ЗМІСТ

					ІАЛЦ.045490.004ПЗ		
Зм	***Лист***	***№ докум.***	***Підп.***	***Дата***			
Розроб.		Шмирьов А.В.			Ком’ютерна система для розпізнавання диктора ***Пояснювальна записка***	***Літ.***	***Аркуш*** 1 / ***Аркушів*** 53
Перев.		Терейковський І.А.					
Н. контр.		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-32	
Затв.		Тарасенко В.П.					

ІАЛЦ.045490.005 Д1. Схема алгоритму задачі розпізнавання диктора.Схема алгоритму

ІАЛЦ.045490.006 Д2.Алгоритм роботи програми для розпізнавання диктора.Схема алгоритму

ІАЛЦ.045490.007 Д3.Структура бази даних системи розпізнавання диктора. Схема струтурна

ІАЛЦ.045490.008 Д4. Струтура програми програми для розпізнавання диктора.Схема струтурна

Перелік скорочень, умовних позначень, термінів

MVC(Model-View-Controller) – модель-інтерфейс-контролер

TDD(test-driven development) – розробка через тестування

DDL(Data Definition Language) – мова опису даних

JSF(JavaServer Faces) – написання веб-додатків

JMX(Java Management Extensions) – керуючі розширення

HID(human interface device) – пристрій для взаємодії з людиною

RFID(Radio Frequency Identification) – радіочастотна ідентифікація

Hasp(Hardware Against Software Piracy) - апаратне забезпечення проти програмного піратства

СКЗ – середнє квадратичне значення

АЧХ – амплітудна частотна характеристика

ВСТУП

Мовлення – це елемент людського життя, який дозволяє пізнавати оточуюче середовище, передавати свої знання іншим людям. Кожна людина має свої голосові характеристики, які створюють особливості голосових органів.

Розпізнавання по голосу має свій початок більш ніж 40 років тому, навіть зараз, ведуться дослідження в даній сфері, але автоматичне розпізнавання диктора в любому середовищі ще далеке від рішення.

Задача по розпізнаванню диктора по мовленню зводиться до того, що б виділити, класифікувати і правильно відреагувати на мовлення із вхідного звукового потоку. Наразі, зараз виділяють дві під задачі на вирішення цього питання: ідентифікація та верифікація людини.

Ідентифікація – процес розпізнання диктора та встановлення його особи шляхом порівняння переданого звукового потоку по зразку збереженого в базі даних. Зазвичай, результатом ідентифікації є ім’я людини, шаблон якого найбільш схожий на вхідний потік.

Верифікація – це процес за допомогою якого, порівнюється запрошена людина що збережена в базі з шаблоном потоку. Результатом є підтвердження особи або негативний результат системи.

Також системи розпізнавання диктора можуть бути як текстозалежні так і текстонезалежні. Текстозалежні можуть розпізнавати фіксовані фрази або фрази які були згенеровані системою і запропоновані користувачу.

Текстонезалежні системи призначені обробляти мовлення довльного тексту.

В даній роботі розглядається задача автоматичної ідентифікації диктора і реалізується алгоритм, який вирішує задачу текстонезалежним способом ідентифікації.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Основи розпізнавання диктора

Завдання розпізнавання дикторів є актуальним завданням в області мовних технологій. Розпізнавання дикторів об'єднує ідентифікацію та верифікацію дикторів.

Ідентифікація диктора - процес визначення особистості за зразком голосу шляхом порівняння даного зразка з шаблонами, збереженими в базі. Результатом процесу ідентифікації є список кандидатів. Система яка реалізується може видавати список фіксованого розміру або приймати рішення про включення користувача в список кандидатів. Якщо передбачена можливість того, що в процесі ідентифікації братиме участь користувач, не зареєстрований в системі, то говорять про ідентифікацію на відкритій множині. В ідеальному випадку для такого користувача система повинна видати порожній список. Якщо всі користувачі, що проходять процедуру ідентифікації, зареєстровані в системі, то кажуть про ідентифікацію на замкнутому безлічі.

Верифікація диктора - процес, при якому за допомогою порівняння наданого зразка з збереженим в базі шаблоном перевіряється запитана ідентичність. Результатом верифікації є позитивне чи негативне рішення. Іноді використовується термін «виявлення по голосу».

У задачі виявлення використовуються дещо інші терміни і пріоритети, але, по суті, верифікація та виявлення є однієї тієї ж завданням.

Крім даної класифікації самі системи розпізнавання можуть бути розділені на текстозалежні і текстонезалежні в залежності від того, відомий чи системі текст, який повинен бути виголошено користувачем, і чи використовує система дану інформацію. При текстозалежному розпізнавані можуть використовуватися як фіксовані фрази, так і фрази, генеровані

системою і запропоновані користувачеві. Текстонезалежні системи призначені обробляти довільну мову.

Існують і інші завдання, пов'язані з розпізнаванням по голосу. До числа таких можна віднести наступне завдання. Нехай сигнал містить запис розмови двох або більше осіб. Частина сигналу розмічається вручну або з використанням алгоритмів навчання без вчителя для вказівки, хто в якій момент говорить. Інша частина сигналу повинна бути розмічена в автоматичному режимі. У цьому завданні крім ідентифікації фрагментів потрібно визначити також їх межі. Таке завдання отримало в англомовній літературі назву speaker diarization («протоколювання дикторів»).

1.2. Опис проблеми розпізнавання диктора

Мова являє собою дуже складний сигнал, в результаті перетворень які проходять на декількох різних рівнях: семантичному, язиковому, рівні голосового апарату людини та на рівні фізичного звуку. Різниця яка зустрічається в цих перетвореннях тягнуть за собою різницю мовного сигналу кожної людини. При вирішені задачі по розпізнаванню диктора все ці відмінності можна використовувати як індивідуальну характеристику голоса будь-якої особистості.

На сьогоднішній день існують такі проблеми які заважають розпізнавати голос диктора без помилок, які потрібно врахувати при вирішені задачі:

- Емоційний стан диктора
- Складна акустична обстановка
- Різні канали зв’язку при навчанні та розпізнаванні
- Природні зміни голосу.

Загальна думка базується на тому, що на даний момент немає надійного варіанту для того щоб встановити чи належать записи однієї і тієї ж людині.

Наприклад, в криміналістиці можна тільки припустити, що два голосових сигнали належать одній і тієї ж людині. В умовах телефонного каналу важко визначити навіть стать або вік людини.

В силу малої вибірки мовних сигналів довірчий інтервал оцінки правдоподібності приналежності двох записів мови одного й того ж диктора настільки великий, що однозначне рішення неможливо.

Відповідно до цією думкою, в судовій практиці експертний висновок про ідентичність записів мови не приймається в якості юридичного докази. Це цілком логічно, оскільки в практиці кримінального розслідування при візуальної ідентифікації особистості потрібно порівняння з деякою кількістю схожих осіб, тоді як рішення про ідентичність голосів, засноване тільки на порівнянні перехоплених записів мовного сигналу і голоси підозрюваного, без порівняння з голосами безлічі інших дикторів, містить високий ризик помилки.

1.3 Аналіз існуючих рішень

Біометричні системи мають свій початок впровадження в український ринок в середині 90-их років. З невідомих причин, чи то поганого розвитку технологій, чи то через секретність, оскільки ця система розроблялась під безпеку, усі проекти які доходили до нашого ринку були з закордону. На той час собівартість продукту та його ціна біла доволі великою. Наприклад, найпростіша система фізичного контролю доступу коштувала дуже великих грошей, приблизно $10 000. Подібні проекти того часу так і не вийшли на масову популяцію через таку велику ціну і отримало, скоріше, характер новомодної екзотики. На сьогоднішній день подібні системи подешевшали, приблизно в 10 разів, але все одно це великі гроші і тому це одна із перших причин такого великого попиту в нашій країні виключно економічна – пристрої стали набагато дешевше. Інша причина зводиться до об’єктивної

потреби замовника створити систему, яка буде грамотно і чітко виконувати задачу безпеки як на підприємстві так і в офісі або в будинку.

Багато експертів вважає, що в наш час широко використовується спосіб дактилоскопічної ідентифікації. Дуже мало використовується систем які працюють за технологією розпізнавання по сітчатці ока, голосу або іншим біометричним ознакам. Але є компанії які використовують біометричні технології. Наприклад, «Приватбанк» вже впровадив систему розпізнавання робітника за відбитком пальця, із інших більш відомих компаній також можна назвати компанію «Макдональдс», у них також встановлена система контролю робочого часу за біометричною системою, також з'явився попит на дактилоскопічні системи від частих осіб, які встановлюють їх у своїх особистих домівках.

Українські розробки відрізняються своєю не повнотою, але існують і професійні зразки, але, нажаль, про якісь великі та об'ємні продажі щось казати ще занадто рано. Найчастіше ринок біометричної безпеки в нашій країні являють закордоні фірми які через наших партнерів проваджують свої системи на нашому ринку.

Систему Facelt, наприклад, представляє група компаній "Дан-ком", інженерна компанія "Солінг" активно впроваджує систему розпізнавання осіб німецького виробництва SmartEye, компанія "Біометричні системи" в основному спеціалізується на постачанні імпортного дактилоскопічного обладнання, але в цій компанії ведуться розробки програмного забезпечення для ідентифікації користувача. Хотілося відзначити розробки компанії «Центр мовних технологій». Вони розробили весь комплекс програм для ідентифікації користувача, для управління комп'ютером за допомогою голосу. Також вони можуть розробити будь-який забезпечення за потребами замовника. Одна з програм VoiceCom - бібліотека розпізнавання голосових команд, володіє наступними характеристиками:

Можливі області застосування: контроль обладнання за допомогою голосу; мовної запит для баз даних, можливо, по телефону; пошук за ключовими словами в WAV файлах; вбудовування голосових функцій в автономні пристрої - програмування DSP.

Переваги: велика швидкодія алгоритмів, невеликі вимоги до пам'яті, адаптація до шумів, незалежність від мови і акценту.

Можливості: Одночасне розпізнавання 100-200 команд в дикторозалежному і 30-50 команд в дикторонезалежному варіанті, можливість структурування для практично необмеженого словника, дикторонезалежне розпізнавання словника в 10-20 слів по телефону; Початок роботи після того, як ви скажете ключове слово (це є підтвердженням того, що система зреагує тільки на вашу команду, а не на що-небудь інше)

Технічні характеристики специфікації: Основа для розробки Borland Delphi 7.0

Вимоги: Конфігурація ПК Pentium 4 1500 або вище, RAM 256 Мб, пам'ять жорсткого диску більше 20Гб, Windows, Стандартна звукова карта, мікрофон

Плюси: Висока швидкодія алгоритмів, невеликі вимоги до пам'яті, адаптація до шумів, незалежність від мови і акценту. Подальший розвиток, підтримка розробником.

Недоліки: Інформацію о вартості системи не можливо дізнатись поки не буде реальної зацікавленості в розробці продукту.

Також існує декілька розробок в цій області за доволі невелику ціну. Наприклад:

- «Web-TalkIt». Виробник «USA Grover Industries», офіційний сайт http://www.groverind.com;
- «Труффальдіно». Виробник «Центр мовних технологій», офіційний сайт http://www.speechpro.com;

- VoiceNet VRS 2000. Виробник «USA Grover Industries», офіційний сайт http://www.groverind.com

Але найбільше розчарування в цих системах є те що вони не мають функції ідентифікації за голосом, а тільки керування комп’ютером за допомогою голосу. Але розроблені програмні забезпечення використовують ті ж самі математичні алгоритми, тому його можливо без навантажень модифікувати під подібні задачі.

Більшість прогнозів зводиться до того, що впровадження біометричних систем безпеки на наш ринок придбає в недалекому майбутньому лавинний характер. Пошук рішень для боротьби з наростаючою глобальною загрозою тероризму так чи інакше призведе до практичного використання досягнень в цій області. Інтенсивний розвиток мультимедійних, цифрових технологій і, як наслідок, їх здешевлення дозволяють не тільки розробити принципово нові підходи в проблемі ідентифікації особистості, а й впровадити їх в широке повсюдне використання.

Існуючі сьогодні системи розпізнавання голосу ґрунтуються на зборі всієї доступної (часом навіть надлишкової) інформації, необхідної для розпізнавання користувача.

Замість цього проводиться процес, першим кроком якого є початкове трансформування вводиться для скорочення оброблюваного обсягу так, щоб її можна було б піддати комп'ютерному аналізу. Наступним етапом є спектральне подання мови, вийшло шляхом перетворення Фур'є. Результат перетворення Фур'є дозволяє не тільки стиснути інформацію, але і дає можливість сконцентруватися на важливих аспектах мови, які інтенсивно вивчалися в сфері експериментальної фонетики. Спектральне подання досягнуто шляхом використання широко-частотного аналізу запису.

