночасно

Обробка десяти файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	20:24:22	20:27:16	0:02:54	0,02
Name_2	5 хв	20:24:24	20:27:18	0:02:54	0,02
Name_3	5 хв	20:24:25	20:27:19	0:02:54	0,02
Name_4	5 хв	20:24:27	20:27:21	0:02:54	0,02
Name_5	5 хв	20:24:29	20:27:23	0:02:54	0,02
Name_6	5 хв	20:24:31	20:27:25	0:02:54	0,02
Name_7	5 хв	20:24:32	20:27:27	0:02:55	0,02
Name_8	5 хв	20:24:34	20:27:28	0:02:54	0,02
Name_9	5 хв	20:24:35	20:27:29	0:02:54	0,02
Name_10	5 хв	20:24:36	20:27:31	0:02:55	0,02

Нижче наведені ключові результати:

Обробка одного файлу

- *Час обробки***:** середній час обробки одного файлу тривалістю 5 хвилин становить 2 хвилини 59 секунд.

- *Точність розпізнавання:* WER = 0,02.

Обробка трьох файлів одночасно

- *Час обробки:* середній час обробки кожного з трьох файлів — 2 хвилини 58 секунд.

- *Точність розпізнавання:* WER = 0,02 для всіх файлів.

- *Висновок:* система показала однаковий час обробки незалежно від одночасної кількості оброблюваних файлів у цьому сценарії.

Обробка п'яти файлів одночасно

- *Час обробки:* середній час обробки становить від 2 хвилин 57 секунд до 2 хвилин 58 секунд для кожного файлу.

- *Точність розпізнавання:* WER = 0,02 для всіх файлів.

- *Висновок:* система залишалася стабільною і не зазнала зниження швидкості або точності.

Обробка десяти файлів одночасно

- *Час обробки:* середній час становить 2 хвилини 54 секунди для більшості файлів.

- *Точність розпізнавання:* WER = 0,02 для всіх файлів.

- *Висновок:* система успішно справляється з обробкою десяти файлів одночасно без зниження продуктивності чи точності.

Загальний висновок

Усі сценарії показали стабільну роботу системи. Незначна варіативність у часі обробки пояснюється особливостями розподілу серверних ресурсів Microsoft Azure Speech-to-Text API.

3.4.3 Вплив швидкості інтернет-з'єднання на ефективність

Для перевірки впливу швидкості інтернет-з’єднання були проведені додаткові тести. У межах тестування використовувалися наступні сценарії:

Швидкість 1 Гбіт/с (основний тест): показала стабільну роботу системи з часом обробки в середньому 2 хвилини 54 секунди на файл, що відображено в таблицях 3.1, 3.2, 3.3 та 3.4.

Швидкість 100 Мбіт/с: незначне збільшення часу обробки — до 3 хвилин 5 секунд. При цьому точність розпізнавання залишалася незмінною (WER = 0,02), що відображено в таблицях 3.5, 3.6, 3.7 та 3.8.

Таблиця 3.5

Результати обробки одного файлу зі швидкістю інтернет-з’єднання 100 Мбіт/с

Обробка одного файлу					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	21:04:28	21:07:33	0:03:05	0,02

Таблиця 3.6

Результати обробки трьох файлів одночасно зі швидкістю інтернет-з’єднання 100 Мбіт/с

Обробка трьох файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	21:12:11	21:15:16	0:03:05	0,02
Name_2	5 хв	21:12:12	21:15:16	0:03:04	0,02
Name_3	5 хв	21:12:14	21:15:19	0:03:05	0,02

Таблиця 3.7

Результати обробки трьох файлів одночасно зі швидкістю інтернет-з’єднання 100 Мбіт/с

Обробка п'яти файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	21:29:43	21:32:47	0:03:04	0,02
Name_2	5 хв	21:29:45	21:32:50	0:03:05	0,02
Name_3	5 хв	21:29:47	21:32:51	0:03:04	0,02
Name_4	5 хв	21:29:48	21:32:53	0:03:05	0,02
Name_5	5 хв	21:29:50	21:32:55	0:03:05	0,02

Таблиця 3.8

Результати обробки трьох файлів одночасно зі швидкістю інтернет-з’єднання 100 Мбіт/с

