НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
__інформаційних систем і технологій_
(назва кафедри)

_____ / Швиденко М. З. /
(підпис) (ПІБ)

“___” _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Розробка аудіопрогравача з пітримкою інструментів ШІ»

Спеціальність 122 – «Комп’ютерні науки»

Гарант освітньої програми
_____д. ек. н, професор_____ _____ _____ Руденський Р.А_____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
_____ к. ек. н._____ _____ __Стариченко Є. М._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _Мельниченко Владислав Віталійович_
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
__інформаційних систем і технологій__
(назва кафедри)

___к.ек. н, доцент _____ Швиденко М. З.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
“___” _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Мельниченко Владислав Віталійович

Спеціальність 122 – «Комп’ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Розробка аудіопрогравача з пітримкою інструментів ШІ» затверджена наказом ректора НУБіП України від 16.12.2024 № 2246 “С”
2. Термін подання завершеної роботи на кафедру_____02.06.2025 р._____
(рік, місяць, число)

3. Вихідні дані до роботи: надання інформації про облік військовозобов'язаних на території України у вигляді списків та таблиць.
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Дата видачі завдання “16”_грудня_ 2024 р.

Керівник бакалаврської кваліфікаційної роботи _____/ Стариченко Є. М./
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____/ Мельниченко В. В. /
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У дипломній роботі розглянуто процес розробки багатофункціонального аудіоплеєра для операційної системи Linux із підтримкою елементів штучного інтелекту. Основною метою є створення автономного музичного програвача з інтуїтивно зрозумілим інтерфейсом, розширеним функціоналом (відтворення, пошук, створення плейлістів, звітність) та можливістю інтелектуальних рекомендацій без необхідності постійного доступу до Інтернету.

Система орієнтована на користувачів віком 15–25 років з достатнім рівнем технічної грамотності. Для реалізації проєкту застосовувалися мова програмування Python, база даних PostgreSQL, бібліотеки Gstreamer, Mutagen, Pandas, scikit-learn та PyGObject. У роботі також здійснено порівняльний аналіз існуючих рішень на ринку, сформульовано функціональні й нефункціональні вимоги, побудовано діаграми UML, реалізовано логічну модель бази даних та тригери для автоматизації обробки інформації. Робота підкреслює актуальність розробки з огляду на зростання популярності Linux-середовища та підвищений інтерес до відкритих мультимедійних рішень.

Бакалаврська кваліфікаційна робота складається з чотирьох розділів.

У розділі аналізу предметної області було визначено потреби користувачів та актуальність розробки, а також розглянуто особливості платформи Linux у контексті створення програмного забезпечення.

У розділі інформаційне забезпечення системи було побудовано ER-діаграму, детально описано структуру бази даних, та розглянуто процес її створення та налаштування, включаючи реалізацію тригерів для автоматизації та контролю даних.

У розділі прикладне програмне забезпечення було обґрунтовано вибір мови програмування Python та бібліотеки GTK для розробки основного застосунку, а також окремо розглянуто вибір мови та бібліотек для реалізації моделі штучного інтелекту.

У розділі впровадження системи було описано вимоги до апаратного та програмного забезпечення для розгортання системи, деталізовано склад інсталяційного пакета, а також розглянуто різні типи та методи тестування, що забезпечують коректну та стабільну роботу програми.

ABSTRACT

The thesis considers the process of developing a multifunctional audio player for the Linux operating system with support for artificial intelligence elements. The main goal is to create an autonomous music player with an intuitive interface, extended functionality (playback, search, playlist creation, reporting) and the ability to make intelligent recommendations without the need for constant access to the Internet. The system is aimed at users aged 15–25 with a sufficient level of technical literacy. The project was implemented using the Python programming language, the PostgreSQL database, the Gstreamer, Mutagen, Pandas, scikit-learn, and PyGObject libraries. The work also includes a comparative analysis of existing solutions on the market, the formulation of functional and non-functional requirements, the construction of UML diagrams, the implementation of a logical database model, and triggers for automating information processing. The work emphasizes the relevance of the development in view of the growing popularity of the Linux environment and the increased interest in open multimedia solutions.The Bachelor's qualification paper consists of four sections.

The domain analysis section examines user needs and the relevance of the development, as well as the peculiarities of the Linux platform in the context of software creation.

The information system support section details the construction of an ER-diagram, describes the database structure, and outlines the process of its creation and configuration, including the implementation of triggers for data automation and control.

The application software section justifies the choice of the Python programming language and the GTK library for developing the main application, and separately discusses the selection of the language and libraries for implementing the artificial intelligence model.

The system implementation section covers the hardware and software requirements necessary for system deployment, details the composition of the installation package, and examines various types and methods of testing that ensure the correct and stable operation of the program.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Отримання завдання	16 грудня 2024	
2	Аналіз предметної області	січень 2025	
3	Моделювання предметної області	лютий 2025	
4	Проєктування програмної системи	березень 2025	
5	Розробка програми і тестування	Березень-квітень 2025	
6	Економічне дослідження розробки та експлуатації програмної системи	травень 2025	
7	Написання пояснювальної записки	травень 2025	
8	Перевірка на плагіат	травень 2025	
9	Проходження нормо контроля	травень 2025	
10	Проходження передзахисту	2 червня 2025	
11	Захист роботи	червень 2025	

Студент _____ Мельниченко В. В.

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

к.ек.н., доцент Стариченко Є. М.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі користувача вже важко здивувати новітніми аудіоплеєрами з підтримкою штучного інтелекту, які забезпечують високоякісне відтворення музичного контенту. Проте, прагнення споживачів до розширених можливостей, які не завжди залежать від постійного доступу до мережі Інтернет та надають більшу свободу вибору режимів відтворення, вимагає нових підходів. Частиною сучасного тренду є усвідомлення того, що надмірне покладання на ШІ може призвести до неточностей або зменшити цінність досвіду через його надмірну автоматизацію, що іноді сприймається як фінансово або часово затратне.

Метою дипломної роботи є створення багатофункціонального аудіоплеєра для операційних систем Linux, який забезпечуватиме автоматизовані елементи керування, такі як клавіатурні скорочення, та інтегруватиме ШІ-помічника для покращення користувацького досвіду прослуховування музики.

Актуальність цього проєкту підтверджується зростаючим попитом на зручні та автоматизовані рішення для відтворення музики. Можливість керувати музикою без додаткових рухів, а також автоматизоване перемикання треків значно підвищують комфорт користувача.

Додатково, спостерігається значне зростання популярності операційних систем Linux. За останні три роки використання Linux-систем збільшилося на 25% у порівнянні з попереднім роком, тоді як операційні системи Windows втрачають свою аудиторію. Це може бути зумовлено, зокрема, високими технічними вимогами Windows 11 та закінченням підтримки безпеки для Windows 10 у 2025 році. Зростання популярності Linux серед програмістів та звичайних користувачів підтверджується статистикою за 2024 рік. [4]:

З огляду на зростання використання Linux-систем у різних сферах, набуття навичок роботи з ШІ в цьому середовищі стає дедалі актуальнішим.

Також варто зазначити потенціал розширення використання Linux-систем, зокрема Ubuntu, на мобільних пристроях. Очікується, що з оновленнями Android 15-16, консоль буде інтегрована в систему, що може сприяти масовому впровадженню Linux на смартфонах, враховуючи, що Android вже базується на Linux.

Для розробки аудіоплеєра використовуються такі основні технології:

- Python
- PostgreSQL

Для забезпечення функціональності програми необхідні наступні бібліотеки Python, які встановлюються за потреби:

- Gstreamer
- Mutagen
- psycopg2
- pandas
- PyGObject
- scikit-learn

Використання даних бібліотек гарантує стабільну та ефективну роботу програми.

Створений аудіоплеєр з підтримкою штучного інтелекту є відповіддю на сучасні вимоги користувачів, які прагнуть до автоматизації та розширених можливостей у відтворенні музики, при цьому зберігаючи контроль над процесом та незалежність від постійного доступу до Інтернету, особливо в контексті зростаючої популярності Linux-систем.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області та постановка завдання

Для успішної реалізації проєкту важливо чітко визначити цільову аудиторію. Це дасть змогу адаптувати функціонал, інтерфейс та маркетингову стратегію програми. Розроблюваний додаток призначений для користувачів операційної системи Linux.

Дослідження [2] показують, що серед користувачів Linux спостерігається значний переважання чоловіків. Аналіз джерел, таких як "Why are there so few women in Linux?", вказує на те, що низька кількість жінок у Linux-спільноті пов'язана не з відсутністю інтересу чи здібностей, а скоріше із соціальними та психологічними бар'єрами.

Для визначення вікового діапазону користувачів розглянемо декілька аспектів:

- Музика: Музика є універсальною і актуальною для всіх вікових груп, від дітей до літніх людей.
- Користання комп'ютера: Комп'ютерами активно користуються люди віком від 5 до 50 років. Технічна грамотність високо цінується в сучасному світі.
- Використання Linux: Користувачами Linux переважно є люди віком від 12 до 35 років, що пояснюється складністю системи та її специфічними вимогами. Проте, Linux також може бути встановлений на старіші комп'ютери для батьків або інших членів родини через його низькі системні вимоги, що розширює потенційний віковий діапазон до 12-40 років.
- Встановлення мультимедійних програм: Дорослі користувачі часто уникають самостійного встановлення таких програм.

Зважаючи на необхідність певної технічної грамотності для встановлення та налаштування програми в Linux, найбільш релевантним віковим сегментом є

15-25 років. Підлітки та молоді люди у цьому віці мають більше вільного часу та інтересу до освоєння нових інструментів та технологій. Для цієї аудиторії важливим є детальний опис процедури встановлення програми на GitHub.

З огляду на вищезазначене, цільовою аудиторією програмного додатку є переважно чоловіки віком 15-25 років. Це дозволяє нам припустити, що вони є технічно досвідченими користувачами, які здатні освоювати складніші функції та цінуватимуть сучасний, можливо, "брутальний" дизайн та інноваційні можливості програми. Ми можемо відхилятися від "стандартних" рішень, надаючи більш розширені функціональні можливості, що компенсується підвищеною інтерактивністю та сучасним стилем, який є привабливим для цієї демографічної групи.

Вимоги бізнес рівня.

Розробка та поширення додатку здійснюється без жодних фінансових витрат для кінцевого користувача. Це означає, що програма не вимагає інвестицій або будь-яких грошових внесків для свого повноцінного функціонування. Таким чином, економічні ризики та потенційні втрати для користувачів повністю відсутні. Зазначений підхід унеможливлює необхідність перетворення програми на платний продукт або обмеження доступу до її функціоналу комерційною ліцензією.

Аудіоплеєр планується поширювати за моделлю відкритого програмного забезпечення (Open Source). Це передбачає повний доступ до її вихідного коду, що дозволяє будь-кому вільно завантажувати, вивчати, модифікувати та адаптувати програму відповідно до власних потреб. Такий підхід не тільки сприяє прозорості та довірі з боку спільноти, але й відкриває широкі можливості для:

- Спільного вдосконалення: Користувачі та розробники можуть долучатися до покращення функціоналу.
- Кастомізації: Програма може бути легко налаштована під індивідуальні вимоги.

- Стабільного розвитку: Спільнота допомагає виявляти та виправляти помилки, підвищуючи якість продукту.

Завдяки відкритості вихідного коду, користувачі з необхідними технічними навичками можуть не лише адаптувати програму під свої специфічні потреби, але й вносити нові функції або усувати виявлені недоліки. Це повністю відповідає філософії вільного програмного забезпечення, де основними цінностями є свобода вибору, модифікації та колективної розробки.

Опишемо ключові вимоги до функціоналу програми з позиції кінцевого користувача, що забезпечить інтуїтивно зрозуміле та ефективне використання.

Функціональні вимоги:

- Відтворення музики: Відтворення обраного треку здійснюється за допомогою одного натискання на елемент у списку відтворення.
- Пошук треку: Інтерфейс повинен включати поле "Пошук" для швидкого фільтрування музичних композицій за назвою або ім'ям виконавця.
- Додавання музичних папок: Забезпечити функцію додавання папок, що містять музичні файли. Натискання кнопки "+" відкриває діалогове вікно для вибору папки, після чого всі MP3-файли з обраної директорії автоматично імпортуються до бази даних програми.
- Видалення даних: Передбачити кнопку "–", яка дозволяє очистити всю інформацію з бази даних програми та оновити список відтворення.
- ШІ-рекомендації: Реалізувати окрему кнопку з іконкою "AI", яка буде пропонувати наступний трек на основі аналізу історії прослуховувань користувача за допомогою алгоритмів штучного інтелекту.
- Генерація звітів: Надати можливість генерації звітів про прослуховування. Це передбачає наявність діалогового вікна "Звіт" з календарями для вибору часового періоду та функцією експорту згенерованих даних.
- Налаштування інтерфейсу: Інтерфейс повинен включати кнопку

"Налаштування", яка відкриває панель з перемикачами для зміни теми оформлення та керування звуковими ефектами.

Вимоги до інтерфейсу користувача (UI/UX):

- Темна тема: За замовчуванням інтерфейс програми повинен використовувати темну тему оформлення. Колір фону: #121212. Колір акцентних елементів: #ff4081.
- Прогрес-бар: Інтерфейс повинен відображати прогрес відтворення поточного треку за допомогою візуального елемента, такого як Gtk.ProgressBar.
- Часові лейбли: Поточний час відтворення та загальна тривалість треку повинні відображатися у форматі "XX:XX" (хвилини:секунди).

Ці вимоги є основою для розробки функціоналу та дизайну програми, орієнтованої на максимальну зручність та ефективність для кінцевого користувача.

Обов'язкові функції, які повинна виконувати програма для задоволення потреб користувача.

1 CRUD треків: Реалізація повного циклу управління музичними треками:

- C (Create): Імпорт MP3-файлів з обраних користувачем папок.
- R (Read): Зчитування та відображення метаданих треків (назва, виконавець, альбом тощо).
- U (Update): Можливість редагування метаданих треків.
- D (Delete): Видалення треків з бази даних.
- Усі метадані треків повинні зберігатися в базі даних PostgreSQL.

2 Плейлісти: Можливість створення, збереження та завантаження користувацьких плейлістів.

3 Керування відтворенням: Забезпечення миттєвого відгуку на натискання кнопок:

- Play/Pause: Запуск та призупинення відтворення.
- Next: Перехід до наступного треку.

- Prev: Перехід до попереднього треку.

4 Функція пошуку: Динамічне оновлення списку треків у відповідь на введення тексту в поле Gtk.Entry для фільтрації за назвою або виконавцем.

5 ШІ-модуль: Розробка модуля штучного інтелекту, який аналізуватиме дані з таблиці try_listening (історія прослуховувань) та викликатиме API рекомендацій для пропозиції наступного треку.

6 Звітність: Формування звітів у форматі Excel-файлу, що включатиме:

- Топ-5 найбільш прослуховуваних треків.
- Топ-1 найбільш прослуховуваний виконавець.
- Загальний час прослуховування за обраний період.

7 Стабільний прогрес-бар: Безперебійне оновлення індикатора прогресу відтворення (Gtk.ProgressBar) за допомогою таймера GLib.timeout_add без "зависань" інтерфейсу.

Обробка зображень: Читання вбудованих обкладинок альбомів з MP3-файлів за допомогою бібліотеки mutagen та їх масштабування за допомогою GdkPixbuf.

8 Логування: Запис інформації про помилки та події відтворення у спеціалізований файл app.log.

9 Регуляція гучності: Наявність повзунка або іншого елемента керування для зміни рівня гучності звуку через Gstreamer.

Нефункціональні вимоги

Цей розділ визначає якісні атрибути та обмеження, яким повинна відповідати програма, не впливаючи безпосередньо на її основний функціонал.

1 Відповідність стилю: Всі елементи керування користувацького інтерфейсу (кнопки, списки, поля введення) повинні бути оформлені відповідно до визначених CSS-правил.

2 Кросплатформенність: Програма повинна бути сумісною та запускатися на різних дистрибутивах Linux з використанням бібліотеки GTK 3 та мови

програмування Python 3.

3 Час старту: Час завантаження інтерфейсу та підключення до бази даних не повинен перевищувати 2 секунди.

4 Продуктивність UI: Користувацький інтерфейс не повинен "підвисати" або гальмувати при роботі з великою кількістю треків (понад 10 000).

5 Надійність: Система повинна забезпечувати:

- Автоматичне перепідключення до бази даних у разі виникнення тайм-ауту з'єднання.
- Коректну обробку помилок для запобігання збоям.

6 Безпека: Забезпечення захисту конфіденційних даних, зокрема:

- Захист рядків підключення до бази даних (паролі не повинні зберігатися у відкритому коді).
- Шифрування звітів, якщо це буде необхідно для захисту чутливої інформації.

7 Локалізація: Підтримка двох мов інтерфейсу: української та англійської.

8 Масштабованість: Архітектура програми повинна дозволяти подальше розширення функціоналу, включаючи додавання плагінів або підтримку додаткових аудіоформатів.

9 Тестованість: Наявність модульних тестів для перевірки коректності роботи ключових функцій (імпорт треків, відтворення, генерація звітів).

10 Лог-ротація: Автоматичне обмеження розміру файлу app.log до 5 MB з архівацією або видаленням старих лог-файлів для запобігання надмірному використанню дискового простору.

Для забезпечення розширеного функціоналу аудіоплеєра, зокрема функцій ШІ-рекомендацій, ми працюємо з багатогранною інформацією про музичні композиції. Кожна пісня аналізується за такими акустичними властивостями [1]:

- Функції Фур'є (DFT/FFT): Дозволяють заглибитися в спектральний

аналіз, виявляючи басові лінії, середньочастотні та високочастотні компоненти звуку.

- Короткочасне перетворення Фур'є (STFT): Аналогічно DFT/FFT, STFT використовується для аналізу басового рівня та спектрального складу звуку в часовій області.
- Гармонія та мелодія: Модель гармонійної та синусоїдальної декомпозиції, що дозволяє відстежувати фундаментальну частоту та гармоніки мелодії, визначаючи її музичну структуру.
- Текстура та тембр: Залишковий компонент, що описує шумові частоти сигналу, визначає щільність та "колір" звуку, формуючи його унікальний тембр.
- Ритм та темп: Виявлення перехідних процесів та обчислення пульсу допомагає аналізувати ритмічну структуру треку, визначаючи його темп.
- Динаміка та артикуляція: Метод оцінки амплітудної огинаючої та стохастичної компоненти, що описує динамічний контраст та експресивність виконання.
- Звукові перетворення та фільтрація: Практики фільтрації, морфінгу та часово-частотного масштабування демонструють, як впливати на різні частотні діапазони, змінюючи звучання композиції.

Крім акустичних даних, програма також оперує метаданими, які можна отримати з файлів або зовнішніх джерел:

- Виконавець: Інформація про автора або групу, що створила пісню, включаючи їхнє ім'я, походження, основний жанр та інструментальний склад (якщо це група).
- Жанр: Класифікація пісні за музичним жанром, що є ключовим для пошуку та рекомендацій.
- Альбом: Сукупність треків, що можуть бути частиною стандартного альбому (зазвичай 12-13 пісень) або EP (3-6 пісень).

- Текст: Лірика пісні та мова, якою вона виконується.
- У майбутньому, у випадку помилок бібліотеки при отриманні даних з MP3-файлів, передбачається можливість автоматичного парсингу серверів для вилучення та збереження цих метаданих у текстовому форматі в базі даних.

Створювана система класифікується як інформаційно-довідкова та мультимедійна система для прослуховування музики.

Програма розробляється з такими ключовими характеристиками:

- Локальна система: Функціонує переважно в офлайн-режимі, використовуючи локальні ресурси.
- Підтримка бази даних: Дані зберігаються та керуються за допомогою бази даних.
- Графічний фреймворк: Інтерфейс користувача створюється за допомогою графічного фреймворку.

Часткова автоматизація: Користувач обирає пісню для відтворення, тоді як програма автоматично керує процесом. Також планується інтеграція елементів автоматизації та можливість легкого налаштування системи через конфігураційний файл, що дозволить змінювати поведінку програми без значних зусиль.

1.2 Огляд існуючих рішень

Серед найпопулярніших рішень для прослуховування музики, що домінують на ринку, варто виділити Spotify та YouTube. У екосистемі Linux особливо поширеним є Rhythmbox. Розглянемо детальніше особливості кожного з них.

Spotify — це провідна кросплатформна сервісна платформа для потокового відтворення музики, доступна як на мобільних пристроях, так і на персональних комп'ютерах [8]. Вона користується надзвичайною популярністю в операційній системі Windows, де пропонується для завантаження через

Microsoft Store. Для користувачів Linux існують спеціальні версії Spotify, включаючи можливість керування відтворенням через консоль за допомогою спеціального пакета.

