

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Вебплатформа для надання відгуків на музичні
твори з використанням засобів штучного інтелекту»**

Виконав студент групи ІПЗм-22-1
ГОРДЕЄВ Ростислав Сергійович

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
 ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
 ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
 КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії

програмного забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2023 р.

ЗАВДАННЯ на кваліфікаційну роботу

Здобувач вищої освіти: **ГОРДЕЄВ Ростислав Сергійович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Вебплатформа для надання відгуків на музичні твори з використанням засобів штучного інтелекту»,**

затверджена Наказом закладу вищої освіти від **«18» вересня 2023 р. № 479/с**

Вихідні дані для роботи: Предметом дослідження є використання вебпрактик та архітектури «клієнт-сервер» для створення вебплатформи для обговорення та надання відгуків поціновувачами музики на музичні твори; використання засобів штучного інтелекту для створення музичних рекомендацій. Об'єктом дослідження є вебплатформа, що надає можливість обговорення та надання відгуків поціновувачами музики на музичні твори; а також створення моделі музичної рекомендаційної системи.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	22.09.2023р.	22.09.2023р.
2	Вакалюк Т.А.	22.09.2023р.	22.09.2023р.
3	Вакалюк Т.А.	22.09.2023р.	22.09.2023р.

РЕФЕРАТ

Випускна робота магістра включає в себе розробку вебплатформи для обговорення музики і також науковий аналіз методів створення рекомендаційних систем. Робота складається з музичної вебсистеми та пояснювальної записки.

Пояснювальна записка до випускної роботи містить 83 сторінок, 49 ілюстрацій, 14 таблиць, 3 додатки та 21 використане джерело.

В першому розділі були описані існуючі розробки схожих вебплатформ для надання музичних відгуків і взято для розгляду декілька музичних рекомендаційних систем. Також було проаналізовано наявні методи створення музичних рекомендаційних систем. На кінець, було обрано засоби для розробки вебсайту та рекомендаційної системи.

В другому розділі було змодельовано архітектуру бази даних та самої вебплатформи для надання музичних відгуків. Крім цього, було реалізовано музичну рекомендаційну систему за допомогою обраних алгоритмів.

В третьому розділі було продемонстровано практичну реалізацію створеної вебсистеми.

У висновках було підведено підсумки про те, що вдалося дослідити та розробити в кваліфікаційній роботі

КЛЮЧОВІ СЛОВА: ВЕБПЛАТФОРМА, МУЗИЧНА РЕКОМЕНДАЦІЙНА СИСТЕМА, НЕЙРОННА МЕРЕЖА, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, МАТРИЧНА ФАКТОРИЗАЦІЯ.

ABSTRACT

The master's thesis includes the development of a web platform for discussing music and a scientific analysis of methods for creating recommendation systems. The work consists of a music web system and an explanatory note.

The explanatory note to the thesis contains 83 pages, 27 illustrations, 14 tables, 3 appendices, and 21 used sources.

The first section described existing developments of similar web platforms for providing music reviews and considered several music recommendation systems. It also analyzed the existing methods of creating music recommendation systems. Finally, the tools for developing the website and recommendation system were selected.

The second section modeled the architecture of the database and the web platform itself for providing music reviews. In addition, a music recommendation system was implemented using selected algorithms.

The third section demonstrated the practical implementation of the created web system.

The conclusions summarized the research and development in the qualification work.

KEYWORDS: WEB PLATFORM, MUSIC RECOMMENDATION SYSTEM, NEURAL NETWORK, COLLABORATIVE FILTERING, MATRIX FACTORIZATION.

ЗМІСТ

ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ НАЯВНИХ МЕТОДІВ ТА РІШЕНЬ ДЛЯ РЕАЛІЗАЦІЇ ВЕБПЛАТФОРМИ ТА МОДЕЛІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ.....	9
1.1. Аналіз існуючих аналогів вебплатформи для надання відгуків.....	9
1.2. Аналіз існуючих аналогів музичних рекомендаційних систем.....	12
1.3. Аналіз існуючих методів розробки музичної рекомендаційної системи	15
1.4. Інструментальні засоби реалізації програмного продукту	27
Висновки до першого розділу.....	30
РОЗДІЛ 2.ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ТА МОДЕЛІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ	32
2.1. Проєктування бази даних вебплатформи для надання музичних відгуків	32
2.2. Архітектура та узагальнена структура музичної вебплатформи	39
2.3. Розробка музичної рекомендаційної системи	42
Висновки до другого розділу	50
РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ ВЕБПЛАТФОРМИ ДЛЯ НАДАННЯ МУЗИЧНИХ ВІДГУКІВ	52
3.1. Порядок роботи вебплатформи для надання музичних відгуків	52
3.2. Тестування музичної вебплатформи	66
Висновки до третього розділу.....	71
ВИСНОВКИ.....	73
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	75

