3 РОЗРОБКА ПРОГРАМИ ДЛЯ РЕАЛІЗАЦІЇ МЕТОДИКИ РОЗПІЗНАВАННЯ МОВНИХ КОМАНД

За виконаним аналізом теми розпізнавання мовних команд у першому розділі та розробленої методики у другому розділі було створено програму з використанням мовних команд для підтвердження ефективності розробленої методики.

3.1 Вимоги до розробленої програми

Вхідні потоки: аудіо сигнал з мікрофону або аудіо файл.

Вихідні потоки: дія в залежності від вхідної команди або дія, якщо команда не була розпізнана.

3.1.1 Функціональні вимоги

Система повинна забезпечувати можливість виконання наступних функцій:

- додавання нової команди інтерфейсу;
- показ списку усіх команд;
- редагування та видалення існуючих команд інтерфейсу;
- редагування та скидання конфігурації програми;
- експорт у файл та експорт з файлу конфігурації програми;
- показ інформації про використані у системі методи отримання характерних ознак аудіо даних та класифікації команд;
- показ графіків характерних ознак аудіо даних;
- реагування на розпізнану команду;
- реагування на невдале розпізнавання команди.

3.1.2 Нефункціональні вимоги

Функціонування програми можливе за наступних вимогах:

- оперативна пам'ять об'ємом 2 Гбайт, не менше;
- внутрішню пам'ять, 800 Мбайт, не менше;
- процесор, 1.7 ГГц, не менше;
- операційна система Windows XP або вище, або Linux;
- інтерпретатор мови Python для версії 3.5 або вище;
- бібліотеки мови Python: NumPy, SciPy, LibROSA, Matplotlib, Scikit-learn, Shogun, PyMongo, PyQt5;
- СУБД MongoDB версії 4.0.0 або вище.

3.2 Вибір технічних засобів реалізації програми ЛМВ

3.2.1 Вибір мови програмування

В якості мови програмування було обрано мову Python. Python — це інтерпретована об'єктно-орієнтована мова програмування високого рівня. Інтерпретованість означає, що код програми не компілюється (перетворюється у машинні команди), а виконується рядок за рядком за допомогою спеціальної програми-інтерпретатора. Вона має надзвичайно легкий та доступний у розмінні синтаксис. Також на вибір вплинуло те, що код програми, через особливості язику Python, скорочується, по меншій мірі, у повтори рази, порівняно з такими мовами програмування, як C++ та Java. У порівнянні з усіма іншими мовами програмування, розробка на Python проводиться значно швидше завдяки такому синтаксису та великій кількості стандартних функцій і бібліотек, за допомогою яких можна виконати розробку майже будь-якої програми.

Велика кількість зовнішніх бібліотек, тобто розроблених різними людьми, які не є офіційними представниками розробки мови Python, у тому числі з

машинного навчання, дають змогу автоматизувати такі моменти роботи, як отримання характерних ознак аудіо даних або управління комп’ютером після розпізнавання команди. Загальна швидкість інтерпретації та виконання вихідного коду мовою Python є значно нижчою, ніж у C++ та C#, але за допомогою бібліотек, частина яких написана мовою C, можна отримати значне покращення у швидкості виконання.

3.2.2 Вибір бібліотек

3.2.2.1 Бібліотеки обробки та маніпуляції даними

NumPy (Numeric Python) – це бібліотека з відкритим вихідним кодом. Вони забезпечують функціонал, який можна порівняти з функціоналом MatLab. NumPy надає базові методи для маніпуляції з багатомірними масивами і матрицями. Будь-який алгоритм, який може бути виражений у вигляді послідовності операцій над масивами (матрицями) і реалізований з використанням NumPy, працює так само швидко, як еквівалентний код, що виконується в MatLab.

SciPy (Scientific Python) розширює функціонал NumPy величезною колекцією корисних алгоритмів, таких як мінімізація, перетворення Фур'є, регресія, і інших прикладних математичних технік, призначених для виконання наукових і інженерних розрахунків.

3.2.2.2 Бібліотеки отримання характерних ознак

LibROSA – це бібліотека, написана мовою Python, створена для аналізу музики та аудіо даних. У ній присутні усі необхідні компоненти для отримання характерних ознак з аудіо даних. Серед можливих до отримання ознак є, наприклад, мел-частотні кепстральні коефіцієнти, спектральні характеристики, швидкість перетину нульового рівня тощо.

3.2.2.3 Бібліотеки для демонстрації даних

Matplotlib – це бібліотека для візуалізації даних двомірних графіки. Вона дає змогу створення звичайних графіків, гістограм, точкових графіків, діаграм спектрів тощо. Користувач може вказати осі координати, решітку, додавати написи та пояснення, використовувати логарифмічну шкалу або полярні координати. Генеровані зображення графіків можуть бути використані в інтерактивній графіці, в наукових публікаціях, графічному інтерфейсі користувача, веб-додатках, де вимагається побудова діаграм тощо. Велика та досить легка до розуміння документація значно спрощує створення графіків. Також можливе створення трьомірних графіків за допомогою спеціального додаткового інструментарію.

Для реалізації графічного інтерфейсу користувача використовується PyQt5. Він реалізований як комплект Python-модулів, включає в себе близько 620 класів і 6000 функцій і методів. Це мультиплатформовий інструментарій, який запускається на більшості операційних систем, серед яких Unix, Windows і MacOS [7].

3.2.2.4 Бібліотеки для класифікації аудіо даних

Scikit-learn – це бібліотека з відкритим вихідним кодом для машинного навчання. Вона включає в себе різні алгоритми класифікації, регресії та кластеризації, у тому числі такі алгоритми, як машина опорних векторів, random forest, gradient boosting, k-means та DBSCAN. Бібліотека призначена для взаємодії з чисельними та науковими бібліотеками Python NumPy та SciPy.

Shogun – це безкоштовна бібліотека програмного забезпечення з відкритим кодом, написана на C++. Вона вміщує в себе численні алгоритми та структури

даних для проблем машинного навчання. Бібліотека має інтерфейси для мов програмування Octave, Python, R, Java, Lua, Ruby і C # за допомогою SWIG.

3.2.3 Вибір СУБД

В якості програми управління базою даних (СУБД) була обрана MongoDB через її відповідності потребам створюваної програми до зберігання великої кількості аудіо даних. Це є особливістю нереляційних баз даних – можливість збереження та ефективної маніпуляції великими об'ємами даних, до того ж вони забезпечують високу пропускну здатність.