Існує припущення, що Spotify співпрацює з Apple, надаючи інформацію про майбутні оновлення за два тижні до їхнього офіційного релізу. Щодо політики компанії, Spotify активно підтримує Україну, обмежуючи доступ для користувачів з країн-агресорів, хоча обійти ці обмеження можливо за допомогою VPN та інших методів.

Основний функціонал Spotify:

- Зручна пошукова система: Дозволяє легко знаходити виконавців, треки та альбоми.
- Профілі виконавців: Містять біографії, інформацію про тури та концерти, а також щомісячні рейтинги топ-10 треків за кількістю прослуховувань.
- Детальна інформація про альбоми: Надається хронологія випусків, включаючи EP та спільні роботи (feat.) з посиланнями на інших виконавців.
- Персоналізовані плейлісти: Можливість створювати, редагувати, видаляти та відтворювати власні плейлісти, синхронізовані між пристроями.
- ШІ-рекомендації: Система пропонує нові треки на основі аналізу прослуханої музики. Хоча вона інтелектуальна, якість рекомендацій може варіюватися.
- Щоденні мікси та релізи тижня: Автоматично генеровані добірки музики, включаючи топ української музики. Суб'єктивно, такі добірки не завжди відповідають специфічним музичним вподобанням.
- Офлайн-режим: Доступний для преміум-користувачів, дозволяє завантажувати музику на пристрій для прослуховування без підключення до Інтернету.
- Спільне прослуховування: Можливість створювати групи для

прослуховування музики з друзями, знайомими чи родиною.

YouTube [10] став домінуючою платформою для обміну відеоконтентом, що надає можливості для заробітку творцям та є джерелом навчання та розваг для користувачів. Музичні групи та виконавці часто публікують свої кліпи та треки саме тут, використовуючи платформу для просування та монетизації контенту. З кожним роком збільшується кількість рекламних інтеграцій. YouTube доступний як онлайн-версія, так і на мобільних пристроях. На Linux-системах існують сторонні рішення для встановлення YouTube Music, тоді як на Windows таких нативних рішень немає.

Основні функції YouTube:

- Безкоштовний доступ: Широкий вибір кліпів та музики.
- Інтерактивні можливості: Коментування, оцінки ("вподобайки"), можливість включати ремікси та запити. Ремікси можуть бути проблематичними через часті повтори, що може дратувати.
- Автоматична черга відтворення: Формується на основі попередніх переглядів або пошукових запитів.
- Створення та обмін плейлістами: Можливість створювати та ділитися власними списками відтворення.
- Розширена пошукова система: Дозволяє знаходити композиції не лише за назвою, альбомом чи виконавцем, а й за фрагментом тексту або обкладинкою відео.
- Опції перегляду: Вибір між переглядом відео або прослуховуванням аудіо.
- Фоновий режим: Доступний після оплати місячної передплати.
- Синхронізація з Google-акаунтом: Надає доступ до всіх попередніх налаштувань та бібліотеки на різних пристроях.

Rhythmbox — це музичний плеєр, розроблений спеціально для операційних систем Linux, зокрема для робочого оточення GNOME [9]. Він є

одним із найпопулярніших MP3-плеєрів серед користувачів Linux і часто встановлюється за замовчуванням у таких дистрибутивах, як Fedora. Rhythmbox орієнтований на зручне відтворення локальної музики. Система була розроблена з акцентом на швидкість за допомогою мови програмування C++.

Основні функції Rhythmbox:

- Автоматичне сканування та додавання треків: Сортує музику за жанром, альбомом та виконавцем. Хоча система прагне до точності, можуть виникати некоректні назви виконавців або розділення одного виконавця на декілька записів. Назви треків та жанрів здебільшого коректні.
- Створення плейлістів: Можливість створювати нові списки відтворення.
- Пошук: Функція пошуку за піснею та альбомом.
- Редагування та видалення: Можливість редагувати метадані та видаляти пісні.
- Підтримка плагінів: Дозволяє розширювати функціонал програми за допомогою плагінів, розроблених іншими спільнотами.
- Підтримка інтернет-радіо: Можливість прослуховування іноземних радіостанцій.
- Простота та стабільність: Розробка на C++ сприяла забезпеченню високої швидкості та відносної стабільності, хоча іноді можуть виникати неточності в категоризації даних.

В таблиці 1.1 наведене порівняння трьох описаних вище музичних платформ та плеєрів та дозволяє оцінити їхні переваги та недоліки за ключовими критеріями.

Таблиця 1.1

Порівняння музичних платформ

Критерій	**Spotify**	**YouTube / YouTube Music**	**Rhythmbox**
1. Платформа	Хмарна, стрімінг	Хмарна, відео/аудіо платформа	Десктопна, локальна музика
2. Автоматичний підбір музики	За допомогою ШІ	За допомогою ШІ або історії переглядів (ідентично)	Відсутній
3. Використання ШІ	Є (персоналізація, настрій, плейлісти)	Є (алгоритмічні рекомендації)	Відсутній
4. Алгоритмічне "нав'язування"	Частково, особливо без преміум	Так, алгоритм змінює чергу	Відсутнє, повністю ручний вибір
5. Можливість слухати без впливу алгоритмів	Частково (офлайн або свої списки)	Практично неможливо	Повна свобода вибору
6. Пошук за настроєм чи жанром	Підтримується	Можна, але менш зручно	Вручну або за тегами
7. Офлайн-режим	Є у преміум-версії1	Є у підписників Premium	Працює з локальними файлами, завжди доступний
8. Інтеграція з іншими сервісами	Широка (Discord, Last.fm, Smart TV)	Google Assistant, Android TV	Обмежена (Last.fm, подкасти)
9. Простота інтерфейсу	Сучасний, зручний	Перевантажений відео та рекламою	Класичний, простий інтерфейс
10. Якість звуку	До 320 Kbps у преміум	До 256 Kbps	Залежить від якості локального файлу
11. Потреба в Інтернеті	Потрібен для стрімінгу	Потрібен для стрімінгу	Не потрібен (офлайн)
12. Наявність реклами	Залежить від регіону (Японія: кожні 2-3 пісні; Україна: 1 реклама на 20-50 пісень)	30-40 секунд реклами на 15 хвилин прослуховування (за досвідом користувача)	Відсутня

Алгоритмічне нав'язування контенту є суттєвим недоліком сучасних платформ, особливо коли йдеться про розважальний контент.

Платформа YouTube активно просуває короткі відео (Shorts), які, попри свою привабливість, часто є часово затратними і не надають конкретної інформації. ШІ, розвиваючись, ефективно адаптує цей контент під уподобання користувача, що може призвести до надмірного споживання часу. Це створює "вірусний" ефект: користувачі без конкретної мети, зайшовши на YouTube,

легко "залипають" на годинами, витрачаючи більше часу, ніж планувалося. Такий алгоритмічний підхід може обмежувати свободу вибору користувача, оскільки він постійно отримує той контент, який, за оцінкою ШІ, йому сподобається, а не той, який він міг би активно шукати. Для багатьох це призводить до емоційного та розумового перевантаження. Хоча повнометражні відео надають більше змістовної інформації, вони також можуть мати подібний ефект "залипання". Особистий досвід показує, що краще провести годину за комп'ютерною грою, ніж випадково "залипнути" на YouTube, що може викликати більшу нервовість та втому. Важливо зазначити, що ці алгоритми не завжди ефективно поширюються на музику, оскільки у 90% випадків пропонуються вже прослухані або подібні композиції.

Хоча система рекомендацій Spotify є більш помірною, вона також має свої недоліки. Алгоритм часто віддає перевагу музиці, яка є найбільш популярною або часто прослуховуваною, а не тій, що може справді сподобатися користувачеві своєю унікальністю. Цей підхід, ймовірно, був посилений для користувачів без преміум-підписки на мобільних пристроях, які мали обмежені можливості випадкового вибору. Це призводить до того, що багато пісень, які могли б бути цікавими, пропускаються і не потрапляють у рекомендації через низький показник прослуховування. Водночас, у моменти, коли вся звичайна музика набридла і хочеться просто розслабитися, жоден алгоритм не може повністю задовольнити цю потребу; у таких випадках рятує лише повний випадковий вибір пісень з телефону.

1.3 Моделювання предметної області

Для ефективного моделювання предметної області та чіткого визначення напрямку розробки використовуються UML-діаграми. Ці діаграми дозволяють:

- Конкретизувати цілі: Чітко описати завдання, що стоять перед командою або індивідуальним розробником програми.

- Відобразити взаємодію: Деталізувати можливі дії користувача та реакції програми на конкретні сценарії.
- Зрозуміти функціонал: Візуалізувати ключові можливості програми (наприклад, наявність облікового запису, можливість придбання преміум-підписки тощо).
- Надати покрокове розуміння: Проілюструвати кожен етап взаємодії, від введення даних до отримання результату.

Діаграма варіантів використання (Use-Case Diagram)

Діаграма варіантів використання описує взаємодію між Акторами (зовнішніми об'єктами, що взаємодіють із системою) та Варіантами використання (функціями, які система надає). У нашому випадку головними акторами є "Користувач" та "ШІ". Додавання "Програми" як окремого актора було б менш коректним, оскільки вона є інструментом, що виконує дії, а не окремим суб'єктом. [11]

На діаграмі варіантів використання дії зазвичай розташовуються за пріоритетом або послідовністю, починаючи з найважливіших зверху.

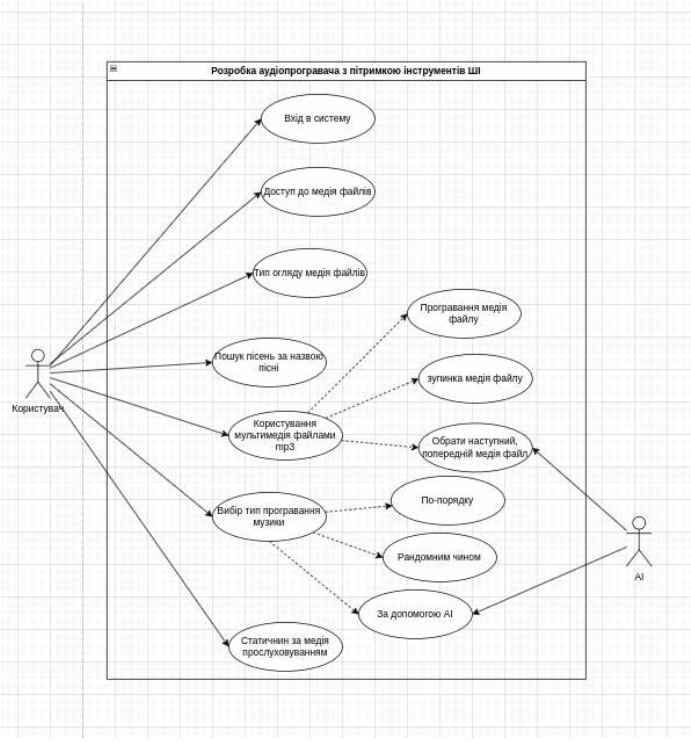


Рис. 1.1 – Діаграма Use-case

Згідно діаграми Use-case (рис. 1.1) «Користувач» може:

- Вхід в систему: Аутентифікація користувача для доступу до функцій програми.
- Надання доступу до файлів: Дозвіл програмі на доступ до локальних музичних файлів.
- Вибір типу огляду медіафайлів: Вибір способу відображення медіатеки (наприклад, за альбомами, виконавцями, жанрами).
- Виконання пошуку пісень за назвою: Пошук конкретних композицій за їхньою назвою.
- Керування відтворенням мультимедійних файлів:
  - Програвання медіафайлу: Запуск відтворення обраного треку.
  - Запуск медіафайлу: Повторний запуск треку або його відновлення.
  - Запуск наступного/попереднього медіафайлу: Перехід до наступного або попереднього треку в черзі відтворення. Ці дії можуть бути вдосконалені «ШІ» та/або «ШІ» може згенерувати наступний трек за алгоритмом.
- Вибір типу програвання мультимедіафайлу: Вибір режиму відтворення:
  - Послідовно: Відтворення треків у порядку списку.
  - Рандомно: Випадковий порядок відтворення.
  - За допомогою штучного інтелекту: Використання ШІ для формування черги відтворення.
- Отримання статистики прослуховувань: Запит та перегляд статистики прослуховувань мультимедіа за обраний період.

«ШІ» може:

- Додавання даних про вибір треків до бази даних: Фіксація в базі даних вибору користувачем наступних/попередніх треків, що дозволяє ШІ навчатися.
- Генерація ідеального треку в режимі «ШІ»: На основі зібраних

даних, ШІ формує ідеальну послідовність треків для користувача.

- Генерація послідовності треків за допомогою «ШІ»: Створення черги відтворення, оптимізованої ШІ.
- Ці елементи діаграми надають більш конкретне розуміння цілей та функціоналу, які ми прагнемо реалізувати у програмі.

Діаграма послідовності є ключовим інструментом для детального опису покрокових взаємодій між елементами системи. Вона ілюструє послідовність повідомлень, які передаються між об'єктами, та їхню реакцію на дії користувача [13]. Ця діаграма тісно пов'язана з діаграмою варіантів використання, надаючи детальний опис кожної події та її наслідків.

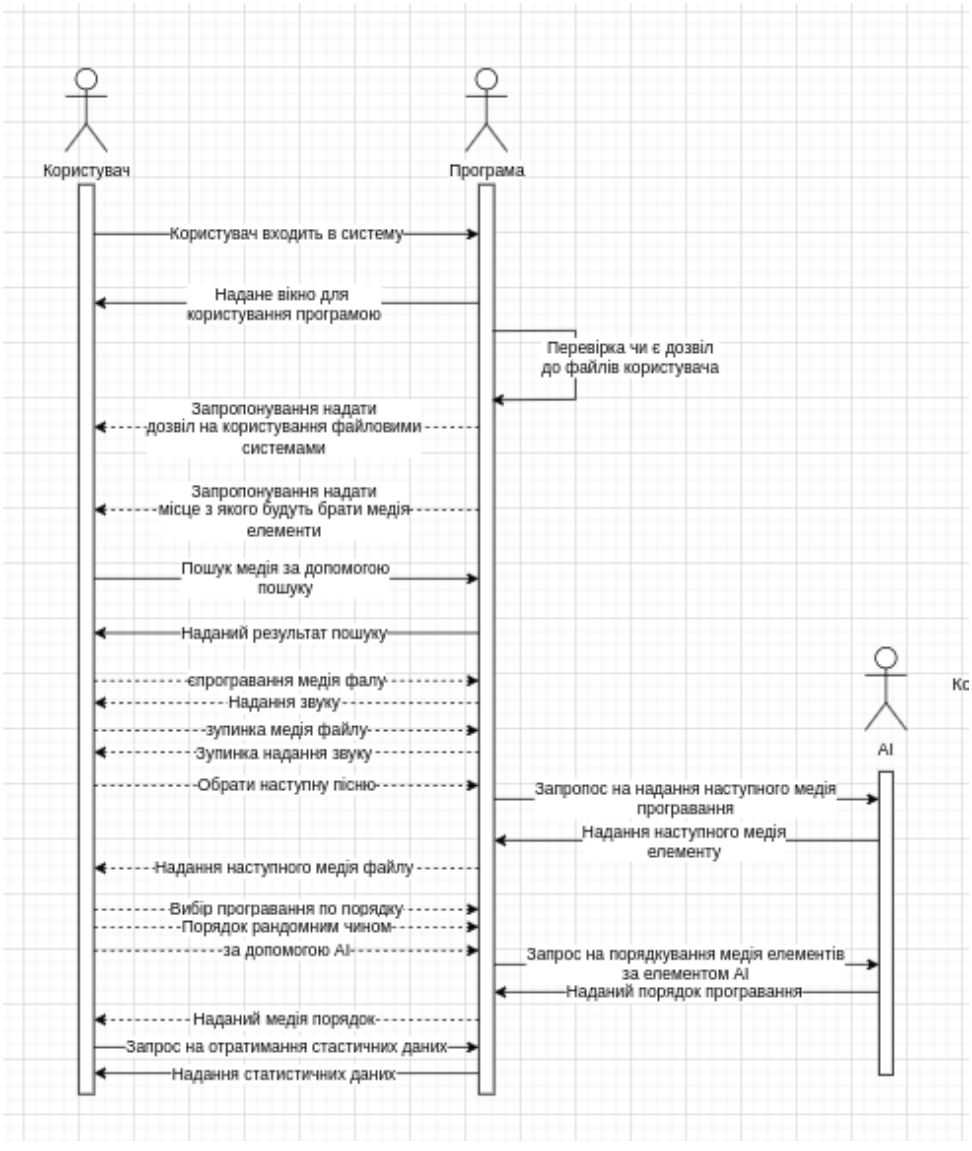


Рис. 1.2 – Діаграма активності1

Згідно діаграми послідовності (рис. 1.2):

1. Користувач: Входить в систему.
2. Програма: Надає вікно для користування програмою.
3. Програма: Перевіряє наявність доступу до файлової системи.
4. Програма: Якщо доступу немає, надсилає запит на його надання.
5. Програма: Запитує вказівку папки у файловій системі для імпорту MP3-файлів.
6. Користувач: Вводить пошуковий запит у поле пошуку.
7. Програма: Надає результати пошуку.
8. Користувач: Запускає відтворення медіафайлу.
9. Програма: Відтворює звуковий потік медіафайлу.
10. Користувач: Зупиняє відтворення медіафайлу.
11. Програма: Зупиняє звуковий потік.
12. Користувач: Обирає запуск наступної пісні.
13. Програма: Надсилає запит на генерацію наступної пісні до ШІ.
14. ШІ: Надає найкращий підбір наступного медіаелемента.
15. Програма: Відтворює користувачеві наступну пісню, обрану ШІ.
16. Користувач: Обирає відтворення музики по порядку.
17. Програма: При наступному відтворенні надає пісню, що йде наступною у візуальному списку.
18. Користувач: Обирає відтворення музики рандомним чином.
19. Програма: При наступному відтворенні надає абсолютно випадкову пісню.
20. Користувач: Обирає відтворення музики за допомогою ШІ.
21. Програма: Надсилає запит на генерацію плейліста до ШІ.
22. ШІ: Надає згенерований плейліст.
23. Програма: Відтворює музику з плейліста, і при потребі наступний трек також береться з цього плейліста.
24. Користувач: Надсилає запит на отримання статистичної інформації.

25. Програма: Надає статистичну інформацію.

Діаграма активності деталізує послідовність дій та процесів, що відбуваються в системі, дозволяючи створити алгоритм для їх реалізації. Вона зазвичай розробляється на основі діаграм варіантів використання або послідовності, щоб описати кожну дію крок за кроком. **[12]**

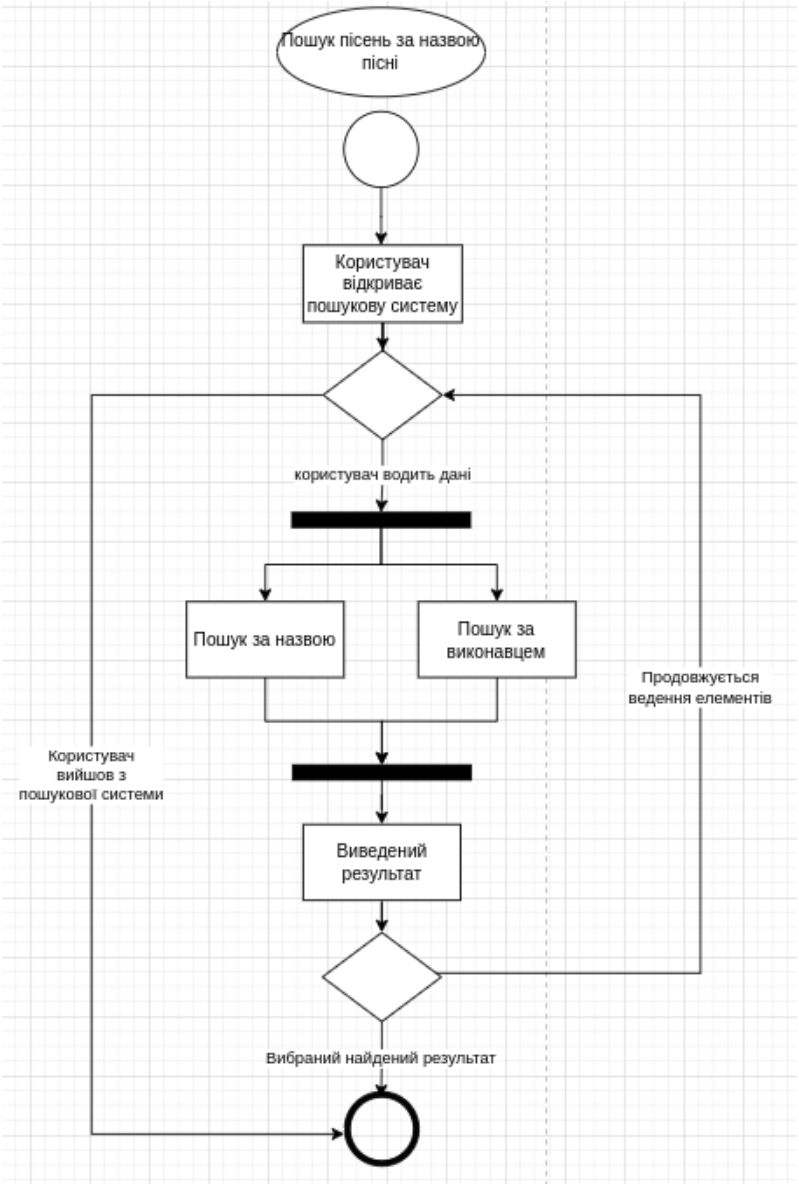


Рис. 1.3 – Діаграма актививності

Опис діаграми активності (для елемента "Доступ до медіафайлів" та "Пошукова система"):

Користувач відкриває пошукову систему.