ВСТУП

Актуальність теми. Музична індустрія продовжує демонструвати стабільне зростання рік за роком. Згідно з аналізом, проведеним Міжнародною федерацією фонографічної індустрії, останнім часом галузь виросла на 18,5%[3]. Паралельно з цим, відбувається розширення ринку онлайн-музичних платформ, серед яких особливо популярними стають стрімінгові сервіси. Ці платформи пропонують величезний вибір музичних треків, що може ускладнити пошук потрібної музики для користувачів. Щоб спростити цей процес, корисними є системи, які пропонують музичні рекомендації, виходячи з індивідуальних переваг користувачів.

Зростання популярності сервісів на кшталт Spotify, YouTube та Netflix призвело до збільшення значення рекомендаційних систем у нашому повсякденному житті. Вони використовуються у різних додатках, включаючи такі області як: музика, книги, кіно, електронна комерція, новини, ТБ-програми, інтернет-ресурси, туризм тощо[4].

Наукові та бізнес-спільноти проявляють значний інтерес до розробки рекомендаційних систем, які б могли з високою точністю передбачати переваги користувачів. Ці системи не тільки допомагають користувачам у виборі продукції, але й приносять вигоду бізнесу. Наприклад, Netflix провів конкурс з призовим фондом у мільйон доларів для створення алгоритму рекомендаційної системи, який був би ефективнішим від існуючого[5]. Відповідно до статті з бізнес-журналу Школи Уортон, рекомендаційні алгоритми Netflix спрямовані на пропонування нішових продуктів, що допомагає знизити інвентаризаційні витрати компанії. Про це також дізнались такі компанії-гіганти, як Amazon та Apple [4].

Метою дослідження є створення платформи, що надасть засоби для вираження своєї думки про музичний проєкт; і також аналіз ролі рекомендаційних систем у сучасній музичній індустрії та створення своєї моделі рекомендаційної системи.

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

- Проаналізувати існуючі розробки вебплатформ, що надають можливість написання відгуків на музичні проєкти;
- Проаналізувати існуючі алгоритми створення моделей рекомендаційних систем;
- Обрати технології розробки вебпроєкту;
- Спроектувати архітектуру вебплатформи;
- За допомогою спроектованої системи реалізувати програмний продукт та музичну рекомендаційну систему;

Об'єктом дослідження є вебплатформа, що надає можливість обговорення та надання відгуків поціновувачами музики на музичні твори; а також створення моделі музичної рекомендаційної системи.

Предметом дослідження є використання вебпрактик та архітектури «клієнт-сервер» для створення вебплатформи для обговорення та надання відгуків поціновувачами музики на твори; використання засобів штучного інтелекту для створення музичних рекомендацій.

Наукова новизна отриманих результатів полягає у тому, що:

- спроектовано власний підхід створення музичної рекомендаційної системи з використанням штучного інтелекту;
- реалізовано власну нейронну мережу, що надасть максимально точні рекомендації для створеної вебсистеми.

Практичне значення полягає у тому, що:

- створено вебплатформу, що надає можливість виражати думки різними способами – від різних видів обговорення музичного твору (написання відгуків, чат) до різного типу оцінювання (оцінка, реакція);
- розроблено власну музичну рекомендаційну систему, що надає список схожих музичних проєктів/виконавців, враховуючи оцінювання інших користувачів вебсистеми.

Апробація результатів роботи. Матеріали дослідження за тематикою кваліфікаційної роботи:

— Аналіз існуючих алгоритмів музичних рекомендаційних систем / Гордєєв Р.С., Граф М.С. // Стаття з наукового журналу «Технічна інженерія» .— 19 грудня 2022. – № 2 (90) 2022. – С. 86-93[1].

— Розробка соціальної мережі для меломанів / Гордєєв Р.С., Вакалюк Т.А. // Тези доповідей III Всеукраїнської науково-практичної інтернет-конференції "СУЧАСНІ КОМП'ЮТЕРНІ ТА ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ". – 12-19 грудня 2022. – С. 346-348[2].

