МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

ХІЛІНСЬКИЙ ОЛЕКСАНДР БОРИСОВИЧ

ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОФАЙЛІВ

Спеціальність: 122 Комп'ютерні науки

Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
ГРИШАНОВИЧ ТЕТЯНА ОЛЕКСАНДРІВНА
кандидат фіз.-мат наук, доцент кафедри
комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №__
засідання кафедри комп'ютерних наук
та кібербезпеки від _____ 2025 р.
Завідувач кафедри
(_____) Гришанович Т. О.

ЛУЦЬК-2025

ЗМІСТ

ВСТУП

У сучасному цифровому світі музичні вебплатформи стали невід'ємною частиною повсякденного життя мільйонів користувачів. Зростання популярності потокових музичних сервісів зумовлено зручністю доступу до величезних музичних бібліотек, можливістю персоналізації контенту та соціальними функціями обміну музичними уподобаннями. За даними міжнародних досліджень, ринок потокових музичних сервісів продовжує демонструвати стабільне зростання, що підкреслює актуальність розробки інноваційних рішень у цій сфері.

Сучасні користувачі потребують інтуїтивно зрозумілих інтерфейсів, високої якості звучання, швидкого завантаження контенту та можливості створення персональних плейлистів. Водночас існуючі рішення часто мають обмеження щодо функціональності або складності використання, що створює потребу у розробці нових вебплатформ з урахуванням найкращих практик користувацького досвіду та сучасних вебтехнологій.

Метою кваліфікаційної роботи є проєктування та розробка функціональної вебплатформи для прослуховування музики з інтуїтивним користувацьким інтерфейсом та ефективною системою управління аудіоконтентом.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

- проаналізувати сучасні тенденції розвитку вебплатформ для прослуховування музики;
- розробити архітектуру вебплатформи з урахуванням сучасних стандартів веброзробки;
- спроектувати базу даних для зберігання інформації про музичний контент та користувачів;
- реалізувати основний функціонал платформи з використанням HTML, CSS, JavaScript та PHP;
- протестувати працездатність розробленої системи та виявити можливості для оптимізації.

Об’єктом дослідження є методи та технології проєктування і розробки вебплатформи для прослуховування музики з використанням сучасних вебтехнологій та принципів користувацького досвіду.

Предметом дослідження є процес створення вебплатформ для потокового прослуховування музичного контенту.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБПЛАТФОРМ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОКОНТЕНТУ

1.1. Аналіз сучасного стану онлайн-сервісів для прослуховування музики

Індустрія онлайн-прослуховування музики переживає період динамічного розвитку, зумовленого зростанням популярності потокових сервісів та зміною споживчих звичок користувачів. За останні десятиріччя спостерігається перехід від фізичних носіїв та цифрових завантажень до моделі потокового доступу до музичного контенту [1].

Сучасні тенденції розвитку галузі характеризуються кількома ключовими напрямами. Перш за все, відбувається масштабна цифрова трансформація музичної індустрії, що проявляється у зростанні частки цифрових доходів від продажу музики. Згідно з дослідженнями International Federation of the Phonographic Industry, потокові сервіси забезпечують понад 65% глобальних доходів музичної індустрії [2].

Технологічний прогрес значно впливає на якість користувацького досвіду онлайн-сервісів. Впровадження технологій штучного інтелекту та машинного навчання дозволяє створювати персоналізовані рекомендації, що підвищує залученість користувачів та тривалість сесій прослуховування. Алгоритми аналізу музичних переваг стають все більш складними, враховуючи не лише історію прослуховування, але й контекстуальні фактори, такі як час доби, настрій та активність користувача [3].

Соціальний аспект музичного споживання набуває все більшого значення. Користувачі прагнуть ділитися своїми музичними відкриттями, створювати спільні плейлисти та взаємодіяти з іншими любителями музики. Це призводить до інтеграції соціальних функцій у музичні платформи та розвитку музичних спільнот навколо артистів та жанрів.

Мобільність стає критично важливим фактором успіху музичних сервісів. Зростання використання смартфонів та планшетів вимагає оптимізації

вебплатформ для мобільних пристроїв та розробки responsive дизайну. Progressive Web Apps технології дозволяють створювати вебдодатки з функціональністю, близькою до нативних мобільних програм [4].

Якість звучання залишається одним з ключових конкурентних переваг. Розвиток технологій стиснення аудіо та збільшення пропускної здатності інтернет-з'єднань дозволяє пропонувати високоякісний звук, включаючи форми Hi-Res Audio та просторове аудіо. Водночас важливим залишається баланс між якістю звуку та ефективністю передачі даних.

Глобалізація музичного контенту створює нові виклики та можливості. Локальний контент набуває важливості для залучення аудиторії в різних регіонах, що вимагає підтримки багатомовності та врахування культурних особливостей. Разом з тим, глобальні тренди та віральний контент сприяють міжкультурному обміну та відкриттю нової музики.

Аналіз сучасного стану показує, що успішні музичні вебплатформи повинні поєднувати високу технічну якість з глибоким розумінням потреб користувачів, забезпечувати персоналізований досвід та підтримувати соціальну взаємодію навколо музичного контенту.

1.2. Огляд технологій потокової передачі аудіоконтенту в вебсередовищі

Потокова передача аудіоконтенту в вебсередовищі базується на комплексі технологій, що забезпечують ефективну доставку та відтворення звукових файлів у реальному часі. Сучасні вебстандарти та протоколи дозволяють створювати високоякісні музичні додатки безпосередньо у браузері без необхідності встановлення додаткового програмного забезпечення.

HTML5 Audio API є фундаментальною технологією для роботи зі звуком у веббраузерах [5]. Елемент <audio> підтримує основні аудіоформати, включаючи MP3, AAC, OGG та WAV, забезпечуючи крос-браузерну сумісність. Web Audio API надає розширені можливості для обробки звуку, включаючи аналіз частотного спектру, застосування аудіоефектів та точне управління відтворенням.

Adaptive Bitrate Streaming (ABS) є ключовою технологією для забезпечення якісного прослуховування при різних умовах мережі [6]. Протокол автоматично адаптує якість потоку залежно від доступної пропускної здатності, забезпечуючи безперервне відтворення. Основними реалізаціями ABS для аудіо є HTTP Live Streaming (HLS) та Dynamic Adaptive Streaming over HTTP (DASH).

Content Delivery Networks (CDN) відіграють критичну роль у забезпеченні швидкої доставки аудіоконтенту глобальній аудиторії [7]. Технології кешування та географічного розподілу серверів мінімізують затримки та забезпечують стабільне відтворення. Провідні CDN, такі як CloudFlare, AWS CloudFront та Google Cloud CDN, пропонують спеціалізовані рішення для потокового контенту.

Progressive Web Apps (PWA) технології дозволяють створювати вебдодатки з можливостями, аналогічними нативним мобільним програмам [8]. Service Workers забезпечують офлайн-доступ до музики, Push API дозволяє надсилати сповіщення про нові релізи, а Media Session API інтегрується з системними медіа-контролами.

Для розробки сучасних музичних вебплатформ рекомендується використовувати комбінацію HTML5, CSS3 та JavaScript з фреймворками. Backend-рішення можуть базуватися на Node.js, PHP або Python з використанням баз даних MySQL або PostgreSQL для зберігання метаданих та Redis для кешування.

1.3. Особливості проєктування користувацького інтерфейсу музичних платформ

Користувацький інтерфейс є критично важливим елементом музичних вебплатформ, що безпосередньо впливає на залученість користувачів та загальне сприйняття сервісу. Проєктування ефективного інтерфейсу для музичних додатків вимагає врахування специфічних особливостей взаємодії з аудіоконтентом та поведінкових патернів користувачів [5].

Основні принципи проєктування користувацького інтерфейсу музичних платформ базуються на концепції мінімалізму та інтуїтивності. Користувачі повинні мати можливість швидко знаходити потрібний контент, керувати відтворенням та навігувати між різними секціями платформи без додаткового навчання. Це досягається через застосування знайомих візуальних метафор, логічну структуру навігації та зрозумілу ієрархію інформації.

Центральним елементом будь-якої музичної платформи є медіаплеєр, який повинен забезпечувати зручне управління відтворенням. Стандартні елементи керування включають кнопки відтворення, паузи, перемотування, регулювання гучності та індикатор прогресу відтворення. Сучасні тенденції передбачають мінімалістичний дизайн з великими, легко натискуваними елементами, особливо важливими для мобільних пристроїв.

Персоналізація інтерфейсу відіграє важливу роль у створенні унікального досвіду для кожного користувача. Адаптивні рекомендації, персональні плейлисти та збережені треки повинні бути легко доступними з головної сторінки.

Responsive дизайн є обов’язковою вимогою для сучасних музичних платформ. Інтерфейс повинен адаптуватися до різних розмірів екранів, зберігаючи функціональність та естетичну привабливість

Кольорова схема та типографіка значно впливають на сприйняття платформи. Темні теми стали популярними для музичних додатків, оскільки створюють атмосферу зосередженості на контенті та зменшують втому очей під час тривалого використання. Контрастність елементів повинна забезпечувати хорошу читабельність текстової інформації.

Доступність (accessibility) інтерфейсу забезпечує можливість використання платформи людьми з різними потребами. Підтримка навігації з клавіатури, сумісність з екранними читачами та альтернативний текст для зображень є важливими аспектами інклюзивного дизайну.

Простота використання залишається ключовим фактором успіху музичних платформ. Мінімалістичний підхід до дизайну, зосередження на основному функціоналі та уникнення перевантаження інтерфейсу зайвими елементами сприяють кращому користувацькому досвіду. Особливо це актуально для платформ,

що прагнуть забезпечити безперешкодний доступ до музики без відволікання уваги на рекламні блоки чи складні меню.

1.4. Огляд та аналіз аналогічних програмних розробок

Для визначення вимог до розробки музичної вебплатформи та формування конкурентних переваг необхідно провести комплексний аналіз існуючих програмних рішень аналогічного призначення. Дослідження провідних музичних сервісів дозволить виявити їх сильні та слабкі сторони, а також визначити можливості для створення простішої та доступнішої альтернативи.

Назва: Spotify

Розробник: Spotify Technology S.A.

Архітектура: Web application з підтримкою мобільних додатків (iOS, Android) та десктопних клієнтів (Windows, macOS, Linux)

Мова реалізації: Backend реалізований на Java та Python, frontend використовує JavaScript та React, мобільні додатки розроблені на нативних технологіях.

Spotify є найбільшою глобальною платформою потокового прослуховування музики з понад 500 мільйонами активних користувачів. Технічна реалізація базується на мікросервісній архітектурі з використанням Apache Cassandra для зберігання метаданих та власних алгоритмів рекомендацій на основі машинного навчання.

Переваги: Високоякісні персоналізовані рекомендації завдяки розвиненим алгоритмам, величезна музична бібліотека з регулярними оновленнями, інтуїтивний інтерфейс з зручною навігацією, розвинені соціальні функції для взаємодії користувачів, крос-платформенна сумісність та синхронізація між пристроями.

Недоліки: Присутність реклами у безкоштовній версії, що перериває прослуховування через кожні 3-6 треків, обмежені можливості безплатного використання включаючи відсутність офлайн-доступу та довільного вибору треків на мобільних пристроях, відносно складний інтерфейс для нових користувачів з великою кількістю функцій, необхідність стабільного інтернет-з’єднання для комфортного використання.

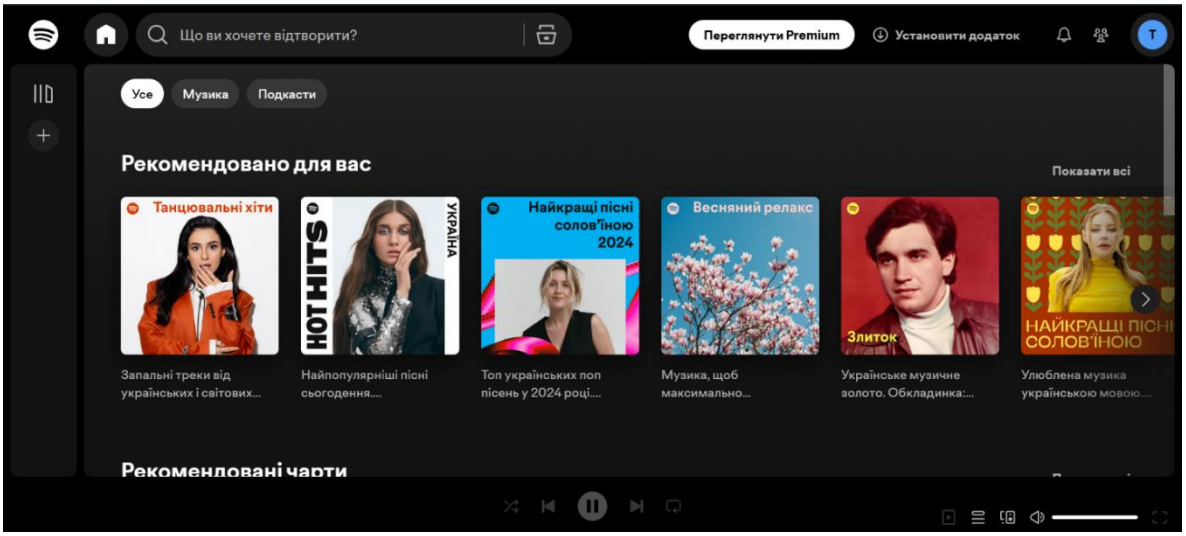


Рисунок 1.1 – Сторінка Spotify

Назва: Apple Music

Розробник: Apple Inc.

Архітектура: Web application з нативними додатками для iOS, macOS, Android, Windows та інтеграція з Apple TV

Мова реалізації: Swift для iOS/macOS додатків, Java для Android, вебтехнології (HTML5, CSS, JavaScript) для браузерної версії

Apple Music представляє преміум-сервіс від Apple Inc., що позиціонується з акцентом на високу якість звучання та глибоку інтеграцію з екосистемою Apple. Платформа пропонує тільки платну підписку без безкоштовної версії, що може створювати бар'єр для входу нових користувачів.

Переваги: Висока якість звучання з підтримкою Lossless Audio та просторового звуку, повна відсутність реклами в усіх тарифних планах, глибока інтеграція з екосистемою Apple для безшовного досвіду між пристроями, якісна курація контенту музичними експертами та редакторами, стабільна робота та швидкий відгук на всіх підтримуваних пристроях.

Недоліки: Повна відсутність безкоштовної версії, що створює високий бар'єр входу для нових користувачів, висока щомісячна вартість підписки порівняно з конкурентами, значно обмежена функціональність на пристроях поза екосистемою Apple, менш розвинені алгоритми автоматичних рекомендацій

порівняно з Spotify, складність використання для користувачів, незнайомих з продуктами Apple.

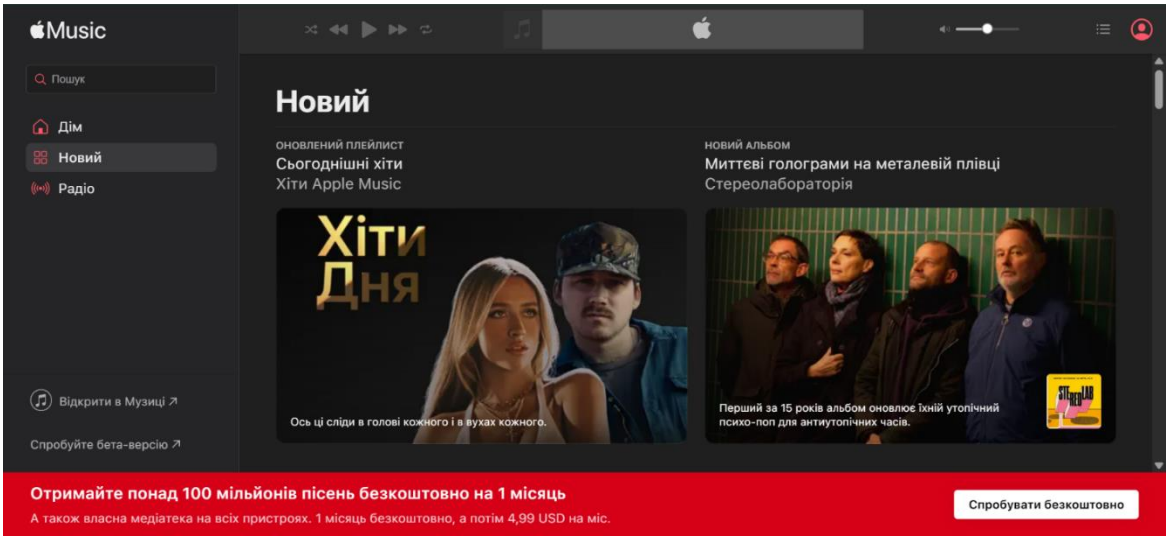


Рисунок 1.2 – Головна сторінка Apple Music

Назва: YouTube Music

Розробник: Google LLC

Архітектура: Web application з мобільними додатками для iOS та Android, інтеграція з YouTube платформою

Мова реалізації: Frontend реалізований на JavaScript з використанням Angular framework, backend на Java та Python, мобільні додатки на нативних технологіях

YouTube Music є музичним сервісом від Google LLC, що унікально поєднує потоковий аудіо з величезною відеобібліотекою YouTube. Платформа пропонує freemium модель з рекламою у безкоштовній версії та розширеними можливостями у YouTube Music Premium.