Хоча спектральне подання мови дуже корисно, необхідно пам'ятати, що досліджуваний сигнал дуже різноманітний.

Різноманітність виникає з багатьох причин, включаючи:

- відмінності людських голосів;
- рівень мови мовця;
- варіації у вимові;
- нормальне варіювання руху артикуляторів (мови, губ, щелепи, піднебіння).

Для усунення негативного ефекту впливу варіювання голосового тракту на процес розпізнавання мови було використано безліч методів. Найбільш вдалі форми трансформації, використаної для скорочення відмінностей, були вперше представлені Сако & Чібо і називалися динамічними спотвореннями (dynamic time warping). Техніка динамічного спотворення використовується для тимчасового витягування і скорочення відстані між спотвореним спектральним поданням і шаблоном для мовця. Використання даної техніки дало поліпшенні точного розпізнавання приблизно на 20-30%.

Метод динамічного спотворення використовують практично всі комерційно доступні системи розпізнавання, що показують високу точність повідомлення при використанні. Спочатку сигнал перетворюється в спектральне подання, де визначається нечисленний, але високоінформативний набір параметрів. Потім визначаються кінцеві вихідні параметри для варіювання голосу (слід зазначити, що дана задача не є тривіальною) і виробляється нормалізація для складання шкали параметрів, а також для визначення ситуаційного рівня мови.

Вищеописані змінені параметри використовуються потім для створення шаблону. Шаблон включається в словник, який характеризує вимовляння звуків при передачі інформації мовцем, що використовують цю систему. Далі в процесі розпізнавання нових мовних зразків (вже зазнали нормалізації і отримали свої параметри), ці зразки порівнюються з шаблонами, вже наявними в словнику, використовуючи динамічний спотворення і схожі метричні виміри.

Як вже говорилось раніше, рішення цієї задачі моє свій початок ще з 90-их років, наразі не має масштабних проектів які вирішують це питання. Зайнявшись цим питанням, я почав шукати програми або роботів які вирішують розпізнавання диктора по голосу, я знайшов систему яка використовується в банку «Пріорбанк», програму «ALIZE» та японського розумного робота «Pepper», який збирається піти до школи.

Порівняння системи в банку «Пріорбанк»

На сьогоднішній день зростає кількість людей які телефонують до банку з запитом потребуючих персональної ідентифікації клієнта. Технологія додаткової авторизації по голосу дозволяє суттєво зменшити час обслуговування клієнта.

Але, навіть, у такому банку система може давати похибки і видати помилку. Наприклад, видавши позитивну відповідь «чужому» так само і давши негативну відповідь «своєму».

Кожного місяця банк «Пріорбанк» отримує десятки тисяч дзвінків і якщо оператор буде на кожного витрачати ще час для ідентифікації клієнта, то час обслуговування буде надзвичайно довгим. За власним досвідом тільки ідентифікація займає приблизно 30 секунд, а тільки потім вирішення питання. А питання зазвичай типу «Чому не пройшов платіж?», «Чому в мене заблокована карточка?» або «Яка в мене заборгованість?» та інші. З новою системою час вирішення питання займає в середньому 60 секунд.

На рис. 1 ми можемо побачити порівняння часу обслуговування клієнта стандартним способом ідентифікації клієнта та з програмою автоматичної ідентифікації клієнта за голосом.

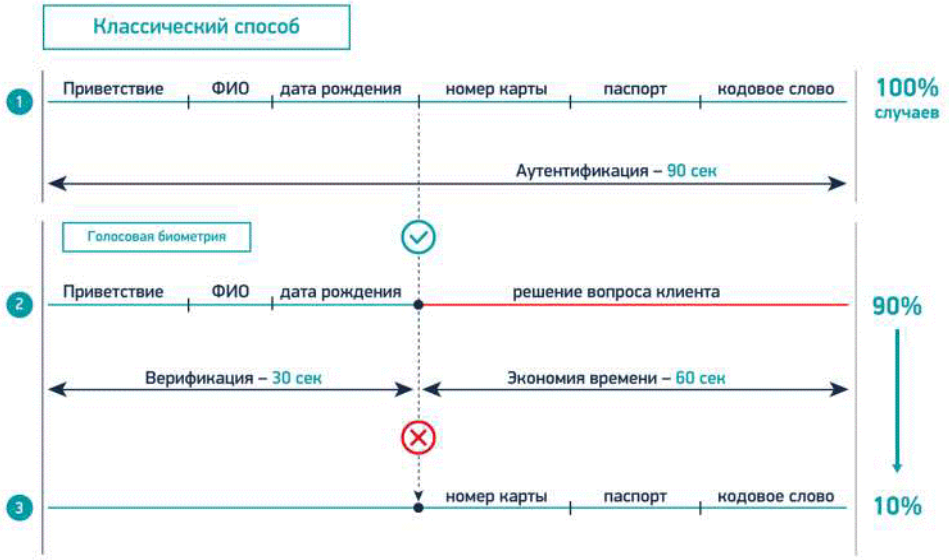

Рисунок 1 – Порівняння часу обслуговування клієнта.

Ідентифікація проходить за таким алгоритмом:

- Надходить дзвінок до банку і в цей же момент запускається програма для розпізнавання диктора
- В режимі реального часу отримуються данні диктора і аналізуються з шаблоном, який знаходиться в базі даних
- Результати перевірки клієнта показуються на екрані оператора(рис. 2)

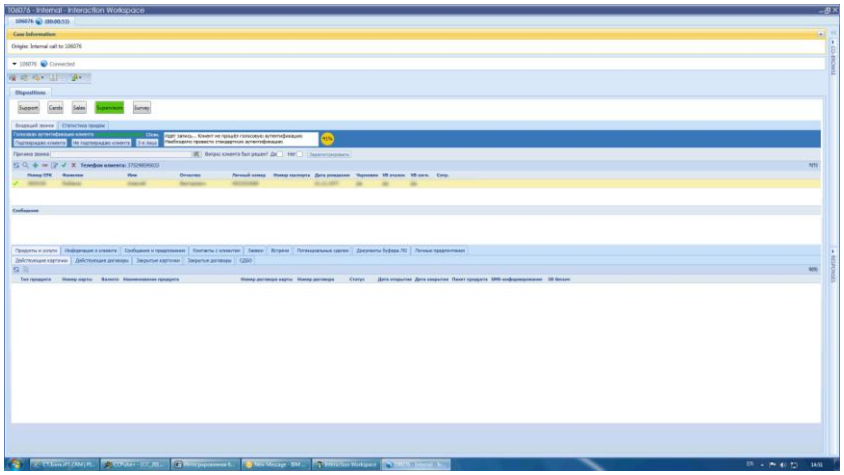

Рис. 2 – Результати перевірки клієнта по голосу

Переваги:

- Великий відсоток правильного результату(70-90%)
- Швидкість розпізнавання(близько 30 секунд)
- Після верифікації не слухає диктора

Недоліки:

- Залежить від інтернету
- Орієнтований тільки на банк
- Закритий вихідний код
- Мало відомий

Порівняння програми «ALIZE»

ALIZE проект – відкритий вихідний код платформи(LGPL) для біометричної ідентифікації. Основна ціль ALIZE створення низькорівневго та високорівневого фреймворку для створення програми по розпізнанню диктора верифікації або ідентифікаї по голосу. Включає в себе ядро та високорівневу бібліотеку, яка керує функціями у ядрі.

ALIZE був написаний під багаторівневу структуру. Це означає що програма працює під різними операційними система, такими як Windows, Linux та MacOS.

Переваги:

- Відкритий вихідний код
- Має високу оцінку від великих закордонних компаній
- Ведуться розробки для інтеграції для мобільних телефонів
- Використовується для наукових цілей

Недоліки:

- Потребує складних математичних підрахунків
- Немає документації
- Не є повноцінною програмою
- Відома в дуже вузькому колі користувачів

Порівняння японського розумного робота «Pepper»

Компанія NanoJam представила у світ свого унікального робота, який може адаптуватися до оточуючого середовища, спілкування з людьми, а також розуміти настрій співрозмовника. У словнику цього робота знаходиться 4500 слів для спілкування з людьми і він весь час поповнюється. Для розмов він використовує 4 мікрофона розволожених по всьому корпусу робота.

На даний час немає повних характеристик робота, але зі статей ЗМІ можна дізнатись що робот самонавчається і вміє розпізнавати диктора по голосу.

Переваги:

- Повноцінна програма для спілкування і розпізнавання диктора
- Запущена у масове виробництво
- Багатофункціональний

Недоліки:

- Не зовсім популярний
- Дуже велика ціна
- Велика черга на покупку
- Низький запас батареї

1.4. Обґрунтування теми дипломного проекту

На сьогоднішній день немає чіткого вирішення питання для ідентифікації та верифікації диктора по голосу. Є програми та, навіть, роботи які зможуть це зробити, але вони або вузько спеціалізовані, або не є повноцінними програмами, або ж коштують дуже великих грошей. Тому проаналізувавши всі ці фактори, я вирішив взятися за цю тему та розробити програму, яка буде завершена, яку можна використовувати не тільки в вузькому напрямку, а в більш масштабному, і в якої буде відкритий вихідний код.

2. ОПИС МАТЕМАТИЧНОГО ЗАБЕЗПЕЧННЯ

2.1. Математична модель розпізнавання диктора

В цьому дипломному проекті, я вирішив базувати свою систему на методі математичної моделі розпізнавання диктора статичного підходу. На мою думку він не важкий в реалізації і не потребує важких навантажень на систему.

Нехай реалізація слова представляється послідовністю елементів $X_l = (x_1, x_2, \ldots, x_l)$. Розпізнавана реалізація послідовно порівнюється з еталонними реалізаціями слова. При порівнянні реалізації з еталоном обидва елементи розтягуються шляхом повторення окремих елементів реалізації і еталона до деякої загальної довжини одним з методів нормалізації.

Нехай X^1_{l1} і X^2_{l2} дві реалізації деякого слова k. Звичайною мірою близькості є умова $P(x^1_{i_1}, x^2_{i_2}/k)$ спільності походження цих реалізацій.

Припустимо, що безліч різних реалізацій X_{lk}-го слова породжується з деякого еталонного сигналу $E_q = (e_1, \ldots, e_q)$, довжина якого коротше всіх можливих реалізацій X_{lk}-го слова. Застосуємо до E_q оператор розтягування $w = (W_1, \ldots, W_q)$ з множини(рис. 3)

$$W(q,l) = \left\{ w: W_0 = 0, 0 < m \le W_s - W_{s-1} \le M; W_q = l \right\}.$$

Рисунок 3 – Оператор розтягування

Результатом буде еталонний сигнал E_l дліни L(рис. 4)

$$wE_q = (\underbrace{e_1, \ldots e_1}_{W_1}, \ldots, \underbrace{e_s, \ldots e_s}_{W_s - W_{s-1}}, \ldots, \underbrace{e_q, \ldots e_q}_{l - W_{q-1}})$$

Рисунок 4 – Еталонний сигнал довжини L

X_l виходить з E шляхом спотворення деяким адитивним шумом.(рис. 5)

$$H_l = (h_1, h_2, \ldots, h_l): \quad x_i = e^i + h_i$$

Рисунок 5 – Адаптивний шум

У даній моделі враховуються нелінійні варіації темпу проголошення. Задавши конкретні значення m і M, можна звузити безліч W(q, l), не допускаючи надмірних спотворень темпу проголошення слова. У сигналі X_l з прототипом $E_1=wE_q$ близько q сегментів (крайній лівий елемент сегмента), які відповідають фонемам або частинам фонем.

