

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр» за
спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

**«Вебзастосунок для інтерактивного прослуховування та
візуалізації музичного контенту»**

Виконав студент групи ВТ-22-2
КРАМАРОВСЬКИЙ Ренат Русланович

Керівник роботи:
ФАНТ Микола Олександрович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КРАМАРОВСЬКИЙ Ренат Русланович**

Керівник роботи: **ФАНТ Микола Олександрович**

Тема роботи: **«Вебзастосунок для інтерактивного прослуховування та візуалізації музичного контенту»**, затверджена Наказом закладом вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **методи та засоби розробки вебзастосунків для потокового відтворення, та інтерактивної візуалізації музичного контенту.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Фант М.О.	02.03.2026	02.03.2026
2	Фант М.О.	02.03.2026	02.03.2026
3	Фант М.О.	02.03.2026	02.03.2026

РЕФЕРАТ

Випускна кваліфікаційна робота присвячена проектуванню та розробці багатофункціонального музичного вебзастосунку, що поєднує інструменти інтерактивної взаємодії з аудіоконтентом, розгалужену систему модерації та засоби соціальної комунікації.

У роботі проведено аналіз предметної області цифрових мультимедійних платформ, досліджено наявні світові аналоги музичних стрімінгових сервісів, обґрунтовано вибір монолітної клієнт-серверної архітектури та інструментальних засобів розробки.

Метою роботи є створення масштабованого вебзастосунку для публікації, прослуховування та керування музичним контентом, що забезпечує користувачів засобами персоналізації, соціальної взаємодії та візуального супроводу до прослуховуваного аудіоконтенту.

Об'єктом дослідження є процес розробки та функціонування музичного вебдодатку, що забезпечує зберігання аудіофайлів, їхнє відтворення з низькою затримкою та синхронну візуалізацію частотних характеристик звуку в реальному часі, включно з такими соціальними складовими як коментування композицій та управління правами доступу до власних списків відтворення.

Випускна кваліфікаційна робота містить 68 сторінок основного тексту, 38 ілюстрацій, 18 таблиць, додатки та 30 використаних літературних джерел.

Ключові слова: ВЕБЗАСТОСУНОК, ЗАВАНТАЖЕННЯ, ЗБЕРІГАННЯ, ПРОСЛУХОВУВАННЯ, КЕРУВАННЯ ДОСТУПОМ, ACL, МОДЕРАЦІЯ КОНТЕНТУ, АКУСТИЧНИЙ ВІДБИТОК МУЗИЧНИЙ СЕРВІС, ТРЕК, СПЕКТРОГРАМА, ВІЗУАЛІЗАЦІЯ, PWA, AWS, SDK, S3.

					ІПЗ.КР.Б-121.26 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебзастосунок для інтерактивного прослуховування та візуалізації музичного контенту	Літ.	Арк.	Аркуші
Розроб.		Крамаровський Р.Р.					3	83
Перевір.		Фант М.О.						
Реценз.		Єфремов Ю.М.				ФІКТ, гр. ВТ-22-2		
Зав.каф.		Вакалюк Т.А.						

ABSTRACT

The final qualification work is devoted to the design and development of a multifunctional music web application that combines tools for interactive interaction with audio content, an extensive moderation system and social communication tools.

The work analyzes the subject area of digital multimedia platforms, investigates existing global analogues of music streaming services, justifies the choice of a monolithic client-server architecture and development tools.

The purpose of the work is to create a scalable web application for publishing, listening and managing music content, which provides users with personalization tools, social interaction and visual accompaniment to the audio content being listened to.

The object of the research is the process of developing and operating a music web application that provides storage of audio files, their playback with low latency and synchronous visualization of frequency characteristics of sound in real time, including such social components as commenting on compositions and managing access rights to their own playlists.

The final qualification work contains 68 pages of main text, 38 illustrations, 18 tables, appendices and 30 used literary sources.

Keywords: WEB APPLICATION, DOWNLOADING, STORAGE, LISTENING, ACCESS CONTROL, ACL, CONTENT MODERATION, ACOUSTIC IMPRINT MUSIC SERVICE, TRACK, SPECTROGRAM, VISUALIZATION, PWA, AWS, SDK, S3.

					<i>ІПЗ.КР.Б-121.26 - ПЗ</i>			
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>	<i>Вебзастосунок для інтерактивного прослуховування та візуалізації музичного контенту</i>	<i>Літ.</i>	<i>Арк.</i>	<i>Аркушів</i>
<i>Розроб.</i>		<i>Крамаровський Р.Р.</i>					4	83
<i>Перевір.</i>		<i>Фант М.О.</i>						
<i>Реценз.</i>		<i>Єфремов Ю.М.</i>				<i>ФІКТ, гр. ВТ-22-2</i>		
<i>Зав.каф.</i>		<i>Вакалюк Т.А.</i>						

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ЗНАЧЕНЬ	7
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ РОЗРОБКИ ВЕБ-СИСТЕМ ДЛЯ ІНТЕРАКТИВНОГО ПРОСЛУХОВУВАННЯ АУДІО КОНТЕНТУ	11
1.1. Постановка задачі	11
1.2. Аналіз аналогів програмного продукту	13
1.3. Вибір архітектури розроблюваної системи	22
1.4. Обґрунтування вибору засобів реалізації та вимоги до апаратного забезпечення	25
Висновки до першого розділу	28
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ІНТЕРАКТИВНОГО ПРОСЛУХОВУВАННЯ ТА ВІЗУАЛІЗАЦІЇ МУЗИЧНОГО КОНТЕНТУ	30
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	30
2.2. Розробка бази даних системи	39
2.3. Проектування та реалізація алгоритмів роботи системи	45
2.4. Реалізація функціональних модулів та компонентної структури вебзастосунку	49
Висновки до другого розділу	53
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБЗАСТОСУНКОМ ..	54
3.1. Порядок встановлення та налаштування системи	54
3.2. Структура інтерфейсу вебзастосунку	57
3.3. Тестування роботи програмного комплексу	71
Висновки до третього розділу	74

ВИСНОВКИ.....	76
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	78
ДОДАТКИ.....	81

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ЗНАЧЕНЬ

FFT – Fast Fourier Transform

СКБД – Система Керування Базою Даних

PWA – Progressive Web App

ACL – Access Control List

API – Application Programming Interface

S3 – Simple Storage Service

SDK – Software Development Kit

JWT – JSON Web Token

TTFB – Time To First Byte

AWS – Amazon Web Services

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність розробки вебзастосунку для інтерактивного прослуховування та візуалізації музичного контенту зумовлена стрімким розвитком технологій клієнтської обробки медіаданих та зміною парадигми дистрибуції цифрового контенту, зі звичайного пасивного прослуховування застарілих репозиторіїв музики.

Згідно зі звітами IFPI Global Music Report, частка стрімінгових доходів становить понад 67% світового ринку звукозапису. Проте висока конкуренція серед платформ висуває нові вимоги до досвіду користувача під використання цих самих платформ: сучасний користувач потребує інструментів прямого впливу на контент. Це підтверджується активним впровадженням відповідних функцій у Spotify та «Spatial Audio» в Apple Music, що свідчить про перехід від лінійного відтворення до абсолютно нового сприйняття звуку, де візуальний ряд (спектрограми, анімації) та незвичні звукові ефекти є невід'ємною частиною медіапотоків.

Технологічна актуальність роботи полягає у вирішенні проблеми серверного перевантаження. Традиційні методи обробки аудіо (наприклад, нарізка треків для прев'ю або створення рингтонів) вимагають значних ресурсів на стороні сервера. Впровадження бібліотек на базі WebAssembly дозволяє реалізувати ці функції безпосередньо в браузері клієнта, оскільки сучасні пристрої можуть собі дозволити такі операції без втрати продуктивності.

Критично важливим аспектом є питання безпеки та інтелектуальної власності. Велика кількість аматорського та незалежного контенту потребує складних механізмів модерації та контролю доступу. В умовах сьогодення, створення платформ, що забезпечують повний цикл перевірки треку перед публікацією та захист приватних плейлистів, є необхідною відповіддю на виклики несанкціонованого розповсюдження контенту, та втручання в особистий простір користувачів.

Зрештою, актуальність підсилюється розвитком концепції Offline-First.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдяки стандартам Progressive Web Apps, стає можливим подолання залежності музичних сервісів від стабільності інтернет-з'єднання або взагалі його наявності, оскільки завдяки впровадженню ресурсам користувач матиме змогу зберегти до певної кількості музичних композицій для подальшого прослуховування без наявного інтернет-підключення.

Метою роботи є проектування та розробка музичного вебзастосунку, що забезпечує повний цикл менеджменту аудіоконтенту: від клієнтської обробки та завантаження на платформу до візуалізації частотних характеристик звуку в реальному часі та автономного функціонування системи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати архітектурні рішення сучасних стрімінгових сервісів та методи інтерактивної взаємодії з медіаданими;
- дослідити та впровадити технологію WebAssembly для реалізації безсерверного редагування аудіофайлів на стороні клієнта;
- розробити програмні модулі для динамічної візуалізації аудіо з використанням Web Audio API та Canvas API;
- спроектувати та реалізувати систему розмежування прав доступу для керування приватністю контенту та багаторівневий механізм модерації;
- забезпечити підтримку офлайн-режиму за допомогою стандартів PWA, та механізмів локального кешування медіаданих.

Об'єкт дослідження - процес функціонування мультимедійних вебплатформ, що охоплює методи стрімінгу аудіоконтенту, керування користувацьким доступом та забезпечення інтерактивності в межах системи.

Предмет дослідження - методи та засоби клієнтської обробки аудіо сигналів, алгоритми реального часу для візуалізації частотного спектра звуку, а також архітектурні патерни побудови систем модерації та контролю доступу.

Методи аналізу та інформаційна база дослідження ґрунтуються на застосуванні порівняльного моніторингу існуючих аналогів мультимедійних платформ, а також на методах структурного аналізу UML за допомогою

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

побудови відповідних діаграм. Окрім цього, розрахунок частотного спектра звуку в реальному часі реалізовано на основі методів цифрової обробки сигналів та алгоритмів швидкого перетворення Фур'є.

Практичне значення отриманих результатів полягає у створенні працездатного вебзастосунку з автономним офлайн-плеєром, де винесення обробки сигналів на клієнтську сторону оптимізує використання серверних ресурсів, а інтегрована модель контролю доступу та багаторівневої модерації гарантує безпеку медіапростору, та швидкість опрацювання заявок завантаження.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ РОЗРОБКИ ВЕБ-СИСТЕМ ДЛЯ ІНТЕРАКТИВНОГО ПРОСЛУХОВУВАННЯ АУДІО КОНТЕНТУ

1.1. Постановка задачі

Сучасний етап розвитку цифрових технологій розподілу та збереження медіаконтенту характеризується стрімким збільшенням обсягів аудіоданих, що циркулюють у глобальній мережі. Традиційні підходи до побудови вебресурсів, які обмежуються виключно статичним відтворенням звукових файлів, наразі втрачають свою актуальність. Розвиток індустрії вимагає переходу до створення комплексних мультимедійних екосистем, здатних забезпечити високий рівень інтерактивності, низьку затримку передачі даних та розширені інструменти соціальної комунікації. У зв'язку з цим виникає гостра інженерна потреба в автоматизації процесів дистрибуції контенту, розробці засобів його інтерактивної візуалізації, а також у створенні швидких і надійних механізмів фільтрації та модерації користувацьких завантажень. Дослідження у цьому напрямку зумовлюють доцільність проектування та розробки багатофункціонального музичного вебзастосунку, який має концептуально об'єднати в собі функції високопродуктивного стрімінгу, підсистеми спектрального аналізу звукових хвиль та автономні компоненти для безперервного доступу до даних.

Для успішного вирішення поставленої інженерної задачі у межах кваліфікаційної роботи першочергово необхідно реалізувати модульний підхід до організації архітектури майбутнього програмного комплексу. Такий підхід дозволить чітко розмежувати зони відповідальності між ізольованими компонентами системи, забезпечити високу масштабованість платформи та спростити її подальше технологічне розширення. Під час проектування клієнт-серверного ядра застосунку замість класичних REST API ендпоінтів доцільно передбачити інтеграцію сучасного механізму серверних дій (Server Actions) фреймворку Next.js. Впровадження цієї технології дозволить безпосередньо зв'язати інтерфейсні компоненти клієнтської сторони з логікою виконання на

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

сервері в межах єдиних функціональних модулів. Це закладає основу для забезпечення наскрізної типізації даних за допомогою TypeScript та Prisma ORM - від моменту введення інформації користувачем до безпосереднього виконання транзакції у реляційній базі даних PostgreSQL, мінімізуючи ризики виникнення логічних помилок при передачі об'єктів.

Окремим критично важливим завданням у межах розробки архітектури безпеки та хмарної інтеграції є проєктування ефективного механізму взаємодії з віддаленим об'єктним сховищем AWS S3. З метою оптимізації мережевого трафіку та уникнення надмірного навантаження на обчислювальні потужності основного сервера застосунку, планується відмовитися від транзитного пересилання медіафайлів. Замість цього підлягає реалізації стратегія прямих завантажень контенту з браузера користувача до хмарної інфраструктури. Для забезпечення безпеки цієї операції необхідно спроектувати алгоритм динамічної генерації тимчасових підписаних посилань (Presigned URLs) на стороні сервера. Таке рішення дозволить клієнтському модулю безпечно передавати великі обсяги бінарних даних безпосередньо у сховище, гарантуючи високу швидкість обробки черг навіть за умов низької пропускної здатності каналу зв'язку у кінцевого споживача. Разом із цим, постановка задачі передбачає розробку та впровадження надійної підсистеми автентифікації та розмежування прав доступу на основі криптографічних токенів JWT.

У цей же суцільний контур безпеки та контролю якості даних необхідно інтегрувати автоматизований конвеєр дворівневої модерації завантажуваного контенту. Цей механізм має забезпечувати ізоляцію нових аудіофайлів у статусі чернеток до моменту їх перевірки відповідальними особами, а також підтримувати управління скарг користувачів із автоматичним зворотним зв'язком через email-сповіщення.

Фінальним технологічним процесом, що підлягає проєктуванню, є забезпечення безперебійної роботи медіаплеєра в умовах повної відсутності інтернет-підключення. Для вирішення цієї задачі планується впровадження

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

стандартів Progressive Web Apps на базі життєвого циклу Service Workers, що дозволить реалізувати стратегію локального кешування інтерфейсу та медіаресурсів в IndexedDB, гарантуючи безперервний доступ до обраних композицій через адаптивний офлайн-режим.

За умови виконання всіх окреслених інженерних кроків створюваний програмний комплекс забезпечить оптимальний баланс між швидкодією, безпекою та відмовостійкістю.

1.2. Аналіз аналогів програмного продукту

На сьогодні ринок стрімінгових сервісів та аудіоплатформ є надзвичайно насиченим, що зумовлює необхідність проведення порівняльного аналізу існуючих рішень для виявлення їхніх архітектурних переваг та функціональних обмежень. Для детального аналізу було обрано три найбільш релевантні системи.

Першим і одним із найближчих аналогів за своєю моделлю взаємодії з аудиторією виступає міжнародна платформа **SoundCloud**. Дана система повністю реалізує концепцію відкритої публікації, де кожен зареєстрований учасник має можливість виступати одночасно у ролі слухача та незалежного автора, завантажуючи власний оригінальний аудіоконтент.

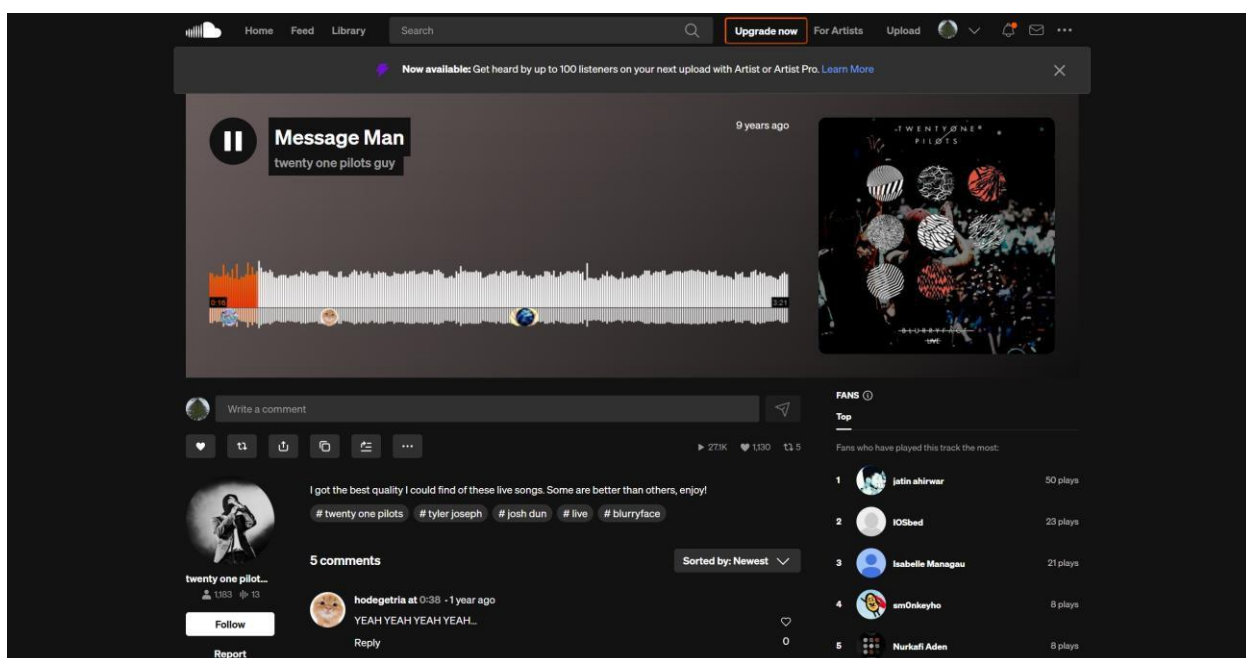


Рис. 1.1. Відображення треку на платформі SoundCloud

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

Також варто відзначити реалізацію коментарів, яка дозволяє користувачам залишати дописи, прив'язані до конкретного часового моменту треку, і під час програвання відображує їх безпосередньо на екрані та дублює в блоці під плеєром (рис. 1.2).

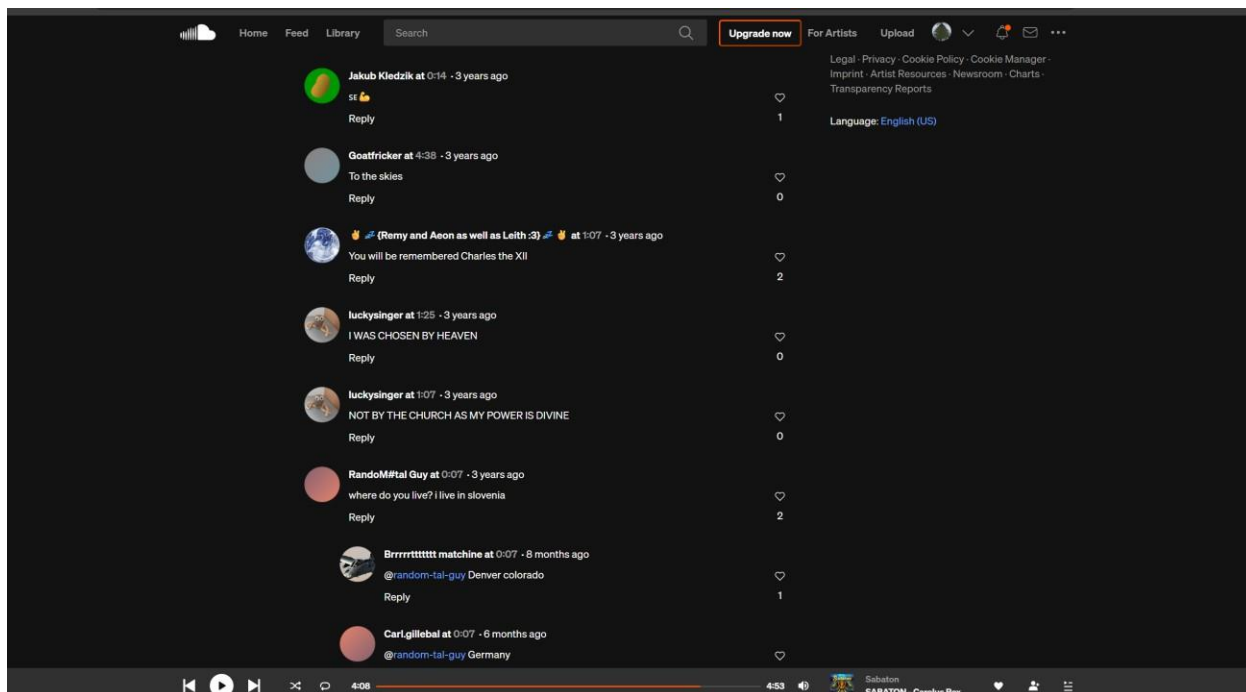


Рис. 1.2. Блок коментарів користувачів на платформі SoundCloud

З переваг варто зазначити відкритість платформи до авторів, звичайних осіб які просто хочуть поділитись своєю творчістю та легкість завантаження власного контенту (рис. 1.3). Та систему модерації яка основана на автоматичному скануванні завантажуваного контенту за для уникнення можливих проблем з авторськими правами, якщо ж перевірку пройдено, трек користувача одразу отримує помітку що його трек прийнято, та може ділитись ним з будь ким, навіть не зареєстрованим користувачем.

Не далеко від модерації знаходиться і неймовірна підтримка авторів, система завжди відстоює права своїх користувачів усіма зусиллями, та надає безліч зручних інструментів для них, в тому числі окрему панель для відслідковування активності по кожній з завантажених композицій.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

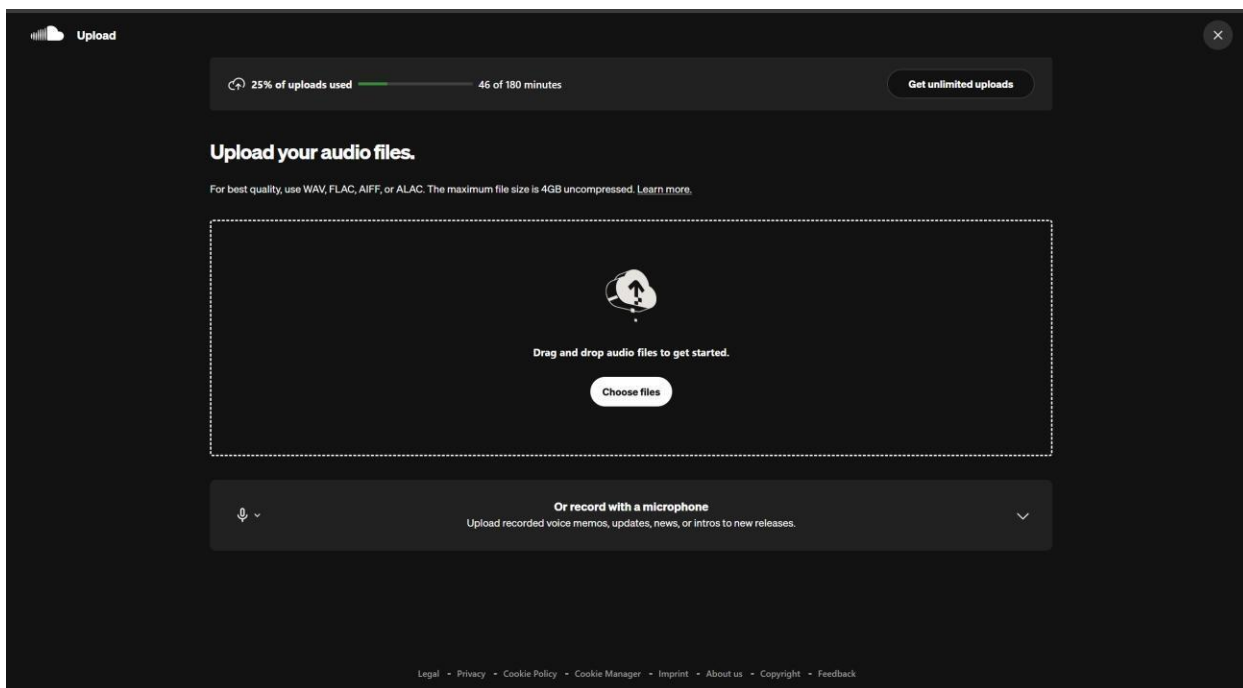


Рис. 1.3. Сторінка завантаження аудіо-контенту на платформі SoundCloud

Другим аналогом, що займає позицію світового лідера у сфері дистрибуції та стрімінгу ліцензійного аудіоконтенту, є платформа Spotify (рис. 1.4). Дослідницький інтерес до цієї системи в межах випускної кваліфікаційної роботи зумовлений її високоефективною архітектурою збереження, структурування та розподілу мультимедійних даних, що охоплюють не лише музичні треки, а й подкасти та аудіокниги.