Переваги: Найбільша бібліотека музичного контенту в світі, включаючи рідкісні треки, live-виступи та неофіційні версії, можливість перегляду музичних відео в тому ж додатку, хороші рекомендації завдяки потужним алгоритмам Google, доступність безкоштовної версії з обмеженим функціоналом, синхронізація з особистим Google-акаунтом та іншими сервісами Google.

Недоліки: Присутність реклами у безкоштовній версії, що може переривати-ся частіше ніж у Spotify, складність знаходження високоякісних студійних версій треків через велику кількість користувацького контенту, перевантажений інтерфейс з множиною опцій та функцій, нижча максимальна якість звуку порівняно з конкурентами, іноді неточні або відсутні метадані через велику кількість користувацького контенту.

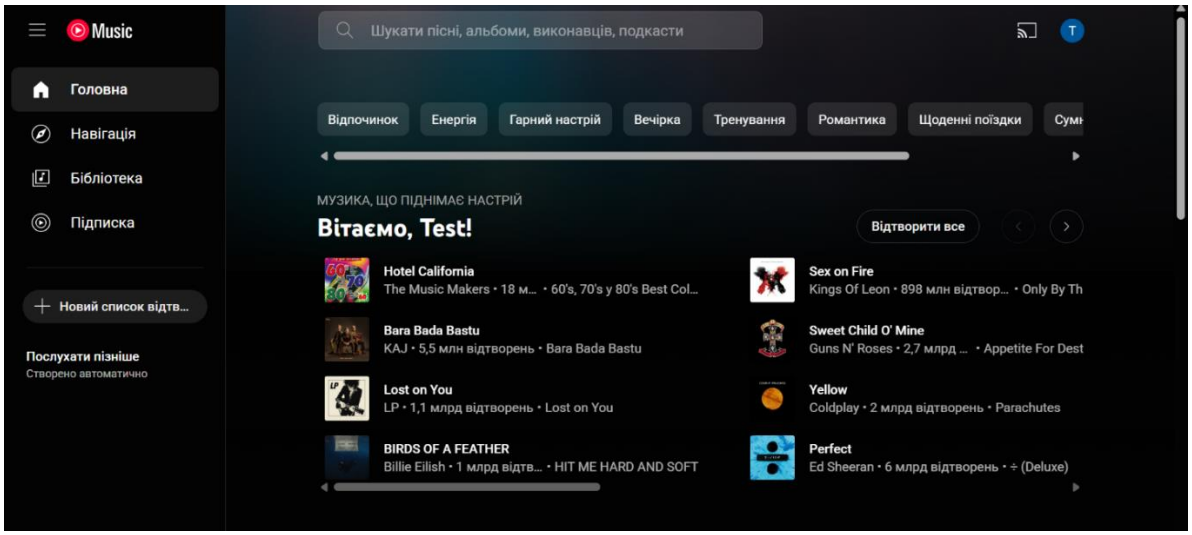


Рисунок 1.3 – Сторінка YouTube Music

Таблиця 1.1 Порівняльний аналіз провідних музичних платформ

Характеристика	Spotify	Apple Music	YouTube Music
Розробник	Spotify Technology S.A.	Apple Inc.	Google LLC
Архітектура	Web + мобільні/десктоп	Web + нативні додатки	Web + мобільні додатки
Мова реалізації	Java, Python, JavaScript	Swift, вебтехнології	JavaScript, Angular, Java
Модель монетизації	Freemium + Premium	Тільки підписка	Freemium + Premium
Бібліотека	100+ млн треків	100+ млн треків	100+ млн треків + відео
Якість звуку	До 320 kbps	До 24-біт/192 кГц	До 256 kbps
Безкоштовна версія	З рекламою та обмеженнями	Відсутня	З рекламою
Складність інтерфейсу	Середня	Середня	Висока
Реклама	Присутня у безкоштовній	Відсутня	Присутня у безкоштовній
Простота використання	Середня	Висока	Низька
API для розробників	Повний доступ	Обмежений	Повний доступ

Проведений детальний аналіз демонструє, що всі розглянуті платформи мають спільні недоліки, які створюють значні можливості для розробки альтернативного рішення.

На основі результатів аналізу можна сформулювати конкретні вимоги до програмного продукту, який має реалізувати мету та завдання роботи. Розроблювана музична вебплатформа повинна забезпечувати простий та інтуїтивний інтерфейс без зайвих елементів, зосереджуючись на основному функціоналі прослуховування музики.

Платформа має забезпечувати швидке завантаження та миттєвий відгук системи, мінімальні вимоги до інтернет-з’єднання для комфортного використання, базовий але повний набір функцій для прослуховування музики та управління плейлистами, адаптивний дизайн для коректного відображення на різних пристроях та розмірах екранів. Такий підхід дозволить створити унікальний продукт, який займе специфічну нішу серед користувачів, що цінують простоту використання та принципово не бажають платити за підписку або терпіти рекламні переривання під час прослуховування улюбленої музики.

РОЗДІЛ 2
ОСОБЛИВОСТІ ПРОЄКТУВАННЯ ТА РОЗРОБКИ ВЕБПЛАТФОРМИ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОФАЙЛІВ

2.1. Постановка задачі, призначення та вимоги до програмного засобу

2.1.1. Постановка задачі

Основною задачею розробки є створення вебплатформи, яка забезпечить користувачам простий та зручний доступ до прослуховування музичного контенту без реклами та додаткових обмежень. Платформа повинна надавати інтуїтивний інтерфейс для пошуку, відтворення та організації музичних композицій з акцентом на простоту використання та швидкодію.

Розробка включає створення повнофункціональної вебсистеми, що складається з клієнтської частини для взаємодії з користувачами, серверної частини для обробки запитів та управління даними, а також бази даних для зберігання інформації про музичний контент та користувачів. Система повинна забезпечувати стабільну роботу при одночасному доступі множини користувачів та підтримувати сучасні вебстандарти для кроссбраузерної сумісності.

2.1.2. Призначення програмного засобу

Вебсайт призначений для забезпечення безкоштовного доступу до прослуховування музики широкому колу користувачів незалежно від їх технічної підготовки та фінансових можливостей. Основною цільовою аудиторією є користувачі, які прагнуть отримати якісний досвід прослуховування музики без необхідності оплати підписки або перегляду реклами.

Платформа орієнтована на користувачів різних вікових груп, від підлітків до дорослих, які цінують простоту та зручність використання технологічних рішень. Особливу увагу приділено потребам користувачів з обмеженим досвідом роботи з комп’ютерними технологіями, для яких складні інтерфейси можуть створювати бар’єри у використанні.

2.1.3. Функціональні вимоги

Система повинна забезпечувати відтворення аудіофайлів у популярних форматах з підтримкою стандартних елементів управління відтворенням, включаючи відтворення, паузу, зупинку, перемотування та регулювання гучності. Функціонал пошуку має дозволяти користувачам знаходити музичні композиції за назвою треку, ім’ям виконавця або назвою альбому з підтримкою автодоповнення та фільтрації результатів.

Платформа повинна надавати можливість створення та управління персональними плейлистами з функціями додавання, видалення та зміни порядку композицій. Система управління користувачами має включати реєстрацію нових акаунтів, автентифікацію та збереження персональних налаштувань і плейлистів.

Інтерфейс має забезпечувати перегляд інформації про музичні композиції, включаючи назву треку, виконавця, альбом, тривалість та обкладинку. Система повинна підтримувати адаптивний дизайн для коректного відображення на різних пристроях та розмірах екранів.

2.1.4. Нефункціональні вимоги

Продуктивність системи має забезпечувати швидке завантаження сторінок з часом відгуку не більше 2 секунд при стандартному інтернет-з’єднанні. Початок відтворення музичної композиції після вибору користувачем повинен відбуватися протягом 3 секунд незалежно від розміру аудіофайлу.

Інтерфейс користувача повинен бути інтуїтивно зрозумілим та не потребувати додакового навчання для основних операцій. Навігація між розділами платформи має бути логічною та послідовною з мінімальною кількістю кліків для досягнення цільових функцій.

Система повинна коректно функціонувати у всіх сучасних веббраузерах, включаючи Chrome, Firefox, Safari та Edge. Адаптивний дизайн має забезпечувати зручність використання на настільних комп’ютерах, планшетах та смартфонах.

Надійність платформи передбачає стабільну роботу при одночасному доступі до 100 користувачів з можливістю масштабування при зростанні навантаження. Система повинна мати механізми обробки помилок та відновлення після збоїв без втрати користувацьких даних.

2.1.5 Вимоги до безпеки

Платформа повинна забезпечувати захист персональних даних користувачів згідно з сучасними стандартами інформаційної безпеки. Паролі користувачів мають зберігатися у зашифрованому вигляді з використанням сучасних алгоритмів хешування.

Система повинна мати захист від основних типів вебатак, включаючи SQL-ін’єкції, XSS-атаки та CSRF-атаки. Доступ до адміністративних функцій має бути обмежений авторизованими користувачами з відповідними правами.

2.2. Вибір моделі розробки програмного засобу для прослуховування аудіофайлів

2.2.1. Аналіз моделей життєвого циклу програмного забезпечення

Життєвий цикл програмного забезпечення представляє собою структурований підхід до розробки, що визначає послідовність етапів від початкового планування до завершення експлуатації системи. Згідно зі стандартом ISO/IEC 12207:1995, життєвий цикл включає основні процеси планування, аналізу вимог, проектування, реалізації, тестування, впровадження та супроводу.

Каскадна модель передбачає послідовне виконання етапів розробки, де кожний наступний етап починається лише після повного завершення попереднього. Ця модель забезпечує чітку структуру проекту та детальну документацію на кожному етапі, що особливо важливо для академічних проектів. Однак каскадна модель має обмежену гнучкість та не дозволяє вносити зміни після завершення етапу без повернення до попередніх стадій.

Ітераційна модель розділяє розробку на декілька коротких циклів, кожний з яких включає всі основні етапи життєвого циклу. Це дозволяє отримувати робочі версії продукту на ранніх стадіях та вносити корективи на основі зворотного зв’язку. Ітераційний підхід особливо ефективний для проектів з нечіткими або змінними вимогами.

Спіральна модель поєднує елементи каскадної та ітераційної моделей з додатковим акцентом на аналізі ризиків. Кожна ітерація включає планування, аналіз ризиків, розробку та оцінку результатів. Ця модель підходить для складних проектів з високим рівнем невизначеності та потенційними ризиками.

2.2.2. Обґрунтування вибору моделі розробки

Для розробки платформи обрано модифіковану каскадну модель з елементами ітераційності. Цей вибір обґрунтований специфікою академічного проекту, чіткими вимогами до функціональності та обмеженими часовими рамками виконання роботи.

Каскадна основа забезпечує структурованість розробки та детальну документацію кожного етапу, що відповідає вимогам навчального процесу. Послідовне виконання етапів дозволяє зосередитися на якісному виконанні кожної фази без розпорошення уваги на паралельні задачі.

Елементи ітераційності включені для можливості уточнення та покращення рішень на основі результатів попередніх етапів. Особливо це стосується етапів проектування інтерфейсу та реалізації, де можливі корективи для покращення користувацького досвіду.

2.2.3. Етапи життєвого циклу розробки

Етап аналізу та планування включає детальне вивчення предметної області, аналіз існуючих рішень та формування технічного завдання. На цьому етапі визначаються функціональні та нефункціональні вимоги до системи, обирається

технологічний стек та планується архітектура майбутньої платформи. Результатом етапу є технічне завдання та план розробки.

Етап проектування системи передбачає створення детальної архітектури вебплатформи, включаючи структуру бази даних, API інтерфейси та алгоритми основних функцій. Розробляються діаграми архітектури, схема бази даних та mockup інтерфейсу користувача. Особлива увага приділяється проектуванню інтерфейсу для забезпечення простоти та інтуїтивності використання.

Етап реалізації включає програмування всіх компонентів системи згідно з розробленою архітектурою. Розробка ведеться послідовно: спочатку створюється база даних та серверна частина, потім клієнтський інтерфейс та інтеграція компонентів. На цьому етапі важливо дотримуватися стандартів кодування та забезпечувати коментування коду для подальшого супроводу.

Етап тестування та налагодження передбачає комплексну перевірку функціональності платформи, виявлення та усунення помилок.

Етап документування та впровадження включає створення документації користувача, технічної документації та рекомендацій щодо розгортання системи. Розробляється керівництво користувача з описом всіх функцій платформи та інструкції щодо встановлення і налаштування системи.

2.2.4. Переваги обраної моделі

Структурованість каскадної моделі забезпечує чітке планування та контроль виконання проекту, що критично важливо в умовах академічної роботи з фіксованими дедлайнами. Послідовне виконання етапів дозволяє зосередитися на якісному виконанні кожної фази без втрати фокусу.

Детальна документація на кожному етапі створює повну картину процесу розробки та полегшує подальший супровід системи. Це особливо важливо для кваліфікаційної роботи, де необхідно продемонструвати розуміння всього процесу створення програмного забезпечення.

Включення елементів ітераційності дозволяє вносити покращення та корективи без порушення загальної структури проекту. Це забезпечує гнучкість розробки при збереженні основних переваг каскадної моделі.

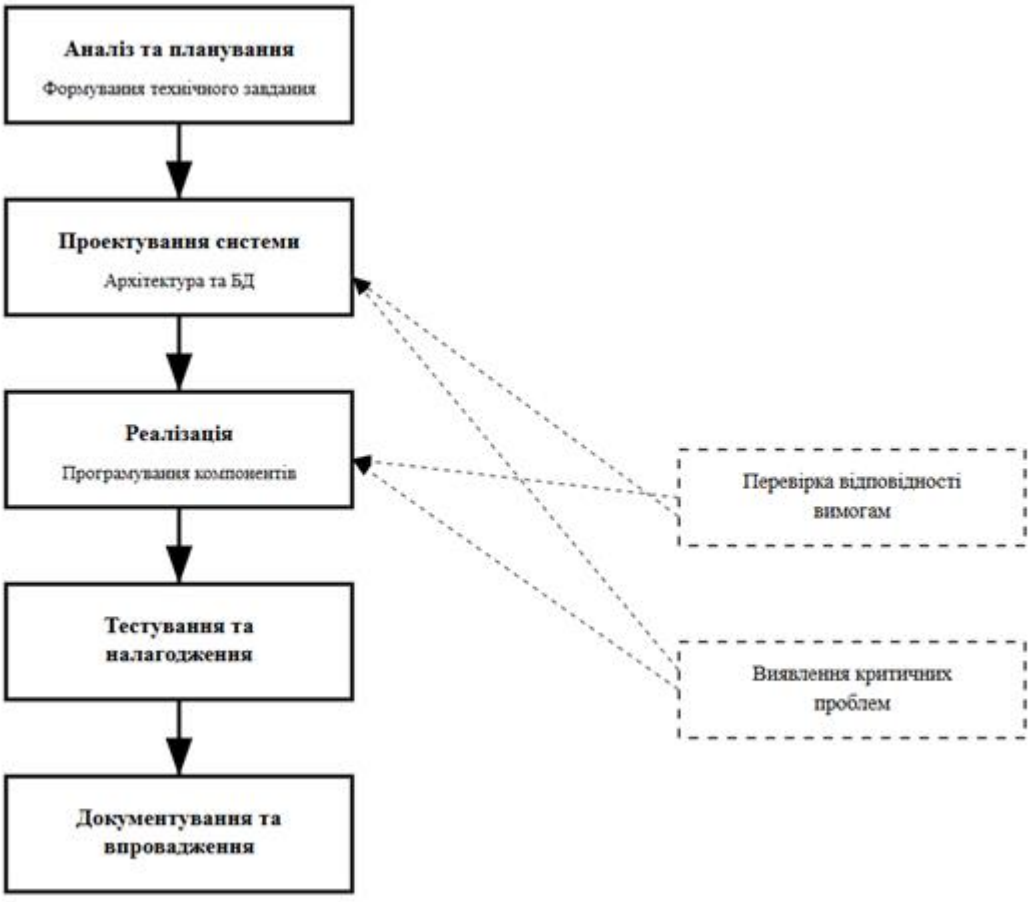


Рисунок 2.1 - Модель життєвого циклу розробки вебплатформи

Обрана модель розробки оптимально підходить для створення вебплатформи в рамках кваліфікаційної роботи, забезпечуючи баланс між структурованістю процесу та гнучкістю реалізації. Чітка послідовність етапів дозволяє ефективно планувати роботу та контролювати прогрес, а можливість внесення коректив забезпечує якість кінцевого продукту.

