ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Вебзастосунок для прослуховування аудіофайлів з використанням багатокритеріальних фільтрів та рекомендаційної системи»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп’ютерні науки»
Освітня програма «Комп’ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-21
Денис КОРНІЄНКО
Керівник: старший викладач кафедри
комп’ютерних наук
Олег ЛУКУТІН

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2025 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри
комп'ютерних наук
_____Сергій МІЧКІВСЬКИЙ
«____»____20___р

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Корнієнко Денис Дмитрович

Тема роботи	Вебзастосунок для прослуховування аудіофайлів з використанням багатокритеріальних фільтрів та рекомендаційної системи
Номер та дата наказу про затвердження теми	№121-7 від 24 грудня 2024 року
Коротка постановка завдання	Вебзастосунок для прослуховування аудіофайлів з використанням багатокритеріальних фільтрів та рекомендаційної системи
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1. Banks A. Вивчення React: Функціональна веб-розробка з React і Redux. // O'Reilly Media, 2020. (дата звернення: 07.04.2025) 2. Greg L. Full-stack React проекти: Вивчення розробки на стоці MERN шляхом створення сучасних веб-застосунків за допомогою MongoDB, Express, React та Node.js. (дата звернення: 07.04.2025)
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має передбачити теоретичне, системотехнічне або експериментальне дослідження складного спеціалізованого завдання або практичної проблеми в галузі комп'ютерних наук, яке характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 27 грудня 2024р.

Керівник Олег ЛУКУТІН

Здобувач освітнього ступеня бакалавра Денис КОРНІЄНКО

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
	Підготовчий етап		
1	Вибір напряму дослідження	02.12.2024 р.	*виконано*
2	Формування теми та призначення керівника	16.12.2024 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	23.12.2024 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	27.12.2024 р.	*виконано*
	Основний етап		
5	Розробка концепції кваліфікаційної роботи	13.01.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	20.01.2025 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	10.03.2025 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	24.03.2025 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	31.03-04.04.2025 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	07.04.2025 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	14.04.2025 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	21.04.2025 р.	*виконано*
	Завершальний етап		
13	Представлення рукопису для перевірки на плагіат	28.04-04.05.2025 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	05.05-11.05.2025 р.	*виконано*
15	Передзахист кваліфікаційної роботи	12.05-16.05.2025 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	19.05-06.06.2025 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	09.06-15.06.2025 р.	*виконано*
18	Доопрацювання доповіді та презентації для захисту	09.06-15.06.2025 р.	*виконано*
19	Захист кваліфікаційної роботи	16.06-22.06.2025 р.	*виконано*

Керівник Олег ЛУКУТІН

Здобувач освітнього ступеня бакалавра Денис КОРНІЄНКО

Корнієнко К.Д. Вебзастосунок для прослуховування аудіофайлів з використанням багатокритеріальних фільтрів та рекомендаційної системи

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 – Комп'ютерні науки (освітня програма – Комп'ютерні науки) СО Бакалавр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікативних технологій, кафедра комп'ютерних наук, Київ, 2025.

Описано розробку веб-застосунку для прослуховування аудіо файлів, який забезпечую зручне управління медіа-контентом, використовуючи багатокритеріальні фільтри для сортування та пошуку аудіо треків, а також реалізує рекомендаційну систему для покращення користувацького досвіду.

Ключові слова: вебзастосунок, аудіо файли, багатокритеріальний фільтр, пошукова система, рекомендаційна система.

Рис. 14. Бібліограф.: 19 найм.

Kornienko D.D. Web application for listening to audio files using multi-criteria filters and a recommender system.

Project explanatory note by specialty 122 – Computer science. – «KROK» University, Educational and Scientific Institute of Information and communication technologies, Department of Computer Science, Kyiv, 2025.

The development of a web application for listening to audio files is described, which provides convenient media content management by using multi-criteria filters for sorting and searching audio tracks, also implements a recommendation system to enhance the user experience.

Keywords: web-application, audio files, multi-criteria filter, search system, recommendation system.

Fig. 14. Bibliography: 19 Items.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

React – JavaScript-бібліотека, призначена для створення динамічних користувацьких інтерфейсів. Вона дозволяє ефективно оновлювати сторінки без перезавантажень, використовуючи віртуальний DOM, компоненти та хуки[5].

React Query – бібліотека для управління станом серверних даних. Автоматично кешує, оновлює та синхронізує дані з бекенда[6].

React Select – компонент для створення кастомізованих випадаючих списків[7].

React Icons – набір іконок, представлених у вигляді React-компонентів[8].

Typescript – мова програмування, яка є надбудовою над JavaScript з підтримкою статичної типізації. Дозволяє уникати багатьох помилок на етапі розробки[9].

TailwindCSS – утилітарний CSS-фреймворк, який дозволяє швидко створювати адаптивні інтерфейси[10].

WaveSurfer.js – JavaScript-бібліотека для візуалізації аудіохвиль. Дозволяє створювати інтерактивні waveform-компоненти, які відображають аудіохвилю та забезпечують його програвання[11].

Sonner – сучасна бібліотека для виведення нотифікацій. Забезпечує інформування користувача про успішне виконання дій[12].

Framer Motion – бібліотека для створення анімацій. Використовується для плавного зникнення/з’явлення елементів, переходів та інших інтерактивних ефектів[13].

MongoDB – документально-орієнтована нереляційна база даних, яка зберігає дані у форматі BSON[14].

ВСТУП

Актуальність теми. Сучасний світ свідчить про зростання інтересу до цифрового контенту, зокрема до аудіо файлів, серед користувачів як національного, так і міжнародного рівня. Число людей, які звертають увагу на аудіо контент, постійно зростає, оскільки доступ до музики та подкастів стає все більш зручним завдяки новітнім технологіям.

Одним із потужних інструментів розповсюдження аудіо файлів є веб-застосунки, які дозволяють організувати процес пошуку, прослуховування та рекомендацій аудіо треків з будь-якого місця, враховуючи різноманітні уподобання користувачів.

Таким чином, розробка веб-застосунку для прослуховування аудіо файлів з використанням багатокритеріальних фільтрів, пошукової та рекомендаційної системи є важливим і необхідним напрямком практичних досліджень, оскільки це дозволяє не тільки полегшити доступ до аудіо файлів, а й покращити персоналізований досвід користувачів, пропонуючи контент на основі їхніх інтересів та вподобань.

Мета дослідження. Метою проєкту є розробка програмного забезпечення для прослуховування аудіо файлів з використанням багатокритеріальних фільтрів, пошукової та рекомендаційної системи, що дозволяє користувачам зручно знаходити, сортувати та отримувати персоналізовані рекомендації аудіо треків відповідно до їхніх інтересів.

Для досягнення мети проєкту були виконані наступні завдання:

- розглянуто існуючі аналоги веб-застосунків для прослуховування аудіо файлів;
- визначені вимоги до програмного забезпечення системи;
- досліджено методи та підходи до створення рекомендаційної системи та багатокритеріальних фільтрів для сортування аудіо треків;
- спроектовано базу даних для зберігання інформації про аудіо файли

- розроблено веб-інтерфейс, який дозволяє користувачам шукати, переглядати та прослуховувати аудіо треки;
- створено систему фільтрації, що дозволяє клієнтам швидко знаходити треки відповідно до їхніх вимог та критеріїв, таких як виконавець, дату випуску тощо;
- розроблено програмне забезпечення системи, здійснено інтеграцію всіх компонентів та тестування їх працездатності.

Об’єктом дослідження є візуалізація та управління аудіо контентом, методи та алгоритми фільтрації аудіо треків за заданими критеріями, а також методи та підходи до формування рекомендацій для користувачів на основі їхніх інтересів.

Предметом дослідження є вебзастосунок для прослуховування аудіо файлів з використанням багатокритеріальних фільтрів, пошукової та рекомендаційної системи для персоналізованого досвіду користувачів.

Методи дослідження. Дослідження проводилося за допомогою різних наукових та дослідницьких методів, таких як аналіз існуючих рішень у сфері аудіо платформ для вивчення підходів до фільтрації, пошуку та рекомендацій. Також застосовувалися практичні методи програмування для розробки та вдосконалення веб-застосунку.

Практичне значення. Використання сучасних методів розробки покращує швидкість обробки аудіо даних, а також забезпечує надійність та безпеку при роботі з медіа-контентом. Інтеграція багатокритеріального пошуку дозволить користувачам швидко знаходити аудіо треки за різними параметрами, а рекомендаційна система покращить персоналізацію та допоможе підвищити взаємодію з користувачами.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОФАЙЛІВ З ВИКОРИСТАННЯМ БАГАТОКРИТЕРІАЛЬНИХ ФІЛЬТРІВ ТА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

1.1 Предметна область

Вебзастосунок для прослуховування аудіо файлів – це інтерактивна онлайн-платформа, що дозволяє користувачам прослуховувати, шукати, сортувати та отримувати рекомендації щодо аудіо треків через веб-браузер у режимі реального часу.

Основною функцією веб-застосунку є використання багатокритеріальних фільтрів, які дозволяють користувачам здійснювати пошук аудіо файлів за різними параметрами, такими як жанр, виконавець, дата випуску аудіо тощо. Це значно спрощує процес навігації та допомагає швидко знаходити потрібний контент відповідно до індивідуальних уподобань.

Метою проєкту є створення платформи для організації прослуховування аудіо контенту, яка надає користувачам змогу зручно керувати своїми аудіо треками, отримувати інформацію про композиції, а також відкривати для себе нову музику завдяки вбудованій рекомендаційній системі.

1.2 Огляд аналогів

На сучасному ринку існує велика кількість веб-застосунків та сервісів для прослуховування аудіо, серед яких найбільш популярними є Spotify, SoundCloud, Deezer, YouTube Music та Apple Music. Кожен з них має власні особливості, переваги та обмеження, які варто врахувати при розробці власного рішення.

Spotify (рис. 1.1) – один із найпопулярніших сервісів для прослуховування музики. Він має велику музичну бібліотеку, зручні добірки та хорошу систему рекомендацій[1].

Переваги:

- велика база треків від відомих артистів;
- персоналізовані добірки та плейлисти;
- зручний інтерфейс;
- працює на всіх платформах.

Недоліки:

- багато функцій тільки з підпискою;
- обмежена фільтрація;
- немає розширеного пошуку;
- локальне завантаження тільки з преміумом.

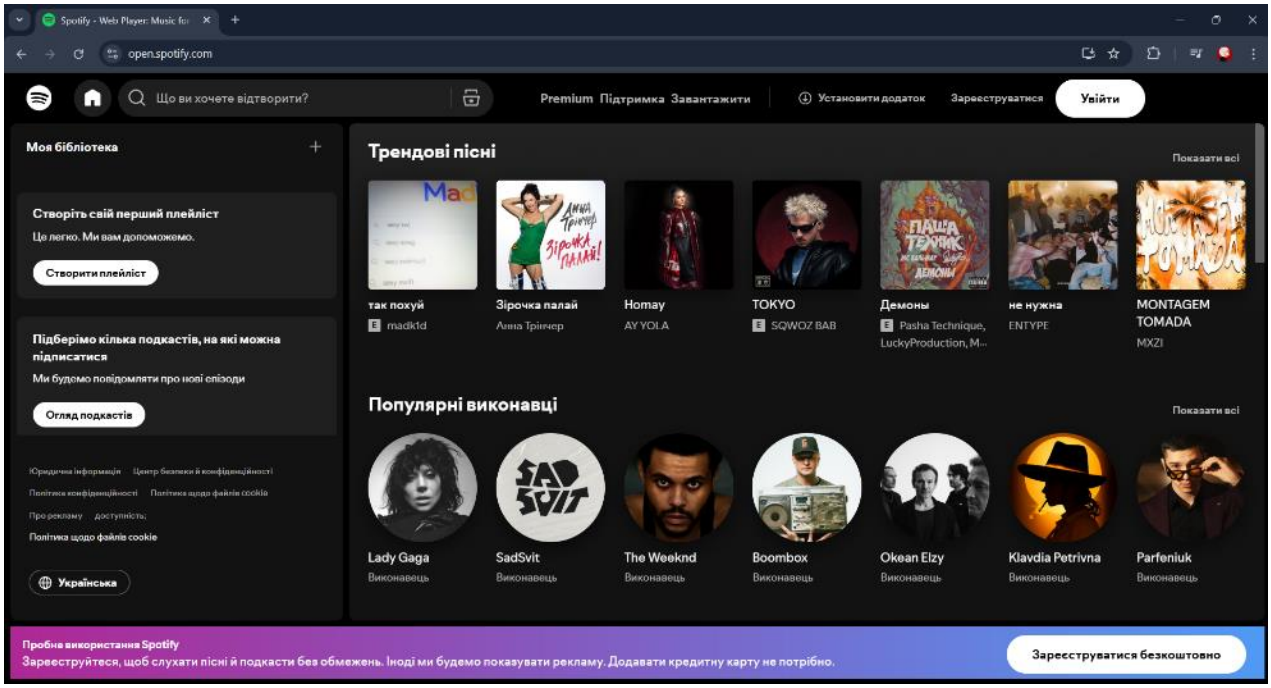

Рисунок 1.1 – Вебзастосунок «Spotify»

Джерело:[1]

SoundCloud (рис. 1.2) – це платформа, яка орієнтується більше на незалежних музикантів та аматорів[2].

Переваги:

- майданчик для нових та незалежних артистів;

- можна завантажувати власні треки;
- відкритість та різноманітність контенту.

Недоліки:

- обмежені можливості фільтрації;
- інтерфейс не завжди зручний;
- менше ліцензованої музики.

