МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Вебзастосунок пошуку музичних партнерів та обміну композиціями

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22010809

Здобувач _____В. В. Іванов

підпис

«__» _____ 2024 р.

Керівник PhD, ст. викладач _____ І. О. Кандиба

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент ___ _____ А. О. Алексєєва

підпис

«__» _____ 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри Інженерії

Програмного Забезпечення

_____ Є. О. Давиденко

«_22__» ___грудня____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 408 факультету комп'ютерних наук

_____Іванов Віктор Вікторович_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Вебзастосунок пошуку музичних партнерів та обміну композиціями _____

Затверджена наказом по ЧНУ від «22» ____грудня_____ 2023 р. № ___269___

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є вебзастосунок для обміну композиціями з музичними партнерами_____

_____

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проектування програмного забезпечення;
- моделювання та проектування програмного забезпечення;

– розробка програмного забезпечення;

– здійснення тестування роботи програмного забезпечення;

5. Перелік графічних матеріалів

Презентація

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
А. О. Алексєєва	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи PhD, ст викладач Кандиба Ігор Олександрович

(посада, прізвище, ім’я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

Іванов Віктор Вікторович

(прізвище, ім’я, по батькові)

_____

(підпис)

Дата видачі завдання «_22_» грудня 2023р.

КАЛЕНДАРНИЙ ПЛАН

виконання бакалаврської кваліфікаційної роботи

Тема: «Вебзастосунок пошуку музичних партнерів та обміну композиціями»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	22.12.23	22.12.23	Виконано
2	Огляд літератури за темою роботи	04.01.24	04.01.24	Виконано
3	Складання календарного плану КРБ	15.01.24	15.01.24	Виконано
4	Аналіз предметної області	20.01.24	22.01.24	Виконано
5	Розробка проектних рішень	22.01.24	30.01.24	Виконано
6	Моделювання та конструювання	22.01.24	30.01.24	Виконано
7	Кодування ПЗ	22.01.24	30.01.24	Виконано
8	Розробка керівництва користувача	20.03.24	20.03.24	Виконано
9	Розробка частини з охорони праці	20.03.24	20.03.24	Виконано
10	Оформлення КРБ та презентації	20.03.24	20.03.24	Виконано
11	Відгук керівника КРБ	29.03.24	29.03.24	Виконано
12	Попередній захист	03.06.24	05.06.24	
13	Рецензування			
14	Захист кваліфікаційної роботи			

Розробив студент _____Іванов Віктор Вікторович_____ _____

(прізвище, ім’я, по батькові) (підпис)

«15» __січня__ 2024 р.

Керівник роботи _PhD, ст викладач___Кандиба Ігор Олександрович ___

(посада, прізвище, ім’я, по батькові) (підпис)

«15» __січня__ 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок пошуку музичних партнерів та обміну композиціями»

Студент 408 гр,: Іванов Віктор Вікторович

Керівник: PhD, ст. викладач Кандиба І. О.

Кваліфікаційна робота присвячена розробці та впровадженню вебзастосунку, спрямованого на спрощення процесу пошуку музичних партнерів та обміну композиціями серед музикантів та артистів.

Об'єктом кваліфікаційної роботи є процес створення вебзастосунку для публікації музики та текстових постів.

Предметом кваліфікаційної роботи є інструментарій розробки вебзастосунку для реалізації функцій публікації, обміну, прослуховування музики та текстових постів.

Метою кваліфікаційної роботи є розробка вебзастосунку спільного створення музики, обміну композиціями та отримання зворотнього зв'язоку від аудиторії.

У першому розділі було проаналізовано предметну область, зокрема галузь пошуку музичних партнерів, та визначено цільову платформу для розроблювальної системи — вебзастосунок. Проведений аналіз існуючих застосунків комунікації музичних партнерів дозволив виявити кілька ключових недоліків наявних систем, таких як обмежений функціонал, відсутність можливості публікувати контент для певної аудиторії, а також застарілий дизайн. На основі аналізу було сформульовано специфікацію вимог до застосунку, яка включає ключові функції, такі як пошук музичних партнерів, обмін композиціями, комунікація в режимі реального часу та безпека даних користувачів.

У другому розділі розроблені проектні рішення для системи пошуку музичних партнерів та обміну композиціями згідно з вимогами, визначеними у попередньому розділі. Проектні рішення включають опис функцій системи через сценарії

використання та графічні моделі, зокрема діаграми використання, послідовності, класів та компонентів. Це дозволило створити чітке уявлення про архітектуру та функціональність системи, що стало основою для подальшої розробки.

У третьому розділі було створено базове оточення для нашого застосунку. Шлях до досягнення цієї мети включав вибір технологій, мов програмування та інструментів, які найкращим чином відповідають нашим потребам і сприятимуть успішному розвитку проекту.

У четвертому розділі налаштовано та вдало використано сучасні технології як для бекенду, так і для фронтенду.

КРБ викладена на 68 сторінок, вона містить 4 розділи, 39 ілюстрацій, 20 джерел в переліку посилань

Ключові слова: вебзастосунок, музичні партнери, обмін композиціями, музиканти, артисти.

ABSTRACT

of the Bachelor's Thesis

"Web Application for Finding Musical Partners and Exchanging Compositions"

Student of group 408: Ivanov Viktor Viktorovich

Supervisor: PhD, senior lecturer Kandyba I. O.

The qualification work is dedicated to the development and implementation of a web application aimed at simplifying the process of finding music partners and exchanging compositions among musicians and artists.

The object of the qualification work is the process of creating a web application for publishing music and text posts.

The subject of the qualification work is the toolset for developing a web application to implement the functions of publishing, exchanging, and listening to music and text posts.

The goal of the qualification work is to develop a web application for collaborative music creation, exchange of compositions, and receiving feedback from the audience.

In the first section, the domain of music partner search was analyzed, identifying the target platform for the system—a web application. The analysis of existing music partner communication applications revealed several key shortcomings, such as limited functionality, the inability to publish content for a specific audience, and outdated design. Based on this analysis, the specification of requirements for the application was formulated, including key functions such as searching for music partners, exchanging compositions, real-time communication, and data security.

In the second section, project solutions for the music partner search and composition exchange system were developed according to the requirements defined in the previous section. These project solutions include the description of system functions through use case scenarios and graphical models, such as use case diagrams, sequence diagrams, class diagrams, and component diagrams, creating a clear understanding of the system's architecture and functionality, which formed the basis for further development.

In the third section, the basic environment for our application was created. This involved selecting the technologies, programming languages, and tools that best meet our needs and support the successful development of the project.

In the fourth section, modern technologies were successfully configured and utilized for both the backend and frontend.

The qualification work is presented on 68 pages, containing 4 sections, 39 illustrations, and 20 sources in the reference list.

Keywords: web application, music partners, composition exchange, musicians, artists.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

BE – backend

FE – frontend

UML – unified modeling language

UI – user interface

DB – database

ПЗ – програмне забезпечення

ВСТУП

У сучасному цифровому світі соціальні мережі стають все більш популярними серед користувачів, що активно обмінюються інформацією та спілкуються з іншими. Створення соціальної мережі з можливістю публікації музики та текстових постів відкриває нові можливості для музикантів та поціновувачів музики. Цей проект може стати платформою для спільного творчого процесу, обміну ідеями та звуковими записами, а також сприяти розвитку музичної спільноти та підтримці творчих ініціатив.

Об'єктом кваліфікаційної роботи є процес створення вебзастосунку для публікації музики та текстових постів.

Предметом кваліфікаційної роботи є інструментарій розробки вебзастосунку для реалізації функцій публікації, обміну, прослуховування музики та текстових постів.

Метою кваліфікаційної роботи є розробка вебзастосунку спільного створення музики, обміну композиціями та отримання зворотнього зв'язоку від аудиторії.

Для досягнення цієї мети визначено наступні завдання:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проектування програмного забезпечення;
- моделювання та проектування програмного забезпечення;
- розробка програмного забезпечення;
- здійснення тестування роботи програмного забезпечення.

1 АНАЛІЗ ЗАСТОСУНКІВ ПОШУКУ ТА ВЗАЄМОДІЇ МУЗИЧНИХ ПАРТНЕРІВ

1.1 Опис галузі пошуку музичних партнерів

Галузь пошуку музичних партнерів є важливим сегментом музичної індустрії, який забезпечує взаємодію між музикантами, артистами та іншими учасниками музичної сфери. Зазвичай цей пошук спрямований на виявлення співпраці для створення нових музичних творів, організації концертів, студійних сесій, записів та інших проєктів.

Історично музиканти покладалися на традиційні методи пошуку партнерів, такі як особисті знайомства, місцеві музичні сцени та рекомендації колег. Проте розвиток інтернету та цифрових технологій істотно змінив цю ситуацію, відкривши можливості для більш ефективного та масштабного пошуку партнерів.

З розвитком технологій та інтернету, пошук музичних партнерів значно спростився та набрав популярності. З'явилося безліч вебплатформ та застосунків, що дозволяють музикантам та артистам знаходити один одного, обмінюватися своїми роботами та співпрацювати в реальному часі. Ці платформи також сприяють обміну досвідом, комунікації та розширенню мережі контактів.

Одним із напрямків сучасного ринку є соціальні мережі для музикантів, які поєднують функції знайомств, спілкування та пошуку співпраці. Вони часто пропонують музикантам можливість представити свої профілі, портфоліо, демо-записи та інші матеріали, що допомагають знайти партнерів для співпраці.

Також варто відзначити розвиток алгоритмів рекомендацій та штучного інтелекту, які використовуються для більш ефективного пошуку партнерів, ґрунтуючись на інтересах та стилях музикантів. Це дозволяє покращити процес пошуку та збільшити ймовірність успішної співпраці.

Також критичним аспектом у пошуку музичних партнерів є питання інтелектуальних прав. Музиканти повинні бути поінформовані про свої права на твори, які вони створюють, і обмінюватися ними лише з дотриманням умов інтелектуальної власності. Вебплатформи для пошуку музичних партнерів повинні забезпечувати засоби для реєстрації та захисту інтелектуальної власності користувачів, такі як:

– ліцензії для авторських прав, що дозволяють музикантам встановлювати умови використання своїх робіт;

– засоби відстеження використання музичних творів, щоб забезпечити прозорість та справедливість у співпраці;

– освітні ресурси для музикантів щодо їхніх прав і обов'язків в галузі інтелектуальної власності.

Захист інтелектуальної власності та право на власність над творами є ключовими факторами для забезпечення довіри та безпеки музикантів, що беруть участь у пошуку партнерів через інтернет.

1.2 Огляд та аналіз існуючих застосунків комунікації музичних партнерів

Для виявлення основних вимог до розробки вебзастосунку для музикантів та артистів проведено аналіз існуючих застосунків комунікації музичних партнерів. Аналіз подібних систем проводиться з метою:

– визначення ключових функцій та можливостей: Вивчення наявних застосунків дозволяє визначити основні функції, які повинні бути реалізовані у новому вебзастосунку, такі як співпраця між музикантами, обмін ідеями та ресурсами, а також можливість створювати нові музичні проєкти;

– оцінка переваг та недоліків: Аналіз застосунків допомагає визначити сильні та слабкі сторони кожного з них, що дозволяє уникнути подібних недоліків у новому розробленому застосунку та покращити його функціональність;

– ыдентифікація потенційних можливостей для поліпшення: Вивчення існуючих застосунків дає змогу виявити можливості для вдосконалення, наприклад, у плані зручності користування, налаштувань конфіденційності або підтримки різних форматів контенту;

– оцінка популярності та прийнятності користувачами: Вивчення існуючих застосунків дозволяє оцінити популярність різних функцій та характеристик серед користувачів, що допомагає зробити новий застосунок більш привабливим для цільової аудиторії.

SoundCloud, розроблений SoundCloud Limited, є великою онлайн-платформою для завантаження та відтворення музики (рис 1.1) [1]. Ця платформа займає провідне місце серед музичних сервісів завдяки своїй клієнт-серверній архітектурі, яка забезпечує стабільну роботу та інтерактивність застосунку. Використання JavaScript для фронтенд-частини та Java для бекенд-частини створює потужну та надійну базу для обробки великої кількості запитів і користувачів одночасно. Завдяки такій архітектурі, SoundCloud здатен підтримувати високу швидкість завантаження та відтворення музичних треків, забезпечуючи користувачам якісний досвід прослуховування.

SoundCloud пропонує широкий функціонал, який робить його привабливим для різних категорій користувачів. Серед основних можливостей платформи - завантаження та відтворення музики, що дозволяє музикантам і слухачам з легкістю ділитися своїми треками та відкривати нову музику. Користувачі можуть залишати коментарі під треками, ставити лайки, робити репости та створювати власні плейлисти. Це сприяє активній взаємодії між користувачами та створює спільноту, де

музиканти можуть обговорювати свої роботи та отримувати зворотний зв'язок. Крім того, SoundCloud підтримує функцію співпраці з іншими музикантами, що дає можливість користувачам спільно створювати музичні треки, об'єднуючи свої таланти.

Однією з найбільших переваг SoundCloud є його велика спільнота користувачів. Завдяки широкій базі користувачів, музиканти можуть швидко знаходити нових партнерів для співпраці, ділитися своєю творчістю з великою аудиторією та встановлювати нові зв'язки. Це робить платформу привабливою для як професійних музикантів, так і для аматорів, які тільки починають свій шлях у музичній індустрії. Однак, незважаючи на всі переваги, SoundCloud має й деякі недоліки. Основним недоліком є відсутність можливості обмежити доступ до завантажених матеріалів. Усі матеріали публікуються для загального доступу, що може бути незручним для користувачів, які хочуть ділитися своєю музикою тільки з обраною аудиторією. Це може викликати занепокоєння щодо конфіденційності та безпеки персональної інформації та творчих робіт.

Загалом, SoundCloud є потужною платформою для завантаження, відтворення та обміну музикою, яка пропонує багатий функціонал та велику спільноту користувачів. Однак, для деяких користувачів, відсутність можливості обмеження доступу до завантажених матеріалів може бути значним недоліком, що потребує уваги та можливого вдосконалення в майбутньому.

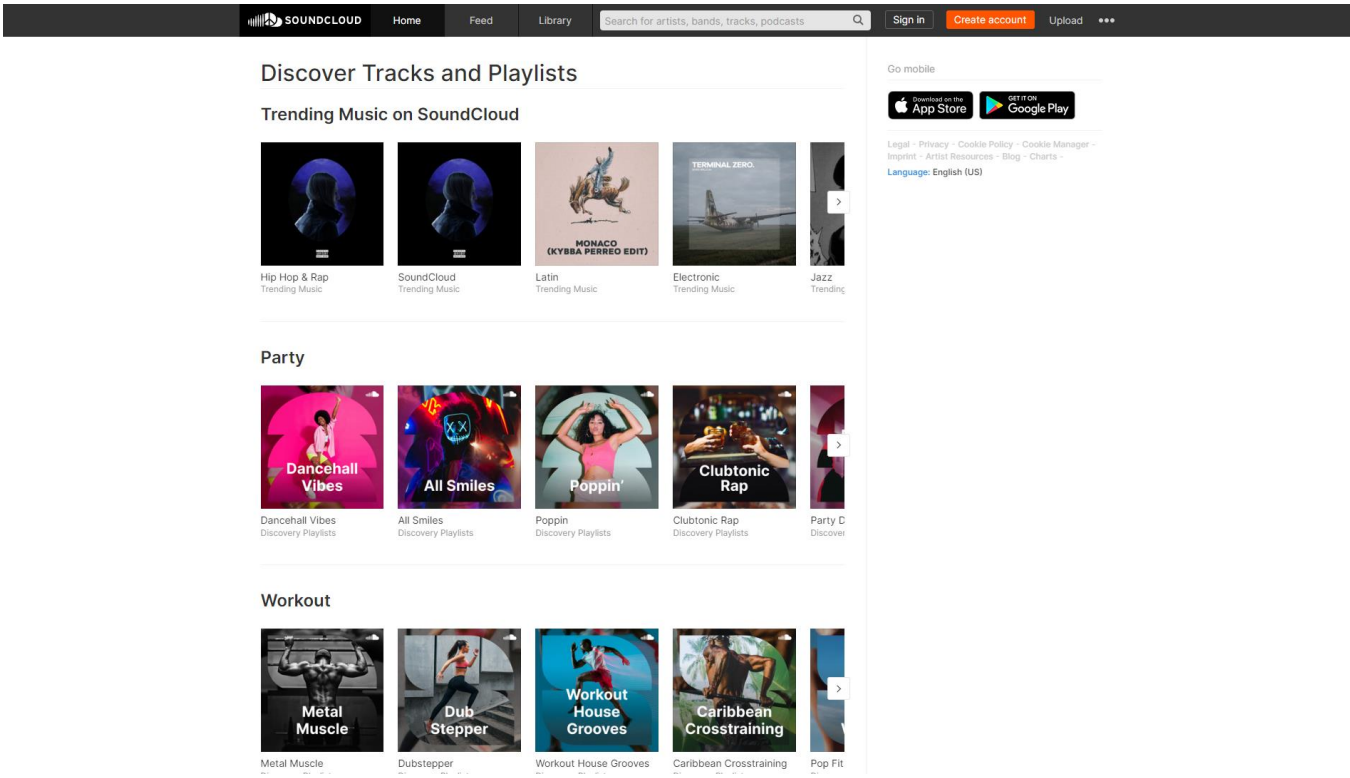

Рисунок 1.1 – Інтерфейс Soundcloud

BandMix, розроблений BandMix LLC, є спеціалізованою платформою для музикантів, яка надає їм можливість співпраці у пошуку гуртів та проєктів (рис 1.2) [2]. В основі її функціонування лежить клієнт-серверна архітектура, реалізована за допомогою PHP як для фронтенд, так і для бекенд-частини. Це забезпечує ефективне керування даними та користувачами, що є критично важливим для надійної роботи платформи. Завдяки використанню PHP, платформа може обробляти великий обсяг даних, зберігаючи стабільність і швидкість роботи, що дозволяє користувачам безперешкодно взаємодіяти з нею.

Однією з головних особливостей BandMix є можливість музикантам розміщувати власні музичні оголошення. Це включає в себе обмін ідеями та ресурсами, а також створення детального профілю, де можна вказати свої музичні навички та інструменти, на яких вони грають (рис. 1.2). Така функціональність дозволяє музикантам точно окреслити свої можливості та потреби, полегшуючи

іншим користувачам пошук відповідних партнерів для співпраці. Крім того, BandMix пропонує зручні інструменти для комунікації, що сприяють швидкому встановленню контактів та обміну інформацією між музикантами.

Платформа також відзначається великою базою активних користувачів, що значно полегшує пошук музичних партнерів у своєму регіоні. Завдяки цьому, музиканти можуть швидко знайти однодумців і розпочати спільні музичні проєкти. Можливість знаходити музикантів поблизу є однією з ключових переваг BandMix, оскільки це сприяє організації живих репетицій та виступів, що є важливим аспектом музичної діяльності. Велика база користувачів також сприяє різноманітності музичних стилів та жанрів, представлених на платформі, що розширює можливості для співпраці.

Однак, незважаючи на численні переваги, BandMix має деякі обмеження. Одним з них є відсутність можливості публікувати власні думки та пісні для певної аудиторії. Усі матеріали, завантажені на платформу, стають доступними для загального публічного перегляду. Це може бути незручним для користувачів, які хочуть ділитися своєю музикою або думками тільки з обраною групою осіб або в рамках певної спільноти. Така відсутність приватності може викликати занепокоєння щодо конфіденційності та безпеки персональної інформації та творчих робіт, особливо серед користувачів, які цінують обмежений доступ до своїх матеріалів.

Таким чином, BandMix є потужною платформою для музикантів, яка пропонує численні можливості для співпраці та пошуку партнерів. Використання PHP для реалізації клієнт-серверної архітектури забезпечує надійну та ефективну роботу системи, а велика база користувачів сприяє швидкому знаходженню відповідних партнерів. Проте, недоліки у вигляді відсутності можливості обмежити доступ до публікацій можуть бути значним мінусом для деяких користувачів, що вимагає подальшого вдосконалення платформи для задоволення потреб всіх її користувачів.

