НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система бронювання діджейсетів»

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Керівник бакалаврської кваліфікаційної роботи к.т.н., доцент	_____ (підпис)	**Ірина БОРОДКІНА**
Виконав	_____ (підпис)	**Володимир САЧКО**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«_06_» _____01_____ 2026 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Сачку Володимиру Володимировичу
Спеціальність «122 Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні науки»
Тема бакалаврської кваліфікаційної роботи «Інформаційна система бронювання діджейсетів»
затверджена наказом від «_06_» __січня__ 2026 р. № _46_ «С»
Термін подання завершеної роботи на кафедру «_22_» _травня_ 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): інформаційна система бронювання діджейсетів; прототип ROOM_9; стек Next.js, TypeScript, Tailwind CSS, Supabase PostgreSQL/Auth/Storage, Cloudflare Workers; вимоги до створення подій, заявок, ролей користувачів, Booking CRM, Event Desk, Sound Vault, Music Lab та Signal Engine.
Перелік питань, що підлягають дослідженню:
Аналіз предметної області бронювання діджейсетів та існуючих рішень; проєктування логічної моделі даних; розроблення інтерфейсної та серверної частини інформаційної системи; реалізація Event Desk, Booking CRM, Sound Vault, Music Lab та Signal Engine; перевірка основних сценаріїв бронювання.
Перелік графічних документів (за необхідності).
ER-модель, діаграма розгортання, use case-діаграма, архітектура компонентів, алгоритм підбору DJ за звуковими референсами, sequence diagram створення booking request, скріншоти інтерфейсу прототипу ROOM_9.
Дата видачі завдання «_06_» _січня_ 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Ірина БОРОДКІНА**

Завдання взяв до виконання _____ (підпис) **Володимир САЧКО**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

Таблиця 0.1 – Перелік умовних позначень та скорочень

Позначення	**Пояснення**
API	інтерфейс програмування застосунків
CRM	система супроводу заявок і взаємодії з клієнтами
ER-модель	логічна модель сутностей і зв’язків між ними
RLS	Row Level Security, політики доступу до рядків бази даних
UI	користувацький інтерфейс
UX	користувацький досвід
MVP	мінімально життєздатний продукт
Sound Vault	персональний музичний архів користувача
Atmosphere Brief	збережений звуковий момент для опису атмосфери події

ВСТУП

Цифрова музична індустрія поступово переходить від простого прослуховування аудіо до складніших форм взаємодії між слухачем, артистом і професійним організатором подій. Для електронної музики важливим є не лише ім’я виконавця, а й характер звучання, енергія сету, темп, драматургія переходів і відповідність конкретному простору. Через це організатору потрібен не загальний опис стилю, а можливість швидко зафіксувати фрагмент, який передає очікувану атмосферу майбутнього виступу.

Актуальність теми пов’язана з потребою в інформаційній системі, що автоматизує бронювання діджейсетів і поєднує створення події, вибір DJ, формування заявки, керування статусами та збереження звукових референсів. На практиці ці дії часто розподілені між різними сервісами: музика розміщується на одних платформах, події ведуться в таблицях, а комунікація відбувається в месенджерах. Прототип ROOM_9 об’єднує ці етапи в одному процесі.

У роботі прототип ROOM_9 розглядається не як окрема музична вебплатформа, а як практична реалізація інформаційної системи бронювання діджейсетів. Музичні модулі використовуються як допоміжний шар для уточнення заявки: організатор може зафіксувати звуковий фрагмент, прикріпити його до події та створити структуровану заявку на виступ DJ.

Метою роботи є розроблення інформаційної системи бронювання діджейсетів, що забезпечує створення подій, керування заявками, ролями користувачів, статусами бронювання та використання звукових референсів для уточнення потреб організатора.

Об’єктом дослідження є процес бронювання діджейсетів для музичних подій. Предметом дослідження є моделі даних, алгоритми та програмні засоби побудови інформаційної системи, у якій звуковий фрагмент використовується як контекст заявки на виступ DJ.

Для досягнення мети у роботі вирішено такі завдання:

- проаналізовано предметну область бронювання діджейсетів та існуючі цифрові рішення;
- сформовано функціональні й нефункціональні вимоги до системи;
- розроблено логічну модель даних і структуру користувацьких сценаріїв;
- спроєктовано архітектуру застосунку, бази даних, безпеки та розгортання;
- реалізовано прототип інформаційної системи з модулями Explore, Track Page, Sound Vault, Artist Dossier, Event Desk, Booking CRM, Music Lab і Signal Engine;
- підготовлено рекомендації щодо впровадження, тестування та подальшого розвитку системи.

У роботі використано методи системного аналізу, ER-моделювання, структурного проєктування, прототипування інтерфейсу, алгоритмізації рекомендацій, ручного функціонального тестування та аналізу політик доступу до реляційних даних [24; 25].

Практичне значення роботи полягає у створенні прототипу інформаційної системи, що демонструє не окрему сторінку або дизайн-макет, а пов’язаний процес бронювання діджейсету. Користувач може пройти шлях від вибору звукового референсу до прикріплення цього моменту до події, створення заявки й перегляду операційного кейсу. Такий сценарій показує зв’язок між предметною областю, моделлю даних, інтерфейсом, алгоритмами та безпекою.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Інформаційна система бронювання діджейсетів реалізована у вигляді прототипу ROOM_9. Публічна частина системи дає змогу прослуховувати DJ-сети, переглядати сторінки артистів і зберігати звукові референси. Професійна частина підтримує організацію подій, формування лайнапу, створення заявок, роботу з райдером, календарем і станом booking case.

Ключова інформаційна одиниця — Atmosphere Brief. Це збережений звуковий момент із timestamp, назвою треку, артистом, енергією, BPM, типом простору і коротким поясненням. У межах системи такий об’єкт не є товаром. Він використовується як бриф, що допомагає організатору точніше описати бажаний характер майбутнього діджейсету.

Базова роль користувача — Listener. Після реєстрації користувач може слухати музику, зберігати треки й моменти, створювати плейлисти та формувати власний Sound Vault. Професійні можливості DJ, Organizer і Venue відкриваються пізніше через Role Verification Center.

Предметна область має багаторольову природу. Один і той самий користувач може бути слухачем, DJ, організатором або представником venue. Жорсткий вибір ролі під час реєстрації ускладнює вхід у систему й погано відповідає реальному сценарію, коли користувач спочатку знайомиться з музичним матеріалом, а вже згодом відкриває професійні інструменти. Тому в прототипі ROOM_9 застосовано listener-first модель, у якій професійні ролі є додатковими доступами, а не окремими типами акаунтів.

1.2 Аналіз вимог до програмної системи

Вимоги сформовано за сценаріями бронювання, а не лише за сторінками інтерфейсу. Одна сторінка може обслуговувати різні ролі: Track Page потрібна слухачеві для прослуховування, організатору для вибору звукового контексту, а DJ для перевірки того, як його матеріал представлено в системі.

Нефункціональні вимоги включають стабільність аудіоплеєра, контроль переповнення тексту, адаптивність для мобільних браузерів, обмеження запитів до бази даних, безпечну роботу з файлами та передбачувану поведінку інтерфейсу у locked state. Для системи, що працює з довгими назвами треків, псевдонімами артистів і динамічними картками, контроль overflow є не косметичним, а функціональним аспектом якості.

Таблиця 1.1

Функціональні вимоги до інформаційної системи бронювання діджейсетів

Категорія	**Вимога**	**Призначення**
Автентифікація	Реєстрація, вхід, підтвердження пошти, відновлення пароля	Безпечний доступ до персональних і професійних даних
Музика	Прослуховування треків, глобальний player, queue, repeat-one	Підбір звукового контексту для бронювання
Sound Vault	Збережені треки, Atmosphere Briefs, плейлисти, історія	Архів звукових референсів для майбутніх заявок
DJ profile	Аватар, cover, релізи, Sound Proof, SoundCloud, trust drawer	Публічне представлення артиста
Event Desk	Події, slots, прикріплення Atmosphere Brief	Перехід від звуку до структури події
Booking CRM	Заявки, case file, статуси, чат, райдер, escrow preview	Професійний супровід букінгу
Безпека	RLS, owner-scoped data, storage policies	Захист приватних даних і файлів
Адаптивність	Робота у desktop,	Доступність основних

Категорія	Вимога	Призначення
	iPhone, Android та iPad browser	сценаріїв на різних пристроях

Таблиця 1.2

Ролі користувачів інформаційної системи та їх можливості

Роль	Основна потреба	Ключові функції	Обмеження доступу
Listener	Слухати й зберігати музику	Explore, Track Page, Sound Vault, playlists, queue	Не має доступу до професійних CRM-дій
DJ	Публікувати музику й відповідати на заявки	Uploads, Releases, Artist Dossier, Music Lab, Rider	Керує тільки власним профілем і контентом
Organizer	Підбирати артистів і керувати заявками	Event Desk, Booking CRM, Calendar, Case File	Бачить тільки власні події та пов’язані заявки
Venue	Керувати подіями фізичного простору	Venue events, lineup slots, recurring calendar	Редагує тільки власні venue-дані
Admin	Модерувати систему	Reports, users, content	Не є учасником звичайного

Роль	Основна потреба	Ключові функції	Обмеження доступу
		moderation, statistics	booking flow

1.3 Моделювання предметної області

Основний сценарій бронювання починається з вибору звукового референсу. Користувач відкриває Explore, слухає трек, переходить на Track Page, зберігає момент у Sound Vault, прикріплює Atmosphere Brief до слоту події та створює букінгову заявку. Далі заявка супроводжується у Booking Case.