MongoDB – це документоорієнтована СУБД з відкритим вихідним кодом. Вона написана на C++, тому її легко перенести на найрізноманітніші платформи, такі як Windows, Linux, MacOS та Solaris. Якщо реляційні бази даних зберігають рядки, то MongoDB зберігає документи. На відміну від рядків документи можуть зберігати складну за структурою інформацію. Документ можна уявити як сховище ключів і значень. Ключ являє собою просту мітку, з якою асоційований певний частина даних.

3.3 Розробка внутрішньої структури програми

Було розроблено інформаційну систему ЛМВ за допомогою мовних команд для реалізації методики розпізнавання команд. Структурна схема програми з використанням повних команд зображена на рис. 3.2.

Розпізнавання у розробленій системі може відбуватися одним з трьох основних способів: за допомогою повних аудіо команд, за допомогою визначення лише кореню з команди або за допомогою визначення команди за префіксом та коренем слова.

Структурна схема програми з розпізнаванням команд за допомогою префіксів та коренів команди зображена на рис. Г. Система з використанням

лише коренів команд є майже ідентичної до попередньої, єдина різниця у використанні замість бібліотеки частин слів бібліотеки коренів команд.

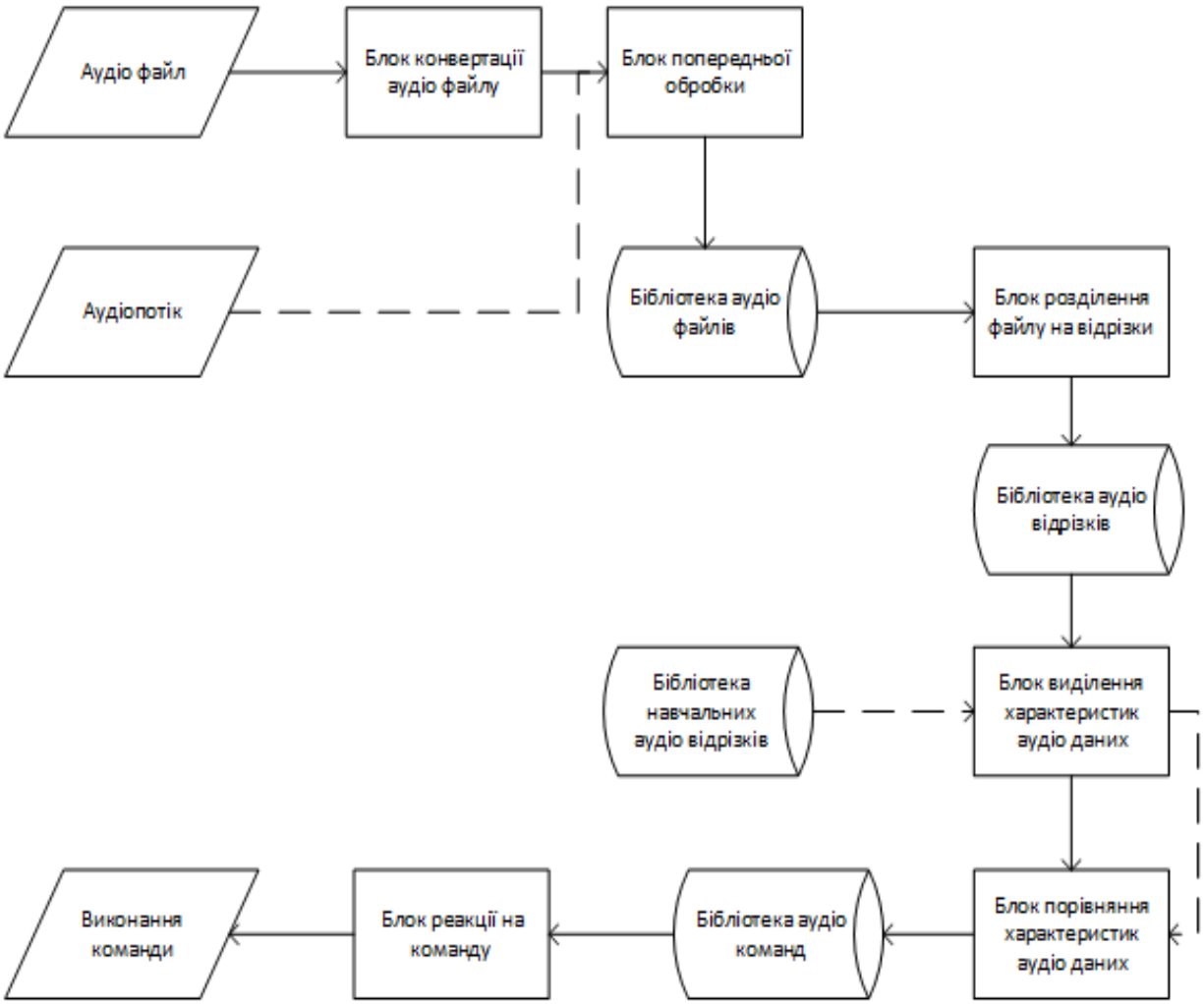


Рисунок 3.2 — Структурна схема програми при використанні повних команд

Вхідними даними для програми можуть бути аудіосигнал, отриманий, наприклад, з мікрофону або аудіо файл. Підтримуються наступні формати аудіо файлів у якості вхідних даних для розробленої інформаційної програми: wav, mp3, ac3, aac та ogg.

Блок конвертації аудіо файлу використовується, якщо вхідними даними є аудіо файл, але формат його відрізняється від файлу wav, з яким відбувається

робота у подальшому виконанні. Вхідними даними для цього блоку можуть бути тільки формати аудіо, які підтримуються розробленою системою. У блоці попередньої обробки виконується зменшення рівню шуму та видалення з аудіосигналу надлишкових даних, які не грають великої ролі у розпізнаванні та майже не сприймаються людським слухом. Це необхідно для зменшення кількості оброблюваних даних та прискорення їх обробки та розпізнавання команд.

Бібліотека аудіо файлів є колекцією елементів, яка працює з аудіо файлами або аудіосигналами. Це дані з невідомою внутрішньою структурою, тобто вони можуть містити декілька команд або не містити жодної.

Блок розділення файлу на відрізки займається проблемою, коли у файлі знаходиться декілька команд одразу. Файл ділиться на відрізки, між якими є тиша, яка продовжується певну кількість часу.

Бібліотека аудіо відрізків створюється після розділення файлу на деяку їх кількість. Ці дані утримують у собі рівно одну аудіо команду, але не обов'язково ця команда є серед відомих команд інтерфейсу.

