МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ЗАСТОСУНКУ ДЛЯ АКУСТИЧНОЇ ДІАГНОСТИКИ ДВИГУНА АВТОМОБІЛЯ НА БАЗІ PYTHON»**

Виконав: студент 4 курсу, групи 6.1211-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
О.Д. Ключков
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Горбенко В.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Ключкову Олександру Дмитровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка застосунку для акустичної діагностики двигуна автомобіля на базі Python

керівник роботи Горбенко Віталій Іванович, к.ф-м.н, доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Відомості.
2. Розгляд методів, технологій, платформ.
3. Програмна реалізація.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.01.2025	
2.	Збір вихідних даних.	24.01.2025	
3.	Обробка методичних та теоретичних	10.02.2025	
	джерел.		
4.	Розробка першого розділу.	03.03.2025	
5.	Розробка другого розділу	04.04.2025	
6.	Розробка третього розділу.	16.05.2025	
7.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
8.	Захист кваліфікаційної роботи.	18.06.2025	

Студент _____ О.Д. Ключков
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.І. Горбенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка застосунку для акустичної діагностики двигуна автомобіля на базі Python»: 43 с., 31 рис., 1 табл., 24 джерела, 1 додаток.

АУДІОДАНІ, СПЕКТРАЛЬНИЙ АНАЛІЗ, LIBROSA, PYQT, PYTHON.

Об’єкт дослідження – автоматизація та візуалізація аналізу аудіоданих, збережених у файлах стандартних аудіо форматів.

Мета роботи: розробка застосунку спектрального аналізу аудіоданих від двигуна автомобіля, що працює, засобами мови програмування Python.

Методи дослідження: методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

В результаті роботи створено додаток акустичної діагностики двигуна автомобіля на мові програмування Python. Для цього було проаналізовано додатки зі схожою функціональністю, розглянуто їх переваги, зручність, недоліки. Перед розробкою додатка, було створено UML діаграми. При реалізації використані бібліотеки Python: Librosa, SciPy, PyQt, Matplotlib та NumPy.

SUMMARY

Bachelor's qualifying paper "Development of an Application for Acoustic Diagnostics of a Car Engine Using Python": 43 pages, 31 figures, 1 table, 24 references, 1 supplement.

AUDIO DATA, LIBROSA, PYQT, PYTHON, SPECTRAL ANALYSIS.

The object of the study are automation and visualization of audio data analysis stored in standard audio format files.

The aim of the study is development of an application for spectral analysis of audio data from a running car engine using the Python programming language.

Research methods: methods of collecting and analyzing software requirements, methods of modeling, designing, constructing, and testing software.

As a result of the work, a program for acoustic diagnostics of a car engine was created using the Python programming language. For this, applications with similar functionality were analyzed, their advantages, ease of use and disadvantages were considered. UML diagrams were created before creating the program. When implementing the application, Python libraries were used: Librosa, SciPy, PyQt, Matplotlib and NumPy.

ЗМІСТ

Завдання на кваліфікаційну роботу……..….2
Реферат 4
Summary 5
Вступ..... 7
1 Відомості 8
1.1 Особливості аудіоаналізу діагностики двигуна автомобіля..... 8
1.2 Практична реалізація і напрямки використання аудіо діагностики 9
2 Розгляд методів, технологій, платформ 12
2.1 Бібліотеки для реалізації додатку..... 12
2.2 Діаграма прецедентів..... 14
2.3 Діаграма класів 15
2.4 Діаграма послідовностей..... 16
3 Програмна реалізація..... 18
3.1 Аналіз програми 18
3.2 Робота програми..... 23
Висновки 35
Перелік посилань..... 36
Додаток А Лістинг додатку..... 39

ВСТУП

Ремонт автомобіля – доволі складний та дорогий процес. Однак серйозних проблем автомобіля можна запобігти, якщо своєчасно їх діагностувати. Одним із методів раннього діагностування проблем – є метод акустичної діагностики. Деякі досвідченні механіки можуть розпізнавати ці проблеми на слух, але можливості автоматичних програм аудіо діагностики перевищують фізичні можливості людини. Автоматична аудіо діагностика заснована на обробці звукових сигналів із застосуванням методів цифрової обробки сигналів та машинного навчання.

Перспективи автоматичної діагностики аудіо великі. Попит не обмежується лише автолюбителями, а й включає допомогу в контролі якості продукції автопідприємств, що випускається, а також перевірка якості наданих послуг з ремонту автомобілів на станціях техобслуговування.

Завдяки широким можливостям Python та його багатому набору бібліотек, таких як Librosa, SciPy, PyQt та Numpy розробка автоматизованих систем діагностики стає доступною та ефективною. В 2017 році науковці Масачусетського Технологічного Інституту заявили о початку розробки додатку для аудіо діагностики автомобілів. В 2020 році чеський концерн Škoda auto заявив про створення додатку Sound Analyser для діагностики автомобілів Škoda. На поточний час 245 дилерів ŠKODA в 14 країнах в даний час тестують Sound Analyser в реальних умовах.

1 ВІДОМОСТІ

1.1 Особливості аудіоаналізу діагностики двигуна автомобіля

Аудіоаналіз – це процес дослідження та ідентифікація якогось явища, предмета чи істоти за допомогою звука. За допомогою аудіоаналізу легко ідентифікувати: вид птахів за їх співом, людину за голосом або несправність за звуком який видає механізм під час роботи. Очевидною перевагою аудіоаналізу є те, що єдине що потрібно для дослідження – це звук.

Як правило для аудіоаналізу використовують спектрограми. Спектрограма – це графік, який дозволяє візуалізувати звукові сигнали. Так ось X представляє час, ось Y – частоту, колір або яскравість пікселів – це амплітуда (або інтенсивність) звуку на даній частоті у конкретний момент часу. Завдяки спектрограмам можна навіть аналізувати звуки які не чутні вуху людини, наприклад ультразвук. Так компанія UE Systems inc. Elmsford, NY спеціалізується на виробництві сканерів та програм які здатні аналізувати ультразвук для пошуку несправностей підшипників та пробоїв електромереж [1].

Окремим випадком аудіоаналізу є акустична діагностика несправностей двигуна автомобіля. Традиційна діагностика автомобіля тривалий, дорогий та трудомісткий процес який можна значно полегшити за допомогою акустичної діагностики. Простий дзвін, тріск, свист, скрип та інші нехарактерні для нормальної роботи двигуна звуки в автомобілі можуть свідчити про певні проблеми [2].

Наприклад, раптовий стукіт низького тону в нижній частині двигуна, що посилюється при навантаженні та збільшенні оборотів двигуна, може бути спричинено розбалансуванням корінних підшипників. Таке буває при сильно зношених двигунах. Часто проблема посилюється падінням рівня оливи (загоряється сигналізатор аварійного рівня оливи), що може привести до перегріву двигуна і виходу його з ладу.

Ритмічний, дзвінкий, металевий звук середнього тону в середній частині блоку циліндрів може бути викликаний шатунними підшипниками. Якщо зі збільшенням навантаження звук зростає і навіть падає рівень оливи – їхати далі не можна, треба евакуювати автомобіль на станцію техобслуговування.

Свист відразу після запуску двигуна та різкого натискання на газ говорять про знос ременя генератора. Його потрібно затягнути чи замінити на новий.

Своєчасна діагностика цих проблем зменшує об’єм робіт і вартість ремонту. Деякі досвідченні механіки можуть розпізнавати ці проблеми на слух, але можливості автоматичних програм аудіо діагностики перевищують фізичні можливості людини.

Хоча акустична діагностика двигуна не замінить повноцінної діагностики яку може надати традиційна автомайстерня, однак акустична діагностика може значно полегшити ідентифікацію та своєчасний ремонт несправностей.

1.2 Практична реалізація і напрямки використання аудіо діагностики

Існує багато напрямів для застосування програм автоматичної аудіо діагностики. Попит не обмежується лише автолюбителями, а й включає допомогу в контролі якості продукції автопідприємств, що випускається, а також перевірка якості наданих послуг з ремонту автомобілів на станціях техобслуговування.

На поточний момент в багатьох країнах розробляють програми які здатні за допомогою акустичної діагностики знаходити несправності автомобіля. Вже існують програми для діагностики двигуна, наприклад: Autune, Sound Analyser від Škoda, Bosch audio analyzer або ClingClanger [3-5, 7].

Так в 2017 році науковці Масачусетського Технологічного Інституту заявили про початок розробки додатку для аудіо діагностики автомобілів. Додаток буде аналізувати звук та вібрації автомобіля, для виявлення проблем, навіть до того, як транспортний засіб покаже якісь очевидні симптоми. Таким чином, додаток для смартфона, розроблений командою Массачусетського технологічного інституту, може заощадити середньому водієві $125 на рік та скоротити загальну витрату палива на кілька процентних пунктів. Для вантажівок економія може становити до $600 на рік, не рахуючи вигоди від уникнення поломок, які можуть призвести до втрати доходу [3].

Також в 2020 році чеський концерн Škoda auto заявив про створення додатку Sound Analyser для діагностики автомобілів Škoda. Програма записує шуми, що видаються відповідним транспортним засобом під час його руху, та порівнює їх із збереженими звуковими зразками. У разі виникнення будь-яких невідповідностей програма використовує алгоритм, щоб визначити, яка проблема виникла в автомобілі і як її можна усунути. Одже, цей додаток зможе скоротити загальний час на виявлення та усунення поломок. Вже зараз 245 дилерів ŠKODA в 14 країнах тестують Sound Analyser в реальних умовах [4]. До переваг цього додатку можна віднести високу точність та можливість встановити безкоштовно на телефон, однак, програма працює тільки з автомобілями ŠKODA.

В 2019 році компанія Bosh, відомий виробник автомобільних та промислових технологій, випустила програму Sound Analyzer для автомобільних двигунів [5]. Програма Sound Analyzer від Bosch була представлена в рамках проєкту SoundSee, розробленого у співпраці з компанією Astrobotic Technology Inc. Спочатку ця технологія призначалася для аналізу звуків на Міжнародній космічній станції (МКС), де вона використовувалася для діагностики поломок обладнання по звуку за допомогою штучного інтелекту та машинного навчання [6]. Хоча спочатку SoundSee була розроблена для космічних досліджень, згодом Bosch адаптувала цю технологію для комерційного використання, включаючи аналіз

звуків в автомобільних двигунах та інших промислових програмах. До переваг можна віднести високу точність та підтримка вантажних та промислових двигунів. До недоліків можна віднести її коштовність та доступність тільки для великих підприємств.

В 2014 році компанія Noise Tech створила програму ClingClanger яка може ідентифікувати поломки автомобіля по запису звука. В ClingClanger знаходиться бібліотека автомобільних шумів з якими програма порівнює записи роботи, та на основі яких програма оцінює стан двигуна [7]. До переваг можна віднести можливість роботи без доступу в Інтернет та підтримку великої кількості моделей автомобілів. До недоліків можна віднести те що в безкоштовній версії обмежена функціональність.

Переваги та недоліки кожного з додатків наведено в таблиці 1.1.

Таблиця 1.1 – Переваги і недоліки додатків

	Переваги	**Недоліки**
Autune (мобільний додаток)	– простий у використанні – доступно на смартфоні	– низька точність – невелика база даних поломок
Sound Analyser ŠKODA (мобільний додаток)	– доступно на смартфоні – висока точність	– працює тільки з автомобілями ŠKODA
Bosch audio analyzer	– висока точність – підтримка вантажних та промислових двигунів	– дуже дорогий, – недоступний для приватних осіб
ClingClanger	– працює офлайн – підтримка великої кількості моделей автомобілів	– більшість функцій присутні тільки в платній версії

2 РОЗГЛЯД МЕТОДІВ, ТЕХНОЛОГІЙ, ПЛАТФОРМ

2.1 Бібліотеки для реалізації додатку

При реалізації додатку аудіо діагностики двигуна було використано багато бібліотек, наприклад: Librosa, SciPy, PyQt, NumPy та Matplotlib.

Librosa – це бібліотека python для аналізу музики та аудіо. Вона надає методи та класи, необхідні для створення додатків, які працюють зі звуковою інформацією. Так в додатку використовується функція librosa.stft(). Короткочасне перетворення Фур'є або Short-time Fourier transform (STFT) представляє сигнал у частотно-часової області шляхом обчислення дискретних перетворень Фур'є (DFT) в коротких вікнах, що перекриваються [8]. Окрім функції librosa.stft() в додатку використовується функція librosa.load(), яка завантажує аудіофайл як часовий ряд із плаваючою точкою [9]. А також функція librosa.amplitude_to_db() для перетворення амплітудної спектрограми на спектрограму в масштабі децибел [10].

SciPy – це бібліотека для мови програмування Python, призначена для виконання наукових та інженерних розрахунків. Зокрема в додатку використовується метод scipy.spatial.distance.cosine() який відповідає за косинусну схожість, і обчислює косинусну відстань між двома одновимірними масивами [11].

PyQt – це великий набір бібліотек Python, для розробки інтерфейсу користувача. Існує 3 версії: PyQt6, PyQt5 та PyQt4, що підтримують відповідні версії фреймворку Qt [12], в даному проєкті використовується версія PyQt5. При розробці поточного додатка були використані такі класи PyQt5, як QApplication, QMainWindow, QFileDialog, QMenuBar, QMenu, QAction, QLabel. Класи QApplication та QMainWindow відповідає за потік управління, а також за основні налаштування програми з інтерфейсом користувача [13, 14]. Клас QFileDialog надає діалогове вікно, що дозволяє користувачам вибирати файли чи каталоги [15]. Класи QMenuBar та QMenu використовують для

створення та налаштування горизонтальної панелі меню [16, 17]. Клас QAction використовують для представлення випадаючого списку при натисканні на кнопки меню, а також для додавання функціоналу кнопкам меню [18]. Нарешті, клас QLabel використовується для додавання графічного елементу Label, який використовується для представлення для представлення зображень та текстової інформації. При представленні зображень використовується метод класу setPixmap() який дозволяє додавати зображення в графічний елемент Label. Для додавання тексту використовується метод setText() [19].