Обробка десяти файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	21:48:42	21:51:46	0:03:04	0,02
Name_2	5 хв	21:48:44	21:51:49	0:03:05	0,02
Name_3	5 хв	21:48:46	21:51:50	0:03:04	0,02
Name_4	5 хв	21:48:48	21:51:53	0:03:05	0,02
Name_5	5 хв	21:48:50	21:51:55	0:03:05	0,02
Name_6	5 хв	21:48:52	21:51:57	0:03:05	0,02
Name_7	5 хв	21:48:54	21:51:59	0:03:05	0,02
Name_8	5 хв	21:48:56	21:52:00	0:03:04	0,02
Name_9	5 хв	21:48:58	21:52:03	0:03:05	0,02
Name_10	5 хв	21:49:00	21:52:05	0:03:05	0,02

Швидкість 10 Мбіт/с: спостерігалося подовження часу обробки до 3 хвилин 45 секунд на файл. Точність розпізнавання не змінилася, що відображено в таблицях 3.9, 3.10, 3.11 та 3.12.

Таблиця 3.9

Результати обробки одного файлу зі швидкістю інтернет-з’єднання 10 Мбіт/с

Обробка одного файлу					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	23:17:09	23:20:53	0:03:44	0,02

Таблиця 3.10

Результати обробки трьох файлів одночасно зі швидкістю інтернет-з’єднання 10 Мбіт/с

Обробка трьох файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	23:25:09	23:28:53	0:03:44	0,02
Name_2	5 хв	23:25:11	23:28:56	0:03:45	0,02
Name_3	5 хв	23:25:13	23:28:58	0:03:45	0,02

Таблиця 3.11

Результати обробки п’яти файлів одночасно зі швидкістю інтернет-з’єднання 10 Мбіт/с

Обробка п'яти файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	23:31:23	23:35:08	0:03:45	0,02
Name_2	5 хв	23:31:25	23:35:10	0:03:45	0,02
Name_3	5 хв	23:31:27	23:35:12	0:03:45	0,02
Name_4	5 хв	23:31:29	23:35:14	0:03:45	0,02
Name_5	5 хв	23:31:31	23:35:15	0:03:44	0,02

Таблиця 3.12

Результати обробки десяти файлів одночасно зі швидкістю інтернет-з’єднання 10 Мбіт/с

Обробка десяти файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	23:47:48	23:51:33	0:03:45	0,02
Name_2	5 хв	23:47:50	23:51:34	0:03:44	0,02
Name_3	5 хв	23:47:52	23:51:37	0:03:45	0,02
Name_4	5 хв	23:47:54	23:51:39	0:03:45	0,02
Name_5	5 хв	23:47:56	23:51:41	0:03:45	0,02
Name_6	5 хв	23:47:58	23:51:43	0:03:45	0,02
Name_7	5 хв	23:48:00	23:51:44	0:03:44	0,02
Name_8	5 хв	23:48:02	23:51:46	0:03:44	0,02
Name_9	5 хв	23:48:04	23:51:49	0:03:45	0,02
Name_10	5 хв	23:48:06	23:51:51	0:03:45	0,02

Висновки

Точність роботи системи (WER) не залежить від швидкості інтернет-з’єднання в умовах достатнього рівня стабільності каналу.

Зменшення швидкості інтернет-з’єднання призводить до збільшення часу передачі даних на сервер і, відповідно, до подовження загального часу обробки файлів.

Для досягнення найкращої продуктивності рекомендовано використовувати швидкість інтернет-з’єднання не менше 100 Мбіт/с.

3.5 Висновки до розділу

Результати тестування продемонстрували, що розроблена система є стабільною, продуктивною і точною. Вона успішно справляється з обробкою від одного до десяти файлів одночасно без втрати якості розпізнавання. Вплив швидкості інтернет-з’єднання обмежується лише часом обробки, тоді як точність залишається стабільною навіть при повільних з'ядднаннях. Таким чином, система відповідає заявленим цілям і може бути використана для багатокористувацького розпізнавання мовлення на основі хмарних API.