Блок розділення аудіо відрізку на частини слова знаходить у команді префікс та корінь слова. Вони використовуються лише у другому варіанті розпізнавання команд системою.

Бібліотека навчальних частин слів є колекцією префіксів та коренів команд, за допомогою яких виконувалося навчання програми. Якщо користувач захоче додати нові команди до програми керування комп'ютером, ці дані про команду будуть добавлені у базу даних для можливості подальшого їх розпізнавання. Для адекватного розпізнавання користувачу слід виконати додавання додаткових варіантів промовляння цієї команди.

Блок виділення характеристик аудіо даних може приймати на свій вхід елементи з бібліотеки аудіо файлів, також з бібліотеки відрізків та частин слів. На виході цього блоку є одні з багатьох характеристик аудіо даних, наприклад їх спектрограма або мел частотний кепстральний коефіцієнт.

Блок порівняння характеристик аудіо даних приймає на вхід ці характеристики та порівнює з відповідними характеристиками аудіо даних у базі, завдяки яким проводилося навчання програми. Вихідним елементом є розпізнана команда з бібліотеки аудіо команд. Якщо жодна команда не має схожих характеристик з оброблюваною, користувач може добавити цю команду у базу в якості нової.

Бібліотека аудіо команд вміщає в себе усі команди, що можуть бути розпізнані системою. У кожної команди є певна дія або реакція, яка виконується при її розпізнаванні.

Блок реакції на команду виконує взаємодію з операційною системою та реагує на кожну з розпізнаних команд. У кожної команди є параметр важливості, а саме шкоди, до якої може призвести неправильне розпізнавання саме цієї команди. Тобто, коли розпізнається ця команда, але насправді на аудіосигналі немає такої команди. Отже при великій важливості команди та малій схожості цих команд є велика ймовірність, що команда не буде виконана. Адже краще буде, якщо користувач повторить команду більш чітко, ніж неправильно розпізнати команду, яка може бути небезпечною для даних користувача. Наприклад, при розпізнаванні команди «виконати виділення», що можна з легкістю прийняти за команду «виконати видалення» і видалити важливі документи. Тому і був створений параметр важливості команди.

Бібліотека навчальних аудіо відрізків є колекцією аудіо даних з бази, які використовуються при розпізнаванні команд за допомогою повних слів без скорочень до кореня тощо.

Блок обробник аудіо бази даних виконує усю роботу з базою даних, у якості якої виступає MongoDB через свою малогабаритність та швидкість взаємодії, що є критичним у подібній інформаційній системі.

3.3 Розробка алгоритмів функціонування програми

Аудіо дані можна отримати з мікрофону або з файлу. Алгоритм отримання

аудіосигналу з мікрофону зображено на рис. 3.3. Коли виникає звук на аудіосигналі, запис якого відбувається за кожною секундою, ці дані записуються у пам'ять разом з усіма іншими даними до того моменту, коли на наступній ділянці сигналу не буде нічого, окрім тиші.

Далі цей отриманий аудіосигнал передається на поділ за тишою для відділення можливих команд. Усі команди будуть розпізнані послідовно, або не будуть, в залежності від якості отриманого сигналу.

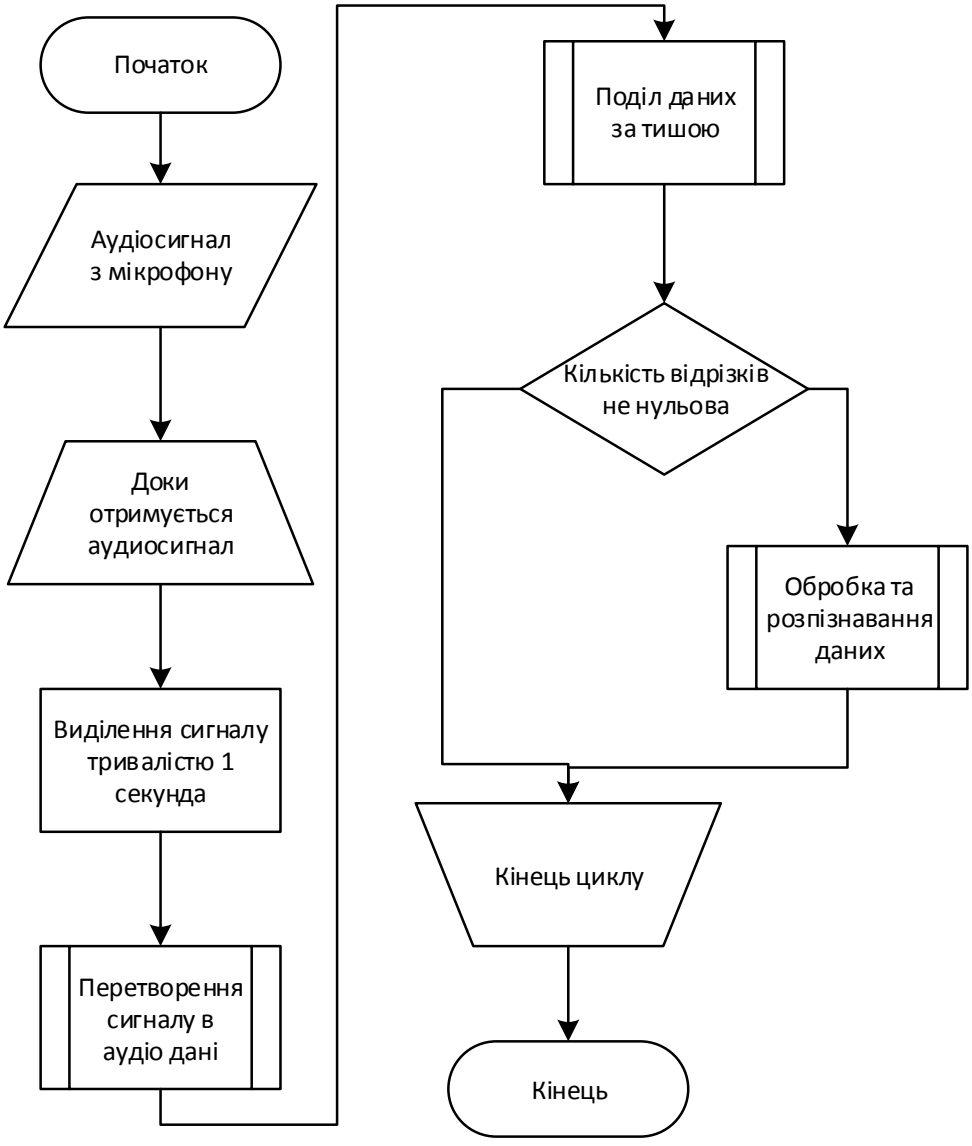


Рисунок 3.3 — Алгоритм отримання аудіо даних з мікрофону

Якщо відбулося виділення хоча б одної команди, вона передається на обробку та розпізнавання, адже на сигналі може бути не тільки команди та тиша, а також і шум, що може спричинити помилкове спрацьовування програми. Така обробка сигналу буде проводитися до моменту закінчення використання мікрофону користувачем.