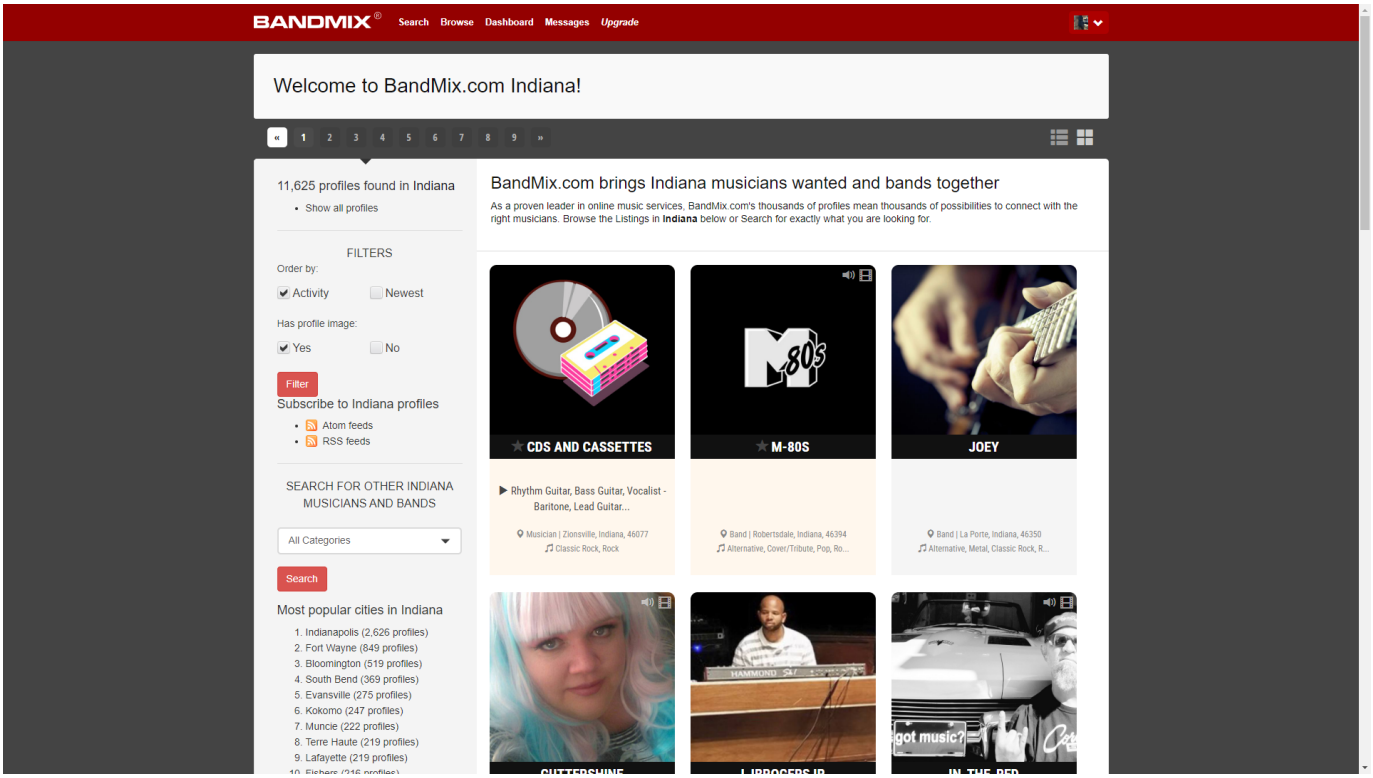

Рисунок 1.2 – Інтерфейс BandMix

Kompoz, створений компанією Kompoz Inc., є інноваційною платформою, орієнтованою на колаборативне середовище для створення музики (рис 1.3) [3]. Його клієнт-серверна архітектура, реалізована за допомогою PHP для фронтенд і бекенд частин, забезпечує ефективну та зручну взаємодію між користувачами. Використання PHP дозволяє платформі обробляти великі обсяги даних та забезпечує стабільність і швидкодію при виконанні різноманітних операцій, таких як завантаження та редагування музичних композицій. Ця технологія дозволяє створювати надійну інфраструктуру, що підтримує багатокористувацькі сесії та забезпечує безперервність роботи.

Основна концепція Kompoz полягає у створенні музики в колаборативному середовищі, де музиканти з усього світу можуть спільно працювати над музичними проєктами. Платформа дозволяє користувачам створювати та додавати різні інструментальні доріжки, обговорювати їх та вносити зміни до композицій (рис. 1.3).

Це сприяє взаємодії та обміну ідеями між музикантами, що в свою чергу покращує якість кінцевого продукту. Завдяки можливості одночасної роботи над композицією з різних місць, Kompoz забезпечує гнучкість та адаптивність у процесі створення музики, що особливо важливо у сучасному глобалізованому світі.

Kompoz також пропонує простоту використання та зручний інтерфейс, що робить його доступним навіть для новачків. Платформа містить інтуїтивно зрозумілі інструменти для завантаження та редагування музичних треків, що дозволяє користувачам швидко освоїтися та почати створювати музику. Крім того, Kompoz має активну спільноту музикантів, які готові ділитися своїм досвідом та допомагати новим користувачам. Ця спільнота є важливим ресурсом для обміну знаннями та натхненням, що сприяє зростанню та розвитку кожного учасника.

Однією з ключових функцій Kompoz є можливість обговорення музичних треків та внесення змін на основі зворотного зв'язку від інших користувачів. Це створює умови для активної співпраці та вдосконалення композицій, оскільки музиканти можуть отримувати конструктивну критику та пропозиції від своїх колег. Така інтерактивність сприяє більш творчому підходу до створення музики та допомагає уникнути багатьох поширених помилок на ранніх етапах роботи над композицією.

Проте, незважаючи на численні переваги, Kompoz має і свої недоліки. Основним з них є відсутність можливості публікувати власні думки та пісні для певної аудиторії, оскільки весь контент на платформі є загальнодоступним для всіх користувачів. Це може бути проблематичним для тих музикантів, які бажають обмежити доступ до своїх робіт лише для певної групи людей або в рамках приватних проєктів. Відсутність таких налаштувань конфіденційності може викликати занепокоєння щодо безпеки та приватності творчих робіт.

Таким чином, Kompoz є потужною платформою для колаборативного створення музики, що пропонує багатий функціонал для спільної роботи та обміну ідеями. Використання PHP для реалізації клієнт-серверної архітектури забезпечує стабільну

роботу системи, а активна спільнота користувачів сприяє творчому розвитку та вдосконаленню музичних композицій. Проте, обмеження щодо налаштувань приватності контенту вимагають подальшого вдосконалення платформи для забезпечення комфортних умов роботи для всіх категорій користувачів.

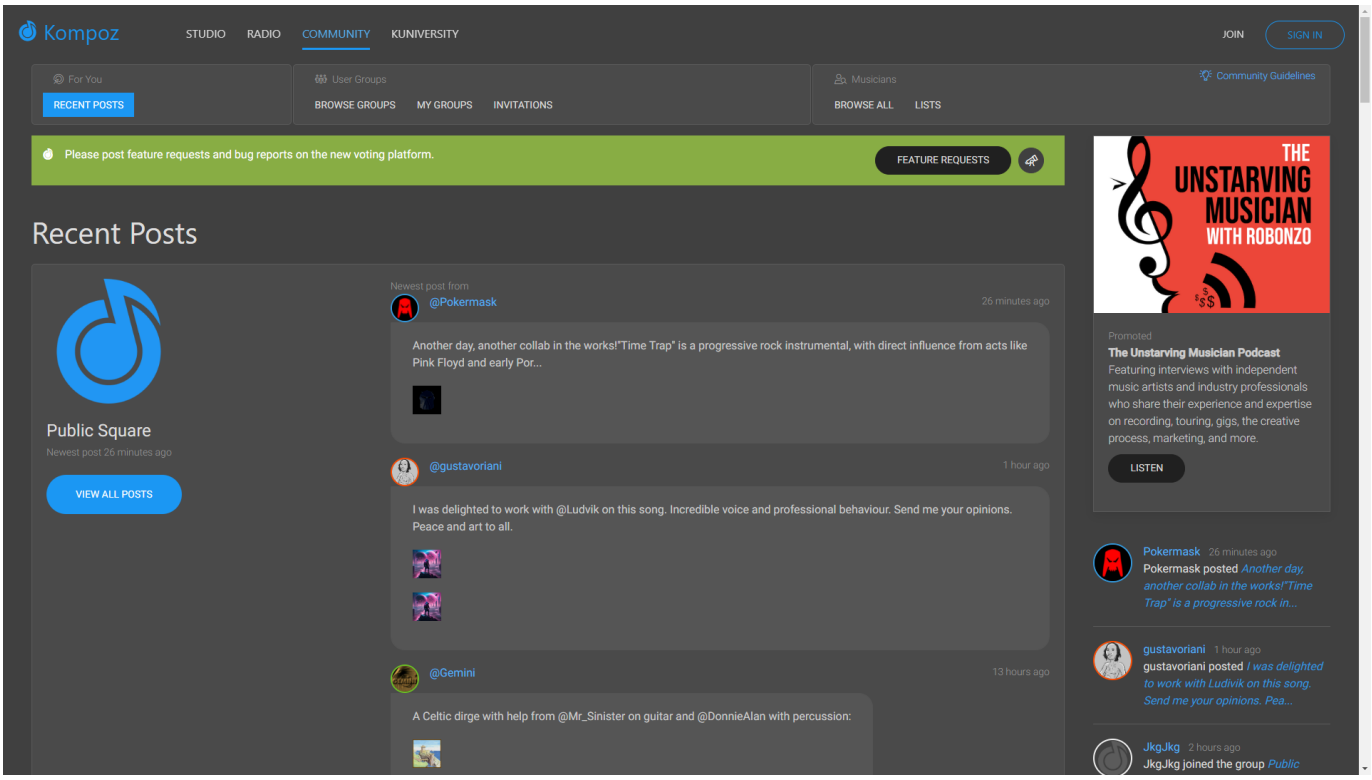

Рисунок 1.3 – Інтерфейс Kompoz

1.3 Специфікація вимог до застосунку

Призначення та межі застосунку

1) Призначення застосунку, для якого розробляється програмне забезпечення:

Призначенням застосунку є забезпечення зручного пошуку музичних партнерів та обміну музичними композиціями серед музикантів та артистів.

2) Погодження, що ухвалені в програмній документації

Було погоджено, що застосунок буде реалізовано як вебзастосунок із використанням сучасних технологій, таких як React, NestJS, та Supabase.

3) Межі проєкту ПЗ

Крайня дата завершення роботи над вебзастосунком – 12.06.2024 р.

Загальний опис

1) Сфера застосування

Даний вебзастосунок може застосовуватись будь-якою особою, яка зацікавлена у пошуку музичних партнерів, обміні музичними композиціями та отриманні зворотного зв'язку від аудиторії.

2) Характеристики користувачів

Основні характеристики користувачів: наявність смартфону, планшету або ПК та доступу до мережі Інтернет.

3) Загальна структура і склад системи

Основні частини для створення програмного забезпечення: сервер, база даних, вебзастосунок, API.

4) Загальні обмеження

Єдине обмеження для роботи з ПЗ – наявність доступу до мережі Інтернет.

Функції застосунку

1) Функція пошуку музичних партнерів

Опис функції: Застосунок надає можливість пошуку музикантів за різними критеріями, такими як жанр музики, інструменти, досвід, локація тощо.

Вхідна і вихідна інформація: Користувач вводить критерії пошуку, система надає список потенційних партнерів.

Функціональні вимоги: Доступ до бази даних користувачів та мережі Інтернет.

2) Функція обміну композиціями

Опис функції: Користувачі можуть завантажувати власні музичні композиції на платформу, обмінюватися ними та отримувати зворотний зв'язок.

Вхідна і вихідна інформація: Користувач завантажує композицію, інші користувачі можуть прослуховувати та коментувати.

Функціональні вимоги: Доступ до бази даних композицій та мережі Інтернет.

3) Функція комунікації в режимі реального часу

Опис функції: Застосунок підтримує можливість обміну повідомленнями між користувачами в режимі реального часу.

Вхідна і вихідна інформація: Користувач надсилає повідомлення, інший користувач отримує його негайно.

Функціональні вимоги: Доступ до мережі Інтернет та серверної частини для обробки повідомлень.

4) Функція зворотного зв'язку від аудиторії

Опис функції: Користувачі можуть отримувати відгуки, оцінки та коментарі щодо своїх композицій від інших користувачів.

Вхідна і вихідна інформація: Користувач отримує відгуки та оцінки, які зберігаються у системі.

Функціональні вимоги: Доступ до бази даних користувачів та мережі Інтернет.

Вимоги до інформаційного забезпечення

1) Джерела і зміст вхідної інформації (даних)

У вебзастосунку основним джерелом вхідної інформації є користувачі, які заповнюють свої профілі, завантажують композиції та взаємодіють з іншими користувачами.

2) Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги до даного пункту відсутні.

Висновки до розділу 1

У розділі було проаналізовано предметну область, зокрема галузь пошуку музичних партнерів, та визначено цільову платформу для розроблювальної системи — вебзастосунок. Проведений аналіз існуючих застосунків комунікації музичних партнерів дозволив виявити кілька ключових недоліків наявних систем, таких як

обмежений функціонал, відсутність можливості публікувати контент для певної аудиторії, а також застарілий дизайн.

На основі аналізу було сформульовано специфікацію вимог до застосунку, яка включає ключові функції, такі як пошук музичних партнерів, обмін композиціями, комунікація в режимі реального часу та безпека даних користувачів. Завдяки цьому можна розробити систему, що вигідно виділиться серед аналогічних застосунків та забезпечить ефективну взаємодію між музикантами та артистами.

Результатом проведеної роботи в цьому розділі стало виявлення специфікації вимог до розроблюваного програмного забезпечення, яке має потенціал зайняти провідне місце в цій ніші.

2 МОДЕЛЮВАННЯ ФУНКЦІОНАЛЬНОСТІ ЗАСТОСУНКУ

Після проведеного аналізу галузі пошуку музичних партнерів та огляду існуючих застосунків, виявлено основні вимоги до вебзастосунку для забезпечення ефективної комунікації та співпраці музикантів та артистів. На основі специфікації вимог було сформовано чітке уявлення про функціональність майбутньої системи.

Далі ми перейдемо до моделювання функціональності системи за допомогою UML [4]. Це дозволить детально розглянути та візуалізувати різні сценарії використання системи, а також проаналізувати взаємодію між різними компонентами та користувачами. Таким чином, ми зможемо забезпечити оптимальне проектування вебзастосунку, що відповідає всім необхідним вимогам та очікуванням користувачів.

2.1 Моделювання сценаріїв використання системи

Діаграма прецедентів системи відображає різні сценарії використання, що охоплюють основні функції та взаємодії користувачів з вебзастосунком пошуку музичних партнерів та обміну композиціями. Ця діаграма допомагає зрозуміти, як користувачі будуть взаємодіяти з системою, які дії вони можуть виконувати та які функції доступні їм у рамках застосунку.

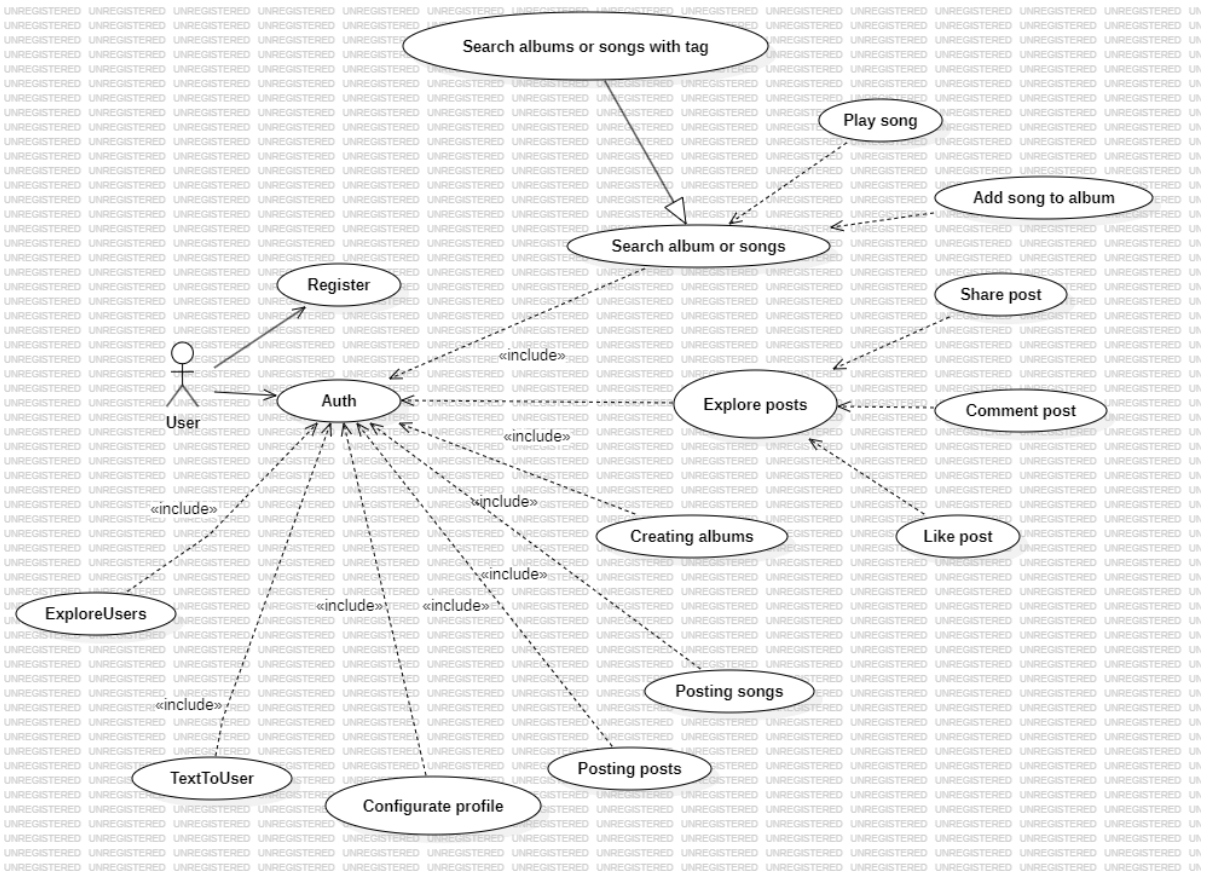

Рисунок 2.1 – Діаграма прецедентів системи

На діаграмі на рисунку 2.1 основними прецедентами системи є:

- реєстрація;
- вхід до системи;
- налаштування профілю;
- публікація постів;
- публікація пісень;
- створення альбомів;
- перегляд публікацій;
- схвалення постів;
- коментування постів;
- розповсюдження постів;

- пошук пісень та альбомів;
- додання пісні до альбому;
- прослуховування пісень;
- пошук користувачів;
- спілкування з користувачами.

Розглянемо детальніше кожен прецидент за допомогою діаграм діяльності та послідовності.

2.2 Моделювання поведінки користувача

Діаграма діяльності відображає послідовність операцій і дій, що відбуваються під час виконання певного процесу в системі. Вона є корисним інструментом для моделювання та візуалізації логіки процесу, відображаючи потік діяльності між різними станами системи або об'єктами. На наступній діаграмі і наведена логіка використання застосунку на основі сформованих прецедентів.

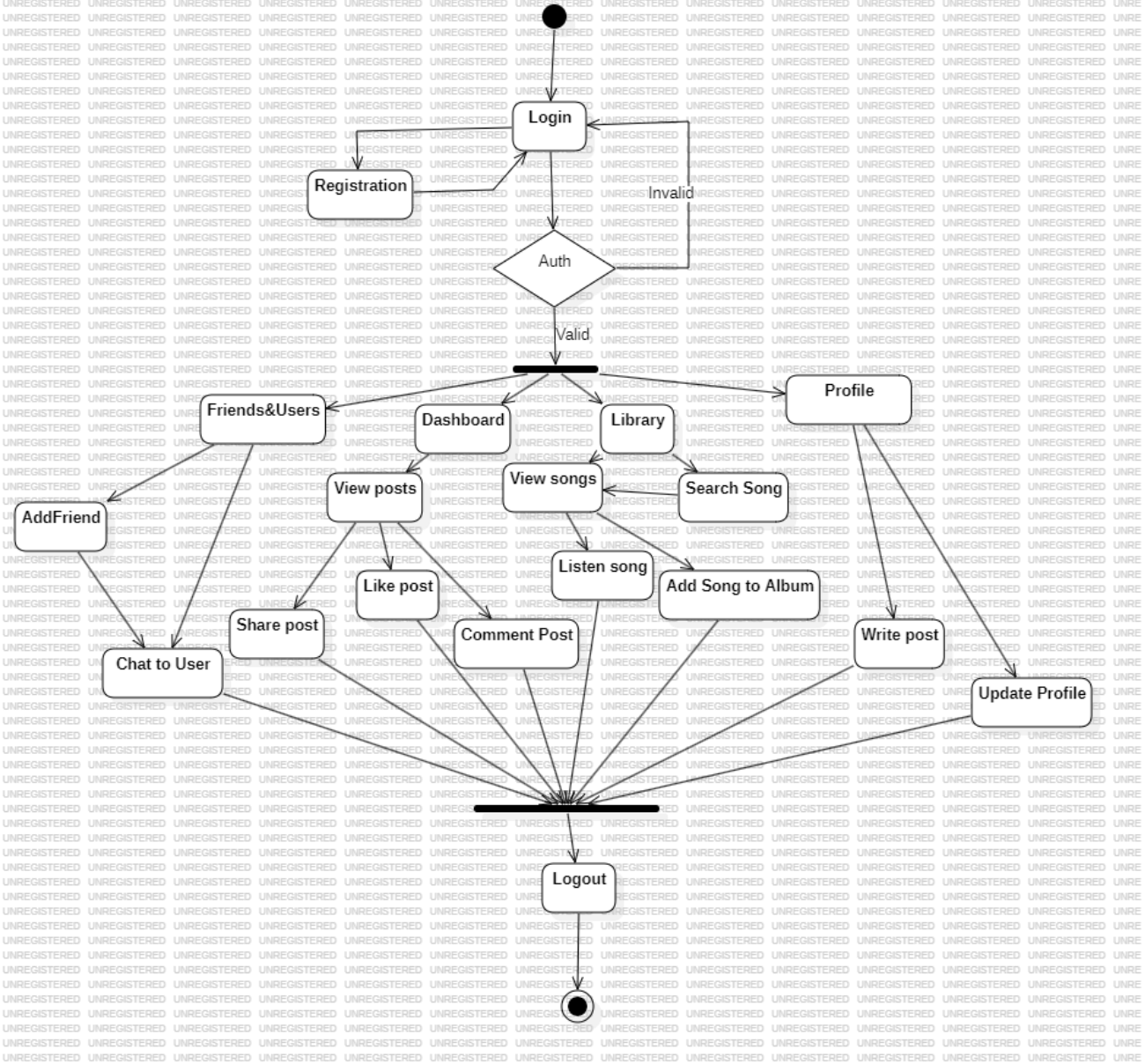

Рисунок 2.2 – Діаграма діяльності системи

Діаграма послідовності ж відображає хронологічний порядок дій і взаємодій між різними об'єктами системи під час виконання певного прецеденту. Ця діаграма є корисним інструментом для ілюстрації послідовності операцій, які відбуваються в системі під час взаємодії користувачів з вебзастосунком. Нижче наведені діаграми послідовності для кожного прецеденту.