2.3. Загальний опис проєкту

2.3.1. Архітектура вебплатформи

Архітектура вебплатформи побудована за принципами трирівневої клієнт-серверної моделі, що забезпечує чіткий розподіл відповідальності між компонентами системи та створює основу для масштабованого та підтримуваного рішення. Система складається з рівня представлення, бізнес-логіки та рівня даних, кожен з яких виконує специфічні функції та взаємодіє з іншими через чітко визначені інтерфейси.

Рівень представлення реалізований з використанням HTML5, CSS3 та JavaScript, забезпечуючи інтерактивний користувацький інтерфейс для відтворення музики та навігації по платформі. Цей рівень відповідає за відображення музичного контенту, обробку користувацьких дій та взаємодію з аудіо API браузера для відтворення звукових файлів.

Рівень бізнес-логіки реалізований на PHP та включає обробку HTTP-запитів, управління сесіями користувачів, пошук по базі даних та логіку авторизації. Цей рівень містить основну функціональність системи, включаючи алгоритми пошуку музики, управління лайками та завантаженнями, а також адміністративні функції для управління контентом.

Рівень даних базується на MySQL базі даних, що зберігає інформацію про музичні композиції, користувачів та їх взаємодії з системою. База даних забезпечує надійне зберігання та швидкий доступ до інформації через оптимізовані SQL-запити.

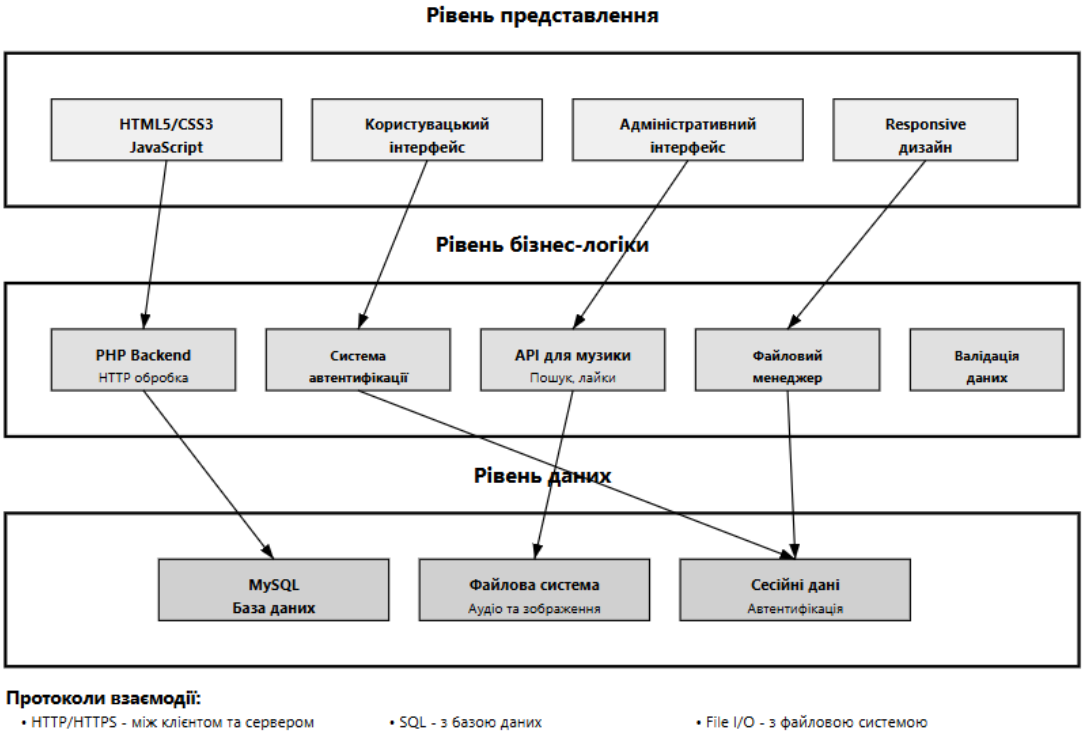


Рисунок 2.2 - Архітектура вебплатформи

Взаємодія між компонентами відбувається через HTTP-протокол для комунікації між клієнтом та сервером, а також через SQL-запити для взаємодії з базою даних. Система підтримує RESTful принципи для організації API, що забезпечує зрозумілість та розширюваність архітектури.

2.3.2. Карта навігації сайту

Навігаційна структура вебсайту розроблена з урахуванням принципів простоти та інтуїтивності, забезпечуючи користувачам легкий доступ до всіх функцій системи. Структура сайту організована таким чином, щоб мінімізувати кількість кліків для досягнення будь-якої цільової функції та створити логічний потік користувацької взаємодії.

Головна сторінка служить центральним хабом платформи, де користувачі можуть переглядати весь доступний музичний контент у вигляді структурованої таблиці. Кожен елемент списку містить основну інформацію про композицію,

включаючи назву, виконавця, дату додавання та елементи управління для відтворення, лайків та завантаження.

Функція пошуку інтегрована безпосередньо в заголовок сторінки, дозволяючи користувачам швидко знаходити потрібні композиції без переходу на окремі сторінки. Система автодоповнення та фільтрації результатів забезпечує ефективний пошук навіть при частковому введенні назви.

Адміністративна частина доступна через спеціальний URL-шлях /admin та включає дві основні сторінки: панель управління музикою та форму додавання нового контенту. Доступ до адміністративних функцій контролюється системою автентифікації, що перевіряє права користувача перед надання доступу.

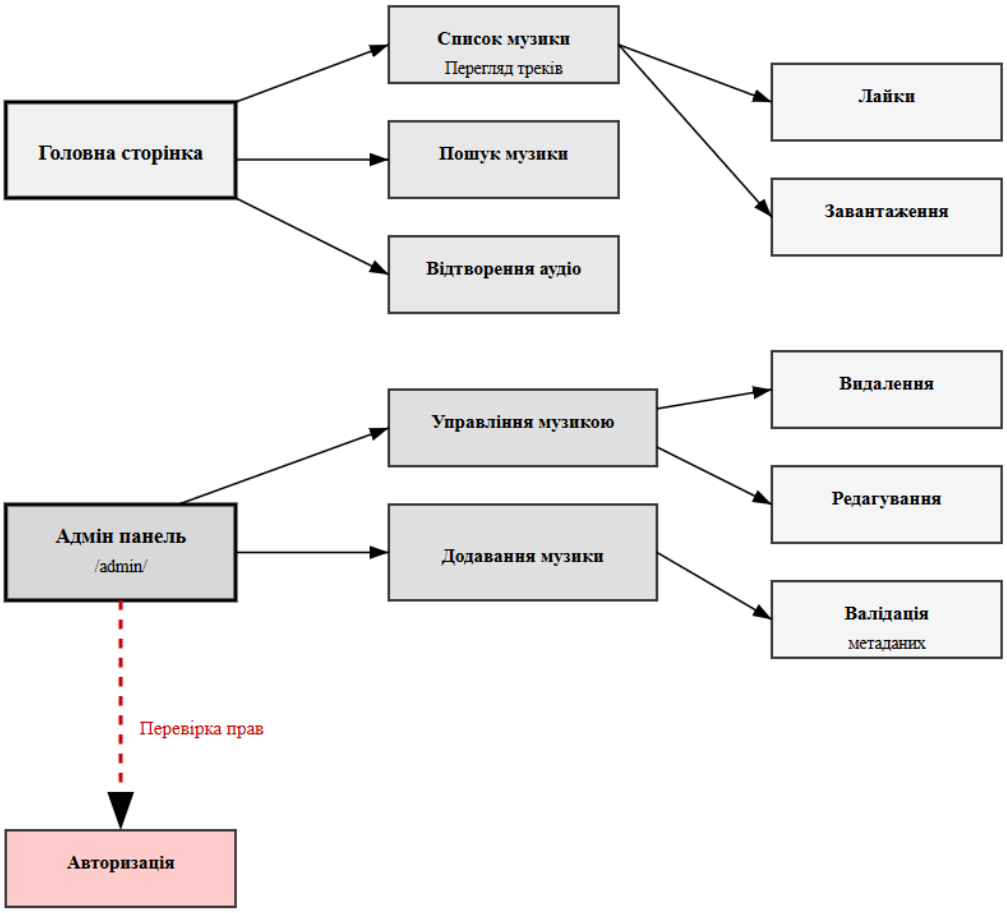


Рисунок 2.3 - Карта навігації вебплатформи

Навігаційна структура враховує різні ролі користувачів та забезпечує відповідний рівень доступу до функціональності. Звичайні користувачі мають доступ лише до перегляду та взаємодії з музичним контентом, тоді як адміністратори отримують додаткові права для управління системою.

2.3.3. Проєктування бази даних

Процес проєктування бази даних для "MusicHub" включав детальний аналіз різних систем управління базами даних та вибір оптимального рішення для специфічних потреб проекту

MySQL є однією з найпопулярніших реляційних СУБД з відкритим кодом, розробленою компанією Oracle Corporation. Система характеризується високою швидкістю виконання запитів, особливо для операцій читання, що критично важливо для музичних платформ з інтенсивним переглядом контенту. MySQL підтримує різні движки зберігання, включаючи InnoDB для транзакційних операцій та MyISAM для швидкого читання даних. Широка підтримка вебхостинг провайдерами робить MySQL доступним вибором для більшості проектів.

Широка підтримка MySQL вебхостинг провайдерами забезпечує гнучкість у виборі серверних рішень та зменшує витрати на розгортання системи. Простота налаштування та адміністрування MySQL робить її ідеальним вибором для проектів з обмеженими ресурсами на системне адміністрування.

Концептуальна модель бази даних для платформи побудована навколо центральної сутності музичної композиції, яка містить всю необхідну інформацію для функціонування платформи. Модель спроектована з урахуванням принципів нормалізації для забезпечення цілісності даних та мінімізації надлишковості.

Основна сутність "Музична композиція" містить атрибути, що описують всі аспекти музичного контенту: ідентифікаційну інформацію (унікальний ідентифікатор, назва, автор), технічні характеристики (файли аудіо та зображень), метрики взаємодії користувачів (лайки, завантаження) та службову інформацію (дата додавання, посилання).

Дизайн бази даних враховує специфіку вебдодатків, де швидкість відгуку критично важлива для користувацького досвіду. Логічна структура оптимізована для найчастіших операцій: відображення списку композицій, пошук за назвою або автором, та оновлення лічильників взаємодії користувачів.

Логічна реалізація бази даних включає єдину таблицю music__list, що містить всю необхідну інформацію про музичні композиції. Така денормалізована структура обрана свідомо для максимізації продуктивності читання за рахунок незначного збільшення обсягу зберігання.

Структура таблиці music__list:

- id (INT AUTO_INCREMENT PRIMARY KEY) - унікальний ідентифікатор композиції, що служить первинним ключем та забезпечує швидкий доступ до записів;
- name (VARCHAR(255) NOT NULL) - повна назва музичної композиції як вона відображається користувачам;
- author (VARCHAR(255) NOT NULL) - ім’я виконавця або автора музичного твору;
- downloads (INT NOT NULL DEFAULT 0) - лічильник загальних завантажень композиції;
- download_user (INT NOT NULL DEFAULT 0) - лічильник завантажень користувачами через інтерфейс;
- likes (INT NOT NULL DEFAULT 0) - кількість лайків від користувачів;
- link (VARCHAR(255) NOT NULL) - посилання на зовнішній ресурс, пов’язаний з композицією;
- date_added (TIMESTAMP DEFAULT CURRENT_TIMESTAMP) - автоматична мітка часу додавання;
- name_music (VARCHAR(255) NOT NULL) - ім’я файлу аудіокомпозиції в файловій системі;
- img (VARCHAR(255) NOT NULL) - ім’я файлу зображення обкладинки.
-

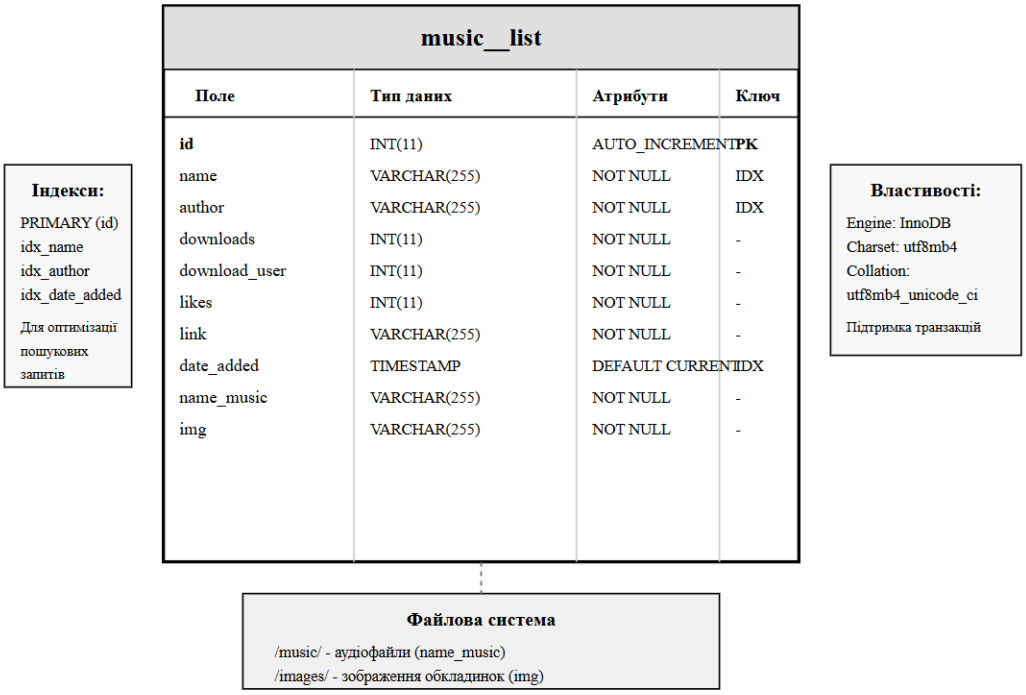


Рисунок 2.4 – Логічна схема бази даних

2.3.4. Проєктування інтерфейсу користувача

Проєктування користувацького інтерфейсу вебплатформи для прослуховування музики базується на принципах мінімалістичного дизайну та максимальної зручності використання. Основна ціль інтерфейсу полягає в створенні простого та інтуїтивного досвіду для прослуховування музики без зайвих елементів, що можуть відволікати користувача від основної функціональності.

Кольорова схема інтерфейсу побудована на темних тонах з яскравими зеленими акцентами, що створює сучасний та стильний вигляд платформи. Темний фон зменшує втому очей під час тривалого використання та створює фокус на музичному контенті. Зелені акценти використовуються для виділення інтерактивних елементів та створення візуальної ієрархії.

Заголовок платформи містить логотип "MusicHub" та поле пошуку, розташовані горизонтально для оптимального використання простору екрану. Поле

пошуку має чіткий placeholder текст та кнопку відправки з іконкою пошуку, що робить його призначення очевидним для користувачів.

Основний контент організований у вигляді таблиці, що забезпечує структурований перегляд музичних композицій. Кожен рядок містить номер композиції, інформацію про трек з мініатюрою обкладинки, дату додавання, тривалість композиції, кількість лайків, кількість завантажень та кнопку завантаження файлу.

Медіаплеєр інтегрований безпосередньо в кожен рядок таблиці, дозволяючи користувачам відтворювати музику без переходу на окремі сторінки. HTML5 audio елемент забезпечує нативні елементи управління відтворенням, що знайомі користувачам з різних платформ.

Адаптивний дизайн забезпечує коректне відображення інтерфейсу на різних пристроях. На мобільних пристроях таблиця трансформується для зручного перегляду на малих екранах, зберігаючи всю необхідну функціональність.

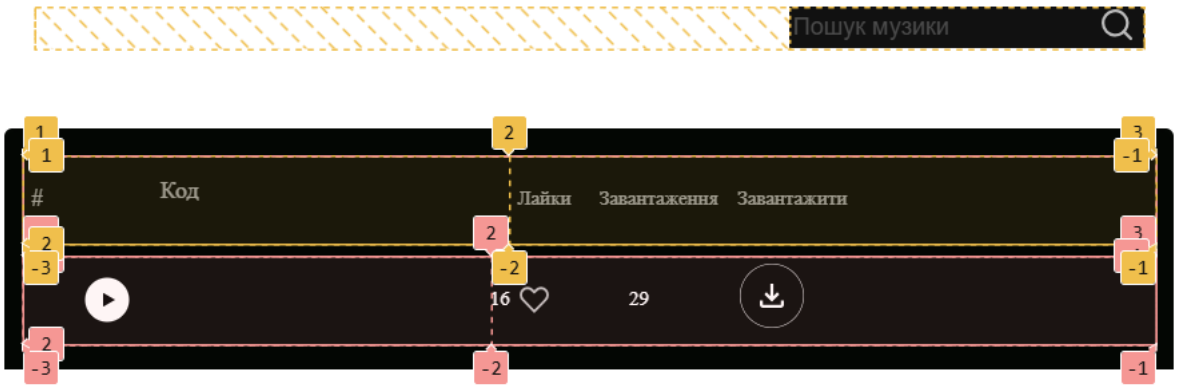


Рисунок 2.5 - Структура користувацького інтерфейсу

Інтерактивні елементи мають чіткі візуальні стани для hover та active дій, забезпечуючи зворотний зв’язок з користувачем. Кнопки та посилання мають достатній розмір для зручного натискання як на настільних комп’ютерах, так і на сенсорних пристроях.