РОЗДІЛ 1. АНАЛІЗ НАЯВНИХ МЕТОДІВ ТА РІШЕНЬ ДЛЯ РЕАЛІЗАЦІЇ ВЕБПЛАТФОРМИ ТА МОДЕЛІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

1.1. Аналіз існуючих аналогів вебплатформи для надання відгуків

Для оптимальної розробки музичної вебплатформи важливо вивчити існуючі приклади в цій сфері. Такий аналіз дозволить краще зрозуміти, які функції та особливості є ефективними, а які можна вдосконалити. Розглянемо два, вже реалізованих, вебпроекти для цих цілей.

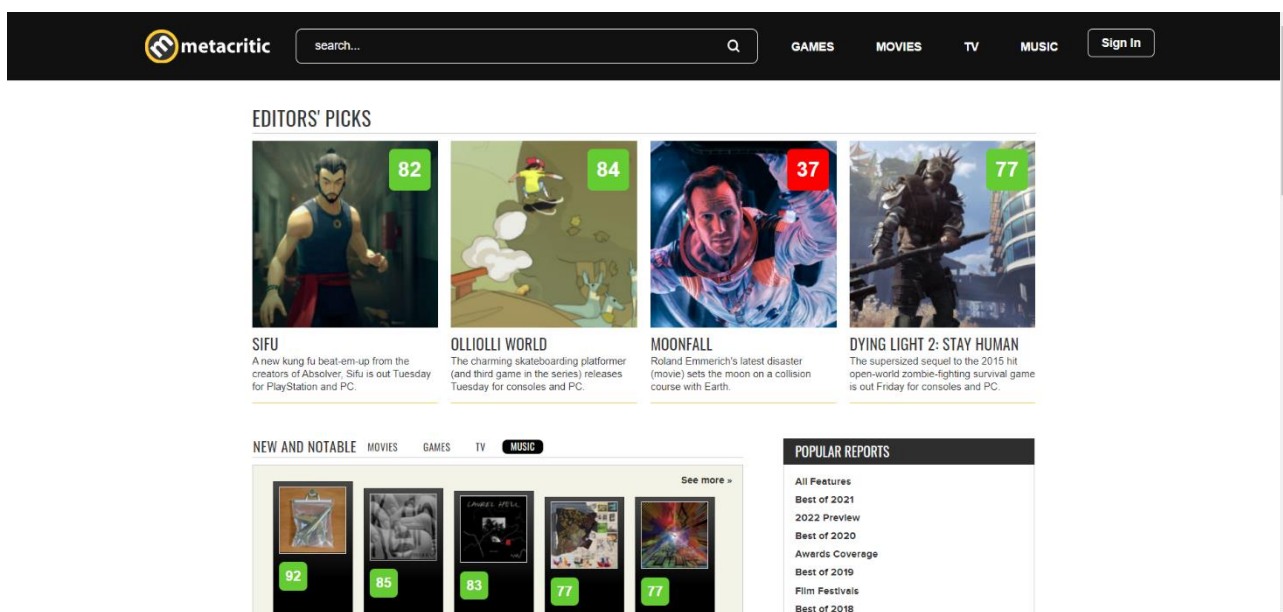


Рис. 1.1. Головна сторінка сайту «Metacritic»

"Metacritic" (див. рис. 1.1) є вебсайтом, де зібрані рецензії на музичні альбоми, відеоігри, фільми і телешоу. Для переходу на сторінку конкретного музичного твору можна скористатися головною сторінкою або пошуком. На сторінці твору відображаються відгуки з оцінками, середній бал твору та його деталі. Оцінки надаються як критиками, так і користувачами, а для узагальненого визначення ставлення до твору використовується трьохкольорова система (зелений, жовтий, червоний), що відображає загальну привабливість продукту. Сторінка також показує позицію альбому в рейтингу кращих за рік.

