Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка модуля «Аналіз музичних уподобань користувачів для персоналізованих плей-листів» інформаційної системи музичного сервісу

(тема)

Виконав:
здобувач 4 року навчання,
групи ІТУ-21-1
Олександр ЛЕВІН
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні технології управління
(повна назва освітньої програми)

Керівник: доц. Віктор БОРИСЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри ІУС _____ Костянтин ПЕТРОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Інформаційних управляючих систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні технології управління
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
" 19 " травня 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Левіну Олександру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка модуля «Аналіз музичних уподобань користувачів для створення персоналізованих плей-листів» інформаційної системи музичного сервісу

затверджена наказом по університету від " 19 " травня 2025 р. № 370Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії " 17 " червня 2025 р.

3. Вихідні дані до роботи відомості про музичні сервіси; матеріали переддипломної практики; публікації та інтернет-джерела з досліджуваної проблеми.

4. Перелік питань, що потрібно опрацювати у роботі аналіз предметної області та постановка задачі дослідження; огляд і аналіз сучасного стану розглянутої проблеми; формулювання завдання розробки; опис архітектури об'єкта розробки на рівні функцій; розробка та обґрунтування елементів інформаційної, математичної та програмної забезпечуючої системи модуля.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Дослідження предметної області та визначення актуальності автоматизації аналізу музичних уподобань користувачів	20.05.2025 –21.05.2025	Виконано
2	Огляд існуючих музичних платформ та сервісів персоналізованих рекомендацій	22.05.2025 –24.05.2025	Виконано
3	Постановка задачі розробки модуля для генерації плейлистів	25.05.2025 –30.05.2025	Виконано
4	Проєктування функціональної структури інформаційної системи	31.05.2025 –03.06.2025	Виконано
5	Розробка концепції та структури бази даних для збереження треків і вподобань	04.06.2025 –06.06.2025	Виконано
6	Реалізація програмних модулів збору, обробки та класифікації музичних даних	07.06.2025 –09.06.2025	Виконано
7	Оформлення пояснювальної записки	10.06.2025 –12.06.2025	Виконано
8	Захист кваліфікаційної роботи	18.06.2025	Виконано

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Віктор БОРИСЕНКО
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 79 с., 16 рис., 4 табл., 1 дод., 16 джерел.

АВТОМАТИЗАЦІЯ, АНАЛІЗ НАСТРОЮ, БАЗА ДАНИХ, ВИЗНАЧЕННЯ ЖАНРУ, ІНТЕРФЕЙС, КЛАСИФІКАЦІЯ ТЕКСТУ, МАШИННЕ НАВЧАННЯ, , МУЗИЧНІ ВПОДОБАННЯ, ОБРОБКА ТЕКСТІВ, ШТУЧНИЙ ІНТЕЛЕКТ.

У сучасному цифровому суспільстві музика є невід'ємною частиною повсякденного життя, а кількість доступного контенту стрімко зростає. Користувачам стає все складніше знаходити треки, що відповідають їхнім інтересам і емоційному стану. Саме тому виникає потреба в розробці інтелектуального модуля, яка б автоматично аналізувала музичні вподобання користувача та пропонувала персоналізовані плейлисти.

У рамках проєкту досліджуються існуючі рекомендаційні системи, методи текстової класифікації. Основна мета - створити модуль, що використовує штучний інтелект для аналізу текстів пісень і створення персоналізованих плейлистів. Система має враховувати особисті вподобання, частоту прослуховування, емоційні характеристики композицій,.

У проєкті було розроблено інформаційну систему, що збирає дані з музичних сервісів, аналізує тексти пісень, визначає жанр, настрій тощо. На основі цих даних створюється персоналізований список треків. Також передбачено врахування популярності пісень та кількості їх прослуховувань для підвищення точності рекомендацій.

Програмна частина створена з використанням мови Python та бібліотек transformers, langdetect [5]. Для використано моделі машинного навчання: zero-shot-класифікатор для жанру й настрою, а також модель для визначення мови. У результаті формується адаптивний плейлист відповідно до користувача.

ABSTRACT

Thesis: 79 pages, 16 figures, 4 tables, 1 appendix, 16 sources.

ARTIFICIAL INTELLIGENCE, AUTOMATIZATION, CLASSIFICATION OF TEXTS, DATABASE, GENRE DETECTION, LANGUAGE MODEL, MACHINE LEARNING, MOOD ANALYSIS, MUSIC PREFERENCES, PERSONALIZED PLAYLIST, RECOMMENDATION SYSTEM, TEXT PROCESSING.

In today's digital society, music is an integral part of everyday life, and the amount of available content is growing rapidly. Users find it increasingly difficult to discover tracks that match their interests and emotional states. This creates the need for an intelligent module capable of automatically analyzing user's music preferences and generating personalized playlists based on song genre, mood, and language.

This project explores existing music systems, text classification methods, and approaches to detecting mood and language in songs. The main goal is to develop a module that uses artificial intelligence to analyze song lyrics and generate personalized playlists. The system takes into account personal preferences, listening frequency, and emotional characteristics of the compositions to provide relevant music.

In this project, an information system was developed that collects data from Spotify [1] and Deezer [2], analyzes song's lyrics, and determines the genre, mood, and language of tracks. All collected information is stored in a structured database. Based on this data, a personalized tracklist is generated for each user. The system also considers track popularity to improve recommendation accuracy.

The software component is built using Python and libraries such as transformers, langdetect, requests, and sqlite3. For classification, machine learning models are used: a zero-shot classifier for genre and mood, and a language detection model. Track data is retrieved from external APIs, and the processing results are stored in a database. As a result, an adaptive playlist is formed according to the preferences.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ІС – інформаційна система

ІП – інтерфейс користувача

КТ – класифікація тексту

ММ – мовна модель

МН – машинне навчання

МУ – музичні уподобання

ОТ – обробка текстів

ПП – персоналізований плейлист

РС – рекомендаційна система

ШІ – штучний інтелект

AI – Artificial Intelligence

DB – Database

GUI – Graphical User Interface

ML – Machine Learning

NLP – Natural Language Processing

TC – Text Classification

TP – Text Processing

ЗМІСТ

ВСТУП

У сучасному цифровому просторі музичні стрімінгові сервіси стали невід'ємною частиною повсякденної життя мільйонів користувачів. Стрімінговий сервіс - це онлайн-платформа, яка дозволяє прослуховувати музыку в режимі реального часу без необхідності її попереднього завантаження. Такі сервіси, як Spotify [1], Apple Music [2], Deezer чи YouTube Music [3], надають доступ до мільйонів композицій за підпискою або безкоштовно з рекламою. Попит на якісний музичний контент постійно зростає, що зумовлює потребу в ефективних інструментах персоналізації. Водночас, незважаючи на досягнення провідних платформ, більшість з них використовують закриті, непрозорі алгоритми, які не дозволяють гнучко налаштовувати логіку формування рекомендацій відповідно до індивідуальних потреб користувача.

На глобальному рівні спостерігається тенденція до впровадження систем, які не лише аналізують історію прослуховувань, а й враховують жанрові, мовні, емоційні параметри композицій, що дозволяє формувати контекстно-залежні добірки. Проте більшість доступних рішень не підтримують глибокої інтеграції з іншими системами або не надають можливості створення власних алгоритмів добору.

Актуальність теми обумовлена потребою в автономному модулі для музичного сервісу, який міг би використовувати методи штучного інтелекту та обробки мови для аналізу вподобань користувача, визначення жанру, настрою й мови треків, і на основі цього формувати персоналізовані музичні плейлисти. Такий підхід дозволить покращити якість рекомендацій, підвищити релевантність контенту.

Метою цієї роботи є розробка інформаційного модуля, який забезпечує автоматизований аналіз музичних уподобань користувача з урахуванням лінгвістичних та емоційних характеристик текстів пісень. Очікувана сфера застосування це музичні сервіси, інтелектуальні інформаційні системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області музичного сервісу

У сучасній цифровій середовищі музичні сервіси відіграють важливу роль у повсякденній житті користувачів, надаючи зручний доступ до мільйонів композицій зі всього світу. Такі сервіси працюють як централізовані платформи для зберігання, пошуку, прослуховування та персоналізації музичного контенту. Вони об'єднують у собі функціональність потокової передачі даних, урахування активності користувача, інструментів рекомендацій.

Структура музичного сервісу передбачає наявність кількох основних відділів, кожен з яких виконує свою роль у підтримці роботи всієї системи. Центральне місце власник сервісу, який відповідає за стратегічне управління платформою, укладання договорів з правовласниками та контроль за дотриманням авторських прав. Саме він ухвалює рішення про доступ до контенту та співпрацю з музичними лейблами.

Важливим є адміністративний відділ, який займається технічною підтримкою сервісу, додаванням нових музичних творів, оновленням інформації в базі даних та забезпеченням стабільної роботи платформи. Працівники цього відділу контролюють процес завантаження контенту, перевіряють його на відповідність вимогам та слідкують за актуальністю даних.

Відділ роботи з користувачами забезпечує контролює процес реєстрації, авторизації та підтримки клієнтів. Саме цей відділ відповідає за комфорт користувача при взаємодії з сервісом.

Аналітичний відділ відповідає за обробку історії прослуховувань, виявлення вподобань користувачів, аналіз поведінки та формування рекомендацій. На основі зібраної статистики він створює персоналізовані добірки треків, які відповідають смакам кожного слухача.

Загальна структура сервісу побудована так, щоб забезпечити повний цикл -

від надходження музичного твору до його прослуховування кінцевим користувачем, із дотриманням технічних, юридичних та аналітичних вимог.

Серед основних функцій музичного сервісу можна виділити:

− реєстрація та аутентифікація користувача створення облікового запису, вхід за допомогою логіна та пароля;

− завантаження музичного контенту адміністраторами;

− відображення каталогу композицій надання користувачеві доступу до структурованої бібліотеки треків;

− відтворення вибраного музичного контенту;

− облік прослуховувань;

− гарантування дотримання юридичних та авторських прав.

Музичний сервіс є джерелом інформації для нашого модуля - саме через нього виходять дані про історію прослуховувань користувача, його улюблених виконавців та треки. Завдяки цьому модуль може проаналізувати вподобання слухача, витягти потрібні дані для подальшої обробки а також використовувати них для створення і відображення нового персоналізованого плейлиста. Таким чином, наш модуль не дублює функціональність музичного сервісу, а наоборот - доповнює її, додаючи аналіз та кастомізацю музичного контенту.

1.2 Опис поточного стану процесу, що автоматизується

У поточних умовах формування музичних плейлистів у стрімінгових сервісах зазвичай виконується вручну або напівручним способом. Користувач самостійно обирає улюблені композиції, додає їх у списки відтворення, орієнтуючись на особистий смак, популярність треків, рекомендації від платформи або результати пошуку. Такий процес є трудомістким, суб’єктивним і не завжди враховує реальні музичні вподобання слухача.

Формування плейлисту включає кілька послідовних кроків таких як

відкриття стрімінгової платформи, ручний перегляд треків, їх прослуховування, сортування за смаком та додавання до окремого списку. У більшості випадків користувач орієнтується на загальні алгоритмічні рекомендації або наявні добірки без можливості впливати на критерії відбору. Також немає інструментів, які б враховували особисту статистику прослуховувань такі як переважаючі мови текстів, улюблені настрої чи найчастіше прослуховувані жанри.

Деяка часткова автоматизація вже реалізована у вигляді внутрішніх рекомендацій від сервісів Spotify [1], YouTube Music [2] ,Apple Music [3] тощо, однак вона закрита для зовнішньої кастомізації користувач не може переглядати логіку добору, а також не має можливості сформувати власну модель рекомендацій на основі відкритих даних або історії слухання.

Наявні стрімінгові сервіси не дозволяють інтегрувати зовнішні модулі персоналізації безпосередньо в процес формування добірки. У зв’язку з цим процес залишається ручним і неадаптивним, що призводить до зниження точності відповідності музики смакам слухача.

На рисунку 1.1 наведено IDEF0 діаграму, що ілюструє модель поточного процесу створення плейлисту в ручному режимі. Діаграма охоплює ключові етапи - від ініціації користувачем процесу пошуку музики до фінального створення списку треків. Як видно з моделі, у процесі практично відсутня аналітика, врахування історії прослуховувань, контексту чи інтелектуального аналізу даних.

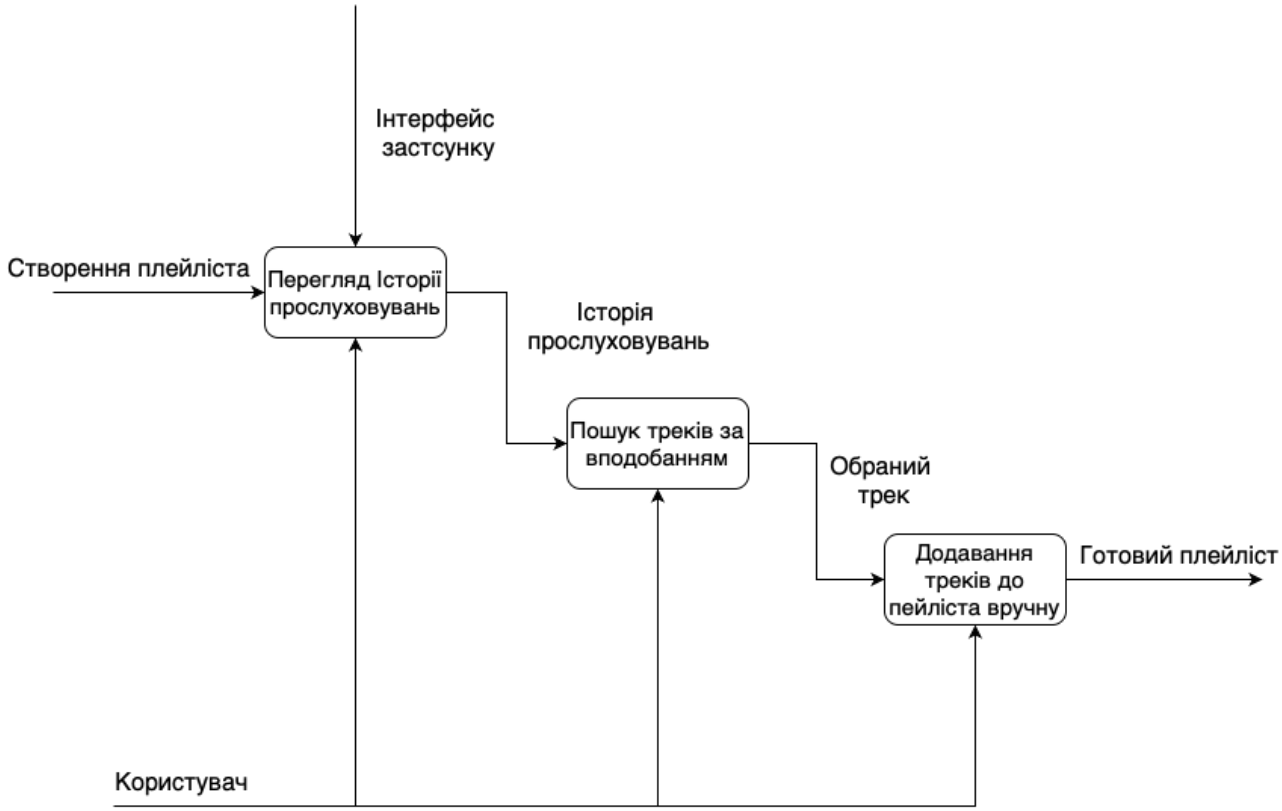

Рисунок 1.1 – Поточний процес створення плейлістів - діаграма IDEF0

Існуюча процедура потребує глибокої автоматизації з використанням методів аналізу даних та машинного навчання. Запропонований у дослідженні модуль має змінити підхід до формування плейлистів зробити його адаптивним і персоналізованим.

