**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА**

**Навчально-науковий інститут фізико-технічних та комп’ютерних наук**

**кафедра комп’ютерних наук**

**Сервіс для прослуховування аудіо контенту**

**Курсова робота**

**Рівень вищої освіти – перший (бакалаврський)**

***Виконав:***

студент 3 курсу, 344-А групи

Кульчицький Андрій Сергійович

***Керівник****:*

Оксана Мирославівна Яцько

**Чернівці – 2024Чернівецький національний університет імені Юрія Федьковича**

Навчально-науковий інститут фізико-технічних та комп’ютерних наук

Кафедра Комп’ютерних наук

Спеціальність Комп’ютерні науки

Освітній ступінь Бакалавр

Форма навчання денна курс 3 група 344А

ЗАТВЕРДЖУЮ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Завідувач кафедри | | |  | Ушенко Ю.О. |
|  | | | ( підпис) | (ініціали, прізвище) |
|  |  |  | 2024 р. | |

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТА

|  |
| --- |
| Кульчицький Андрій Сергійович |
| (прізвище , ім’я, по батькові) |

1. Тема роботи

|  |  |
| --- | --- |
| Сервіс для прослуховування аудіо контенту | |
|  |  |

затверджена протоколом засідання кафедри від « » 2024 року № 1

1. Термін подання студентом закінченої роботи 24.05.2024
2. Вхідні дані до роботи

|  |
| --- |
| Середовища Visual Studio Code , бібліотеки ReactJS , Express.js , MongoDB , Node.js , Mongoose, Firebase |

1. Зміст розрахунково-пояснювальної записки (перелік питань, які треба

розробити)

|  |
| --- |
| Вступ |
| Розділ 1 аналіз предметної ОБЛАСТІ |
| Розділ 2 ПРОЕКТУВАННЯ СИСТЕМИ. АНАЛІЗ ПІДХОДІВ |
| РОЗДІЛ 3 Особливості кодової імплементації програмного |
| застосунку |
| Розділ 4 ОГЛЯД ІМПЛЕМЕНТОВАНИХ ФУНКЦІОНАЛЬНИХ |
| МОЖЛИВОСТЕЙ ІНФОРМАЦІЙНОЇ СИСТЕМИ. |
| Висновки |

Список використаних джерел

1. Перелік графічного, наочного матеріалу

|  |
| --- |
| Скріншоти інтерфейсу, коду, сайтів-аналогів |

1. Консультант(и) курсової роботи

|  |
| --- |
| Асистент кафедри Яцько О.М. – надав консультації на рахунок проектування системи а так же в написані курсової |

**КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КУРСОВОЇ РОБОТИ**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів курсової роботи | Термін виконання етапів роботи | Примітки |
| 1 | Отримання завдання на курсову роботу | 03.02.2024 | виконано |
| 2 | Аналіз предметної області, дослідження літератури та матеріалів на задану тему | 16.02.2024 | виконано |
| 3 | Аналіз існуючих аналогів програмного забезпечення | 24.02.2024 | виконано |
| 4 | Постановка задачі за темою курсової роботи | 11.03.2024 | виконано |
| 5 | Вибір інструментальних засобів розробки системи | 25.03.2024 | виконано |
| 6 | Проєктування структури та алгоритму роботи розроблюваної системи | 6.04.2024 | виконано |
| 7 | Розробка компонентів системи | 28.04.2024 | виконано |
| 8 | Розгортання та тестування розробленої системи | 2.05.2024 | виконано |
| 9 | Написання розділів пояснювальної записки | 12.05.2024 | виконано |
| 10 | Представлення закінченої роботи на перевірку | 19.05.2024 | виконано |
| 11 | Захист курсової роботи | 24.05.2024 | виконано |

|  |  |  |
| --- | --- | --- |
| **Студент** |  | Кульчицький А.С. |
|  | (підпис) |  |
| **Науковий керівник** |  | Яцько О.М. |
|  | (підпис) |  |
| «\_\_\_» \_\_\_\_ 2024 р. | |  |

## АНОТАЦІЯ

У цій курсовій роботі ведеться розробка програмного забезпечення – сервісу для прослуховування аудіо контенту. У роботі проаналізовано сьогоденні тенденцій розвитку ринку аудіо контенту та програмних інновацій у данній області. Визначено основні функціональні вимоги до програмного застосунку, а також опис до завдання, мети роботи та MVP-версію продукту.

Надалі розглянуті основні принципи архітектури системи та список технологій для розробки back-end частини, включаючи модулі безпеки, аутентифікації користувачів, завантаження та видалення музики на платформу, інтеграції з firebase.

Крім того, описана реалізація front-end частини програми, включаючи модулі аутентифікації, списку музики, альбомів та виконавців, керування іншими користувачами та прослуховування аудіо контенту.

У розділі "Огляд імплементованих функціональних можливостей інформаційної системи" описані функціональності для адміністратора та звичайного користувача.

В результаті роботи спроектовано та реалізовано функціонал та візуальну частину для платформи для взаємодії з аудіо контентом, яка відповідає стандартам розробки програмного забезпечення та потребам користувачів.

Сторінок – 40, Зображень – 21 , Посилань на джерела - 10

Ключові слова : Програмний забезпечення, аудіо, прослуховування, аналіз, функціональні вимоги, архітектура, back-end, front-end, модулі, безпека, аутентифікація, користувачі, список пісень, Firebase, користувач, адміністратор.

А.С. Кульчицький

**ANNOTATION**

In this term paper, the development of software - a service for listening to audio content is underway. The paper analyzes the current trends in the development of the audio content market and software innovations in this area. The main functional requirements for the software application, as well as a description of the task, the purpose of the work, and the MVP version of the product are defined.

Further, the main principles of the system architecture and a list of technologies for the development of the back-end part, including security modules, user authentication, uploading and deleting music to the platform, integration with firebase, are considered.

In addition, the implementation of the front-end part of the program is described, including modules for authentication, music list, albums and artists, managing other users and listening to audio content.

The section "Overview of the implemented functionality of the information system" describes the functionality for the administrator and the regular user.

As a result of the work, the functionality and visual part for the platform for interacting with audio content, which meets the standards of software development and user needs, was designed and implemented.

Pages – 40, Images – 33, Links to sources – 10

Keywords : Software, audio, listening, analysis, functional requirements, architecture, back-end, front-end, modules, security, authentication, users, playlist, Firebase, user, admin.

А.S. Kulchytskyi

## ЗМІСТ

[АНОТАЦІЯ 4](#_Toc167425916)

[ЗМІСТ 2](#_Toc167425917)

[Вступ 4](#_Toc167425918)

[РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 6](#_Toc167425919)

[1.1 Аналіз сучасних тенденцій розвитку предметної області та відповідного ринку сучасних програмних рішень 6](#_Toc167425920)

[1.2 Аналіз сервісів-аналогів 6](#_Toc167425921)

[1.3 Функціональні вимоги програмного застосунку 8](#_Toc167425922)

[1.3.1 Визначення завдання і мети роботи 8](#_Toc167425923)

[1.3.2 Опис продукту 8](#_Toc167425924)

[1.3.3 MVP-версія продукту 9](#_Toc167425925)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ. АНАЛІЗ ПІДХОДІВ 10](#_Toc167425926)

[2.1 Проектування структури бази даних 10](#_Toc167425927)

[2.2 Розробка структурно-логічної схеми бекенду та архітектурних компонентів застосунку 13](#_Toc167425928)

[2.2.1 Модуль Аутентифікації та авторизації 14](#_Toc167425929)

[2.2.2Модуль користувачів 15](#_Toc167425930)

[2.2.3 Модуль Аудіо 15](#_Toc167425931)

[2.2.4 Модуль Авторів 16](#_Toc167425932)

[2.2.5 Модуль Альбомів 16](#_Toc167425933)

[2.2.6 Модуль взаємодії з Firebase 17](#_Toc167425934)

[2.3 UX/UI дизайн програмного застосунку 18](#_Toc167425935)

[2.3.1 Персоналії 19](#_Toc167425936)

[2.3.2 Wireframes 20](#_Toc167425937)

[РОЗДІЛ 3 Особливості кодової імплементації програмного застосунку. 23](#_Toc167425938)

[3.1 Back-end розробляємої платформи. 23](#_Toc167425939)

[3.1.1 Особливості кодової імплементації модуля безпеки, аутентифікації та авторизації 23](#_Toc167425940)

[3.1.2 Особливості кодової імплементації модуля Аудіо 24](#_Toc167425941)

[3.1.3 Особливості кодової імплементації модуля api 26](#_Toc167425942)

[3.2 Front-end розробляємої платформи 28](#_Toc167425943)

[3.2.1 Особливості імплементації модуля відображення контенту 29](#_Toc167425944)

[3.2.2 Особливості кодової імплементації модуля плеєра 30](#_Toc167425945)

[3.2.3 Особливості кодової імплементації модуля додавання контенту 33](#_Toc167425946)

[РОЗДІЛ 4 ОГЛЯД ІМПЛЕМЕНТОВАНИХ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ІНФОРМАЦІЙНОЇ СИСТЕМИ. 34](#_Toc167425947)

[4.1 Функціонал Адміністратора 34](#_Toc167425948)

[4.2 Функціонал звичайного користувача 35](#_Toc167425949)

[ВИСНОВОК 36](#_Toc167425950)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 38](#_Toc167425951)

[ДОДАТОК А 38](#_Toc167425952)

## Вступ

**Мета:**

Розробити багато функціональний та зручний у користуванні веб-сайт для прослуховування аудіо контенту, який забезпечить можливістю легко та в чудовій якості прослуховувати музичні композиції.

**Завдання:**

1. **Аналіз ринку та потреб користувачів:**

* Провести дослідження ринку для виявлення основних конкурентів та їх пропозицій.
* Визначити цільову аудиторію та її потреби.
* Вивчити сучасні тенденції у сфері прослуховування аудіо.

1. **Розробка архітектури сайту:**

* Спроектувати логічну структуру сайту, включаючи основні розділи та підрозділи.
* Визначити функціональні можливості сайту, такі як реєстрація користувачів, можливість прослуховувати музику, завантажувати аудіо файли, додавати альбоми та нових авторів

1. **Дизайн та юзабіліті:**

* Розробити інтерфейс користувача (UI), що забезпечить зручну навігацію та естетичний вигляд сайту.
* Забезпечити високий рівень юзабіліті (UX), щоб користувачі могли легко зручно користуватись програвачем.

1. **Технічна реалізація:**

* Обрати відповідні технології та платформи для розробки сайту (HTML, CSS, JavaScript, фреймворки, CMS).
* Реалізувати back-end частину сайту, включаючи базу даних для зберігання самих аудіофайлів та інформації щодо альбомів та авторів.

Такий підхід дозволить створити продукт, що відповідатиме вимогам користувачів та сприятиме бізнесу у сфері прослуховування аудіо контенту.

Проєкт має на меті дослідити ключові аспекти сервісів для прослуховування музики та розробити відповідне програмне рішення, яке відповідатиме сучасним стандартам ринку. Важливо врахувати потреби користувачів у зручності використання сайту. Аналіз ринку та конкурентів допоможе визначити унікальні переваги розробленого продукту.