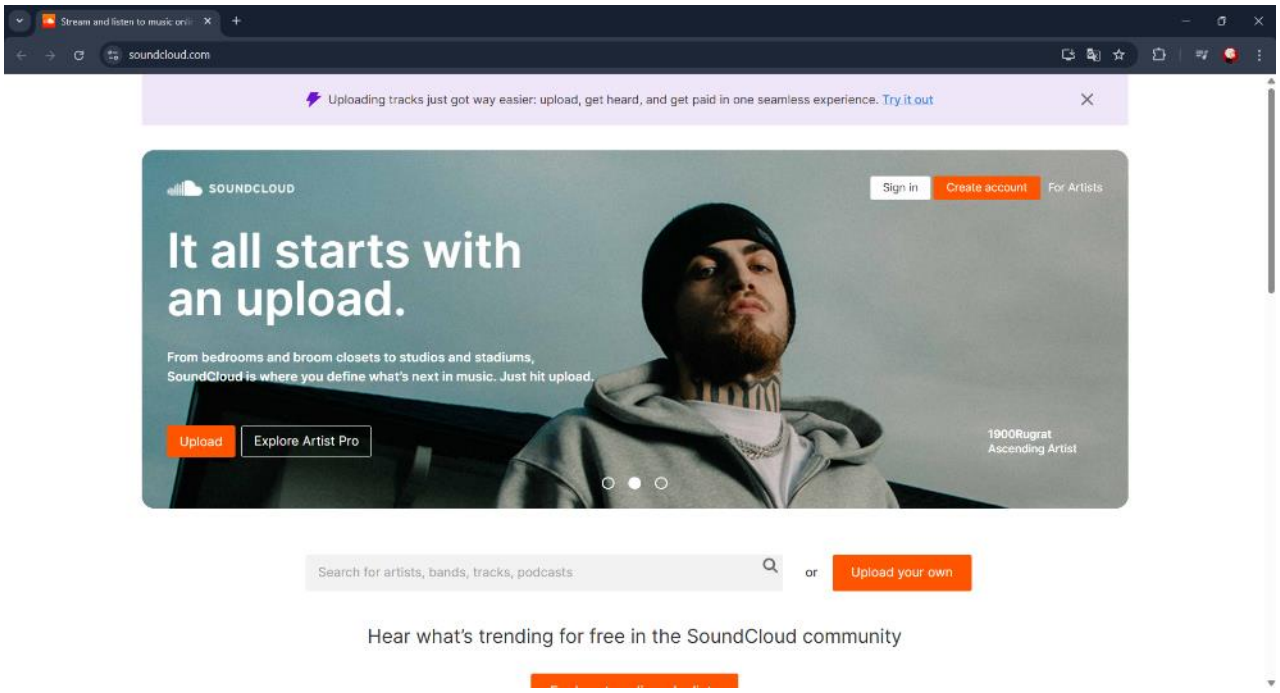

Рисунок 1.2 – Вебзастосунок «SoundCloud»

Джерело:[2]

YouTube Music (рис. 1.3) – платформа, яка поєднує переваги звичайного стрімінгу з гнучкістю відеоплатформи YouTube[3].

Переваги:

- величезна база треків через інтеграцію з YouTube;
- доступ до рідкісних записів і лайвів;
- легко знайти кавери або неофіційні версії.

Недоліки:

- незручний інтерфейс для чистого музичного прослуховування;

- багато відео не є музикою;
- мало фільтрів;
- багато реклами без підписки.

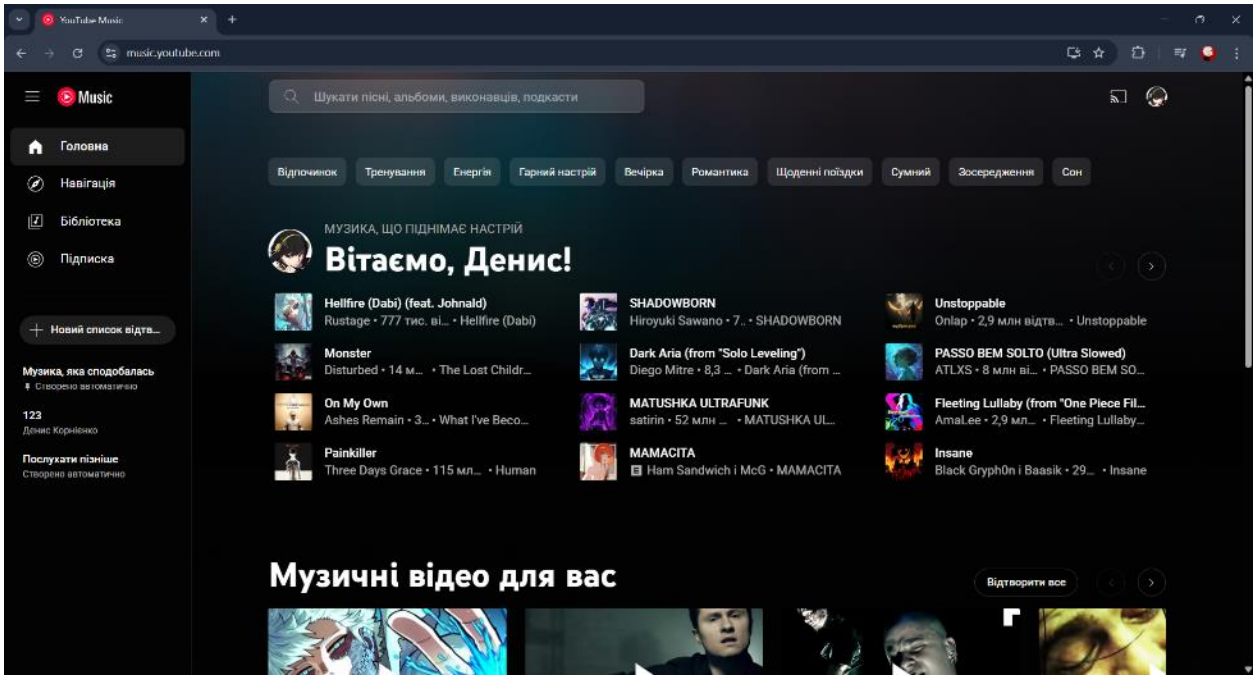

Рисунок 1.3 – Вебзастосунок «YouTube Music»

Джерело:[3]

Apple Music (рис 1.4) – це сервіс для стрімінгу музики, який особливо зручний для користувачів продуктів Apple[4].

Переваги:

- висока якість звуку;
- глибока інтеграція з IOS;
- гарна добірка музики за жанрами;
- професійні рекомендації від кураторів.

Недоліки:

- найкраще працює тільки в екосистемі Apple;
- базовий пошук і фільтрація;
- платна модель без безкоштовної версії;
- інтерфейс не завжди зручний.

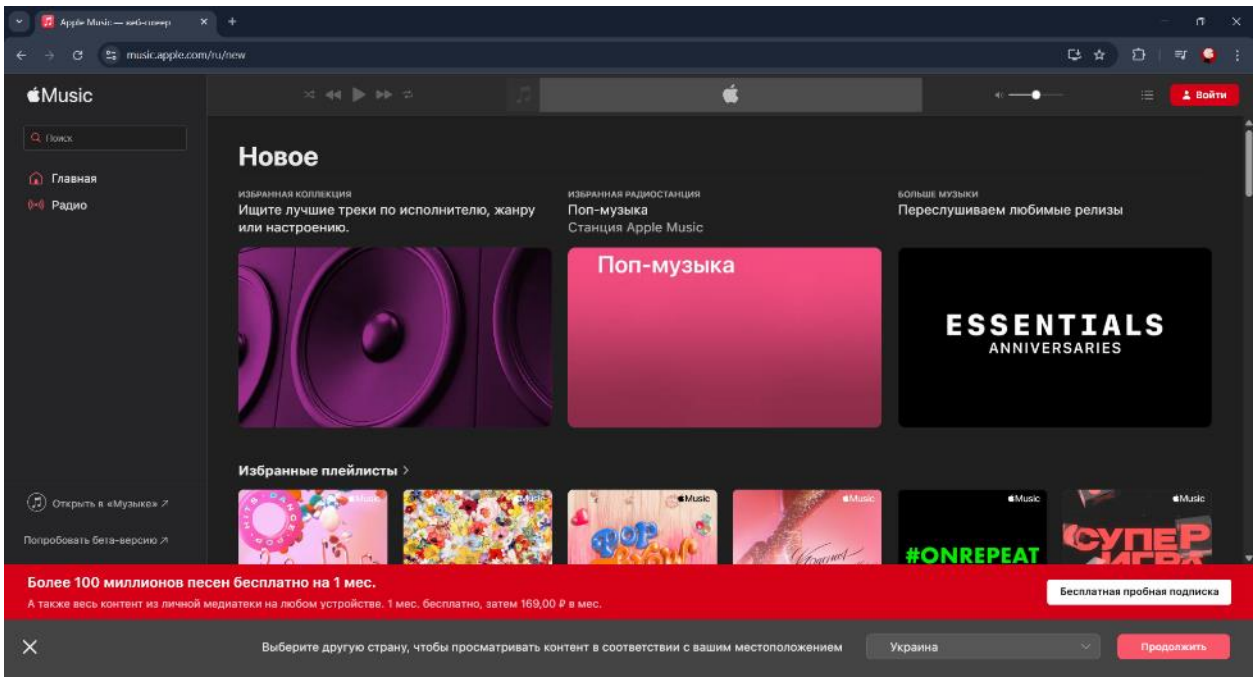

Рисунок 1.4 – Вебзастосунок «Apple Music»

Джерело:[4]

1.3 Визначення потенційних конкурентних переваг розробки

Потенційні конкурентні переваги розробки веб-застосунку для прослуховування аудіо файлів можуть бути:

1. Сучасні технології веб-розробки. Завдяки використанню передових інструментів та фреймворків, сайт забезпечує швидку роботу, адаптивність та зручний інтерфейс для користувачів на будь-яких пристроях.
2. Багатокритеріальний фільтр. Платформа дозволяє шукати й сортувати аудіо за кількома параметрами одночасно (жанр, виконавець, дата створення аудіо), що значно покращує зручність пошуку.
3. Фокус на зручності та персоналізації. Платформа орієнтована на комфортне прослуховування аудіо з можливістю підлаштування під індивідуальні вподобання користувача.

1.4 Постановка задачі

Постановка завдання передбачає врахування ряду важливих аспектів. Основна мета – створення сучасного веб-застосунку для прослуховування

музики, який буде інтуїтивно зрозумілим, функціональним та адаптованим до потреб кінцевого користувача. Проект має забезпечити зручний інтерфейс для взаємодії з аудіо контентом, розширені можливості пошуку та рекомендацій, а також стабільну роботу на клієнтському та серверному рівнях.

У межах цього проекту визначено такі основні задачі:

- розробка основного функціоналу аудіо програвача, зокрема:
  - відтворення треків у зручному інтерфейсі;
  - можливість ставити на паузу, перемотувати та регулювати гучність;
  - перемикання візуального відображення треків;
  - реалізація світлої та темної теми оформлення.
- реалізація системи пошуку та фільтрації:
  - пошук треків за назвою, автором та жанром;
  - фільтрація за кількома критеріями одночасно;
  - оптимістичне завантаження даних з використанням пагінації.
- розробка та інтеграція системи взаємодії користувачів із контентом, а саме:
  - можливість ставити лайки трекам;
  - додавання коментарів;
- створення простої рекомендаційної системи, яка формує перелік популярних треків на основі сукупної кількості лайків та коментарів;
- забезпечення якісної взаємодії між фронтендом і бекендом за допомогою REST API, створення контролерів для обробки запитів, а також впровадження кешування та ефективного завантаження даних на клієнтську частину.

Висновки до розділу 1

Було проведено дослідження предметної області, пов’язаної з веб-застосунком для аудіостримінгу, як сучасного інструменту мультимедійного контенту та цифрової розваги.

Здійснено аналіз існуючих платформ із подібною функціональністю, виявлено їхні ключові можливості, переваги та недоліки. Це дозволило визначити актуальні потреби користувачів і сформувати бачення конкурентних переваг майбутнього продукту.

Використання автоматизованих функцій у застосунку, таких як фільтрація треків за кількома критеріями, персоналізовані рекомендації, динамічне оновлення інформації про треки, інтерактивні елементи плеєра – сприяє підвищенню зручності використання платформи. Це важливо для забезпечення якісного користувацького досвіду, зростання зацікавленості аудиторії та формування лояльності.

Крім того, така функціональність дозволяє ефективніше організувати робочі процеси в рамках проєкту, спрощує підтримку та розширення платформи, а також підвищує її конкурентоспроможність серед інших рішень на ринку.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОФАЙЛІВ

2.1 Проєктування структури

Розробка веб-додатку для прослуховування аудіо файлів потребує аналізу варіантів використання та висування функціональних та нефункціональних вимог до програмного продукту.

Функціональні вимоги визначають, яку саме функціональність має реалізовувати вебзастосунок:

- надання користувачу можливості прослуховування треків;
- пошук треків за фільтрами: автор, назва, жанр;
- можливість перемикання між світлою та темною темою;
- вибір вигляду відображення треків (табличний / повноекранний);
- додавання лайків до треків;
- написання коментарів до треків;
- формування рекомендаційної системи на основі популярності треків (лайки + коментарі);
- динамічне оновлення контенту без перезавантаження сторінки;
- адаптивність інтерфейсу до різних розмірів екранів.