1.3 Проблеми й недоліки існуючого підходу

У процесі аналізу предметної області було виявлено низку проблем, що свідчать про недостатній рівень автоматизації при формуванні персоналізованих музичних плей-листів:

- користувач самостійно додає треки до списку, не маючи інструментів для автоматичного добору композицій на основі своїх вподобань;
- відсутність системи, яка б автоматично підбирала треки на основі

попередніх прослуховувань або жанрових вподобань, змушує користувача витрачати час на самостійний пошук;

− дані про взаємодію користувача з контентом не використовуються для формування рекомендацій;

− плей-листи не враховують емоційний стан або ситуаційний контекст користувач сам вирішує, яка музика йому підходить у певний момент;

− немає автоматизованого механізму, що групує композиції за жанром, темпом, звучанням тощо для створення збалансованих добірок;

− списки відтворення не оновлюються автоматично відповідно до змін у музичних смаках користувача або появи нових релевантних треків;

− плей-листи формуються локально, без автоматичного перенесення до стрімінгових платформ, що створює додаткові кроки для користувача;

Ці проблеми підтверджують необхідність розробки спеціалізованого модуля, який дозволить підвищити точність рекомендацій, зменшити навантаження на користувача та зробити систему більш адаптивною й ефективною.

1.4 Постановка задач дослідження

У сучасних музичних сервісах персоналізовані добірки відіграють ключову роль у взаємодії з користувачем. У межах кваліфікаційної роботи передбачено створення окремого модуля, що дозволяє автоматизувати аналіз музичних вподобань користувача на основі його поведінкових даних та формувати індивідуальні плей-листи без потреби у ручному втручанні.

Розробка модуля базується на ідеї покращення релевантності музичних добірок за рахунок глибшої обробки таких параметрів, як історія прослуховувань, частота взаємодії з треками, жанрові та ритмічні вподобання, тощо. Основна функція модуля - зменшити навантаження на користувача під час пошуку музики та забезпечити дійсно індивідуальний контент.

З технічного боку, реалізація передбачає:

− побудову системи збору та структуризації користувацьких даних;

− розробку алгоритмів, які зможуть виявляти закономірності в поведінці слухача;

− створення механізму автоматичного формування плей-листів на основі цих висновків;

− побудову інтеграції з API зовнішніх стрімінгових сервісів для передачі готових добірок;

− розробку базової системи візуалізації, що демонструє основні переваги користувача наприклад, домінуючі жанри або зміни в стилях протягом місяця.

Окрім технологічного блоку, очікується, що реалізація модуля позитивно вплине на загальний рівень взаємодії з інформаційною системою музичного сервісу. Йдеться про підвищення точності рекомендацій, скорочення часу на пошук треків, покращення зручності, адаптивність і, як наслідок, зростання користувацької лояльності.

Наше дослідження спрямоване не лише на реалізацію окремого функціонального компонента, а й на інтиграцію системи персоналізації у мизичні сервіси, яка відповідатиме іх вимогам і залишатиметься гнучкою для подальшого розвитку.

2 ОГЛЯД І АНАЛІЗ СУЧАСНОГО СТАНУ СИСТЕМ ПЕРСОНАЛІЗАЦІЇ МУЗИЧНОГО КОНТЕНТУ ТА МЕТОДІВ РЕКОМЕНДАЦІЇ НА ОСНОВІ ВПОДОБАНЬ КОРИСТУВАЧІВ

2.1 Огляд існуючих систем для створення музичних плейлістів

У сучасних умовах розвитку цифрових систем та стрімінгових сервісів проблема ефективної персоналізації музичного контенту набуває особливої актуальності.

Одним із найпопулярніших сервісів є Spotify [1], який використовує поєднання колаборативної фільтрації, аудіоаналізу та моделей машинного навчання. Рекомендації формуються на основі історії прослуховувань користувачів, а також характеристик треків (ритм, тональність, енергійність тощо) [9]. Через API доступна функція отримання тільки історії прослуховувань, однак немає доступу до внутрішніх механізмів побудови рекомендацій. Це унеможливлює повноцінне використання системи для побудови модулів для музичних систем з адаптивною логікою.

Youtube music [2] реалізує сценарний підхід до генерації рекомендацій «музика для відпочинку», «для концентрації», базуючись на поведінкових патернах інших слухачів [9]. Алгоритми побудовані на великих обсягах історичних даних, однак система не відкрита для кастомізації логіки або використання алгоритмів у інших системах. Аналогічну закритість демонструє Apple Music [3], де рекомендації також формуються автоматично, але зовнішні інструменти не мають доступу до цих данних.

Більш відкритою платформою є last.fm [4], яка реалізує механізм "scrobbling" - фіксації прослуховувань користувача. API дозволяє отримати профіль прослуховувань і загальні вподобання, однак якість рекомендацій обмежена тільки популярностю пісень, без глибокого аналізу емоційного чи мовного контексту. Іншим прикладом є Pandora [5], де застосовується music genome project - система ручної класифікації треків за сотнями параметрів.

Незважаючи на високу точність, сервіс повністю закритий для інтеграції.

Порівняння зазначених систем наведено в таблиці 2.1, де проаналізовано ключові характеристики кожного рішення, включаючи доступність, можливість налаштування логіки рекомендацій та рівень персоналізації.

Таблиця 2.1 - Порівняння систем

Система	Метод	Доступність	Контекст	Власна логіка	Обмеження
Spotify	Колаборативна	Частково	Так	Ні	Алгоритми закриті, відсутній відкритий API
Apple Music	Поведінкова	Ні	Частково	Ні	Відсутній відкритий API
YouTube Music	Сценарна	Ні	Частково	Ні	Інтеграція неможлива
Last.fm	Taste Profile	Так	Частково	Частково	Немає повного контролю
Pandora	Ручна класифікація	Ні	Так	Ні	Закрита, точна, але фіксована
Deezer	Гібридна	Частково	Так	Частково	Обмежені можливості налаштувань

Серед ключових недоліків наведених систем варто виділити:

- обмежений доступ до логіки формування рекомендацій;
- неможливість щодо експортування добірок у інші системи;
- відсутність можливості гнучко налаштовувати критерії добору;

- недоступність внутрішніх аналітичних інструментів для обробки даних;
- відсутність класифікації за настрієм, мовою, жанром тощо.

2.2 Обгрунтування необхідності розробки модуля для аналізу музичних вподобань та створення плейлістів

Аналіз сучасних стрімінгових платформ показує, що попри високу популярність таких систем, як Spotify [1], Apple Music [3], YouTube Music [2], Last.fm [4] та інші жодна з яких не забезпечує повної гнучкості, прозорості та адаптивності, необхідних для побудови незалежного модуля генерації персоналізованих плейлистів. Переважна більшість платформ має закриту архітектуру, обмежену відкритість або взагалі відсутній доступ до логіки формування рекомендацій. Це унеможливлює реалізацію власних алгоритмів та інтеграцію цих систем у музичний сервіс.

У сучасному інформаційному середовищі виникає потреба у відкритому рішенні, яке б дозволяло:

- автоматично формувати плейлисти на основі не лише жанрових, а й
- враховувати індивідуальну історію прослуховувань;
- підтримувати інтеграцію з музичними сервісами;
- реалізовувати власну логіку добору треків, незалежну від вбудованих механізмів закритих систем;

Розробка модуля дозволить подолати обмеження існуючих рішень, створити більш гнучку, настроювану та відкриту систему персоналізації.

3 ФОРМУЛЮВАННЯ ЗАВДАННЯ РОЗРОБКИ

3.1 Опис вимог до об'єкта розробки

3.1.1 Опис функціональних вимог

Визначення функціональних вимог до програмного модуля формує основу для розробки логіки, структури та взаємодії компонентів системи. В межах цього проекту модуль повинен реалізовувати повний цикл взаємодії з користувачем та стрімінговими платформами від авторизації до експорту згенерованого плейліста.

Система повинна забезпечувати аутентифікацію та авторизацію користувача через відповідну музичну платформу Spotify. Після успішної авторизації необхідно реалізувати здобуття історії прослуховувань, що включає список треків, улюблених виконавців, частоту взаємодії та ідентифікатори композицій. Ці дані надалі використовуються для побудови користувацького профілю.

Наступним етапом є отримання текстів музичних творів із відкритих сторонніх API на основі пари виконавця та назви пісні. У разі недоступності тексту трек позначається як неповний для аналізу.

На основі зібраного текстового матеріалу виконується ідентифікація параметрів музичного твору - жанру, настрою та мови композиції. Для цього використовують попередньо натреновані моделі штучного інтелекту.

Далі система повинна забезпечити генерацію персоналізованого плейліста, у якому треки обираються відповідно до виявлених улюблених користувачів.

Після автоматичної генерації передбачається можливість ручного коригування параметрів добірки: змінити ваги характеристик жанру, настрою, мови, популярностіь, переглянути список.

Завершальним функціональним етапом є експорт сформованого плейліста до музичного сервісу слухача.

3.1.2 Опис нефункціональних вимог

Перед тим як запускати модуль у реальне використання, важливо не лише визначити, що саме він повинен робити, але й як добре і зручно він це робитиме. Саме для цього формуються нефункціональні вимоги - вони описують такі речі, як швидкість роботи, зручність для користувача, безпека, стабільність і можливість у майбутньому розширити або змінити систему. Це допомагає зробити сервіс не просто функціональним, а надійним, безпечним і гнучким.

По-перше, система має працювати стабільно, навіть якщо щось пішло не так - наприклад, немає інтернету або не відповідає сторонній сервіс. У таких випадках додаток не повинен закриватися чи зависати.

Також важлива швидкість. Система повинна швидко формувати плейлист - бажано не довше 30 секунд при обробці до 500 треків. А всі запити до зовнішніх сервісів, наприклад Spotify або Deezer, повинні виконуватись не довше ніж 10 секунд.

Ще одна важлива річ - можливість масштабування. Якщо користувачів стане більше або зросте обсяг даних, система має підтримувати перехід з простої бази даних SQLite на більш потужну, як PostgreSQL.Flask [6]-сервер, на якому побудований наш модуль, повинен легко запускатися у багатопоточному режимі і працювати стабільно навіть при великому навантаженні.

Безпека - ще один критичний момент. Усі особисті дані користувача наприклад, токени доступу повинні бути захищені й не зберігатися у відкритому вигляді. Доступ до музичних сервісів має відбуватися через безпечну авторизацію OAuth у музичному сервісі, а жодні паролі не повинні передаватися чи зберігатися.

Модуль повинен бути сумісним із різними платформами - запускатися як на Windows, так і на macOS або Linux. Крім того, система має працювати з Python 3.10 і новішими версіями.

Отже, ці нефункціональні вимоги потрібні, щоб система працювала

надійно, швидко, безпечно і зручно, і щоб її можна було легко підтримувати та вдосконалювати в майбутньому.

3.2 Обґрунтування мети і критеріїв ефективності об'єкта розробки

Метою створення модуля аналізу музичних уподобань користувача є реалізація модуля, який здатен формувати індивідуальні добірки треків. Це особливо важливо для музичних сервісів, які важливо+ мати власний алгоритм рекомендацій.

З точки зору виробничо-господарської доцільності, розробка дозволяє створити систему для аналізу вподобань користувача у сфері цифрових медіа. Такий модуль може застосовуватися у власних музичних сервісах та бути додатковим інструментом підвищення залученості аудиторії. Оскільки більшість сучасних сервісів концентруються на однотипних, неадаптивних рекомендаціях, наш підхід орієнтується на якість персоналізації, а не тільки на кількість взаємодій.

З науково-технічної точки зору, система демонструє практичне використання NLP для аналізу текстів пісень і zero-shot класифікації для визначення настрою. Ми використовуємо модель BART-Large-MNLI [8] для визначення емоційного тону композицій, що демонструє гнучкість застосування універсальних моделей. Крім того, аналіз ведеться на основі результатів із двох музичних сервісів Spotify і Deezer, що дозволяє створити об'єктивніший профіль користувача, ніж у випадках, коли використовується тільки одна джерело даних.

Економічна ефективність нашого модуля полягає у його відкритості, підвищених можливостей до повторного використанні та мінімальних вимогах до розгортання. Ми використовуємо легковаговий сервер, що забезпечує API-комунікацію між клієнтом і серверною частиною. База даних реалізована на SQLite, що дозволяє швидкий старт без витрат на інфраструктуру. При

масштабуванні модуль може бути адаптований під інші СУБД, а сервер замінений на іншу більш продуктивну технологію.

Критерії ефективності об'єкта розробки:

− персоналізація рекомендацій: треки у плейлисті повинні відповідати реально виявленим перевагам користувача, з урахуванням не лише жанру, але й мови і настрою;

− надійність збереження результатів: усі обчислення та дані мають фіксуватись у базі даних;

− адаптивність алгоритмів: зміни в поведінці користувача повинні відображатися у нових плейлистах без необхідності ручного втручання;

− можливість експорту: результати повинні бути доступні для експорту як у вигляді локального файлу, так і на музичний сервіс;

− гнучкість логіки добору користувач або розробник повинен мати можливість змінювати критерії генерації без втручання в код через інтерфейс.

Обмеження розробки:

− музичні сервіси мають обмеження на доступ до повної історії прослуховувань або створення плейлистів, що унеможливлює повну автоматизацію без додаткових дозволів;

− у моделях класифікації результати іноді можуть бути нечіткими або контекстно неадаптивними, особливо для пісень без тексту;

− обсяг даних обмежено базами SQLite, що знижує ефективність при великій кількості користувачів, тому доведеться масштабувати на більш потужні СУБД при реальному навантаженні.