ВИСНОВОК

У процесі виконання дипломної роботи було розроблено клієнтську програму для перетворення мовлення в текст із використанням хмарних API, що є сучасним і перспективним напрямом у галузі інформаційних технологій. Проведено детальний аналіз сучасних технологій розпізнавання мовлення, включаючи провайдерів Microsoft Azure, OpenAI та Google Cloud. Було розглянуто ключові особливості кожного API, включно з точністю, швидкістю обробки, масштабованістю та вартістю послуг. Розроблений програмний продукт забезпечує підтримку багатокористувацького доступу [2], дозволяючи різним категоріям користувачів працювати з системою на основі їхніх прав доступу. Цей функціонал є важливим для забезпечення гнучкості та безпеки використання системи в багатокористувацькому середовищі.

Під час тестування програмного продукту було підтверджено високу ефективність використання хмарних рішень для розпізнавання мовлення. Система демонструє стабільну роботу, адаптуючись до різних типів аудіо вхідних даних, і дозволяє значно зекономити час порівняно з традиційними методами ручного транскрибування. Ця розробка має потенціал для впровадження в реальних умовах, зокрема в освітніх установах, бізнесі та сервісах клієнтської підтримки.

У рамках дипломної роботи було успішно досягнуто всіх поставлених цілей, що включають, таких як: проведення аналізу сучасних хмарних рішень для розпізнавання мовлення, визначення їх переваг і недоліків; створення програмного продукту з підтримкою багатокористувацького доступу [8], що дозволяє налаштовувати права доступу залежно від потреб користувачів; проведення порівняльного аналізу API Microsoft Azure, OpenAI та Google Cloud за ключовими показниками, включно з точністю розпізнавання, швидкістю обробки та вартістю; розробка системи, яка дозволяє інтегрувати функціонал API для забезпечення максимального рівня точності та адаптивності до різних типів завдань.

Результати дослідження підтвердили, що розроблений програмний продукт відповідає сучасним вимогам і має великий потенціал для подальшого вдосконалення.

Для подальшого розвитку даного проекту пропонується:

Розширення підтримки додаткових API: Додати інтеграцію з іншими провайдерами API для розпізнавання мовлення, що дозволить підвищити універсальність системи.

Обробка мультимедійних файлів: Реалізувати можливість попереднього аналізу якості аудіо [7] та автоматичного вибору найбільш підходящого провайдера для обробки конкретного завдання.

Автоматичний вибір API: Інтегрувати функціонал автоматичного аналізу вхідних даних для визначення найефективнішого API [4] залежно від параметрів, таких як якість аудіо та вимоги до точності.

Покращення безпеки: Додати двофакторну аутентифікацію [6] та інші механізми захисту для забезпечення максимальної безпеки облікових записів користувачів із високим рівнем доступу.

Мобільна версія: Розробити мобільний додаток, що дозволить користувачам отримувати доступ до системи з будь-якого пристрою, підвищуючи зручність і доступність продукту.

ПЕРЕЛІК ПОСИЛАНЬ

1. OpenAI Platform. Docs. Speech to text. Supported languages: https://platform.openai.com/docs/guides/speech-to-text/supported-languages

2. Necula, Sabina. (2024). Exploring The Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications.

3. A. Seagraves. Benchmarking Top Open Source Speech Recognition Models: Whisper, Facebook wav2vec2, and Kaldi.

4. Murge, Lokesh & Sampangi, Sandhya. (2024). REST AS A WEB ARCHITECTURAL DESIGN 1.

5. Kore, Pratik & Lohar, Mayuresh & Surve, Madhusudan & Jadhav, Snehal. (2022). API Testing Using Postman Tool.

6. Hunt, John. (2023). Pip and Conda Virtual Environments.

7. Bredin, Hervé. (2023). pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe.

8. Desai, Krutika & Fiaidhi, Jinan. (2022). Developing a Social Platform using MERN Stack.

9. Desai, Krutika & Fiaidhi, Jinan. (2022). Developing a Social Platform using MERN Stack.

10. Murge, Lokesh & Sampangi, Sandhya. (2024). REST AS A WEB ARCHITECTURAL DESIGN 1.

11. A. Vaswani et al., “Attention is all you need,” Advances in neural information processing systems, 2017.

12. A. Radford, J. W. Kim, et al. (2022). Robust Speech Recognition via Large-Scale Weak Supervision.

13. OpenAI Platform. Docs. Speech to text. Supported languages: https://platform.openai.com/docs/guides/speech-to-text/supported-languages