Особливу увагу під час аналізу було приділено аналітичним підходам до формування рекомендаційних систем. Алгоритми платформи здійснюють комплексний математичний аналіз історії прослуховувань конкретного користувача, виявляють відповідні закономірності у його поведінкових факторах (прослуховування певних композицій вночі, та інших удень) та враховують специфіку географічного регіону для генерації персоналізованих добірок, що демонструє високу ефективність моделі автоматизації складання рекомендаційних добірок.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

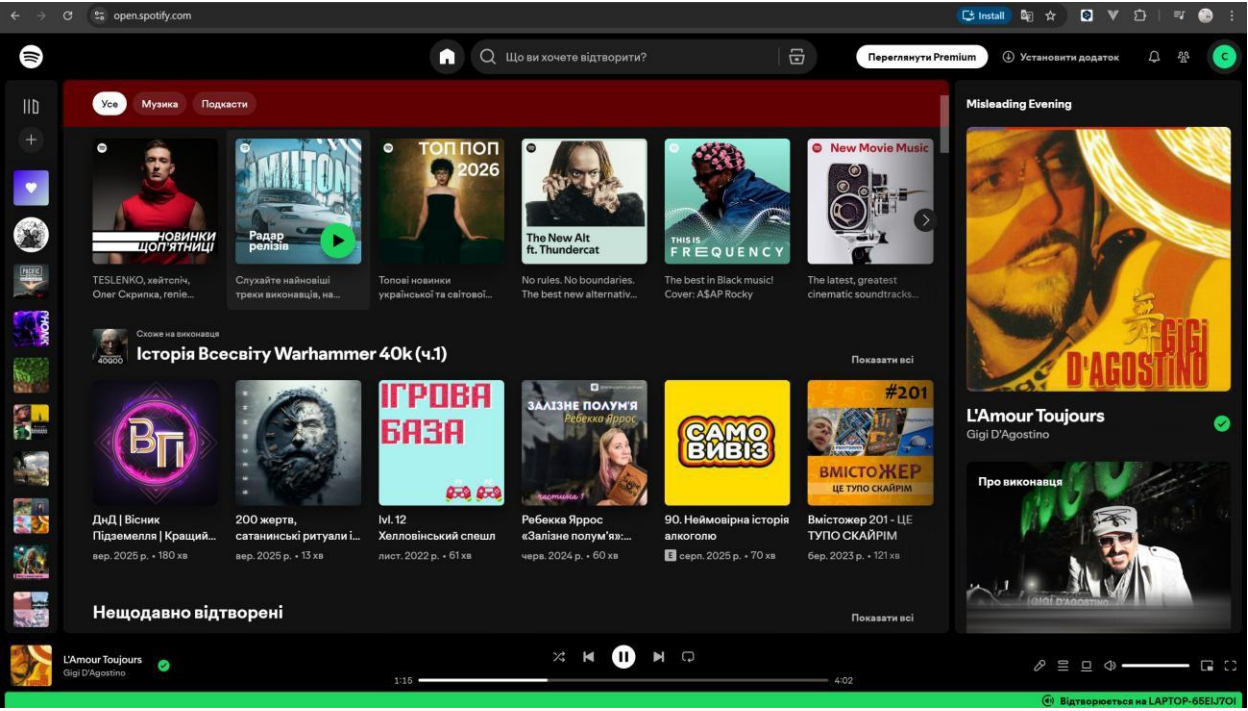


Рис. 1.4. Головна сторінка платформи Spotify

Окремо можна виділити можливість створення «Shared Playlists» за допомогою додавання співавторів (рис. 1.5), де декілька користувачів можуть редагувати один список відтворення, додавати або видаляти композиції.

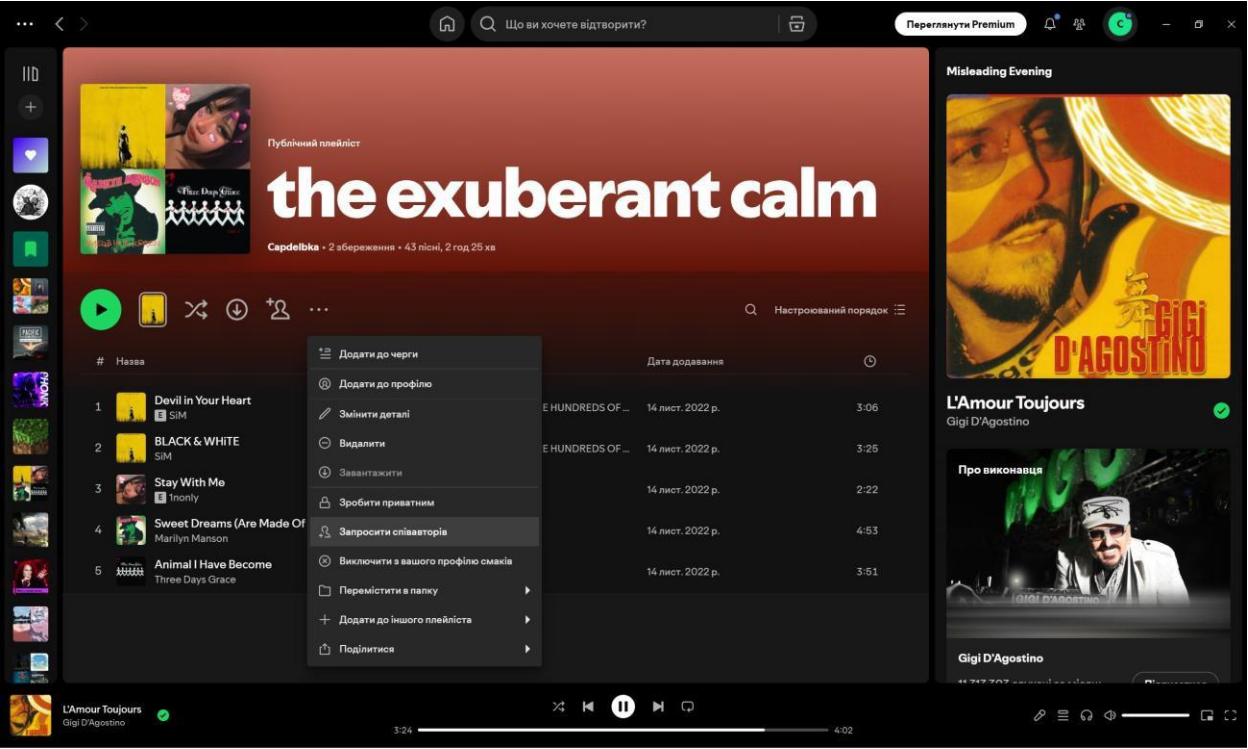


Рис. 1.5. Додавання співавторів на платформі Spotify

З переваг можна виділити наявність десктопних та мобільних версій з підтримкою офлайн-режиму, в якому користувач з преміум підпискою має змогу зберегти улюблені треки та прослуховувати їх без інтернет-підключення.

Головним недоліком платформи є високий ступінь закритості її екосистеми. Пересічний користувач позбавлений можливості швидкого публічного завантаження власних аудіоматеріалів, а модерація як правило займає набагато більше часу та цілком можливі постійні відмови через сувору внутрішню політику сервісу. Для завантаження існує окремий сервіс від розробників Spotify який несе назву «Spotify for Artists» (рис. 1.6)

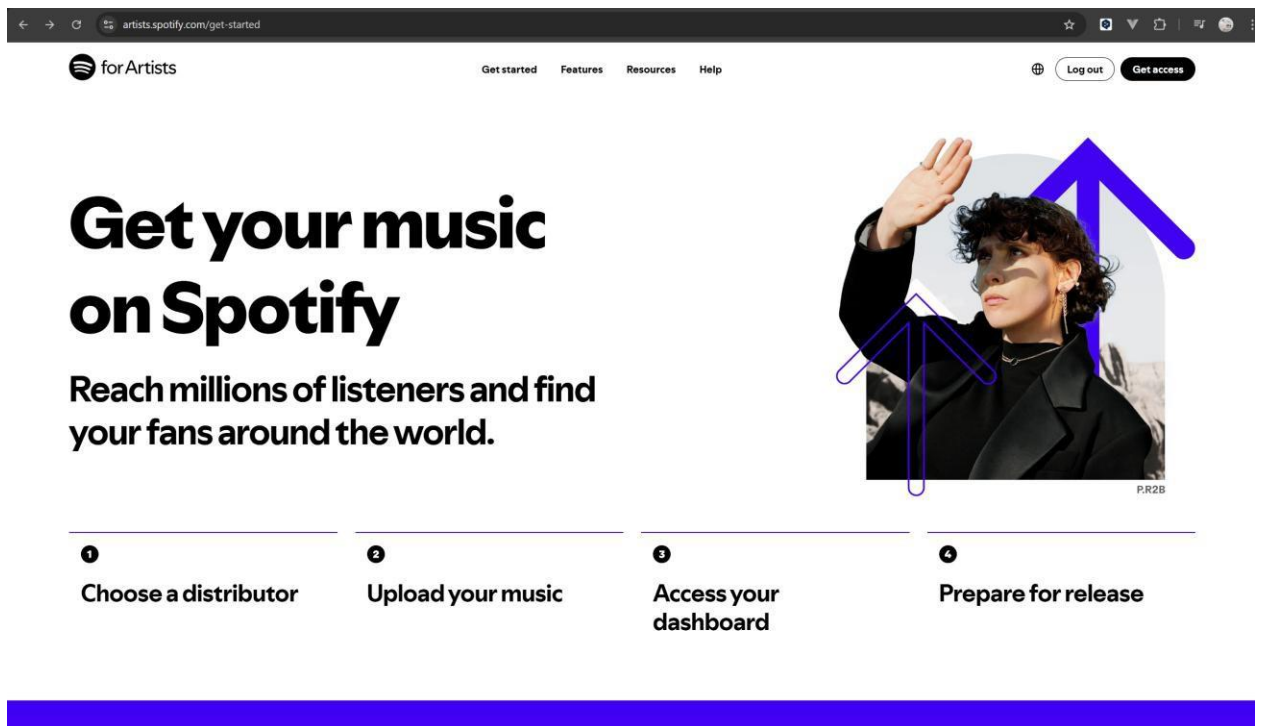


Рис. 1.6. Головна сторінка платформи для авторів від Spotify

Фактично він являється таким собі гідом, який містить в собі всю інформацію, з якою необхідно ознайомитись перед тим як пройти процедуру отримання доступу до частини платформи для завантаження власної творчості (рис. 1.7). Тут же після отримання доступу стає можливим розпочати процедуру завантаження власної композиції, яка являє собою багатоетапну перевірку на відповідність правилам платформи.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

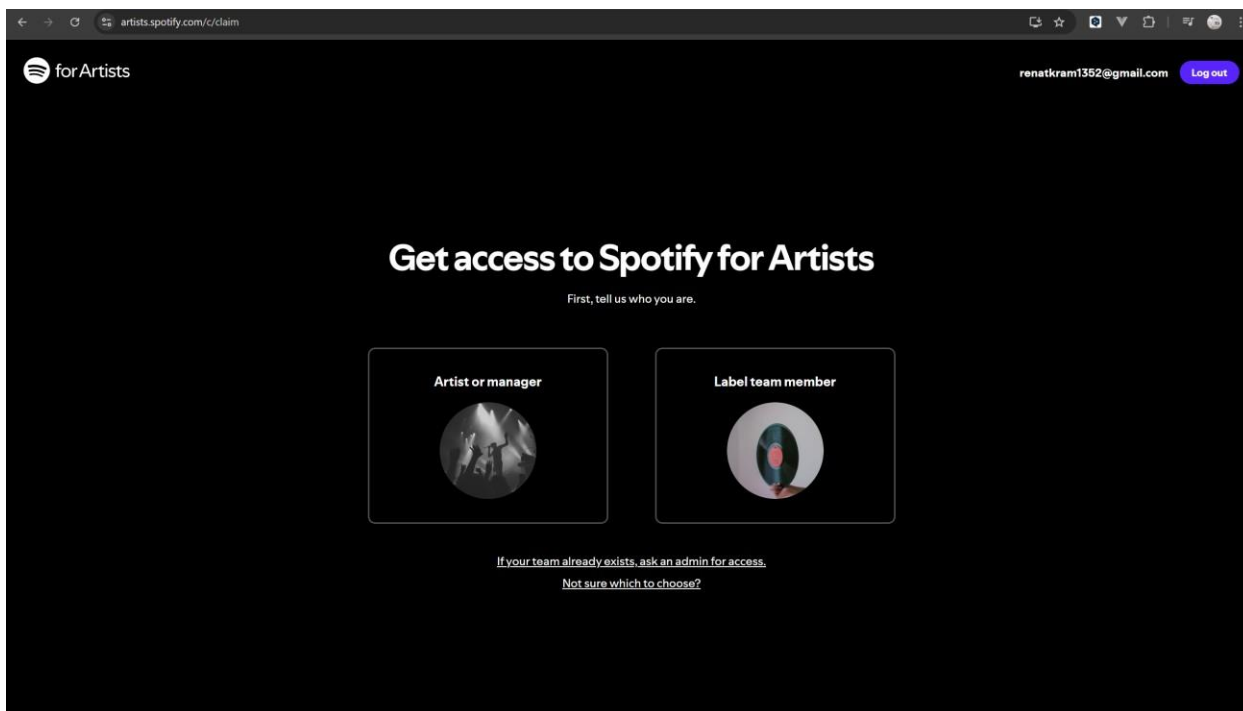


Рис. 1.7. Форма для отримання доступу до завантаження треків на платформі Spotify

Останнім аналогом для порівняння та вивчення є французький медіасервіс Deezer. Попри меншу глобальну популярність порівняно з лідерами ринку, дана платформа володіє унікальними технологічними перевагами. Зокрема, Deezer став одним із перших сервісів, який масштабував підтримку Hi-Fi звуку, задовольнивши потреби вибагливого кола користувачів у прослуховуванні музики без втрати якості.

Крім того, розробники активно впроваджують інтелектуальні алгоритми рекомендацій на основі машинного навчання, прикладом чого є фірмова система динамічного підбору контенту «Flow», яка підлаштовує чергу відтворення під настрій слухача, базуючись на тому що користувач слухає і в який період часу, паралельно зіставляючи список дібраних треків із обраними в ручну користувачем. Що і дозволяє підібрати список композицій з високою точністю.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

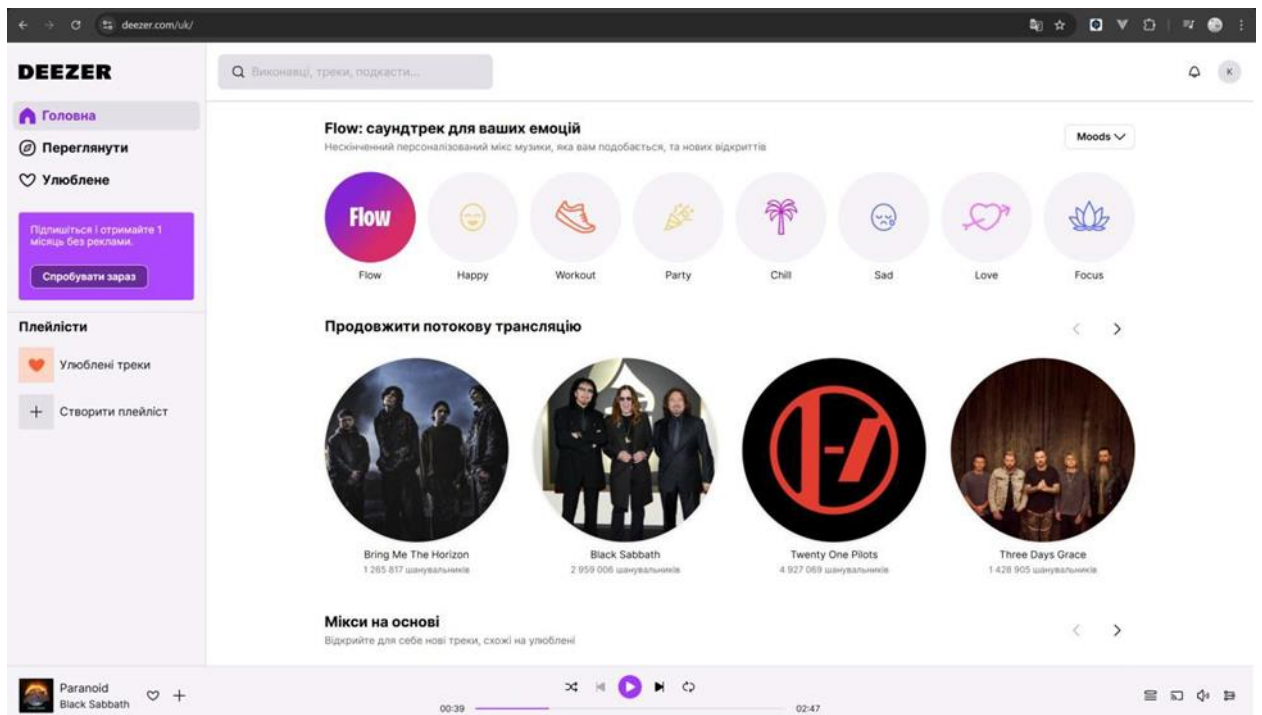


Рис. 1.8. Головна сторінка платформи Deezer

Аналіз користувацького інтерфейсу та функціональних можливостей платформи дозволив виділити низку її архітектурних переваг та недоліків. Серед ключових переваг Deezer варто відзначити інтуїтивно зрозумілий, ергономічний та неперевантажений зайвими елементами інтерфейс, що забезпечує високий рівень зручності під час навігації каталогом медіаресурсів, та швидкий час до опанування усіх можливих інструментів платформи.

Водночас програмний комплекс має вагомі функціональні обмеження. Зокрема, повноцінний офлайн-режим та трансляція звуку високої чіткості заблоковані у базовій версії. Крім того, вбудований інтерфейс візуалізації аудіопотоку є повністю статичним. У контексті дистрибуції контенту на платформі зафіксовано суворі інфраструктурні ліміти: незалежні автори позбавлені можливості безпосереднього публічного завантаження власних треків для спільного доступу. Замість цього сервіс пропонує взаємодію через обмежений перелік офіційних посередників-дистриб'юторів, які здійснюють

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

тривалу процедуру пре-модерації та ліцензування завантажуваного контенту. Проте, як альтернативне інженерне рішення, на сайті реалізовано можливість імпорту власних аудіофайлів у приватному режимі (рис. 1.9). У такому сценарії використання архітектура Deezer функціонує в режимі ізольованого хмарного сховища, призначеного суто для персонального збереження та прослуховування локальних медіафайлів користувача.

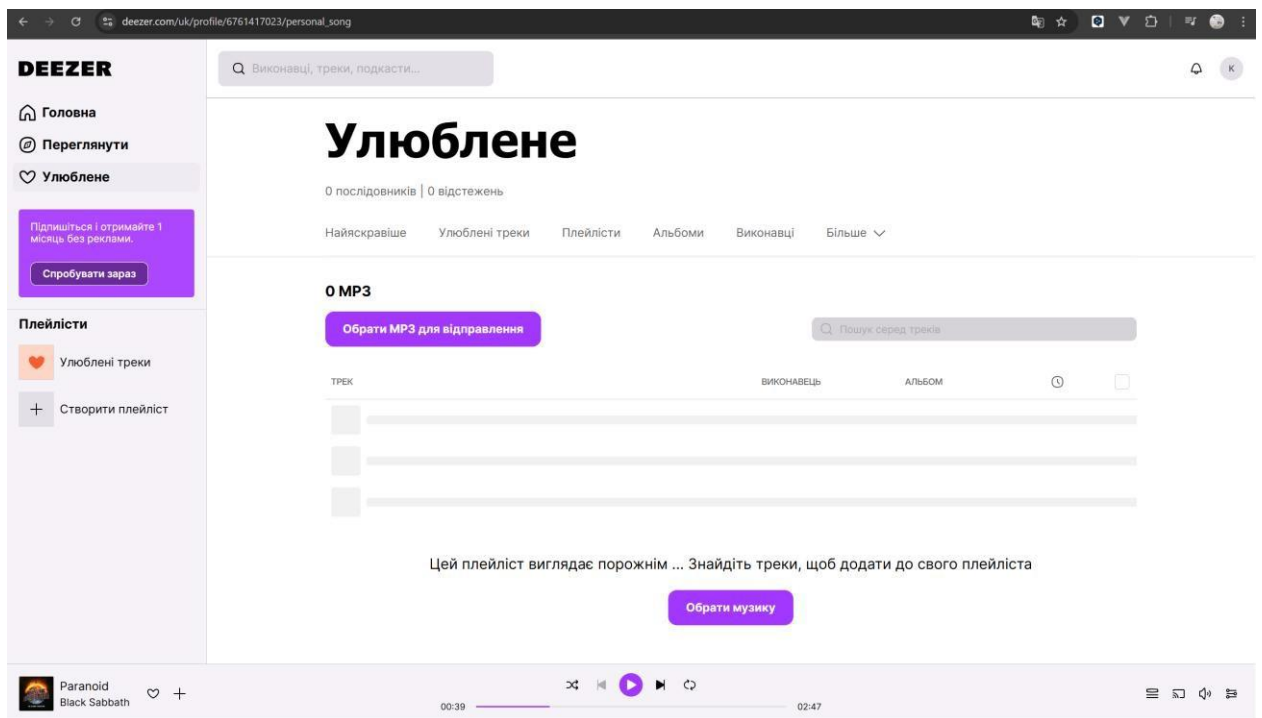


Рис. 1.9. Сторінка завантаження власних треків на платформі Deezer

Робота з плейлистами також зручно реалізована, є можливість обрати режим доступу до плейлисту (публічний\приватний) та запросити інших користувачів системи до редагування списку доданих треків за допомогою запрошувального посилання (рис. 1.10), де можна вказати текст повідомлення яке користувач побачить в окремому блоці після переходу за цим посиланням.

Впровадження такого функціоналу спільного редагування дозволяє трансформувати статичні списки відтворення в інтерактивні соціальні хаби, де користувачі можуть спільно формувати музичні добірки в реальному часі.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

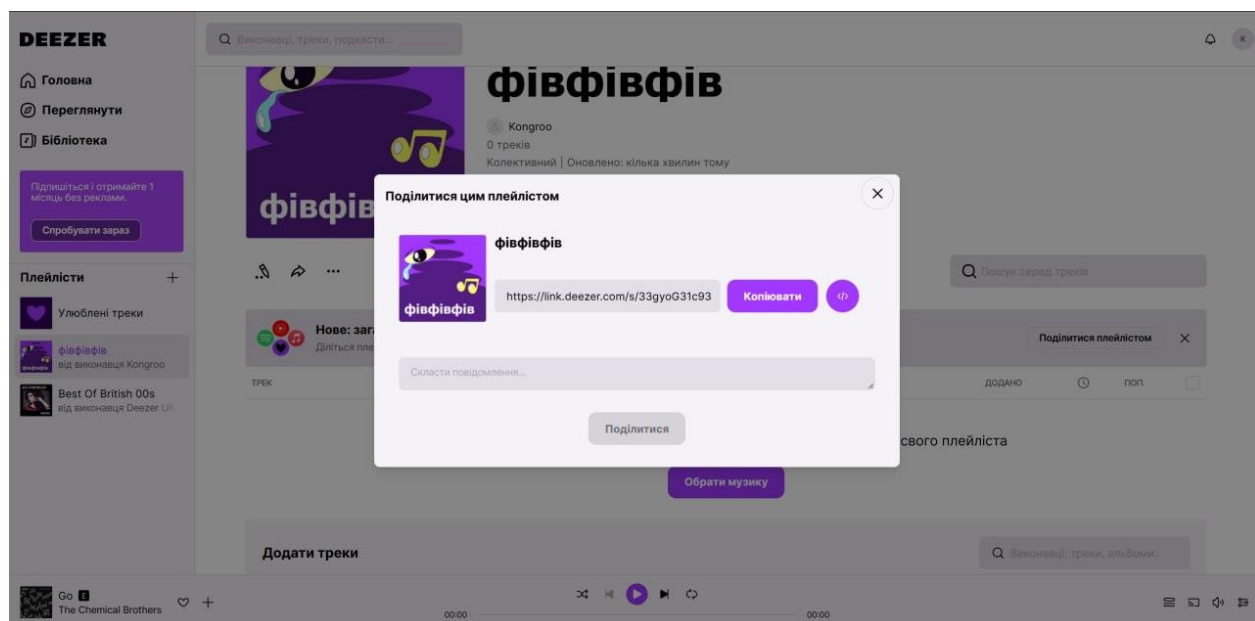


Рис. 1.10. Модальне вікно додавання співавторів

Таблиця 1.1

Таблиця порівняння музичних платформ

Функція / Платформа	SoundCloud	Spotify	Deezer	Monkefy
Завантаження контенту користувачем	Так	Ні	Ні	Так
Пошук та фільтрація композицій	Так	Так	Так	Так
Клієнтське редагування	Ні	Spotify For Artists	Ні	Так
Інтеграція рекомендаційних моделей на основі штучного інтелекту	Ні	Так	Так	Ні
Інтерактивна візуалізація звуку	Базова (статична хвиля)	Ні	Ні	Розширена (кругова/лінійна)
Коментування з прив'язкою до часу	Так	Ні	Ні	Так (тайм-код коментарі)
Офлайн-режим	Так	Так	Так	Так
Механізм модерації	Алгоритмічна/Пост-фактум	ШІ Аналіз + Перевірка людиною	Алгоритмічна	Попередня/Пост-фактум для скарг
Керування доступом плейлистів (ACL)	Базове (Public/Private)	Спільні плейлисти	Спільні плейлисти	Розширене керування правами доступу

Статистика для авторів	Так. Розширена	Окремий сервіс Spotify for Artists	Окремий сервіс Deezer for Creators	Так
------------------------	----------------	------------------------------------	------------------------------------	-----

Аналіз показав, що хоча існуючі сервіси мають потужний функціонал, вони або є закритими для вільних публікацій (Spotify, Deezer), або не надають можливості глибокої графічної взаємодії зі звуком у реальному часі. Розроблювана система заповнює цю нішу, пропонуючи поєднання соціальної платформи з відкритим завантаженням та інструментів інтерактивної візуалізації.