4 ОПИС АРХІТЕКТУРИ ОБ'ЄКТА РОЗРОБКИ НА РІВНІ ФУНКЦІЙ

4.1 Опис архітектури системи

Цей підрозділ присвячений аналізу архітектури розроблюваної інформаційної системи, що є критично важливим етапом у процесі проєктування. Архітектура визначає, як саме організована взаємодія між усіма компонентами системи - користувачами, зовнішніми модулями, адміністративним персоналом та іншими сторонами. Вона візуалізує потоки даних, розподіл обов’язків та межі відповідальності кожного учасника системи.

Центральним об’єктом архітектури є інформаційна система "Музичного сервісу", яка виконує роль основного обробника, зберігача й маршрутизатора даних. Вона об'єднує всі зовнішні й внутрішні джерела інформації в єдиний функціональний комплекс.

Одним із ключових учасників є слухач який є користувачем сервісу, який через свій інтерфейс надсилає до системи:

− дані для авторизації, щоб підтвердити свою особу або створити обліковий запис;

− вихідні параметри для формування добірки як назва добірки чи тощо;

− запити на відтворення композицій або всієї добірки, що ініціюють відповідні дії в системі задля відтворення творів.

У відповідь система надає слухачу:

− персоналізовану добірку треків, сформовану відповідно до його вподобань;

− каталог музичних творів, що дає можливість шукати й обирати треки самостійно;

− плейліст, створений на музичній платформі, з можливістю миттєвого прослуховування.

Для реалізації аналітики та генерації плейлистів в системі передбачено «Модуль аналізу та генерації музичних плейлистів». Цей модуль не взаємодіє з

користувачем напряму, але працює з даними, які надходять від ІС "Музичного сервісу". Система передає йому історію прослуховувань слухача, на основі чого модуль виконує обробку та повертає сформований плейліст, що враховує жанр, мову, настрій і популярність.

Окремо в архітектурі представлено власника музичного сервісу. Його участь полягає у правовому врегулюванні доступу до творів. Він надає до системи дані для оформлення договору, на основі яких за допомогою ІС формується офіційний договір на використання музичних творів, що гарантує законне розміщення контенту та надання стімінгових послух.

Також важливу роль виконує адміністратор - представник технічного персоналу, який підтримує актуальність бази треків модуля. Він надсилає до системи запити на завантаження файлів нових музичних творів, а також повну інформацію про них наприклад назва, автор, тощо. Система у відповідь генерує повідомлення про результат завантаження, яке може містити підтвердження або опис помилки.

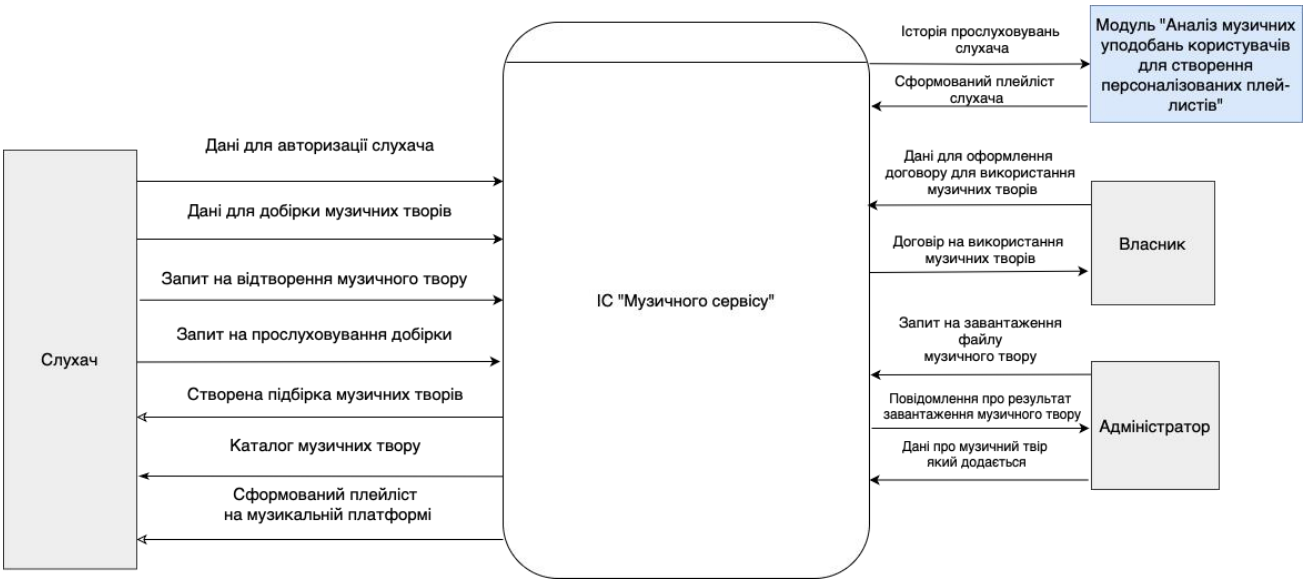

Рисунок 4.1 – Контекст на діаграма потоків даних ІС "Музичного сервісу"

Діаграма потоків даних першого рівня декомпозиції інформаційної системи

"Музичний сервіс" яка наведена на рисунку 4.2 деталізує взаємодію між функціональними модулями, базою даних та зовнішніми учасниками - такими як слухач, адміністратор і власник музичного сервісу. Ця модель розкриває внутрішню логіку ІС а саме обробки запитів, доступу до інформації та управління музичними творами й обліковими записами.

Центральною функціональною ланкою виступає ІС "Музичний сервіс", яка підтримується за рахунок взаємодії з базою даних та численними модулями, кожен з яких відповідає за окремий процес. Одним із ключових модулів є модуль аналізу та генерації музичних плейлистів, який отримує з функції API-експорту та імпорту даних історію прослуховувань користувача. На основі цієї інформації модуль формує персоналізований плейліст, що передається назад через API для подальшого використання. Сама функція API звертається до бази даних, щоб отримати історію прослуховувань та дані про користувача, з якими працює аналітичний модуль.

Окремий блок архітектури пов’язаний з власником музичного контенту, який надсилає дані для оформлення договору до модуля отримання та обліку юридичних прав. Цей модуль опрацьовує дані, реєструє договір у системі й зберігає інформацію до бази даних у вигляді даних про юридичні права на твір. У відповідь власник отримує офіційний договір на використання музичних творів, що юридично закріплює співпрацю. Слухач взаємодіє з кількома модулями. На етапі входу або реєстрації він надсилає дані для автентифікації до модуля реєстрації та авторизації, який, у свою чергу, зберігає дані про користувача в БД. Для перевірки коректності модуль також отримує з БД дані для верифікації, після чого повертає слухачу підтвердження про автентифікацію.

Після входу користувач може відтворювати музику через відповідний модуль. Для цього він надсилає запит на відтворення музичного твору, а система у відповідь повертає відтворений аудіотрек. Паралельно модуль надсилає в базу дані про факт прослуховування, які надалі використовуються для побудови рекомендацій.

Також слухач може ініціювати формування персональної добірки через

модуль формування каталогу та добірки музичних творів. Він надсилає дані про вподобанного йому музичного твору, а у відповідь отримує каталог доступних треків і саму підбірку. Для цього модуль звертається до БД, отримуючи дані про жанри та наявні твори, а результати - сформовану підбірку зберігає назад у базу.

Адміністратор бере участь у розширенні каталогу треків. Він надсилає до модуля додавання нового музичного твору як запит на завантаження файлу, так і дані про твір такі як назва, автор, жанр тощо. У відповідь модуль повертає повідомлення про результат завантаження. Для коректного додавання інформації модуль передає до бази даних файл музичного твору та його опис, а також отримує з бази список наявних жанрів та авторів, щоб уникнути помилок.

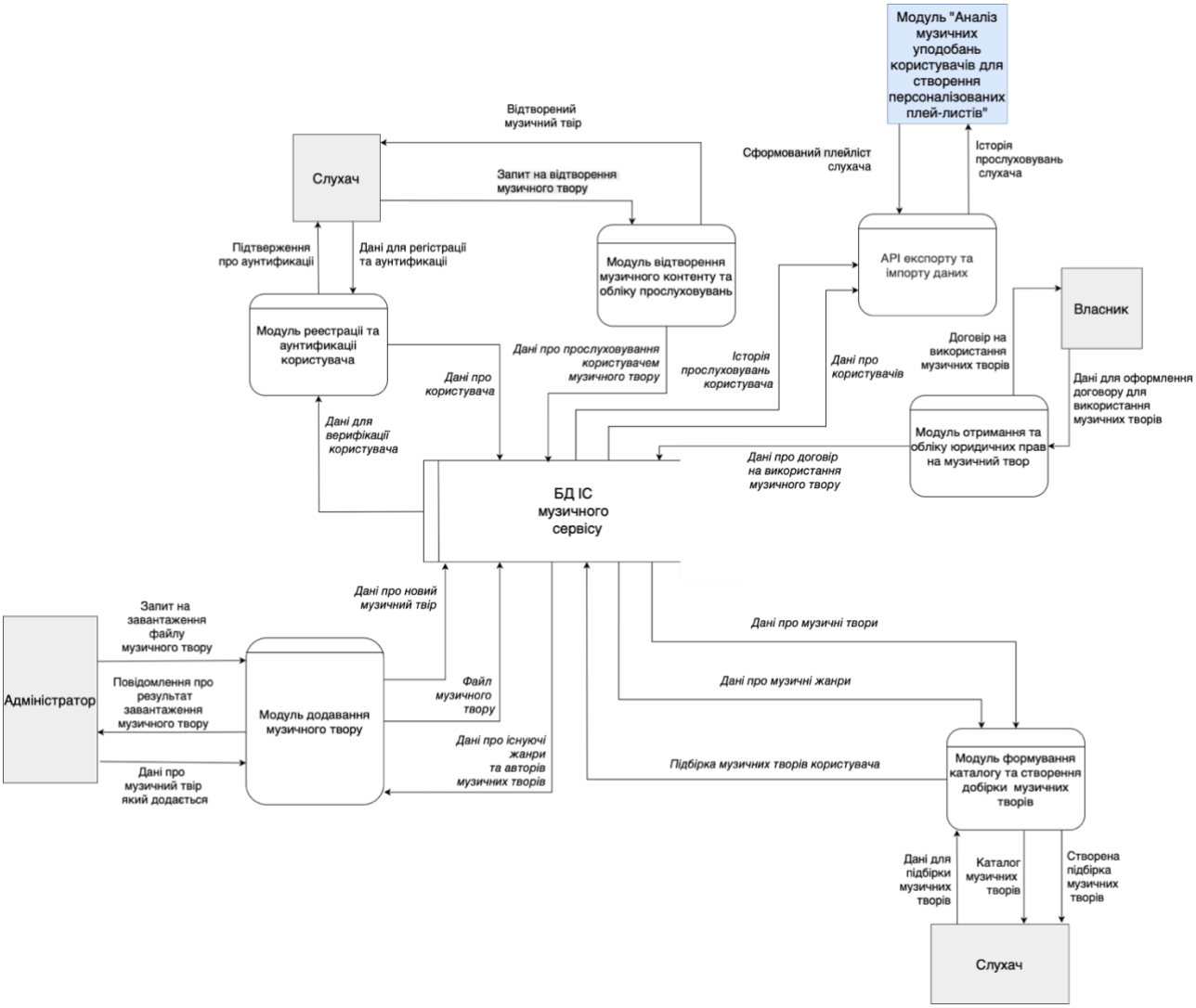

Рисунок 4.2 – Діаграма потоків даних ІС "Музичний сервіс" першого рівня декомпозиції

Діаграма потоків даних першого рівня декомпозиції є логічним продовженням загальної архітектурної моделі ІС "Музичного сервісу". Якщо попередня діаграма демонструвала ролі користувачів та загальні інформаційні взаємодії на високому рівні, то дана діаграма деталізує внутрішні процеси, пов’язані з отриманням, обробкою, зберіганням і передачею даних у межах модулю аналізу та генерації музичних плейлистів. Це дозволяє краще зрозуміти, як саме працюють ключові функції системи та як інформація переміщується між компонентами.

Зверху діаграми зображена ІС "Музичний сервіс", яка надсилає в процес "Імпорт та експорт даних щодо історії прослуховувань користувача" такі дані: перелік музичних творів, історію прослуховувань і інформацію про користувача. Цей процес виконує імпорт історії прослуховувань і єкспорт персоналізованих плейлистів. У результаті роботи процесу до системи повертається сформований плейліст, який потім може бути переданий користувачу або збережений.

Цей же процес, пов'язаний з історією прослуховувань, передає дані про прослуховування до бази даних ІС, а у відповідь отримує інформацію про користувачів, яка потрібна для подальшої обробки й персоналізації.

Паралельно з цим у системі діє користувач - Слухач, який взаємодіє з функцією "Генерування плейлісту". Він надсилає свої вподобання, а система у відповідь формує та повертає йому згенерований плейліст. У свою чергу, ця функція також надсилає дані про згенерований плейліст до бази даних, щоб зберегти його для подальшого використання. З бази функція отримує перелік музичних творів та вподобання користувача, що дозволяє формувати плейлист на основі актуальних і повних даних. Крім того, ідентифікатор згенерованого плейлісту передається до процесу "Імпорт та експорт історії", щоб зв’язати згенерований результат з єкспортом плейлісту до музичного сервісу.

Окремо діаграма охоплює взаємодію із зовнішнім сервісом - Lyrics.ovh, що використовується для обробки текстів пісень. Цей сервіс отримує запит на пошук тексту музичного твору від процесу "Отримання тексту музичного твору", і у відповідь надає результат пошуку тексту. Потім цей текст передається до

наступного процесу "Ідентифікація параметрів вподобань музичних творів", який на його основі визначає жанр, мову, настрій та інші параметри. У відповідь ця функція повертає назву та автора музичного твору, уточнюючи метадані.

Результати ідентифікації, тобто параметри вподобань музичних творів, передаються до бази даних для збереження. Зі свого боку, база надає функції "Ідентифікації" історію прослуховувань користувача, які можуть вплинути на точність класифікації.