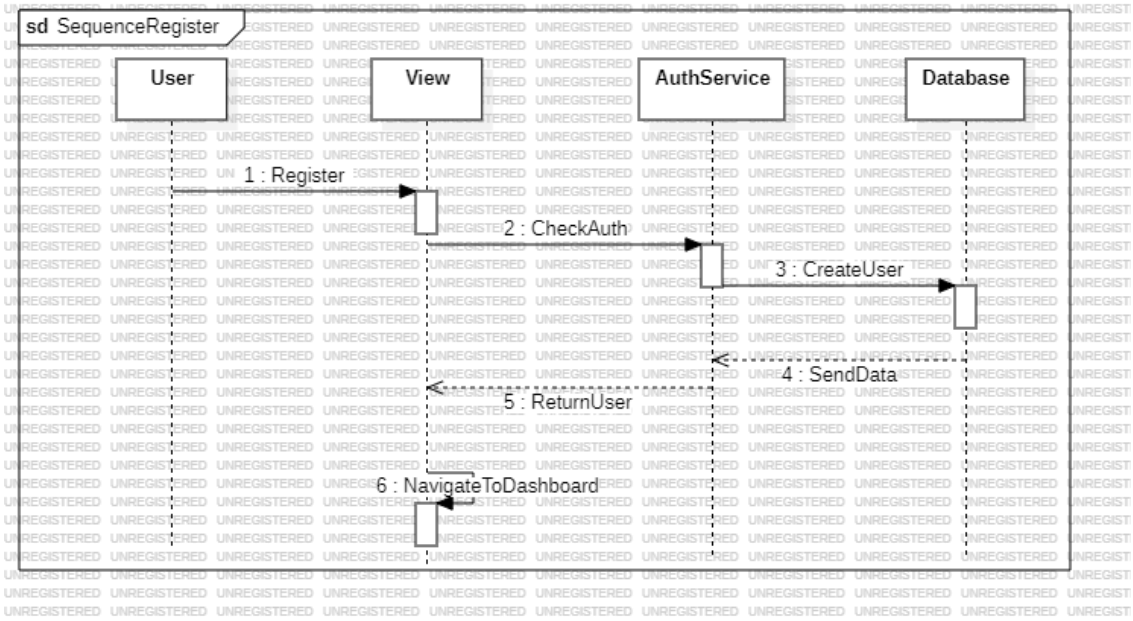

Рисунок 2.3 – Діаграма послідовності реєстрації

На рис. 2.3 на діаграмі видно, що аби зареєструватися, користувачу необхідно відправити свої данні до сервісу, після чого відповідний сервіс створить цього користувача.

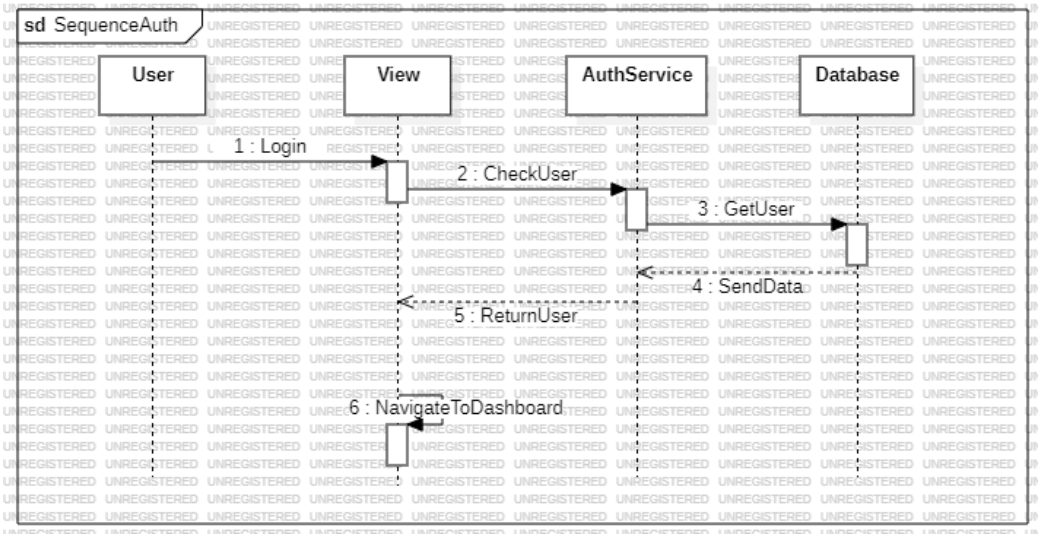

Рисунок 2.4 – Діаграма послідовності логіну

На рис. 2.4 на діаграмі видно, що аби залогінитись, користувачу необхідно відправити свої данні до сервісу, після чого відповідний сервіс поверне данні стосовно відповідного користувача.

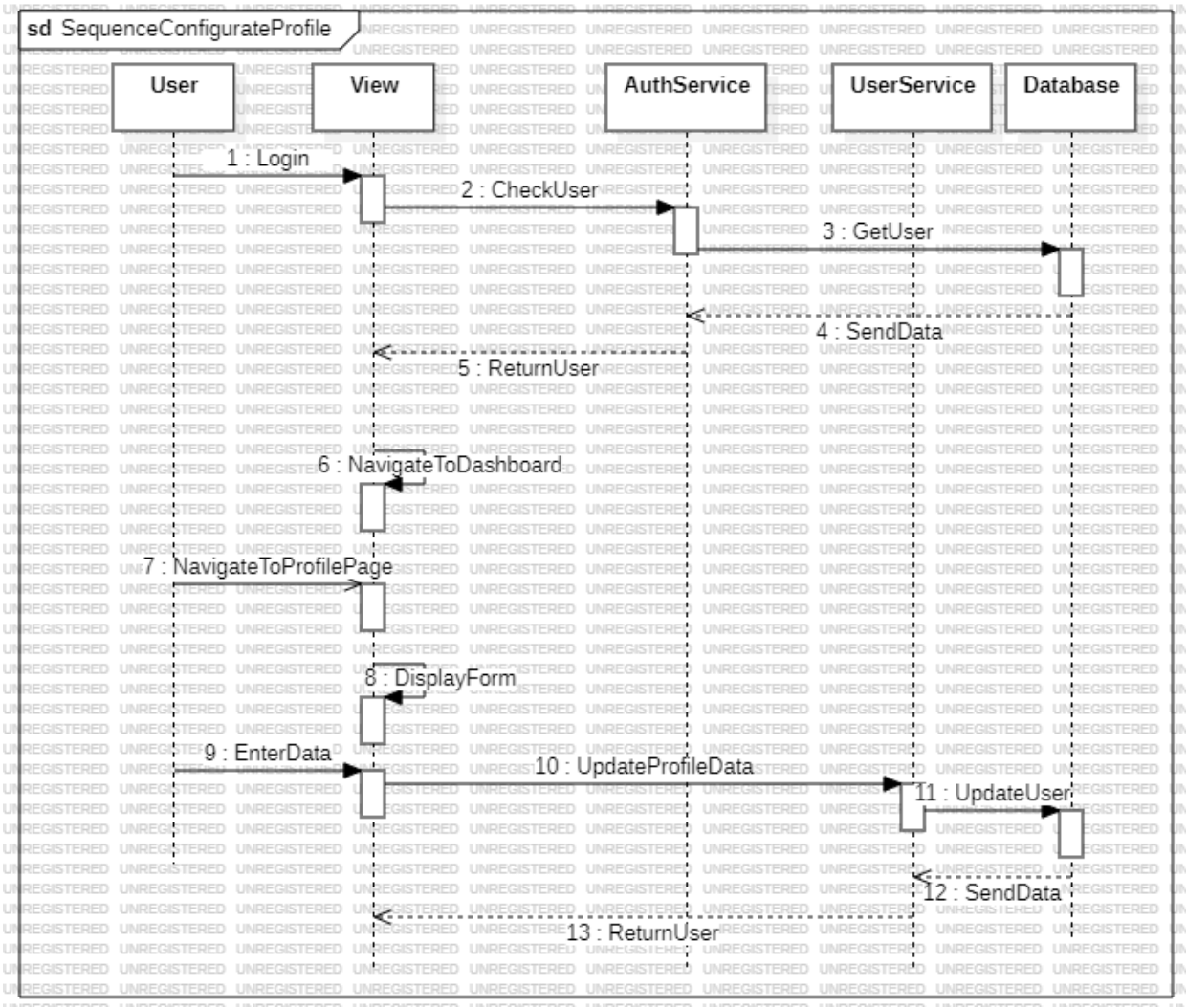

Рисунок 2.5 – Діаграма послідовності налаштування профілю

На рис. 2.5 на діаграмі видно, що аби налаштувати профіль, користувачу необхідно відправити певні данні, після чого відповідний сервіс змінить данні стосовно користувача.

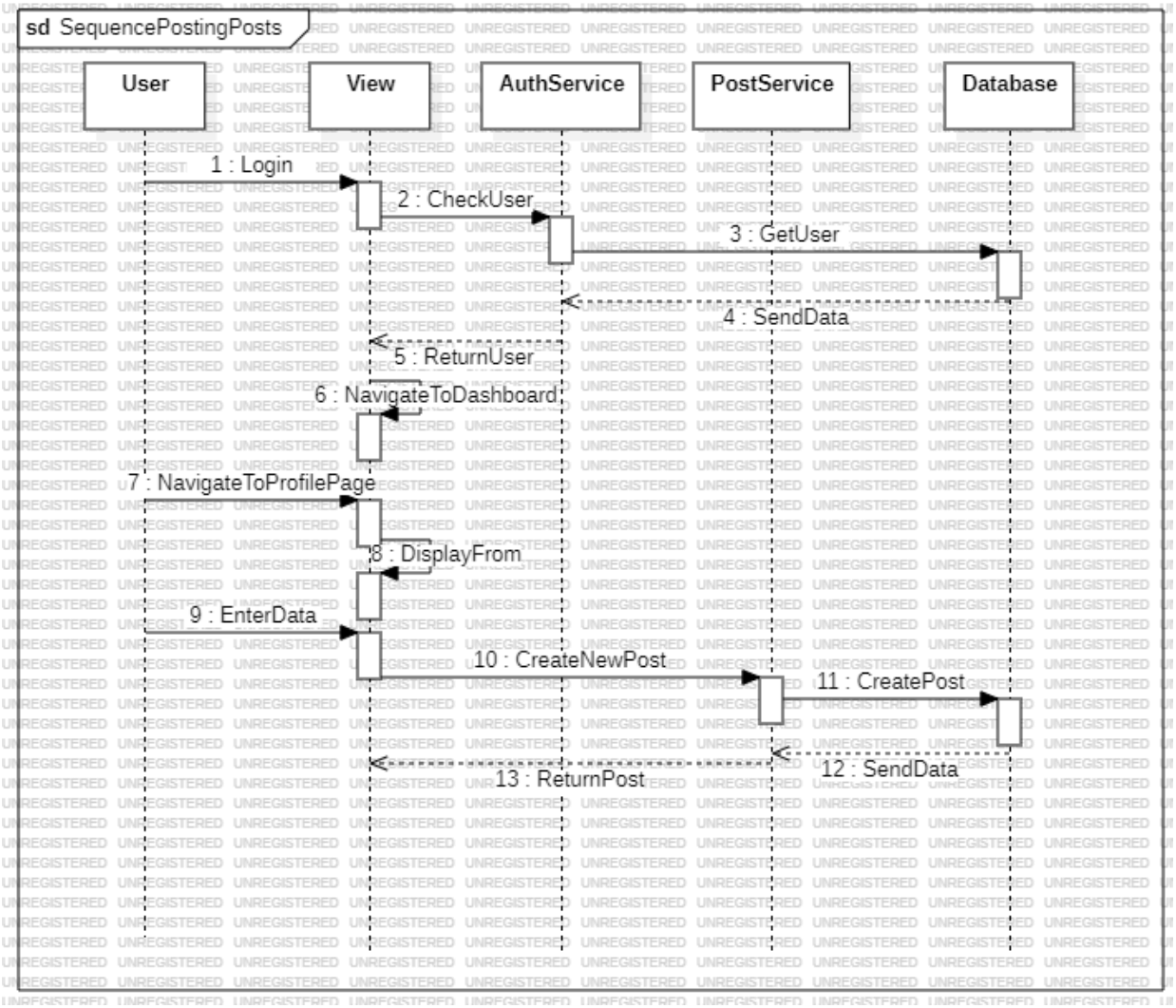

Рисунок 2.6 – Діаграма послідовності публікації постів

На рис. 2.6 на діаграмі видно, що аби опублікувати пост, користувачу необхідно відправити данні цього посту, після чого відповідний сервіс його опублікує.

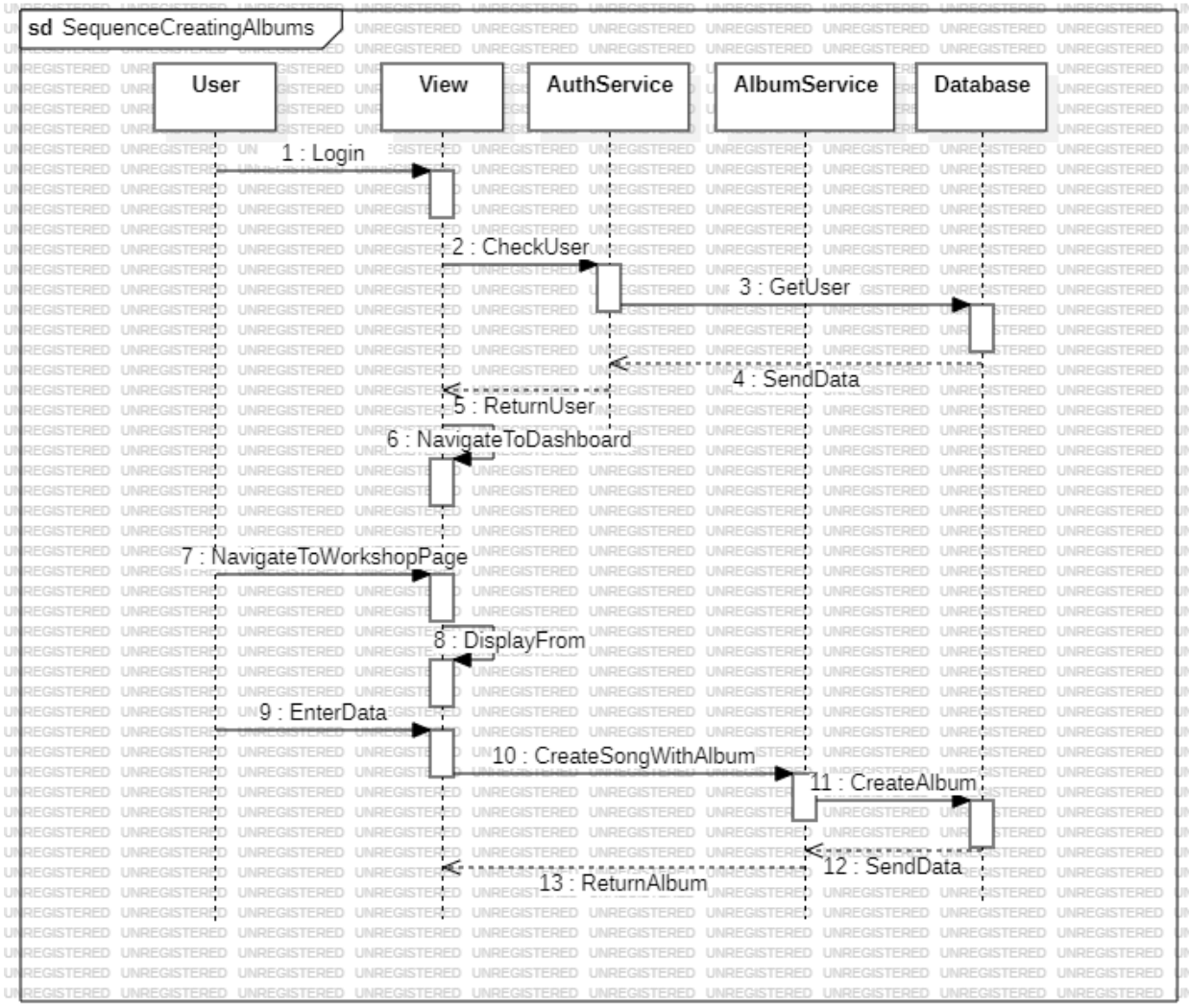

Рисунок 2.7 – Діаграма послідовності створення альбомів

На рис. 2.7 на діаграмі видно, що аби створити альбом користувачу необхідно ввести та відправити деякі данні, після чого відповідний сервіс створить цей альбом.

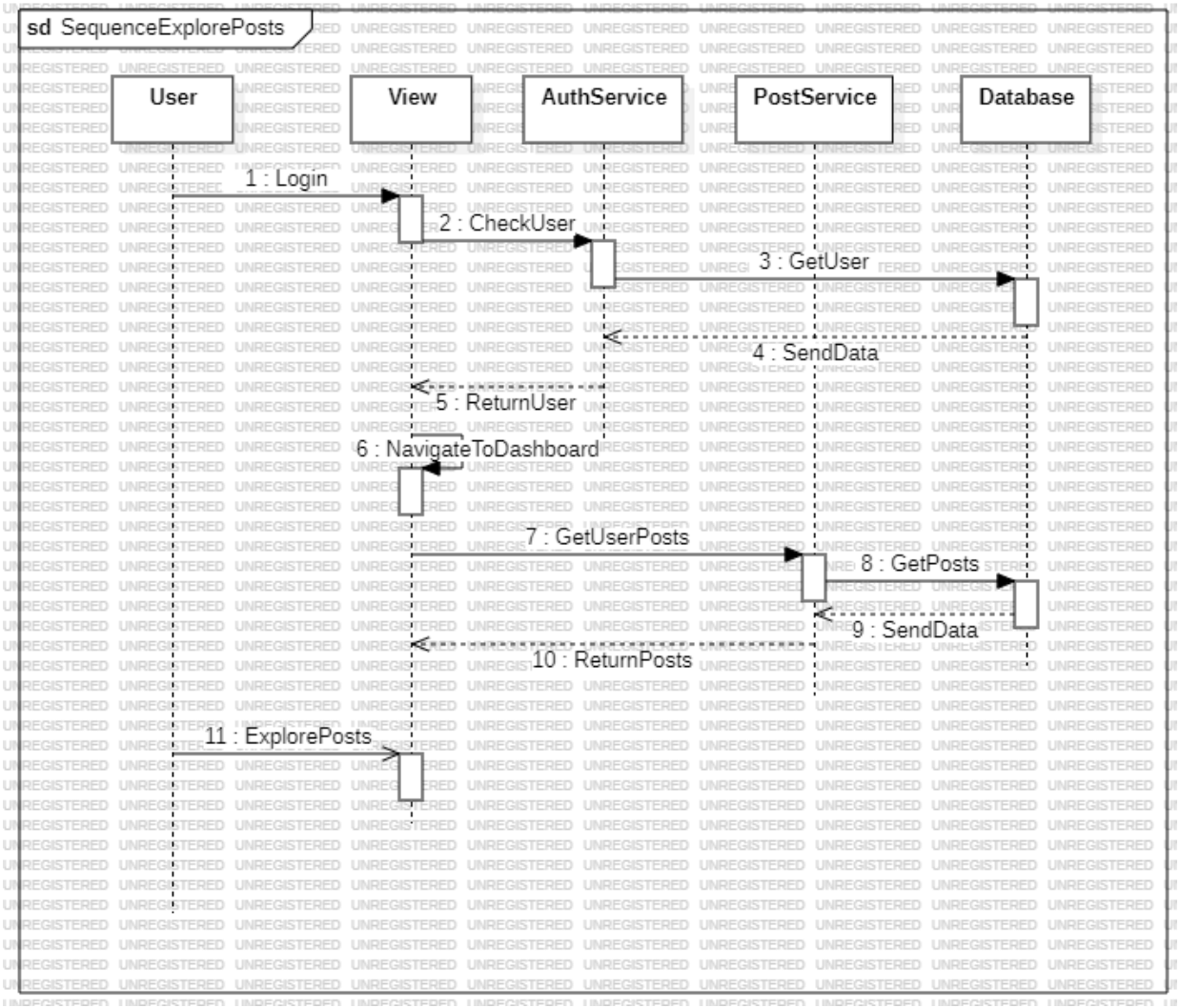

Рисунок 2.8 – Діаграма послідовності перегляду постів

На рис. 2.8 на діаграмі видно, що аби переглянути пости, користувачу необхідно зайти на дешборд сторінку, після чого відповідний сервіс відправить необхідні публікації.