14. Mikolov, Tomas et al. (2013). Efficient Estimation of Word Representations in Vector Space.

15. Microsoft Azure Docs. (2024). Speech Service Documentation.

16. Tilkov, Stefan, & Vinoski, Steve. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs.

17. Google Cloud's AI and Machine Learning services - https://cloud.google.com/products/ai/

18. Microsoft Azure's Scalability and Performance - https://azure.microsoft.com/en-us/features/scalability/

19. Node.js Official Documentation - https://nodejs.org/en/docs/

20. MongoDB Official Documentation - https://www.mongodb.com/docs/

21. Microsoft Azure Speech Service - https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/

22. Journal of Speech, Language, and Hearing Research - https://pubs.asha.org/journal/jslhr

23. Microsoft Azure Cognitive Services - https://azure.microsoft.com/en-us/services/cognitive-services/

24. IEEE Access Journal on Speech Recognition - https://ieeexplore.ieee.org/document/8485823

25. Node.js Design Philosophy - https://nodejs.org/en/about/

26. Speech Communication Journal - https://www.journals.elsevier.com/speech-communication

27. Software Architecture for Developers - https://www.oreilly.com/library/view/software-architecture-patterns/

28. Understanding WER in Speech Recognition - https://ieeexplore.ieee.org/document/6237557

29. Modern Web Development - https://www.oreilly.com/library/view/modern-web-development/

30. Advantages of Cloud Computing - https://dl.acm.org/doi/10.1145/3371396

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інтеграція Speech-to-Text API в багатокористувацькі системи для розпізнавання тексту»

Виконав: здобувач вищої освіти гр. ІСДМ-63
Віктор Золочевський

Керівник: Андрій БОНДАРЧУК
д.т.н., професор

Київ, 2024

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

Актуальність

Попит на точні та швидкі системи розпізнавання мовлення в сучасному світі зростає

Обчислювальні ресурси	Економічна ефективність	Масштабованість	Інноваційний підхід	Технологічні переваги
Хмарні сервіси надають потужні ресурси для обробки мовлення навіть в умовах шуму, різних акцентів та діалектів	Зниження витрат на інфраструктуру та оплата за використання	Легке масштабування залежно від потреб організації	Автоматизація процесів за допомогою розпізнавання мовлення сприяє оптимізації робочих процесів	Використання передових технологій дозволяє значно підвищити ефективність і продуктивність організацій

Інтеграція хмарних API забезпечує високі стандарти якості та доступність сервісів

Метою розробка клієнтської програми для автоматичного перетворення мовлення в текст із використанням хмарних API, таких як Microsoft Azure, OpenAI та Google Cloud, для забезпечення високої точності, швидкості та зручності.

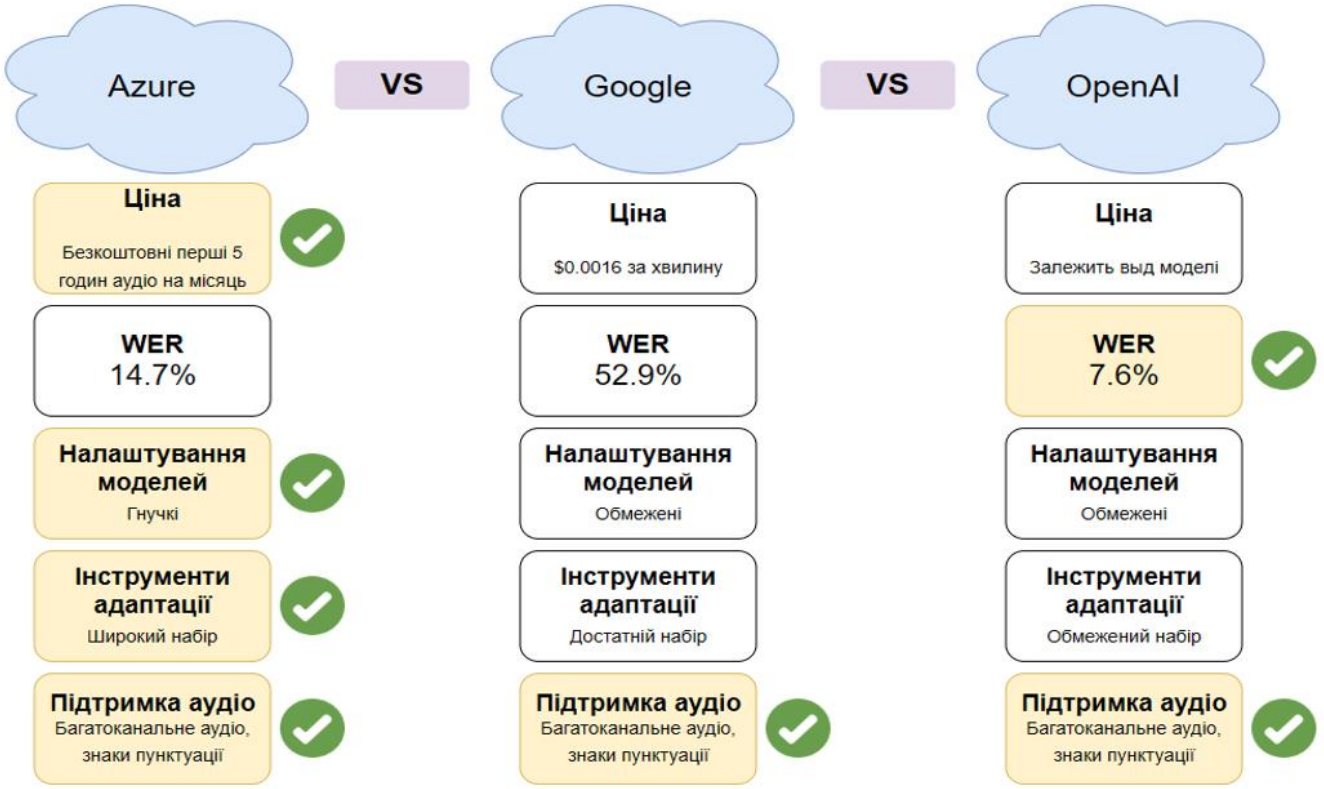
Об'єкт дослідження – технології автоматичного розпізнавання мовлення.