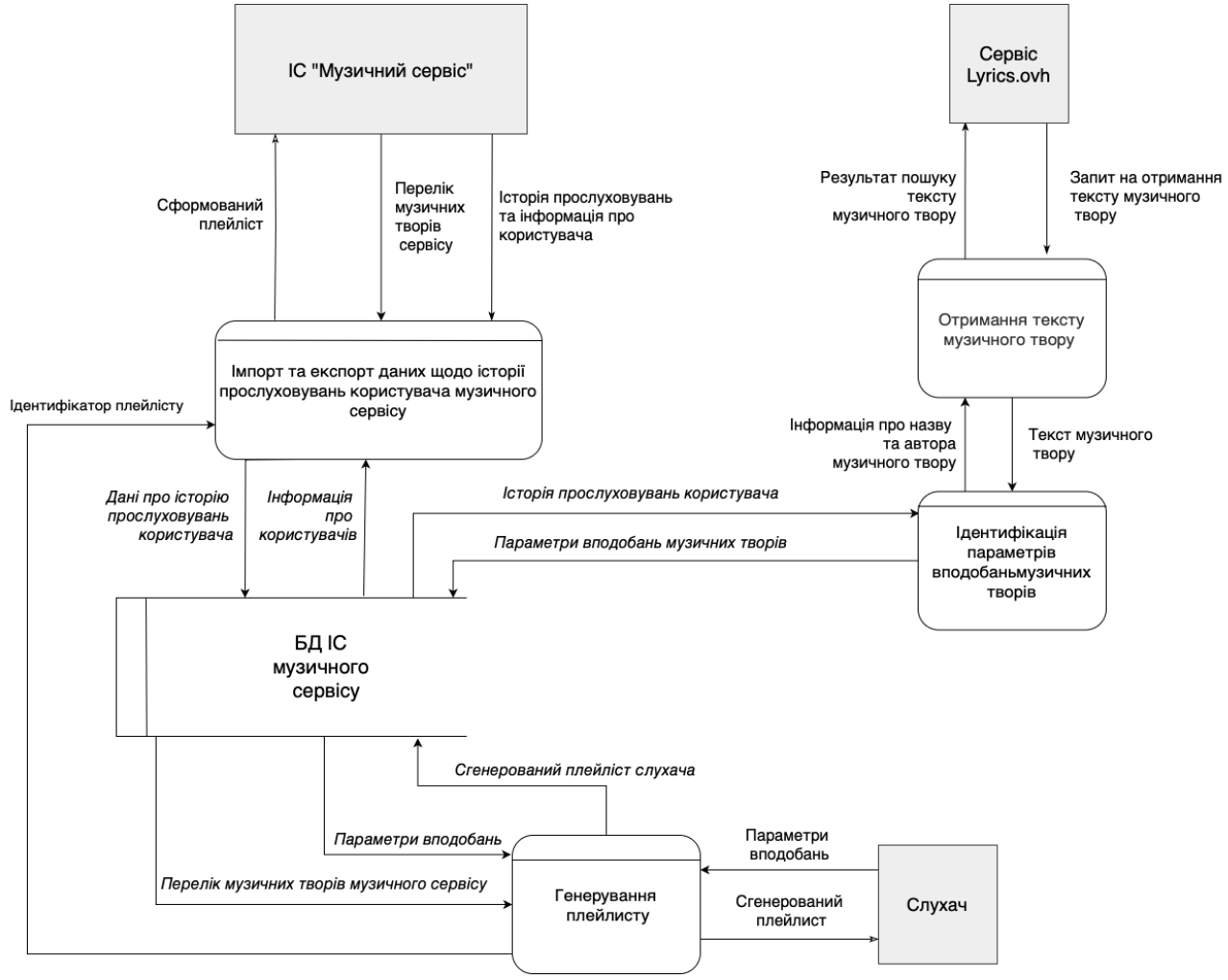

Рисунок 4.3 – Діаграма потоків даних модулю «Аналіз музичних уподобань користувачів для створення персоналізованих плей-листів» другого рівня декомпозиції

У результаті аналізу трьох діаграм, можна зробити висновок, що побудована система має чітко структуровану та логічно організовану архітектуру,

яка забезпечує ефективну взаємодію між усіма учасниками такими як слухач, адміністратор, власник контенту та аналітичні модулями.

5 РОЗРОБКА ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ІНФОРМАЦІЙНОЇ ЗАБЕЗПЕЧУЮЧОЇ СИСТЕМИ

5.1 Загальна характеристика інформаційного забезпечення

Інформаційне забезпечення це один з ключових компонентів будь-якої інформаційної системи, оскільки воно визначає, які саме дані будуть зберігатися, оброблятися, передаватися між модулями та виводитися користувачу. У межах проєкту інформаційне забезпечення виконує роль центрального сховища даних про треки, характеристики пісень та результати класифікації.

Основу інформаційного забезпечення модуля становить реляційна база даних, яка містить відомості про музичні треки, зібрані з платформ через відповідні API або класифіковані методами машинного навчання. Для кожного треку зберігаються такі параметри, як назва, виконавець, рейтинг популярності та кількість прослуховувань. Окрім цього, в базі зберігаються результати NLP-аналізу визначення настрою, жанру та мови.

Важливим елементом інформаційного забезпечення є також збереження вподобань користувача тобто зведеної статистики про те, які жанри, мови та настрої переважають у прослуханому контенті. Ці дані використовуються для генерації персоналізованого плейлисту.

З технічної точки зору, система реалізована на базі SQLite. Усі взаємодії з базою даних реалізовано через ORM SQLAlchemy [8], що дозволяє зручно працювати з об'єктами Python без написання SQL коду вручну.

Крім того, база даних тісно інтегрована з backend-сервером. Сервер відповідає за отримання даних з API, їх обробку, класифікацію треків, оновлення бази та формування запитів до даних для побудови рекомендацій.

5.2 Обґрунтування вибору СУБД і методу організації даних

Для ефективної реалізації модуля аналізу музичних вподобань користувачів було необхідно обрати таку систему управління базами даних, яка забезпечує швидкий доступ до інформації, простоту у розгортанні, підтримку реляційної структури та сумісність із Python-середовищем. Було розглянуто декілька альтернативних варіантів, серед яких PostgreSQL, MySQL, MongoDB та SQLite порівняльний аналіз наведено у таблиці 5.1.

Таблиця 5.1 - Порівняння популярних СУБД:

Характеристика	SQLite	PostgreSQL	MySQL	MongoDB
Тип СУБД	Реляційна, вбудована	Реляційна, серверна	Реляційна, серверна	Документо-орієнтована, NoSQL
Встановлення	Не потребує, вбудована	Потребує налаштування сервера	Потребує налаштування сервера	Потребує окремого запуску
Швидкість роботи	Висока для локальних задач	Висока на великих обсягах даних	Висока при читанні	Висока на неструктурованих даних
Підтримка транзакцій	Так	Так	Так	Частково
Підтримка ORM (SQLAlchemy)	Так	Так	Так	Частково (через MongoEngine)
Підходить локальних застосунків	Так	Складніше у налаштуванні	Складніше у налаштуванні	Надлишкове для простих задач

Після аналізу було обрано SQLite як основну СУБД для реалізації системи. Головними перевагами цього вибору є:

− простота інтеграції з мовою програмування Python і сумісність з фреймворком сервера без потреби в додаткових сервісах.

− легка вага та автономність, що дозволяє працювати без окремого серверного процесу достатньо одного файлу бази даних.

− підтримка транзакцій, ключів, індексів, реляційних зв'язків тобто всього, що потрібно для реалізації складної структури даних.

− підтримка ORM SQLAlchemy [8], яка дозволяє ефективно працювати з базою в об'єктно-орієнтованому стилі.

− ідеально підходить для локального зберігання даних, де не передбачається одночасний доступ великої кількості користувачів.

Хоча такі СУБД як PostgreSQL чи MySQL могли б бути доцільними для розгортання у промислових масштабах або у хмарному середовищі, для даного проєкту їх використання було б надмірним і додатково ускладнювало б архітектуру.

5.3Логічна модель даних

Логічна модель даних описує, які саме сутності існують у системі, які атрибути вони мають та як пов’язані між собою. Вона не залежить від конкретної реалізації бази даних або мови запитів. Логічна модель дозволяє зрозуміти структуру інформації та логіку взаємодії між частинами системи ще до початку технічної реалізації. У нашому випадку логічна модель зображена на рисунку 5.1 відображає, як побудована система зберігання даних для аналізу музичних вподобань користувача та формування персоналізованих плейлистів.

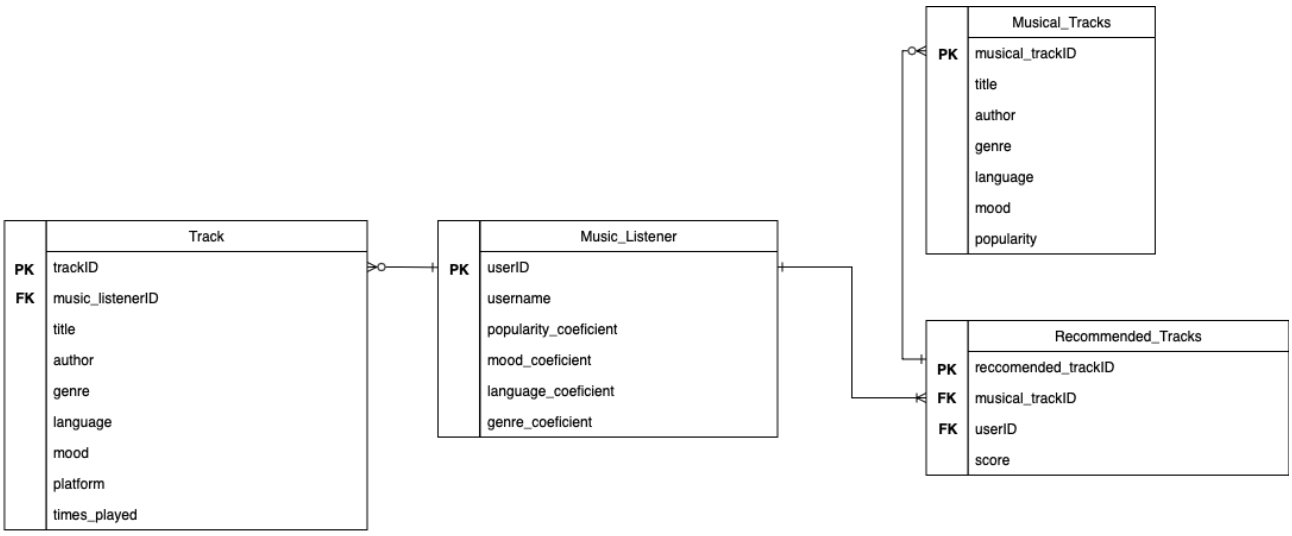

Рисунок 5.1 – Логічна модель даних системи.

Основна сутність - Music_Listener (користувач), яка зберігає ім’я та коефіцієнти жанру, мови, настрою і популярності. Ці коефіцієнти формуються на основі історії прослуховувань.

З користувачем пов’язана сутність Track - це всі треки, які він слухав. Вони містять інформацію про композицію, платформу, а також кількість прослуховувань.

Каталог усіх доступних треків представлений сутністю Musical_Tracks. Це незалежна база, з якої обираються треки для рекомендацій.

Сутність Recommended_Tracks зберігає результати рекомендацій - тобто треки, які система підібрала користувачу, разом з оцінкою відповідності (score).

Наша логічна модель бази даних охоплює: з одного боку - користувача і його взаємодію з системою (через прослуховування), з іншого - треки, які можна рекомендувати, та результати аналізу у вигляді згенерованих пропозицій.

5.4 Фізична модель даних

Фізична модель даних є важливою частиною розробки будь-якої

інформаційної системи, оскільки вона визначає, як саме дані будуть зберігатися в базі, які таблиці будуть використовуватись, як вони пов’язані між собою, які типи даних застосовуються. Це дозволяє забезпечити узгодженість, цілісність і ефективну обробку інформації під час роботи системи. У нашому випадку фізична модель яка наведена на рисунку 5.2 даних дає змогу організовано зберігати інформацію про користувачів, прослухані треки, загальний каталог треків і систему рекомендацій.

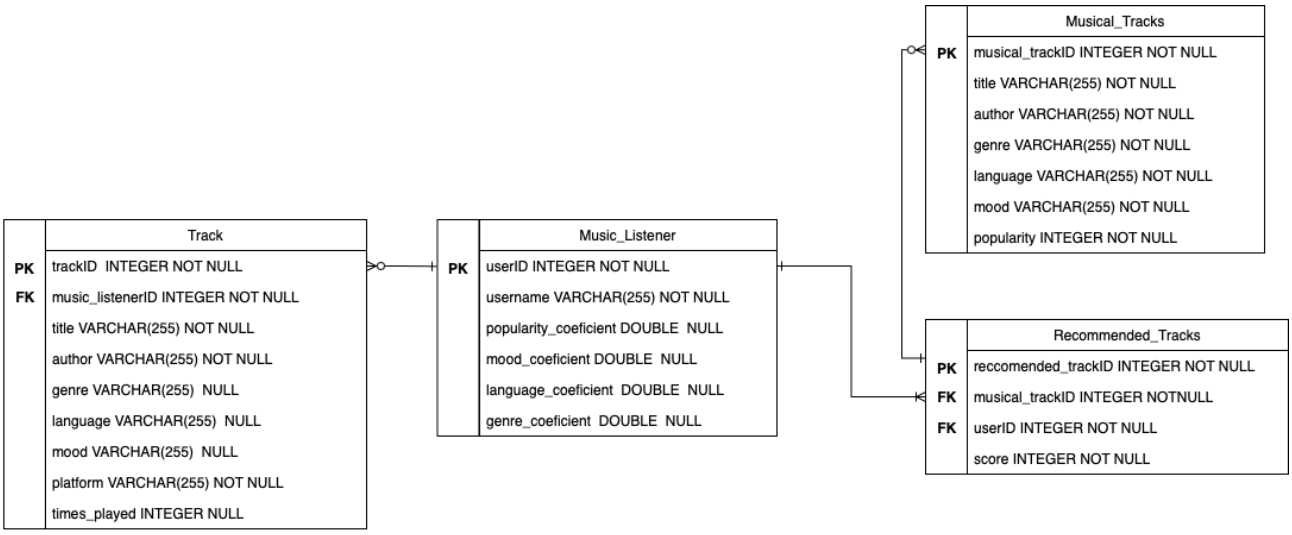

Рисунок 5.2 – Фізична модель даних системи у вигляді ER-діаграми

У нашем модулі у БД використовується чотири таблиці: Track, Music_Listener, Musical_Tracks та Recommended_Tracks. Кожна з них виконує чітко визначену функцію, а разом вони утворюють цілісну структуру, яка дозволяє аналізувати вподобання користувача, будувати профіль слухача та формувати індивідуальні музичні рекомендації. Взаємозв’язки між таблицями реалізовані за допомогою зовнішніх ключів.

Таблиця Track скрипт створення якої зображено у лістингу 5.1- зберігає інформацію про всі треки, які були прослухані конкретним користувачем. Вона містить такі поля, як назва композиції (title), автор (author), жанр, мова, настрій, платформа тобто звідки саме був прослуханий трек - наприклад, Spotify чи Deezer, а також кількість прослуховувань. Ця таблиця пов’язана з таблицею

Music_Listener через поле music_listenerID, яке виступає зовнішнім ключем. Таким чином, кожен запис у таблиці Track відповідає конкретному користувачу.