Під час розробки програмного забезпечення слід враховувати актуальні технології та тенденції у сфері прослуховування аудіо контенту. Це допоможе розробити сервіс , який буде конкурентоспроможним. Необхідно також вивчити найкращі практики та інновації у галузі для успішного впровадження програмного продукту.

Результатом цього дослідження стане веб-додаток, який спростить процес прослуховування музики та підвищить зручність і швидкість доступу до контенту.

Основні функціональні можливості веб-додатку включатимуть, можливість прослуховування треків, систему аутентифікації для звичайних користувачів, а також функції адміністратора: додавання та видалення музики, авторів, альбомів, управління користувачами та надання або позбавлення прав адміністратора. Додаткові функції, такі як створення персоналізованих плейлистів, збереження улюблених треків та налаштування особистого профілю, також можуть бути реалізовані у майбутньому для забезпечення більш персоналізованого та зручного досвіду користувачів.

Таким чином, розроблений веб-додаток сприятиме розвитку бізнесу у сфері прослуховування аудіо контенту, задовольняючи потреби користувачів.

## РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

### 1.1 Аналіз сучасних тенденцій розвитку предметної області та відповідного ринку сучасних програмних рішень

На данний момент є великий вибір програмних рішень для прослуховування аудіо контенту, що відповідає на зростаючий попит споживачів на зручні та ефективні методи доступу до музики. Основні тенденції розвитку цієї предметної області можна розділити на декілька ключових аспектів:

* Зростання популярності онлайн-сервісів для прослуховування музики. За останні роки спостерігається значне зростання популярності онлайн-сервісів для прослуховування музики через збільшення кількості аудіо контенту та його доступності.
* Конкуренція: За останній час значно збільшилась кількість компаній, які пропонують програмні рішення для онлайн-замовлення їжі. Це призвело до зростання конкуренції на ринку та стимулює компанії до постійного вдосконалення своїх продуктів і сервісів.

### 1.2 Аналіз сервісів-аналогів

Для розуміння поточної ситуації на ринку та визначення основних конкурентних переваг і недоліків, розглянемо деякі популярні сервіси для прослуховування аудіо контенту, зокрема музики. Ми проаналізуємо їх функціональність, зручність користування, а також переваги і недоліки.

1. Spotify є одним з найбільш популярних музичних сервісів у світі, що дозволяє користувачам стрімінг музики, створення плейлистів, а також відкриває доступ до величезної бібліотеки музичних творів різних жанрів.

Переваги:

* Велика бібліотека музики.
* Можливість створювати та ділитися плейлистами.
* Доступність на різних платформах (мобільні додатки, веб-версія, настільні додатки).
* Висока якість звуку.

Недоліки:

* Безкоштовна версія обмежена рекламою та деякими функціями.
* Деякі функції, як-от можливість прослуховування офлайн, доступні тільки в преміум версії.

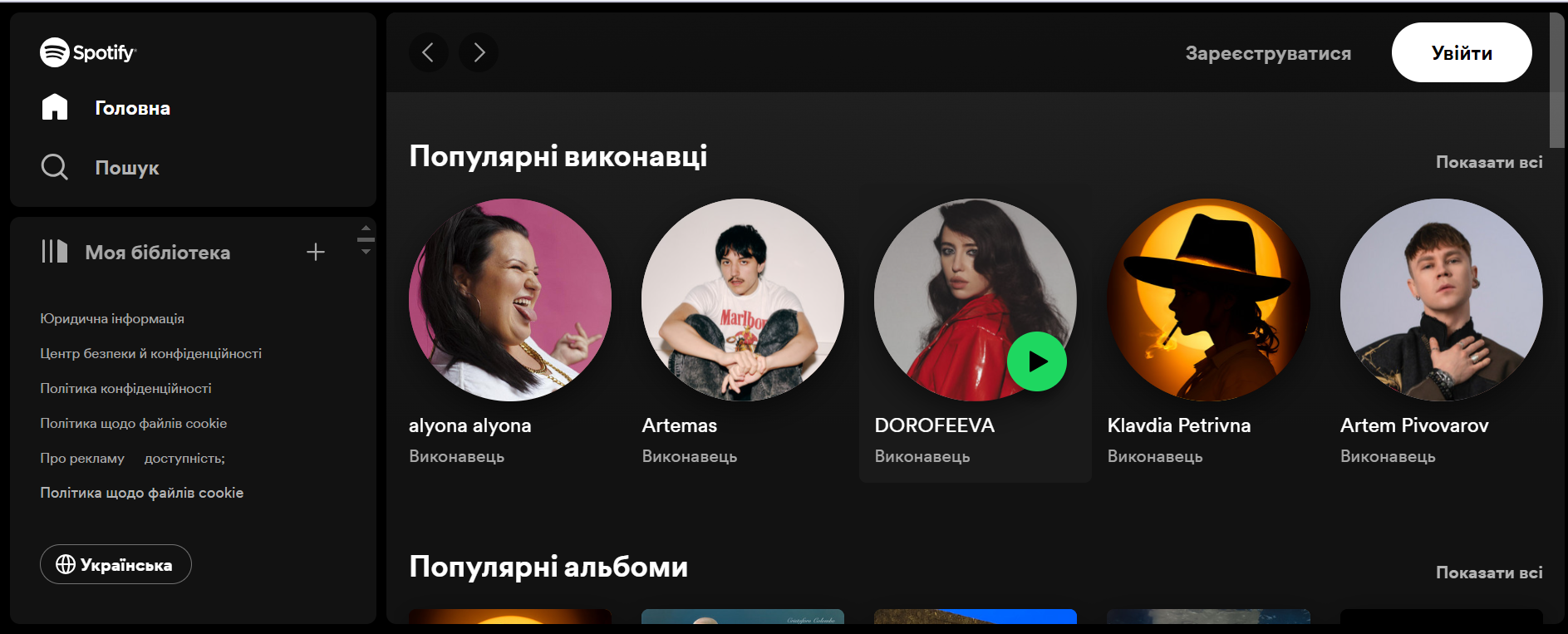


Рис.1.1 – Вебсайт Spotify головна сторінка

2. Apple Music це сервіс для стрімінгу музики від Apple, який пропонує велику бібліотеку пісень, ексклюзивні треки та інші музичні ресурси.

Переваги:

* Глибока інтеграція з екосистемою Apple.
* Доступ до ексклюзивного контенту.
* Висока якість звуку та велика бібліотека треків.
* Можливість завантажувати музику для офлайн прослуховування.
* Включення радіо станцій і плейлистів, створених кураторами.

Недоліки:

* Відсутність безкоштовної версії.
* Більшість функцій доступні тільки для користувачів продуктів Apple.
* Обмежені можливості налаштування та кастомізації плейлистів порівняно з конкурентами.

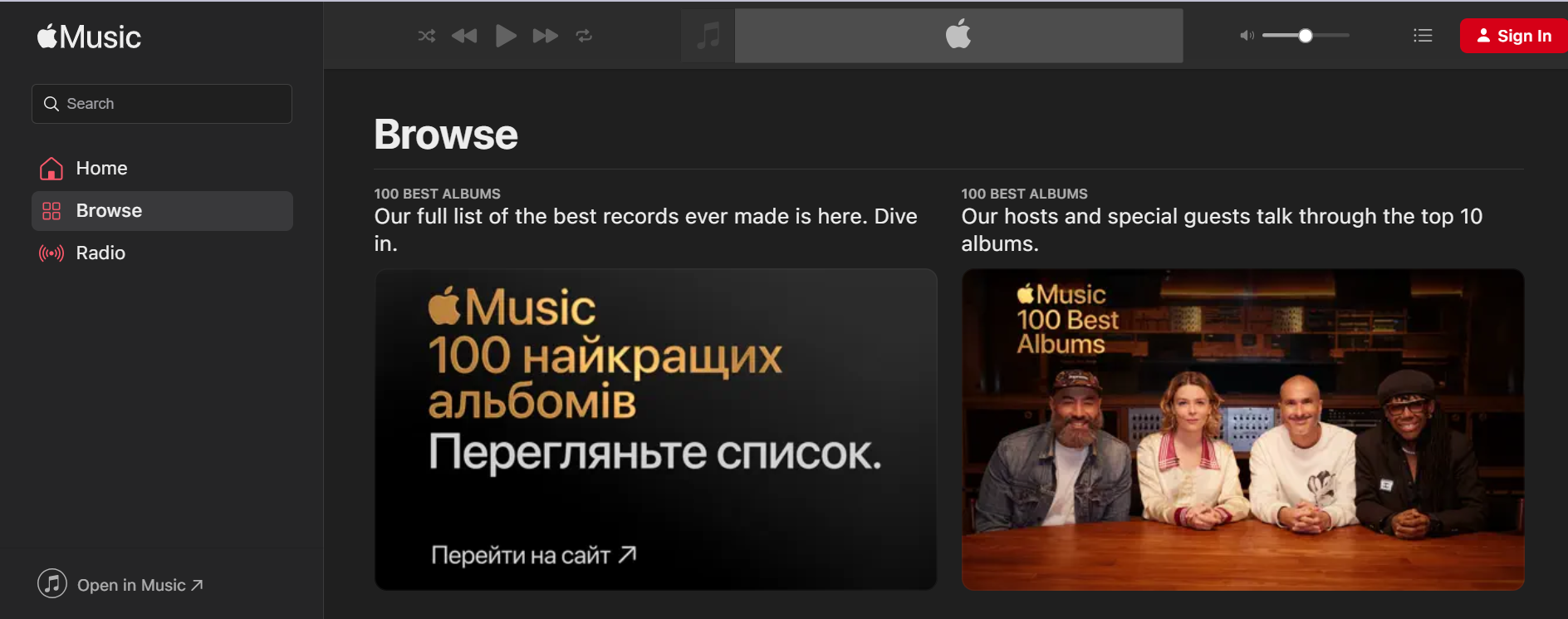


Рис.1.2 – Вебсайт Apple Music головна сторінка

### 1.3 Функціональні вимоги програмного застосунку

1.3.1 Визначення завдання і мети роботи

Основною метою є розробка та реалізація інноваційного сервісу для прослуховування аудіо контенту, що надасть користувачам доступ до великої бібліотеки музики та забезпечить зручний та персоналізований досвід прослуховування.

Ключові аспекти завдання та мети роботи включають:

* Розробка інтерфейсу для прослуховування музики в реальному часі.
* Надання можливості аутентифікації користувачів для забезпечення безпеки та персоналізації досвіду користувача.
* Розробка адміністративного інтерфейсу для керування контентом, включаючи можливість додавання та видалення музики, керування користувачами та їхніми правами.

Робота спрямована на створення високоякісного та інтуїтивно зрозумілого продукту, який задовольнить потреби користувачів у сфері музичного стрімінгу та надасть їм нові можливості для насолодження музикою.

1.3.2 Опис продукту

Застосунок це інноваційний сервіс для прослуховування аудіо контенту, який

надає користувачам можливість насолоджуватися музикою у будь-який зручний для них момент. Основна мета нашого продукту - забезпечити зручний та доступний прослуховування.