Досить широкий клас розподілів $p(h_i)$ описується виразом зображеним на рис. 6

$$p(h_i) = p(x_i / e^i) = c \cdot \exp(-d(x_i, e^i));$$

$$p(X_l / wE_q) = \prod_{i=1}^{l} p(x_i / e^i) = c^l \cdot \exp\left(-\sum_{s=1}^{q} \sum_{i=W_{s-1}+1}^{W_s} d(x_i, e_s)\right).$$

Рисунок 6 – Клас розподілів

Тоді умовна ймовірність породження реалізацій X^1_{l1} та X^2_{l2} деяких еталонним елементом E_q зображена на рис. 7

$$p(X^1_{l_1}, X^2_{l_2} / w^1, w^2, E_q) = c_1 \cdot \exp\left(-\sum_{u=1}^{2} \sum_{s=1}^{q} \sum_{i=W^4_{s-1}+1}^{W^4_s} d(x^4_i, e_s)\right)$$

Рисунок 7 – Умовна ймовірність породження

Так як необхідні для інтегрування апріорні розподілу невідомі, деяку оцінку можна отримати, замінюючи інтегрування максимізацій по заважаючим параметрам. Таким чином, міру близькості можна записати як зображено на рис. 8

Вибір міри схожості $d(x_i, e_i)$ залежить від застосовуваного опису мовних сигналів і в значній мірі визначається зручностями обчислень. В цій роботі я використовую міру схожості скалярного складання векторів(рис. 9), оскільки він найбільш точніше порівняння і має елемент автокореляції.

$$\tilde{p}(X_{l_1}^1, X_{l_2}^2 / k) = \max_{E_q, w^1, w^2} p(X_{11}^1, X_{12}^2 / k);$$

$$G(X_{l_1}^1, X_{l_2}^2) = \min_{E_q, w^1, w^2} \sum_{u=1}^{2} \sum_{s=1}^{q} \sum_{i=W_{s-1}^4+1}^{W_s^4} d(x_i^4, e_s) =$$

$$\min_{q, w^1, w^2} \min_{(e_1, e_2, \ldots, e_q)} \sum_{u=1}^{2} \sum_{s=1}^{q} \sum_{i=W_{s-1}^4+1}^{W_s^4} d(x_i^4, e_s).$$

Рисунок 8 – Розрахунок міри близькості

$$d(x_i, e_i) = -\alpha(x_i)(x_i, e_i), \quad (x_i, e_i)$$

Рисунок 9– Скалярне складання векторів

2.2. Математичні алгоритми

Звук, утворений коливаннями всього діапазону частот, подібний до того, спектр якого показаний на рис. 10, називається шумом. Тлумачення цього слова, прийняте в техніці, відрізняється від загальновизнаного. Свист високого тону (видається, наприклад, старим монітором) може вважатися шумом в побутовому сенсі. Але у цього звуку є чітко визначений спектр частот, і, отже, він не може вважатися шумом в технічному сенсі цього слова.

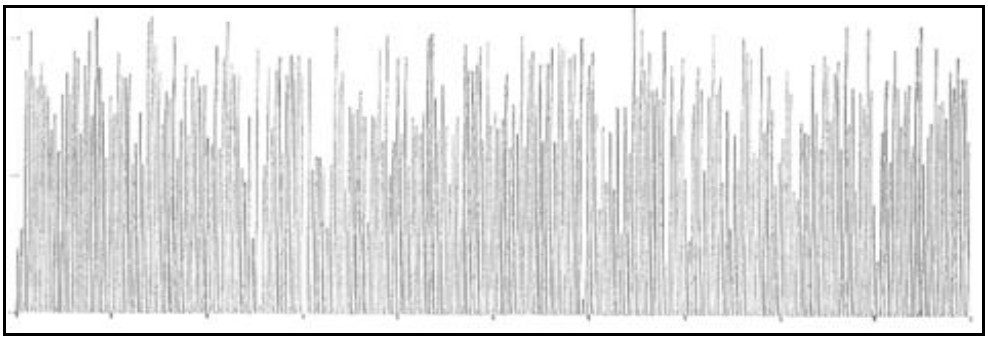

Рисунок 10 - Спектр частот шуму

Шумом може бути наприклад рух повітря - незалежно від того, чи це дихає людини або шуршання вітру в мікрофоні. Можна сказати, що ніжні звуки флейти в деякій мірі витягуються з шуму, виробленого видихом людиною повітря.

Так як шум містить всі частоти, флейта може виділити в ньому потрібні і посилити їх.

Якщо аналізувати дискретні значення (відліки) рівня шуму (а не спектр його частот), то вийде, випадкова вибірка. Гарним джерелом шуму є високоякісний генератор випадкових чисел.

Для того щоб отримати чіткі спектральні характеристики звуку їх потрібно відчистити від зайвих шумів.

Вхідний дискретний звуковий сигнал обробляється фільтрами, для того щоб позбутися від перешкод виникають при записі за формулою.

$$X_i = (X_i - 0{,}9 * X_{i-1})\left[0{,}54 - 0{,}46 * \cos\left((i-6) * \frac{2*\pi}{180}\right)\right], \qquad (1)$$

де X_i - набір дискретних значень звукового сигналу.

Після обробки в сигналі шукається початок і кінець запису, а так як шуми вже відфільтровані, то початок фрагмента буде характеризуватися сплеском сигналу, якщо шукати з X_0. Відповідно якщо шукати з X_n вниз, то сплеск буде характеризувати кінець фрагмента. Таким чином отримаємо початок і кінець фрагмента в масиві дискретних значень сигналу. У нематематичному вигляді це означає, що ми знайшли слово сказане користувачем в мікрофон, яке потрібно усереднити з іншими характеристиками голосу.

2.2.1. Середнє значення амплітуди

Крім висоти тону людина відчуває і іншу характеристику звуку - гучність. Фізичні величини, найбільш точно відповідні гучності, - це шоковий тиск (для звуків в повітрі) і амплітуда (для цифрового або електронного подання звуку).

Якщо говорити про відцифрований сигнал, то амплітуда - це значення вибірки. Аналізуючи мільйони дискретних значень рівня одного і того ж звуку, можна сказати про пікової амплітуді, тобто про абсолютну величину

максимального з отриманих дискретних значень рівня звуку. Щоб уникнути спотворення, викликаного спотворенням обмеження сигналу при цифрового запису звуку (дане спотворення виникає в тому випадку, якщо величина пікової амплітуди виходить за межі, які визначаються форматом зберігання даних), необхідно звернути увагу на величину пікової амплітуди. При цьому потрібно зберігати ставлення сигнал або шум на максимально досяжному рівні.

Основною причиною різної гучності звуків є різний тиск, який тисне ними на вуха. Можна сказати, що хвилі тиску мають різні рівні потужності. Хвилі, що несуть велику потужність, з більшою силою впливають на механізм вух. Електричні сигнали, що йдуть по проводах, також передають потужність. По проводам звук зазвичай передається у вигляді змінної напруги, і миттєва потужність цього звуку пропорційна квадрату напруги. Щоб визначити повну потужність за період часу, необхідно підсумувати всі значення миттєвої потужності за цей період.

Мовою математики це описується інтегралом $\int v_t^2 dt$, де v_t - це напруга в заданий момент часу.

Оскільки ви використовуєте звук, представлений дискретними значеннями, вам не знадобиться брати інтеграл. Досить просто скласти квадрати відліків. Середнє значення квадратів дискретних значень пропорційно середньої потужності.

Так як моментальна потужність залежить від квадрата моментальної амплітуди, має сенс аналогічним чином підібрати схоже співвідношення, що зв'язує середню амплітуду і середню потужність. Спосіб, яким це можна зробити, полягає у визначенні середньої амплітуди (СКЗ). Замість того, щоб обчислювати середнє значення безпосередньо амплітуди, ми спочатку будуємо в квадрат отримані значення, обчислюємо середнє значення отриманого безлічі, а потім витягуємо з нього корінь. Метод СКЗ застосовується в тому випадку, коли необхідно обчислити середнє для

мінливого значення. Алгебраїчно це виражається наступним образом: нехай у нас N значень і $x_{(i)}$ це амплітуда і-ого дискретного значення.

Потужність пропорційна зведеної у квадрат величиною дискретного значення. Це означає, що для переходу до реальної потужності, цю величину необхідно помножити на деякий коефіцієнт. Для цього не потрібні точні дані електричної потужності, так що, насправді, нас не цікавлять точні числа, швидше за відносна потужність.

Відносна потужність вимірюється в белах, а частіше в децибелах. Щоб порівняти два звуки, береться ставлення їх потужності. Десятковий логарифм цього відношення і є відмінність в белах; якщо множити число, на десять, то вийде значення в децибелах. Наприклад, якщо потужність одного сигналу перевершує потужність іншого в два рази, то перший сигнал буде голосніше на 10log10 (2) = 3,01 дБ.

Для пошуку цього значення функція робить розрахунки за формулою:

$$\sqrt{\frac{1}{N}\sum_{i=0}^{N}(x(i))^2}\,, \qquad (2)$$

Для початку ми беремо квадрат значення для того, тому що потужність пропорційна зведеної у квадрат величиною дискретного значення. Це означає, що для переходу до реальної потужності, цю величину необхідно помножити на деякий коефіцієнт. Для цього не потрібні точні дані електричної потужності, так що, насправді, нас не цікавлять точні числа, скоріше – відносна потужність. Відносна потужність вимірюється в белах, але частіше в децибелах.

Є кілька причин, за якими за допомогою вимірювань, проведених в децибелах, вдається добре апроксимувати то, як людина відчуває гучність. По-перше, почуття слуху у людини дуже близько до логарифму: відчувається різниця і гучності двох звуків залежить від ставлення, а не від різниці

потужностей кожного зі звуків. Хоча це буде і не зовсім коректно, було б непогано розглядати децибел як мінімально відчувається зміна гучності.

Ще один аспект, для якого вимірювання в децибелах дають точну картину відчуття людини - це те, що відчувається гучність дуже сильно залежить від відносної потужності. Зокрема, відома акустична ілюзія, звана маскування. Якщо звук утворюється двома незалежними компонентами і одна з цих компонент набагато голосніше інший, то більш тиха компонента часто буде нечутно. Фактично, слух людини «налаштовується» до рівня більш гучного звуку і більш тихий звук чується набагато більш тихим, ніж він є насправді. Це особливо відноситься до тих ситуацій, коли у цих звуків дуже близькі висоти тону.

2.2.2. Розкладання Фур’є

На відміну від ряду Фур'є, перетворення Фур'є розкладає функцію не по дискретними частотами (набір частот ряду Фур'є, за якими відбувається розкладання, взагалі кажучи, дискретний), а по безперервним.

Якщо поглянути на те, як співвідносяться коефіцієнти ряду Фур'є і результат перетворення Фур'є, іменований, власне, спектром.

Спектр перетворення Фур'є - в загальному випадку, функція комплексна, що описує комплексні амплітуди відповідних гармонік. Тобто, значення спектра - це комплексні числа, чиї модулі є амплітудами відповідних частот, а аргументи - відповідними початковими фазами. На практиці, розглядають окремо амплітудний спектр і фазовий спектр.

Однак, перетворення Фур'є зіставляє безперервної в часі, нескінченної функції іншу, безперервну по частоті, нескінченну функцію - спектр. Як бути, якщо у нас немає нескінченної в часі функції, а є лише якась записана її дискретна в часі частина? Відповідь на це питання дає подальшої розвиток перетворення Фур'є - дискретне перетворення Фур'є .

Дискретне перетворення Фур'є покликане вирішити проблему необхідності безперервності і нескінченності в часі сигналу. По суті, ми вважаємо, що вирізали якусь частину нескінченного сигналу, а всю іншу тимчасову область вважаємо цей сигнал нульовим.

Математично це означає, що, маючи досліджувану нескінченну в часі функцію $f_{(t)}$, ми множимо її на деяку віконну функцію $w_{(t)}$, яка наближається до нуля всюди, але нас цікавить інтервалу часу.

Якщо «виходом» класичного перетворення Фур'є є спектр - функція, то «виходом» дискретного перетворення Фур'є є дискретний спектр. І на вхід теж подаються відліки дискретного сигналу. Решта властивості перетворення Фур'є не змінюються.

Нам потрібно лише знати про Фур'є - образ синусоїдального сигналу, який ми і будемо намагатися відшукати в нашому спектрі. У загальному випадку, це пара дельта-функцій, симетрична щодо нульової частоти в частотної області.

Ми розглядаємо не вихідну функцію, а деякий її твір з віконної функцією. Тоді, якщо спектр вихідної функції, а віконної, то спектром твори буде така неприємна операція, як згортка цих двох спектрів.

Цей ефект називають також розтікання спектру. А шуми, що з'являються внаслідок розтікання спектру, відповідно, бічними пелюстками.

Для боротьби з бічними пелюстками застосовують інші, не прямокутні віконні функції. Основною характеристикою «ефективності» віконної функції є рівень бічних пелюсток. Зведена таблиця рівнів бічних пелюсток для деяких часто використовуваних віконних функцій приведена нижче.

Основною проблемою в нашій задачі є те, що бічні пелюстки можуть маскувати інші гармоніки, що лежать поруч.

Інший підхід до боротьби з розтікання спектру полягає в відніманні з сигналу гармонік, що створюють це саме розтікання.