Інша бібліотека, що використовується в розробленому додатку, це бібліотека **Matplotlib** це комплексна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python. Зокрема в додатку використовується такі методи класу pyplot як matplotlib.subplots, matplotlib.colorbar, title, xlabel, ylabel [20]. Ці методи використовуються для створення і візуалізації графіка спектрограми і спектра. Для візуалізації спектра також використано метод yscale() з параметром ‘log’ для встановлення логарифмічного масштабу осі Y.

Бібліотека **NumPy** додає підтримку великих багатовимірних масивів і матриць, а також великий набір високорівневих математичних функцій для роботи з цими масивами. В проєкті використовуються такі функції бібліотеки як fft.fft, abs, fft.fftfreq. Функція numpy.fft.fft() використовується для обчислення одновимірного дискретного перетворення Фур’є [21]. Функція numpy.absolute, яка також має скорочення numpy.abs, використовується в додатку для обчислення амплітуди [22]. А функція fft.fftfreq використовується для обчислення частоти [23].

Для реалізації архітектури програми було обрано шаблон програмування MVC (Model-View-Controller). Шаблон програмування MVC передбачає поділ системи на три взаємопов’язані частини: модель, вид та контролер [24].

Модель відповідає за дані, обробку даних та структуру програми. Модель реагує на команди з контролера та видає інформацію.

Вид візуалізує інформацію, яку він отримує від моделі. Вид відображає дані на рівні інтерфейсу користувача.

Контролер забезпечує взаємодію з системою: обробляє дії користувача, перевіряє отриману інформацію та передає її моделі. Контролер визначає, як програма буде реагувати на дії користувача.

2.2 Діаграма прецедентів

Для опису програми було використано Use Case Diagram (діаграма варіантів використання) (див. рис. 2.1). Діаграма варіантів використання в UML – це діаграма, що відображає відносини між акторами та прецедентами і є складовою моделі прецедентів, що дозволяє описати систему на концептуальному рівні.

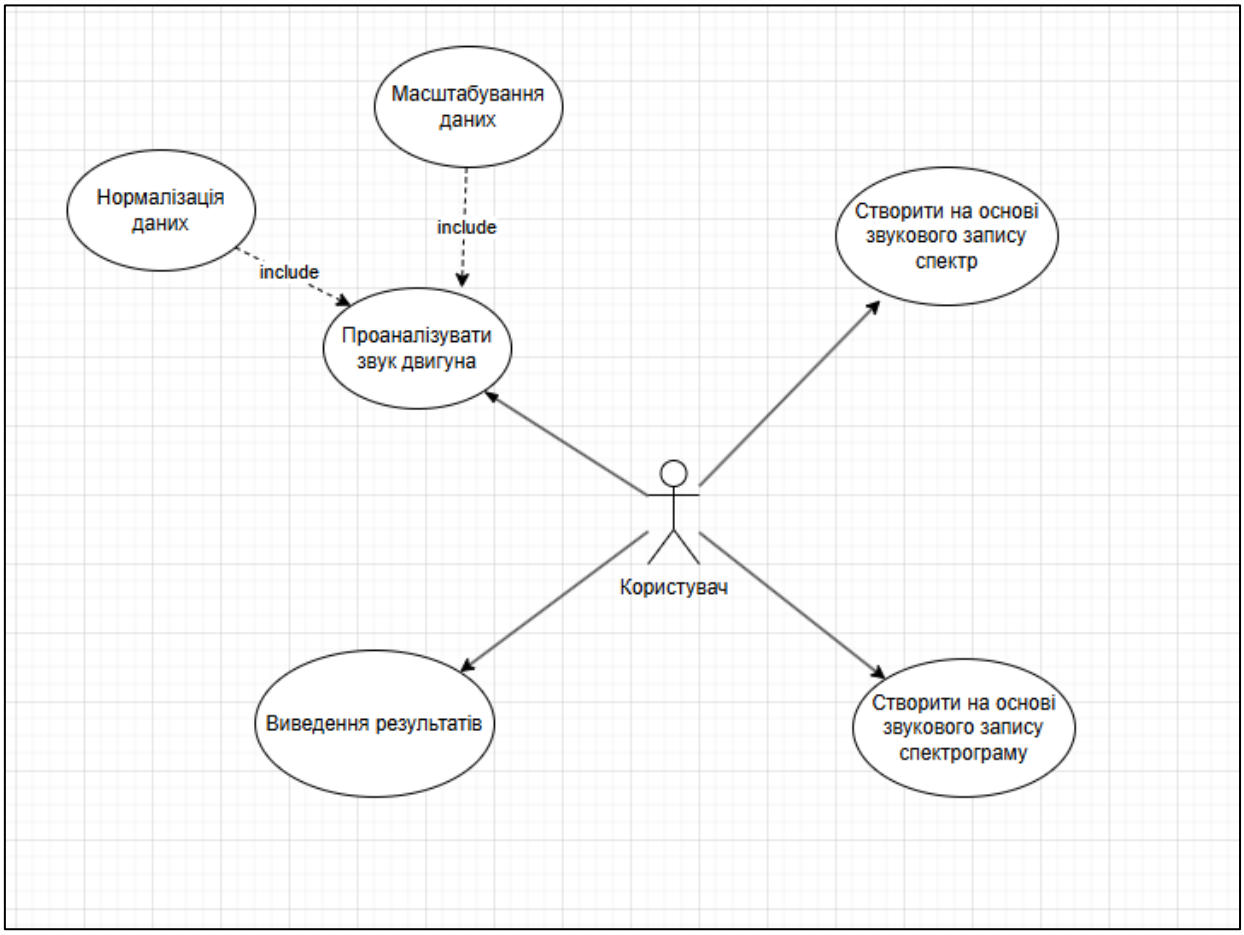


Рисунок 2.1 – Діаграма прецедентів

Актор – це безліч логічно пов’язаних ролей, що виконуються при взаємодії з прецедентами чи сутностями. Актором може бути людина чи інша система, підсистема чи клас. На рисунку зображено один актор – користувач.

Прецеденти (варіанти використання) – це опис поведінки системи, коли вона взаємодіє з актором. На малюнку можна побачити що актор-користувач має наступні варіанти використання: «Створити на основі звукового запису спектрограму», «Створити на основі звукового запису спектр», «Проаналізувати звук двигуна», «Виведення результатів». При виконанні прецеденту «Проаналізувати звук двигуна» також буде виконуватись «Нормалізація даних» та «Масштабування даних».

2.3 Діаграма класів

Для опису програмної реалізації було обрано діаграму класів (див. рис. 2.2), яка представляє структуру системи шляхом моделювання її класів, властивостей, операцій і зв’язків між об’єктами.

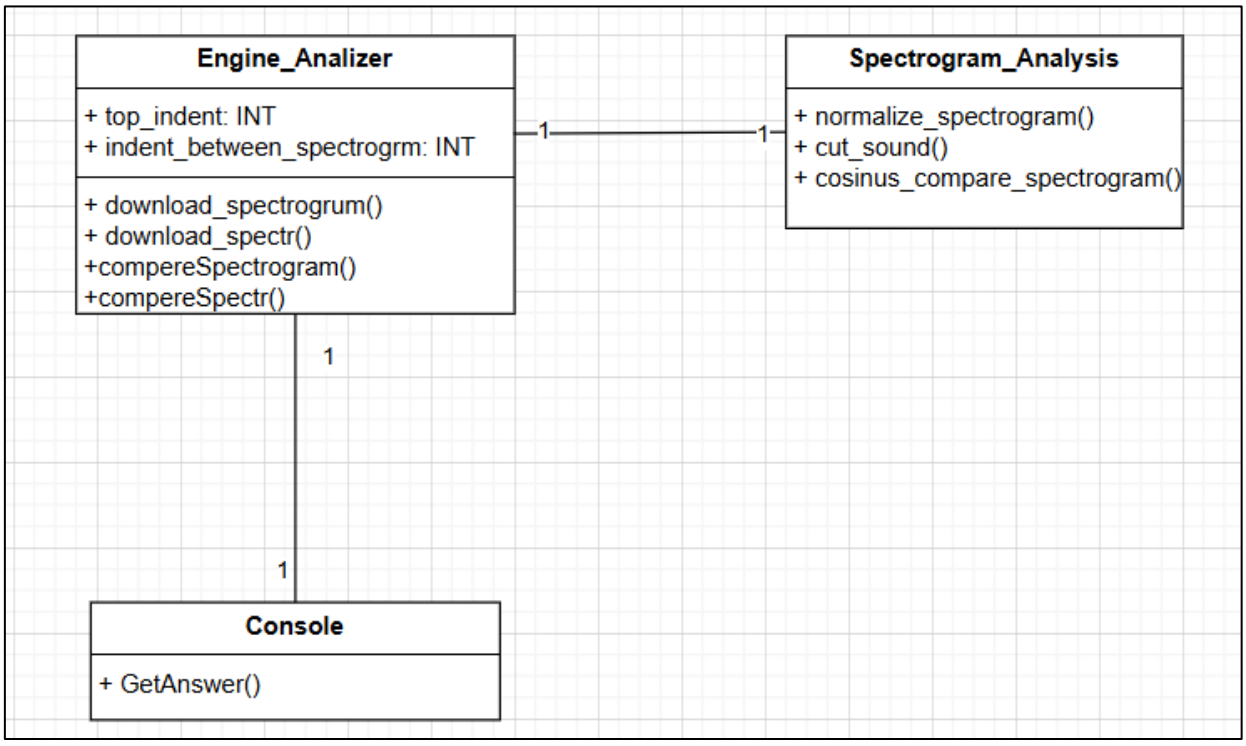


Рисунок 2.2 – Діаграма класів

На діаграмі, класи представлені у вигляді прямокутників, які містять три відділення: Верхня частина містить назву класу. Середнє відділення містить атрибути класу. Нижній відсік містить операції, які може виконувати клас. Класи з'єднані зв'язками. На діаграмі представлено три класи: Engine_Analizer, Spectrogram_Analysis, Console. Більш детальний опис методів кожного класу наведено в розділі 3.1 Аналіз програми.

Зв'язок (або відношення) – це загальний термін, що охоплює конкретні типи логічних зв'язків, які зустрічаються на діаграмі класів та на діаграмі об'єктів. UML визначає наступні типи зв'язку: асоціація, спадкування, залежність, агрегація, композиція. На діаграмі, всі три класи з'єднано зв'язком асоціації.

2.4 Діаграма послідовностей

На діаграмі послідовностей (див. рис. 2.3) представлено поведінку системи за функцією «Аналіз двигуна на наявність несправностей».

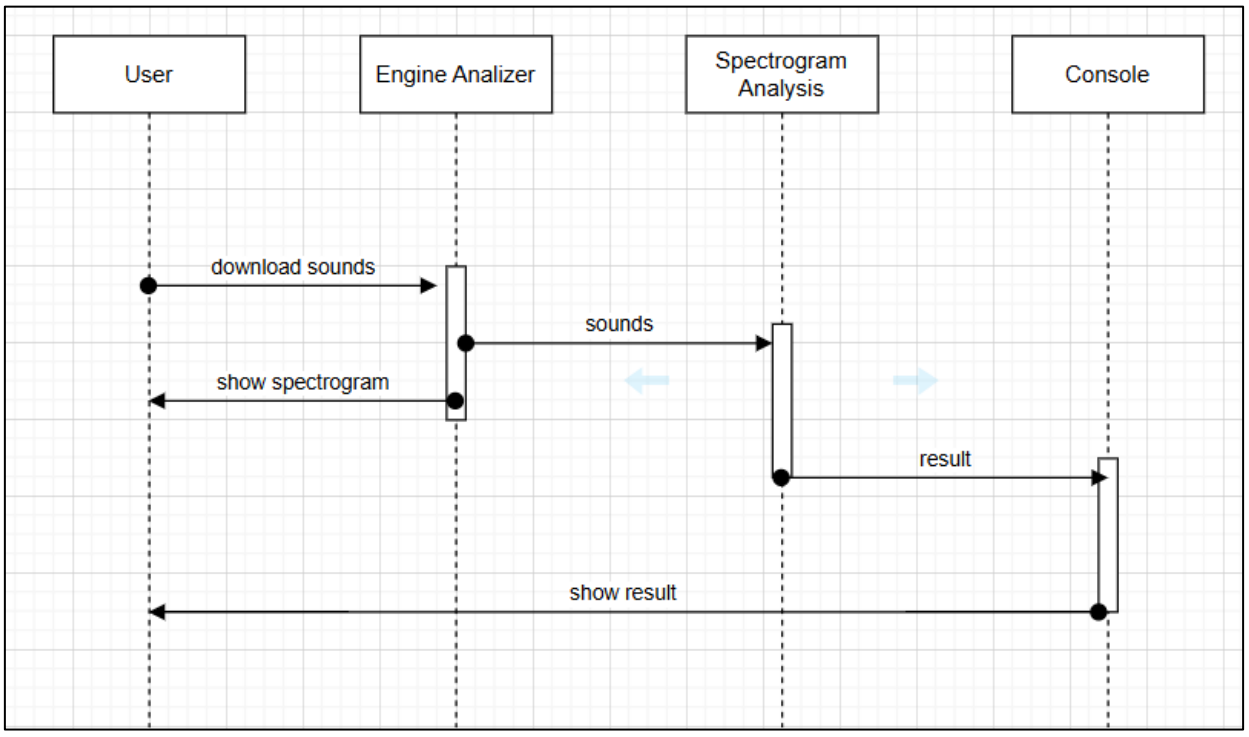


Рисунок 2.3 – Діаграма послідовності

Як можна побачити: користувач завантажує звук роботи двигуна в клас Engine_Analizer. Клас Engine_Analizer передає дані в клас Spectrogram_Analysis який відповідає за аналіз звуку, після чого передає відповідь в клас Console.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Аналіз програми

Розглянемо деякі класи детальніше. Клас Engine_Analizer – основний клас, в моделі MVC відповідає контролеру (Controller). Клас Engine_Analizer працює с користувачем та приймає від нього запити, а потім передає класу Spectrogram_Analysis, який виконує роль моделі. Розглянемо деякі основні методи класу Engine_Analizer.