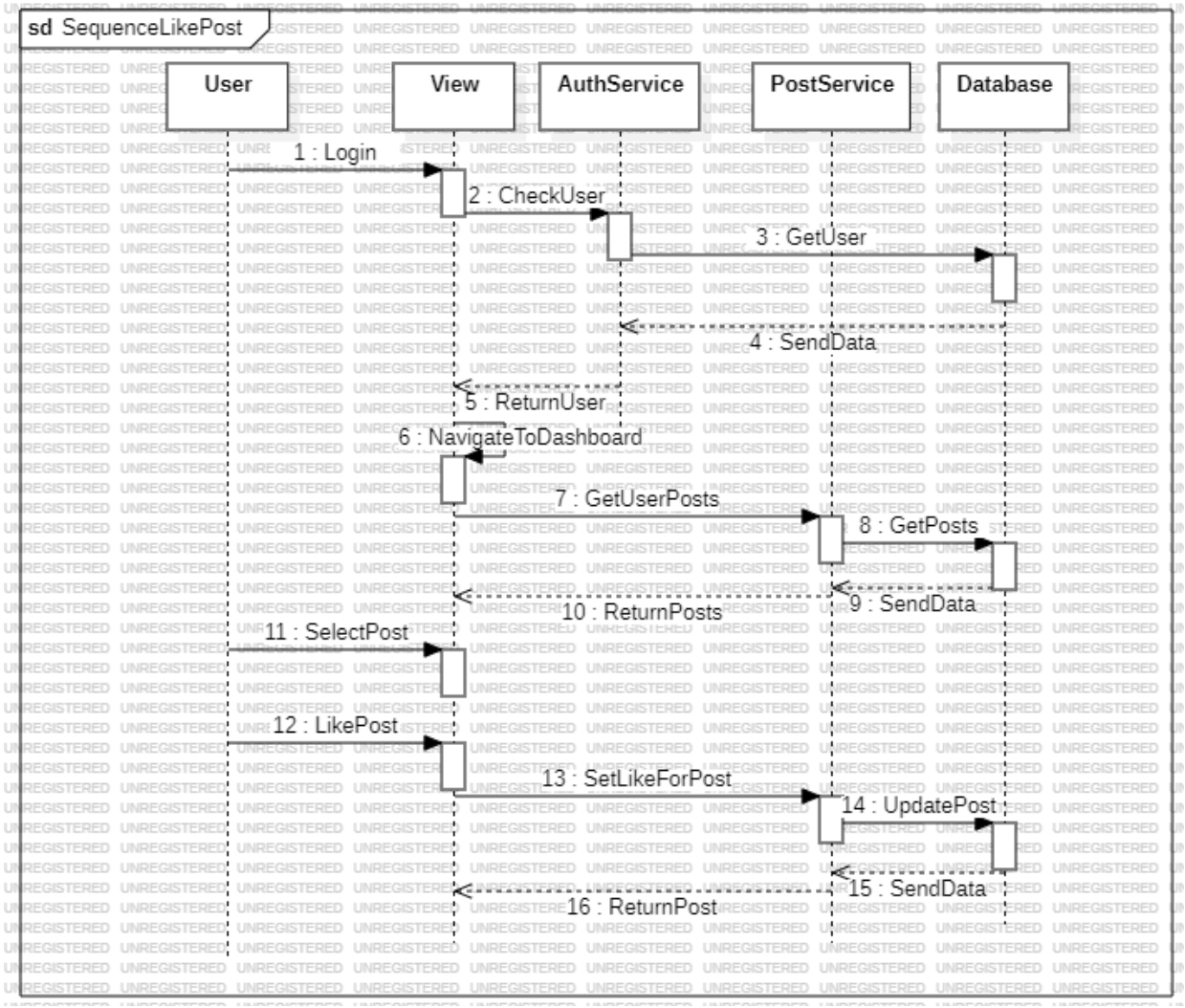

Рисунок 2.9 – Діаграма послідовності схвалення постів

На рис. 2.9 на діаграмі видно, що аби схвалити пост, користувачу необхідно лайкнути його, після чого відповідний сервіс його схвалить.

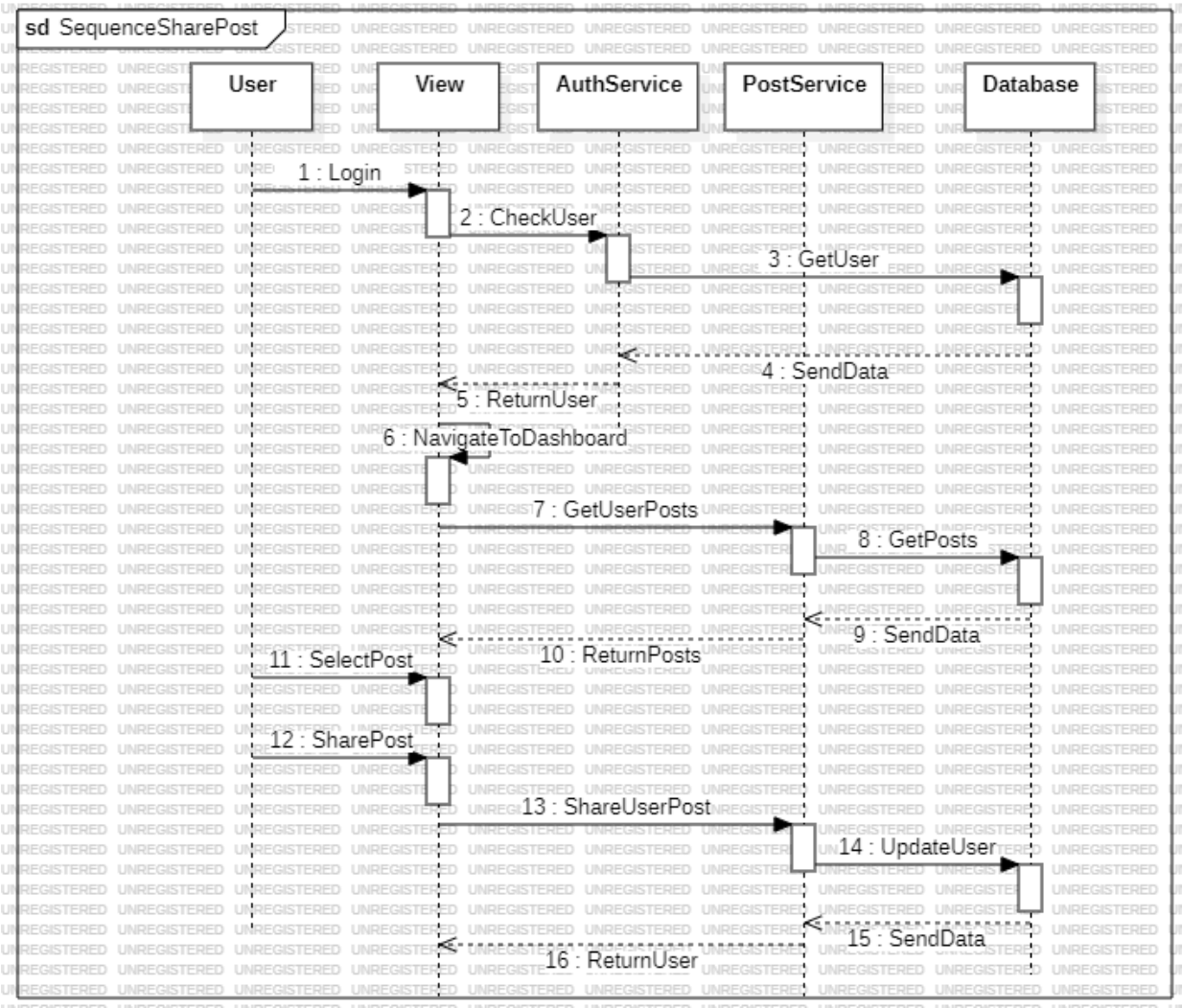

Рисунок 2.10 – Діаграма послідовності поширення постів

На рис. 2.10 на діаграмі видно, що аби поширити пост, користувачу необхідно обрати пост та виконати функцію поширення, після чого відповідний сервіс збереже пост до вашого профілю.

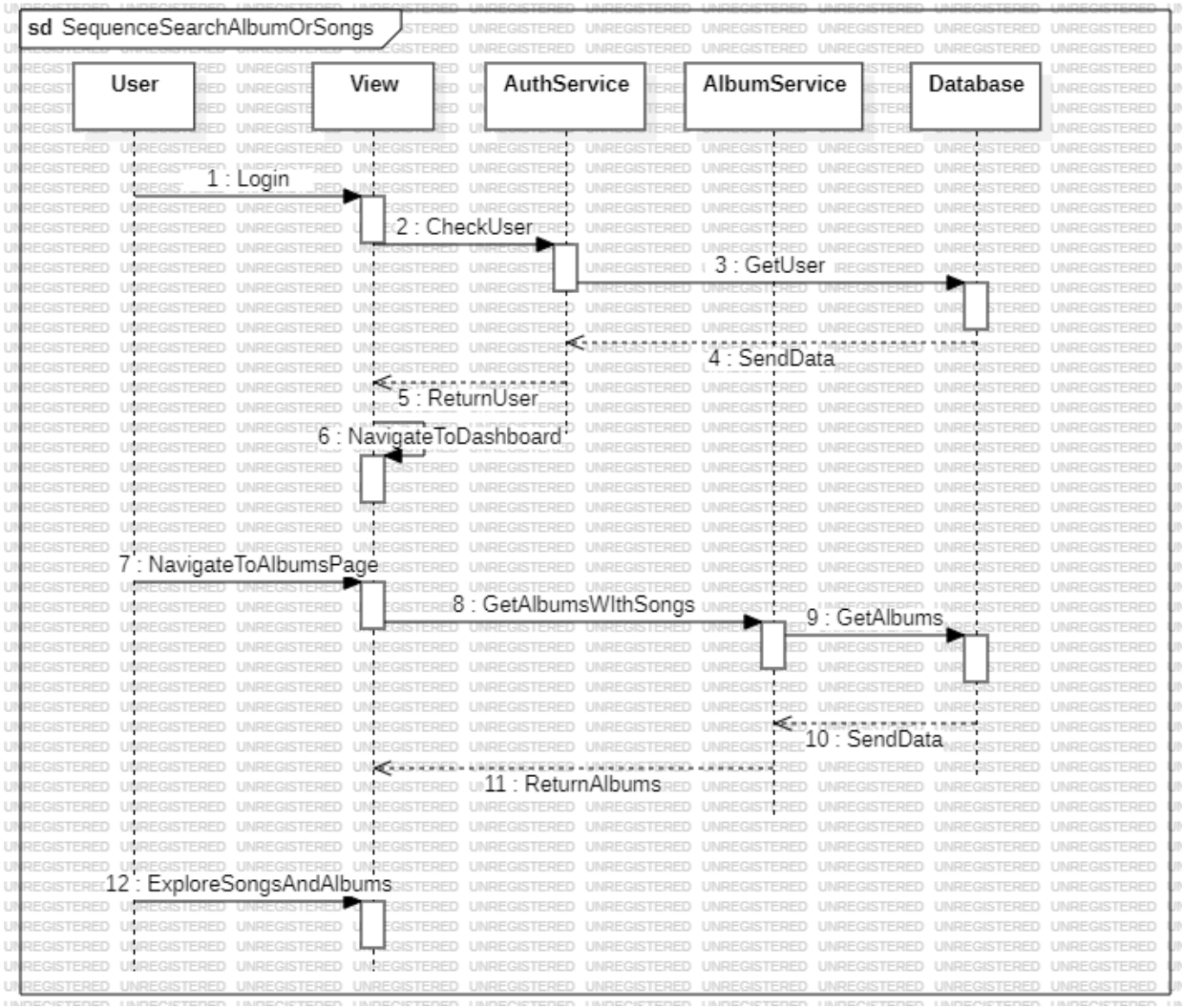

Рисунок 2.11 – Діаграма послідовності пошуку пісень та альбомів

На рис. 2.11 на діаграмі видно, що аби переглянути пісні та альбоми, користувачу необхідно зайти до бібліотеки, після чого відповідний сервіс надасть альбоми та пісні.

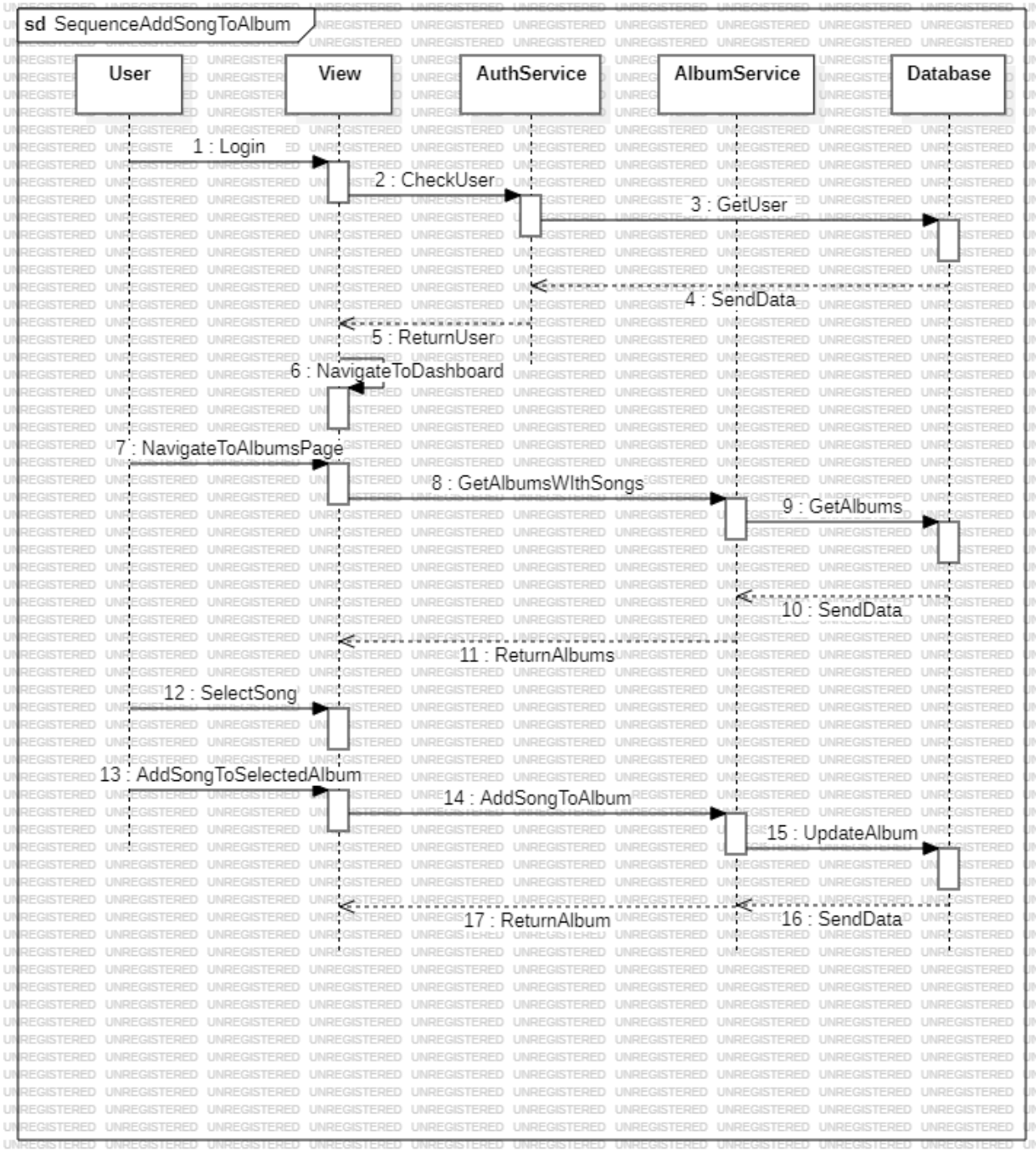

Рисунок 2.12 – Діаграма послідовності додання пісні до альбому

На рис. 2.12 на діаграмі видно, що аби додати пісню до альбому, користувачу необхідно обрати деяку пісню, після чого відповідний сервіс додасть її.

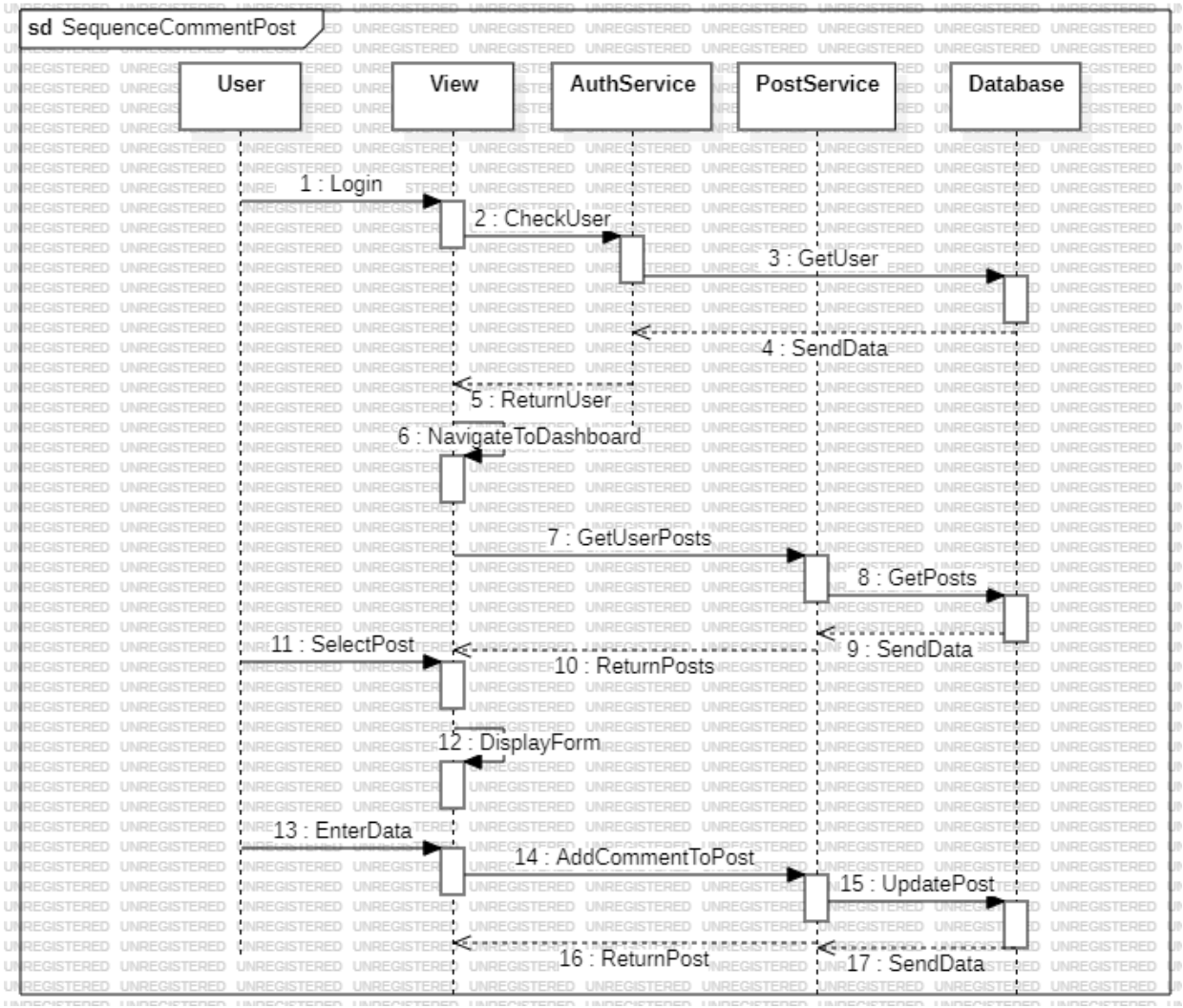

Рисунок 2.13 – Діаграма послідовності коментування постів

На рис. 2.13 на діаграмі видно, що аби коментувати деякий пост, користувачу необхідно обрати пост та відправити данні, після чого відповідний сервіс додасть ваш коментар до посту.

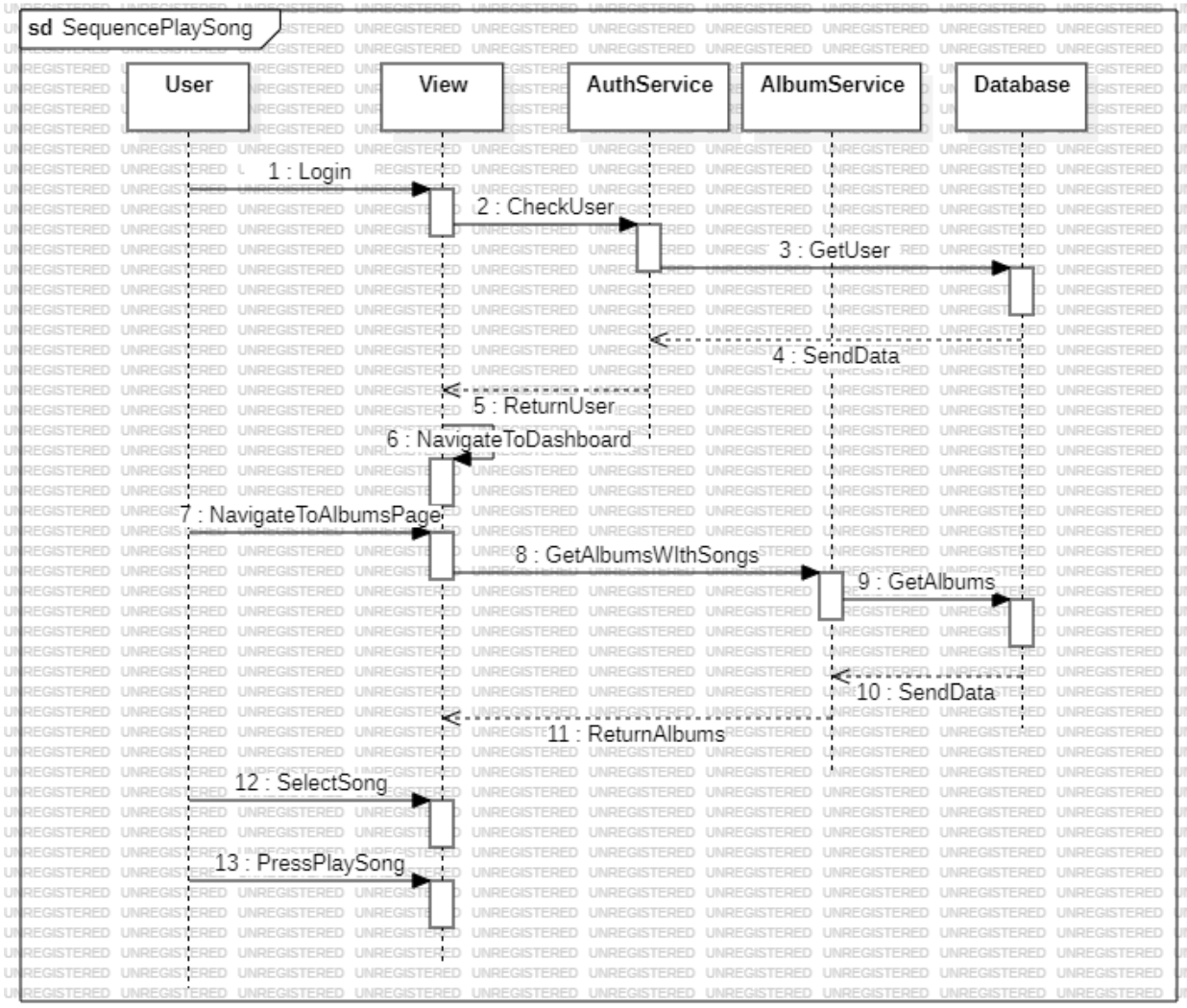

Рисунок 2.14 – Діаграма послідовності прослуховування пісень

На рис. 2.14 на діаграмі видно, що аби прослухати пісню користувачу необхідно обрати пісню після чого відповідний сервіс надасть вам пісню до прослуховування.