Основні функції продукту:

* Стрімінг музики: Користувачі можуть прослуховувати музику в реальному часі без необхідності завантаження її на пристрій.
* Бібліотека аудіо контенту: Наш продукт має велику бібліотеку треків, альбомів та виконавців різних жанрів, що дозволяє користувачам знайти свої улюблені композиції.
* Аутентифікація користувачів: Для забезпечення безпеки та персоналізації досвіду користувача ми надаємо можливість авторизації та аутентифікації користувачів.
* Адміністративні функції: Для адміністраторів ми надаємо інтерфейс для керування контентом, зокрема можливість додавання та видалення музики, а також керування користувачами.

1.3.3 MVP-версія продукту

Наша MVP-версія продукту включає в себе наступні функціональні можливості, необхідні для забезпечення базового функціонування платформи:

* Стрімінг музики: Користувачі можуть прослуховувати музику з вже наявної бази треків без можливості додавання власних файлів.
* Базова бібліотека аудіо контенту: У нашому MVP вже доступна обмежена кількість треків різних жанрів для прослуховування.
* Програвання та пауза відтворення: Функціонал кнопок програвання та паузи відтворення для контролю музики.
* Базовий інтерфейс користувача: Ми надаємо мінімалістичний інтерфейс для прослуховування музики та навігації між треками.
* Аутентифікація користувачів: Користувачі можуть увійти в систему з використанням основних облікових записів (наприклад, через інші свої аккаунти).

Наша MVP-версія продукту спрямована на демонстрацію основного функціоналу та перевірку його прийнятності для користувачів. Ми прагнемо залучити перші фідбеки від користувачів та вдосконалити наш продукт на основі їхніх потреб і вимог.

## РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ. АНАЛІЗ ПІДХОДІВ

### 2.1 Проектування структури бази даних

У проекті ми будемо використовувати MongoDB, що є документ-орієнтованою базою даних, для зберігання та управління інформацією про користувачами, піснями, авторами та альбомами. Нижче наведено детальний опис структури бази даних.

* Колекція "user": В цій колекції зберігається інформація про користувачів , які зарєестувались на сайті. Дивитись на Рис 2.1. Схема "UserSchema" включає такі поля:
* **name**: Імя користувача(String).
* **email**: Пошта користувача (String).
* **imabeURL**: Фото користувача (String).
* **User\_id**: ID користувача (String).
* **Email\_verfied**: Верифікація користувача (Boolean).
* **role**: Роль у системі(String).
* **auth\_time**: час авторизації (String).

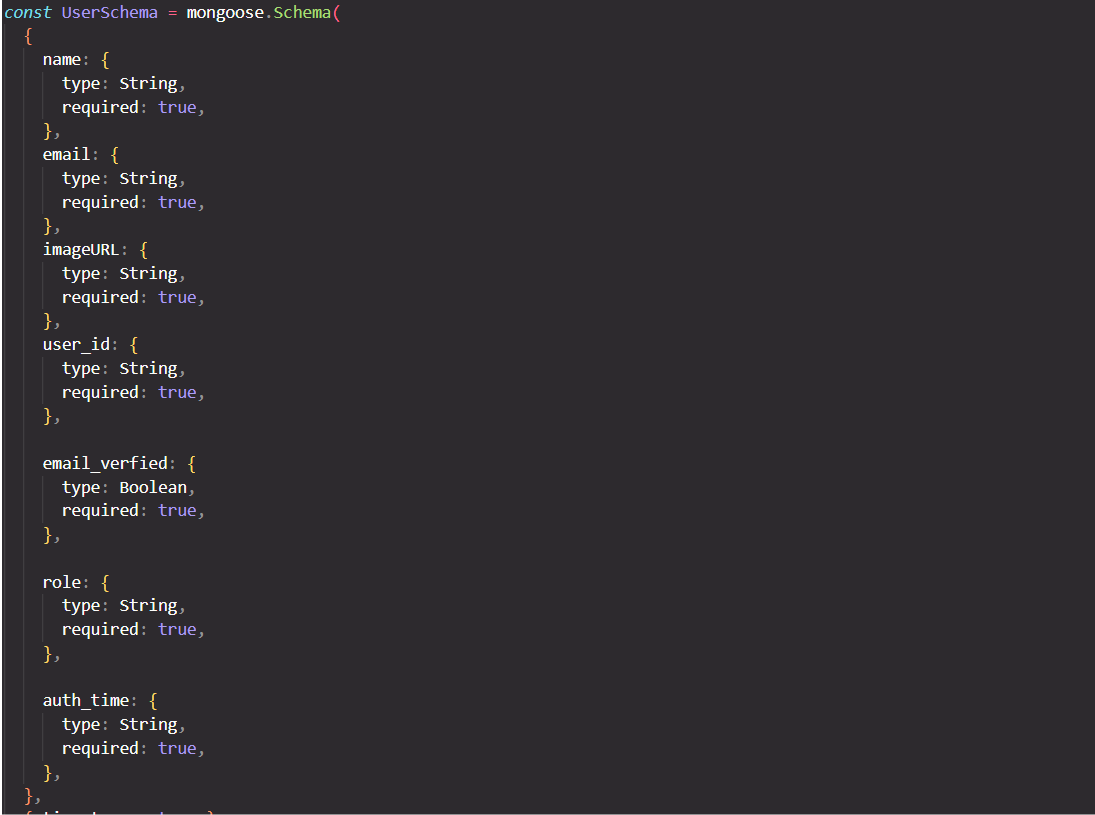


Рис. 2.1 – Колекція “User”

* Колекція "album": У цій колекції зберігається інформація про альбоми виконавців. Дивитись на Рис 2.2 .Схема "albumSchema" має такі поля:
* name: Назва альбому (String).
* imageURL: Зображення альбому (String)

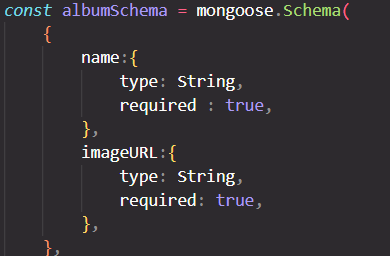


Рис 2.2 – Колекція “Album”

* Колекція "artist": У цій колекції зберігається інформація про виконавців. Дивитись на Рис 2.3 і 2.4. Схема "artistSchema" містить наступні поля:
* name: Ім'я артиста (String).
* imageUR: Зображення артиста (String)
* X: X акаунт артиста (String).
* Instagram: Instagram акаунт артиста (String)

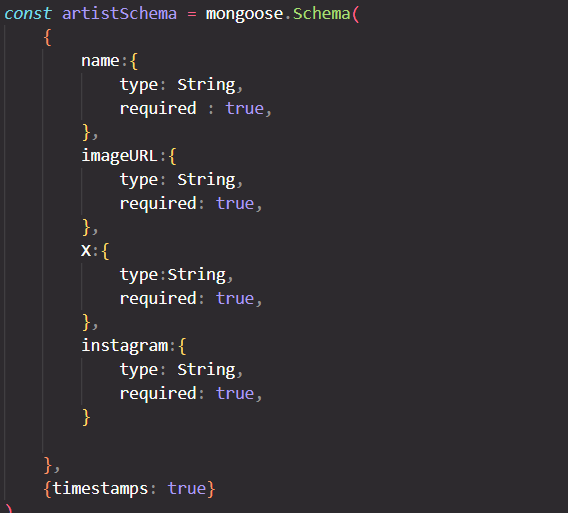


Рис 2.3 – Колекція “Artist”

* Колекція "song": У цій колекції зберігається інформація про аудіофайли, які знаходяться системі. Дивитись на Рис 2.3 і 2.4. Схема "songSchema" містить наступні поля:
* name: Назва пісні (String).
* imageURL: Зображення пісні (String).
* songURL: Сама пісня (String).
* album: Альбом пісні (String).
* artist: Виконавець пісні (String).
* Genre Жанр пісні (String).

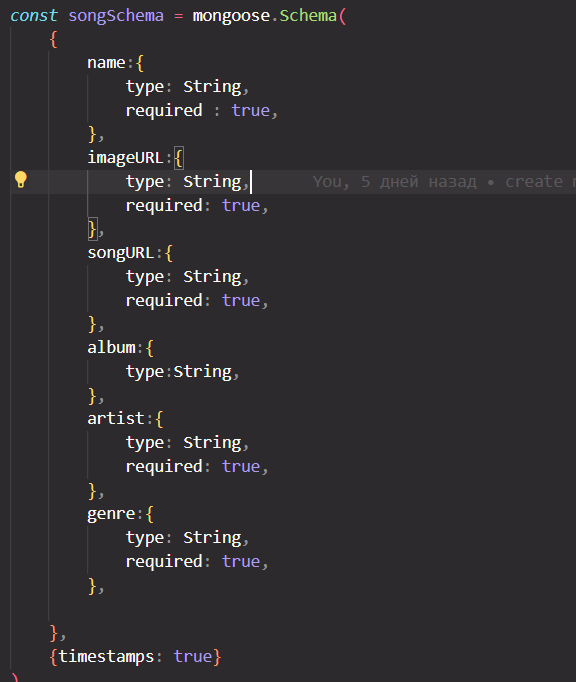


Рис. 2.4 – Колекція “Song”

### 2.2 Розробка структурно-логічної схеми бекенду та архітектурних компонентів застосунку

При проектуванні back-end та архітектури застосунку для сайту з прослуховування аудіо контенту ми будемо використовувати MongoDB, Express.js, React.js, Node.js, mongoose, firebase.

Основні компоненти back-end включають:

* Роутери (Routers): Роутери визначать маршрути та обробники для різних запитів, таких як отримання списку пісень, артистів, альбомів та виконанню операцій з ними.
* Контролери (Controllers): Контролери відповідають за логіку обробки запитів, виконуючи різні операції з базою даних та повертаючи відповіді на запити.
* Моделі (Models): Моделі представляють структуру та взаємодію з даними в базі даних MongoDB, забезпечуючи доступ до них через ORM (об'єктно-реляційне відображення).
* Middleware: Middleware використовується для обробки запитів до сервера перед тим, як вони досягнуть маршрутів та контролерів.

Бекенд буде побудований з використанням Node.js та Express.js, які дозволять нам створити API для взаємодії з базою даних MongoDB та фронтендом. та firebase який дозволить зробити систему автентифікації та зберігати файли з зображеннями та аудіофайли

2.2.1 Модуль Аутентифікації та авторизації

Firebase Authentication Service використовується для забезпечення аутентифікації користувачів та управління їхніми обліковими записами. Основні функціональні можливості включають:

* Реєстрація та вхід в систему: Користувачі можуть створювати нові облікові записи або увійти в систему за допомогою своїх існуючих облікових записів.
* Соціальна аутентифікація: Підтримка різних провайдерів, таких як Google, Facebook, Twitter, які дозволяють користувачам автентифікуватися за допомогою своїх інших облікових записів.

Firebase Authorization Middleware відповідає за перевірку та обробку токенів доступу, що надається Firebase Authentication Service. Основні функціональні можливості включають:

* Перевірка токенів доступу: Перевірка правильності та актуальності токенів доступу, отриманих від Firebase Authentication Service.
* Правила доступу: Налаштування прав доступу для різних ендпоінтів API на основі ролей користувачів та інших умов.
* Автоматичне продовження сесій: Можливість автоматичного продовження сесій користувачів за допомогою токенів оновлення, що надаються Firebase.