Тобто, встановивши амплітуду, частоту і початкову фазу гармоніки, можна відняти її з сигналу, при цьому ми приберемо і «дельта-функцію», відповідну їй, а разом з нею і бічні пелюстки, породжувані їй. Інше питання полягає в тому, як же точно дізнатися параметри потрібної гармоніки. Недостатньо просто взяти потрібні дані з комплексної амплітуди. Комплексні амплітуди спектра сформовані цілими частотам, однак, ніщо не заважає гармоніці мати і дробову частоту. В цьому випадку, комплексна амплітуда як би розпливається між двома сусідніми частотами, і точну її частоту, як і інші параметри, встановити не можна.

Для встановлення точної частоти і комплексної амплітуди потрібної гармоніки, ми скористаємося прийомом, широко застосовуваним у багатьох галузях інженерної практики – гетероденірования.

Роздивимось варіант, що вийде, якщо помножити вхідний сигнал на комплексну гармоніку. Спектр сигналу зрушиться вправо.

Цією властивістю ми і скористаємося, зрушуючи спектр нашого сигналу вправо, до тих пір, поки гармоніка не стане ще більше нагадувати дельта-функцію (тобто, поки деяке локальне ставлення сигнал або шум не досягне максимуму). Тоді ми і зможемо обчислити точну частоту потрібної гармоніки, і відняти її з вихідного сигналу для зменшення ефекту розтікання спектру.

Будемо повторювати описані процедури до тих пір, поки не виріжемо всі присутні гармоніки, і спектр не буде нагадувати нам спектр білого шуму.

Основна ідея швидкого перетворення Фур'є полягає в тому, що кожну другу вибірку можна використовувати для отримання половинного спектра. Формально це означає, що формула дискретного перетворення Фур'є може бути представлена у вигляді двох сум. Перша містить всі парні компоненти оригіналу, друга - всі непарні

$$A_f = \sum_{t=0}^{N/2-1} s_{2t} e^{-2\pi t f/(N/2)} + e^{-2\pi t f/N} * \sum_{t=0}^{N/2-1} s_{2t+1} e^{-2\pi t f/(N/2)}, \quad (3)$$

Подання мовної інформації в частотній області піддається деякими перевагами. По-перше, це дає досить чіткий опис звуків мови. По-друге, в початковій стадії сприйняття в вусі людини проводиться деякий грубий аналіз. Таким чином, характерні особливості, які виявляються в результаті частотного аналізу, грають важливу роль в процесах сприйняття і розпізнавання голосової інформації. Тому важливо знайти спектральну щільність апериодической функції.

Безперервний мовний сигнал, як і будь-який інший можна представити у вигляді:

$$X(t) = \sum_{n=0}^{\infty} a_n U(t), \tag{4}$$

якщо безліч функцій U_n є ортогональними, тобто задовольняє умові

$$\int_0^T U_n(t) * U_m(t) dt = \begin{cases} C, если \ \ n = m \\ 0, если \ n \neq m \end{cases}, \tag{5}$$

де С - константа.

У цьому випадку значення коефіцієнтів a_n визначаються виразом:

$$a_n = \frac{1}{T} \int_0^T X(t) * U_n(t) dt, \tag{6}$$

Залежно від виду використовуваних ортогональних функцій розрізняють кілька видів перетворень. За допомогою пари перетворень Фур'є можна виразити зв'язок між аперіодичною функцією часу f (t) і її комплексним спектром F (w):

$$\mathrm{F(w)} = \int_{-\infty}^{\infty} \mathrm{f(t)} * \mathrm{e}^{-\mathrm{iwt}} \mathrm{dt}, \tag{7}$$

$$f(t) = \frac{1}{2x} \int_{-\infty}^{\infty} F(w) * e^{iwt} dw, \tag{8}$$

Спектральна форма включає в себе дві основні операції: аналого-цифрового перетворення - перетворення сигналу з хвилі звукового тиску в цифровий сигнал; цифрова фільтрація - виділення головної частоти сигналу.

2.2.3. «Вікно Кайзера»

Завдання розрахунку кращих вікон фактично зводиться до математичної задачі відшукання обмежених у часі функцій, перетворення Фур'є яких найкращим чином апроксимують функції, обмежені по частоті, тобто мають мінімальну енергію за межами заданого інтервалу частот. При вирішенні цього завдання в замкнутій формі для безперервних функцій часу був виведений клас так званих витягнутих сфероїдальних хвильових функцій. Ці функції мають досить складний вид, тому Кайзер в якості найкращого вікна запропонував відносно просту апроксимацію цих функцій. Ця апроксимація, названа вікном Кайзера.

Частотна характеристика дискретного вікна Кайзера в замкнутій формі не отримана, але Кайзер показав, що для неперервної функції вікна частотна характеристика пропорційна

Вікно Кайзера є, по суті, оптимальним вікном в тому сенсі, що воно являє собою послідовність кінцевої довжини, яка має мінімум енергії спектра за межами деякої заданої частоти.

Віконні функції мають виняткову важливість в спектральному аналізі сигналів. В даному випадку, ми наведемо приклади для деяких віконних функцій, а також їх зовнішній вигляд в тимчасовій і частотній областях.

Узагальнимо основні частотні характеристики спектра віконної функції, що дозволяють порівнювати різні вікна між собою. Для цього розглянемо нормовану амплітудно-частотну характеристику віконної функції.

Нормування амплітуди проводиться для обліку коефіцієнта ослаблення. Таким чином, всі АЧХ матимуть максимум дорівнює одиниці на нульовій

частоті. Оскільки ширина головної пелюстки залежить від тривалості вікна в часі то введено нормування частоти.

Таким чином, форма нормованої АЧХ віконної функції не буде змінюватися при зміні тривалості вікна. Тоді можна ввести наступні нормовані параметри:

- Нормована ширина головної пелюстки АЧХ за рівнем 0,5 (-3 дБ) визначається як нормована смуга при деякому децибелі.
- Нормована ширина головної пелюстки АЧХ по нульового рівня.
- Максимальний рівень бічних пелюсток.

Можна помітити, що ширина головної пелюстки прямокутного вікна дорівнює 2. Тоді можна ввести параметр, що складає скільки раз нормована ширина головної пелюстки АЧХ нульового рівня заданого вікна ширше ніж прямокутного вікна. Залежно від параметра вікна ділять на вікна з високою роздільною здатністю і вікна низького дозволу.

Отримавши спектральне подання сигналу його потрібно відчистити від шумів. Людський голос володіє відомими характеристиками, і тому ті області які не можуть бути характеристиками голосу потрібно погасити.

Для цього застосуємо функцію, яка отримала назву «вікно Кайзера»

$$y = \frac{x}{2}, \tag{9}$$

$$I_0(b) = \sum_{n=1}^{50} \left(\frac{I_o(b)_{n-1} * y}{n} \right)^2, \tag{10}$$

$$I_{01} = \frac{1}{I_0(b)}, \tag{11}$$

$$X_i = X_i * \left[b * \sqrt{1 - \left(\frac{2*k}{n-1} \right)^2} \right] * I_{01}, \tag{12}$$

2.2.4. Математичне очікування

Основним параметром, використовуваним для ідентифікації, є міра подібності двох звукових фрагментів. Для її обчислення необхідно порівняти спектрограми цих фрагментів. При цьому спочатку порівнюються спектри, отримані в окремому вікні, а потім обчислені значення усереднюються.

Для порівняння двох фрагментів використовувався наступний підхід:

X[1..N] і Y[1..N] - масиви чисел, одного і того самого розміру N, що містять значення спектральної потужності першого і другого фрагментів відповідно.

Тоді міра подібності між ними обчислюється за такою формулою:

$$f_{xy} = \left| \frac{\sum_i (x_i - M_x)(y_i - M_y)}{\sqrt{\sum_i (x_i - M_x)^2} * \sqrt{\sum_i (y_i - M_y)^2}} \right|, \qquad (13)$$

де M_x і M_y - математичні очікування для масивів X[] і Y[] відповідно, обчислюється за наступною формулою:

$$M_z = \frac{1}{N} \sum_1^N z_i , \qquad (14)$$

Даний спосіб обчислення міри схожості двох фрагментів представлених у вигляді спектра є найоптимальнішим для задачі ідентифікації людини по його голосу.

2.3. Висновок

Отже, проаналізувавши вимоги до задачі, а також існуючі варіанти, прочитавши літературу, було вирішено написати програму, яка була би

простою у користуванні, не потребувала би потужної системи і виконувала свої функції.

Я обрав математичний метод статичного підходу оскільки він простий у реалізації, не потребує великих математичних розрахунків, найменш поширений. Також, проаналізувавши алгоритми для розпізнавання диктора, було обрано алгоритм середнього значення амплітуди, розклад Фур'є та «вікно Кайзера» оскільки у цій комбінації вони дають найбільш точну характеристику звукового сигналу і мають велику швидкодію.

До того ж, було обрано метод порівняння еталонів математичного очікування, оскільки він вираховує весь масив даних і ми отримуємо результат і відсотковому відношенні, що в свій час дає змогу порівняти декілька шаблонів і дати більш точну ймовірність правильного розпізнавання диктора.

3. Розробка комп’ютерної системи розпізнавання диктора

3.1 Вибір технологій для розробки.

Перевагою системи Visual Studio Express з використанням мови C# над іншими системами розробки додатків є його безкоштовна платформа. Система прийняття та обробки звукового сигналу буде працювати на самій системі що пришвидшить його роботу та зробить повністю безкоштовним для використання. Таким чином було обрано цю систему з використанням мови C#, яка включає в себе велику кількість бібліотек та не потребує придбання ліцензій та дозволяє реалізувати потужну систему для вирішення поставленої задачі.

Також хотілось би відмітити, що я вирішив зупинитись на шаблоні який працює з класами та об’єктами за технологією MVC. На сьогоднішній день ця технологія дуже відома і дуже часто використовується програмістами, навіть, великі компанії дуже часто використовують цю систему, адже вона зрозуміла і будується на простій структурі.

MVC є популярним підходом до розробки , реалізує архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення. Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача. Таким чином ми маємо універсальну зручну та гнучку парадигму, яка дозволяє пришвидшити та зробити зручнішим процес розробки та відладки додатків на об’єктно-орієнтованих мовах. Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у

великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності. Архітектурний шаблон MVC поділяє програму на три частини. У тріаді до обов'язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами. NAUDIO поділяє систему на три самостійні частини: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було зазначено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних. Компонент виводу Вигляд може мати декілька взаємопов'язаних областей, наприклад, різні таблиці і поля форм, в яких відображається інформація. У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача (зміна положення курсора миші, натиснення кнопки або введення даних в текстове поле).

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі, або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через Контролер внесе зміни до Моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

При розробці моделі було використано технологію розробки через тестування. Розробка через тестування (англ. Test-driven development (TDD)) — технологія розробки програмного забезпечення, яка використовує короткі

ітерації розробки, що починаються з попереднього написання тестів, які визначають необхідні покращення або нові функції. Кожна ітерація має на меті розробити код, який пройде ці тести. Нарешті, програміст або група вдосконалюють код для погодження змін. Один із ключових моментів TDD полягає у тому, що підготовка тестів перед написанням самого коду пришвидшує процес внесення змін. Керована тестами розробка вимагає від розробника створення автоматизованих модульних тестів, які визначають вимоги до коду безпосередньо перед написанням самого коду. Тест містить перевірки умов, які можуть або виконуватися, або ні. Коли вони виконуються, кажуть, що тест пройдено. Проходження тесту підтверджує поведінка, передбачувана програмістом. Розробники часто використовують програмні каркаси для тестування, для створення та автоматизації запуску наборів тестів. На практиці модульні тести покривають критичні та нетривіальні ділянки коду. Це може бути код, схильний до частих змін, код, від роботи якого залежить працездатність великої кількості іншого коду, або код з великою кількістю залежностей. Середовище розробки повинно швидко реагувати на невеликі модифікації коду. Архітектура програми повинна базуватися на використанні безлічі сильно пов’язаних компонентів, які слабо залежать одне від одного , завдяки чому тестування коду спрощується. TDD не тільки пропонує перевірку коректності, а й впливає на дизайн програми. Спираючись на тести, розробники можуть швидше уявити, яка функціональність необхідна користувачеві. Таким чином, деталі інтерфейсу з'являються задовго до остаточної реалізації рішення. Зрозуміло, що до тестів застосовуються такі ж вимоги щодо якості коду, як і до основного коду.

Процедури у C# представляють собою колекцію програмних інструкцій, яка виконує операції в базі даних, зберігається під певним ім'ям і обробляється як єдине ціле. Така технологія реалізації процедур дозволяє додатково

економити апаратні засоби та застосовувати пришвидшення у процесорах Intel.