Нефункціональні вимоги, які відносяться до властивостей та характеристик системи, що не пов’язані безпосередньо з функціональністю:

- система повинна швидко реагувати на дії користувача, не перевищуючи 1 секунду очікування при завантаженні списку треків;
- архітектура повинна підтримувати розширення функціоналу в майбутньому без необхідності повного переписування системи;
- інтерфейс має бути інтуїтивно зрозумілим, з чіткими візуальними підказками;

- система має коректно обробляти помилки;
- застосунок повинен коректно відображатися в основних браузерах;
- підтримка мобільних пристроїв з різними розмірами екрану.

Для опису функціональності проекту також використаємо діаграму прецедентів.

Діаграма прецедентів (Use Case діаграма) – це тип діаграми, що використовується в процесі моделювання систем для зображення функціональних можливостей системи з точки зору її користувачів. Основна мета такої діаграми полягає у визначенні та наочному представленні функціональних вимог системи, а також взаємодії між користувачами (акторами) і самою системою (рис. 2.1).

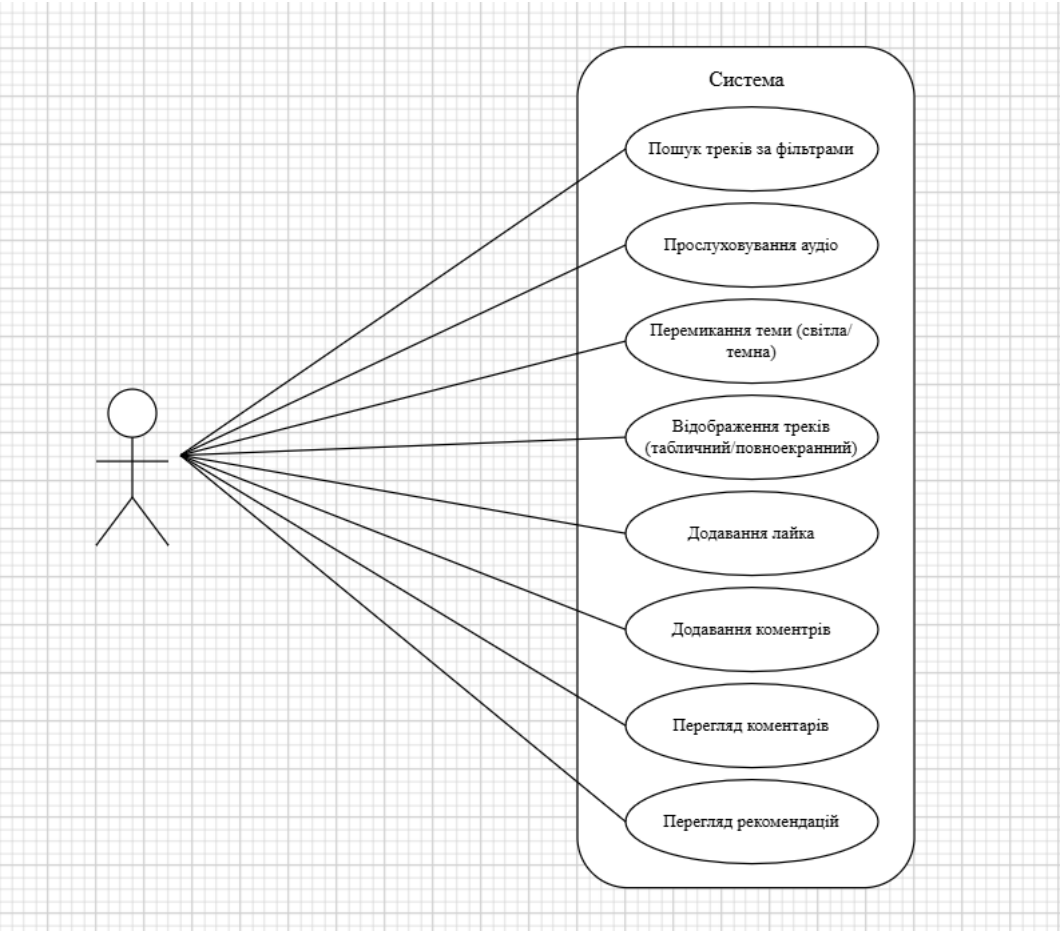

Рисунок 2.1 – Діаграма прецедентів

Джерело: розроблено автором

2.2 Моделювання даних

Для опису зберігання та використання даних в проекті використаємо діаграми класів та «сутність-зв’язок».

Діаграма класів – це тип діаграми, що застосовується в моделюванні програмного забезпечення для графічного представлення структури класів та взаємозв’язків між ними в системі. Її основне призначення – показати, які класи містить програма, їхні атрибути і методи, а також способи взаємодії між цими класами (рис. 2.2).

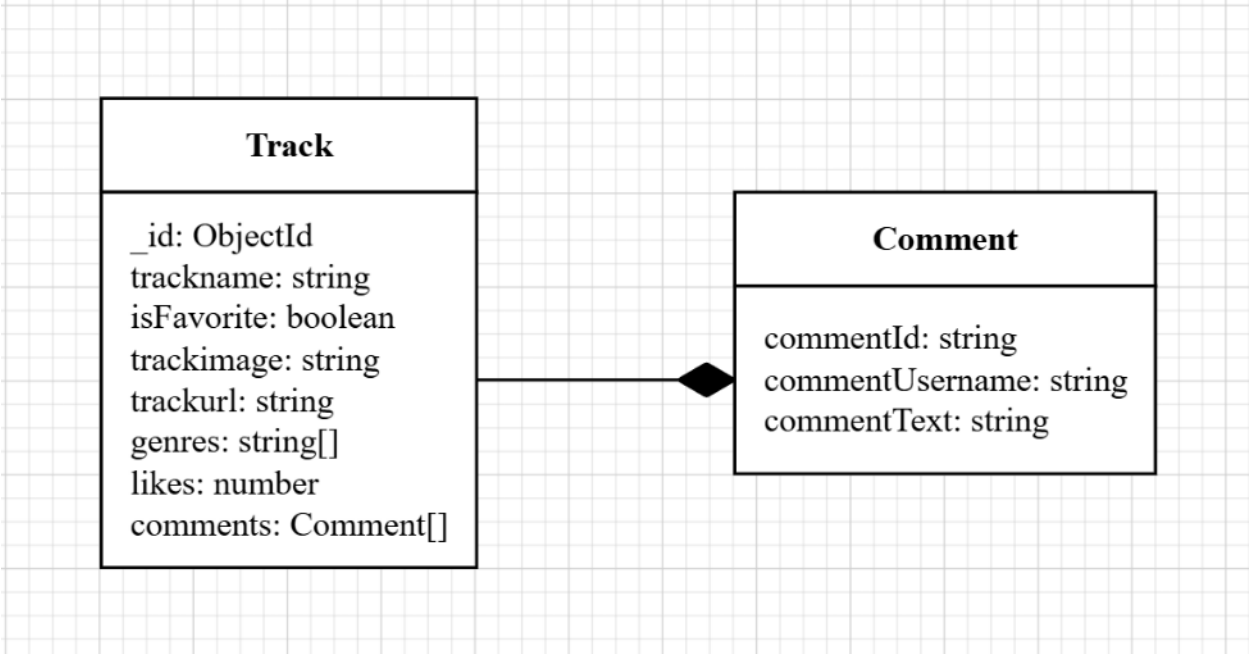

Рисунок 2.2 – Діаграма класів

Джерело: розроблено автором

«Сутність-зв’язок» діаграма (Entity-Relationship diagram) – це тип діаграми, що використовується в базах даних для візуалізації структури даних та взаємозв’язків між різними сутностями. Вона допомагає розуміти структуру бази даних, описуючи, які сутності існують, як вони пов’язані між собою та які атрибути вони мають (рис. 2.3).

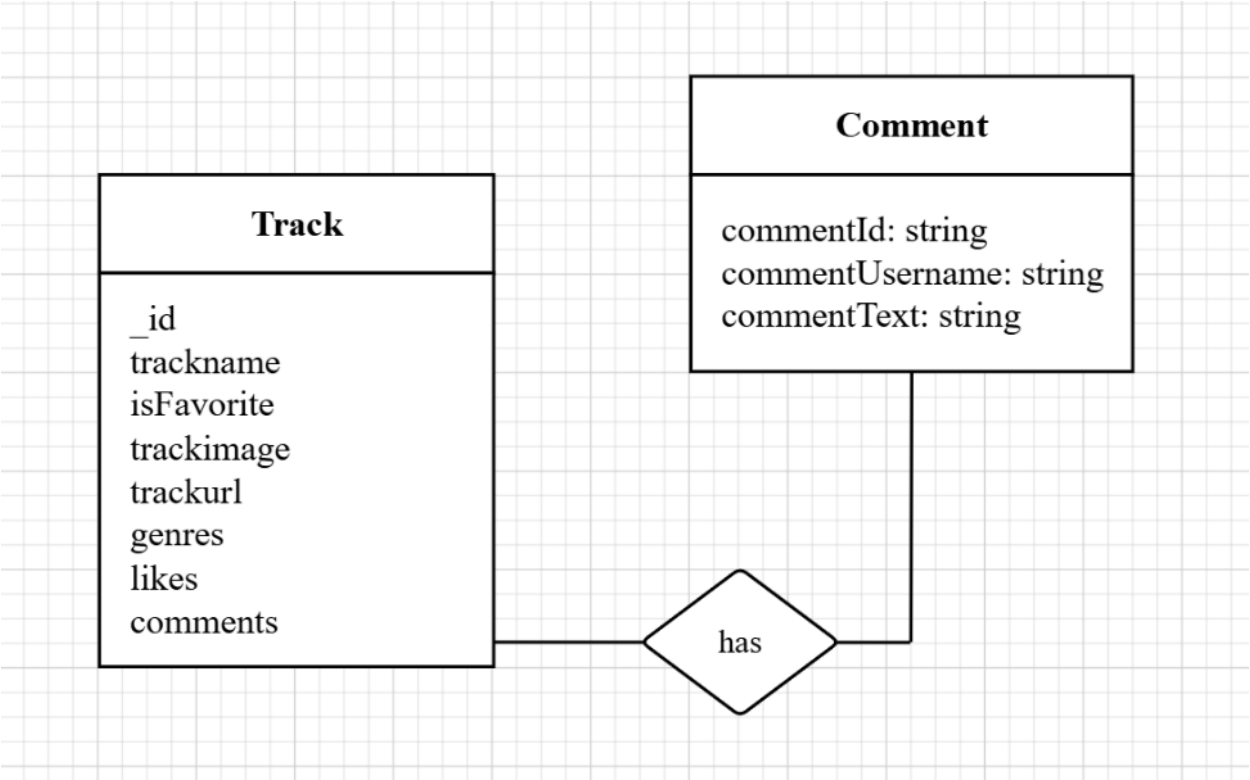

Рисунок 2.3 – Entity-Relationship діаграма

Джерело: розроблено автором

2.3 Моделювання процесів

У процесі розробки веб-застосунку було виконано моделювання даних на основі функціональних вимог до системи. Основна мета моделювання – формалізувати структуру зберігання інформації у базі даних MongoDB, а також логіку взаємодії користувача з даними через інтерфейс.

Була розроблена ER-діаграма, яка демонструє зв’язки між сутностями. Основною сутністю системи є треки, кожен з яких може мати кілька коментарів. Сутність коментар є вкладеною в об’єкт треку.

Ключові концепції моделі:

- уся інформація про треки зберігається в одному документі, що дозволяє ефективно обробляти запити до MongoDB;
- коментарі реалізовані як вбудований масив документів у полі comments;

- для реалізації пошуку за кількома критеріями використовуються комбіновані фільтри за полями trackName, trackAuthor, genres.
- рекомендаційна система будується на базі агрегованих полів likes + кількість comments.

Діаграма активності – це тип діаграми в області моделювання, який використовується для візуалізації послідовності дій або процесів у системі. Вона дозволяє ілюструвати різноманітні сценарії виконання взаємодії між об’єктами або учасниками в системі, відображаючи порядок виконання окремих операцій, рішень та умовних переходів.

У випадку системи прослуховування аудіо файлів, ця діаграма показує порядок взаємодії користувачів з веб-застосунком (рис. 2.4).