Поділ аудіо файлів за тишою є важливим для визначення присутності команди в даних, він зображений на рис. 3.4.

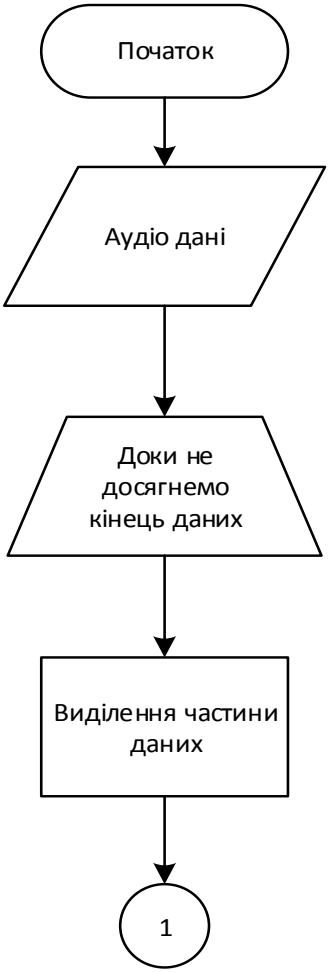


Рисунок 3.4 — Алгоритм поділу аудіо даних за тишою, аркуш 1

Алгоритм шукає початок та кінець ділянок тиші, яка менше деякого порогового значення, а потім видаляє цю тишу з аудіо даних. При цьому зазвичай залишається деяка невелика кількість тиші або додається власноруч для

збільшення розуміння слова при прослуховуванні.

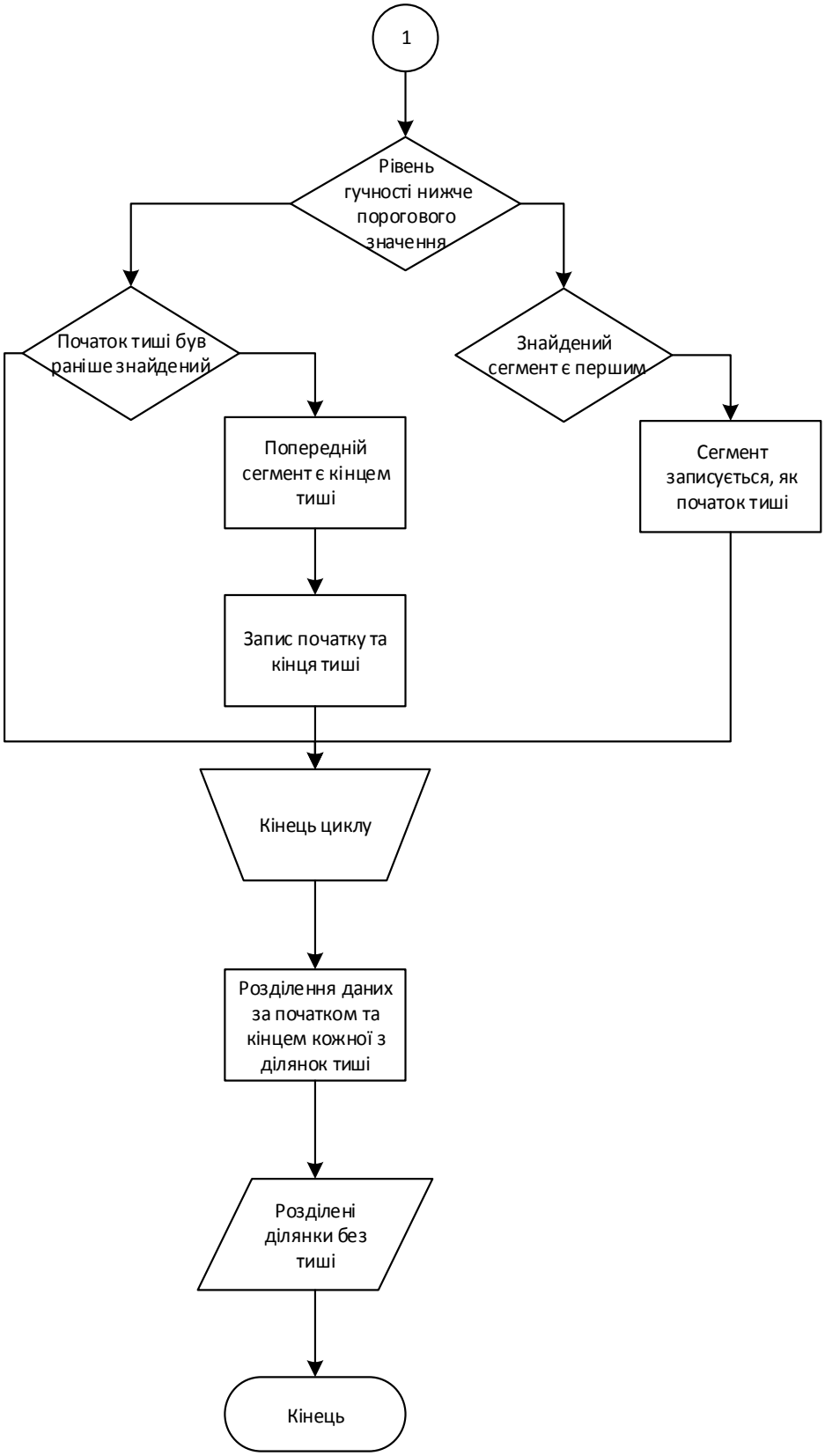


Рисунок 3.4 — Алгоритм поділу аудіо даних за тишою, аркуш 2

Це дозволяє розділити багато команд на аудіосигналі або аудіо файлі. Для цього між командами повинні бути зроблені паузи не меншої тривалості, ніж деяке порогове значення, яке задається у налаштуваннях програми.