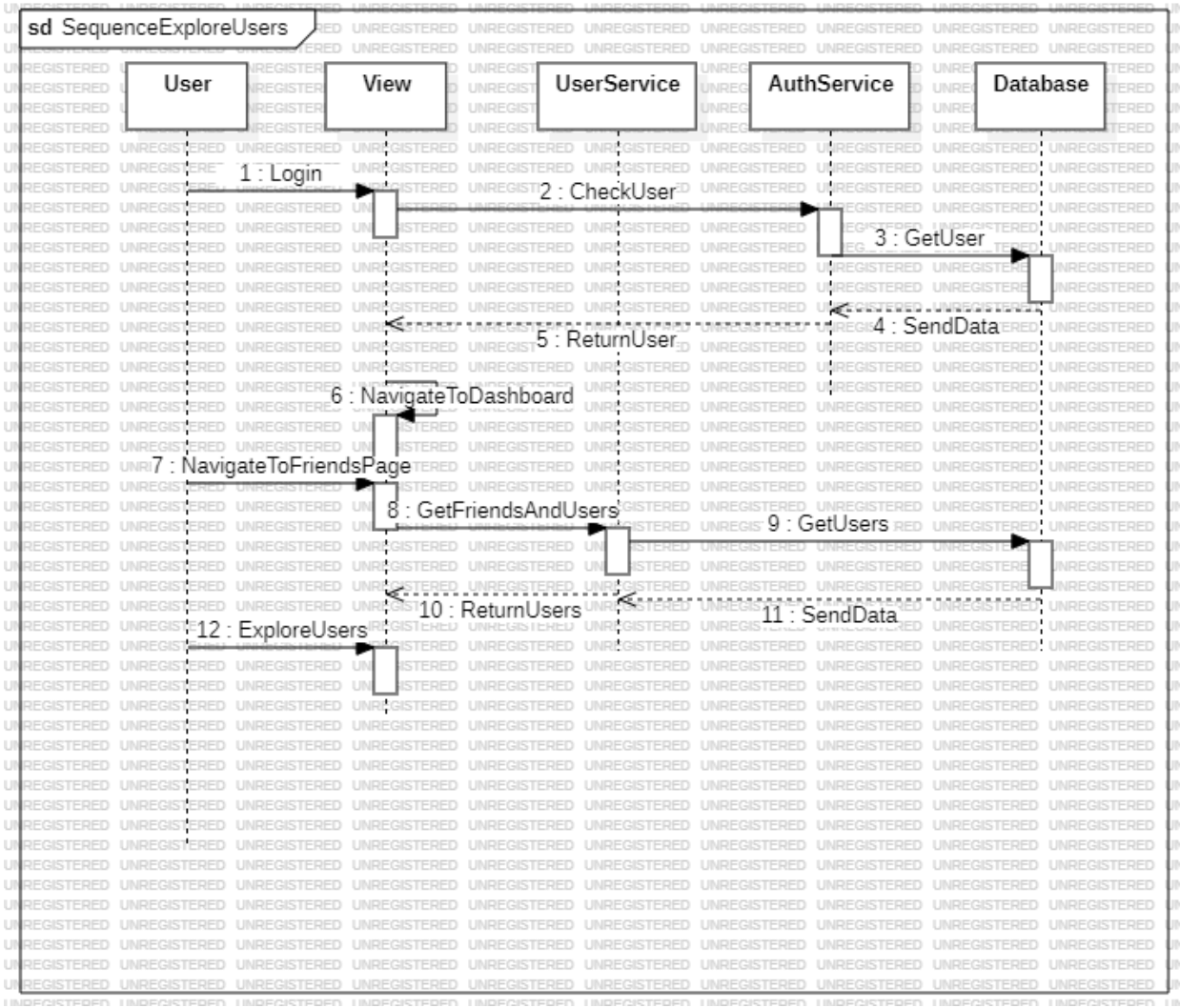

Рисунок 2.15 – Діаграма послідовності пошуку користувачів

На рис. 2.15 на діаграмі видно, що аби знайти користувача, вашому користувачу необхідно перейти до сторінки з користувачами, після чого відповідний сервіс надасть вам список з іншими користувачами.

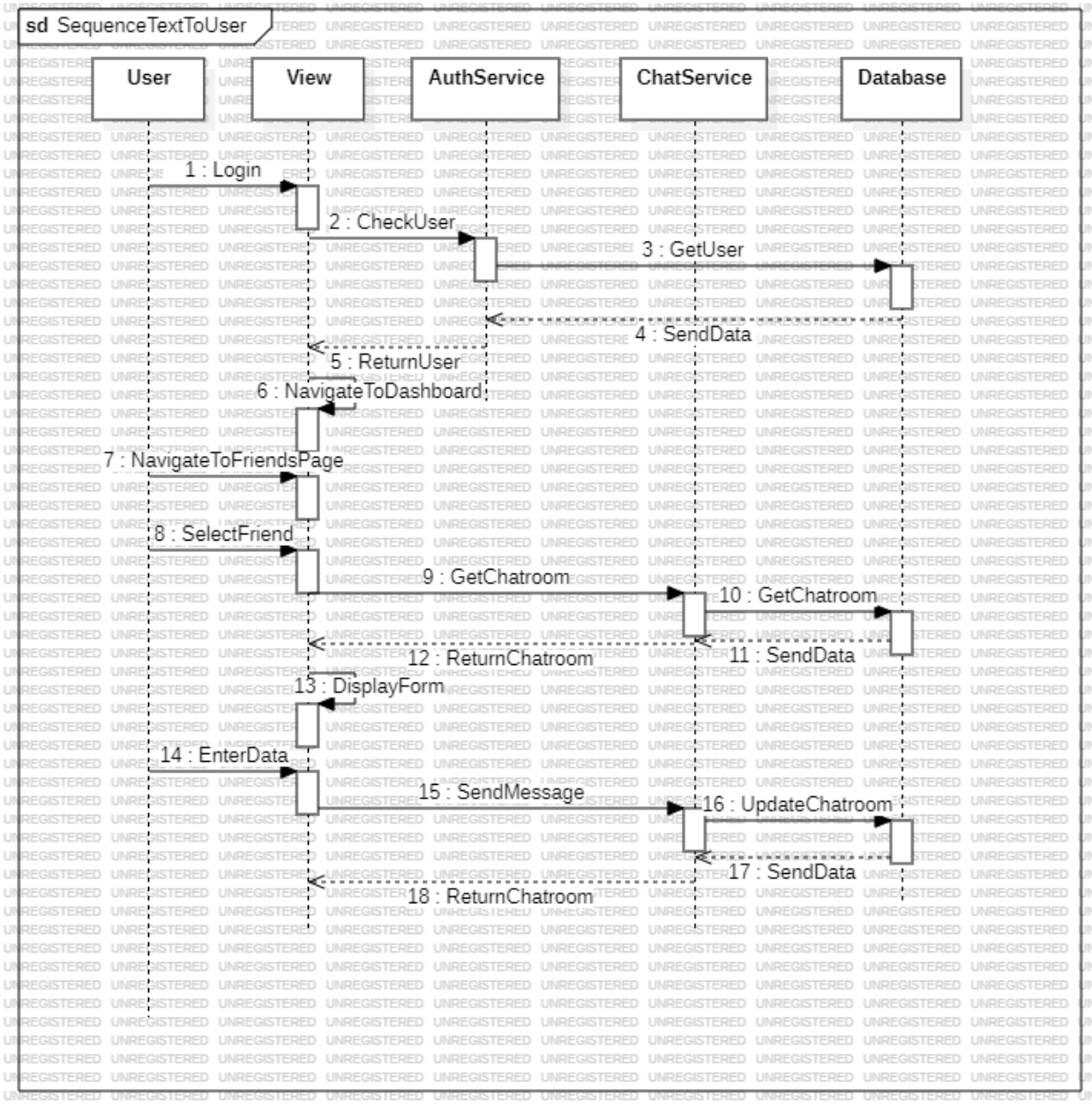

Рисунок 2.16 – Діаграма послідовності написання повідомлення користувачу

На рис. 2.16 на діаграмі видно, що аби надіслати повідомлення, користувачу необхідно ввести відповідні данні, після чого відповідний сервіс відправить ваше повідомлення.

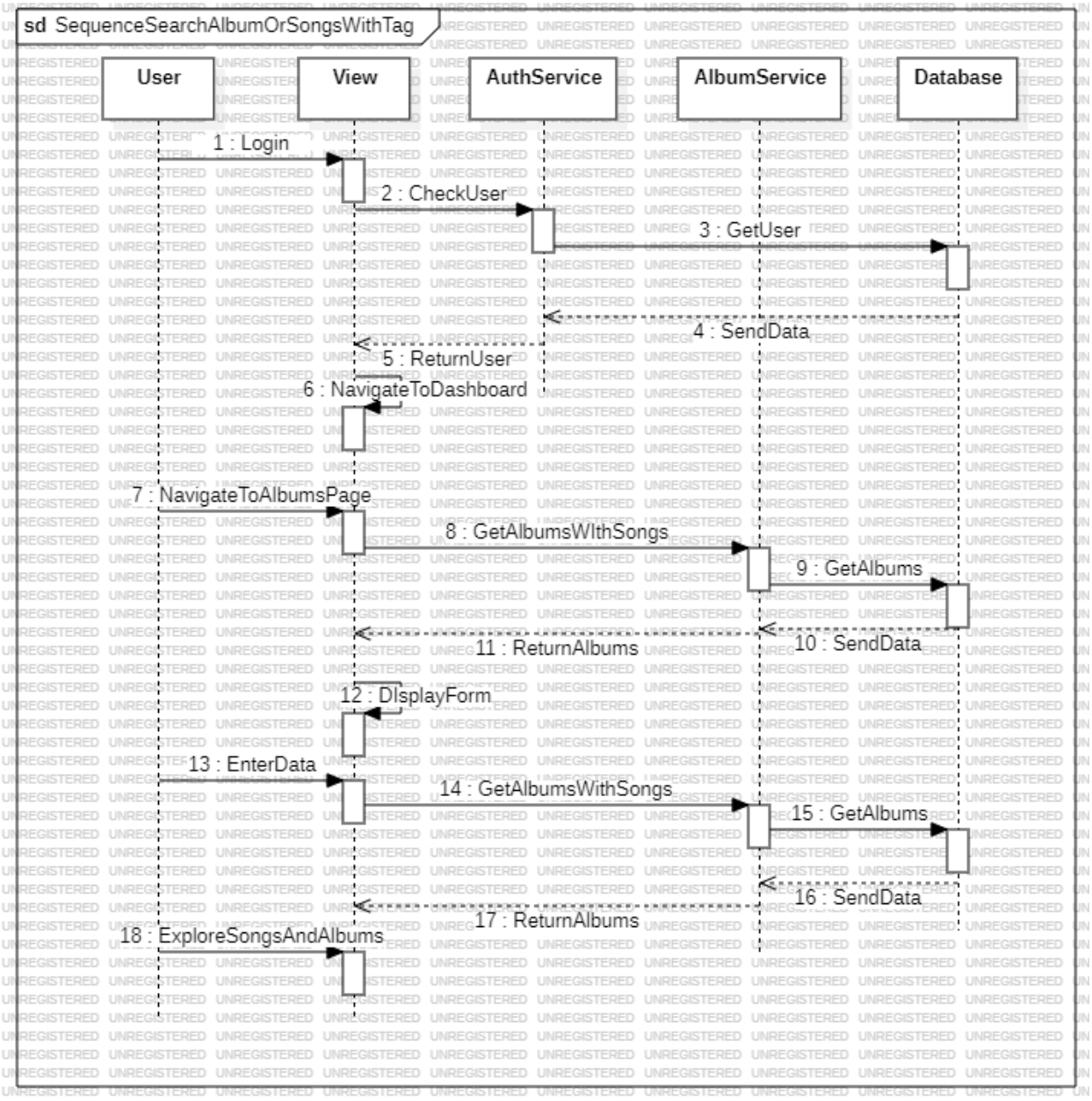

Рисунок 2.17 – Діаграма послідовності пошуку пісень за тегом

На рис. 2.17 на діаграмі видно, що аби знайти пісню за тегом користувачу необхідно ввести тег, після чого відповідний сервіс надасть вам бажані пісні.

Висновки до розділу 2

У другому розділі розроблені функціональні моделі для системи пошуку музичних партнерів та обміну композиціями згідно з вимогами, визначеними у

попередньому розділі. Основні рішення включають визначення ключових функцій системи, таких як реєстрація та аутентифікація користувачів, створення та редагування профілю, пошук музичних партнерів, обмін композиціями, комунікація в режимі реального часу та управління контентом. Ці функції були деталізовані через сценарії використання, що забезпечують наочне розуміння процесу взаємодії користувачів із системою.

Для представлення вимог були використані графічні моделі, зокрема діаграми використання, послідовності. Діаграми використання демонструють, як різні користувачі взаємодіють із системою та які функції вони можуть виконувати. Діаграми послідовності ілюструють порядок взаємодій між користувачами та системою під час виконання певних функцій.

Ці діаграми дозволили створити чітке уявлення про архітектуру та функціональність системи, що стало основою для подальшої розробки. Деталізовані сценарії використання та графічні моделі допомогли точно визначити, як користувачі взаємодіятимуть із системою, забезпечивши її зручність та ефективність. Такий підхід сприяв створенню надійного та масштабованого вебзастосунку, який відповідає потребам музикантів та артистів у пошуку партнерів та обміні композиціями.

3 КОНСТРУЮВАННЯ ВЕБЗАСТОСУНКУ ПОШУКУ МУЗИ ЧНИХ ПАРТНЕРІВ ТА ОБМІНКУ КОМПОЗИЦІЯМИ

3.1 Діаграма класів та компонентів

На основі сформованих сервісів та моделей у базі данних необхідно сформувати діаграму класів.

Діаграма класів є однією з найважливіших діаграм об'єктно-орієнтованого моделювання. Вона представляє структуру системи, показуючи класи, їх атрибути та методи, а також взаємозв'язки між ними. Кожен клас описує конкретний об'єкт або сутність у системі, а атрибути та методи визначають його властивості та поведінку.

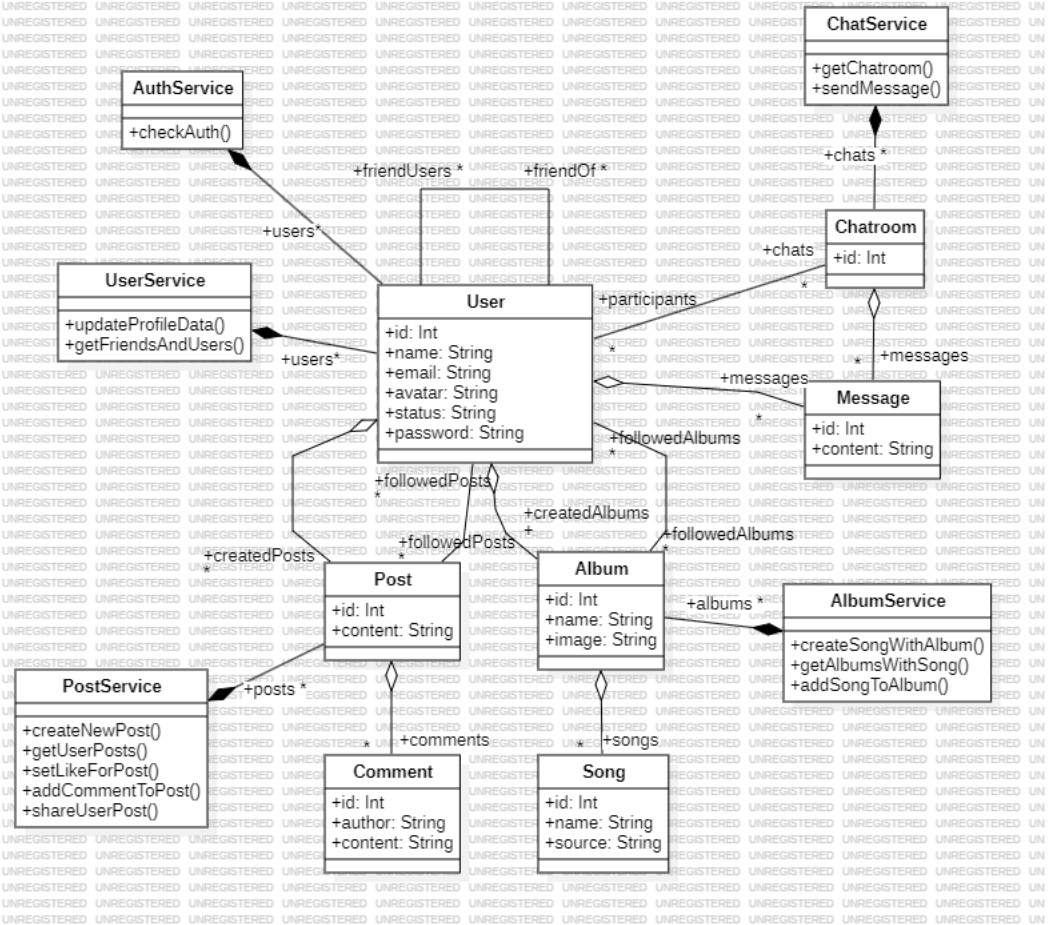

Рисунок 3.1 – Діаграма класів системи

Через сервіси з якими взаємодіє FE частина, такі як ChatService, AlbumService, PostService, UserService та AuthService, користувач маніпулює відповідними моделями: User, Chatroom, Message, Album, Song, Comment, Post.

Діаграма компонентів представляє архітектуру системи на вищому рівні, показуючи основні компоненти системи та взаємозв'язки між ними. Кожен компонент є самостійною частиною системи, що виконує певні функції або представляє конкретний модуль.

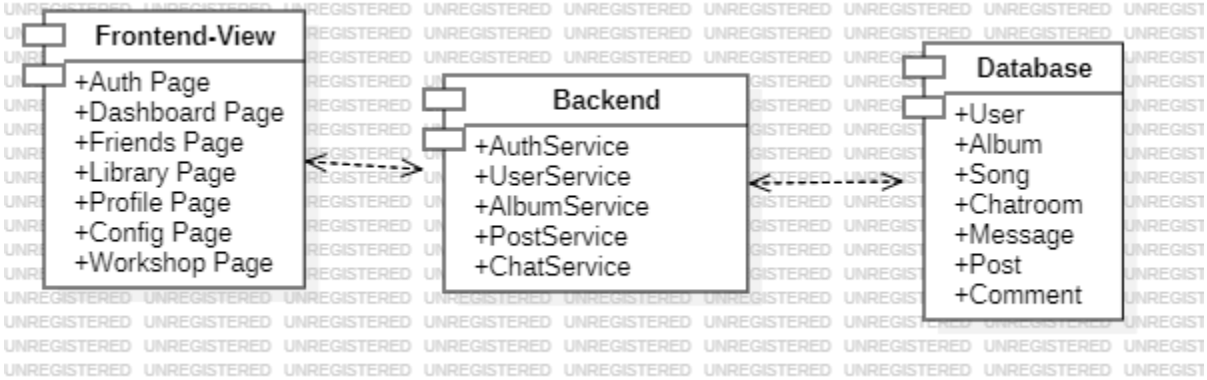

Рисунок 3.2 – Діаграма компонентів системи

Саме на цій діаграмі видно частину прдставлення для користувача сторінки. Серед них сторінка автинтефікації, дєшборду, сторінка з друзями, бібліотека, профіль користувачи, сторінка налаштування та сторінка майстерні. Крім того, на діаграмі видно вище згадані сервіси та моделі.

Тепер, коли сформовано основні вимоги до системи, необхідно створити інтерфейс для самого користувача. Розробка макетів інтерфейсу дозволить нам візуалізувати функціональність застосунку та забезпечити зручний та ефективний користувацький досвід. Ми сконцентруємося на створенні макетів ключових сторінок, які відображають основні функції системи, такі як пошук музичних партнерів, обмін композиціями та профіль користувача. Ці макети будуть використані як основа для подальшого розроблення та реалізації інтерфейсу застосунку.

Також, в процесі розробки будуть обрані мови програмування та технології, що відповідають вимогам проекту та сприятимуть його успішному втіленню. Вибір мов програмування та технологій буде здійснюватися з урахуванням вимог до функціональності та потреб користувачів, ефективності, масштабованості та зручності розробки та підтримки системи.