- Потік рішень:
  - Користувач може вийти з пошукової системи, і функція завершиться.
  - Користувач вводить дані (пошуковий запит).
- Користувач надає дані (підтверджує введення).
- Дія:
  - Пошукова система надає можливі елементи за назвою пісні.
  - Пошукова система надає можливі елементи за назвою групи.
- Результат: Виведений результат пошукової системи.
- Потік рішень:
  - Користувач може продовжувати пошук (повернутися до кроку 2).
- Користувач може обрати музику, і пошук завершується.

Даталогічний опис інших додаткових функціональних елементів знаходиться у Додатку А.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Логічна модель даних

Для ефективного зберігання та управління інформацією, з якою працює програма, була розроблена логічна модель даних. Ця модель є основою для створення структури бази даних SQL. Варто зазначити, що в процесі реалізації моделі можливі незначні зміни, оскільки початкове проєктування є динамічним процесом.

ER-діаграма (Entity-Relationship Diagram) дозволяє візуалізувати сутності (таблиці) бази даних та зв'язки між ними. Вона слугує шаблоном для побудови реальної бази даних. [14]

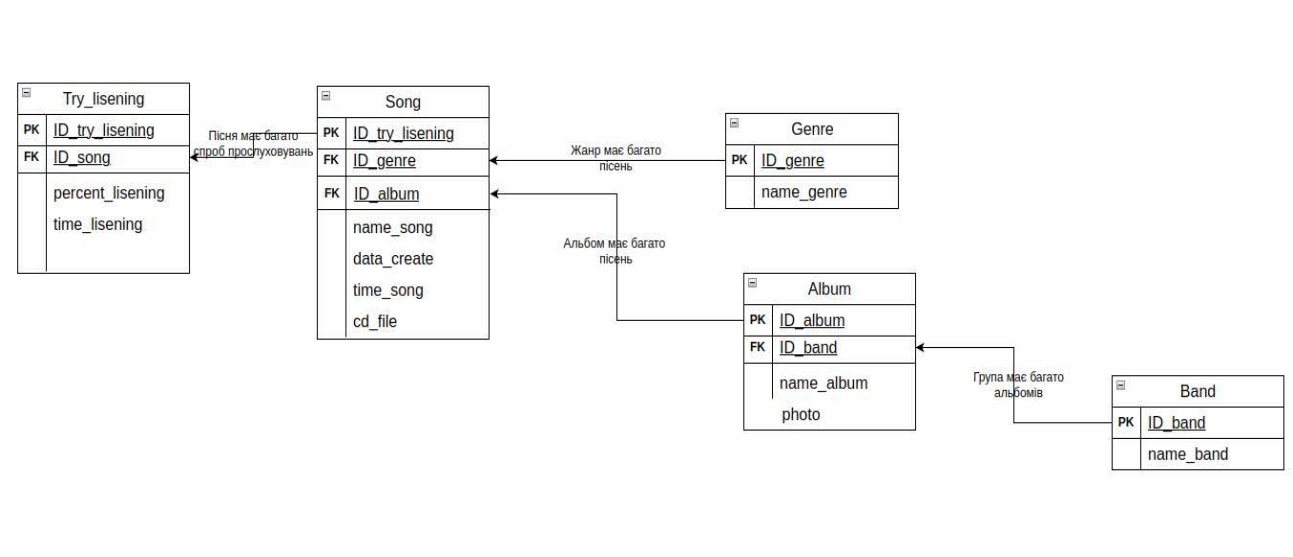


Рис. 2.1 – ER-діаграма

Опис ER-діаграми (рис. 2.1):

- Сутність: Band (Гурт)
  - ID_band (PK): Унікальний ідентифікатор гурту (первинний ключ).
  - name_band: Назва гурту.
- Сутність: Album (Альбом)
  - ID_album (PK): Унікальний ідентифікатор альбому (первинний ключ).
  - ID_band (FK): Зовнішній ключ, що посилається на Band.ID_band,

встановлюючи зв'язок з гуртом.

◦ name_album: Назва альбому.

◦ photo: Обкладинка (фото) альбому.

- Сутність: Genre (Жанр)

◦ ID_genre (PK): Унікальний ідентифікатор жанру (первинний ключ).

◦ name_genre: Назва жанру.

- Сутність: Song (Пісня)

◦ ID_song (PK): Унікальний ідентифікатор пісні (первинний ключ).

◦ ID_genre (FK): Зовнішній ключ, що посилається на Genre.ID_genre, встановлюючи зв'язок з жанром.

◦ ID_album (FK): Зовнішній ключ, що посилається на Album.ID_album, встановлюючи зв'язок з альбомом.

◦ name_song: Назва пісні.

◦ date_create: Дата створення (релізу) пісні.

◦ time_song: Тривалість пісні (у форматі HH:MM:SS).

◦ cd_file: Шлях або ім'я файлу MP3 на диску.

- Сутність: Try_listening (Спроби прослуховування)

◦ ID_try_listening (PK): Унікальний ідентифікатор запису прослуховування (первинний ключ).

◦ ID_song (FK): Зовнішній ключ, що посилається на Song.ID_song, встановлюючи зв'язок з піснею.

◦ percent_listening: Відсоток прослуханого часу пісні.

◦ time_listening: Фактичний час прослуховування пісні.

Зв'язки між таблицями:

- Band до Album:

◦ Кардинальність: Один до багатьох (1:N).

◦ Опис: Один гурт може мати багато альбомів.

◦ Реалізація: Здійснюється через зовнішній ключ Album.ID_band, який

посилається на Band.ID_band.

- Album до Song:
  - Кардинальність: Один до багатьох (1:N).
  - Опис: Один альбом може містити багато пісень.
  - Реалізація: Здійснюється через зовнішній ключ Song.ID_album, який посилається на Album.ID_album.
- Genre до Song:
  - Кардинальність: Один до багатьох (1:N).
  - Опис: Один жанр може бути присвоєний багатьом пісням.
  - Реалізація: Здійснюється через зовнішній ключ Song.ID_genre, який посилається на Genre.ID_genre.
- Song до Try_listening:
  - Кардинальність: Один до багатьох (1:N).
  - Опис: Одна пісня може мати багато записів прослуховування (спроб).
  - Реалізація: Здійснюється через зовнішній ключ Try_listening.ID_song, який посилається на Song.ID_song.

Нормалізація даних

Нормалізація бази даних є критично важливим етапом проєктування, що дозволяє оптимізувати структуру таблиць, зменшити надмірність даних, уникнути аномалій оновлення та підвищити ефективність роботи з даними. Це особливо актуально для баз даних значних розмірів, що можуть містити мільйони або навіть трильйони записів. Більшість компаній дотримуються нормалізації принаймні до Третьої нормальної форми (3NF).

Для відповідності 1NF, база даних повинна задовольняти такі вимоги:

- Відсутність списків: Жодна таблиця не повинна містити в собі елементи, які є списками або повторюваними групами даних. Наприклад, у таблиці Song не має бути вбудованого списку з таблиці Try_listening, оскільки одна пісня може мати багато спроб прослуховування.

- Атомарні та унікальні значення: Усі значення в стовпцях повинні бути атомарними (неподільними) та однозначно інтерпретованими. Назви стовпців мають бути короткими та чіткими, наприклад, cd_file замість cd_file_from_laptop_user.
- Незалежність порядку: Порядок зберігання рядків і стовпців не повинен впливати на інтерпретацію даних.

Таблиця знаходиться у 2NF, якщо вона відповідає 1NF і кожен неключовий атрибут повністю залежить від первинного ключа. Це означає, що, якщо первинний ключ є складеним (складається з кількох атрибутів), жоден неключовий атрибут не повинен залежати лише від частини цього складеного ключа. У нашому випадку кожна таблиця має одиночний первинний ключ (PK), що автоматично задовольняє цю вимогу.

Таблиця знаходиться у 3NF, якщо вона відповідає 2NF і не має транзитивних залежностей. Це означає, що жоден неключовий атрибут не повинен залежати від іншого неключового атрибута. Простіше кажучи, кожен неключовий атрибут повинен прямо залежати від первинного ключа, а не від іншого неключового атрибута.

Існують різні тлумачення 3NF. Деякі джерела [51] надають більш формальні визначення. Однак, у спрощеному вигляді, як це було виявлено на YouTube, це означає, що жоден атрибут у таблиці не може бути виведений або створений за допомогою іншого неключового атрибута в тій же таблиці.

У контексті нашої таблиці Try_listening (ID_try_listening > ID_song, percent_listening, time_listening):

Залежність між percent_listening (відсотком прослуховування) та time_listening (фактичним часом прослуховування) існує, оскільки percent_listening може бути обчислений на основі time_listening та загальної тривалості пісні. Формально, це може бути інтерпретовано як транзитивна залежність, якщо time_listening є неключовим атрибутом, що визначає percent_listening. Однак, для наших цілей, де percent_listening використовується

для реалізації системи тригерів (що буде детальніше розглянуто в пункті "2.3.3 Створення тригерів"), ця залежність не є критичною і може бути прийнятною для підвищення функціональності, не порушуючи загальну структуру бази даних.

Класифікація атрибутів

Атрибути (стовпці) в нашій логічній моделі даних можна класифікувати за їхнім призначенням:

Ідентифікаційні атрибути (Первинні ключі - PK):

- ID_band (PK гурту)
- ID_album (PK альбому)
- ID_genre (PK жанру)
- ID_song (PK пісні)
- ID_try_listening (PK запису прослуховування)

Зв'язкові атрибути (Зовнішні ключі - FK):

- Album.ID_band (FK → Band.ID_band)
- Song.ID_album (FK → Album.ID_album)
- Song.ID_genre (FK → Genre.ID_genre)
- Try_listening.ID_song (FK → Song.ID_song)

Описові атрибути:

- Band.name_band – назва гурту
- Album.name_album – назва альбому
- Genre.name_genre – назва жанру
- Song.name_song – назва пісні

Історичні атрибути:

- Song.date_create – дата створення (релізу) пісні
- Song.time_song – тривалість пісні

Технічні/Метадані:

- Album.photo – обкладинка альбому (зберігається як BLOB, двійкові

дані)

- Song.cd_file – шлях до MP3-файлу на диску
- Try_listening.percent_listening – відсоток прослуханого часу
- Try_listening.time_listening – фактичний час прослуховування

2.2 Вибір системи управління інформаційною базою

Вибір оптимальної системи управління базами даних (СУБД) є ключовим етапом у розробці будь-якого програмного забезпечення. Кожна СУБД має свої унікальні переваги, потужні можливості та сфери оптимального застосування. Деякі краще підходять для веб-додатків з великою кількістю операцій читання, інші — для складної аналітики або геопросторових запитів. Важливо враховувати ліцензійні умови, підтримку спільноти, доступні інструменти адміністрування та кросплатформеність.

PostgreSQL — це потужна об'єктно-реляційна СУБД (ORDBMS), яка поєднує традиційну реляційну модель з можливостями об'єктно-орієнтованого підходу [15]. Вона широко використовується аналітиками даних, фахівцями зі штучного інтелекту, розробниками бекенду та мобільних додатків.

Ключові особливості:

- Кросплатформенність: Чудово працює на Linux, Windows, macOS, BSD, а також у Docker-контейнерах.
- Відкритий код: Доступ до вихідного коду гарантує прозорість, відсутність "слідкування" та можливість спільного вдосконалення безпеки. Будучи безкоштовним продуктом, вона не може бути продана (за винятком преміум-планів для корпоративного використання).
- Повна підтримка ACID-транзакцій: Забезпечує коректність і надійність обробки даних.
- Розширювані типи даних: Підтримує JSONB, масиви, гео-дані, що дозволяє зберігати різноманітні типи інформації, включаючи фотографії.

- Індекси: Має потужні індекси GIN, GiST, BRIN для оптимізації пошуку.
- Розширені функції SQL: Підтримує CTE (WITH), віконні функції, тригери та процедури (детальніше про тригери буде в 2.3.3).
- Адміністрування: Зручне адміністрування через `pgAdmin 4` та `psql`.

MySQL — це СУБД, відома своєю відносною простотою встановлення та налаштування. Вона часто використовується веб-розробниками та DevOps-інженерами. [16]

Функціонал:

- Просте налаштування: Легкість інсталяції.
- Master-slave реплікація: Підтримка реплікації "з коробки".
- Кросплатформенність: Працює на Linux, Windows, macOS.
- Офіційний GUI: Наявність MySQL Workbench.
- Обмежена підтримка нових SQL-стандартів: Деякі розширені функції, як-от CTE та віконні функції, з'явилися пізніше порівняно з іншими СУБД.

Microsoft SQL Server — це СУБД, яка часто використовується в екосистемі Windows, особливо в компаніях, що орієнтуються на рішення від Microsoft [21]. Її перевага — глибока інтеграція з іншими продуктами Windows. Це вибір для .NET-розробників, корпоративних DBA та системних архітекторів.

Функції:

- Повна інтеграція: Глибока інтеграція з .NET та Active Directory.
- Оптимізація для Windows: Коректно працює переважно на Windows.
- Адміністрування: Через SSMS (SQL Server Management Studio) та Azure Data Studio.
- T-SQL розширення: Власна мова розширення SQL.

- Express-версія: Безкоштовна, але з жорсткими обмеженнями.

Oracle Database — це корпоративна база даних, відома своєю високою надійністю та безпекою, часто використовується у великих компаніях, банках та урядових установах [17]. Це вибір для корпоративних DBA, архітекторів систем високої доступності. Хоча вона потужна, її мінусами є висока вартість та складність адміністрування для пересічного користувача.

Функції:

- Широка сумісність: Добре працює на всіх системах: Windows, Linux, Solaris.
- Адміністрування: Через Oracle SQL Developer.
- Oracle RAC: Рішення для кластеризації.
- PL/SQL: Власна процедурна мова для розробки.
- Flashback-функції: Можливість відновлення даних до попередніх станів.
- Data Guard: Функція для аварійного відновлення (DR).
- Висока вартість ліцензій та складність адміністрування.

SQLite — це легковагова, вбудована СУБД, яка не потребує окремого серверного процесу і зберігає дані безпосередньо у файлах [18]. Завдяки низьким вимогам до ресурсів, вона набула значної популярності на мобільних платформах. Основні користувачі: мобільні розробники (Android, iOS), розробники десктопних утиліт, embedded-інженери.

Функціонал:

- Кросплатформенність: Підтримується на Linux, Windows, macOS, iOS, Android.
- Відкритий код: Прозорий доступ до вихідного коду.
- Адміністрування: Через CLI (`sqlite3`) або GUI-інструменти (наприклад, DB Browser for SQLite).
- Безсерверна архітектура: Не вимагає серверної частини.

- Динамічна типізація: Значення зберігаються як TEXT/NUMERIC, що забезпечує гнучкість.

IBM DB2 — комерційна СУБД від IBM, розроблена для великих корпоративних середовищ та архівування великих обсягів даних [19]. Вона оптимізована для великих аналітичних навантажень та інтеграції з AI-запитами. Основні користувачі: enterprise-архітектори даних, аналітики великих даних у великих корпораціях.

Функції:

- Широка сумісність: Працює на Windows, Linux, AIX.
- Адміністрування: Через IBM Data Studio та DB2 Control Center.
- Оптимізація під Big Data та AI: Оптимізована для аналізу великих даних та запитів ШІ.
- Інтеграція: Тісна інтеграція з IBM Watson та Big Data.
- Масштабованість: Підтримка Multi-Instance та pureScale для кластеризації.

MS Access — це СУБД, яка є частиною пакету Microsoft Office [20]. Вона часто використовується як перша база даних для вивчення в освітніх закладах. Її перевагами є зручність та простота у формуванні форм та звітів. Призначена для офісних користувачів та розробників простих бізнес-додатків без глибоких знань СУБД.

Для дипломного проєкту обрано ***PostgreSQL***. Цей вибір обумовлений декількома ключовими факторами:

- Спільність поглядів: PostgreSQL активно використовується спільнотою однодумців, що забезпечує широку підтримку та доступ до ресурсів.
- Об'єктно-реляційність: Завдяки об'єктно-реляційній моделі, PostgreSQL є гнучкою та потужною, що дозволяє легко працювати з різними типами даних.

- Гнучкість налаштування: Простота та швидкість налаштування системи.
- Локальний та мережевий доступ: Можливість використовувати як локально, так і через мережу.

Єдиним, проте не критичним, мінусом є відносно складне встановлення на деяких дистрибутивах Linux (деталі будуть розглянуті в пункті "2.3.1 Короткі деталі встановлення програми бази даних").

2.3 Створення інформаційної бази

Для створення та роботи з базою даних у проєкті необхідно встановити PostgreSQL та pgAdmin. Процес встановлення залежить від дистрибутива Linux.

Наприклад, для Fedora встановлення PostgreSQL відбувається покроково через термінал, згідно з офіційними інструкціями. Після встановлення та активації PostgreSQL (`systemctl enable postgresql.service`, `systemctl start postgresql.service`), для роботи з `pgAdmin` потрібно активувати віртуальне середовище (`source pgadmin4/bin/activate`) та запустити програму (`pgadmin4`). Це дозволить отримати доступ до інтерфейсу `pgAdmin` через веббраузер за адресою `http://127.0.0.1:5050`.

У випадку з Arch Linux, встановлення `PostgreSQL` через пакетні менеджери `pacman` та `yay` було успішним. Однак, при спробі встановлення `pgAdmin4` виникли циклічні помилки, що унеможливило його коректну роботу. Тому було прийнято рішення використовувати онлайн-версію pgAdmin, що забезпечило робоче середовище для взаємодії з базою даних, попри те, що вона тепер не є повністю локальною.

Після успішного встановлення та налаштування `PostgreSQL` та `pgAdmin`, можна приступити до створення бази даних та таблиць згідно з

логічною моделлю.

Для запуску сервера PostgreSQL використовуємо команди:

```
systemctl enable postgresql.service
systemctl start postgresql.service
```

Після чого в терміналі з'явиться повідомлення про запуск `pgAdmin` та адреса для доступу через браузер:

```
Starting pgAdmin 4. Please navigate to http://127.0.0.1:5050 in your browser.
 * Serving Flask app 'pgadmin'
 * Debug mode: off
```

У веб-інтерфейсі `pgAdmin` потрібно буде зареєструвати та додати нову базу даних (рис. 2.2).