Наступний алгоритмом, що виконує основну функціональність програми, а саме розпізнавання команди зображено на рис. Г. Необхідність зменшення об’єму даних виникає тоді, коли у налаштуванні вибраний цей пункт. Не всі дані необхідні для розпізнавання, звичайно їх більше, ніж потрібно. Деякі дані можуть навіть погано вплинути на розпізнавання через те, що вони несуть інформацію про людину, яка говорить, її унікальні характеристики промовляння звуків, темп мовлення тощо. Отже видалення надлишкової інформації може покращити розпізнавання команд у деяких випадках у залежності від якості вхідних аудіо даних.

Ще одною головною функцією програми є можливість додавати власні команди для використання, як інтерфейс ЛМВ. Алгоритм додавання команди представлений на рис. 3.5.

Згідно з алгоритмом, під час виконання додавання команди у базу даних, у першу чергу перевіряється чи є подібна команда у системі, тобто застосовується спроба розпізнати цю команду.

Якщо розпізнавання вдалося, то користувач попереджається про її наявність та робить вибір: продовжити додавання команди, при цьому нова команда створена не буде, а усі екземпляри еталонів, що він надасть, будуть добавлені до вже існуючої команди; або припинити додавання команди, що спричинить видалення вже доданого екземпляру еталона.

Невдале розпізнавання команди означає, що нова команда буде створена у базі даних. Після цього користувач повинен надати достатню кількість екземплярів еталонів команди для забезпечення більшої адекватності та достовірності розпізнавання.

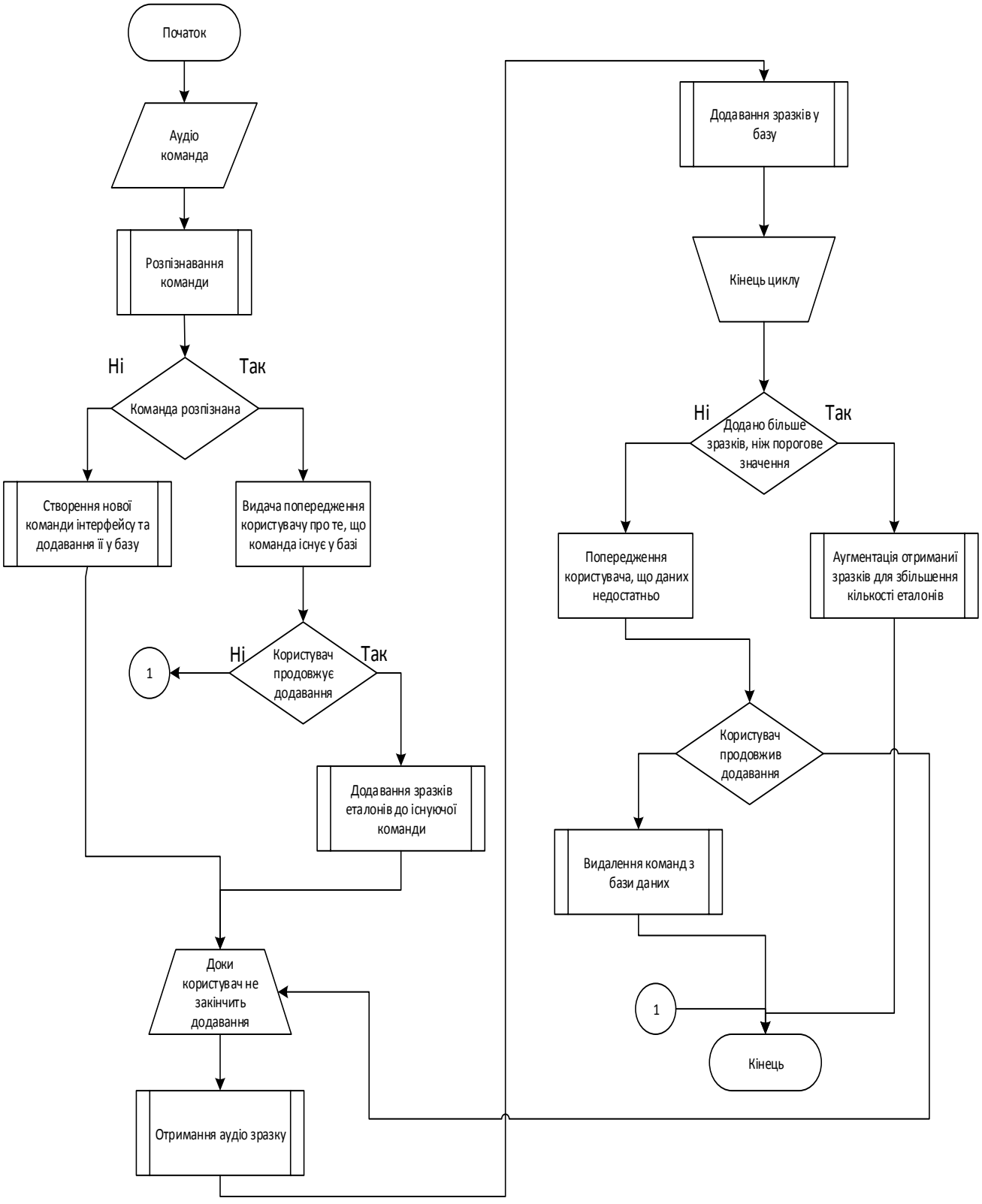


Рисунок 3.5 — Алгоритм додавання команди у систему

Якщо еталонів недостатньо, користувач буде повідомлений про це та повинен буде надати додаткові еталони або відмінити усі внесені зміни, тобто

видалити створену нову команду та усі її екземпляри промовляння.

3.4 Розробка класової структури програми

Усі класи програми можна умовно розділити на три групи:

- entities — ті, що є сутностями представлення аудіо даних;
- libraries — ті, що є колекціями для роботи з сутностями даних;
- performers — ті, що виконують певну діяльність, яка залежить від необхідності виконати обробку або зробити щось інше з початковими даними;

Коротка характеристика класів та їх ролі в розробленій інформаційній системі, яка зображена на рис. Д:

AudioConvertor — виконує конвертацію аудіо файлів з одного формату на інший.

AudioLibrary — загальній клас, від якого наслідуються усі інші сутності бібліотек. Є колекцією з даними, дає змогу зберігати та оброблювати дані різних класів.

AudioFilesLibrary — бібліотека файлів або аудіосигналів. Дає змогу завантажити файл з жорсткого диску або отримати дані з аудіопотоку, також підтримує одночасне додавання багатьох файлів.

AudioUnitsLibrary — бібліотека аудіо частин, яка дає змогу отримати дані з бази або розділенням файлу.