Лістинг 5.1 – Формування таблиці Track

```
CREATE TABLE Track (
    trackID INTEGER PRIMARY KEY,
    music_listenerID INTEGER NOT NULL,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255),
    genre VARCHAR(255),
    language VARCHAR(255),
    mood VARCHAR(255),
    platform VARCHAR(255) NOT NULL,
    times_played INTEGER,
    FOREIGN KEY (music_listenerID) REFERENCES
Music_Listener(userID) ON DELETE CASCADE
);
```

Таблиця Music_Listener процесс створення якої зображно у лістингу 5.2 є центральною у моделі - вона зберігає інформацію про кожного користувача: його унікальний ідентифікатор (userID), ім’я користувача (username), а також коефіцієнти вподобань за жанром, настроєм, мовою та популярністю. Ці коефіцієнти формуються на основі аналізу історії прослуховувань і використовуються для подальшого формування персоналізованих рекомендацій. Таблиця має зв’язки з двома іншими - Track та Recommended_Tracks.

Лістинг 5.2 – Формування таблиці Music_Listener

```
CREATE TABLE Music_Listener (
    userID INTEGER PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    popularity_coeficient DOUBLE,
    mood_coeficient DOUBLE,
    language_coeficient DOUBLE,
    genre_coeficient DOUBLE
);
```

Таблиця Musical_Tracks створення якої зображено на лістингу 5.3 виконує роль основного каталогу всіх доступних треків, з яких можуть формуватися рекомендації. Вона містить поля з інформацією про композицію: назву, автора,

жанр, мову, настрій та популярність. Ця таблиця не пов’язана безпосередньо з користувачами, але її дані активно використовуються під час генерації плейлистів, оскільки саме з неї система добирає релевантні треки.

Лістинг 5.3 – Формування таблиці Musical_Tracks

```
CREATE TABLE Musical_Tracks (
    musical_trackID INTEGER PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255),
    genre VARCHAR(255),
    language VARCHAR(255),
    mood VARCHAR(255),
    popularity INTEGER
);
```

Таблиця Recommended_Tracks зберігає результати роботи рекомендаційного модуля. Кожен запис у цій таблиці відповідає певному треку, який був рекомендований конкретному користувачу, разом із полем score, що вказує рівень відповідності треку до вподобань користувача. Таблиця має зовнішні ключі як до Musical_Tracks, так і до Music_Listener, що дозволяє точно визначити, кому була запропонована композиція і на основі чого.

Лістинг 5.4 – Формування таблиці Recommended_Tracks

```
CREATE TABLE Recommended_Tracks (
    recommended_trackID INTEGER PRIMARY KEY,
    musical_trackID INTEGER NOT NULL,
    userID INTEGER NOT NULL,
    score INTEGER NOT NULL,
    FOREIGN KEY (musical_trackID) REFERENCES
Musical_Tracks(musical_trackID) ON DELETE CASCADE,
    FOREIGN KEY (userID) REFERENCES Music_Listener(userID) ON
DELETE CASCADE
);
```

Така структура фізичної моделі забезпечує зрозумілу, масштабовану й ефективну основу для збереження і обробки даних у системі персоналізованих музичних рекомендацій.

6 РОЗРОБКА Й ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ МАТЕМАТИЧНОЇ ЗАБЕЗПЕЧУЮЧОЇ СИСТЕМИ

6.1 Мета створення математичного забезпечення

У нашій системі математичне забезпечення відіграє ключову роль, оскільки дозволяє автоматично аналізувати вподобання користувача та на основі цього формувати персоналізовані музичні плейлисти. Без чітко визначених математичних моделей система не змогла б об'єктивно порівнювати треки між собою, оцінювати їх відповідність уподобанням слухача чи визначати, які саме композиції будуть для нього найбільш підходящими.

Основна мета математичного забезпечення полягає у побудові моделей, які можуть кількісно оцінити характеристики кожного треку такі як жанр, настрій, мова виконання і зіставити їх із зібраним профілем користувача. Наприклад, якщо користувач найчастіше слухає спокійну музику англійською мовою в жанрі Pop, система повинна вміти зрозуміти ці патерни поведінки, зберегти їх як набір числових значень і надалі порівнювати з іншими композиціями.

Крім того, математичні моделі використовуються для класифікації треків на основі тексту пісень за допомогою методів машинного навчання. Також із застосуванням вагових коефіцієнтів система визначає, наскільки добре конкретна пісня відповідає перевагам користувача. У підсумку, саме математичне забезпечення дозволяє реалізувати логіку рекомендацій.

6.2 Опис математичної моделі вподобань користувача

у розроблюваному модулі вподобання користувача формуються автоматично на основі історії прослуховувань. Коли користувач слухає музику через Spotify або Deezer, система фіксує жанр, мову та настрій кожного треку,

який було відтворено. Ця інформація накопичується й дозволяє виявити, які типи музики переважають у його вподобаннях.

Щоб перевести ці вподобання у числову форму, ми використовуємо просту математичну модель на основі ймовірностей. Для кожного жанру, мови та настрою ми обчислюємо відсоткове співвідношення - тобто наскільки часто цей параметр зустрічається серед усіх прослуханих треків. Наприклад, якщо користувач прослухав 100 треків, із яких 30 - у жанрі pop, то вага жанру pop становитиме:

$$\mathrm{Pgenre(g)} = \frac{\mathrm{count_g}}{\mathrm{total}} = \frac{100}{30} = 0.3,$$

де Pgenre(g) – ймовірність або вага жанру gg у вподобаннях користувача;

countg - кількість треків у жанрі, які були прослухані користувачем;

total - загальна кількість усіх треків, прослуханих користувачем.

Коефіцієнт Pgenre(g) показує, яку частину від загальної музичної активності займає певний жанр. Аналогічно обчислюються ймовірності для мов та настроїв, формуючи повноцінний профіль уподобань користувача.

6.3 Модель обчислення релевантності треку

У процесі формування персоналізованого плейлиста важливо оцінити, наскільки кожен трек відповідає вподобанням конкретного слухача. Для цього використовується математична модель обчислення релевантності, яка дозволяє визначити «цінність» треку з точки зору жанрових, емоційних та мовних вподобань, а також з урахуванням його популярності.

Релевантність треку T обчислюється за такою формулою:

$$Score(T) = \omega_1 \mathrm{P_{genre}}(T) + \omega_2 \mathrm{P_{mood}}(\mathrm{T}) + \omega_3 \mathrm{P_{language}}(\mathrm{T}) + \omega_4 \mathrm{P_{popularity}}(\mathrm{T}),$$

де $P_{genre}(T)$ - відповідність жанру треку до жанрових уподобань користувача;

$P_{mood}(T)$ - відповідність настрою треку до переважних емоцій користувача;

$P_{language}(T)$ - відповідність мови треку до мовних уподобань;

$P_{popularity}(T)$ - нормалізоване значення популярності треку серед інших;

$\omega_1, \omega_2, \omega_3, \omega_4$ - вагові коефіцієнти, які визначають значущість кожного фактору в загальному результаті.

Усі вагові коефіцієнти слухач може корегувати самостійно, використовуючи програмний інтерфейс, після чого програма обчислює релевантність кожного треку з бази даних усіх пісень після чого сортує їх від найбільшого до найменшого тим самим пісня с найбільшим score найвірогідніше сподобається користувачу та попаде в рекомендований плей-ліст.

6.4 Використання NLP-моделей у класифікації

У нашій системі аналізу музичних вподобань ми використовуємо методи машинного навчання (ML), які є складовою частиною штучного інтелекту. Машинне навчання дозволяє комп'ютеру самостійно знаходити закономірності в даних і приймати рішення без жорстко прописаних правил. Завдяки цьому наша система може автоматично розуміти, до якого жанру належить пісня, який у неї настрій, якою мовою вона написана і на основі цього будувати персоналізовані рекомендації для кожного користувача.

Для класифікації жанру, настрою ми використовуємо zero-shot класифікацію - метод, який дозволяє моделі обирати правильну категорію серед запропонованих, навіть якщо вона раніше не бачила прикладів цих категорій. Наприклад, якщо ми даємо пісню, в якій модель раніше не зустрічала тег «sad», вона все одно може визначити ймовірність, що ця пісня має сумний настрій. Це особливо корисно, оскільки музика постійно змінюється, і не завжди можливо мати готові навчальні дані для кожної нової категорії. Ці моделі працюють на

основі великих трансформерів, таких як BART(Bidirectional and Auto-Regressive Transformers) [8], які навчені розуміти зв'язки між словами, фразами та сенсом тексту.

Математично, задача формалізується як багатокласова класифікація, де обирається найбільш ймовірна мітка з множини можливих. Наприклад, для визначення настрою треку T із текстом lyrics використовується формула 6.1:

$$mood(T) = \arg\max_{m \in M} Prob(m \mid lyrics), \quad (6.1)$$

де M - множина можливих міток настрою (наприклад, "happy", "sad", "calm", "energetic");

$Prob(m \mid lyrics)$ - ймовірність того, що текст належить до настрою m;

arg max - оператор, що обирає мітку з найвищою ймовірністю.

Результатом розрахунків є список ймовірностей по кожному класу, який наведено в таблиці 6.1.

Таблиця 6.1 – Вирогідності визначення настрою тексту пісень

Label	Probability
happy	0.65
sad	0.20
aggressive	0.10
romantic	0.05

Подібні формули (6.2-6.3) використовуються і для класифікації жанру та мови:

$$genre(T) = \arg\max_{g \in G} Prob(g \mid lyrics), \quad (6.2)$$

$$language(T) = \arg\max_{l \in L} Prob(l \mid lyrics), \quad (6.3)$$

Обирається найвища ймовірність, і відповідний клас присвоюється треку.

Для автоматичного виявлення мови тексту пісні ми застосовуємо бібліотеку langdetect [9], яка є обгорткою над мовною моделлю Google Language Detection API. Ця модель належить до категорії мовної класифікації, яка аналізує статистичні характеристики тексту зокрема, частотність символів, співвідношення буквосполучень, довжину слів та лексичну структуру.

Бібліотека langdetect використовує наївний байєсівський підхід з попередньо зібраною статистикою для понад 50 мов. Під час обробки тексту вона повертає найвірогіднішу мову з вказанням її ймовірності. На рисунку 6.1 наведено приклад результату, який повертає langdetect під час визначення найбільш ймовірної мови треку:

```
Input: "Je suis très content aujourd'hui"
Output: {"language": "fr", "probability":
0.997}
{"language": "es", "probability": 0.003}
```

Рисунок 6.1 – Приклад результату визначення найбільш ймовірної мови треку

Якщо система не може визначити мову тексту, тобто вирогідність усіх можливих мов менша за 0.5 (50%), тоді система маркерує пісню як пісню з невизначенною мовою. Поріняльну таблицую технологій машинного навчання, які використовувались, наведено у таблиці 6.2.

Таблиця 6.2 – Порівняння технологій машинного навчання

Характеристика	Для визначення мови (langdetect)	Для настрою/жанру (bart-large-mnli)
Тип моделі	Статистична	Глибока нейромережа
Алгоритм	Наївний байєс	Zero-shot (MNLI)
Мова тексту	Будь-яка	Будь-яка
Вхідні дані	Текст пісні	Текст пісні
Вихід	Назва мови + імовірність	Настрій/Жанр + імовірність
Обмеження	Точність	Неточність при невеликих обсягах даних

7 РОЗРОБКА ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ПРОГРАМНОЇ ЗАБЕЗПЕЧУЮЧОЇ СИСТЕМИ

7.1 Загальна характеристика програмної системи

Розроблена програмна система є модулем музичної системи Spotify, що дозволяє створювати персоналізовані плейлисти. Система реалізована у вигляді клієнт-серверної архітектури, де сервер обробляє запити, виконує обчислення й зберігає результати, а клієнтська частина забезпечує взаємодію з користувачем через графічний інтерфейс.

Серверна частина побудована на Python з використанням Flask. Вона відповідає за всю бізнес-логіку: авторизацію через Spotify, збирання даних про треки, запити до сторонніх API Lyrics.ovh [7], Deezer API, NLP-аналіз текстів пісень та збереження результатів у базу.

Клієнтська частина реалізована з використанням JavaFX [11] - платформи для побудови настільних інтерфейсів у Java. Інтерфейс користувача поділений на логічні екрани: авторизація, перегляд статистики, генерація плейлиста, експорт результатів. Всі діаграми, таблиці та графіки реалізовані інтерактивно: кругові діаграми для жанрів, мов і настрою, гістограми для популярності, таблиці треків для перегляду історії прослуховування.

Окрему увагу приділено модулю обробки текстів. Для класифікації настрою й жанру використовується zero-shot модель facebook/bart-large-mnli із бібліотеки transformers. Для визначення мови застосовується модель xlm-roberta-base-language-detection, яка підтримує багатомовну класифікацію та працює з текстами пісень з Lyrics.ovh. У разі відсутності текстів аналіз проводиться за назвою треку, що знижує точність, але дозволяє не втрачати дані.

Усі дані після обробки зберігаються у структурованій формі. База містить окремі таблиці для треків з платформ Spotify і Deezer, окрему - для узагальненої бібліотеки треків, а також таблиці з перевагами користувача та рекомендованими треками. При генерації рекомендацій система обчислює оцінку релевантності

кожного треку на основі минулих прослуховувань слухача тобто система буде йому рекомендовати переважно музику з такими же самими параметрами для бльшої кастомізації в інтерфейсі передбачена опція змінювати вагу цих коефіцієнтів щоб трішки разнообразити вподобання користувача. Це дозволяє налаштувати результат під побажання слухача.

Наша система орієнтована на автономне розгортання усі компоненти можуть запускатися локально, без потреби у складній інфраструктурі. Також система підтримує різні операційни системи та працює стабільно.

7.2 Опис компонентів системи

Модуль складається з трьох основних компонентів: клієнтської частини, серверної частини та бази даних (рисунок 7.1). Клієнтська частина дозволяє пройти авторизацію через Spotify або надати посилання на профіль Deezer, переглянути власні музичні вподобання у вигляді діаграм, налаштувати вагові коефіцієнти, і здійснити експорт згенерованої добірки до мизичного сервісу. Комунікація між клієнтом і сервером здійснюється за допомогою HTTP-запитів до REST API.