3.2 Розробка макетів інтерфейсу

Тепер, коли сформовано основні вимоги до системи, необхідно створити UI для самого користувача. Важливо, щоб інтерфейс був зручним, інтуїтивно зрозумілим та естетично привабливим. Для цього були розроблені макети інтерфейсу, які відображають усі ключові функції застосунку.

Відповідно до вимоги реалізації функціоналу з логіну та входу до свого аккаунту, було створено мокап логін сторінки. На ній користувач буде мати змогу увійти до свого аккаунту.

Рисунок 3.3 – Макет сторінки авторизації

Відповідно до вимоги реалізації функціоналу з реєстрації користувачів, було створено мокап сторінки реєстрації. На ній користувач буде мати змогу створити свій обліковий запис.

Рисунок 3.4 – Макет сторінки реєстрації

Відповідно до вимоги реалізації функціоналу з перегляду та коментування публікацій було створено мокап сторінки з постами. На ній користувач буде мати змогу переглядати та коментувати пости.

Рисунок 3.5 – Макет сторінки з постами

Відповідно до вимоги реалізації функціоналу з редагування власних даних, перегляду та створення публікацій, було створено мокап сторінки профілю. На ній

користувач буде мати змогу переглядати та створювати власні публікації а також налаштовувати свій профіль.

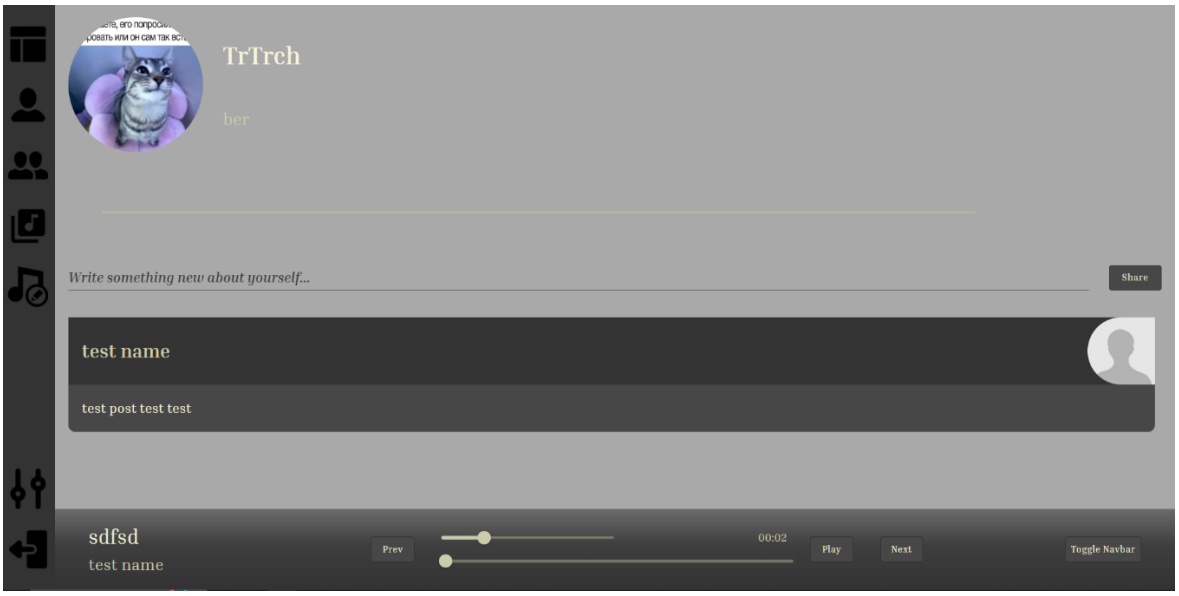

Рисунок 3.6 – Макет сторінки профілю

Відповідно до вимоги реалізації функціоналу з перегляду користувачів та додання їх да власних друзів, було створено мокап сторінки з друзями. На ній користувач буде мати змогу додавати собі друзів .

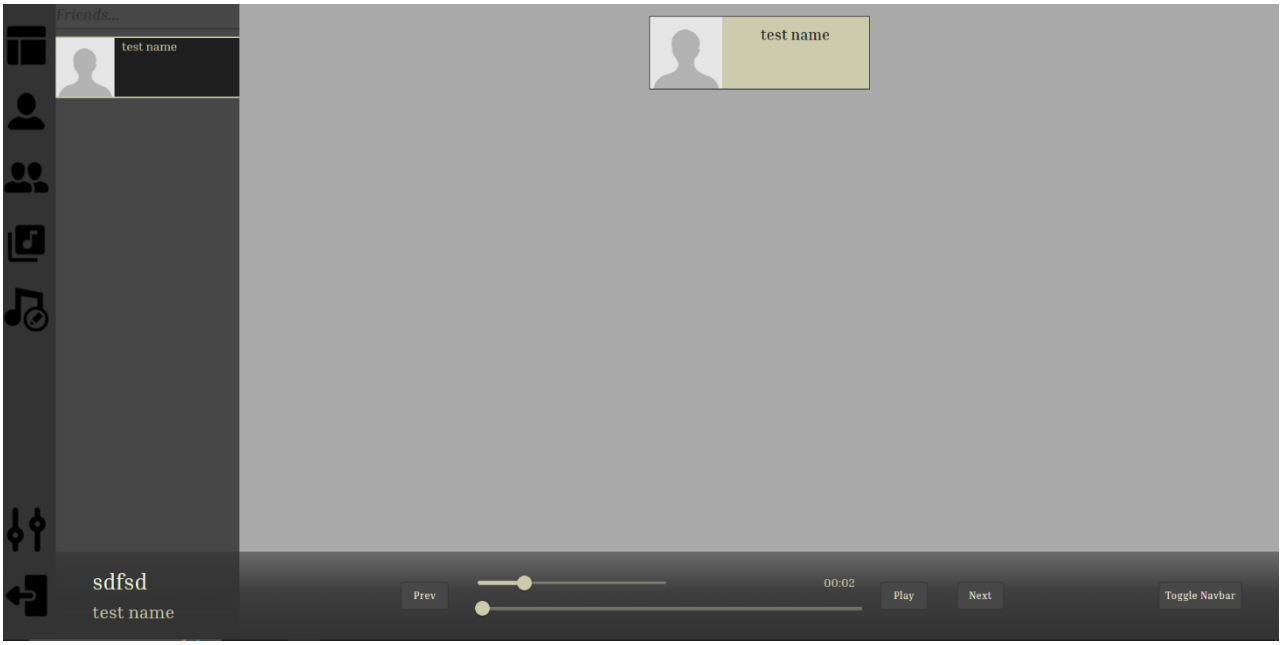

Рисунок 3.7 – Макет сторінки з друзями

Відповідно до вимоги реалізації функціоналу з переписок між користувачами було створено мокап чат сторінки. На ній користувач буде мати змогу переписуватись з іншими користувачами .

Рисунок 3.8 – Макет сторінки з чатом

Відповідно до вимоги реалізації функціоналу з прослуховування та перегляду альбомів та композицій, було створено мокап сторінки з піснями. На ній користувач буде мати змогу прослуховувати та взаємодії з композиціями та альбомами.

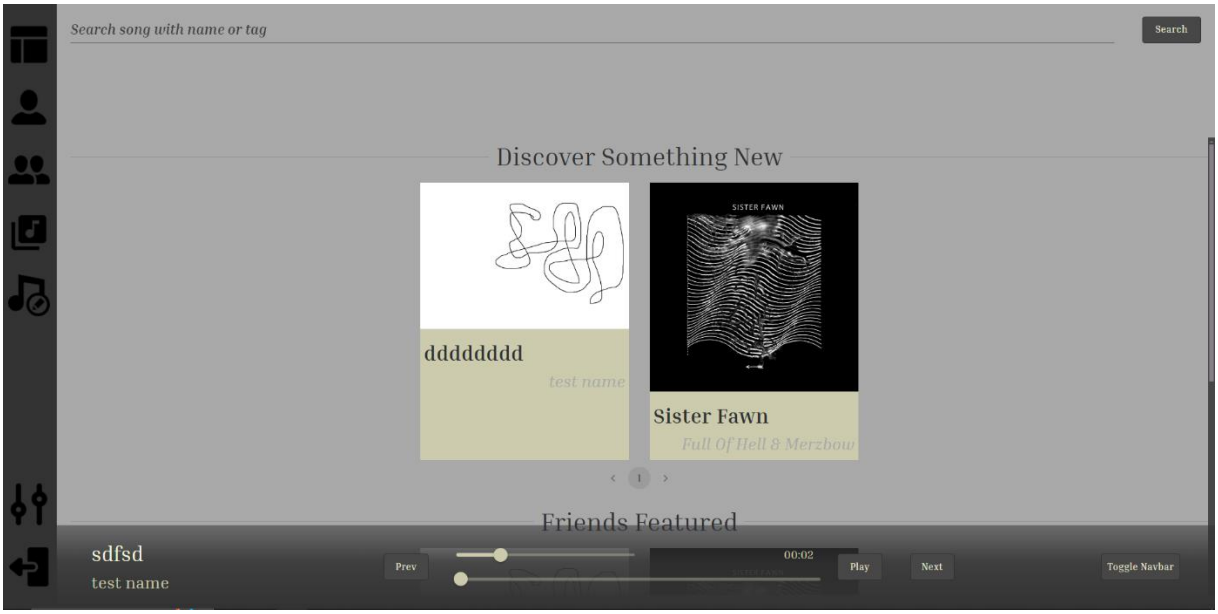

Рисунок 3.9 – Макет сторінки з піснями

Відповідно до вимоги реалізації функціоналу з прослуховування конкретних композицій було створено мокап сторінки відповідного альбому. На ній користувач буде мати змогу обрати та прослухати пісню.

Рисунок 3.10 – Макет попапу з піснями

Відповідно до вимоги реалізації функціоналу з публікацій альбомів та композицій, було створено мокап. Сторінки публікації альбому. На ній користувач буде мати змогу опублікувати альбом чи пісню.

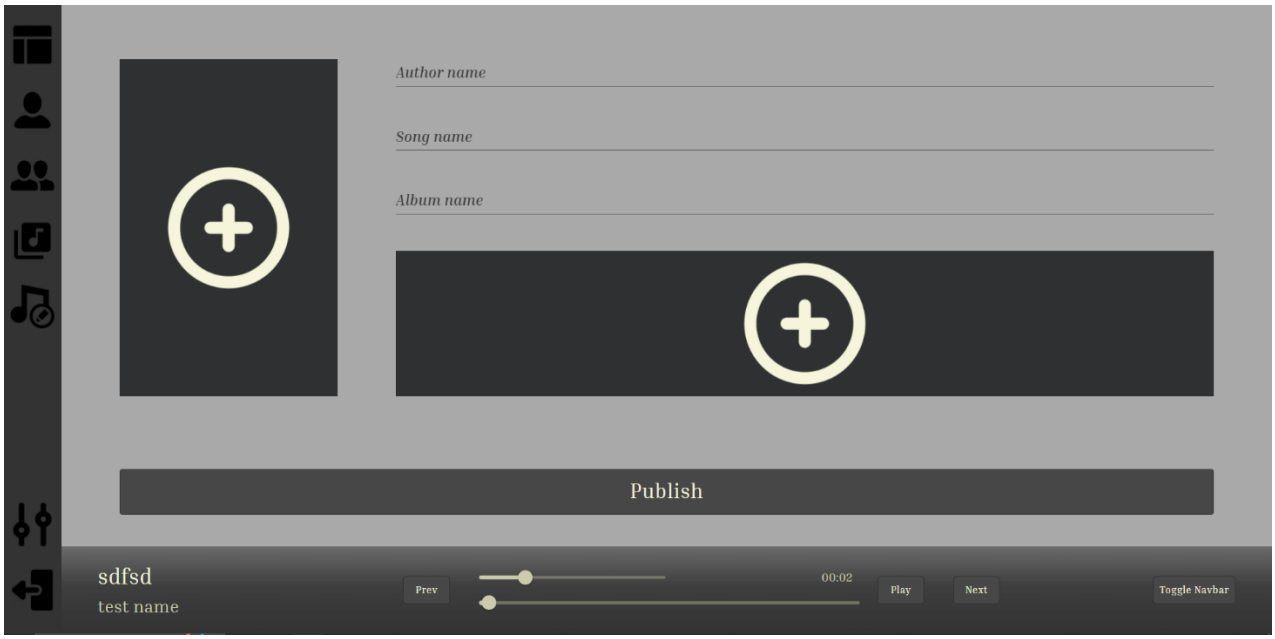

Рисунок 3.11 – Макет форми публікації альбому

Таким чином були створенні сторінки до застосунку, які реалізують основний його функціонал за поставленими вимогами.

3.3 Вибір технологій та мов програмування

Тепер, коли перед нами стоїть завдання розробити готовий інтерфейс та функціонал для нашого застосунку, ми звертаємось до потужних інструментів, які допоможуть нам втілити наші ідеї в реальність.

За основу розробки фронтенду ми обрали React [5]. Ця бібліотека визнана своєю високою швидкістю та ефективністю у створенні динамічних інтерфейсів. Завдяки їй ми зможемо створити зручні та інтуїтивно зрозумілі користувацькі інтерфейси, які забезпечать приємний досвід користувача.

Для керування станом додатка ми вибрали Redux [6] разом з Redux toolkit [7]. Цей інструмент дозволить нам ефективно управляти складним станом додатка та забезпечить передбачувану реакцію на дії користувачів. Його використання дозволить нам зробити наш застосунок більш масштабованим та підтримуваним.

У якості серверного фреймворка ми обрали Nest.js [9]. Цей фреймворк, побудований на основі Node.js [8], дозволить нам створити потужне та масштабоване API для нашого застосунку. Він забезпечить зручну структуру проекту та допоможе зберігати наш код організованим та легким для розуміння.

При роботі з базою даних ми використовуватимемо Prisma ORM [12]. Цей інструмент дозволить нам зручно взаємодіяти з базою даних та забезпечить безпеку наших даних. Завдяки йому ми зможемо швидко розробляти та модифікувати схему бази даних, не втрачаючи при цьому продуктивності та надійності.

Для збереження та управління даними ми обрали Supabase [10], що використовує SQLite [11]. Цей сервіс надає широкий спектр можливостей для роботи з базою даних та забезпечує високу надійність та безпеку наших даних. Використання Supabase дозволить нам швидко розгорнути наше ПЗ та зосередитися на його функціоналі, не турбуючись про інфраструктуру бази даних.

3.4 Створення та налаштування базового оточення для застосунку

Для створення та налаштування базового оточення для нашого застосунку ми використовуємо NX для організації монорепозиторію [21]. NX надає зручні інструменти для управління кількома додатками та бібліотеками в одному проекті, що дозволяє нам ефективно розробляти та підтримувати наш код.

Наш монорепозиторій містить два основних застосунки, для FE та BE частини, які будуть розроблятися окремо. Кожен з цих застосунків має свою власну структуру та функціонал, але вони можуть використовувати спільні бібліотеки та інструменти.

Для кожного з наших застосунків ми також створюємо по бібліотеці - одну для компонентів FE та іншу для компонентів BE. Це дозволяє нам ефективно організовувати та повторно використовувати код між різними частинами наших застосунків.

Для створених бібліотек ми також створюємо окрему бібліотеку для типів, DTO та інших спільних об'єктів, які можуть використовуватися в обох застосунках.

Окрім того, ми налаштовуємо змінні оточення для наших застосунків, щоб мати зручний та гнучкий спосіб керування конфігурацією на різних етапах розробки. Це дозволяє нам швидко переключатися між різними середовищами, такими як розробка, тестування та виробництво, зберігаючи при цьому консистентність у нашому коді.

Для забезпечення високої якості коду ми використовуємо лінтери, які допомагають виявляти та виправляти потенційні проблеми ще до того, як вони стануть причиною серйозних проблем. Це допомагає забезпечити чистоту та читабельність нашого коду, а також зменшує ймовірність виникнення помилок під час розробки.

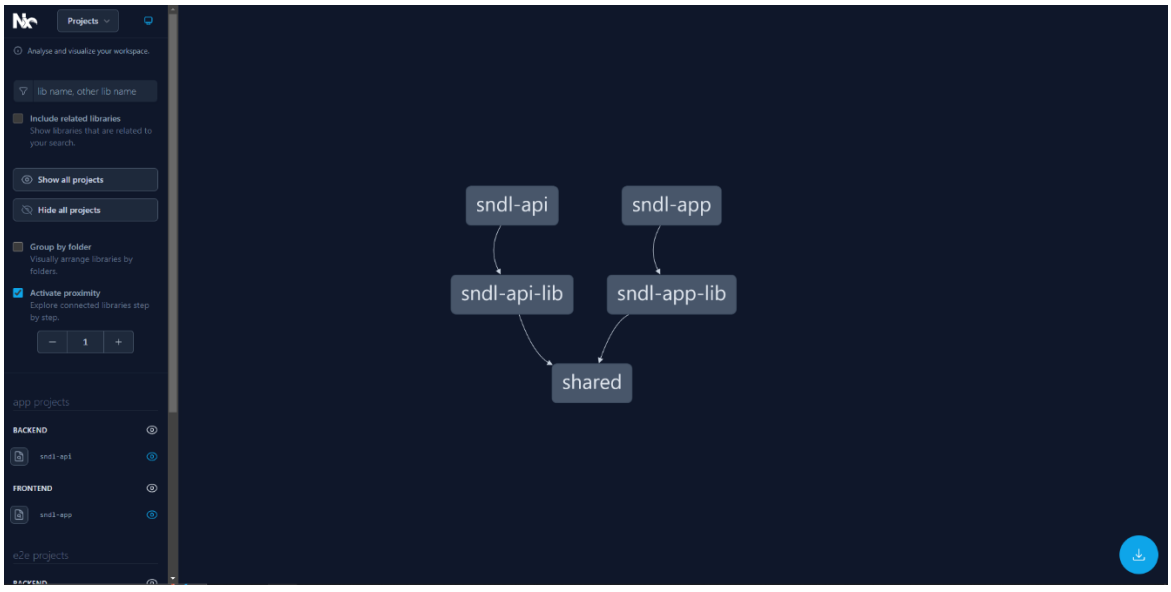

Рисунок 3.10 – NX діаграма залежностей бібліотек

NX також надає можливість переглядати створені бібліотеки, застосунки та їхні залежності, що також, допомагає зорієнтуватись у великих проектах.

Висновки до розділу 3

Цей розділ був присвячений створенню базового оточення для застосунку. Шлях до досягнення цієї мети включав вибір технологій, мов програмування та інструментів, які найкращим чином відповідають нашим потребам і сприятимуть успішному розвитку проекту.

Було обрано NX для організації монорепозиторію, оскільки він надає потужні засоби для управління кількома додатками та бібліотеками в одному проекті. Такий підхід дозволяє ефективно розробляти та підтримувати код, забезпечуючи його стабільність та швидкість розгортання.

Для розробки фронтенду обрано React, який є однією з найпопулярніших бібліотек для створення користувацьких інтерфейсів веб-додатків. Використання React дозволяє нам розробляти швидкі та інтерактивні інтерфейси, які забезпечать зручність користувачам нашого застосунку.

Для бекенду обрано NestJS, який є потужним фреймворком для розробки серверних додатків на Node.js. NestJS надає зручний та ефективний спосіб розробки складних серверних додатків, забезпечуючи високу продуктивність та надійність.

Для спільної роботи між фронтендом та бекендом був використан Redux, який є потужною бібліотекою для керування станом додатку, та Prisma ORM, яка надає зручний інтерфейс для роботи з DB.

Для збереження та обробки даних використовуємо Supabase як базу даних та сховище, що дозволяє нам ефективно управляти даними та забезпечувати їх безпеку.

Вибір цих технологій та інструментів дозволяє створити потужний та ефективний застосунок, який задовольнить потреби наших користувачів і забезпечить високу якість обслуговування.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ПОШУКУ МУЗИЧНИХ ПАРТНЕРІВ

Переходимо до етапу налаштування та створення основних компонентів нашого застосунку. Використовуючи NX для організації монорепозиторію, ми забезпечимо, щоб фронтенд та бекенд знаходилися в одному проекті, що значно спростить процес розробки та підтримки.

Перед тим як приступити до безпосередньої розробки, необхідно виконати важливий крок – налаштування бази даних. Це включає в себе створення та конфігурацію структури бази даних, а також забезпечення ефективних шляхів комунікації між базою даних та нашим бекендом. Використовуючи Prisma ORM, ми зможемо встановити зручний та безпечний зв’язок з базою даних, що дозволить нам ефективно керувати даними та забезпечити їхню цілісність.

Забезпечивши налаштування бази даних та встановивши шляхи для комунікації з бекендом, ми створимо стабільну основу для подальшої розробки нашого застосунку. Це дозволить нам зосередитися на розробці функціоналу та інтерфейсу, знаючи, що наша інфраструктура надійно підтримує всі необхідні процеси.

4.1 Створення та налаштування БД

Для налаштування Supabase спершу було зареєстровано новий проект на платформі. У налаштуваннях проекту знайдено URL-адреси для доступу до бази даних і сховища файлів, а також ключі для аутентифікації. Ці URL-адреси використовуються для встановлення з'єднання між бекендом і базою даних.