Треба сказати про різницю між основними мовами програмування на сьогоднішній день, які могли б бути обраними для цілей розробки – C#, Java або Python. Мови C#, Java та Python з'явилися в різний час. Мова Java був створений задовго до появи C#, та лише на рік раніше ніж з'явився Python. Python має свій початок ще з 1980-х років, але вийшов у світ лише 1991 року. Java був розроблений компанією Sun Microsystems в 1990 р, а в 1995 була випущена перша бета-версія Java. Створення C# було анонсовано в 2000 році, а в 2002 році вийшла перша версія платформи .NET, що підтримує C#. Таким чином, якщо Java створювався спираючись більшою мірою на досвід мов Objective C і C, то для C#, як і для Python, такою опорою були C++ і сам Java. І, незважаючи на свою назву, C# виявився ближче до Java, ніж до C++.

З точки зору розробника мови Java і C# дуже схожі, але вони обидва дуже сильно відрізняються від Python. Усі три мови є строго типізовані, об'єктно-орієнтовані та увібрали в себе багато чого з синтаксису C++, але на відміну від C++, простіше в освоєнні для початківців. Java і C# запозичили з C набір основних ключових слів і службових символів, в тому числі фігурні дужки для виділення блоків. На відміну від Python мови Java і C# спираються на збірку сміття. Усі мови супроводжуються багатими колекціями бібліотек. Але є в мовах також свої особливості і відмінності, сильні і слабкі сторони. C# врахував багато недоліків Java, і виправив їх у своїй реалізації. Але і Java з Python не стоять на місці, розвиваючись паралельно з C#.

Тригер - це особливий тип збереженої процедури, яка активується при зміні даних у зазначеній таблиці за допомогою однієї або декількох операцій зміни даних: UPDATE, INSERT або DELETE. Також можна створити тригери рівня бази даних, що спрацьовують у відповідь на оператори мови DDL. Такі тригери можна використовувати для виконання завдань адміністрування бази даних, таких як аудит і упорядкування операцій в базі даних. Розробка

процедур і тригерів зручна за рахунок набору операторів мов Visual Basic і Visual C#, особливо при реалізації складної процедурної логіки, необхідної для застосування бізнес-правил. Крім того, платформа .NET Framework містить безліч бібліотек. Особливий інтерес представляють ті, які дозволяють використовувати численні засоби криптографії, великі математичні бібліотеки і зовнішній доступ до веб-служб, файлів і комунікаційних систем типу "бізнес-бізнес".

В основі програмування на C# лежить технологія апплетів. C# Servlet .NET — стандартизований .NET для створення динамічного контенту до сервера, використовуючи платформу C#. Апплет може зберігати інформацію між багатьма транзакціями, використовуючи сесії, або редагування. Servlet .NET, що міститься в пакеті C#.servlet, описує взаємодію контейнера і апплета. контейнер — це компонент сервера, що створений для взаємодії з апплетами. Він відповідає за управління життєвим циклом апплетів, перетворення у певний апплет та забезпечення того, щоб клієнт, який зробив запит, мав відповідні права доступу

Для розробки додатку на C# було обрано використовувати технологію C#.NET Faces. C#.NET Faces — це NAUDIO каркас програмування, технологія для додатків, що написані на C#. Він слугує для того, щоб полегшувати розробку користувацьких інтерфейсів для C# додатків. На відміну від більшості NAUDIO пакетів, які керуються запитами, підхід JSF ґрунтується на використанні компонентів. Стан компонентів користувацького інтерфейсу зберігається, коли користувач запитує нову сторінку й потім відновлюється, якщо запит повторюється. Користь технології JSF обумовлена, в першу чергу, наявністю специфікації JSF. Специфікація дозволяє розробляти JSF пакети з різним призначенням та різною внутрішньою структурою. Вона лиш гарантує, що каркас буде підпорядкований певній структурі. Група експертів підтримує в актуальному

стані еталонну реалізацію (reference implementation) JSF. Це дозволяє, як використовувати її в реальних застосунках, так і розвивати інші реалізації конкурентів. Гарними сферами застосування еталонної реалізації JSF є невеликі пакети.

Як альтернативу до використання C#.NET Faces, було розглянуто фреймворк Bukkase NAUDIO. Bukkase Framework — це програмний каркас (пакет) з відкритим кодом та контейнера з підтримкою інверсії управління для платформи C#. Bukkase Framework складається з декількох модулів, які надають широкий спектр послуг:

- Контейнер інверсії управління: конфігурація компонентів додатків і управління життєвим циклом об'єктів C#, здійснюється головним чином через інверсію управління.
- Аспектно-орієнтоване програмування: дозволяє реалізувати наскрізні процедури.
- Доступ до даних: робота з реляційною системою управління базами даних на платформі C# з використанням JDBC і об'єктно-реляційні відображення та інструментів з NoSQL баз даних.
- Управління транзакціями: об'єднує кілька .NET, управління транзакціями та координує операції для C#-об'єктів
- Модель-Вид-Контролер (Model-View-Controller): програмний каркас на основі HTTP апплета, що забезпечує створення додатків і служб RESTful.
- Аутентифікація і авторизація: налаштовані процеси безпеки, які підтримують цілий ряд стандартів, протоколів та інструментів.
- Віддалене управління: конфігураційній вплив і управління C#-об'єктами для місцевої (локальної) або віддаленої конфігурації через JMX.

- Тестування: підтримка класів для написання загальних модульних тестів та інтеграційних тестів

Безумовно, для великих корпоративних проектів Bukkase NAUDIO – найкращий вибір, але для невеликого проекту, основна складність якого полягає у моделі алгоритму, C#.NET Faces забезпечує достатньо потужний засіб для створення додатків, при цьому даний фреймворк не є вимогливим до апаратного забезпечення в цілому.

Для розробки інтерфейсу було використано каркас .NET APIc. .NET APIc — це безкоштовний набір інструментів з відкритим вихідним кодом, призначений для створення пакетів та застосунків, який містить шаблони CSS та WINAPI для графіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення C#. Він спрощує розробку динамічних пакетів і застосунків. .NET APIc — це клієнтський пакет, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері.

C# копіює концепцію віртуальних методів C++: віртуальний метод повинен бути явно оголошений з ключовим словом virtual, інші методи віртуальними не є. Таке вибіркове оголошення віртуальних методів введено в C #, так як оголошення всіх методів віртуальними може сильно уповільнити виконання. Крім того, C # вимагає явного оголошення про перекриття віртуального методу в похідному класі ключовим словом override. Якщо потрібно приховати (hide) віртуальний метод, тобто просто ввести новий метод з тим же ім'ям і сигнатурою, потрібно вказати ключове слово new (в разі відсутності якого компілятор видає попередження). Забороняється затуляти абстрактні методи. Оголошення override-методу з ключовим словом sealed забороняє перевизначати (override) цей метод в класах-нащадках, проте як і раніше дозволяє приховати його.

У Java, навпаки, всі відкриті методи, крім статичних, є віртуальними, а перевизначити метод так, щоб механізм віртуальності не включили, неможливо. Метод завжди віртуально перекриває метод базового класу з тими ж ім'ям і сигнатурою, якщо він є. Ключове слово final дозволяє заборонити створення методу з такою ж сигнатурою в похідних класах.

Підхід Java синтаксично простіше, він гарантує, що завжди викликається метод саме того класу, до якого належить об'єкт. З іншого боку, віртуальність дійсно потрібна не завжди, а накладні витрати на виклик віртуальних методів дещо більше, оскільки ці виклики зазвичай не проходять інлайн підстановку і вимагають додаткового звернення до таблиці віртуальних методів (хоча деякі реалізації JVM, включаючи реалізацію Sun, реалізують інлайн підстановку для віртуальних методів, що найбільш часто викликаються).

3.2 Структура та реалізація

Розроблена система складається з зовнішнього інтерфейсу для операційної системи Windows з .NET Framework 4.6 та новіше. Така комбінація дозволяє використовувати додаток на широкому спектрі пристроїв під керуванням ОС Windows. Для цієї цілі в процесі тестування системи було використано потужну робочу станцію обладнану мікрофоном. Завдяки простій принциповій організації та використанні великої кількості апаратно пришвидшуваного коду було досягнена висока швидкодія системи загалом, гарна чистота коду та легкість відлаштування. Первинний вибір технологій підтвердив себе в процесі кінцевого налагодження, яке зайняло всього один день, що дуже мало, як для системи з такою кількістю коду.

Програма була створена у відповідності за допомогою технології MVC, тому можна сказати, що вона складається умовно з трьох частин це модель, контролер та інтерфейс користувача. На рис. 11 зображено загальну схему роботи технології MVC.

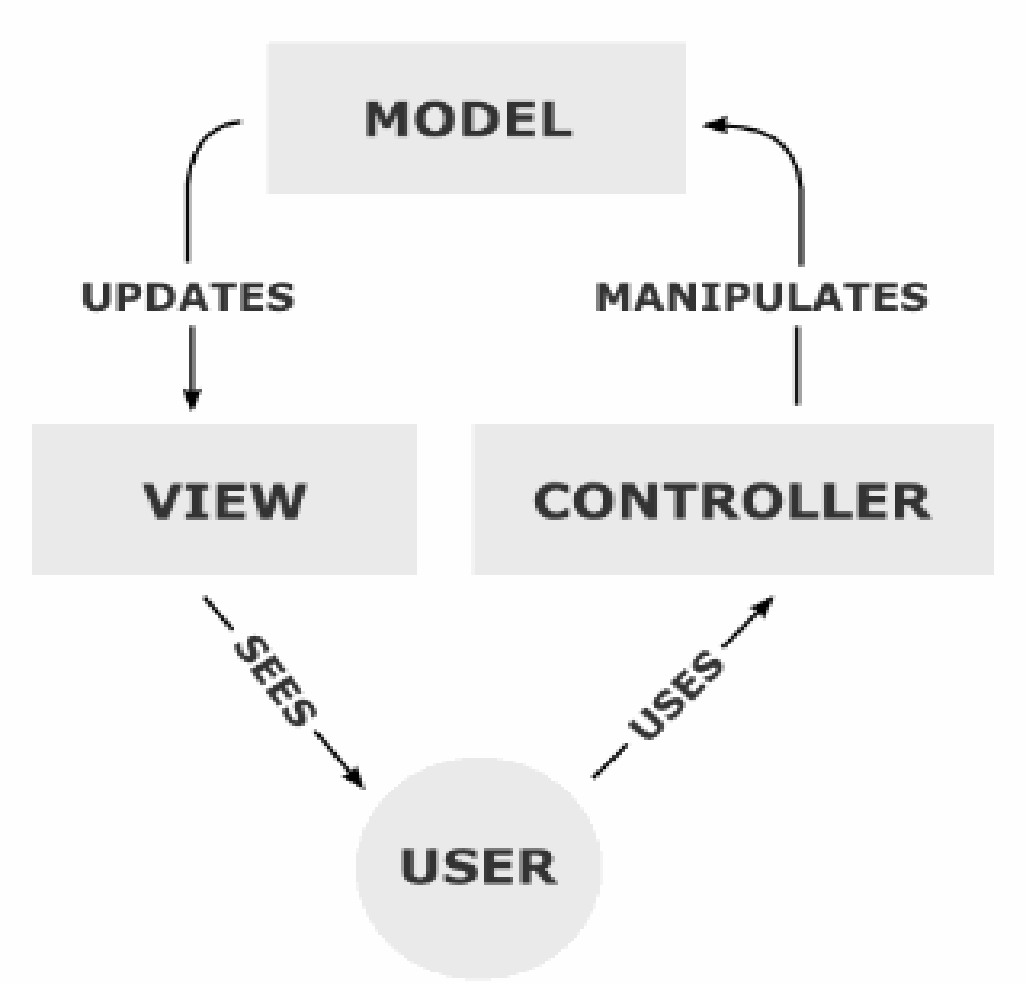

Рисунок 11 – Загальна схема роботи технології MVC

Контролер складається з класу та об’єктів, що зв’язує модель та інтерфейс користувача. Контролер виконаний за технологією C# Bean Dependency Injection і представлений класом GetBean.C#. C# Bean – це клас C#, який виконаний за певними правилами, що дозволяє системі керувати його життєвим циклом. Розроблений клас існує протягом всієї сесії, отже при отриманні даних від інтерфейсу користувача або від моделі, даний клас буде існувати до закриття сесії, що виключає можливість втрати даних.

Модель відповідає за обробку отриманого аудіо-сигналу, отриманих результатів від процесу ідентифікації диктора та всіх дій користувача.