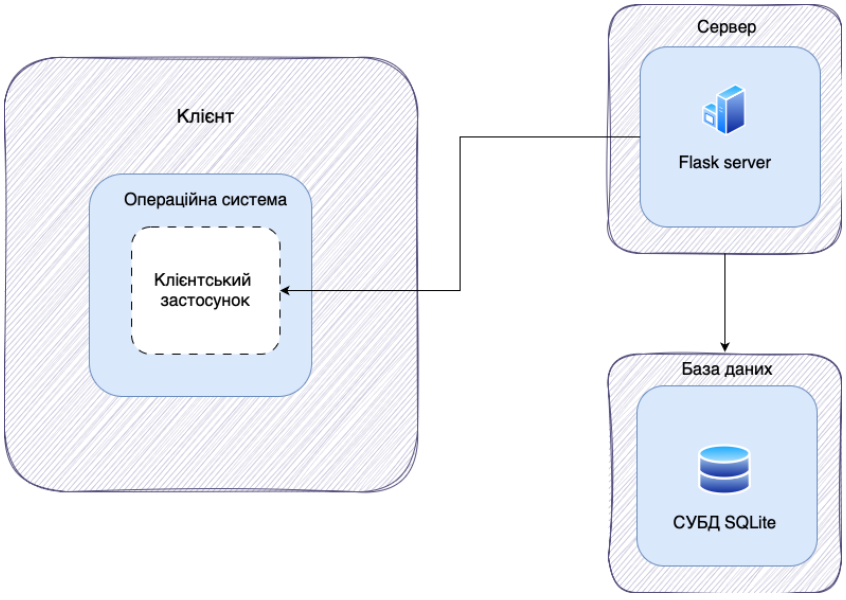

Рисунок 7.1 – Схема програмних компонентів модуля

Серверна частина реалізована на сервері використовуючи мову програмування Python, який дозволяє швидко обробляти запити та реалізовувати бізнес-логіку. Сервер обробляє запити від клієнта через такі ключові маршрути: /spotify/profile для отримання історії прослуховувань і даних профілю Spotify, /deezer/profile для витягу треків із Deezer, /playlist/recommended-v2 для генерації рекомендацій, а також /playlist/export для експорту плейлисту назад у Spotify. Авторизація користувача в Spotify реалізована через функцію OAuth2 яка після переходу за URL авторизації користувач надає дозвіл, сервер отримує access token і зберігає його тимчасово в середовище виконання.Після авторизації сервер виконує запити до API стрімінгових сервісів. Наприклад, у випадку Spotify використовують запит до https://api.spotify.com/v1/me/top/tracks, а в разі Deezer - до https://api.deezer.com/user/{id}/tracks. Дані про треки доповнюються текстами пісень через Lyrics.ovh, після чого проходять обробку з використанням моделей машинного навчання. Кожен трек отримує визначення за трьома ознаками та додатковими метриками такими як популярність, кількість прослуховувань, після чого зберігається до бази даних.

У процесі формування персоналізованого плейлиста сервер обчислює статистичний профіль користувача переваги за жанром, настроєм і мовою. Після цього всі треки з бази оцінюються за формулою релевантності, в яку входять вагові коефіцієнти, задані користувачем. Алгоритм обчислює загальний score для кожного треку і формує список з найрелевантніших композицій. За запитом /playlist/export цей список може бути надіслано назад до Spotify: спочатку створюється новий плейлист, а потім у нього додаються композиції.

Сервер постійно взаємодіє з базою даних: додає нові записи, оновлює лічильники, витягує дані для візуалізації й генерації плейлистів.

Компонент JavaFX GUI виконує роль клієнтської частини: відображає інтерфейс користувача, надсилає запити на сервер, приймає відповіді та візуалізує їх у вигляді таблиць, діаграм та графіків. Усі запити надсилаються через HTTP до компонента Flask Server.

Flask Server є центральним елементом бекенду. Він приймає запити від

клієнта, обробляє їх та взаємодіє з іншими підсистемами

Flask виконує роль обробника, координатора й аналітика між усіма іншими компонентами. Всі результати аналізу передаються назад клієнту для подальшої візуалізації та взаємодії.

7.3 Діаграми програмної структури

7.3.1 Діаграма класів

На діаграмі класів рисунок 7.2 зображено ключові класси та функції програмного модуля, які реалізують повний цикл роботи з музичними даними - від запуску інтерфейсу до генерації персоналізованого плейлиста та його експорту. Кожен клас виконує чітко визначену роль і має набір методів, які дозволяють ізольовано обробляти відповідну частину логіки.

Зверху діаграмми знаходиться клас MusicGUI - це головний клас клієнтської частини на JavaFX, який відповідає за запуск програми, побудову сцен, перемикання між вікнами авторизації, перегляду статистики, генерації плейлистів. Саме тут викликаються методи візуалізації діаграм, ініціалізується робота з SpotifyAPIHandler, DeezerAPIHandler та RecommendationService. Він є точкою входу в застосунок, зв’язаний із класами нижчого рівня через TrackModelService.

AppInitializer відповідає за ініціалізацію всіх сервісів, що будуть використовуватись у MusicGUI. Він отримує налаштування з класу AppConfig, де можуть зберігатися, наприклад, адреса бекенду, токени або інші параметри запуску. Це дозволяє централізовано керувати конфігурацією всього клієнта.

Track - це перевантажений класс для представлення одного треку в клієнті. Вона містить поля: title, author, genre, mood, language, popularity та score. Клас не має логіки, але слугує структурою даних, яка використовується в таблицях і графіках. Його обробляє клас TrackModelService, який оновлює дані у таблицях,

перетворює JSON у список Track, формує список для діаграм тощо.

SpotifyAPIHandler виконує логіку авторизації користувача в Spotify через OAuth2, отримує access token, відправляє запити до API (/me, /me/top/tracks, /me/playlists) і повертає JSON-об'єкти з інформацією про користувача. Він викликається безпосередньо з TrackModelService. Також цей клас передає дані в RecommendationService.

SpotifyExportService працює у зв'язці з RecomendationService і відповідає за експорт згенерованого плейлиста: створює плейлист та додає треки. Він читає назву плейлиста та список URI з клієнта і виконує запити до Spotify. DeezerAPIHandler працює аналогічно до Spotify-хендлера, але для Deezer. Оскільки Deezer не підтримує OAuth у публічному вигляді, він використовує посилання на профіль і витягує треки через API. Також робить додаткові запити до /album/{id} для отримання genre.

DeezerGenreService - допоміжний клас, який визначає жанр треку через Deezer API. Він використовується, коли класифікатор не може точно визначити жанр.

LyricService відповідає за отримання текстів пісень через API https://api.lyrics.ovh/v1/{artist}/{title}. Цей текст потім передається у NLPService для визначення жанру, мови та настрою.

NLPService - ключовий клас для обробки текстів. Метод detectMood(text) повертає один з шести настроїв; detectGenre(text) - один із жанрів, заданих у системі. Цей клас працює з текстами з LyricService.

RecommendationService - центральна логіка генерації персоналізованого плейлиста. Метод generateRecommendedPlaylist(platform, weights) виконує:

- зчитування вподобань користувача з бази;
- фільтрацію та оцінку кожного треку
- збереження результатів у RecommendedPlaylist;
- формування відповіді у форматі JSON для клієнта.

Усі API-хендлери працюють із REST-запитами до бекенду і не зберігають стан, що дозволяє їх легко тестувати й підміняти. Обробка NLP і побудова

плейлистів виконуються на боці сервера, а клієнт лише ініціює виклик і візуалізує результат.

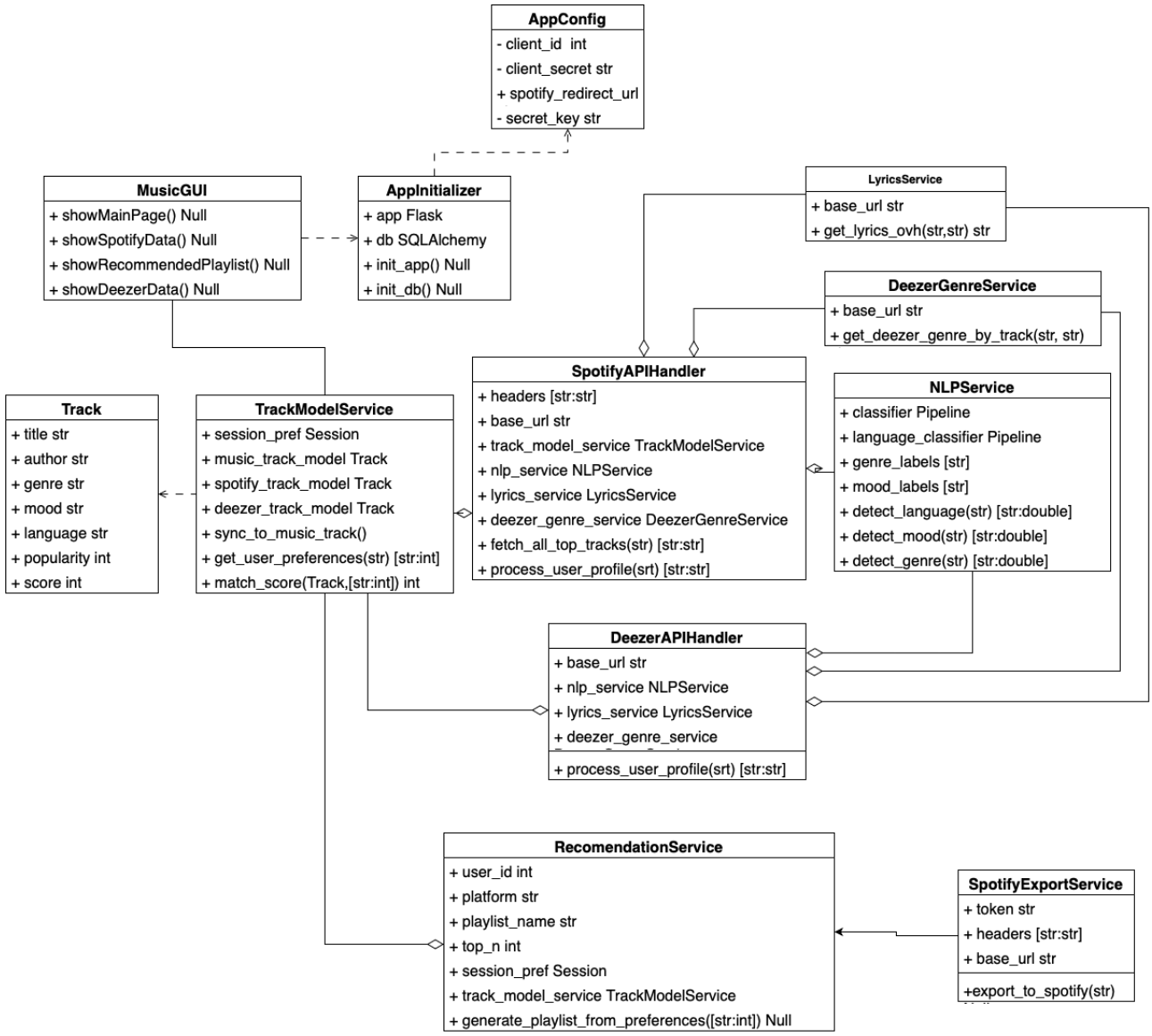

Рисунок 7.2 - Архітектура побудованої системи у вигляді діаграми класов

7.3.2 Діаграма активності

Діаграма активности для розроблюваного модуля зображена на рисунку 7.3. Вона наочно демонструє повний сценарій взаємодії користувача з програмою модуля. Вона охоплює ключові гілки логіки, залежно від обраної музичної платформи - Spotify або Deezer.

На початку користувач потрапляє до головного меню, де обирає платформу для аналізу. У разі вибору Spotify, система спрямовує користувача на сторінку авторизації Spotify. Після успішної авторизації відбувається завантаження профілю користувача, зокрема топ-треків, жанрів, артистів. Отримані дані обробляються на сервері, і користувач отримує можливість переглянути аналітику у вигляді діаграм.

У випадку вибору Deezer, користувач вводить посилання на свій публічний профіль Deezer. Система автоматично витягує інформацію про історію прослуховувань і теж формує аналітичні графіки за аналогічною логікою, використовуючи модулі класифікації.

Незалежно від обраної платформи, після перегляду вподобань користувач переходить до налаштування рекомендацій тут вводяться коефіцієнти ваг , назва плейлиста, а також визначається, з якої саме платформи брати треки для формування добірки.

Далі система виконує логіку рекомендації сервер порівнює треки з бази з вподобаннями користувача, обчислює score кожному треку та формує список з найрелевантніших. Цей плейлист повертається клієнту, де виводиться у вигляді таблиці та гістограми. З’являється можливість експорту плейлиста у музичний сервіс Spotify де автоматично створюється новий плейлист у профілі користувача та заповнюється підібраними треками.

Ця діаграма демонструє повний робочий цикл застосунку, з урахуванням обробки різних варіантів дій користувача.

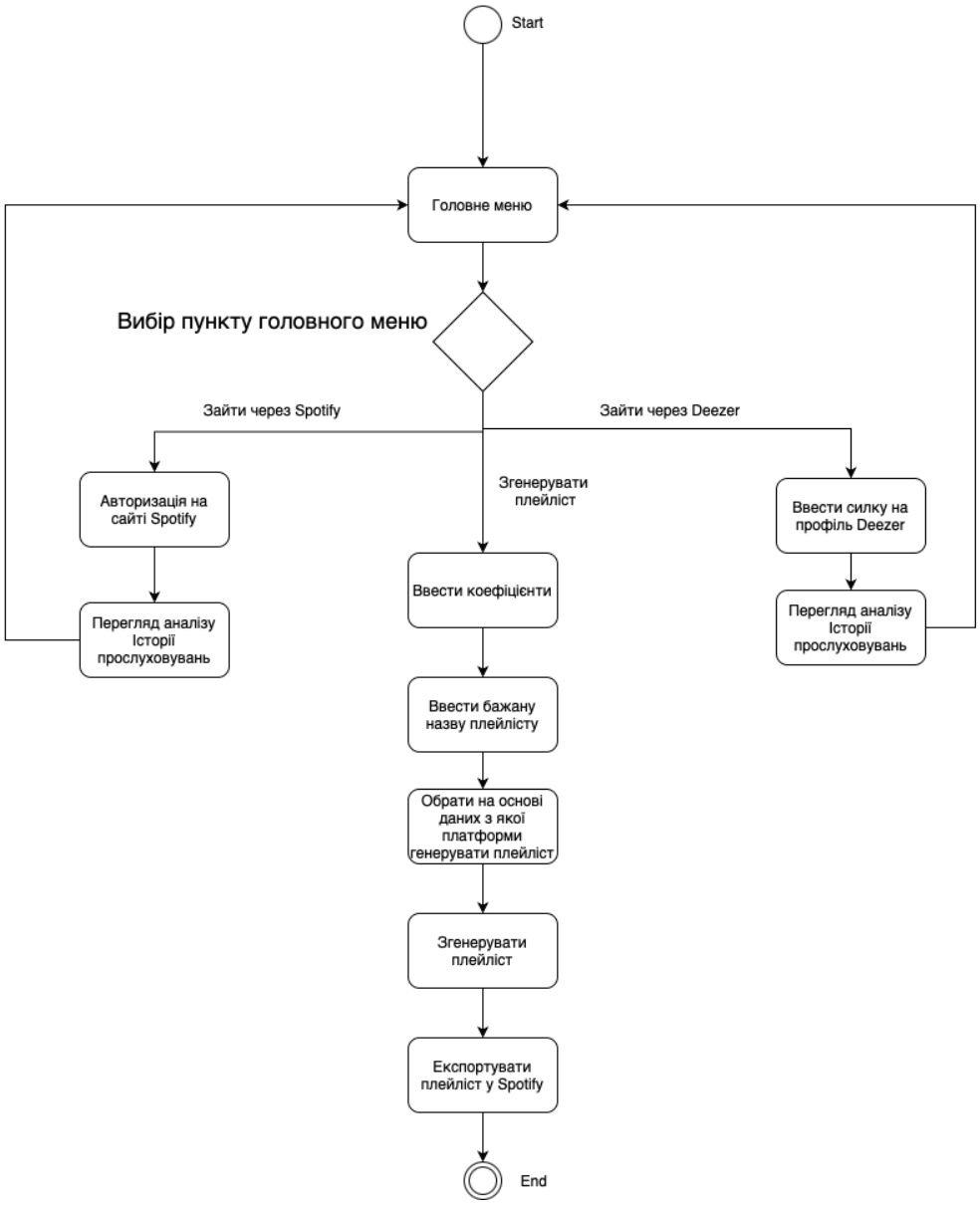

Рисунок 7.3 – Діаграма активності модуля

7.4 Опис екранних форм (інтерфейсу)

Графічний інтерфейс системи реалізовано за допомогою JavaFX. Кожна функціональна частина реалізована у вигляді окремої форми-сцени, між якими користувач може переходити. Інтерфейс підтримує інтерактивну взаємодію, візуалізацію статистики у вигляді діаграм і таблиць, а також забезпечує зворотній зв’язок наприклад, повідомлення про успішний експорт плейлиста.