2.3.5. Проєктування адміністративної панелі

Адміністративна панель вебсайту розроблена як окрема підсистема з власним інтерфейсом та функціональністю, призначеною для управління музичним контентом платформи. Доступ до адміністративних функцій здійснюється через спеціальний URL-шлях /admin, що забезпечує логічне відокремлення від користувацької частини сайту.

Архітектура адміністративної панелі базується на популярному шаблоні SB Admin 2, що забезпечує професійний вигляд та повний набір UI компонентів для ефективного управління контентом. Бічна навігаційна панель містить основні розділи системи управління: головну сторінку з переліком всієї музики та сторінку додавання нового контенту.

Система автентифікації адміністраторів перевіряє роль користувача через сесійну змінну role зі значенням ‘admin’. При спробі доступу до адміністративних сторінок без відповідних прав система автоматично перенаправляє користувача на сторінку авторизації.

Головна сторінка адміністративної панелі відображає всі музичні композиції у вигляді таблиці з повною інформацією про кожен трек. Таблиця включає ID композиції, назву, кількість завантажень, кількість лайків, посилання на зовнішній ресурс, дату додавання та дії для редагування або видалення контенту.

Форма додавання нової музики організована як окрема сторінка з набором полів для введення всіх необхідних метаданих. Форма включає поля для назви композиції, посилання на зовнішній ресурс, імені автора, завантаження аудіофайлу та зображення обкладинки. Використання enctype="multipart/form-data" забезпечує коректне завантаження файлів на сервер.

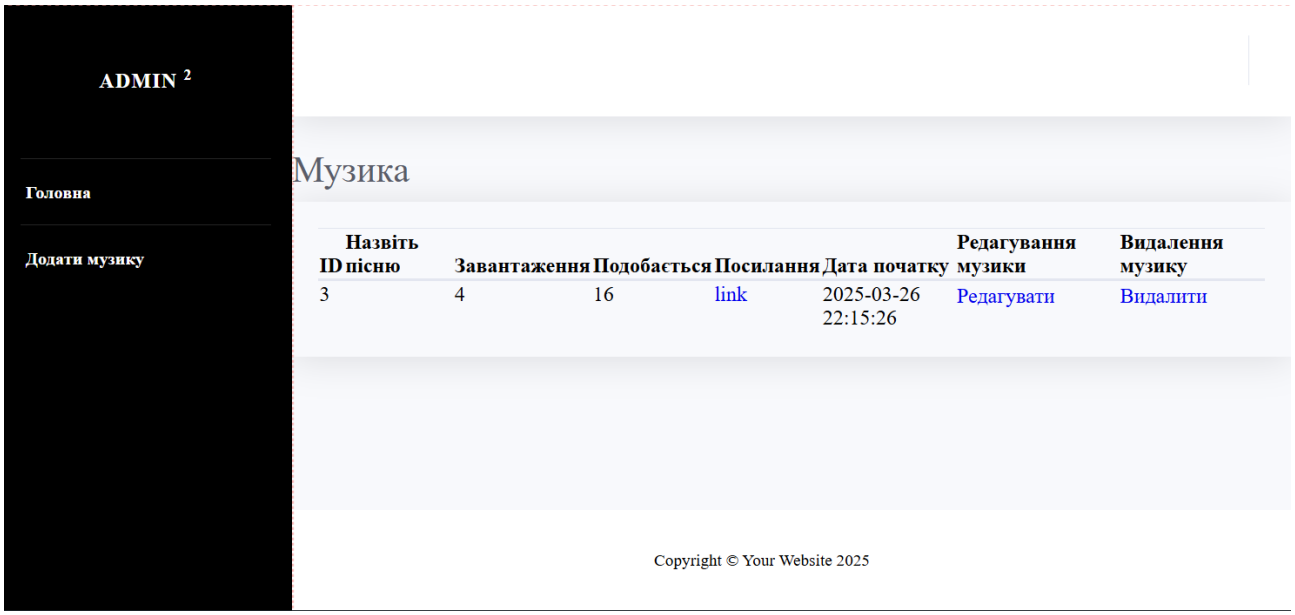


Рисунок 2.6 - Структура адміністративної панелі

Обробка файлів в адміністративній панелі включає валідацію типів файлів, генерацію унікальних імен для запобігання конфліктам та збереження в відповідних директоріях файлової системи. Аудіофайли зберігаються в директорії /music, а зображення обкладинок в директорії /images.

Функціональність редагування дозволяє адміністраторам оновлювати метадані існуючих композицій через окрему форму, що попередньо заповнена поточними значеннями. Функція видалення включає видалення записів з бази даних та відповідних файлів з файлової системи для запобігання накопиченню непотрібних даних.

Інтерфейс адміністративної панелі адаптований для ефективної роботи з великою кількістю контенту, включаючи сортування, пагінацію та пошук по таблиці даних. Використання DataTables бібліотеки забезпечує розширену функціональність для управління табличними даними.

Безпека адміністративної панелі забезпечується через перевірку прав доступу на кожній сторінці, валідацію вхідних даних та використання підготовлених SQL-запитів для запобігання ін’єкційним атакам. Логування адміністративних дій може бути додано для аудиту змін в системі.

2.3.6. Алгоритми обробки та відтворення аудіоконтенту

Ефективна обробка та відтворення аудіоконтенту є центральною технічною задачею вебплатформи, що вимагає комплексного підходу до проектування алгоритмів взаємодії з музичними файлами. Архітектура системи побудована навколо принципу потокової обробки даних, що забезпечує мінімальні затримки та оптимальне використання ресурсів як сервера, так і клієнтського пристрою.

Основний алгоритм потокового відтворення базується на технології HTML5 Audio API та реалізує прогресивне завантаження аудіоконтенту. Процес починається з формування запиту до сервера на основі унікального ідентифікатора композиції з бази даних, після чого система генерує URL шлях до файлу у форматі /music/{filename}. Ключовою особливістю алгоритму є використання атрибуту preload="metadata", що дозволяє розпочати відтворення до повного завантаження файлу через попереднє завантаження лише метаданих композиції.

Для забезпечення плавного відтворення без переривань система використовує математичну модель буферизації, де розмір буферу розраховується за наступною формулою:

$$B = (R \times T)/8 \qquad (2.1)$$

де B – розмір буферу в байтах,

R – бітрейт аудіофайлу в бітах за секунду,

T – час буферизації в секундах, що зазвичай становить 3-5 секунд. Наприклад, для стандартного MP3 файлу з бітрейтом 320 kbps оптимальний розмір буферу складає 156.25 КБ, що забезпечує безперервне відтворення навіть при короткочасних затримках мережі.

Тісно пов’язаний з відтворенням алгоритм пошуку музичного контенту забезпечує швидкий доступ користувачів до потрібних композицій через оптимізовані SQL запити з використанням оператора LIKE та індексованих полів. Система реалізує інтелектуальне ранжування результатів пошуку, де коефіцієнт релевантності розраховується за наступною формулою:

$$R = w_1 \times P + w_2 \times L + w_3 \times S \qquad (2.2)$$

де R – загальний коефіцієнт релевантності,

P – позиційний коефіцієнт (1.0 для збігу на початку, 0.5 для збігу в середині),

L – коефіцієнт довжини збігу (відношення довжини збігу до загальної довжини поля),

S – коефіцієнт популярності (логарифм кількості лайків + 1),

w_1, w_2, w_3 – вагові коефіцієнти (0.5, 0.3, 0.2 відповідно).

Такий підхід забезпечує не лише швидкий пошук, але й представлення найбільш релевантних результатів у верхній частині списку.

Ефективність пошуку безпосередньо впливає на швидкість доступу до аудіоконтенту, що в свою чергу визначає необхідність оптимального алгоритму обробки та збереження файлів на етапі їх завантаження в систему. Процес включає багаторівневу валідацію через перевірку MIME-типів, контроль розміру файлів та генерацію унікальних імен через комбінацію timestamp та випадкових значень. Для зображень обкладинок застосовується алгоритм масштабування з збереженням пропорцій згідно з наступною формулою:

$$S_{new} = min(W_{max}/W_{orig}, H_{max}/H_{orig}) \times S_{orig} \qquad (2.3)$$

де S_new – новий розмір зображення,

W_max, H_max – максимальні розміри (зазвичай 400×400 пікселів),

W_orig, H_orig – оригінальні розміри зображення,

S_orig – оригінальний розмір.

Це оптимізує баланс між якістю відображення та швидкістю завантаження вебсторінки.

Успішне збереження та індексація аудіофайлів створює основу для реалізації алгоритму керування відтворенням, що координує роботу множини аудіо елементів на вебсторінці. Система використовує глобальну змінну currentAudio для відстеження активного відтворення та автоматично зупиняє попередню композицію при запуску нової, запобігаючи одночасному відтворенню множини треків. Такий підхід не лише покращує користувацький досвід, але й оптимізує використання системних ресурсів браузера.

Завершальним компонентом архітектури є алгоритм відстеження метрик взаємодії користувачів, що збирає дані про лайки, завантаження та тривалість прослуховування через асинхронні AJAX запити.

Використання атомарних SQL операцій типу:

```
UPDATE music__list SET likes = likes + 1 WHERE id = ?
```

гарантує цілісність лічильників навіть при одночасному доступі множини користувачів. Зібрані метрики не лише відображають популярність контенту, але й служать вхідними даними для алгоритмів ранжування пошуку та пріоритизації кешування, створюючи замкнений цикл оптимізації системи на основі реальної поведінки користувачів.

Таким чином, всі описані алгоритми функціонують як єдина екосистема, де кожен компонент використовує результати роботи інших для забезпечення оптимальної продуктивності та користувацького досвіду вебплатформи .

2.4. Обґрунтування вибору інструментальних засобів розробки

Процес вибору інструментальних засобів для розробки вебплатформи для прослуховування музики базувався на комплексному аналізі сучасних технологій веброзробки з урахуванням специфічних вимог проекту, обмежень ресурсів та необхідності забезпечення високої продуктивності системи. Розглядалися різні варіанти технологічних стеків, кожен з яких має свої переваги та обмеження для реалізації музичної платформи.

Frontend технології. Для клієнтської частини розглядалися три основні підходи: використання чистих вебтехнологій (HTML5, CSS3, JavaScript), застосування фреймворків (React, Vue.js, Angular) та гібридні рішення. Чисті вебтехнології забезпечують максимальну продуктивність та мінімальні вимоги до ресурсів, що критично важливо для музичних додатків. Фреймворки пропонують більші можливості для складних інтерактивних інтерфейсів, але збільшують розмір завантажуваного коду та вимоги до браузера користувача.

HTML5 служить структурною основою вебплатформи, визначаючи семантичну розмітку сторінок та забезпечуючи нативну підтримку мультимедійного контенту через елемент <audio>. Це дозволяє створювати доступний та SEO-оптимізований контент без необхідності використання сторонніх плагінів для відтворення музики.

CSS3 відповідає за візуальне оформлення та стилізацію інтерфейсу, включаючи створення адаптивного дизайну через медіа-запити, анімації переходів та градієнтні ефекти. Використання CSS Grid та Flexbox забезпечує гнучкість макету та коректне відображення на різних пристроях.

JavaScript реалізує інтерактивну логіку платформи, управління DOM елементами, обробку подій користувача та асинхронні запити до сервера. Мова забезпечує динамічну поведінку інтерфейсу без перезавантаження сторінок, що критично важливо для безперервного прослуховування музики.

Backend рішення. Серед серверних технологій аналізувалися PHP, Node.js, Python (Django/Flask) та Java (Spring). PHP характеризується простотою розгортання, широкою підтримкою хостинг-провайдерів та низьким порогом входу для розробки. Node.js забезпечує високу продуктивність для додатків реального часу та єдину мову для frontend та backend. Python пропонує елегантний синтаксис та потужні бібліотеки, але вимагає більше ресурсів сервера.

Системи управління базами даних**.** Вибір MySQL як основної СУБД був детально обґрунтований у розділі проєктування бази даних з урахуванням специфічних вимог музичної платформи.

HTML5, CSS3, JavaScript обрані як основа frontend розробки завдяки нативній підтримці аудіо API, що є критично важливим для музичної платформи. HTML5 Audio API забезпечує повний контроль над відтворенням аудіо без необхідності використання сторонніх плагінів. CSS3 надає достатні можливості для створення сучасного адаптивного дизайну з анімаціями та переходами. JavaScript забезпечує всю необхідну інтерактивність без додаткових залежностей.

HTML5 у проекті "MusicHub" використовується для створення семантично правильної структури сторінок з акцентом на доступність та SEO-оптимізацію. Елемент <audio> з атрибутом preload="metadata" дозволяє швидко

завантажувати інформацію про тривалість композицій без повного завантаження аудіофайлів, що значно покращує швидкодію платформи.

CSS3 реалізує візуальний дизайн платформи з використанням сучасних можливостей стилізації. Застосування CSS Grid для структури таблиці музики забезпечує гнучкий макет, а CSS-анімації створюють плавні переходи при наведенні на елементи інтерфейсу. Медіа-запити забезпечують адаптивність дизайну для коректного відображення на мобільних пристроях.

JavaScript виконує роль головного координатора інтерактивності, управляючи відтворенням аудіо, обробкою форм пошуку та динамічним оновленням інтерфейсу. Використання Vanilla JavaScript без додаткових бібліотек забезпечує мінімальний розмір завантажуваного коду та максимальну швидкодію.

PHP вибрана як серверна мова програмування через її оптимальне співвідношення простоти розробки та функціональності. PHP має відмінну підтримку роботи з файлами, що важливо для обробки аудіо та зображень. Широка підтримка хостинг-провайдерів та наявність готових рішень для типових задач веброзробки значно прискорюють процес створення платформи.

MySQL обрана як СУБД завдяки характеристикам, детально описаним у розділі проєктування бази даних, що включають високу продуктивність операцій читання та простоту адміністрування.

Visual Studio Code використовується як основне інтегроване середовище розробки завдяки його універсальності, широкій підтримці розширень та вбудованим інструментам для роботи з вебтехнологіями. Редактор забезпечує синтаксичну підсвітку, автодоповнення та інтеграцію з системами контролю версій.

MAMP (macOS, Apache, MySQL, PHP) застосовується для створення локального середовища розробки, що точно відтворює умови продакшн сервера. Це забезпечує ідентичність поведінки додатку в процесі розробки та після розгортання на хостингу.

Для ефективного налагодження використовувався комплекс інструментів, що забезпечили всебічний аналіз роботи системи. Основним інструментом для налагодження клієнтської частини служили Browser Developer Tools, вбудовані в сучасні браузери.

Browser Developer Tools використовувалися для аналізу HTML-структури, CSS-стилів, JavaScript-коду та мережевого трафіку. Вкладка Elements дозволяла в реальному часі аналізувати DOM-структуру та динамічно редагувати стили. Console використовувалася для відстеження JavaScript помилок, виведення діагностичної інформації та тестування фрагментів коду. Network панель забезпечувала моніторинг HTTP-запитів, включаючи завантаження аудіофайлів та AJAX-запити для лайків.

phpMyAdmin застосовувався для прямого аналізу SQL-запитів, перевірки цілісності даних у базі та моніторингу продуктивності запитів. Особливо корисним було тестування prepared statements та аналіз планів виконання запитів для оптимізації індексів.

2.5. Особливості програмної реалізації та основні режими роботи

Програмна реалізація вебсайту організована за модульним принципом, що забезпечує логічне розділення функціональності та спрощує подальший супровід системи. Архітектура включає три основні функціональні модулі: пошуку та відтворення музики, управління контентом через адміністративну панель та систему автентифікації користувачів.

2.5.1. Модуль пошуку та відтворення музики

Центральним компонентом користувацького інтерфейсу є модуль пошуку та відтворення музичного контенту, реалізований в файлі index.php. Модуль забезпечує відображення каталогу музики, функцію пошуку та інтегровані елементи управління відтворенням.