Цей модуль використовується для забезпечення безпеки та контролю доступу до ресурсів застосунку, забезпечуючи користувачам доступ лише до дозволених операцій.

2.2.2Модуль користувачів

Модуль користувачів відповідає закористувачів, їх аутентифікацію та авторизацію. Основні компоненти цього модуля включають:

* Реєстрація користувачів , користувачі матимуть можливість створити обліковий запис на нашому веб-сайті,за допомогою Firebase Authentication Service . Ці дані будуть перевірені та збережені у базі даних після успішної реєстрації.
* Вхід користувача , зареєстровані користувачі матимуть можливість ввійти до свого облікового запису за допомгою Firebase Authorization Middleware буде перевірено чи вони зареєстровані.

2.2.3 Модуль Аудіо

Модуль аудіо відповідає за управління та обробку аудіо контенту в системі. Основні функціональні можливості модулю аудіо включають:

* Завантаження та зберігання аудіо файлів: Модуль аудіо відповідає за приймання та зберігання аудіофайлів у визначеному форматі та якості.
* Обробка аудіо даних: Модуль забезпечує функції обробки аудіо даних, такі як конвертація форматів, обрізання, зміна гучності тощо.
* Стрімінг аудіо контенту: Модуль забезпечує можливість стрімінгу аудіо контенту користувачам, що дозволяє їм прослуховувати музику онлайн без необхідності завантаження файлів на пристрій.
* Інтеграція з іншими сервісами: Модуль може забезпечувати можливість інтеграції з іншими платформами та сервісами для отримання або обміну аудіо контентом.

Модуль аудіо відіграє ключову роль у функціоналі системи для прослуховування музики, забезпечуючи зручний та надійний доступ до аудіо контенту користувачам.

* + 1. Модуль Авторів

Модуль авторів відповідає за управління та обробку інформації про музичних виконавців в системі. Основні функціональні можливості модулю авторів включають:

* Додавання та зберігання інформації про виконавців: Модуль забезпечує можливість додавання та зберігання даних про музичних виконавців, такі як їхнє ім'я, жанр, роки активності, фотографії тощо.
* Управління метаданими: Модуль дозволяє додавати та редагувати метадані про виконавців, такі як біографічна інформація, посилання на соціальні мережі, дискографія тощо.
* Статистика та аналітика: Модуль може збирати дані про використання музичних виконавців користувачами, такі як кількість прослуховувань, популярність виконавців тощо, для подальшого аналізу та використання.

Модуль авторів допомагає забезпечити повноту та різноманітність музичного контенту в системі, дозволяючи користувачам отримувати доступ до інформації про улюблених виконавців та їхню музику.

2.2.5 Модуль Альбомів

Модуль альбомів відповідає за управління та обробку інформації про музичні альбоми в системі. Основні функціональні можливості модулю альбомів включають:

* Додавання та зберігання інформації про альбоми: Модуль забезпечує можливість додавання та зберігання даних про музичні альбоми, такі як назва, рік випуску, жанр, обкладинка тощо.
* Управління метаданими: Модуль дозволяє додавати та редагувати метадані про альбоми, такі як список треків, список виконавців, опис, посилання на обкладинку тощо.

Модуль альбомів допомагає забезпечити систему повноцінним та організованим способом представлення музичного контенту, дозволяючи користувачам знаходити та слухати аудіо матеріали у відповідно до їхніх вподобань.

2.2.6 Модуль взаємодії з Firebase

Модуль взаємодії з Firebase відповідає за інтеграцію та взаємодію з різними сервісами, які надає Firebase, для забезпечення різноманітних функцій та можливостей в додатку. Основні функціональні можливості модулю взаємодії з Firebase включають:

* Аутентифікація користувачів: Модуль взаємодії з Firebase використовує сервіс Firebase Authentication для забезпечення безпечної та надійної аутентифікації користувачів. Він дозволяє користувачам входити в додаток за допомогою різних методів, таких як електронна пошта та пароль, а також соціальні мережі.
* Зберігання та синхронізація даних: Firebase Realtime Database або Firestore можуть використовуватися для зберігання та синхронізації даних між користувачами та пристроями. Це дозволяє забезпечити доступ до оновленої інформації в реальному часі.
* Зберігання файлів: Firebase Storage використовується для зберігання медіафайлів, таких як зображення обкладинок альбомів, аудіофайли музики тощо.
* Аналітика та звіти: Firebase Analytics дозволяє збирати дані про використання додатку користувачами, а Firebase Crashlytics - виявляти та відстежувати помилки в додатку.

Модуль взаємодії з Firebase забезпечує потужні можливості для розробки додатків з використанням хмарних сервісів, що дозволяє розширити функціонал додатку та підвищити його продуктивність і безпеку.

### 2.3 UX/UI дизайн програмного застосунку

Щоб програмний застосунок для прослуховування музики був успішним та зручним для користувачів, він має мати такі ключові елементи UX/UI дизайну:

1. Інтуїтивний і простий інтерфейс користувача: Користувачі мають легко розуміти, як користуватися застосунком з першого погляду. Меню та елементи управління повинні бути зрозумілими та доступними.
2. Зручний пошук та фільтрація: Додаток повинен мати можливість швидкого пошуку музики за назвою, виконавцем, жанром тощо. Фільтри можуть допомогти користувачам точніше знаходити потрібну музику.
3. Персоналізація рекомендацій: Система рекомендацій може пропонувати користувачам музику, яка відповідає їхнім вподобанням на основі їхньої історії прослуховування.
4. Програвання в реальному часі: Користувачі мають бути здатні легко програвати музику та керувати відтворенням зручним способом, включаючи кнопки програвання, паузи, перемотування та регулювання гучності.
5. Створення та керування плейлистами: Додаток повинен дозволяти користувачам створювати власні плейлисти, додавати та видаляти треки, редагувати їхні назви та порядок відтворення.
6. Синхронізація між пристроями: Якщо додаток підтримується на декількох пристроях, важливо мати можливість синхронізувати дані, так щоб користувач міг продовжити прослуховування там, де він завершив на іншому пристрої.
7. Графічний дизайн та обкладинки альбомів: Графіка та обкладинки альбомів мають бути привабливими та залучати увагу користувачів, а також відображати настрій музики.
8. Інтеграція з соціальними мережами: Можливість ділитися улюбленою музикою або плейлистами з друзями через соціальні мережі може збільшити вовлеченість користувачів.

Успішний UX/UI дизайн дозволить зробити додаток зручним та привабливим для користувачів, що сприятиме його популярності та успіху на ринку.

2.3.1 Персоналії

У UX дизайні програмного застосунку для прослуховування музики, персоналії є ключовим елементом, оскільки вони допомагають зрозуміти потреби та очікування користувачів і спрямовують розробників на створення ефективного та задовільного досвіду користувача. Ось кілька типових персоналій та їх ролі в UX дизайні програмного застосунку для прослуховування музики:

1. Марія:

Роль: Марія - активний слухач, який любить відкривати нову музику та досліджувати різні жанри.

Очікування: Простий та зручний інтерфейс для швидкого пошуку та прослуховування музики різних жанрів. Рекомендації на основі попередніх прослуховувань для відкриття нових виконавців і альбомів.

1. Петро:

Роль: Петро - користувач, який слухає популярну музику та великими фанатом конкретних виконавців або жанрів.

Очікування: Можливість створення власних плейлистів з улюбленою музикою. Опції підписки на артистів для отримання сповіщень про нові релізи.

1. Користувач Вікторія:

Роль: Користувач Вікторія - користувач, який слухає музику з різних жанрів, не має конкретних уподобань.

Очікування: Функція рекомендацій на основі особистих уподобань та прослуховувань. Простий інтерфейс з можливістю випадкового відтворення музики.

1. Олексій:

Роль: Олексій - користувач, який веде власний музичний блог або подкаст та регулярно додає власний контент.

Очікування: Зручні інструменти для завантаження та публікації аудіофайлів. Функція взаємодії зі слухачами, така як коментарі або можливість створення власних плейлистів.

Ці персоналії допомагають UX дизайнерам розуміти потреби та очікування різних типів користувачів та створювати інтерфейс, який відповідає їхнім потребам і вподобанням.

2.3.2 Wireframes

Wireframes - макети, які потрібні для проектування інтерфейсу користувача та дизайну. Вони використовуються для візуалізації розташування компонентів на етапі розробки продукту. Для створення Wireframes було використано застосунок Figma. Figma інструмент для створення дизайну, яка дозволяє створювати макети, прототипи, редагувати зображеня та інші дизайн-елементи в спільній робочій області, доступній для співпраці.

Загалом, використання Figma створювати wireframes для програмного застосунку, що сприяє якості та ефективності дизайн-процесу.

Ось приклади wireframes для програмного застосунку "Інформація про їжу":

* Інтерфейс головної сторінки (Рис 2.5)
* Інтерфейс панелі інструментів(Рис 2.6)
* Інтерфейс панелі управління користувачами (Рис 2.7)
* Інтерфейс списку пісень (Рис 2.8)
* Інтерфейс добавлення Артиста (Рис. 2.9)
* Інтерфейс входу в акаунт (Рис. 2.10)