1.3. Вибір архітектури розроблюваної системи

Варто розуміти, що фактично архітектура програмного забезпечення являє собою фундаментальну організацію системи, що втілена в її компонентах, їхніх взаємозв'язках між собою та із зовнішнім середовищем. Вона визначає принципи, що спрямовують проєктування та розвиток продукту. Головною метою архітектурного проєктування є мінімізація трудовитрат на етапах розробки, тестування та подальшої підтримки системи, забезпечуючи при цьому високу продуктивність та масштабованість. Оскільки під час розробки можуть трапитись як конфлікти певних ідей так і неможливість реалізації у зв'язку з обраним стеком технологій попередній аналіз та вибір технології так необхідний.

Процес побудови архітектури є комплексним та ітераційним. Він базується на аналізі архітектурно значущих вимог. Під час аналізу альтернативних шляхів реалізації системи обов'язок розробника полягає у виборі найбільш збалансованого рішення для вирішення комплексної проблеми, що відповідає обмеженням проєкту та очікуванням зацікавлених сторін будь-то замовник чи сам розробник.

Для об'єктивного оцінювання архітектурних патернів було проведено порівняльний аналіз трьох найбільш поширених парадигм за ключовими

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

технічними визначеннями (табл. 1.2).

Таблиця 1.2

Порівняльна таблиця архітектурних шаблонів проєктування

Критерій порівняння	Монолітна архітектура	Мікросервісна архітектура	Модульна архітектура
Швидкість початкової розробки	Висока	Низька	Середня
Складність розгортання	Мінімальна	Висока	Середня
Синхронізація даних в контексті транзакцій з базою даних	Висока	Низька	Висока
Міжсервісні затримки	Відсутні	Присутні. Залежно від обраного протоколу	Мінімальні
Ізоляція критичних збоїв	Низька	Найвища	Середня
Вимоги до обчислювальних ресурсів	Середні	Високі	Середні

На основі проведеного аналізу, для практичної реалізації інтерактивної музичної платформи було обрано клієнт-серверну модель у поєднанні з концепцією модернізованого модульного моноліту. Застосована модель є галузевим стандартом для сучасних високонавантажених вебзастосунків і передбачає суворий розподіл системних функцій. Набір серверних компонентів забезпечує обробку центральної бізнес-логіки, пряму взаємодію з реляційною базою даних, виконання операцій з файлами та суворе управління матрицею прав доступу. Водночас клієнтська частина відповідає за безпосереднє представлення даних кінцевому користувачу, забезпечення високого рівня інтерактивності через механізми реактивності користувацького інтерфейсу, низькорівневу обробку звукового потоку та рендеринг графічних спектрограм, а також за трансляцію асинхронних запитів на сервер для їхнього подальшого опрацювання.

Мікросервісна архітектура розглядалася як один із пріоритетних варіантів для забезпечення максимальної масштабованості та гнучкості розробки. Дана концепція передбачала повний розподіл інфраструктури вебплатформи на автономні, ізольовані сервіси, зокрема винесення обробки

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

аудіопотоків, механізмів автентифікації користувачів та генерації спектрограм у незалежні функціональні вузли. Проте детальний аналіз специфіки проєкту змусив відхилити цей підхід через низку стримувальних чинників, першочерговим серед яких стала надмірна складність проектування інфраструктури. Необхідність розгортання додаткових рівнів оркестрації контейнерів, конфігурування систем міжсервісного моніторингу та логування суттєво збільшувала загальні терміни розробки системи, що було недоцільно на поточному етапі життєвого циклу програмного комплексу.

Поряд із цим виникли побоювання щодо появи мережевих затримок під час комунікації між розподіленими вузлами. У мультимедійній системі, де критично важливою виступає миттєва реакція інтерфейсу на дії кінцевого користувача, як-от оперативний імпорт медіафайлів чи синхронне коментування на часових мітках, постійний мережевий транзит даних між мікросервісами міг спровокувати небажаний ефект який би впливав на швидкодію додатка. Додатковим аргументом проти мікросервісного підходу стала проблема забезпечення цілісності інформаційних потоків. Практична реалізація та підтримка складних розподілених транзакцій між окремими базами даних кожного сервісу виявилася технічно надлишковою для поточного масштабу платформи й створювала невиправдані ризики порушення консистентності даних під час паралельних запитів.

Модульна архітектура виступила проміжною архітектурною альтернативою, що пропонує чітке логічне розділення компонентів у межах єдиного простору сирцевого коду. Це дозволяє структурувати внутрішні зв'язки додатка за принципом слабкої зв'язаності, що в майбутньому значно полегшить вірогідний перехід до мікросервісів у разі виникнення такої інфраструктурної потреби. З огляду на це, хоча елементи модульності були інтегровані в структуру проєкту, як основний каркас системи було затверджено саме монолітний підхід.

Таке інженерне рішення було обумовлене насамперед аспектами забезпечення цілісності даних, адже використання єдиної централізованої бази

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

даних у межах моноліту дозволяє гарантувати відсутність конфліктів інформації на рівні SQL-запитів без ризику виникнення розсинхронізації між модулями.

Крім того, монолітний підхід забезпечує високу продуктивність програмного комплексу на початкових етапах його життєвого циклу. Взаємодія між усіма функціями та модулями системи відбувається всередині одного процесу, що гарантує мінімальний час відгуку сервісу та є критично важливим для збереження позитивного користувацького досвіду під час роботи з інтерактивним плеєром.

1.4. Обґрунтування вибору засобів реалізації та вимоги до апаратного забезпечення

Для реалізації системи було обрано інструментарій, що дозволяє поєднати можливості серверного рендерингу із високою інтерактивністю клієнтської частини.

Як основну мову програмування було обрано TypeScript, її використання дозволяє мінімізувати кількість помилок ще на етапі написання коду завдяки строгій статичній типізації, а також забезпечує ефективну взаємодію з інструментами автодоповнення в середовищі розробки. У межах монолітної архітектури фреймворку Next.js це гарантує наскрізну відповідність структур даних між серверною та клієнтською частинами без ризику розходження типів. Окреслений підхід є критично важливим для безпомилкової обробки об'єктів медіафайлів та складних сутностей плейлистів.

Як основний Fullstack-фреймворк для побудови вебзастосунку було обрано Next.js. Дане технологічне рішення базується на бібліотеці React і пропонує ефективний гібридний підхід до архітектури додатка. Цей механізм дозволяє раціонально перерозподілити обчислювальні ресурси на сторону сервера під час генерування статичних сторінок, які не потребують складної інтерактивної взаємодії. У такий спосіб досягається суттєве розвантаження клієнтської частини додатка, що звільняє апаратні потужності пристрою

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

користувача для виконання ресурсномістких операцій, наприклад, розрахунку та відмальовування спектральних діаграм, які генеруються безпосередньо у браузері.

Поряд із цим, архітектурна модель Next.js надає інструментарій Server Actions, який дозволяє описувати ключову серверну логіку у вигляді ізолюваних асинхронних функцій. Завдяки цьому операції імпорту аудіотреків, авторизації або додавання текстових коментарів виконуються без створення додаткових REST API ендпоінтів. Впровадження даного інструменту значно спрощує загальну кодову базу, зменшує обсяг клієнтського бандла, що завантажується в браузер, та гарантує високий рівень безпеки під час виконання конфіденційних операцій на стороні сервера.

Для збереження даних обрано Supabase, що базується на реляційній системі керування базами даних PostgreSQL. PostgreSQL є стандартом надійності, що підтримує складні зв'язки та транзакції за ACID необхідні для менеджменту користувачів та їх медіатек. Що ж до переваг безпосередньо платформи Supabase, перш за все, вона надає готову та надійну систему автентифікації та авторизації користувачів, що дозволяє зосередитися на бізнес-логіці без необхідності проектування складних механізмів безпеки з нуля. Доступ до об'єктного сховища медіафайлів у Supabase поєднує в собі простоту інтеграції з гнучкістю налаштувань, що є критично важливим для масштабування системи в майбутньому.

Важливою архітектурною особливістю є можливість використання вбудованих плагінів безпосередньо в екосистемі платформи, зокрема розширення pg_cron. Це дозволяє автоматизувати періодичні завдання бази даних, як-от очищення застарілих чернеток або оновлення статистичних показників авторів, без залучення сторонніх сервісів. Завдяки легкій процедурі розгортання та повній сумісності з хмарними хостингами, такими як Vercel, Supabase забезпечує безперервний цикл розробки та стабільну роботу додатку в мережі, гарантуючи при цьому високу продуктивність при обробці запитів до даних.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

Для зберігання медіафайлів (аудіозаписів у форматах MP3/WAV та графічних зображень обкладинок) було обрано сервіс AWS S3. Оскільки база даних призначена виключно для метаданих, використання S3 дозволяє винести навантаження з обслуговування файлів на спеціалізовану інфраструктуру.

Ключовим фактором вибору саме цього рішення стала висока безпека, що реалізується через механізм підписаних посилань. Це дозволяє створювати систему модерації та приватного доступу до треків шляхом генерації тимчасових токенів, що унеможливорює пряме завантаження контенту неавторизованими користувачами, та нівелює можливість використання сховища в свої цілях злоумисниками. Також важливу роль відіграє глибока інтеграція з офіційним SDK для Node.js, що спрощує взаємодію хмарного сховища із серверними екшенами. Такий підхід забезпечує швидку передачу потоку даних від клієнта до сховища, гарантуючи необхідну швидкодію для стабільної роботи вебзастосунку в умовах реального часу.

Для забезпечення стабільного функціонування, високої швидкодії та масштабованості розроблюваної системи під час проєктування було сформовано чіткий перелік технічних вимог до інфраструктури розгортання та пристроїв кінцевих користувачів. Ці критерії охоплюють як апаратні можливості клієнтської сторони, необхідні для безперебійного аналізу аудіо та плавного графічного рендерингу візуалізацій, так і параметри серверного середовища виконання й хмарних сховищ (табл. 1.3).

Таблиця 1.3

Вимоги до апаратного забезпечення

Найменування області	Технічні параметри	Мінімальні технічні
Клієнське середовище виконання	Процесор	Тактова частота мінімум 2.0 ГГц для забезпечення обчислень алгоритмів FFT.
	Оперативна пам'ять	Обсягом не менше 4 ГБ для буферизації аудіопотоку та стабільного функціонування SPA-інтерфейсу.

	Відеокарта (GPU)	Графічний ядро із підтримкою стандартів WebGL/Canvas
	Мережеве з'єднання	Доступ до мережі Інтернет зі швидкістю не менше 15 Мбіт/с для
Серверне середовище та зберігання даних	Платформа для розгортання	Середовище виконання Node.js (версія 22.x або новіша)
	СКБД	Реляційна система управління базами даних PostgreSQL
	Обчислювальні потужності	Хмарний хостинг із підтримкою вертикального або горизонтального автоматичного масштабування під пікові навантаження.
	Об'єктне сховище	Інтеграція з хмарною інфраструктурою AWS S3 для надійного збереження і дистрибуції бінарних аудіофайлів.
Безпека	Режим роботи та доступ	Рольова модель обмеження прав доступу (ACL) для захисту від несанкціонованого керування приватними плейлистами.
	Захист облікових даних	Обов'язкове криптографічне хешування паролів користувачів за допомогою відповідних алгоритмів на стороні сервера.

Висновки до першого розділу

В межах першого розділу було здійснено всебічне дослідження теоретичних та практичних засад створення систем для інтерактивної взаємодії з медіаконтентом. Проведений порівняльний аналіз існуючих рішень у галузі музичного стрімінгу дозволив виявити стійку тенденцію до децентралізації обчислювальних процесів та зростання попиту на інструменти візуального супроводу аудіоконтенту.

Особлива увага в ході аналізу була приділена вибору технологічного стеку, який зміг би забезпечити високий рівень швидкодії при збереженні безпеки передачі даних. Використання сучасних вебфреймворків у поєднанні з хмарними об'єктними сховищами та технологіями клієнтського виконання програмного коду визначено як найбільш оптимальний підхід для побудови

масштабованої платформи. У цьому контексті детально досліджено переваги делегування ресурсомістких операцій зчитування сигналів безпосередньо на сторону клієнта, що дозволяє суттєво знизити постійне апаратне навантаження на центральний сервер. Таке поєднання технологій дає змогу реалізувати гнучкі моделі керування доступом та забезпечити стабільність роботи сервісу в умовах змінного мережевого з'єднання, що є критично важливим для сучасних розподілених вебсистем.

За результатами аналізу було сформовано модель функціонування програмного продукту, яка базується на принципах інтерактивності, соціальної залученості типового користувача та безпеці у використанні. Визначена стратегія розробки передбачає створення середовища, де візуалізація та цифровий аналіз звуку інтегровані в єдиний інтерфейс, що забезпечує користувачу повний контроль над медіаданими. Сформована постановка задачі та обґрунтовані етапи реалізації створюють все необхідне для переходу до наступного етапу роботи.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		29

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ІНТЕРАКТИВНОГО ПРОСЛУХОВУВАННЯ ТА ВІЗУАЛІЗАЦІЇ МУЗИЧНОГО КОНТЕНТУ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Проектування архітектури вебзастосунку для інтерактивного прослуховування музичного контенту базується на забезпеченні високого рівня інтерактивності та чіткому розмежуванні функціональних обов'язків між різними групами користувачів (описано далі). Система орієнтована на надання повного циклу взаємодії з музичним контентом: від анонімного прослуховування до керування власною медiateкою та адміністрування глобального пулу треків.

Функціональні вимоги до проектованої системи передбачають розмежування прав доступу на основі впровадження чотирьох базових користувацьких ролей. Початковим рівнем взаємодії володіє роль гостя, яка описує неавторизованого відвідувача платформи. Дана категорія користувачів має доступ виключно до базового функціоналу системи, що включає можливість прослуховування публічно доступного аудіоконтенту та перегляду загальних метаданих музичних композицій. Наступним етапом градації прав є роль зареєстрованого користувача, який отримує значно ширший спектр інтерактивних можливостей. Такий учасник системи спроможний імпортувати власний контент, формувати персональні й публічні плейлисти, взаємодіяти з іншими авторами.

Для забезпечення правомірного та безпечного функціонування вебдодатку в архітектуру закладено роль модератора. На дану відповідальну особу покладаються обов'язки щодо моніторингу чистоти інформаційного простору платформи, що реалізується шляхом перевірки нових завантажень з, а також оперативного реагування на скарги користувачів стосовно порушення авторських прав чи правил спільноти. Найвищим рівнем доступу та

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

повноважень у системі наділена роль адміністратора. Цей профіль призначений для здійснення повного контролю за глобальними технічними параметрами вебзастосунку.

Для формалізації взаємодії між об'єктами системи та програмою було розроблено діаграму варіантів використання представлену на рис. 2.1. Ця модель відображає сценарії роботи, що включають як стандартні операції відтворення, так і специфічні для платформи функції, такі як коментування з часовими мітками та динамічне перемикання типів візуалізації частот.



Рис. 2.1. Діаграма варіантів використання системи «Monkefy»

Як видно з діаграми, система побудована на принципі успадкування прав доступу, де кожна вища роль включає в себе можливості попередньої, доповнюючи їх специфічними інструментами керування. Зокрема, сценарій «Прослуховування треків» для авторизованого користувача розширюється через механізм «Керування чергою прослуховування», а сценарій завантаження треків передбачає обов'язкову стадію пре-модерації. Більш детальний опис кожного з основних варіантів використання системи наведено в таблицях нижче.

Таблиця 2.1

Специфікація варіанта використання «Завантаження власних треків»

Елемент опису	Зміст
Короткий опис	Дозволяє авторизованому користувачу додати аудіофайл до системи з можливістю попередньої обрізки.
Актор	Користувач
Передумова	Користувач авторизований у системі.
Результат	Створення чернетки завантажуваного аудіо, очікуючого на вердикт модератора.

Таблиця 2.2

Специфікація варіанта використання «Керування типом візуалізації»

Елемент опису	Зміст
Короткий опис	Зміна алгоритму відображення частотних характеристик звуку під час відтворення.
Актор	Користувач
Передумова	Активний режим прослуховування.
Результат	Зміна візуального супроводу аудіосигналу.

Таблиця 2.3

Специфікація варіанта використання «Керування чергою треків»

Елемент опису	Зміст
Короткий опис	Перевірка нових композицій на відповідність правилам платформи перед публікацією.
Актор	Модератор
Передумова	Наявність нових заявок у черзі.
Результат	Публікація треку в глобальному пулі або видалення

Таблиця 2.4

Специфікація варіанта використання «Керування плейлистами»

Елемент опису	Зміст
Короткий опис	Організація персональних добірок треків з налаштуванням прав доступу.
Актор	Користувач
Передумова	Наявність авторизації та завантажених треків у системі.
Результат	Збереження структурованої медіатеки з визначеними правами видимості.

Таблиця 2.5

Специфікація варіанта використання «Коментування завантажених треків»

Елемент опису	Зміст
Короткий опис	Додавання текстового відгуку, що синхронізується з конкретним моментом відтворення треку.
Актор	Користувач
Передумова	Активний режим прослуховування композиції. Або перехід на сторінку з інформацією про композицію.
Результат	Поява коментаря, який відображатиметься у відповідній вкладці для всіх користувачів, а у випадку візуалізації з коментарями, буде виникати у певний момент часу на екрані.

Таблиця 2.6

Специфікація варіанта використання «Керування чергою скарг»

Елемент опису	Зміст
Короткий опис	Реагування на повідомлення користувачів про невідповідний контент або порушення авторських прав.
Актор	Модератор
Передумова	Наявність активних скарг від користувачів у системній черзі.
Результат	Підтримка правомірного та якісного пулу композицій у системі.

Специфікація варіанта використання «Перегляд статистики по системі»

Елемент опису	Зміст
Короткий опис	Отримання узагальнених даних про активність користувачів та стан серверних ресурсів.
Актор	Адміністратор
Передумова	Вхід до панелі керування з правами адміністратора.
Результат	Оцінка ефективності роботи системи та використаних ресурсів

Виходячи з описаних вище специфікацій та визначених варіантів використання, можна зробити висновок, що розробка вебзастосунку для інтерактивного прослуховування та візуалізації музичного контенту, ведеться для того, щоб втілити в життя певну екосистему, для авторів та звичайних слухачів, де вони зможуть взаємодіяти один з одним певним чином. На відміну від класичних репозиторіїв музики, вебзастосунок розроблюваний в межах даної кваліфікаційної роботи створює інтерактивну екосистему, де функціонал прослуховування є лише базовим рівнем взаємодії. Пріоритетним напрямком розробки став розвиток інструментарію, що дозволяє користувачу безпосередньо впливати на форму та структуру контенту. Це реалізується через інтеграцію засобів цифрової обробки сигналів на стороні клієнта, що дає змогу маніпулювати тривалістю треків та отримувати миттєву візуальну відповідь у вигляді деталізованих спектрограм, які відображають фізичну природу звуку через амплітудно-частотні коливання.

Зручність експлуатації платформи забезпечується через відмову від громіздких інтерфейсів на користь мінімалістичного дизайну, де кожна дія користувача має передбачуваний результат. Завдяки впровадженню архітектури PWA (Progressive Web App), систему вдасться перетворити на універсальний інструмент, який у випадку складних мережеских умов, як от низька швидкість або взагалі її відсутність, надає користувачу доступ до збережених заранню композицій, навіть без доступу до мережі інтернету.

Архітектура зберігання даних базується на розмежуванні метаданих та

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		34

бінарних об'єктів. Медіафайли (аудіо та графічні обкладинки треків та списків відтворення) розміщуються у зовнішньому об'єктному сховищі AWS S3. Це забезпечує високу швидкість доступу до контенту та знижує навантаження на процесор основного сервера застосунку, який відповідає виключно за бізнес-логіку та управління сесіями користувачів.

Безпека системи реалізована через багатофакторну модель захисту. Автентифікація користувачів здійснюється за допомогою JSON Web Tokens (JWT), що передаються через захищені HTTP-only cookies. Доступ до файлів у сховищі обмежений механізмом Presigned URLs, який генерує тимчасові підписані посилання для кожного окремого запиту, унеможливаючи прямий несанкціонований доступ до ресурсів. Керування доступом до приватних колекцій та плейлистів виконується на основі моделі ACL, що перевіряється на рівні серверних екшенів перед виконанням будь-яких маніпуляцій з даними.

Соціальний функціонал системи включає механізм тайм-кодованого коментування, де кожен запис прив'язаний до конкретного цілочисельного значення часової шкали треку. Цілісність контенту в загальному доступі гарантується модулем пре-модерації: завантажені об'єкти отримують статус публічних лише після верифікації адміністратором або модератором. Обробка скарг та системні сповіщення про зміну статусу контенту автоматизовані через інтегровані поштові сервіси.

Об'єктно-орієнтована структура застосунку базується на взаємодії ключових сутностей: користувача, треку, плейлиста та звіту модерації. Дана модель дозволяє масштабувати систему шляхом додавання нових функціональних блоків без зміни базових протоколів безпеки та обміну даними.

Нефункціональні вимоги визначають критерії якості, продуктивності та безпеки, яким повинен відповідати вебзастосунок для забезпечення стабільної експлуатації. В межах дипломного проєкту ці вимоги розподілені за типами користувачів та технічними параметрами взаємодії з медіаконтентом.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

Зокрема, для сесій неавторизованих відвідувачів першочерговим критерієм продуктивності є забезпечення безперебійного стрімінгу публічних композицій та висока швидкість завантаження статичних сторінок без потреби в попередній ідентифікації пристрою. При цьому вимоги безпеки обмежують рівень доступу для даної категорії виключно операціями читання, що виключає можливість несанкціонованої модифікації метаданих треків або користувацьких коментарів.

Для автентифікованих у системі користувачів нефункціональні вимоги фокусуються на гарантуванні цілісності персональних даних, суворій ізоляції приватних плейлистів на рівні бази даних та забезпеченні мінімального часу відгуку системи під час виконання операцій асинхронного завантаження й клієнтського редагування контенту.

Нарешті, для персоналу управління, що охоплює ролі модераторів та адміністраторів платформи, критично важливими критеріями якості є надання пріоритетного доступу до черг обробки заявок, підвищена відмовостійкість інструментів адміністрування та гарантоване відображення актуального стану медіапростору в реальному часі без затримок синхронізації. Такий комплексний підхід до структурування нефункціональних параметрів дозволяє закласти надійний фундамент для подальшої програмної реалізації всіх модулів розроблюваного вебзастосунку.

За вимогами до часових характеристик реакції мультимедійної системи встановлюються чіткі інженерні критерії, що визначають динаміку взаємодії користувача з інтерфейсом та швидкість обробки медіаданих. Зокрема, час відгуку клієнтського інтерфейсу на інтерактивні дії, такі як натискання елементів керування відтворенням, перемикання аудіотреків або зміна поточного типу візуалізації, не повинен перевищувати 150 мс, що є критично важливим для забезпечення безперервної візуальної синхронізації та підтримки високого рівня ергономічності платформи. Водночас затримка початку відтворення програвання аудіо після ініціалізації запиту має відбуватися в межах 1–2 секунд, що планується досягти шляхом оптимізації

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		36

розміру чанків передачі даних та використання мереж доставки контенту хмарного об'єктного сховища. Продуктивність операцій, пов'язаних із низькорівневою обробкою звукових сигналів за допомогою бінарного рантайму WebAssembly на стороні клієнта, має становити від 3 до 8 секунд для файлів об'ємом до 15 МБ, залежно від апаратних обчислювальних потужностей кінцевого пристрою.

Окремим технологічним критерієм ефективності виступає синхронізація модулів графічного рендерингу, де частота оновлення динамічної спектрограми звуку повинна бути не меншою за 30 кадрів на секунду, а в умовах використання сучасних дисплеїв — адаптивно відповідати частоті 60 FPS для забезпечення плавності коливань відповідно до амплітуди звукового сигналу в реальному часі. Поряд із часовими характеристиками, фундаментальним критерієм надійності проектується безпека облікових даних користувачів. З метою мінімізації ризиків компрометації конфіденційної інформації у базі даних, у межах архітектури безпеки передбачається збереження паролів виключно у вигляді незворотних криптографічних хеш-сум із використанням стійкого алгоритму bcrypt із оптимально підбіраною кількістю ітерацій шифрування.

Також не маловажним аспектом є те що система має перехоплювати виняткові ситуації (наприклад, перевищення ліміту завантаження або помилка S3 сховища) та виводити інформативні повідомлення без переривання загального циклу роботи застосунку, для того щоб покращити досвід користувача, та не відлякувати його неперехопленими помилками які можуть призвести до повної зупинки функціонування системи.

Об'єктна структура системи спроектована за принципом розділення відповідальностей, де кожен функціональний вузол представлений окремим модулем серверних екшенів. Така організація дозволяє забезпечити наскрізну типізацію та безпечну взаємодію між клієнтською частиною та базою даних без необхідності створення проміжних шарів у вигляді контролерів чи сервісів.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

Центральне місце в архітектурі займають модулі, що забезпечують повний життєвий цикл медіаконтенту та управління правами користувачів.

Логічна структура основних компонентів системи побудована за принципом розподілу обов'язків, де першочерговим елементом виступає модуль управління автентифікацією. Даний вузол забезпечує безпечний вхід та реєстрацію користувачів у системі, інтегруючи логіку криптографічного хешування та механізми генерації безпечних токенів JWT. Модуль безпосередньо взаємодіє із сутністю User для проведення валідації облікових даних на стороні сервера та встановлення захищених сесійних HTTP-only cookies, що закладає надійну основу безпеки всього вебзастосунку.