Алгоритм завантаження та відображення музичного каталогу:

```php
// Ініціалізація сесії для збереження користувацьких дій
session_start();
require_once './admin/function/db.php';

// Базовий запит для завантаження всього каталогу музики
$music__list = mysqli_query($connect, "SELECT * FROM `music__list`");
$music__list = mysqli_fetch_all($music__list);

// Обробка пошукового запиту через GET параметр
if (isset($_GET['searchInput'])) {
    $searchInput = trim($_GET['searchInput']); // Очищення введених даних

    // Підготовлений запит для запобігання SQL-ін'єкціям
    $stmt = mysqli_prepare($connect, "SELECT * FROM `music__list` WHERE `name` LIKE ?");
    $searchParam = "%" . $searchInput . "%"; // Формування шаблону пошуку
    mysqli_stmt_bind_param($stmt, "s", $searchParam);
    mysqli_stmt_execute($stmt);

    $result = mysqli_stmt_get_result($stmt);
    $music__list = mysqli_fetch_all($result);
    mysqli_stmt_close($stmt);
}

// Завантаження інформації про лайки з сесії користувача
$likedMusic = isset($_SESSION['liked_music']) ? $_SESSION['liked_music'] : [];
?>
```

Система пошуку реалізує безпечний підхід через використання підготовлених запитів (prepared statements), що виключає можливість SQL-ін'єкцій. Пошук здійснюється за частковим збігом в назві композиції з автоматичним додаванням wildcards для знаходження результатів у будь-якій частині назви.

Генерація HTML структури для відображення музичного контенту:

```php
<div class="music__table-row">
    <div class="music__table-column">
        <!-- Нумерація композицій та індикатор відтворення -->
        <div class="music__table-col number-list">
            <p><?= $counter++ ?></p>
            <img src="./assets/img/playing.svg" alt="">
        </div>

        <!-- Інтегрований аудіо плеєр з HTML5 Audio API -->
        <div class="music__table-col">
            <div class="name__music">
                <!-- Нативний HTML5 аудіо елемент з метаданими -->
                <audio src="./music/<?= $music__item[6] ?>" preload="metadata" controls></audio>
                <div class="name__music-img">
                    <img class="play" src="./assets/img/play.svg" alt="play">
                </div>
                <div class="name__music-txt">
                    <h3><?= $music__item[1] ?></h3>
                </div>
```

Кожна композиція відображається через HTML5 audio елемент з атрибутом preload="metadata", що забезпечує швидке завантаження основної інформації про трек без повного завантаження аудіофайлу. Це оптимізує початкове завантаження сторінки при збереженні можливості миттєвого початку відтворення.

Система лайків та завантажень:

```
<!-- Відображення лайків з динамічною зміною іконки -->
<?php $isLiked = isset($likedMusic[$music__item[0]]) && $likedMusic[$music__item[0]]; ?>
<a href="./admin/function/music/like.php?id=<?= $music__item[0] ?>" class="music__table-col likes">
    <p><?= $music__item[4] ?></p>
    <?php
    // Умовне відображення іконки лайку
    if ($isLiked) {
        echo '<img src="./assets/img/Heart_active.svg" alt="Liked">';
    } else {
        echo '<img src="./assets/img/Heart.svg" alt="Like">';
    }
    ?>
</a>

<!-- Лічильник завантажень та кнопка завантаження -->
<div class="music__table-col">
    <p><?= $music__item[2] + $music__item[9] ?></p>
</div>
<div class="music__table-col">
    <a target="_blank" href="./admin/function/music/download.php?id=<?= $music__item[0] ?>">
        <img src="./assets/img/download-bttn.svg" alt="download">
    </a>
</div>
```

Система відстеження взаємодії користувачів реалізована через сесійні дані для лайків та окремі лічильники для завантажень. Це дозволяє запобігти повторним лайкам від одного користувача в межах сесії при збереженні можливості множинних завантажень.

2.5.2. Модуль управління контентом через адміністративну панель

Адміністративна панель реалізує повний цикл CRUD операцій (Create, Read, Update, Delete) для управління музичним контентом платформи. Система забезпечує безпечний доступ через перевірку ролей та валідацію всіх вхідних даних.

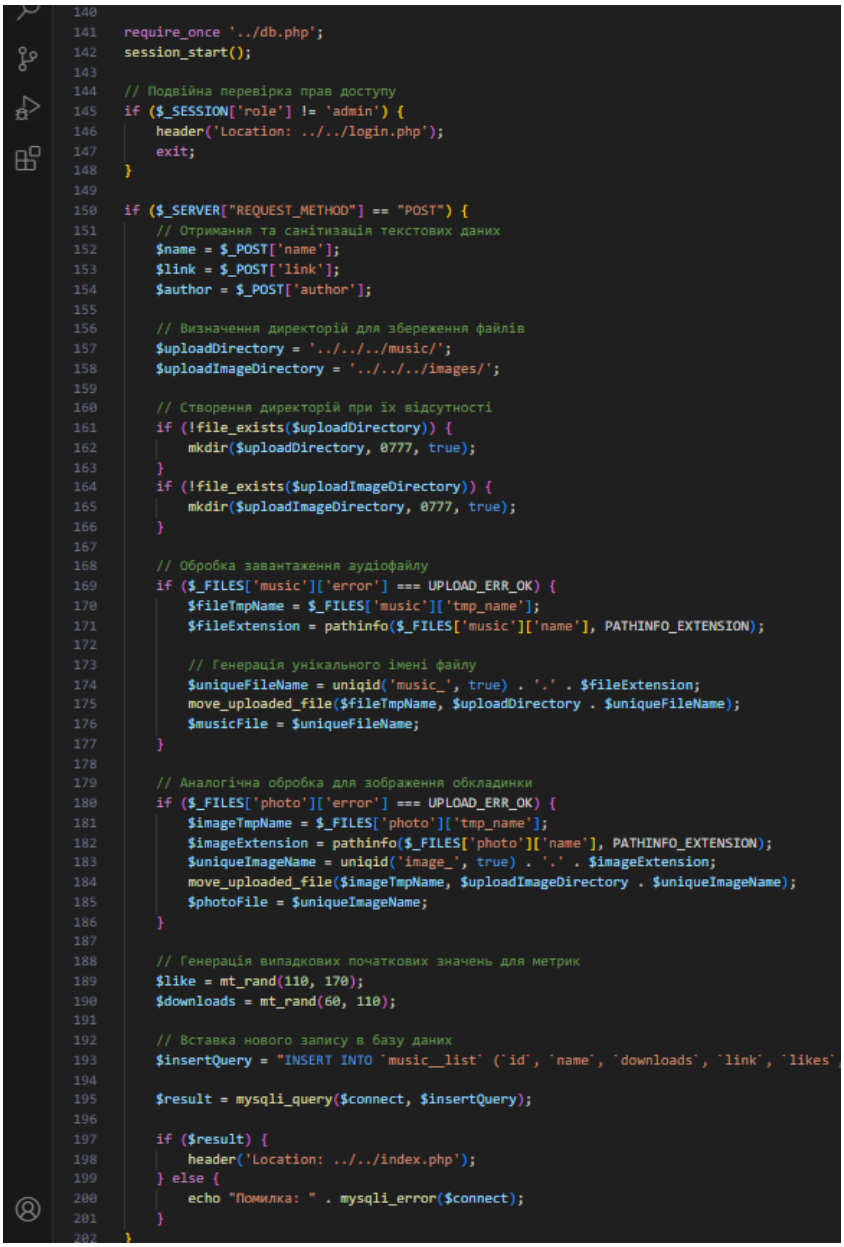
Система автентифікації та контролю доступу:

```php
session_start();

// Перевірка прав адміністратора для доступу до панелі
if ($_SESSION[`role`] != `admin`) {
    header(`Location: ./login.php`);
    exit;
}
```

Кожна сторінка адміністративної панелі включає обов'язкову перевірку ролі користувача через сесійну змінну. При відсутності відповідних прав система автоматично перенаправляє на сторінку авторизації.

Алгоритм додавання нового музичного контенту:



```php
require_once '../db.php';
session_start();

// Подвійна перевірка прав доступу
if ($_SESSION['role'] != 'admin') {
    header('Location: ../../login.php');
    exit;
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Отримання та санітизація текстових даних
    $name = $_POST['name'];
    $link = $_POST['link'];
    $author = $_POST['author'];

    // Визначення директорій для збереження файлів
    $uploadDirectory = '../../../music/';
    $uploadImageDirectory = '../../../images/';

    // Створення директорій при їх відсутності
    if (!file_exists($uploadDirectory)) {
        mkdir($uploadDirectory, 0777, true);
    }
    if (!file_exists($uploadImageDirectory)) {
        mkdir($uploadImageDirectory, 0777, true);
    }

    // Обробка завантаження аудіофайлу
    if ($_FILES['music']['error'] === UPLOAD_ERR_OK) {
        $fileTmpName = $_FILES['music']['tmp_name'];
        $fileExtension = pathinfo($_FILES['music']['name'], PATHINFO_EXTENSION);

        // Генерація унікального імені файлу
        $uniqueFileName = uniqid('music_', true) . '.' . $fileExtension;
        move_uploaded_file($fileTmpName, $uploadDirectory . $uniqueFileName);
        $musicFile = $uniqueFileName;
    }

    // Аналогічна обробка для зображення обкладинки
    if ($_FILES['photo']['error'] === UPLOAD_ERR_OK) {
        $imageTmpName = $_FILES['photo']['tmp_name'];
        $imageExtension = pathinfo($_FILES['photo']['name'], PATHINFO_EXTENSION);
        $uniqueImageName = uniqid('image_', true) . '.' . $imageExtension;
        move_uploaded_file($imageTmpName, $uploadImageDirectory . $uniqueImageName);
        $photoFile = $uniqueImageName;
    }

    // Генерація випадкових початкових значень для метрик
    $like = mt_rand(110, 170);
    $downloads = mt_rand(60, 110);

    // Вставка нового запису в базу даних
    $insertQuery = "INSERT INTO `music__list` (`id`, `name`, `downloads`, `link`, `likes`

    $result = mysqli_query($connect, $insertQuery);

    if ($result) {
        header('Location: ../../index.php');
    } else {
        echo "Помилка: " . mysqli_error($connect);
    }
}
```

Алгоритм забезпечує комплексну обробку завантажуваних файлів з автоматичною генерацією унікальних імен через функцію uniqid(), що практично

виключає колізії імен файлів. Система створює необхідні директорії автоматично та перевіряє успішність завантаження через аналіз коду помилки.

Функціонал редагування існуючого контенту:

```
// Отримання ідентифікатора композиції для редагування
$id = $_GET['id'];

// Завантаження поточних даних композиції
$result = mysqli_query($connect, "SELECT * FROM `music__list` WHERE `id` = $id");
$music__item = mysqli_fetch_assoc($result);
?>

<!-- Форма редагування з попередньо заповненими полями -->
<form action="./function/music/edit.php" enctype="multipart/form-data" method="post" id="add-music">
    <!-- Прихований ідентифікатор для збереження зв'язку з записом -->
    <input type="hidden" name="id" value="<?= $music__item['id']; ?>">

    <!-- Поля форми з поточними значеннями -->
    <div class="form__list">
        <p>Назва пісні</p>
        <input type="text" name="name" value="<?= $music__item['name']; ?>" required>
    </div>

    <!-- Попередній перегляд поточних медіафайлів -->
    <div class="form__list">
        <p>Музичний файл</p>
        <input type="file" name="music" accept="audio/*">
        <!-- Відтворення поточного аудіофайлу -->
        <audio controls src="../music/<?= $music__item['name_music']; ?>"></audio>
    </div>

    <div class="form__list">
        <p>Картинка до музичного файла</p>
        <input type="file" name="photo" accept="image/*">
        <!-- Відображення поточного зображення -->
        <img src="../images/<?= $music__item['img']; ?>" alt="">
    </div>
```

Форма редагування забезпечує візуальний попередній перегляд поточних медіафайлів, дозволяючи адміністратору переконатися в правильності вибраного контенту перед внесенням змін.

2.5.3. Система безпеки та видалення контенту

Безпечне видалення записів з валідацією:

```php
<?php
require_once '../db.php';
session_start();

// Перевірка прав доступу
if ($_SESSION['role'] != 'admin') {
    header('Location: ../../login.php');
    exit;
}

// Валідація параметрів запиту
if (isset($_GET['id']) && is_numeric($_GET['id'])) {
    $id = $_GET['id'];

    // Використання підготовленого запиту для безпечного видалення
    $stmt = mysqli_prepare($connect, "DELETE FROM `music__list` WHERE `music__list`.`id` = ?");

    if ($stmt) {
        mysqli_stmt_bind_param($stmt, 'i', $id); // Прив'язка цілочисельного параметра
        mysqli_stmt_execute($stmt);
        mysqli_stmt_close($stmt);

        header('Location: ../../index.php');
    } else {
        echo "Помилка підготовки запиту";
    }
} else {
    echo "Некоректний ідентифікатор";
}
```

Система видалення реалізує багаторівневу перевірку безпеки: валідацію прав доступу, перевірку існування та коректності параметрів, використання підготовлених запитів для запобігання SQL-ін'єкціям.

Модуль автентифікації та управління сесіями:

```php
<!-- Форма входу в адміністративну панель -->
<form class="user" action="./function/auth/login.php" method="post">
    <div class="form-group">
        <input type="email" class="form-control form-control-user"
               name="email" placeholder="Введіть адресу електронної пошти..." required>
    </div>
    <div class="form-group">
        <input type="password" class="form-control form-control-user"
               name="password" placeholder="Пароль" required>
    </div>
    <button class="btn btn-primary btn-user btn-block">Логін</button>
</form>
<?php
// Модуль завершення сесії (exit.php)
session_start();
session_destroy(); // Повне знищення сесійних даних

header('Location: ./login.php'); // Перенаправлення на сторінку входу
exit;
```

Система автентифікації забезпечує безпечне управління сесіями з повним знищенням сесійних даних при виході та перенаправленням на сторінку авторизації.

Реалізована архітектура забезпечує повний цикл управління музичним контентом з дотриманням принципів безпеки, валідації даних та зручності використання як для кінцевих користувачів, так і для адміністраторів платформи.

2.6. Організація тестування та налагодження програмного засобу

Процес тестування вебплатформи "MusicHub" здійснювався поетапно з використанням комплексного підходу, що включав функціональне тестування, тестування безпеки, тестування продуктивності та тестування сумісності. Організація тестування передбачала як ручне тестування інтерфейсу користувача, так і автоматизовану перевірку серверної логіки через спеціалізовані інструменти.

Тестування проводилося в локальному середовищі MAMP з подальшою перевіркою на тестовому сервері. Використовувалися Browser Developer Tools для аналізу клієнтської частини, phpMyAdmin для верифікації операцій з базою даних та вбудовані засоби PHP для налагодження серверного коду. Кожен модуль тестувався окремо, після чого проводилася інтеграційна перевірка взаємодії між компонентами системи.

Особлива увага приділялася тестуванню функцій, специфічних для музичної платформи: відтворення аудіо, пошуку композицій, управління лайками та завантаженнями. Тестування HTML5 Audio API проводилося на різних аудіоформатах для забезпечення кроссбраузерної сумісності.

Деталі тестування розміщені в додатку Г.

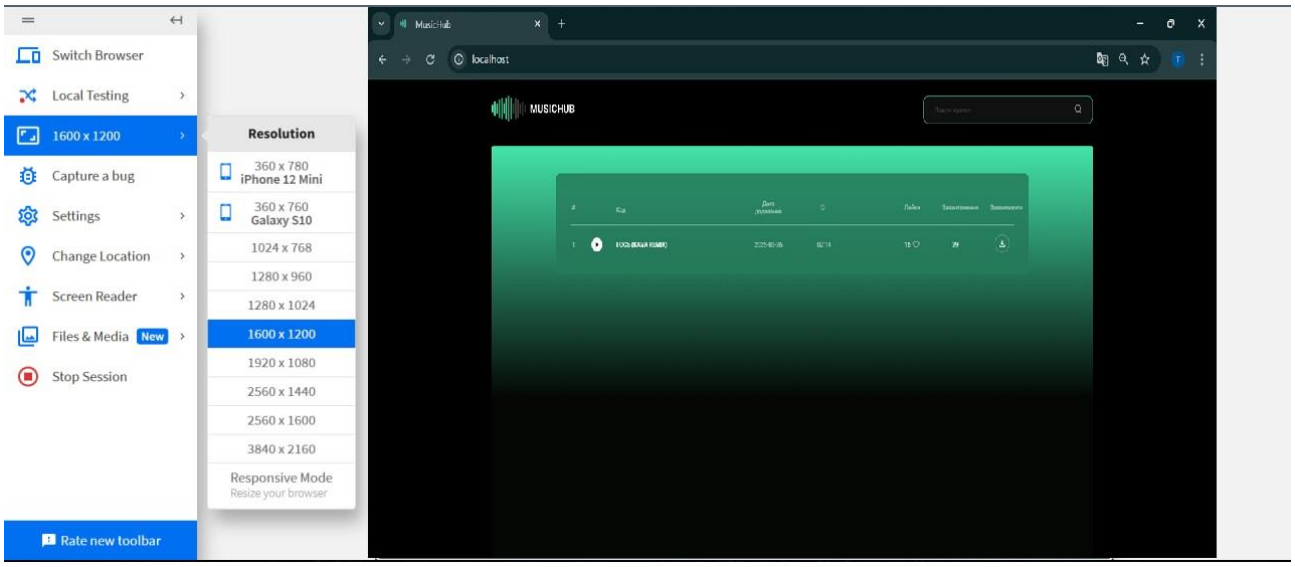


Рисунок 2.7 – Тестування платформи в Browser Developer Tools

Рекомендації щодо покращення функціональності:

1. Додавання підтримки додаткових аудіоформатів (FLAC, AAC)
2. Реалізація системи плейлистів для користувачів
3. Впровадження системи рейтингів та коментарів до композицій
4. Додавання функції перемішування та повтору треків

Рекомендації щодо оптимізації продуктивності:

1. Впровадження кешування запитів до бази даних
2. Оптимізація розміру зображень через автоматичне стиснення
3. Реалізація ледачого завантаження для великих списків композицій
4. Додавання CDN для статичних ресурсів

Тестування аудіофункцій проводилося з використанням різних форматів файлів та розмірів для перевірки коректності роботи HTML5 Audio API. Використовувалися тестові файли різної якості (128kbps, 320kbps) та тривалості для перевірки продуктивності системи.

