

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 «Інженерія програмного забезпечення»)
на тему:

«Додаток з продажу музичних треків»

Виконав студент групи ПЗк-23-1
ДАВИДОВ Віктор Сергійович

Керівник роботи:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Рецензент:
САВІЦЬКИЙ Роман Святославович

Житомир – 2025

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

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ДАВИДОВ Віктор Сергійович**

Керівник роботи: **ГОДЛЕВСЬКИЙ Юрій Олександрович**

Тема роботи: **«Додаток з продажу музичних треків»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: архітектура, інтерфейс та функціональні можливості веб-додатку з продажу музичних треків, зокрема взаємодія між користувачами, управління треками та інтеграція платіжних систем і процес організації онлайн-продажу музичного контенту з використанням веб-технологій.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Годлевський Ю.О.	27.02.2025р.	27.02.2025р.
2	Годлевський Ю.О.	27.02.2025р.	27.02.2025р.
3	Годлевський Ю.О.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, джерел інформації, додатків та програмної розробки.

Текстова частина викладена на 66 сторінках друкованого тексту.

Джерела інформації містять 25 найменувань і займають 2 сторінки. Пояснювальна записка має 7 додатків. У роботі наведено 24 ілюстрацій та 16 таблиць.

Метою роботи є розробка веб-платформи для продажу музичних треків, яка забезпечує зручне завантаження, пошук, прослуховування та придбання музики.

У роботі проаналізовано функціональні вимоги користувачів, сформовано технічне завдання. Наведено концептуальну, логічну та фізичну моделі системи. Описано архітектуру програмного комплексу, алгоритми взаємодії користувача з платформою, а також реалізацію інтерфейсу з використанням React, Ant Design та CSS-модулів. Додатки містять фрагменти коду, приклади роботи системи та технічну документацію.

КЛЮЧОВІ СЛОВА: ТРЕК, STRIPE, ВЕБ-ДОДАТОК, ПЛЕЙЛИСТ, БІТ

					ІПЗ.КР.Б-121-25-ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Давидов В.С.			Додаток з продажу музичних треків. Пояснювальна записка			Літ.	Арк.	Аркушів	
Керівник		Годлевський Ю.О								3	115
Резецентт		Савіцький Р.С.						Житомирська політехніка, група ІПЗк-23-1			
Н. контр.		Тичина О.П.									
Зав. каф.		Вакалюк Т.А.									

ABSTRACT

The Bachelor's qualification thesis consists of an introduction, three chapters, conclusions, references, and appendices.

The main text is presented on 66 pages of printed material.

The list of references includes 25 sources and occupies 2 pages.

The explanatory report contains 7 appendices. The work includes 24 illustrations and 16 tables.

The total volume of the thesis is 115 pages.

The aim of the thesis is to develop a web platform for selling music tracks, which provides convenient uploading, searching, listening, and purchasing of music.

The work analyzes the functional requirements of users and formulates the technical specification. Conceptual, logical, and physical models of the system are presented.

The software architecture is described, including user interaction algorithms and the implementation of the user interface using React, Ant Design, and CSS Modules.

The appendices include code fragments, system operation examples, and technical documentation.

KEYWORDS: TRACK, STRIPE, WEB APPLICATION, PLAYLIST, BEAT

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ДОДАТКУ З ПРОДАЖІ МУЗИЧНИХ ТРЕКІВ	8
1.1. Постановка задачі.....	8
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури веб-додатку з продажу музичних треків.....	16
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	17
Висновки до першого розділу	19
2. ПРОЕКТУВАННЯ ВЕБ ДОДАТКУ З ПРОДАЖУ МУЗИЧНИХ ТРЕКІВ.....	20
2.1. Варіанти використання та об'єктно-орієнтована структура системи.....	20
2.2. Розробка бази даних системи.....	30
2.3. Проектування та реалізація алгоритмів роботи системи	33
Висновки до другого розділу	37
3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ПЛАТФОРМОЮ З ПРОДАЖУ МУЗИЧНИХ ТРЕКІВ.....	39
3.1. Порядок встановлення та налаштування параметрів платформи	39
3.2. Структура інтерфейсу веб-платформи	41
3.3. Порядок роботи з веб-платформою	53
Висновки до третього розділу	61
ВИСНОВКИ.....	62
ДЖЕРЕЛА ІНФОРМАЦІЇ.....	64
ДОДАТКИ.....	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

API – Прикладний програмний інтерфейс

БД – База даних

ІС – Інформаційна система

ПК – Персональний комп'ютер

ORM – Object-Relational Mapping (технологія взаємодії з базами даних через об'єкти)

JWT – JSON Web Token (токен для автентифікації користувачів)

UI – User Interface (користувацький інтерфейс)

CRUD – Create, Read, Update, Delete (основні операції з даними)

Stripe – Сервіс для інтеграції платіжних рішень в онлайн-системи

REST – Representational State Transfer (архітектурний стиль для створення API)

RTK – Redux Toolkit (бібліотека для управління станом у React)

Node.js – JavaScript-середовище виконання на стороні сервера

React – Бібліотека JavaScript для створення користувацьких інтерфейсів

Express – Легкий вебфреймворк для Node.js

MongoDB – Документоорієнтована база даних NoSQL

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми полягає у тому, що сучасний розвиток музичної індустрії значною мірою залежить від цифрових платформ, які забезпечують ефективну взаємодію між музичними продюсерами та слухачами. Онлайн-сервіси, такі як BeatStars чи SoundClick, стали невід’ємною частиною бізнес-моделі для багатьох незалежних виконавців, дозволяючи їм монетизувати свою творчість, розширювати аудиторію та працювати без посередників. Однак, навіть попри популярність таких платформ, зберігається потреба у створенні нових рішень, що поєднують у собі ширший функціонал, зручність користування та інструменти для ефективної взаємодії між усіма учасниками музичного процесу.

Метою випускної кваліфікаційної роботи бакалавра є розробка функціонального веб-додатку з продажу музичних треків, який забезпечуватиме зручний, інтуїтивно зрозумілий інтерфейс для прослуховування, придбання та управління музичним контентом. Додаток також має на меті підтримку взаємодії між продюсерами та слухачами, що сприятиме розвитку цифрового музичного середовища.

Завдання, необхідні для досягнення поставленої мети, охоплюють аналіз аналогічних платформ з метою виявлення їх сильних і слабких сторін, обґрунтований вибір технологій та архітектурних рішень, проєктування і реалізацію клієнтської й серверної частин системи з урахуванням ролей користувачів, створення бази даних для зберігання треків, інформації про користувачів і покупки, інтеграцію платіжної системи для забезпечення безпечних транзакцій, реалізацію функцій авторизації, додавання до вподобаного, створення плейлистів і коментування.

Об’єктом дослідження є процес організації онлайн-продажу музичного контенту з використанням веб-технологій.

Предметом дослідження є архітектура, інтерфейс та функціональні можливості веб-додатку з продажу музичних треків, зокрема взаємодія між користувачами, управління треками та інтеграція платіжних систем.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				7
Змн.	Арк.	№ докум.	Підпис	Дата		

1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ДОДАТКУ З ПРОДАЖІ МУЗИЧНИХ ТРЕКІВ

1.1. Постановка задачі

У сучасному цифровому світі музична індустрія зазнає постійних змін, і особливо важливою стала інтеграція ІТ-рішень у творчі процеси. Розвиток веб-технологій відкрив широкі можливості для створення онлайн-середовищ, які забезпечують не лише прослуховування та розповсюдження музики, а й повноцінну взаємодію між авторами, продюсерами та слухачами. Зокрема, велика популярність музичних маркетплейсів свідчить про затребуваність подібних рішень.

Метою кваліфікаційної роботи бакалавра є створення додатку з продажу музики, використовуючи сучасні веб-технології. Ця платформа дозволить музикантам, продюсерам та слухачам взаємодіяти між собою, купувати, продавати та просувати музичний контент.

Сучасні цифрові технології значно змінили музичну індустрію, відкривши нові можливості для незалежних виконавців. Продаж музики через онлайн-платформи дозволяє авторам напряду взаємодіяти зі своєю аудиторією, отримуючи прибуток без посередників. Яскравим прикладом успішної музичної платформи є BeatStars, яка дозволяє музикантам продавати свої композиції та співпрацювати з іншими виконавцями.

Додаток з продажу музики включатиме низку програмних модулів, кожен з яких відповідатиме за певний аспект функціональності платформи. Модуль продажу музики дозволить продюсеру завантажувати свої біти [21], встановлювати жанр, настрій та отримувати прибуток через інтегровані платіжні системи. Завдяки модулю прослуховування користувачі зможуть попередньо оцінити композицію перед покупкою.

Кожен виконавець матиме змогу створити персональну сторінку з детальною інформацією про себе та керувати власним контентом. Модуль

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				8
Змн.	Арк.	№ докум.	Підпис	Дата		

рекомендацій використовуватиме алгоритми персоналізації для підбору релевантної музики згідно зі смаками користувача. Модуль комунікації забезпечить обговорення треків, можливість залишати коментарі та надсилати повідомлення.

Управління контентом здійснюватиметься через модуль адміністрування, який дозволяє перевіряти нові завантаження, обробляти скарги та підтримувати безпеку системи. Уся платформа буде реалізована як веб-додаток із можливістю створення акаунту для продажу чи купівлі музики. Особливу увагу буде приділено захисту авторських прав та запобіганню незаконному використанню контенту.

Розробка проєкту включає кілька ключових етапів. Першим кроком є створення дизайну вебсайту, що охоплює загальну концепцію інтерфейсу, розробку UI-компонентів та оформлення сторінок [1]. Наступним етапом є реалізація основного функціоналу: впровадження логіки авторизації з урахуванням різних ролей користувачів, створення віддаленої бази даних для зберігання інформації, імплементація бекенду та налаштування взаємодії між серверною і клієнтською частинами. Окремо буде розроблена система ліцензування для захисту музичного контенту [3].

Завершальним етапом стане тестування додатку, що включатиме перевірку функціональності, інтерфейсу, безпеки та адаптивності дизайну до різних пристроїв. У результаті буде створена повноцінна платформа, яка забезпечить зручний та безпечний процес продажу музики для виконавців і простий доступ до улюблених композицій для слухачів.

Окрім основного функціоналу, важливим аспектом є забезпечення високого рівня зручності користування та безпеки. Для цього передбачається впровадження адаптивного дизайну, що дозволить комфортно використовувати платформу як на комп'ютерах, так і на мобільних пристроях. Також планується реалізація багатомовної підтримки (наприклад, українська, англійська), що розширить цільову аудиторію та зробить платформу більш доступною для користувачів з різних країн.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				9
Змн.	Арк.	№ докум.	Підпис	Дата		

З метою захисту авторських прав та інтелектуальної власності буде інтегровано механізми захисту контенту — водяні знаки, унікальні ID треків, обмеження на завантаження без покупки. Також платформа забезпечуватиме захищену обробку фінансових транзакцій через популярні платіжні шлюзи (наприклад, Stripe, PayPal), що гарантує конфіденційність і безпеку даних користувачів [15].

Після завершення тестування та впровадження системи аналітики буде можливо відстежувати поведінку користувачів на платформі, що дозволить надалі вдосконалювати її функціонал і покращувати користувацький досвід. Також можлива реалізація додаткових функцій у майбутньому, таких як: система push-сповіщень про нові релізи або знижки, внутрішній маркетплейс для продажу бітів, вокалу чи шаблонів, інтеграція з музичними стрімінговими сервісами, можливість створення колаборацій між виконавцями.

Таким чином, створення даного веб-додатку не лише стане демонстрацією практичних навичок розробника у сфері веб-програмування, дизайну інтерфейсів та побудови архітектури клієнт-серверних додатків, але й матиме реальну практичну цінність. Реалізована платформа сприятиме розвитку сучасного музичного середовища, надаючи незалежним виконавцям, саунд-продюсерам та авторам музики ефективний інструмент для монетизації своєї творчості, пошуку партнерів для колаборацій та розширення власної аудиторії.

Слухачі, у свою чергу, отримають доступ до великого каталогу якісного та унікального музичного контенту, з можливістю його прослуховування, придбання, а також взаємодії з виконавцями. Інтеграція соціальних функцій, системи рекомендацій і підтримка безпечних платіжних операцій робить платформу не лише технологічно сучасною, а й максимально зручною для щоденного використання. У результаті, даний проєкт є прикладом того, як за допомогою інформаційних технологій можна створити функціональне та корисне рішення, що відповідає потребам реального ринку.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				10
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів програмного продукту

На сьогоднішній день існує кілька популярних платформ для продажу та ліцензування музичних бітів. До них відносяться BeatStars, Traktrain та SoundClick. Всі ці сервіси надають продюсерам можливість завантажувати музику, встановлювати ціни та ліцензійні умови, а також взаємодіяти з покупцями.

Додаток «BeatStars» відрізняється стильним сучасним дизайном, що підкреслює професіоналізм та інновації платформи, а також логічною структурою і інтуїтивно зрозумілим меню, що забезпечує зручне управління. Функції додатку включають можливість прослуховування і покупки музичних треків, управління корзиною і процесом оформлення замовлення, а також можливість зв'язатися з продюсером для вирішення проблем.

Мобільна версія додатку дозволяє зручно користуватись платформою з будь-якого пристрою, дозволяючи користувачам робити покупки і управляти своїми треками в будь-який час і в будь-якому місці. Інтеграція соціальних медіа полегшує затвердження, підвищує популярність додатку та дозволяє користувачам легко ділитися своїми улюбленими треками зі своїми друзями.

Крім того, додаток BeatStars підтримує персоналізацію профілю користувача, де продюсери можуть оформлювати свої сторінки у власному стилі, додаючи банери, опис, соціальні посилання та відео. Це дозволяє кожному виконавцю або продюсеру створити унікальне враження для потенційних покупців.

Ще одним вагомим аспектом є аналітичні інструменти, які надає платформа. BeatStars забезпечує користувачів докладною статистикою щодо прослуховувань, вподобань, продажів та взаємодій з треками. Це дозволяє продюсерам краще розуміти свою аудиторію, аналізувати ефективність власної діяльності та приймати зважені рішення щодо просування контенту. Також сервіс підтримує декілька типів ліцензій — від стандартних до ексклюзивних — що відкриває ширші можливості для комерціалізації музики.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				11
Змн.	Арк.	№ докум.	Підпис	Дата		

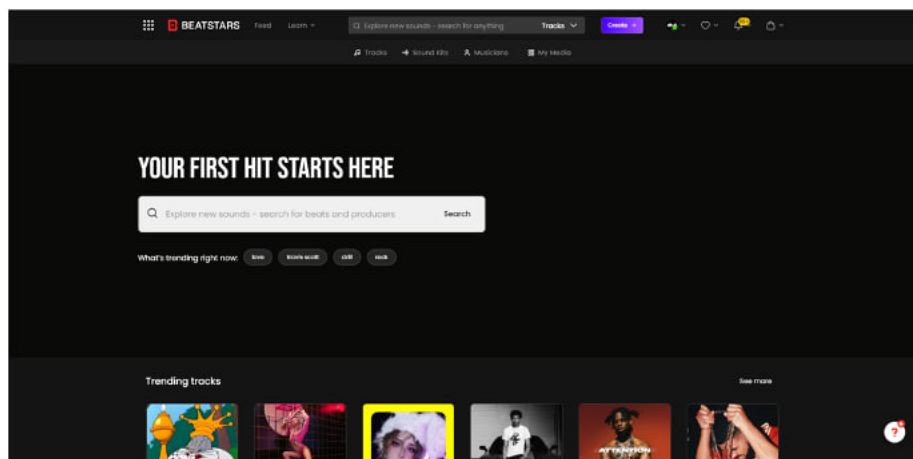


Рисунок 1.1 Вигляд головної сторінки веб-додатку BeatStars

Серед основних переваг додатку «BeatStars» можна виділити наступні:

Швидка робота та висока продуктивність: Завдяки оптимізованому коду та використанню сучасних фреймворків, додаток працює швидко та стабільно, навіть при високому навантаженні.

Захист особистих даних: Забезпечується високий рівень безпеки облікових записів, зокрема через двохфакторну аутентифікацію та шифрування даних.

Адаптивний дизайн: Інтерфейс автоматично підлаштовується під розміри екрана, забезпечуючи комфортне користування як на смартфонах, так і на планшетах та ПК.

Зручна система фільтрації та пошуку: Користувачі можуть швидко знаходити потрібні треки або виконавців завдяки продуманим категоріям, тегам та пошуковим алгоритмам.

Система рекомендацій: Алгоритми на основі вподобань користувача пропонують релевантні треки, що сприяє підвищенню взаємодії з платформою.

Можливість монетизації для авторів: Продюсери можуть не лише завантажувати свої треки, але й продавати їх без посередників, отримуючи дохід напряму.

Аналітика для авторів: Додаток надає продюсерам детальну статистику щодо кількості прослуховувань, вподобань та продажів.

Додаток «TrackTrain» має простий та інтуїтивно зрозумілий інтерфейс, який забезпечує простоту використання та швидкий доступ до музичних

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				12
Змн.	Арк.	№ докум.	Підпис	Дата		

інструменталів. Функціонал включає в себе профілі виконавців, рейтинги та огляди прослуховувань, щоб допомогти користувачам приймати обґрунтовані рішення щодо вибору музики, контактну інформацію для взаємодії з музикантами тощо.

Мобільна версія додатку ідеально поєднується з гаджетами і забезпечує легкий доступ з будь-якого пристрою. Інтеграція з соціальними мережами полегшує доступ і схвалення офіційних сторінок виконавця, сприяючи підвищенню залученості користувачів. Додатку підвищує довіру користувачів, використовуючи новітні технології безпеки для захисту особистих даних і фінансових транзакцій

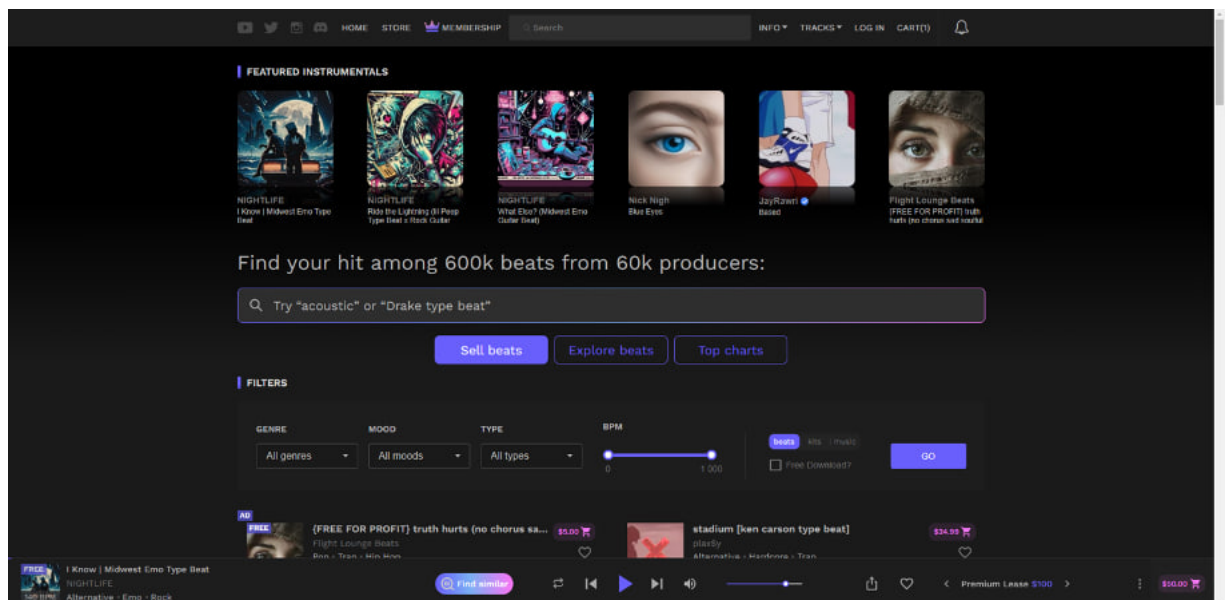


Рисунок 1.2 Вигляд головної сторінки веб-додатку Tracktrain

Основні переваги додатку «Tracktrain» полягають у його персоналізованому підході, зручності використання та функціональності. Система рекомендацій аналізує поведінку користувачів і пропонує інструментали, що відповідають їхнім уподобанням. Прозора система оцінювання дозволяє переглядати рейтинги та відгуки, що підвищує якість вибору. Профілі виконавців містять детальну інформацію — біографії, дискографію, посилання на соціальні мережі та контакти для співпраці.

Купівля музики реалізована через зручну систему ліцензування, яка дає змогу швидко придбати трек для комерційного або некомерційного

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				
Змн.	Арк.	№ докум.	Підпис	Дата		13

використання. Важливою особливістю є можливість прямої взаємодії між виконавцями та клієнтами, що сприяє індивідуальним замовленням та колабораціям. Контент на платформі регулярно оновлюється, забезпечуючи різноманітність і актуальність. Безпека особистих даних та фінансових операцій гарантується завдяки сучасним технологіям шифрування та автентифікації. Додаток є адаптивним і підтримує роботу на різних платформах, зокрема Android та iOS.

Додаток «Soundclick» є сучасним онлайн-ресурсом для музикантів та слухачів з яскравим і стильним дизайном, що підкреслює його професійний вигляд і відповідає сучасним веб-трендам. Інтерфейс додатку інтуїтивно зрозумілий і зручний, що дозволяє користувачам легко знаходити необхідну інформацію. Візуальна привабливість та зручність користування забезпечують позитивний досвід користувачів.

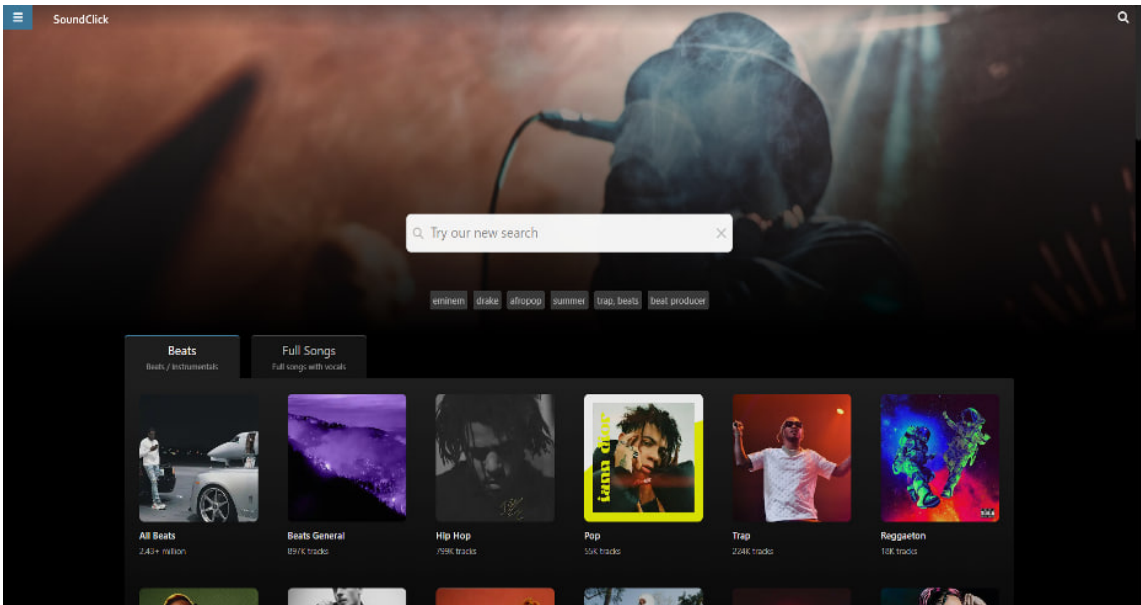


Рисунок 1.3 Вигляд головної сторінки веб-додатку SoundClick

Сучасні музичні платформи пропонують широкий вибір функціональності як для слухачів, так і для артистів. Однією з головних переваг є наявність великої бібліотеки музики різних жанрів, що охоплює як роботи відомих артистів, так і композиції незалежних виконавців. Користувачі можуть безкоштовно прослуховувати музику онлайн, що забезпечує легкий доступ до творчості без необхідності оплат.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Для музикантів платформи відкривають можливість виставляти свої треки на продаж, налаштовувати ціни та ліцензійні умови. Також вони мають змогу оформлювати власні профілі, додавати біографічну інформацію, посилання на соціальні мережі, альбоми, відео та взаємодіяти з фанатами через коментарі, блоги або повідомлення. Інструменти для просування, як платні, так і безкоштовні, дозволяють артистам підвищувати видимість своєї музики. До того ж, зручна аналітика допомагає відстежувати статистику прослуховувань, продажів та активність слухачів, що сприяє ефективнішому розвитку творчості.

Значну роль відіграють рейтинги та чарти, які оновлюються щодня чи щотижня, що дозволяє користувачам орієнтуватися в популярних треках і стежити за актуальними трендами. Крім того, підтримка мобільних пристроїв гарантує зручний доступ до платформи в будь-якому місці.

Виходячи з наявних переваг та недоліків, складемо порівняльну таблицю вищезгаданих програмних аналогів (табл. 1.1)

Таблиця 1.1

Характеристика	Beatstars	Tracktrain	Soundclick
Прослуховування треків	+	+	+
Продаж музики	+	+	+
Персоналізовані профілі	+	-	+
Система рекомендацій	-	+	-
Адаптивність до мобільних пристроїв	+	-	+
Інтерфейс користувача	Сучасний	Мінімалістичний	Стандартний
Можливість створювати плейлисти	+	-	+-
Пошук музичних треків за жанром	+	+	+
Залишення коментарів	+	+-	-

Аналізуючи функціональність подібних сервісів, можна виокремити основні можливості: купівля та продаж музики, система плейлистів і персоналізованих рекомендацій, інтеграція з соціальними мережами, а також вбудовані маркетингові інструменти (реклама, аналітика, розсилки).

Попри розвинений функціонал, у таких платформ є й певні недоліки. До них належать високі комісії з продажу, обмежені можливості кастомізації особистої сторінки продюсера, відсутність інтеграції з деякими платіжними системами та часткова закритість функціоналу — не всі можливості доступні у безкоштовній версії.

Отже, за проведенням аналізом можна визначити, що додаток “BeatStars” видається найбільш привабливим із врахуванням його сучасного дизайну та широкого спектру послуг, які надаються у сфері музикального бізнесу. Він також відзначається гарною функціональністю, адаптацією під мобільні пристрої та наявністю інтеграції з соціальними мережами, що робить його привабливим для користувачів.