ANTS FROM UP THERE

by **Black Country, New Road**

Ninja Tune | Release Date: Feb 4, 2022



Summary	Critic Reviews	User Reviews	Details & Credits
	92 Metascore Universal acclaim based on 15 Critic Reviews What's this?	9.7 User Score Universal acclaim based on 54 Ratings Your Score: 0	Record Label: Ninja Tune Genre(s): Pop/Rock, Alternative/Indie Rock, New Wave/Post-Punk Revival, Experimental Rock More Details and Credits »
Buy Now Buy on amazon.com			

CRITIC REVIEWS	USER REVIEWS	AWARDS & RANKINGS
Positive: 15 Mixed: 0 Negative: 0 100 The Independent (UK) Feb 3, 2022 The sheer grace and ambition of Ants... will prove tough for 2022 to top. A huge leap forward, headfirst into the unknown. All this publication's reviews Read full review	Positive: 14 Mixed: 0 Negative: 0 10 Lodz Feb 5, 2022 Music at its best. No superfluous sounds. Perfectly crafted songs that really sound. A little bit like Mahavishnu (King Crimson) meet Mark... Expand Helpful? ☐ YES ☐ NO 0 of 0 users found this helpful	#1 Best Album of 2022 #5 Most Discussed Album of 2022 ESSENTIAL LINKS New & Recent Releases A-Z by Artist New & Recent Releases by Date Upcoming Release Calendar 2022 High Scores Best Albums of the Decade All-Time High Scores

Рис. 1.2. Сторінка альбому на сайті «Metacritic»

Одним із недоліків вебсайту "Metacritic" є не зовсім зручний інтерфейс для перегляду відгуків. Розміри блоків з відгуками та розмір шрифту є досить малими, що ускладнює процес читання. Крім того, на сайті можна лише писати відгуки, іншого функціоналу для обговорення, як, наприклад, форуми чи коментарі, немає.

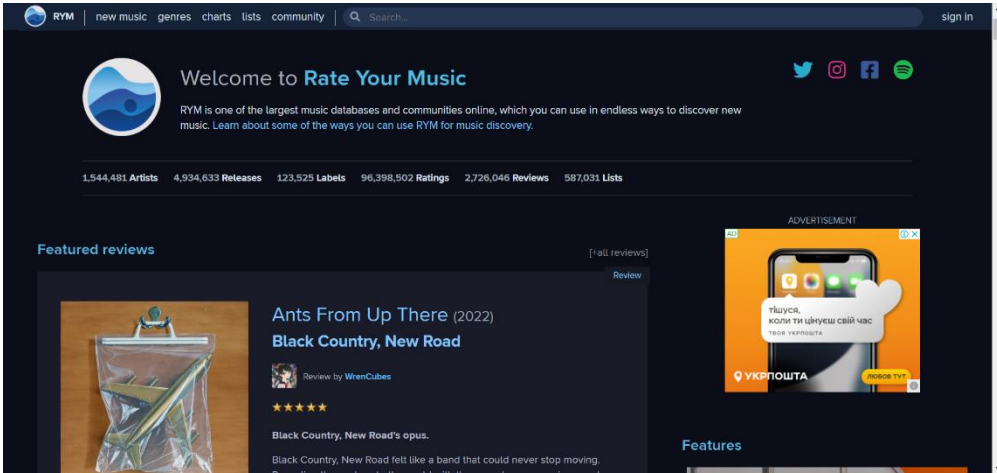


Рис. 1.3. Головна сторінка сайту «RateYourMusic»