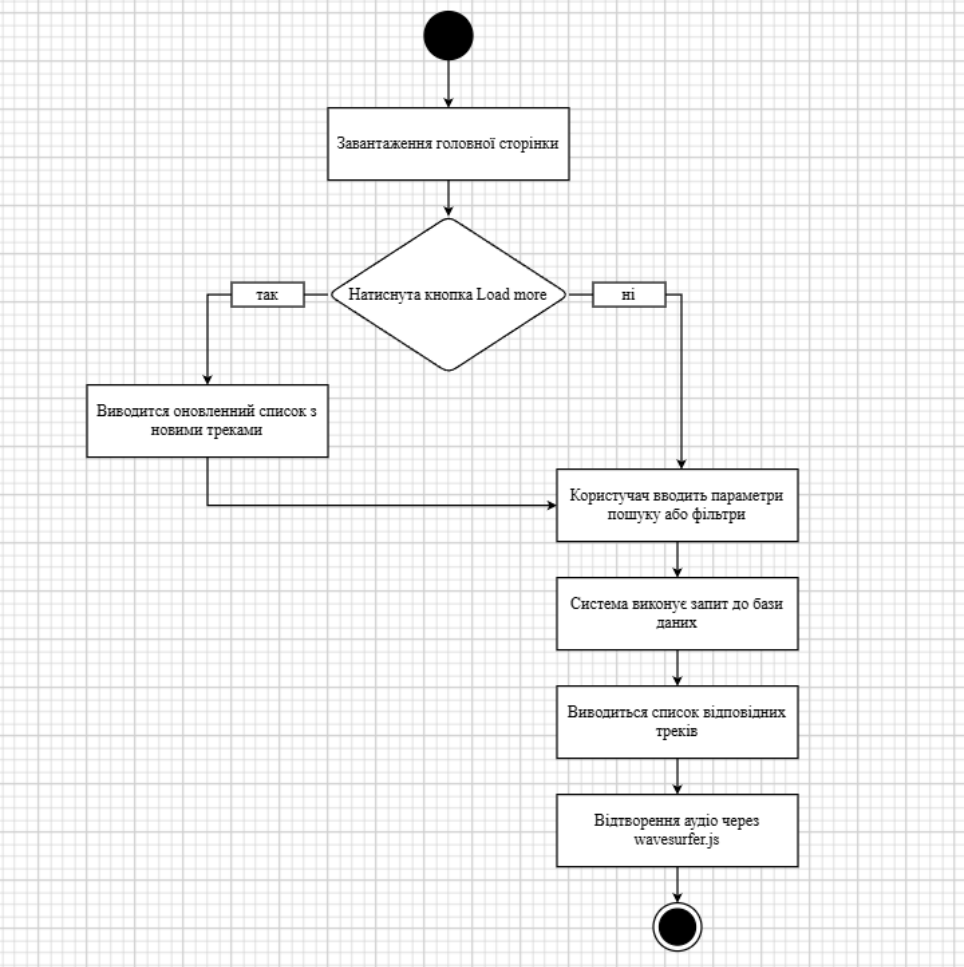

Рисунок 2.4 – Діаграма активності

Джерела: розроблено автором

Висновки до розділу 2

У другому розділі було проведено аналіз функціональних та нефункціональних вимог до веб-застосунку для прослуховування аудіо файлів та виконано його структурне проектування. Було визначено основні функціональні модулі системи, такі як пошук і фільтрація треків, відтворення аудіо, взаємодія з треками (лайки, коментарі), а також реалізація рекомендаційної системи.

Було виконано моделювання сутностей бази даних за допомогою ER-діаграми, що дозволило візуалізувати логіку збереження та зв’язків між об’єктами. Також розроблено діаграму активності, яка ілюструє логіку взаємодії користувача з веб-застосунком – від завантаження сторінки до відтворення музики та взаємодії з треками.

Запропонована архітектура враховує ефективність обробки даних, масштабність та зручність для кінцевого користувача. Отримані результати створюють надійну основу для реалізації клієнтської та серверної частини застосунку у наступних етапах розробки.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІО ФАЙЛІВ

3.1 Особливості реалізації

3.1.1 Засоби розробки

Для реалізації веб-застосунку було використано сучасні інструменти та бібліотеки, які забезпечують масштабованість, підтримку, продуктивність і комфорт.

Frontend (Клієнтська частина):

- React – JavaScript-бібліотека для побудови інтерфейсів. Забезпечує компонентну архітектуру, що полегшує повторне використання коду та тестування[5];
- TypeScript – надмножина JavaScript, яка вводить статичну типізацію. Дозволяє уникати багатьох помилок ще на етапі написання коду[9];
- TailwindCSS – утилітарна CSS-бібліотека для швидкої та адаптивної верстки. Спрощує підтримку дизайну та не потребує написання класичних CSS-файлів[10];
- React-Query – бібліотека для роботи з API-запитами. Відповідає за кешування, повторні запити, статуси завантаження, що значно полегшує управління асинхронними операціями[6];
- WaveSurfer.js – бібліотека для побудови аудіо хвиль та контролю над програванням. Застосовується для відтворення треків з візуалізацією[11];
- React-Select – UI-бібліотека для створення зручних і настроюваних випадаючих списків, що використовуються у фільтрації[7];
- React-Icons – набір іконок для покращення інтерфейсу[8];

- Sonner – бібліотека для показу повідомлень та повідомлень про помилки (toasts)[12];
- Framer Motion – сучасна бібліотека для створення анімацій компонентів у React. Додає плавність і динамічність до UI[13].

Backend (Серверна частина):

- Node.js – середовище виконання JavaScript на стороні сервера. Висока продуктивність та асинхронність виконання[16];
- Express.js – мінімалістичний фреймворк для побудови RESTful API. Забезпечує швидку розробку та гнучку маршрутизацію[17];
- Mongoose – ODM (Object Data Modeling) для MongoDB. Дає змогу визначати схеми та працювати з документами бази даних у зручному форматі[15]

Database (База даних):

- MongoDB – документо-орієнтована база даних NoSQL. Ідеально підходить для зберігання складних об’єктів, таких як треки з коментарями[14];
- Cloudinary – сервіс для хостингу мультимедійних файлів (зображень та аудіо), що дозволяє зберігати URL-адреси без завантаження великих обсягів даних на власний сервер[18].

Інструменти розробника:

- Visual Studio Code – основне середовище розробки;
- Postman – для тестування API[19];
- ESLint + Prettier – для підтримки єдиного стилю коду та попередження синтаксичних помилок.

3.1.2 Опис архітектури програмного забезпечення

Архітектура застосунку має клієнт-серверну модель:

- Клієнт (Frontend):
    - відповідає за відображення інтерфейсу, пошук, фільтрацію, програвання треків, лайків, коментарів, перемикання теми;

- всі дії користувача обробляються за допомогою компонентів React та передаються через API на сервер.

- Сервер (Backend):
  - реалізовано на Express.js
  - обробляє запити REST API: отримання списку треків з пагінацією, додавання лайків, збереження коментарів, тощо;
  - підключення до MongoDB здійснюється через бібліотеку Mongoose.
- База даних (MongoDB):
  - зберігає документи з інформацією про треки, їх жанри, кількість лайків, коментарів та інші поля;

3.1.3 Реалізація сховища та опис доступу до даних

У якості бази даних для зберігання інформації використовується MongoDB – документо-орієнтована NoSQL-система, яка зберігає дані у вигляді JSON-подібних документів. Це дозволяє гнучко моделювати структуру об’єктів без жорсткого визначення схеми, що є зручним для динамічних веб-застосунків.

Основна колекція: audioBox

Кожен документ у колекції містить такі поля:

1. _id: (ObjectID) – унікальний ідентифікатор документа;
2. trackName: (String) – назва треку;
3. trackAuthor: (String) – автор треку або група;
4. isFavorite: (Boolean) – чи є трек улюбленим;
5. trackImage: (String) – посилання на зображення;
6. trackUrl: (String) – посилання на аудіо файл;
7. genres: (String[]) – список жанрів, до яких належить трек;
8. likes: (Number) – кількість лайків;
9. comments (Comment[]) – список коментарів, кожен з яких містить автора та текст.

Приклад одного документа продемонстровано на рис 3.1.

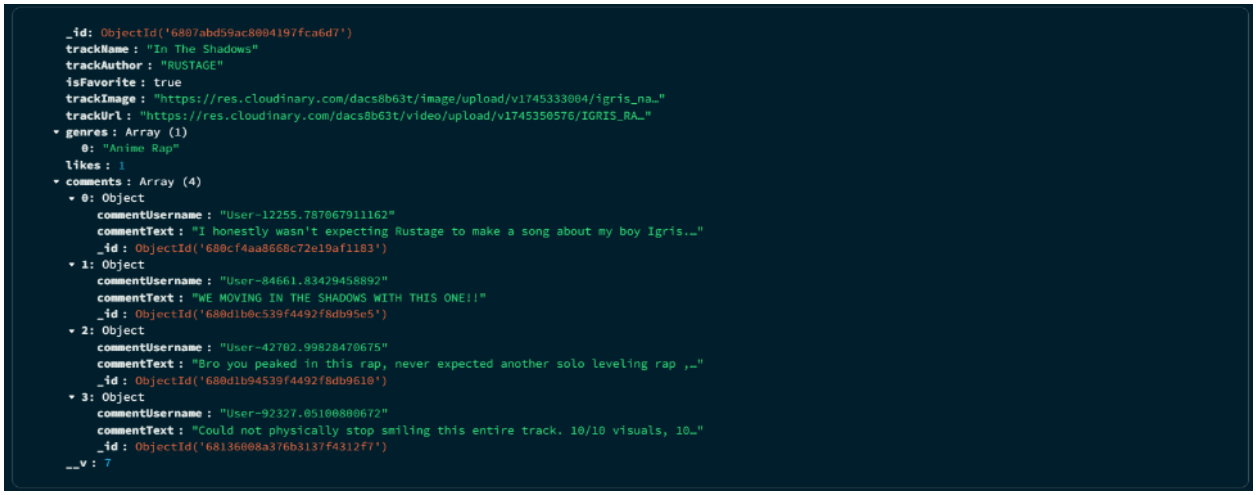

Рисунок 3.1 – Приклад даних треку

Джерело: розроблено автором

Модель Audio у середовищі Node.js реалізована за допомогою бібліотеки Mongoose (рис. 3.2):

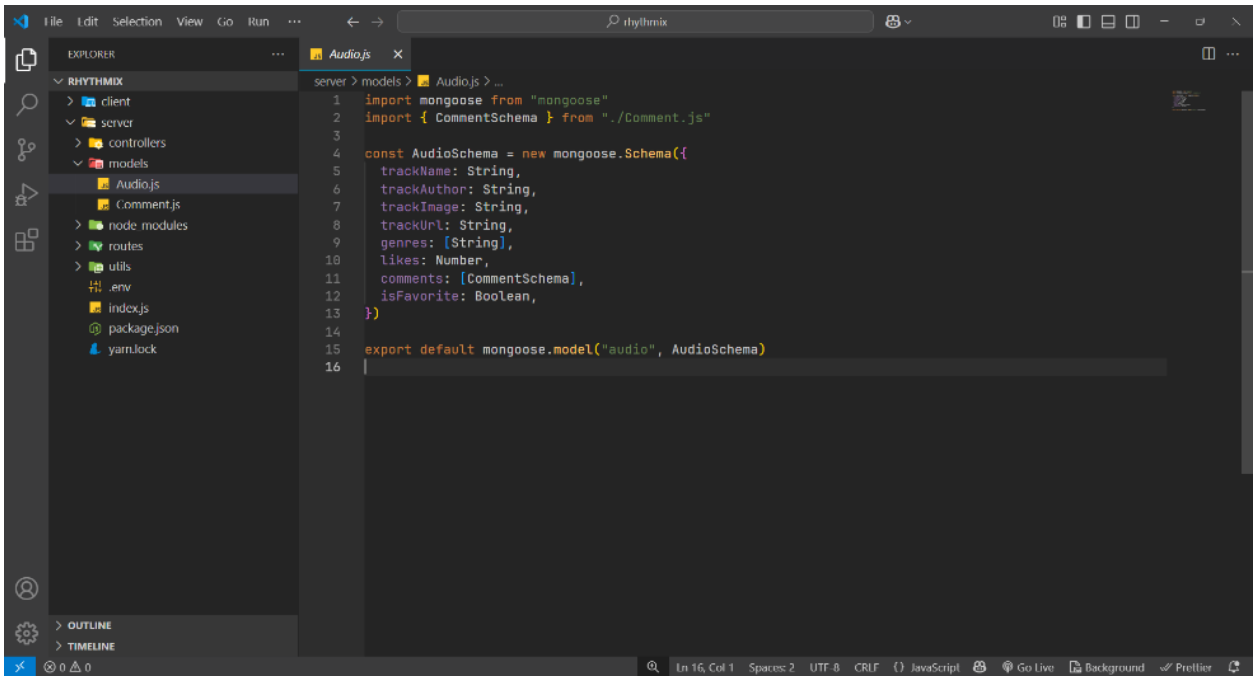

Рисунок 3.2 – Модель Audio

Джерело: розроблено автором

Модель Comment наведено на рис. 3.3.

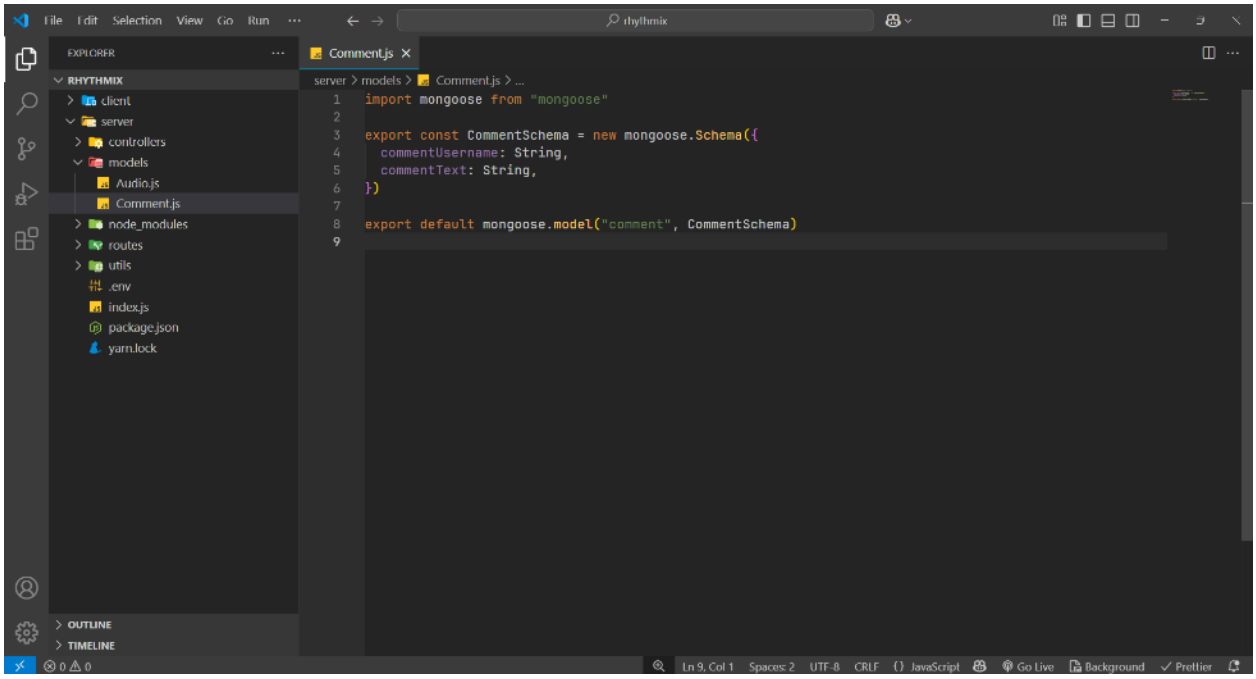

Рисунок 3.3 – Модель Comment

Джерело: розроблено автором

Доступ до даних реалізовано через API маршрути:

1. (GET) /tracks?offset=0&limit=10 – отримання треків із пагінацією;
2. (POST) /tracks/:trackId/like – лайк/анлайк;
3. (POST) /tracks/:trackId/comments – додавання коментаря;
4. (GET) /tracks/:trackId/comments – отриманню списку коментарів;
5. (GET) /tracks/:trackId/recommended – отримання рекомендованих треків.