DividedUnitsLibrary — бібліотека коренів та префіксів. Дає можливість завантажити дані з бази або розділити аудіо частини на компоненти слів.

AudioCommandsLibrary — бібліотека команд, які підтримує система. Вони завантажуються з бази програми.

AudioEntity — загальній клас для сутності аудіо даних, від якого наслідуються усі інші.

AudioFile — сутність файлу або аудіопотоку з даними.

AudioUnit — сутність аудіо частини.

Root — сутність кореню слова.

Prefix — сутність префіксу слова.

AudioCommand — сутність аудіо команди. На відміну від усіх попередніх сутностей, клас не наслідується від AudioEntity, бо в цій сутності немає безпосередньо аудіо даних.

AudioPreprocessor — виконує попередню обробку аудіо даних для зменшення їх об’єму та поліпшення розпізнавання.

AudioFeatureExtractor — виконує отримання вихідних характеристик за різними параметрами з аудіо даних.

AudioRepresentator — виконує показ даних у вигляді графіків у двомірному та тримірному просторі.

AudioDataComparator — виконує порівняння поданих характеристик від двох сутностей та знаходить ті, що схожі. Вихідним параметром є розпізнана команда для інтерфейсу.

GUICreator — клас відповідає за створення та роботу з графічним інтерфейсом користувача. За допомогою нього з’являється можливість використання усіх функцій програми. Взаємодія з ними можлива завдяки об'єкту класу MainProcessor.

MainProcessor — працює з усіма іншими класами за допомогою створених об'єктів цих класів. Він отримує дані з графічного інтерфейсу та передає до інших класів на обробку. Таким чином відбувається внутрішня робота програми.

3.5 Послідовність передачі даних між елементами програми

Послідовність передачі даних під час розпізнавання команди розглядалася лише для двох випадків: використання у базі даних повних слів команд та використання частин слів, тобто префіксів та коренів. Третій випадок, де

використовуються лише корені слів не розглядався через велику схожість на другий випадок. Передачу даних при першому випадку зображено на рис. Е, другий випадок — на рис. Є.

Згідно з діаграмою, аудіо файл зчитується з жорсткого диску та передається для конвертації у формат wav, з яким працює система. У даному випадку вважається, що користувач використовує аудіо файли замість аудіосигналу з мікрофону і ці файли мають розширення одного з підтримуваних конвертором форматів, але не wav. Після конвертації файл записується знову на диск у вказану директорію.

Потім цей файл зчитується з диску та передається до класу, що виконує попередню обробку даних, тобто зменшує рівень шумової похибки та позбавляється надлишкових даних. Далі файл записується у бібліотеку аудіо файлів, тобто колекцію для зберігання і роботи з ними.

Далі аудіо дані передаються до класу, що виконує поділ усіх даних на частини, між якими є тиша. Тобто кожна з цих частин є можливою командою. Усі розділені частини записуються у бібліотеку аудіо частин (аудіо одиниць). З бази даних зчитуються початкові аудіо частини для їх подальшого порівняння з отриманими частинами.

Потім усі аудіо частині (ті, що отримані з бази даних та ті, що отримані з файлу) подаються до класу, що виконує генерацію та вилучення характеристик з даних. Це дозволяє оцінити схожість даних за різноманітними параметрами та вибрати найбільш схожу команду.

Ці характеристики подаються до класу, що зрівнює схожість кожної з пар аудіо частин різними способами та визначає найбільш схожу команду або визначає, що такої команди в базі даних немає.

Отримана команда подається до класу, що виконує дію, пов'язану з цією командою. Дія безпосередньо залежить від операційної програми, на якій використовуються система.

Передача даних при використанні частин слів досить схожа на

використання повних команд. Різницею є те, що аудіо частини не є найменшим представником аудіо даних. Це місце займають корені та префікси слів.

Вони отримуються шляхом подальшого ділення аудіо частин за допомогою спеціального класу. Ці частини слова зберігаються у бібліотеці частин слів та саме з них отримуються вихідні ознаки для порівняння за допомогою ДЧ.

У базі даних записані ці початкові частини слів, вони використовуються тим же чином, що і аудіо частини у попередньому випадку. Після порівняння обирається найбільш схожі частини слова, при цьому ці частини відповідають певній команді. Таким чином виконується розпізнавання.

На рис. 3.6 зображена передача даних при редагуванні команди.