Метод download_spectrogrum відповідає за завантаження запису звука та перетворення звукового запису у спектрограму (див. рис. 3.1).

```
def download_spectrogrum(self, label):
    options = QFileDialog.Options()
    file_path, _ = QFileDialog.getOpenFileName(self, "Виберіть файл", "", "Звукові файли (*.mp3)",
options=options)
    if file_path:
        y, sr = librosa.load(file_path, sr=None)
        D = librosa.stft(y)
        S_db = librosa.amplitude_to_db(np.abs(D), ref=np.max)
        fig, ax = plt.subplots(1)
        librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='log', cmap='magma')
        plt.colorbar(format='%+2.0f дБ')
        plt.title(os.path.basename(file_path))
        plt.ylabel('Частота')

        fig.savefig("spectrum.png")
        pixmap = QtGui.QPixmap("spectrum.png")
        label.setPixmap(pixmap)
        label.setScaledContents(True)
        return y, sr
    else:
        return None, None
```

Рисунок 3.1 – Метод download_spectrogrum

При виконані методу програма спочатку взаємодіє з користувачем і відкриває діалогове вікно вибору файлу. Після вибору користувачем файлу програма досліджує чи існує такий файл на комп’ютері. Якщо такий файл

відсутній, робота методу завершується. Якщо існує, програма завантажує цей файл та розкладає його на аудіосигнал та частоту дискретизації. Після цього, використовуємо STFT(короткочасне перетворення Фур'є) для створення спектрограми. Після створення спектрограми система візуалізує її. Далі метод закінчується повертаючи аудіосигнал та частота дискретизації для використання в інших методах класу.

Для аналізу звукового запису і виявлення проблем існує метод analize_spectrogram (див. рис. 3.2). При виконані цього методу спочатку перевіряється чи було завантажено звуковий запис та чи було створено спектрограму. Якщо звуковий запис не був завантажений виводиться повідомлення про помилку та робота методу завершується. Якщо звуковий запис був завантажений, то виконується метод Breakdown_Analysis класу Spectrogram_Analysis, який повертає список несправностей, знайдених методом.

```
def analize_spectrogram(self):
    if(self.label.pixmap() is not None):
        Breakdowns = []
        SpectrAnalysis = Spectrogram_Analysis.Spectrogram_Analysis()
        Breakdowns = SpectrAnalysis.Breakdown_Analysis(self.y, self.sr)
        if len(Breakdowns) != 0:
            self.ConsoleWindow.GetAnswer("Знайдено проблема! " + ", ".join(Breakdowns) + "!")
        else:
            self.ConsoleWindow.GetAnswer(f"Проблеми відсутні!")
    else:
        self.ConsoleWindow.GetAnswer("Помилка! Завантажте спектрограму!")
```

Рисунок 3.2 – Метод analize_spectrogram

Окрім спектрограм програма може будувати спектри (залежність частоти від амплітуди) (див. рис. 3.3). Як і при побудові спектрограми спочатку з'являться діалогове вікно вибору файлу. Після вибору користувачем файлу програма досліджує чи існує такий файл на комп'ютері. Якщо такий файл відсутній, робота методу завершується. Якщо існує, програма завантажує файл та розкладає його на аудіосигнал та частоту дискретизації. Потім відбувається обчислення ряду Фур'є. Оскільки перетворення Фур'є для

реальних сигналів симетричне, можна аналізувати лише позитивну частину спектра. Після створення спектрограми система візуалізує її. Далі метод завершує роботу і повертає аудіосигнал та частоту дискретизації для використання в інших методах класу.

```
def download_spectr(self, label):
    options = QFileDialog.Options()
    file_path, _ = QFileDialog.getOpenFileName(self, "Виберіть файл", "", "Усі файли (*);;Звукові файли
(*.mp3)", options=options)
    if file_path:
        #обчислення спектра
        y, sr = librosa.load(file_path, sr=None)
        n = len(y)
        fft = np.fft.fft(y)
        magnitude = np.abs(fft)  # Амплитуди
        frequency = np.fft.fftfreq(n, 1/sr)  # Частоти

        # Лишаємо тільки позитивні частоти
        positive_freqs = frequency[:n//2]
        positive_magnitude = magnitude[:n//2]

        # Візуалізація спектра
        fig, ax = plt.subplots(1)
        plt.plot(positive_freqs, positive_magnitude, color='blue')
        plt.yscale('log')
        plt.title(os.path.basename(file_path))
        plt.xlabel("Частота (Гц)")
        plt.ylabel("Інтенсивність (Амплітуда)")
        fig.savefig("spectrum.png")
        pixmap = QtGui.QPixmap("spectrum.png")
        label.setPixmap(pixmap)
        label.setScaledContents(True) #маштабування зображення до розміру label
        return y, sr
    else:
        return None, None
```

Рисунок 3.3 – Метод download_spectr

Для аналізу даних існує клас Spectrogram_Analysis. В моделі MVC відповідає моделі (Model). В класі Spectrogram_Analysis існує декілька методів головним у класі є метод Breakdown_Analysis (див. рис. 3.4). В методі Breakdown_Analysis система порівнює завантажений користувачем запис з записами у власній бібліотеці. При наявності збігів вона повідомляє користувачу. Для того щоб зрозуміти принцип порівняння, розглянемо метод cosinus_compare_spectrgrum.

```
def Breakdown_Analysis(self, y1, sr1):
    folder_path = "Sound_Recording\\"
    Breakdown = []
    ConsoleWindow = Console.Console()
    for file_name in os.listdir(folder_path):
        if os.path.isfile(os.path.join(folder_path, file_name)):
            y2, sr2 = librosa.load(folder_path + file_name, sr=None)
            cos_sim = Spectrogram_Analysis.cosinus_compare_spectrgrum(self, y1, y2, sr1, sr2)
            if(cos_sim > 0.9):
                Breakdown.append(os.path.splitext(os.path.basename(file_name))[0])
    return Breakdown
```

Рисунок 3.4 – Метод Breakdown_Analysis

Як можна побачити в методі cosinus_compare_spectrgrum використовується косинусна схожість (див. рис. 3.5). Косинусна схожість працює за таким принципом: спочатку обчислюється вектори значень, а потім порівнюється кут між векторами. Косинусна схожість була обрана через те, що в цьому методі дивиться не на самі довжини векторів, а на кут між ними. Якщо кут маленький (вектори дивляться приблизно в один бік) – значить вони схожі. У відповідь функція виводить значення від -1 до 1. Якщо значення знаходиться ближче до 1 то звукові записи схожі. Якщо до -1 або 0 то звукові записи не схожі. В методі cosinus_compare_spectrgrum використовуються методи cut_sound та normalize_spectrogram.

```
def cosinus_compare_spectrgrum(self, y1, y2, sr1, sr2):

    y1, y2 = Spectrogram_Analysis.cut_sound(y1, y2)
    # Перетворення в спектрограми
    S1 = librosa.stft(y1)
    S2 = librosa.stft(y2)

    # Амплітудний спектр
    S1_db = librosa.amplitude_to_db(np.abs(S1), ref=np.max)
    S2_db = librosa.amplitude_to_db(np.abs(S2), ref=np.max)

    S1_db = Spectrogram_Analysis.normalize_spectrogram(S1_db)
    S2_db = Spectrogram_Analysis.normalize_spectrogram(S2_db)
    # Перетвореня спектрограм у вектори

    vec1 = S1_db.flatten()
    vec2 = S2_db.flatten()
    cos_sim = 1 - cosine(vec1, vec2)
    return cos_sim
```

Рисунок 3.5 – Метод cosinus_compare_spectrgrum

В методі cut_sound два звукові записи порівнюються та довший з них обрізається (див. рис. 3.6).

```
def cut_sound(y1, y2):
    if len(y1) > len(y2):
        y1 = y1[:len(y2)]
    elif len(y1) < len(y2):
        y2 = y2[:len(y1)]
    return y1, y2
```

Рисунок 3.6 – Метод cut_sound

В методі normalize_spectrogram дані нормалізуються (див. рис. 3.7). Процес нормалізації включає вирівнювання амплітудних рівнів спектрограм, щоб вони знаходилися в одному діапазоні значень. Для нормалізації: спочатку, визначається мінімальне та максимальне значення в спектрограмі. Потім кожне значення у спектрограмі зменшується на spectrogram_min, щоб мінімальне значення дорівнювало 0. Потім результат ділиться на (spectrogram_max - spectrogram_min), щоб максимальне значення дорівнювало 1. Отримане значення помножується на (max_val - min_val) , для зміни масштабу значень, та додається min_val для зсуву діапазону, щоб мінімальне значення дорівнювало min_val, а максимальне – max_val.

```
def normalize_spectrogram(spectrogram, min_val=0, max_val=1):
    spectrogram_min = spectrogram.min()
    spectrogram_max = spectrogram.max()
    normalized_spectrogram = (spectrogram - spectrogram_min) / (spectrogram_max - spectrogram_min)
    normalized_spectrogram = normalized_spectrogram * (max_val - min_val) + min_val
    return normalized_spectrogram
```

Рисунок 3.7 – Метод normalize_spectrogram

Клас Console відповідає за представлення даних, і відповідає представленню (View) в моделі MVC. В класі представлений інтерфейс і метод GetAnswer для виводу відповіді програми (див. рис. 3.8).

```
from PyQt5.QtWidgets import QWidget, QLabel


class Console(QWidget):
    def __init__(self):
        super().__init__()
        self.resize(800, 200)
        self.label = QLabel(self)
        self.label.setGeometry(10, 0, self.width() - 10, self.height())
        self.setWindowTitle("Console")

    def GetAnswer(self, Answer):
        text = self.label.text()
        self.label.setText(text + "\n" + Answer)
```

Рисунок 3.8 – Метод normalize_spectrogram

3.2 Робота програми

Розглянемо роботу програми більш детально. Після запуску програми відкриваються два вікна основне вікно та вікно консолі (див. рис. 3.9).

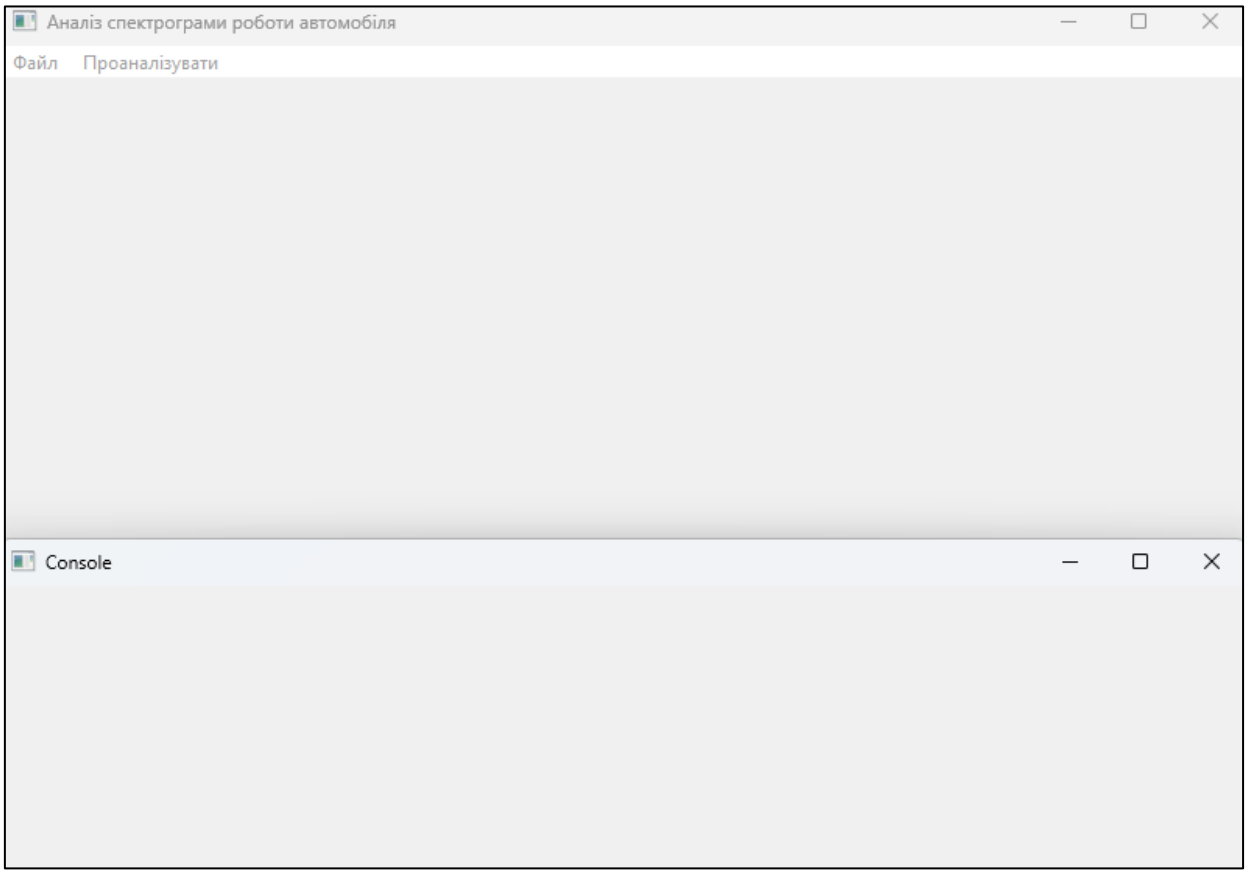


Рисунок 3.9 – Програма після запуску

В графічному елементі menubar представлений основний функціонал додатку. В меню «Файл» присутні такі функції: «Завантажити файл як спектрограму», «Завантажити файл як спектр» (див. рис. 3.10).



Рисунок 3.10 – Функціонал додатку меню «Файл»

В меню «Проаналізувати» присутні функції: «Аналіз запису», «Порівняти спектрограми», «Порівняти спектри» (див. рис. 3.11).



Рисунок 3.11 – Функціонал додатку меню «Проаналізувати»

Далі продемонстровано робота функції програми «Аналіз запису». Для початку, завантажмо файл як спектрограму, для цього в меню файл виберемо пункт «Завантажити файл як спектрограму» (див. рис. 3.12).

Після цього з’явиться вікно вибору файлу. Після вибору файлу користувачем, програма завантажить у додаток звуковий запис як спектрограму (див. рис. 3.13).