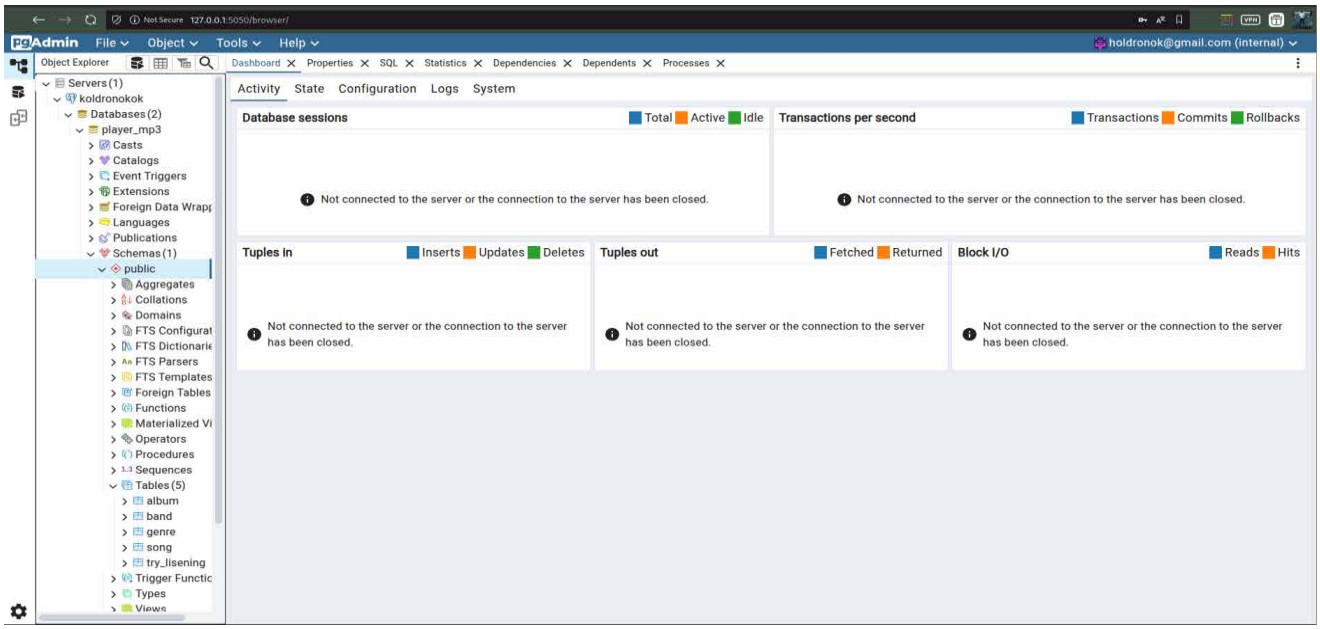


Рис. 2.2 – Інтерфейс web-pgadmin4

Для створення таблиць використовується мова SQL. Створення таблиці `Genre`:

```
SQL
CREATE TABLE Genre(
    ID_genre SERIAL PRIMARY KEY,
```

```
    name_genre VARCHAR(255) NOT NULL
);
```

Весь код для створення таблиць знаходиться у Додатку Б.

Основні типи даних, використані для стовпців:

- Genre:
  - `ID_genre` – `SERIAL` (автоматично зростаюче ціле число)
  - `name_genre` – `VARCHAR(255)` (рядок до 255 символів)
- Band:
  - `ID_band` – `SERIAL`
  - `name_band` – `VARCHAR(255)`
- Album:
  - `ID_album` – `SERIAL`
  - `name_album` – `VARCHAR(255)`
  - `photo` – `BYTEA` (двійкові дані для зберігання зображень)
  - `ID_band` – `SERIAL` (зовнішній ключ)
- Song:
  - `ID_song` – `SERIAL`
  - `name_song` – `VARCHAR(255)`
  - `date_create` – `VARCHAR(4)` (наприклад, для року релізу)
  - `time_song` – `TIME` (тривалість пісні)
  - `cd_file` – `TEXT` (шлях до файлу MP3)
  - `ID_album` – `SERIAL` (зовнішній ключ)
  - `ID_genre` – `SERIAL` (зовнішній ключ)
- Try_listening:
  - `ID_try_listening` – `SERIAL`
  - `percent_listening` – `INTEGER` (ціле число)
  - `time_listening` – `TIME` (фактичний час прослуховування)

- `date_listening` – `DATE` (дата прослуховування)
- `random` – `BOOLEAN` (логічний тип, вказує, чи було прослуховування випадковим)
- `ID_song` – `SERIAL` (зовнішній ключ)

Для візуалізації структури бази даних та зв'язків між елементами також можна використовувати Діаграму класів, яка деталізує атрибути та методи кожного класу, а також взаємозв'язки між ними.

Тригери є потужним інструментом для автоматизації та контролю дій у базі даних, дозволяючи реалізовувати бізнес-логіку та запобігати помилкам. Вони дозволяють автоматизувати рутинні операції, що є критично важливими протягом усього життєвого циклу бази даних. Код для створення тригерів знаходиться у Додатку С.

Тригер 1: Автоматичний розрахунок відсотка прослуховування (`percent_listening`)

Цей тригер автоматизує заповнення поля `percent_listening` у таблиці `Try_listening`, запобігаючи можливим помилкам при ручному або програмному введенні. Використовуючи `song.time_song` (загальна тривалість пісні) та `try_listening.time_listening` (фактичний час прослуховування), тригер автоматично обчислює `percent_listening` при додаванні нових даних. Результат виконання подано на рис. 2.3 - 2.4:

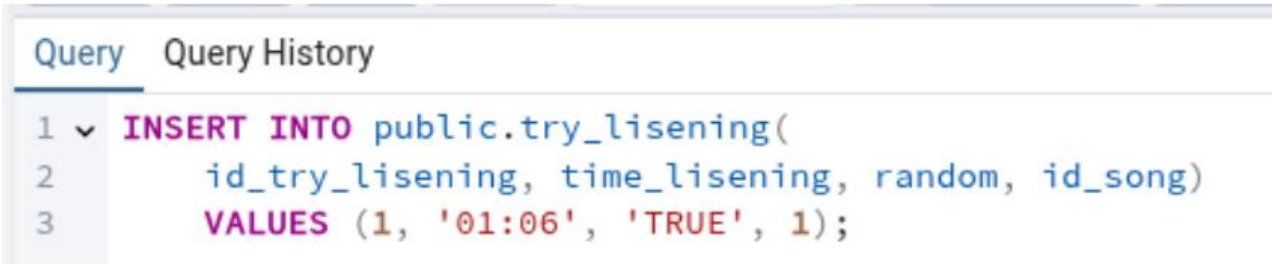


Рис. 2.3 – Створення рядка значень

Рис 2.4 – Перегляд значень

При додаванні запису до таблиці `Try_listening` ми вказуємо `id`, `time_listening`, `random`, `id_song`. Поле `percent_listening` не вказується, оскільки воно заповнюється автоматично тригером. Після додавання запису, при виведенні даних, можна побачити, що `percent_listening` вже розраховано та додано до запису.

Тригер 2: Валідація відсотка прослуховування (`percent_listening`)

Цей тригер забезпечує цілісність даних, запобігаючи додаванню записів, де `percent_listening` перевищує 100%. Кожного разу, коли виконується спроба додати запис до таблиці `Try_listening` з некоректним відсотком (тобто `time_listening` більший за `time_song`, що призводить до `percent_listening` > 100%), тригер генерує помилку, блокуючи операцію.

Тестування дії:

Спочатку визначаємо максимальну тривалість пісні (`time_song`) для конкретного `id_song`:

Рис. 2.5 – Виведення значень рядка

```
SELECT time_song FROM Song WHERE ID_song = [ваш_id_пісні];
```

Потім, спробуємо додати хибні дані до `Try_listening`, де `time_listening` перевищує `time_song`, що спричинить розрахунок `percent_listening` більше 100%.

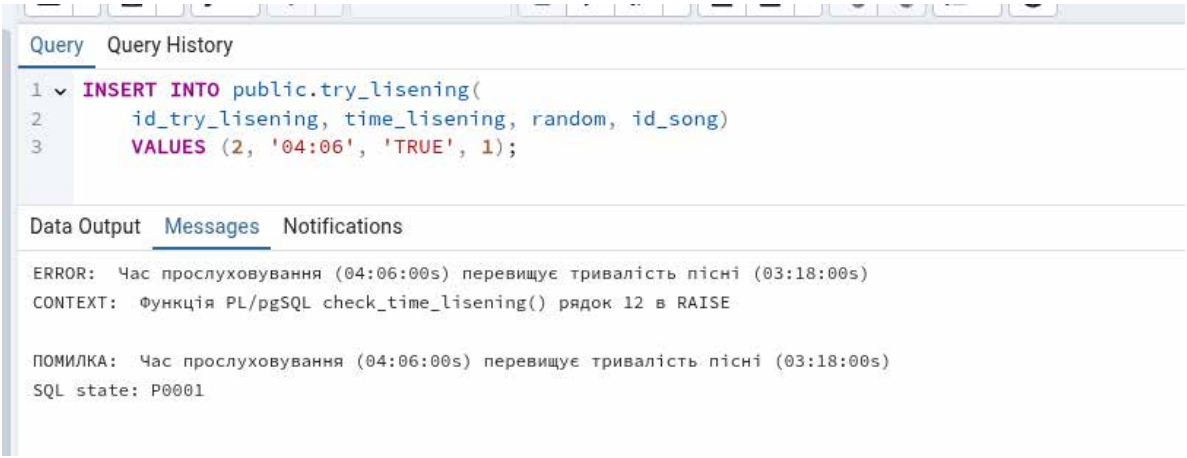


Рис. 2.6 – Виведена помилка при створенні даних

```
INSERT INTO Try_listening (ID_song, time_listening, random, date_listening)
VALUES (1, '00:05:00', FALSE, '2025-05-25'); -- Припустимо, Song з ID_song=1 має тривалість 3 хвилини
```

Виконання такого скрипта призведе до помилки (рис. 2.6), оскільки тригер виявить некоректні дані та заблокує їх додавання до бази даних.

Повний код тригерів подано в додатку В. Хоча база даних для програмного додатку відносно невелика, закладені основи дозволять їй ефективно масштабуватися разом з подальшим розвитком програми.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Організаційна структура програмного забезпечення

У підрозділі 1.1 визначалися попередні вимоги до архітектури додатку. У даному розділі розглянуто архітектурні рішення, пов'язані з вибором операційної системи, а саме Linux, як основної платформи для розробки.

Linux — це ядро операційної системи, тоді як дистрибутив — це повноцінна операційна система, побудована на цьому ядрі. Протягом року проводилося тестування багатьох дистрибутивів Linux для забезпечення оптимальної роботи нашої програми.

Кожен дистрибутив Linux складається з таких компонентів:

- Linux Kernel (Ядро Linux): Центральна частина операційної системи, яка безпосередньо взаємодіє з апаратним забезпеченням комп'ютера. Без нього функціонування системи неможливе.
- Робоче середовище: Відповідає за графічний інтерфейс користувача. Приклади: GNOME, KDE Plasma, XFCE, Cinnamon, Hyprland. Цей елемент є ключовим для нашої програми, оскільки зміна робочого середовища може вплинути на відображення та поведінку деяких елементів програми.
- Утиліти: Невеликі службові програми для допоміжних або адміністративних функцій (налаштування системи, моніторинг процесів, файловий менеджер тощо). Важливо враховувати можливі відмінності у відкритті файлового менеджера через нашу програму порівняно з прямим викликом.
- Пакети: Програми встановлюються через систему управління пакетами (наприклад, APT у Debian/Ubuntu, DNF у Fedora, Pacman у Arch). Ці системи забезпечують доступ до перевіреного програмного забезпечення, а також дозволяють встановлювати програми від незалежних розробників.

- Драйвери: Програмні модулі, що забезпечують взаємодію операційної системи з апаратними компонентами (вебкамера, графічна карта, Wi-Fi, Bluetooth). Хоча проблеми з драйверами можуть виникати на будь-якій платформі, у Linux вони є частим явищем. Однак, правильне встановлення відповідних драйверів зазвичай вирішує ці проблеми.

Сучасні робочі середовища Linux, такі як GNOME, KDE Plasma та Hyprland, активно розвиваються в напрямку автоматизації та кастомізації. На відміну від Windows, де користувачі здебільшого покладаються на мишку, ці середовища надають розширені можливості для керування системою за допомогою гарячих клавіш та автоматизованих процесів.

- ***GNOME*:** Відомий своєю інтеграцією пошукової функції. Користувач може просто почати писати, і система одразу шукатиме елементи. Натискання клавіші SUPER (Win) відкриває доступ до пошуку будь-якого елемента, від файлу музики до програми. [22]
- ***Hyprland*:** Орієнтований на повну автоматизацію оболонки, зокрема, автоматичне розміщення вікон для оптимального використання простору. [24]
- ***KDE Plasma*:** Пропонує простий перехід для користувачів Windows завдяки звичному інтерфейсу, але водночас надає широкі можливості для глибокого налаштування. [23]

Ці особливості підкреслюють, що Linux-системи найкраще підходять для автоматизованих програм та інтеграції нових технологій автоматизації, що є однією з ключових переваг для нашого проєкту.

Особливості файлової системи Linux

Для розуміння структури програмного забезпечення в середовищі Linux важливо знати організацію файлової системи. В таблиці 3.1 наведено опис стандартних каталогів Linux та їхнє значення для розробки та розгортання MP3-плеєра. Ця інформація ґрунтується на загальноприйнятих стандартах [3].

Таблиця 3.1

Файлова система Linux

Каталог	Опис
/	Кореневий каталог, від якого відгалужуються всі інші теки.
/bin	Основні виконувані програми, доступні для всіх користувачів.
/sbin	Системні виконувані файли, призначені для адміністрування системи.
/usr	Додаткові програми та бібліотеки. Містить підтеки:
/usr/bin	Більшість користувацьких програм.
/usr/sbin	Адміністраторські інструменти.
/usr/lib	Бібліотеки.
/etc	Глобальні конфігураційні файли системи та сервісів.
/var	Змінні дані, які змінюються під час роботи системи. Містить:
/var/log	Журнали (логи).
/var/cache	Кеш пакетного менеджера.
/var/spool	Черги пошти/друку.
/home	Домашні теки користувачів (наприклад, /home/username).
/tmp	Тимчасові файли, які зазвичай очищаються при перезавантаженні системи.
/dev	Спеціальні файли пристроїв (драйвери в юзерспейсі), наприклад /dev/sda.
/proc	Віртуальна файлова система з інформацією про процеси та ядро.
/sys	Віртуальна файлова система для взаємодії з ядром через sysfs.
/mnt	Точка монтування для тимчасових файлових систем (ручне монтування).
/media	Автоматичні точки монтування зовнішніх носіїв (USB, CD/DVD).
/opt	Додаткове програмне забезпечення, встановлене вручну (сторонні пакунки).
/lib64	Бібліотеки, необхідні для завантаження системи та виконання програм з /bin і /sbin на 64-бітних системах.

При розробці та розгортанні власного MP3-плеєра в Linux необхідно звернути особливу увагу на наступні теки, які є ключовими для функціонування програми:

Таблиця 3.2

Файлова система програмного додатку

Призначення	Каталог	Опис
Джерела аудіофайлів	~/Music/	Стандартне розташування для музичних файлів (MP3, OGG, FLAC).
Кеш та тимчасові дані	/tmp/foronok-koldronokok/	Кеш обробки звуку, що зберігається протягом поточного сеансу роботи програми.
Налаштування гарячих клавіш	~/.config/hypr/conf/keybindings/default/	Розташування для створених системних скорочень (наприклад, для Hyprland).
	~/.config/foronok/keybindings.conf	Файл конфігурації гарячих клавіш для самої програми.
Системні бібліотеки та плагіни	/usr/lib/gstreamer-1.0/	Встановлені плагіни GStreamer для аудіо/відео відтворення.
	/usr/lib64/gstreamer-1.0/	Аналогічно, для 64-бітних дистрибутивів.
	/etc/gstreamer-1.0/	Глобальні системні налаштування GStreamer.

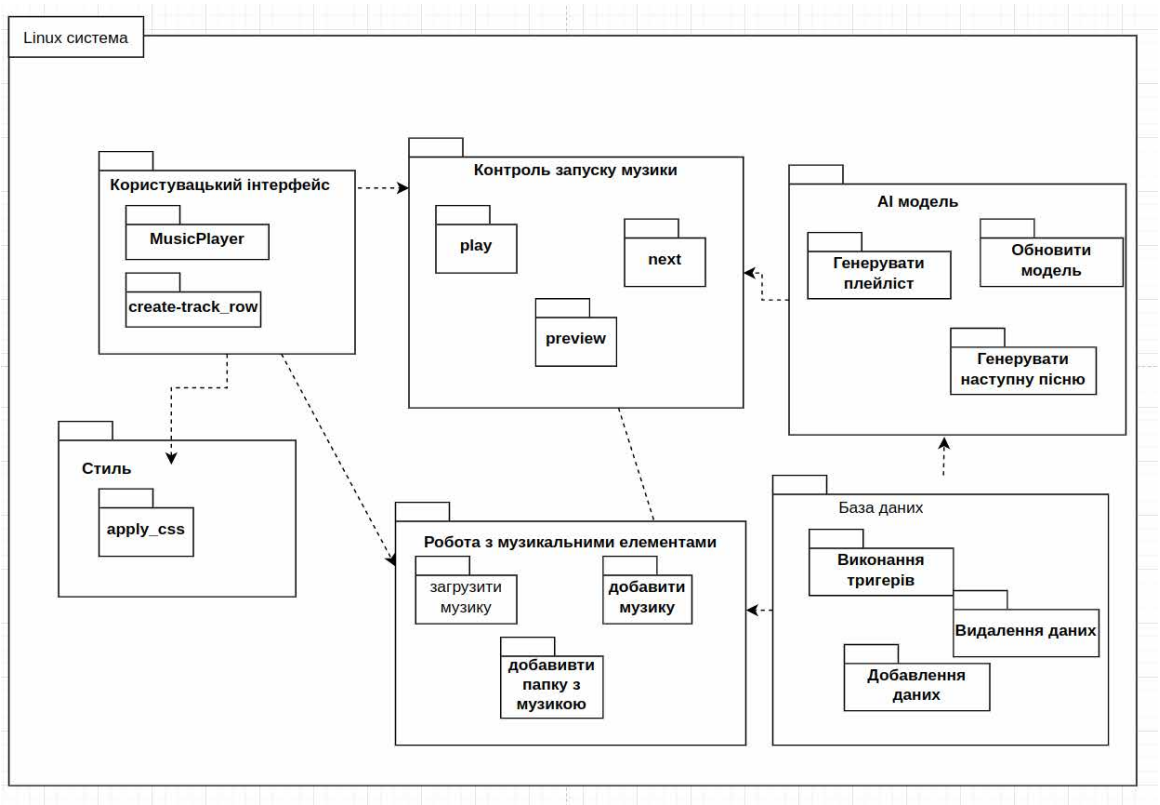


Рис. 3.1 – Діаграма пакетів

На рисунку 3.1 подано діаграму пакетів - UML-діаграму, яка візуалізує архітектуру програмної системи, представляючи її як набір логічно пов'язаних пакетів (модулів) та їхні залежності. Вона допомагає краще зрозуміти структуру проєкту, розподіл відповідальності між компонентами та потік використання даних і сервісів.

Опис елементів діаграми пакетів

Усі пакети взаємодіють у межах пакета "Linux система", що вказує на їхнє функціонування в середовищі операційної системи Linux.

1 Пакет "Користувацький інтерфейс":

- MusicPlayer: Відповідає за основні функції відтворення та паузи, а також відображення інформації про поточний трек.
- create-track_row: Генерує елементи списку треків, включаючи назву, час та кнопки керування.

2 Пакет "Стиль":

- apply_css: Застосовує візуальне оформлення інтерфейсу, включаючи кольори, шрифти та розташування елементів.

3 Пакет "Контроль запуску музики":

- play: Запускає відтворення поточного треку.
- next: Переходить до наступного треку в черзі відтворення.
- preview: Здійснює попередній перегляд початку треку або його уривка.

4 Пакет "Робота з музикальними елементами":

- Завантажити музику: Сканує файлову систему та індексує аудіофайли.
- Додати музику: Додає окремий файл до локальної бібліотеки програми.
- Додати папку з музикою: Рекурсивно імпортує всі треки з обраної теки.

5 Пакет "AI модель":

- Генерувати плейлист: Формує набір треків на основі історії прослуховувань користувача.
- Генерувати наступну пісню: Рекомендує наступний трек під час відтворення.
- Оновити модель: Здійснює тренування алгоритму на нових даних прослуховувань.

6 Пакет "База даних":

- Виконання тригерів: Забезпечує автоматизацію та контроль даних, як описано в розділі "2.3.3 Створення тригерів".
- Видалення даних: Відповідає за видалення даних (зокрема, музичних треків) з бази даних.
- Додавання даних: Додає нові дані (наприклад, імпортовану музику) до бази даних MP3-плеєра.

Взаємодія пакетів:

- Користувацький інтерфейс — це основна точка взаємодії з користувачем, що відображає дані та дозволяє керувати програмою.
- AI модель — відповідає за інтелектуальну генерацію рекомендацій та плейлістів.
- Контроль запуску музики — керує логікою відтворення треків.
- Робота з музикальними елементами — забезпечує функції завантаження та управління аудіофайлами.
- Стиль — відповідає за візуальне оформлення та привабливість інтерфейсу.
- База даних — слугує центральним сховищем, звідки програма отримує та куди зберігає дані, включаючи додавання та видалення музики з плейліста.

3.2 Вибір інструментарію для створення ППЗ

C++ — це потужна компіляційна мова програмування, яка ефективно поєднує можливості процедурного та об'єктно-орієнтованого підходів [25]**.** Вона є незамінною там, де ключовими є висока швидкодія, точний контроль над системними ресурсами та мінімізація накладних витрат програми. C++ широко застосовується розробниками ігор, системного та вбудованого програмного забезпечення, а також фахівцями, що створюють високопродуктивні додатки. Ця мова особливо популярна в Linux-системах.