1.3. Вибір архітектури веб-додатку з продажу музичних треків

Під час створення веб-додатку для організації робочого часу важливо обрати структуру, яка забезпечить надійну роботу системи, її зручну підтримку, безпечний доступ до даних і можливість масштабування. У цьому проекті було обрано класичну архітектуру з поділом на клієнтську, серверну частини та окремий рівень зберігання даних.

Була обрана архітектурна модель, яка базується на принципі поділу обов’язків між трьома основними складовими:

Інтерфейс користувача (frontend) відповідає за візуальну частину та взаємодію з користувачем [4]. Застосовано фреймворк React у поєднанні з бібліотеками Redux та RTK Query для роботи з даними [23], а також Ant Design для реалізації гнучкого інтерфейсу [16].

Серверна логіка (backend) обробляє запити, реалізує ключову функціональність системи, перевіряє права доступу та здійснює взаємодію з базою даних. В основі — середовище Node.js з Express [7].

Сховище даних (database) використовується для збереження інформації про працівників, розклади, змінні запити, статистику тощо. Було обрано

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				16
Змн.	Арк.	№ докум.	Підпис	Дата		

MongoDB — документо-орієнтовану базу, що дозволяє легко адаптуватися до змін у структурі збережених даних [9].

Обрана архітектура веб-додатку має низку переваг, серед яких масштабованість, що дозволяє незалежно масштабувати кожен компонент у залежності від навантаження. Такий підхід сприяє розподіленій розробці, де фронтенд і бекенд можуть розвиватися паралельно без взаємних обмежень. Значну увагу приділено захисту даних: використовується безпечна система автентифікації на основі токенів або сесій, що забезпечує надійність обробки персональної інформації користувачів. Крім того, завдяки модульності, кожен компонент системи є ізольованим, що спрощує підтримку, оновлення й масштабування додатку [17].

Під час проєктування також було проаналізовано інші архітектурні підходи. Зокрема, монолітна архітектура вирізняється простотою розгортання, однак вона є менш гнучкою в контексті майбутнього масштабування та модернізації. З іншого боку, мікросервісна архітектура добре підходить для великих систем з високими вимогами до масштабованості, проте потребує значних ресурсів для налаштування, підтримки та забезпечення міжсервісної взаємодії.

З урахуванням потреб і обсягів проєкту, обрана модель виявилася найбільш збалансованим і практичним рішенням, що дозволяє швидко реалізувати функціональну систему з можливістю подальшого розширення.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки сучасної веб-системи з керування розкладом було обрано архітектурну модель клієнт-серверного типу. Такий підхід забезпечує чітке розділення функцій між візуальним інтерфейсом та серверною логікою, що сприяє масштабованості, структурованості та зручності в супроводі проєкту.

Серверна логіка реалізована з використанням Node.js у зв'язці з фреймворком Express, що дозволяє організувати ефективну обробку HTTP-

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				17
Змн.	Арк.	№ докум.	Підпис	Дата		

запитів через REST API. Кожен функціональний блок винесено в окремі маршрути, які делегують обробку до відповідних контролерів. Це спрощує підтримку та розширення додатку.

Для взаємодії з базою даних використовується MongoDB, яка забезпечує гнучке зберігання структурованих і динамічних даних. Робота з базою реалізована через Mongoose — обгортку для MongoDB, яка дозволяє описувати структуру даних у вигляді моделей, проводити валідацію, працювати з індексами та використовувати middleware-функції. Всі моделі розміщено окремо, що полегшує підтримку та повторне використання [11].

Весь серверний код написано на TypeScript, що забезпечує статичну типізацію, покращує якість коду та знижує ризик помилок. Основні об'єкти — запити, відповіді, моделі — описані через інтерфейси, що дозволяє зберігати узгодженість на всіх рівнях проєкту. Також для підтримки стилю коду застосовується ESLint [25].

Обробка помилок централізована: створено middleware, яке перехоплює винятки, що виникають під час виконання запитів, і повертає уніфіковану відповідь. Це дозволяє легко знаходити проблеми й уніфікувати логування.

Фронтенд частина додатку створена на основі бібліотеки React, яка дозволяє створювати динамічний інтерфейс із використанням компонентного підходу. Інтерфейс поділений на незалежні елементи — сторінки, форми, таблиці, які можна легко комбінувати й масштабувати.

Для побудови інтерфейсу застосовано бібліотеку Ant Design (ANTD) — набір готових UI-компонентів, що значно прискорює розробку. Компоненти адаптивні, підтримують кастомізацію і добре підходять для створення сучасного вигляду з урахуванням UX/UI принципів. Система стилів налаштована таким чином, щоб легко підтримувати єдиний стиль на всіх сторінках.

Дані з бекенду отримуються через REST API, запити організовані в окремі модулі, що спрощує їх повторне використання і дає змогу централізовано керувати обробкою відповідей, статусів та помилок.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Також клієнтська частина реалізована на TypeScript, що забезпечує узгодженість типів із сервером, покращує читабельність і полегшує налагодження під час розробки.

Висновки до першого розділу

У першому розділі було розглянуто особливості функціонування існуючих платформ для продажу музики, зокрема BeatStars, Traktrain та SoundClick. На основі проведеного аналізу було визначено, що для створення сучасного, функціонального та безпечного веб-додатку доцільно використовувати набір перевірених технологій, які дозволять реалізувати повноцінну систему для взаємодії виконавців та слухачів.

Проектований додаток повинен підтримувати можливості завантаження музичних треків, прослуховування, монетизації контенту, а також забезпечувати захищену авторизацію, ефективну систему управління користувачами та платіжну інтеграцію. Для цього потрібно обрати такі інструменти, які гарантують масштабованість, надійність та зручність у розробці.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				19
Змн.	Арк.	№ докум.	Підпис	Дата		

2. ПРОЕКТУВАННЯ ВЕБ ДОДАТКУ З ПРОДАЖУ МУЗИЧНИХ ТРЕКІВ

2.1. Варіанти використання та об'єктно-орієнтована структура системи

Веб-платформа з продажу музичних треків передбачає взаємодію між двома основними типами користувачів — слухачами та продюсером (адміністратором). Слухачі можуть реєструватися, переглядати каталог треків, здійснювати їх прослуховування, додавати треки до улюблених або до плейлистів, а також купувати ліцензії на біти через інтегровану платіжну систему. Після покупки трек з'являється у списку придбаного контенту користувача з можливістю його завантаження у відповідному форматі.

Продюсер (власник платформи), який одночасно виконує роль адміністратора, має доступ до адміністративного інтерфейсу, через який він завантажує нові треки, змінює їх опис, оновлює обкладинки та керує цінами. Крім того, продюсер отримує повідомлення через форму зворотного зв'язку та може модерувати коментарі або контент. Статистика переглядів і вподобань треків дозволяє відслідковувати популярність контенту та визначати його комерційний потенціал.

Для кращого розуміння функціональної моделі системи на рисунку зображено діаграму варіантів використання. Вона демонструє основні дії двох типів користувачів — слухача та продюсера (який також виконує функції адміністратора). З боку слухача реалізовано пошук музики, попереднє прослуховування треків, додавання вподобаного контенту, створення плейлистів, залишення коментарів та оформлення покупок. Продюсер, у свою чергу, може завантажувати нові біти, редагувати або видаляти їх, переглядати аналітику взаємодії з контентом (лайки, покупки), а також здійснювати модерацію контенту й обробляти повідомлення, надіслані через контактну форму.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Таким чином, діаграма демонструє ключові дії кожного типу користувача в межах платформи, з урахуванням ролей та доступного функціоналу, забезпечуючи цілісну картину логіки роботи сервісу (рис. 2.1).



Рисунок 2.1 Діаграма варіантів використання

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.11.

Опис прецеденту «Вхід/Реєстрація»

Таблиця 2.1

Назва варіанту використання	Вхід/Реєстрація
Основні актори	Продюсер (Адміністратор), Слухач
Короткий опис	Користувач вводить адресу електронної пошти та пароль для входу в систему або створює новий обліковий запис. Слухачі можуть реєструватися самостійно, роль продюсера встановлюється вручну через базу даних

Опис прецеденту «Пошук музики»

Таблиця 2.2

Назва варіанту використання	Пошук музики
Основні актори	Слухач
Короткий опис	Слухач може переглядати каталог музичних треків та використовувати фільтри для пошуку за жанром, BPM, ключем, настроєм або популярністю

Опис прецеденту «Попереднє прослуховування треків»

Таблиця 2.3

Назва варіанту використання	Попереднє прослуховування треків
Основні актори	Слухач
Короткий опис	Користувач може слухати демо-версії треків на головній сторінці або в каталозі без необхідності завантаження. Програваач працює глобально під час навігації по сайту

Опис прецеденту «Додавання лайків»

Таблиця 2.4

Назва варіанту використання	Додавання лайків
Основні актори	Слухач
Короткий опис	Користувач може вподобати трек. Усі лайки зберігаються в обліковому записі користувача. Лайк також впливає на популярність треку

Опис прецеденту «Залишення коментарів»

Таблиця 2.5

Назва варіанту використання	Залишення коментарів
Основні актори	Слухач
Короткий опис	Користувач може залишати текстові відгуки до треків. Також можливе редагування чи видалення власних коментарів. Продюсер має право на модерацію

Опис прецеденту «Створення плейлиста»

Таблиця 2.6

Назва варіанту використання	Створення плейлиста
Основні актори	Слухач
Короткий опис	Слухач може створювати власні плейлисти, додавати треки до них та організовувати свою музичну колекцію

Опис прецеденту «Купівля треків»

Таблиця 2.7

Назва варіанту використання	Купівля треків
Основні актори	Слухач
Короткий опис	Користувач обирає трек, ліцензію (MP3, WAV або Stems), додає в кошик і здійснює оплату через Stripe. Після цього трек доступний для завантаження у профілі

Опис прецеденту «Завантаження треків»

Таблиця 2.8

Назва варіанту використання	Завантаження треків
Основні актори	Продюсер (Адміністратор)
Короткий опис	Продюсер (власник платформи) додає трек через форму: назва, жанр, BPM, файли (MP3, WAV, ZIP), обкладинка

Опис прецеденту «Редагування / Видалення треків»

Таблиця 2.9

Назва варіанту використання	Редагування / Видалення треків
Основні актори	Продюсер (Адміністратор)
Короткий опис	Продюсер може редагувати інформацію про трек або повністю видалити його з бази

Опис прецеденту «Модерація контенту»

Таблиця 2.10

Назва варіанту використання	Модерація контенту
Основні актори	Продюсер (Адміністратор)
Короткий опис	Продюсер має змогу видаляти коментарі, які порушують правила платформи, або приховувати контент, що не відповідає вимогам

Опис прецеденту «Отримання повідомлень від користувачів»

Таблиця 2.11

Назва варіанту використання	Отримання повідомлень від користувачів
Основні актори	Продюсер (Адміністратор)
Короткий опис	Через контактну форму слухач може надіслати повідомлення продюсеру. Повідомлення надходить на email, пов'язаний з обліковим записом

Для забезпечення надійної роботи веб-платформи з продажу музичних треків, система повинна ефективно виконувати всі ключові функції — від авторизації користувачів до обробки платежів, керування треками та взаємодії між слухачами та продюсером. Вимоги до системи формуються з урахуванням сучасних підходів до UX-дизайну, безпеки та масштабованості.

Основна мета проєкту полягає у створенні інтуїтивного середовища для взаємодії між продюсером (власником платформи) та слухачами: завантаження, прослуховування, вподобання та купівля треків, залишення коментарів, а також зручна організація плейлистів. Важливою частиною системи є також функціонал адміністратора, який об'єднаний з роллю продюсера — саме він здійснює модерацію контенту, керування треками та обробку повідомлень від користувачів.

Платформа реалізується у вигляді адаптивного веб-додатку, доступного з будь-якого пристрою. Це дозволяє слухачам зручно знаходити потрібну музику за допомогою фільтрів, слухати демо, створювати добірки та купувати біти з ліцензією, що відповідає їхнім потребам. Продюсер, у свою чергу, отримує простий інструмент для керування своїм контентом, аналізу статистики й отримання прибутку від продажів.

Вимоги користувачів:

Можливість реєстрації та редагування профілю (фото, біографія та інші особисті дані);

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				
Змн.	Арк.	№ докум.	Підпис	Дата		25

Можливість завантаження та продажу бітів, додавання обкладинок, жанру та метаданих;

Створення плейлистів за жанром, настроєм або популярністю;

Пошук та фільтрація бітів за назвою, продюсером, жанром;

Можливість прослуховування демо-версій бітів через вбудований плеєр;

Можливість замовлення бітів або додавання їх до обраного;

Вимоги адміністратора веб-додатку:

Управління обліковими записами користувачів: можливість перегляду, редагування та видалення акаунтів користувачів.

Управління контентом: можливість перевірки прав на використання бітів та їх видалення чи редагування.

Аналітика та звіти: отримання даних про популярність бітів, активність користувачів для кращого управління платформою.

Характеристика об'єкта комп'ютеризації:

Користувач додатку буде мати можливість слухати музику, шукати та фільтрувати біт, а також зручно працювати з додатком завдяки простому та зрозумілому інтерфейсу.

Нефункціональні вимоги:

Сприйняття – інтуїтивно зрозумілий інтерфейс з елементами керування, швидким доступом до всіх функцій і підказками.

Надійність – середній час безвідмовної роботи системи має бути не меншим за 30 робочих днів.

Продуктивність – система повинна підтримувати мінімум 50 одночасних активних користувачів без збоїв.

Можливість експлуатації – система повинна мати можливість збільшувати потужності з розширенням кількості користувачів, не впливаючи на її продуктивність.

Аналіз вимог користувачів та діаграма варіантів використання:

Use Case діаграма відображає основні сценарії взаємодії між двома ключовими типами користувачів системи — слухачем і продюсером. Слухач —

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				26
Змн.	Арк.	№ докум.	Підпис	Дата		

це звичайний зареєстрований користувач, який може переглядати каталог треків, здійснювати пошук за фільтрами, прослуховувати демо-версії бітів, додавати треки до вподобаних, створювати персональні плейлисти, залишати коментарі під треками, оформлювати замовлення та завантажувати куплені композиції. Крім того, користувач має можливість надіслати повідомлення продюсеру через спеціальну форму зв'язку.

Зі свого боку, продюсер (який водночас виконує роль адміністратора системи) має доступ до розширеного функціоналу. Він може завантажувати нові біти на платформу, редагувати вже наявні треки, змінюючи їхні метадані або обкладинку, видаляти застарілий або проданий контент, а також отримувати повідомлення від слухачів. Продюсер також здійснює модерацію коментарів, контролюючи дотримання правил спільноти.

Таким чином, Use Case діаграма відображає повну логіку взаємодії користувачів із системою залежно від їх ролі. Вона ілюструє типові дії, які здійснюють слухач і продюсер у межах веб-платформи, а також демонструє, як реалізований функціонал дозволяє їм взаємодіяти з музичним контентом, здійснювати покупки, формувати добірки треків і підтримувати зворотний зв'язок між сторонами.

Системні вимоги:

Платформа повинна відповідати вимогам для комп'ютера з наступною конфігурацією:

4 ГБ оперативної пам'яті

Процесор з тактовою частотою від 1 GHz

Операційна система, яка підтримує використання інтернету

Браузер, що підтримує сучасні веб-технології (HTML5, JavaScript, React)
NodeJs, MongoDB

Підтримка для роботи з базами даних через API

У серці системи повинна бути використана реляційна СУБД для зберігання даних (наприклад, MongoDB або MySQL)

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Аудіофайли бітів, завантажених користувачами, повинні зберігатися на хмарному сховищі для зручного доступу та масштабованості.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать програмну реалізацію додатку. На рисунку 2.2. наведено Діаграму класів, яка використовується для його роботи

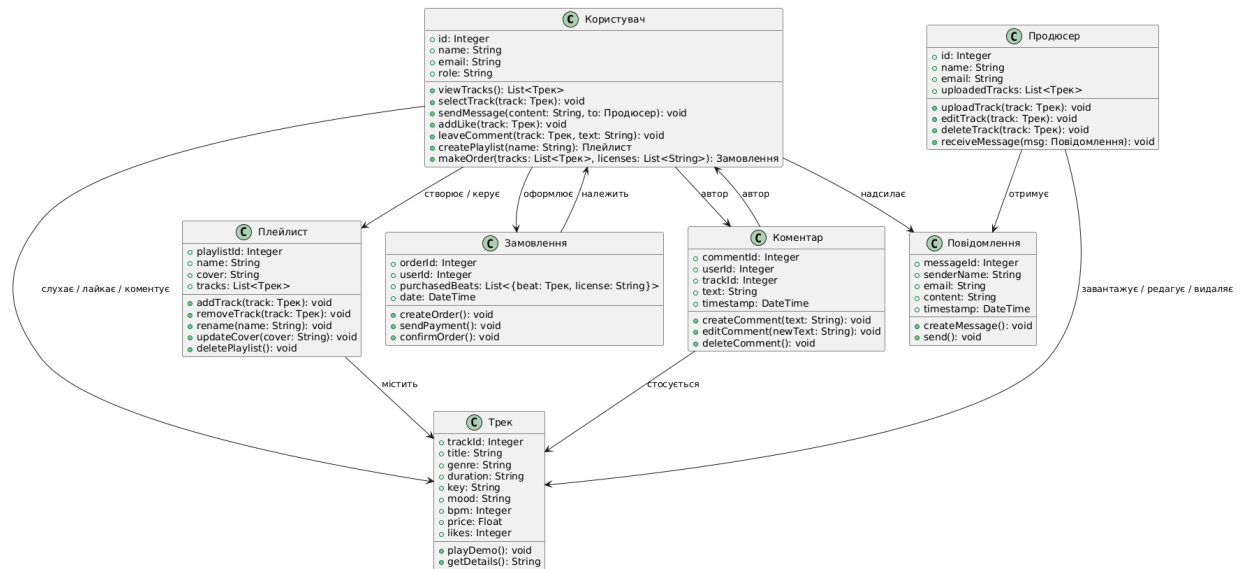


Рисунок 2.2 Діаграма класів

Опис діаграми класів:

Діаграма класів моделює основні сутності та їхню взаємодію в системі з продажу музичних треків. Центральним є клас користувача, який представляє слухача платформи. Після авторизації він отримує доступ до функціоналу: перегляд бітів, прослуховування демо-версій, додавання треків у вподобані, створення плейлистів, залишення коментарів і купівля треків. Користувач має унікальний ідентифікатор, ім'я, email та роль у системі (user). Серед методів класу: перегляд списку доступних треків, вибір треку для прослуховування, надсилання повідомлення продюсеру через форму зворотного зв'язку, додавання лайка до треку, створення коментаря, створення плейлиста, а також оформлення замовлення на купівлю.

Продюсер виступає адміністратором системи. Це спеціалізований користувач, якому вручну через базу даних надається роль "admin". Продюсер може завантажувати нові треки, редагувати їх метадані, видаляти контент і

отримувати повідомлення від користувачів. Він також має унікальний ідентифікатор, ім'я, електронну пошту та список завантажених треків.

Клас треку моделює окрему музичну композицію, яка може бути прослухана, лайкнута, додана до плейлиста або придбана. Трек має унікальний ідентифікатор, назву, жанр, тривалість, ВРМ, тональність, настрій, ціну та кількість лайків. Доступні методи включають відтворення демо-версії та перегляд детальної інформації про трек.

Клас плейлиста дозволяє користувачеві зберігати вподобані треки в окремій колекції. Плейлист має унікальний ідентифікатор, назву, обкладинку та список треків. Можливості включають додавання або видалення треку, перейменування плейлиста, зміну обкладинки та видалення всього плейлиста.

Операція купівлі треку реалізована через клас замовлення. Замовлення містить унікальний ідентифікатор, посилання на користувача, що його створив, масив об'єктів із інформацією про куплені треки та вибрані ліцензії, а також дату покупки. Методи передбачають створення замовлення, проведення оплати та її підтвердження.

Клас коментар описує повідомлення, які користувач залишає під треком. У ньому зберігаються унікальний ідентифікатор, посилання на користувача й трек, зміст коментаря та час його публікації. Передбачено створення, редагування та видалення коментарів.

Окремий клас описує повідомлення, які надсилаються користувачами через форму зв'язку. Він містить ім'я відправника, електронну адресу, зміст повідомлення, дату й час відправлення. Система обробляє такі повідомлення як email-запити, що потрапляють до продюсера.

Загалом діаграма класів відображає повний цикл роботи користувачів з платформою — від авторизації та прослуховування музики до взаємодії з контентом і оформлення замовлення.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				29
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2. Розробка бази даних системи

Для збереження інформації про користувачів, треки, плейлисти та інші дані в додатку використовується база даних MongoDB, яка є реляційною системою керування базами даних. Усі дані зберігаються у відповідних таблицях з чітко визначеними зв'язками між ними [2].

База даних складається з наступних основних таблиць:

Users – зберігає дані користувачів системи.

Beats – містить інформацію про музичні треки.

Playlists – зберігає дані про створені плейлисти.

Actions – зберігає коментарі до треків.

Опис кожної з таблиць бази даних наведений у таблицях 2.12 – 2.15.

Опис таблиці «Users»

Таблиця 2.12

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
email	varchar	Електронна пошта користувача
password	varchar	Пароль
username	varchar	Ім'я користувача
avatar	varchar	Шлях до зображення користувача
role	enum	Роль (admin, user)
likedBeats	array	Список бітів, які лайкнув користувач
purchasedBeats	array	Список бітів, які придбав користувач

Опис таблиці «Beats»

Таблиця 2.13

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
img	varchar	Шлях до зображення обкладинки треку
title	varchar	Назва музичного треку
time	varchar/time	Тривалість треку у форматі HH:MM:SS
bpm	int	Темп треку в ударах за хвилину

Продовження таблиці 2.13

tags	array of string	Список тегів (виконавці, стиль тощо)
price	float	Ціна треку
key	varchar	Музичний ключ (наприклад, G Minor)
mood	array of string	Настрій треку (наприклад, Relaxed, Bouncy)
genre	varchar	Жанр
soundSrc	varchar	Шлях до MP3 файлу
wavSrc	varchar	Шлях до WAV файлу
zipSrc	varchar	Шлях до ZIP архіву зі стемами
likes	int	Кількість лайків
isExclusiveSold	boolean	Чи проданий трек на ексклюзив
likedBy	array of object	Масив об'єктів тих, хто лайкнули трек

Опис таблиці «Playlists»

Таблиця 2.14

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
user_id	Int	ID користувача, який створив плейлист
playlistName	varchar	Назва плейлиста
playlistPreview	varchar	Шлях до обкладинки плейлисту
beats	array of string	Список доданих бітів до плейлисту

Опис таблиці «Actions»

Таблиця 2.15

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
beat_id	int	ID треку
login	Int	Юзернейм користувача
action	int	Тип дії
value	text	Текст коментаря
avatar	varchar	Шлях до зображення користувача
hidden	int	Прихований коментар чи ні
createdAt	datetime	Дата залишення коментаря
updatedAt	datetime	Дата оновлення коментаря

У контексті збереження та обробки інформації у додатку з продажу музичних треків, дані користувачів, треків, плейлистів, лайків, коментарів та інші пов'язані об'єкти зберігаються у базі даних MongoDB. Це документоорієнтована система управління базами даних, яка дозволяє зручно працювати зі структурами даних у форматі JSON.

MongoDB зберігає інформацію у вигляді колекцій документів. Кожен документ містить набір ключів і значень, які можуть бути вкладеними, що дозволяє ефективно зберігати складну інформацію, пов'язану з користувачами, треками, їх прослуховуванням і взаємодією [5].

Усі основні операції — створення, оновлення, видалення та читання даних — реалізовані через API-запити до серверної частини, написаної на Node.js. Ці запити обробляють дії користувача (наприклад, купівлю треку або додавання його до улюблених) та взаємодіють із базою даних у реальному часі. Це дозволяє гарантувати, що кожна дія користувача (наприклад, натискання "лайк", створення плейлиста або завантаження нового біта продюсером) буде збережена та оброблена належним чином.

Також для доступу до цих даних використовуються контролери бекенду, які реалізують відповідні маршрути і логіку. Таким чином, уся система зберігання та взаємодії з даними в додатку є централізованою, надійною та забезпечує високу швидкість обробки запитів без необхідності використання сторонніх технологій типу блокчейн

Крім того, структура даних у MongoDB дозволяє реалізувати гнучкі зв'язки між об'єктами, наприклад, зберігати масив ідентифікаторів лайкнутих бітів у документі користувача або список коментарів безпосередньо в документі треку. Це дозволяє зменшити кількість запитів до бази під час побудови інтерфейсу користувача та підвищити продуктивність системи. Для забезпечення безпеки доступу до даних усі запити, що змінюють інформацію, захищені механізмом авторизації з використанням JWT-токенів [17], а також дотримуються правил кросдоменної взаємодії (CORS) [24]. Завдяки цьому зберігається цілісність інформації та гарантується, що лише авторизований

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				32
Змн.	Арк.	№ докум.	Підпис	Дата		

користувач може здійснювати певні дії, наприклад, редагувати свій профіль, залишати коментарі або купувати треки. У комплексі, така організація бази даних дозволяє забезпечити масштабованість і надійність веб-платформи при одночасній підтримці високої продуктивності та швидкості обробки запитів.

2.3. Проектування та реалізація алгоритмів роботи системи

Додаток з продажу музичних треків побудований з орієнтацією на зручність взаємодії користувача з основними функціями платформи — пошуком, прослуховуванням, вподобанням, збереженням у плейлистах та покупкою музики. Інтерфейс розроблено таким чином, щоб навіть новий користувач зміг швидко зорієнтуватися в системі.

Процес взаємодії починається з авторизації або реєстрації. У разі успішного входу слухач потрапляє на головну сторінку, звідки може перейти до каталогу треків. У розділі «Пошук» користувач обирає спосіб взаємодії: або задає параметри пошуку (жанр, настрій, bpm, ім'я продюсера), або переглядає рекомендації й популярні треки. Після цього система відображає релевантні результати, з яких слухач може обрати біт для попереднього прослуховування.