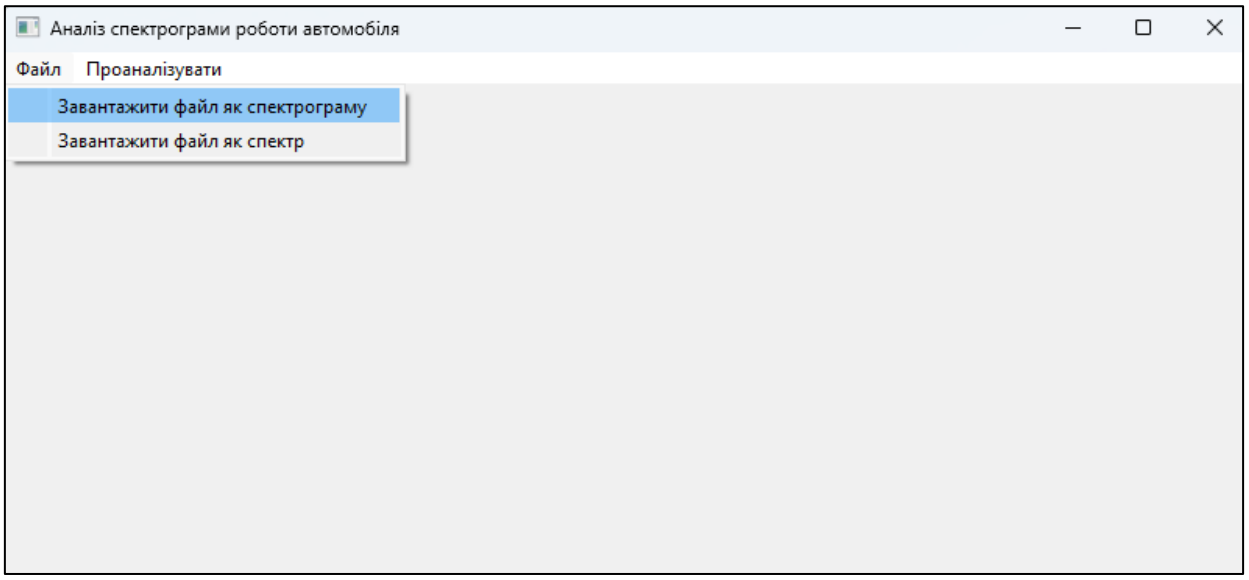


Рисунок 3.12 – Пункт «Завантажити файл як спектрограму»

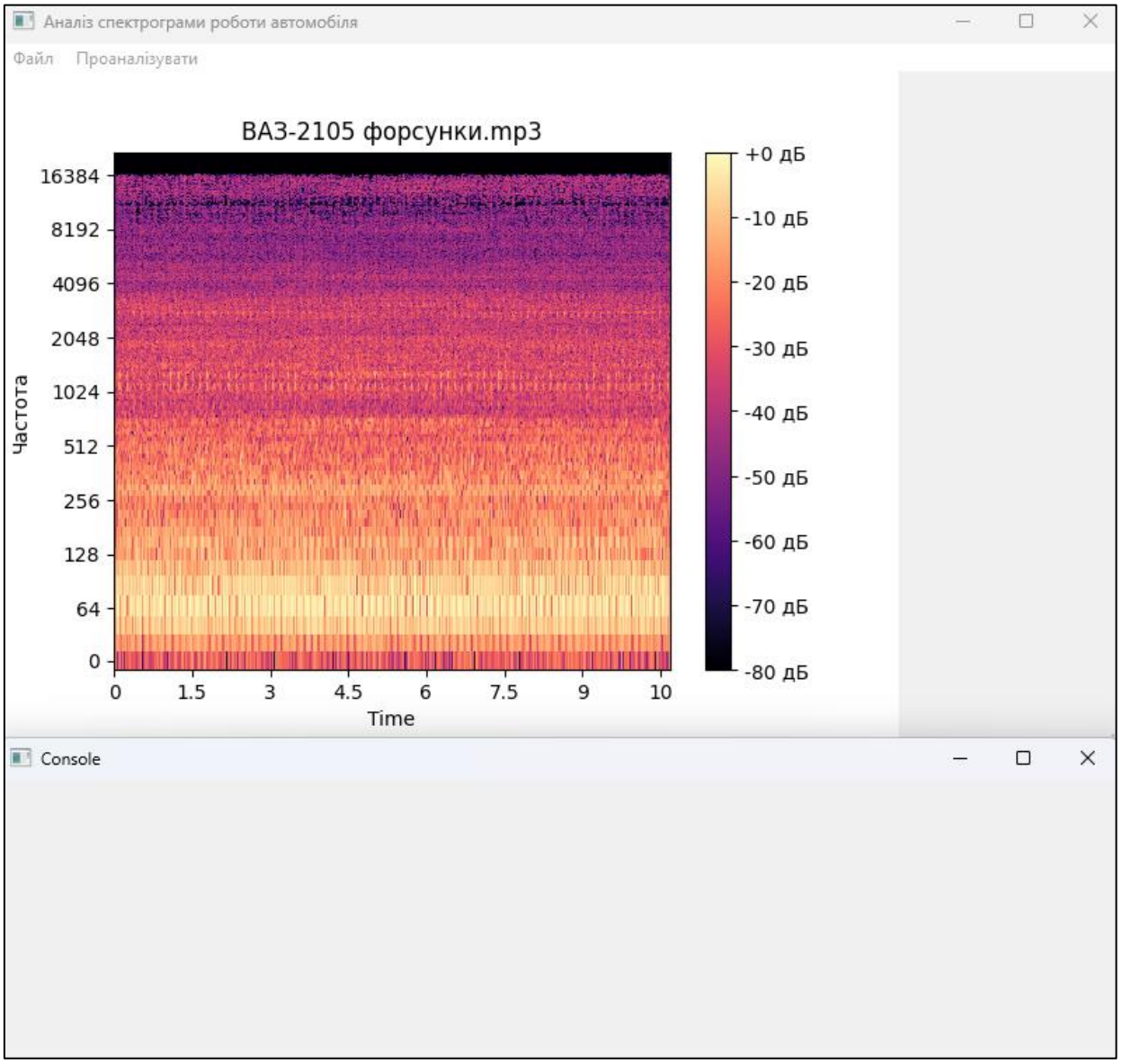


Рисунок 3.13 – Завантаження файлу як спектрограми

Після завантаження, виберемо пункт меню «Аналіз запису», як зображено на рисунку 3.14, і програма проаналізує звуковий запис.

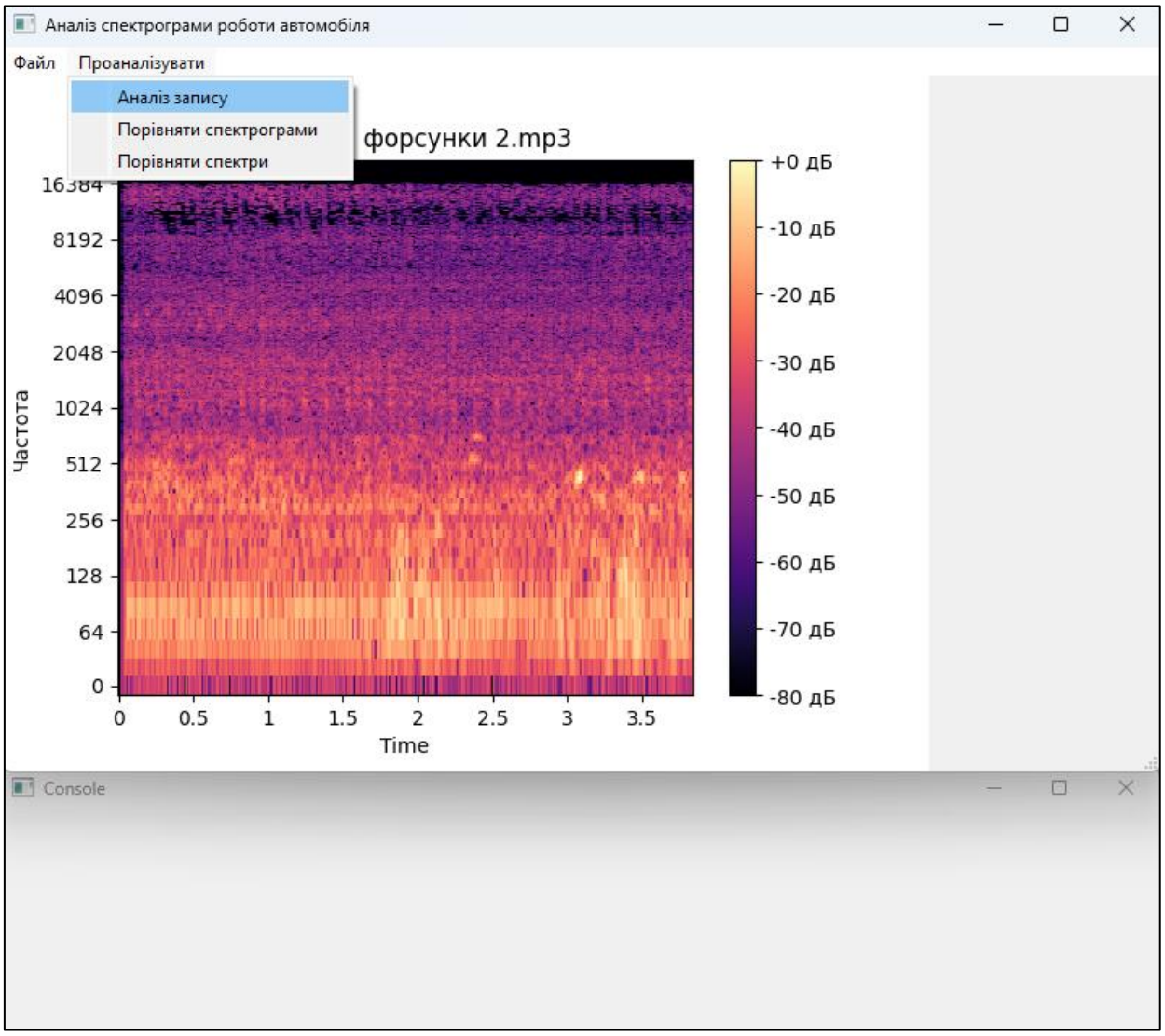


Рисунок 3.14 – Пункт «Аналіз запису» в меню «Проаналізувати»

Під час аналізу додаток порівнює завантажений запис з записами у власній бібліотеці. Якщо програма знаходить співпадіння, то виводить її.

Після аналізу додаток виведе в консоль відповідь про те, зламаний двигун чи ні (див. рис. 3.15 та 3.16).

Програма здатна відслідковувати ряд несправностей, наприклад: розбовтані форсунки, проблеми з підшипником на ремені (див. рис. 3.17), проблеми з ременем (див. рис. 3.18), програма також може відслідковувати зламані диски або колодки (див. рис. 3.19).