Ключові особливості:

- Висока продуктивність: Забезпечує надзвичайну швидкість виконання завдань, що робить її ідеальною для вимогливих до ресурсів програм.
- Прямий контроль над пам'яттю: Дозволяє розробникам детально керувати розподілом і використанням пам'яті, що є важливим для оптимізації та безпеки (звідси її популярність серед фахівців з кібербезпеки).
- Універсальність: Підходить для створення широкого спектра додатків, включаючи драйвери, ігри та мультимедійні програми.
- Багатий набір бібліотек: Має велику кількість потужних бібліотек, таких як QT, SDL, PortAudio, які значно спрощують розробку.
- Кросплатформенність: Підтримується на більшості операційних систем, включаючи Linux, Windows та macOS.

Python — це мова з динамічною типізацією, відома своєю лаконічністю, високим рівнем абстракції та широкими можливостями [26]**.** Вона особливо популярна в галузях аналітики даних, машинного навчання та швидкого прототипування. Основними користувачами Python є Data Scientists, AI/ML інженери, аналітики даних, веб-розробники та фахівці з автоматизації. Зокрема, на Linux-середовищах, таких як Hyprland, багато функцій автоматизації

реалізовані саме за допомогою Python.

Ключові особливості:

- Простий та читабельний код: Завдяки синтаксису, що нагадує природну мову, Python є легким для вивчення та розуміння.
- Потужні бібліотеки для ШІ/ML: Має величезну екосистему бібліотек, таких як TensorFlow, PyTorch, scikit-learn, що робить її лідером у сфері штучного інтелекту.
- Швидке прототипування: Дозволяє швидко створювати робочі прототипи ідей завдяки своїй гнучкості.
- Вбудовані засоби: Надає зручні інструменти для роботи з API, файлами та базами даних.
- Кросплатформенність: Функціонує на Linux, Windows та macOS.

Java — це багатоплатформова мова програмування, яка виконується у віртуальному середовищі JVM (Java Virtual Machine), що забезпечує високу портативність ("напиши один раз — запускай всюди") **[28]**. Вона широко застосовується у великих корпоративних інформаційних системах. Хоча Java також використовується для роботи зі штучним інтелектом, вона не настільки потужна в цій галузі, як Python. Типові користувачі: корпоративні розробники, розробники Android-додатків, фахівці з Enterprise backend та розробники великих інформаційних систем.

Ключові особливості:

- Висока портативність: Можливість запускати один і той самий код на різних платформах без змін.
- Добра підтримка багатопоточності: Ефективно працює з паралельними обчисленнями.
- Бібліотеки для GUI: Має вбудовані бібліотеки для розробки графічних інтерфейсів, такі як Swing та JavaFX.
- Взаємодія з базами даних: Забезпечує легку взаємодію з різними

базами даних через JDBC.

- Широка сумісність: Підтримується на Linux, Windows, Android та macOS.

C# — це сучасна мова програмування, що є частиною екосистеми .NET**.** Вона глибоко інтегрована з операційною системою Windows, що робить її дуже зручною для розробки десктопних додатків, зокрема музичних програвачів. Типові користувачі: десктопні .NET-розробники, розробники бізнес-додатків, розробники ігор на Unity та програмісти, орієнтовані на платформу Microsoft. [27]

Ключові особливості:

- Зручний інструментарій: Чудово інтегрується з Visual Studio, надаючи потужний набір інструментів для розробки.
- Бібліотеки для GUI: Має власні бібліотеки для розробки графічних інтерфейсів, такі як WinForms та WPF.
- Інтеграція з базами даних: Добра інтеграція з базами даних через Entity Framework.
- Кросплатформенність: Можливість створювати кросплатформенні застосунки завдяки .NET Core / MAUI, що дозволяє запускати їх на Windows, Linux та macOS.

Вибір мови програмування для створення програмного продукту

На етапі вибору мови програмування для основного програмного продукту розглядалися різні варіанти. Попри переваги C# і C++, дані мови більш орієнтований на Windows.

Під час аналізу існуючих програм у Linux, було приділено увагу на ефективність поєднання Python та бібліотеки GTK. Це дозволило створити просту, зрозумілу та візуально привабливу програму. Дослідження показали, що такий підхід добре реалізований для Linux-систем.

Проведені тести на двох різних дистрибутивах Linux — Fedora (з робочим

середовищем GNOME) та Arch Linux (з робочим середовищем Hyprland) — показали приємне здивування щодо портативності програми. Візуально програма адаптується до особливостей середовища: у GNOME верхня частина програми має стандартні елементи управління вікном, тоді як у Hyprland ці елементи відсутні, що відповідає його філософії тайлінгового віконного менеджера. Крім того, файловий менеджер, який використовується при додаванні нових елементів, також відрізняється залежно від робочого середовища. Варто зазначити, що деякі візуальні елементи, такі як колір, можуть незначно відрізнятися під час розробки та тестування.

Особливості роботи з бібліотеками в Python

Важливо розуміти, що не всі бібліотеки можна вільно використовувати для комерційної розробки програмного забезпечення. Деякі бібліотеки мають ліцензії, які вимагають або придбання ліцензії, або сплати відсотка від доходу, або відкриття вихідного коду вашого проєкту. Оскільки ця програма не призначена для заробітку грошей і має відкритий код, це питання мене не турбує.

У проєкті використовуються такі вбудовані бібліотеки Python: os, time, subprocess, re, datetime, timedelta. Ці модулі дозволяють працювати з операційною системою, регулювати регулярні вирази, керувати процесами, а також маніпулювати часом та датами. Ці елементи зазвичай встановлені разом із Python і доступні без додаткових дій. Вбудовані модулі Python поширюються під Python Software Foundation License, що дозволяє їх вільне використання без обмежень у будь-яких системах.

Зовнішні бібліотеки Python

Для розширення функціоналу, зокрема для роботи з графічним інтерфейсом, графікою, мультимедіа та системними подіями, використовуються такі зовнішні бібліотеки через GObject Introspection: gi, Gtk, Gdk, GdkPixbuf, Gst, GLib, Pango. Ці бібліотеки не встановлюються через pip, а інсталюються безпосередньо через систему Linux, що забезпечує їх коректну

інтеграцію.

Описані бібліотеки поширюються під ліцензією LGPL 2.1+ (Lesser General Public License). Це означає, що їх можна використовувати у відкритих та комерційних проєктах. Однак, якщо ви модифікуєте саму бібліотеку GTK, то ці модифікації повинні бути також відкриті. Просте використання бібліотеки для створення медіаплеєра не вважається модифікацією.

- ***psycopg2*** – це популярна бібліотека для роботи з базою даних PostgreSQL у середовищі Python [29]. У дипломному проєкті вона відповідає за підключення та взаємодію з PostgreSQL. Ліцензія: LGPL 3: Дозволяє вільне використання в будь-яких проєктах, включаючи комерційні.
- ***Pandas*** – це широко використовувана бібліотека для роботи з великими обсягами даних [30]**.** У додатку вона застосовується для генерації звітів та читання даних з бази даних. Ліцензія: BSD 3-Clause дозволяє використання без обмежень у комерційних та відкритих проєктах.
- ***mutagen*** – бібліотека, призначена для читання та редагування метаданих аудіофайлів (MP3, FLAC, AAC, OGG тощо) [31] . Ліцензія: GPL v2: дозволено використовувати у відкритих некомерційних проєктах, за умови, що весь проєкт також має відкритий код. Недопустимо: Використання у комерційних проєктах або проєктах із закритим вихідним кодом без придбання відповідної ліцензії.
- ***spotipy*** – це Python-клієнт для API Spotify. Він дозволяє отримувати інформацію про треки, плейлісти та акаунт користувача [32]**.** Ця бібліотека була використана для тренування моєї моделі на прослуханих треках. Ліцензія: MIT.
- ***joblib*** – бібліотека для збереження/завантаження об'єктів та кешування результатів обчислень [33]. Використано для збереження моделі ШІ у вигляді єдиного файлу. Ліцензія: BSD.
- ***scikit-learn*** – потужна бібліотека машинного навчання, яка чудово працює з ШІ [34]. Зокрема, модуль sklearn.neighbors. NearestNeighbors дозволяє

створювати ефективні моделі рекомендаційних систем. Ліцензія: BSD.

Підсумовуючи вибір бібліотек для дипломного проєкту, можна зробити висновок про переваги створення додатку з відкритим кодом, без придбання ліцензій на окремі з них. Зокрема, розробник mutagen отримує дохід від комерційного використання своєї бібліотеки. Тому, при розробці комерційних проєктів, надзвичайно важливо уважно вивчати ліцензії всіх використовуваних бібліотек, щоб уникнути юридичних проблем.

3.3. Алгоритмізація та програмування програмних модулів

Опис графічного алгоритму GTK в середовищі python.

Для створення графічного інтерфейсу створюваного MP3-плеєра в середовищі Python використовується бібліотека GTK. Одним з ключових елементів GTK для компонування віджетів є Gtk.Box.

Gtk.Box — це контейнер, який дозволяє впорядковувати віджети (елементи інтерфейсу) в одному рядку або стовпці. Його поведінка залежить від параметра orientation:

- Gtk.Orientation.VERTICAL: Віджети розташовуються вертикально, один під одним.
- Gtk.Orientation.HORIZONTAL: Віджети розташовуються горизонтально, один поруч з іншим.

Додатково, можна задати spacing — це кількість пікселів між сусідніми віджетами всередині контейнера.

main_box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=10)

Наведений програмний код створює головний контейнер main_box, який буде розташовувати всі вкладені елементи вертикально, з відступом у 10 пікселів між ними (рис. 3.2).

Рис. 3.2 – Програмний інтерфейс

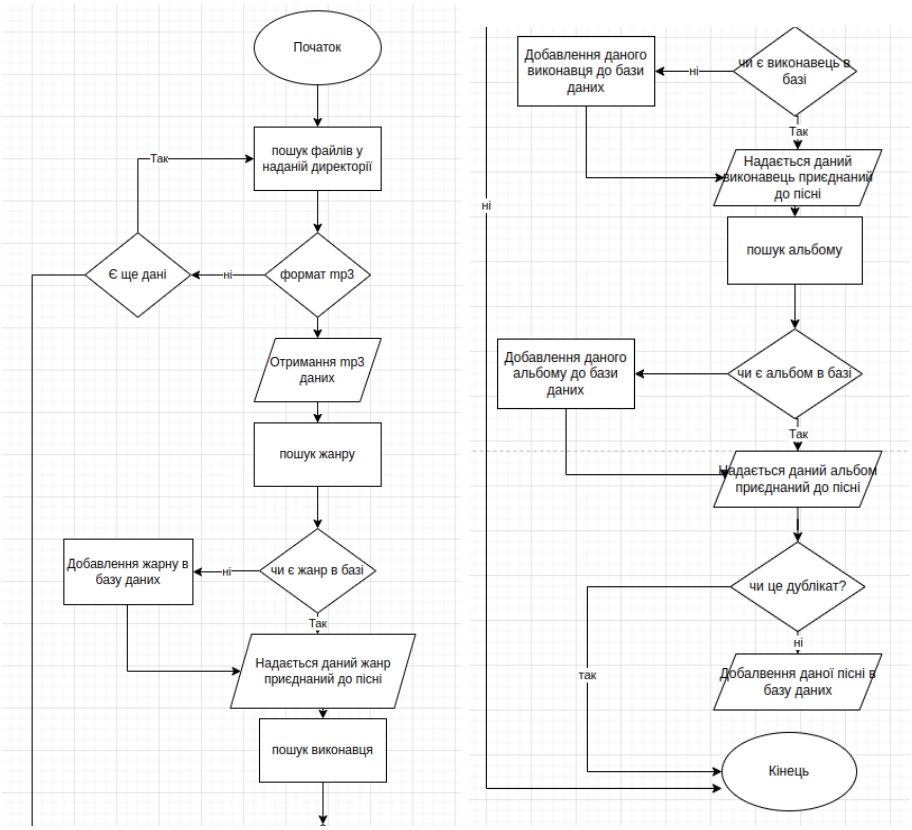


Рис. 3.3 – Блок-діаграма по пошуковій системі

Наведена блок-діаграма (рис. 3.3) дозволяє описати код, або подібні елементи, для кращого розуміння дії за допомогою простих елементів.

Алгоритм виконання програми, згідно даної діаграми:

1 крок — Початок

- Починається виконання програми.

2 крок — Пошук файлів у директорії

- Програма знаходить усі файли у заданій папці (директорії).

3 крок — Чи є ще файли?

- Так — перейти до наступного кроку (4).
-
- Ні — перейти до кінця (завершення програми).
- Якщо так — повертається на 4 крок.

4 крок — Чи має файл формат mp3?

- Так — перейти до 5 кроку.
- Ні — повертається на 2 крок (обробляти наступний файл).

5 крок — Отримання тегів з даних

- Програма витягує з mp3-файлу: жанр, виконавця, альбом, назву пісні

6 крок — Пошук жанру

- Перевірка: чи є жанр у базі даних?

7 крок — Чи є жанр у базі?

- Так — перейти до 9 кроку. Діаграма 3.5 Блок-схема
- Ні — перейти до 8 кроку.

8 крок — Додавання жанру у базу даних

- Додаємо новий жанр у базу.
- Потім перейти до 9 кроку.

9 крок — Прив’язка жанру до пісні

- Зв’язуємо жанр із піснею (наприклад, через зовнішній ключ або зв’язувальну таблицю).

10 крок — Пошук виконавця

- Перевірка: чи є виконавець у базі?

11 крок — Чи є виконавець у базі?

- Так — перейти до 13 кроку. Ні — перейти до 12 кроку.

12 крок — Додавання виконавця у базу даних

- Додаємо нового виконавця.
- Потім перейти до 13 кроку.

13 крок — Прив’язка виконавця до пісні

- Зв’язуємо виконавця з піснею.

14 крок — Пошук альбому

- Перевірка: чи є альбом у базі?

15 крок — Чи є альбом у базі?

- Так — перейти до 17 кроку.
- Ні — перейти до 16 кроку.

16 крок — Додавання альбому у базу даних

- Додаємо новий альбом у базу.

17 крок — Прив’язка альбому до пісні

- Зв’язуємо альбом із піснею.

18 крок — Чи є дублікати пісні?

- Перевіряємо, чи така пісня вже існує в базі.

19 крок — Додавання пісні

- Якщо пісня нова, додаємо її у базу даних.
- Повернення до 2 кроку — обробляється наступний файл.

Кінець

Додавання даних

1. Ініціалізація бази даних у програмному середовищі

Для взаємодії з базою даних у програмі необхідно ініціалізувати підключення.

```python
def connect_to_db():  8 usages  bontonent
    try:
        return psycopg2.connect(
            dbname="player_mp3",
            user="postgres",
            password="koldronokok",
            host="localhost",
            port="5432"
        )
    except Exception as e:
        print(f"Error connecting to database: {e}")
        return None
```

Рис.3.4 –Підключення до бази

2 Ініціалізація кнопки та її функціонал

Для додавання файлів до бібліотеки програми використовується кнопка. Її натискання викликає відповідну функцію.

3 Функція вибору папки та обробка MP3-файлів

У функції on_add_folder_clicked користувачу надається можливість обрати папку (рис. 3.5). Після вибору, програма сканує цю папку, знаходить усі файли з розширенням .mp3, обробляє їх (витягує метаінформацію за допомогою відповідної бібліотеки) і додає ці дані до бази даних.

```python
def on_add_folder_clicked(self, btn):  1 usage  bontonent
    dlg = Gtk.FileChooserDialog(
        title="Оберіть папку",
        parent=self,
        action=Gtk.FileChooserAction.SELECT_FOLDER,
    )
    dlg.add_buttons(
        Gtk.STOCK_CANCEL, Gtk.ResponseType.CANCEL,
        Gtk.STOCK_OPEN, Gtk.ResponseType.OK,
    )
    if dlg.run() == Gtk.ResponseType.OK:
        self.load_music_from_folder(dlg.get_filename())
    dlg.destroy()
```

Рис. 3.5 – Операція вибору папки.

Логіка додавання файлів:

1. Користувач натискає кнопку "Додати папку з музикою".
2. Відкривається діалог вибору папки.
3. Після вибору папки, програма отримує шлях до неї.
4. Програма ітерує по всіх файлах у вибраній папці та її підпапках.
5. Для кожного `.mp3` файлу використовується бібліотека `mutagen` для вилучення таких метаданих, як назва пісні, виконавець, альбом, тривалість.
6. Отримана інформація зберігається в базі даних за допомогою `psycopg2`, відповідно до визначеної ER-діаграми.

Весь код, що стосується додавання даних, знаходиться у Додатку В

Алгоритм роботи ШІ

Основна мета алгоритму ШІ — створити та оновлювати конфігураційний файл config.json, який містить статистику прослуховувань треків користувача, а також навчити модель, що прогнозує ймовірність повного прослуховування пісні. Для цього використовується підключення до Spotify API.

1. Підключення до Spotify API

Для коректної роботи алгоритму ШІ необхідно отримати дані для навчання моделі. Це досягається шляхом підключення до Spotify API (рис. 3.6).

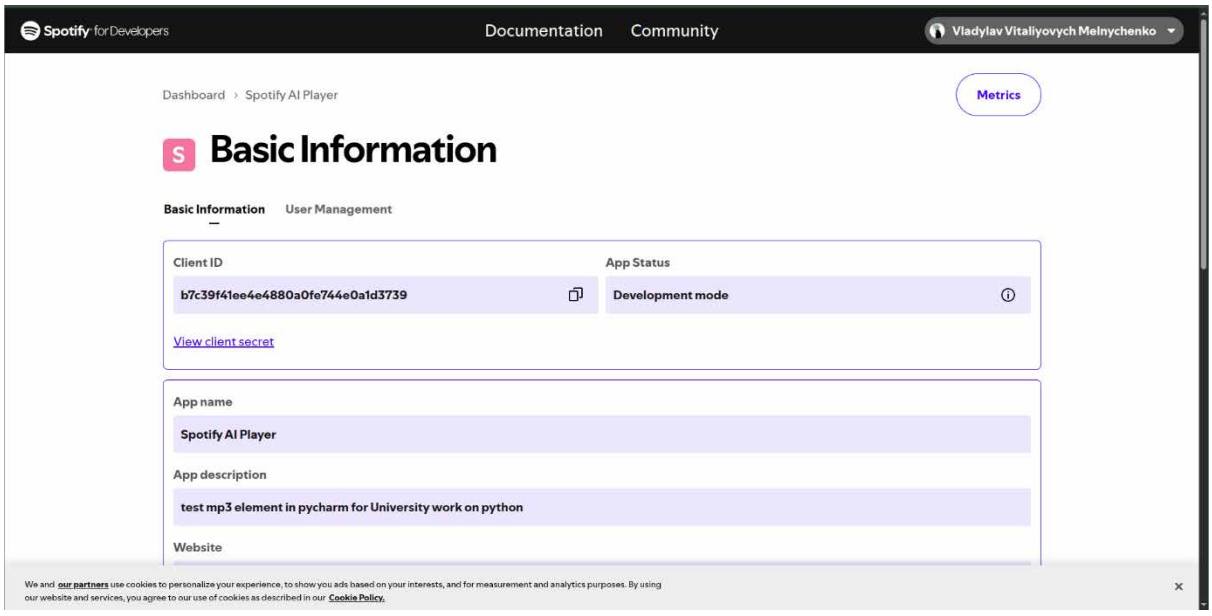


Рис. 3.6 – Spotify developer сайт

Основна мета алгоритму ШІ — створити та оновлювати конфігураційний файл `config.json`, який містить статистику прослуховувань треків користувача, а також навчити модель, що прогнозує ймовірність повного прослуховування пісні. Для цього використовується підключення до Spotify API.

За допомогою .py коду ствоюється підключення:

sp = spotipy.Spotify(auth_manager=SpotifyOAuth(...))

Цей блок використовує бібліотеку spotipy для аутентифікації користувача через OAuth2 і отримання доступу до його прослуханих треків. Параметри задаються в ньому:

client_id, client_secret — API ключі(вказані у вікні)

redirect_uri — куди повертається відповідь після авторизації

scope — доступ до останніх прослуховувань та збережених треків.

2. Завантаження останніх прослуханих треків

Функція Workspace_recent_tracks(limit=50) відповідає за отримання до 50 останніх треків, які користувач слухав. Для кожного з цих треків збирається детальна інформація, включаючи:

- Назву треку.
- Ім'я виконавця.
- Назву альбому.
- Дату релізу.
- Тривалість у мілісекундах.
- Фактичний час прослуховування.
- Обчислений percent_played: Відсоток пісні, яка була прослухана користувачем.