3.2 Конструювання системи

Конструювання системи є критично важливою фазою, у якій визначається архітектура усієї системи. На рис 3.4 показано діаграму компонентів застосунку. Він складається з 3 основних частин: клієнтська, серверна та база даних.

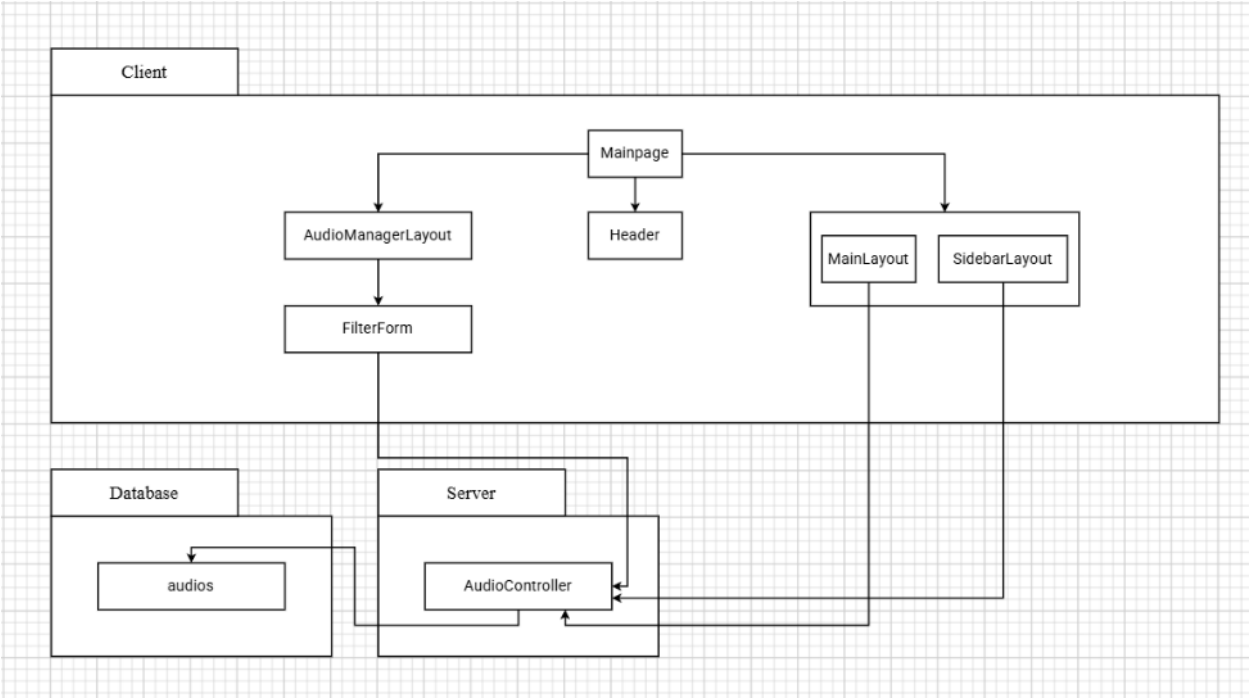

Рисунок 3.4 – Component Diagram

Джерело: розроблено автором

У клієнтській частині реалізовано кілька основних React-компонентів:

- Mainpage – головна сторінка застосунку, що ініціалізує завантаження і рендеринг основних частин інтерфейсу;
- Header – верхня панель, яка містить головний заголовок додатку;
- AudioManagerLayout – головний контейнер для компонента, що керує виведенням треків;
- FilterForm – форма, як дозволяє користувачу застосовувати фільтри до списку треків за автором, жанром або назвою;
- MainLayout та SidebarLayout – компоненти, які відповідають за розміщення та структуру основної області контенту та бокової панелі відповідно.

У серверній частині є такий компонент:

- AudioController – контролер, який обробляє HTTP-запити від клієнта:
    - отримання списку
    - пошук за фільтрами

- додавання лайків та коментарів
- отримання рекомендованих треків тощо

У базі даних є такий компонент:

- зберігається колекція audios, яка містить документи з інформацією про аудіо треки
- сервер взаємодіє з цією колекцією через контролер, виконуючи CRUD-операції.

3.3 Тестування системи

Тестування системи здійснювалося з метою перевірки правильності функціонування всіх основних компонентів веб-застосунку, виявлення можливих помилок та недоліків, а також забезпечення стабільної роботи системи при взаємодії з користувачем.

Були застосовані такі види тестування:

- Модульне тестування (Unit testing)
  - функції фільтрації аудіо треків за критеріями;
  - логіку зміни теми;
  - функції керування лайками;
  - генерацію коментарів з автоматичними іменами користувачів;
  - перевірку логіки пагінації;
- Функціональне тестування
  - пошук треків за автором/назвою/жанром;
  - перемикання вигляду треків;
  - взаємодія з фільтрами;
  - відображення рекомендованих треків;
  - роботи кнопки підгрузки нових 10-ти треків.
- UI/UX тестування
  - адаптивність інтерфейсу під різні розміри екрану;
  - читабельність у світлій і темній темі;

- інтуїтивність взаємодії з фільтрами.

- Тестування продуктивності
  - завантаження перших 10-ти треків відбувається швидко;
  - нові треки додаються без перезавантаження сторінки;
  - відсутність затримок при фільтрації навіть при великому обсязі даних.

3.4 Використання системи

Перший кроком використання системи – перехід за посиланням. Після переходу користувач потрапляє на головну сторінку (рис. 3.5).

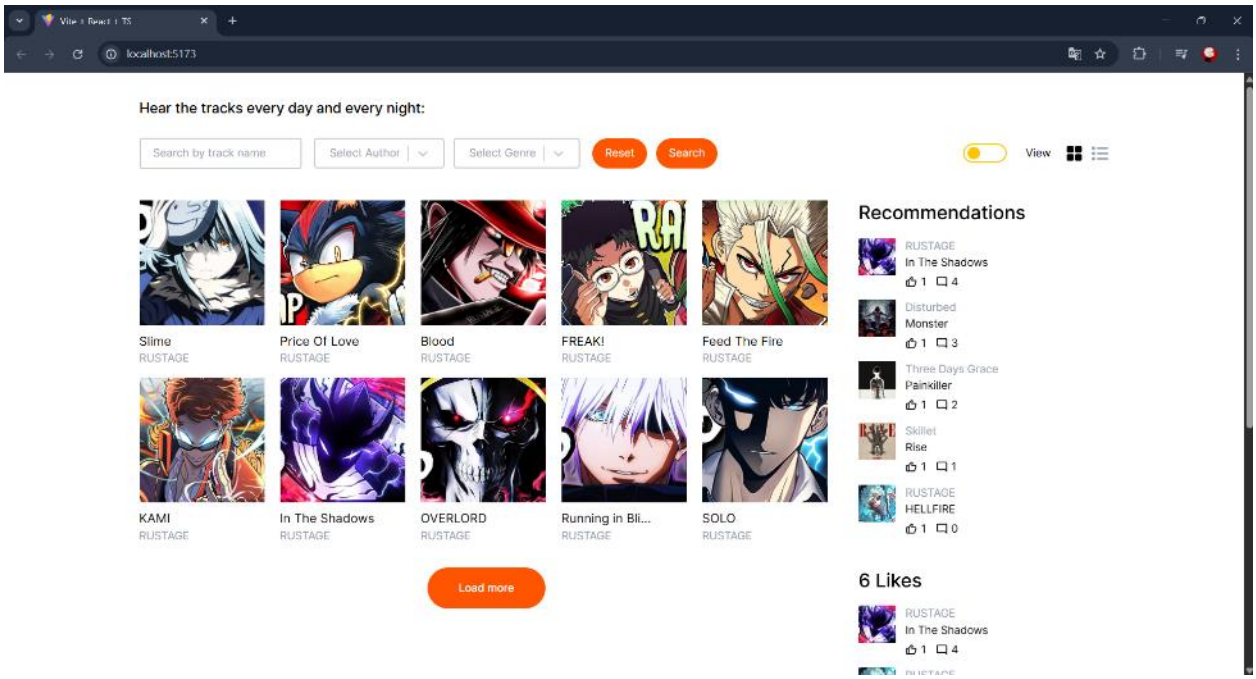

Рисунок 3.5 – Головна сторінка застосунку

Джерело: розроблено автором

На головній сторінці представлені такі компоненти:

- компонент Header, в якому записано головний заголовок застосунку;
- нижче від компонентом Header представлено компоненти управління аудіо треками. Перший компонент – форма фільтрації

треків по різним критеріям (назва, автор/група, жанр), другий компонент – управління зовнішній виглядом аудіо треків (табличний/повноекранний), третій компонент – перемикання теми (світла/темна).

- за цими компонентами йдуть головний контент застосунку, який ділиться на AudiosList та Recomendations і Favorites. Зліва у нас представлено список з перших 10 аудіо треків з можливість збільшення кількості при натисканні кнопки “Load more”, праворуч відображається рекомендаційна система та список улюблений треків в одній оболонці.

Тепер більш детально розберемо присутній функціонал. Спочатку глянемо, як застосунок виглядає в темній темі (рис. 3.6).

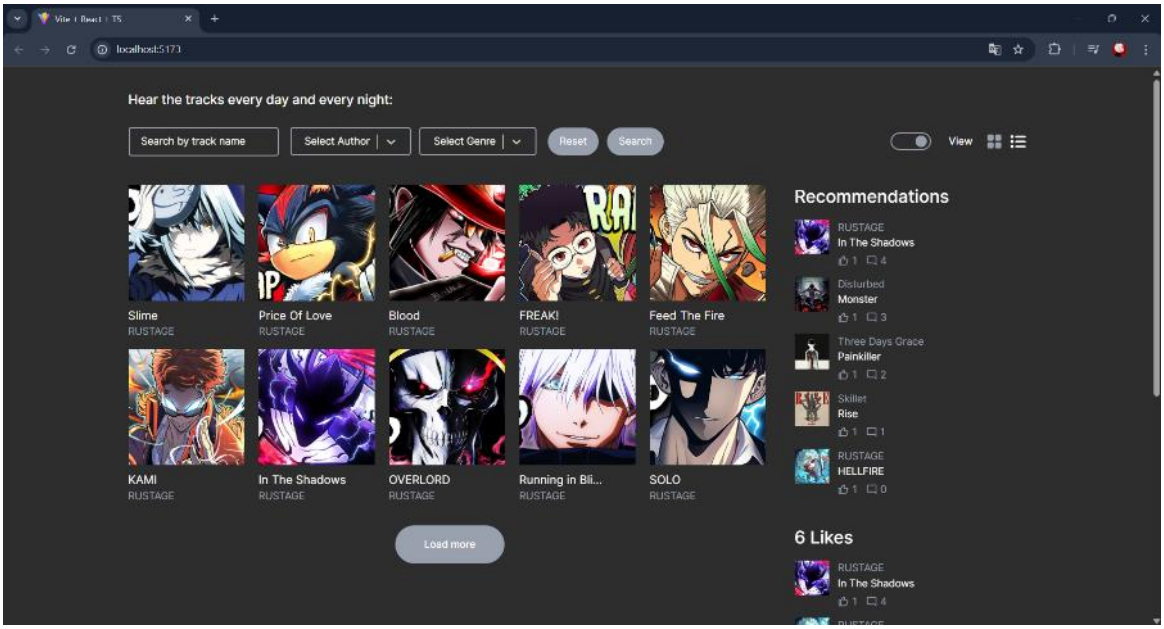

Рисунок 3.6 – Темна тема застосунку

Джерело: розроблено автором

Зміна з світлої теми на темну відбувається дуже швидко. Також, щоб кожного разу після перезавантаження сторінки тема залишалася, використовується локальне сховище для зберігання теми. Якщо ми закриємо додаток і знову відкриємо, тема залишиться темною.

Наступне, що ми подивимось це інший зовнішній вигляд аудіо треків (повноекранний). На рис. 3.7 представлено зовнішній вигляд повноекранних аудіо треків.