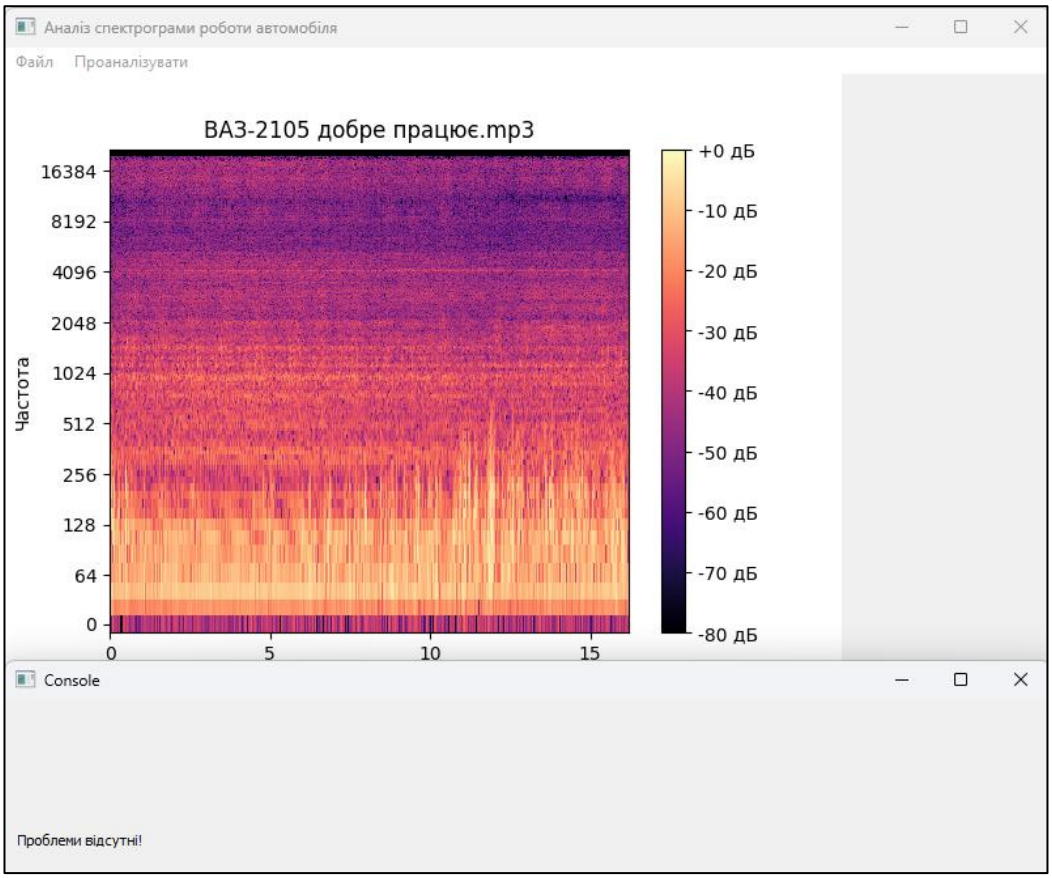


Рисунок 3.15 – Проблеми в двигуні відсутні

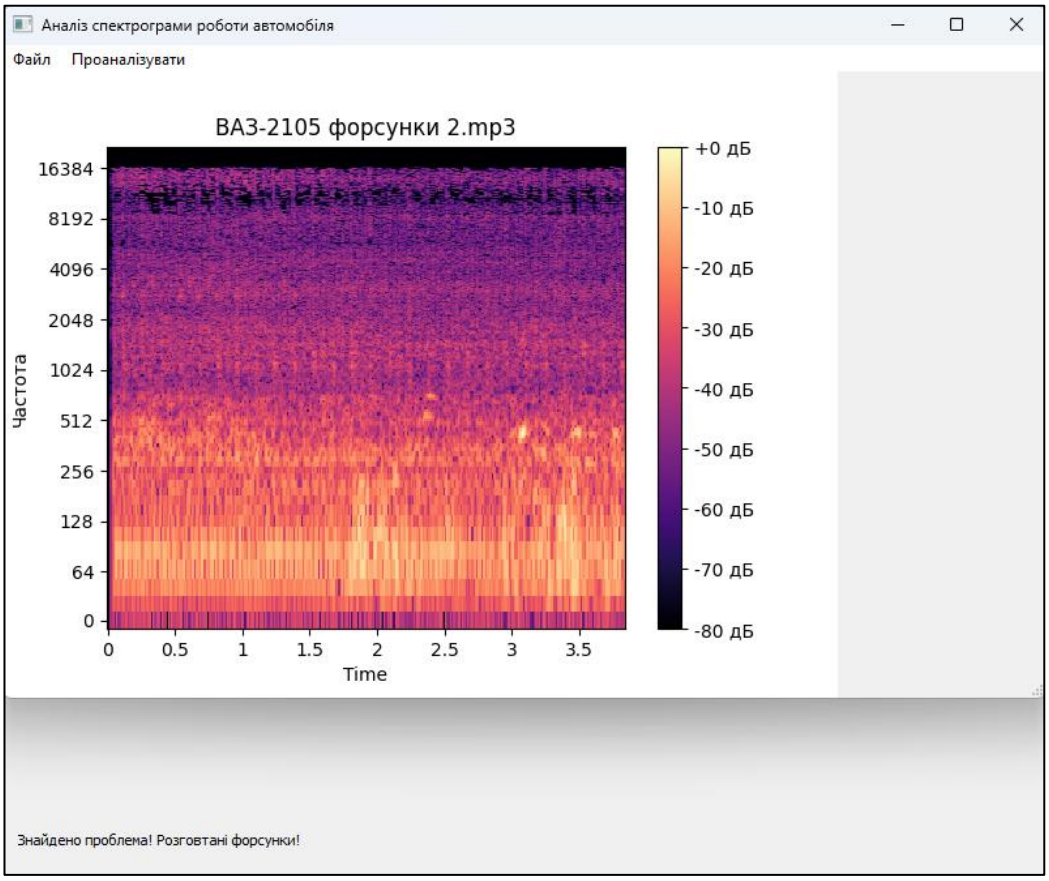


Рисунок 3.16 – Проблеми з форсунками в двигуні

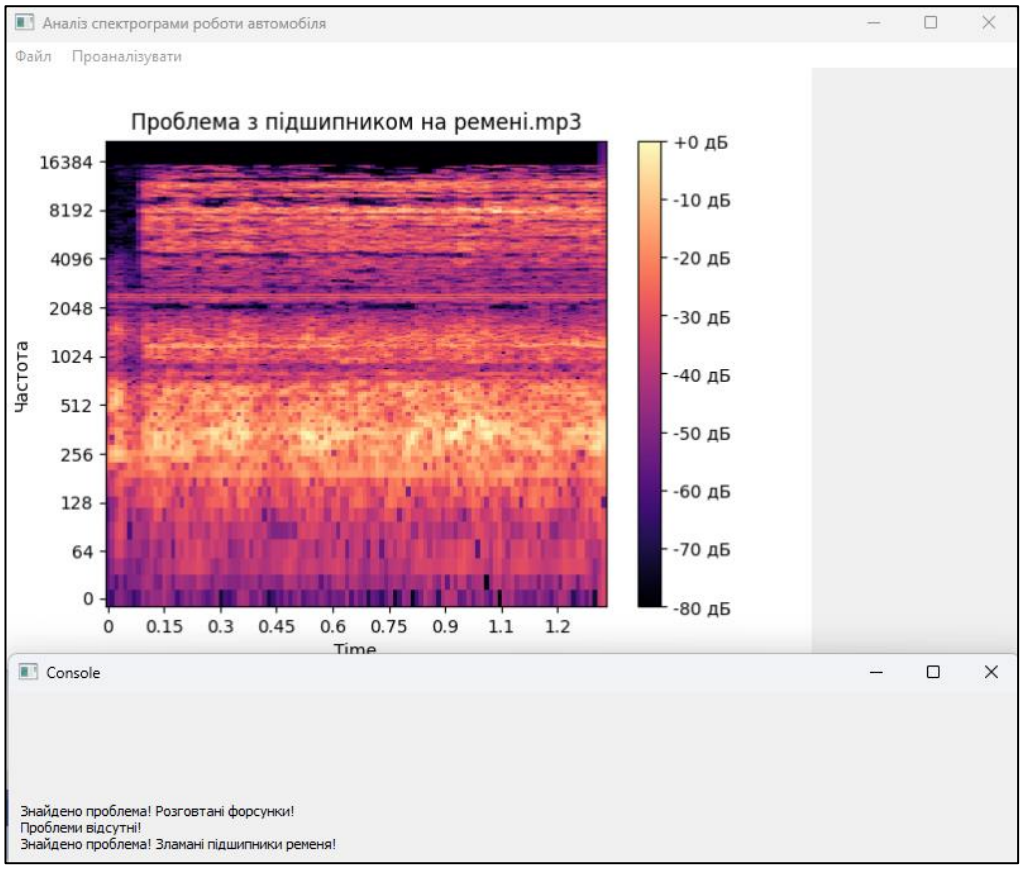


Рисунок 3.17 – Проблеми з підшипником на ремені

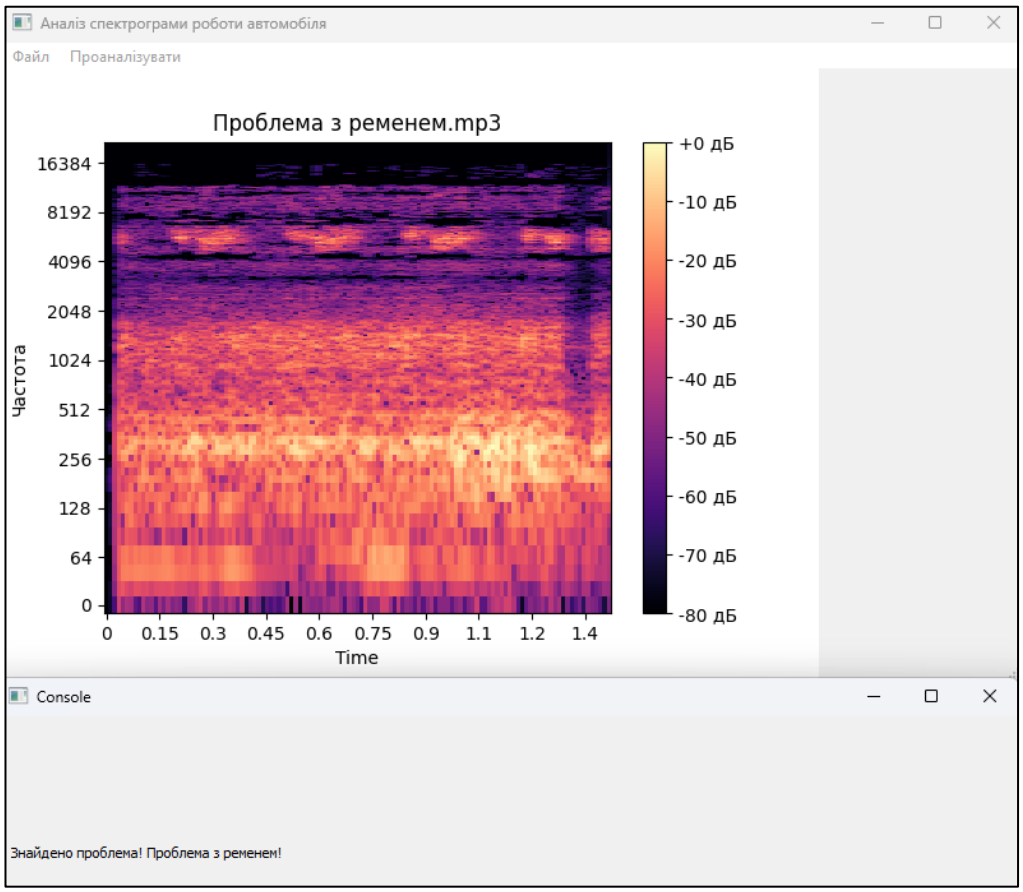


Рисунок 3.18 – Проблеми з ременем

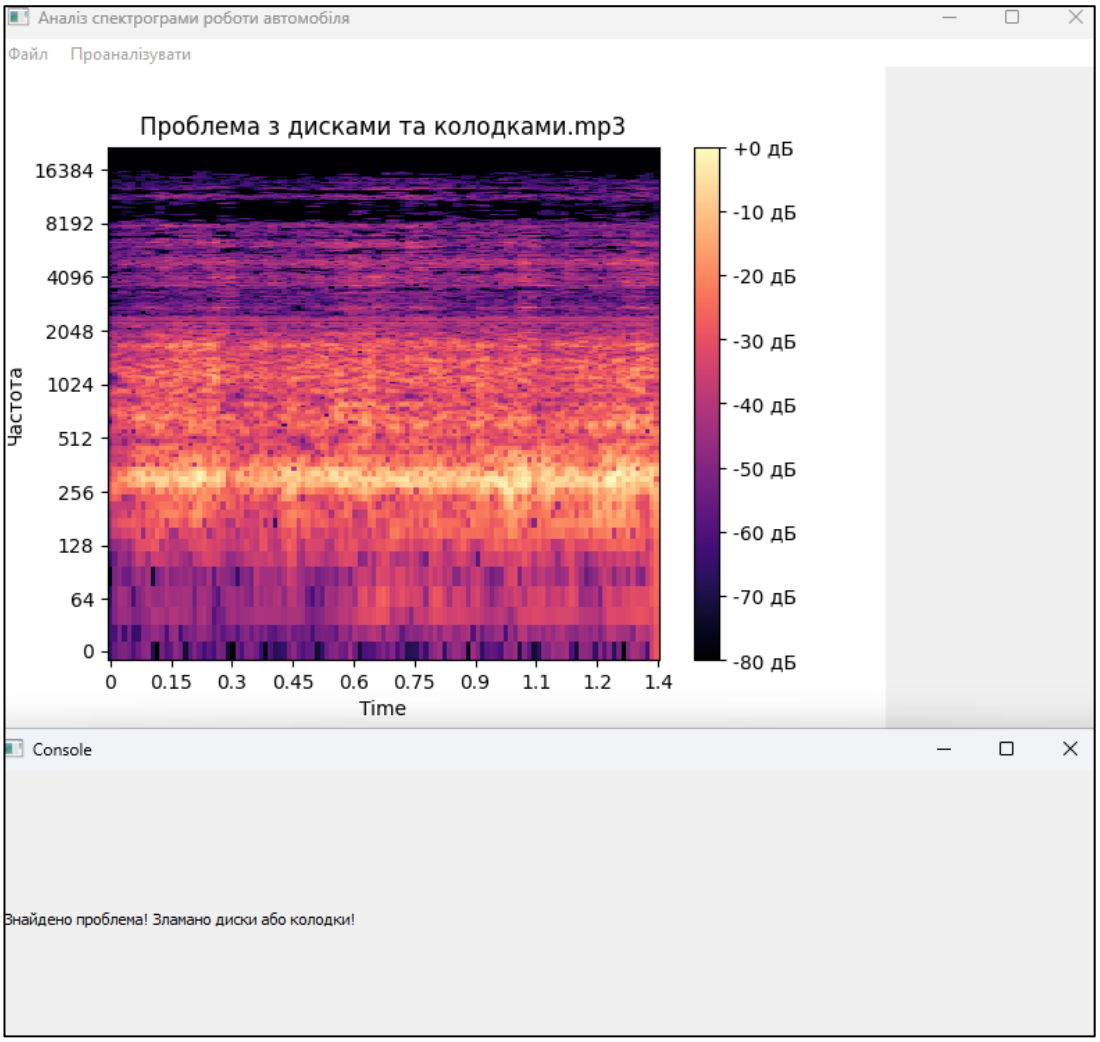


Рисунок 3.19 – Зламані диски або колодки

Окрім роботи зі спектрограмами, програма може працювати з графіком залежності частоти від амплітуди, іншими словами, спектрами, для цього потрібно вибрати меню «Файл» та вибрати пункт меню «Звантажити файл як спектр» (див. рис. 3.20).



Рисунок 3.20 – Пункт «Завантажити файл як спектр» в меню «Файл»

Після цього з’явиться вікно вибору файлу. Після вибору файлу користувачем, програма завантажить у додаток звуковий запис як спектрограму (див. рис. 3.21).