Рис. 1.1 – Основний сценарій бронювання діджейсету в прототипі ROOM_9

1.4 Огляд інформаційних джерел та існуючих рішень

Для порівняння розглянуто музичні платформи, сервіси подій і професійні CRM-рішення. SoundCloud і Spotify мають сильний music-first досвід, але не підтримують операційне бронювання діджейсетів. Resident Advisor і DICE допомагають відкривати події, однак не перетворюють конкретний звуковий момент на структурований бриф заявки. Gigwell-подібні CRM наближені до професійного процесу, проте мають слабший музичний контекст для підбору DJ.

Таблиця 1.3

Порівняння існуючих рішень для задачі бронювання діджейсетів

Рішення	**Сильні сторони**	**Обмеження для задачі бронювання діджейсетів**
SoundCloud	Завантаження музики, профілі артистів, соціальне поширення	Немає Event Desk, Booking CRM і звукового брифу заявки
Spotify	Стабільний player, бібліотека, персоналізація	Немає професійної взаємодії організатора й DJ
Resident Advisor	Контекст клубної культури, події, артисти, міста	Звуковий момент не стає формальним об’єктом процесу
DICE	Події, квитки, мобільний сценарій	Фокус на продажі квитків, а не на підборі артиста через sound proof
Gigwell-подібні CRM	Статуси, переговори, документи, календар	Недостатній музичний discovery-шар для слухача й організатора

1.5 Постановка завдання

Потрібно розробити інформаційну систему бронювання діджейсетів, що підтримує наскрізний шлях від вибору звукового референсу до професійної заявки. Результатом має бути pre-release прототип із реальними маршрутами, базою даних, авторизацією, storage, role unlock, Sound Vault, Event Desk, Booking CRM, Music Lab та Signal Engine.

Окреме завдання полягає у коректному розмежуванні реалізованого й запланованого функціоналу. У прототипі реалізовано музичний player,

збереження моментів, базові плейлисти, релізи, рольові доступи, booking case, rider upload у демонстраційній логіці, escrow preview та deterministic recommendation. Реальні платежі, повноцінна live-stream інфраструктура, адміністративна модерація і ML-рекомендації залишаються перспективою наступних версій та не змінюють основної теми роботи — бронювання діджейсетів.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Логічна модель інформаційної системи включає базові профілі користувачів, рольові доступи, DJ-профілі, треки, релізи, збережені треки й моменти, плейлисти, події, слоти лайнапу, букінги, повідомлення, платежі preview-рівня, аудіофічі та нотифікації. Основні зв’язки показують перехід від користувача до персонального Sound Vault, від DJ до музичного контенту, від події до слотів, а від слоту до букінгового кейсу.

У моделі даних важливо відокремити об’єкти музичного споживання від професійних операцій. Трек, реліз, плейлист і збережений момент належать до music-first шару. Подія, слот лайнапу, букінг, повідомлення, райдер і escrow preview належать до операційного шару. Зв’язок між цими шарами створюється через saved_moments і source_work_id у booking-заявці.

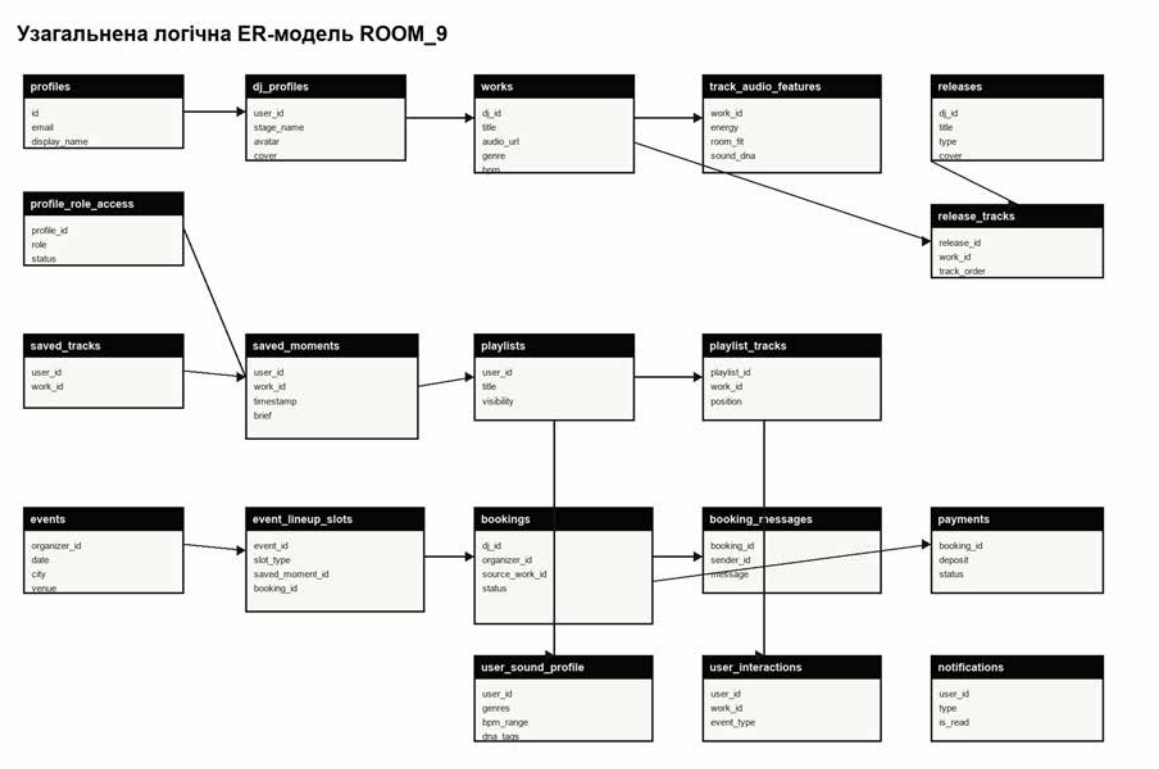

Рис. 2.1 – Узагальнена логічна ER-модель інформаційної системи бронювання діджейсетів

Таблиця 2.1

Основні сутності логічної моделі даних системи бронювання

Сутність	**Призначення**	**Ключові зв’язки**
profiles	Базовий обліковий запис	1:N із role access, playlists, saved moments
profile_role_access	Рольові unlock-стани	N:1 із profiles
dj_profiles	Публічний і професійний профіль артиста	1:N із works, releases, availability
works	Треки та сети	N:1 із DJ, 1:N із saved_moments, track_audio_features
releases	Альбоми, EP, сети, сингли	N:M із works через release_tracks
saved_moments	Збережені Atmosphere Briefs	N:1 із user і work, можуть вести до booking
playlists	Персональні добірки	1:N із playlist_tracks
events	Події організатора або venue	1:N із event_lineup_slots
event_lineup_slots	Слоти лайнапу	N:1 із event, може містити saved_moment і booking
bookings	Професійна заявка	N:1 із DJ, organizer, event slot
booking_messages	Лог і чат кейсу	N:1 із bookings
track_audio_features	Ознаки звучання	1:1 або N:1 із works
user_sound_profile	Агрегований профіль смаку	1:1 із profiles
notifications	Системні повідомлення	N:1 із profiles

2.2 Архітектура програмної системи

Прототип ROOM_9 побудовано як модульний вебзастосунок на основі Next.js App Router [1]. Інтерфейс реалізовано на React [2] і TypeScript [3]. Дані зберігаються в Supabase PostgreSQL [4], автентифікація виконується через Supabase Auth, а медіафайли розміщуються в Supabase Storage. Для розгортання використано OpenNext і Cloudflare Workers [7; 8].

Для бакалаврського прототипу обрано модульний моноліт, а не мікросервісну архітектуру [25]. Це рішення спрощує розробку, деплой і тестування, оскільки всі маршрути, UI-компоненти та інтеграції залишаються в одному Next.js-проєкті. Подальший поділ на окремі сервіси доцільний лише після появи незалежного навантаження на streaming, recommendations, payments або analytics.

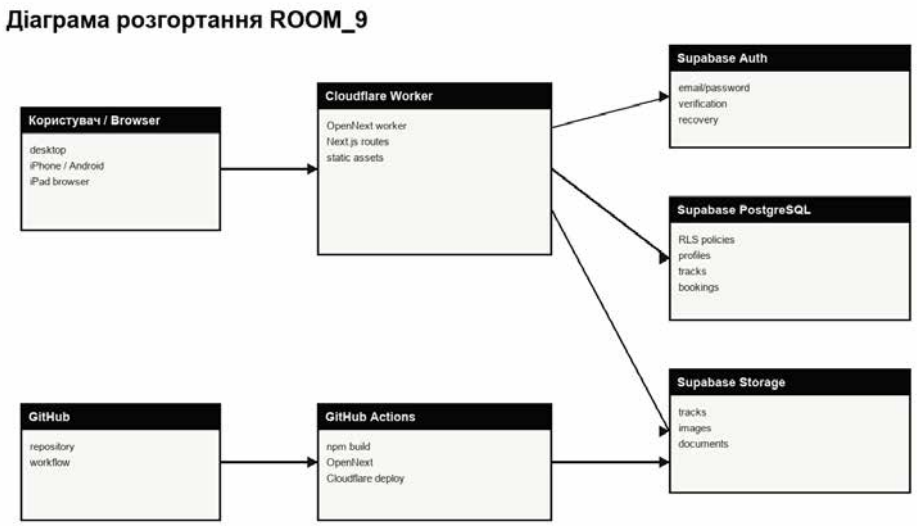

Рис. 2.2 – Діаграма розгортання системи

2.3 Проєктування користувацьких сценаріїв

Інтерфейс прототипу включає публічний шар і workspace-шар. Публічний шар містить Home, Explore, Track Page, Artist Dossier, Release Page, Events, Streams і Library. Workspace-шар містить Dashboard, Bookings, Events, Calendar, Analytics, Streams, Music Lab і Settings. Такий поділ не змішує музичний досвід із професійними операціями бронювання.