Інтерфейс користувача (рис. 12) відповідає за вивід результатів роботи програми на екран. Інтерфейс користувача розроблений у вигляді

вікна за допомогою технологій WinAPI та .NET APIc. Одним з пріоритетів при розробці програмного забезпечення було мінімізувати навантаження на клієнтську апаратну частину, отже модулі інтерфейсу користувача ніяких обчислень не відбувається.

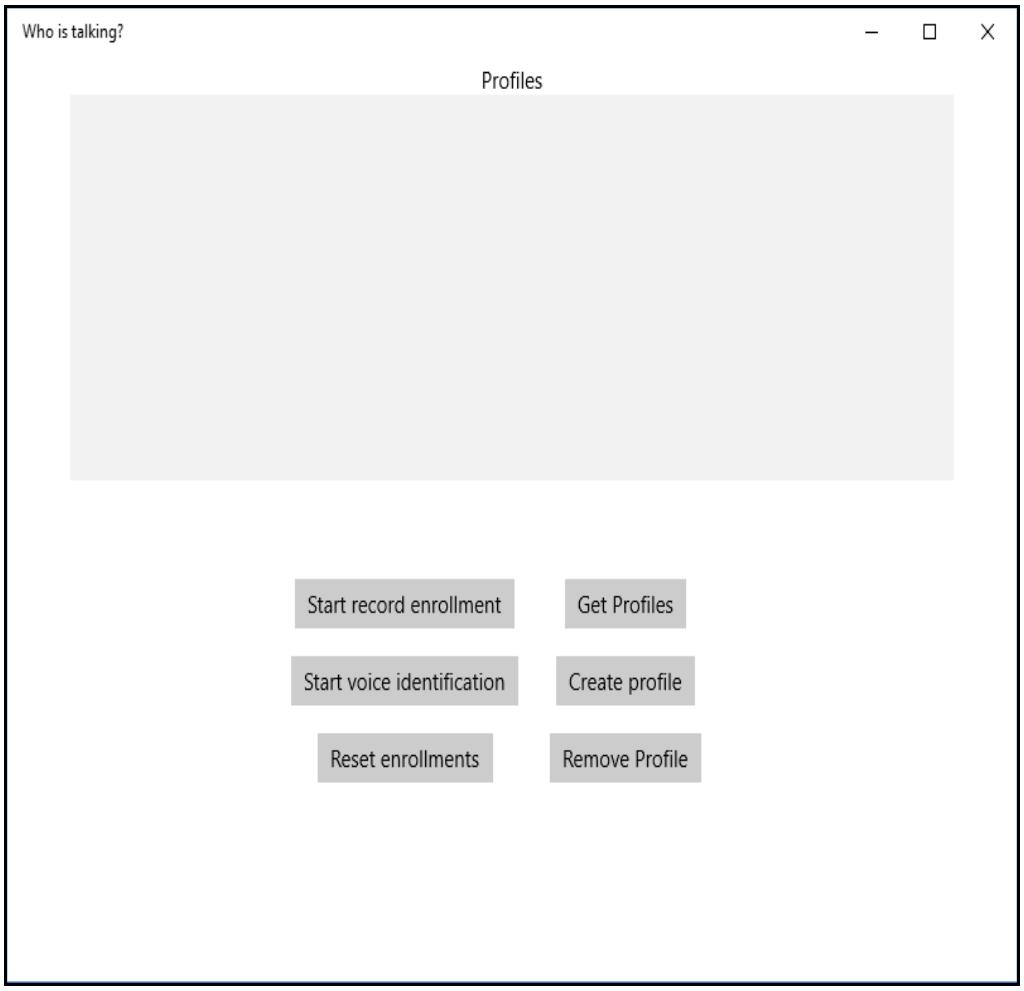

Рисунок 12 – Інтерфейс користувача

Як можна побачити з рис. 12 інтерфейс користувача не є складним, а, навіть, є дуже простим і зрозумілим.

У верхній частині програми у нас є вікно яке виводить список наших профілів, нижче основні 6 кнопок для роботи програми. Розглянемо їх більш детально.

Як тільки програма завершила свій запуск, необхідно натиснути кнопку «Get Profiles», коли користувач натискає кнопку, то функція GetProfiles() відправляє сигнал до контролеру, в свій час контролер обробляє інформацію

що користувач натиснув кнопку для отримання списку про файлів і функція GetProfiles_Controller() відправляє сигнал до моделі і вже там функція GetProfiles_Model() робить запит до бази даних «main» і з таблиці users отримує кожен профіль і через контролер повертає дані до інтерфейсу і виводить на екран про файли які були в базі даних. (рис. 13)

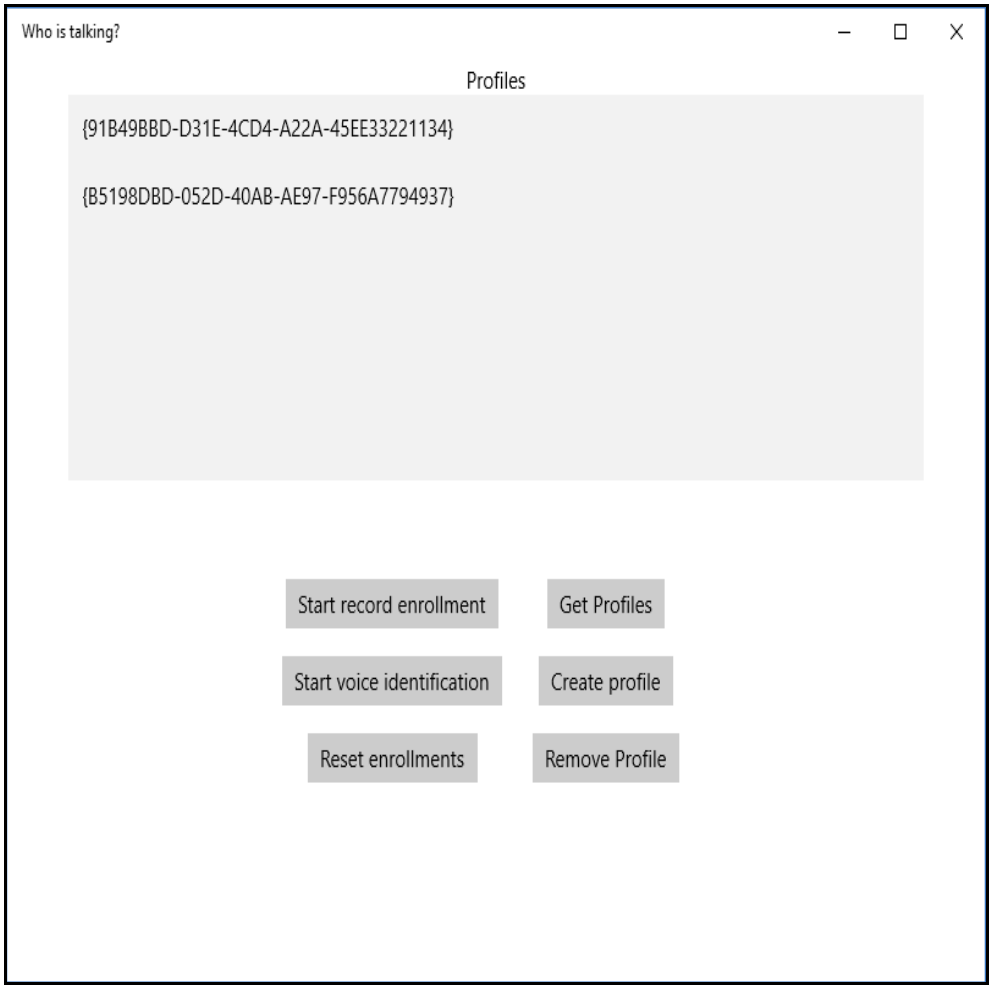

Рисунок 13 – Результат роботи кнопки «Get Profiles»

Кнопки «Create profile» і «Remove Profile» працюють за таким самим принципом як і «Get Profiles», але їх відмінність полягає в тому що перша кнопка працює за функцією SELECT на мові MySQL, «Create profile» викликає функцію INSERT(рис. 14)

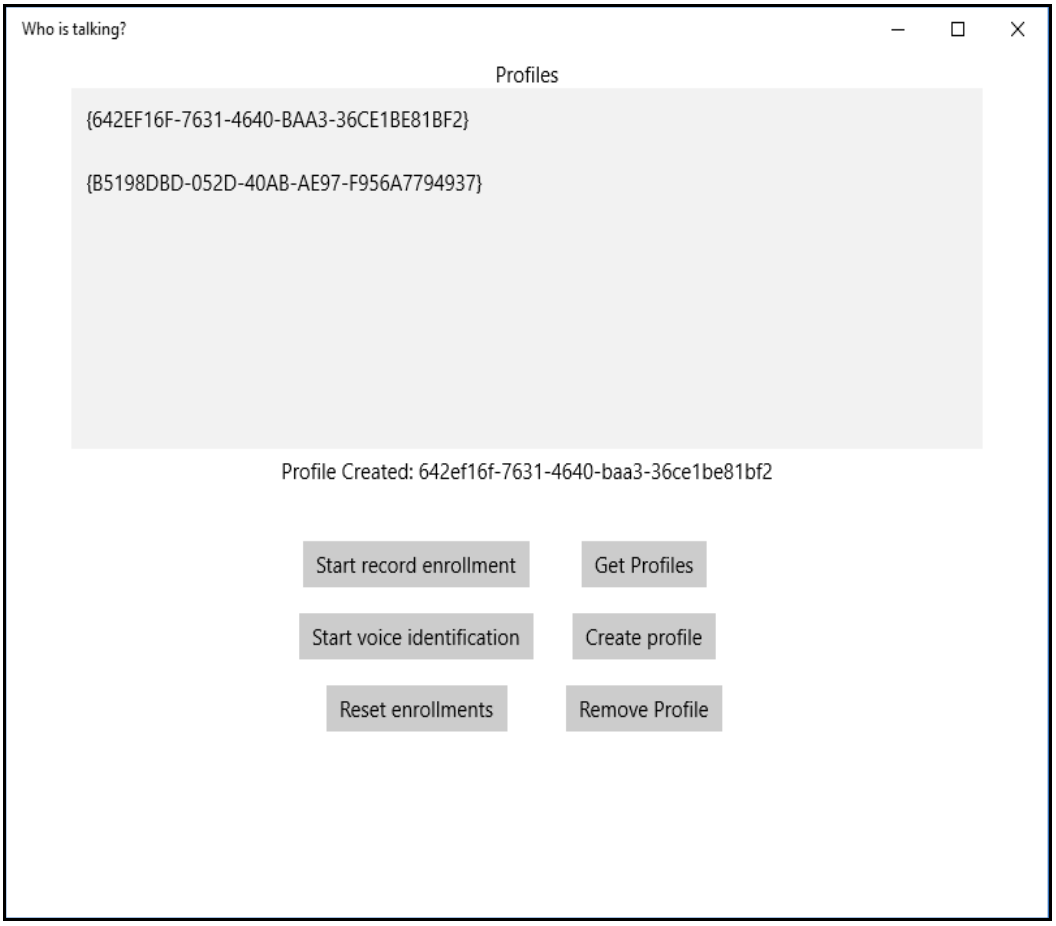

Рисунок 14 – створення профілю

«Remove Profile» виконує функцію DELETE на мові MySQL. (рис. 15)

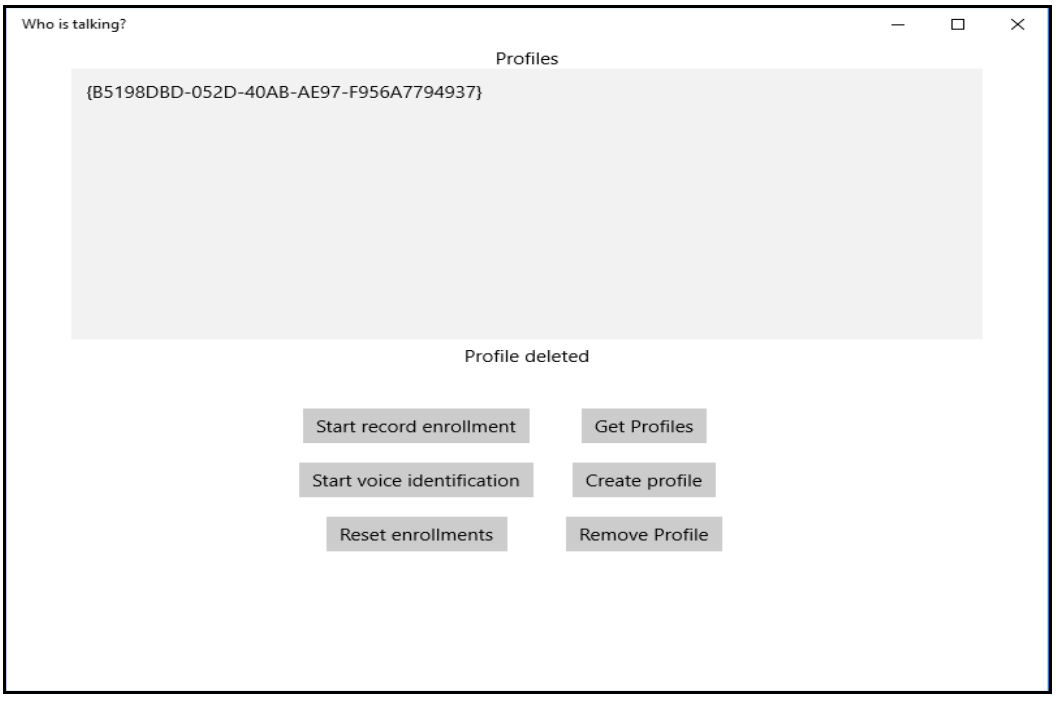

Рисунок 15 – Видалення профілю

Коли ми закінчили роботу з профілями, починаємо роботу з аудіо. Для того щоб ми змогли розпізнавати диктора, спочатку для нього потрібно створити шаблон. Для створення шаблону натискаємо кнопку «Start record enrollment». Коли ми натискаємо кнопку починає свою роботу функція btnRecordEnroll_Click(). Ця функція відсилає сигнал до функції у контролері RecordEnroll() для початку запису аудіо з мікрофону. Щоб закінчити запис необхідно натиснути кнопку яка змінилась з «Start record enrollment» на «Stop record enrollment». Після натискання даний аудіо файл відправляється до функції фільтрації шумів Noise_filter().