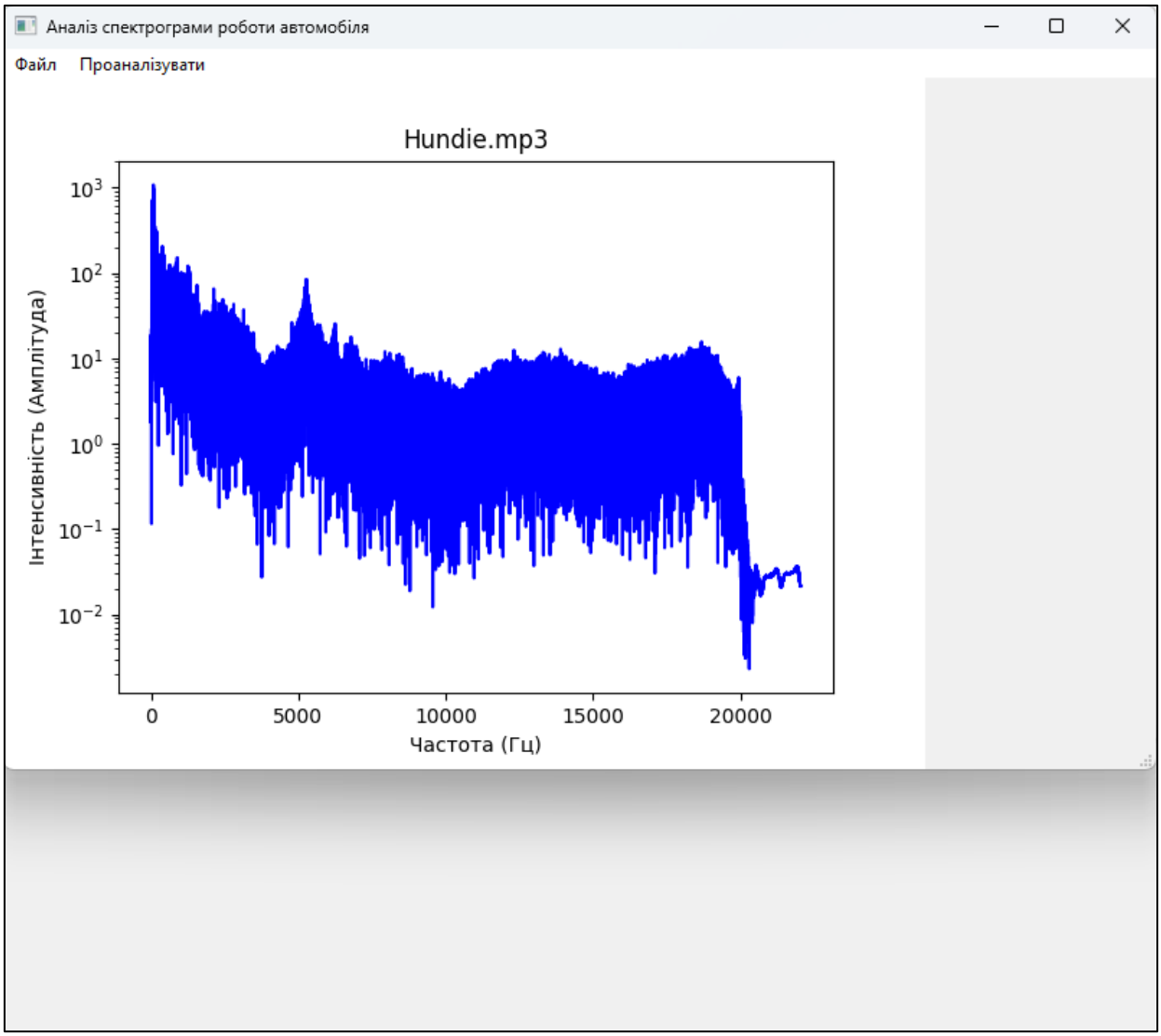


Рисунок 3.21 – Залежності частоти від амплітуди

Через те що програма працює зі звуковим записом а не з графіком, вона також може аналізувати спектри. Для цього потрібно вибрати пункт «Аналіз запису». Після чого додаток виведе результат (див. рис. 3.22).

Програма може порівнювати спектри та спектрограми та знаходити відсоток схожості, для цього потрібно вибрати пункт «Порівняти спектри» (див. рис. 3.23) або пункт «Порівняти спектрограми» (див. рис. 3.26) в меню «Проаналізувати».

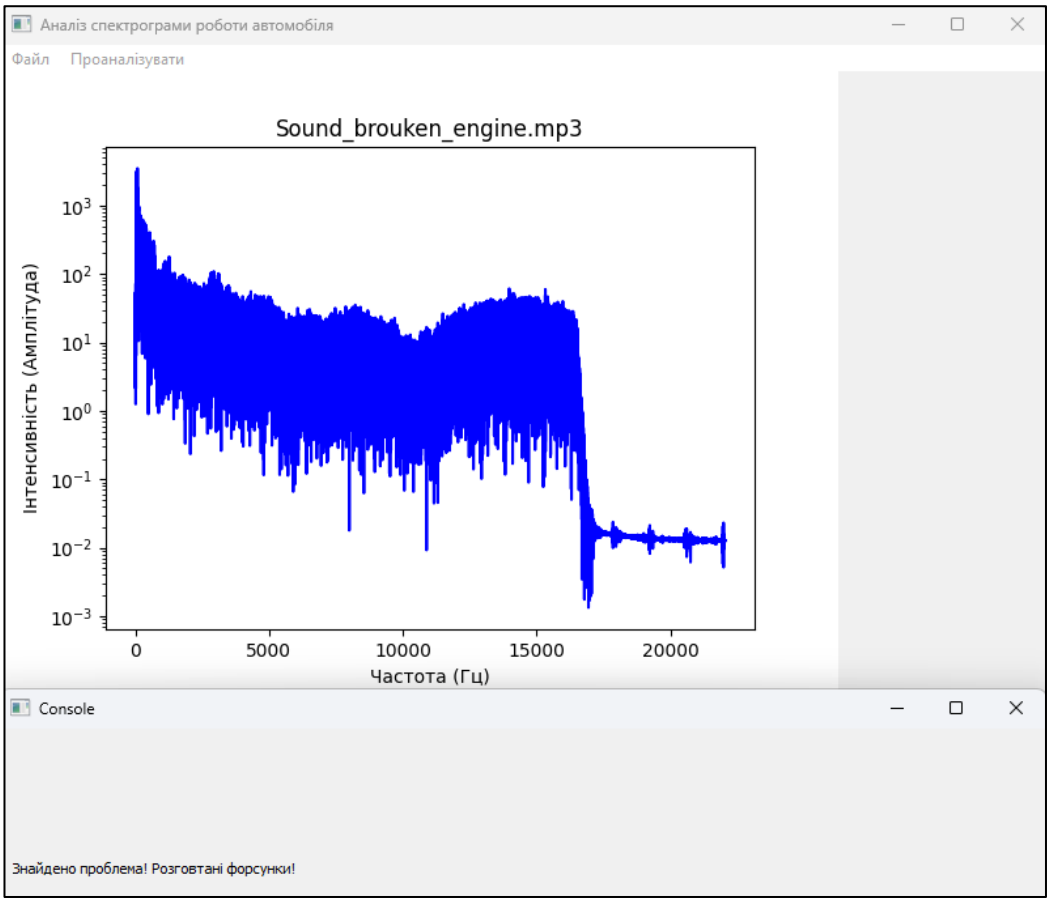


Рисунок 3.22 – Аналіз спектру

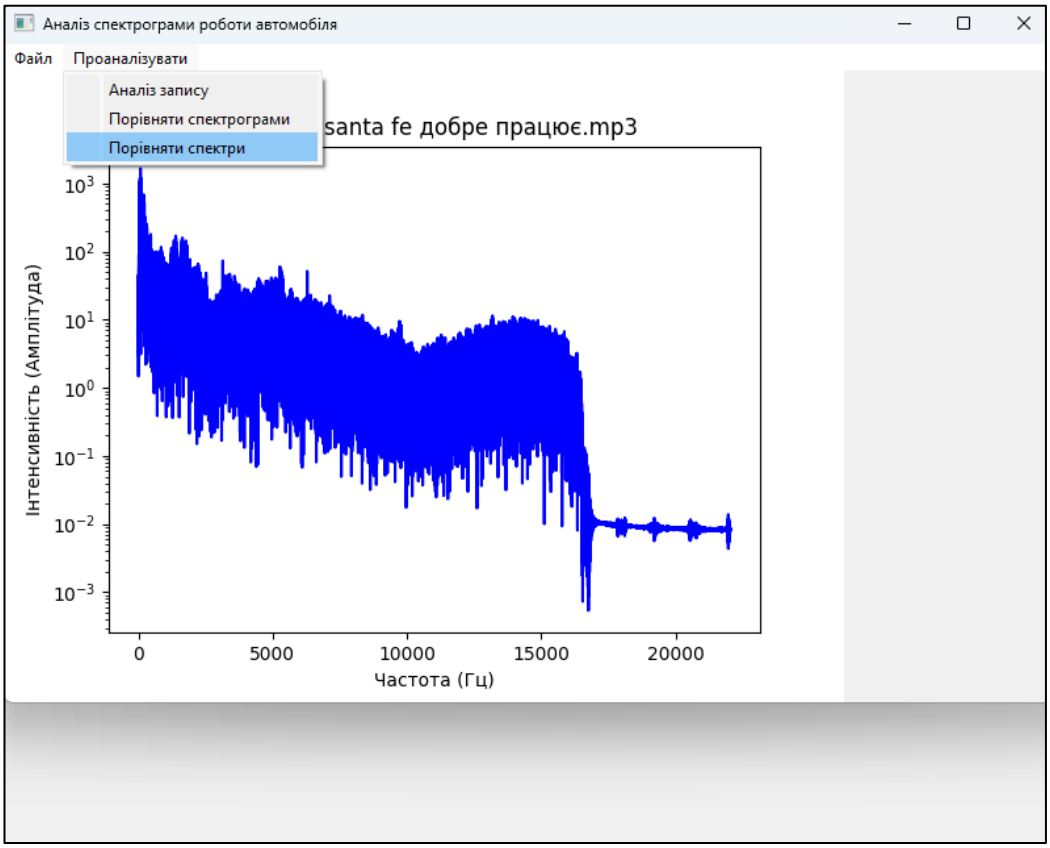


Рисунок 3.23 – Пункт меню «Порівняти спектри»

При натисканні кнопки «Порівняти спектри», відкриється вікно вибору файлу з яким додаток буде порівнювати поточний спектр. Після вибору файлу додаток завантажить його та проведе аналіз.

Після порівняння, програма виведе відсоток схожості двох спектрів (див. рис. 3.24 та 3.25).

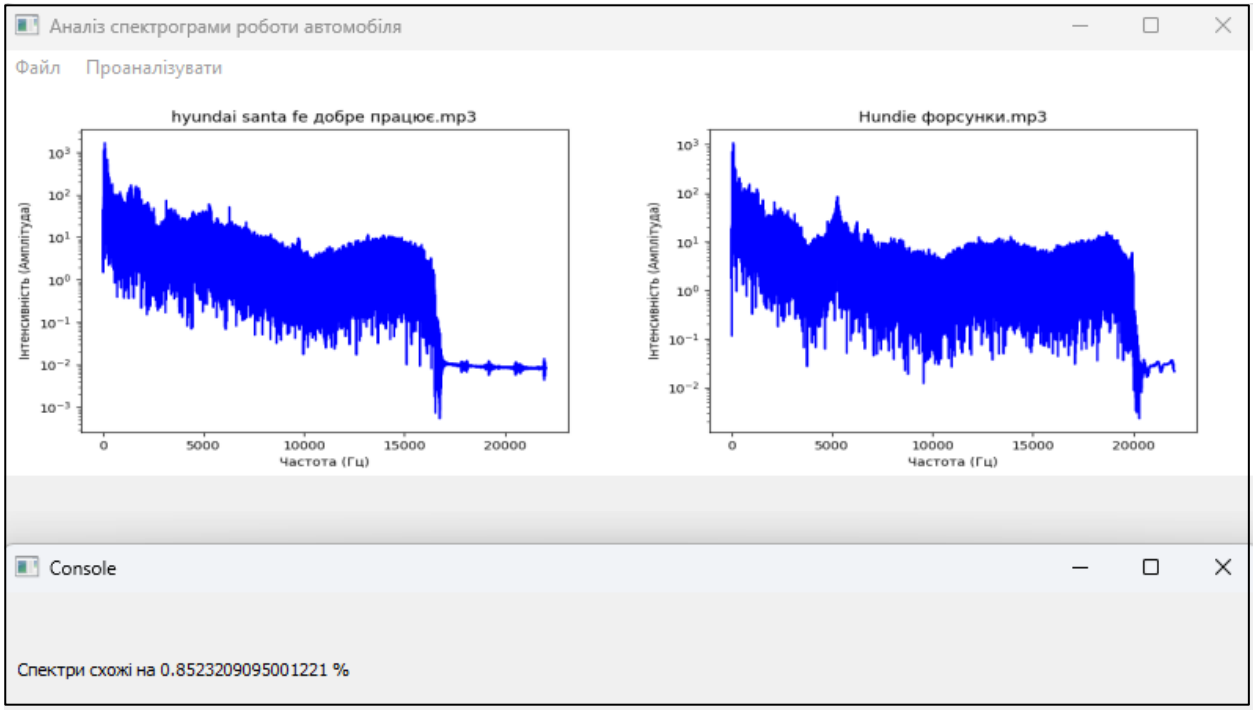


Рисунок 3.24 – Порівняння двох спектрів

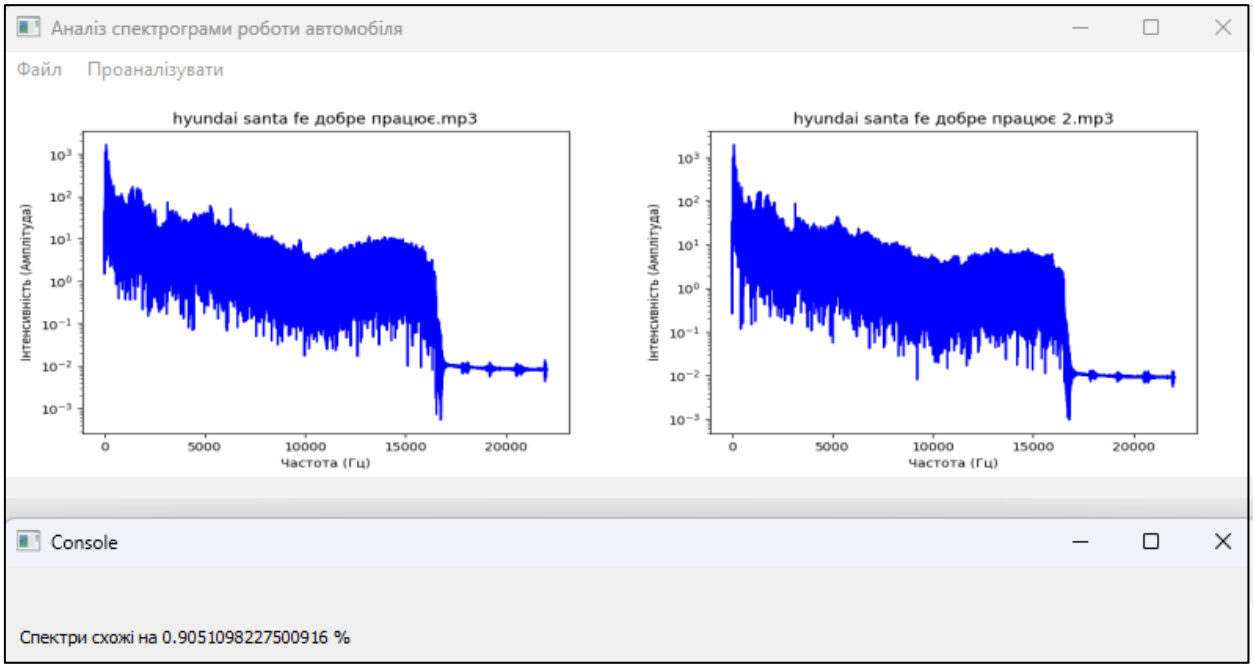


Рисунок 3.25 – Порівняння двох спектрів

Аналогічним чином програма працює зі спектрограмами. Після натискання кнопки «Порівняти спектрограми», відкриється діалогове вікно, що дозволяє вибрати файл, який містить другий спектр для порівняння.