Якщо трек зацікавив, користувач має можливість додати його до вподобаних або у власний плейлист. Далі він переходить до процесу покупки — обирає ліцензію, оплачує трек через платіжну систему (Stripe) і отримує можливість його завантаження разом із ліцензією у PDF-форматі.

Описаний алгоритм охоплює основний сценарій взаємодії звичайного користувача — слухача. Інші функціональні сценарії для продюсера (завантаження, редагування та видалення треків, перегляд статистики, модерація контенту) реалізовані за аналогічним підходом з урахуванням відповідних ролей. Усі ключові етапи поведінки користувача логічно структуровані й забезпечують послідовність та передбачуваність у навігації додатком. Алгоритм дій слухача, починаючи з моменту входу в додаток і до завантаження придбаного треку, представлено нижче (рис 2.3).

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				
Змн.	Арк.	№ докум.	Підпис	Дата		33



Рисунок 2.3 Діаграма активності

2.4. Реалізація функціоналу додатку з продажу музичних треків

Платформа побудована як модульна система, в якій чітко розділено функціональність між основними складовими: інтерфейсом користувача, модулями керування контентом, плейлистами, повідомленнями, базою даних і API-сервісами. На першому етапі взаємодії користувач проходить реєстрацію та авторизацію, після чого залежно від ролі — слухач або продюсер — отримує відповідний набір можливостей. У системі передбачено лише одну роль із правами адміністратора — це продюсер, який вручну призначається через базу даних. Слухач, у свою чергу, може шукати й прослуховувати треки, створювати плейлисти, зберігати вподобане, залишати коментарі та оформлювати покупки. Весь контент і дії користувача обробляються у відповідних модулях: модулі управління бітами, модулі плейлистів, модулі повідомлень тощо. Завдяки такій структурі забезпечено надійне розмежування функцій, масштабованість платформи та легкість підтримки. Кодова база підтримується у репозиторії з використанням системи контролю версій Git [14].

Завантаження бітів продюсером:

Продюсер, який є адміністратором системи, після авторизації може додавати нові треки до платформи. Завантаження відбувається через спеціальну форму, де зазначається назва композиції, її жанр, настрій, BPM, тональність, супровідні теги, а також прикріплюються відповідні файли у форматах MP3, WAV та ZIP (зі стемами). Обов'язковим є завантаження обкладинки, яка потім відображається у картці треку. Система автоматично зберігає файли у відповідні директорії згідно з категорією.

Прослуховування та додавання до улюблених:

Слухачі мають можливість попередньо прослуховувати біти прямо з головної сторінки або зі сторінки каталогу. Кожен трек супроводжується кнопкою відтворення, яка активує вбудований глобальний плеєр. Завдяки цьому трек не переривається навіть при переході між сторінками. Авторизовані користувачі можуть натискати на іконку «лайку», щоб додати трек у список

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				35
Змн.	Арк.	№ докум.	Підпис	Дата		

вподобаних. Цей функціонал дозволяє швидко зберігати треки для подальшого прослуховування або купівлі.

Покупка треку та збереження у профілі:

Після прослуховування користувач може придбати обраний трек, вибравши одну з доступних ліцензій: MP3, WAV або ZIP із стемами. Після підтвердження покупки через інтегровану платіжну систему (Stripe), користувач перенаправляється на сторінку підтвердження, де може завантажити придбані файли та відповідну PDF-ліцензію. Усі покупки зберігаються в особистому кабінеті. Якщо трек був придбаний з ексклюзивною ліцензією, він автоматично зникає з каталогу, вподобаних і плейлистів інших користувачів.

Коментування треків:

Користувачі можуть залишати коментарі до треків, ділитися враженнями та думками. Коментарі зберігаються разом із зазначенням користувача, часу публікації та його аватаром. Система дозволяє редагувати або видаляти власні коментарі, а адміністратор має змогу може приховувати повідомлення у випадках порушення правил платформи.

Для візуалізації логічної структури системи та розподілу її функціональних частин було побудовано діаграму компонентів, яка наведена на рисунку 2.4

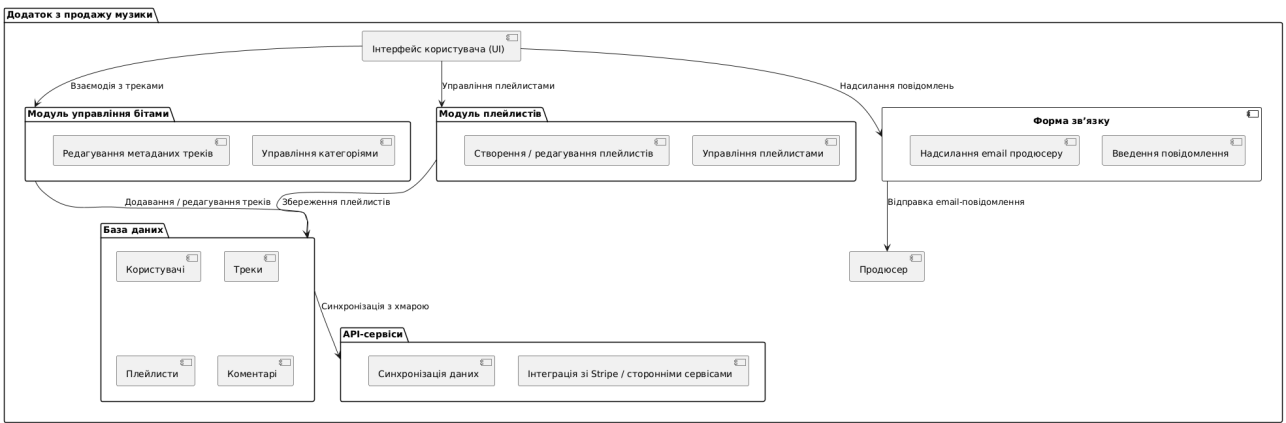


Рисунок 2.4 Діаграма компонентів

Діаграма компонентів системи «Додаток з продажу музичних треків» відображає основні частини платформи та їхню взаємодію. Основним компонентом виступає клієнтський додаток, розроблений за допомогою

бібліотеки React. Через нього користувачі отримують доступ до функціоналу, що включає перегляд треків, додавання їх до улюбленого, покупку музики, а також взаємодію з іншими користувачами через коментарі. Компонування клієнтського інтерфейсу реалізовано з використанням CSS Flexbox [20], що забезпечує адаптивну побудову елементів на сторінці.

Компонент інтерфейсу користувача підключається до бекенд-сервісу, реалізованого на Node.js. Бекенд відповідає за обробку запитів, авторизацію користувачів, валідацію введених даних, керування треками та обліковими записами. Він також надає API для обміну даними з базою даних MongoDB, де зберігається інформація про користувачів, треки, лайки, коментарі та плейлисти.

Окремий компонент відповідає за обробку платежів. Інтеграція з платіжною системою дозволяє слухачам швидко купувати треки відповідно до обраної ліцензії. Після успішної транзакції користувач отримує доступ до завантаження купленого треку, а продюсер бачить відповідне оновлення статистики у своєму кабінеті.

Ще одним важливим компонентом системи є адміністративна панель, яка забезпечує модерацію контенту, управління користувачами та перегляд аналітики. Адміністратор має можливість видаляти або схвалювати завантажені біти, переглядати активність користувачів та контролювати дотримання ліцензійних умов.

Комунікація між компонентами відбувається через REST API. Для зручності користувачів реалізована система авторизації та аутентифікації з підтримкою ролей, що дозволяє розмежовувати доступ до функцій платформи залежно від типу користувача.

Висновки до другого розділу

У другому розділі було виконано детальне проектування веб-додатку з продажу музичних треків. Розглянуто варіанти використання системи для різних ролей користувачів – слухачів, та продюсерів. Сформовано діаграму варіантів

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				37
Змн.	Арк.	№ докум.	Підпис	Дата		

використання та об'єктно-орієнтовану структуру додатку, на основі чого описано основні класи, їх методи та взаємозв'язки.

Було спроектовано базу даних, яка містить необхідні таблиці для збереження інформації про користувачів, треки, плейлисти, коментарі та лайки. Окрему увагу приділено алгоритмам взаємодії між модулями та опису роботи функціоналу додатку з точки зору користувача.

Описано структуру компонентної побудови додатку та основні функції, які забезпечують повноцінну взаємодію між користувачем і системою. У результаті, спроектована система забезпечує необхідну функціональність для публікації треків, прослуховування, взаємодії з контентом і купівлі музики у зручному інтерфейсі.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				38
Змн.	Арк.	№ докум.	Підпис	Дата		

3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ПЛАТФОРМОЮ З ПРОДАЖУ МУЗЧИНИХ ТРЕКІВ

3.1. Порядок встановлення та налаштування параметрів платформи

Встановлення та налаштування програмної системи «Додаток з продажу музичних треків» передбачає розгортання її основних компонентів — клієнтської частини на базі React, серверної частини з використанням Node.js та Express, а також бази даних MongoDB для зберігання всієї інформації про користувачів, треки, покупки, лайки та інші дії. Усі компоненти системи працюють у вебсередовищі, а доступ до функціоналу реалізовано через браузер. На рисунку 3.1 зображено діаграму розгортання додатку, що ілюструє архітектуру взаємодії між клієнтом, сервером і базою даних (рис. 3.1).

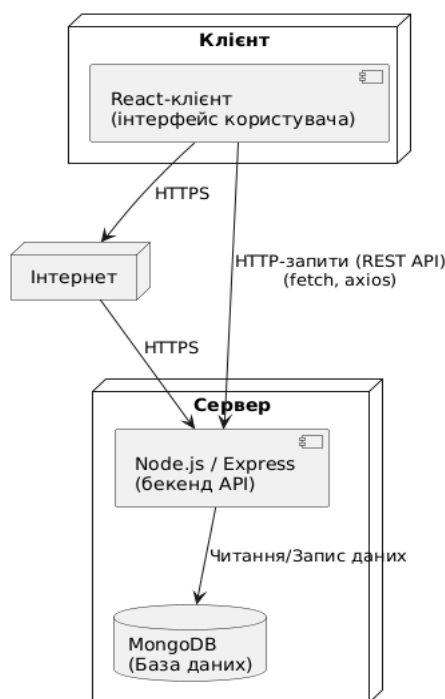


Рис. 3.1. Діаграма розгортання платформи з продажу музичних треків

Серверна частина додатку реалізована за допомогою середовища Node.js у поєднанні з фреймворком Express. Вона відповідає за обробку HTTP-запитів, автентифікацію та авторизацію користувачів, збереження й оновлення даних у базі, а також керування статичними ресурсами, зокрема аудіофайлами та зображеннями обкладинок [8].

База даних MongoDB використовується для збереження структурованої інформації, пов'язаної з треками, користувачами, лайками та купівлями. Завдяки нереляційній природі MongoDB забезпечується гнучка структура документів, що дозволяє зберігати вкладені масиви даних, наприклад, масив обраних треків або масив лайків із зазначенням часу [10].

Клієнтська частина платформи побудована з використанням бібліотеки React. Вона забезпечує інтерактивний інтерфейс користувача, який дозволяє продюсерам завантажувати треки, а слухачам — переглядати каталог, слухати демо, залишати лайки та купувати ліцензії. React-компоненти обробляють дії користувача в режимі реального часу [13], а взаємодія з сервером відбувається через асинхронні запити до REST API [18]. Завдяки використанню React Router реалізована маршрутизація без перезавантаження сторінок, що підвищує швидкодію та зручність використання системи [19]

Для налаштування серверного середовища застосовується файл .env, у якому зберігаються ключові змінні: адреса підключення до MongoDB, секретні ключі для шифрування токенів, порти для запуску клієнта й сервера, а також шляхи до папок для збереження завантажених файлів [22]. Після розгортання усіх компонентів система готова до роботи в локальному середовищі, однак її можна також адаптувати до хмарного розміщення.

Налаштування ролей користувачів відбувається безпосередньо на рівні бази даних або через відповідні API-запити. Кожному користувачеві надається роль або звичайного слухача, або продюсера, що впливає на доступний функціонал. Адміністратори мають змогу переглядати або модифікувати дані користувачів і треків, контролювати зміст і видаляти порушення, якщо така функція передбачена.

Таким чином, розгортання платформи є поетапним процесом, що охоплює налаштування середовища виконання, запуск бекенду та фронтенду, підключення до бази даних і створення базових облікових записів для тестування. Вся система побудована таким чином, щоб бути масштабованою, адаптивною та зручною для подальшого розширення.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				40
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2. Структура інтерфейсу веб-платформи

Веб-платформа «Додаток з продажу музичних треків» реалізована з використанням сучасних інтерфейсних технологій, що забезпечують зручність взаємодії як для продюсерів, так і для слухачів. Інтерфейс побудований на базі React — JavaScript-бібліотеки, що дозволяє створювати динамічні та гнучкі користувацькі інтерфейси [6]. Основна увага при розробці була приділена інтуїтивності навігації, швидкості відгуку та наочному відображенню контенту.

Після запуску додатку користувач потрапляє на головну сторінку, де відображаються представлені треки. Кожен трек виводиться у вигляді окремої картки з обкладинкою, назвою, тривалістю, тегами, BPM, а також ціною. У користувача є можливість негайно прослухати демо-версію треку без необхідності переходу на іншу сторінку [12]. Завдяки вбудованому аудіоплеєру з підтримкою глобального контексту, відтворення треку не припиняється навіть при навігації між сторінками додатку. (рис. 3.2)

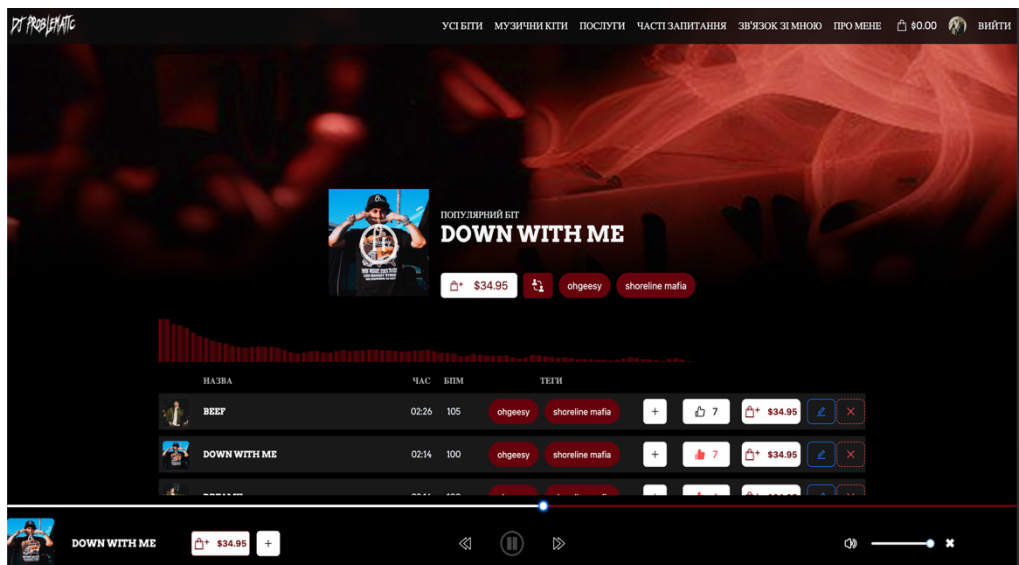


Рис. 3.2. Головна сторінка платформи

Для повноцінної взаємодії з платформою, зокрема для можливості вподобувати або купувати треки, користувач має авторизуватись у системі. Процес авторизації реалізований через окрему сторінку, де потрібно ввести адресу електронної пошти та пароль, вказані при реєстрації. У випадку помилкового введення даних система повідомляє про помилку, а при успішній

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				
Змн.	Арк.	№ докум.	Підпис	Дата		41

авторизації — перенаправляє користувача на головну сторінку платформи. Також передбачено можливість швидкого переходу до реєстрації, якщо користувач ще не має облікового запису.

Після входу до системи для користувача активуються персоналізовані функції, зокрема можливість зберігати вподобані треки, створювати та редагувати власні плейлисти, оформлювати покупки й переглядати історію придбаних композицій. Крім того, інтерфейс адаптується до статусу користувача: у хедері з’являється аватарка з доступом до профілю, а деякі кнопки змінюють свій стан — наприклад, кнопка «Увійти» замінюється на ім’я користувача. Таким чином, авторизація є обов’язковим кроком для розблокування повного функціоналу системи та збереження індивідуального прогресу користувача в межах платформи (рис. 3.3).

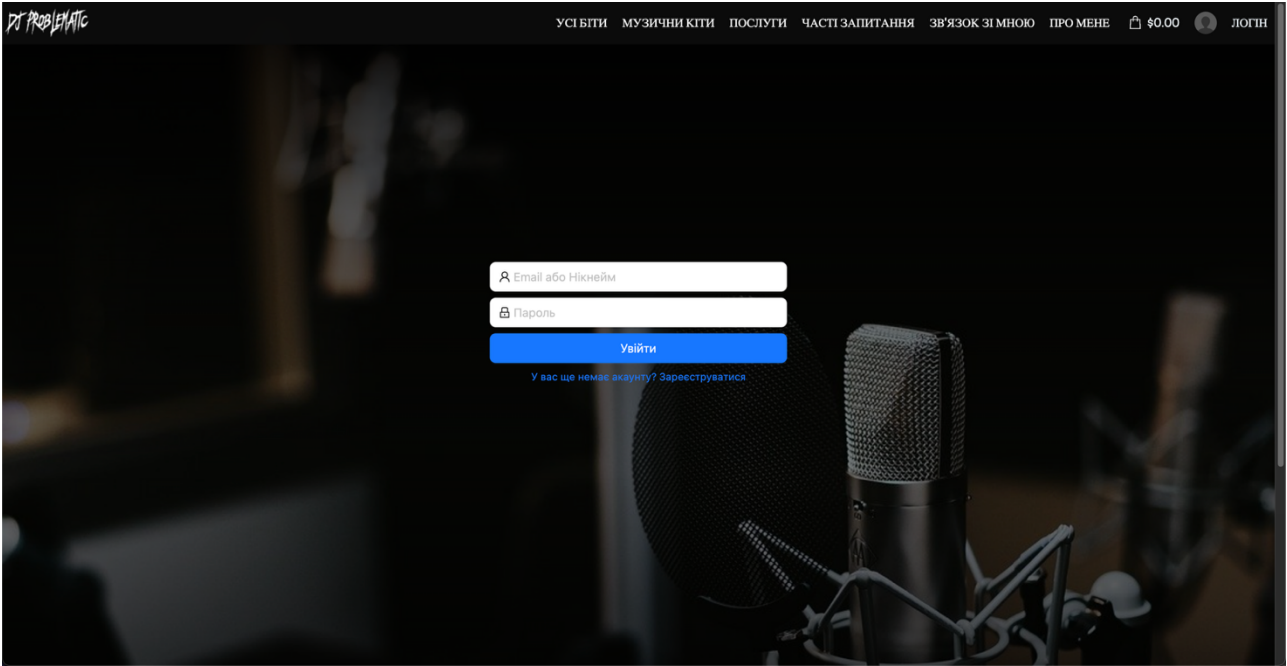


Рис. 3.3. Сторінка авторизації

Окрему увагу в інтерфейсі приділено профілю користувача. Після авторизації користувач має доступ до особистого кабінету, де відображаються його аватар, ім’я, адреса електронної пошти, пароль. Звідси також можна змінити аватар, оновити персональні дані та переглянути історію взаємодії з платформою. (рис. 3.4)

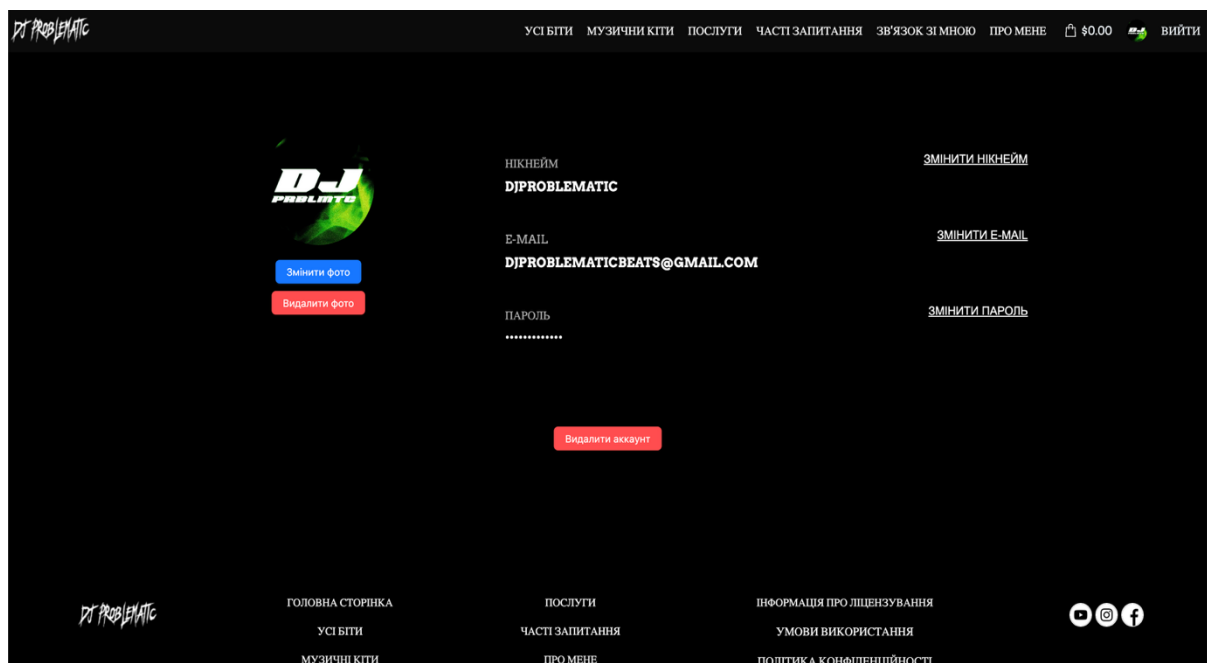


Рис. 3.4. Сторінка профілю користувача

Користувач має змогу вподобати трек, натиснувши на відповідну іконку лайку на його картці на головній сторінці, або на сторінці біта. Усі вподобані треки автоматично додаються у збережені біти користувача у вигляді окремого списку, який у подальшому можна фільтрувати за жанром, ключем, настроєм, темпом, і також за популярністю. Такий функціонал значно полегшує подальший пошук релевантної музики, особливо для постійних клієнтів платформи

Сторінка збережених бітів адаптована для швидкої навігації: треки згруповані в таблицю зі зручною структурою, де кожен біт можна миттєво прослухати, додати до кошика або перемістити в один із власних плейлистів. Завдяки цьому користувач не лише зберігає улюблені композиції, а й має змогу ефективно працювати з ними в рамках подальших креативних або комерційних завдань (рис. 3.5).

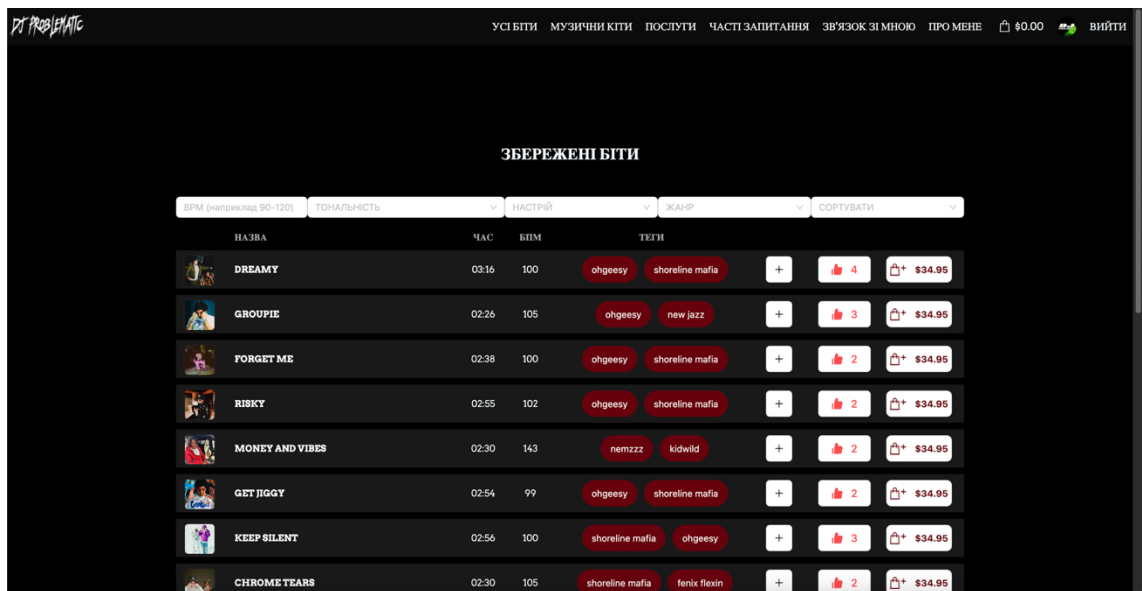


Рис. 3.5. Сторінка вподобаних треків

Продюсер після авторизації отримує доступ до окремого функціоналу. У профілі з'являється можливість завантаження нових треків через спеціальну форму, де необхідно вказати назву композиції, жанр, тональність, настрій, теги, BPM, прикріпити обкладинку, а також додати три файли: MP3-версію, WAV-версію та ZIP-архів зі стемами. Після успішного завантаження трек з'являється у списку всіх бітів та стає доступним для інших користувачів (рис. 3.6).