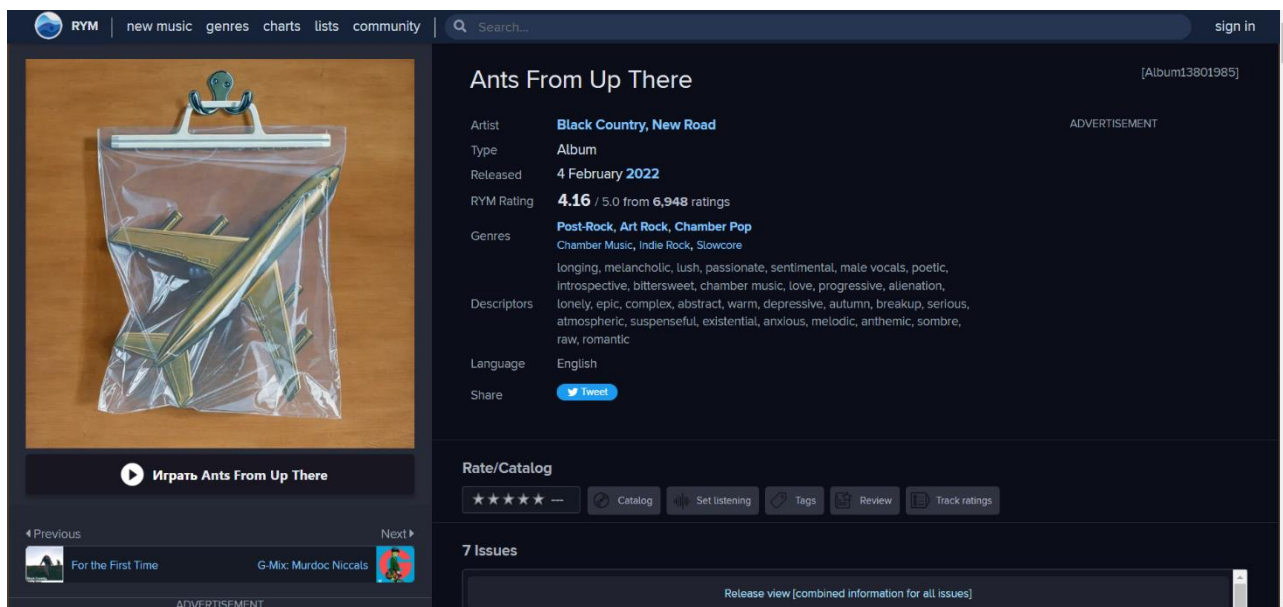


Рис. 1.4. Сторінка музичного твору «RateYourMusic»

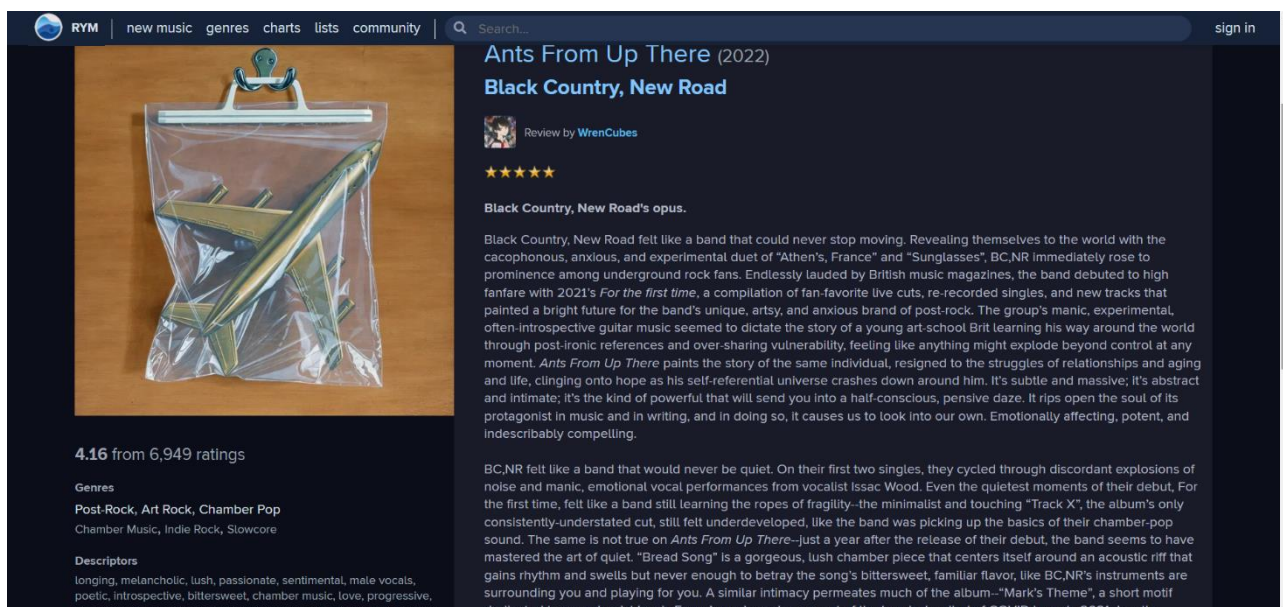


Рис. 1.5. Сторінка відгуку «RateYourMusic»

"RateYourMusic" є вебсайтом, який надає користувачам можливість оцінювати та рецензувати музичні альбоми, сингли, відео, бутлеги та фільми. Головна сторінка сайту включає популярні та нові релізи, а також статистичну інформацію про сайт (див. рис. 1.3). Для переходу до конкретного музичного твору можна використовувати головну сторінку чи пошук. На сторінці музичного твору представлені фотографія, оцінки, деталі та відгуки (див. рис. 1.4).