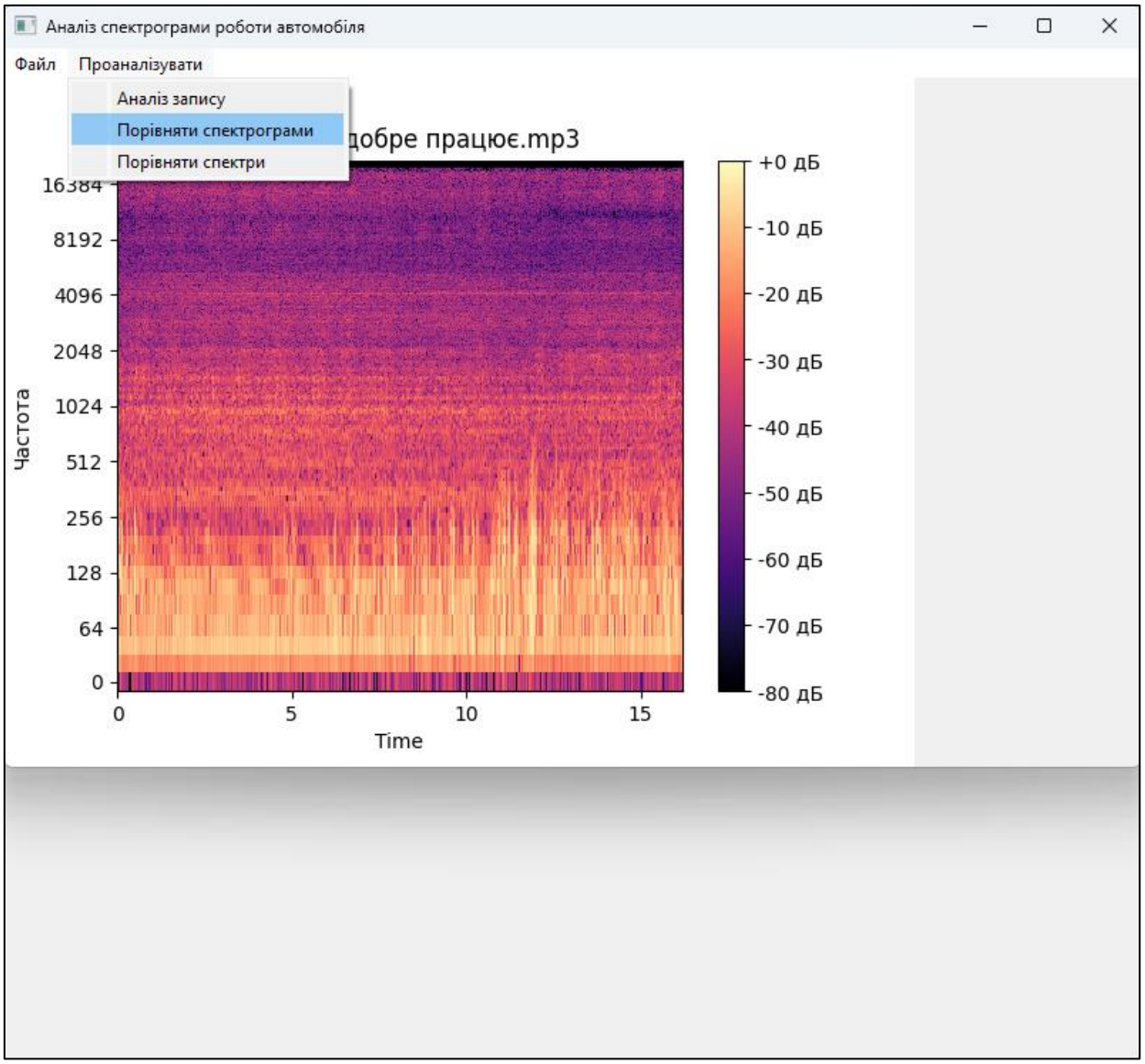


Рисунок 3.26 – Пункт меню «Порівняти спектрограми»

Після того, як буде вибрано файл і підтверджено свій вибір, програма завантажить його дані.

Потім програма автоматично проаналізує обидва спектри та визначить ступінь їхньої схожості, відобразивши результат у вигляді відсотка подібності, як показано на рисунках 3.27 та 3.28.

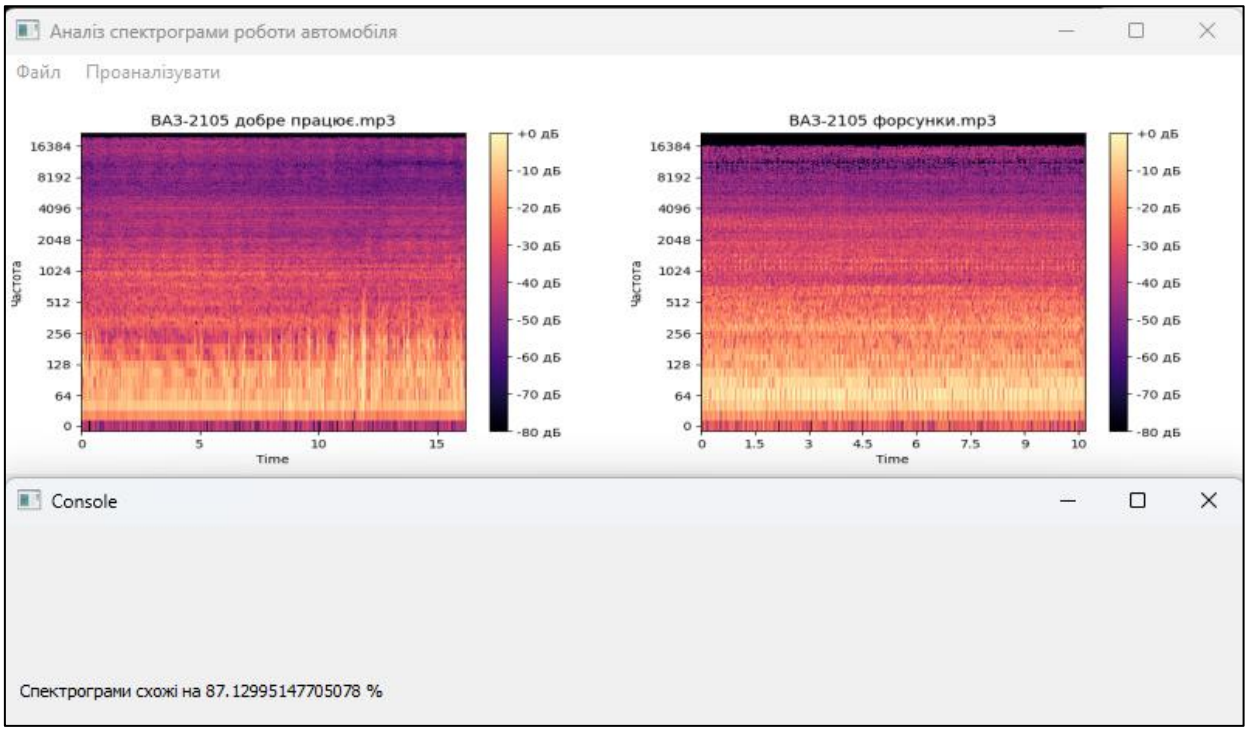


Рисунок 3.27 – Порівняння двох спектрограм

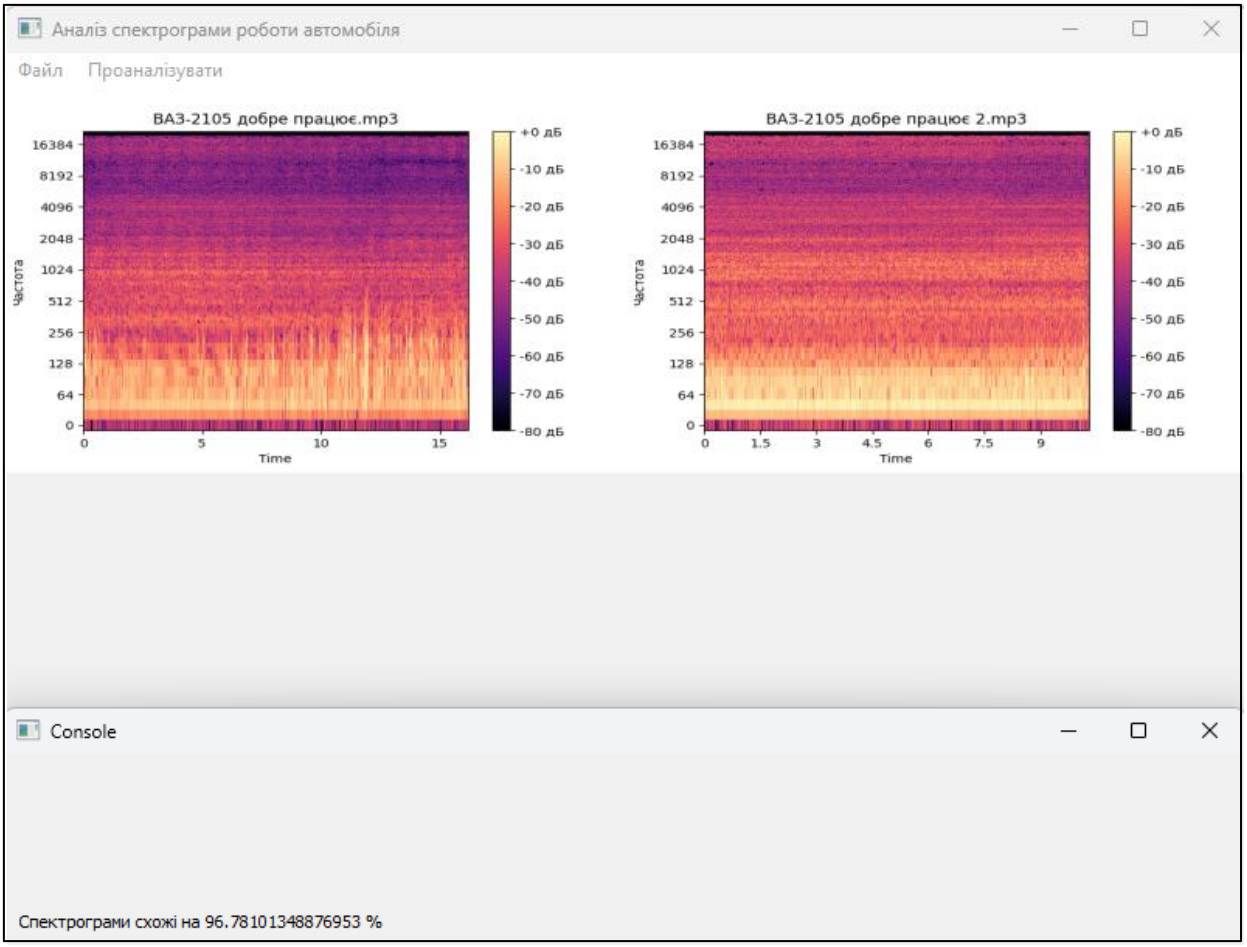


Рисунок 3.28 – Порівняння двох спектрограм

ВИСНОВКИ

У цій кваліфікаційній роботі було розглянуто проблему створення додатку акустичної діагностики двигуна автомобіля на мові python. Було розглянуто інші програми зі схожою функціональністю, розглянуто їх переваги, зручність, недоліки. Було побудовано UML діаграми та обрано патерни програмування.

В результаті дослідження було розроблено додаток, який може будувати спектрограми та спектри звукового запису, порівнювати звукові записи, аналізувати записи роботи двигуна автомобіля та робити висновок про те, чи є несправності у двигуні.

Основною перевагою додатку є можливість швидкої попередньої оцінки стану двигуна без необхідності демонтажу вузлів та агрегатів. Було створено інструмент, який може бути використаний як водіями, так і технічними фахівцями для оперативної оцінки стану двигуна. Крім того, робота може стати основою для подальших досліджень та вдосконалення системи діагностики, наприклад, за рахунок впровадження методів машинного навчання та нейромереж для більш точної класифікації несправностей.

Перспективними напрямками розвитку проєкту є розширення бібліотеки звукових патернів несправностей та реалізація адаптивної системи навчання на основі накопичених даних від користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Your equipment is trying to tell you something. URL: https://www.uesystems.com/en-gb/ultrasound-technology/ (дата звернення: 09.02.2025).

2. Романов М. В. 10 звуків, які розкажуть про несправності автомобіля. URL: https://www.avtomir.ua/articles/10-zvukiv-yaki-rozkazhut-pro-nespravnosti-avtomobilya/ (дата звернення: 09.02.2025).

3. Chandler D. L. Let your car tell you what it needs. URL: https://news.mit.edu/2017/software-let-your-car-tell-you-what-it-needs-1026 (дата звернення: 09.02.2025).

4. ŠKODA AUTO uses artificial intelligence for even more accurate car diagnostics. URL: https://www.skoda-storyboard.com/en/press-releases/skoda-auto-uses-artificial-intelligence-for-even-more-accurate-car-diagnostics-2/ (дата звернення: 09.02.2025).