Предмет дослідження – методи інтеграції, порівняння та використання хмарних API для створення багатокористувацької клієнтської програми.

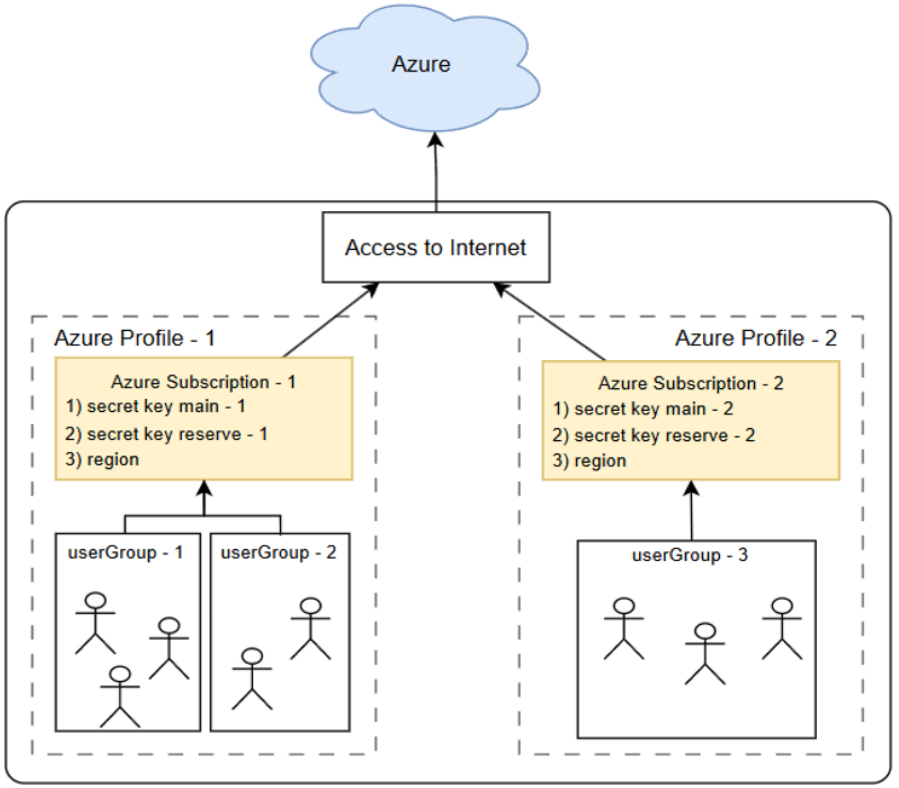
Основні завдання:

- Проаналізувати існуючі хмарні рішення для розпізнавання мовлення;
- Визначити критерії порівняння, на основі яких провести вибірковий аналіз сучасних хмарних рішень дял розпізнавання мовлення;
- На основі обраного хмарного сервісу побудувати багатокористувацьку платформу для автоматичного розпізнавання мовлення аудіо файлів, з можливістю її використання в різних галузях;
- Зробити висновки щодо ефективності розробленої платформи та надати рекомендації для її подальшого вдосконалення.