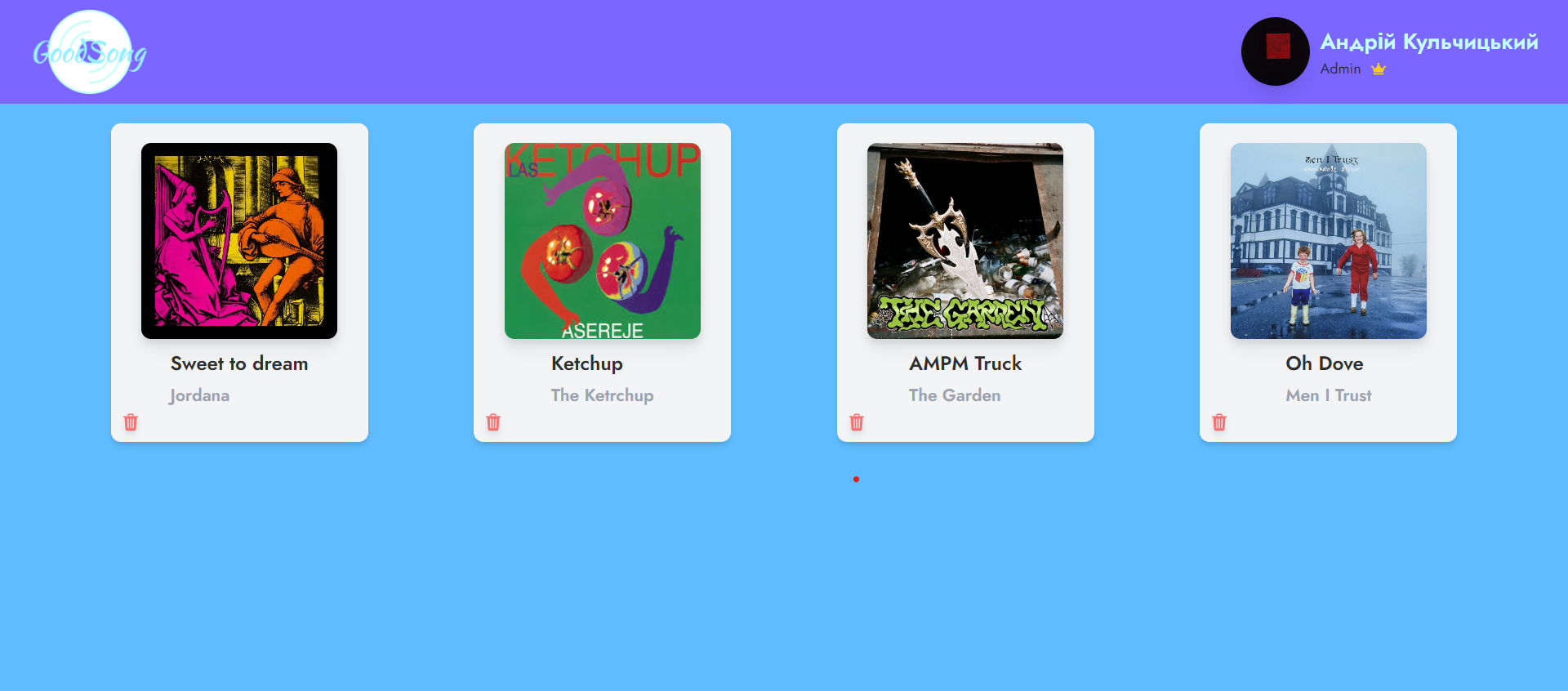


Рис. 2.5 – Макет головного меню

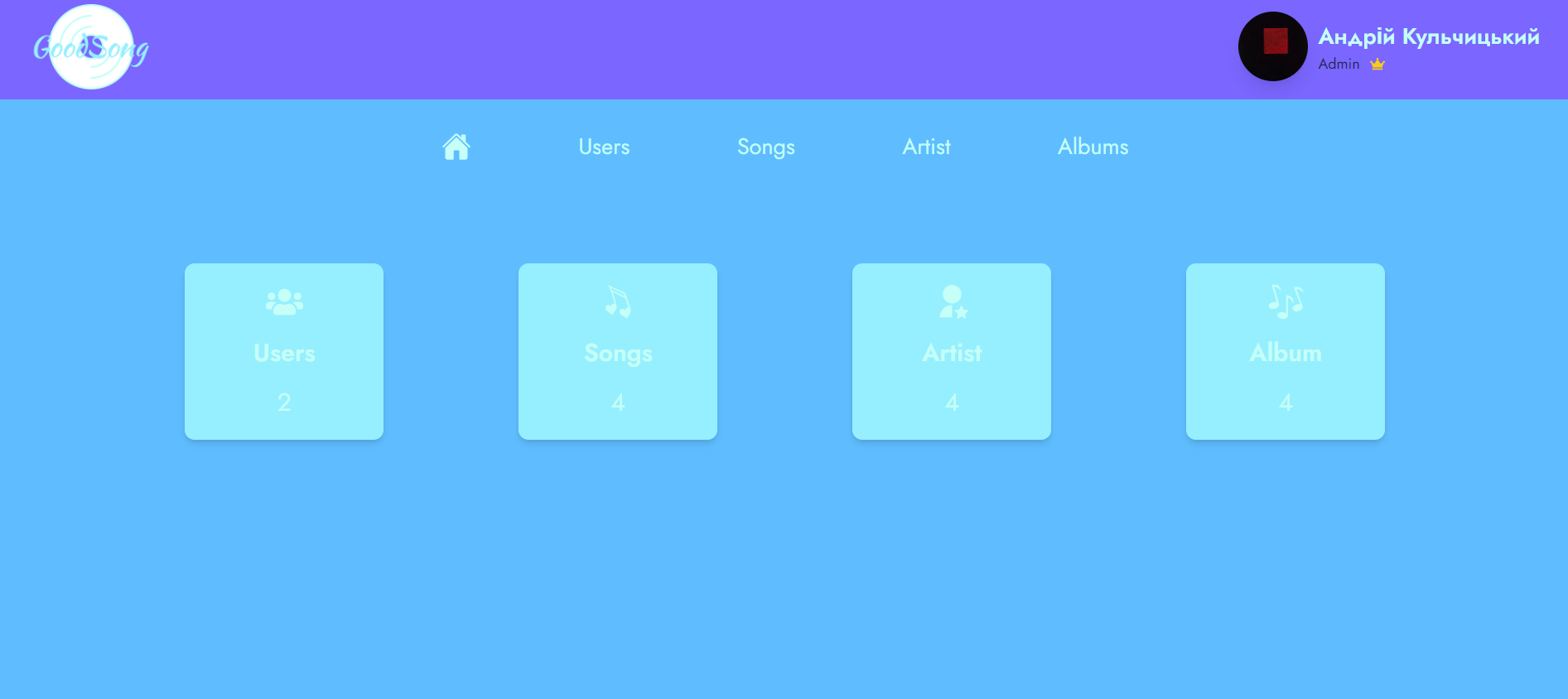


Рис. 2.6 – Макет панелі інструментів Адміна

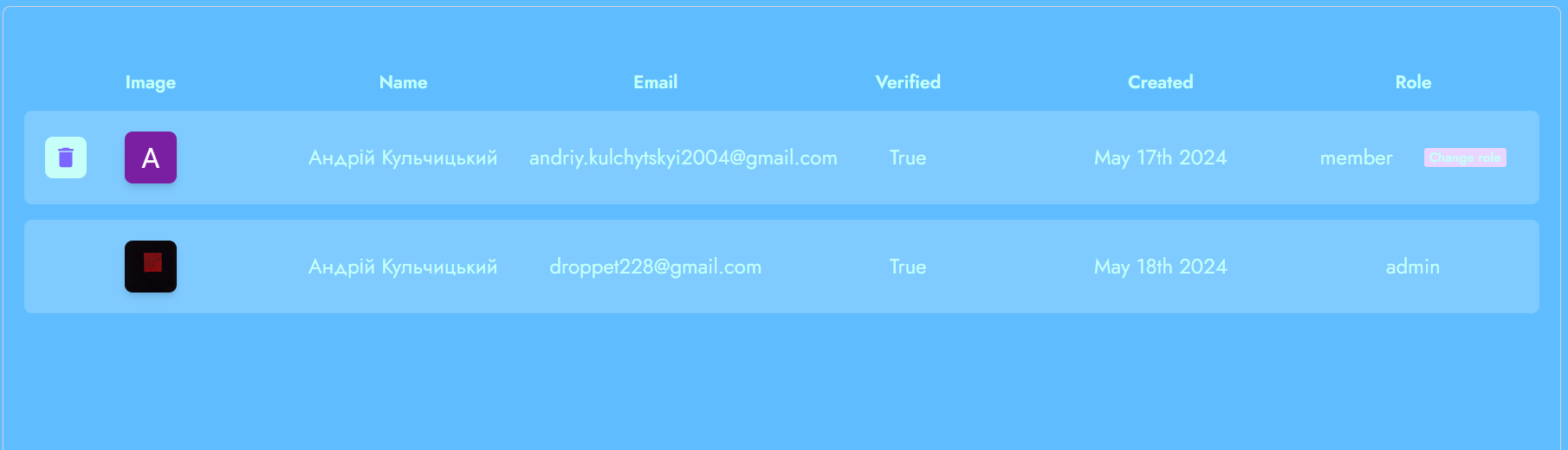


Рис. 2.7 – Макет панелі керування користувачами

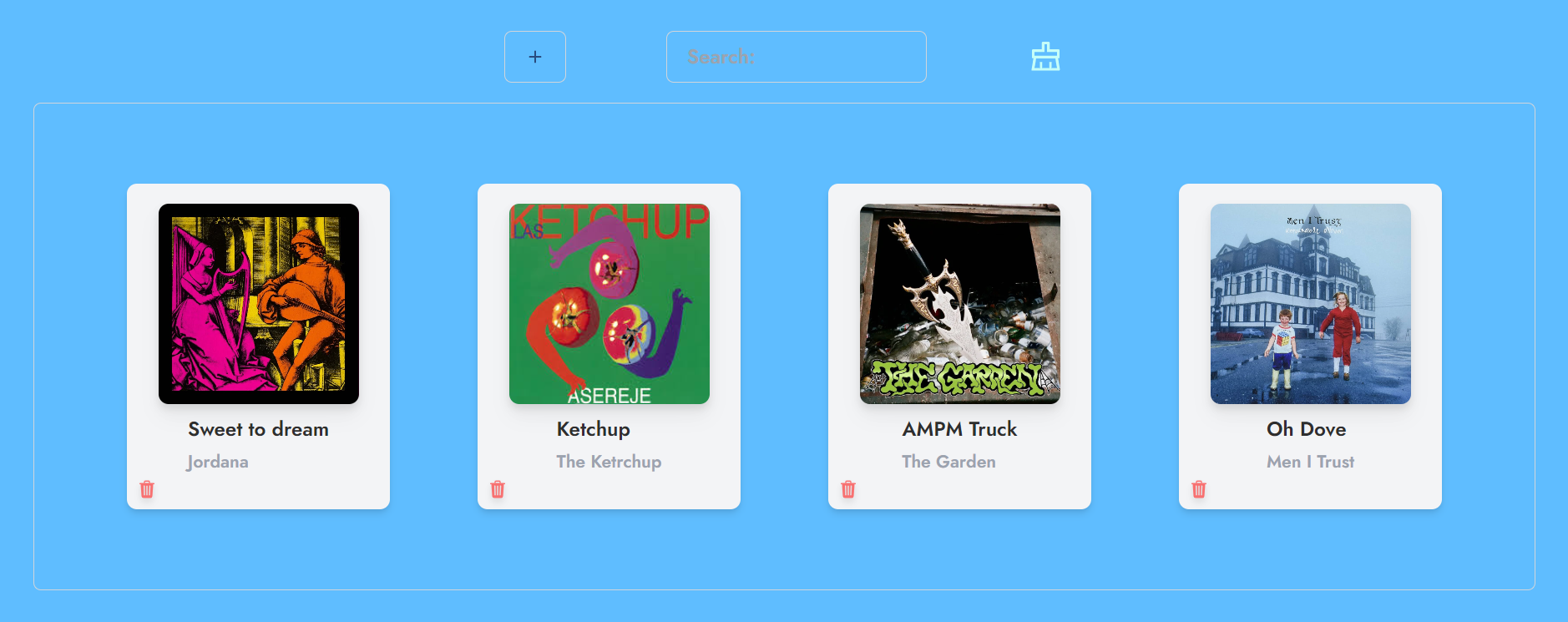


Рис. 2.8 – Макет списку пісень

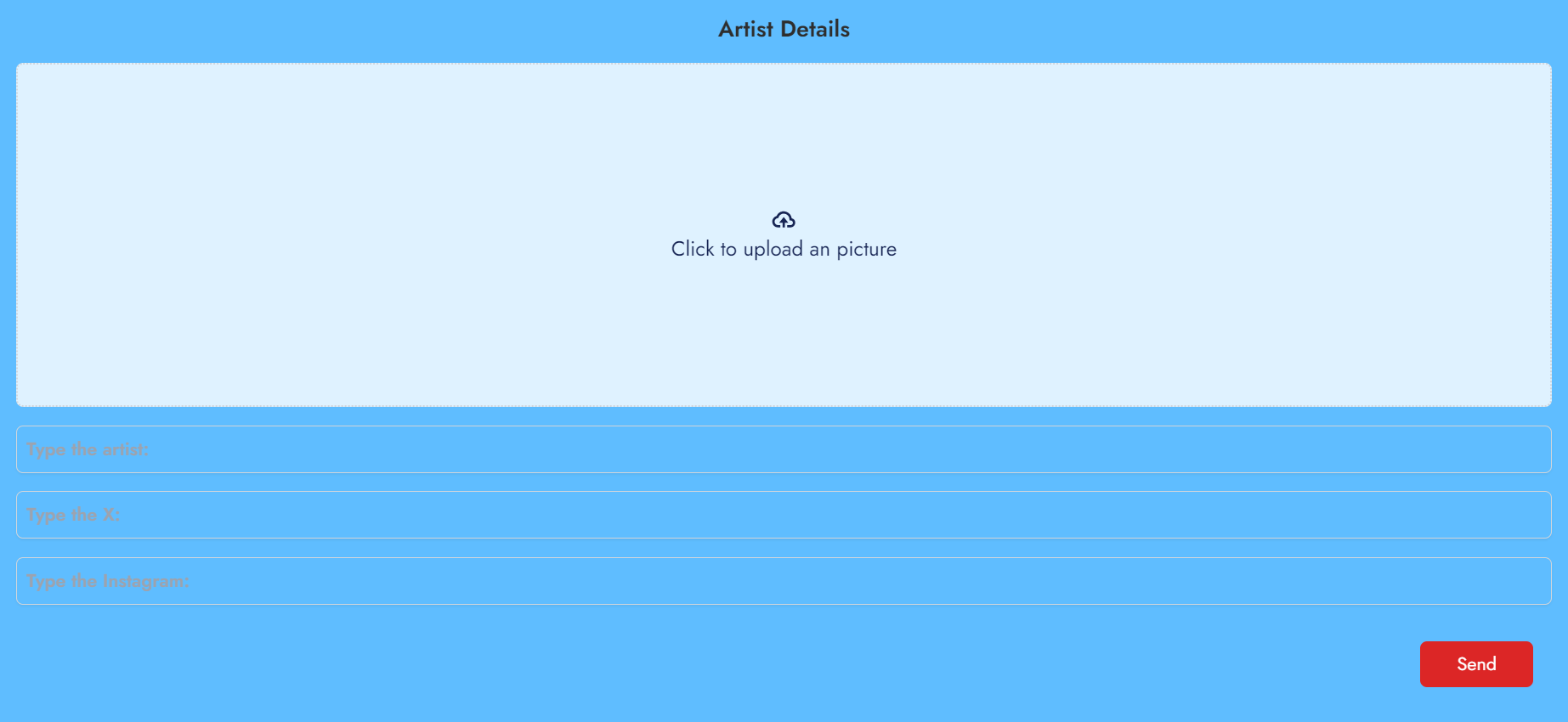


Рис. 2.9 – Макет додавання артиста



Рис. 2.10 – Макет вхід в акаунт

## РОЗДІЛ 3 Особливості кодової імплементації програмного застосунку.

### 3.1 Back-end розробляємої платформи.

3.1.1 Особливості кодової імплементації модуля безпеки, аутентифікації та авторизації

У даному розділі розглянемо особливості кодової імплементації модуля безпеки, аутентифікації та авторизації для нашого сервісу прослуховування аудіо контенту. Основна мета цього модуля – забезпечити безпечний доступ користувачів до сервісу та надати адміністраторам можливість керувати контентом і користувачами.

На Рис.(3.1) фрагмент коду для перевірки аутентифікації за допомогою firebase. Першим кроком у процесі аутентифікації є перевірка наявності заголовка authorization у запиті. Якщо заголовок відсутній, сервер повертає відповідь з кодом 500 та повідомленням "Invalid Token". Токен витягується з заголовка авторизації та верифікується за допомогою методу verifyIdToken сервісу Firebase Admin SDK. Якщо верифікація не вдається, сервер повертає відповідь з кодом 500 та повідомленням "Not Authorized". Після успішної верифікації токена виконується перевірка наявності користувача у базі даних за його user\_id. Якщо користувач не існує, викликається функція newUserData для створення нового запису користувача. Якщо користувач існує, викликається функція updateUserData для оновлення даних користувача. У разі виникнення помилки під час верифікації токена або перевірки існування користувача, сервер повертає відповідь з кодом 500 та повідомленням про помилку.