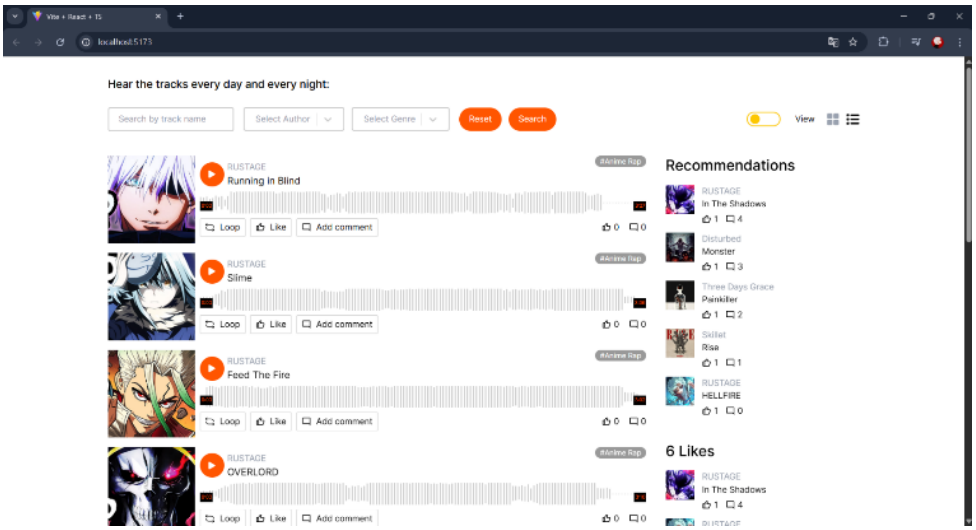

Рисунок 3.7 – Повноекранний вигляд аудіо треків

Джерело: розроблено автором

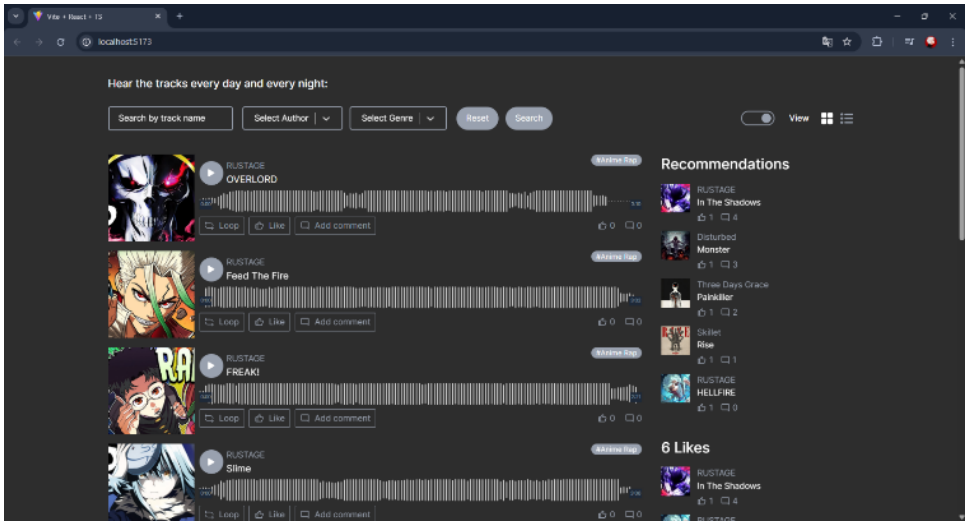

Рисунок 3.8 – Повноекранний вигляд аудіо треків в темній теми

Джерело: розроблено автором

На рис 3.8 також було продемонстровано зовнішній вигляд повноекранних аудіо треків в темній темі.

Тепер глянемо, як працює підгрузка ще 10-ти нових треків. На рис 3.9 буде зображено список аудіо треків до підгрузки 10-ти нових треків, на рис 3.10 буде зображено список з 10-ти перших треків та 10-ти нових.

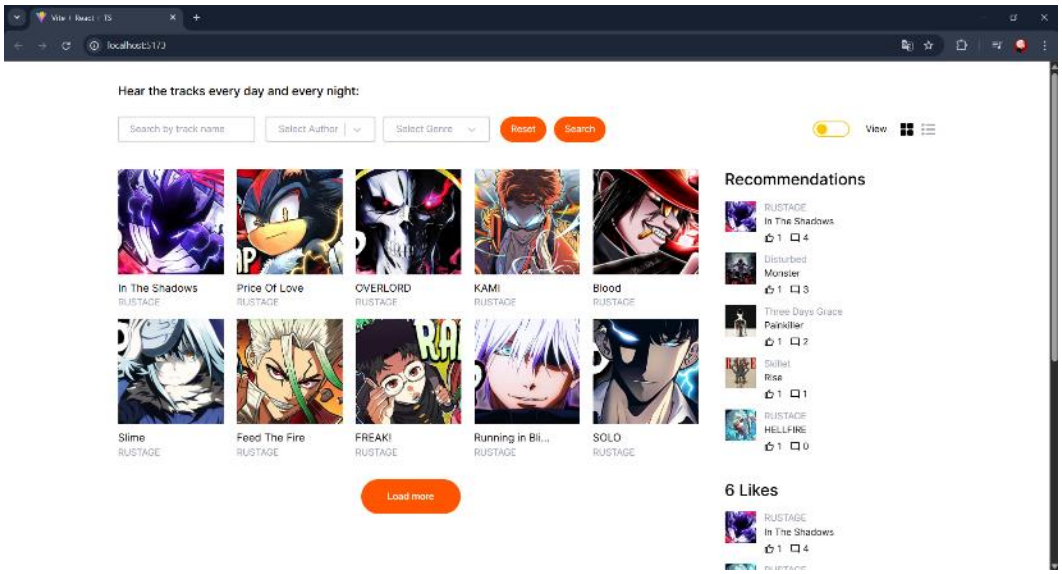

Рисунок 3.9 – Список аудіо треків до підгрузки

Джерело: розроблено автором

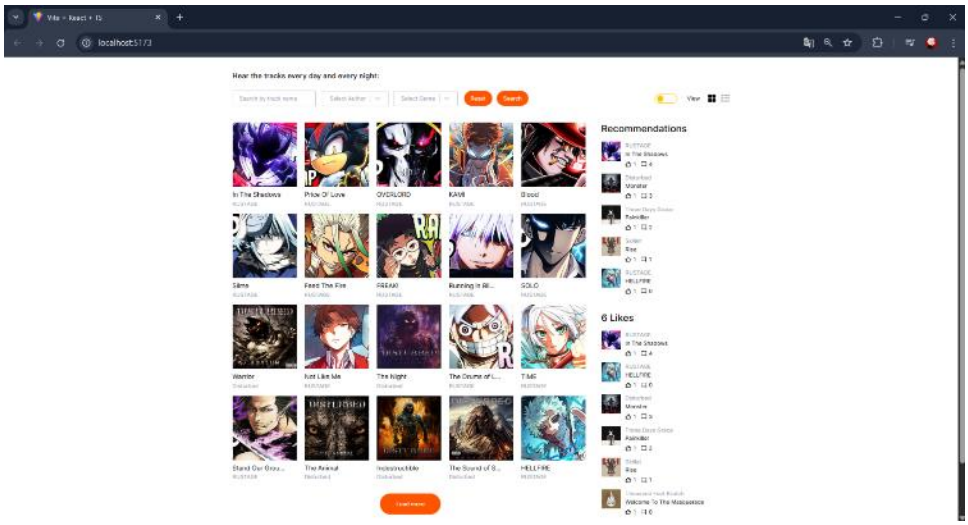

Рисунок 3.10 – Список аудіо треків після підгрузки

Джерело: розроблено автором

Тепер розглянемо аудіо плеєр. Коли ми виберемо трек, внизу зв’явиться аудіо плеєр. На рис 3.11 буде продемонстровано його зовнішній вигляд.

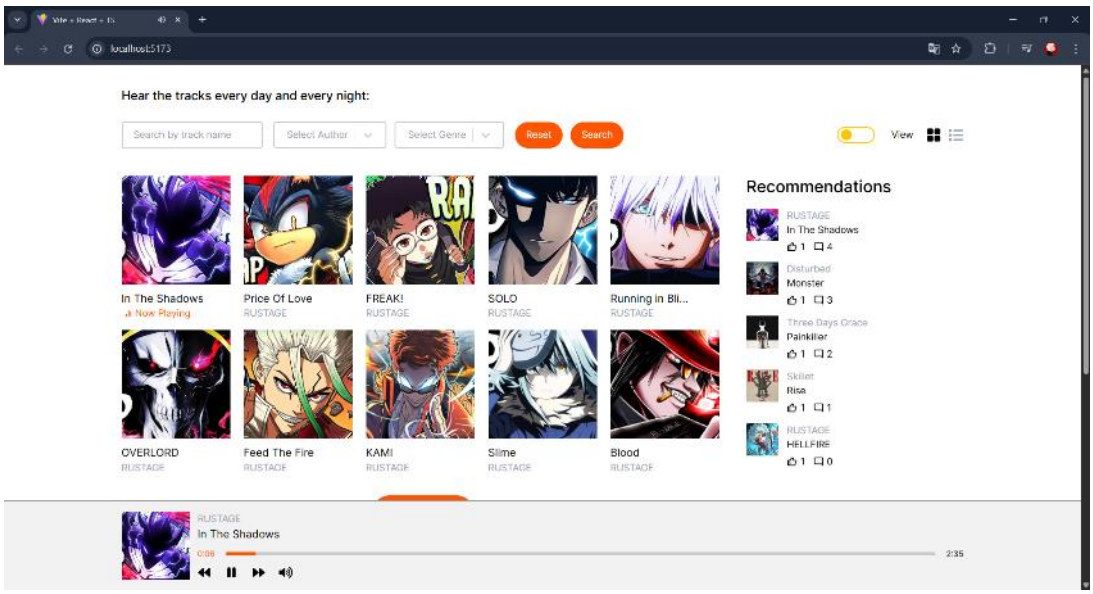

Рисунок 3.11 – Аудіо плеєр

Джерело: розроблено автором

На рис. 3.12 показано як виглядає аудіо плеєр коли встановлена темна тема.

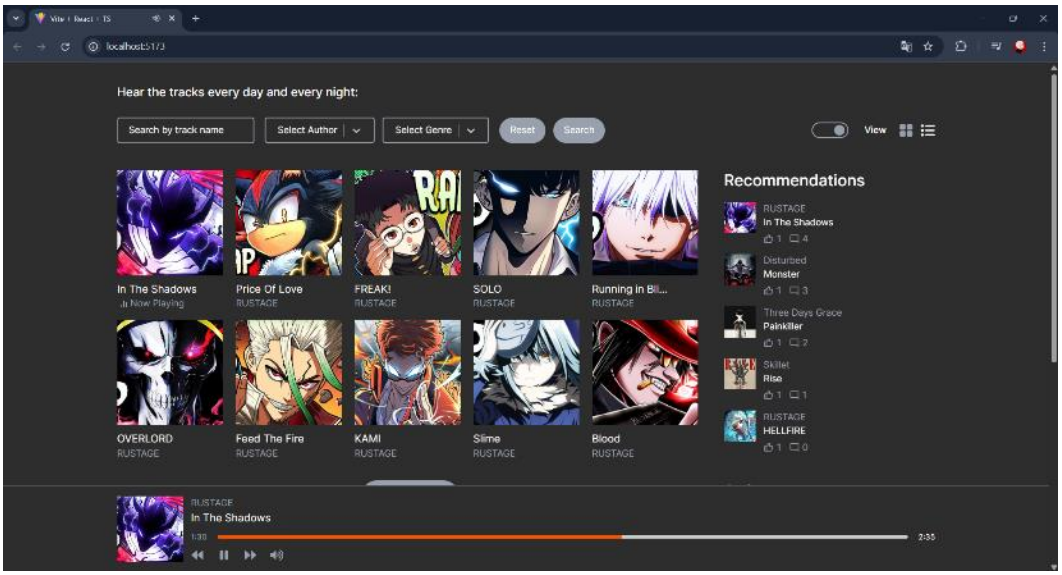

Рисунок 3.12 – Аудіо плеєр (темна тема)

Джерело: розроблено автором

На рис 3.13 відображаються список всіх коментарів аудіо

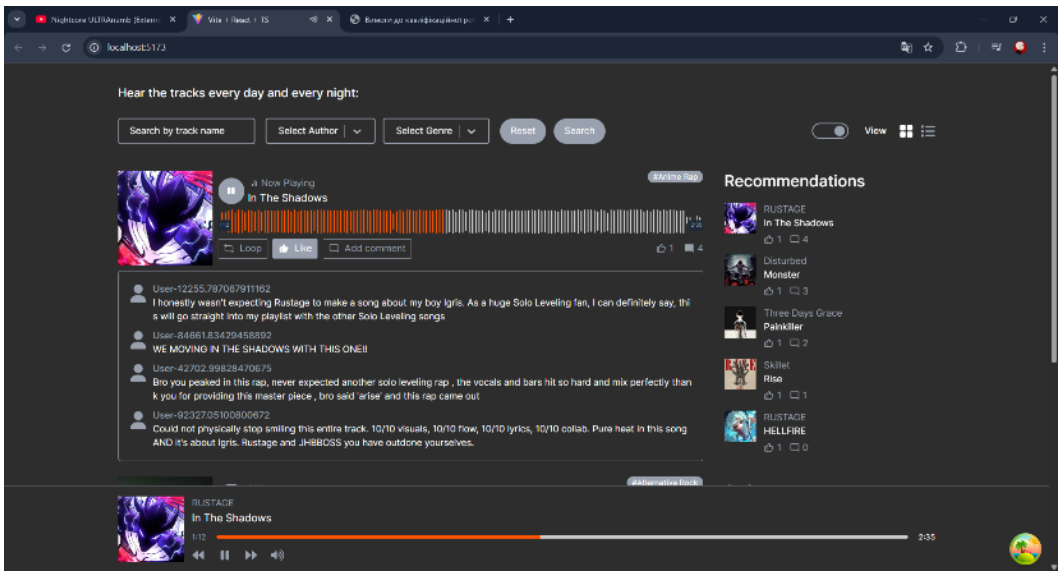

Рисунок 3.13 – відображення коментарів

Джерело: розроблено автором

Тепер глянемо пошук та фільтрацію. На рис 3.14 - 3.21 зображено відображення списку треків в залежності від вибраних користувачем критерій