Наступним, найбільш складним архітектурним вузлом системи є модуль обробки медіаконтенту, який реалізує дворівневу логіку створення та верифікації цифрових матеріалів. Він відповідає за первинну ініціалізацію чернеток аудіозаписів, фіналізацію текстових і технічних метаданих, а також за гнучке управління статусами сутностей Track у базі даних. Серверні дії даного модуля забезпечують автоматичну генерацію тимчасових підписаних посилань (Presigned URLs) для взаємодії з хмарним об'єктним сховищем AWS S3, що дозволяє клієнтському браузеру безпечно та асинхронно завантажувати важкі бінарні файли безпосередньо у хмару, повністю минаючи транзитний сервер застосунку і заощаджуючи його обчислювальні ресурси.

Для забезпечення правомірності розміщеного цифрового контенту та підтримання безпеки інформаційного простору в архітектуру інтегровано модуль системної модерації та звітності.

На основі описаних вимог та архітектури системи фактично на модулях, можна вибудовувати діаграму логічної структури (рис. 2.2) яка фактично являє собою коротке представлення основних модулів системи які і вибудовують ядро вебзастосунку, на ній показано взаємодію між різними компонентами від сутності користувача до хмарного сховища.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		38

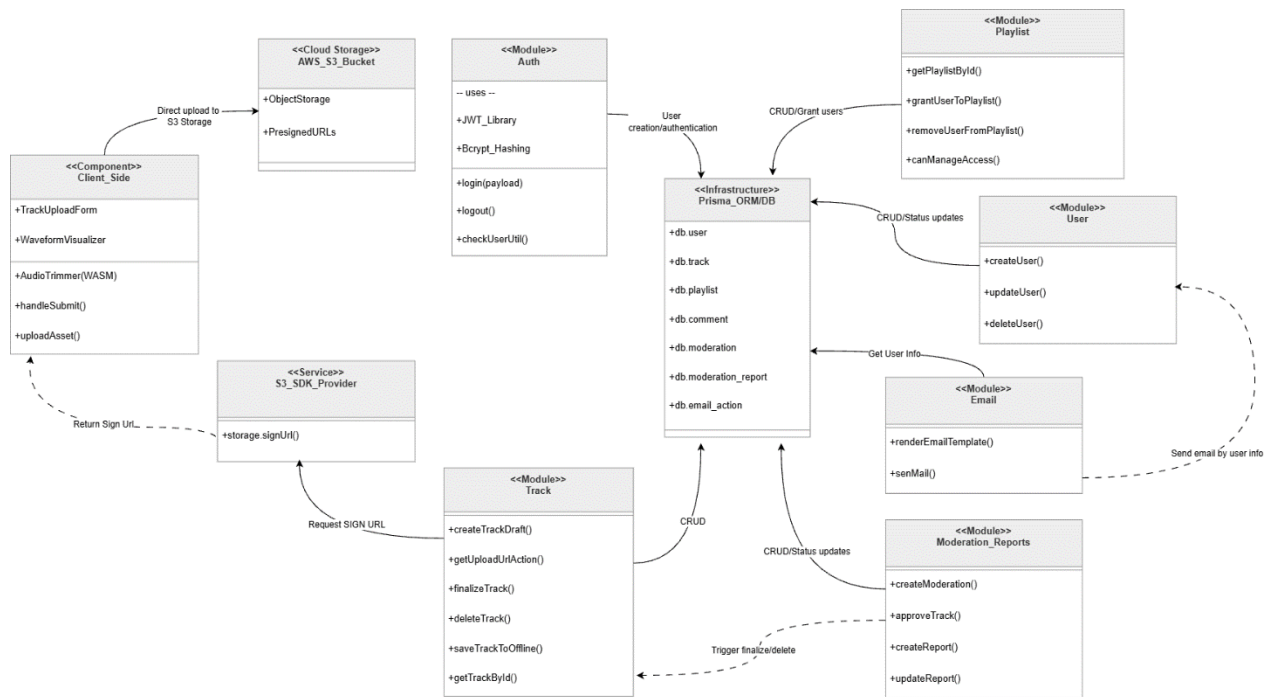


Рис. 2.2. Діаграма логічної структури проекту

2.2. Розробка бази даних системи

Для забезпечення надійного зберігання структурованої інформації, управління метаданими та підтримки реляційних зв'язків у вебзастосунк було обрано систему управління базами даних PostgreSQL, розгорнуту на хмарній платформі Supabase. Вибір PostgreSQL обумовлений її високою відмовтійкістю, підтримкою складних типів даних та механізмів індексації. Використання Supabase як хмарного провайдера надає можливість інтеграції розширень, таких як pg_stat, що дозволяє автоматизувати періодичні завдання (наприклад, очищення застарілих чернеток, або очищення даних які було видалено, адже системою мається на увазі що дані не будуть видалятися ще певний час після підтвердження видалення безпосередньо на стороні клієнта) безпосередньо в екосистемі БД, не навантажуючи сервер винесенням даної логіки на бекенд частину.

Для взаємодії з базою даних застосовано Prisma ORM. Використання даного інструменту дозволяє описувати структуру даних через декларативну схему, забезпечуючи автоматичну генерацію міграцій та наскрізну типізацію

на всіх рівнях розробки.

Нижче описано структуру усіх використаних сутностей схеми бази даних.

Таблиця 2.8

Опис таблиці «User»

Поле	Тип	Призначення
id	Uuid (PK)	Унікальний ідентифікатор користувача
role	UserRole (Enum)	Роль в системі (admin, moderator, user, unverified)
email	VarChar(255)	Електронна пошта (унікальна)
password	VarChar(255)	Хешований пароль користувача
username	VarChar(50)	Унікальний нікнейм користувача
displayName	VarChar(100)	Ім'я, що відображається в профілі
bio	Text	Короткий опис профілю
avatarUrl	VarChar(500)	Посилання на зображення аватара
createdAt	Timestamp	Дата та час створення акаунту
updatedAt	Timestamp	Дата та час оновлення акаунту
deletedAt	Timestamp	Дата та час поверхневого видалення користувача

Таблиця 2.9

Опис таблиці «Track»

Поле	Тип	Призначення
id	Uuid (PK)	Унікальний ідентифікатор треку
title	VarChar(120)	Назва композиції
captions	Text	Опис або текст треку
listenCount	Int	Кількість прослуховувань
authorName	VarChar(255)	Сценічне ім'я автора
uploaderId	Uuid (FK)	Ідентифікатор користувача, що завантажив трек, посилання на таблицю User
trackPicUrl	VarChar(500)	Посилання на обкладинку у сховищі S3
trackUrl	VarChar(500)	Посилання на аудіофайл у сховищі S3
isApproved	Boolean	Чи було прийнято модераторами
coverExtension	ImageExtension (Enum)	Розширення обкладинки

trackExtension	AudioExtension (Enum)	Розширення аудіофайлу
coverUrl	VarChar(500)	Посилання на обкладинку треку
uploadedAt	Timestamp	Дата та час завантаження треку
updatedAt	Timestamp	Дата та час оновлення акаунту
deletedAt	Timestamp	Дата та час поверхневого видалення треку

Таблиця 2.10

Опис таблиці «Playlist»

Поле	Тип	Призначення
id	Uuid (PK)	Унікальний ідентифікатор плейлиста
author_id	Uuid (FK)	Посилання на власника або автора плейлиста
title	VarChar(120)	Назва плейлиста
description	Text	Опис вмісту добірки
playlist_pic_url	VarChar(500)	Посилання на обкладинку плейлиста
is_public	Boolean	Видимість (публічний/приватний)
created_at	Timestamp	Дата створення
updated_at	Timestamp	Дата оновлення метаданих
deleted_at	Timestamp	Дата поверхневого видалення плейлисту

Таблиця 2.11

Опис таблиці «Playlist_Track»

Поле	Тип	Призначення
id	Uuid (PK)	Унікальний ідентифікатор запису зв'язку
playlist_id	Uuid (FK)	Посилання на таблицю плейлистів
track_id	Uuid (FK)	Посилання на таблицю треків
order	Int	Позиція треку в списку відтворення
created_at	Timestamp	Час додавання треку до плейлиста

Таблиця 2.12

Опис таблиці «User_Playlist»

Поле	Тип	Призначення
id	Uuid (PK)	Ідентифікатор запису прав доступу
playlist_id	Uuid (FK)	Посилання на конкретний плейлист
user_id	Uuid (FK)	Посилання на користувача
access	PlaylistAccess (Enum)	Рівень прав (view, edit, owner тощо) Визначено перерахуванням
created_at	Timestamp	Дата надання доступу
updated_at	Timestamp	Дата зміни рівня прав

Таблиця 2.13

Опис таблиці «Email_Actions»

Поле	Тип	Призначення
id	Uuid (PK)	Унікальний токен дії
type	ActionType	Тип операції (верифікація або скидання). Визначено перерахуванням
data	Json	Корисне навантаження (payload) у форматі JSON
created_at	Timestamp	Час ініціалізації дії
updated_at	Timestamp	Останнє оновлення стану дії

Таблиця 2.14

Опис таблиці «Comment»

Поле	Тип	Призначення
id	Uuid (PK)	Унікальний ідентифікатор коментаря
user_id	Uuid (FK)	Посилання на автора (користувач) текстового повідомлення
track_id	Uuid (FK)	Посилання на трек до якого залишено коментар
content	Text	Текст коментаря
moderation_status	ModerationStatus	Стан перевірки тексту коментаря
time_stamp	Int	Часова мітка (в секундах) у аудіофайлі

продовж. табл. 2.14

created_at	Timestamp	Дата публікації
updated_at	Timestamp	Дата редагування
deleted_at	Timestamp	Дата поверхневого видалення коментаря

Таблиця 2.15

Опис таблиці «Moderation»

Поле	Тип	Призначення
id	Uuid (PK)	Ідентифікатор процесу перевірки
track_id	Uuid (FK)	Посилання на запис треку, що проходить модерацію
moderator_id	Uuid (FK)	Посилання на призначеного системного модератора
status	ModerationStatus	Результат перевірки (pending, accepted, declined)
conclusion	Text	Текстове пояснення рішення
created_at	Timestamp	Час створення запиту на модерацію
updated_at	Timestamp	Час винесення вердикту

Таблиця 2.16

Опис таблиці «Moderation_Report»

Поле	Тип	Призначення
id	Uuid (PK)	Унікальний ідентифікатор скарги
track_id	Uuid (FK)	Посилання на оскаржуваний трек
comment_id	Uuid (FK)	Посилання на оскаржуваний коментар
user_id	Uuid (FK)	Посилання на користувача, який подав скаргу
moderator_id	Uuid (FK)	Посилання на модератора (користувача), що обробляє скаргу
reason	Text	Опис причини скарги
conclusion	Text	Результат розгляду скарги
status	ReportStatus	Поточний стан (pending, accepted, declined)

2.3. Проектування та реалізація алгоритмів роботи системи

Для забезпечення високого рівня надійності системи, детальний аналіз архітектури основних процесів і бізнес-логіки системи, виконано за допомогою моделювання через інструментарій UML-діаграм.

Зокрема, розроблений алгоритм завантаження медіаконтенту та взаємодії з хмарною інфраструктурою описує процес додавання нового аудіотреку як комплексну розподілену операцію. У цьому процесі синхронно задіяні три сторони: клієнтський браузер користувача, сервер застосунку в ролі якого виступають серверні дії, та віддалене об'єктне сховище AWS S3.

Головною інженерною метою реалізації даного алгоритму є мінімізація транзитного мережевого трафіку через основний сервер додатка. Побудована діаграма послідовності завантаження треку (рис. 2.4) детально відображає повний шлях проходження інформації про трек до сховища.

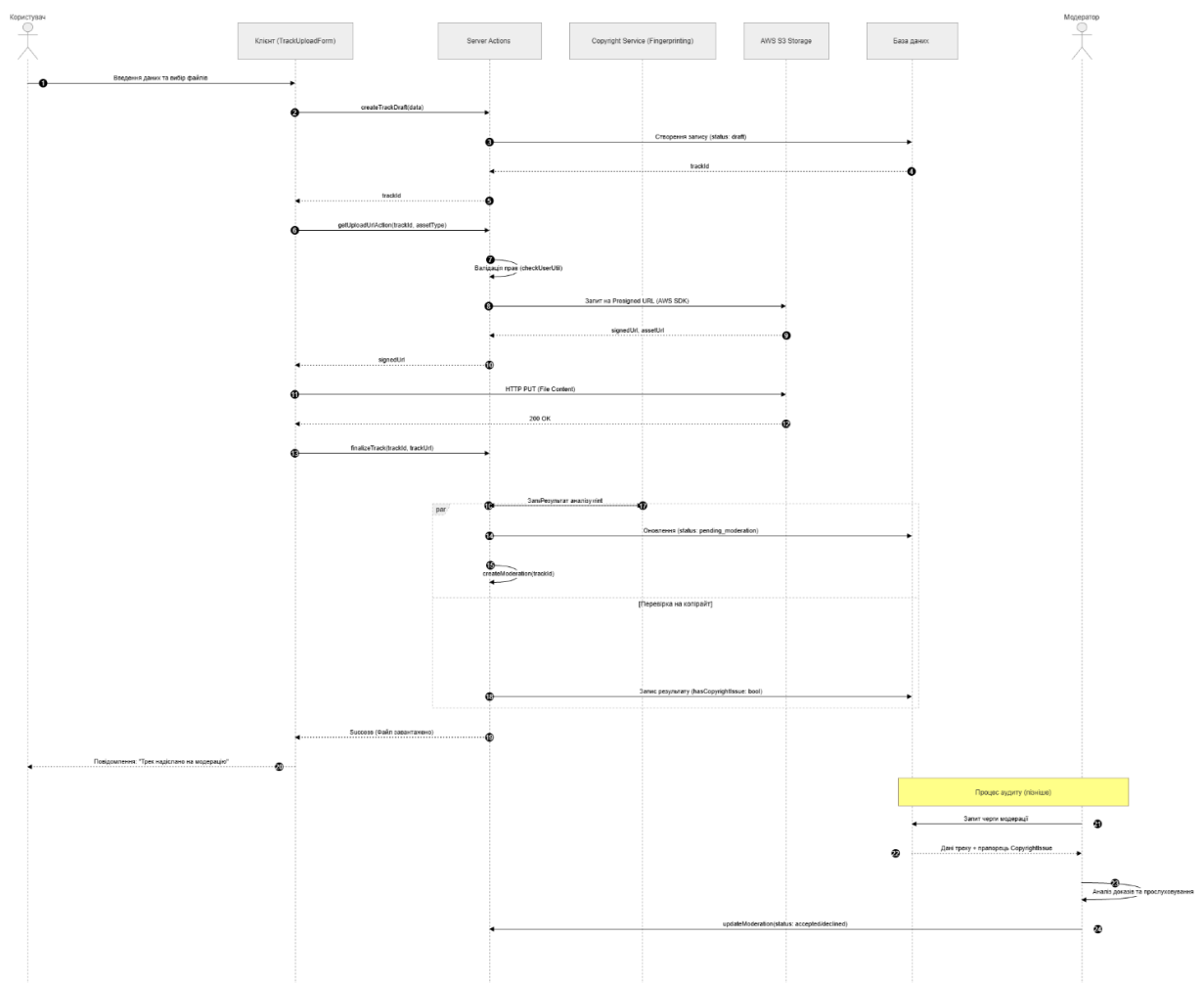


Рис. 2.4. Діаграма послідовності завантаження аудіо на платформу

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

Після успішного завантаження файлу в S3 та виклику методу `finalizeTrack`, система паралельно створює запис у таблиці `Moderation`. Цей запис пов'язує сутність треку із сесією перевірки, зберігаючи початковий статус `pending`. До моменту винесення вердикту трек залишається прихованим від глобального пошуку. Важливою особливістю пайплайну є інтеграція з сервісом ідентифікації контенту. Одночасно зі створенням сесії модерації, система виконує запит до сервісу цифрових відбитків (`fingerprinting`). У разі виявлення збігів з існуючою базою авторських прав, у системі встановлюється службовий флаг `hasCopyrightIssue`.

Авторизований персонал (Модератор/Адміністратор) в свою чергу через спеціалізований інтерфейс керування отримує доступ до черги запитів. Алгоритм дозволяє здійснити повний аудит об'єкта: перегляд метаданих, прослуховування аудіопотоку, а також врахування автоматично згенерованого звіту про авторські права. Наявність відмітки про потенційне порушення прав власності не блокує трек автоматично, проте слугує критичним сигналом для модератора.

За результатом аудиту модератор ініціює метод `updateModeration`, вносячи текстовий висновок та встановлюючи статус `accepted` або `declined`. При винесенні вердикту система автоматично змінює статус треку в основній таблиці та ініціює відправку Email-повідомлення автору. У разі схвалення об'єкт стає доступним для прослуховування всіма категоріями користувачів. У випадку відхилення трек отримує мітку видалення (`deletedAt`).

В свою чергу для мінімізації витрат на хмарну інфраструктуру, система інтегрована з `S3 Lifecycle Rules` [17]. Файли, що отримали статус відхилених або видалених, автоматично підлягають фізичному стиранню зі сховища після 14-денного періоду за умови відсутності запитів до них. Це забезпечує автоматизовану очистку без додаткових серверних ресурсів.

Переходячи до наступного модулю варто також виділити окремо алгоритм валідації прав доступу, який спрацьовує при кожному зверненні до приватних ресурсів (наприклад, редагування плейлиста). Він базується на

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

ієрархічній перевірці: спочатку перевіряється відповідність сесії (токен в cookies), а потім - специфічні права в таблиці зв'язків.

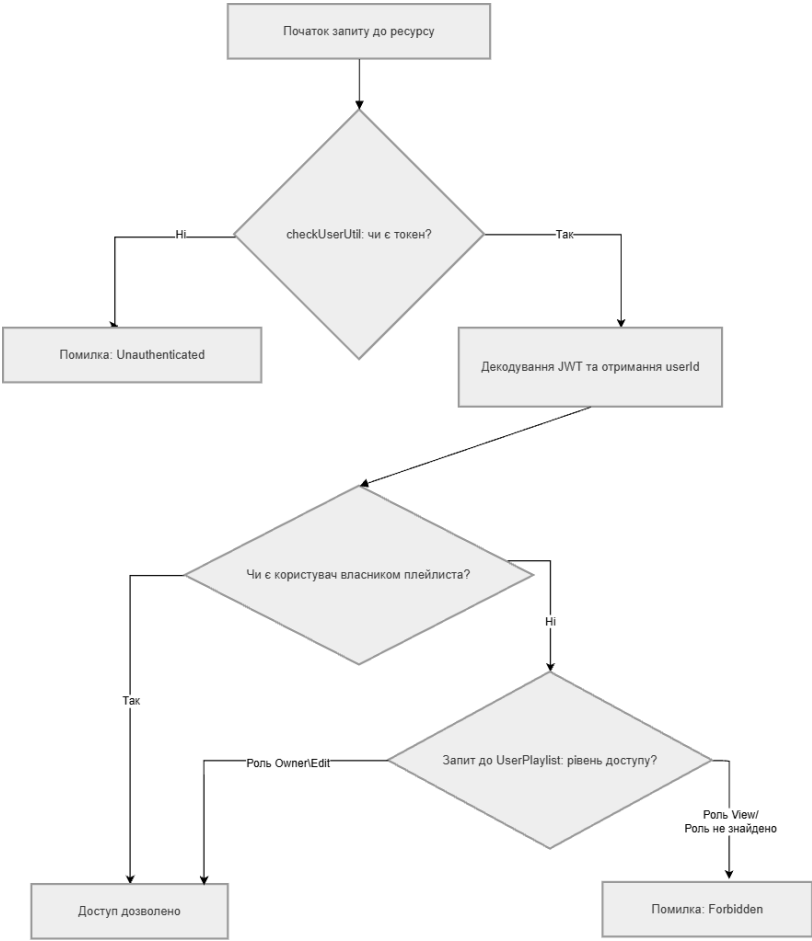


Рис. 2.5. Діаграма діяльності перевірки доступу до ресурсу редагування доступу до плейлисту

Алгоритм відображення та взаємодії з об'єктами типу «Плейлист» базується на динамічній перевірці прав доступу, що зберігаються в сутності UserPlaylist. Початкова фільтрація контенту відбувається на рівні глобального сховища: у загальному переліку користувачу доступні лише власні добірки та об'єкти з прапором isPublic. Прямий доступ до приватних плейлистів інших користувачів заблокований на рівні маршрутизації до моменту отримання відповідної ролі в ACL.

Логіка розширення функціональних можливостей при зміні рівня доступу в межах платформи реалізована за чітким ієрархічним принципом. Базовим ступенем у цій структурі є рівень слухача, призначення якого

дозволяє користувачу ініціювати запит до сторінки приватного плейлиста. На цьому етапі алгоритм авторизації забезпечує доступ виключно до методів читання списку композицій та їх безпосереднього відтворення, проте жорстко блокує будь-які спроби модифікації метаданих або внутрішньої структури добірки.

Наступним етапом розширення повноважень є рівень редактора, отримання прав якого активує доступ до інтерфейсу безпосереднього керування контентом. Користувач отримує санкційований дозвіл на виклик модальних вікон модифікації даних, що включають зміну заголовка, текстового опису та графічної обкладинки медіатеки. Крім того, на рівні редактора стає доступною функція зміни статусу публічності плейлиста, а також надає можливість додавати існуючі в системі музичні композиції до поточної добірки.

Найвищим ступенем у наведеній ієрархії виступає рівень власника, який надає користувачу повний адміністративний контроль над створеною сутністю. Власнику відкривається доступ до спеціалізованого модуля управління правами, де реалізовано механізм динамічного делегування ролей іншим учасникам системи. Крім того, володіння цією роллю надає виключне право на ініціалізацію процедури безповоротного видалення плейлиста та пов'язаних медіаданих із бази даних системи.

Процес модифікації прав доступу тісно інтегрований із підсистемою сповіщень. Кожна операція призначення нової ролі або зміни поточного рівня повноважень ініціює алгоритм надсилання електронного листа на верифіковану адресу користувача. Система в автоматичному режимі генерує інформаційні повідомлення на основі попередньо скомпільованих шаблонів, які містять вичерпні дані про тип наданого доступу та пряме посилання на відповідний медіаресурс. Це забезпечує високий рівень прозорості взаємодії між учасниками платформи та підтверджує легітимність отриманих повноважень. При цьому валідація кожного з етапів виконання описаного алгоритму дублюється на стороні сервера через обов'язкову перевірку сесії та

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

контексту користувача, що повністю виключає можливість маніпуляції клієнтським інтерфейсом для обходу встановлених безпекових обмежень.

2.4. Реалізація функціональних модулів та компонентної структури вебзастосунку

Фактично всі модулі, та архітектура всього проекту базується на надаваному Next.js каркасі, який в свою чергу надає можливість зручно розділяти серверний та клієнтські рівні. А програмний комплекс спроектовано за модульними принципами де кожний окремих модуль містить в собі набір як клієнтський так і серверних компонентів разом з серверними екшенами які відповідають за роботу з безпосередньо базою даних та інформацією яка в ній зберігається.

Для забезпечення роботи офлайн та фоновому режиму вебзастосунку інтегровано бібліотеку NextPWA, конфігурація якої описується безпосередньо в файлі «next-config.js». Що в свою чергу забезпечує автоматичну реєстрацію фонових сервіс-воркерів, без необхідності описувати купу файлів вручну, прямо в браузері клієнта, паралельно очищуючи застарілі закешовані файли. Той же самий ресурс і займається кешуванням категорично необхідних сторінок як от «Збережені» за для їх валідної роботи без прямого підключення до мережі. Там же й описано логіку відображення відповідної сторінки заглушки, на яку користувача буде перенаправлено у випадку якщо він спробує перейти на певний маршрут який не було передбачено в стратегії кешування без підключення до інтернету.

Для вирішення завдання з програванням аудіо в офлайн режимі було задіяно бібліотеку Dexie обгортку над IndexedDB, яка в свою чергу являється апі для роботи з локальним сховищем, в тому числі для об'єктів як от великі відео та аудіо файли. Опис схеми локальної бази даних saved_tracks реалізовано через декларативне створення сховища репозиторіїв tracks, де первинним ключем виступає унікальний ідентифікатор треку з основної реляційної бази даних. Процес збереження треку до локального сховища

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

реалізовано у відповідному серверному екшені `saveTrackToOffline`, в якому відбувається паралельне стягування аудіо файлу відповідного треку та його збереження з усім переліком необхідних полів як от назваа або довживно аудіодоріжки. Система початково налаштована таким чином що вона шукає трек офлайн, і тільки якщо його не знайдено йде запит на серверну частину та отримуються дані звідти.

Для збереження цих самих медіафайлів використовується об'єктне хмарне сховище `AWS S3`. А взаємодія з ним відбувається через надаваний `AWS S3 SDK` функціонал. Для ізоляції логіки розроблено сервісний клас `Storage`, який інкапсулює ініціалізацію клієнта `S3Client`, та роздає його екземпляр якщо така логіка потрібна, без необхідності щоразу перестворювати новий об'єкт підключення. Основним методом класу є `signUrl`, який реалізує концепцію безпечного завантаження файлів з клієнта. Де для реалізації безпечного та минуючого навантаження для сервера завантаження використовується метод `getSignedUrl` надаваний `SDK`, яке в свою чергу має обмежений час життя в 300 секунд та при запиті буде очікувати саме той файл який було початково вказано, що виключає підміну контенту. Генерація безпечних адрес для завантаження асетів контролюється захищеним серверним екшеном `getUploadUrlAction`. Перед викликом хмарного `SDK` модуль виконує комплекс перевірок безпеки, як от перевірку на авторство та відповідність типів того чи іншого ресурсу, і тільки у випадку успішного проходження цих перевірок на клієнт повертається пара значень з шляхом за яким буде збережено файл та посилання для його завантаження.