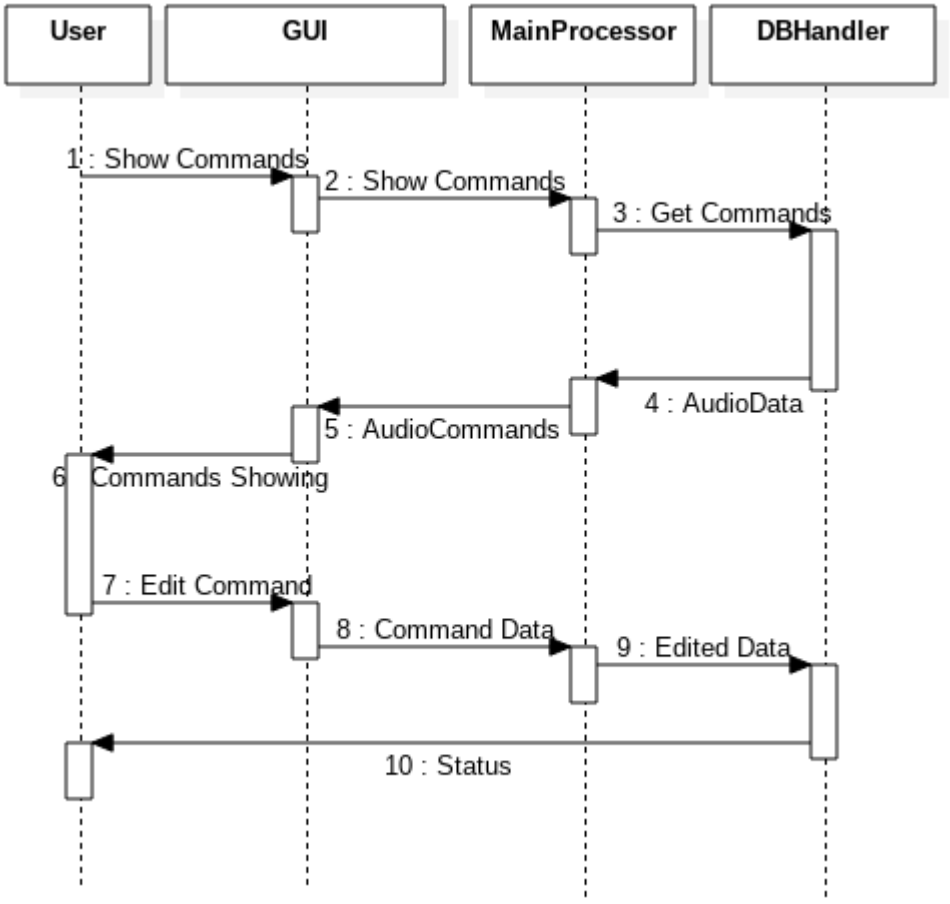
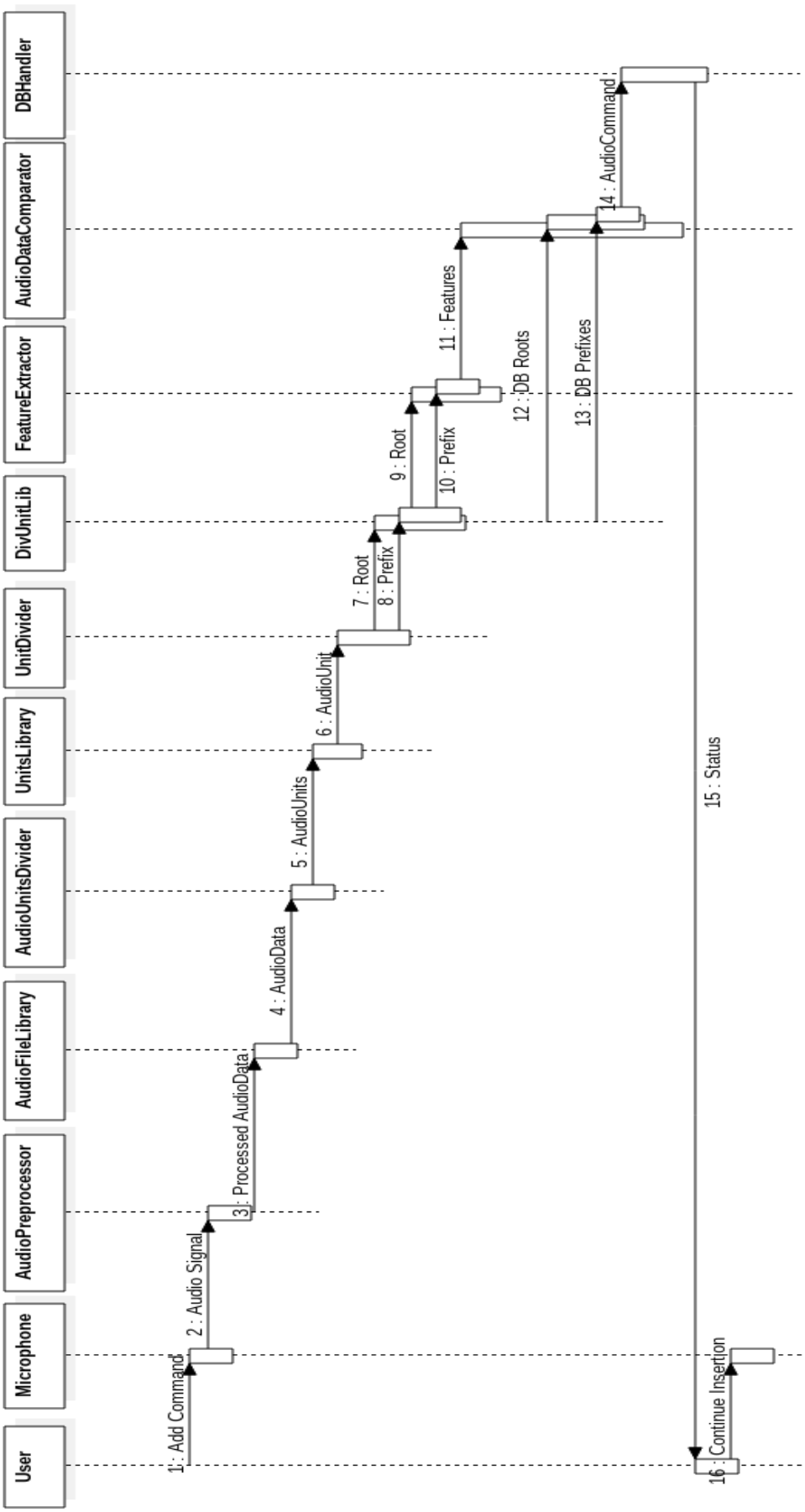


Рисунок 3.6 — Передача даних для функціональності редагування команди

На рис. 3.7 передачу даних при додаванні нової команди у систему:



					ІС КРМ 122 035 ПЗ	Лист
Змін.	Лист	№ докум.	Підпис	Дата		2

При цьому спочатку відбувається спроба розпізнати цю команду, а потім, в залежності від результатів розпізнавання, відбувається додавання її у базу даних.

3.6 Інструкція з користування системою

3.6.1 Загальний вигляд графічного інтерфейсу програми

Робота користувача з програмою починається з відкриття головного вікна програми. Воно має різний вигляд в залежності від налаштувань програми. Вигляд за замовчуванням представляю собою звичайний заголовок, головне меню та дві кнопки для розпізнавання команд: кнопка, яка активує роботу з мікрофоном та кнопка, що починає діалог пошуку та відкриття файлу з жорсткого диску користувача. Головне вікно програми за замовчуванням для режиму «Розпізнавання» представлене на рис. 3.3:

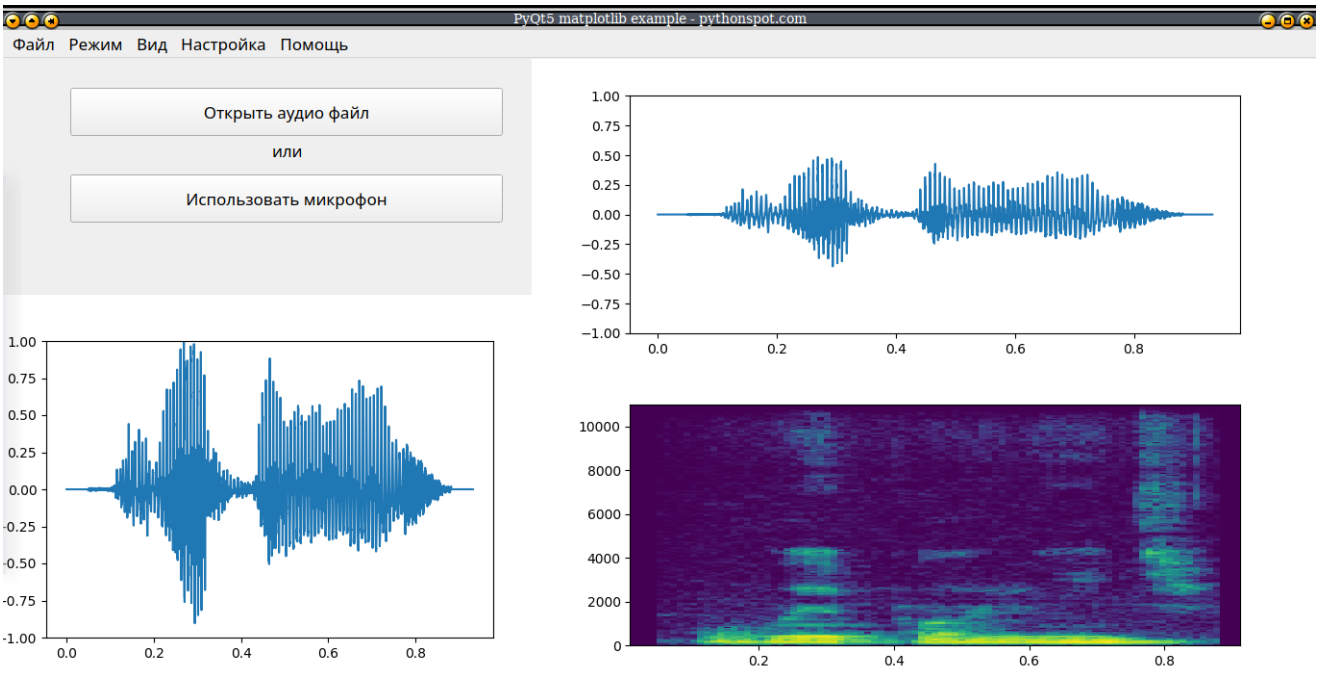


Рисунок 3.3 — Головне вікна за замовчуванням для режиму «Розпізнавання»

Основна взаємодія з системою відбувається за допомогою головного меню.

На вкладці «Файл» можна побачити наступні варіанти взаємодії:

- «відкрити» — відкриття файлу з жорсткого диска користувача;
- «закрити» — зупинити роботу з файлом та зробити головне вікно програми таким, яке воно було до відкриття файлу;
- «відкрити директорію» — завантажити усі файли з директорії для розпізнавання;
- «вихід» — вихід з програми.

Вкладка «Вид» має наступні варіанти взаємодії:

- «показати хвильовий сигнал» — показати хвильовий графік за даними з тестового файлу у головному вікні;
- «показати спектрограму» — показати графік спектрограми даних;
- «показати порівняльну ознаку» — показати графік за ознакою, яка у цей момент використовується для розпізнавання команд.

Вкладка «Режим» дає можливість переключатись на різні режими роботи програми, їх існує три:

- «розпізнавання» — стандартний режим роботи програми, де розпізнається команда та, у залежності від налаштування, виконується якась дія чи лише видається повідомлення про розпізнану команду;
- «додавання» — додавання нової команди у базу, вона потребує обов'язково зробити декілька зразків промовляння команди для подальшого адекватного її розпізнавання;
- «показ списку» — виконується показ списку усіх команд з можливістю видалення або редагування кожної з них.

Вкладка «Налаштування» дає можливість працювати з налаштуваннями програми:

- «редагування» — виконується відкриття вікна з користувальницькими налаштуваннями, наприклад, обиранням класифікатора для використання;
- «сброс налаштувань» — повернення налаштувань за замовчуванням;
- «імпорт» — завантаження файлу з налаштуваннями;

– «експорт» — імпортування налаштувань у файл за вибором користувача з можливістю подальшого використання.

Вкладка «Допомога» дає коротку характеристику з використовуваних ознак для виділення з аудіо файлів та використовуваних класифікаторів для розпізнавання команд у системі.

3.6.2 Приклад виконання функції розпізнавання команди з аудіо файлу

1) відкрити головне вікна програми, у меню «Режим» обрати «Розпізнавання»;
2) у меню «Вид» обрати всі з представлених там варіантів;
3) у меню «Налаштування» обрати пункт «Редагування»;
4) у ньому зняти галочку з пункту «Виконувати реакцію на команду» та натиснути «Зберегти»;
5) натиснути на кнопку «Відкрити файл» у головному вікні;
6) у відкритому діалозі обрати необхідний файл та натиснути кнопку «ОК»;
7) відкриється нове вікно з розпізнаною командою.

3.6.3 Приклад виконання функції додавання нової команди

1) відкрити головне вікно програми, у меню «Режим» обрати «Додавання»;
2) у меню «Вид» обрати всі з представлених там варіантів;
3) натиснути кнопку «Використання мікрофону» та сказати команду;
4) в залежності від того, чи є така команда в базі, буде виданий результат;
5) якщо такої команди немає, виконувати додавання щонайменше ще чотири рази;
6) натиснути кнопку «Завершення додавання»;
7) якщо екземплярів команди добавлено достатньо, вона буде додана у систему.

3.7 Висновки до третього розділу

У результаті виконання третього розділу магістерського дослідження було спроектовано та створено систему людино-машинної взаємодії з комп’ютером, що використовує мовні команди для роботи.

Проектування почалося з аналізу необхідних варіантів використання програми та побудови діаграми прецедентів, що визначила основні можливості використання програми: додавання, редагування, видалення та розпізнавання команд одним з трьох способів.

Наступним кроком при проектуванні стало створення структурної схеми програми для кожного зі способів розпізнавання команд. За цими структурними схемами було створено загальний алгоритм розпізнавання команди.

Подальше проектування складалося з створення класів, що виконують усю функціональність програми та опис їх призначення. Потім за цими даними було створено загальну діаграму класів. Усі класи програми можна розділити на три групи: сутність, що зберігає в собі якісь дані, бібліотека, що працює з сутностями та є колекцією елементів, та класи-виконавці, що безпосередньо виконують якусь функцію.

Наступним було створено діаграми передачі даних для більшого розуміння того, як та куди передаються дані між класами. За допомогою цих діаграм було створено прототип програми, а також було створено графічний інтерфейс користувача для більш зручнішого доступу до даних. Користувачам було надано інструкцію з використання графічного інтерфейсу програми та усіх її функцій з прикладами.