Після того як ми відсіяли шуми надалі данні переходять у функцію Average_amplitude(). Дана функція шукає середнє значення амплітуди, оскільки як говорилось у пункті 2.2.1 голос людини може мати різну гучність у певних умовах.

Коли ми встановили середнє значення гучності переходимо до наступної функції Fourier_transforms().

Я не зупиняюся на виборі частоти відцифрування сигналу, хоча цей вибір грає важливу роль в завданні моделювання сигналу. Тому надалі ми передаємо наші дані до функції Kaiser_window().

Після пройденого такого великого шляху відцифрування і накладання «вікна Кайзера» ми передаємо отриманні данні до функції у моделі New_template(), яка за допомогою функції INSERT на мові MySQL додає для обраного профілю шаблон голосу. На рис. 16 зображено вдалу процедуру запису шаблону для диктора. Для коректної роботи програми необхідно створити хоча б три шаблони для одного профілю.

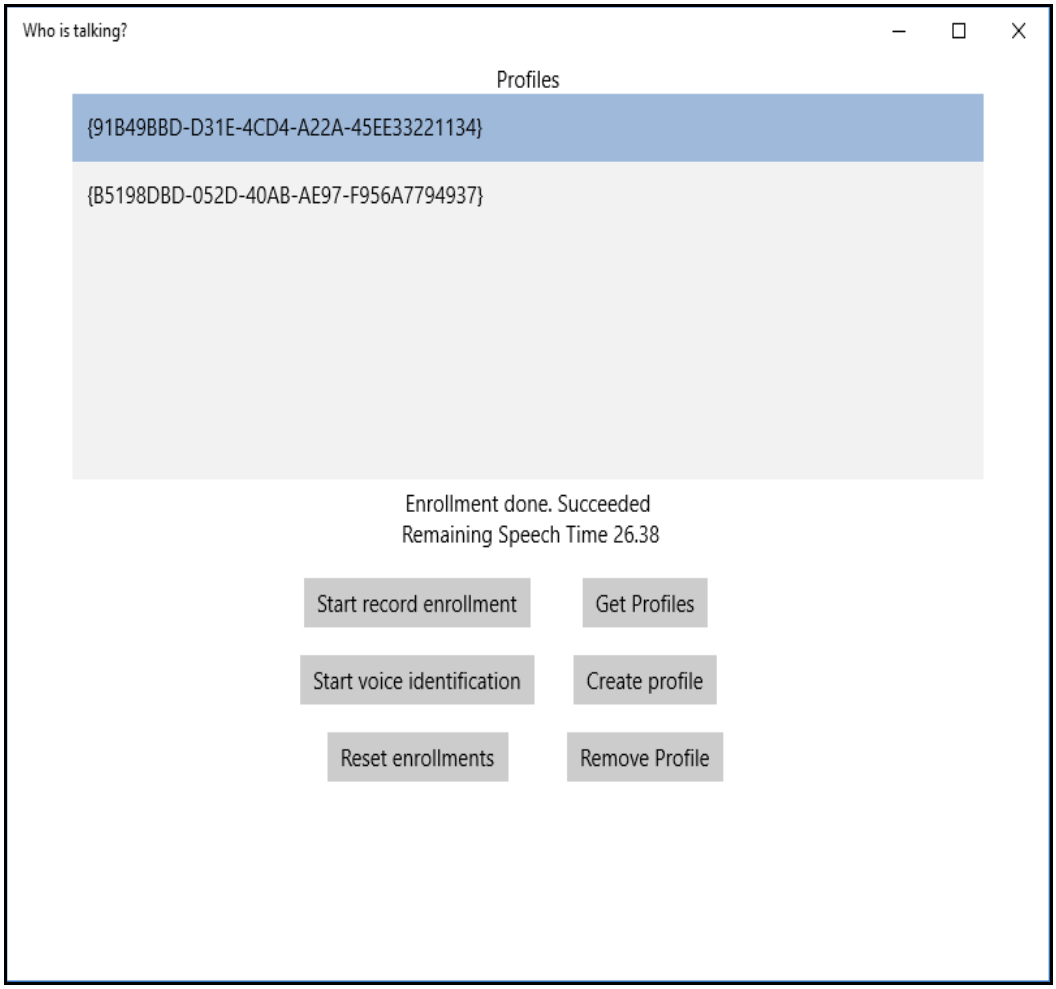

Рис. 16 – Результат додавання шаблону

Коли ми пройшли етап так званого «навчання» системи і створили декілька варіантів, переходимо до етапу розпізнавання диктора. Для цього необхідно натиснути кнопку «Start voice identification». Після того як ми натиснули кнопку починається запис голосу з мікрофона. Для закінчення запису необхідно натиснути кнопку яка змінилась з «Start voice identification» на «Stop voice identification». Коли ми натиснули цю кнопку, починається такий самий алгоритм як і для додавання шаблону диктора, а саме фільтрація шумів, знаходження середнього значення амплітуди(гучності), перетворення Фур'є, накладання «вікна Кайзера», але після цього дані не передаються до функції додавання нового шаблону, а передається до функції Comparison(), яка порівнює данні з еталонними образами.

Спочатку порівнюються по одному шаблону для кожного профілю, знаходиться максимально схожий шаблон, після цього порівнюються усі шаблони для цього диктора. Якщо середній відсоток більший за 90 то вважається що ідентифікація пройшла успішною і повертається профіль диктора, як показано на рис. 17, якщо менше то повертається результат що ідентифікація не пройшла успішно і просить повторити спробу або додати ще один шаблон для свого профілю(Рис. 18).

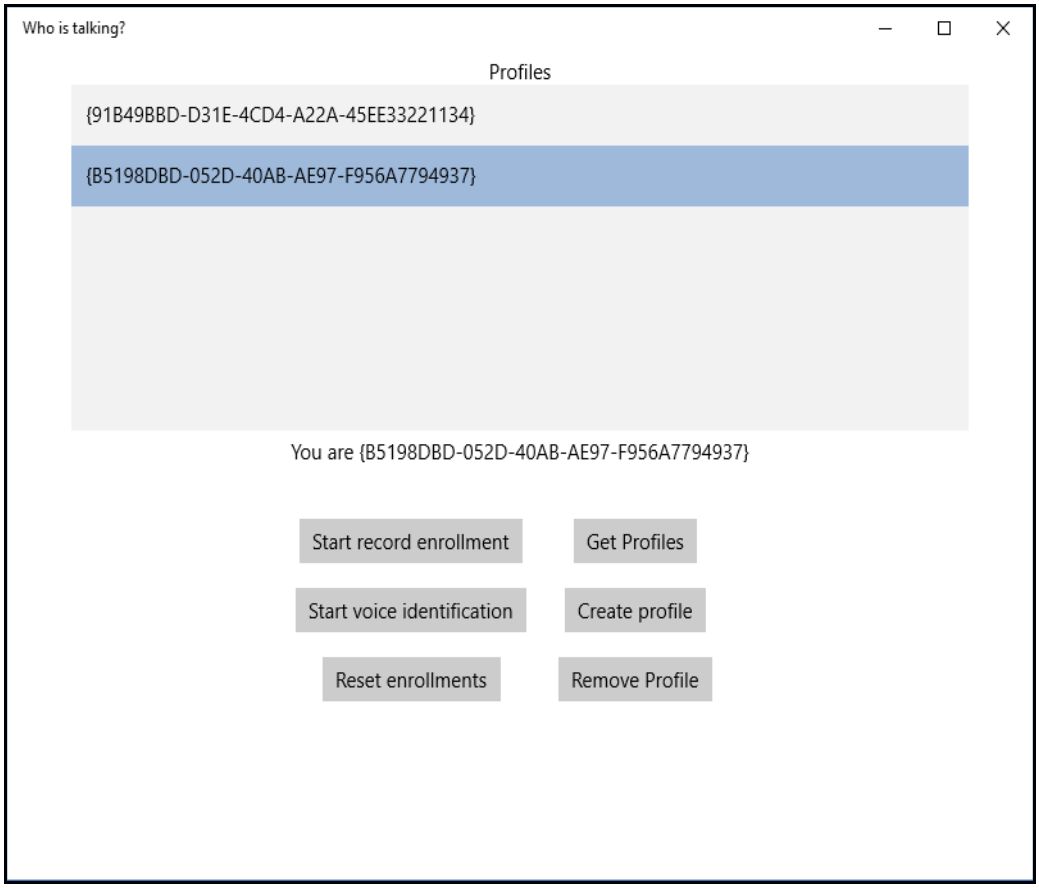

Рис. 17 – Результат успішної ідентифікації

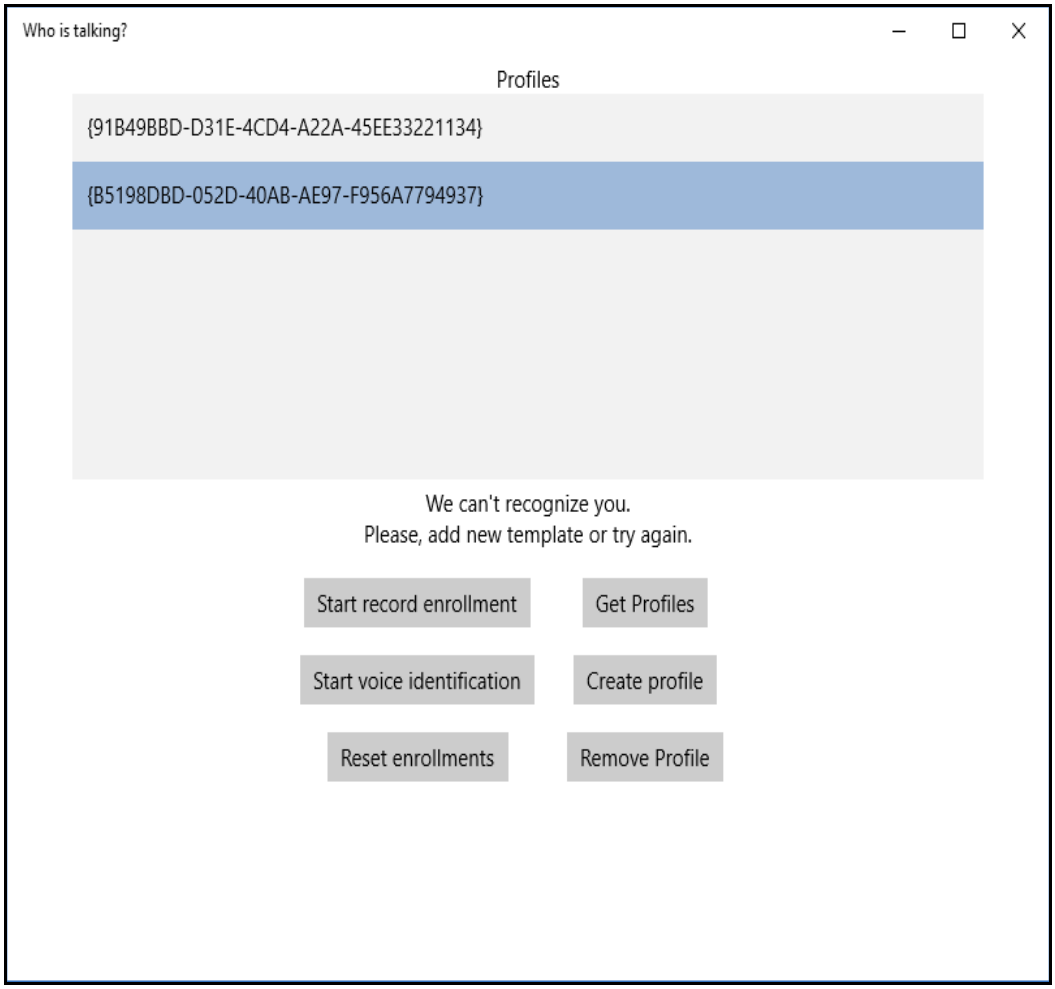

Рис. 18 – Невдала спроба розпізнавання

Також передбачена кнопка для видалення всіх шаблонів для обраного диктора. При натисканні кнопки «Reset enrollments» данні передаються до функції Reset_enrollments_Controller() яка передає сигнал до функції Reset_enrollments_Model() якій потрібно видалити усі шаблони для обраного профілю. Видалення відбувається за допомогою сценарію DELETE мови MySQL(рис. 19). Після видалення шаблонів можна записати нові шаблони для того самого диктора, або записати для іншої людини. Або можна видалити профіль диктора, хоча для цього не потрібно спочатку видаляти шаблони, програма передбачає функцію видалення шаблонів коли видаляєш диктора. Коли тиснемо кнопку «Delete profile» вона відправляє сигнал на ту ж саму функцію що і «Reset enrollments», але до того як видалить диктора.