Також важливим компонентом системи являється ядро візуалізації аудіо в реальному часі. В рамках проєкту це реалізован через апі надаване браузером яке зветься `Web Audio API`, інструментарій якого дозволяє перехоплювати аудіопотік який в даний момент декодується браузером та розкласти спектр аудіо на відповідні хвилі коливання та частоти.

Зосередок обробки звуку знаходиться в глобальному контексті звідки паралельно з цим відбувається управління звуком, його гучністю, прогресом

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

програвання та павзою разом чергою відтворення, яка реалізована через окремих хук. Побудова конвеєра обробки аудіосигналу виконується методом `initAudioAnalyzer` (див. Додаток В). Алгоритм створює екземпляр базового керуючого класу `AudioContext` та вузол математичного аналізу фонограм `AnalyzerNode`. Тут же оптимізації навантаження на пристрій користувача розмір FFT зафіксовано на рівні 256 точок що в майбутньому перетворюється на 128 значень частот на відповідному часовому відрізку. Взагалі що таке розмір FFT [21] або ж алгоритму швидкого перетворення Фур'є. Фактично це ніби контейнер з визначеними розмірами, який в певний часовий момент обирає маленький відрізок звукової хвилі, розбиває його на 256 шматків в нашому випадку і відносно нуля надає цим шматкам якусь точку, а саме значення, від нуля до 255 що і відповідає частотам аудіо в обраний момент часу.

Для передачі масиву частот у компоненти графічного інтерфейсу розроблено кастомний хук `useVisualizer` (див. Додаток Б). Логіка його роботи полягає в організації безперервного циклу опитування буфера аналізатора із частотою оновлення екрана пристрою в 60 кадрів. Це реалізовано за допомогою системного API браузера `requestAnimationFrame`. При кожному такті анімації хук створює масив байтів `Uint8Array`, розмір якого строго відповідає поточній властивості аналізатора в 128 точок, після чого і виконується заповнення масиву числовими значеннями від 0 до 255 який перезаписується та проковує перерендер компонентів які залежать від стану цього масиву.

Безпосередній рендеринг графічних стовпчиків тої чи іншої реалізації виконуватиметься всередині спеціалізованих клієнтських компонентів (по прикладу звичаної стовпчастої спектограми). Оскільки сирий масив із 128 частот часто є занадто щільним для мобільних екранів і відображає шкалу лінійно (де більша частина частот припадає на високі значення, які людина майже не чує), впроваджено шар математичної нормалізації даних на основі переданого в компонент значення, яке визначає к-сть стовпців в спектограмі.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

На основі цього значення і групуються частоти шляхом розрахунку середньої ціни поділки на кожен стовпець, після чого відповідно отриманим значенням частот і виконується рендер стовпчиків та їх висота.

Окремо варто виділити роботу сервісу для аналізу завантажуваних треків на предмет копірайту. В основі роботи якого лежить хроматичний алгоритм [29], який аналізує отримане аудіо шляхом групування чанків отриманих шляхом математичного перетворення Фур'є за 12 основними напівтонами музичної октави. Оскільки алгоритм ідентифікує саме домінантні гармоніки та ноти в кожен момент часу, система оцінює звуковий потік у контексті його унікальності в плані мелодійності та інструментальної структури, в тому числі вокалу. Це дозволяє аналізувати безпосередньо сутність композиції, а не абстрактні числові масиви чи сухі технічні параметри оцифрованого сигналу. Після аналізу генерується захешований рядок який і являється акустичним відбитком обраного треку, у випадку якщо при запиті на сторонню базу даних знайдено певні збіги, результат подається у вигляді масиву об'єктів де зазначено автора, назву треку та рівень схожості за шкалою від 0 до 1. За для найбільш пасуючого граничного значення було враховано результати випробувань автора алгоритму Chromaprint. У ході експериментів автора [29] алгоритму та відповідно утиліти fpcalc проводилося порівняння оригінальних студійних записів із їхніми копіями, що піддалися екстремальному стисненню до формату MP3 із бітрейтом 32 kbps або ж іншими словами аудіо неможливо поганої якості. Було виявлено, що навіть за умови критичної деградації звуку, яка спричиняє відчутті для людини акустичні шуми та спотворення, хроматичний алгоритм зберігає стійкість і фіксує збіг відбитка на рівні 80%–82%. На основі цього розробник технології дійшов висновку, що падіння коефіцієнта подібності нижче позначки 0.8 свідчить про зміну безпосередньо структури самого аудіосигналу (застосування інших інструментів, зміна темпу чи додавання унікальних перебивок), що вказує на зовсім іншу музичну композицію. Саме тому для автоматичного фільтра копірайту було обрано поріг відповідності в 0.8, який і

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

гарантує точне визначення дублікатів, без хибних висновків через те що трек було піддано стисканню або мінімальним змінам.

Висновки до другого розділу

У другому розділі кваліфікаційної роботи проведено комплексне проектування, декомпозицію та програмну реалізацію архітектурних модулів, бази даних та алгоритмічної основи вебзастосунку

Сформовано чітку структуру системи на основі прецедентів використання для чотирьох ключових ролей. Спроектовано реляційну структуру бази даних на основі декларативних схем Prisma ORM та СКБД PostgreSQL. Структура таблиць та зв'язків між ними оптимізована для швидкої обробки медіаконтенту, ієрархічного збереження метаданих треків, безпечного адміністрування списків доступу та логування користувацьких скарг.

На основі розроблених алгоритмів інтегровано логіку взаємодії з хмарним об'єктним сховищем AWS S3 за допомогою короткочасних препідписаних посилань, що дозволило повністю оптимізувати транзитний трафік через сервер та забезпечити надійне збереження медіафайлів

Визначено ключові та критично необхідні в пильному аналізі компоненти як от інтерактивний модуль цифрової обробки та візуалізації частот звукових хвиль у реальному часі на базі Web Audio API із застосуванням швидкого перетворення Фур'є та офлайн роботи всієї ситеми.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБЗАСТОСУНКОМ

3.1. Порядок встановлення та налаштування системи

Головний модуль системи, що охоплює інтерфейсну частину та логіку ядра обробки даних, було розгорнуто на спеціалізованій хмарній PaaS-платформі Vercel. Вибір цього середовища зумовлений його нативною інтеграцією з фреймворком Next.js та автоматичною підтримкою обраної архітектури серверних компонентів. Увесь процес розгортання та конфігурації проєкту було поділено на три послідовні етапи: автоматизацію розгортання засобами CI/CD шляхом інтеграції віддаленого репозиторію, налаштування етапів збирання готового застосунку разом із генерацією статичних сторінок, а також конфігурацію змінних середовища з усіма необхідними для роботи системи ключами, назвами сховищ та захищеними токенами доступу. Клієнтський пристрій користувача взаємодіє з серверною частиною за допомогою серверних дій (Server Actions).

Для збереження медіафайлів, зокрема аудіофайлів композицій, графічних обкладинок альбомів та аватарів користувачів, було залучено хмарну інфраструктуру об'єктного сховища AWS S3. Конфігурація цього сховища винесена в окремий незалежний модуль, що дозволило реалізувати патерн прямого завантаження даних із клієнтського браузера за допомогою тимчасових підписаних посилань. Таке гібридне інфраструктурне рішення, де логіка застосунку виконується на потужностях Vercel, а важкі статичні файли розподіляються через оптимізовані сховища AWS, дозволило забезпечити максимальну пропускну здатність платформи, мінімізувати мережеві затримки при стрімінгу аудіопотоку та суттєво знизити загальні витрати на обчислювальні ресурси основного сервера, повністю звільнивши його від необхідності займатись завантаження великих файлів на своїй стороні.

Найбільш складною з точки зору обчислювальної потужності частиною

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

системи є сервіс перевірки копірайту. Оскільки інструменти декодування звуку та обчислення акустичних відбитків потребують значних ресурсів процесора та довготривалого утримання з'єднань, цей модуль повністю ізольовано від архітектури ядра та розгорнуто на платформі Railway [27]. Процес розгортання даного сервісу автоматизовано за допомогою контейнеризації, логіка самого сервісу, всі необхідні залежності в тому числі медіа-процесор FFmpeg та системні бібліотеки упаковуються у сервісний Docker-образ, який після цього стягується з репозиторію сервісом Railway читає конфігураційний докерфайл з описаними інструкціями для його розгортання та виділяє для нього необхідну к-сть ресурсів процесора. Отримавши задачу на перевірку, сервіс зчитує файл з AWS S3, за допомогою FFmpeg генерує унікальний акустичний та відправляє його на верифікацію у сторонню базу глобальних відбитків (див. Додаток А), після чого отриманий результат отримується на сервері вебзастосунку та залежно від отриманої відповіді визначає трек як підпадаючий під права власності або ні.

Зберігання довготривалих даних, що охоплюють облікові записи користувачів, метадані плейлистів, черги модерації, текстові коментарі та матрицю прав доступу ACL, покладено на реляційну базу даних PostgreSQL, яка розгорнута на базі хмарної BaaS-платформи Supabase.

Вибір цього інфраструктурного рішення зумовлений високою надійністю збереження транзакційних даних, підтримкою вбудованого пулу з'єднань для оптимізації одночасних запитів, а також можливістю гнучкого масштабування сховища при зростанні обсягів інформації. Архітектурне поєднання зазначених середовищ формує цілісну високопродуктивну інфраструктуру, де кожен компонент виконує чітко ізольовану інженерну задачу.

Для комплексного розуміння топології розгортання та мережевої взаємодії всіх інтегрованих модулів було розроблено загальну високорівневу схему інфраструктури (рис. 3.1). Наведена схема детально демонструє архітектурні межі та протоколи зв'язку, що забезпечують працездатність

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

системи під час виконання користувацьких та адміністративних сценаріїв. На схемі чітко візуалізовано розподіл інформаційних потоків: ядро системи підтримує постійне TCP-з'єднання з реляційним сховищем PostgreSQL для оперативного обміну даними та ініціює асинхронні HTTPS-виклики до ізольованого контейнера на Railway. Прямий канал взаємодії між Next.js PWA Client та об'єктним сховищем AWS S3 відображає архітектурну ізоляцію важкого контенту, що гарантує низький час відгуку інтерфейсу та високу відмовостійкість системи в умовах реальної експлуатації..

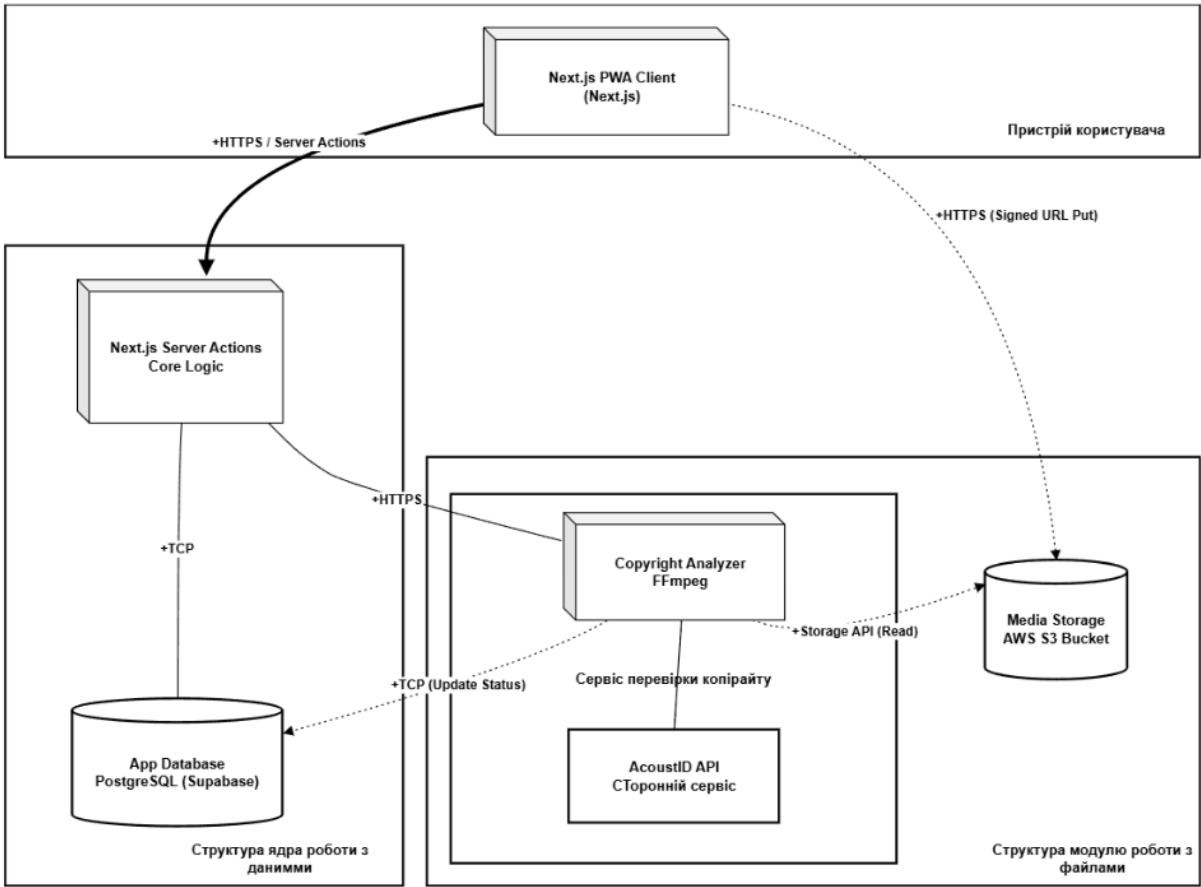


Рис. 3.1. Візуалізація високорівневої структури проєкту

Додатковою перевагою обраного порядку налаштування параметрів інфраструктури є максимальне використання концепції Serverless та Edge-обчислень, наданих платформою Vercel. Замість утримання класичного виділеного сервера VPS, який потребує постійного ручного адміністрування, оновлення пакетів операційної системи та конфігурації вебсерверів типу Nginx, розроблений комплекс повністю спирається на керовані хмарні сервіси.

При цьому статичні елементи інтерфейсу та оптимізовані графічні ресурси миттєво реплікуються на глобальні сервери доставки контенту. Це наближує контент до кінцевого користувача, зводячи TTFB до мінімуму, що в умовах сучасного вебзастосунку є без критично важливим фактором для забезпечення високої якості користувацького досвіду.

3.2. Структура інтерфейсу вебзастосунку.

Додаток початково проектувався в парадигмі нашарування функціоналу, саме тому певні функції не будуть відображатись до тих пір, поки слухач не отримає відповідну роль, або ж не зареєструється в подальшому підтвердивши свою електронну пошту вказану в формі раніше.

При першому вході у веб-застосунок, незареєстрований користувач потрапляє на головну сторінку (рис. 3.2) на якій одразу має можливість обрати трек та прослухати його.

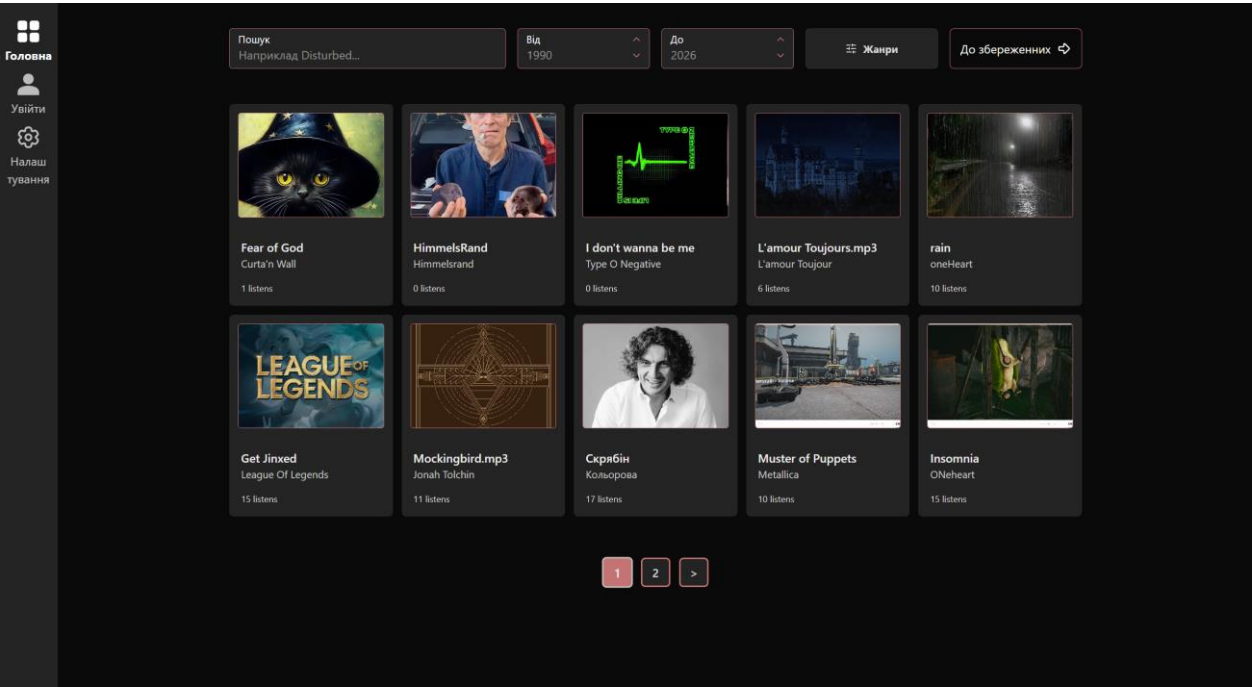


Рис. 3.2. Головна сторінка вебзастосунку

У верхній частині екрану розміщений центр фільтрації (рис. 3.3), де користувач може відсіяти композиції за роком випуску, по жанрам, або ж знайти конкретний трек за назвою або ім'ям автора, всі варіанти фільтрації

можна комбінувати між собою без ніяких конфліктів.

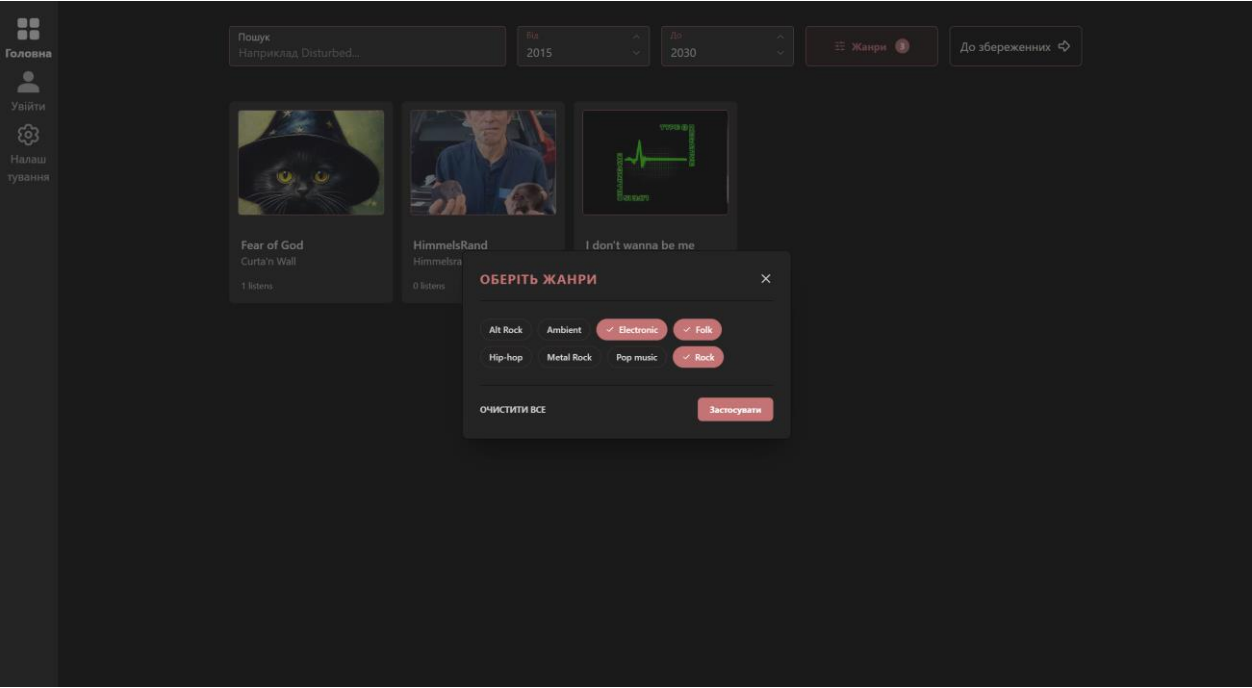


Рис. 3.3. Фільтрація за декількома пунктами

Тут же представлене бокове меню, опції якого змінюються залежно від ролі користувача, у випадку простого гостя це лише три варіанти на вибір, а саме Головна сторінка, сторінка Налаштувань, та можливість перейти до форми реєстрації або авторизації.

Під час програвання треку з'являється відповідний компонент плеєра в нижній частині екрану, звідки користувач має можливість керувати його гучністю або одразу вимкнути звук одним натисканням на відповідну іконку, керувати прогресом програвання та ставити на паузу. По обидва боки від кнопки паузи розміщені кнопки перемикання композицій, функціонування яких тісно переплетене з чергою програвання, де наведено можливість проглянути увесь список разом, перемикатись між в будь якому порядку, видаляти композиції з черги, та додавати їх через відповідну опцію в контекстному меню (рис. 3.4).

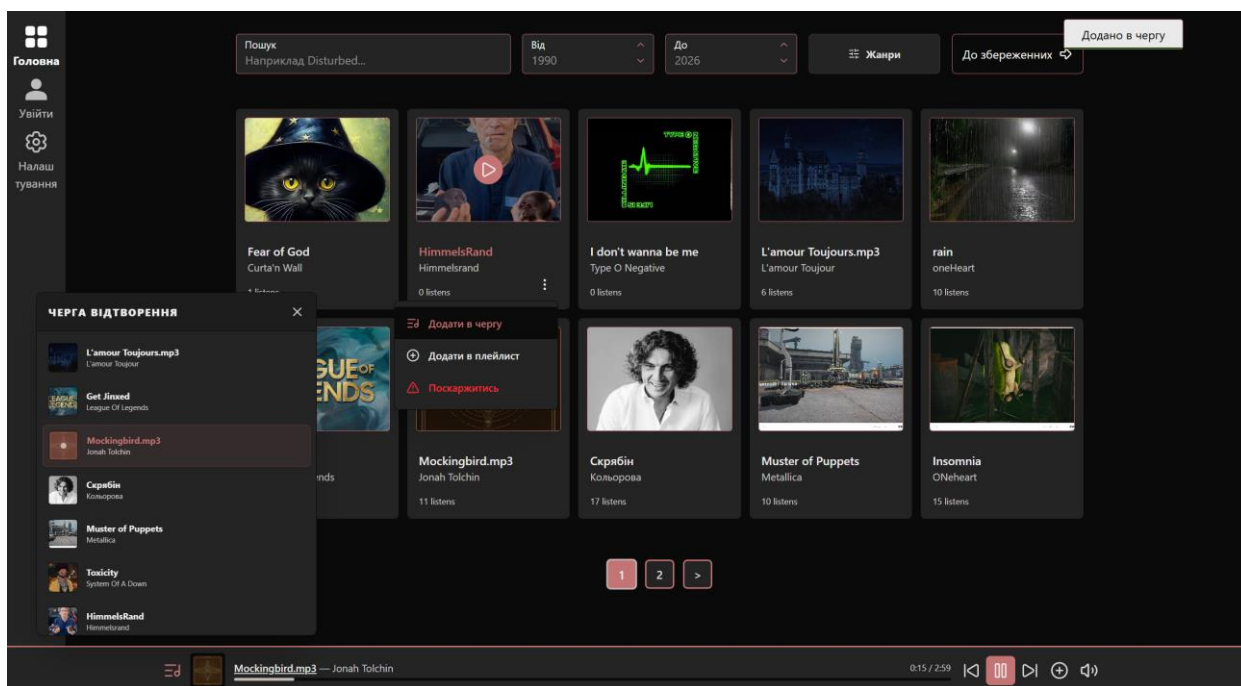


Рис. 3.4. управління чергою програвання

Сторінка входу в власний акаунт являє собою просту форму (рис 3.5) з можливістю увійти або зареєструватись через власний гугл акаунт, без необхідності вгадувати та запам'ятовувати пароль для входу.