Користувацькі сценарії проєктуються так, щоб кожен екран відповідав на чотири практичні питання: де перебуває користувач, який об’єкт є головним, яку дію треба виконати далі та чому цим даним можна довіряти. Для Track Page головним об’єктом є waveform і вибраний момент. Для Sound Vault — збережений музичний архів. Для Event Desk — подія і слот лайнапу. Для Booking Case — операційний стан заявки.

2.4 Проєктування безпеки

Безпека проєктується на рівні автентифікації, рольового доступу, RLS-політик і Storage-політик. Публічні треки, релізи та базові профілі можуть читатися без професійного доступу. Особисті плейлисти, saved moments,

notifications, booking messages і приватні event-дані обмежуються власником або учасниками відповідного процесу.

Ключове правило безпеки полягає у тому, що frontend не використовує service_role key. Клієнтський код працює лише з publishable key, а реальні права визначаються політиками бази даних. Це зменшує ризик випадкового відкриття адміністративного доступу й дає змогу перевіряти поведінку системи двома окремими акаунтами.

Таблиця 2.2

Рівні доступу до даних у системі

Дані	**Читання**	**Створення**	**Оновлення / видалення**
profiles	Власник, частково публічно	Автоматично після реєстрації	Власник або admin
dj_profiles	Публічно	Користувач із DJ access	Власник DJ-профілю
works	Публічні треки читаються всіма	Власник DJ-профілю	Власник треку
playlists	Власник або public playlist	Авторизований користувач	Власник playlist
saved_moments	Власник; пов’язаний DJ у межах кейсу	Авторизований користувач	Власник або пов’язаний DJ у дозволених станах
bookings	Організатор, target DJ, admin	Organizer/Venue access	Учасники відповідно до статусу
events	Публічні події — всі	Organizer/Venue access	Власник події
notifications	Тільки адресат	Система або пов’язаний	Адресат може позначати

Дані	Читання	Створення	Оновлення / видалення
		процес	прочитаними

2.5 Комп’ютерно-наукове обґрунтування моделі рекомендацій

Signal Engine — пояснюваний механізм рекомендацій на основі музичних і поведінкових сигналів. Для спеціальності 122 важливо, що модель описується формально: вона має вхідні множини, вектор ознак, вагові коефіцієнти, функцію оцінки та детермінований алгоритм ранжування.

Позначимо множину треків як T, множину користувачів як U, множину збережених моментів як M, множину плейлистів як P, а множину слотів подій як E. Для треку t формується вектор ознак x(t) = (genre, bpm, room, dna, energy, readiness). Для користувача u формується профіль смаку на основі likes, saves, playlists, queue, listening history і saved moments. Для слоту e задається контекст: Opening, Support, Peak, Closing або Stream.

Загальний бал відповідності подано формулою: score(t,u,e) = w1G + w2B + w3R + w4D + w5U + w6Q, де G — жанровий збіг, B — близькість BPM, R — відповідність типу простору, D — збіг sound DNA, U — поведінкові сигнали користувача, Q — професійна готовність артиста.

Таблиця 2.3

Компоненти моделі рекомендацій Signal Engine

Компонент моделі	Формальне подання	Пояснення
Трек	$t \in T$	Музичний об’єкт із genre, BPM, duration, audio features
Користувач	$u \in U$	Профіль із поведінковими сигналами й role state

Компонент моделі	**Формальне подання**	**Пояснення**
Момент	$m = (t, \tau, l)$	Трек, timestamp τ і label l
Плейлист	$p \in P$	Стійкий сигнал інтересу до групи треків
Слот події	$e \in E$	Opening, Support, Peak, Closing або Stream
Оцінка	score(t,u,e)	Сумарна відповідність треку користувачу або події

Таблиця 2.4

Вагові коефіцієнти рекомендаційної моделі

Сигнал	**Позначення**	**Listening context**	**Event slot context**	**Booking context**
Genre match	G	0.25	0.15	0.10
BPM proximity	B	0.15	0.25	0.20
Room type match	R	0.10	0.25	0.25
Sound DNA match	D	0.20	0.15	0.15
User behavior	U	0.25	0.10	0.10
Artist readiness	Q	0.05	0.10	0.30

Наведені значення є демонстраційними для прототипу. Після накопичення реальних даних їх доцільно уточнювати через експерименти з поведінкою користувачів.

Псевдокод алгоритму наведено з урахуванням базових підходів теорії алгоритмів [24]:

```
Algorithm SignalEngineRecommendation
Input: tracks T, user u, event slot e
Output: ranked list of recommended tracks

1. For each track t in T:
2.     Extract genre, BPM, room type, energy and sound DNA features
3.     Compare track features with user sound profile
4.     Compare track features with event slot requirements
5.     Calculate score(t,u,e) = w1G + w2B + w3R + w4D + w5U + w6Q
6.     Generate explanation for recommendation
7. Sort tracks by score in descending order
8. Return top N tracks with explanations
```

Content-based recommendation краще підходить для раннього прототипу, ніж collaborative filtering, оскільки не потребує великої кількості користувачів і може працювати з уже відомими характеристиками треку [11; 12].

Пояснюваність рекомендацій є окремою вимогою. Користувач має бачити не лише список запропонованих треків, а й причину: збіг BPM, повторення жанру у saved moments, відповідність room type або професійну готовність артиста. Це наближує рекомендацію до інструмента прийняття рішення, а не до непрозорого автоматичного ранжування.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою

Як систему управління інформаційною базою обрано Supabase PostgreSQL. PostgreSQL забезпечує реляційну модель, зовнішні ключі, індекси, транзакційність і політики Row Level Security [5; 6]. Supabase додає керовану автентифікацію, storage, REST API та зручну інтеграцію з клієнтським застосунком.

Для ROOM_9 реляційна модель є зручною, оскільки більшість об’єктів мають чіткі зв’язки: користувач володіє плейлистами, DJ володіє треками, реліз складається з треків, подія має слоти, а слот може бути пов’язаний із заявкою. Документоорієнтоване сховище могло б спростити зберігання частини metadata, але ускладнило б контроль доступу, зовнішні ключі та перевірку цілісності.

Таблиця 3.1

Групи таблиць бази даних

Група таблиць	**Приклади**	**Призначення**
Identity	profiles, profile_role_access	Облікові записи та рольові доступи
Artist	dj_profiles, works, releases, release_tracks	Профілі DJ, треки, альбоми та EP
Vault	saved_tracks, saved_moments, playlists, playlist_tracks	Персональний музичний архів
Booking	bookings, booking_messages, payments	Заявки, чат, escrow preview
Events	events, event_lineup_slots,	Події, слоти лайнапу, календар

Група таблиць	Приклади	Призначення
	availability	
Signals	track_audio_features, user_sound_profile, user_interactions	Дані для рекомендацій і Music Lab

3.2 Розробка інформаційної бази

Інформаційна база реалізована у SQL-сценарії `supabase/schema.sql`. У ньому створено таблиці, індекси, constraints, grants, RLS-політики та Storage buckets. Для Sound Vault-даних використовується user-scoped підхід, що запобігає змішуванню плейлистів, saved moments і notification між акаунтами. Скорочені фрагменти SQL-схеми наведено у Додатку А, а приклади RLS-політик — у Додатку Б.

Для масштабування передбачено індекси за полями, які часто використовуються у фільтрах і зв'язках: `dj_id`, `organizer_id`, `status`, `created_at`, `work_id`, `event_id`, `saved_moment_id`, а також GIN-індекси для масивів жанрів, room type і sound DNA. Це зменшує ризик повільних запитів під час роботи Explore, Sound Vault, Booking CRM та Signal Engine.

3.3 Вибір інструментарію

Основними інструментами розробки стали Next.js, React, TypeScript, Tailwind CSS, Supabase, OpenNext, Cloudflare Workers і GitHub Actions [1–8; 22]. Такий стек дозволяє створити інтерактивний frontend, мати реальну базу даних і storage без окремого backend-сервера, а також розгортати застосунок у serverless-середовищі.

Вибір TypeScript зменшує кількість помилок під час роботи з різними типами об'єктів: треками, релізами, saved moments, event slots, booking cases і role access. Tailwind CSS використовується не як випадковий набір класів, а як інструмент для підтримки єдиної системи відступів, рамок, кольорів і

типографіки [22]. Це важливо для проєкту з великою кількістю операційних екранів.

3.4 Реалізація інтерфейсної системи

Інтерфейс побудований на повторно використовуваних компонентах. Base UI включає Button, Input, Select, Panel і StatusBadge. Music UI містить GlobalPlayer, Waveform, TrackCard і TrackRow. Workspace UI включає WorkspaceSidebar, BookingPipeline, EventSlotCard і RoleAccessPanel.