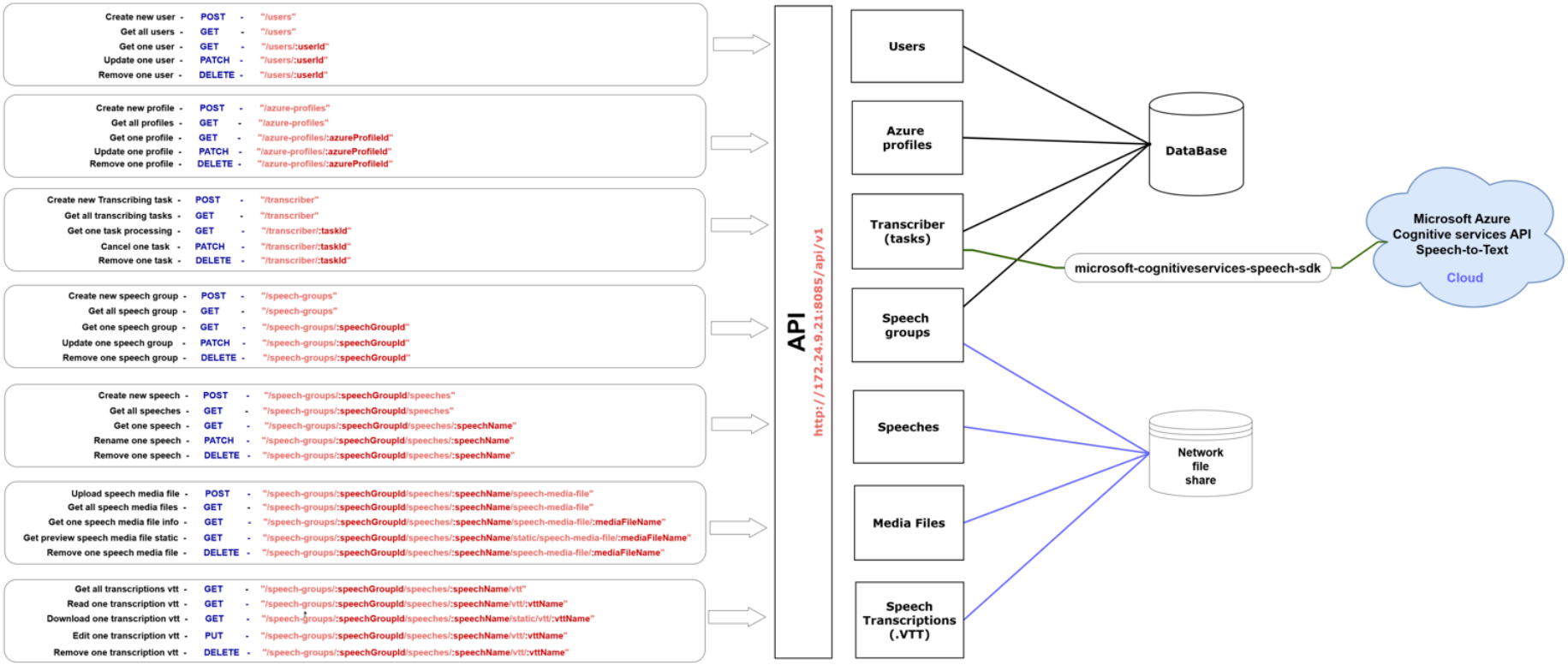
Порівняння


Azure
VS
Google
VS
OpenAI
Ціна
Безкоштовні перші 5 годин аудіо на місяць
WER
14.7%
Налаштування моделей
Гнучкі
Інструменти адаптації
Широкий набір
Підтримка аудіо
Багатоканальне аудіо, знаки пунктуації
Ціна
$0.0016 за хвилину
WER
52.9%
Налаштування моделей
Обмежені
Інструменти адаптації
Достатній набір
Підтримка аудіо
Багатоканальне аудіо, знаки пунктуації
Ціна
Залежить выд моделі
WER
7.6%
Налаштування моделей
Обмежені
Інструменти адаптації
Обмежений набір
Підтримка аудіо
Багатоканальне аудіо, знаки пунктуації