Результати комплексного тестування підтверджують загальну стабільність та функціональність платформи "MusicHub". Основні модуля системи працюють коректно та відповідають встановленим вимогам. Виявлені недоліки стосуються переважно безпеки системи та можуть бути усунені через впровадження додаткових заходів захисту.

Продуктивність системи відповідає встановленим критеріям, а кроссбраузерна сумісність забезпечує широку доступність платформи для користувачів. Адаптивність інтерфейсу гарантує коректне відображення на різних типах пристроїв та розмірах екранів.

Платформа готова до використання в продакшн-середовищі після впровадження рекомендованих заходів безпеки та може служити надійною основою для музичного сервісу з можливістю подальшого розширення функціональності.

2.7. Рекомендації по використанню та впровадженню програмного засобу

2.7.1. Обґрунтування практичного використання та сфери застосування

Вебсайт представляє собою комплексне технічне рішення, яке демонструє високий потенціал для практичного використання в різноманітних сферах музичної індустрії та цифрових медіа. Основним обґрунтуванням доцільності впровадження системи є її архітектурна простота в поєднанні з функціональною повнотою, що дозволяє швидко розгортати надійні музичні сервіси без значних капіталовкладень.

Технічна архітектура платформи базується на перевірених часом технологіях веброзробки, що забезпечує стабільність роботи та доступність кваліфікованих фахівців для подальшого супроводу. Використання стандартних рішень (PHP, MySQL, HTML5 Audio API) мінімізує ризики технологічної застарілості та забезпечує сумісність з більшістю сучасних хостинг-платформ.

Економічна ефективність впровадження обумовлена низькими вимогами до серверної інфраструктури та відсутністю необхідності у спеціалізованому обладнанні для потокового відтворення аудіо. Модульна структура системи дозволяє поетапне впровадження з можливістю масштабування відповідно до зростання аудиторії та функціональних потреб.

2.7.2. Технічні вимоги та системна архітектура для впровадження

Успішне впровадження "MusicHub" значною мірою залежить від правильного планування технічної інфраструктури та дотримання рекомендованих системних вимог. Архітектура системи спроектована з урахуванням можливості поступового масштабування від невеликих локальних інсталяцій до розподілених систем з високим навантаженням.

Мінімальна конфігурація сервера включає одноядерний процесор з тактовою частотою не менше 1.8 GHz, 1 GB оперативної пам’яті та 10 GB вільного дискового простору для базової інсталяції системи. Така конфігурація підходить для тестування та невеликих проектів з аудиторією до 20 одночасних користувачів. Операційна система може бути як Linux-базованою (Ubuntu 18.04+, CentOS 7+), так і Windows Server 2016+, з обов’язковою підтримкою вебсервера Apache 2.4+ або Nginx 1.14+.

Рекомендована продуктивна конфігурація передбачає використання двоядерного процесора з частотою 2.4 GHz або вище, 4 GB оперативної пам’яті та SSD накопичувач обсягом від 50 GB для системи плюс розрахунковий простір для аудіофайлів. Така конфігурація забезпечує комфортну роботу для 50-100 одночасних користувачів з швидким відгуком системи та стабільним потоковим відтворенням.

Критично важливою є конфігурація PHP середовища з версією не нижче 7.4 та обов’язковими розширеннями mysqli для роботи з базою даних, gd для обробки зображень, fileinfo для визначення типів файлів та session для управління користувацькими сесіями. Параметри upload_max_filesize та post_max_size повинні бути встановлені на рівні 50MB для забезпечення завантаження високоякісних аудіофайлів.

2.7.3. Можливості розширення та масштабування системи

Архітектура платформи спроектована з урахуванням можливості поступового зростання як за функціональністю, так і за навантаженням. Модульна структура коду дозволяє додавати нові можливості без порушення існуючого функціоналу, а технологічний стек підтримує різні сценарії масштабування відповідно до потреб проекту.

При зростанні аудиторії понад 100 одночасних користувачів рекомендується перехід до розподіленої архітектури з винесенням бази даних на окремий сервер та використанням CDN (Content Delivery Network) для медіафайлів. База даних може бути розміщена на спеціалізованому сервері з SSD накопичувачами та збільшеною оперативною пам’яттю для кешування запитів.

Використання CDN для аудіофайлів не лише покращує швидкість завантаження для користувачів з різних географічних регіонів, але й значно зменшує навантаження на основний сервер. Сервіси типу CloudFlare, AWS CloudFront або Google Cloud CDN забезпечують глобальну доступність контенту з мінімальними затримками.

Вебплатформа "MusicHub" представляє собою збалансоване рішення, що поєднує простоту впровадження з потенціалом для значного розширення функціональності та масштабування інфраструктури відповідно до зростаючих потреб проекту та його аудиторії.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було успішно створено функціональну вебплатформу для прослуховування музики, яка демонструє ефективний підхід до розробки простих та доступних музичних сервісів. Проведений на початковому етапі аналіз сучасних тенденцій розвитку музичних вебплатформ виявив критичні недоліки існуючих рішень, які стали відправною точкою для формування концепції власного продукту. Виявлена присутність реклами у безкоштовних версіях провідних сервісів, складність їх інтерфейсів та високі бар'єри входу для користувачів підтвердили актуальність розробки альтернативного рішення, орієнтованого на простоту та доступність.

Саме ці висновки з аналітичного дослідження безпосередньо вплинули на архітектурні рішення при проектуванні системи. Розроблена трирівнева клієнт-серверна модель з чітким розподілом відповідальності між компонентами стала логічним результатом потреби в простій, але масштабованій архітектурі. Вибір технологічного стеку (HTML5, CSS3, JavaScript, PHP, MySQL) був зумовлений не лише технічними перевагами, але й необхідністю забезпечити низький поріг входу для розробників та мінімальні вимоги до серверної інфраструктури, що прямо відповідає виявленим в аналізі проблемам складності існуючих рішень.

Реалізація основного функціоналу підтвердила правильність вибраних архітектурних підходів. Потокове відтворення аудіо через HTML5 Audio API, система пошуку з підтримкою часткових збігів та механізм лайків продемонстрували стабільну роботу саме завдяки продуманій структурі бази даних та оптимізованим алгоритмам обробки запитів.

Практична цінність розробленої платформи полягає не стільки в її поточному функціоналі, скільки в демонстрації життєздатності альтернативного підходу до створення музичних сервісів. Простота архітектури та використання стандартних технологій створюють міцну основу для подальшого розвитку, а виявлені недоліки є не стільки проблемами, скільки можливостями для вдосконалення та диференціації на ринку.

Рекомендації щодо подальшого дослідження випливають з комплексної природи виявлених викликів. Дослідження користувацького досвіду має стати пріоритетом для розуміння реальних потреб аудиторії та оптимізації балансу між простотою та функціональністю. Технічні дослідження в галузі оптимізації алгоритмів та впровадження сучасних вебтехнологій відкривають можливості для підтримки конкурентоспроможності продукту. Безпекові дослідження є критично важливими не лише для усунення поточних вразливостей, але й для превентивного захисту від майбутніх загроз.

Загалом, виконана робота успішно досягла поставленої мети створення функціональної вебплатформи, але її головна цінність полягає в демонстрації повного циклу розробки програмного продукту - від аналізу потреб ринку до критичної оцінки результатів та планування подальшого розвитку. Розроблена система заклала міцну основу для трансформації в повнофункціональний музичний сервіс, здатний конкурувати на ринку завдяки унікальному поєднанню простоти використання та відсутності реклами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. International Federation of the Phonographic Industry. Global Music Report 2024: State of the Industry / IFPI. London : IFPI, 2024. 48 p.

2. Garcia M. Modern Web Audio Processing with HTML5 / M. Garcia, L. Johnson // IEEE Computer Society. 2021. Vol. 54, No. 3. P. 78-86.

3. Українська музична платформа та її розвиток / А. К. Мельник, В. О. Петренко, С. М. Іваненко [та ін.] // Культура і мистецтво у сучасному світі. 2020. Вип. 21. С. 89-98.

4. Wilson B. Progressive Web Apps: Building Modern Web Applications / B. Wilson, A. Russell // ACM Computing Surveys. 2020. Vol. 53, No. 4. P. 1-34.

5. W3C Consortium. Web Audio API Specification / W3C. 2021. URL: https://webaudio.github.io/web-audio-api/ (дата звернення: 05.03.2025).

6. Stockhammer T. Dynamic Adaptive Streaming over HTTP: Standards and Design Principles / T. Stockhammer // Proceedings of the 2nd Annual ACM Conference on Multimedia Systems. New York : ACM, 2011. P. 133-144.

7. Netflix Technology Blog. Building a Scalable Media Streaming Platform // Netflix Tech Blog. 2021. URL: https://netflixtechblog.com/building-a-scalable-media-streaming-platform-5f2f853b98d8 (дата звернення: 20.03.2025).

8. Mozilla Developer Network. Web Audio API Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API (дата звернення: 08.03.2025).

9. Spotify Technology S.A. Annual Report 2023: Building the Future of Audio / Spotify. Stockholm : Spotify AB, 2024. 156 p.

10. Apple Music. Technical Specifications and Documentation. URL: https://support.apple.com/apple-music (дата звернення: 15.03.2025).

11. Google LLC. YouTube Music API Documentation. URL: https://developers.google.com/youtube/v3 (дата звернення: 10.03.2025).

12. SoundCloud Limited. SoundCloud API Documentation and Developer Resources. URL: https://developers.soundcloud.com/docs (дата звернення: 14.03.2025).

13. Ткаченко Л. П. Адаптивний вебдизайн: принципи та практика / Л. П. Ткаченко, М. В. Коваль // Інформаційні технології і засоби навчання. 2021. Т. 82, № 2. С. 134-148.

14. Кривонос О. М. Архітектура програмного забезпечення : навчальний посібник / О. М. Кривонос, В. А. Лукашенко. Київ : НАУ, 2020. 244 с.

15. Іванов А. С. Системи управління базами даних : підручник / А. С. Іванов, Т. В. Петрова. Одеса : Астропринт, 2019. 440 с.

16. Петров С. В. Проектування баз даних : теорія і практика / С. В. Петров, О. А. Сидоренко. Харків : Фоліо, 2020. 456 с.

17. Kamps J. Web Development with PHP and MySQL / J. Kamps, T. Welling. 3rd ed. Apress, 2020. 524 p.

18. Мельник В. В. Основи HTML5 та CSS3 : практичний курс / В. В. Мельник, Н. А. Горобець. Київ : Четверта хвиля, 2019. 368 с.

19. Flanagan D. JavaScript: The Definitive Guide / D. Flanagan. – 7th ed. O'Reilly Media, 2020. 1096 p.

20. Коваленко Д. П. Розробка вебдодатків на PHP : практичний посібник / Д. П. Коваленко. Київ : Видавничий дім "Києво-Могилянська академія", 2021. – 328 с.

21. Dahl R. Node.js: Evented I/O for V8 JavaScript / R. Dahl // Proceedings of JSConf. Berlin, 2018. – P. 22-35.

22. Robbins J. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics / J. Robbins. 5th ed. O'Reilly Media, 2018. – 808 p.

23. Hickson I. HTML5 Audio and Video / I. Hickson // W3C Working Draft. 2019. URL: https://www.w3.org/TR/html5/embedded-content-0.html (дата звернення: 12.03.2025).

24. Сидоренко О. В. Мультимедійні технології в веброзробці / О. В. Сидоренко // Комп'ютерні науки та інженерія. 2020. № 3. С. 45-52.

25. Spotify for Developers. Web API Reference. URL: https://developer.spotify.com/documentation/web-api (дата звернення: 16.03.2025).

26. Шевченко А. І. Користувацький досвід у вебдодатках / А. І. Шевченко, О. П. Коваленко // Наукові праці ВНТУ. 2021. № 1. С. 67-74.

27. Бойко Р. В. Вебтехнології та вебдизайн : навчальний посібник / Р. В. Бойко, О. М. Савчук. Львів : Магнолія-2006, 2020. 284 с.

28. Єгоров В. М. Основи вебпрограмування : навчальний посібник / В. М. Єгоров, С. П. Кузьменко. Харків : ХНУРЕ, 2020. 156 с.

29. Поліщук В. В. Основи вебтехнологій : підручник / В. В. Поліщук, А. М. Іващенко. Вінниця : ВНТУ, 2019. 284 с.

30. Левченко М. О. Інтернет-технології та вебдизайн : навчальний посібник / М. О. Левченко, К. В. Пономарьова. Дніпро : НметАУ, 2020. 200 с.

31. Дейтел П. Як програмувати на JavaScript / П. Дейтел, Х. Дейтел ; пер. з англ. Київ : БХВ-Петербург, 2019. 784 с.

32. Нікітін Ю. О. Безпека вебдодатків : навчальний посібник / Ю. О. Нікітін, Р. М. Скорик. Київ : НТУУ "КПІ", 2021. 188 с.

33. Куліков С. С. Тестування програмного забезпечення : підручник / С. С. Куліков. Львів : БаК, 2019. 312 с.

34. Гриценко В. І. Сучасні інформаційні технології : підручник / В. І. Гриценко, Г. О. Райко. Київ : Нора-Друк, 2018. 512 с.

35. Баранов О. А. Інформаційні системи і технології в економіці : підручник / О. А. Баранов, А. В. Гриценко. Київ : Фінанси і статистика, 2019. 368 с.

ДОДАТКИ

Додаток А

Технічне завдання

Найменування програмного забезпечення: Вебплатформа для прослуховування музики «MusicHub».

Підставою для виконання проекту є необхідність створення простої та доступної музичної платформи без реклами та складних інтерфейсів, що забезпечить безкоштовний доступ до прослуховування музичного контенту для широкого кола користувачів.

Призначення розробки: забезпечення зручного доступу користувачів до музичного контенту через вебінтерфейс з можливістю пошуку, відтворення композицій та взаємодії з контентом, а також надання адміністраторам інструментів для ефективного управління музичною бібліотекою.

Дана система повинна мати можливості:

– для користувача: будь-який користувач може переглядати каталог музичних композицій, здійснювати пошук за назвою або автором, відтворювати аудіофайли через вбудований плеєр, ставити лайки композиціям, завантажувати музичні файли;

– для системного адміністратора: управляти музичним контентом через спеціалізовану адміністративну панель, додавати нові композиції з аудіофайлами та зображеннями, редагувати інформацію про існуючі треки, видаляти застарілий контент, моніторити активність користувачів.

Вхідними даними є: музичні композиції з супроводжуючою інформацією. Кожна композиція характеризується: унікальний ідентифікатор, назва композиції, ім’я автора/виконавця, аудіофайл у форматі MP3, зображення обкладинки, посилання на зовнішній ресурс, дата додавання, кількість лайків та завантажень.

Вихідними даними є: повнофункціональна вебплатформа для прослуховування музики з інтуїтивним інтерфейсом та стабільною роботою всіх компонентів системи.

Перелік етапів розробки вебплатформи:

– аналіз існуючих музичних платформ та визначення вимог;

– проектування архітектури системи та бази даних;

– розробка концепції та дизайну користувацького інтерфейсу;

– створення HTML-структури та CSS-стилів;

– програмування серверної логіки на PHP;

– розробка та інтеграція JavaScript функціоналу;

– створення адміністративної панелі;

– тестування функціональності та безпеки;

– оптимізація продуктивності та налагодження.

Вебплатформа що розробляється повинна вирішувати такі задачі: принципові, функціональні, сервісні.

До принципових задач відносять:

– забезпечення надійного зберігання та швидкого доступу до музичної бібліотеки;

– організація мінімалістичного та інтуїтивного дизайну;

– забезпечення кроссбраузерної сумісності та адаптивності;

– мінімізація можливості виникнення помилок при взаємодії користувача з системою;

– контроль якості та безпеки завантажуваного контенту.