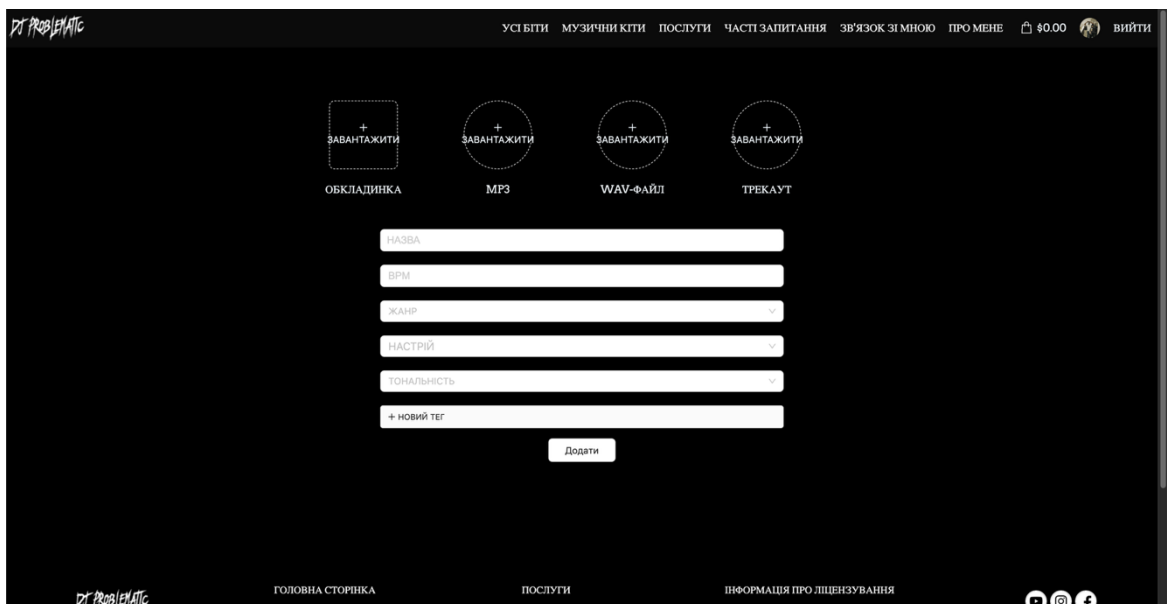


Рис. 3.6. Сторінка завантаження треку

Окрім завантаження, продюсери можуть редагувати інформацію про власні треки або повністю їх видаляти. Це дозволяє оперативно оновлювати

контент, змінювати ціну або обкладинку, а також видаляти неактуальні або продані треки з відкритого доступу (рис 3.7).

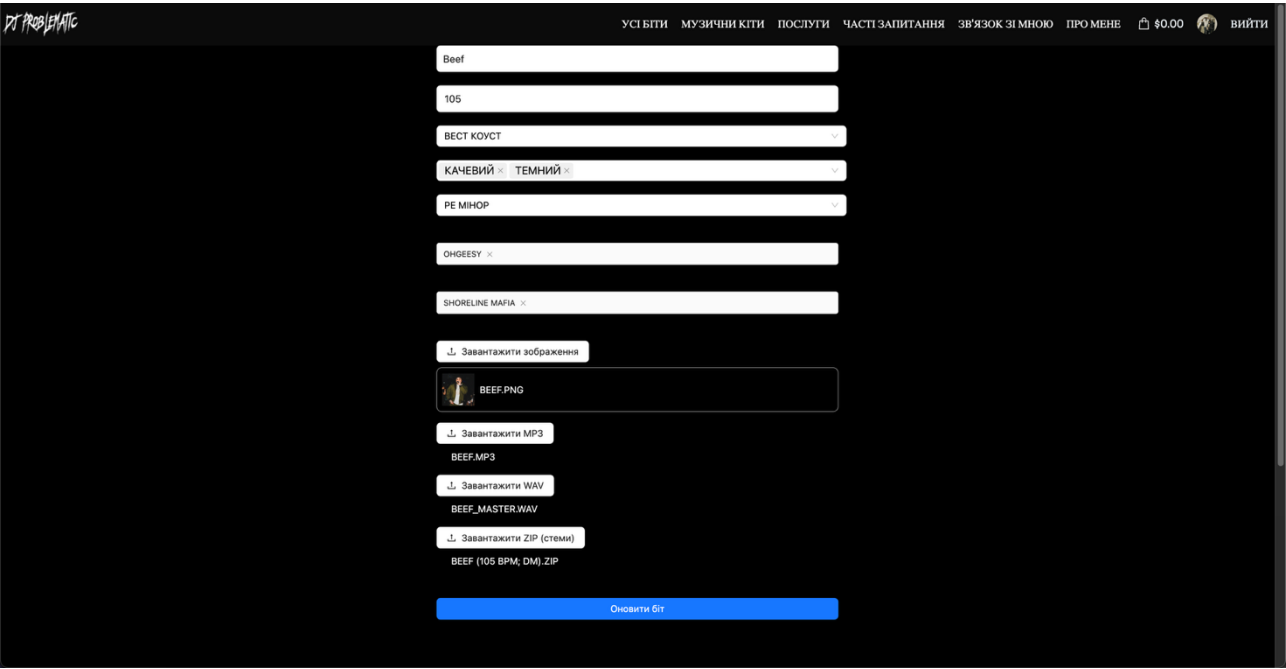


Рис. 3.7. Сторінка редагування треку

Процес купівлі треку максимально спрощено: користувач обирає трек, натискає кнопку придбання, після чого відкривається модальне вікно з вибором ліцензії. Усі придбані треки також зберігаються у профілі користувача для подальшого доступу (рис. 3.8).

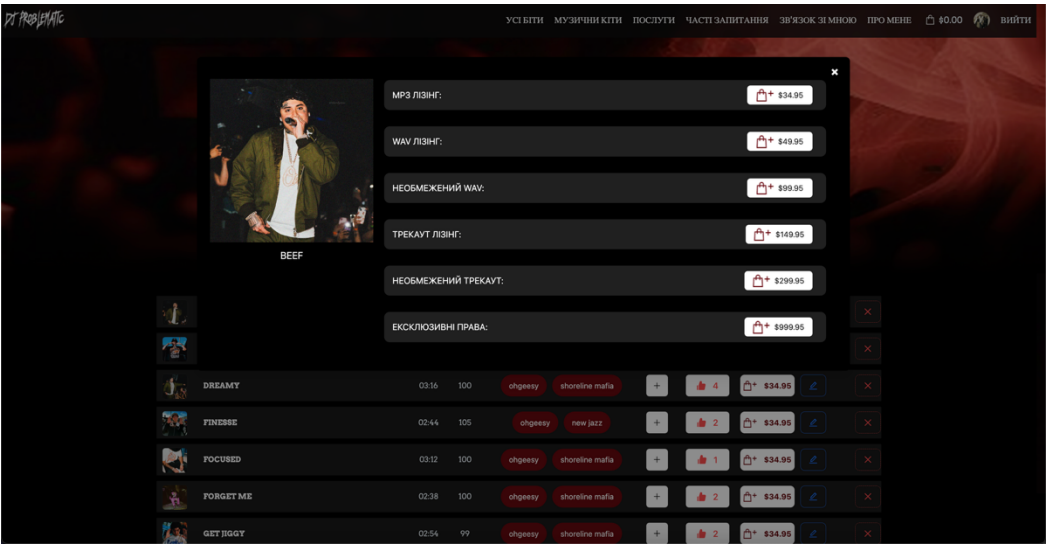


Рис. 3.8. Інтерфейс купівлі треку

Після того як користувач обрав ліцензію, він натискає на кнопку купівлі, і після цього біт потрапляє у кошик (рис. 3.9)

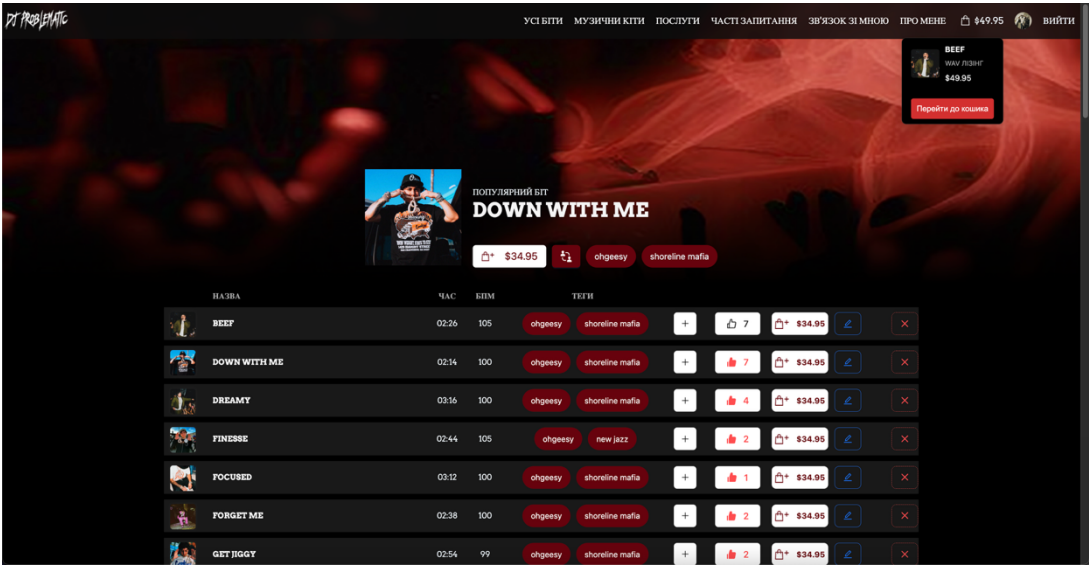


Рис. 3.9. Інтерфейс випадаючого меню кошика

Кнопка «Перейти до кошика» переводить користувача на сторінку його кошика де здійснюється оплата (рис. 3.10)

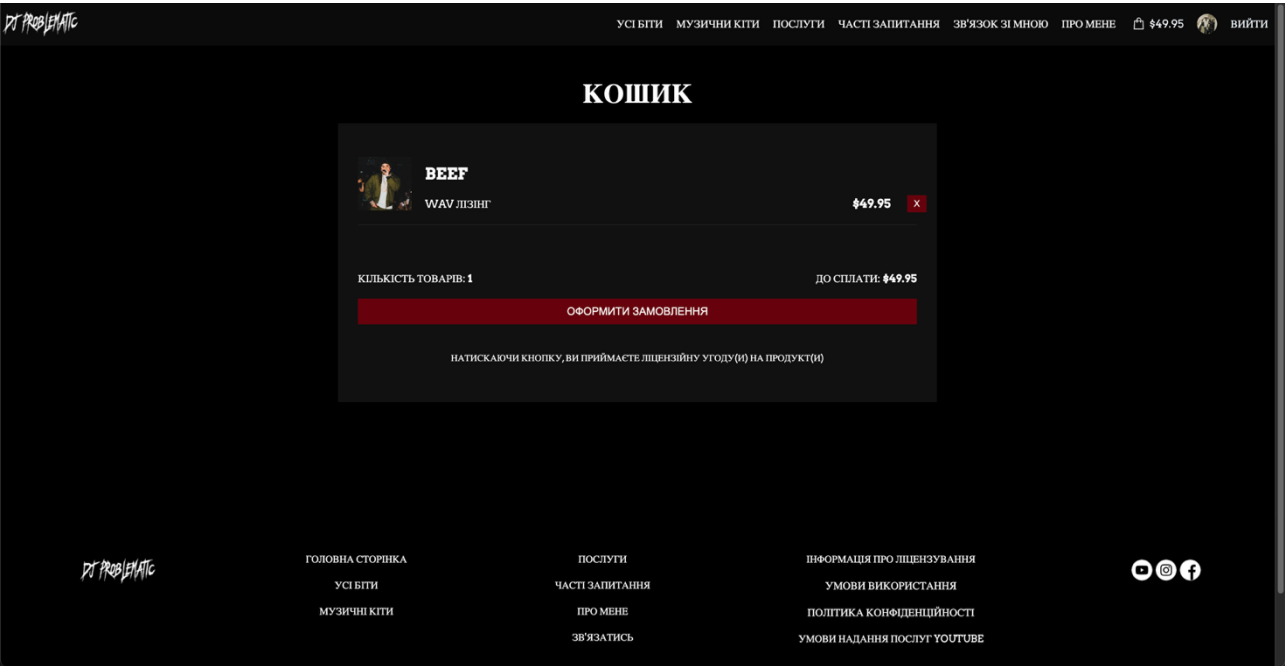


Рис. 3.10. Сторінка кошика

Після натискання кнопки «Оформити замовлення» користувач потрапляє на сторінку оплати, де може використати платіжну систему Stripe для сплати товару.

Інтерфейс сторінки побудовано таким чином, щоб мінімізувати кількість дій для завершення транзакції — користувачеві потрібно лише ввести платіжні реквізити та підтвердити оплату. Завдяки інтеграції з Stripe процес оплати відбувається швидко, безпечно та з урахуванням усіх актуальних стандартів захисту персональних даних і фінансових операцій. У разі успішної транзакції користувач автоматично перенаправляється на сторінку підтвердження покупки (рис. 3.11)

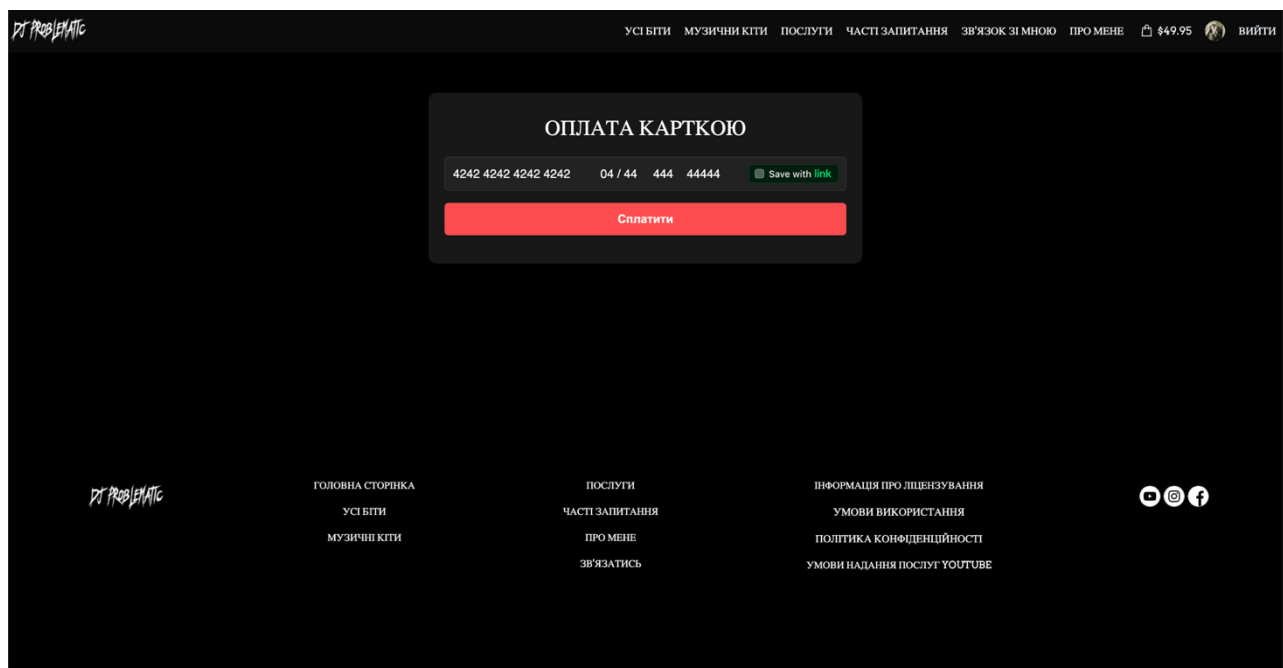


Рис. 3.11. Сторінка оплати

Після успішного підтвердження оплати користувач автоматично перенаправляється на сторінку повідомлення про успішне замовлення. На цій сторінці відображається придбаний трек із можливістю негайного завантаження у відповідному форматі — MP3, WAV або архів зі стемами, залежно від обраної ліцензії. Крім цього, для кожної покупки доступна кнопка завантаження офіційної ліцензії у форматі PDF, яка автоматично генерується системою на основі даних про покупця, трек і тип ліцензії. Інтерфейс сторінки також містить кнопку «Переглянути усі куплені біти», яка здійснює перехід до розділу профілю користувача з повним списком придбаних композицій. У цьому розділі користувач має змогу переглядати історію своїх замовлень, а також повторно завантажувати придбані треки й ліцензії. Якщо трек було придбано за умовами

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				47
Змн.	Арк.	№ докум.	Підпис	Дата		

ексклюзивної ліцензії, він автоматично видаляється з публічного каталогу, списку вподобаних, плейлистів та недоступний для подальшого придбання іншими користувачами (рис. 3.12).

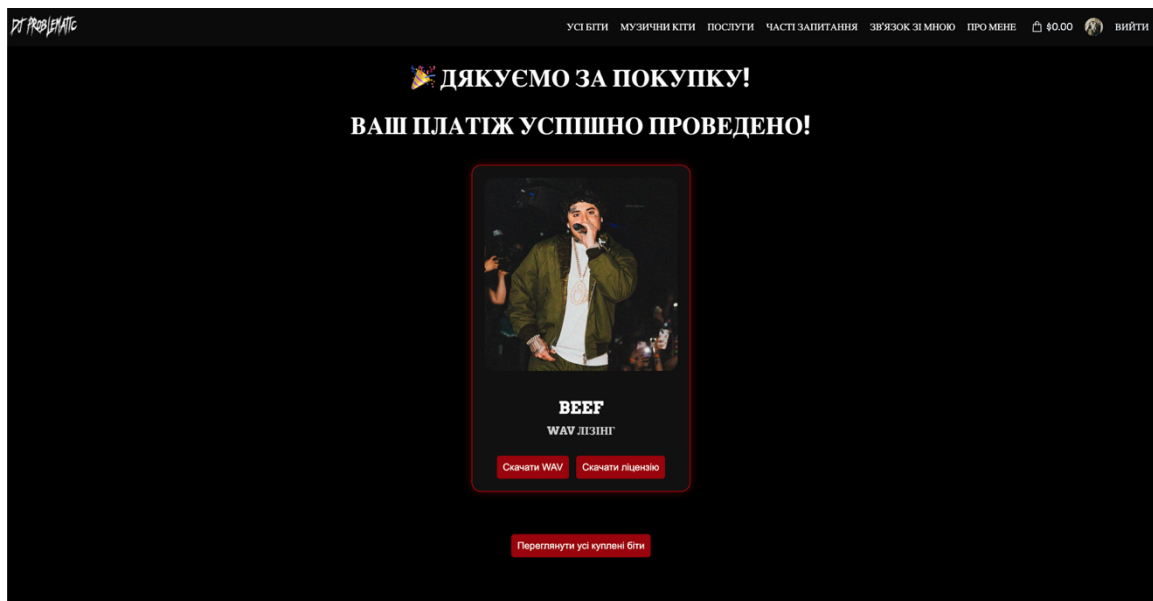


Рис. 3.12. Сторінка успішної оплати

Окремо варто відзначити сторінку окремого біта, яка відкривається після натискання на картку треку. На цій сторінці користувач може ознайомитися з розгорнутою інформацією: повною назвою, BPM, жанром, ключем, настроєм, обкладинкою, кількістю лайків, цінами на різні типи ліцензій. Також реалізована функція залишення коментарів, яка дозволяє слухачам ділитися думками або відгуками про композицію. Зареєстровані користувачі можуть редагувати або видаляти власні коментарі, а адміністратор — модерувати вміст у разі порушень правил. Такий функціонал сприяє створенню інтерактивної спільноти та зворотного зв'язку для продюсерів.

Програвач, який вбудовано, дозволяє не лише слухати демо-фрагмент у високій якості, а й плавно перемикається між треками без оновлення інтерфейсу, що особливо зручно при ознайомленні з великим обсягом музичного матеріалу (рис. 3.13).

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				48
Змн.	Арк.	№ докум.	Підпис	Дата		

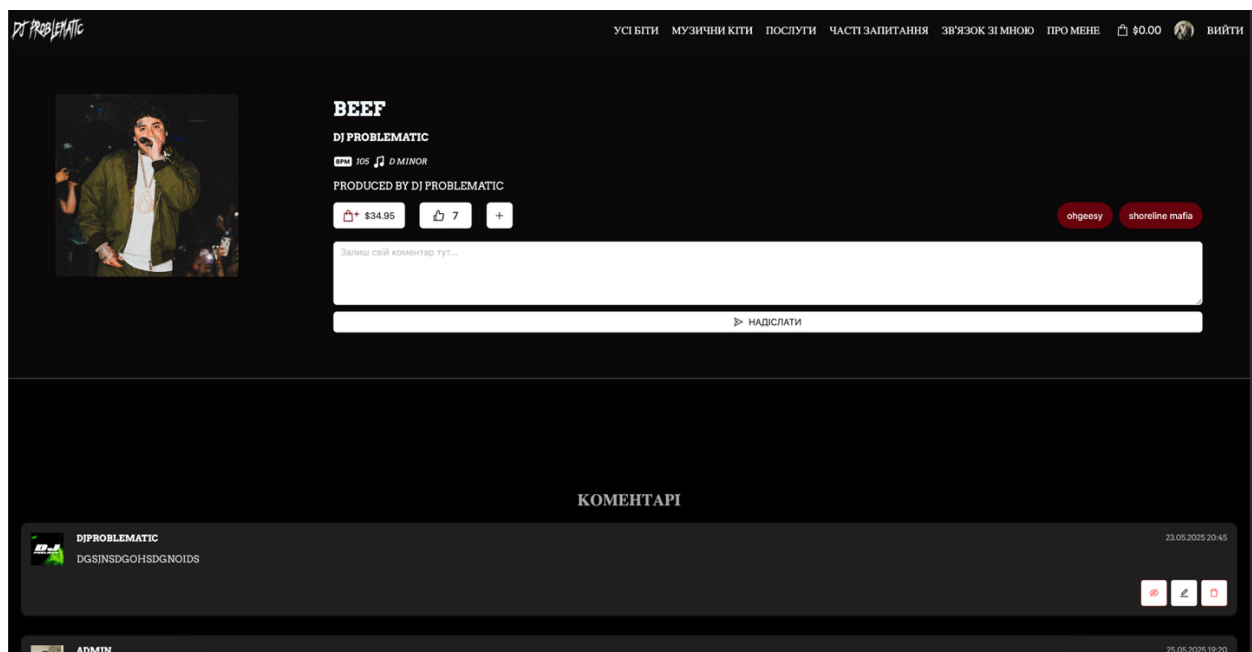


Рис. 3.13. Сторінка біта з розгорнутою інформацією та коментарями

Сторінка «Усі біти» виконує роль повноцінного каталогу музичних композицій, доступних для прослуховування та придбання. Інтерфейс реалізовано у вигляді таблиці або ґрид-компонентів, кожен з яких містить інформацію про біт: обкладинку, назву, тривалість, жанр, BPM, настрій, тональність та ціну. Центральне місце в цій сторінці займає система фільтрації, що дозволяє користувачеві швидко знайти треки відповідно до власних потреб або музичних вподобань.

Фільтрація реалізована за п'ятьма основними параметрами: темпом (BPM), тональністю, жанром, настроєм і популярністю. Користувач може обрати значення для кожного з параметрів із випадających списків або тегів. Зміна значень у фільтрах автоматично оновлює результати без перезавантаження сторінки, що забезпечує швидку взаємодію. Сортвання за популярністю ґрунтується на кількості вподобань (лайків), отриманих кожним треком, що дозволяє виділити найбільш актуальні та прослуховувані композиції на платформі.

Кожен трек можна одразу прослухати через вбудований аудіоплеєр, а також додати до списку обраного або перейти на його індивідуальну сторінку для перегляду детальнішої інформації. Завдяки зручній і зрозумілій структурі

сторінка «Усі біти» є ключовим елементом платформи, який забезпечує легкий доступ до всього доступного контенту й сприяє швидкому вибору треків для покупки або подальшої взаємодії (рис. 3.14).

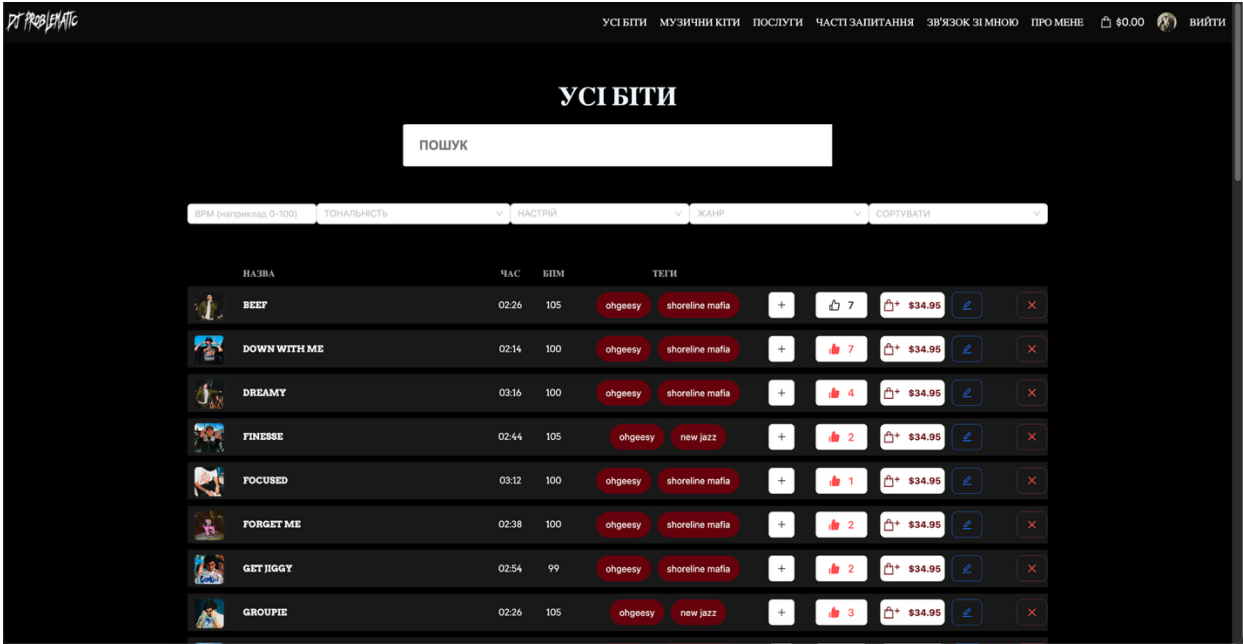


Рис. 3.14. Сторінка каталогу треків з системою фільтрації

Користувач також має можливість організовувати вподобані або потенційно цікаві біти у власні плейлисти. Цей функціонал реалізовано таким чином, щоб додавання треку до плейлисту було доступним із будь-якої сторінки, де відображається трек: з головної, зі сторінки каталогу «Усі біти», а також зі сторінки самого треку. Біля кожного треку розміщена спеціальна кнопка або іконка, яка відкриває модальне вікно з переліком існуючих плейлистів користувача. У цьому вікні можна обрати один або кілька плейлистів, до яких буде додано біт, або створити новий плейлист безпосередньо під час додавання.