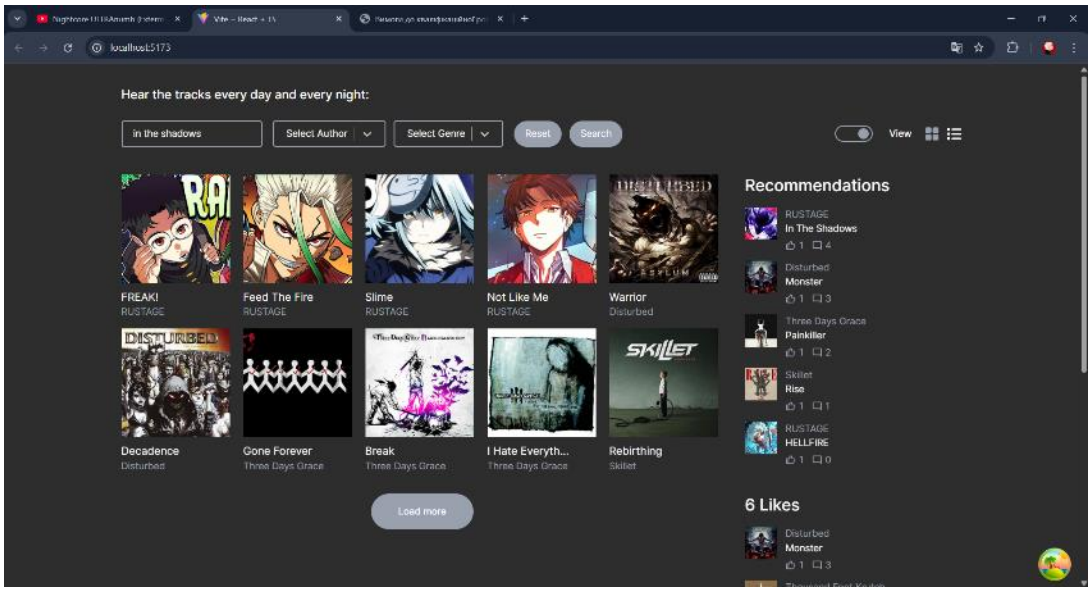

Рисунок 3.14 – До пошуку аудіо по назві

Джерело: розроблено автором

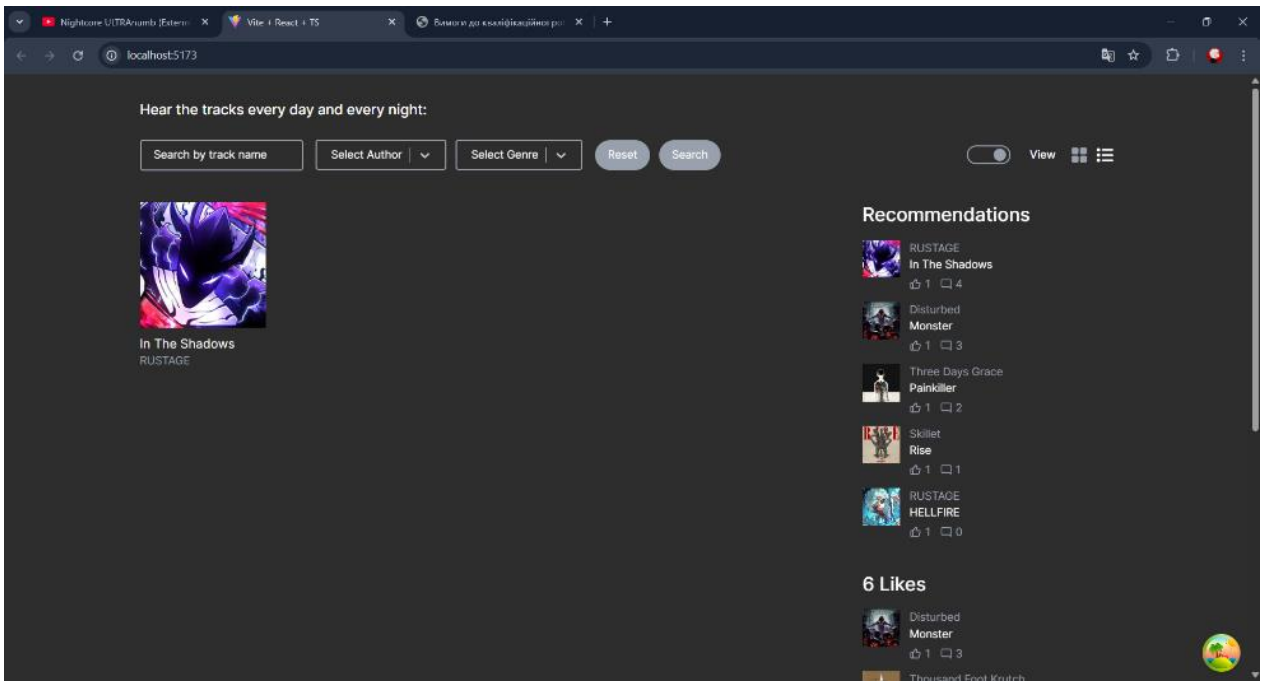

Рисунок 3.15 – Відображення знайденого аудіо по назві

Джерело: розроблено автором

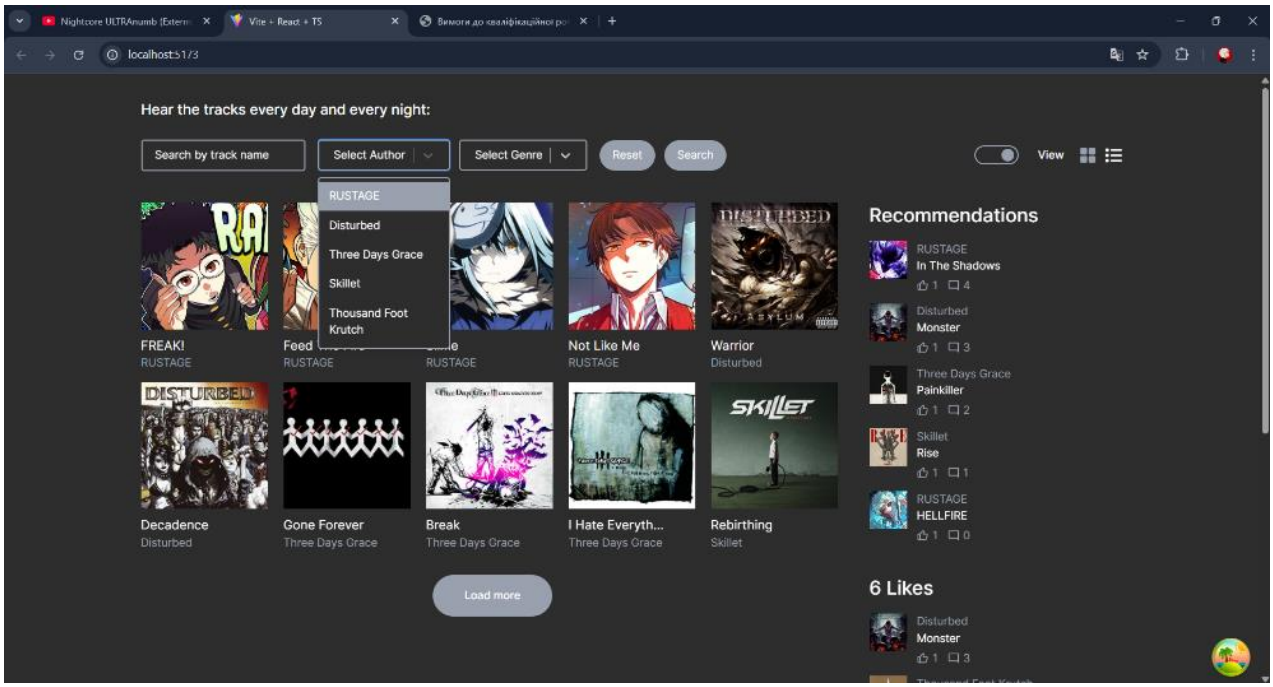

Рисунок 3.16 – До пошуку по автору/групі

Джерело: розроблено автором

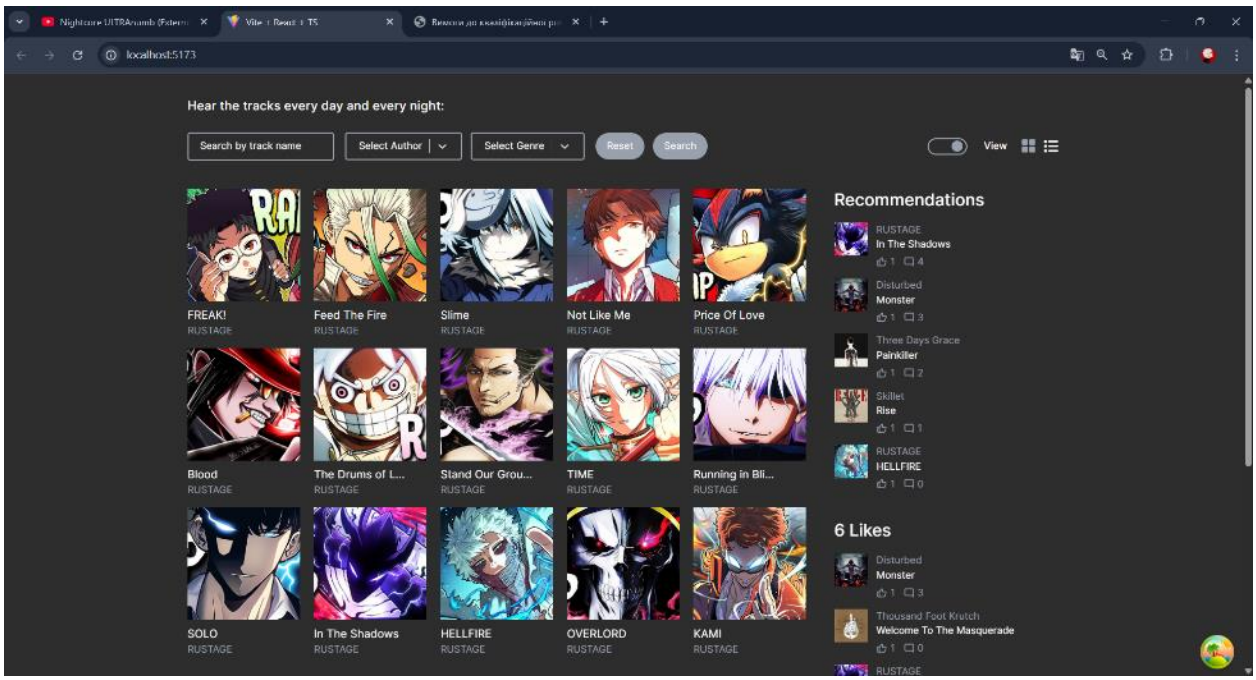

Рисунок 3.17 – Відображення знайдених треків по автору/групі

Джерело: розроблено автором

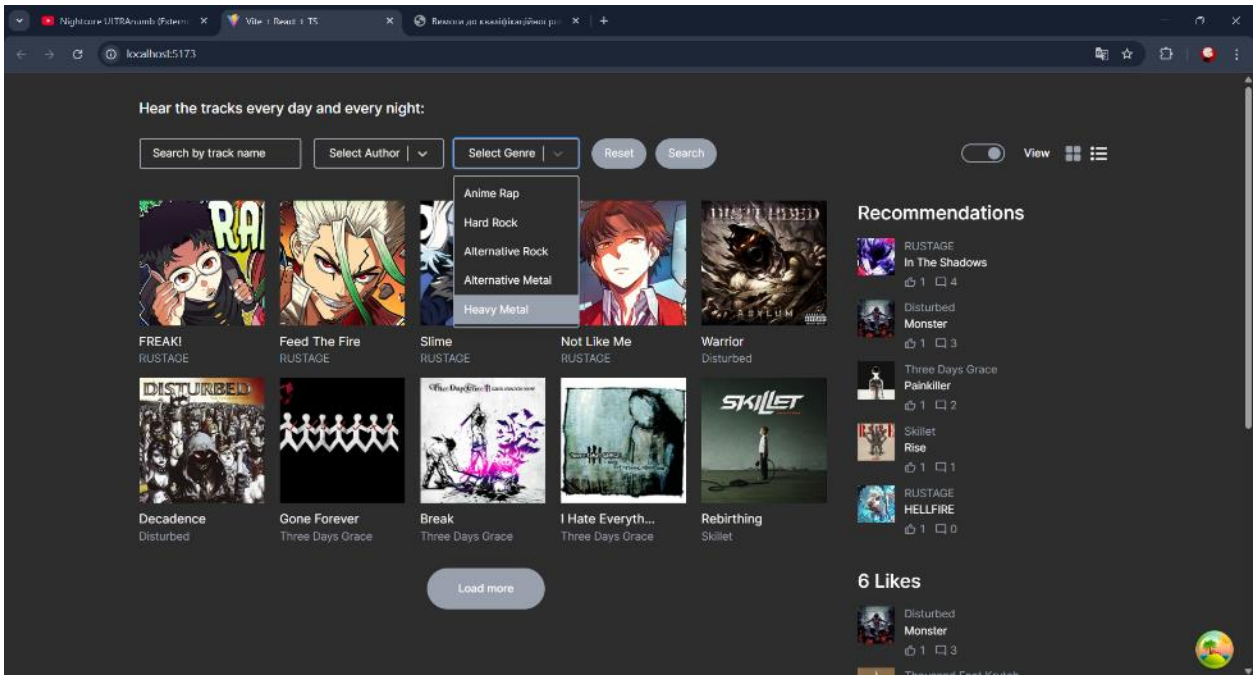

Рисунок 3.18 – До пошуку за жанром

Джерело: розроблено автором

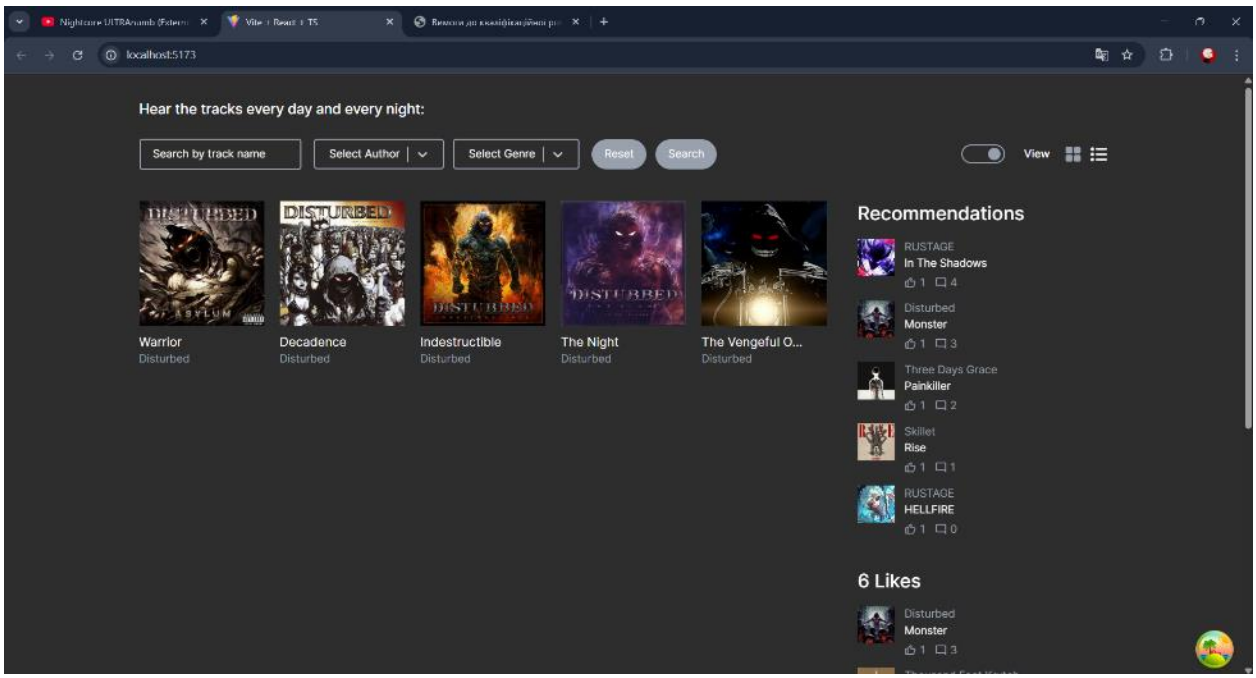

Рисунок 3.19 – Відображення знайдених треків по жанру

Джерело: розроблено автором

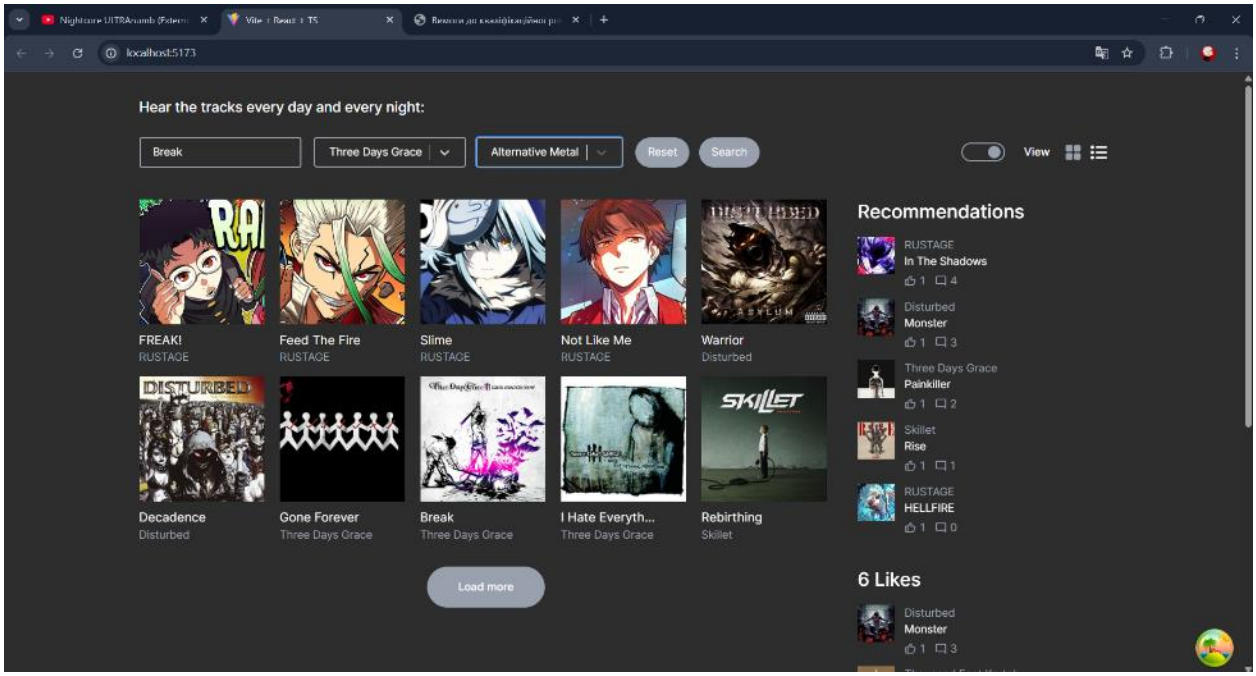

Рисунок 3.20 – До пошуку одночасно всіх критеріях

Джерело: розроблено автором

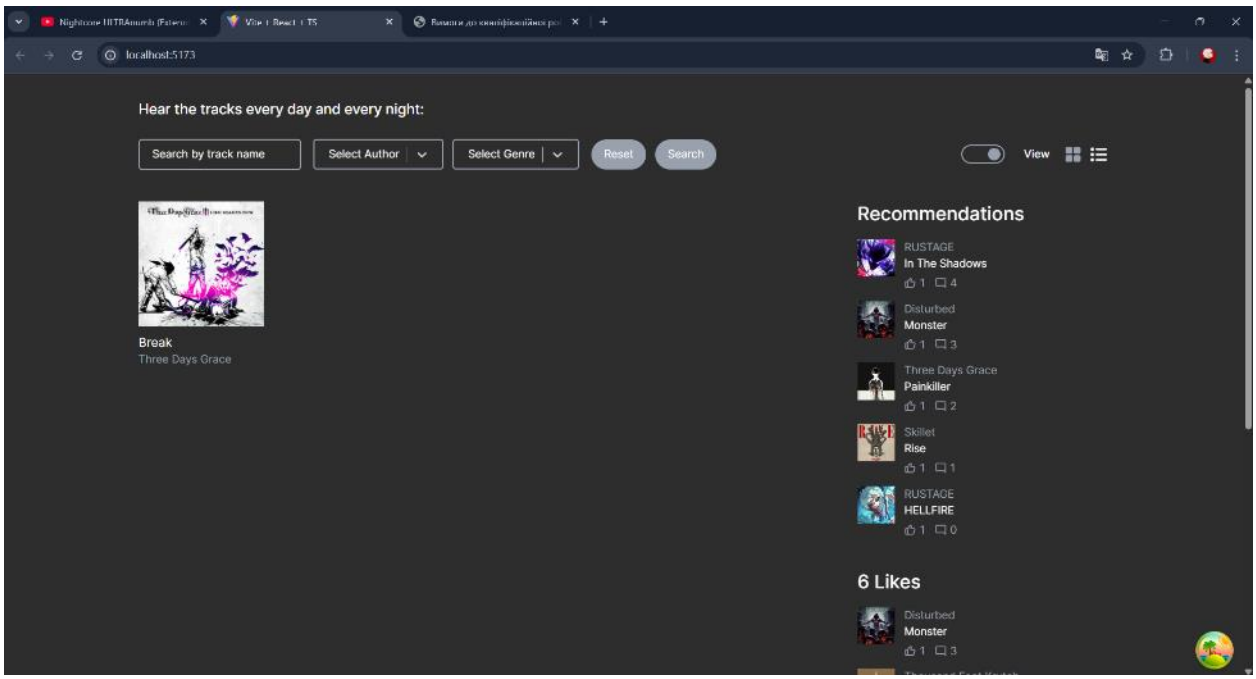

Рисунок 3.21 – Відображення аудіо по критеріям

Джерело: розроблено автором

Висновки до розділу 3

В ході реалізації програмного забезпечення системи було проведено технологічний аналіз створення програмного продукту відповідно до сучасних стандартів розробки програмного забезпечення. Визначено інструменти розробки клієнтської частини – фреймворк React з мовою програмування TypeScript та TailwindCSS для зовнішнього вигляду. Для реалізації серверної частини застосунку обрано Express.js з мовою програмування JavaScript. Збереження даних реалізовано з використанням нереляційної бази даних MongoDB.

Побудовано діаграму компонентів, яка демонструє взаємозв’язки між модулями клієнта, сервером і базою даних.

Проведено тестування системи для забезпечення правильної роботи та відсутності критичних помилок.

Описано використання системи веб-застосунку для прослуховування аудіо файлів.

Отже, в результаті роботи над розділом було створено робочу програмну систему для прослуховування аудіо файлів, що й було завданням на цю кваліфікаційну роботу.

ВИСНОВКИ

В результаті роботи над кваліфікаційною роботою було розроблено вебзастосунок для прослуховування аудіо файлів з підтримкою багатокритеріального фільтрування, пошукової та рекомендаційної системи.

У першому розділі було проведено аналіз предметної області. Розглянуто актуальні тенденції в галузі потокового відтворення аудіо, типові функціональні можливості подібних систем, основні вимоги до веб-застосунків такого типу. Було сформульовано мету, завдання проєкту та визначено функціональні та нефункціональні вимоги до майбутньої системи.