До функціональних задач відносять наступні:

– зберігання метаданих музичних композицій та користувацьких дій у базі даних;

– забезпечення операцій додавання, редагування та видалення музичного контенту;

– модерація та організація музичної бібліотеки через адміністративну панель;

– пошук композицій за назвою або автором з підтримкою часткових збігів;

– відтворення аудіофайлів через HTML5 Audio API;

– відстеження лайків та завантажень користувачів.

До сервісних задач відносяться:

– організація головної сторінки з каталогом музики;

– забезпечення зворотного зв’язку про статус операцій;

– надання зручних інструментів для управління контентом;

– забезпечення стабільної роботи потокового відтворення аудіо.

Основні вимоги до вебплатформи:

– для коректної роботи необхідні сучасні браузери Google Chrome 90+, Mozilla Firefox 88+, Safari 14+, Microsoft Edge 90+;

– мінімальні вимоги до клієнтського пристрою: 1 GB оперативної пам’яті, підтримка JavaScript та HTML5 Audio API;

– платформа оптимізована для роздільної здатності екрану від 1024×768 пікселів;

– використання сучасної темної кольорової схеми з зеленими акцентами;

– забезпечення візуального зворотного зв’язку для всіх інтерактивних елементів.

Вебплатформа ділиться на дві основні частини навігації: користувацький інтерфейс для прослуховування музики та адміністративна панель для управління контентом.

У процесі розробки вебплатформи використовується програмне забезпечення:

– Frontend: HTML5, CSS3, JavaScript для створення користувацького інтерфейсу;

– Backend: PHP для серверної логіки та обробки запитів;

– База даних: MySQL для зберігання метаданих та користувацьких дій;

– Середовище розробки: Visual Studio Code для написання коду;

– Локальний сервер: MAMP для тестування та налагодження;

– Вебсервер: Apache з підтримкою mod_rewrite та PHP.

Оцінка результатів розробки та доцільність її продовження здійснюється за такими представленими матеріалами:

– архів з вихідним кодом вебплатформи на резервному носії;

– детальний опис архітектури системи та інструкції по встановленню;

– звіт про проведене тестування з виявленими недоліками;

– керівництво адміністратора для управління контентом;

– керівництво користувача з описом всіх функцій платформи.

Перелік документів: технічне завдання, пояснювальна записка до проекту, керівництво адміністратора, керівництво користувача, результати тестування системи.

Звітні матеріали подаються у паперовому та електронному вигляді, оформлені згідно з вимогами навчального закладу до курсових робіт.

Тестування виконується відповідно до розробленої програми тестування, що включає функціональне тестування всіх модулів, тестування безпеки, продуктивності та кроссбраузерної сумісності.

Терміни виконання: розробка вебплатформи здійснюється протягом одного семестру згідно з навчальним планом кваліфікаційної роботи.

Критерії приймання: система вважається готовою при успішному проходженні всіх етапів тестування, відповідності технічним вимогам та демонстрації стабільної роботи всього заявленого функціоналу.

Додаток Б
Керівництво користувача

Для зручності роботи з вебплатформою "MusicHub" при проектуванні інтерфейсу враховано, що адміністратор та звичайний користувач мають різні можливості та рівні доступу до функціональності системи.

Для використання даної платформи передбачено два рівні доступу: доступ для звичайного користувача (він має можливість переглядати музичний каталог, здійснювати пошук, прослуховувати композиції, ставити лайки та переходити до кліпу пісні) та доступ для адміністратора (він має повний доступ до управління музичним контентом через спеціалізовану адміністративну панель).

Відкривши платформу, користувач потрапляє на головну сторінку, яка зображена на рисунку В.1.

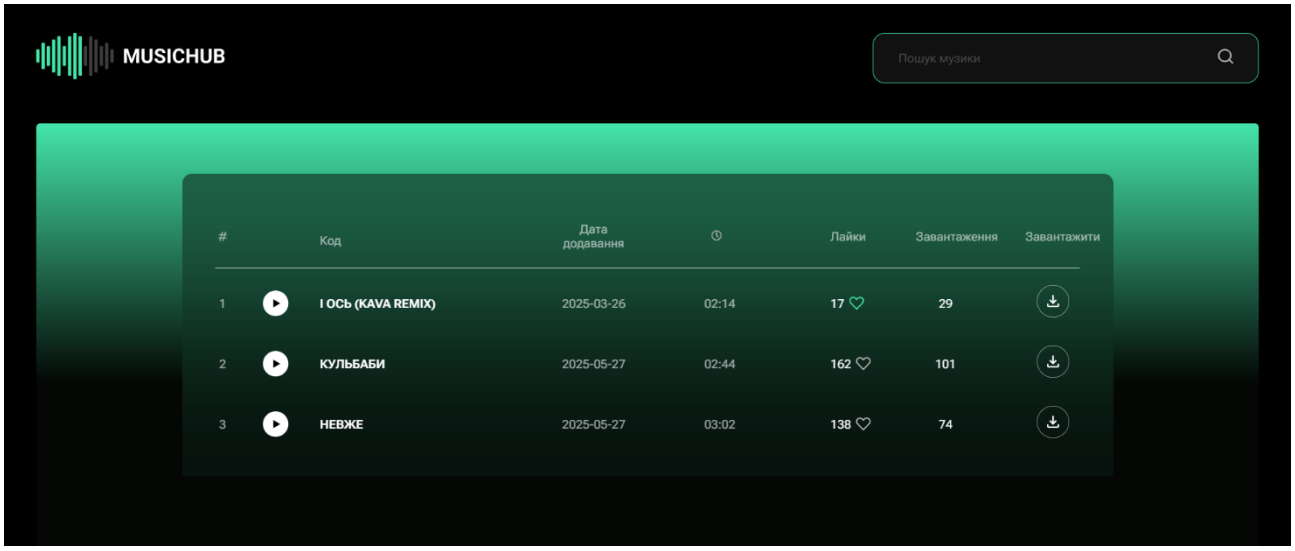


Рисунок В.1 – Головна сторінка

Головна сторінка відображає весь доступний музичний каталог у вигляді структурованої таблиці. Інтерфейс є достатньо простим та інтуїтивним у використанні, всі елементи розміщені компактно для швидкого доступу до функцій платформи. Кожен рядок таблиці містить інформацію про окрему музичну композицію: порядковий номер, назву треку з вбудованим аудіоплеєром, дату додавання, тривалість композиції, кількість лайків, кількість завантажень та кнопку для завантаження файлу.

У верхній частині сторінки знаходиться поле пошуку, яке зображено на рисунку В.2.

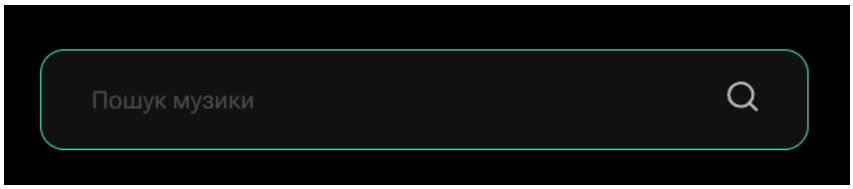


Рисунок В.2 – Поле пошуку музики

Дане поле призначене для швидкого знаходження потрібних композицій на платформі. Користувач може вводити назву треку або ім’я виконавця, система автоматично фільтрує результати за частковими збігами в режимі реального часу.

Для відтворення музики користувач натискає на кнопку відтворення біля назви композиції. Аудіоплеєр, який зображено на рисунку В.3.

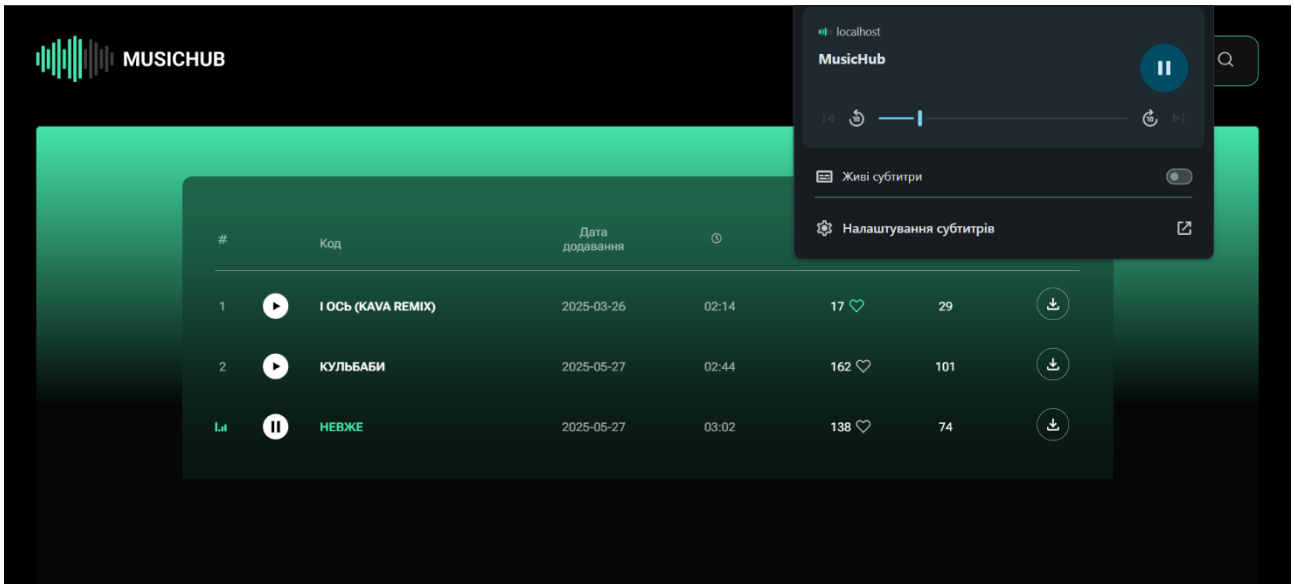


Рисунок В.3 – Вбудований аудіоплеєр

Плеєр використовує стандартні HTML5 елементи управління та дозволяє відтворювати, призупиняти композицію, регулювати гучність та переміщатися по треку. Система автоматично зупиняє попередньо відтворювану композицію при запуску нової.

Користувач може виражати своє схвалення композицій через систему лайків, яка зображена на рисунку В.4.

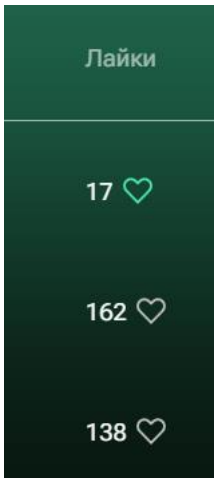


Рисунок В.4 – Система лайків

При натисканні на іконку серця кількість лайків збільшується на одиницю, а іконка змінює свій вигляд на активний стан. Система запобігає повторним лайкам від одного користувача в межах поточної сесії.

Для завантаження музичних файлів на свій пристрій користувач може скористатися кнопкою завантаження.

При натисканні на кнопку завантаження автоматично розпочинається збереження аудіофайлу на пристрій користувача, одночасно збільшується лічильник завантажень для даної композиції.

Результат успішного пошуку музики зображено на рисунку В.5.

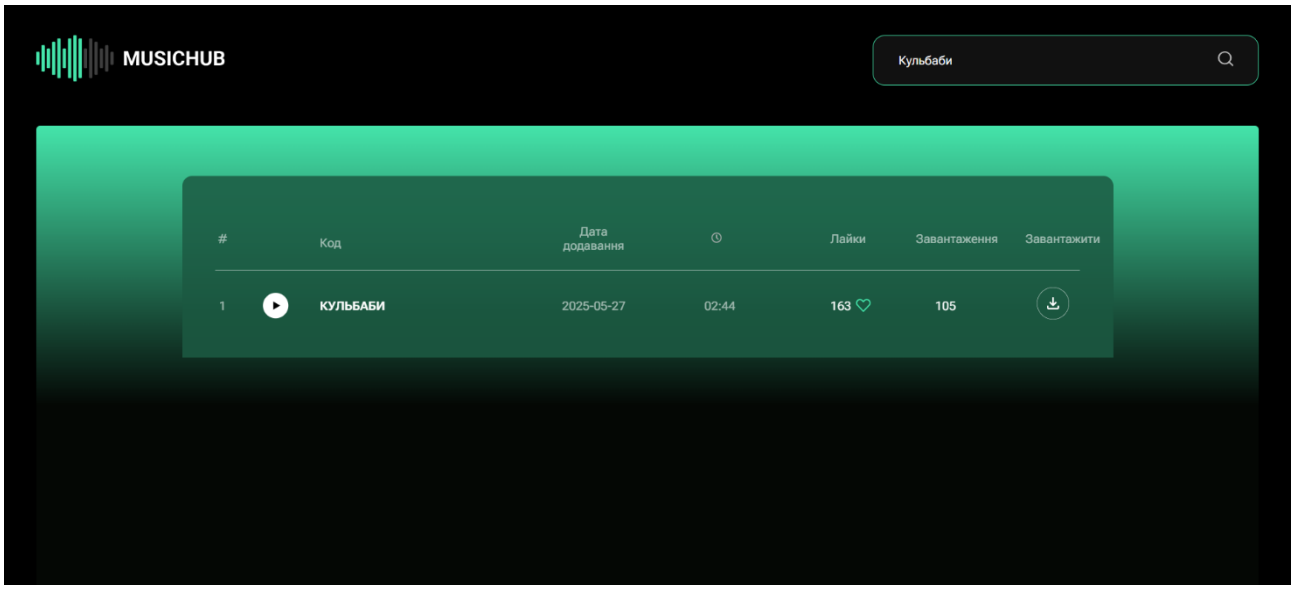


Рисунок В.5 – Результати пошуку

Після введення пошукового запиту система відображає тільки ті композиції, які відповідають критеріям пошуку. Якщо збігів не знайдено, відображається порожня таблиця.

Для доступу до адміністративних функцій необхідно перейти за адресою /admin/ та виконати автентифікацію. Вікно входу для адміністратора зображено на рисунку В.6.

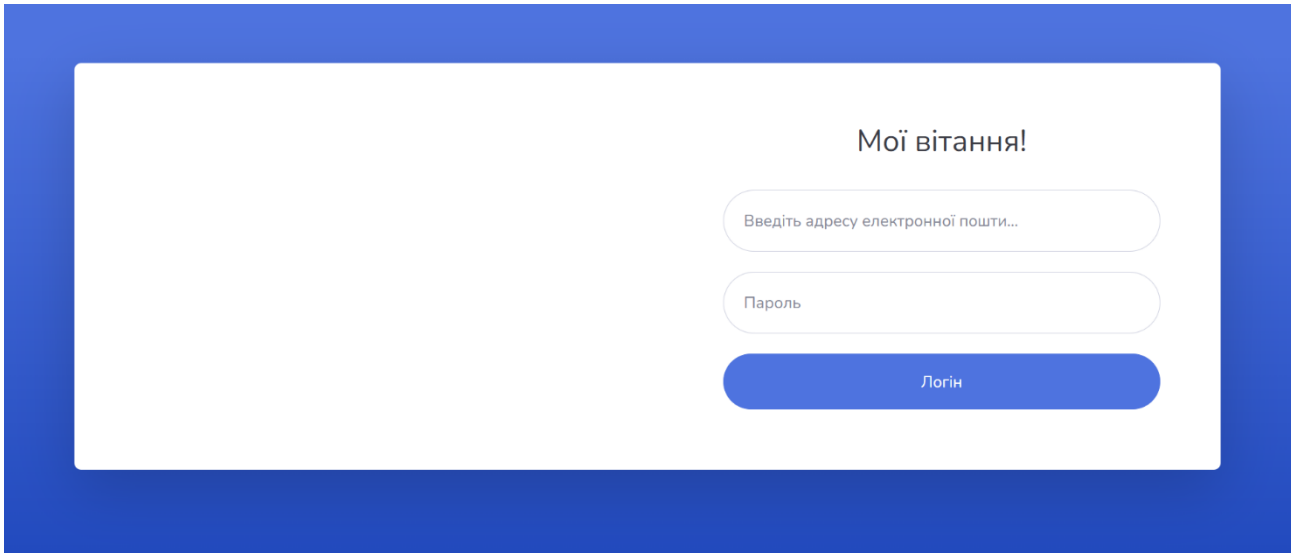


Рисунок В.6 – Сторінка входу адміністратора

Дана сторінка призначена для безпечного входу адміністратора в систему управління контентом. Необхідно ввести електронну адресу та пароль адміністратора.

Після успішної автентифікації адміністратор потрапляє на головну сторінку адміністративної панелі, яка зображена на рисунку В.7.