Візуальна система ROOM_9 поєднує brutalist-ідентичність із вимогами до професійного інструмента. Публічні музичні сторінки можуть бути виразнішими, тоді як workspace-екрани мають залишатися компактними. Кислотно-зелений колір використовується тільки для активних станів, primary CTA, progress, live/success і вибраних моментів.

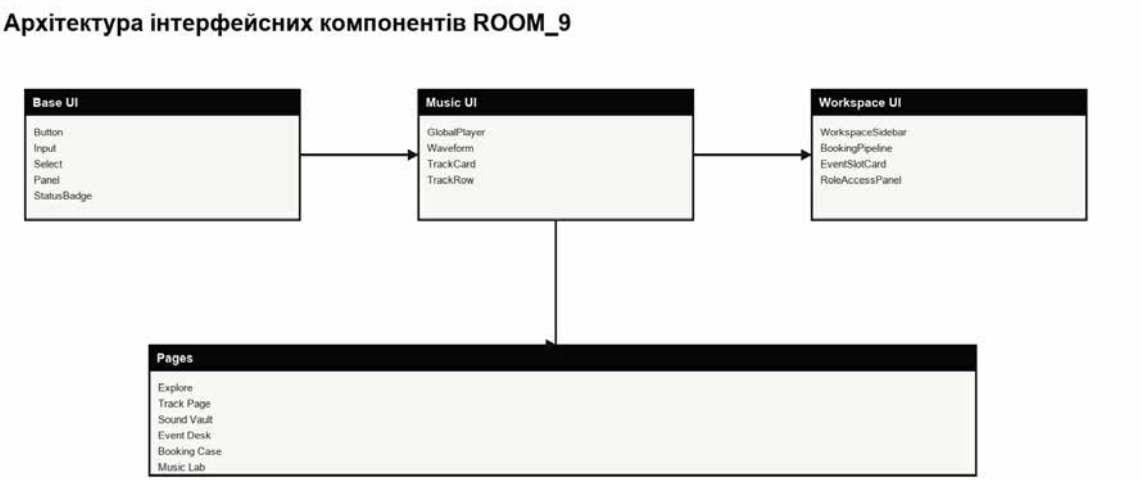

Рис. 3.1 – Архітектура інтерфейсних компонентів ROOM_9

Таблиця 3.2

Основні екрани та дії користувача в системі бронювання

Екран	**Головний об’єкт**	**Первинна дія**	**Вторинні дії**
Explore	Звуковий результат	Прослухати	Зберегти, queue, open track, open artist

Екран	**Головний об’єкт**	**Первинна дія**	**Вторинні дії**
Track Page	Трек і waveform	Обрати момент	Save Moment, Artist Dossier, Use as Brief
Sound Vault	Особистий музичний архів	Відкрити збережений об’єкт	Playlist, queue, booking-ready
Artist Dossier	Артист і sound proof	Прослухати latest set	Follow, SoundCloud, trust drawer, book DJ
Event Desk	Подія і lineup slots	Прикріпити Atmosphere Brief	Choose DJ, send request, open case
Booking Case	Професійна заявка	Виконати next action	Chat, rider, escrow preview

3.5 Алгоритмізація та програмування модулів

Глобальний player реалізований як постійний музичний об’єкт платформи. Він підтримує play/pause, queue, next/previous, repeat-one, progress line, selected moment і дію Use Brief. Важливо, що букінгова дія не замінює базову музичну поведінку.

Моменти Intro, Build, Peak і Closing формуються динамічно з урахуванням duration треку. Це захищає систему від некоректних timestamp-значень для коротких аудіофайлів.

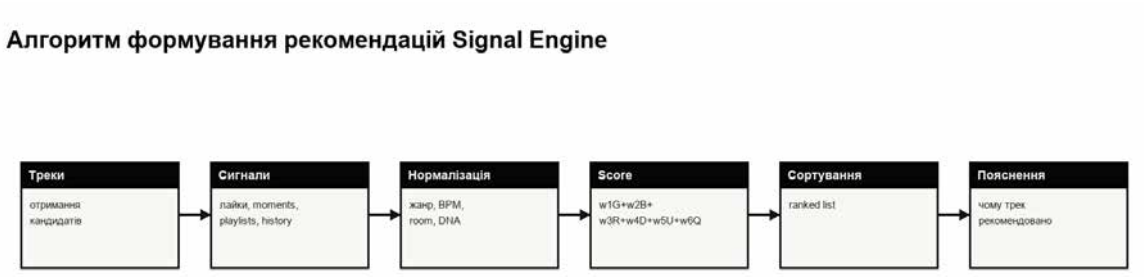

Рис. 3.2 – Алгоритм формування рекомендацій Signal Engine

Під час програмування модулів враховано, що користувач може почати сценарій із різних точок. Наприклад, booking request може створюватися з Track Page, зі збереженого моменту в Sound Vault або зі слоту Event Desk. Тому форма заявки повинна приймати `workId`, `t`, `savedMomentId`, `eventId` і `slotId`, якщо ці значення вже існують у контексті.

3.6 Реалізація Sound Vault

Sound Vault — персональний музичний архів користувача. Він розділений на Tracks, Moments, Playlists, Uploads і Network. Tracks містить збережені та лайкнуті треки, Moments — Atmosphere Briefs, Playlists — користувацькі добірки, Uploads — DJ-контент, Network — артистів і рекомендаційні групи.

На рівні UX Sound Vault не повинен відчуватися як одна велика вкладка. Кожна його частина має власну функцію: Tracks відповідає за прослуховування, Moments — за майбутні професійні наміри, Playlists — за власну систему добірок, Uploads — за DJ-матеріали, Network — за артистів і рекомендаційні зв’язки. Такий поділ дає змогу поступово розвивати продукт без перевантаження стартового екрана.

3.7 Реалізація Booking CRM та Event Desk

Booking CRM подано як професійну дошку зі станами New Requests, Negotiating, Rider Needed, Contract Next, Escrow Preview і Confirmed. Event Desk показує подію та слоти Opening, Support, Peak, Closing і Stream. До кожного слоту можна прикріпити Atmosphere Brief, обрати DJ і створити заявку.

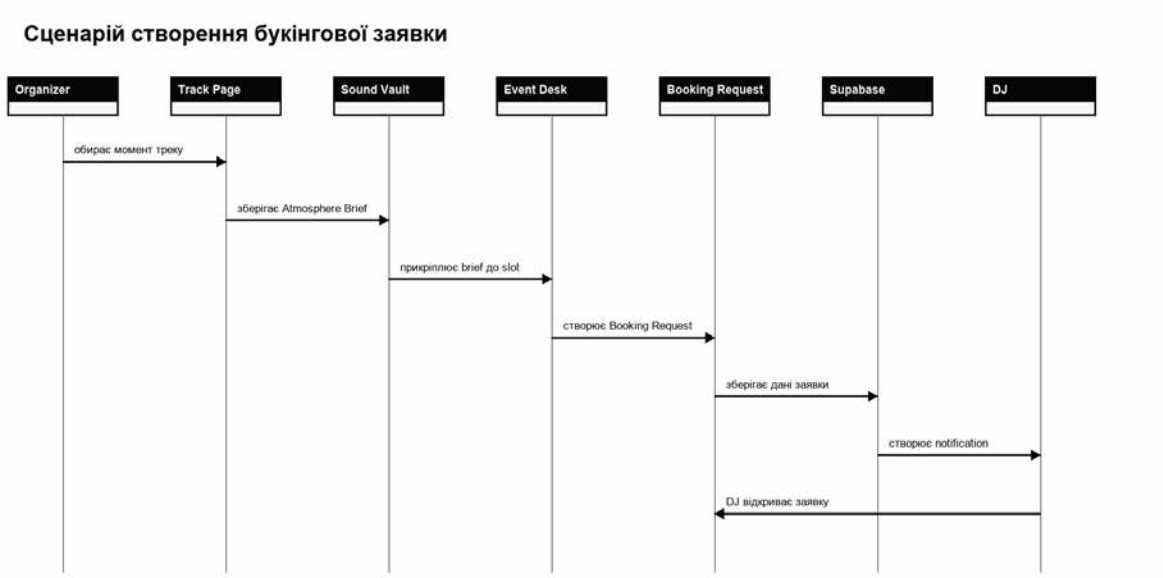

Рис. 3.3 – Сценарій створення букінгової заявки

У демонстраційній версії база може зберігати спрощені статуси, але інтерфейс показує розширену lifecycle-модель. Це дозволяє захистити продуктову логіку без передчасного ускладнення схеми. Для production-версії доцільно додати окремий журнал змін статусів, audit trail і точні права на кожну дію.

3.8 Реалізація Music Lab та Signal Engine

Music Lab — робочий простір DJ для опису музичного матеріалу. У демонстраційній версії він містить waveform preview, cue points, moments editor, energy, room type, sound DNA, EQ sketch і analysis summary. Повноцінна DSP-обробка й real-time equalizer належать до V3/V4.

EQ sketch у поточній роботі є не аудіопроцесором, а структурованим описом частотного характеру треку. Він допомагає зафіксувати, чи має матеріал сильний sub, агресивний mid, м'який high або відкритий air. Ці ознаки можуть бути використані у Signal Engine як частина sound DNA.