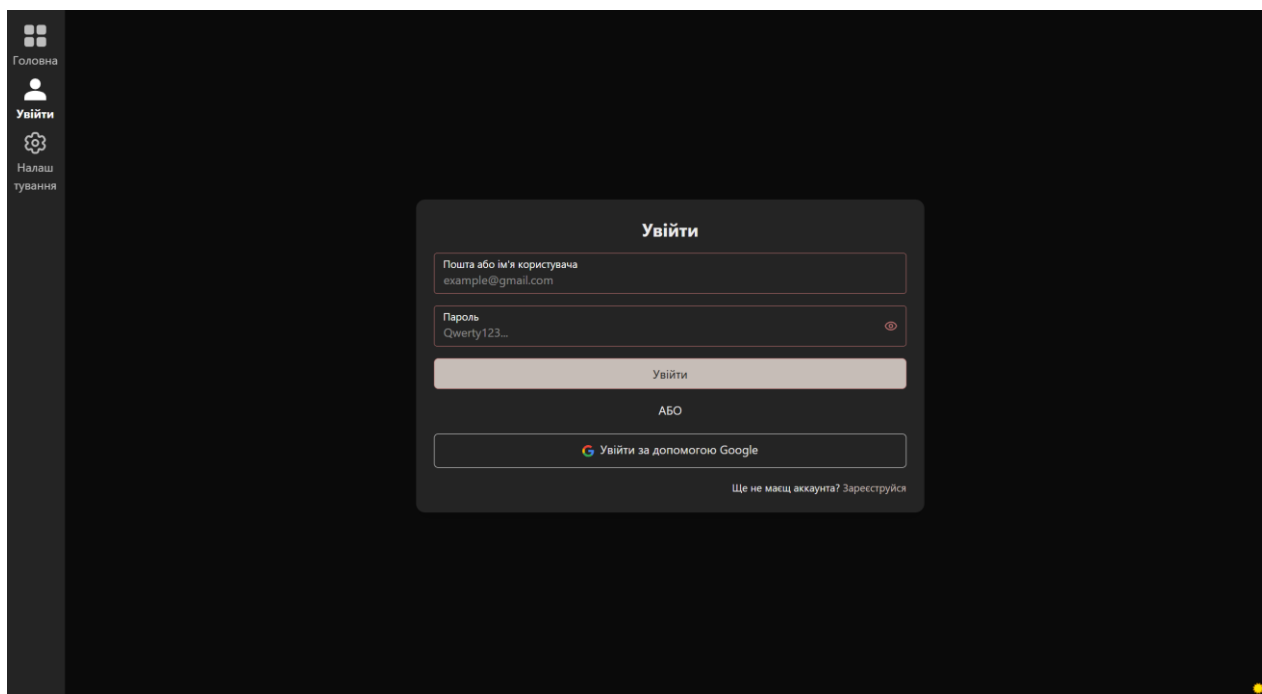


Рис. 3.5. сторінка входу у власний акаунт

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		59

Після успішного входу в акаунт, користувач знову потрапляє на головну сторінку, У випадку якщо ж акаунт було тільки но створено, доступ до вебзастосунку надається, однак обмежений, а іконка кнопки переходу в власний кабінет змінюється на відповідну. Для того щоб підтвердити акаунт користувачи на вказану електронну пошту надійде лист з відповідними інструкціями. Після успішного підтвердження пошти тепер перед користувачем відкривається безліч нових можливостей, від створення власних плейлистів та керуванням додаванням композицій в них, до завантаження власних треків, управління ними, та можливість залишати коментарі до вже існуючи. Однак варто розпочати зі сторінки кабінету (рис. 3.6), де представлено всю інформацію про користувача.

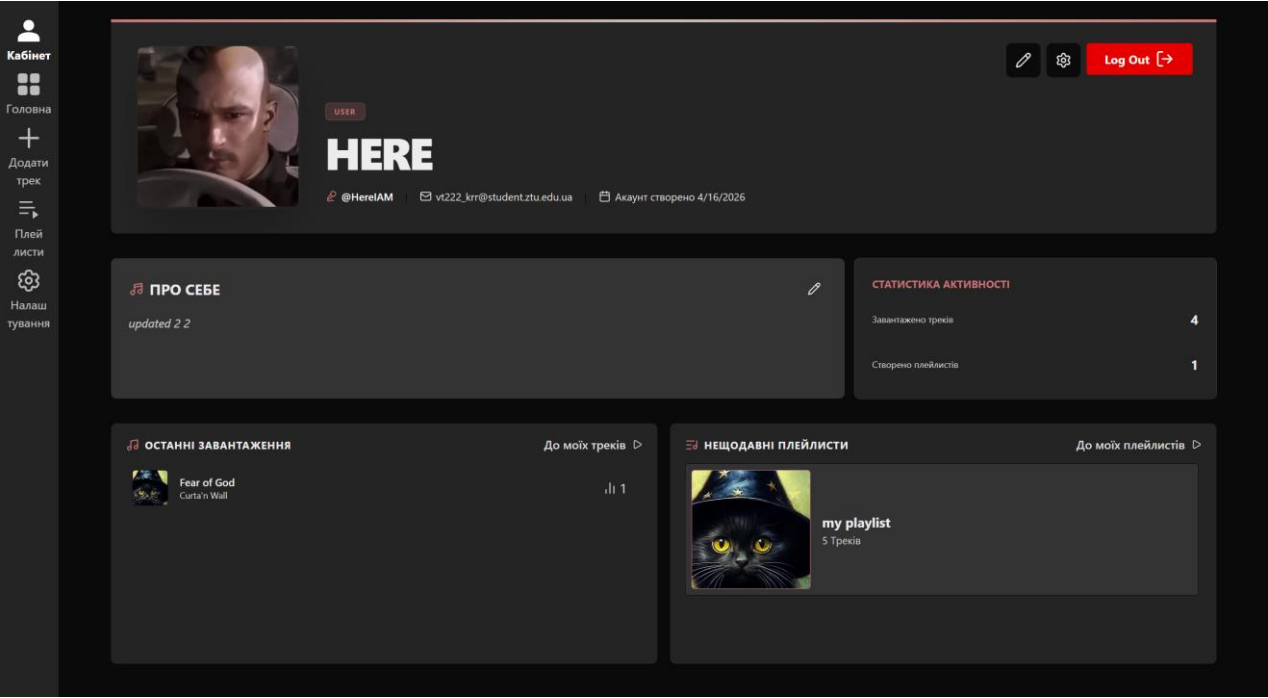


Рис. 3.6. Сторінка профілю користувача

Сторінка побудована з декількох блоків, де верхній два блоки містять в собі особисту інформацію про користувача, дату створення акаунта та опис який можна залишити по бажанню, разом з простою статистикою про те скільки композицій було завантажено даним користувачем, та скільки плейлистів було створено. Тут же представлено кнопка редагування особистої інформації, зміна паролю та фото профілю, якщо це необхідно (рис. 3.7), у

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

випадку якщо користувач все ж вирішить завантажити нову картинку профілю, або змінити вже існуючу, розмір файлу встановлений в налаштуваннях, не дасть йому завантажити занадтке фото, після чого відповідне повідомлення буде показане під компонентом для зміни цього самого фото.

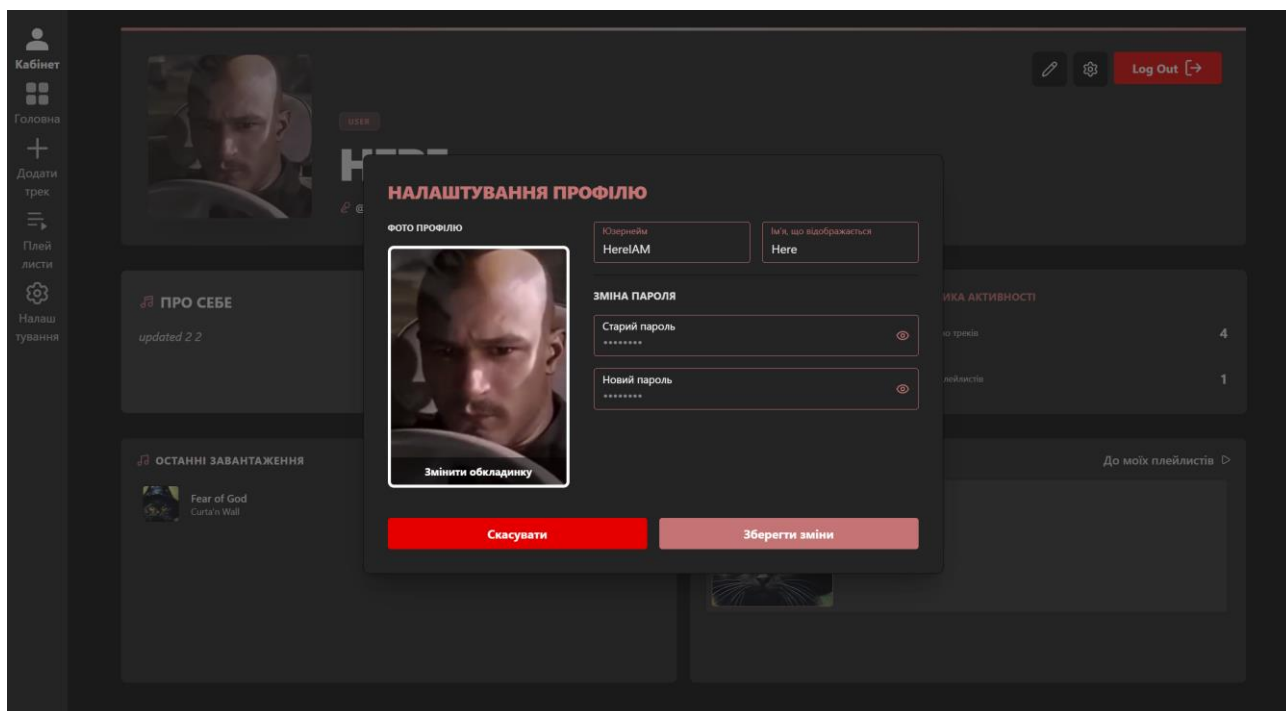


Рис. 3.7. форма редагування профілю

Наступний варіант використання системи полягає у можливості проглянути основну інформацію про трек (рис. 3.8), де представлені такі опції як поскаржитись на нього, після чого скарга потрапить у відповідну чергу та буде очікувати на вердикт відповідного персоналу та зберегти обраний трек офлайн, ці ж функції продубльовано у контекстному меню, у випадку незареєстрованого користувача, іншими словами гостя, дані опції не відображаються. Тут же різні вкладки як от перегляд слів до пісні якщо користувач додав їх при завантаженні, та можливість перейти до вкладки коментарів.

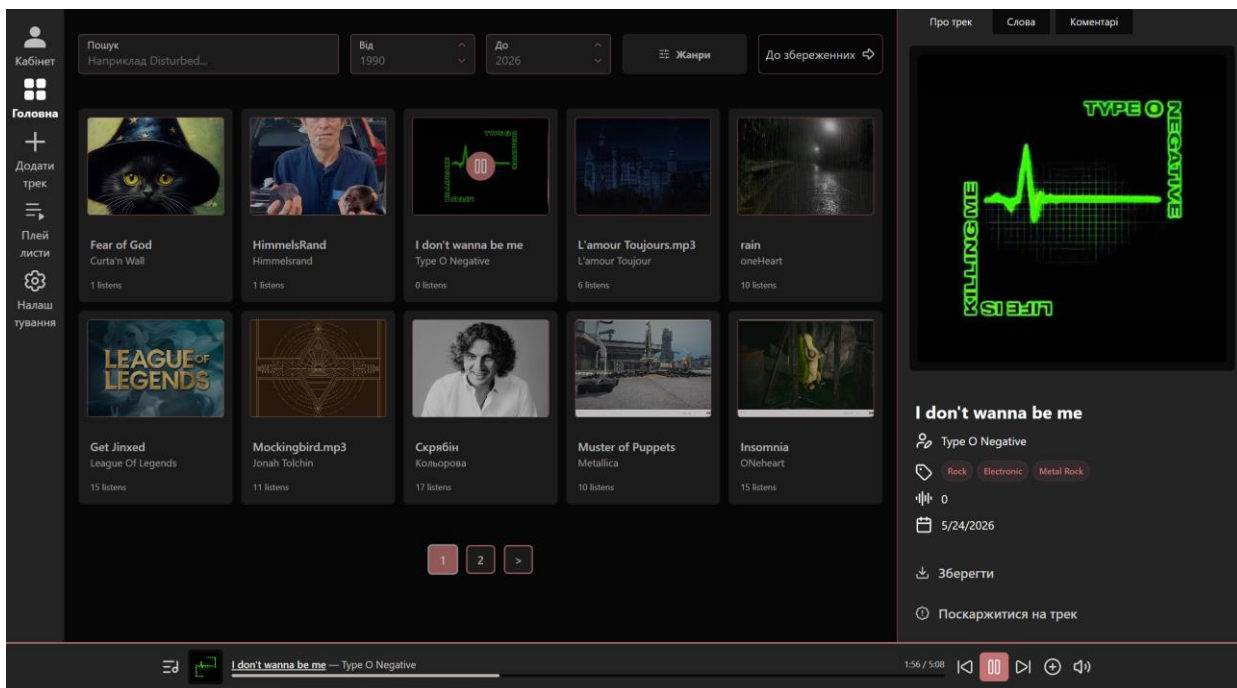


Рис. 3.8. Перегляд детальної інформації про трек

Вкладка з коментарями (рис. 3.9) являє собою перелік вже існуючих коментарів двох видів, а саме з марками часу та без них, при натисканні на час праворуч від тексту коментаря, прогрес програвання треку буде встановлений чітко на той же час. Та можливість залишити власний коментар, знову ж таки з можливістю зазначити час до якого має бути прив'язано даний коментар або ж без нього.

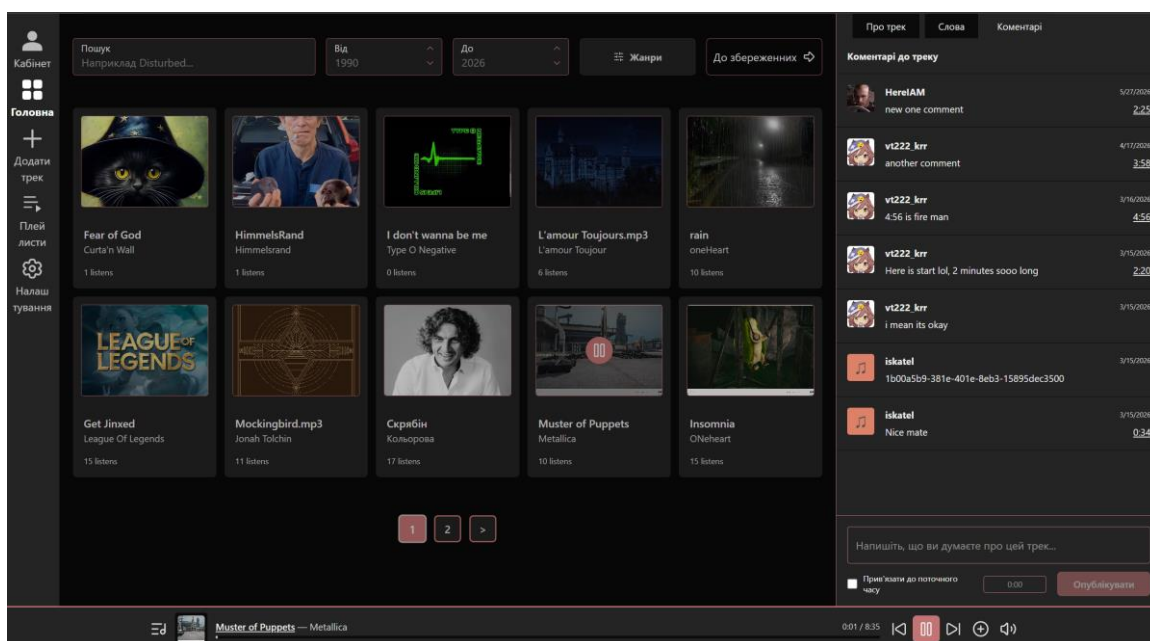


Рис. 3.9. Вкладка коментарів

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

В контекстному меню зареєстрованого користувача з'явилась функція збереження треку. Системою передбачено збереження до 10 композицій за для уникання занадто великого розміру локальної бази даних, після натискання на обрану опцію, трек з'явиться на відповідній сторінці (рис. 3.10) з можливістю прослухати його без прямого підключення до мережі інтернет.

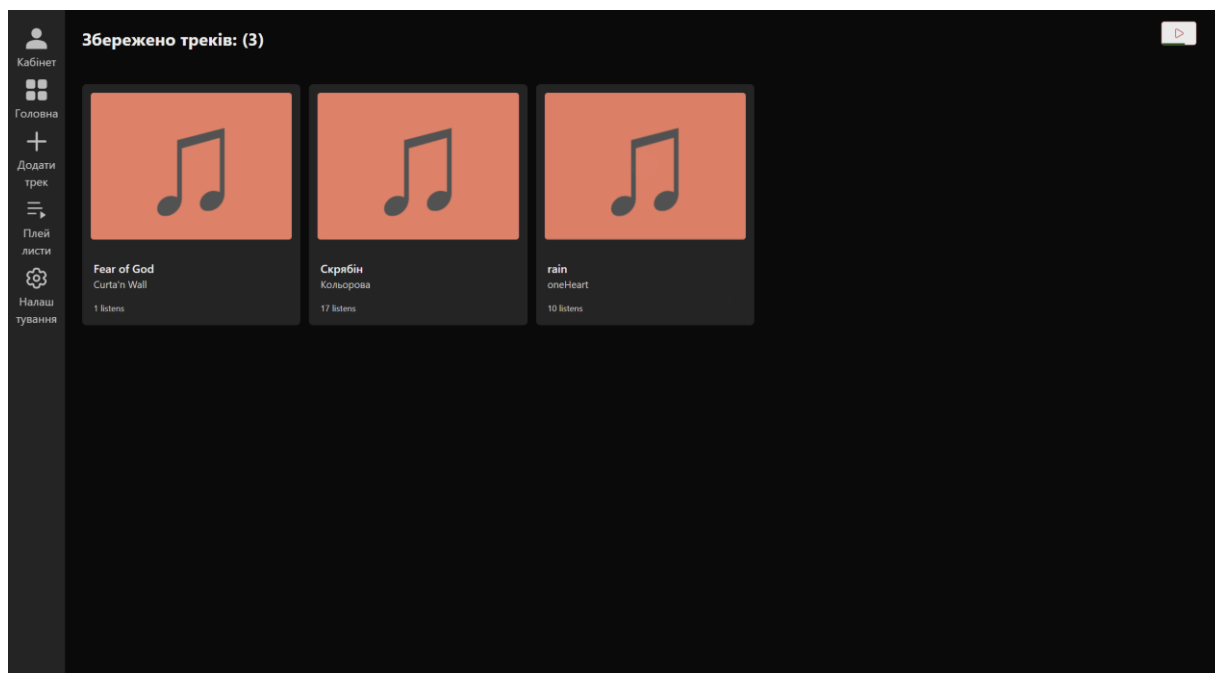


Рис. 3.10. Сторінка збережених треків

Далі сторінка плейлистів, на головній сторінці відображаються всі плейлисти доступні поточному користувачу, а це власноруч створені, відкриті для всіх, тобто публічні, та ті доступ до яких було надано особисто поточному користувачу через відповідну сторінку контролю доступу про яку буде описано нижче. Тут же в блоці зверху є можливість знайти конкретний плейлист за назвою у випадку коли шукати в ручну перебираючи всі списки відтворень за допомогою пагінації занадто довго. Та кнопка відкриття форми (рис. 3.10) для створення нового власного плейлиста, з полями назви опису, полем для завантаження обкладинки плейлиста та полем яке відповідає за встановлення відкритого доступу до створюваного списку відтворення для всіх користувачів системи.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

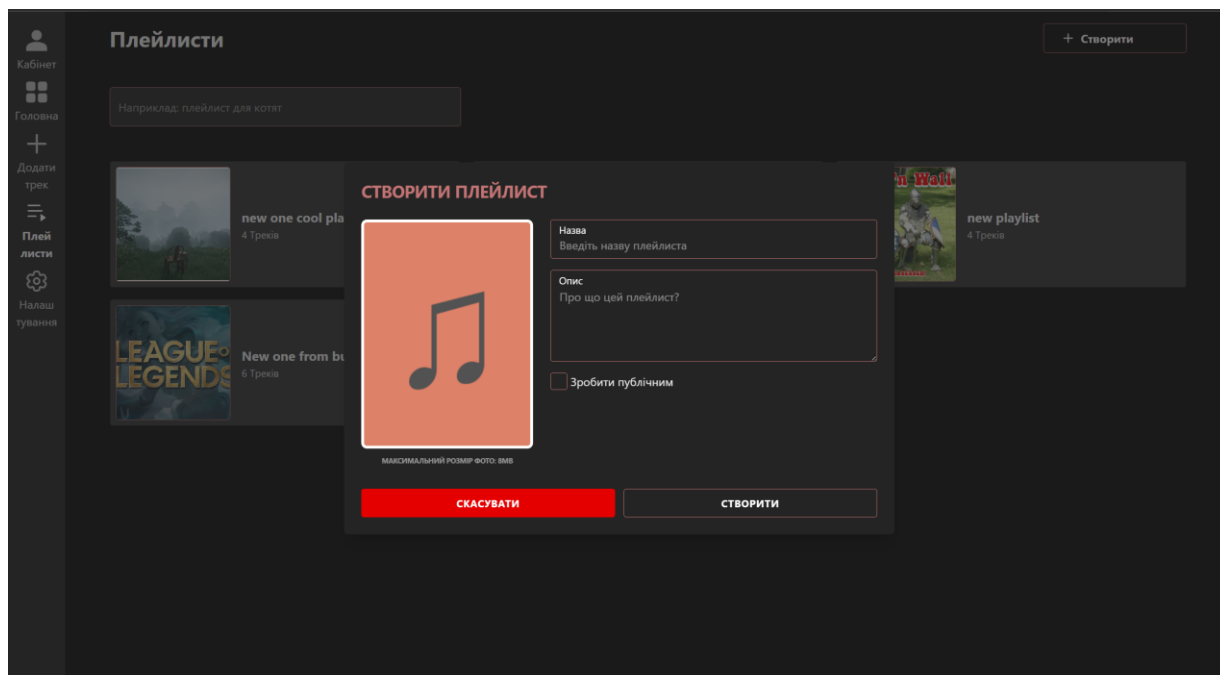


Рис. 3.11. Створення нового списку відтворення

Сторінка обраного списку відтворення з усіма можливими інструментами для керування через найвищий рівень доступу до ресурсу представлена на рисунку 3.12:

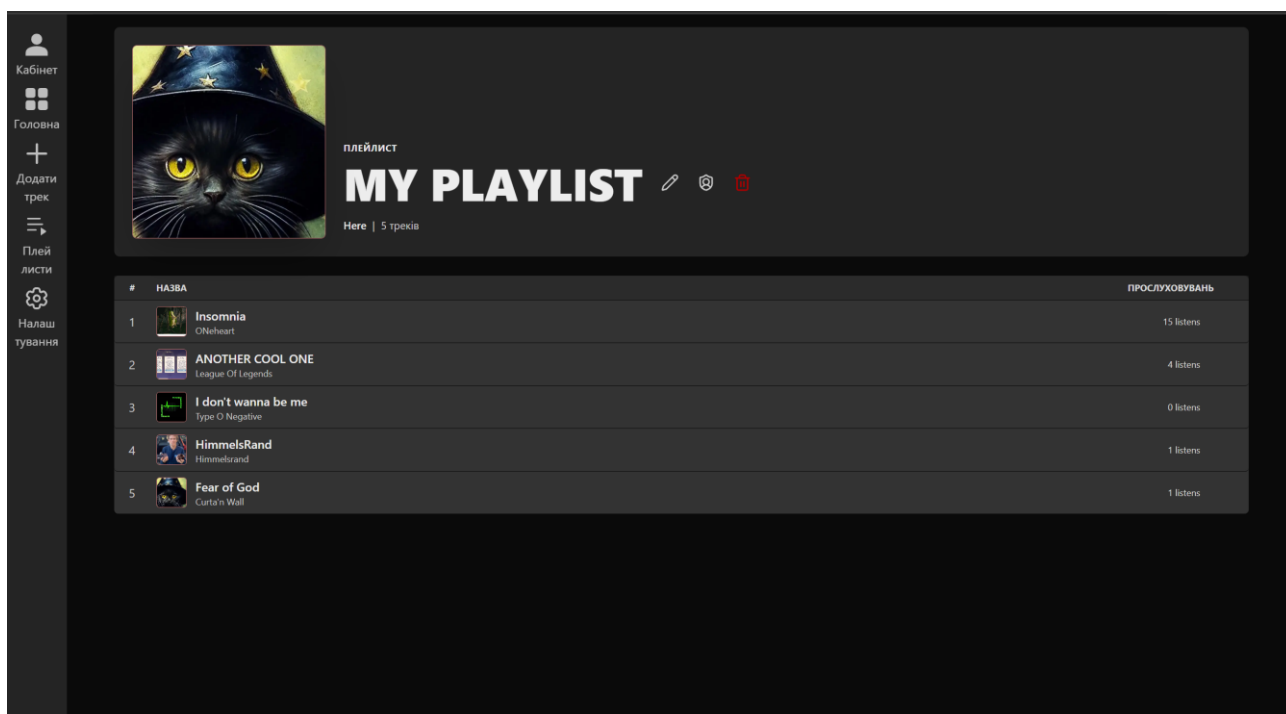


Рис. 3.12. Сторінка обраного списку відтворення

Список треків представлений на сторінці відображається в порядку додавання, тут же є можливість видалити вже існуючи трек зі списку попередньо доданий через форму (рис. 3.13), яка відкривається шляхом натискання відповідних опцій безпосередньо в плеєрі або ж контекстному меню.