Перша форма, яку бачить користувач після запуску програми, - це головне меню (див. рисунок 7.4). Тут можна обрати джерело даних - Spotify або Deezer, а також перейти до генерації плейлиста.

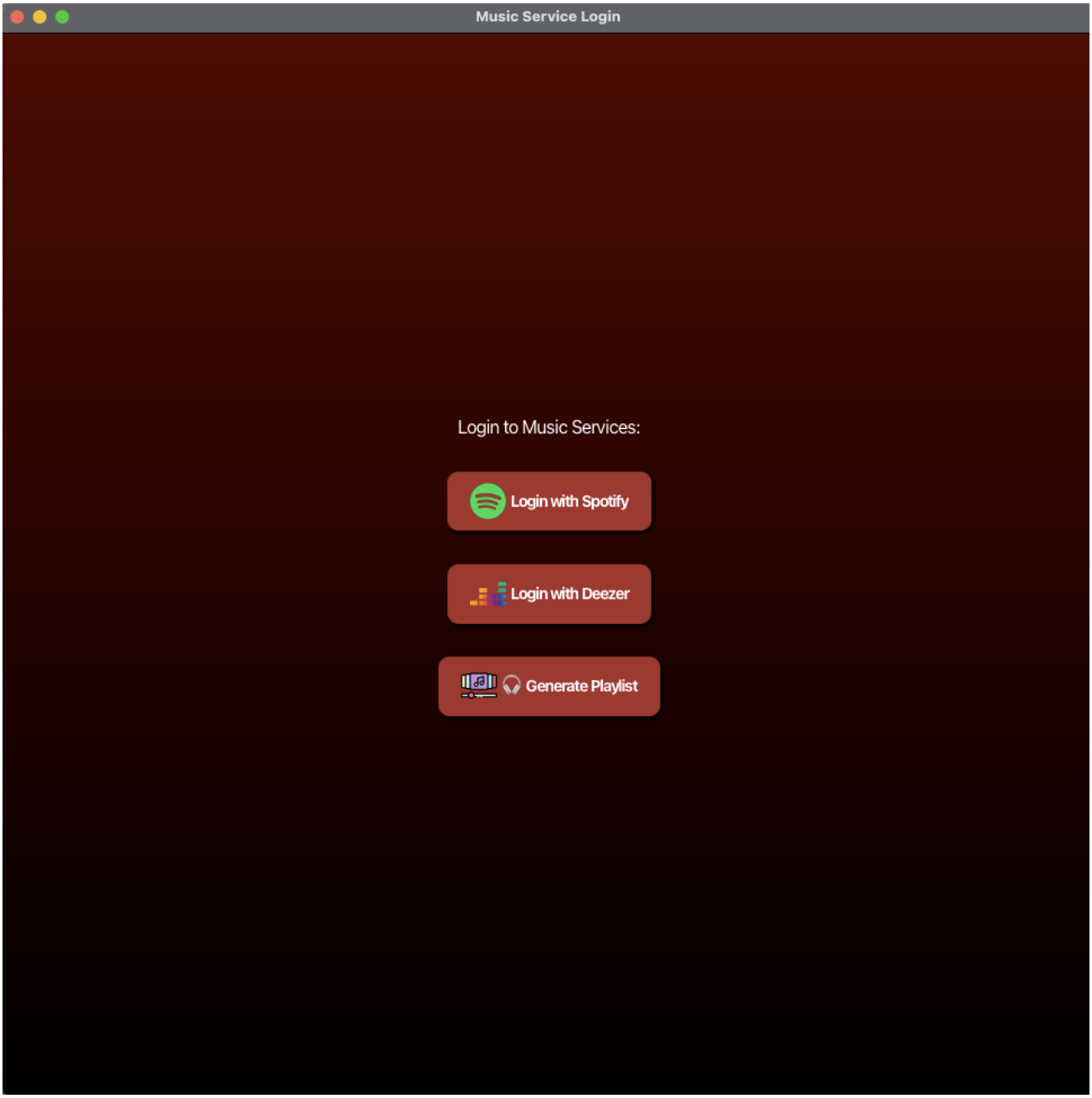

Рисунок 7.4 – Головне меню застосунку

При виборі Spotify відкривається браузер для OAuth-авторизації, після чого користувач автоматично повертається в застосунок рисунок 7.5. Для Deezer реалізовано просту форму, де користувач вводить посилання на свій профіль (див.

рисунок 7.6).

Рисунок 7.5 – Сторінка аналізу прослуховувань користувача на основі даних, які були отримані зі Spotify

Рисунок 7.6 – Сторінка аналізу прослуховувань користувача на основі даних, які були отримані з Deezer

У формі користувач вводить назву майбутнього плейлиста, налаштовує вагові коефіцієнти для жанру, мови, настрою та популярності через слайдери й обирає платформу, на основі якої слід формувати добірку (див. рисунок 7.7). Нижче виводиться таблиця з найрелевантнішими треками та графік, який показує їхні бали.

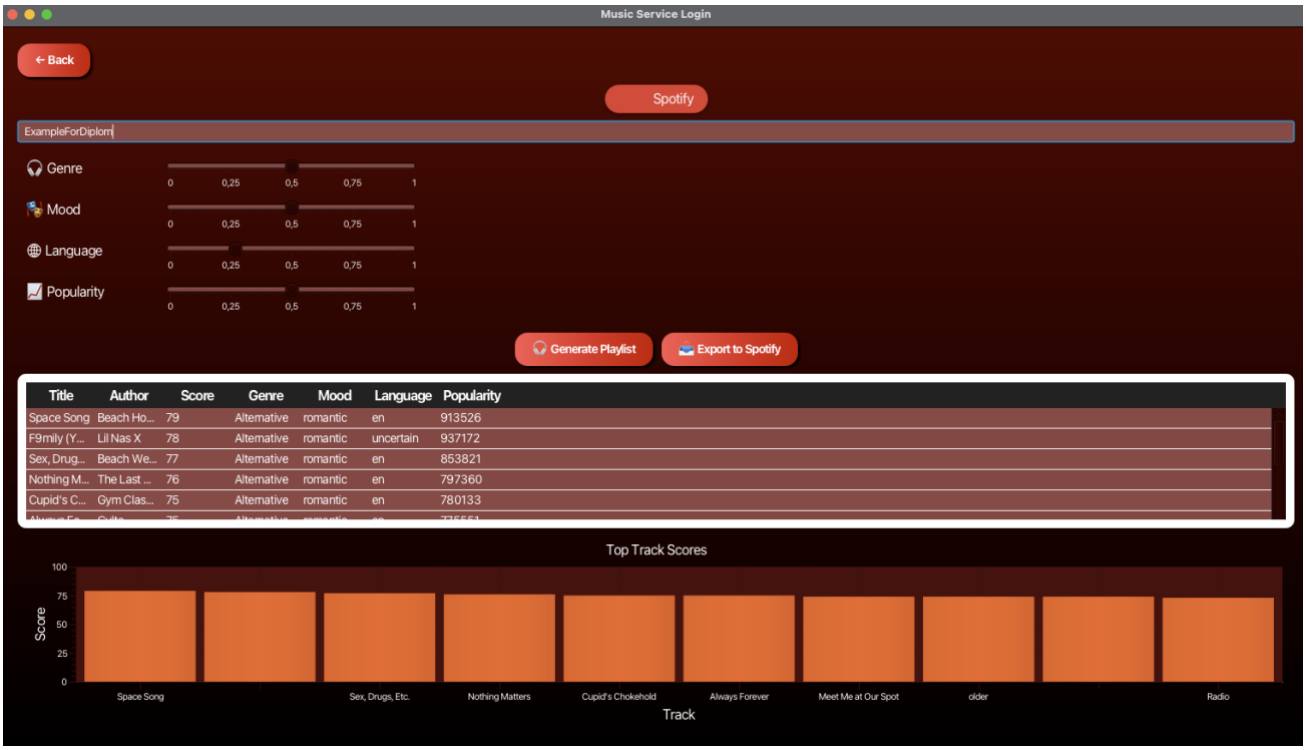

Рисунок 7.7 –Інтерфейс генерації персоналізованого плейлиста

Після генерації треків користувач може натиснути кнопку експорту. У разі успішного створення плейлиста в обліковці Spotify система виводить підтвердження (див. рисунок 7.7). У випадку помилки наприклад, токен неактивний, з’являється повідомлення про помилку. Після чого сгенерований плейліст з’являється (див рисунок 7.9) у нашому аккаунті на сайті Spotify де вже можно буде прослухати музику самостійно.

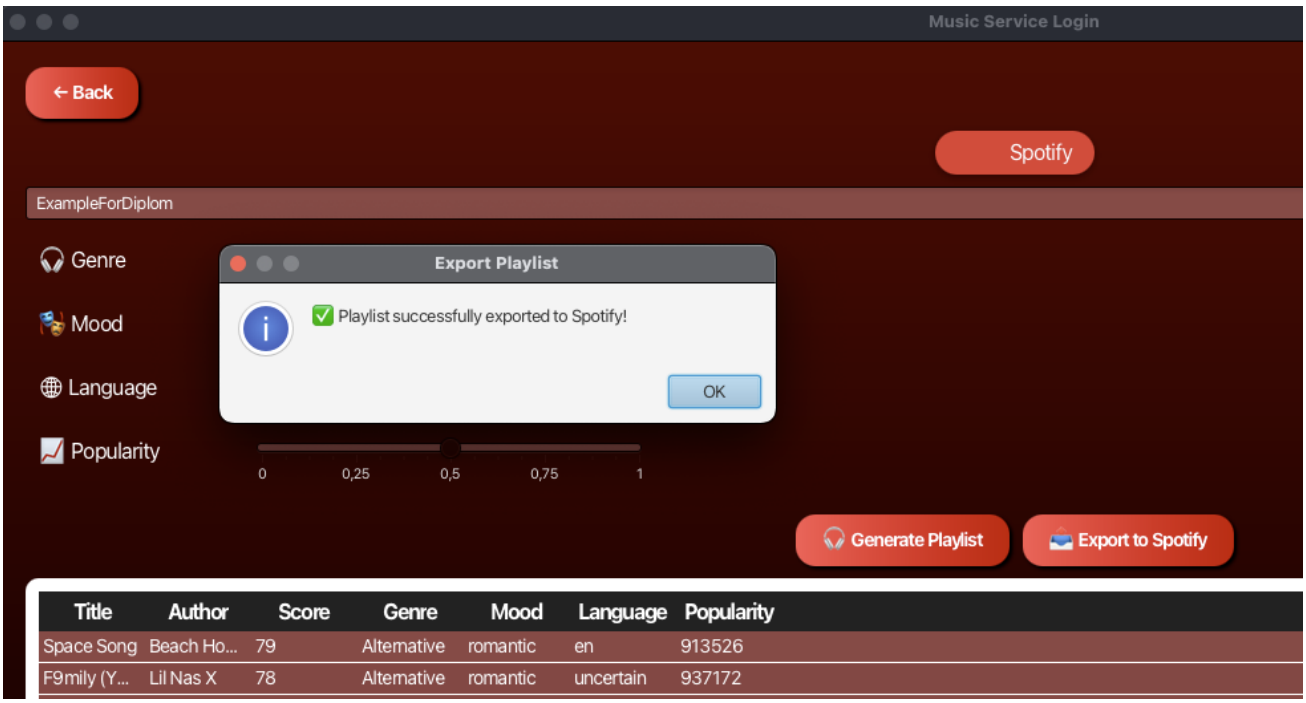

Рисунок 7.8 - Повідомлення про успішний експорт плейлиста

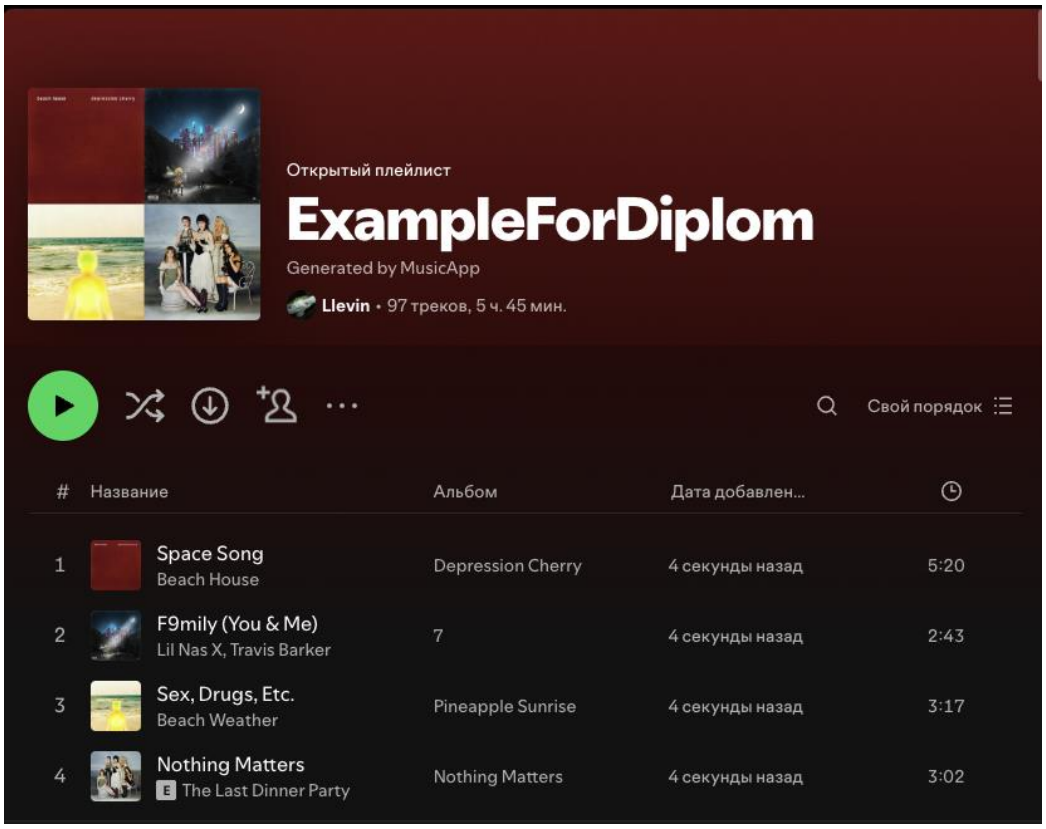

Рисунок 7.9 – Сгенерований плейліст у застосунку Spotify

7.5 Обґрунтування вибору технологій

Під час розробки інформаційної системи було прийнято рішення використовувати сучасні, гнучкі та зручні технології, які забезпечують швидку інтеграцію з сторонніми сервісасми, простоту підтримки, а також комфортну роботу з інтерфейсом.