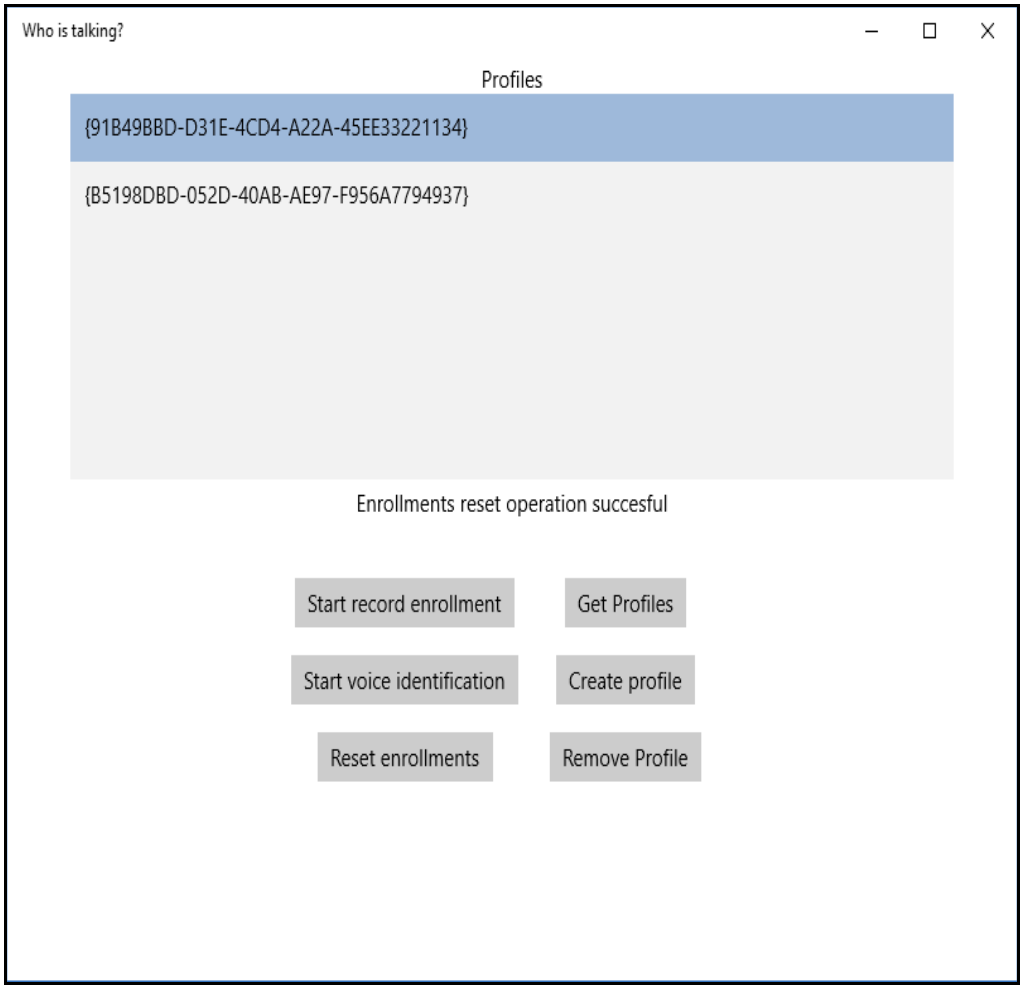

Рис. 19 – Видалення шаблонів диктора

Функції методів розробленого програмного забезпечення зазначені в таблиці 3.1.

Таблиця 3.1

Описання методів програмного забезпечення.

Назва	Опис
btnRecordEnroll_Click()	Метод обробки натиску на клавішу початку запису шаблона
finishEnrollment()	Закінчення запису шаблона і передачі інформації до контролера

Продовження табл 3.1

Назва	Опис
Noise_filter()	Функція для фільтрації шумів в аудіо файлі
Fourier_transforms()	Функція яка розкладає результат отриманий з попередньої функції у ряд Фур’є
Kaiser_window()	Формує масив остаточного результат у для подальшої обробки
New_template()	Сценарій який відповідає за передачу даних для додавання нового шаблону до бази даних
New_template_bd()	Робить запит до бази даних за допомогою функції INSERT для додавання шаблону
btnIdentify_Click()	Обробляє дію користувача на запит початку ідентифікації користувача
finishIdentification()	Обробляє дію користувача на запит закінчення ідентифікації користувача і передачу даних до контролера
Identification()	Функція початку обробки аудіо файлу для ідентифікації користувача

Продовження табл 3.1

Назва	Опис
Comparison()	Обчислення математичного очікування і порівняння отриманого результату з шаблонами що знаходяться в базі даних
Comparison_bd()	Запит до базі даних щоб отримати шаблон для порівняння
btnGetProfiles_Click()	Обробка дії користувача на запит отримання профілів
GetProfiles_Controller()	Передає сигнал до моделі для отримання профілів користувачів
GetProfiles_Model()	Запит до бази даних функцією SELECT для отримання усіх дикторів які є в базі
btnCreateProfile_Click()	Обробка на запит створення нового диктора
CreateProfile_Controller()	Відправка сигналу до моделі для запису нового диктора
CreateProfile ()	Відправляє функцію INSERT для запису нового користувача
btnRemoveProfile_Click()	Отримує сигнал від користувача на видалення диктора
RemoveProfile_Controller()	Дає сигнал моделі на видалення диктора

Продовження табл 3.1

Назва	Опис
RemoveProfile_Model()	За допомогою функції DELETE видаляє користувача за бази даних

Структура бази даних теж виявилась не складної і простою для використання. У таблиці 3.2 наведено структуру бази даних користувачів написаній на мові MySQL.

У таблиці було використано позначення «*» - означає що дане поле є ключем.

Таблиця 3.2

Структура таблиці користувачів.

Назва поля	Тип	Ключ
Ідентифікатор користувача	Лічильник	*
Ім’я користувача	Рядок	

Таблиця 3.3

Структура таблиці шаблонів.

Назва поля	Тип	Ключ
Лічильник шаблонів	Лічильник	*
Ідентифікатор користувача	Числовий	
Ідентифікатор шаблона	Числовий	
Значення спектральної потужності	Числовий	

Проаналізувавши які задачі можна вирішувати за допомогою розпізнавання диктора, я вирішив розробити план на доопрацювання продукту. Оскільки за допомогою ідентифікації користувача можна

видавати повну інформацію про диктора. Після того як ми отримуємо повну інформацію, ми можемо наприклад, розробити програму яка буде керувати системою голосом і можна буде сказати «Зателефонувати мамі» ідентифікувати хто це сказав і набрати який прив'язаний до цього користувача. Або викликати таксі для себе, або людини яка буде викликати і в базі даних будуть міститись основні місця для поїздок. Тобто сказавши «Викликати таксі до роботи», програма буде знати адресу роботи і викличе автомобіль. Можна, навіть, вдосконалити продукт який зберігав в собі банківську інформацію, наприклад, якщо нам потрібно сплатити якийсь товар чи послугу, ми голосом ідентифікуємо диктора і передаємо реквізити банківської картки і тим самим підтверджуємо що це саме цей користувач.

У таблиці 3.4 наведено приклад структури бази даних для контактів диктора.

Таблиця 3.4

Структура таблиці контактів диктора.

Назва поля	Тип	Ключ
Ідентифікатор контакту	Лічильник	*
Ідентифікатор диктора	Числовий	
Ім'я контакту	Рядок	
Прізвище контакту	Рядок	
Стать	Числовий	
Номер телефону	Числовий	
Електрона адреса	Рядок	

ВИСНОВОК

Під час виконання дипломного проекту було досліджено поставлену задачу і було виявлено, що в наш час немає дешевих, простих та зручних програм для розпізнавання диктора. Зі сформованих даних було зрозуміло що потрібна програма яка не потребувала би сильних навантажень на систему, мала основні функції для ідентифікації користувача.

Розпізнавання диктора рухається у різних напрямках, найбільш поширеними методами визначення ознак на сьогоднішній день є методи кепстральних коефіцієнтів та метод статичного підходу. Оскільки задача була розробити не навантажувальну систему, було обрано саме метод статичного підходу, який під час тестів показав високі результати роботи, не високий рівень використання апаратних засобів користувача комп'ютера завдяки апаратному пришвидшенню на процесорах Intel.

Для розробки програмного забезпечення було використано такі засоби: мова програмування високого рівня C#, .NET Framework, засоби для створення вікон WinAPI та .NET API.

Наведені технології чітко справились з задачею, був створений максимально простий інтерфейс для користувача і в той же час інформативний.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Аналіз, розпізнавання і інтерпретація мовних сигналів. Т.К. Вінцюк. 2010
2. Joseph Picone, Ruslan Popov. Методи моделювання сигналу в розпізнаванні мови. Кемерово. 2000
3. Чисельні методи Н.С. Бахвалов. Головна редакція фізико-математичної літератури вид-ва «Наука», М., 1975.
4. Голос як один з каналів невербальної комунікації І. Б. Єсін. Юр. психологія. 2008
5. Автоматичне розпізнавання говорить по голосу Г. С. Рамішвілі. Радио и связь, 1981. - 244 с. - Бібліогр. : С. 58 - 207.
6. Ревтов, Ю. І. Розпізнання людини по голосу і особливостям мови в процесі розслідування: дис. . канд. юр. наук. - Мінськ, 1998. - 45 с.
7. Теплов, Б. М. Розум полководця: Проблеми індивідуальних відмінностей / Б. М. Теплов.-М .: АПН РРФСР, 1961. Чистовіч, Л. А. Фізіологія мови. Сприйняття мови людиною / Л. А. Чис-товіч і ін .. Л.: Наука, 1976. - 388 с.
8. C# апплети [Електронний ресурс]. – 2015. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82. – Дата доступу: листопад 2015.
9. .NET API 2015 [Електронний ресурс]. – 2015. – Режим доступу: https://uk.wikipedia.org/wiki/.NET APIс. – Дата доступу: листопад 2015.
10. Розробка через тестування [Електронний ресурс]. – 2015. – Режим доступу: https://en.wikipedia.org/wiki/Test-driven_development. – Дата доступу: листопад 2015.
11. Model-View-Controller [Електронний ресурс]. – 2015. – Режим доступу: https://ru.wikipedia.org/wiki/Model-View-Controller. – Дата доступу: листопад 2015.

12. C#.NET Faces [Електронний ресурс]. – 2015. – Режим доступу: https://ru.wikipedia.org/wiki/C#.NET_Faces. – Дата доступу: листопад 2015.
13. Bukkase NAUDIO [Електронний ресурс]. – 2015. – Режим доступу: https://uk.wikipedia.org/wiki/Bukkase_Framework. – Дата доступу: листопад 2015.
14. Олпорт, Г. Становлення особистості: обр. праці. М.: Сенс, 2002. - 462 с.