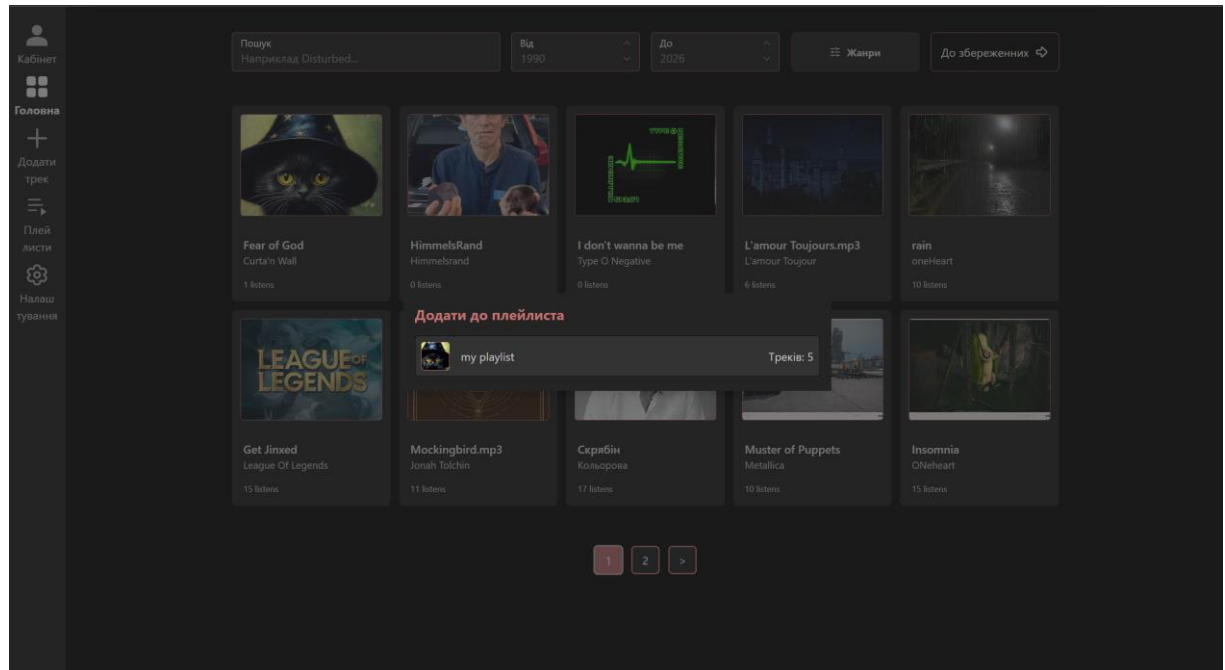


Рис. 3.13. форма додавання обраної композиції до плейлиста

Звідси користувач може перейти на сторінку керування доступом (рис 3.14) де представлено список усіх раніше доданих співавторів до обраної добірки, включно з автором, звідки має можливість додавати будь якого користувача системи за ім'ям, змінювати доступ вже існуючих, надаючи їм на вибір одну з ролей, від звичайного слухача до повноцінного власника плейлиста, або ж взагалі видаляти їх зі списку відповідною кнопкою.

Усі дії супроводжуються листами на вказану при реєстрації та підтверджену пошту користувача зміна прав якого відбувається в даний момент, з повідомленням про те що їм було надано певні права доступу до плейлиста керування якого відбувається, та кнопкою переходу до відповідного списку відтворення.

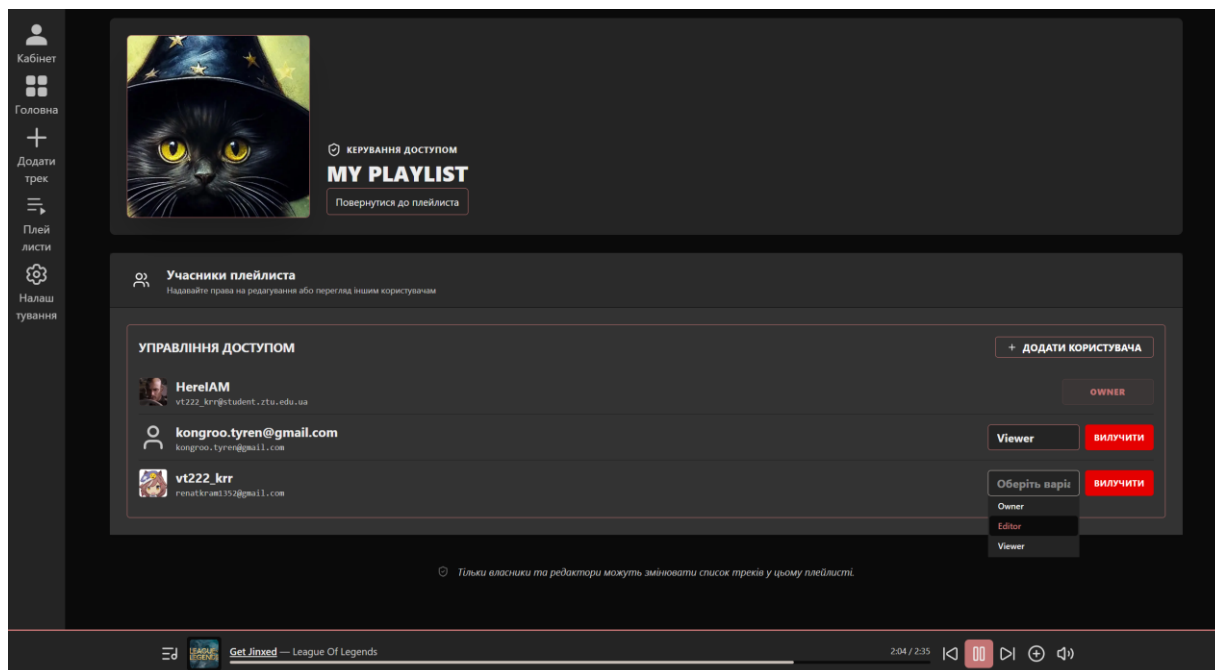


Рис. 3.14. Сторінка керування доступом списку відтворення

Лист який надходить на пошту при зміні прав доступу або ж початковому запрошенні до спільного керування списком відтворення продемонстровано на рисунку 3.15 :

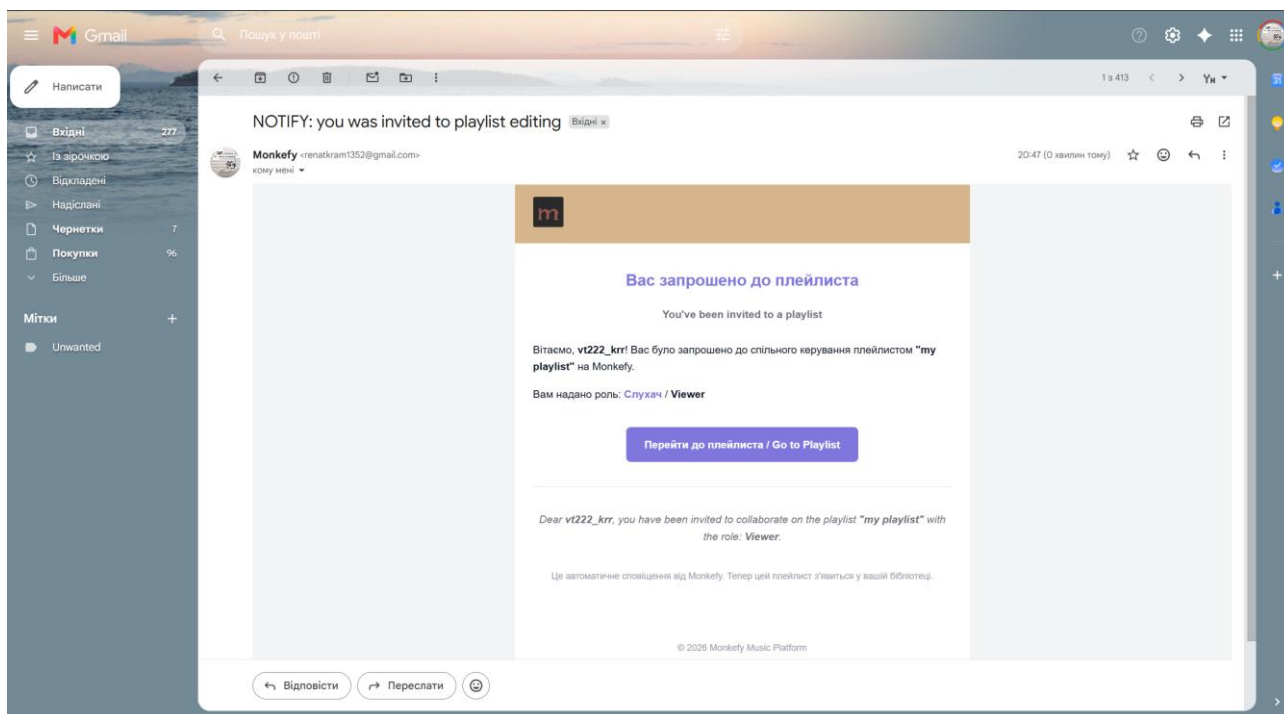


Рис. 3.15. Лист сповіщення про додавання до списку відтворення

Варто зазначити що як зареєстрований користувач так і гість вебзастосунку має можливість перейти до сторінки візуалізації аудіо, де на вибір надається три варіанти, а саме кругова спектрограма (рис. 3.16), стовпаста, та стовпчаста з паралельним відображенням коментарів відносно їх часових міток.

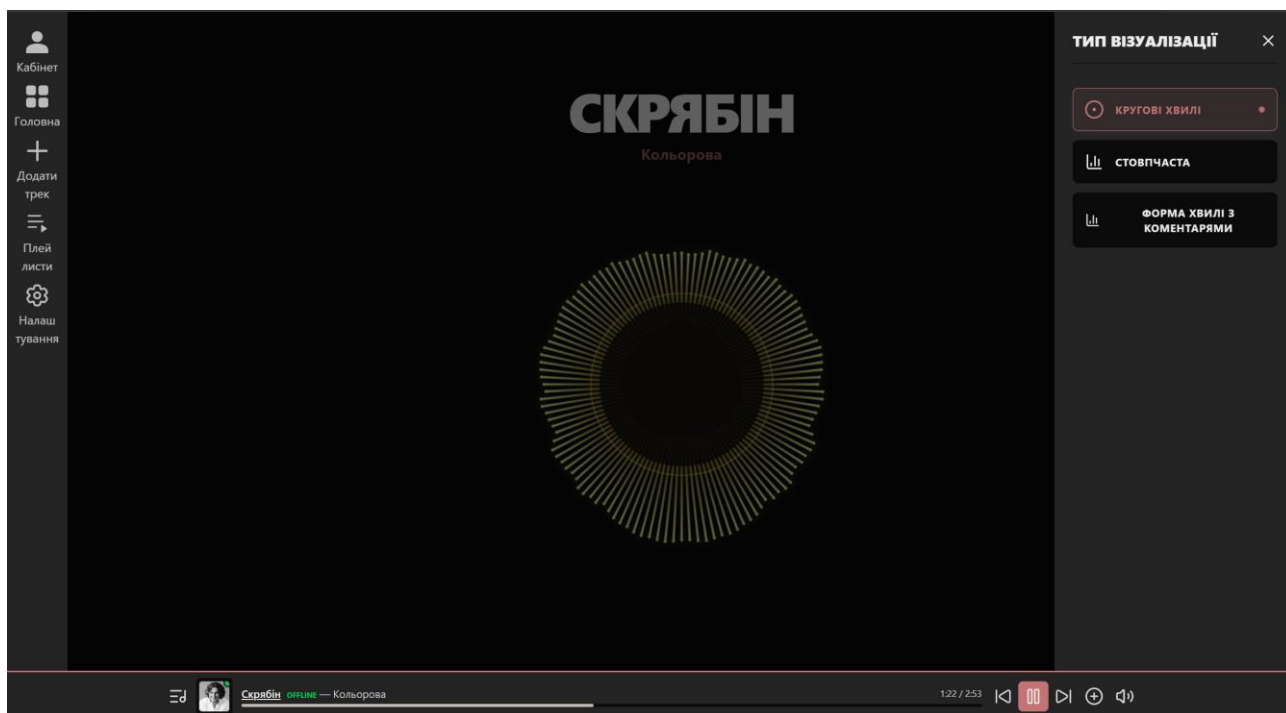


Рис. 3.16. Сторінка візуалізації аудіо в реальному часі з обраною круговою спектрограмою

Користувач що завантажив трек, який успішно пройшов модерацию, та потрапив в глобальний пул композицій, має можливість через контекстне меню перейти до сторінки, яка являє собою центр управління контентом (рис. 3.17), тут же представлена проста статистика по додаванню композиції у списки відтворення користувачів системи за останні 7 днів, та форма редагування завантаженого треку. У випадку ж якщо користувач вирішив знову завантажити аудіо, можливо через помічений брак звуку, або ж застарілу версію, композиція пропадає з глобального пулу та знову потрапляє в чергу модерації, очікуючи вердикту.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

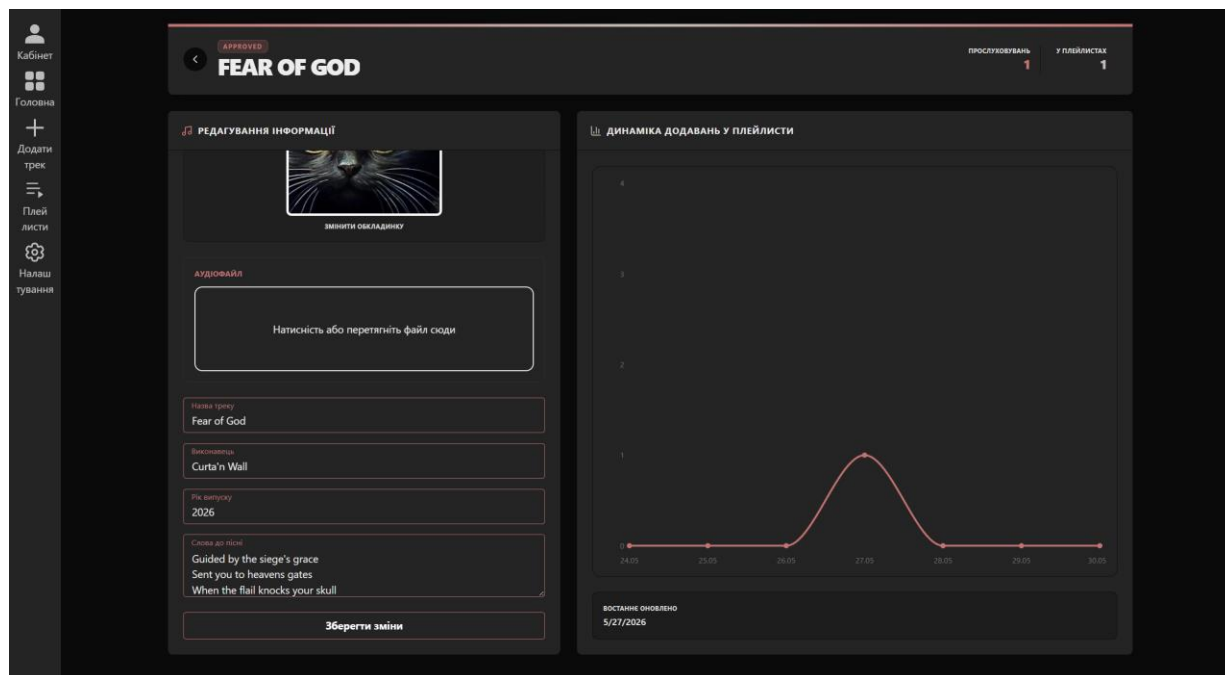


Рис. 3.17. Сторінка керування завантаженої композиції

Останній варіант використання системи звичайним користувачем це ядро веб застосунку, форма завантаження власних треків, де користувач одразу має можливість обрати файл обкладинки та мінімально змінити аудіо файл обрізавши його у відповідному блоці, представлена на рисунку 3.18 :

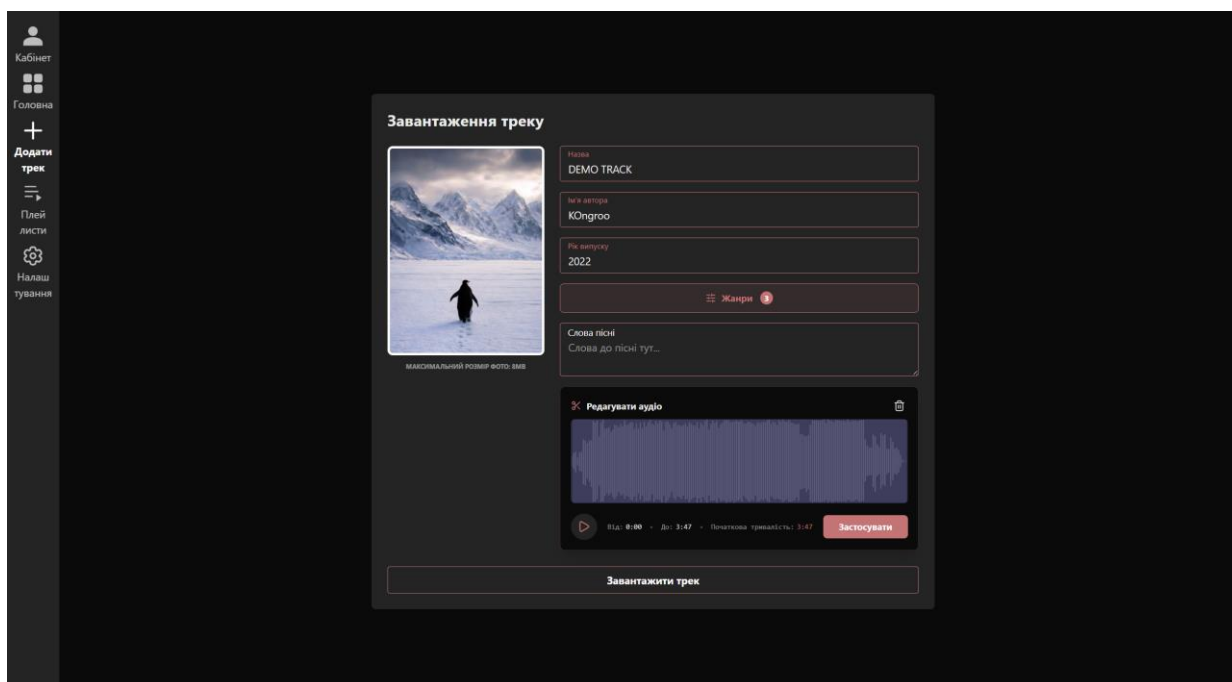


Рис. 3.18. Форма завантаження треку

На черзі сторінка панелі модератора, яка відповідно доступна тільки користувачам з роллю модератора, представлена на рисунку 3.19.

Кабінет Панель модератора Головна Додати трек Плей листи Налаштування	Треки Скарги		Модерація треків				
	Назва треку		Статус	Дата з	Дата по	Сортування	Лист
	Пошук за назвою...		Усі статуси	mm/dd/yyyy --:-- --	mm/dd/yyyy --:-- --	Спочатку нові	20
	38 результатів знайдено						
	Трек	Автор	Модератор	Статус	Дата		
	DEMO TRACK	Here	Не призначено	Очікує	5/30/2026, 09:02 PM		
	Updated	Here	vt222_krr	Відхилено	5/27/2026, 06:59 PM		
	afada	vt222_krr	vt222_krr	Схвалено	5/27/2026, 06:25 PM		
	Fear of God	Here	vt222_krr	Схвалено	5/27/2026, 11:32 AM		
	It works	vt222_krr	vt222_krr	Відхилено	5/27/2026, 11:29 AM		
	Amortalis	vt222_krr	vt222_krr	Відхилено	5/27/2026, 11:23 AM		
	Delete this one	vt222_krr	vt222_krr	Відхилено	5/27/2026, 11:19 AM		
	The last one	vt222_krr	vt222_krr	Відхилено	5/27/2026, 11:11 AM		
	φIBφiφ	vt222_krr	vt222_krr	Відхилено	5/27/2026, 11:09 AM		
	asca	vt222_krr	vt222_krr	Відхилено	5/27/2026, 11:04 AM		
	asdfasd	vt222_krr	vt222_krr	Відхилено	5/27/2026, 10:56 AM		
	asdfasd	vt222_krr	vt222_krr	Відхилено	5/27/2026, 10:54 AM		
	ssfd13r3	vt222_krr	vt222_krr	Відхилено	5/27/2026, 10:52 AM		
	addasd	vt222_krr	vt222_krr	Відхилено	5/27/2026, 10:48 AM		
	adsdas	vt222_krr	vt222_krr	Відхилено	5/27/2026, 10:46 AM		
	adsas	vt222_krr	vt222_krr	Відхилено	5/27/2026, 10:44 AM		
	1123123123	vt222_krr	vt222_krr	Відхилено	5/27/2026, 10:41 AM		
	aadsa	vt222_krr	vt222_krr	Відхилено	5/27/2026, 10:31 AM		

Рис. 3.19. Сторінка черги завантажуваних треків

Тут же представлено список усіх модерацій, за весь час існування системи, звісно ці дані будуть очищуватись через певний проміжок часу, близько трьох місяців, однак до того моменту вони будуть зберігатися у форматі архіву, у верхньому блоці розміщено пункт перемикання між вкладками модерації черги завантажуваних треків та скарг на вже завантажені композиції, сторінки практично ідентичні, в обох варіантах, при натисканні на будь-який з представлених рядків в списку відкривається бічна панель (рис. 3.20) з усією необхідною інформацією про композицію, про користувача що завантажив її, у випадку скарги це ще й причина скарги від користувача що поскаржився. При завантаженні кожна композиція першочергово відправляється в описаний раніше сервіс для перевірки на авторське право, та у випадку знаходження співпадінь, відображається відповідний блок в бічній панелі. Однак остаточний вердикт завжди залишається за модератором.

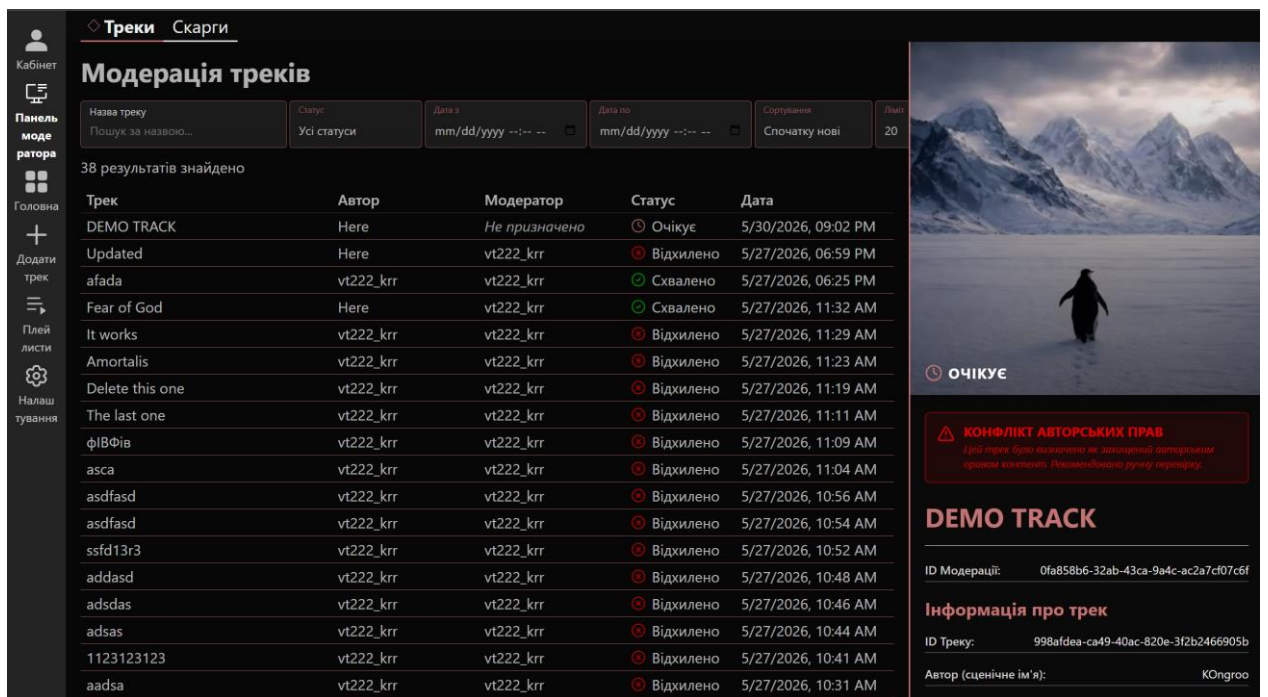


Рис. 3.20. Вигляд бокової панелі сторінки модерації

Процес винесення вердикту, відбувається через відповідний блок внизу бокової панелі та модального вікна з підтвердженням, який представлено на рисунку 3.21 :

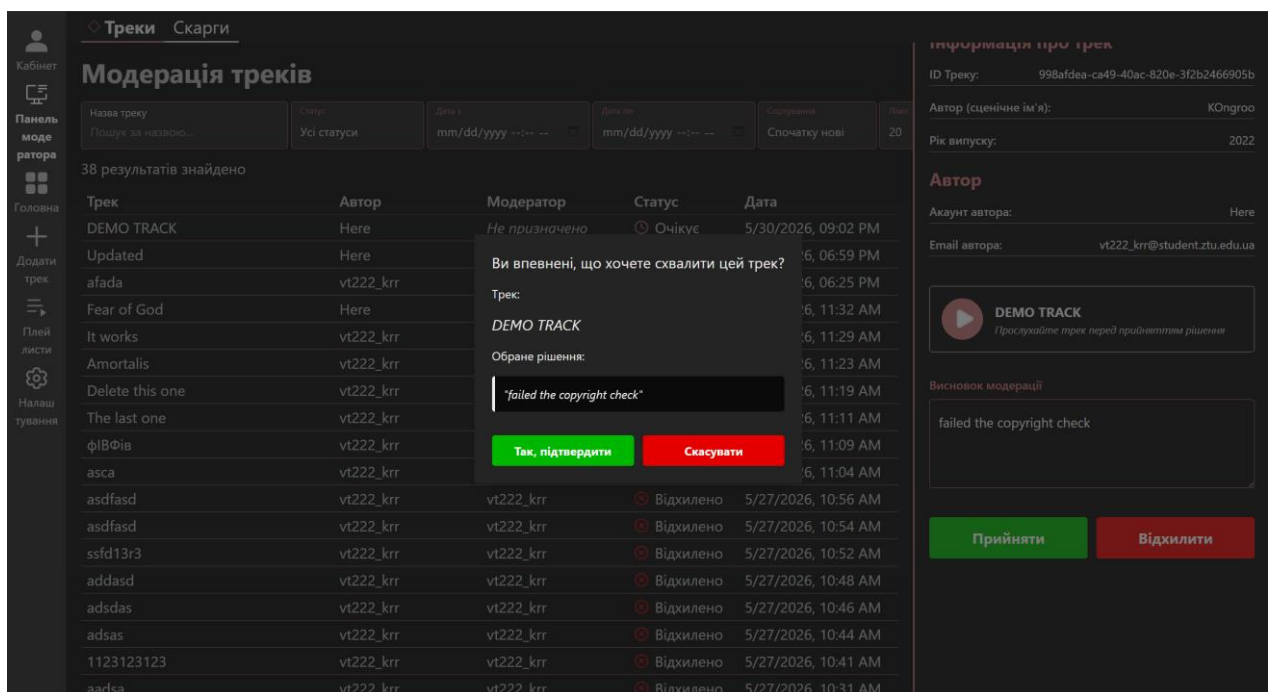


Рис. 3.21. Форма винесення вердикту завантаження

Останнім варіантом використання системи стає сторінка моніторингу інформації по використаним ресурсам (рис. 3.22) на хмарному об'єктовому сховищі AWS S3 яке використовується для зберігання усіх медіафайлів системи, включно з фото профілю користувачів та статичних файлів необхідних наприклад для листів на пошту. Дана сторінка доступна тільки користувачу з роллю Адміністратор, яка включає в собі можливості всіх минулих ролей.