Ця гнучка система дозволяє користувачу зручно групувати біти за різними критеріями: жанром, настроєм, майбутнім призначенням (наприклад, для виступів або релізів), типом ліцензії тощо. Плейлисти зберігаються у профілі користувача та доступні для перегляду і подальшої взаємодії. Таким чином, функція додавання до плейлистів сприяє персоналізації досвіду взаємодії з платформою та полегшує навігацію по улюблених треках навіть за умов великого обсягу доступного контенту (рис. 3.15).

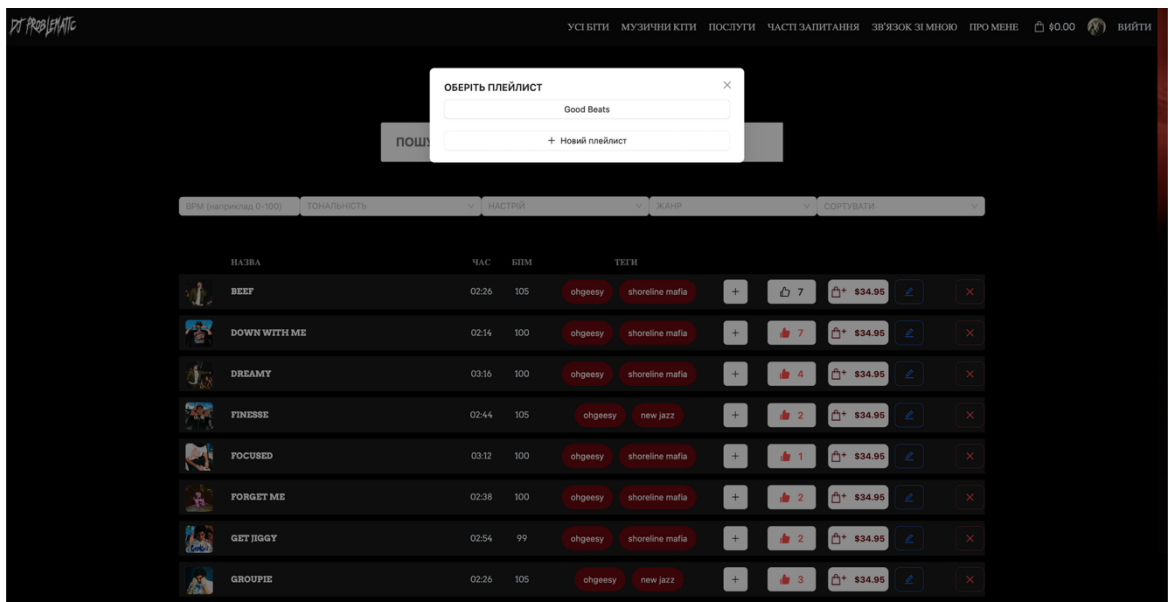


Рис. 3.15. Модальне вікно додавання біта до плейлисту

Усі створені плейлисти користувача згруповані на окремій сторінці «Мої плейлисти», доступ до якої здійснюється через навігаційне меню профілю. На цій сторінці користувач бачить повний перелік власних плейлистів із можливістю відкриття кожного з них для перегляду вмісту (рис. 3.16).

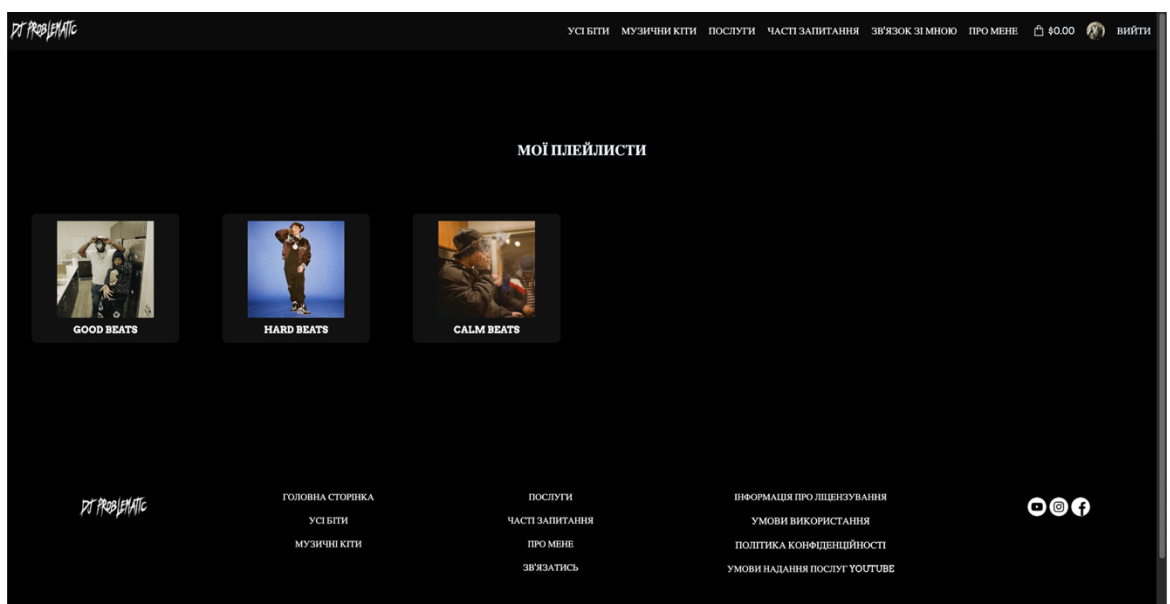


Рис. 3.16. Сторінка плейлистів користувача

Після переходу у конкретний плейлист відкривається таблиця з усіма бітами, що були додані до нього раніше. Для кожного біта в таблиці доступні ті самі дії, що й у загальному каталозі: прослуховування, додавання в кошик, перегляд детальної інформації, а також можливість видалення з плейлисту.

Користувач має змогу перейменувати плейлист або змінити його обкладинку, що дозволяє краще структурувати музичну бібліотеку відповідно до особистих вподобань. Крім того, передбачено можливість видалення плейлисту разом із усіма зв'язками між ним та бітами. Однак самі треки при цьому не видаляються з профілю користувача або загального каталогу, що гарантує збереження всієї купленої або вподобаної музики.

Завдяки реалізації сторінки плейлистів користувач отримує повноцінний інструмент для організації власної колекції, який особливо зручний при роботі з великою кількістю треків. Це створює можливість не лише зберігати улюблені композиції, а й формувати добірки для прослуховування у певному настрої, стилі або під конкретні задачі (рис. 3.17).

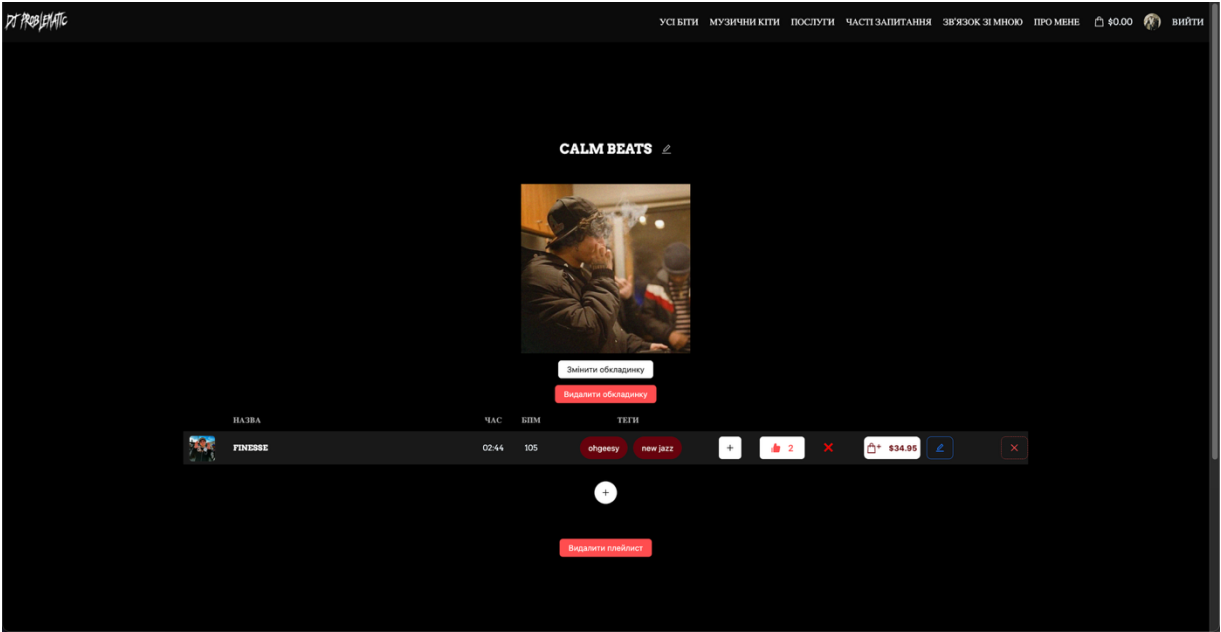


Рис. 3.17. Сторінка плейлиста користувача

Таким чином, інтерфейс веб-платформи розроблено з урахуванням зручності та послідовності дій як для продюсерів, так і для слухачів. Всі елементи системи взаємопов'язані та дозволяють виконувати повний цикл взаємодії з контентом — від перегляду й прослуховування до додавання в обране, оформлення покупки та завантаження ліцензії. Гнучкість функціоналу, можливість організації плейлистів і збереження персональних налаштувань сприяють формуванню індивідуального досвіду кожного користувача.

Завдяки цим рішенням система не лише виконує свої функції, а й створює приємне враження від роботи з нею, що є ключовим фактором для подальшого залучення та утримання аудиторії.

3.3. Порядок роботи з веб-платформою

Робота з веб-платформою передбачає кілька основних сценаріїв взаємодії користувачів із системою. Залежно від ролі — слухача або продюсера — доступний функціонал відрізняється. Усі користувачі платформи проходять обов'язкову процедуру реєстрації, після чого отримують доступ до персонального кабінету. Роль продюсера (власника платформи), на відміну від звичайних слухачів, призначається вручну через базу даних і не доступна для публічної реєстрації. У реалізованій системі адміністратор і продюсер є однією й тією самою особою — тобто саме розробник/власник проєкту має доступ до повного керування контентом.

Сценарій 1: Авторизація та персоналізація інтерфейсу

Після відкриття платформи незареєстрований користувач має змогу переглядати треки, ознайомлюватися з інформацією про них і слухати демо-версії. Для здійснення будь-яких персоналізованих дій (лайків, створення плейлистів, оформлення покупок) обов'язковою є авторизація. На сторінці авторизації потрібно ввести адресу електронної пошти та пароль. У разі помилки користувач отримає відповідне повідомлення, а у випадку успішного входу — буде перенаправлений на головну сторінку з розблокованим функціоналом. Інтерфейс адаптується: з'являється стандартний аватар, меню профілю, а також можливість взаємодії з контентом на персоналізованому рівні.

Сценарій 2: Редагування профілю користувача

Після авторизації користувач отримує доступ до персонального кабінету, в якому може змінювати особисту інформацію. Це включає редагування імені, електронної пошти, пароля, зміну аватарки та видалення аккаунту. У верхній частині профілю розміщено кнопку для зміни аватару: при натисканні

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				53
Змн.	Арк.	№ докум.	Підпис	Дата		

відкривається форма завантаження нового зображення, після чого аватар автоматично оновлюється на всіх сторінках платформи.

Редагування імені та електронної пошти виконується через окрему форму, де користувач вводить нові дані та підтверджує зміни. У разі зміни пароля необхідно ввести поточний пароль, а потім новий — це дозволяє захистити обліковий запис від несанкціонованого доступу. Усі зміни зберігаються в базі даних, а інтерфейс оновлюється без потреби перезавантаження сторінки.

Цей сценарій забезпечує користувачу контроль над особистою інформацією та дозволяє підтримувати її в актуальному стані.

Усі дії користувача, що передбачають персоналізовану взаємодію з платформою — збереження треків, створення плейлистів, оформлення покупок, залишення коментарів, редагування профілю — потребують обов'язкової авторизації. Тому сценарій авторизації є базовим і попереднім для всіх наступних дій. Надалі, описуючи кожен сценарій, вважаємо, що користувач вже успішно увійшов у систему.

Сценарій 3: Прослуховування біту з головної сторінки або зі сторінки «Усі біти»

Один із ключових сценаріїв взаємодії з платформою — це попереднє прослуховування треку. Веб-додаток побудований таким чином, що користувач одразу після входу або навіть без авторизації може знайомитися з доступними бітами на головній сторінці або в розділі «Усі біти».

Кожен трек виводиться у вигляді окремої картки, що містить назву, обкладинку, жанр, BPM, ключ, настрій і ціну. Біля кожного треку розміщено кнопку «Play» (іконка відтворення), натискання на яку запускає вбудований аудіоплеєр. Важливою особливістю реалізації є використання глобального контексту плеєра — завдяки цьому трек продовжує відтворюватися навіть при переході на інші сторінки (наприклад, у профіль або на сторінку конкретного біта).

Після початку відтворення активується візуальний індикатор — звукова анімація або прогрес-бар — що дозволяє користувачу бачити, який трек зараз

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				54
Змн.	Арк.	№ докум.	Підпис	Дата		

грає. У випадку повторного натискання на кнопку відтворення — трек зупиняється або запускається знову.

Таким чином, механізм прослуховування забезпечує швидкий і зручний доступ до контенту, що значно спрощує процес ознайомлення з треком перед його купівлею або додаванням у вподобані чи плейлист.

Сценарій 4: Збереження треку у вподобане

Функціонал збереження треків у вподобане на платформі виконує роль персональної музичної добірки для користувача. Це дає змогу швидко позначати цікаві біти, до яких можна буде повернутися згодом для повторного прослуховування чи подальшої покупки. Щоб скористатися цією функцією, користувач повинен бути авторизований у системі. Після входу на платформу біля кожного треку, незалежно від того, чи це головна сторінка, каталог «Усі біти» або сторінка конкретного треку, з'являється іконка у вигляді сердечка. Один клік по ній додає обраний трек до списку вподобаних, а сама іконка змінює вигляд, вказуючи на те, що збереження відбулося успішно.

Усі додані до вподобаних композиції автоматично відображаються в окремому розділі «Мої вподобані», який доступний через профіль користувача. У цьому розділі реалізовано зручну табличну форму відображення збережених треків, а також систему фільтрів за жанром, настроєм, BPM, ключем та популярністю, що дозволяє швидко знайти потрібну композицію серед вподобаних.

Користувач може безпосередньо з цього розділу прослуховувати біти через глобальний плеєр, додавати обрані треки до плейлистів, надсилати їх у кошик для оформлення покупки або видаляти з добірки, якщо інтерес до них зник. Якщо якийсь трек було придбано іншим користувачем із ексклюзивною ліцензією, система автоматично видаляє його з усіх вподобаних, оскільки такий контент більше недоступний для повторного прослуховування або покупки. Це забезпечує актуальність збережених треків та підтримує логіку ліцензування на платформі.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Сценарій 5: Створення плейлисту користувачем та додавання треків

Після авторизації на платформі як слухач користувач отримує можливість створювати персоналізовані плейлисти для збереження та організації вподобаних треків. Цей механізм дозволяє об'єднувати композиції за різними критеріями — жанром, настроєм, BPM або особистими вподобаннями.

Взаємодія з функціоналом починається з перегляду треків на головній сторінці або у розділі каталогу. Біля кожного треку передбачена кнопка «Додати до плейлисту». Після її натискання відкривається модальне вікно, де користувач може або обрати один із раніше створених плейлистів, або створити новий, вказавши його назву й завантаживши обкладинку. Після підтвердження вибору трек миттєво додається до зазначеного плейлисту.

Усі створені добірки зберігаються в розділі «Мої плейлисти», доступному через меню профілю. Тут відображається список плейлистів користувача у вигляді мініатюр із назвою та обкладинкою. При натисканні на обраний плейлист відкривається його вміст — табличний список треків, які до нього входять.

Усередині одного плейлисту користувач має змогу здійснювати базові дії: відтворювати треки, переглядати їхню деталізацію, додавати окремі біти до кошика для покупки, а також видаляти непотрібні треки з добірки. Крім того, доступна можливість перейменування плейлисту та зміни його обкладинки. За потреби, користувач також може повністю видалити плейлист — при цьому самі треки залишаються в каталозі та профілі.

Цей сценарій забезпечує слухачеві зручний спосіб зберігати, впорядковувати та структурувати улюблену музику на платформі. Особливо корисним він є для користувачів, які планують пізніше повернутися до певних композицій або придбати декілька треків одночасно.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Сценарій 6: Залишення коментаря до треку

Функціонал коментування на платформі виконує важливу роль у забезпеченні взаємодії між слухачами та продюсерами. Він дає змогу користувачам ділитися враженнями від прослуханих треків, залишати відгуки та створювати активне середовище спілкування. Для того щоб залишити коментар, користувач повинен бути авторизований. Після входу до системи він може перейти на сторінку будь-якого треку, де у нижній частині відображається розділ коментарів. Тут користувач бачить уже наявні повідомлення від інших слухачів і має доступ до форми для введення власного тексту.

Після написання коментаря достатньо натиснути кнопку публікації, і повідомлення миттєво з'явиться у загальному потоці з іменем користувача та часом додавання. Усі коментарі автоматично зберігаються в базі даних і доступні для перегляду усім користувачам платформи. Автор коментаря має можливість редагувати або видалити своє повідомлення, а також переглядати реакцію спільноти, якщо реалізована система відповідей або вподобань.

Продюсер, який водночас виконує роль адміністратора, має змогу модерувати коментарі, видаляти порушення або блокувати небажані повідомлення для підтримки комфортного середовища. Такий підхід дозволяє слухачам бути активною частиною платформи, а продюсерам — оперативно реагувати на відгуки та покращувати якість контенту відповідно до очікувань аудиторії.

Сценарій 7: Придбання треку з вибором ліцензії та оплатою

Після прослуховування треку користувач може вирішити його придбати. Для цього необхідна авторизація. Якщо користувач ще не увійшов до системи, при натисканні кнопки «Купити» система запропонує перейти на сторінку авторизації. Після успішного входу відкривається повна функціональність покупки.

На картці біта або на його окремій сторінці користувач натискає кнопку «Купити», після чого з'являється модальне вікно з переліком доступних ліцензій. Кожна ліцензія (MP3, WAV або Stems ZIP) супроводжується описом прав

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				57
Змн.	Арк.	№ докум.	Підпис	Дата		

використання та відповідною ціною. Користувач обирає потрібну ліцензію та натискає кнопку «Додати в кошик». Після цього у випадяючому меню з'являється оновлений кошик, де можна побачити всі додані треки.

Натиснувши «Перейти до кошика», користувач потрапляє на сторінку оформлення замовлення. Тут відображено список обраних бітів, ціни та загальну суму до сплати. Натискання кнопки «Оформити замовлення» переводить користувача на сторінку оплати за допомогою Stripe, де потрібно ввести платіжні реквізити.

Після успішної транзакції система автоматично перенаправляє користувача на сторінку підтвердження покупки. На цій сторінці відображається список нещодавно придбаних треків, з можливістю негайного завантаження файлів у відповідному форматі, а також завантаження автоматично згенерованої PDF-ліцензії.

Додатково присутня кнопка «Переглянути усі куплені біти», яка відкриває розділ «Придбані треки» у профілі користувача. Там зберігається вся історія покупок із можливістю повторного завантаження треків і ліцензій.

Якщо трек було придбано з ексклюзивною ліцензією, він автоматично зникає з публічного каталогу, списку вподобаних, плейлистів інших користувачів і більше не доступний для подальшого придбання.

Сценарій 8: Надсилання повідомлення через сторінку «Зв'язок зі мною»

Функціонал зворотного зв'язку на платформі реалізований через окрему сторінку «Зв'язок зі мною», яка доступна в хедері сайту, а також дублюється у нижній частині головної сторінки. Цей інтерфейс дозволяє слухачам напряму звертатися до продюсера, який водночас виконує функції адміністратора. На сторінці розміщено форму, яка включає поля для введення імені, адреси електронної пошти, теми звернення та самого повідомлення. Після заповнення всіх обов'язкових полів користувач натискає кнопку надсилання, після чого система відправляє введені дані на заздалегідь вказану електронну пошту адміністратора.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				58
Змн.	Арк.	№ докум.	Підпис	Дата		

Після успішної відправки на екрані з'являється підтвердження про доставлення повідомлення. Така форма зворотного зв'язку дозволяє слухачам задавати запитання, залишати відгуки, повідомляти про технічні проблеми або звертатися з приводу індивідуальної співпраці. Продюсер, у свою чергу, має змогу оперативно реагувати на звернення, не залучаючи сторонні месенджери, що робить комунікацію ефективною і зручною для обох сторін.

Сценарій 9: Завантаження треку продюсером

Після авторизації в системі користувач, який має роль продюсера (що водночас виконує функції адміністратора), отримує доступ до функціоналу управління контентом. У його профілі стає активною можливість завантаження нових треків через спеціальну форму. У цій формі потрібно ввести назву композиції, вибрати жанр, вказати BPM, тональність, настрій, а також додати супровідні теги. Обов'язковим є завантаження обкладинки, яка буде відображатися в картці треку на платформі. До кожного треку прикріплюються три типи аудіофайлів — у форматах MP3, WAV та ZIP-архів, що містить окремі стемми.

Після заповнення форми та натискання кнопки «Опублікувати», система перевіряє правильність заповнення і формат файлів, і в разі успішної валідації трек додається до каталогу платформи, стаючи доступним для перегляду та взаємодії іншим користувачам. Якщо ж у формі пропущено обов'язкове поле або завантажено файл у неправильному форматі, користувач отримає повідомлення про помилку.

Одразу після завантаження треку продюсер може перейти до редагування композиції — змінити її ціну, оновити жанр, замінити теги або завантажити нову обкладинку. Детальніше функціонал редагування описується в наступному сценарії.

Сценарій 10: Редагування або видалення треку продюсером

Функціонал редагування або видалення треку доступний виключно користувачам із роллю продюсера. У розділі «Мої треки» продюсер бачить перелік усіх раніше завантажених бітів. Біля кожного треку розміщені відповідні

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				59
Змн.	Арк.	№ докум.	Підпис	Дата		

кнопки керування — «Редагувати» та «Видалити». Натиснувши кнопку редагування, користувач переходить до форми, схожої на початкову форму додавання треку, однак вона вже містить заповнені поля з поточними даними композиції. У межах цієї форми доступне оновлення таких параметрів, як назва треку, жанр, BPM, настрій, ключ, теги, ціна, обкладинка або прикріплені аудіофайли. Після підтвердження змін оновлена інформація одразу відображається як у загальному каталозі, так і на сторінці конкретного треку. Якщо ж продюсер натискає кнопку видалення, трек повністю вилучається з бази даних і автоматично зникає з усіх розділів, включно з плейлистами, кошиками й вподобаними інших користувачів. Такий підхід забезпечує повний контроль над контентом з боку автора.

Таким чином, описані сценарії взаємодії користувачів з веб-платформою охоплюють повний життєвий цикл роботи з музичним контентом — від ознайомлення та прослуховування треку до його придбання, збереження у вподобане або плейлист, а також залишення коментаря. Реалізований функціонал повністю забезпечує потреби як звичайного слухача, так і продюсера, який має можливість публікувати й керувати треками через зручний інтерфейс.

Платформа адаптована до різних ролей користувачів, дозволяє здійснювати персоналізовану навігацію, підтримує безпечну оплату, а також зберігає історію взаємодії з контентом. Особливу увагу приділено простоті інтерфейсу та логічності взаємодії, що дозволяє будь-якому користувачу інтуїтивно працювати з системою без попередньої підготовки.

Загалом, розділ демонструє, що всі заплановані функції успішно реалізовані, а веб-додаток є завершеним, стабільним і готовим до повноцінного використання.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до третього розділу

У третьому розділі було проведено детальний аналіз реалізованого інтерфейсу веб-платформи з продажу музичних треків, розглянуто порядок її встановлення, налаштування та основні сценарії взаємодії користувачів із системою. Сформовано повну картину функціонування додатку як з боку слухачів, так і з боку продюсера (адміністратора).

Визначено ключові елементи інтерфейсу: сторінка авторизації, головна сторінка з каталогом треків, особистий кабінет, сторінки вподобаних композицій, кошика, плейлистів та куплених бітів. Окремо описано функціонал сторінки треку з можливістю залишення коментарів. Продемонстровано, як користувач може виконувати основні дії — прослуховування, збереження у вподобане, додавання до плейлистів, придбання треків та завантаження ліцензій.

Особливу увагу приділено опису ролей користувачів і порядку роботи продюсера — створення, редагування та керування треками. Також описано процедуру обробки платежів через інтеграцію з платіжною системою Stripe та збереження придбаних композицій у профілі.

Загалом реалізована система повністю відповідає поставленим функціональним вимогам: забезпечує зручний і безпечний доступ до музичного контенту, підтримує персоналізовану взаємодію з користувачами, дозволяє ефективно керувати каталогом композицій і формує позитивний користувацький досвід. У перспективі платформа може еволюціонувати в напрямку Web 3.0-рішень [25].

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				61
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було спроектовано та реалізовано повноцінну веб-платформу для продажу музичних треків. Платформа дозволяє забезпечити ефективну взаємодію між продюсерами, які публікують контент, та слухачами, які можуть ознайомлюватись із ним, додавати до улюбленого, створювати персональні добірки, а також здійснювати купівлю з відповідною ліцензією. В основу системи покладено мету — створити універсальне середовище для розміщення, прослуховування та комерційного поширення музичних композицій у вигляді цифрових бітів.

Результатом роботи є реалізація функціонального програмного комплексу, що повністю відповідає поставленим завданням. У системі реалізовано реєстрацію та аутентифікацію користувачів, розмежування доступу відповідно до ролі (слухач або продюсер), можливість продюсера завантажувати треки з обкладинками та метаданими, редагувати чи видаляти їх. Слухачі можуть прослуховувати демо, ставити лайки, залишати коментарі, створювати плейлисти та додавати біти у вподобані. Важливою функціональністю є інтеграція з платіжною системою Stripe, яка дозволяє здійснювати купівлю треків з вибором типу ліцензії (MP3, WAV, ZIP з трекаутами), після чого система забезпечує безпечне завантаження треку та автоматичне збереження PDF-ліцензії. У випадку придбання треку з ексклюзивною ліцензією реалізовано видалення цього треку з публічного каталогу, плейлистів та вподобаних у всіх інших користувачів.