Таблиця 3.3

Сигнали, які використовуються в Signal Engine

Сигнал	**Джерело**	**Використання в рекомендаціях**
Genre	works, user_sound_profile	Підбір треків і артистів за стилем
BPM	works, track_audio_features	Темпова відповідність треку події
Room type	saved_moments, Music Lab	Відповідність warehouse, club, basement, open air
Sound DNA	Music Lab, metadata	Порівняння характеру звучання
Saved moments	Sound Vault	Ознака високого booking intent
Playlist additions	playlist_tracks	Стійкий інтерес користувача
Event slot	event_lineup_slots	Підбір під Opening, Peak або Closing

3.9 Реалізація автентифікації та role unlock

Автентифікація підтримує listener-first реєстрацію, email verification, login, forgot password, update password і збереження next URL. Role Verification Center відкриває DJ, Organizer і Venue інструменти через checklist готовності профілю.

Важливо, що професійні сторінки не мають відкриватися через пряме введення URL для користувача без відповідного доступу. Замість помилки або порожньої сторінки система повинна показати lock state з поясненням, який етап потрібно пройти: підтвердити роль, заповнити профіль, завантажити перший трек, додати обкладинку або райдер.

Таблиця 3.4

Стан ролей та поведінка інтерфейсу

Стан ролі	**Приклад доступу**	**Поведінка інтерфейсу**
Listener	Explore, Track Page, Sound Vault	Професійні сторінки показують locked state
DJ pending	Uploads частково, Music Lab preview	Показано missing requirements
DJ verified	Artist profile, releases, rider, booking response	Професійні дії активні
Organizer verified	Event Desk, Booking CRM, Calendar	Можна створювати події та заявки
Venue verified	Venue profile, recurring events	Доступні venue-specific поля

3.10 Реалізація розгортання та оптимізації

Локальне середовище зафіксоване на localhost:3001. Публічне розгортання виконується через GitHub Actions, OpenNext і Cloudflare Workers. У процесі експлуатації враховано ризик Worker CPU limit: демо-аудіо має бути статичним, списки — обмеженими, а Supabase-запити — точними за колонками.

Для deployment pipeline використовується GitHub як джерело істини. Це спрощує співпрацю й дозволяє Cloudflare брати актуальну версію з репозиторію. Секрети зберігаються в GitHub Secrets або в налаштуваннях

провайдерів, а не в коді. Таке розміщення відповідає базовим вимогам production-підготовки.

Таблиця 3.5

Ризики продуктивності та способи оптимізації

Ризик продуктивності	**Причина**	**Рішення**
Worker CPU limit	Важкі обчислення під час server rendering	Статичні assets, query limits, мінімізація RSC-навантаження
Повільне завантаження медіа	Великі cover/audio	Storage CDN, lazy loading, контроль розміру файлів
Дублювання запитів	Однакові дані в кількох компонентах	Спільні data utilities і select потрібних колонок
Mobile audio failure	Спроба автозапуску без user gesture	Єдиний audio element і запуск після дії користувача
Text overflow	Довгі назви артистів і релізів	min-width:0, line-clamp, wrapping, safe max-widths

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування виконується за наскрізними сценаріями. Для слухача перевіряється Explore → Track Page → Save Moment → Sound Vault. Для організатора або venue — Sound Vault Moment → Event Desk → Booking Request → Booking Case. Для DJ — Upload Track → Music Lab → Create Release → Public Profile → Rider → Accept/Decline.

Окремо перевіряються сценарії відмови: відсутність Supabase-з'єднання, відсутність Storage bucket, заблокована роль, порожній Sound Vault, відсутній rider, некоректний timestamp і довга назва релізу. Такі стани мають показувати зрозуміле повідомлення, а не ламати інтерфейс.

Таблиця 4.1

Тестові сценарії інформаційної системи

Сценарій	**Очікуваний результат**	**Критерій приймання**
Реєстрація	Користувач створюється як listener	Профіль існує, професійні ролі закриті
Підтвердження пошти	Перехід через /auth/callback	Сесія створюється, next URL зберігається
Відновлення пароля	Лист веде до /update-password	Новий пароль зберігається
Прослуховування	Player запускає трек і працює між сторінками	Pause/resume стабільні
Save Moment	Момент зберігається в Sound Vault	Timestamp не перевищує duration
Event Desk	Слот приймає brief	Booking request отримує eventId і slotId
Booking Case	Кейс містить статус, rider, escrow preview	Учасники бачать тільки пов'язаний кейс

Таблиця 4.2

Типи перевірок інформаційної системи

Тип перевірки	**Приклад дії**	**Очікуваний результат**
Функціональна	Like/unlike треку	Стан оновлюється без дублювання
Рольова	Listener відкриває Booking CRM	Показано locked state
Інтеграційна	Saved moment додається до event slot	Заявка створюється з контекстом
RLS	Другий акаунт читає чужий playlist	Дані не повертаються
Storage	DJ завантажує avatar/cover	Зображення синхронізується у профілі й Explore
Візуальна	Довгий release title	Текст не перекриває Atmosphere Brief

4.2 Вимоги до апаратного та програмного забезпечення

Для користувача достатньо сучасного браузера з підтримкою HTML5 Audio і JavaScript [23]. Для розробника потрібні Node.js, npm, Git, доступ до Supabase-проєкту та Cloudflare/GitHub deployment-параметрів.

Мобільна версія розглядається як браузерна адаптація, а не окремий native-застосунок. Для аудіо це означає обов’язковий запуск після дії користувача, компактний player, відсутність перекриття основного контенту й адаптивні touch-цілі для play, queue, like і save moment.

Таблиця 4.3

Вимоги до апаратного та програмного забезпечення

Компонент	**Мінімальна вимога**	**Рекомендована вимога**
Клієнтський пристрій	Сучасний браузер	Chrome, Safari, Edge або Firefox актуальної версії
Локальна розробка	Node.js, npm, .env.local	Node.js 20+, Git, стабільний термінал
База даних	Supabase PostgreSQL	Активний Supabase-проєкт із застосованим schema.sql
Storage	Buckets tracks, images, documents	MIME limits і owner-folder policies
Деплой	Cloudflare Workers	GitHub Actions із Cloudflare API token

4.3 Склад інсталяційного пакету

Таблиця 4.4

Склад інсталяційного пакету

Елемент	**Призначення**
app/	Маршрути Next.js App Router
components/	Reusable UI, player, waveform, workspace components
lib/	Supabase client, auth flow, recommendations, track moments
supabase/schema.sql	Таблиці, індекси, RLS, storage buckets
docs/	Roadmap, deployment notes, RLS audit
.github/workflows/	Автоматизоване розгортання

Елемент	Призначення
package.json	Скрипти dev, lint, build, deploy

4.4 Аналіз ризиків та обмежень

Таблиця 4.5

Аналіз ризиків та обмежень

Категорія ризику	Прояв	Запобіжний захід
Продуктовий	Користувач сприймає Sound Reference як продаж аудіо	Використовувати терміни Atmosphere Brief, Use as Brief, Book DJ
Безпековий	Приватні moments видно іншому акаунту	RLS audit і live-перевірка двома акаунтами
Фінансовий	Неповний payment flow	Залишити escrow preview до інтеграції провайдера
Технічний	Cloudflare CPU limit	Статичні assets, обмежені запити, оптимізація SSR
UX	Dashboard перевантажений	Command Center тільки для next actions і blocked states
Медіа	Нестабільний mobile audio	Запуск через user gesture і єдиний audio element

4.5 Рекомендації щодо супроводу

Після впровадження потрібно регулярно перевіряти RLS-політики, Storage buckets, auth redirects, продуктивність Cloudflare Worker і узгодженість UI-компонентів. Для майбутніх версій заплановано реальні платежі, signed URLs для документів, live-stream infrastructure, admin moderation, розширений Signal Engine і автоматизовані E2E-тести.

Для розвитку продукту найбільш важливими залишаються три напрями. Перший — стабілізація music-first досвіду: player, Sound Vault, playlist CRUD, queue, mobile playback і cover sync. Другий — професійний booking-шар: Event Desk, booking lifecycle, rider, escrow preview і календар конфліктів. Третій — алгоритмічний шар: Music Lab, track_audio_features, user_sound_profile і зрозумілі рекомендації для слухача та організатора.

ВИСНОВКИ

1. Проаналізовано предметну область бронювання діджейсетів та професійного DJ-букінгу. Встановлено, що наявні рішення закривають окремі частини процесу, але не поєднують звуковий контекст, подію, booking-заявку і booking case в одному сценарії.
2. Сформовано модель інформаційної системи бронювання діджейсетів, у якій музичний контекст є допоміжним шаром для створення точнішої booking-заявки.
3. Розроблено логічну модель даних із profiles, role access, dj_profiles, works, releases, saved moments, playlists, events, event slots і bookings.
4. Спроєктовано архітектуру на основі Next.js, React, TypeScript, Tailwind CSS, Supabase PostgreSQL, Supabase Auth, Supabase Storage, OpenNext і Cloudflare Workers.
5. Реалізовано основні модулі прототипу: Explore, Track Page, Artist Dossier, Release Page, Sound Vault, Event Desk, Booking CRM, Booking Case, Calendar Timeline, Settings, Music Lab і Signal Engine.
6. Описано детерміновану рекомендаційну модель Signal Engine з формальними вхідними множинами, вагами, функцією score та псевдокодом.
7. Підготовлено рекомендації щодо тестування, впровадження, безпеки, продуктивності й подальшого розвитку системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Next.js Documentation. URL: https://nextjs.org/docs (дата звернення: 20.05.2026).
2. React Documentation. URL: https://react.dev/reference/react (дата звернення: 20.05.2026).
3. TypeScript Handbook. URL: https://www.typescriptlang.org/docs/ (дата звернення: 20.05.2026).
4. Supabase Documentation. URL: https://supabase.com/docs (дата звернення: 20.05.2026).
5. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 20.05.2026).
6. Supabase Row Level Security Guide. URL: https://supabase.com/docs/guides/database/postgres/row-level-security (дата звернення: 20.05.2026).
7. Cloudflare Workers Documentation. URL: https://developers.cloudflare.com/workers/ (дата звернення: 20.05.2026).
8. OpenNext Documentation. URL: https://opennext.js.org/ (дата звернення: 20.05.2026).
9. MDN Web Docs. HTMLAudioElement. URL: https://developer.mozilla.org/en-US/docs/Web/API/HTMLAudioElement (дата звернення: 20.05.2026).
10. OWASP Application Security Verification Standard. URL: https://owasp.org/www-project-application-security-verification-standard/ (дата звернення: 20.05.2026).
11. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. 3rd ed. New York : Springer, 2022.
12. Pazzani M. J., Billsus D. Content-Based Recommendation Systems. The Adaptive Web. Berlin : Springer, 2007. P. 325–341.