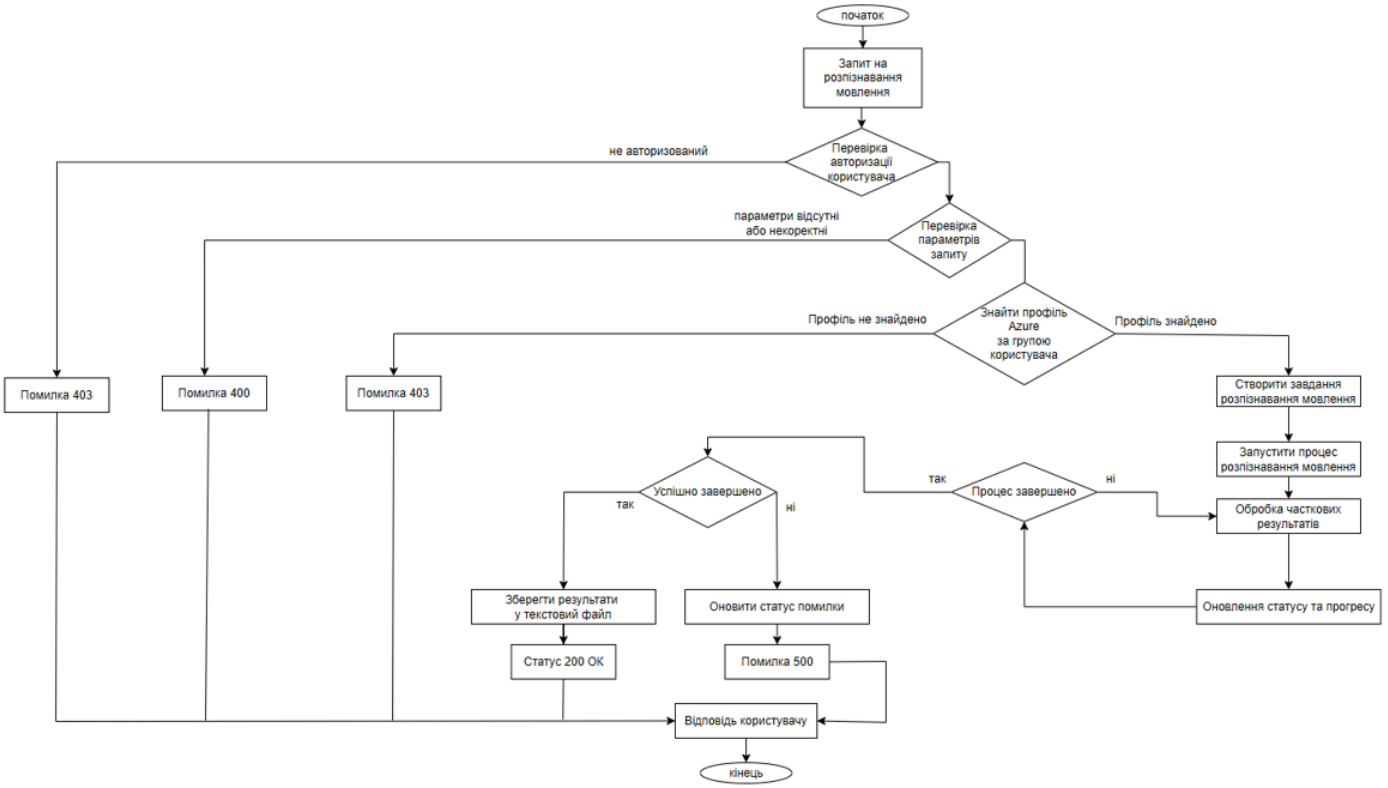
Принцип багатокористувацького доступу



Архітектура



Етапи розпізнавання мовлення



Тестування

Обробка одного файлу					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	20:00:47	20:03:46	0:02:59	0,02

Обробка двох файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	20:05:00	20:07:58	0:02:58	0,02
Name_2	5 хв	20:05:02	20:08:00	0:02:58	0,02
Name_3	5 хв	20:05:03	20:08:02	0:02:59	0,02