Архітектура платформи реалізована з використанням технологій Node.js, Express, MongoDB та React. Серверна частина обробляє запити до API, відповідає за роботу з базою даних та збереження файлів. Клієнтська частина забезпечує динамічну взаємодію з інтерфейсом у режимі реального часу, використовуючи React Router для реалізації SPA-архітектури. Окрему увагу приділено модульності, масштабованості та зручності адаптації інтерфейсу до майбутнього розширення. Налаштування середовища та зберігання ключів реалізовано через .env конфігурацію.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				62
Змн.	Арк.	№ докум.	Підпис	Дата		

Практична цінність розробленої системи полягає в її готовності до використання як платформою незалежного продюсера, так і комерційним стартапом. Платформа адаптивна, сучасна за архітектурою і передбачає масштабування в разі розширення функціоналу. У перспективі можливим напрямом удосконалення є розробка мобільної версії, впровадження системи рекомендацій на базі поведінкової аналітики користувачів, багатомовної підтримки, а також додаткових інструментів монетизації. Таким чином, виконана робота повністю досягає поставленої мети, демонструє високий рівень відповідності технічному завданню та має значний потенціал до практичного застосування та подальшого розвитку.

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				63
Змн.	Арк.	№ докум.	Підпис	Дата		

ДЖЕРЕЛА ІНФОРМАЦІЇ

1. Vice R., Horton A. Mastering React. – Packt Publishing, 2016. – 260 с.
2. Bank C. Full-Stack Web Development with MongoDB and Node.js. – Packt Publishing, 2022. – 320 с
3. Cantelon M., Harter M., Holowaychuk T., Rajlich N. Node.js in Action. – Manning Publications, 2017. – 375 с.
4. Flanagan D. JavaScript: The Definitive Guide. – O'Reilly Media, 2020. – 706 с.
5. Fletcher J. MongoDB: The Definitive Guide. – O'Reilly Media, 2019. – 496 с.
6. Документація React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/learn>
7. Документація NodeJS [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/en>
8. Документація Express.js [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/>
9. Документація MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/docs/>
10. Документація MongoDB Compass [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/products/tools/compass>
11. Документація Mongoose [Електронний ресурс] – Режим доступу до ресурсу: <https://mongoosejs.com/docs/>
12. Документація HTML5 та CSS3 [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web>
13. Документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://devdocs.io/javascript/>
14. Документація Git та GitHub [Електронний ресурс] – Режим доступу до ресурсу: <https://git-scm.com/doc>
15. Документація Stripe API [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.stripe.com/api>

		Давидов В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю				64
Змн.	Арк.	№ докум.	Підпис	Дата		

16. Документація Ant Design [Електронний ресурс] – Режим доступу до ресурсу: <https://ant.design/docs/react/introduce>
17. Документація JWT (JSON Web Tokens) [Електронний ресурс] – Режим доступу до ресурсу: <https://jwt.io/introduction>
18. Документація REST API [Електронний ресурс] – Режим доступу до ресурсу: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
19. Документація React Router [Електронний ресурс] – Режим доступу до ресурсу: <https://reactrouter.com/en/main/start/tutorial>
20. Документація з адаптивної верстки CSS Flexbox [Електронний ресурс] – Режим доступу до ресурсу: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
21. Документація Multer [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/expressjs/multer>
22. Документація dotenv [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/dotenv>
23. Документація Redux Toolkit [Електронний ресурс] – Режим доступу до ресурсу: <https://redux-toolkit.js.org/introduction/getting-started>
24. Документація CORS (Cross-Origin Resource Sharing) [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS>
25. Документація ESLint (Cross-Origin Resource Sharing) [Електронний ресурс] – Режим доступу до ресурсу: <https://eslint.org/docs/latest/>

ДОДАТКИ

		Давидов В.С			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

Фрагмент коду файлу “main_page.jsx”

```

import { PauseCircleOutlined, PlayCircleOutlined } from '@ant-design/icons'
import { Button, Image, Modal, Typography, message } from 'antd'
import Cookies from 'js-cookie'
import { useContext, useEffect, useRef, useState } from 'react'
import { Link, useLocation, useNavigate } from 'react-router-dom'
import { PlayerContext } from '../components/playerContext'
import { useGetAllBeatsQuery } from '../routes/beatsApi'
import PriceModal from '../PriceModal'
import BeatsTable from '../beatsTable/index'
import Footer from '../footer/index'
import Header from '../header/index'
import ShareModal from '../shareModal'
import ExclusiveRights from './license_cards_content/exclusiverights'
import Mp3License from './license_cards_content/mp3lease'
import TrackoutLicense from './license_cards_content/trackoutlease'
import UnlimitedTrackoutLicense from './license_cards_content/unlimitedtrackout'
import UnlimitedWavLicense from './license_cards_content/unlimitedwav'
import WavLicense from './license_cards_content/wavlease'
import styles from './style.module.css'

const Main = (id, imgSrc, title, time, bpm, beatTags, price, soundSrc, key) => {
  const [isModalOpen, setIsModalOpen] = useState(false)
  const [modalContent, setModalContent] = useState("")
  const [beats, setBeats] = useState([])
  const [isShareModalOpen, setIsShareModalOpen] = useState(false)
  const [shareLink, setShareLink] = useState("")
  const [isHovered, setIsHovered] = useState(false)

  const { data } = useGetAllBeatsQuery()
  const query = useQuery()
  const messageRef = useRef(null)
  const sendMail = useRef(null)
  const sendName = useRef(null)
  const sendTitle = useRef(null)
  const [cartBeat, setCartBeat] = useState(null)
  const [isCartModalVisible, setIsCartModalVisible] = useState(false)
  const [featuredBeats, setFeaturedBeats] = useState([])
  const [currentFeaturedIndex, setCurrentFeaturedIndex] = useState(0)
  const navigate = useNavigate()
  const animationIdRef = useRef(null)
  const { playBeat, pauseBeat, setBeatQueue, setCurrentIndex } = useContext(PlayerContext)

  const location = useLocation()

  function useQuery() {
    return new URLSearchParams(useLocation().search)
  }

  const handleCartModalClose = () => {
    setIsCartModalVisible(false)
  }

```

```

}

const handlePlayPauseClick = () => {
  if (!featuredBeat) return

  const beatInfo = {
    _id: featuredBeat._id,
    title: featuredBeat.title,
    imgSrc: featuredBeat.img,
    soundSrc: featuredBeat.soundSrc,
    price: featuredBeat.price,
    bpm: featuredBeat.bpm,
    beatTags: featuredBeat.tags,
    key: featuredBeat.key
  }

  if (String(currentBeat?._id) === String(featuredBeat?._id) && isPlaying) {
    pauseBeat()
  } else {
    setBeatQueue([beatInfo])
    setCurrentIndex(0)
    playBeat(beatInfo)
  }
}

const openModal = (content) => {
  setModalContent(content)
  setIsModalOpen(true)
}

const showCartModal = (beat) => {
  setCartBeat(beat)
  setIsCartModalVisible(true)
}

const closeModal = () => {
  setIsModalOpen(false)
}

const openShareModal = () => {
  setIsShareModalOpen(true)
}

const closeShareModal = () => {
  setIsShareModalOpen(false)
}

const handleDeleteBeat = (idToDelete) => {
  setBeats((prev) => prev.filter((beat) => beat._id !== idToDelete))
}

useEffect(() => {

```

```

fetch('http://localhost:3001/api/beats')
  .then((res) => res.json())
  .then((data) => {
    setBeats(data)

    const maxLikes = Math.max(...data.map((beat) => beat.likes || 0))
    const popularBeats = data.filter((beat) => beat.likes === maxLikes)
    setFeaturedBeats(popularBeats)
  })
  .catch((err) => console.log('Помилка при отриманні бітів:', err))
}, [])

useEffect(() => {
  if (featuredBeats.length > 1) {
    const interval = setInterval(() => {
      setCurrentFeaturedIndex((prevIndex) => (prevIndex + 1) % featuredBeats.length)
    }, 15000)
    return () => clearInterval(interval)
  }
}, [featuredBeats])

useEffect(() => {
  fetch('http://localhost:3001/api/beats')
    .then((res) => res.json())
    .then((data) => setBeats(data))
    .catch((err) => console.log('Помилка при отриманні бітів:', err))
}, [])

useEffect(() => {
  const messageTextarea = messageRef.current
  if (messageTextarea) {
    messageTextarea.style.height = 'auto'
    messageTextarea.style.height = `${messageTextarea.scrollHeight}px`
  }
}, [])

const { analyserRef, isPlaying, isPlayerVisible, stopVisualizer, currentBeat } =
useContext(PlayerContext)

const canvasRef = useRef(null)

useEffect(() => {
  const canvas = canvasRef.current
  const ctx = canvas?.getContext('2d')
  const analyser = analyserRef.current

  if (!canvas || !ctx || !analyser) return

  const bufferLength = analyser.frequencyBinCount
  const dataArray = new Uint8Array(bufferLength)

  const draw = () => {

```

```

animationIdRef.current = requestAnimationFrame(draw)

analyser.getBytesFrequencyData(dataArray)

ctx.clearRect(0, 0, canvas.width, canvas.height)

const barWidth = (canvas.width / bufferLength) * 0.6
let x = 0

for (let i = 0; i < bufferLength; i++) {
  const barHeight = dataArray[i]
  ctx.fillStyle = 'rgb(103, 3, 13)'
  ctx.fillRect(x, canvas.height - barHeight, barWidth, barHeight)
  x += barWidth + 1
}
}

if (isPlayerVisible) {
  if (isPlaying) {
    draw()
  } else {
    analyser.getBytesFrequencyData(dataArray)
    ctx.clearRect(0, 0, canvas.width, canvas.height)

    const barWidth = (canvas.width / bufferLength) * 0.6
    let x = 0

    for (let i = 0; i < bufferLength; i++) {
      const barHeight = dataArray[i]
      ctx.fillStyle = 'rgb(103, 3, 13)'
      ctx.fillRect(x, canvas.height - barHeight, barWidth, barHeight)
      x += barWidth + 1
    }
  }
} else {
  if (animationIdRef.current) {
    cancelAnimationFrame(animationIdRef.current)
    animationIdRef.current = null
  }
  ctx.clearRect(0, 0, canvas.width, canvas.height)
}

return () => {
  if (animationIdRef.current) {
    cancelAnimationFrame(animationIdRef.current)
    animationIdRef.current = null
  }
}
}, [analyserRef, isPlaying, isPlayerVisible])

useEffect(() => {
  const canvas = document.getElementById('mainEqCanvas')

```

```

    if (!isPlayerVisible && canvas) {
      canvas.style.display = 'none'
    }
  }, [isPlayerVisible])

const handleInput = (e) => {
  const textarea = e.target
  textarea.style.height = 'auto'
  textarea.style.height = `${textarea.scrollHeight}px`
}

const handleSendEmail = () => {
  const emailData = {
    email: sendMail.current.value,
    title: sendTitle.current.value,
    name: sendName.current.value,
    message: messageRef.current.value
  }

  if (
    sendMail.current.value !== "" &&
    sendTitle.current.value !== "" &&
    sendName.current.value !== "" &&
    messageRef.current.value !== ""
  ) {
    fetch('http://localhost:3001/api/sendemail/', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(emailData)
    })
      .then((response) => response.json())
      .catch((error) => {
        console.error('Error:', error)
      })
  }
  message.success('Повідомлення успішно надіслано!')
  sendMail.current.value = ""
  sendTitle.current.value = ""
  sendName.current.value = ""
  messageRef.current.value = ""
}

const featuredBeat =
  featuredBeats.length > 0
    ? featuredBeats[currentFeaturedIndex % featuredBeats.length]
    : beats.length
      ? beats.reduce((max, beat) => (beat.likes > (max.likes || 0) ? beat : max), {})
      : null

useEffect(() => {

```

```

const params = new URLSearchParams(location.search)
if (params.get('deleted') === 'true') {
  message.success('Аккаунт удалено')
  params.delete('deleted')
  const newSearch = params.toString()
  const newUrl = location.pathname + (newSearch ? '?' + newSearch : '')
  window.history.replaceState(null, '', newUrl)
}
}, [location])

return (
  <div className={styles.main}>
    <Header />
    <div className={styles.empty}></div>

    <div className={styles.featuredBeat}>
      <div className={styles.beatHeat}>
        <div
          className={styles.imageWrapper}
          onMouseEnter={() => setIsHovered(true)}
          onMouseLeave={() => setIsHovered(false)}
          onClick={handlePlayPauseClick}
          style={{ position: 'relative', cursor: 'pointer' }}>
          <img
            src={featuredBeat?.img}
            alt="beat"
            className={styles.beatHeatLogo}
            style={{ filter: isHovered ? 'brightness(50%)' : 'none' }}
          />
          {(isHovered || String(currentBeat?._id || currentBeat?.id) === String(featuredBeat?._id)) &&
            (String(currentBeat?._id || currentBeat?.id) === String(featuredBeat?._id) && isPlaying ? (
              <PauseCircleOutlined
                style={{
                  position: 'absolute',
                  top: '50%',
                  left: '50%',
                  fontSize: '64px',
                  color: 'rgba(255,255,255,0.8)',
                  transform: 'translate(-50%, -50%)',
                  pointerEvents: 'none',
                  transition: 'opacity 0.3s ease',
                  opacity: 1
                }}
              />
            ) : (
              <PlayCircleOutlined
                style={{
                  position: 'absolute',
                  top: '50%',
                  left: '50%',
                  fontSize: '64px',
                  color: 'rgba(255,255,255,0.8)',

```

```

        transform: 'translate(-50%, -50%)',
        pointerEvents: 'none',
        transition: 'opacity 0.3s ease',
        opacity: 1
      }}
    />
  )})
</div>

<div className={styles.beatInfo}>
  <div className={styles.featuredText}>Популярний Біт</div>
  <div className={styles.beatNameWrapper}>
    {featuredBeat ? (
      <div
        className={styles.beatName}
        style={{ cursor: 'pointer' }}
        onClick={() => {
          if (!featuredBeat) return

          const params = new URLSearchParams({
            _id: featuredBeat._id || "",
            imgSrc: featuredBeat.img || "",
            name: featuredBeat.title || "",
            bpm: featuredBeat.bpm ? String(featuredBeat.bpm) : "",
            beatTags: featuredBeat.tags ? featuredBeat.tags.join(',') : "",
            price: featuredBeat.price ? String(featuredBeat.price) : "",
            key: featuredBeat.key || "",
            soundSrc: featuredBeat.soundSrc || ""
          })

          navigate(`/beatPage?${params.toString()}`)
        }}
      >
        {featuredBeat.title}
      </div>
    ) : (
      <div className={styles.beatName}></div>
    )}
  </div>
  <div className={styles.previewBeatButtons}>
    <Button
      className={styles.purchase}
      onClick={(e) => {
        e.stopPropagation()
        const isAuthorized = Cookies.get('current_login')
        if (featuredBeat) {
          showCartModal({
            _id: featuredBeat._id,
            imgSrc: featuredBeat.img,
            title: featuredBeat.title,
            price: featuredBeat.price,
            soundSrc: featuredBeat.soundSrc
          })
        }
      }}
    />
  </div>

```

```

    }
    if (!isAuthorized) {
      localStorage.setItem('pendingCart', JSON.stringify(showCartModal))
      window.location.href = '/authorization'
      return
    }
  }}>
  <Image preview={false} className={styles.cartPriceImg} src="/mainPage/cart.png" />
  <Typography.Text className={styles.purchaseText}>${featuredBeat?.price ||
'34.95'}</Typography.Text>
</Button>
<Button
  type="primary"
  className={styles.share}
  onClick={() => {
    if (!featuredBeat) return

    const params = new URLSearchParams({
      _id: featuredBeat._id || "",
      imgSrc: featuredBeat.img || "",
      name: featuredBeat.title || "",
      bpm: featuredBeat.bpm ? String(featuredBeat.bpm) : "",
      beatTags: featuredBeat.tags ? featuredBeat.tags.join(',') : "",
      price: featuredBeat.price ? String(featuredBeat.price) : "",
      key: featuredBeat.key || "",
      soundSrc: featuredBeat.soundSrc || ""
    })

    const url = `${window.location.origin}/beatPage?${params.toString()}`

    setShareLink(url)
    setIsShareModalOpen(true)
  }}>
  <Image preview={false} src="/mainPage/share.png" className={styles.shareImg} />
</Button>
{featuredBeat?.tags &&
  featuredBeat.tags.length > 0 &&
  featuredBeat.tags.map((tag, idx) => (
    <Button className={styles.tag} key={idx}>
      <Typography.Text className={idx === 0 ? styles.tagText :
styles.tagText2}>{tag.trim()}</Typography.Text>
    </Button>
  ))}
</div>
</div>
</div>
</div>
{ /* <div className={styles.eq}>
  <Image className={styles.test} preview={false} src="/mainPage/eq.png"></Image>
</div> */ }
<div className={styles.eqWrapper}>
  <div className={styles.eqCanvasContainer}>

```

```

    <canvas ref={canvasRef} id="mainEqCanvas" height="300" />
  </div>
</div>

<BeatsTable beats={beats.slice(0, 10)} onDeleteBeat={handleDeleteBeat}
onPlayBeat={playBeat} />

<div className={styles.buttonWrapper}>
  <Link to="/allbeats">
    <button className={styles.browseAllBeatsButton}>переглянути всі</button>
  </Link>
</div>
<div className={styles.licenses}>
  <div className={styles.licenseinfo}>Інформація про ліцензування</div>
  <div className={styles.licensesmain}>
    <div className={styles.licenseCard}>
      <div className={styles.title}>мп3 лізінг</div>
      <div className={styles.price}>$34.95</div>
      <div className={styles.details}>
        <div>мп3 файл</div>
        <div>2,000 унікальних копій</div>
        <div>1 комерційне використання</div>
        <div>10,000 стрімінгових копій</div>
      </div>
      <button onClick={() => openModal(Mp3License)} className={styles.button}>
        переглянути ліцензію
      </button>
    <div className={styles.bulkdeals}>
      <div>Оптові пропозиції:</div>
      <div>придбай 2 біта, отримай 1 безкоштовно!</div>
      <div>придбай 3 біта, отримай 2 безкоштовно!</div>
      <div>придбай 4 біта, отримай 3 безкоштовно!</div>
      <div>придбай 5 біта, отримай 4 безкоштовно!</div>
    </div>
  </div>
  <div className={styles.licenseCard}>
    <div className={styles.title}>WAV Лізінг</div>
    <div className={styles.price}>$49.95</div>
    <div className={styles.details}>
      <div>WAV + MP3 файл</div>
      <div>5,000 унікальних копій</div>
      <div>2 комерційних використання</div>
      <div>50,000 стрімінгових копій</div>
    </div>
    <button onClick={() => openModal(WavLicense)} className={styles.button}>
      переглянути ліцензію
    </button>
  <div className={styles.bulkdeals}>
    <div>Оптові пропозиції:</div>
    <div>придбай 2 біта, отримай 1 безкоштовно!</div>
    <div>придбай 3 біта, отримай 2 безкоштовно!</div>
    <div>придбай 4 біта, отримай 3 безкоштовно!</div>
  </div>
</div>

```

```

    <div>придбай 5 біта, отримай 4 безкоштовно!</div>
  </div>
</div>
<div className={styles.licenseCard}>
  <div className={styles.title}>Необмежений WAV</div>
  <div className={styles.price}>$79.95</div>
  <div className={styles.details}>
    <div>wav + мп3 Файл</div>
    <div>Необмеження кількість копій</div>
    <div>Необмежене комерційне використання</div>
    <div>Необмежені стрімінгові копії</div>
  </div>
  <button onClick={() => openModal(UnlimitedWavLicense)} className={styles.button}>
    переглянути ліцензію
  </button>
  <div className={styles.bulkdeals}>
    <div>Оптові пропозиції:</div>
    <div>придбай 2 біта, отримай 1 безкоштовно!</div>
    <div>придбай 3 біта, отримай 2 безкоштовно!</div>
    <div>придбай 4 біта, отримай 3 безкоштовно!</div>
    <div>придбай 5 біта, отримай 4 безкоштовно!</div>
  </div>
</div>
<div className={styles.licenseCard}>
  <div className={styles.title}>Трекаут Лізинг</div>
  <div className={styles.price}>$99.95</div>
  <div className={styles.details}>
    <div>Доріжки + WAV + MP3 файл</div>
    <div>2 комерційних використання</div>
    <div>200,000 стрімінгових копій</div>
  </div>
  <button onClick={() => openModal(TrackoutLicense)} className={styles.button}>
    переглянути ліцензію
  </button>
  <div className={styles.bulkdeals}>
    <div>Оптові пропозиції:</div>
    <div>придбай 2 біта, отримай 1 безкоштовно!</div>
    <div>придбай 3 біта, отримай 2 безкоштовно!</div>
    <div>придбай 4 біта, отримай 3 безкоштовно!</div>
    <div>придбай 5 біта, отримай 4 безкоштовно!</div>
  </div>
</div>
<div className={styles.licenseCard}>
  <div className={styles.title}>Необмежений Trackout</div>
  <div className={styles.price}>$149.95</div>
  <div className={styles.details}>
    <div>Доріжки + WAV + MP3 файл</div>
    <div>Необмеження кількість копій</div>
    <div>Необмежене комерційне використання</div>
    <div>Необмежені стрімінгові копії</div>
  </div>

```

```

    <button onClick={() => openModal(UnlimitedTrackoutLicense)}
className={styles.button}>
    переглянути ліцензію
  </button>
  <div className={styles.bulkdeals}>
    <div>Оптові пропозиції:</div>
    <div>придбай 2 біта, отримай 1 безкоштовно!</div>
    <div>придбай 3 біта, отримай 2 безкоштовно!</div>
    <div>придбай 4 біта, отримай 3 безкоштовно!</div>
    <div>придбай 5 біта, отримай 4 безкоштовно!</div>
  </div>
</div>
<div className={styles.licenseCard}>
  <div className={styles.title}>Ексклюзивні права</div>
  <div className={styles.price}>$999.95</div>
  <div className={styles.details}>
    <div>Доріжки + WAV + MP3 файл</div>
    <div>Необмеження кількості копій</div>
    <div>Необмежені стрімінгові копії</div>
    <div>Ексклюзивне право власності</div>
  </div>
  <button onClick={() => openModal(ExclusiveRights)} className={styles.button}>
    переглянути ліцензію
  </button>
  <div className={styles.bulkdeals}>
    <div>Bulk deals:</div>
    <div>придбай 2 біта, отримай 1 безкоштовно!</div>
    <div>придбай 3 біта, отримай 2 безкоштовно!</div>
    <div>придбай 4 біта, отримай 3 безкоштовно!</div>
    <div>придбай 5 біта, отримай 4 безкоштовно!</div>
  </div>
</div>
</div>
<div className="transitionFade"></div>
<div className={styles.services}>
  <div className={styles.servicesinfo}>Послуги</div>
  <div className={styles.servicesmain}>
    <Link to="/services/mixandmastering" className={styles.serviceCard}>
      
      <p className={styles.serviceTitle}>Міксинг та Мастеринг</p>
    </Link>
    <Link to="/services/custombeat" className={styles.serviceCard}>
      
      <p className={styles.serviceTitle}>Кастомний біт</p>
    </Link>
  </div>
  <Link to="/servicespage">
    <button className={styles.browseButton}>ПЕРЕГЛЯНУТИ ВСІ ПОСЛУГИ</button>
  </Link>
</div>

```

```

    <div className={styles.contactForm}>
      <div className={styles.contactInfo}>написати</div>
      <div className={styles.form}>
        <div className={styles.formRow}>
          <input ref={sendName} className={styles.inputField} type="text" placeholder="ім'я" />
        </div>
        <div className={styles.formRow}>
          <input ref={sendMail} className={styles.inputField} type="email" placeholder="пошта"
/>
        </div>
        <div className={styles.formRow}>
          <input ref={sendTitle} className={styles.inputField} type="text" placeholder="назва" />
        </div>
        <div className={styles.formRow}>
          <textarea className={styles.textArea} placeholder="повідомлення"
onInput={handleInput} ref={messageRef}></textarea>
        </div>
        <button className={styles.submitButton} onClick={handleSendEmail}>
          надіслати
        </button>
      </div>
    </div>
    <Footer />
    <Modal title="" width="1000px" visible={isModalOpen} footer={null}
onCancel={closeModal}>
      {modalContent}
    </Modal>
    <ShareModal visible={isShareModalOpen} onClose={closeShareModal}
shareLink={shareLink} />
    <PriceModal
      visible={isCartModalVisible}
      onCancel={handleCartModalClose}
      title={cartBeat?.title}
      imgSrc={cartBeat?.imgSrc}
      beatId={cartBeat?._id}
    />
  </div>
)
}

```

export default Main

Фрагмент коду файлу “beatPage.jsx”

```

import { LikeFilled, LikeOutlined, SendOutlined } from '@ant-design/icons'
import { Button, Image, Input } from 'antd'
import axios from 'axios'
import Cookies from 'js-cookie'
import { useEffect, useState } from 'react'
import { useLocation } from 'react-router-dom'
import CommentTable from '../CommentTable/index'
import BeatsTable from '../beatsTable/index'
import Footer from '../footer/index'
import Header from '../header/index'
import styles from './style.module.css'

const useQuery = () => {
  return new URLSearchParams(useLocation().search)
}

const { TextArea } = Input

const BeatPage = () => {
  const query = useQuery()
  const imgSrc = query.get('imgSrc')
  const title = query.get('name')
  const bpm = query.get('bpm')
  const beatTags = query.get('beatTags')?.split(',') || []
  const price = query.get('price')
  const key = query.get('key')
  const [liked, setLiked] = useState(false)
  const [comment, setComment] = useState("")
  const [data, setData] = useState(null)

  const toggleLike = () => {
    if (!Cookies.get('current_login')) {
      window.location.href = 'http://database/Autorisation.php?window=Login'
      return
    }
    setLiked(!liked)
    const Data = {
      beatName: title,
      login: Cookies.get('current_login'),
      action: 'like',
      text: "",
    }
    console.log(Data)

    if (!liked)
      axios
        .post('http://database/action', Data)
        .then((response) => {
          console.log('УСПЕШНЫЙ ОТВЕТ ОТ СЕРВЕРА:', response.data)
        })
  }
}