13. Aggarwal C. C. Recommender Systems: The Textbook. Cham : Springer, 2016.
14. Nielsen J. 10 Usability Heuristics for User Interface Design. Nielsen Norman Group. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 20.05.2026).
15. Norman D. The Design of Everyday Things. Revised and expanded edition. New York : Basic Books, 2013.
16. W3C. Web Content Accessibility Guidelines (WCAG) 2.2. URL: https://www.w3.org/TR/WCAG22/ (дата звернення: 20.05.2026).
17. SoundCloud for Artists. URL: https://artists.soundcloud.com/ (дата звернення: 20.05.2026).
18. Spotify for Developers Documentation. URL: https://developer.spotify.com/documentation/web-api (дата звернення: 20.05.2026).
19. Resident Advisor. URL: https://ra.co/ (дата звернення: 20.05.2026).
20. DICE. URL: https://dice.fm/ (дата звернення: 20.05.2026).
21. ROOM_9 project repository. URL: https://github.com/sachkovladimir-crypto/room-9 (дата звернення: 20.05.2026).
22. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 22.05.2026).
23. Web-технології та Web-дизайн: Застосування мови HTML для створення електронних ресурсів : навчальний посібник. Вид. 2-ге. Київ : Видавництво Ліра-К, 2020. 212 с.
24. Бородкіна І. Л., Бородкін Г. О. Теорія алгоритмів : навчальний посібник. Київ : Центр учбової літератури, 2018. 184 с.
25. Бородкіна І. Л., Бородкін Г. О. Інженерія програмного забезпечення : навчальний посібник. Київ : Центр учбової літератури, 2018. 204 с.

ДОДАТОК А

Фрагменти SQL-схеми бази даних

```
create table if not exists public.profiles (
  id uuid primary key references auth.users(id) on delete cascade,
  email text,
  display_name text,
  created_at timestamptz default now()
);


create table if not exists public.works (
  id uuid primary key default gen_random_uuid(),
  dj_id uuid references public.dj_profiles(id) on delete cascade,
  title text not null,
  audio_url text,
  genre text,
  bpm integer,
  visibility text default 'public'
);


create table if not exists public.saved_moments (
  id uuid primary key default gen_random_uuid(),
  user_id uuid references public.profiles(id) on delete cascade,
  work_id uuid references public.works(id) on delete cascade,
  timestamp_seconds integer not null,
  moment_label text,
  atmosphere_brief text
);


create table if not exists public.events (
  id uuid primary key default gen_random_uuid(),
  organizer_id uuid references public.profiles(id) on delete cascade,
  title text not null,
  event_date date,
  venue text,
  city text
);


create table if not exists public.event_lineup_slots (
  id uuid primary key default gen_random_uuid(),
  event_id uuid references public.events(id) on delete cascade,
  slot_type text not null,
  saved_moment_id uuid references public.saved_moments(id) on delete set null,
  booking_id uuid references public.bookings(id) on delete set null
);


create table if not exists public.bookings (
  id uuid primary key default gen_random_uuid(),
  dj_id uuid references public.dj_profiles(id),
  organizer_id uuid references public.profiles(id),
  source_work_id uuid references public.works(id),
  source_saved_moment_id uuid references public.saved_moments(id),
  status text default 'pending'
);
```

ДОДАТОК Б

Фрагменти політик Row Level Security

```
create policy "saved_moments_select_own"
  on public.saved_moments for select
  using (user_id = auth.uid());


create policy "playlists_manage_own"
  on public.playlists for all
  using (user_id = auth.uid())
  with check (user_id = auth.uid());


create policy "bookings_select_related"
  on public.bookings for select
  using (
    organizer_id = auth.uid()
    or exists (
      select 1 from public.dj_profiles
      where dj_profiles.id = bookings.dj_id
      and dj_profiles.user_id = auth.uid()
    )
  );
```