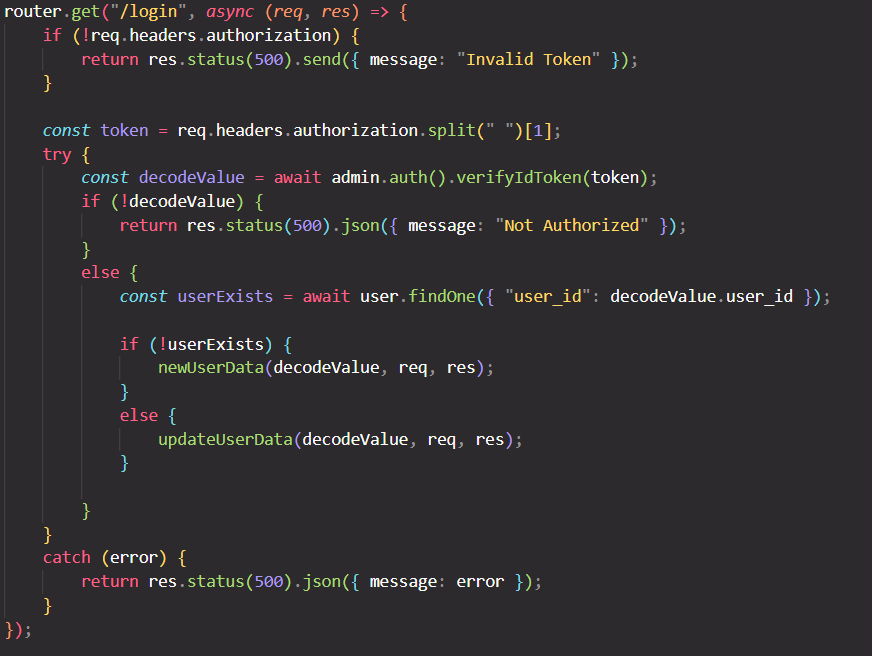


Рис.3.1 – реалізація модуля аутентифікації

3.1.2 Особливості кодової імплементації модуля Аудіо

У цьому розділі детально розглянемо, як виконується імплементація модуля аудіо на стороні back-end, для того щоб користувач міг без проблем завантажувати та слухати аудіо файли. з використанням технологій Node.js, Express, MongoDB та Mongoose. Цей модуль відповідає за управління музичними треками, включаючи їх створення, оновлення, видалення та отримання інформації про них.

Використані технології

* Node.js - платформа, що дозволяє виконувати JavaScript на сервері. Node.js використовується для створення серверної частини додатку.
* Express - веб-фреймворк для Node.js, який спрощує створення веб-додатків та API. Він забезпечує зручну маршрутизацію та обробку запитів.
* MongoDB - документно-орієнтована база даних NoSQL, яка зберігає дані у форматі JSON-подібних документів.
* Axios - бібліотека для здійснення HTTP-запитів з клієнтської сторони. Вона використовується для комунікації між фронтендом і бекендом.

Архітектура модуля

* Роутери (Routers) - відповідають за маршрутизацію HTTP-запитів до відповідних контролерів. Кожен роутер обробляє певні запити, наприклад, для отримання всіх пісень, отримання конкретної пісні за ID, додавання нової пісні, оновлення існуючої пісні та видалення пісні.
* Моделі (Models) - визначають структуру документів у MongoDB. Модель пісні (song) описує поля, які містить кожен документ пісні, такі як назва пісні, URL зображення, URL пісні, альбом, виконавець та жанр.
* Контролери (Controllers) - містять логіку для обробки запитів та взаємодії з базою даних. Контролери використовують моделі для збереження та отримання даних з MongoDB.

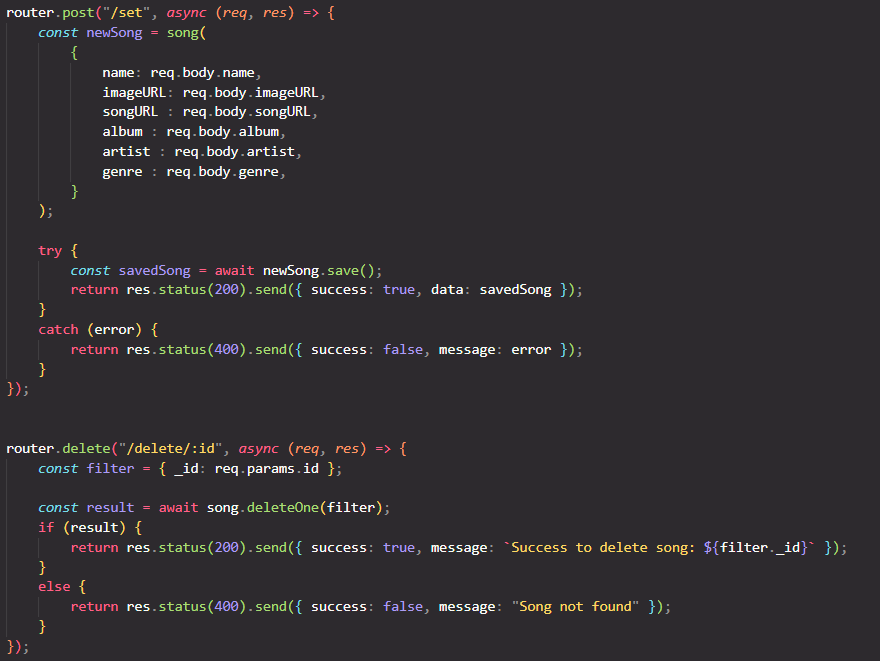
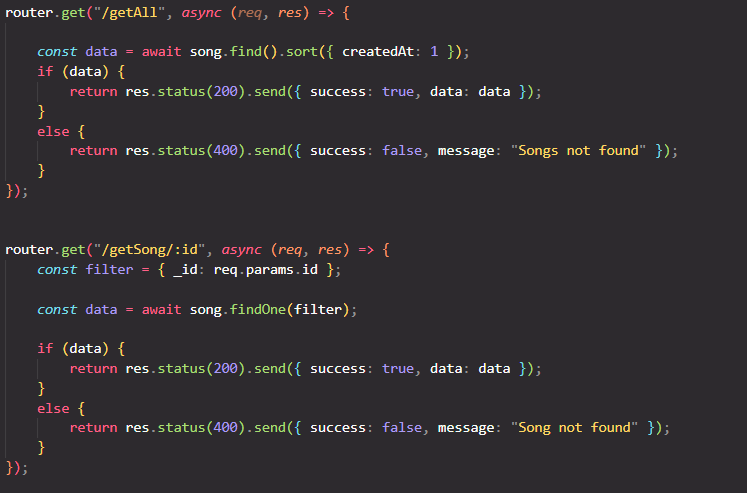


Рис.3.2 та 3.3 - songRoute

Відповідно до Рис.(3.2 , 3.3), тут зображений роутер в якому прописані запити для пісень до бази данних

.