```

```

    })
    .catch((error) => {
      console.error('Ошибка при выполнении POST запроса:', error)
    })
  else {
    axios
      .post('http://database/deleteAction', Data)
      .then((response) => {
        console.log('Успешный удаление ответ от сервера:', response.data)
      })
      .catch((error) => {
        console.error('Ошибка при выполнении POST запроса:', error)
      })
  }
}

axios
  .get(`http://database/takeAction?beat_name=${title}&action=like`)
  .then((response) => {
    setData(response.data)
  })
  .catch((error) => {
    console.error('Ошибка при выполнении POST запроса:', error)
  })
}
useEffect(() => {
  axios
    .get(`http://database/takeAction?beat_name=${title}&action=like`)
    .then((response) => {
      const check =
        Array.isArray(response.data) &&
        response.data.some((element) => element.login.toLowerCase() ===
Cookies.get('current_login'))
      if (check) {
        setLiked(true)
      } else {
        setLiked(false)
      }
      setData(response.data)
    })
    .catch((error) => {
      console.error('Ошибка при выполнении POST запроса:', error)
    })
  }, [title, liked])

const handleInputChange = (e) => {
  setComment(e.target.value)
}

const handleSubmit = () => {
  if (!Cookies.get('current_login')) {
    window.location.href = 'http://database/Autorisation.php?window=Login'
    return
  }

```

```

    }
    const Data = {
      beatName: title,
      login: Cookies.get('current_login'),
      action: 'comment',
      text: comment.trim(),
    }
    axios
      .post('http://database/action', Data)
      .then((response) => {
        console.log('Успешный ответ от сервера:', response.data)
        window.location.reload()
      })
      .catch((error) => {
        console.error('Ошибка при выполнении POST запроса:', error)
      })
  }
}

return (
  <div>
    <Header />
    <div className={styles.beatInfoContainer}>
      <div className={styles.beatHeader}>
        <Image preview={false} src={imgSrc} className={styles.beatImgHeader} />
        <div className={styles.beatHeaderInfo}>
          <span className={styles.titleHeader}>{title}</span>
          <span className={styles.authorHeader}>DJ Problematic</span>
          <span className={styles.infoHeader}>
            <Image preview={false} src="/icons/bpm.png" className={styles.bpmIcon} />
            <span className={styles.beatInfoHeader}>{bpm}</span>
            <Image preview={false} src="/icons/sound.png" className={styles.soundIcon} />
            <span className={styles.beatInfoHeader}>{key}</span>
          </span>
          <span className={styles.desHeader}>
            BUY 2 GET 1 FREE, BUY 3 GET 2 FREE, BUY 4 GET 3 FREE, BUY 5 GET 4 FREE
          </span>
        </div>
        <div className={styles.beatControllHeader}>
          <div className={styles.naviButtons}>
            <Button className={styles.priceButton}>
              <Image className={styles.priceImg} src="/mainPage/cart.png" preview={false} />
              <span className={styles.priceText}>${price}</span>
            </Button>
            <Button className={styles.share} type="primary">
              <Image preview={false} src="/mainPage/share.png" className={styles.shareImg} />
            </Button>
            <Button
              className={styles.likeButton}
              type=""
              icon={liked ? <LikeFilled /> : <LikeOutlined />}
              onClick={toggleLike}
            >
              {liked ? 'Liked' : 'Like'}
            </Button>
          </div>
        </div>
      </div>
    </div>
  )
)

```

```

    </Button>
    <div className={styles.likeValue}>{data?.length}</div>
  </div>
  <div className={styles.tagsButtons}>
    {Array.isArray(beatTags)
      ? beatTags.map((tag) => (
        <Button className={styles.tag} key={tag}>
          <span className={styles.tagText}>{tag}</span>
        </Button>
      ))
      : null}
  </div>
</div>
<div style={{ display: 'flex', flexDirection: 'column', gap: '10px', marginTop: '20px', width:
'100%' }}>
  <TextArea
    rows={4}
    value={comment}
    onChange={handleInputChange}
    placeholder="Write your comment here..."
  />
  <Button type="primary" icon={<SendOutlined />} onClick={handleSubmit}>
    Submit
  </Button>
</div>
</div>
</div>

  <CommentTable title={title} />
  <BeatsTable />

  <Footer />
</div>
)
}

```

export default BeatPage

Фрагмент коду файлу "AllBeatsPage.jsx"

```

import { Input, Select, Space } from 'antd'
import axios from 'axios'
import { useEffect, useState } from 'react'
import { useLocation } from 'react-router-dom'
import Searcher from '../components/Searcher/index'
import BeatsTable from '../components/beatsTable/index'
import Footer from '../components/footer/index'
import Header from '../components/header/index'
import styles from './style.module.css'

const useQuery = () => {
  return new URLSearchParams(useLocation().search)
}

const AllBeats = () => {
  const query = useQuery()
  const [value, setValue] = useState(0)
  const title = query.get('title') || ''
  const [genre, setGenre] = useState([])
  const [mood, setMood] = useState([])
  const [bpm, setBpm] = useState([])
  const [key, setKey] = useState([])
  const [like, setLike] = useState('')
  const [bpmRangeInput, setBpmRangeInput] = useState('')

  const [beats, setBeats] = useState([])

  const handleDeleteBeat = (idToDelete) => {
    setBeats((prev) => prev.filter((beat) => beat._id !== idToDelete))
  }

  useEffect(() => {
    const getBeats = async () => {
      const queryParams = new URLSearchParams()

      let bpmArray = []
      if (bpmRangeInput.trim()) {
        const parts = bpmRangeInput.split('-').map((p) => p.trim())
        const from = parseInt(parts[0])
        const to = parts[1] ? parseInt(parts[1]) : null

        if (!isNaN(from) && to !== null && !isNaN(to)) {
          queryParams.append('bpmFrom', from)
          queryParams.append('bpmTo', to)
        } else if (!isNaN(from)) {
          queryParams.append('bpmFrom', from)
        }
      }

      if (mood.length) mood.forEach((m) => queryParams.append('mood', m))
      if (genre.length) genre.forEach((g) => queryParams.append('genre', g))
      if (key.length) key.forEach((k) => queryParams.append('key', k))
      if (like) queryParams.append('sort', like)
      if (title) queryParams.append('title', title)
    }
  })
}

```

```

try {
  const response = await axios.get(`http://localhost:3001/api/beats?${queryParams.toString()}`)

  if (!bpmRangeInput.trim()) {
    setBeats(response.data)
    return
  }

  const parts = bpmRangeInput.split('-').map((s) => s.trim())
  const fromStr = parts[0] || ""
  const toStr = parts[1] || ""

  const from = parseInt(fromStr)
  const to = parseInt(toStr)

  if (isNaN(from) && isNaN(to)) {
    setBeats(response.data)
    return
  }

  const sortedBeats = response.data.sort((a, b) => a.bpm - b.bpm)

  let beatsInRange = []
  let beatsAboveRange = []
  let beatsBelowRange = []

  sortedBeats.forEach((beat) => {
    if (!isNaN(from) && !isNaN(to)) {
      if (beat.bpm >= from && beat.bpm <= to) beatsInRange.push(beat)
      else if (beat.bpm > to) beatsAboveRange.push(beat)
      else if (beat.bpm < from) beatsBelowRange.push(beat)
    } else if (!isNaN(from) && (toStr === "" || toStr === undefined)) {
      if (beat.bpm >= from) beatsInRange.push(beat)
      else beatsBelowRange.push(beat)
    } else if (!isNaN(to) && (fromStr === "" || fromStr === undefined)) {
      if (beat.bpm <= to) beatsInRange.push(beat)
      else beatsAboveRange.push(beat)
    }
  })

  let finalBeats = []

  if (!isNaN(from) && !isNaN(to)) {
    finalBeats = beatsInRange
  } else if (!isNaN(from) && (toStr === "" || toStr === undefined)) {
    finalBeats = beatsInRange
  } else if (!isNaN(to) && (fromStr === "" || fromStr === undefined)) {
    finalBeats = beatsInRange
  } else {
    finalBeats = [...beatsInRange, ...beatsAboveRange, ...beatsBelowRange]
  }

  setBeats(finalBeats)
} catch (error) {
  console.error('Error fetching beats:', error)
}
}

```

```

    getBeats()
  }, [bpmRangeInput, mood, genre, key, like, title])

return (
  <div className={styles.allbeatsContainer}>
    <Header />
    <div className={styles.beatsNavigator}>
      <span className={styles.allBeatsBoard}>Усі біти</span>
      <Searcher />
      <div className={styles.beatsSortContainer}>
        <Space>
          <Input
            placeholder="BPM (наприклад 0-100)"
            value={bpmRangeInput}
            onChange={(e) => setBpmRangeInput(e.target.value)}
            style={{ width: '200px' }}
          />
        </Space>
        <Select
          mode="multiple"
          allowClear
          placeholder="Тональність"
          className={` ${styles.selectBpm} ${styles.whitePlaceholder}`}
          onChange={(value) => setKey(value)}
          maxTagCount={1}
          options={[
            {
              value: 'A-Flat Minor',
              label: 'Ля-бемоль Мінор'
            },
            {
              value: 'A-Flat Major',
              label: 'Ля-бемоль Мажор'
            },
            {
              value: 'A Minor',
              label: 'Ля Мінор'
            },
            {
              value: 'A Major',
              label: 'Ля Мажор'
            },
            {
              value: 'A-Sharp Minor',
              label: 'Ля-диез Мінор'
            },
            {
              value: 'A-Sharp Major',
              label: 'Ля-диез Мажор'
            },
            {
              value: 'B-Flat Minor',
              label: 'Сі-бемоль Мінор'
            },
            {
              value: 'B-Flat Major',

```

```
    label: 'Ci-бемоль Мажор'
  },
  {
    value: 'B Minor',
    label: 'Ci Мінор'
  },
  {
    value: 'B Major',
    label: 'Ci Мажор'
  },
  {
    value: 'C-Flat Major',
    label: 'До-бемоль Мажор'
  },
  {
    value: 'C Minor',
    label: 'До Мінор'
  },
  {
    value: 'C Major',
    label: 'До Мажор'
  },
  {
    value: 'C-Sharp Minor',
    label: 'До-диез Мінор'
  },
  {
    value: 'C-Sharp Major',
    label: 'До-диез Мажор'
  },
  {
    value: 'D-Flat Major',
    label: 'Ре-бемоль Мажор'
  },
  {
    value: 'D Minor',
    label: 'Ре Мінор'
  },
  {
    value: 'D Major',
    label: 'Ре Мажор'
  },
  {
    value: 'D-Sharp Minor',
    label: 'Ре-диез Мінор'
  },
  {
    value: 'E-Flat Minor',
    label: 'Mi-бемоль Мінор'
  },
  {
    value: 'E-Flat Major',
    label: 'Mi-бемоль Мажор'
  },
  {
    value: 'E Minor',
    label: 'Mi Мінор'
```

```

    },
    {
      value: 'E Major',
      label: 'Mi Мажор'
    },
    {
      value: 'F Minor',
      label: 'Фа Мінор'
    },
    {
      value: 'F Major',
      label: 'Фа Мажор'
    },
    {
      value: 'F-Sharp Minor',
      label: 'Фа-диез Мінор'
    },
    {
      value: 'F-Sharp Major',
      label: 'Фа-диез Мажор'
    },
    {
      value: 'G-Flat Major',
      label: 'Соль-бемоль Мажор'
    },
    {
      value: 'G Minor',
      label: 'Соль Мінор'
    },
    {
      value: 'G Major',
      label: 'Соль Мажор'
    },
    {
      value: 'G-Sharp Minor',
      label: 'Соль-диез Мінор'
    }
  ]
}
/>
<Select
  mode="multiple"
  allowClear
  placeholder="Настрій"
  className={` ${styles.selectMood} ${styles.whitePlaceholder}`}
  onChange={(value) => setMood(value)}
  maxTagCount={1}
  options={[
    {
      value: 'Bouncy',
      label: 'Качевий'
    },
    {
      value: 'Dark',
      label: 'Темний'
    },
    {
      value: 'Calm',

```

```

      label: 'Спокійний'
    },
    {
      value: 'Angry',
      label: 'Злий'
    },
    {
      value: 'Sad',
      label: 'Сумний'
    },
    {
      value: 'Depressed',
      label: 'Депресивний'
    },
    {
      value: 'Lonely',
      label: 'Самотній'
    },
    {
      value: 'Relaxed',
      label: 'Розслаблений'
    },
    {
      value: 'Energetic',
      label: 'Енергійний'
    }
  ]
}
/>

```

```

<Select
  mode="multiple"
  allowClear
  placeholder="Жанр"
  className={` ${styles.selectGenre} ${styles.whitePlaceholder}`}
  onChange={(value) => setGenre(value)}
  maxTagCount={1}
  options={[
    {
      value: 'Hip Hop',
      label: 'Хіп Хоп'
    },
    {
      value: 'West Coast',
      label: 'Вест Коуст'
    },
    {
      value: 'Trap',
      label: 'Трап'
    }
  ]}
/>

```

```

<Select
  allowClear
  placeholder="Сортувати"
  className={` ${styles.selectBpm} ${styles.whitePlaceholder}`}
  onChange={(value) => {
    setLike(value || "")
  }}

```

```

        value={like || undefined}
        options={[
          { value: 'DESC', label: 'Релевантні' },
          { value: 'ASC', label: 'Найменше лайків' }
        ]}
      />
    </div>
    <div className={styles.beatsSearcher}></div>
  </div>

  <BeatsTable
    bpmCategory={bpm === 'All' ? '' : bpm}
    moodCategory={mood === 'All' ? '' : mood}
    genreCategory={genre === 'All' ? '' : genre}
    keyCategory={key === 'All' ? '' : key}
    beats={beats}
    sortByLikes={like === 'All' ? '' : like}
    onDeleteBeat={handleDeleteBeat}
  />

  <Footer />
</div>
)
}

export default AllBeats

```

Фрагмент коду файлу "AuthorizationPage.jsx"

```

import { ExclamationCircleOutlined, LockOutlined, MailOutlined, UserOutlined } from '@ant-design/icons'
import { Button, Flex, Input, Space, Tooltip, Typography, message } from 'antd'
import Cookies from 'js-cookie'
import { useState } from 'react'
import { useDispatch } from 'react-redux'
import Footer from '../components/footer'
import Header from '../components/header'
import { useRegisterUserMutation } from '../routes/usersApi'
import { useGetAuthAccessMutation } from '../routes/usersApi/index'
import { setUser } from '../userSlice'
import styles from './style.module.css'

const AuthorizationPage = () => {
  const { Title } = Typography
  const [email, setEmail] = useState("")
  const [username, setUsername] = useState("")
  const [password, setPassword] = useState("")
  const [repPassword, setRepPassword] = useState("")
  const [passwordError, setPasswordError] = useState(false)
  const [isRegister, setIsRegister] = useState(false)
  const [login, { isLoading }] = useGetAuthAccessMutation()
  const [register] = useRegisterUserMutation()

  const dispatch = useDispatch()

  const handleLogin = async () => {
    if (isRegister && password !== repPassword) {
      message.error('Паролі не співпадають')
      return
    }

    if (isRegister) {
      const emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/

      if (!emailRegex.test(email)) {
        message.error('Введіть коректну електронну пошту')
        return
      }
    }

    if (password.length <= 4) {
      setPasswordError(true)
      return
    }

    try {
      const response = isRegister
        ? await register({ email, password, username }).unwrap()
    }
  }

```

```

: await login({ email, password, username }).unwrap()

message.success(isRegister ? 'Рєєстрація успішна!' : 'Вхід виконано успішно!')
Cookies.set('current_login', response.username)
Cookies.set('current_email', response.email)
Cookies.set('role', response.role || 'user')
localStorage.setItem('avatar', response?.avatar || '/user/avatar.png')
Cookies.set('user_id', response._id || response.id)

const userCart = Cookies.get(response.username)
if (userCart) {
  const cartArr = JSON.parse(userCart)
  const totalPrice = cartArr.reduce((sum, item) => sum + parseFloat(item.price), 0).toFixed(2)
  Cookies.set('totalPrice', totalPrice, { expires: 7 })
  window.dispatchEvent(new Event('cartUpdated'))
} else {
  Cookies.set('totalPrice', '0.00', { expires: 7 })
  window.dispatchEvent(new Event('cartUpdated'))
}

dispatch(
  setUser({
    username: response.username,
    email: response.email,
    role: response.role || 'user'
  })
)

const pendingCart = localStorage.getItem('pendingCart')
if (pendingCart) {
  localStorage.removeItem('pendingCart')
  Cookies.set('cart_beat', pendingCart)
  window.location.href = '/?openCartModal=true'
} else {
  window.location.href = '/'
}
} catch (err) {
  console.error('❌ Помилка авторизації:', err)
  message.error(err?.data?.message || 'Помилка входу')
}
}

return (
  <div className={styles.main}>
    <Header />
    <div className={styles.content}>
      <Flex vertical justify="center" align="center" style={{ height: '80vh' }}>
        <Space direction="vertical" style={{ width: 400 }}>
          {isRegister ? (
            <
              <Input
                size="large"

```

```

placeholder="Нікнейм"
prefix={<UserOutlined />}
value={username}
onChange={(e) => setUsername(e.target.value)}
/>
<Input
  size="large"
  placeholder="Email"
  prefix={<MailOutlined />}
  value={email}
  onChange={(e) => setEmail(e.target.value)}
/>
</>
): (
  <Input
    size="large"
    placeholder="Email або Нікнейм"
    prefix={<UserOutlined />}
    value={email}
    onChange={(e) => {
      setEmail(e.target.value)
      setUsername(e.target.value)
    }}
  />
)
)}

<Input
  size="large"
  placeholder="Пароль"
  prefix={<LockOutlined />}
  type="password"
  value={password}
  onChange={(e) => {
    setPassword(e.target.value)
    setPasswordError(e.target.value.length <= 4 && e.target.value.length !== 0)
  }}
  suffix={
    passwordError && (
      <Tooltip title="Пароль має бути довшим за 4 символи" color="red">
        <ExclamationCircleOutlined />
      </Tooltip>
    )
  }
/>
{isRegister && (
  <Input
    size="large"
    placeholder="Повторіть пароль"
    prefix={<LockOutlined />}
    type="password"
    value={repPassword}
    onChange={(e) => setRepPassword(e.target.value)}
  />
)}

```

```

    />
  })

  <Button type="primary" size="large" style={{ width: 400 }} loading={isLoading}
onClick={handleLogin}>
    {isRegister ? 'Зареєструватися' : 'Увійти'}
  </Button>

  <Button type="link" style={{ padding: 0, marginTop: -10 }} onClick={() =>
setIsRegister(!isRegister)}>
    {isRegister ? 'Вже є акаунт? Увійти' : 'У вас ще немає акаунту? Зареєструватися'}
  </Button>
</Space>
</Flex>
</div>
<Footer />
</div>
)
}

```

```
export default AuthorizationPage
```

Фрагмент коду файлу “CartPage.jsx”

```

import Cookies from 'js-cookie'
import { useEffect, useState } from 'react'
import { useLocation, useNavigate } from 'react-router-dom'
import CheckoutForm from '../components/CheckoutForm/index'
import Footer from '../components/footer'
import Header from '../components/header'
import styles from './style.module.css'

const useQuery = () => {
  return new URLSearchParams(useLocation().search)
}

const CartPage = () => {
  const navigate = useNavigate()
  const query = useQuery()
  const userId = Cookies.get('user_id') || 'guest'
  const imgSrc = decodeURIComponent(query.get('imgSrc') || '')
  const title = decodeURIComponent(query.get('title') || '')
  const lease = decodeURIComponent(query.get('lease') || '')
  const price = decodeURIComponent(query.get('price') || '')
  const userKey = Cookies.get('current_login')
  const [showCheckout, setShowCheckout] = useState(false)

  const [cartItems, setCartItems] = useState(() => {
    const cartItemsCookie = Cookies.get(userId)
    return cartItemsCookie ? JSON.parse(cartItemsCookie) : []
  })
  const handleCheckout = (beatId, totalPrice) => {
    const userId = Cookies.get('user_id')
    Cookies.set('user_id', userId)
    Cookies.set('beat_id', beatId)
    navigate(`/checkout?totalPrice=${totalPrice}`)
  }

  useEffect(() => {
    Cookies.set(userId, JSON.stringify(cartItems), { expires: 7 })
    const totalPrice = cartItems.reduce((total, item) => total + parseFloat(item.price), 0).toFixed(2)
    Cookies.set('totalPrice', totalPrice, { expires: 7 })
    window.dispatchEvent(new Event('cartUpdated'))
  }, [cartItems, userId])

  useEffect(() => {
    if (title && lease && price) {
      const newItem = { imgSrc, title, lease, price, _id: query.get('beatId') }
      const itemExists = cartItems.some((item) => item.title === title && item.lease === lease &&
item.price === price)
      if (!itemExists) {
        const updatedCart = [...cartItems, newItem]
        setCartItems(updatedCart)
      }
    }
  })
}

```

```

    Cookies.set(userId, JSON.stringify(updatedCart), { expires: 7 })
    setTimeout(() => {
      window.dispatchEvent(new Event('cartUpdated'))
    }, 0)
  }
}
}, [imgSrc, title, lease, price])

const handleRemoveItem = (itemId) => {
  const updatedCart = cartItems.filter((_, index) => index !== itemId)
  setCartItems(updatedCart)
  Cookies.set(userId, JSON.stringify(updatedCart), { expires: 7 })
  setTimeout(() => {
    window.dispatchEvent(new Event('cartUpdated'))
  }, 0)
}

const totalPrice = cartItems.reduce((total, item) => total + parseFloat(item.price), 0).toFixed(2)

useEffect(() => {
  Cookies.set('totalPrice', totalPrice, { expires: 7 })
  window.dispatchEvent(new Event('cartUpdated'))
}, [totalPrice])

return (
  <div className={styles.cartPageContainer}>
    <Header />
    <main className={styles.mainContent}>
      <h1>Кошник</h1>
      <div className={styles.cartTable}>
        {cartItems.length > 0 ? (
          cartItems.map((item, index) => (
            <div className={styles.cartItem} key={index}>
              <div className={styles.productInfo}>
                <div className={styles.image}>
                  <img src={item.imgSrc} alt={item.title} />
                </div>
                <div className={styles.productDetails}>
                  <h2>{item.title}</h2>
                  <p>{item.lease}</p>
                </div>
              </div>
              <div className={styles.productPrice}>
                <p>
                  <b>${item.price}</b>
                </p>
                <button className={styles.removeButton} onClick={() =>
handleRemoveItem(index)}>
                  X
                </button>
              </div>
            </div>
          ))
        ) : (
          <div>Корзина пуста</div>
        )
      </div>
    </main>
  </div>
)

```

```

    ))
  ) : (
    <p>Твій кошик наразі пустий</p>
  ))
</div>
<div className={styles.checkoutSection}>
  <div className={styles.priceSummary}>
    <p>
      КІЛЬКІСТЬ ТОВАРІВ: <b>{cartItems.length}</b>
    </p>
    <p>
      До сплати: <b>${totalPrice}</b>
    </p>
  </div>
  {!showCheckout ? (
    <button
      className={styles.checkoutButton}
      disabled={cartItems.length === 0}
      onClick={() => handleCheckout(cartItems[0]?._id, totalPrice)}>
        оформити замовлення
      </button>
    ) : (
      <CheckoutForm totalPrice={totalPrice} cartItems={cartItems} />
    )}
  <p className={styles.termsText}>
    Натискаючи кнопку, ви приймаєте Ліцензійну угоду(и) на продукт(и) <br />
  </p>
</div>
</main>
<Footer className={styles.footer} />
</div>
)
}

```

```
export default CartPage
```

Фрагмент коду файлу “CheckoutForm.jsx”

```
import { CardElement, useElements, useStripe } from '@stripe/react-stripe-js'
import Cookies from 'js-cookie'
import { useState } from 'react'
import { useNavigate } from 'react-router-dom'
import styles from './style.module.css'

const CARD_ELEMENT_OPTIONS = {
  style: {
    base: {
      color: '#fff',
      fontSize: '18px',
      '::placeholder': {
        color: '#bbb'
      },
    },
    fontFamily: 'inherit',
    backgroundColor: '#222'
  },
  invalid: {
    color: '#ff4d4f'
  }
}

const CheckoutForm = ({ totalPrice, cartItems }) => {
  const stripe = useStripe()
  const elements = useElements()
  const [loading, setLoading] = useState(false)
  const [message, setMessage] = useState("")
  const navigate = useNavigate()

  const license = cartItems?.[0]?.lease

  const userId = Cookies.get('user_id')
  const beatId = Cookies.get('beat_id')

  const handleSubmit = async (e) => {
    e.preventDefault()
    setLoading(true)

    if (!beatId) {
      console.error('beatId is undefined. Redirecting to home page.')
      navigate('/')
      return null
    }

    if (!cartItems || !cartItems.length || !cartItems[0].lease) {
      setMessage('Не вдалося визначити тип ліцензії для покупки')
      setLoading(false)
      return
    }
  }
}
```

```

    }

    const userEmail = Cookies.get('current_email') || ''
    const userName = Cookies.get('current_login') || ''

    const response = await fetch('http://localhost:3001/api/stripe/create-payment-intent', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ amount: Math.round(totalPrice * 100), customerEmail: userEmail,
customerName: userName })
    })

    const { clientSecret } = await response.json()

    const { error, paymentIntent } = await stripe.confirmCardPayment(clientSecret, {
      payment_method: { card: elements.getElement(CardElement) }
    })

    if (error) {
      setMessage(`Платіж відхилено: ${error.message}`)
      setLoading(false)
      return
    }

    if (paymentIntent.status === 'succeeded') {
      setMessage('🎉 Дякуємо за покупку!')

      const beatsToSave = cartItems.map((item) => ({
        beatId: item._id || item.id,
        license: item.lease
      })))

      const saveRes = await fetch('http://localhost:3001/api/users/add-purchased-beats', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ userId, beats: beatsToSave })
      })

      if (!saveRes.ok) {
        setMessage('Не вдалося зберегти покупку. Спробуйте ще раз.')
        setLoading(false)
        return
      }

      localStorage.setItem(`purchasedBeats_${userId}`, JSON.stringify(beatsToSave))

      navigate('/success')
    }

    setLoading(false)
  }