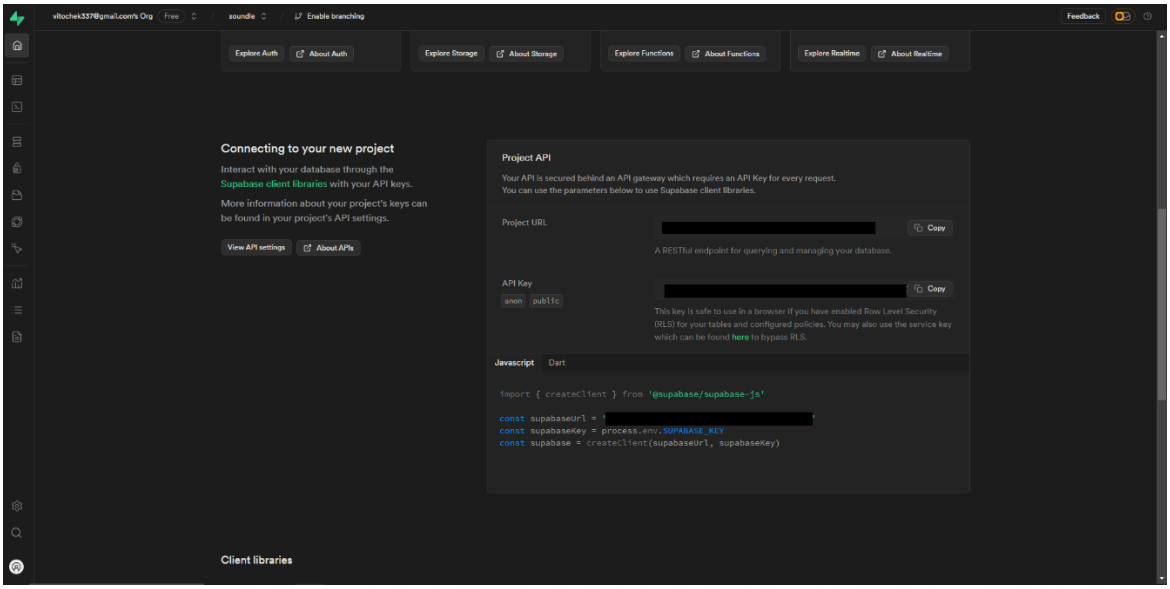

Рисунок 4.1 – Налаштування Supabase та отримання доступу

Після отримання URL-адреси бази даних налаштовано схему бази даних за допомогою Prisma ORM, визначивши необхідні таблиці та їхні взаємозв’язки, такі як User, Chatroom, Post, Comment, Album, Song та Message [14]. Це забезпечило структуроване зберігання та ефективне управління даними. У налаштуваннях проекту також знайдено URL-адресу для доступу до сховища файлів, яка використовується для завантаження та отримання файлів, таких як зображення і музичні композиції.

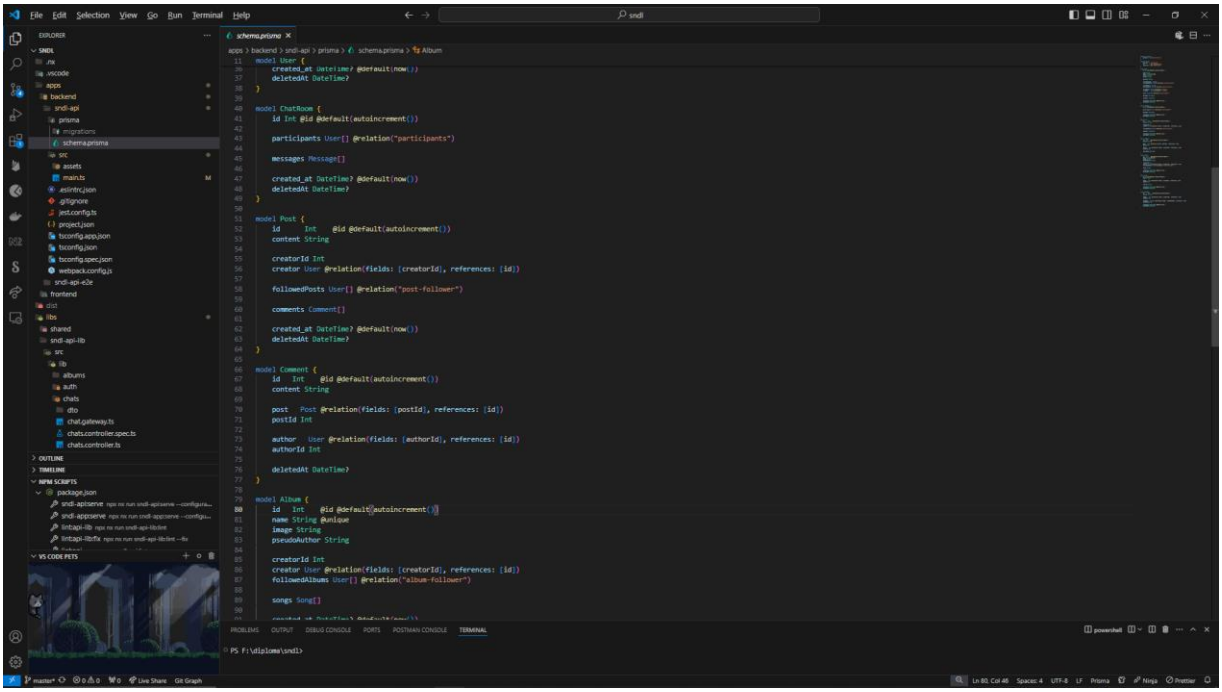

Рисунок 4.2 – Взаємозв’язки до бази даних

Для захисту файлів було налаштовано права доступу в сховищі, що забезпечує безпечне завантаження та зберігання даних. Щоб гарантувати безпеку та аутентифікацію, у налаштуваннях проекту отримано необхідні API ключі та токени аутентифікації. Ці ключі дозволяють безпечно взаємодіяти з базою даних та сховищем файлів. Їх було додано до змінних середовища у проекті, що дозволяє використовувати їх безпосередньо у коді, забезпечуючи безпечний доступ до ресурсів Supabase та захист від несанкціонованого доступу.

Цей процес налаштування Supabase забезпечує надійну основу для роботи з даними та файлами у застосунку, включаючи отримання та конфігурацію URL-адрес, налаштування прав доступу і безпеки, а також використання токенів аутентифікації для захисту ресурсів.

4.2 Створення та налаштуваня бекенду

Основною технологією для бекенд-розробки було обрано NestJS. NestJS - це прогресивний фреймворк для Node.js, який використовує TypeScript та дозволяє

створювати високопродуктивні серверні додатки. Він забезпечує зручний модульний підхід до організації коду, що сприяє легкості у підтримці та розширенні проєкту. Для початку, створимо єдиний модуль, що міститиме інші модулі. Це зробить структуру проєкту зручнішою та легшою для управління.

Також налаштуємо підтримку multipart-запитів, щоб наш застосунок міг приймати файли у форматі multipart/form-data. Це дозволить обробляти запити, що містять зображення та музичні композиції. Для цього інтегруємо відповідні middleware, які будуть обробляти такі запити та забезпечать коректне збереження файлів у сховищі.

Для оптимізації роботи бекенду було використано Fastify у поєднанні з NestJS. Fastify, завдяки своїй високій продуктивності та ефективному управлінню ресурсами, значно прискорив обробку запитів та зменшив затримки [15]. Використання Fastify з NestJS забезпечило стабільність та високу швидкість роботи серверної частини, що особливо важливо для реального часу та масштабованих додатків.

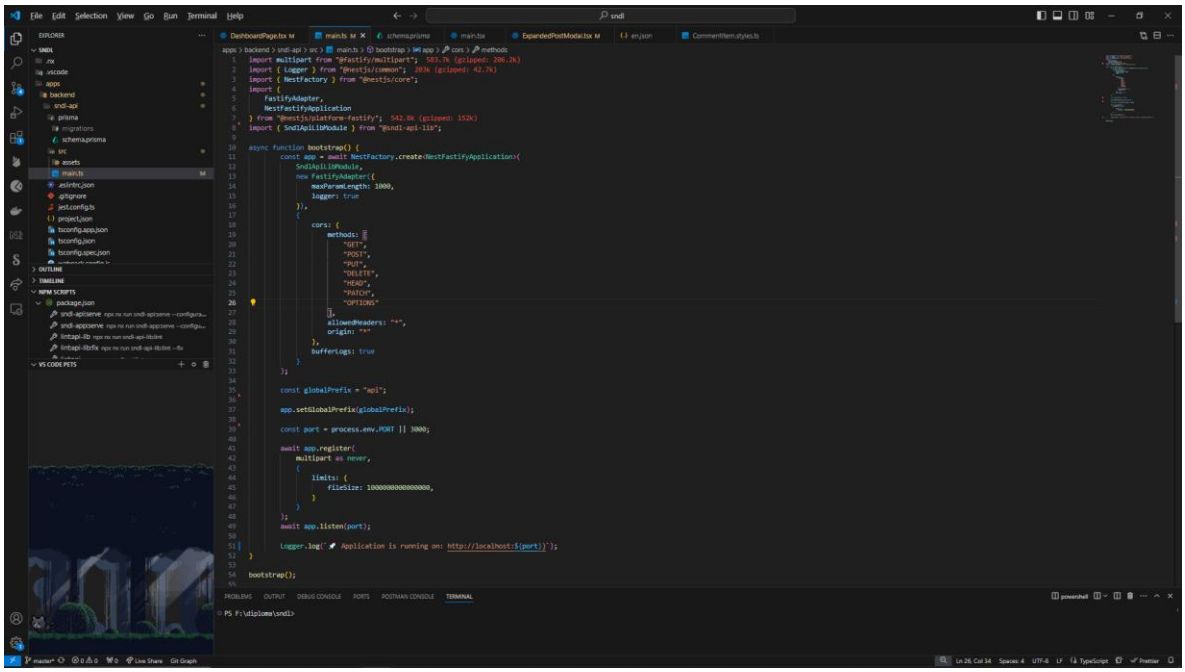

Рисунок 4.3 – Базові налаштування Nest JS

Отже, створивши єдиний модуль для зручності структурування та налаштувавши підтримку multipart-запитів, ми забезпечимо гнучкість і функціональність нашого застосунку. Це дозволить користувачам завантажувати зображення та пісні, а також зберігати їх у безпечний спосіб.

Надалі буде йти мова про відповідньо необхідні сервіси. Білша частина з них не має у створенні нічого складного, але на деякі з них варто звернути увагу.

Сервіс для автинтефікації

Спершу реалізовано ендпоінти для логіну та реєстрації. Користувачі зможуть зареєструватися, надавши необхідну інформацію, таку як ім'я, електронна адреса та пароль. Після реєстрації вони зможуть увійти в систему, отримавши токен доступу, який використовуватиметься для аутентифікації в наступних запитах.

Створено захист за допомогою JWT та Passport [9] з використанням бібліотеки BCrypt [18]. Він перевіряє, чи містить запит токен і чи є цей токен валідним. Це забезпечує, що лише авторизовані користувачі мають доступ до захищених ресурсів і функцій системи. Однак, для запитів логіну та реєстрації ця перевірка не потрібна, оскільки саме через ці запити користувачі отримують свої токени.

Таким чином, налаштовано базову безпеку для нашого застосунку, забезпечуючи аутентифікацію користувачів через логін і реєстрацію та захист доступу до інших ресурсів за допомогою JWT та Passport.

Сервіс для користувачів

Додамо основні ендпоінти для взаємодії з користувачами та управління профілями. Першим ендпоінтом є отримання списку всіх користувачів, що дозволить нашим користувачам легко знаходити та переглядати профілі інших музикантів. Це важлива функція для розширення мережі контактів та пошуку потенційних партнерів для співпраці.

Другий ендпоінт надає можливість отримувати інформацію про конкретного користувача. Це дозволяє детально ознайомитися з профілем та творчістю

потенційного партнера, що може бути корисним для прийняття рішення щодо співпраці.

Третім ендпоінтом є отримання списку друзів користувача. Ця функція спрощує взаємодію між музикантами, дозволяючи користувачам швидко переглядати своїх друзів і підтримувати з ними контакт.

Запит на додавання друзів є четвертим ендпоінтом, який дозволяє користувачам надсилати запити іншим музикантам для додавання їх до списку друзів. Це сприяє розширенню соціальної мережі та полегшує пошук нових партнерів для музичних проєктів.

Нарешті, ендпоінт для редагування власного профілю дозволяє користувачам оновлювати свою інформацію, включаючи завантаження зображень. Підтримка multipart/form-data забезпечує можливість прийому зображень та інших файлів, що є важливою частиною персоналізації профілю. Це гарантує, що профілі користувачів завжди містять актуальну інформацію та зображення, що покращує взаємодію та візуальне представлення користувачів на платформі.

Сервіс для публікацій

Створимо сервіс для управління постами. Перший ендпоінт дозволяє користувачам отримувати список всіх постів на платформі. Це дозволяє їм переглядати різноманітні контенти, які додають інші користувачі.

Другий ендпоінт надає можливість отримувати список постів, які користувачі зберегли у своєму профілі або які вони зашейрили. Це допомагає зберігати цікаві пости та ділитися ними з іншими.

Третій ендпоінт дозволяє користувачам додавати коментарі до існуючих постів. Це сприяє активній взаємодії та обговоренню контенту, що публікується на платформі.

Четвертий ендпоінт дозволяє отримувати інформацію про конкретний пост за його ідентифікатором. Це дає можливість користувачам переглядати деталізовану інформацію про окремі пости.

П'ятий ендпоінт надає можливість користувачам шейрити обраний пост у своєму профілі. Це дозволяє розповсюджувати цікавий контент серед своїх підписників та друзів.

Нарешті, шостий ендпоінт дозволяє користувачам створювати нові пости на платформі. Це ключова функція для висловлення творчих ідей, думок та вражень у вигляді тексту, зображень чи відео.

Завдяки цим ендпоінтам користувачі можуть активно спілкуватися, ділитися контентом та висловлювати свої думки та ідеї на платформі.

Сервіс для альбомів

Створимо сервіс для управління альбомами. Перший ендпоінт дозволяє користувачам отримувати список всіх альбомів, які вони зберегли у своєму профілі. Це дозволяє їм швидко знаходити та переглядати улюблені музичні збірки.

Другий ендпоінт надає можливість користувачам отримувати список нових альбомів на платформі. Це дозволяє вони завжди бути в курсі останніх музичних випусків.

Третій ендпоінт дозволяє користувачам отримувати список альбомів, які недавно були додані їхніми друзями. Це стимулює спільність та обмін музичними враженнями між користувачами.

Четвертий ендпоінт дозволяє користувачам завантажувати пісні до існуючого або нового альбому на платформі. Для цього вони можуть відправити файли з музикою та зображенням обкладинки альбому через мультіпарт.

П'ятий ендпоінт дозволяє користувачам додавати вже завантажені пісні до існуючого альбому на платформі. Це дозволяє створювати комплексні музичні збірки, додавати нові пісні та редагувати вміст альбому.

Нарешті, шостий ендпоінт надає можливість користувачам отримувати інформацію про конкретний альбом за його ідентифікатором. Це дає можливість переглядати деталізовану інформацію про музичні збірки та їхні складові.

Перші три ендпоінти також включають пагінацію для зручності користувачів. Це дозволяє виводити результати пошуку альбомів по частинах, щоб уникнути перевантаження великою кількістю даних та забезпечити швидке завантаження сторінок. Користувачі можуть переходити між сторінками результатів для отримання повного списку альбомів, які вони шукають.

Сервіс для обміну повідомленнями

У сервісі чату наявні ендпоінти для отримання всіх повідомлень між двома користувачами. Однак, особливість полягає в тому, що для реального часу ми використовуємо вебсокети. Вебсокети є протоколом зв'язку між клієнтом та сервером, який дозволяє встановлювати стале двонаправлене з'єднання через веб-протокол HTTP. Це означає, що сервер може ініціювати передачу даних клієнту без очікування запиту від клієнта.

Це передбачає двонаправлене з'єднання між сервером та клієнтом, що дозволяє відправляти та отримувати повідомлення майже миттєво, без необхідності постійних запитів до сервера. Такий підхід забезпечує швидку та ефективну комунікацію між користувачами у чаті.

4.3 Створення та налаштування фронтенду

Тепер, коли ми завершили розробку всіх потрібних сервісів з відповідними ендпоінтами на бекенді, настав час перейти до розробки фронтенду та його підключення до бекенду. Фронтенд буде відповідати за візуальне представлення даних та взаємодію з користувачем, а також за виконання запитів до нашого сервера через розроблені ендпоінти. Тож перейдемо до наступного етапу розробки, де ми зосередимося на створенні фронтенду за допомогою React JS та його інтеграції з

нашим бекендом. Розглянемо детальніше компоненти, що були використані для побудови основи клієнтської частини.

Реакт - це інструмент, що пропонує нам неймовірні можливості для створення динамічних та інтерактивних веб-додатків. Він забезпечує можливість створювати компоненти, які реагують на зміни стану та автоматично оновлюють інтерфейс, що робить користувальницький досвід більш плавним та привабливим.

Для організації навігації та маршрутизації використовується React Router [11], що дозволяє перехідити між сторінками додатка без перезавантаження. Це зробило навігацію в нашому додатку зручною та інтуїтивно зрозумілою для користувача.

Redux разом з Axios становлять невід'ємну частину нашого проекту, дозволяючи нам ефективно керувати станом додатка та взаємодіяти з сервером [20]. Redux забезпечує централізоване зберігання стану, що дозволяє легко відстежувати та змінювати дані, а Axios допомагає нам здійснювати HTTP-запити для обміну даними з сервером.

Графічна бібліотека Material UI допомагає нам швидко створювати красивий та сучасний інтерфейс, використовуючи готові компоненти та стилізацію [15]. Це дозволяє нам ефективно працювати над виглядом та взаємодією з користувачем, зосереджуючись на функціональності.

Intl використовується для міжнародного перекладу текстів, що дозволяє нашому додатку бути доступним для широкої аудиторії користувачів з різних країн та мовних середовищ [19].

Persist Store допомагає нам зберігати токен автентифікації в локальному сховищі користувача, забезпечуючи постійний доступ до додатка без необхідності повторно автентифікуватися після кожного перезавантаження сторінки.

Socket.IO використовується для реалізації RTE у нашому додатку. Це дозволяє нам взаємодіяти з сервером у режимі реального часу через веб-сокети, що дозволяє миттєво отримувати оновлення та повідомлення.

Snackbar використовується для відображення повідомлень про помилки та іншої важливої інформації, яка повертається з сервера. Це робить процес взаємодії з додатком більш інтуїтивним та приємним для користувача, дозволяючи швидко реагувати на помилки та надавати важливі сповіщення.

У процесі розробки фронтенду було також використано Vite.js – сучасний інструмент для створення та оптимізації вебдодатків [12]. Vite.js забезпечив швидкий запуск проєкту та гаряче перезавантаження, що значно прискорило процес розробки та тестування. Використання Vite.js дозволило зменшити час компіляції та зробити процес розробки більш ефективним та зручним.

Тепер, коли основа ясна, перейдемо до основної частини – побудованих сторінок.

Сторінки автинефікації

На сторінці логіну та реєстрації, що були створені відповідно до дизайну у фігмі, ми використовували бібліотеку Formik [16] разом з валідатором Yup [17] для створення форм введення даних. Це дозволило нам зручно взаємодіяти з користувачем та забезпечити валідацію введених даних на клієнтській стороні.

Форми були підключені до відповідних ендпоінтів для обробки логіну та реєстрації. Після успішної автентифікації користувача, ми автоматично перенаправляли його на сторінку з постами, де він міг продовжити взаємодію з додатком.

Сторінка з постами

На сторінці з постами ми створили інтерфейс, що дозволяє користувачам з легкістю переглядати та взаємодіяти з постами. Це включало в себе використання відповідних ендпоінтів для отримання списку постів з сервера. При кожному пості було реалізовано модальне вікно, яке надавало додаткову інформацію про пост та його зміст.

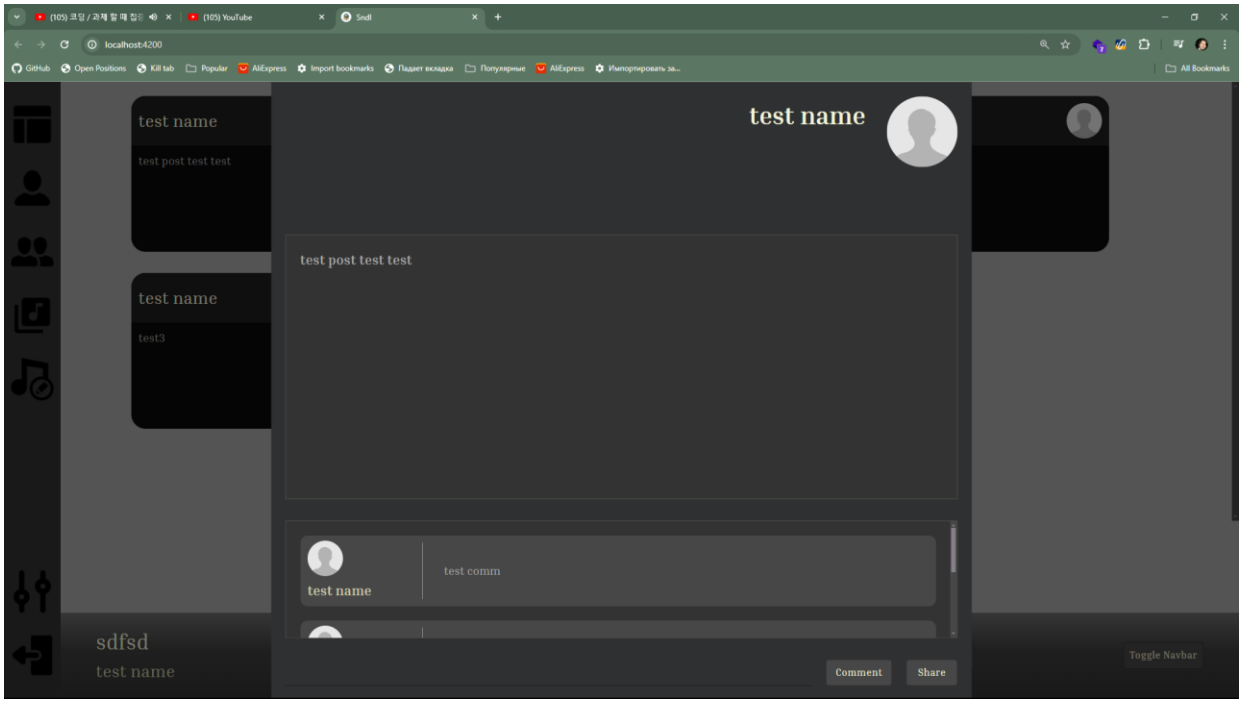

Рисунок 4.4 – Попап з детальним оглядом публікації

Модальне вікно дозволяло користувачам переглядати повну інформацію про пост, включаючи текст, зображення та інші деталі. Крім того, вони могли поділитися постом у своєму профілі або залишити коментар прямо з цього вікна, що забезпечувало зручну взаємодію та спілкування на платформі. Такий підхід дозволяв користувачам вільно висловлювати свої думки та реагувати на контент, не виходячи з основної сторінки з постами.