3.1.3 Особливості кодової імплементації модуля api

Імплементація модуля API для сервісу прослуховування аудіо контенту включає використання різних інструментів та технологій для забезпечення CRUD операцій з аудіо контентом. Процес складається з кількох етапів: створення моделей для зберігання даних, створення роутерів для обробки HTTP-запитів, використання Axios для взаємодії з API на фронтенді та зберігання даних у базі даних MongoDB за допомогою Node.js, Mongoose та Express.

Використані інструменти та технології:

* Node.js: Серверне середовище, що дозволяє використовувати JavaScript для створення бекенду.
* Express: Легкий та гнучкий фреймворк для Node.js, який забезпечує набір функцій для створення веб- та мобільних додатків.
* Mongoose: Бібліотека для роботи з MongoDB, яка забезпечує зручний спосіб взаємодії з базою даних.
* MongoDB: NoSQL база даних, що зберігає дані у форматі документів.
* Axios: Бібліотека для виконання HTTP-запитів з фронтенду, яка підтримує Promise API.

Процес імплементації:

1. Створення моделей Моделі визначають структуру даних, що зберігаються у базі даних MongoDB.
2. Роутери обробляють HTTP-запити, які надходять на сервер, і викликають відповідні функції для виконання дій над даними.
3. 3. Використання Axios на фронтенді
4. Axios використовується для відправки HTTP-запитів до сервера з фронтенду.

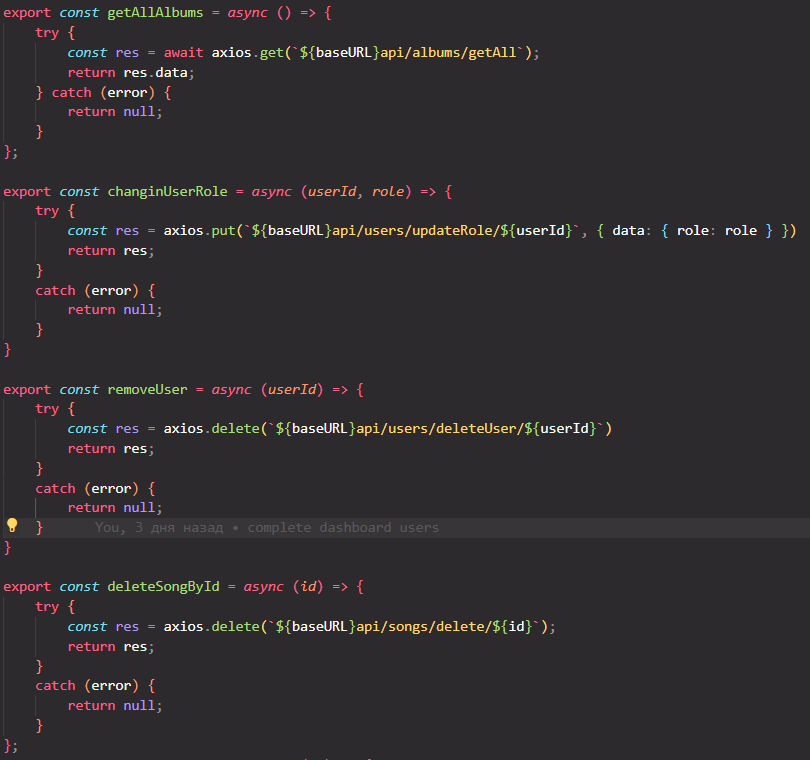


Рис.3.4 та 3.5 - функціонал модуля Api

1. Зберігання даних у MongoDB

Mongoose використовується для зберігання та отримання даних з MongoDB.

Імплементація модуля API включає створення моделей для структурування даних, роутерів для обробки HTTP-запитів, використання Axios для взаємодії з API на фронтенді та зберігання даних у MongoDB за допомогою Mongoose. Цей підхід забезпечує зручний та ефективний спосіб управління аудіо контентом у нашому сервісі.

### 

### 3.2 Front-end розробляємої платформи

Front-end розробляємої платформи відображатиме інтерфейс, з яким користувачі будуть взаємодіяти. Основні завдання front-end розробки включають:

* Дизайн інтерфейсу: Проектування дизайну та, розміщення елементів та оформлення інтерфейсу враховуючи максимальну зручність та естетичність вигляду
* Взаємодія з користувачем: Максимально комфортна та зрозуміла реалізація відповідно до UX/UI стандартів для забезпечення чудового досвіду користуванням
* Взаємодія з сервером для отримання даних для їх відображення а також надсилання данник користувача на сервер

Для розробки front-end частини платформи використовувалось різні технології та інструменти, такі як HTML, CSS, JavaScript або їх фреймворки та бібліотеки (React), а також додаткові бібліотеки (tailwiind, famer-motion) для взаємодії з сервером, маршрутизації, управління станом тощо.

3.2.1 Особливості імплементації модуля відображення контенту

Модуль відображення контенту у нашому сервісі реалізований за допомогою компоненту Card який включає у себе функціонал відображення зображення, та назви елементу, також дає можливість видалити елемент, відпрвляжчи запит через API до сервера та отримаючи відповідь і одразу ж відображаючи її

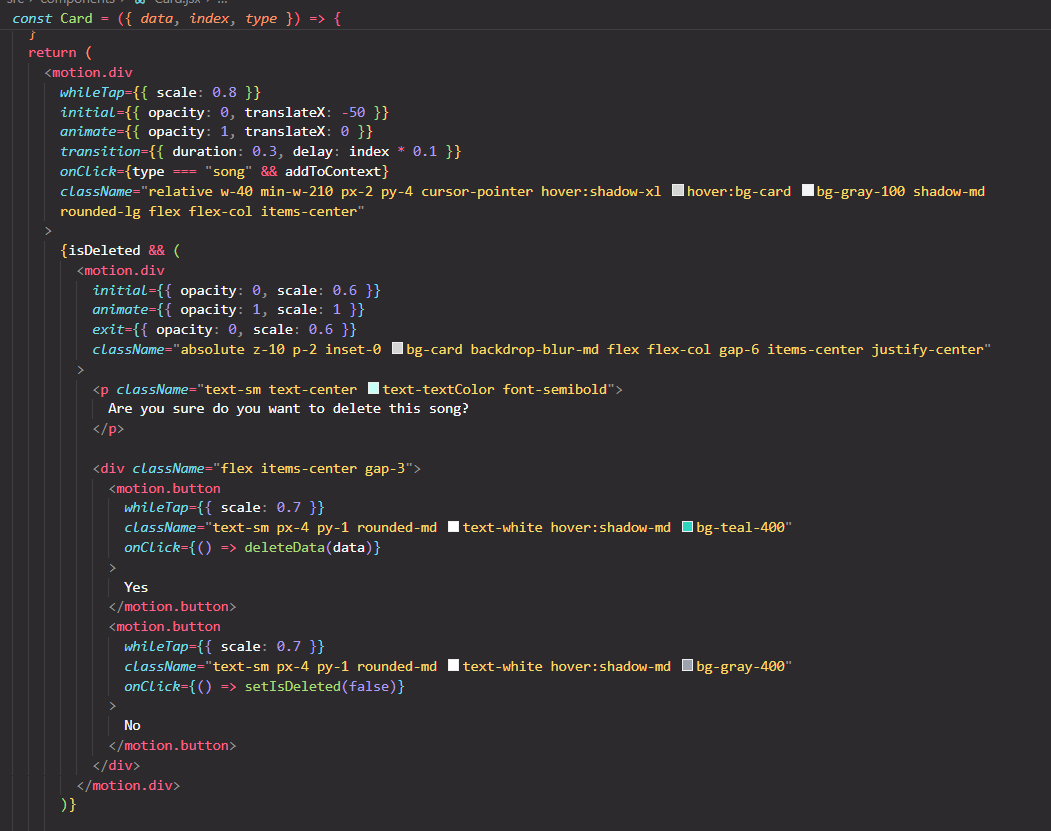


Рис.3.6 – фрагмент коду Card

Були використані такі інструменти як framer-motion для забезпечення легкої та плавної анімації

Цей модуль відображає данні на сторінці та дозволяє взаємодіяти з ними

3.2.2 Особливості кодової імплементації модуля плеєра

У плеєрі музики реалізовано низку функціональностей для відтворення аудіо контенту. Він складається з таких ключових елементів:

* Відтворення аудіо: Використовується компонент AudioPlayer з бібліотеки react-h5-audio-player для відтворення аудіофайлів. Цей компонент забезпечує основні контроли відтворення, такі як play/pause, попередній та наступний трек.
* Список пісень: При кліці на іконку плейлисту відображається компонент PlayListCard, який містить список всіх доступних пісень для відтворення. Кожен елемент списку містить інформацію про пісню, таку як назва, виконавець та альбом.
* Керування плеєром: Реалізовано можливість перемикання між піснями за допомогою кнопок попереднього та наступного треку. Також є кнопка закриття плеєра, яка призначена для закриття вікна плеєра.
* Інтеграція з Redux: Стан плеєра керується через контекст Redux. Це дозволяє зберігати та оновлювати поточну пісню, індекс пісні, стан відтворення тощо.
* Завантаження пісень: При монтуванні компонента перевіряється наявність пісень у стані додатку. Якщо пісні відсутні, вони завантажуються з сервера за допомогою функції getAllSongs.
* У цілому, плеєр музики забезпечує зручний та інтуїтивно зрозумілий інтерфейс для відтворення аудіо контенту та надає користувачам можливість керувати музикою з легкістю.

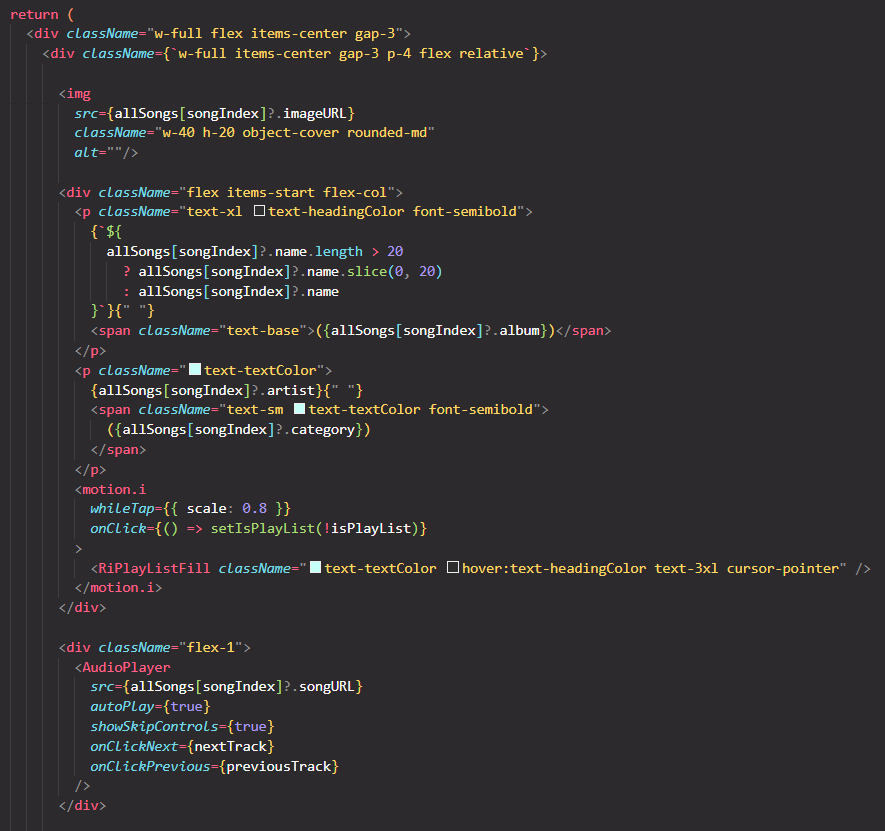


Рис.3.7 та 3.8 – код плеєру

Отже, цей модуль дозволяє користувачам легко керувати музикою, перемикати на попередню та наступну композицію та вибирати її з плейлисту.