```

```
return (  
  <div className={styles.checkoutContainer}>  
    <div className={styles.checkoutTitle}>Оплата картой</div>  
    <form onSubmit={handleSubmit} style={{ width: '100%' }}>  
      <CardElement options={CARD_ELEMENT_OPTIONS} className={styles.stripeInput} />  
      <button type="submit" className={styles.payButton} disabled={!stripe || loading}>  
        {loading ? 'Processing...' : 'Сплатити'}  
      </button>  
      {message && <div className={styles.message}>{message}</div>}  
    </form>  
  </div>  
)  
}
```

```
export default CheckoutForm
```

Фрагмент коду файлу “BeatsTable.jsx”

```

import { CloseOutlined, CopyOutlined, EditOutlined, LikeFilled, LikeOutlined, PlusOutlined }
from '@ant-design/icons'
import { Button, Dropdown, Image, Input, Modal, message } from 'antd'
import axios from 'axios'
import Cookies from 'js-cookie'
import { useContext, useEffect, useState } from 'react'
import { Link, useLocation, useNavigate, useParams } from 'react-router-dom'
import { PlayerContext } from '../components/playerContext'
import PriceModal from '../PriceModal'
import styles from './style.module.css'

const useQuery = () => {
  return new URLSearchParams(useLocation().search)
}

const BeatsTable = ({
  bpmCategory = "",
  moodCategory = "",
  genreCategory = "",
  keyCategory = "",
  sortByLikes = "",
  likedOnlyBy = "",
  playlistBeats = undefined,
  playlistId = null,
  beats = [],
  onDeleteBeat,
  onAddBeat,
  onRemoveBeatFromPlaylist,
  removeOnUnlike = false,
  onPlayBeat
}) => {
  const query = useQuery()
  const [isShareModalVisible, setIsShareModalVisible] = useState(false)
  const [isCartModalVisible, setIsCartModalVisible] = useState(false)
  const [isAdmin, setIsAdmin] = useState(false)
  const [shareLink, setShareLink] = useState(["", ""])
  const [isPlaylistModalVisible, setPlaylistModalVisible] = useState(false)
  const [playlists, setPlaylists] = useState([])
  const [selectedBeatId, setSelectedBeatId] = useState(null)
  const [isCreatingPlaylist, setIsCreatingPlaylist] = useState(false)
  const [newPlaylistName, setNewPlaylistName] = useState("")
  const { id } = useParams()
  const [playlist, setPlaylist] = useState(null)
  const [likedMap, setLikedMap] = useState({})
  const [likesCountMap, setLikesCountMap] = useState({})
  const [liked, setLiked] = useState(false)
  const [likesCount, setLikesCount] = useState(0)
  const beatId = query.get('_id')
  const userId = Cookies.get('user_id')

```

```

const navigate = useNavigate()
const [likesTrigger, setLikesTrigger] = useState(0)

const toggleLike = async (beatId) => {
  const userId = Cookies.get('user_id')
  if (!userId) {
    message.warning('Спочатку увійдіть у свій акаунт')
    window.location.href = '/authorization'
    return
  }

  const wasLiked = likedMap[beatId]
  const beat = beats.find((b) => b._id === beatId)

  try {
    const res = await fetch('http://localhost:3001/api/likes/toggle', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ beatId, userId })
    })

    const data = await res.json()
    setLikesTrigger((v) => v + 1)

    if (wasLiked && removeOnUnlike && typeof onDeleteBeat === 'function') {
      onDeleteBeat(beatId)
    }
    if (!wasLiked && typeof onAddBeat === 'function' && beat) {
      onAddBeat(beat)
    }
  } catch (err) {
    console.error('❌ Помилка лайку:', err)
  }
}

useEffect(() => {
  const userId = Cookies.get('user_id')
  if (!beats.length) return

  const fetchLikes = async () => {
    try {
      const results = await Promise.all(
        beats.map((beat) =>
          axios
            .post('http://localhost:3001/api/likes/check', { beatId: beat._id, userId })
            .then((res) => ({ beatId: beat._id, liked: res.data.liked, likes: res.data.likes }))
        )
      )

      const newLikedMap = {}

```

```

const newLikesCountMap = {}

results.forEach(({ beatId, liked, likes }) => {
  newLikedMap[beatId] = liked || false
  newLikesCountMap[beatId] = likes
})

setLikedMap(newLikedMap)
setLikesCountMap(newLikesCountMap)
} catch (err) {
  console.error('Помилка перевірки лайків:', err)
}
}

fetchLikes()
}, [beats, likesTrigger])

const openPlaylistModal = async (beatId) => {
  try {
    const email = Cookies.get('current_email')
    const userInfo = await axios.post('http://localhost:3001/api/users/getByEmail', { email })
    const userId = userInfo.data._id
    Cookies.set('user_id', userId)

    const res = await axios.get('http://localhost:3001/api/playlists', {
      params: { userId }
    })

    setPlaylists(res.data)
    setSelectedBeatId(beatId)
    setPlaylistModalVisible(true)
  } catch (error) {
    console.error('Помилка завантаження плейлистів', error)
    message.error('Неможливо завантажити плейлисти')
  }
}

const { currentBeat, setCurrentBeat, setCurrentIndex, currentIndex, setBeatQueue, playBeat } =
useContext(PlayerContext)

const [cartBeat, setCartBeat] = useState(null)
const [beatList, setBeatList] = useState([])
const location = useLocation()
const [handlePlaySound, setHandlePlaySound] = useState(false)
const title = new URLSearchParams(location.search).get('title') || ''

const showShareModal = (beat, link) => {
  setShareLink([beat, link])
  setIsShareModalVisible(true)
}

const getUserRights = () => {

```

```

    Cookies.get('current_login') === 'admin' ? setIsAdmin(true) : setIsAdmin(false)
  }

  const showCartModal = (beat) => {
    setCartBeat(beat)
    setIsCartModalVisible(true)
  }

  const handleShareModalClose = () => {
    setIsShareModalVisible(false)
  }

  const handleCartModalClose = () => {
    setIsCartModalVisible(false)
  }

  const handleBeatClick = (beat, index) => {
    const fullQueue = beats.map((b) => ({
      id: b._id,
      rawBeatId: String(b._id),
      imgSrc: b.img,
      title: b.title,
      link: `/beatPage?_id=${beat._id}&name=${encodeURIComponent(beat.title)}`,
      price: b.price,
      soundSrc: b.soundSrc,
      bpm: b.bpm,
      key: b.key,
      beatTags: b.beatTags
    })))

    setBeatQueue(fullQueue)
    setCurrentIndex(index)
    playBeat(beat)
  }

  const nextBeat = () => {
    if (beatList.length === 0 || currentIndex === null) return

    const nextIndex = (currentIndex + 1) % beatList.length
    setCurrentIndex(nextIndex)
    setCurrentBeat(beatList[nextIndex])
  }

  const prevBeat = () => {
    if (beatList.length === 0 || currentIndex === null) return

    const prevIndex = (currentIndex - 1 + beatList.length) % beatList.length
    setCurrentIndex(prevIndex)
    setCurrentBeat(beatList[prevIndex])
  }

  useEffect(() => {

```

```

    getUserRights()
  }, [])

useEffect(() => {
  if (!beats.length) return

  const newBeatList = beats
    .filter((beat) => beat.title && beat.img && beat.soundSrc)
    .map((beat, index) => {
      const mongoId = beat._id
      const tagsArray = beat.tags || []

      const beatPlayerInfo = {
        id: mongoId,
        rawBeatId: String(mongoId),
        imgSrc: beat.img,
        title: beat.title,
        link: `localhost:3000/beatPage?name=${encodeURIComponent(beat.title)}`,
        price: beat.price,
        soundSrc: beat.soundSrc,
        bpm: beat.bpm,
        key: beat.key,
        beatTags: beat.tags
      }

      return {
        id: mongoId,
        beatPlayerInfo,
        beatPlaySrc: beat.soundSrc,
        img: <img className={styles.beatImg} src={beat.img} onClick={() =>
handleBeatClick(beatPlayerInfo, index)} />,
        title: (
          <Link
            to={`/beatPage?_id=${mongoId}&imgSrc=${encodeURIComponent(beat.img)}&name=${encodeURIComponent(
              beat.title
            )}&bpm=${beat.bpm}&beatTags=${encodeURIComponent(tagsArray.join(','))}&price=${beat.price}
            }&key=${encodeURIComponent(beat.key)}&soundSrc=${encodeURIComponent(beat.soundSrc)} `
          >
            <span onClick={() => window.scrollTo(0, 0)} className={styles.titleText}>
              {beat.title}
            </span>
          </Link>
        ),
        time: <span className={styles.time}>{beat.time?.split(':').slice(-2).join(':')}</span>,
        bpm: <span className={styles.bpm}>{beat.bpm}</span>,
        tags: Array.isArray(tagsArray)
          ? tagsArray.map((tag) => (
              <Button className={styles.tag} key={tag}>
                <span className={styles.tagText}>{tag}</span>

```

```

        </Button>
      ))
      : null,
      link: (
        <Dropdown
          menu={{
            items: [
              {
                key: 'share',
                label: 'Поділитися',
                onClick: () =>
                  showShareModal(
                    beat.title,
                    `http://localhost:3000/beatPage?imgSrc=${encodeURIComponent(beat.img)}&name=${encodeURIComponent(beat.title)}&bpm=${beat.bpm}&beatTags=${encodeURIComponent(
                      tagsArray.join(',')
                    )}&price=${beat.price}&key=${encodeURIComponent(beat.key)}&soundSrc=${encodeURIComponent(beat.soundSrc)}`
                  )
              },
              {
                key: 'addToPlaylist',
                label: 'Додати до плейлиста',
                onClick: () => openPlaylistModal(mongoId)
              }
            ]
          }}
          trigger={['click']}
          placement="bottomRight">
            <Button type="default" icon={<PlusOutlined />} />
          </Dropdown>
        ),
      like: (
        <Button className={styles.likeButton} onClick={() => toggleLike(mongoId)}>
          {likedMap[mongoId] ? <LikeFilled style={{ color: '#ff4d4f', fontSize: 18 }} /> :
        <LikeOutlined style={{ fontSize: 18 }} />}
          <span className={styles.likeTextWithCount}>
            <span className={` ${styles.likeCount} ${likedMap[mongoId] ? styles.likedCount :
              ''}`>{likesCountMap[mongoId] || 0}</span>
          </span>
        </Button>
      ),
      price: (
        <Button
          className={styles.priceButton}
          onClick={(e) => {
            e.stopPropagation()
            const isAuthorized = Cookies.get('current_login')
            const beatObj = {
              imgSrc: beat.img,

```

```

        title: beat.title,
        price: beat.price,
        soundSrc: beat.soundSrc,
        _id: beat._id
      }
      if (!isAuthorized) {
        localStorage.setItem('pendingCart', JSON.stringify(beatObj))
        window.location.href = '/authorization'
        return
      }
      showCartModal(beatObj)
    }>
    <Image className={styles.priceImg} src="/mainPage/cart.png" preview={false} />
    <span className={styles.priceText}>${beat.price}</span>
  </Button>
),
delete: isAdmin ? (
  <Button type="dashed" ghost danger className={styles.deleteBeatButton} onClick={() =>
handleDeleteBeat(beat.title, beat._id)}>
    <CloseOutlined />
  </Button>
) : null,
edit: isAdmin ? (
  <Button type="primary" ghost className={styles.editBeatButton} onClick={() =>
navigate(`/edit/${beat._id}`)}>
    <EditOutlined />
  </Button>
) : null
}
})

```

```

    setBeatList(newBeatList)
  }, [beats, likedMap, likesCountMap])

```

```

const handleDeleteBeat = async (titleToDelete, idToDelete) => {
  try {
    await axios.post('http://localhost:3001/api/deleteBeat', {
      title: titleToDelete
    })
    if (onDeleteBeat) onDeleteBeat(idToDelete)
    message.success('Біт успішно видалено')
  } catch (error) {
    console.error('❌ Помилка видалення біта:', error)
    message.error('Не вдалося видалити біт')
  }
}

```

```

const handleRemoveBeatFromPlaylist = async (idToDelete) => {
  try {
    await axios.post('http://localhost:3001/api/playlists/removeBeat', {
      playlistId,
      beatId: idToDelete
    })
  }
}

```

```

    })
    message.success('Біт видалено з плейлиста')
    if (onRemoveBeatFromPlaylist) onRemoveBeatFromPlaylist(idToDelete)
  } catch (err) {
    message.error('Не вдалося видалити біт')
  }
}

const addBeat = (id, imgSrc, title, time, bpm, beatTags, price, soundSrc, key, mongoId) => {
  const beatPlayerInfo = {
    id: mongoId,
    rawBeatId: String(mongoId),
    imgSrc,
    title,
    link: `localhost:3000/beatPage?name=${encodeURIComponent(title)}`,
    price,
    soundSrc
  }
  const newRow = {
    id: mongoId,
    beatPlayerInfo,
    beatPlaySrc: soundSrc,
    img: <img className={styles.beatImg} src={imgSrc} onClick={() =>
handleBeatClick(beatPlayerInfo, beatList.length)} />,
    removeFromPlaylist: playlistId ? (
      <Button danger type="text" size="small" onClick={() =>
handleRemoveBeatFromPlaylist(mongoId)} style={{ fontSize: '14px' }}>
      ✖
    </Button>
    ) : null,

    title: (
      <Link
to={` /beatPage?_id=${mongoId}&imgSrc=${encodeURIComponent(imgSrc)}&name=${encodeURIComponent(title)}&bpm=${bpm}&beatTags=${encodeURIComponent(
  beatTags.join(',')
)}&price=${price}&key=${encodeURIComponent(key)}`}>
      <span onClick={() => window.scrollTo(0, 0)} className={styles.titleText}>
        {title}
      </span>
    </Link>
  ),
    time: <span className={styles.time}>{time?.split(':').slice(-2).join(':')}</span>,
    bpm: <span className={styles.bpm}>{bpm}</span>,
    tags: Array.isArray(beatTags)
      ? beatTags.map((tag) => (
        <Button className={styles.tag} key={tag}>
          <span className={styles.tagText} style={{ display: 'flex' }}>
            {tag}
          </span>
        </Button>
      ))
      : []
  }
}

```

```

    ))
    : null,
    link: (
      <Dropdown
        menu={{
          items: [
            {
              key: 'share',
              label: 'Поділитися',
              onClick: () =>
                showShareModal(
                  title,
                    `/beatPage?imgSrc=${encodeURIComponent(imgSrc)}&name=${encodeURIComponent(title)}&bpm=${bpm}&beatTags=${encodeURIComponent(
                      beatTags.join(',')
                    )}&price=${price}&key=${encodeURIComponent(key)}`
                )
            },
            {
              key: 'addToPlaylist',
              label: 'Додати до плейлиста',
              onClick: () => openPlaylistModal(mongoId)
            }
          ]
        }}
        placement="bottomRight"
        trigger={['click']}>
        <Button type="default" icon={<PlusOutlined />} />
      </Dropdown>
    ),

    like: (
      <Button className={styles.likeButton} type="" onClick={() => toggleLike(mongoId)}>
        {likedMap[mongoId] ? <LikeFilled style={{ color: '#ff4d4f', fontSize: 18 }} /> :
        <LikeOutlined style={{ fontSize: 18 }} />}
        <span className={styles.likeTextWithCount}>
          <span className={` ${styles.likeCount} ${likedMap[mongoId] ? styles.likedCount :
            ''}`>{likesCountMap[mongoId] || 0}</span>
          </span>
        </Button>
    ),

    price: (
      <Button
        className={styles.priceButton}
        onClick={(e) => {
          e.stopPropagation()
          const isAuthorized = Cookies.get('current_login')
          const beat = { imgSrc, title, price, soundSrc, _id: mongoId }
          if (!isAuthorized) {
            localStorage.setItem('pendingCart', JSON.stringify(beat))

```

```

        window.location.href = '/authorization'
        return
    }
    showCartModal(beat)
  }>
  <Image className={styles.priceImg} src="/mainPage/cart.png" preview={false} />
  <span className={styles.priceText}>$34.95</span>
</Button>
),
delete: (
  <Button type="dashed" ghost danger className={styles.deleteBeatButton} onClick={() =>
handleDeleteBeat(title)}>
    <CloseOutlined />
  </Button>
)
}

setBeatList((prevBeatList) => [...prevBeatList, newRow])
}

const handleCopy = (link) => {
  navigator.clipboard
    .writeText(link)
    .then(() => {
      message.success('Посилання скопійовано!')
    })
    .catch(() => {
      message.error('Не вдалося скопіювати посилання')
    })
}

useEffect(() => {
  const query = new URLSearchParams(window.location.search)
  const shouldOpenCart = query.get('openCartModal') === 'true'
  const pendingCart = Cookies.get('cart_beat')

  if (shouldOpenCart && pendingCart) {
    try {
      const beat = JSON.parse(pendingCart)
      showCartModal(beat)
    } catch (e) {
      console.error('Помилка розбору cart_beat из cookie:', e)
    }

    Cookies.remove('cart_beat')
    window.history.replaceState(null, '', window.location.pathname)
  }
}, [])

return (
  <div className={styles.tableBeatsContainer}>
    <table className={styles.tableBeats}>

```

```

<thead>
  <tr>
    <th className={styles.beatTableCol}></th>
    <th className={styles.titleTableCol}>назва</th>
    <th className={styles.timeTableCol}>Час</th>
    <th className={styles.bpmTableCol}>бпм</th>
    <th className={styles.tagsTableCol}>теги</th>
    <th className={styles.linkTableCol}></th>
    <th className={styles.likeTableCol}></th>
    <th className={styles.priceTableCol}></th>
  </tr>
</thead>
<tbody>
  {beats
    .filter((beat) => beat.title && beat.img && beat.soundSrc)
    .map((beat, index) => {
      const mongoId = beat._id
      const tagsArray = beat.tags || []
      return (
        <tr className={styles.beatRow} key={mongoId}>
          <td className={styles.beatTableCol}>
            <img
              className={styles.beatImg}
              src={`http://localhost:3001/${beat.img}`}
              onClick={() =>
                handleBeatClick(
                  {
                    id: mongoId,
                    rawBeatId: String(mongoId),
                    imgSrc: beat.img,
                    title: beat.title,
                    link:
                      `/beatPage?_id=${beat._id}&imgSrc=${encodeURIComponent(beat.img)}&name=${encodeURIComponent(beat.title)}&bpm=${beat.bpm}&beatTags=${encodeURIComponent(beat.tags || []).join(',')}&price=${beat.price}&key=${encodeURIComponent(beat.key)}&soundSrc=${encodeURIComponent(beat.soundSrc)}`,
                    price: beat.price,
                    soundSrc: beat.soundSrc,
                    bpm: beat.bpm,
                    beatTags: beat.tags,
                    key: beat.key
                  },
                  index
                )
              }
            />
          </td>
          <td className={styles.titleTableCol}>
            <Link

```

```

to={` /beatPage? _id=${mongoId}&imgSrc=${encodeURIComponent(beat.img)}&name=${encodeURIComponent(beat.title)}&bpm=${beat.bpm}&beatTags=${encodeURIComponent(
    tagsArray.join(',')
)}&price=${beat.price}&key=${encodeURIComponent(beat.key)}&soundSrc=${encodeURIComponent(beat.soundSrc)} ` }>
    <span onClick={() => window.scrollTo(0, 0)} className={styles.titleText}>
        {beat.title}
    </span>
</Link>
</td>
<td className={styles.timeTableCol}>{beat.time?.split(':').slice(-2).join(':')}</td>
<td className={styles.bpmTableCol}>{beat.bpm}</td>
<td className={styles.tagsTableCol}>
    {Array.isArray(tagsArray)
    ? tagsArray.map((tag) => (
        <Button className={styles.tag} key={tag}>
            <span className={styles.tagText} style={{ display: 'flex' }}>
                {tag}
            </span>
        </Button>
    ))
    : null}
</td>
<td className={styles.linkTableCol}>
    <Dropdown
        menu={{
            items: [
                {
                    key: 'share',
                    label: 'Поділитися',
                    onClick: () =>
                        showShareModal(
                            beat.title,
                            `http://localhost:3000/beatPage? _id=${beat._id}&imgSrc=${encodeURIComponent(beat.img)}&name=${encodeURIComponent(beat.title)}&bpm=${beat.bpm}&beatTags=${encodeURIComponent(
                                tagsArray.join(',')
                            )}&price=${beat.price}&key=${encodeURIComponent(beat.key)}&soundSrc=${encodeURIComponent(beat.soundSrc)} `
                        )
                },
                {
                    key: 'addToPlaylist',
                    label: 'Додати до плейлиста',
                    onClick: () => openPlaylistModal(mongoId)
                }
            ]
        }}
        trigger={['click']}>

```

```

        <Button type="default" icon={<PlusOutlined />} />
      </Dropdown>
    </td>
    <td className={styles.likeTableCol}>
      <Button className={styles.likeButton} type="" onClick={() =>
toggleLike(mongoId)}>
        {likedMap[mongoId] ? (
          <LikeFilled style={{ color: '#ff4d4f', fontSize: 18 }} />
        ) : (
          <LikeOutlined style={{ fontSize: 18 }} />
        )}
        <span className={styles.likeTextWithCount}>
          <span className={` ${styles.likeCount} ${likedMap[mongoId] ? styles.likedCount :
"}}`}>
            {likesCountMap[mongoId] || 0}
          </span>
        </span>
      </Button>
    </td>
    {playlistId && (
      <td>
        <Button
          danger
          type="text"
          size="small"
          onClick={() => handleRemoveBeatFromPlaylist(mongoId)}
          style={{ fontSize: '16px', left: '15px' }}>
            ✖
        </Button>
      </td>
    )}
  <td className={styles.priceTableCol}>
    <Button
      className={styles.priceButton}
      onClick={(e) => {
        e.stopPropagation()
        const isAuthorized = Cookies.get('current_login')
        const beatObj = {
          imgSrc: beat.img,
          title: beat.title,
          price: beat.price,
          soundSrc: beat.soundSrc,
          _id: beat._id
        }
        if (!isAuthorized) {
          localStorage.setItem('pendingCart', JSON.stringify(beatObj))
          window.location.href = '/authorization'
          return
        }
        showCartModal(beatObj)
      }}>
    <Image className={styles.priceImg} src="/mainPage/cart.png" preview={false} />

```

```

        <span className={styles.priceText}>$34.95</span>
      </Button>
    </td>

    {isAdmin && (
      <td className={styles.editBeatButtonContainer}>
        <Button
          type="primary"
          ghost
          size="large"
          className={styles.editBeatButton}
          onClick={() => navigate(`/edit/${beat._id}`)}>
            <EditOutlined />
          </Button>
        </td>
      )}
    {isAdmin && (
      <td className={styles.deleteBeatButtonContainer}>
        <Button
          type="dashed"
          ghost
          danger
          size="large"
          className={styles.deleteBeatButton}
          onClick={() => handleDeleteBeat(beat.title, beat._id)}>
            <CloseOutlined />
          </Button>
        </td>
      )}
    </tr>
  )
  )}
</tbody>
</table>
{isAdmin && (
  <Link to={`/${addbeat}`}>
    <Button
      className={styles.beatSrc}
      style={{ marginTop: '20px', width: '100px', height: '35px' }}
      shape="round"
      icon={<PlusOutlined />}
      size="middle"
    />
  </Link>
)}
<Modal open={isShareModalVisible} footer={null} onCancel={handleShareModalClose}
title={shareLink[0]}>
  <div style={{ display: 'flex', alignItems: 'center', gap: '10px' }}>
    <Input value={shareLink[1]} readOnly />
    <Button icon={<CopyOutlined />} onClick={() => handleCopy(shareLink[1])} />
  </div>
</Modal>

```

```

    {cartBeat && (
      <PriceModal
        visible={isCartModalVisible}
        onCancel={handleCartModalClose}
        title={cartBeat.title}
        imgSrc={cartBeat.imgSrc}
        price={cartBeat.price}
        soundSrc={cartBeat.soundSrc}
        beatId={cartBeat._id}
      />
    )}

    <Modal open={isPlaylistModalVisible} onCancel={() => setPlaylistModalVisible(false)}
    footer={null} title="Оберіть плейлист">
      {playlists.length === 0 ? (
        <p style={{ marginBottom: 12 }}>У вас ще немає плейлистів</p>
      ) : (
        playlists.map((playlist) => (
          <Button
            key={playlist._id}
            block
            style={{ marginBottom: 10 }}
            onClick={async () => {
              try {
                await axios.post('http://localhost:3001/api/playlists/addBeat', {
                  playlistId: playlist._id,
                  beatId: selectedBeatId
                })
                message.success(`Біт додано до "${playlist.playlistName}"`)
                setPlaylistModalVisible(false)
              } catch (err) {
                if (axios.isAxiosError(err) && err.response?.status === 409) {
                  message.warning('Цей біт вже є в цьому плейлисті')
                } else {
                  message.error('Не вдалося додати біт')
                }
              }
            }}
            {playlist.playlistName}
          </Button>
        ))
      )}

      {!isCreatingPlaylist ? (
        <Button type="dashed" block icon={<PlusOutlined />} onClick={() =>
        setIsCreatingPlaylist(true)} style={{ marginTop: 10 }}>
        Новий плейлист
      </Button>
      ) : (
        <Input
          placeholder="Введіть назву плейлиста"

```

```

    value={newPlaylistName}
    onChange={(e) => setNewPlaylistName(e.target.value)}
    style={{ marginTop: 10 }}
  />
  <Button
    type="primary"
    block
    disabled={!newPlaylistName}
    onClick={async () => {
      try {
        const userId = Cookies.get('user_id')
        const res = await axios.post('http://localhost:3001/api/playlists', {
          userId,
          playlistName: newPlaylistName,
          playlistPreview: '/db/img/default.jpg',
          initialBeatId: selectedBeatId
        })

        const newPlaylist = res.data.playlist
        setPlaylists((prev) => [...prev, newPlaylist])
        setNewPlaylistName("")
        setIsCreatingPlaylist(false)
        setPlaylistModalVisible(false)
        message.success(`Плейлист "${newPlaylist.playlistName}" створено`)
      } catch (err) {
        if (axios.isAxiosError(err) && err.response?.status === 409) {
          message.warning('Цей біт вже є в цьому плейлисті')
        } else {
          console.error(err)
          message.error('Не вдалося створити плейлист')
        }
      }
    }}
    style={{ marginTop: 8 }}>
    Зберегти
  </Button>
</>
)}
</Modal>
</div>
)
}

export default BeatsTable

```