3. Робота з JSON (config.json)

Для зберігання та управління статистикою прослуховувань використовується файл config.json.

Функція load_config(): Зчитує наявну статистику прослуховувань пісень з файлу config.json.

Функція save_config(): Зберігає оновлений словник даних назад у файл.

Структура config.json виглядає приблизно так:

```
{
  "Назва Пісні Виконавець": {
    "count": 5,          // Кількість прослуховувань
    "duration_avg": 210000, // Середня тривалість прослуховування (мс)
    "percent_played_avg": 96.7, // Середній відсоток прослуховування
    "played_timestamps": [...] // Часові мітки прослуховувань
    // ...інші статистичні дані
  }
}
```

4. Оновлення статистики для моделі

Функція update_training_data(df, config_data) бере нові дані про прослуховування (df) та оновлює інформацію у config_data. Ця функція виконує такі ключові дії:

Підраховує count: Загальну кількість прослуховувань кожного треку.

Обчислює duration_avg: Середню тривалість прослуховування треку.

Обчислює percent_played_avg: Середній відсоток відтворення пісні.

5. Навчання моделі прогнозу прослуховування

Модель LinearRegression навчається на основі зібраних даних. Її мета — передбачити percent_played_avg для нових пісень, тим самим оцінюючи, наскільки ймовірно користувач прослухає ту чи іншу композицію повністю.

Вхідні дані (X): Включають кількість прослуховувань та середню тривалість треку.

Вихідні дані (y): Є середнім відсотком прослуховування.

Після навчання модель зберігається у файл (наприклад, listening_model.pkl) за допомогою бібліотеки joblib для подальшого

використання.

6. Основна логіка та використання в MP3-плеєрі

Функція main() координує весь процес: вона завантажує конфігурацію, отримує нові треки, оновлює config.json та навчає/оновлює модель ШІ.

У MP3-плеєрі цей ШІ-алгоритм є основою для системи рекомендацій:

config.json виступає як база знань про музичні вподобання користувача.

Поле percent_played_avg є ключовим показником, що вказує на ймовірність того, що користувач повністю прослухає трек.

При запуску плеєра або генерації списку відтворення, алгоритм обиратиме пісні з найвищим percent_played_avg. Це забезпечує максимально приємне прослуховування, оскільки логіка вибору пісні базується не лише на загальних вподобаннях (жанр, рейтинг), а й на реальному досвіді прослуховування, зафіксованому у config.json.

Наприклад, функція recommend_top_tracks(config, n=10) може відсортувати треки за percent_played_avg і повернути топ-N рекомендованих пісень. Результатом є інтелектуальна система вибору музики, яка адаптується до слухача на основі його реальних дій.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування є критично важливим етапом у розробці програмного забезпечення, оскільки воно забезпечує правильність роботи програми. Проте, ручне тестування не може виявити всі можливі помилки, тому часто використовуються автоматизовані тести. Важливо розуміти, що тестування — це не одноразова дія, а безперервний процес, що триває протягом усього життєвого циклу продукту. Під час розробки цього продукту було проведено численні тести, і кожен наступний день приносив нові ідеї для перевірок.

Типи тестувань можна класифікувати за їхньою метою та підходом до виконання. Цей розділ ґрунтується на загальноприйнятих методологіях тестування (додаткову інформацію можна знайти в онлайн-ресурсах).

Функціональне тестування: Цей тип тестування спрямований на перевірку відповідності програми встановленим вимогам і очікуваній поведінці її функцій.

- Unit-тестування (Модульне тестування): Перевіряє коректність роботи окремих, найменших компонентів або модулів програми (наприклад, окремої функції чи класу).
- Інтеграційне тестування: Перевіряє взаємодію між різними модулями або компонентами програми, щоб переконатися, що вони працюють разом правильно.
- Тестування прийняття (Приймальне тестування): Оцінює готовність системи до впровадження та використання кінцевими користувачами. Перевіряє, чи відповідає програма бізнес-вимогам і чи задовольняє потреби замовника.

Нефункціональне тестування. Цей вид тестування оцінює нефункціональні аспекти програмного забезпечення, які впливають на його якість, такі як продуктивність, безпека, зручність використання та сумісність.

- Тестування продуктивності: Перевіряє швидкодію та стабільність системи під різними навантаженнями (наприклад, час відгуку, використання ресурсів).
- Тестування безпеки: Оцінює захищеність системи від несанкціонованого доступу, атак та інших загроз безпеці даних.
- Тестування зручності використання (Usability testing): Оцінює, наскільки інтерфейс користувача зручний, інтуїтивно зрозумілий та легкий у використанні. Це включає тестування програми як з мишкою, так і без неї (наприклад, за допомогою гарячих клавіш).
- Тестування сумісності: Перевіряє коректну роботу програми на різних апаратних і програмних конфігураціях (наприклад, на різних операційних системах, версіях бібліотек).

3. Методи тестування:

Це підходи, які використовуються для виконання тестів.

- Ручне тестування: Виконується тестувальником вручну, без використання автоматизованих інструментів. Часто застосовується для перевірки користувацького інтерфейсу та зручності використання.
- Автоматизоване тестування: Використання спеціальних програмних інструментів для автоматичного виконання тестів та перевірки результатів. Це дозволяє швидше виявляти регресійні помилки.
- Тестування чорної скриньки (Black-box testing): Тестування проводиться без доступу до внутрішньої структури, коду або реалізації програми. Тестувальник взаємодіє з програмою через її інтерфейс, перевіряючи входи та виходи.
- Тестування білої скриньки (White-box testing): Тестування

проводиться з повним доступом до внутрішньої структури програми, її коду та логіки. Це дозволяє перевіряти внутрішні шляхи, умови та цикли.

Проведення тестів

Тестування є життєво важливим етапом у розробці програмного забезпечення для забезпечення його коректної роботи. Ручне тестування, хоч і необхідне, не може виявити всі можливі помилки, тому часто доповнюється автоматизованими тестами. Важливо пам'ятати, що тестування — це не разова подія, а безперервний процес. Під час розробки цього продукту було проведено численні тести, і кожен день приносив нові ідеї для перевірок.

1. Функціональне тестування

Функціональне тестування спрямоване на перевірку того, що система виконує свої функції відповідно до вимог.

Тест 1: Завантаження треків із бази даних

Для виконання цього тесту необхідно додати елементи до програми.

1. Перевірка даних у базі даних (за допомогою pgAdmin):

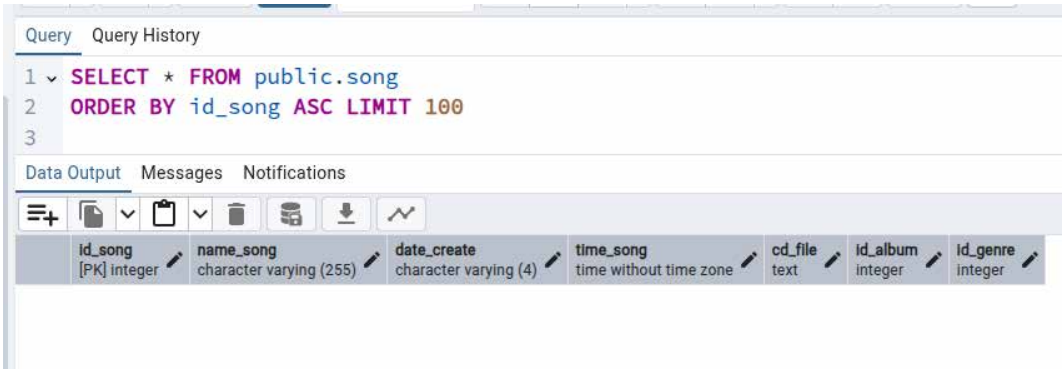


Рис. 4.1 – Виведення даних

Спочатку перевіряємо, чи є вже якісь дані в базі даних.

- Візуальна перевірка:

2. Додавання пісень через програму:

Використовуємо функціонал програми для додавання нових пісень.

Візуальна перевірка:

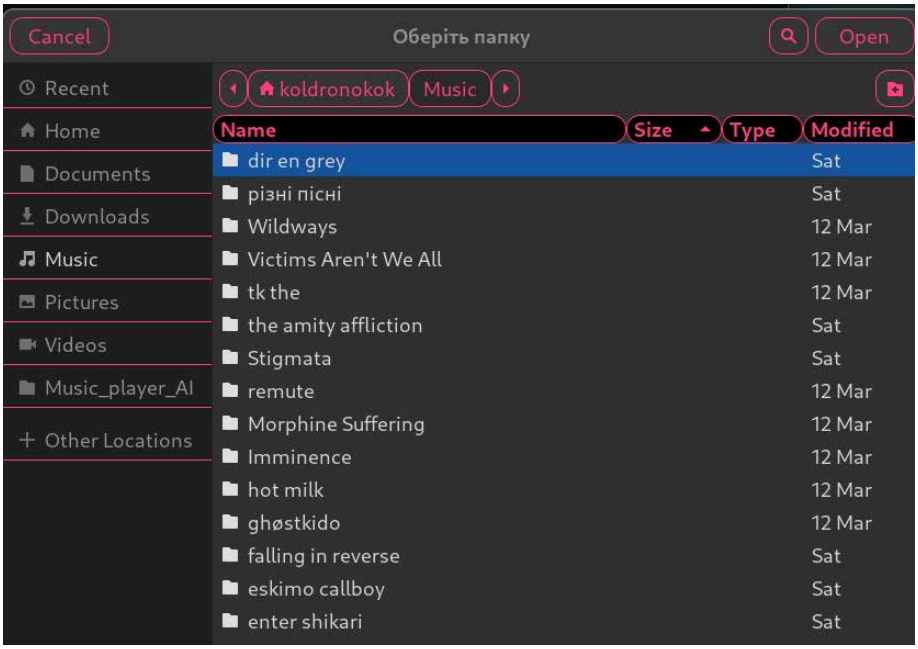
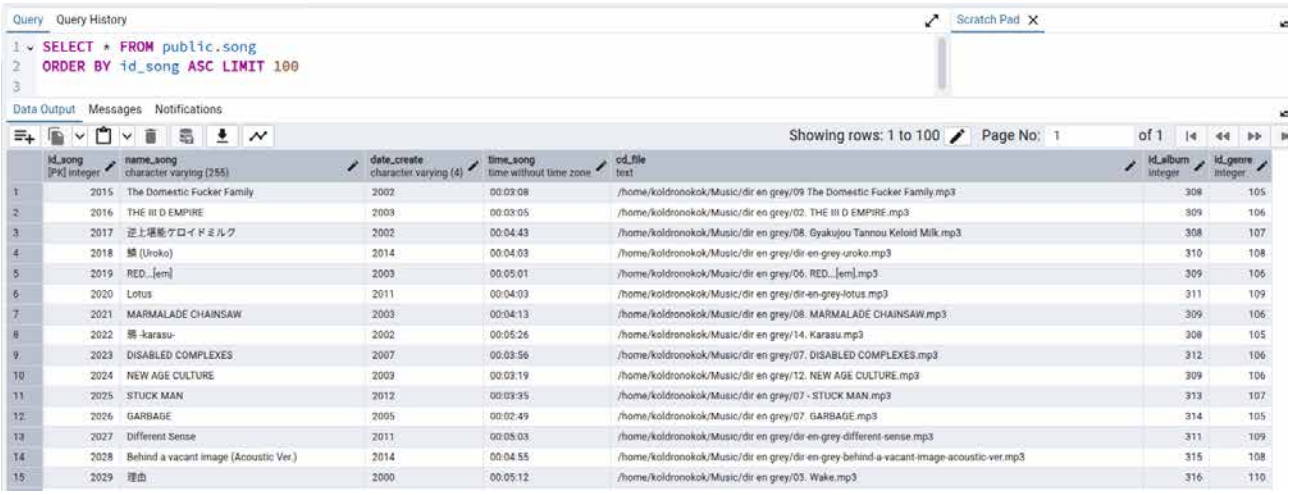


Рис. 4.2 – Добавлення пісень

3. Перевірка, чи додалися дані до бази даних:

- Після додавання пісень ми знову перевіряємо базу даних, щоб переконатися, що нові записи з'явилися.
- Візуальна перевірка:



	id_song [PK] integer	name_song character varying (255)	date_create character varying (4)	time_song time without time zone	cd_file text	id_album integer	id_genre integer
1	2015	The Domestic Fucker Family	2002	00:03:08	/home/koldronokok/Music/dir en grey/09 The Domestic Fucker Family.mp3	308	105
2	2016	THE III D EMPIRE	2003	00:03:05	/home/koldronokok/Music/dir en grey/02. THE III D EMPIRE.mp3	309	106
3	2017	逆上場処ゲロイドミルク	2002	00:04:43	/home/koldronokok/Music/dir en grey/08. Gyakujou Tannou Keloid Milk.mp3	308	107
4	2018	鱗 (Uroko)	2014	00:04:03	/home/koldronokok/Music/dir en grey/dir-en-grey-uroko.mp3	310	108
5	2019	RED...[em]	2003	00:05:01	/home/koldronokok/Music/dir en grey/06. RED...[em].mp3	309	106
6	2020	Lotus	2011	00:04:03	/home/koldronokok/Music/dir en grey/dir-en-grey-lotus.mp3	311	109
7	2021	MARMALADE CHAINSAW	2003	00:04:13	/home/koldronokok/Music/dir en grey/08. MARMALADE CHAINSAW.mp3	309	106
8	2022	枯-karasu-	2002	00:05:26	/home/koldronokok/Music/dir en grey/14. Karasu.mp3	308	105
9	2023	DISABLED COMPLEXES	2007	00:03:56	/home/koldronokok/Music/dir en grey/07. DISABLED COMPLEXES.mp3	312	106
10	2024	NEW AGE CULTURE	2003	00:03:19	/home/koldronokok/Music/dir en grey/12. NEW AGE CULTURE.mp3	309	106
11	2025	STUCK MAN	2012	00:03:35	/home/koldronokok/Music/dir en grey/07 - STUCK MAN.mp3	313	107
12	2026	GARBAGE	2005	00:02:49	/home/koldronokok/Music/dir en grey/07. GARBAGE.mp3	314	105
13	2027	Different Sense	2011	00:05:03	/home/koldronokok/Music/dir en grey/dir-en-grey-different-sense.mp3	311	109
14	2028	Behind a vacant image (Acoustic Ver.)	2014	00:04:55	/home/koldronokok/Music/dir en grey/dir-en-grey-behind-a-vacant-image-acoustic-ver.mp3	315	108
15	2029	理由	2000	00:05:12	/home/koldronokok/Music/dir en grey/03. Wake.mp3	316	110

Рис. 4.3 – Виведення даних

***Результат*:** Дані про пісні успішно додано після виконання дії. Тест пройдено успішно.

Тест 2: Формування звіту

Цей тест перевіряє коректність генерації звітів програмою.

1. Створення звіту:

- У програмі ми обираємо функцію формування звіту та задаємо необхідні дати.

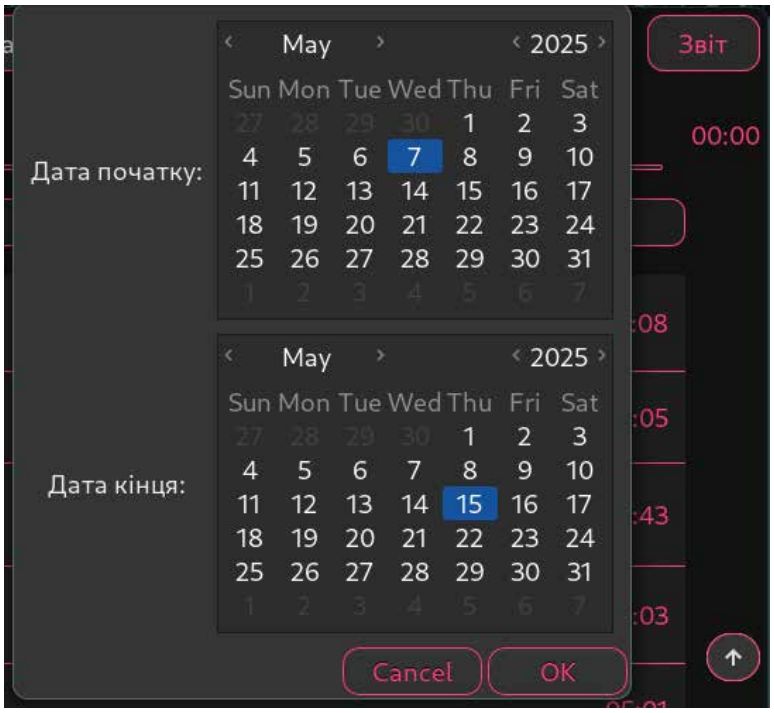


Рис. 4.4 – Створення звіту

- Візуальна перевірка:

2. Перевірка вкладок звіту:

- Перша вкладка: Відображає топ-5 пісень за весь час.
- Візуальна перевірка:

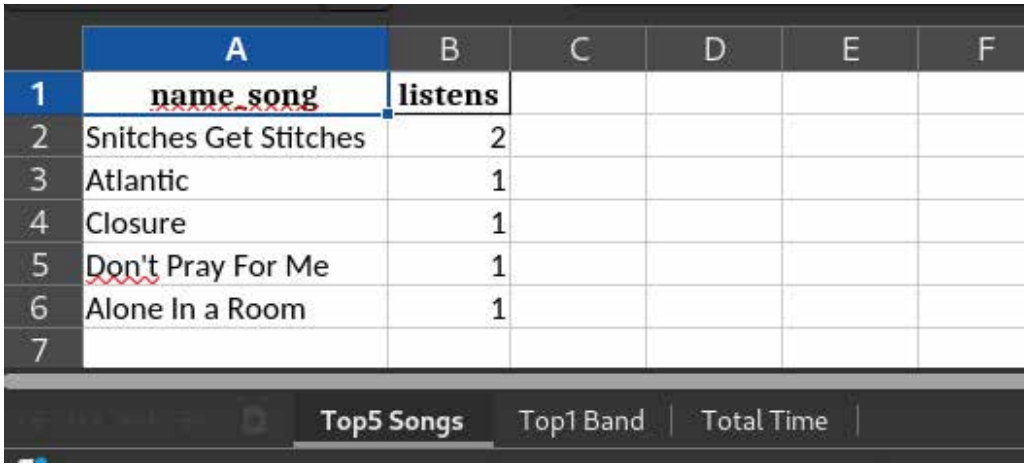


Рис. 4.5 – Експорт звіту в Excel: дані про топ 5 пісень

- Друга вкладка: Відображає топ виконавців за весь час.
- Візуальна перевірка:

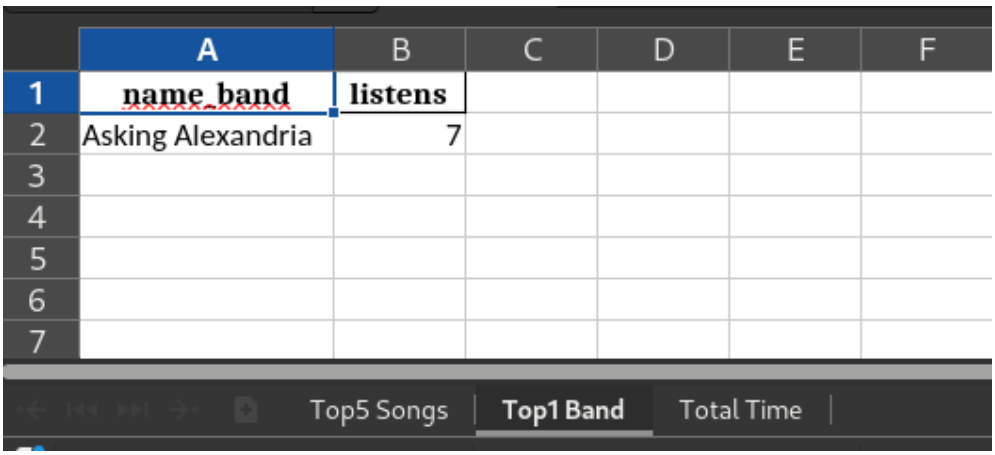


Рис. 4.6 – Експорт звіту в Excel: Дані про топ виконавця

- Третя вкладка: Відображає загальну кількість витрачених годин на прослуховування.
- Візуальна перевірка:

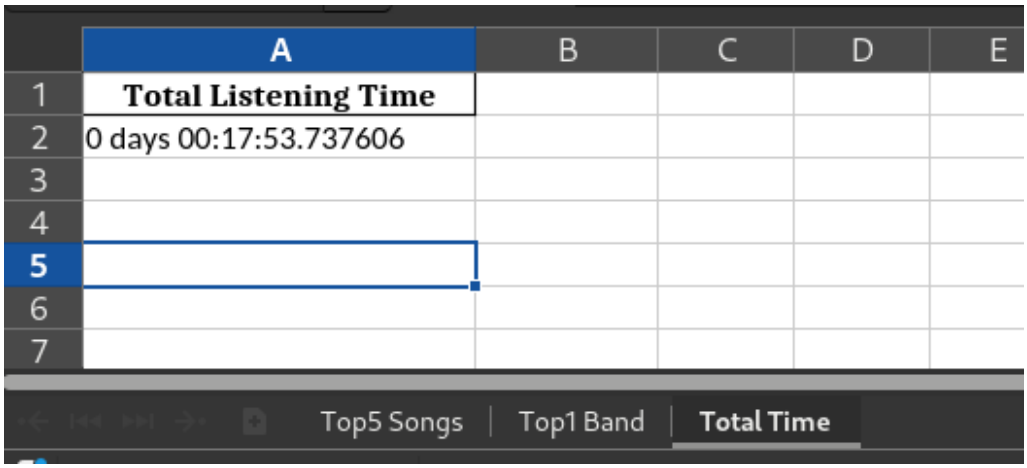


Рис. 4.7 – Експорт звіту в Excel: Дані про кількість потрачених годин

Результат: Функціональні вимоги щодо формування звіту виконані. Також було перевірено коректність роботи при некоректних датах, і в будь-якому випадку звіт надається.