Обробка п'яти файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	20:09:48	20:12:46	0:02:58	0,02
Name_2	5 хв	20:09:50	20:12:48	0:02:58	0,02
Name_3	5 хв	20:09:52	20:12:49	0:02:57	0,02
Name_4	5 хв	20:09:54	20:12:51	0:02:57	0,02
Name_5	5 хв	20:09:56	20:12:53	0:02:57	0,02

Обробка десяти файлів одночасно					
Файл	Хронометраж	Старт	Фініш	Час опрацювання	WER
Name_1	5 хв	20:24:22	20:27:16	0:02:54	0,02
Name_2	5 хв	20:24:24	20:27:18	0:02:54	0,02
Name_3	5 хв	20:24:25	20:27:19	0:02:54	0,02
Name_4	5 хв	20:24:27	20:27:21	0:02:54	0,02
Name_5	5 хв	20:24:29	20:27:23	0:02:54	0,02
Name_6	5 хв	20:24:31	20:27:25	0:02:54	0,02
Name_7	5 хв	20:24:32	20:27:27	0:02:55	0,02
Name_8	5 хв	20:24:34	20:27:28	0:02:54	0,02
Name_9	5 хв	20:24:35	20:27:29	0:02:54	0,02
Name_10	5 хв	20:24:36	20:27:31	0:02:55	0,02

Пропозиції подальшого розвитку

- **Розширення підтримки додаткових API**: Додати інтеграцію з іншими провайдерами API для розпізнавання мовлення, що дозволить підвищити універсальність системи.
- **Обробка мультимедійних файлів**: Реалізувати можливість попереднього аналізу якості аудіо [7] та автоматичного вибору найбільш підходящого провайдера для обробки конкретного завдання.
- **Автоматичний вибір API**: Інтегрувати функціонал автоматичного аналізу вхідних даних для визначення найефективнішого API [4] залежно від параметрів, таких як якість аудіо та вимоги до точності.
- **Покращення безпеки**: Додати двофакторну аутентифікацію [6] та інші механізми захисту для забезпечення максимальної безпеки облікових записів користувачів із високим рівнем доступу.
- **Мобільна версія**: Розробити мобільний додаток, що дозволить користувачам отримувати доступ до системи з будь-якого пристрою, підвищуючи зручність і доступність продукту.

Висновок

- У процесі виконання дипломної роботи було розроблено клієнтську програму для перетворення мовлення в текст із використанням хмарних API, що є сучасним і перспективним напрямом у галузі інформаційних технологій. Проведено детальний аналіз сучасних технологій розпізнавання мовлення, включаючи провайдерів Microsoft Azure, OpenAI та Google Cloud. Було розглянуто ключові особливості кожного API, включно з точністю, швидкістю обробки, масштабованістю та вартістю послуг. Розроблений програмний продукт забезпечує підтримку багатокористувацького доступу [2], дозволяючи різним категоріям користувачів працювати з системою на основі їхніх прав доступу. Цей функціонал є важливим для забезпечення гнучкості та безпеки використання системи в багатокористувацькому середовищі.

- Під час тестування програмного продукту було підтверджено високу ефективність використання хмарних рішень для розпізнавання мовлення. Система демонструє стабільну роботу, адаптуючись до різних типів аудіо вхідних даних, і дозволяє значно зекономити час порівняно з традиційними методами ручного транскрибування. Ця розробка має потенціал для впровадження в реальних умовах, зокрема в освітніх установах, бізнесі та сервісах клієнтської підтримки.

- У рамках дипломної роботи було успішно досягнуто всіх поставлених цілей, що включають, таких як: проведення аналізу сучасних хмарних рішень для розпізнавання мовлення, визначення їх переваг і недоліків; створення програмного продукту з підтримкою багатокористувацького доступу [8], що дозволяє налаштовувати права доступу залежно від потреб користувачів; проведення порівняльного аналізу API Microsoft Azure, OpenAI та Google Cloud за ключовими показниками, включно з точністю розпізнавання, швидкістю обробки та вартістю; розробка системи, яка дозволяє інтегрувати функціонал API для забезпечення максимального рівня точності та адаптивності до різних типів завдань.

- Результати дослідження підтвердили, що розроблений програмний продукт відповідає сучасним вимогам і має великий потенціал для подальшого вдосконалення.