3.2.3 Особливості кодової імплементації модуля додавання контенту

У цьому модулі реалізовано можливість додавання нових пісень, виконавців та альбомів до платформи. Основні етапи та функціональність компонента включають:

Завантаження файлів:

* Використовується компонент FileUploader, який надає можливість вибрати та завантажити файли зображень або аудіофайли.
* Використано функціонал Firebase для завантаження файлів у хмарне сховище. Файли обробляються за допомогою Firebase Storage API.

Управління станом:

* Використано хуки стану React, такі як useState та useEffect, для керування станом компонентів.
* Кожна дія, така як завантаження файлу, супроводжується станом завантаження, що відображається користувачу.

Взаємодія з сервером:

* Використовується асинхронні запити до сервера з допомогою функцій getAllArtists, getAllAlbums, saveNewSong, saveNewArtist, saveNewAlbum, щоб отримати дані про виконавців, альбоми, а також зберегти нові пісні, виконавців та альбоми.
* Дані з сервера оновлюються за допомогою диспетчера Redux.

Інтерфейс користувача:

* Для комфортнішого вибору кнопки фільтрації для вибору виконавців, альбомів та жанрів.
* Зображення завантажується для кожного виконавця та альбому, що дозволяє користувачу встановлювати візуальне представлення.

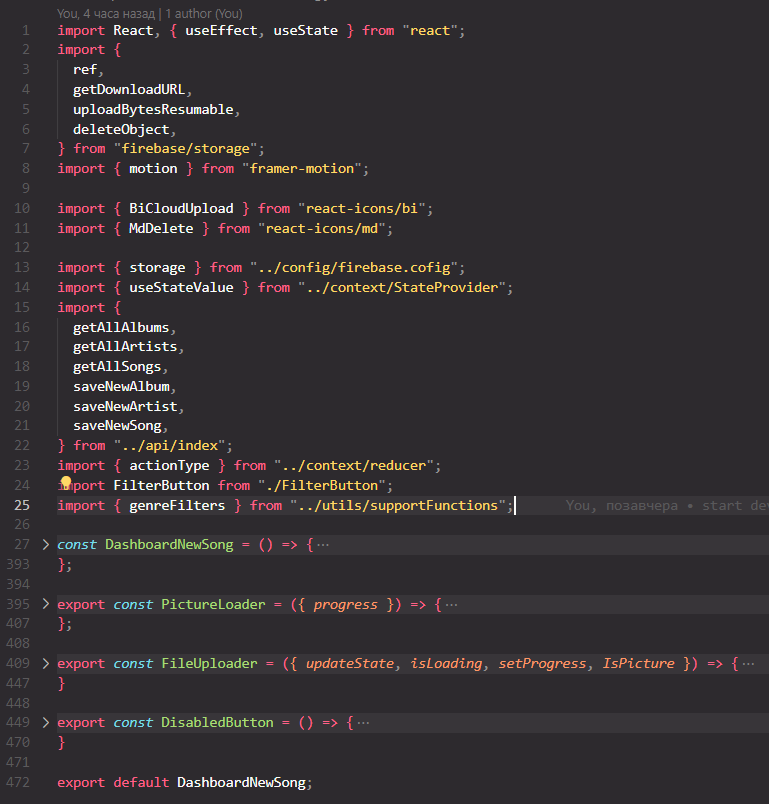


Рис.3.9 -Форма для додавання контенту на сайт

* У цілому, компонент цей надає можливість додавати новий контент до платформи зручним та інтуїтивно зрозумілим способом, забезпечуючи користувачам зручний інтерфейс та швидку взаємодію з системою.

## РОЗДІЛ 4 ОГЛЯД ІМПЛЕМЕНТОВАНИХ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ІНФОРМАЦІЙНОЇ СИСТЕМИ.

4.1 Функціонал Адміністратора

Функціонал адміністратора зазвичай включає в себе широкий спектр можливостей для управління контентом та налаштуваннями платформи. Ось деякі типові функції адміністратора:

Модуль DashboardNewSong() містить у собі 3 форми які дозволяють додавати новий контент на сайт, швидко та зручно оновлювати списки пісень та їх виконавців:

* Додавання нових пісень, виконавців та альбомів до бази даних платформи.
* Редагування та видалення існуючого музичного контенту.
* Можливість створення та редагування плейлистів для користувачів.
* Управління зображеннями та аудіофайлами:
* Завантаження, видалення та керування зображеннями, що використовуються для пісень, альбомів та виконавців.
* Завантаження та керування аудіофайлами, які використовуються для музичного контенту.

Модуль DashboardUser() дає доступ до списку всіх користувачів сайту які були зареєстровані дозволяючи адміністрації:

* Додавання, видалення та редагування облікових записів користувачів.
* Управління правами доступу користувачів, таких як ролі (наприклад, адміністратор, модератор, звичайний користувач).

4.2 Функціонал звичайного користувача

Функціонал звичайного користувача складається з ряду модулів, які надають можливість користувачам використовувати різноманітні функції та можливості системи.

Функціонал звичайного користувача може включати такі можливості:

Перегляд музичного контенту:

* Користувач може переглядати доступні пісні, альбоми та виконавців.
* Є можливість фільтрувати музичний контент за різними критеріями, такими як жанр, виконавець, альбом і т.д.

Прослуховування музики:

* Користувач може відтворювати пісні з платформи за допомогою вбудованого аудіо плеєра.
* Можливість перемикання між треками, регулювання гучності та інші стандартні функції плеєра.

Перегляд інформації про виконавців та альбоми:

* Користувач може дізнатися більше про виконавців та альбоми, переглянувши їх інформаційні сторінки, включаючи біографії, фотографії, рейтинги тощо.

Загалом, функціонал звичайного користувача спрямований на забезпечення зручного та цікавого музичного досвіду, надаючи різноманітні можливості для пошуку, прослуховування та взаємодії з музичним контентом.

## ВИСНОВОК

Під час виконання завдання було спроектовано та реалізовано програмний продукт – сервіс для прослуховування аудіо контенту. Щоб досягти цього було використано чимало інструментів. Спочатку було проведено аналіз сучасного ринку, його вимоги та тенденції, що дозволило скласти чіткий список вимог до продукту. Наступним кроком було створено макет продукту з впровадженням стандартів UX/UI для зручності використання.

Другий етап це редагування back-end додатку. Створивши базу даних яка зберігала в собі данні про користувачів, артистів, альбомів та музики. Також був розроблений функціонал для обробки запитів до бази даних та поверненню відповіді.

У третьому етапі було спроектовано front-end візуальна частина якого відповідала макету щоб взаємодія користувача була максимально комфортна та легка.

Результатом стала програма яка має багатофункціональність та зручність у використанні, який відповідає сучасним стандартам та закриває потреби користувачів.

Загалом, функціонал адміністратора та звичайного користувача включає в себе широкий спектр можливостей для управління та використання музичної платформи.

Адміністратор може ефективно керувати контентом та користувачами, додавати та редагувати музичний контент, керувати зображеннями та аудіофайлами, а також надавати доступ та налаштовувати ролі користувачів. Звичайний користувач, у свою чергу, може насолоджуватися доступним музичним контентом, прослуховувати пісні, дізнаватися більше про виконавців та альбоми, а також використовувати функції пошуку та фільтрації для зручного перегляду.

Загалом, ці два типи користувачів спільно створюють живе та динамічне співтовариство на музичній платформі, яке дозволяє насолоджуватися музикою та розвивати музичну культуру.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MongoDB [Електронний ресурс] - <https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/>
2. Tailwindcss [Електронний ресурс] - <https://tailwindcss.com/>
3. Axios [Електронний ресурс] - <https://axios-http.com/docs/intro>
4. Firebase [Електронний ресурс] - <https://firebase.google.com/>
5. React icons [Електронний ресурс] - <https://react-icons.github.io/react-icons/>
6. React [Електронний ресурс] - <https://react.dev>
7. Spotify [Електронний ресурс] - [https://open.spotify.com/](https://open.spotify.com/%20)
8. Apple Music [Електронний ресурс] – <https://music.apple.com/ua/browse>
9. NodeJS [Електронний ресурс] - <https://www.w3schools.com/nodejs/nodejs_intro.asp>
10. Express [Електронний ресурс] - <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs>

## ДОДАТОК А

**song.js –** запити для взаємодії з музикою**:**

const router = require("express").Router();

const song = require("../models/song");

router.get("/getAll", async (req, res) => {

const data = await song.find().sort({ createdAt: 1 });

if (data) {

return res.status(200).send({ success: true, data: data });

}

else {

return res.status(400).send({ success: false, message: "Songs not found" });

}

});

router.get("/getSong/:id", async (req, res) => {

const filter = { \_id: req.params.id };

const data = await song.findOne(filter);

if (data) {

return res.status(200).send({ success: true, data: data });

}

else {

return res.status(400).send({ success: false, message: "Song not found" });

}

});

router.post("/set", async (req, res) => {

const newSong = song(

{

name: req.body.name,

imageURL: req.body.imageURL,

songURL : req.body.songURL,

album : req.body.album,

artist : req.body.artist,

genre : req.body.genre,

}

);

try {

const savedSong = await newSong.save();

return res.status(200).send({ success: true, data: savedSong });

}

catch (error) {

return res.status(400).send({ success: false, message: error });

}

});

router.delete("/delete/:id", async (req, res) => {

const filter = { \_id: req.params.id };

const result = await song.deleteOne(filter);

if (result) {

return res.status(200).send({ success: true, message: `Success to delete song: ${filter.\_id}` });

}

else {

return res.status(400).send({ success: false, message: "Song not found" });

}

});

router.put("/update/:id", async (req, res) =>{

const filter = { \_id: req.params.id };

const options = {

upsert : true,

new: true,

};

try {

const result = await song.findOneAndUpdate(filter,

{

name: req.body.name,

imageURL: req.body.imageURL,

songURL : req.body.songURL,

album : req.body.album,

artist : req.body.artist,

genre : req.body.genre,

},

options);

return res.status(200).send({ success: true, data: result });

}

catch (error) {

return res.status(400).send({ success: false, message: error });

}

});

module.exports = router;