2. Нефункціональне тестування (Non-Functional Testing)

Нефункціональне тестування оцінює нефункціональні аспекти програмного забезпечення, такі як продуктивність, безпека, зручність

використання та сумісність.

Тест 3: Адаптивність екрану під будь-які розміри

Цей тест перевіряє, наскільки гнучко інтерфейс програми адаптується до різних розмірів вікна, зокрема до 1/2 та 1/4 розміру екрану.

- 1/4 екрану: Перевірка відображення програми при зменшенні вікна до чверті екрану.
- Візуальна перевірка:

Рис. 4.8 – Екран 1/4

- 1/2 екрану: Перевірка відображення програми при зменшенні вікна до половини екрану.
- Візуальна перевірка:

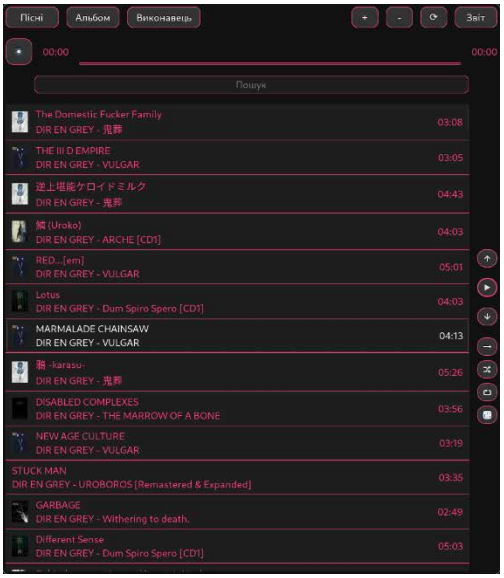


Рис. 4.9 – Екран 1/2

Результат: Програма успішно адаптується до різних розмірів екрану.

Тест 4: Структурне тестування – тестування елементів з дуже великими назвами

Цей тест спрямований на виявлення візуальних дефектів або порушень інтерфейсу при відображенні елементів з надмірно довгими назвами.

- Початковий вигляд (з дефектом):
- Візуальна перевірка:

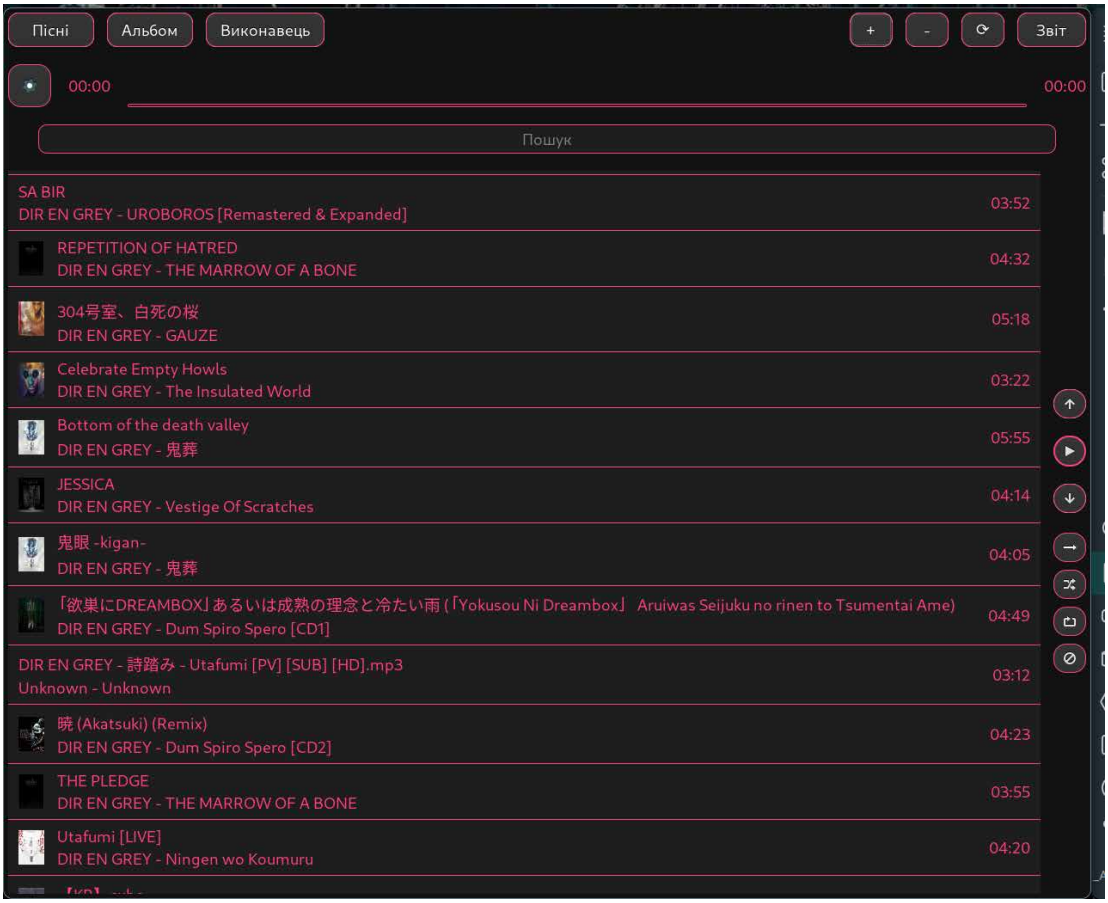


Рис. 4.10 – Екран з дефектом

Результат: Даний тест був провалений, оскільки виявлено дефект у відображенні довгих назв. Для виправлення цього елементу було додано функцію фільтрування тексту.

Виправлення:

- Виклик функції:

```
artist = clean_if_long(artist_)
album = clean_if_long(album_)
genre = clean_if_long(genre_)
title = clean_if_long(title_)
```

Рис. 4.11 – Виклик функції

```
def clean_if_long(s: str, max_len: int = 40) -> str:  4 usages  bontonent
    if len(s) > max_len:
        s = re.sub( pattern: r'\s*\(.*?\)\s*',  repl: ' ', s)
        s = re.sub( pattern: r'\s{2,}',  repl: ' ', s).strip()
    return s
```

- Виконання функції:

Рис.4.12 – Виконання функції

Головна ціль цих тестів полягає не в демонстрації функціоналу програми, а в виявленні та перевірці максимально можливих помилок як за допомогою програмних, так і за допомогою ручних методів.

4.2 Вимоги до апаратного та програмного забезпечення

Для забезпечення стабільної та ефективної роботи MP3-плеєра з функціоналом ШІ, важливо визначити мінімальні та рекомендовані вимоги до апаратного та програмного забезпечення.

Апаратні вимоги до найважливіших компонентів

Визначення апаратних вимог базується на аналізі найбільш навантажених компонентів системи:

1. PostgreSQL

Відповідає за зберігання, обробку та обслуговування запитів до бази даних. Працює у фоновому режимі, обробляючи запити від клієнтської частини (GTK-програми), включно з навчальними циклами системи рекомендацій.

Апаратні вимоги:

- Пам'ять (RAM): Мінімум 1 ГБ для простої роботи, рекомендовано 2–4 ГБ для підтримки кешу запитів та стабільної роботи з навчальними циклами.
- Процесор (CPU): Мінімум 2 ядра, рекомендовано 4 ядра (особливо при регулярному оновленні моделей ШІ).
- Диск: SSD-диск рекомендовано. Не менше 500 МБ під базу даних (залежить від кількості пісень та історії прослуховування).

2. GTK-застосунок

Опис: Виконує основну логіку програми, відображає інтерфейс, забезпечує взаємодію з користувачем, а також передає та отримує дані з бази даних. Містить блоки з обробкою MP3-файлів, відображенням даних, а також реалізує машинне навчання для формування рекомендацій.

Апаратні вимоги:

- Пам'ять (RAM): Мінімум 2 ГБ, рекомендовано 4 ГБ (особливо при обробці великих обсягів даних або виконанні машинного навчання).
- Процесор (CPU): Мінімум 2 ядра, рекомендовано 4 ядра (через фонові обчислення та оновлення рекомендацій).
- Дисплей: Підтримка роздільної здатності від 1024x768 пікселів.

3. Scikit-learn (для ШІ)

Опис: Використовується для навчання та покращення моделей ШІ. Головна мета — забезпечити програму алгоритмом для персоналізованих рекомендацій користувачу.

Апаратні вимоги:

При навчальному процесі:

- RAM: Мінімум 2 ГБ, рекомендовано 4–8 ГБ при великих обсягах прослуховувань (1000+ записів).
- CPU: Рекомендовано 4 ядра і більше.

При фоновому процесі (використання моделі):

- RAM: 1–2 ГБ (достатньо для розгортання моделі середнього розміру).
- CPU: Мінімум 2 ядра (при використанні моделі в реальному часі).

Загальні апаратні вимоги

Зведені апаратні вимоги для всієї системи:

Оперативна пам'ять (RAM): Мінімум:

- 2 ГБ — для базового функціонування всіх компонентів (GTK, PostgreSQL, модель ШІ).
- Рекомендовано: 4–8 ГБ — для комфортної роботи, швидкого навчання моделей на значних обсягах даних (~1000+ записів), ефективної фонової обробки та кешування.

1. Процесор (CPU):

- Мінімум: 2 ядра — достатньо для запуску GTK-програми, базових запитів до PostgreSQL та використання моделі ШІ.
- Рекомендовано: 4 ядра — забезпечує швидке навчання моделей, стабільну обробку запитів та плавну роботу інтерфейсу без затримок.

2. Накопичувач (диск):

- Тип: SSD обов'язково — значно пришвидшує доступ до бази даних, завантаження MP3-файлів та обробку даних.
- Об'єм: Мінімум 500 МБ під базу даних + додатковий простір для аудіофайлів (від ~1 ГБ залежно від кількості MP3).

Графіка / дисплей:

Роздільна здатність: Від 1024×768 пікселів — для коректного

відображення GTK-інтерфейсу.

Вимоги до програмного забезпечення

- Операційна система: Linux (Ubuntu 20.04+, Arch, Debian 11+ або інший дистрибутив з підтримкою `systemd`).
- PostgreSQL: версія 13+.
- Python: версія 3.9+.
- Бібліотеки Python:
  - `PyGObject`: версія 3.44.1+
  - `psycopg2-binary`: версія 2.9.9+
  - `mutagen`: версія 1.47.0+
  - `pandas`: версія 2.2.2+
  - `openpyxl`: версія 3.1.2+
  - `scikit-learn`: версія 1.4.2+
  - `numpy`: версія 1.26.4+
  - `joblib`: версія 1.4.0+
  - `spotipy`: версія 2.23.0+
  - `requests`: версія 2.31.0+

Наведені версії є оптимальними для цього проєкту, хоча можливе використання і новіших версій цих елементів.

4.3 Склад інсталяційного пакета

Діаграма розміщення

Діаграма розміщення допомагає візуалізувати структуру інсталяційного пакета та розташування компонентів програми після встановлення, що полегшує розуміння архітектури проєкту (рис. 4.13).

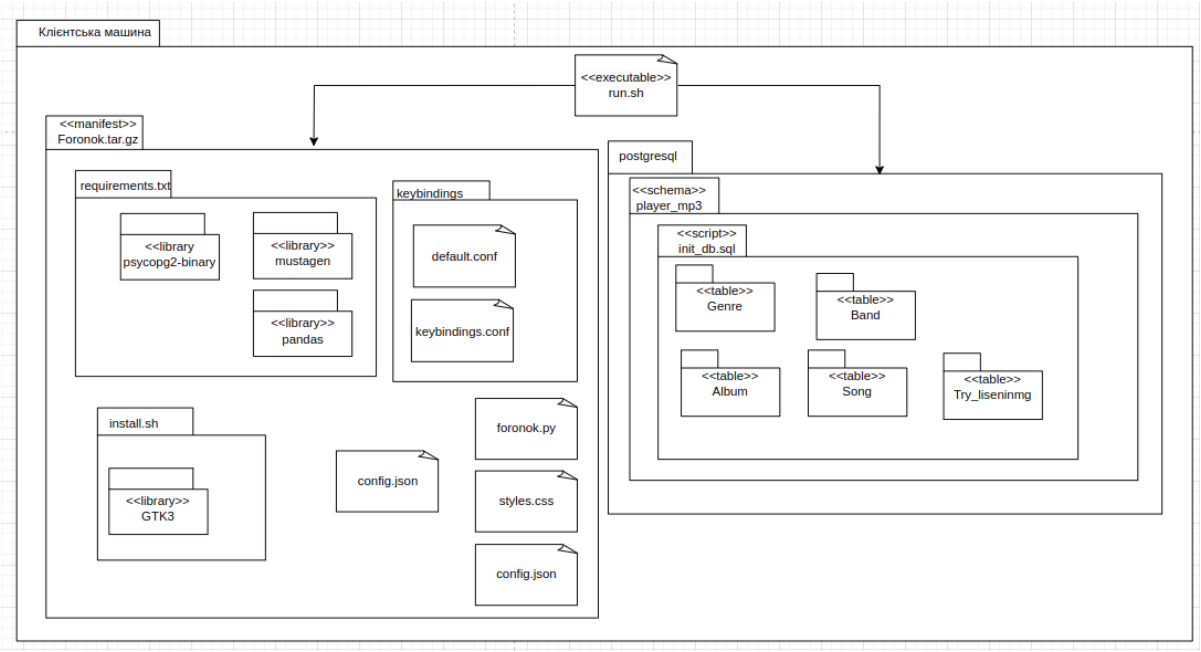


Рис. 4.13 – Діаграма розміщення

Опис елементів діаграми:

1. Клієнтська машина: На клієнтській машині розгортається додаток Foronok, який встановлюється з архіву `Foronok.tar.gz`. Цей архів містить:

- `install.sh`: Скрипт, що встановлює необхідні системні залежності, зокрема графічну бібліотеку GTK3.
- `foronok.py`: Головний виконуваний Python-скрипт програми.
- `styles.css`: Файл стилів для відображення інтерфейсу користувача.
- `config.json`: Файл конфігурації, що зберігає налаштування програми.
- `keybindings/`: Директорія, що містить файли для конфігурації користувацьких прив'язок клавіш:
- `default.conf`: Налаштування для запуску самої програми через комбінацію клавіш.
- `keybindings.conf`: Скрипти для внутрішньої роботи програми з клавішами.

- `requirements.txt`: Файл, що перелічує додаткові Python-залежності, які необхідно встановити перед запуском програми (наприклад, `psycopg2-binary` для з'єднання з PostgreSQL, `mutagen` для роботи з MP3-тегами та `pandas` для обробки даних).

2. Сервер бази даних PostgreSQL: На сервері бази даних розгорнута схема `player_mp3`. Ініціалізація цієї бази даних здійснюється через скрипт `init_db.sql`. Схема містить п'ять основних таблиць:

- `Genre`: Зберігає інформацію про музичні жанри.
- `Band`: Містить дані про виконавців або гурти.
- `Album`: Зберігає інформацію про музичні альбоми.
- `Song`: Основна таблиця з деталями про пісні.
- `Try_listening`: Лог прослуховування, який може використовуватись для побудови рекомендацій ШІ.

Загальна картина: Додаток `foronok.py` встановлює з'єднання з базою даних PostgreSQL через бібліотеку `psycopg2`, використовуючи для запуску скрипт `run.sh`. Робота клієнтської частини програми повністю залежить від доступності та коректності схеми `player_mp3`, оскільки всі дії з музичними даними та історією прослуховування здійснюються через неї.

Можливість встановити програму через GitHub

Linux-система дозволяє виконувати більшість дій через консоль, за винятком безпосереднього використання графічних програм. Щоб встановити програму або проєкт з GitHub на Linux-системі через термінал, потрібно виконати наступні кроки:

1. Встановити Git:

sudo apt update # Оновлення списків пакетів (для Debian/Ubuntu)

sudo apt install git # Встановлення Git

(Примітка: для інших дистрибутивів використовуйте відповідний пакетний менеджер, наприклад dnf install git для Fedora або pacman -S git для

Arch Linux).

2. Клонувати репозиторій з GitHub:

git clone https://github.com/koldronokok/Music_player_AI.git

3. Перейти в директорію проєкту:

cd Music_player_AI

4. Ознайомитися з інструкцією: Більшість репозиторіїв містять файл README.md або INSTALL.md з інструкціями. Його можна переглянути за допомогою:

Bash

cat README.md

5. Встановити залежності:

chmod +x install.sh # Надати скрипту дозволи на виконання

./install.sh # Запустити скрипт встановлення залежностей

6. Запустити програму:

chmod +x run.sh

./run.sh

Якщо користувач бажає запустити програму вручну, він може виконати:

python3 foronok.py

Для того, щоб система запрацювала, необхідно інсталювати файли, що входять до складу інсталяційного пакету, згідно з діаграмою розміщення. Склад інсталяційного пакету чітко визначений.

Можливість встановити програму через репозиторії

Реалізація проєкту також передбачає можливість розповсюдження через репозиторій Flatpak. Flatpak — це система для розгортання та керування програмним забезпеченням на Linux, що дозволяє розробникам публікувати програми, які працюють незалежно від дистрибутива, забезпечуючи регулярні оновлення. (посилання [6]).

Для публікації проєкту у Flatpak необхідно мати такі файли:

- `requirements.txt`: Список Python-залежностей.
- `org.example.Foronok.yaml`: Flatpak-маніфест, що описує, як зібрати та встановити програму.
- `org.example.Foronok.desktop`: Файл з коротким описом програми та її метаданими для інтеграції в робоче середовище.
- `icon.png`: Іконка програми (розміри від 64x64 до 512x512 px).
- Бажано також мати файл `README.md` для виведення проєктної інформації.

Ключові умови для публікації у Flatpak:

- Програма має бути у публічному репозиторії (наприклад, GitHub / GitLab).
- Повинен бути тег релізу (наприклад, `v1.0.0`).
- Ліцензія має бути відкритою (наприклад, MIT, GPL, Apache).

Для публікації необхідно створити директорію `org.example.Foronok/` та додати до неї всі перелічені файли.

Опис файлових системних потреб:

- `org.example.Foronok.desktop`: Файл-опис для інтеграції програми в меню додатків.
- `org.example.Foronok.yaml`: Цей файл детально описує процес збірки та встановлення програми у форматі Flatpak.

Код Flatpak-маніфесту (`org.example.Foronok.yaml`) знаходиться у

ВИСНОВКИ

У рамках цієї дипломної роботи було проведено ґрунтовне дослідження та створено багатофункціональний аудіоплеєр для операційної системи Linux із підтримкою штучного інтелекту, який здатен працювати в автономному режимі без потреби у постійному підключенні до Інтернету.

Здійснено повноцінне моделювання програмної системи — розроблено діаграми варіантів використання, активності, послідовності, а також ER-діаграму для побудови логічної моделі бази даних. Це дало змогу ретельно спланувати архітектуру додатку.

Розроблено функціонал, що відповідає потребам цільової аудиторії — молодих користувачів Linux віком 15–25 років, які мають технічну підготовку та шукають альтернативу стрімінговим сервісам з надмірною автоматизацією та рекламою.

Інтеграція штучного інтелекту реалізована через механізм рекомендацій, який враховує історію прослуховувань користувача. Це підвищує персоналізованість досвіду без жорсткої залежності від зовнішніх алгоритмів стрімінгових сервісів.