5. AI-driven Audio Analytics. URL: https://www.keenfinity-group.com/xc/en/solutions/video-systems/intelligent-audio-analytics/ (дата звернення: 14.04.2025).

6. Bosch sends sensor system to ISS. URL: https://www.bosch.com/stories/acoustic-sensors/ (дата звернення 14.04.2025).

7. Udy J. ClingClanger App Helps Diagnose Car Noises. URL: https://www.motortrend.com/news/clingclanger-app-helps-diagnose-car-noises (дата звернення: 14.04.2025).

8. Librosa.stft documentation. URL: https://librosa.org/doc/0.10.2/generated/librosa.stft.html#librosa.stft (дата звернення: 11.02.2025).

9. Librosa.load documentation. URL: https://librosa.org/doc/0.10.2/generated/librosa.load.html#librosa.load (дата звернення: 11.02.2025).

10. Librosa.amplitude_to_db documentation. URL: https://librosa.org/doc/0.10.2/generated/librosa.amplitude_to_db.html#librosa-amplitude-to-db (дата звернення: 11.02.2025).

11. Scipy.spatial.distance.cosine documentation. URL: https://docs.scipy.org/doc/scipy-1.12.0/reference/generated/scipy.spatial.distance.cosine.html (дата звернення: 11.02.2025).

12. What is PyQt? URL: https://www.riverbankcomputing.com/software/pyqt/intro (дата звернення: 03.02.2025).

13. QApplication PyQt documentation. URL: https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QApplication.html (дата звернення: 03.02.2025).

14. QMainWindow PyQt documentation. URL: https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QMainWindow.html (дата звернення: 03.02.2025).

15. QFileDialog PyQt documentation. URL: https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QFileDialog.html (дата звернення: 03.02.2025).

16. QManuBar PyQt documentation. URL: https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QMenuBar.html (дата звернення: 03.02.2025).

17. QManu PyQt documentation. URL: https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QMenu.html (дата звернення: 03.02.2025).

18. QAction PyQt documentation. URL: https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QAction.html (дата звернення: 03.02.2025).

19. QLabel PyQt documentation. URL: https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QLabel.html (дата звернення: 03.02.2025).

20. Matplotlib.pyplot documentation. URL: https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html (дата звернення: 10.02.2025).

21. Numpy.fft.fft documentation. URL: https://numpy.org/doc/stable/reference/generated/numpy.fft.fft.html (дата звернення: 10.02.2025).

22. Numpy.absolute documentation. URL: https://numpy.org/doc/stable/reference/generated/numpy.absolute.html#numpy.absolute (дата звернення: 10.02.2025).

23. Fft.fftfreq documentation. URL: https://numpy.org/doc/stable/reference/generated/numpy.fft.fftfreq.html#numpy.fft.fftfreq (дата звернення: 10.02.2025).

24. MVC: Model, View, Controller. URL: https://www.codecademy.com/article/mvc (дата звернення: 12.02.2025).

ДОДАТОК А

Лістинг додатку

А.1 Клас Engine_Analizer

```
from PyQt5 import QtGui
from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QMenuBar, QMenu, QAction, QLabel

import os
import sys
import librosa
import librosa.display
import matplotlib.pyplot as plt

import numpy as np

import Spectrogram_Analysis
import Console

class Engine_Analizer (QMainWindow):
    top_indent = 20                 #Відступ зверху зображення спектограм
    indent_between_spectrogram = 10     #Відступ між двума зображеннями спектограм при порівнянні

    def __init__(self):
        super().__init__()
        self.init_ui()
    def init_ui(self):
        font = QtGui.QFont()
        font.setPointSize(10)

        self.menubar = QMenuBar(self)
        self.menubar.setGeometry(0, 0, 800, self.top_indent)
        self.menuFile = QMenu(self.menubar)
        self.menuFile.setTitle("Файл")
        self.menuAnalize = QMenu(self.menubar)
        self.menuAnalize.setTitle("Проаналізувати")

        self.actionAnalize = QAction()
        self.actionAnalize.setText("Аналіз запису")
        self.actionAnalize.triggered.connect(self.analize_spectrogram)
        self.menuAnalize.addAction(self.actionAnalize)
        self.actionCompareSpectrogrum = QAction()
        self.actionCompareSpectrogrum.setText("Порівняти спектрограми")
        self.actionCompareSpectrogrum.triggered.connect(self.compare_spectrogram)
        self.menuAnalize.addAction(self.actionCompareSpectrogrum)
        self.actionCompareSpectr = QAction()
        self.actionCompareSpectr.setText("Порівняти спектри")
        self.actionCompareSpectr.triggered.connect(self.compareSpectr)
        self.menuAnalize.addAction(self.actionCompareSpectr)

        self.setMenuBar(self.menubar)

        self.actionDownload = QAction()
        self.actionDownload.setText("Завантажити файл як спектрограму")
        self.actionDownload.triggered.connect(self.load_spectrogram)
```

```
        self.menuFile.addAction(self.actionDownload)
        self.actionSpectr = QAction()
        self.actionSpectr.setText("Завантажити файл як спектр")
        self.actionSpectr.triggered.connect(self.load_spectr)
        self.menuFile.addAction(self.actionSpectr)

        self.menubar.addAction(self.menuFile.menuAction())
        self.menubar.addAction(self.menuAnalize.menuAction())

        self.label = QLabel(self)
        self.label.move(0, self.top_indent)
        self.label.resize(640, 480)

        self.label2 = QLabel(self)
        self.label2.move(self.label.width() + 5, self.top_indent)
        self.label2.resize(0, 0)

        self.setWindowTitle("Аналіз спектрограми роботи автомобіля")
        self.resize(800, 500)

        self.ConsoleWindow = Console.Console()
        self.ConsoleWindow.show()


    def download_spectrogrum(self, label):
        options = QFileDialog.Options()
        file_path, _ = QFileDialog.getOpenFileName(self, "Виберіть файл", "", "Звукові файли (*.mp3)", options=options)
        if file_path:
            #обчислення спектрограми
            y, sr = librosa.load(file_path, sr=None)
            D = librosa.stft(y)
            S_db = librosa.amplitude_to_db(np.abs(D), ref=np.max)
            # Візуалізація спектрограми
            fig, ax = plt.subplots(1)
            librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='log', cmap='magma')
            plt.colorbar(format='%+2.0f дБ')
            plt.title(os.path.basename(file_path))
            plt.ylabel('Частота')

            fig.savefig("spectrum.png")
            pixmap = QtGui.QPixmap("spectrum.png")
            label.setPixmap(pixmap)
            label.setScaledContents(True)  #маштабування зображення до label
            return y, sr
        else:
            return None, None

    def download_spectr(self, label):
        options = QFileDialog.Options()
        file_path, _ = QFileDialog.getOpenFileName(self, "Виберіть файл", "", "Усі файли (*);;Звукові файли (*.mp3)",
options=options)
        if file_path:
            #обчислення спектра
            y, sr = librosa.load(file_path, sr=None)
            n = len(y)
            fft = np.fft.fft(y)
            magnitude = np.abs(fft)  # Амплитуди
            frequency = np.fft.fftfreq(n, 1/sr)  # Частоти

            # Лишаємо тільки позитивні частоти
            positive_freqs = frequency[:n//2]
            positive_magnitude = magnitude[:n//2]

            # Візуалізація спектра
```

```
        fig, ax = plt.subplots(1)
        plt.plot(positive_freqs, positive_magnitude, color='blue')
        #plt.xscale('log')
        plt.yscale('log')
        plt.title(os.path.basename(file_path))
        plt.xlabel("Частота (Гц)")
        plt.ylabel("Інтенсивність (Амплітуда)")
        fig.savefig("spectrum.png")
        pixmap = QtGui.QPixmap("spectrum.png")
        label.setPixmap(pixmap)
        label.setScaledContents(True) #маштабування зображення до розміру label
        return y, sr
    else:
        return None, None

def load_spectr(self):
    self.y, self.sr = self.download_spectr(self.label)

def compareSpectr(self):
    if(self.label.pixmap() is not None):          # Якщо є перше зображення
        y2, sr2 = self.download_spectr(self.label2)
        if(self.label2.pixmap() is not None):      # Якщо є друге зображення
            self.label.resize(int(self.width() / 2), int(self.height() / 2))    #змінюємо розмір
            self.label2.move(self.label.width(), self.top_indent)
            self.label2.resize(self.label.width(), self.label.height())
            SpectrAnalysis = Spectrogram_Analysis.Spectrogram_Analysis()
            cos_sim = SpectrAnalysis.cosinus_compare_spectrgrum(self.y, y2, self.sr, sr2)
            self.ConsoleWindow.GetAnswer(f"Спектри схожі на {cos_sim} %")
    else:
        self.ConsoleWindow.GetAnswer("Помилка! Завантажте спектр!")

def load_spectrogram(self):
    self.y, self.sr = self.download_spectrogrum(self.label)

def compare_spectrogram(self):
    if(self.label.pixmap() is not None):          # Якщо є перше зображення
        y2, sr2= self.download_spectrogrum(self.label2)
        if(self.label2.pixmap() is not None):      # Якщо є друге зображення
            self.label.resize(int(self.width() / 2), int(self.height() / 2))
            self.label2.move(self.label.width(), self.top_indent)
            self.label2.resize(self.label.width(), self.label.height())

            SpectrAnalysis = Spectrogram_Analysis.Spectrogram_Analysis()
            cos_sim = SpectrAnalysis.cosinus_compare_spectrgrum(self.y, y2, self.sr, sr2)
            self.ConsoleWindow.GetAnswer(f"Спектрограми схожі на {cos_sim * 100} %")
    else:
        self.ConsoleWindow.GetAnswer("Помилка! Завантажте спектрограму!")

def analize_spectrogram(self):
    if(self.label.pixmap() is not None):          # Якщо є перше зображення
        Breakdowns = []
        SpectrAnalysis = Spectrogram_Analysis.Spectrogram_Analysis()
        Breakdowns = SpectrAnalysis.Breakdown_Analysis(self.y, self.sr)
        if len(Breakdowns) != 0:
            self.ConsoleWindow.GetAnswer("Знайдено проблема! " + ", ".join(Breakdowns) + "!")
        else:
            self.ConsoleWindow.GetAnswer(f"Проблеми відсутні!")
    else:
        self.ConsoleWindow.GetAnswer("Помилка! Завантажте спектрограму!")

def resizeEvent(self, event):
    self.resize_label()
```

```
        super().resizeEvent(event)

    def resize_label(self):
        if(self.label2.width() != 0):
            delta_width = self.width() - (self.label.width() + self.label2.width() + 5)
            delta_height = self.height() - (self.label.height() + self.top_indent)
            png_width = self.label.width() + int(min(delta_width, delta_height)/2)  # Зміна ширини (2 малюнки)
            png_height = self.label.height() + int(min(delta_width, delta_height)/2)  # Зміна висоти
            self.label.resize(png_width, png_height)
            self.label2.move(self.label.width() + self.indent_between_spectrogram, self.top_indent)
            self.label2.resize(png_width, png_height)  # width, height
        else:
            delta_width = self.width() - self.label.width()
            delta_height = self.height() - (self.label.height() + self.top_indent)
            png_width = self.label.width() + min(delta_width, delta_height)  # Зміна ширини
            png_height = self.label.height() + min(delta_width, delta_height)  # Зміна висоти
            self.label.resize(png_width, png_height)


if __name__ == "__main__":
    app = QApplication(sys.argv)
    Engine_Analizer_App = Engine_Analizer ()
    Engine_Analizer_App.show()
    sys.exit(app.exec_())
```

A.2 Клас Spectrogram_Analysis

```
from scipy.spatial.distance import cosine

import librosa.display
import numpy as np
import os

import Console

class Spectrogram_Analysis():

    def normalize_spectrogram(spectrogram, min_val=0, max_val=1):
        spectrogram_min = spectrogram.min()
        spectrogram_max = spectrogram.max()
        normalized_spectrogram = (spectrogram - spectrogram_min) / (spectrogram_max - spectrogram_min)
        normalized_spectrogram = normalized_spectrogram * (max_val - min_val) + min_val
        return normalized_spectrogram

    def cut_sound(y1, y2):
        if len(y1) > len(y2):
            y1 = y1[:len(y2)]
        elif len(y1) < len(y2):
            y2 = y2[:len(y1)]
        return y1, y2

    def cosinus_compare_spectrgrum(self, y1, y2, sr1, sr2):

        y1, y2 = Spectrogram_Analysis.cut_sound(y1, y2)
        # Перетворення в спектрограми
        S1 = librosa.stft(y1)
```

```
        S2 = librosa.stft(y2)
        # Амплітудний спектр
        S1_db = librosa.amplitude_to_db(np.abs(S1), ref=np.max)
        S2_db = librosa.amplitude_to_db(np.abs(S2), ref=np.max)

        S1_db = Spectrogram_Analysis.normalize_spectrogram(S1_db)
        S2_db = Spectrogram_Analysis.normalize_spectrogram(S2_db)

        # Перетворення спектрограм у вектори
        vec1 = S1_db.flatten()
        vec2 = S2_db.flatten()

        cos_sim = 1 - cosine(vec1, vec2)
        return cos_sim

    def Breakdown_Analysis(self, y1, sr1):
        folder_path = "Sound_Recording\\"
        Breakdown = []
        ConsoleWindow = Console.Console()
        for file_name in os.listdir(folder_path):
            if os.path.isfile(os.path.join(folder_path, file_name)):
                y2, sr2 = librosa.load(folder_path + file_name, sr=None)
                cos_sim = Spectrogram_Analysis.cosinus_compare_spectrgrum(self, y1, y2, sr1, sr2)
                if(cos_sim > 0.9):
                    Breakdown.append(os.path.splitext(os.path.basename(file_name))[0])
        return Breakdown
```

A.3 Клас Console

```
from PyQt5.QtWidgets import QWidget, QLabel


class Console(QWidget):
    def __init__(self):
        super().__init__()
        self.resize(800, 200)
        self.label = QLabel(self)
        self.label.setGeometry(10, 0, self.width() - 10, self.height())
        self.setWindowTitle("Console")

    def GetAnswer(self, Answer):
        text = self.label.text()
        self.label.setText(text + "\n" + Answer)
```