Сторінка з профілем

На сторінці профілю ми забезпечили користувачам можливість переглядати та редагувати свою особисту інформацію. Це включало в себе відображення даних про користувача, таких як ім'я, аватар, дата народження тощо. Крім того, користувачі мали можливість змінювати ці дані, якщо вони бажали оновити свій профіль.

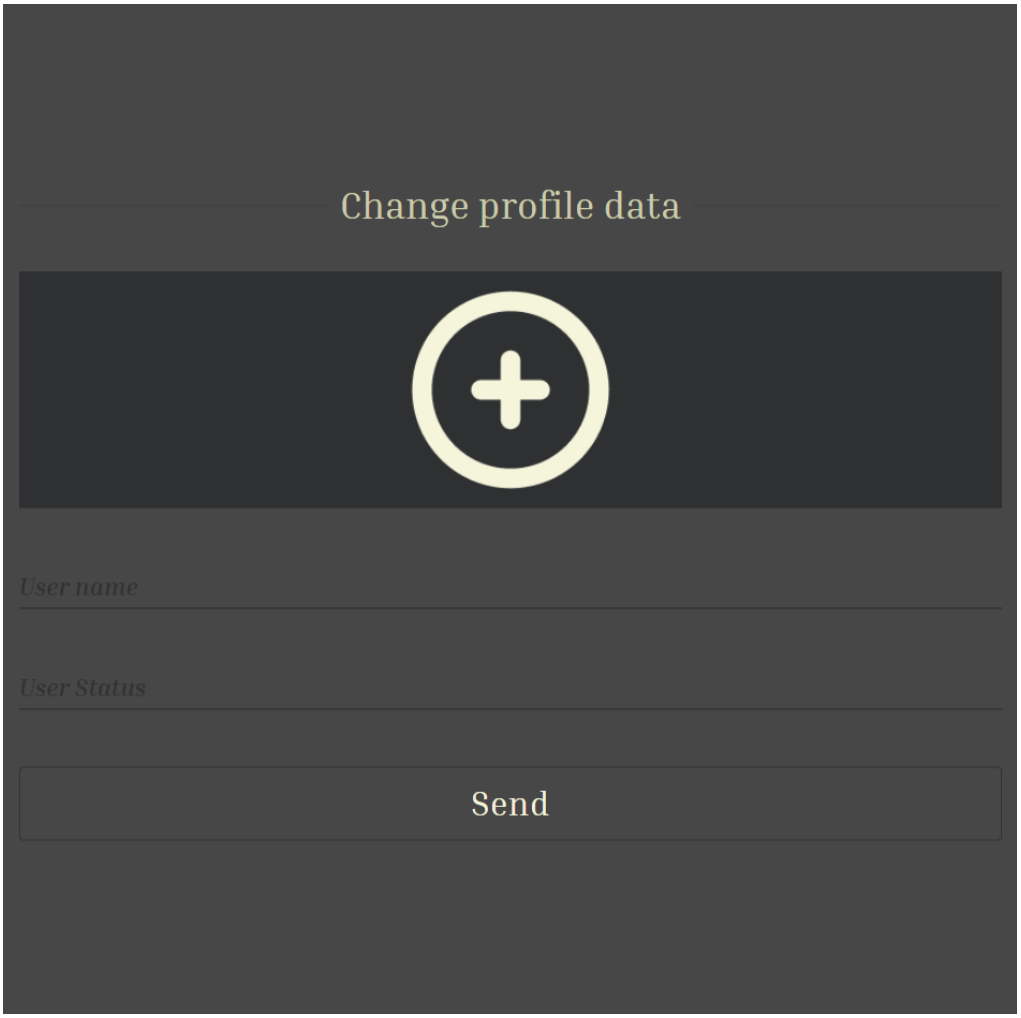

Рисунок 4.5 – Форма редагування данних користувача

Окрім цього, на сторінці профілю була можливість публікувати нові пости, які відображалися на головній сторінці після додавання. Користувачі також мали доступ до списку зашарених постів, які вони обмінювалися з іншими користувачами. Це дозволяло їм легко відстежувати та переглядати контент, яким вони обмінювалися з іншими користувачами на платформі.

Сторінка з користувачами

На сторінці з користувачами ми створили зручний інтерфейс, який дозволяє користувачам взаємодіяти зі своїми друзями та іншими користувачами. У лівій частині екрану розташований сайдбар, де відображається список друзів користувача.

Коли користувач вибирає одного зі своїх друзів зі списку, у правій частині екрану з'являється вікно чату.

Чат з цим обраним користувачем працює через веб-сокети, що забезпечує миттєву передачу повідомлень у реальному часі. Користувачі можуть спілкуватися зі своїми друзями безпосередньо на цій сторінці, обмінюючись текстовими повідомленнями.

Якщо користувач не обрав жодного друга зі списку, у правій частині екрану з'являється список інших користувачів, які доступні для взаємодії. Користувач може переглядати цей список, переглядати профілі інших користувачів, а також надсилати їм повідомлення або надавати їх до списку друзів.

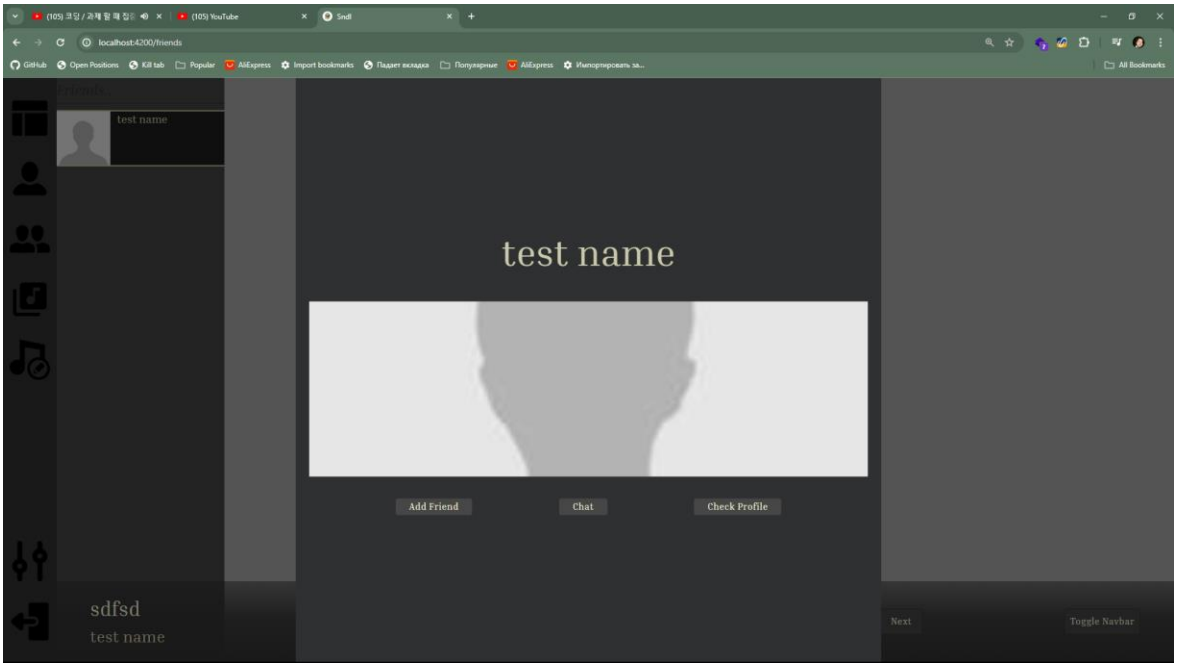

Рисунок 4.6 – Попап детального гляду користувача

При виборі користувача зі списку, користувач має можливість взаємодіяти з цим користувачем на різних рівнях. Він може додати його до списку друзів, якщо він ще не там, написати йому приватне повідомлення або переглянути його профіль, щоб отримати більше інформації про нього. Такий підхід дозволяє зручно взаємодіяти з іншими користувачами та підтримувати соціальні зв'язки на платформі.

Сторінка з піснями

На сторінці з альбомами ми створили зручний інтерфейс для перегляду різних типів альбомів. Ця сторінка містить три основних розділи: "Збережені альбоми", "Нові альбоми" і "Популярні серед друзів". Кожен з цих розділів представлений у вигляді таблиці, яка була створена за допомогою компонентів Material UI. Усі таблиці мають пагінацію, що дозволяє зручно переглядати велику кількість альбомів.

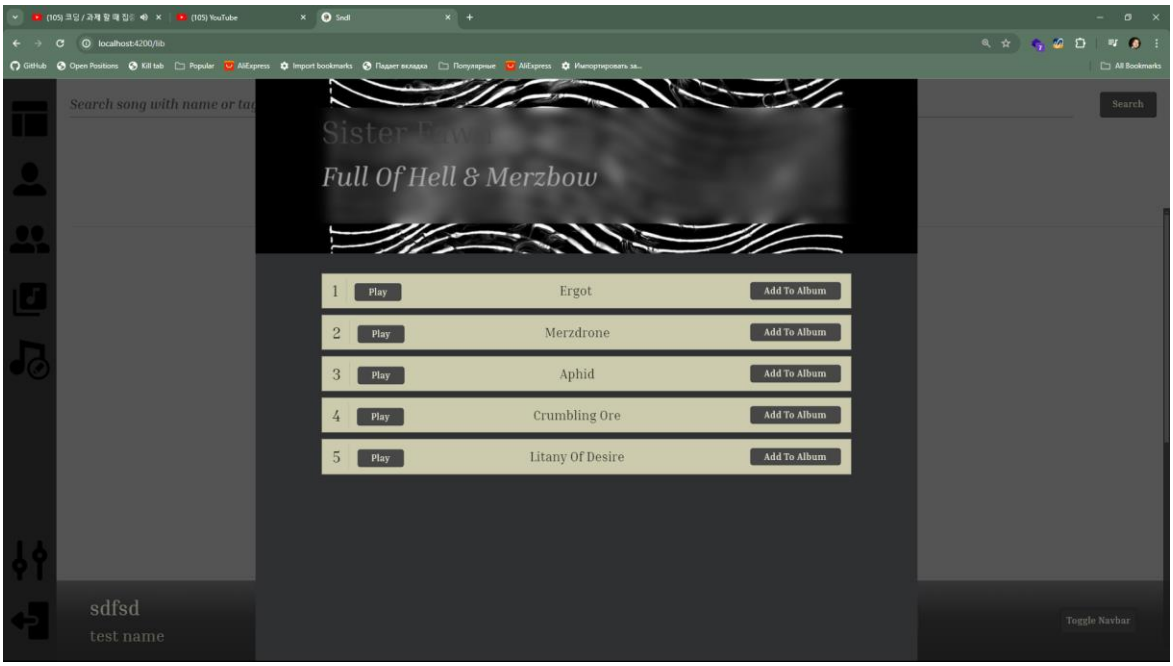

Рисунок 4.7 – Попап з детальним оглядом альбому

При виборі конкретного альбому з таблиці, користувач має можливість здійснити кілька дій. Він може зберегти альбом, щоб мати змогу легко знайти його пізніше. Також він може переглянути список пісень, що входять до цього альбому, а також почати прослуховування їх безпосередньо на сторінці. Це створює зручний інтерфейс для відкриття та взаємодії з музичними альбомами для користувачів.

Сторінка для публікацій композицій

На сторінці для публікацій композицій ми створили форму для користувачів, яка дозволяє легко додавати нові пісні до альбому. Форма реалізована за допомогою

бібліотеки Formik для зручного управління введеними даними, а також використовує валідатори Yup для перевірки правильності введених даних.

Користувач може вибрати альбом, до якого він хоче додати пісню. Якщо альбом вже існує, він буде доданий до нього. Якщо альбому ще не існує, то створиться новий альбом, і пісня буде додана до нього.

Форма також має можливість завантажити саму пісню та зображення для альбому, щоб забезпечити повну інформацію про нову композицію. Крім того, користувач може ввести назву пісні та автора альбому, щоб зробити їх доступними для інших користувачів.

Висновки до розділу 4

У процесі розробки налаштовано та вдало використано сучасні технології як для бекенду, так і для фронтенду. Для збереження даних було використано базу даних Supabase, що надає потужні можливості та забезпечує надійність та швидкість доступу до інформації.

На бекенді для розробки серверної частини було використано Nest.js, який дозволив ефективно організувати структуру проєкту та швидко і просто розробляти API. Ключовими технологіями бекенду були Nest мультіпарт для роботи з файлами, JWT для забезпечення безпеки та Passport.js для авторизації.

У фронтенді для створення користувацького інтерфейсу використовувалась бібліотека React разом з React Router для реалізації маршрутизації та Redux для керування станом додатка. Для стилізації та побудови інтерфейсу була використана графічна бібліотека Material UI. Додатково були використані Axios для здійснення HTTP-запитів та бібліотека Socket.IO для реалізації реального часу у чат-сервісі.

Ці технології дозволили створити надійний, ефективний та зручний застосунок, який відповідає всім вимогам та потребам користувачів. Використання сучасних інструментів та технологій дозволило створити додаток, який пропонує користувачам

зручний та функціональний інтерфейс, а також забезпечує високу швидкість та безпеку обробки їхніх даних.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено вебзастосунок для пошуку музичних партнерів, що забезпечує ефективну взаємодію між музикантами та артистами.

Теоретичним підґрунтям роботи став детальний аналіз предметної області, зокрема галузі пошуку музичних партнерів, та аналіз аналогічних систем. Це дозволило виявити кілька ключових недоліків наявних застосунків, таких як обмежений функціонал, відсутність можливості публікувати контент для певної аудиторії, а також застарілий дизайн. На основі цього аналізу було сформульовано специфікацію вимог до застосунку, що включає пошук музичних партнерів, обмін композиціями, комунікацію в режимі реального часу та безпеку даних користувачів.

У другому розділі було розроблено проектні рішення для системи пошуку музичних партнерів та обміну композиціями. Проектні рішення включають опис функцій системи через сценарії використання та графічні моделі, зокрема діаграми використання, послідовності, класів та компонентів.

У третьому розділі було створено базове оточення для застосунку. Для організації монорепозиторію обрано NX, що забезпечує ефективне управління кількома додатками та бібліотеками в одному проекті. Для фронтенду обрано React, який дозволяє створювати швидкі та інтерактивні інтерфейси. На бекенді використано NestJS у поєднанні з Fastify для підвищення продуктивності, а також бібліотеки JWT та Passport.js для забезпечення безпеки та авторизації. Для керування станом додатку використано Redux, а для роботи з базами даних – Prisma ORM. Дані зберігаються та обробляються у базі даних Supabase.

У процесі розробки налаштовано та вдало використано сучасні технології для бекенду та фронтенду. На бекенді було використано Nest.js, що дозволило ефективно організувати структуру проєкту та розробляти API. Ключовими технологіями бекенду стали Nest мультіпарт для роботи з файлами, JWT для забезпечення безпеки та

Passport.js для авторизації. На фронтенді використовувалась бібліотека React разом з React Router для маршрутизації та Redux для керування станом додатка. Для стилізації та побудови інтерфейсу використано Material UI, Axios для HTTP-запитів та Socket.IO для реалізації чату в реальному часі.

Ці технології дозволили створити надійний, ефективний та зручний вебзастосунок, який відповідає всім вимогам та потребам користувачів. Використання сучасних інструментів та технологій забезпечило додатку зручний та функціональний інтерфейс, високу швидкість та безпеку обробки даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Soundcloud: вебсайт. URL: https://soundcloud.com/ (дата звернення 01.04.2024)

2) Bandmix: вебсайт. URL: https://www.bandmix.com/account/ (дата звернення 01.04.2024)

3) Kompoz: вебсайт. URL: https://www.kompoz.com/ (дата звернення 01.04.2024)

4) Alhir S. S. UML in a Nutshell. O'Reilly, 1998. 290 p. (дата звернення 01.04.2024)

5) Документація React Js: вебсайт. URL: https://react.dev/learn (дата звернення 01.04.2024)

6) Документація Redux: вебсайт. URL: https://redux.js.org/introduction/getting-started (дата звернення 01.04.2024)

7) Документація Redux toolkit: https://redux-toolkit.js.org/introduction/getting-started вебсайт. URL: (дата звернення 01.04.2024)

8) Документація Node Js: вебсайт. URL: https://nodejs.org/docs/latest/api/ (дата звернення 01.04.2024)

9) Документація Nest Js: вебсайт. URL: https://docs.nestjs.com/ (дата звернення 01.04.2024)

10) Документація Supabase: вебсайт. URL: https://supabase.com/docs (дата звернення 01.04.2024)

11) Документація SQLite: вебсайт. URL: https://www.sqlite.org/docs.html (дата звернення 01.04.2024)

12) Документація Prisma: вебсайт. URL: https://www.prisma.io/docs (дата звернення 01.04.2024)

13) Implementing an Auth Guard with JWT tokens in Nest.js: вебсайт. URL: https://medium.com/@bhanushaliyash2000/implementing-an-auth-guard-with-jwt-tokens-in-nest-js-92176a9c3457 (дата звернення 01.04.2024)

14) Prisma relations: вебсайт. URL: https://medium.com/yavar/prisma-relations-2ea20c42f616 (дата звернення 01.04.2024)

15) Документація React Router: вебсайт. URL: https://reactrouter.com/en/main (дата звернення 01.04.2024)

16) Документація Vite: вебсайт. URL: https://vitejs.dev/guide/ (дата звернення 01.04.2024)

17) Документація Nx Workspace: вебсайт. URL: https://nx.dev/getting-started/intro (дата звернення 01.04.2024)

18) Документація Fastify: вебсайт. URL: https://fastify.dev/docs/latest/ (дата звернення 01.04.2024)

19) Документація Material UI: вебсайт. URL: https://mui.com/material-ui/getting-started/ (дата звернення 01.04.2024)

20) Документація Formik: вебсайт. URL: https://formik.org/docs/overview (дата звернення 01.04.2024)

21) Документація Yup: вебсайт. URL: https://github.com/jquense/yup/tree/pre-v1 (дата звернення 01.04.2024)

22) Документація BCrypt: вебсайт. URL: https://github.com/kelektiv/node.bcrypt.js#readme (дата звернення 01.04.2024)

23) Документація React Intl: вебсайт. URL: https://formatjs.github.io/docs/react-intl/ (дата звернення 01.04.2024)

24) Документація Axios: вебсайт. URL: https://axios-http.com/ru/docs/intro (дата звернення 01.04.2024)

ДОДАТОК А

Структура бази даних Prisma ORM

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider  = "postgresql"
  url       = env("DATABASE_URL")
  directUrl = env("DIRECT_URL")
}

model User {
  id Int @id @default(autoincrement())

  name String
  email String @unique
  avatar String
  status String

  password String

  createdAlbums  Album[]
  followedAlbums Album[] @relation("album-follower")

  createdPosts  Post[]
```

```
    followedPosts Post[] @relation("post-follower")

    friendUsers User[] @relation("friends")
    friendOf    User[] @relation("friends")

    chats ChatRoom[] @relation("participants")

    messages Message[]

    comments Comment[]

    created_at DateTime? @default(now())
    deletedAt DateTime?
}

model ChatRoom {
    id Int @id @default(autoincrement())

    participants User[] @relation("participants")

    messages Message[]

    created_at DateTime? @default(now())
    deletedAt DateTime?
}
```

```
model Post {
  id      Int    @id @default(autoincrement())
  content String

  creatorId Int
  creator User @relation(fields: [creatorId], references: [id])

  followedPosts User[] @relation("post-follower")

  comments Comment[]

  created_at DateTime? @default(now())
  deletedAt DateTime?
}

model Comment {
  id   Int    @id @default(autoincrement())
  content String

  post   Post @relation(fields: [postId], references: [id])
  postId Int

  author   User @relation(fields: [authorId], references: [id])
  authorId Int

  deletedAt DateTime?
```

```
}

model Album {
  id   Int    @id @default(autoincrement())
  name String @unique
  image String
  pseudoAuthor String

  creatorId Int
  creator User @relation(fields: [creatorId], references: [id])
  followedAlbums User[] @relation("album-follower")

  songs Song[]

  created_at DateTime? @default(now())
  deletedAt DateTime?
}

model Song {
  id Int @id @default(autoincrement())
  name String

  album   Album @relation(fields: [albumId], references: [id])
  albumId Int

  sourse String
```

```
  created_at DateTime? @default(now())
  deletedAt DateTime?
}

model Message {
  id      Int    @id @default(autoincrement())
  content String

  sender   User @relation(fields: [senderId], references: [id])
  senderId Int

  chatRoom   ChatRoom @relation(fields: [chatRoomId], references: [id])
  chatRoomId Int

  created_at DateTime? @default(now())
  deletedAt DateTime?
}
```