Програма відповідає принципам відкритого програмного забезпечення — вона поширюється безкоштовно, її код може бути змінений користувачами, що сприяє розвитку спільноти та потенційному вдосконаленню проєкту. Та протестована на кількох дистрибутивах Linux, з урахуванням особливостей різних графічних оболонок, що підтвердило її стабільність, адаптивність та продуктивність.

За результатами дослідження, розроблена система має значний потенціал для подальшого розвитку та застосування. В її основу покладено інноваційне технічне рішення: поєднання бази даних, інструментів машинного навчання та стороннього API (Spotify) для створення персоналізованого музичного середовища. Такий підхід дозволив розробити автономний, зручний та

ефективний інструмент для слухачів, які цінують гнучкість, конфіденційність та стабільну роботу без зайвих ресурсних витрат.

На основі проведеного аналізу та отриманих результатів можна стверджувати, що впровадження подібних рішень є технічно доцільним, економічно ефективним та перспективним, особливо з урахуванням сучасного розвитку апаратного забезпечення. У процесі розробки було запропоновано власне бачення побудови системи музичних рекомендацій, адаптоване під потреби локального користувача та сучасні тенденції Linux-платформи.

Результати дипломної роботи підтверджують, що розроблений аудіоплеєр здатен задовольнити запити сучасного користувача Linux, поєднуючи автономність, інтелектуальні можливості, відкритість і гнучкість.

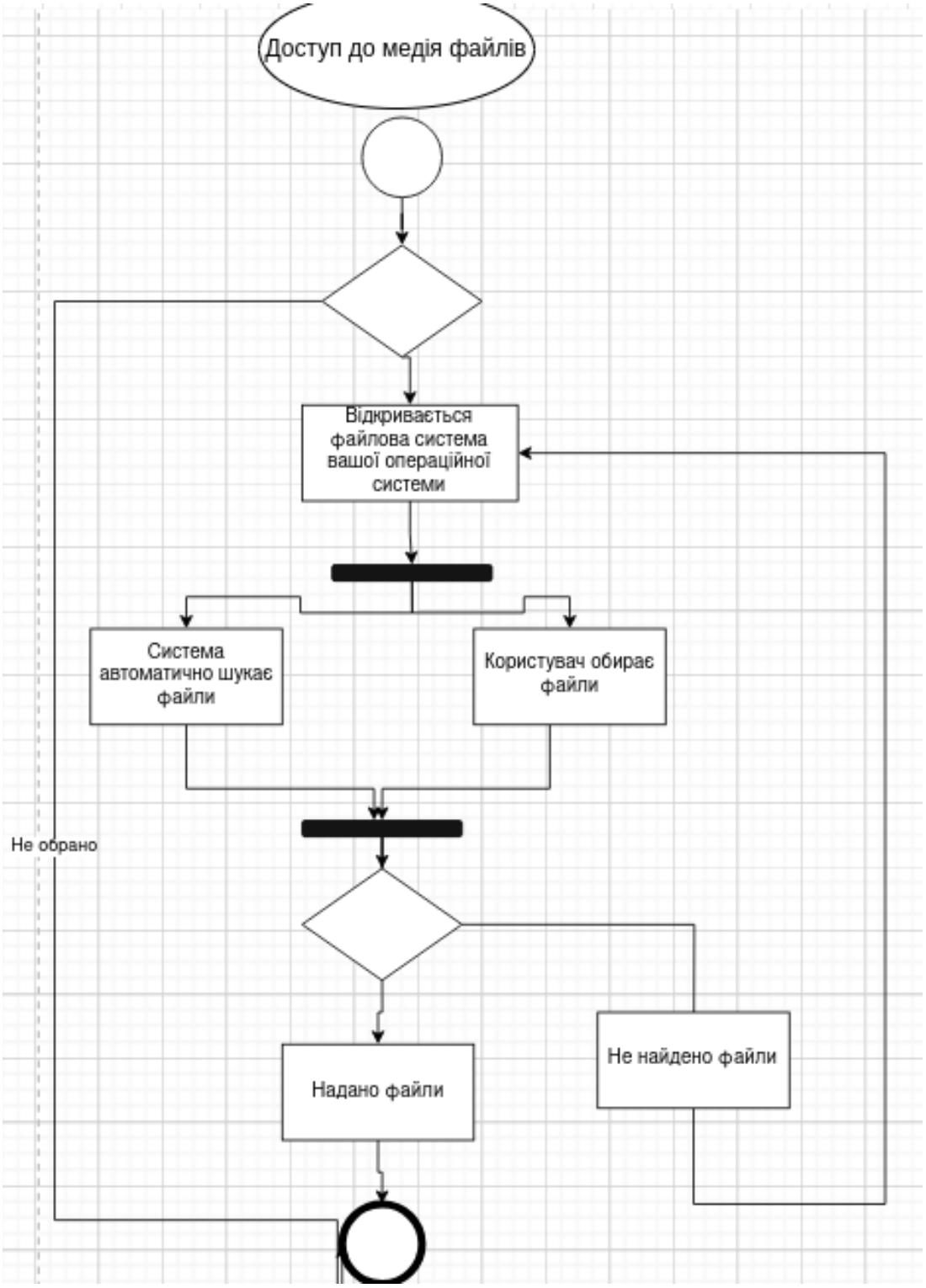
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

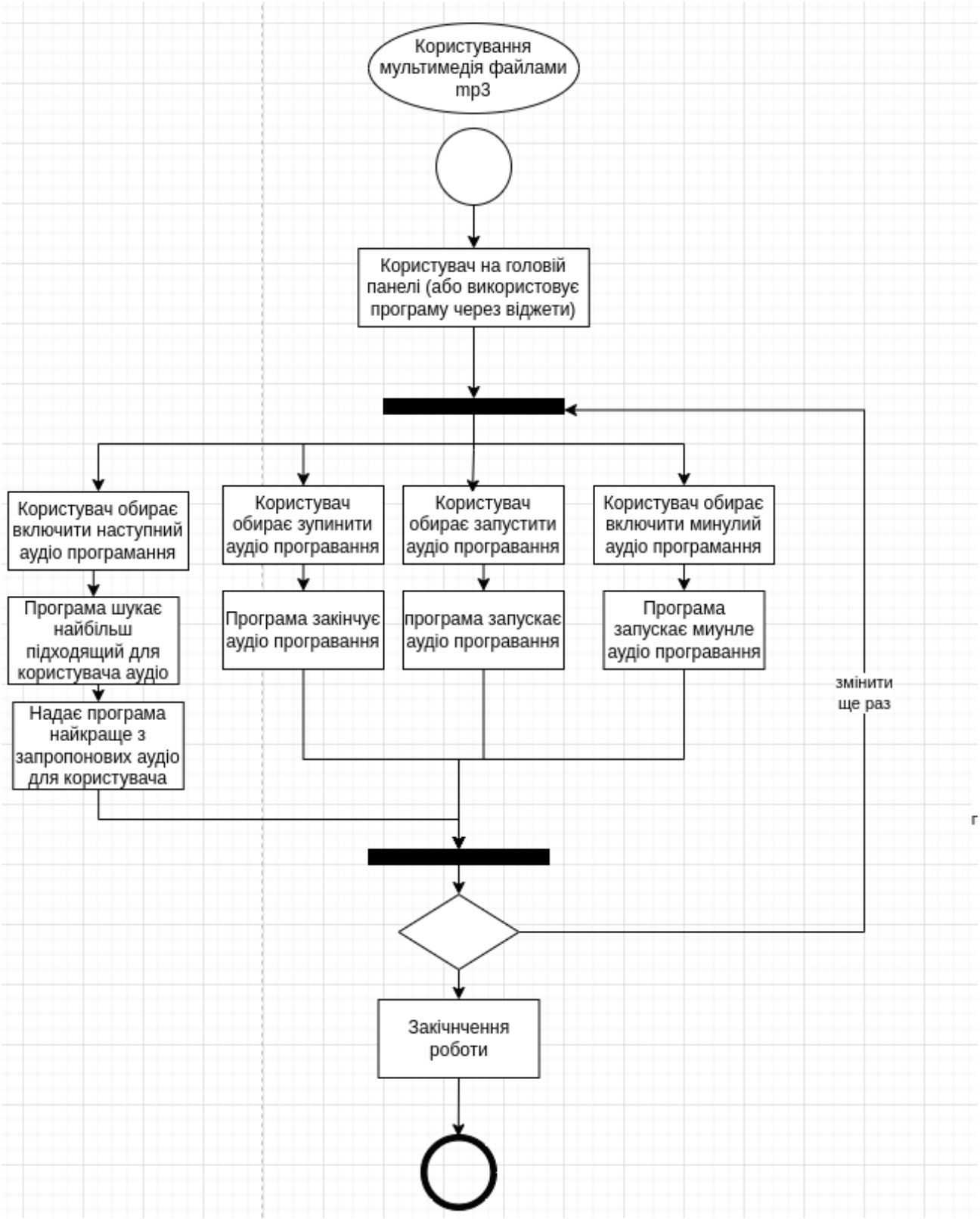
1. Audio Signal Processing for Music Applications. Coursera. URL: https://www.coursera.org/learn/audio-signal-processing (Дата звернення: 05.03.2025).
2. Encourage Women in Linux. The Linux Documentation Project. URL: https://tldp.org/HOWTO/Encourage-Women-Linux-HOWTO/x106.html (Дата звернення: 01.05.2025).
3. Filesystem Hierarchy Standard. Wikipedia. URL: https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard (Дата звернення: 12.05.2025).
4. Linux Statistics: Users, Market Share and Facts in 2024. Truelist Blog. URL: https://truelist.co/blog/linux-statistics/ (Дата звернення: 24.05.2025).
5. Normal Forms in DBMS. GeeksforGeeks. URL: https://www.geeksforgeeks.org/normal-forms-in-dbms/ (Дата звернення: 09.07.2025).
6. Why Flathub? Flathub Documentation. URL: https://docs.flathub.org/docs/for-users/why-flathub (Дата звернення: 22.05.2025).
7. Види тестування програмного забезпечення. Guru99. URL: https://www.guru99.com/uk/types-of-software-testing.html (Дата звернення: 18.05.2025).
8. Підтримка–Spotify. Spotify. URL: https://support.spotify.com/ua-uk/#getting-started (Дата звернення: 01.04.2025).
9. Rhythmbox – музичний програвач для GNOME. Wikipedia. URL: https://en.wikipedia.org/wiki/Rhythmbox (Дата звернення: 02.04.2025).
10. Що таке YouTube Music. Digital Trends. URL: https://www.digitaltrends.com/home-theater/what-is-youtube-music/ (Дата звернення: 03.04.2025).
11. Побудова діаграми варіантів використання (Use Case Diagram). Creately. URL: https://creately.com/guides/use-case-diagram-tutorial/ (Дата звернення: 08.04.2025).
12. Діаграма активностей UML. Lucidchart. URL: https://www.lucidchart.com/pages/uml-activity-diagram (Дата звернення: 09.04.2025).
13. Діаграма послідовності UML. Education Wiki. URL: https://uk.education-wiki.com/8710755-uml-sequence-diagram (Дата звернення: 10.04.2025).
14. Діаграма зв’язків сутностей (ER-діаграма). Database Star. URL: https://www.databasestar.com/entity-relationship-diagram/ (Дата звернення: 11.04.2025).
15. Розробникам – PostgreSQL. PostgreSQL. URL: https://www.postgresql.org/developer/ (Дата звернення: 15.04.2025).
16. MySQL Workbench – інструмент для моделювання БД. MySQL. URL:

https://www.mysql.com/products/workbench/ (Дата звернення: 16.04.2025).
17. Oracle Cloud Infrastructure – можливості та переваги. Oracle. URL: https://www.oracle.com/cloud/why-oci/ (Дата звернення: 17.04.2025).
18. Про SQLite. SQLite. URL: https://sqlite.org/about.html (Дата звернення: 18.04.2025).
19. Система керування базами даних IBM Db2. IBM. URL: https://www.ibm.com/db2 (Дата звернення: 19.04.2025).
20. Microsoft Access – база даних для Microsoft 365. Microsoft. URL: https://www.microsoft.com/en-us/microsoft-365/access (Дата звернення: 20.04.2025).
21. Інструменти розробника для SQL Server. Microsoft. URL: https://www.microsoft.com/en-us/sql-server/developer-tools (Дата звернення: 21.04.2025).
22. Про проєкт GNOME. GNOME. URL: https://www.gnome.org/about (Дата звернення: 23.04.2025).
23. Платформа розробки KDE. KDE. URL: https://develop.kde.org (Дата звернення: 24.04.2025).
24. Hyprland – динамічний композитний менеджер Wayland. Hyprland. URL:https://hyprland.org (Дата звернення: 26.04.2025).
25. Мова програмування C++. Wikipedia. URL: https://uk.wikipedia.org/wiki/C%2B%2B (Дата звернення: 26.04.2025).
26. Офіційна документація Python. Python.org. URL: https://www.python.org/doc/ (Дата звернення: 25.04.2025).
27. C# – мова програмування. Wikipedia. URL: https://en.wikipedia.org/wiki/C_Sharp_(programming_language) (Дата звернення: 26.04.2025).
28. Java для початківців – навчальні матеріали. Dev.java. URL: https://dev.java/learn/ (Дата звернення: 26.04.2025).
29. Бібліотека psycopg2 – адаптер PostgreSQL для Python. PyPI. URL: https://pypi.org/project/psycopg2/ (Дата звернення: 27.04.2025).
30. Бібліотека pandas – обробка та аналіз даних. PyPI. URL: https://pypi.org/project/pandas/ (Дата звернення: 28.04.2025).
31. Бібліотека mutagen – обробка тегів медіафайлів. PyPI. URL: https://pypi.org/project/mutagen/ (Дата звернення: 29.04.2025).
32. Бібліотека spotipy – робота з API Spotify. PyPI. URL: https://pypi.org/project/spotipy/ (Дата звернення: 25.04.2025).
33. Бібліотека joblib – збереження моделей та обчислення. PyPI. URL: https://pypi.org/project/joblib/ (Дата звернення: 22.04.2025).
34. Бібліотека scikit-learn – машинне навчання в Python. PyPI. URL: https://pypi.org/project/scikit-learn/ (Дата звернення: 21.04.2025).

ДОДАТОК А

Діаграма активностей додаткових елементів




Користування мультимедія файлами mp3
Користувач на головій панелі (або використовує програму через віджети)
Користувач обирає включити наступний аудіо програмання
Користувач обирає зупинити аудіо програвання
Користувач обирає запустити аудіо програвання
Користувач обирає включити минулий аудіо програмання
Програма шукає найбільш підходящий для користувача аудіо
Програма закінчує аудіо програвання
програма запускає аудіо програвання
Програма запускає миунле аудіо програвання
Надає програма найкраще з запропонових аудіо для користувача
змінити ще раз
Закічнчення роботи

ДОДАТОК Б

Скрипт створення таблиць бази даних

```
CREATE TABLE Band(
	ID_band SERIAL PRIMARY KEY,
	name_band VARCHAR(255) NOT NULL
);

CREATE TABLE Album(
	ID_album SERIAL PRIMARY KEY,
	name_album VARCHAR(255) NOT NULL,
	photo BYTEA,
	ID_band SERIAL,
	FOREIGN KEY (ID_band) REFERENCES Band(ID_band)
);

CREATE TABLE Song(
	ID_song SERIAL PRIMARY KEY,
	name_song VARCHAR(255) NOT NULL,
	date_create VARCHAR(4) NOT NULL,
	time_song TIME NOT NULL,
	cd_file TEXT NOT NULL,
	ID_album SERIAL,
	ID_genre SERIAL,
	FOREIGN KEY (ID_album) REFERENCES Album(ID_album),
	FOREIGN KEY (ID_genre) REFERENCES Genre(ID_genre)
);

CREATE TABLE Try_lisening
(
	ID_try_lisening SERIAL PRIMARY KEY,
	precent_lisening INTEGER,
	time_lisening TIME,
	date_lisening DATE,
	random bool,
	ID_song SERIAL,
	FOREIGN KEY (ID_song) REFERENCES Song(ID_song)
);
```

ДОДАТОК В

Тригери бази даних

percent_lisening:

```
CREATE OR REPLACE FUNCTION update_precent_lisening()
RETURNS TRIGGER AS $$
DECLARE
  song_duration INTERVAL;
BEGIN
  -- Отримуємо тривалість пісні
  SELECT time_song INTO song_duration
  FROM Song
  WHERE ID_song = NEW.ID_song;
  -- Обчислюємо відсоток (якщо обидва значення існують)
  IF song_duration IS NOT NULL AND NEW.time_lisening IS NOT NULL THEN
    NEW.precent_lisening := EXTRACT(EPOCH FROM NEW.time_lisening) / EXTRACT(EPOCH FROM song_duration) * 100;
  ELSE
    NEW.precent_lisening := NULL;
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_calc_precent_lisening
BEFORE INSERT OR UPDATE ON Try_lisening
FOR EACH ROW
EXECUTE FUNCTION update_precent_lisening();
```

Тригер 2

```
CREATE OR REPLACE FUNCTION check_time_lisening()
RETURNS TRIGGER AS $$
DECLARE
  song_duration INTERVAL;
BEGIN
  -- Отримуємо тривалість пісні
  SELECT time_song INTO song_duration
  FROM Song
  WHERE ID_song = NEW.ID_song;
  -- Перевірка
  IF NEW.time_lisening > song_duration THEN
```

```
            RAISE EXCEPTION 'Час прослуховування (%s) перевищує тривалість пісні (%s)', NEW.time_lisening, song_duration;
        END IF;
        RETURN NEW;
    END;
    $$ LANGUAGE plpgsql;
    CREATE TRIGGER trg_check_time_lisening
    BEFORE INSERT OR UPDATE ON Try_lisening
    FOR EACH ROW
    EXECUTE FUNCTION check_time_lisening();
```

ДОДАТОК Д

Скрипт створення початкових даних

```
def load_music_from_folder(self, folder_path):
    connection = connect_to_db()
    if not connection:
        return
    cursor = connection.cursor()
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            if not file.endswith(".mp3"):
                continue
            file_path = os.path.join(root, file)
            try:
                audio = EasyID3(file_path)
                id3 = ID3(file_path)
                mp3_info = MP3(file_path)
            except Exception as e:
                print(f"Не вдалося прочитати файл: {file_path}, помилка: {e}")
                continue
            title = audio.get("title", [file.replace(".mp3","").strip()])[0]
            artist = audio.get("artist", ["Unknown"])[0]
            album = audio.get("album", ["Unknown"])[0]
            genre = audio.get("genre", ["Unknown"])[0]
            year = audio.get("date", ["????"])[0]
            date_create = f"{year}"
            duration_sec = int(mp3_info.info.length)
            time_song = (datetime.min + timedelta(seconds=duration_sec)).time()
            norm_genre = normalize_name(genre)
            norm_artist = normalize_name(artist)
            norm_album = normalize_name(album)
            # Extract album photo
            album_photo = None
            for tag in id3.items():
                if isinstance(tag[1], APIC):
                    album_photo = tag[1].data
                    break
            # Жанр
            cursor.execute(
                "SELECT ID_genre FROM Genre WHERE lower(name_genre) =
%s", (norm_genre,)          )
            genre_id = cursor.fetchone()
```

```
                if not genre_id:
                    cursor.execute(
                        "INSERT INTO Genre(name_genre) VALUES (%s) RETURNING ID_genre", (genre,)            )
                    genre_id = cursor.fetchone()
                # Гурт
                cursor.execute(
                    "SELECT ID_band FROM Band WHERE lower(name_band) = %s", (norm_artist,)          )
                band_id = cursor.fetchone()
                if not band_id:
                    cursor.execute(
                        "INSERT INTO Band(name_band) VALUES (%s) RETURNING ID_band", (artist,)            )
                    band_id = cursor.fetchone()
                # Альбом
                cursor.execute(
                    "SELECT ID_album FROM Album WHERE lower(name_album) = %s AND ID_band = %s",
                    (norm_album, band_id[0])          )
                album_id = cursor.fetchone()
                if not album_id:
                    cursor.execute(
                        "INSERT INTO Album(name_album, ID_band, photo) VALUES (%s, %s, %s) RETURNING ID_album",
                        (album, band_id[0], album_photo)
                    )
                    album_id = cursor.fetchone()
                # Пісня
                cursor.execute(
                    """INSERT INTO Song(name_song, date_create, time_song, cd_file, ID_album, ID_genre)
                       VALUES (%s, %s, %s, %s, %s, %s)""",
                    (title, date_create, time_song, file_path, album_id[0], genre_id[0])
)
                connection.commit()
            cursor.close()
            connection.close()
            self.load_music_from_db()  # Перезавантажуємо список після додавання нових елементів
```