У другому розділі зосереджено увагу на моделюванні майбутнього програмного забезпечення. Побудовано інформаційну модель системи, розроблено діаграму сутностей (ER-діаграму), що описує структуру зберігання даних, а також діаграму активності, яка ілюструє основні сценарії взаємодії користувача із системою.

У третьому розділі було безпосередньо реалізовано програмну систему. Визначено засоби розробки: стек технологій MERN (MongoDB, Express.js, React, Node.js) бібліотеки та інструменти (TypeScript, TailwindCSS, react-query, wavesurfer.js тощо). Описано архітектуру застосунку, побудовано діаграму компонентів, реалізовано структуру бази даних і логіку взаємодії з нею. Проведено тестування системи на функціональність, коректність фільтрації та продуктивність інтерфейсу.

У підсумку реалізована система повністю відповідає поставленим вимогам і демонструє стабільну роботу в межах заявленого функціоналу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Spotify [Електронний ресурс] URL: https://open.spotify.com/ (дата звернення 07.04.2025)
2. SoundCloud [Електронний ресурс] URL: https://soundcloud.com/ (дата звернення 07.04.2025)
3. YouTube Music [Електронний ресурс] URL: https://music.youtube.com/ (дата звернення 07.04.2025)
4. Apple Music [Електронний ресурс] URL: https://music.apple.com/ua/new (дата звернення 07.04.2025)
5. React документація. URL: https://react.dev/ (дата звернення 03.05.2025)
6. React-Query документація. URL: https://tanstack.com/query/latest (дата звернення 03.05.2025)
7. React-Select документація. URL: https://react-select.com/home (дата звернення 03.05.2025)
8. React-Icons офіційний ресурс: URL: https://react-icons.github.io/react-icons/ (дата звернення 03.05.2025)
9. Typescript документація: URL: https://www.typescriptlang.org/ (дата звернення 03.05.2025)
10. TailwindCSS документація. URL: https://tailwindcss.com/ (дата звернення 03.05.2025)
11. Wavesurfer.js документація. URL: https://wavesurfer.xyz/ (дата звернення 03.05.2025)
12. Sonner документація. URL: https://ui.shadcn.com/docs/components/sonner (дата звернення 03.05.2025)
13. Framer Motion документація. URL: https://motion.dev/ (дата звернення 03.05.2025)
14. MongoDB документація. URL: https://www.mongodb.com/ (дата звернення 03.05.2025)
15. Mongoose документація. URL: https://mongoosejs.com/ (дата звернення 03.05.2025)

16. Node.js документація. URL: https://nodejs.org/uk (дата звернення 03.05.2025)
17. Express.js документація. URL: https://expressjs.com/ (дата звернення 03.05.2025)
18. Cloudinary офіційний ресурс. URL: https://cloudinary.com/ (дата звернення 03.05.2025)
19. Postman офіційний ресурс. URL: https://www.postman.com/ (дата звернення 03.05.2025)