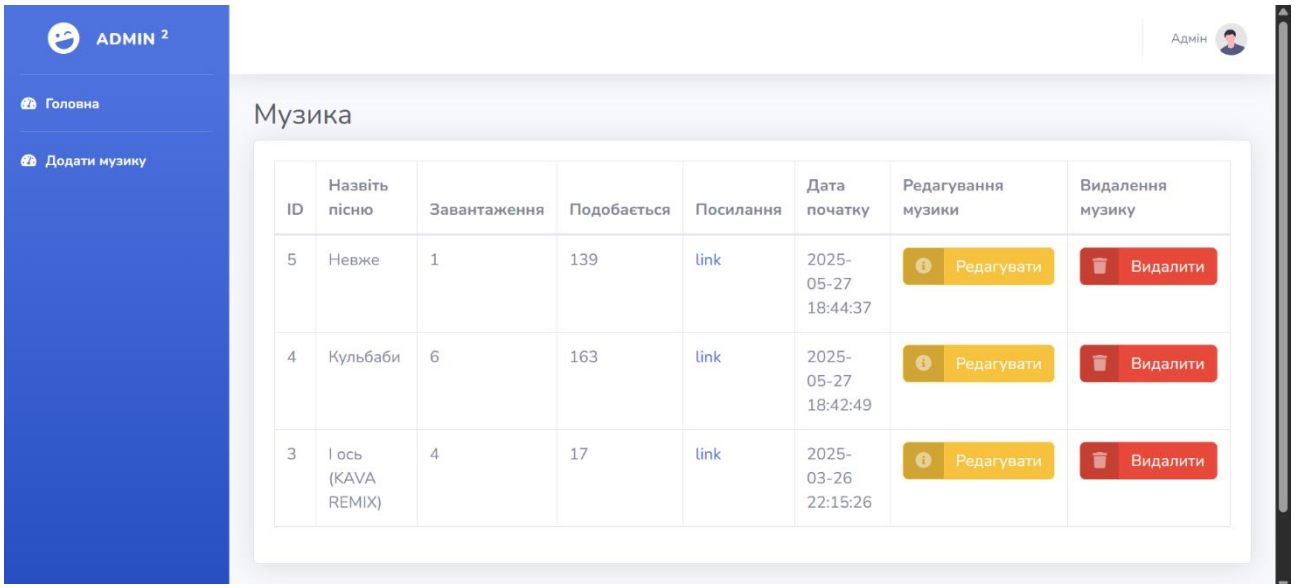


Рисунок В.7 – Головна сторінка адміністративної панелі

Головна сторінка адмін-панелі відображає всі музичні композиції у вигляді таблиці з додатковими колонками для управління: ID композиції, назва, кількість завантажень, кількість лайків, посилання на зовнішній ресурс, дата додавання та кнопки для редагування і видалення.

Для додавання нових композицій адміністратор використовує спеціальну форму, яка зображена на рисунку В.8.

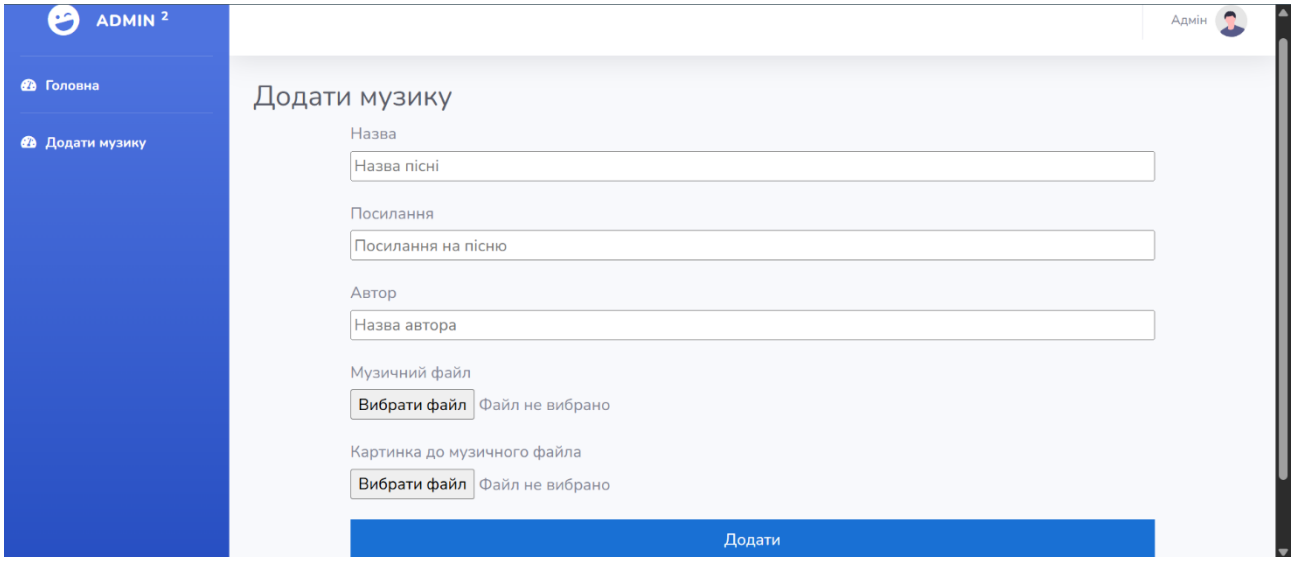


Рисунок В.8 – Форма додавання нової музики

Дана форма дозволяє адміністратору завантажити новий музичний контент, вказавши назву композиції, ім’я автора, посилання на зовнішній ресурс, а також завантажити аудіофайл та зображення обкладинки. Всі поля є обов’язковими для заповнення.

Процес редагування існуючої композиції зображено на рисунку В.9.

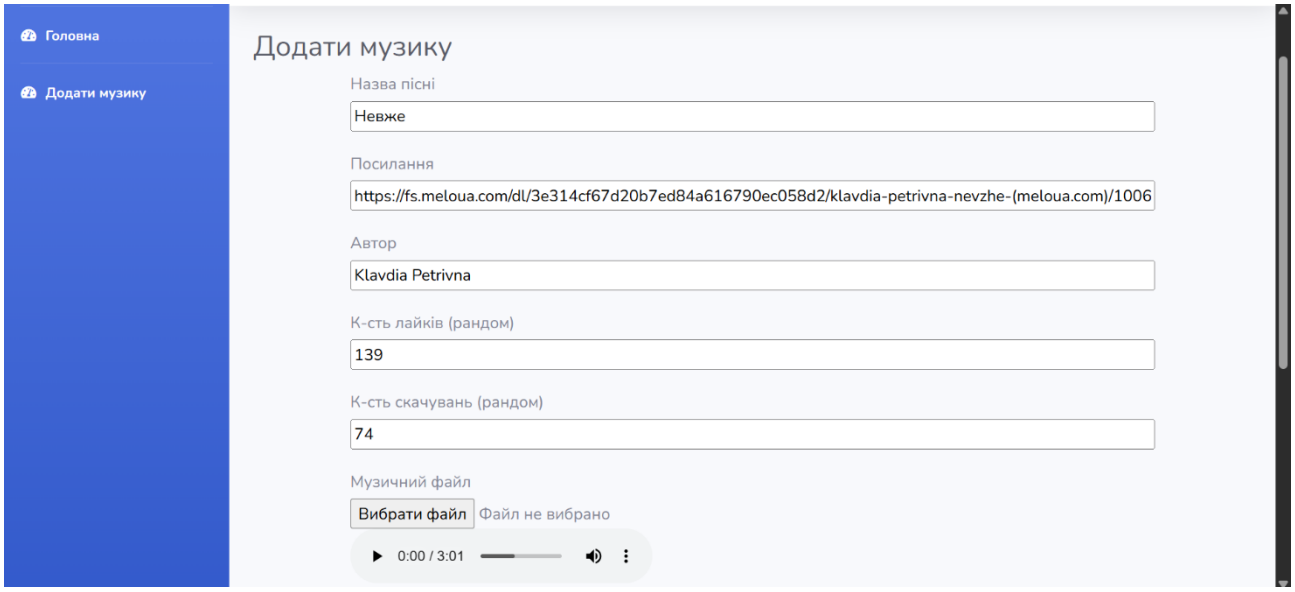


Рисунок В.9 – Форма редагування музики

Форма редагування попередньо заповнена поточними даними композиції та дозволяє адміністратору змінювати будь-які параметри, включаючи заміну аудіофайлу та зображення обкладинки. Поточні медіафайли відображаються для попереднього перегляду.

Після успішного збереження нової композиції система автоматично перенаправляє адміністратора на головну сторінку панелі, де можна побачити новододаний контент у списку.

Процес видалення композиції зображено на рисунку В.10.

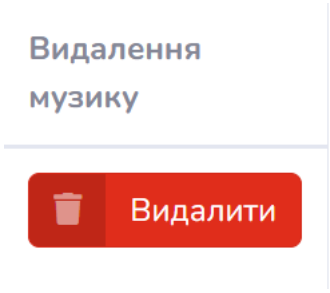


Рисунок В.10 – Видалення композиції

При натисканні на кнопку видалення композиція негайно видаляється з бази даних. Система не запитує додаткового підтвердження, тому адміністратор повинен бути обережним при виконанні цієї операції.

Вихід з адміністративної панелі здійснюється через відповідну кнопку в меню, яку зображено на рисунку В.11.

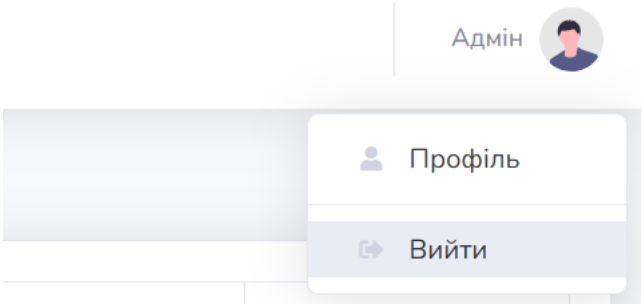


Рисунок В.11 – Кнопка виходу з адмін-панелі

При виході система повністю знищує сесію адміністратора та перенаправляє на сторінку входу для запобігання несанкціонованому доступу.

Додаток В

Тестування вебплатформи

Таблиця Г.1

Тестування модуля відтворення та управління музикою

Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
Відтворення MP3 файлу	Натискання кнопки Play на композиції	Початок відтворення, оновлення UI	Аудіо відтворюється, кнопка змінюється на Pause	Успішно
Пауза відтворення	Натискання кнопки Pause	Зупинка відтворення, збереження позиції	Відтворення призупинено на поточній позиції	Успішно
Перемикання між треками	Запуск нової композиції під час відтворення іншої	Зупинка попереднього, запуск нового треку	Попередній трек зупинився, новий розпочався	Успішно
Відображення тривалості	Завантаження композиції з metadata	Показ тривалості треку в інтерфейсі	Тривалість відображається як MM:SS	Успішно
Управління гучністю	Зміна повзунка гучності	Зміна рівня звуку відтворення	Гучність змінюється відповідно до позиції	Успішно
Завантаження великого файлу	MP3 файл розміром 8MB	Поступове завантаження, можливість відтворення	Файл завантажується прогресивно, відтворення починається	Успішно

Таблиця Г.2

Тестування модуля пошуку музичного контенту

Тестовий сценарій	Параметри пошуку	Очікуваний результат	Фактичний результат	Статус
Пошук за назвою композиції	"І ось"	Відображення композицій з відповідною назвою	Знайдено 1 композицію "І ось (KAVA REMIX)"	Успішно
Пошук за автором	"Геля Зозуля"	Список всіх композицій автора	Відображено композиції виконавця Геля Зозуля	Успішно
Частковий пошук	"KAVA"	Композиції з частковим збігом	Знайдена композиція з "KAVA" в назві	Успішно
Пошук без результатів	"Невідомий виконавець"	Порожній список або повідомлення	Відображається порожній список	Успішно
Пошук з спеціальними символами	"І ось!"	Обробка спецсимволів у запиті	Пошук працює коректно, спецсимволи екрануються	Успішно
Пошук з порожнім запитом	Порожнє поле пошуку	Відображення всього каталогу	Показані всі доступні композиції	Успішно

Таблиця Г.3

Тестування системи лайків та завантажень

Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
Перший лайк композиції	Клік на іконку лайку	Збільшення лічильника на 1, зміна іконки	Лічильник збільшився, іконка стала активною	Успішно

Повторний лайк в сесії	Повторний клік на лайк	Запобігання подвійному лайку	Лайк не додається повторно в межах сесії	Успішно
Завантаження музичного файлу	Клік на кнопку завантаження	Початок завантаження файлу, збільшення лічильника	Файл завантажується, лічильник оновлюється	Успішно
Множинні завантаження	Кілька кліків на завантаження	Кожне натискання збільшує лічильник	Лічильник коректно відображає кількість завантажень	Успішно
Відображення популярності	Композиція з великою кількістю лайків	Коректне відображення числових значень	Лічильники відображаються правильно	Успішно

Таблиця Г.4

Тестування модуля автентифікації адміністратора

Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
Вхід з правильними даними	Email: admin@admin, Password: admin	Успішна авторизація, доступ до панелі	Користувач авторизований, створена сесія admin	Успішно
Вхід з неправильним паролем	Email: admin@test.com, Password: wrong	Повідомлення про помилку	Відображається помилка авторизації	Успішно

Вхід з порожніми полями	Email: "", Password: ""	Валідаційна помилка	HTML5 валідація блокує відправку форми	Успішно
Доступ без авторизації	Пряме звернення до /admin/index.php	Перенаправлення на сторінку входу	Автоматичне перенаправлення на login.php	Успішно
Вихід з системи	Клік на кнопку "Вийти"	Знищення сесії, перенаправлення	Сесія знищена, користувач перенаправлений	Успішно

Таблиця Г.5

Тестування модуля управління музичним контентом

Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
Додавання нової композиції	Назва, автор, аудіофайл MP3, зображення JPG	Успішне створення запису в БД	Композиція додана, присвоєно унікальний ID	Успішно
Додавання без аудіофайлу	Тільки текстові поля заповнені	Валідаційна помилка	HTML5 валідація вимагає аудіофайл	Успішно
Додавання з великим файлом	MP3 файл розміром 15MB	Успішне завантаження або помилка розміру	Файл завантажений, генерується унікальне ім’я	Успішно
Редагування існуючої композиції	Зміна назви та автора	Оновлення даних в базі	Дані оновлені, зберігаються зміни	Успішно

Видалення композиції	ID існуючої композиції	Видалення запису з БД	Запис видалений, файли залишаються	Успішно
Спроба видалення неіснуючого ID	ID = 999999	Помилка або ігнорування	Система коректно обробляє помилку	Успішно

Таблиця Г.6

Тестування безпеки системи

Тип атаки	Тестовий випадок	Результат тестування	Статус
SQL-ін’єкція в пошуку	Введення ‘; DROP TABLE music__list; -- в поле пошуку	Дані екрануються через prepared statements	Захищено
XSS-атака	Введення <script>alert(‘XSS’)</script> в назву композиції	Скрипт відображається як текст	Захищено
Несанкціонований доступ до адмін-панелі	Звернення до /admin/ без авторизації	Перенаправлення на сторінку входу	Захищено
Завантаження шкідливих файлів	Спроба завантаження .php файлу як аудіо	Перевірка MIME-типу файлу	Захищено
CSRF-атака на форми	Відправка форми з іншого домену	Відсутність CSRF-токенів	Частково

Таблиця Г.7

Тестування продуктивності

Операція	Об’єм даних	Час відповіді	Критерій	Статус
Завантаження головної сторінки	10 композицій	0.8 сек	< 2 сек	Успішно
Пошук без фільтрів	50 записів	0.3 сек	< 1 сек	Успішно
Пошук з фільтром по назві	50 записів	0.4 сек	< 1 сек	Успішно
Завантаження аудіофайлу	MP3 файл 5MB	1.2 сек	< 3 сек	Успішно
Відгук на лайк	Оновлення лічильника	0.2 сек	< 0.5 сек	Успішно
Додавання нової композиції	З аудіо 3MB та зображенням 500KB	2.1 сек	< 3 сек	Успішно

Таблиця Г.8

Тестування кроссбраузерної сумісності

Браузер	Версія	Відтворення аудіо	UI відображення	JavaScript функції	Статус
Google Chrome	119.0	Повна підтримка	Коректне відображення	Всі функції працюють	Успішно
Mozilla Firefox	120.0	Повна підтримка	Коректне відображення	Всі функції працюють	Успішно
Microsoft Edge	119.0	Повна підтримка	Коректне відображення	Всі функції працюють	Успішно
Safari	17.1	Обмежена підтримка autoplay	Незначні відмінності в стилях	Основні функції працюють	Частково
Opera	105.0	Повна підтримка	Коректне відображення	Всі функції працюють	Успішно

Таблиця Г.9

Тестування адаптивності інтерфейсу

Розширення екрану	Розмір	Відображення таблиці музики	Форма пошуку	Медіаплеєр	Статус
Desktop HD	1920x1080	Повна таблиця з усіма колонками	Горизонтальне розташування	Стандартні контроли	Успішно
Desktop Standard	1366x768	Повна таблиця з усіма колонками	Горизонтальне розташування	Стандартні контроли	Успішно
Laptop	1280x720	Адаптивна ширина колонок	Скорочена форма пошуку	Компактні контроли	Успішно
Tablet Landscape	1024x768	Скорочені колонки	Адаптивне розташування	Збільшені кнопки	Успішно