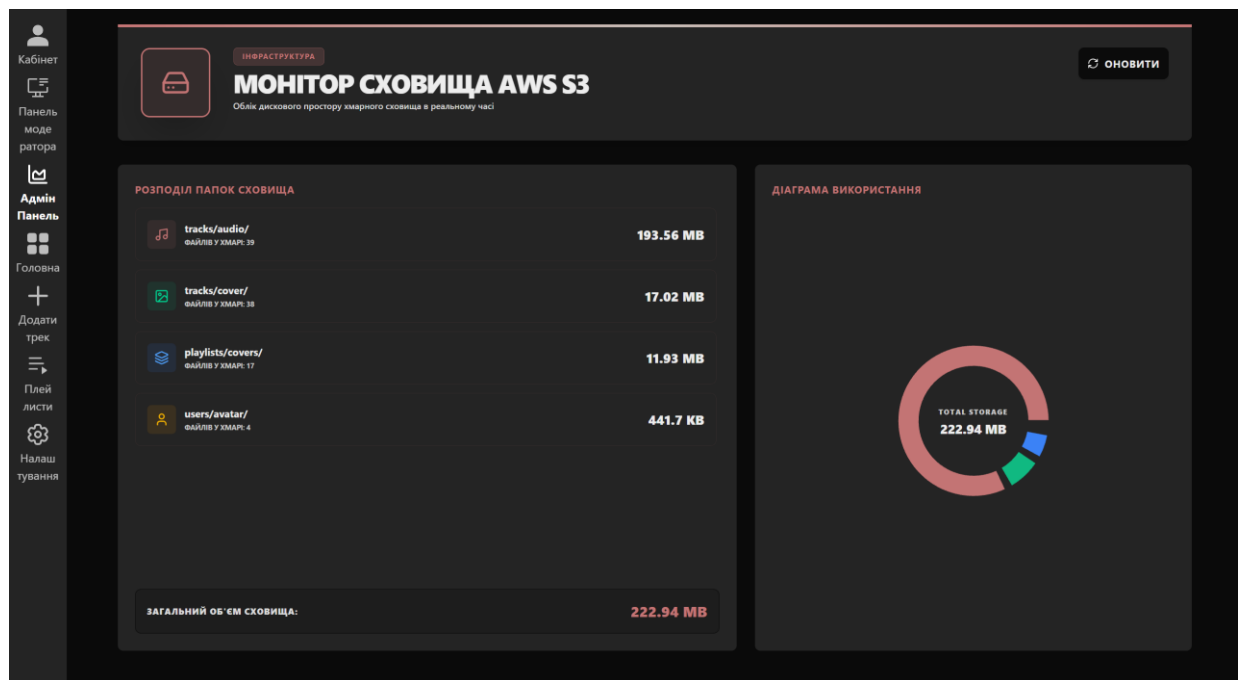


Рис. 3.22. Сторінка моніторингу використаних ресурсів системи

3.3. Тестування роботи програмного комплексу

Окремим пунктом процесу розробки варто виділити тестування всіх можливих варіантів використання системи, в тому числі валідацію основних форм та аналіз отриманих часових міток при запитах на сервер та отриманні запитуваних даних.

Тестування розробленого програмного комплексу здійснювалося ручним методом та проходило у кілька ітерацій. Повторні цикли перевірок дозволили послідовно виявити критичні архітектурні та інтерфейсні дефекти, провести їх чітке налагодження та виконати фінальну регресійну перевірку.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

Крім того, випробування проводилися на декількох типах фізичних пристроїв із різною роздільною здатністю екранів та відношенням сторін у декількох популярних сучасних браузерів а саме Google Chrome, Firefox та Opera, що дозволило переконатися в універсальності та адаптивності створеного вебдодатку.

Для систематизації процесу мануальної верифікації було сформовано перелік базових тест-кейсів, які відображають критичні функціональні операції ядра платформи (табл. 3.1).

Таблиця 3.1

Перелік тест-кейсів функціонального тестування платформи

Назва	Модуль	Очікуваний результат	Відповідність
Створення чернетки треку	Модуль завантаження	Чернетка успішно створюється в БД при заповненні обов'язкових текстових полів після чого потрапляє на модерацію. Незаповнені опціональні поля не блокують транзакцію.	Відповідає
Перевищення ліміту на модерацію треків	Модуль завантаження	При спробі ініціювати 6-й запит на завантаження за наявності 5 активних треків у статусі pending_moderation система блокує операцію та повертає помилку.	Відповідає
Пряме завантаження файлу в S3	Інфраструктура (AWS S3)	Клієнтський браузер успішно отримує Presigned URL та транслює аудіофайл за протоколом HTTP PUT в обхід основного сервера додатка. У випадку підміни файлу, завантаження блокується вказаними правилами сховища.	Відповідає
Фільтрація медіатеки за декількома опціями	Модуль пошуку	Система валідно працює у випадку коли користувач у відповідних компонентах фільтрації обирає одразу декілька варіантів, напр. пошук за назвою, починаючи з 2020 року випуску та з жанром Metal	Відповідає
Валідація розміру завантажуваних файлів	Модуль завантаження та створення списку відтворення	У випадку якщо користувач завантажує файл, розмір якого більше ніж вказано у відповідній формі, користувач бачить чітке повідомлення помилки яке описує причину відмови завантаження	Відповідає

Чітке зображення частотного спектру звуку без затримок	Модуль візуалізації	Користувач бачить чітко зображені кругову та стовпчасту спектрограми, звук та коливання різних частот, відповідають змінам у висоті стовпчиків спектрограми, без помітних затримок.	Відповідає
Винесення вердикту модерації	Модуль модерації	Після винесення вердикту модерацією, користувач що ініціював процедуру завантаження композиції до системи отримує чіткі повідомлення на пошту з відповідним статусом, та повідомленням якщо його було додано	Відповідає

Під час тестування особливу увагу було приділено суворій валідації вхідних даних у формах (рис. 3.23). Усі форми, що відповідають за модифікацію та зміну параметрів уже існуючих сутностей бази даних, спроектовані таким чином що у випадку виникнення помилок (відсутність обов'язкових полів, некоректний формат введення тощо) система миттєво реагує та повертає користувачу зрозумілі й інформативні текстові сповіщення.

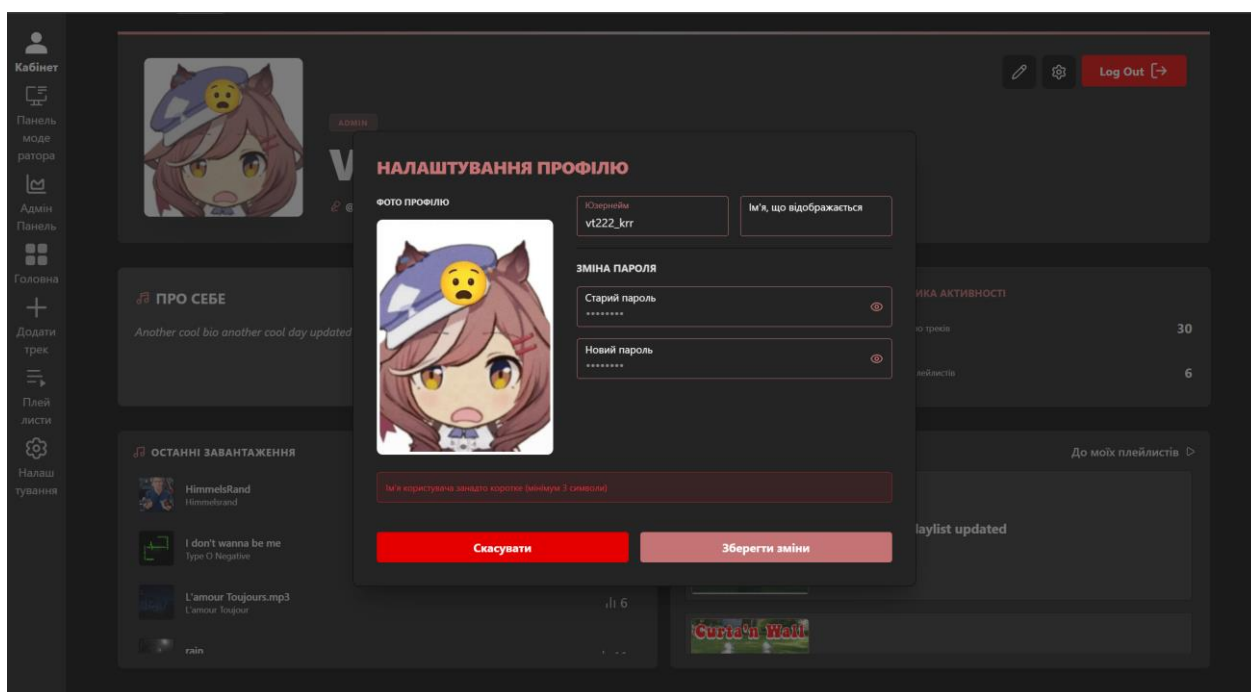


Рис. 3.23. Приклад валідації форми редагування профілю

Основними компонентами системи, які першочергово підверглися

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		73

тестуванню стала спроможність безперервно та без затримок відтворювати аудіоконтент та його візуалізації. Перевірка підтвердила, що завантаження та буферизація медіафайлів для безперервного програвання відбуваються у межах розрахованого та описаного раніше допустимого часу очікування. Аудіоплеєр коректно взаємодіє з чергою відтворення, оперативно реагує на команди перемикавання треків, паузи та регулювання гучності без затримок звукового потоку. А сторінка візуалізації контенту відмальовує хвили у всіх трьох представлених видах спектрограм, враховує роздільну здатність екранів. Окремим пунктом стала швидкодія візуалізації, адже як правило обробка звуку на клієнті може стати занадто ресурсозалежним процесом, однак через обраний алгоритм та впровадження оптимального розміру вікна алгоритму, навантаження на пристрій не відчувається навіть на мобільних пристроях.

Модуль модерації та аналізу на авторські права теж було піддано тестуванню. У результаті чого тестування підтвердило здатність підсистеми паралельно оперувати декількома запитами до сторонніх сервісів без утворення внутрішніх черг на сервері. Процедура побудови акустичного відбитка звуку та звірки його із глобальною базою даних проходить у межах встановлених часових лімітів), забезпечуючи модератора оперативним результатом для прийняття рішення щодо публікації того чи іншого треку.

Загалом, результати проведення багаторазових мануальних перевірок підтвердили стійкість розробленого програмного комплексу до помилок користувача та швидку реакцію інтерфейсу на події від декількох користувачів одночасно.

Висновки до третього розділу

У третьому розділі було успішно реалізовано етап практичного розгортання, інфраструктурного налаштування та оптимізації надійності розроблюваного мультимедійного комплексу. На основі аналізу сучасних хмарних методологій було спроектовано та впроваджено гібридну архітектуру

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

розміщення платформи, де логіку ядра застосунку та інтерфейсні шари інтегровано у PaaS-середовище Vercel із автоматизацією CI/CD, а збереження й дистрибуцію важких бінарних об'єктів делеговано розподіленій інфраструктурі сховища AWS S3.

Таке інженерне рішення дозволило суттєво знизити обчислювальне навантаження на основні серверні вузли завдяки реалізації паттерну прямих клієнтських завантажень за допомогою тимчасових підписаних посилань. Безпеку збереження даних було додатково підсилено на рівні конфігурації хмарного бакету через суворе визначення політик доступу та конфігурацію правил спільного використання ресурсів між різними джерелами, що унеможливило сторонню модифікацію медіапростору

Окрему увагу приділено ізоляції найбільш ресурсомісткого мікросервісу копірайт-аналізу який було розгорнуто на незалежній платформі Railway.

У другому підрозділі наведено детальний аналіз структури розробленого користувацького інтерфейсу, включно з усіма можливими ролями системи та відповідний список функціональних повноважень, від неавторизованого гостя до модераторів та адміністраторів платформи.

У третьому параграфі описано процес тестування усіх основних функціональних можливостей системи в декілька ітерацій, в тому числі адаптивність та підтримуваність різними браузерами та пристроями.0

Результати проведених випробувань повністю підтвердили стійкість до відмов, низьку затримку відгуку інтерфейсних компонентів та повну готовність створеного вебдодатку до експлуатації.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		75

ВИСНОВКИ

В результаті виконання дипломної роботи було успішно реалізовано вебдодаток для інтерактивного прослуховування та візуалізації музичного контенту. Головна мета дослідження, яка полягала у створенні сучасної, відмовостійкої системи із вбудованими інструментами автоматизованої модерації контенту, була повністю досягнута. При проєктуванні та розробці програмного комплексу було вирішено комплекс інженерних та науково-практичних завдань, що дозволило забезпечити ефективне функціонування об'єкта дослідження як єдиної екосистеми включаючи всі описані під час дослідження завдання.

Відповідно до поставленого завдання, у межах програмного комплексу було успішно впроваджено та протестовано всі основні модулі системи, окремо варто виділити інтерактивний музичний плеєр із можливістю безперервного відтворення аудіопотоку, динамічного формування черги треків, регулювання параметрів звуку та збереження прогресу прослуховування разом з системою візуалізації аудіосигналів на основі частотного спектру, в реальному часі. А також автоматизована система премодерації завантажуваних треків на предмет дотримання авторських прав за допомогою хроматичного алгоритму на основі акустичних відбитків аудіо.

Ефективність та стабільність функціонування системи були досягнуті завдяки використанню сучасного технологічного стеку та закладанню мікросервісних інженерних патернів. Базовим каркасом системи став фреймворк Next.js, який дозволив реалізувати гібридну архітектуру рендерингу, забезпечивши високу швидкість первинного завантаження сторінок. Як реляційне сховище структурованих даних було використано систему керування базами даних PostgreSQL, взаємодія з якою оптимізована через Prisma ORM. Для довготривалого збереження важких медіафайлів було інтегровано хмарне об'єктове сховище AWS S3. Оптимізація взаємодії зі сховищем реалізована за допомогою механізму тимчасових підписаних

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		76

посилань, що дозволило клієнтським браузерам завантажувати файли в обхід основного сервера, радикально знижуючи навантаження на обчислювальні ресурси ядра платформи.

Впровадження окремого сервісу для перевірки композицій на авторські права на основі аналізу акустичних відбитків, дозволяє реалізувати захисний шар, для того щоб пришвидшити публікацію контенту незалежних виконавців безпосередньо на платформу, та уникнути юридичних проблем через використання ліцензованого контенту.

Напрямки подальшого вдосконалення програмного комплексу та перспективних наукових досліджень полягають в використанні алгоритмів машинного навчання для побудови системи рекомендацій композицій для окремого користувача на основі аналізу його історії прослуховувань та жанрових вподобань, або навіть за віковими групами. Крім того, подальші плани по впровадженню нового функціоналу передбачають впровадження протоколу WebSockets для реалізації функцій групового прослуховування, або аналогу прямих ефірів для музикантів.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		77

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bartsch M. A., Wakefield G. H. Audio Thumbnailing of Popular Music Using Chroma-Based Representations. IEEE Transactions on Multimedia. 2005. Vol. 7, no. 1. P. 96–104. URL: <https://dub.ucsd.edu/CATbox/Reader/ThumbnailingMM05.pdf> (дата звернення: 03.02.2026).
2. Booz R., Fritchey G. Introduction to PostgreSQL for the data professional. 17th ed. Red Gate Books, 2025. 240 p.
3. Cano P., Batlle E., Kalker T., Haitsma J. A Review of Audio Fingerprinting. Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology. 2005. Vol. 41, no. 3. P. 271–284. URL: <https://ismir2005.ismir.net/proceedings/1019.pdf> (дата звернення: 14.03.2026).
4. Hesmondhalgh D. Music Streaming around the World. London : Routledge, 2025. 288 p.
5. Lazzarini V. Spectral Music Design: A Computational Approach. Oxford : Oxford University Press, 2021. 488 p.
6. Lerch A. An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics. 2nd ed. Wiley-IEEE Press, 2023. 320 p.
7. Müller M. Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications. 2nd ed. Springer, 2021. 513 p.
8. Müller M. Fundamentals of Music Processing: Using Python and Jupyter Notebooks. Springer Nature, 2021. 495 p.
9. Pelly J. Mood Machine: The Rise of Spotify and the Costs of the Perfect Playlist. New York : Hachette Books, 2025. 320 p.
10. Spilker H. Platformed! How Streaming, Algorithms and Artificial Intelligence are Shaping Music Cultures. Equinox Publishing, 2024. 256 p.
11. Thomas G. Real-World Next.js: Build scalable, commercially viable web applications using Next.js. Packt Publishing, 2022. 314 p. URL: <https://library.knu.edu.af/opac/temp/14271.pdf> (дата звернення: 07.02.2026).

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

12. Yan K., Hoiem D., Sukthankar R. Computer Vision for Music Identification. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). 2005. P. 881–888. URL: <https://dhoiem.cs.illinois.edu/publications/cvpr2005-mr.pdf>

13. AcoustID project statistics. AcoustID: Open audio fingerprinting. 2026. URL: <https://acoustid.org/stats> (дата звернення: 22.04.2026).

14. Amazon S3 Lifecycle Policies for Media Content Optimization. AWS Documentation. 2026. URL: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-mgmt.html> (дата звернення: 11.03.2026).

15. AnalyserNode Interface Specification in Web Audio API. W3C Recommendation. 2025. URL: <https://www.w3.org/TR/webaudio/#AnalyserNode> (дата звернення: 27.02.2026).

16. Connecting to PostgreSQL database with Prisma ORM. Prisma Documentation. 2026. URL: <https://www.prisma.io/docs/postgres/database/connecting-to-your-database> (дата звернення: 17.02.2026).

17. Docker Containerization Best Practices for Node.js Applications. Docker Docs. 2025. URL: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/ (дата звернення: 28.04.2026).

18. Essentia tutorial: Audio fingerprinting with Chromaprint. Essentia Library Documentation. 2025. URL: https://essentia.upf.edu/tutorial_fingerprinting_chromaprint.html (дата звернення: 04.06.2026).

19. Fast Fourier Transform (FFT) in Web Audio API AnalyserNode. MDN Web Docs. 2026. URL: <https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode/fftSize> (дата звернення: 04.06.2026).

20. GitHub Actions Documentation: Automating workflows. GitHub Docs. 2026. URL: <https://docs.github.com/en/actions> (дат звернення: 15.04.2026).

21. GitLab CI/CD Pipeline Configuration Reference. GitLab Docs. 2026.

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

URL: <https://docs.gitlab.com/ee/ci/> (дата звернення: 17.04.2026).

22. Likhite S. Why you should use nuqs: Smarter URL state management for React and Next.js. Medium. 2024. URL: <https://medium.com/@ruverd/why-you-should-use-nuqs-smarter-url-state-management-for-react-next-js-26a8b51ca1ac> (дата звернення: 10.03.2026).

23. Next.js App Router API Reference: Use Server Directive. Next.js Documentation. 2026. URL: <https://nextjs.org/docs/app/api-reference/directives/use-server#using-server-functions-in-a-client-component>

24. Next-intl configuration with Next.js App Router plugin. Next-intl Documentation. 2026. URL: <https://next-intl.dev/docs/usage/plugin> (дата звернення: 22.02.2026).

25. Node.js v24.0.0 Runtime Environment API Reference. Node.js Documentation. 2026. URL: <https://nodejs.org/docs/latest/api/> (дата звернення: 26.02.2026).

26. Pg_cron: Run periodic jobs in PostgreSQL databases. Citus Data Documentation. 2025. URL: https://github.com/citusdata/pg_cron (дата звернення: 04.04.2026).

27. Supabase as a Serverless Database Solution. Bejamas Hub. 2024. URL: <https://bejamas.com/hub/serverless-database/supabase> (дата звернення: 12.02.2026).

28. Svec L. How does Chromaprint work? Oxygene Blog. 2011. URL: <https://oxygene.sk/2011/01/how-does-chromaprint-work/> (дата звернення: 23.04.2026).

29. Tailwind CSS v4.0: Developer Reference Guide. Tailwind CSS Docs. 2025. URL: <https://tailwindcss.com/docs>

30. YouTube Content ID System: How Matching and Policies Work. Third Chair Blog. 2024. URL: <https://usethirdchair.com/blog/content-id-system-youtube-how-matching-and-policies-work> (дата звернення: 19.05.2026).

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		80

ДОДАТКИ

Пошук збігів акустичних відбитків

```
app.post('/check-copyright', upload.single('audio'), async (req: Request, res: Response):
Promise<any> => {
  if (!req.file) return res.status(400).json({ error: 'No file uploaded' });
  const filePath = req.file.path;
  const fixedPath = `${filePath}_fixed.wav`;
  try {
    const { stderr } = await exec(`ffmpeg -y -i "${filePath}" -ar 44100 -ac 2
"${fixedPath}"`);
    const stats = fs.statSync(fixedPath);
    const { fingerprint, duration } = await exec(`fpcalc -length 120 -json "${fixedPath}"`)
    .then(({ stdout }) => JSON.parse(stdout))
    .catch((err) => {
      if (err.stdout) return JSON.parse(err.stdout);
      throw err;
    });
    const acoustIdUrl =
`https://api.acoustid.org/v2/lookup?client=${ACOUSTID_KEY}&duration=${Math.roun
d(duration)}&fingerprint=${fingerprint}&meta=recordings+releases`;
    const response = await fetch(acoustIdUrl);
    const data: any = await response.json();
    res.json({
      success: true,
      isCopyrighted: (data.results || []).some((match: any) => match.score > 0.8),
      score: data.results?.[0]?.score || 0,
      matches: data.results?.[0]?.recordings || []
    });
  }
});
```

```
    });  
  } catch (error) {  
    console.error("Copyright analysis failed:", error);  
    res.status(500).json({ error: 'Processing failed', details: error instanceof Error ?  
error.message : 'Unknown error' });  
  } finally {  
    [filePath, fixedPath].forEach(path => {  
      if (fs.existsSync(path)) fs.unlinkSync(path);  
    });  
  }  
});
```

Реалізація кастомного хука для отримання масиву частот

```
export const useVisualizer = () => {  
  const { analyser, isPlaying } = usePlayer();  
  const [frequencies, setFrequencies] = useState<number[]>(new Array(128).fill(0));  
  const animationRef = useRef<number | null>(null);  
  useEffect(() => {  
    if (!analyser) return;  
    const dataArray = new Uint8Array(analyser.frequencyBinCount);  
  
    const update = () => {  
      const context = analyser.context;  
      if (context instanceof AudioContext && context.state === 'suspended') {  
        context.resume();  
      }  
      analyser.getBytesFrequencyData(dataArray);  
      setFrequencies(Array.from(dataArray));  
      animationRef.current = requestAnimationFrame(update);  
    };  
    animationRef.current = requestAnimationFrame(update);  
    return () => {  
      if (animationRef.current) cancelAnimationFrame(animationRef.current);  
    };  
  }, [analyser]);  
  
  return frequencies;  
};
```

Ініціалізація аналізатора частот

```
const initAudioAnalyzer = () => {  
  if (analyser || !audioRef.current || sourceRef.current) return;  
  
  if (audioContextRef.current) {  
    if (audioContextRef.current.state === 'suspended') {  
      audioContextRef.current.resume();  
    }  
    return;  
  }  
  
  const AudioContextClass = window.AudioContext || (window as  
any).webkitAudioContext;  
  const context = new AudioContextClass();  
  const analyserNode = context.createAnalyser();  
  analyserNode.fftSize = 256;  
  
  try {  
    if (!sourceRef.current) {  
      const source = context.createMediaElementSource(audioRef.current);  
      source.connect(analyserNode);  
      analyserNode.connect(context.destination);  
      sourceRef.current = source;  
    }  
    audioContextRef.current = context;  
    setAnalyser(analyserNode);  
  }
```

```
} catch (e) {  
    console.warn("Analyzer connection failed:", e);  
}  
};
```