Python було обрано як основну мову для бекенд-частини через його виразний синтаксис, багатий набір бібліотек для обробки тексту та інтеграції з REST API, а також швидкість розробки.

Фреймворк Flask надає інструментарій для побудови RESTful API. Він дозволяє легко визначати маршрути (/spotify/profile, /playlist/recommended-v2, /preferences) і реалізовувати серверну логіку. Flask підтримує інтеграцію з крос-доменну взаємодію, що забезпечує безперешкодну взаємодію з клієнтською частиною, реалізованою окремо.

Для побудови графічного інтерфейсу застосовано JavaFX. Він забезпечує гнучке компонування, підтримку анімацій, кастомізованих стилів, а також побудову діаграм, таблиць та слайдерів, необхідних для інтерактивної взаємодії з користувачем.

Клієнт реалізує асинхронні HTTP-запити через Java HttpClient, що дозволяє не блокувати UI при зверненні до бекенду.

Deezer API використовується у відкритому доступі без потреби авторизації, що дозволяє витягувати треки з публічного профілю за посиланням.

API Lyrics.ovh безкоштовний та відкритий сервіс для пошуку текстів музичного твору.

Для класифікації настрою та жанру використовується бібліотека transformers від Hugging Face з моделлю facebook/bart-large-mnli [4] у режимі zero-shot classification як доступна та універсальна модель для класифікации тексту.

7.6 Програмна реалізація проекту

У цьому розділі представлені ключові функції, що реалізують основну бізнес-логіку програмної системи, а саме авторизацію користувача, збирання даних про музику, аналіз текстів пісень, генерацію рекомендацій та експорт у Spotify. Всі ці дії описані у вигляді лістингів коду (див. лістинги 7.1–7.6), які пов’язані між собою послідовним процесом.

Функція авторизації користувача через Spotify (login_spotify) представлена у лістингу 7.1. Вона формує URL із параметрами OAuth2 для перенаправлення користувача на офіційну сторінку авторизації Spotify, де запитує дозволи на читання інформації про треки та плейлисти, а також дозвіл на їх створення. Це необхідно для подальшого отримання доступу до персональних даних користувача та формування плейлистів у його акаунті.

Лістинг 7.1 – Програмний код авторизаціїї користувача через Spotify

```
@app.route("/login/spotify")  # Реєструє маршрут Flask, який 
викликається при вході користувача через Spotify
def login_spotify():
    params = {
        "client_id": SPOTIFY_CLIENT_ID,  # ID клієнта, 
зареєстрованого в Spotify Developer Dashboard
        "response_type": "code",  # Запит на отримання коду 
авторизації
        "redirect_uri": SPOTIFY_REDIRECT_URI,  # URI, на який буде 
здійснено редірект після авторизації
        "scope": "user-read-private user-read-email user-library-
read user-top-read playlist-read-private playlist-modify-private 
playlist-modify-public"
        # Дозволи, які запитуються у користувача
    }
    url = "https://accounts.spotify.com/authorize?" + 
urllib.parse.urlencode(params)
    # Формується URL авторизації Spotify з параметрами
    return redirect(url)  # Здійснюється перенаправлення 
користувача на Spotify для входу
```

Після авторизації застосунок викликає функцію отримання історії

прослуховування треків слухача, яка показана у лістингу 7.2. Вона виконує послідовні запити до Spotify API, обробляє результаті і формує повний список з треків, які користувач слухав найчастіше за останній тривалий період. Ці треки використовуються як вхідні дані для побудови профілю вподобань.

Лістинг 7.2 – Функція отримання історії прослуховування треків на музичному сервісі Spotify

```
def fetch_all_top_tracks(headers):
    all_tracks = []  # Ініціалізується порожній список для всіх треків
    url = "https://api.spotify.com/v1/me/top/tracks?limit=50&time_range=long_term"
    # Перший URL для запиту топ-треків користувача
    while url:  # Виконується, поки є наступна сторінка
        resp = requests.get(url, headers=headers).json()  # GET-запит до Spotify API
        items = resp.get("items", [])  # Отримання списку треків з JSON-відповіді
        all_tracks.extend(items)  # Додавання треків до основного списку
        url = resp.get("next")  # Наступна сторінка результатів (або None, якщо закінчилось)
    return all_tracks  # Повернення повного списку топ-треків
```

Для більш глибокого аналізу, кожен трек додатково обробляється через API lyrics.ovh, що реалізовано у лістингу 7.3. Тут за допомогою даних про виконавця та назви пісні отримується її текст. Якщо текст не знайдено то система повертає None, що може вплинути на точність наступної класифікації.

Лістинг 7.3 – Функція отримання тексту пісні

```
def get_lyrics_ovh(artist: str, title: str) -> str | None:
    url = f"https://api.lyrics.ovh/v1/{artist}/{title}"  # Формується запит до API lyrics.ovh
    response = requests.get(url)  # Надсилається GET-запит
    if response.status_code == 200:  # Перевірка, чи запит успішний
        return response.json().get("lyrics")  # Повернення тексту пісні
    return None  # Якщо статус-код не 200 - повертається None
```

Отримані тексти аналізуються за допомогою zero-shot класифікатора у функції detect_mood та подібних функцій для класифікації жанру чи язику, показаній у лістингу 7.4. Вона визначає настрій композиції.

Лістинг 7.4 – Функція класифікації настрою за допомогою zero-shot моделі

```
def detect_mood(text: str) -> str:
    labels = ["happy", "sad", "romantic", "angry", "energetic",
"calm"]
    # Кандидатні мітки для zero-shot класифікації
    try:
        result = classifier(text, candidate_labels=labels)
        # Виклик zero-shot класифікатора з текстом та мітками
        return result["labels"][0] if result and
result.get("labels") else "N/A"
        # Повертається найбільш вірогідна мітка або "N/A" якщо
результат пустий
    except Exception as e:
        print(f"[WARN] Mood detection failed: {e}")
        # Виведення помилки в консоль у випадку проблем
        return "N/A"  # Повернення "N/A" у разі винятку
```

Далі, система формує персоналізований плейлист, орієнтуючись на вподобання користувача. Цей процес описаний у лістингу 7.5. Функція generate_recommended_playlist_from_preferences обчислює оцінку релевантності (score) для кожного треку з бази на основі жанру, настрою, мови та популярності, з урахуванням вагових коефіцієнтів, заданих користувачем. Отриманий список впорядковується за рейтингом та зберігається як набір рекомендацій.

Лістинг 7.5 – Програмний код генерації персоналізованого плейлиста

```
def generate_recommended_playlist_from_preferences(n=100,
platform="spotify", user_id=1, playlist_name="My Playlist"):
    preferences = get_user_preferences(platform)
    # Отримання збережених вподобань користувача
    candidates = session_pref.query(TrackPreference).all()
    # Витяг усіх треків-кандидатів з бази
    RecommendedPlaylist.query.filter_by(userID=user_id).delete()
    # Очистка старих результатів для цього користувача
    ranked = []  # Список відсортованих треків
```

```
    for c in candidates:
        score = 0
        score += genre_weight * preferences["genres"].get(c.genre,
0)
        # Додається бал за жанр
        score += mood_weight * preferences["moods"].get(c.mood, 0)
        # Додається бал за настрій
        score += lang_weight *
preferences["languages"].get(c.language, 0)
        # Додається бал за мову
        ranked.append((score, c))  # Трек з його балом додається до
списку
    ranked.sort(reverse=True, key=lambda x: x[0])
    # Список сортується за спаданням оцінки (від найпідходящего до
менш підходящего)
```

Останній крок - це експорт згенерованого плейлиста в обліковий запис користувача Spotify. Цей процес детально описаний у лістингу 7.6. Спочатку система створює новий плейлист у акаунті користувача через відповідний API-запит, а потім виконує пошук кожного треку за назвою й автором, отримує Spotify URI і додає його до нового плейлиста. Користувач отримує готовий список треків без необхідності ручного додавання.

Лістинг 7.6 – Функція експорту плейлиста в Spotify

```
def export_to_spotify(playlist, name, token):
    headers = {"Authorization": f"Bearer {token}", "Content-Type":
"application/json"}
    # Заголовки запиту з токеном авторизації
    profile = requests.get("https://api.spotify.com/v1/me",
headers=headers).json()
    # Отримання інформації про користувача
    user_id = profile.get("id")  # Витяг user_id із відповіді
    res =
requests.post(f"https://api.spotify.com/v1/users/{user_id}/playlist
s", headers=headers, json={"name": name})
    # Створення нового плейлиста в акаунті користувача
    playlist_id = res.json().get("id")  # Витяг ID нового плейлиста
    for item in playlist:
        search_url =
f"https://api.spotify.com/v1/search?q=track:{item['title']}
artist:{item['author']}&type=track&limit=1"
        # Формується запит на пошук треку по назві та виконавцю
        track_res = requests.get(search_url,
headers=headers).json()
        uri = track_res["tracks"]["items"][0]["uri"]
```

```
        # Витягнутий Spotify URI для треку

requests.post(f"https://api.spotify.com/v1/playlists/{playlist_id}/
tracks", headers=headers, json={"uris": [uri]})
        # Додавання треку до створеного плейлиста
```

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було реалізовано повноцінний модуль для аналізу музичних уподобань користувачів та автоматичного формування персоналізованих плейлистів. Основними функціональними можливостями розробленого модулю є інтеграція з музичними сервісами, збір історії прослуховувань, обробка текстів пісень, визначення жанру, мови та настрою композицій, побудова користувацького профілю та створення релевантних добірок із можливістю їх експорту. Усі функції протестовано на реальних даних, результати виводяться у графічному та табличному вигляді, що забезпечує зручність сприйняття.

Порівняно з існуючими музичними платформами, розроблений модуль відрізняється відкритістю логіки формування рекомендацій, можливістю змінювати вагові коефіцієнти вподобань і прозорістю механізму добору треків. Більшість сучасних стрімінгових сервісів використовують закриті алгоритми, не дозволяючи користувачеві впливати на процес формування плейлистів або адаптувати його під власні потреби. Натомість створене рішення забезпечує гнучке налаштування та контроль над параметрами персоналізації, а також дає змогу розширювати функціонал без втрати цілісності архітектури.

Новизна одержаних результатів полягає в інтеграції одразу трьох рівнів аналізу: поведінкового – з використанням історії прослуховувань, лінгвістичного – завдяки визначенню мови тексту пісні та емоційного – за допомогою визначення настрою композиції. Для цього використано сучасні методи обробки природної мови та моделі штучного інтелекту, зокрема zero-shot класифікацію та бібліотеку langdetect. Крім того, модуль передбачає механізм гнучкого налаштування ваг кожного параметра, що забезпечує персоналізацію.

Практична значущість розробленого модуля полягає в можливості його інтеграції до існуючих бізнес-процесів у сфері стрімінгових музичних сервісів. Система може бути використана для підвищення залученості користувачів музичних сервісів, створення персоналізованих добірок, аналітики поведінки

користувачів. Надалі доцільно реалізувати підтримку інших стрімінгових платформ, додати аудіоаналіз треків без текстів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Spotify for Developers [Електронний ресурс]. - Режим доступу: https://developer.spotify.com (дата звернення: 13.05.2025).

2. YouTube Data API - Google Developers [Електронний ресурс]. - Режим доступу: https://developers.google.com/youtube (дата звернення: 13.05.2025).

3. Apple Music API Documentation [Електронний ресурс]. - Режим доступу: https://developer.apple.com/documentation/applemusicapi (дата звернення: 13.05.2025).

4. Last.fm API [Електронний ресурс]. - Режим доступу: https://www.last.fm/api (дата звернення: 13.05.2025).

5. Pandora Music Genome Project - Overview [Електронний ресурс]. - Режим доступу: https://www.pandora.com/about/mgp (дата звернення: 13.05.2025).

6. SoundCloud for Developers [Електронний ресурс]. - Режим доступу: https://developers.soundcloud.com (дата звернення: 13.05.2025).

7. Lyrics.ovh API Documentation [Електронний ресурс]. - Режим доступу: https://lyricsovh.docs.apiary.io - Назва з екрана.

8. facebook/bart-large-mnli - Hugging Face [Електронний ресурс]. - Режим доступу: https://huggingface.co/facebook/bart-large-mnli - Назва з екрана.

9. Langdetect - Language detection library for Python [Електронний ресурс]. - Режим доступу: https://pypi.org/project/langdetect - Назва з екрана.

10. Flask Documentation [Електронний ресурс]. - Режим доступу: https://flask.palletsprojects.com - Назва з екрана.

11. JavaFX Official Site [Електронний ресурс]. - Режим доступу: https://openjfx.io - Назва з екрана.

12. SQLAlchemy - Python SQL Toolkit and ORM [Електронний ресурс]. - Режим доступу: https://www.sqlalchemy.org - Назва з екрана.

13. Anupam Biswas, Emile Wennekes, Alicja Wieczorkowska, Rabul Hussain Laskar, Advances in Speech and Music Technology - Switzerland: Nature, 2023. - 309

14. Nguyen, T. L., Nguyen, N. D. A survey of music recommendation systems and future perspectives // IEEE Access. - 2020. - Vol. 8. - P. 87678–87708. - DOI: 10.1109/ACCESS.2020.2991734.

15. ДСТУ 3008:2015. Державний стандарт України. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення. - К.: ДП «УкрНДНЦ», 2016. - 31 с.

16. ДСТУ 8302-2015. Бібліографічне посилання. Загальні положення та правила складання. - Київ: ДП «УкрНДНЦ», 2016. - 20 с.