ДОДАТОК В

Скріншоти інтерфейсу ROOM_9

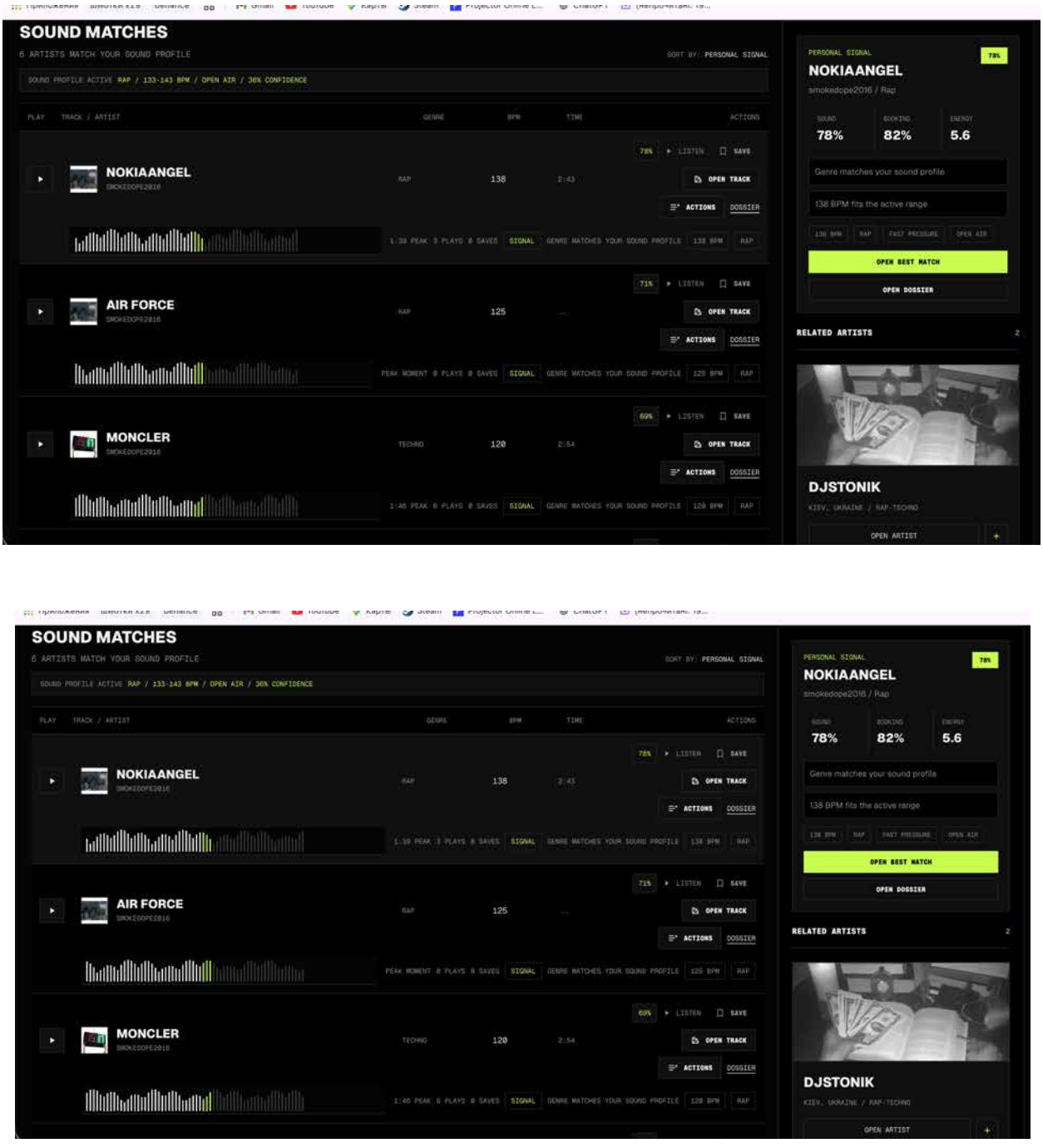

Рис. В.1 – Explore page із пошуком, результатами та Global Player

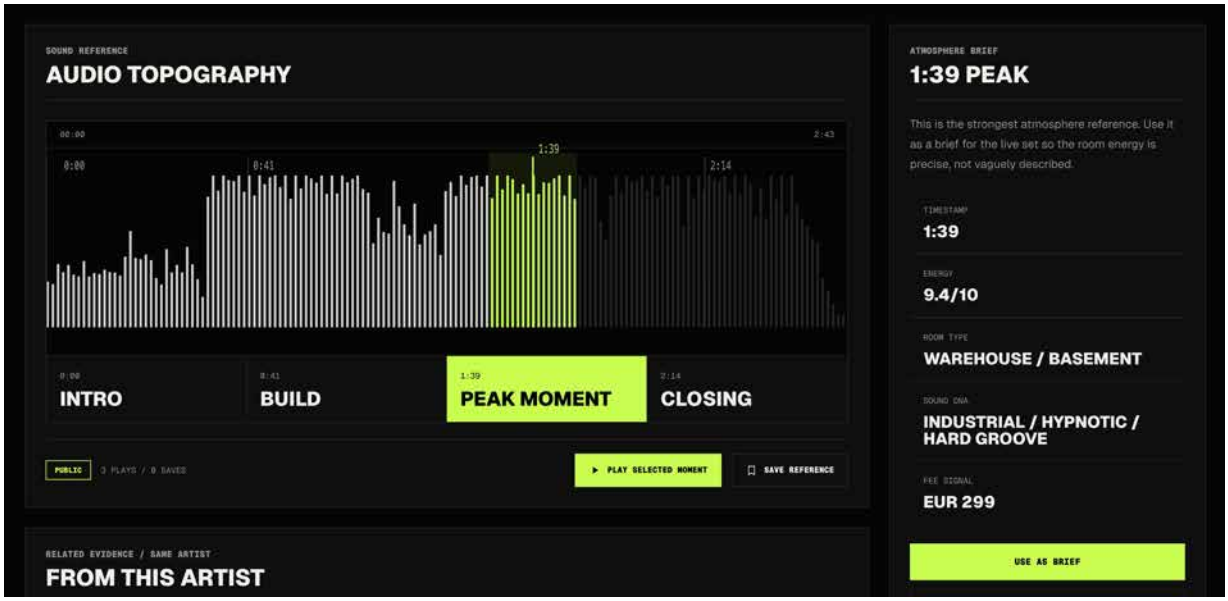

Рис. В.2 – Track Page із waveform, моментами та Sound Reference

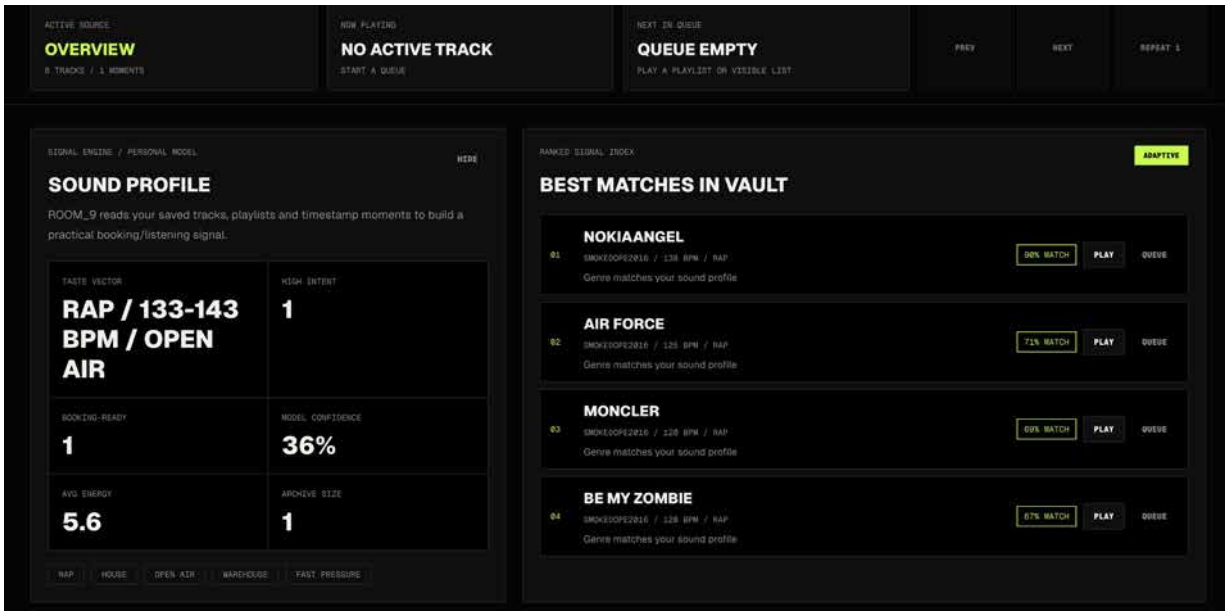

Рис. В.3 – Sound Vault із персональними звуковими референсами

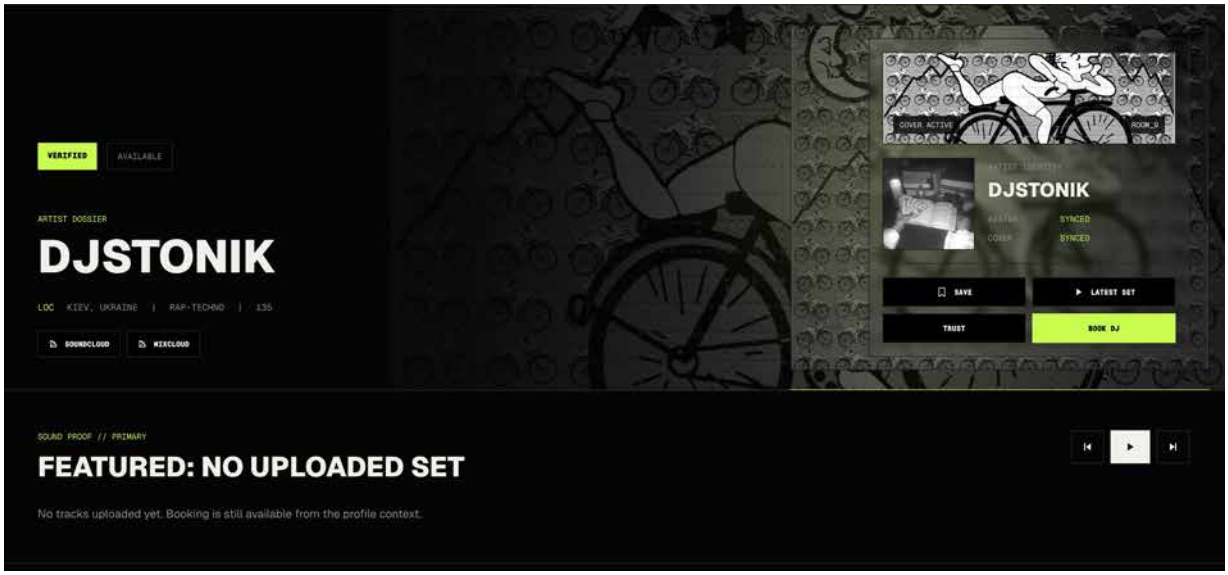

Рис. В.4 – Artist Dossier / DJ Profile із Sound Proof та booking CTA

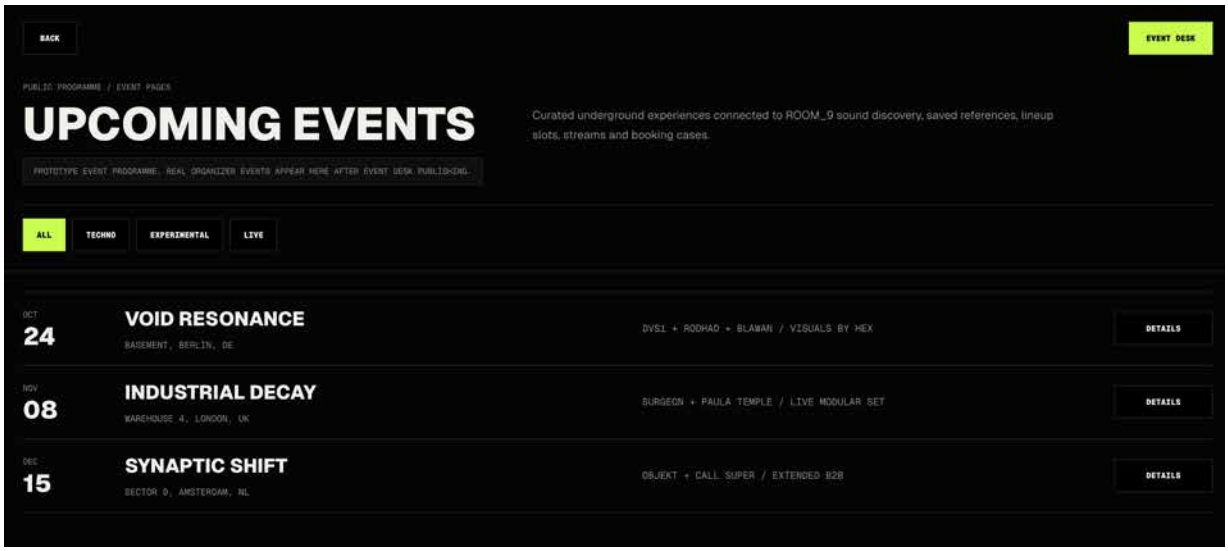

Рис. В.5 – Event Desk із подіями та слотами лайнапу

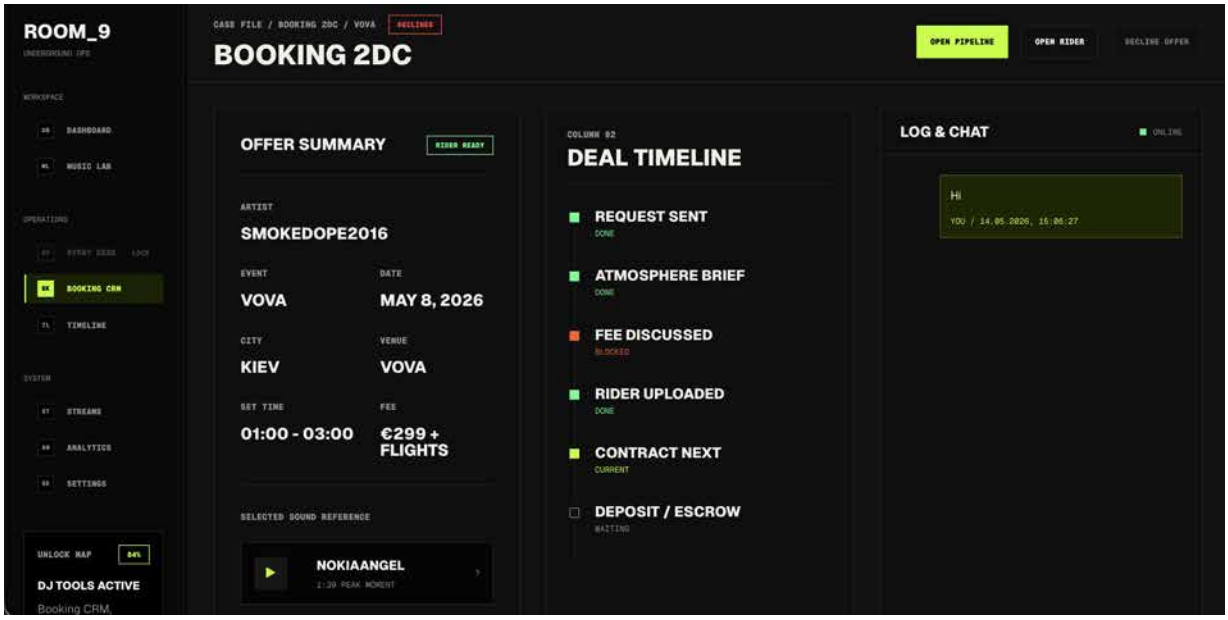

Рис. B.6 – Booking Case із факторами події, статусом та escrow preview

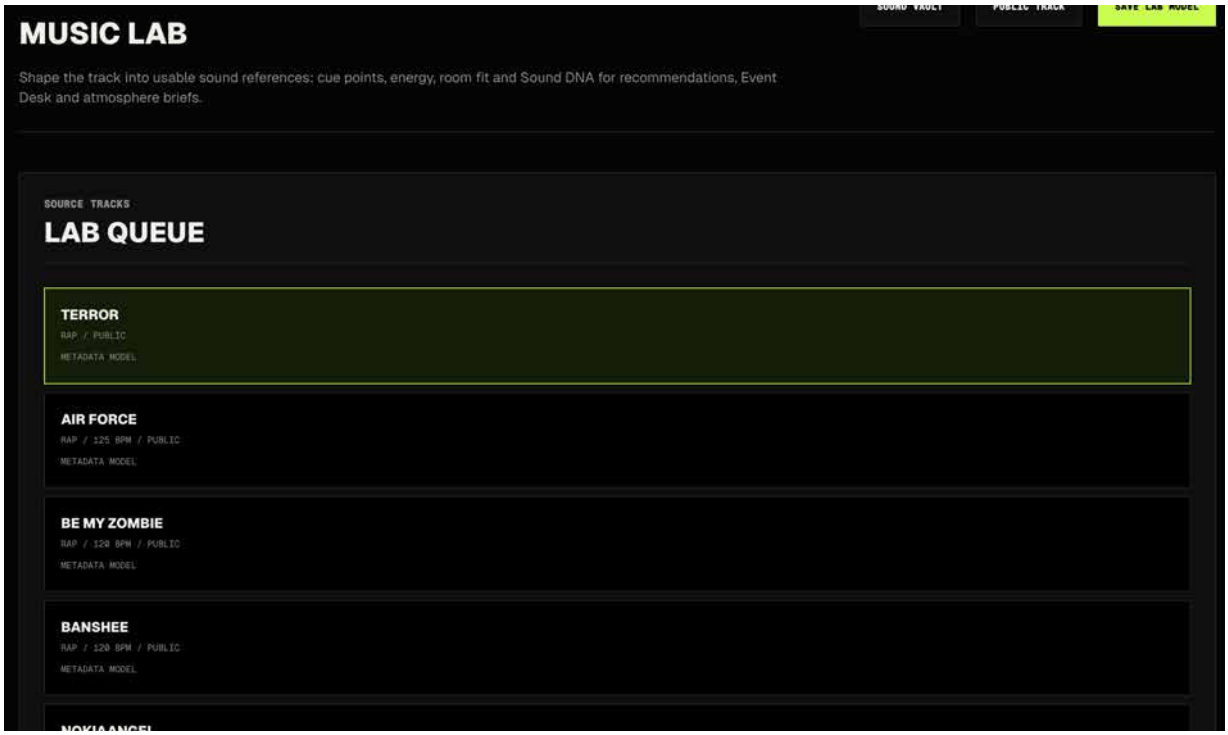

Рис. B.7 – Music Lab із cue points, EQ sketch та analysis summary

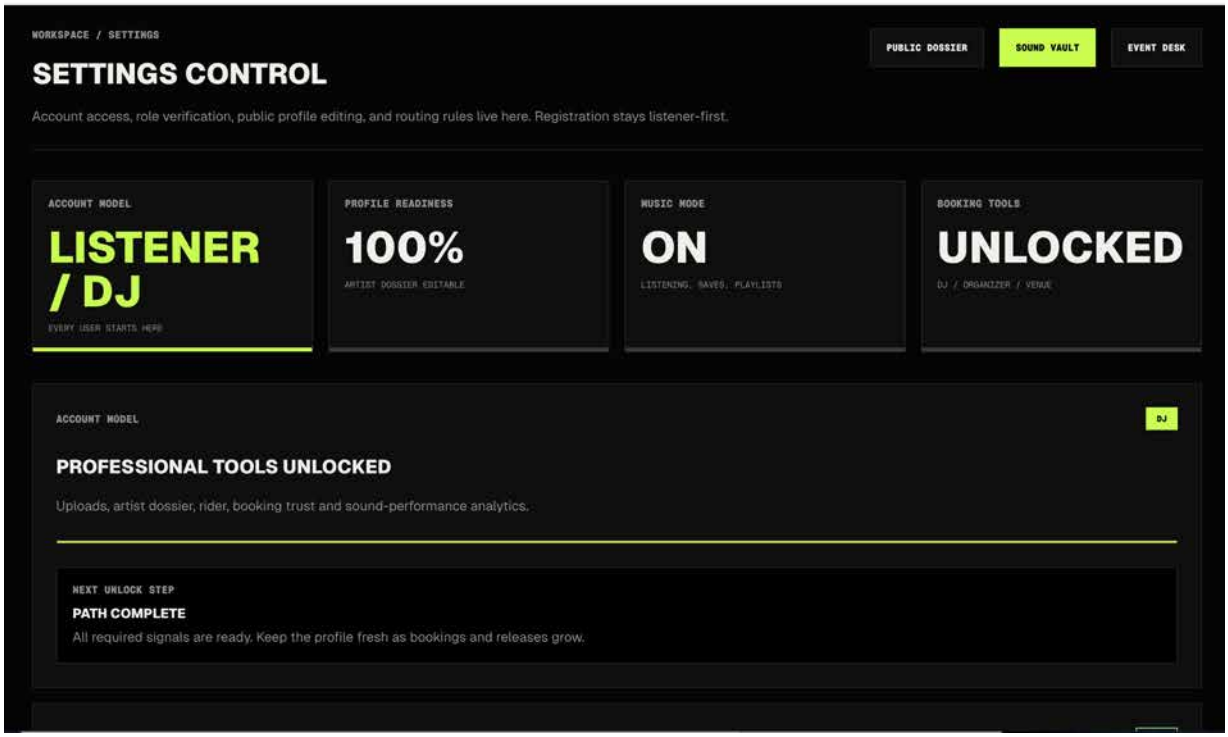

Рис. B.8 – Settings / Role Verification Center

ДОДАТОК Г

Фрагмент алгоритму Signal Engine

```
Algorithm SignalEngineRecommendation
Input: tracks T, user u, event slot e
Output: ranked list of recommended tracks


for each track t in T:
    features = extract_features(t)
    user_match = compare(features, sound_profile(u))
    event_match = compare(features, slot_requirements(e))
    score = w1*G + w2*B + w3*R + w4*D + w5*U + w6*Q
    explanation = build_explanation(features, user_match, event_match)
return sort_by_score_descending(T)
```

ДОДАТОК Д

Інструкція локального запуску

1. Встановити залежності командою `npm install`.
2. Створити файл `.env.local` у корені проєкту.
3. Додати `NEXT_PUBLIC_SUPABASE_URL`.
4. Додати `NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY`.
5. Застосувати SQL-сценарій `supabase/schema.sql` у Supabase SQL Editor.
6. Запустити застосунок командою `npm run dev`.
7. Відкрити `http://localhost:3001` у браузері.

ДОДАТОК Е

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Сачко Володимир Володимирович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

- Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Інформаційна система бронювання діджейсетів» є результатом моєї особистої праці.
- Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
- Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [ ] інструменти ШІ ChatGPT для:
        - [ ] перекладу іншомовних джерел;
        - [x] стилістичного редагування та перевірки граматики;
        - [ ] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«_24_» __05_____ 2026 р. _____ **Володимир САЧКО**

(підпис)