Однією з переваг "RateYourMusic" є ширший спектр можливостей для обговорення музики, включаючи відгуки, списки та форуми для дискусій. Читання відгуків є зручнішим завдяки більшому розміру блоку та шрифту (див. рис. 1.5).

Проте, сторінки музичних творів на сайті можуть бути незручними для використання через перенасиченість різноманітною інформацією, яка погано структурована, ускладнюючи читання та навігацію.

Після аналізу існуючих вебсайтів можемо визначити кілька ключових елементів, які варто інтегрувати у розробку нової соціальної мережі:

- Створення системи оцінювання музичних творів: Це дозволить користувачам висловлювати свою думку щодо музичних творів, що збільшить взаємодію та залученість на платформі.

- Розрахунок загальної оцінки музичного твору: Об'єднання оцінок від критиків та користувачів у середню оцінку допоможе формувати об'єктивне враження про твір.

- Використання системи кольорів для візуального оцінювання: Кольорова система (зелений, жовтий, червоний) є зручною для швидкого визначення загального ставлення до музичного твору.

- Читабельний дизайн: Важливо забезпечити легкість використання та читання інформації на сайті, включаючи адекватний розмір шрифтів та чітку структуру сторінок.

З урахуванням цих аспектів можна створити ефективну та привабливу для користувача вебплатформу для надання музичних відгуків.

1.2. Аналіз існуючих аналогів музичних рекомендаційних систем

Для того, щоб проаналізувати існуючі рекомендаційні системи, візьмемо дві популярні платформи: *Spotify* та *YouTube Music*.

Алгоритм рекомендацій *Spotify* – це поєднання різних технік, які спрямовані на надання користувачам персоналізованих рекомендацій музики на

основі їх історії прослуховування та уподобань. З основних його компонентів можна виділити такі:

— **Колаборативна фільтрація:** Колаборативна фільтрація - це методика, яка рекомендує елементи на основі вподобань інших користувачів. В контексті Spotify це означає рекомендацію пісень або виконавців, які сподобалися користувачам з подібними звичками прослуховування. Колаборативна фільтрація може бути орієнтованою на користувача (знаходження користувачів із подібними смаками) або орієнтованою на об'єкт (знаходження схожих пісень/виконавців на основі їх характеристик).

— **Фільтрація за вмістом:** Цей підхід рекомендує елементи на основі їхніх характеристик і атрибутів. У випадку Spotify це включає аналіз аудіохарактеристик пісень, таких як темп, тональність, ритмічність, енергійність тощо. Пісні зі схожими аудіохарактеристиками можуть бути рекомендовані користувачам, які вже висловили перевагу пісням з такими характеристиками.

— **Обробка природної мови (NLP):** Spotify також може аналізувати назви пісень, імена виконавців та плейлисти, створені користувачами, для розуміння текстових зразків. Це може надати додатковий контекст для здійснення рекомендацій.

— **Історія та поведінка користувача:** Spotify бере до уваги історію прослуховування користувача, включаючи пісні, які вони вподобали, плейлисти, які створили, та альбоми, які зберегли. Це допомагає побудувати профіль музичних уподобань користувача.

— **Неявний та явний відгук:** Spotify бере до уваги як неявний відгук (історія прослуховування, кількість пропусків, кількість програвань), так і явний відгук (лайки, пальці вгору/вниз), щоб зрозуміти уподобання користувачів.

— **Техніки факторизації матриці:** Ці техніки розкладають матрицю взаємодії користувач-елемент для виявлення внутрішніх закономірностей та уподобань. Факторизація матриці допомагає заповнити відсутні значення та передбачити відносини користувач-елемент.

— **Глибоке навчання та нейронні мережі:** Spotify може використовувати нейронні мережі для вивчення складних закономірностей з даних. Рекурентні нейронні мережі (RNN) або механізми уваги можуть використовуватися для виявлення послідовних закономірностей у прослуховуванні користувачів.

— **Контекстні фактори:** Spotify може брати до уваги додаткові контекстні фактори, такі як час доби, день тижня та розташування користувача, для забезпечення більш релевантних рекомендацій.

Наступним розглянемо алгоритм YouTube Music. Ось деякі характеристики:

— **Профіль користувача та історія прослуховування:** YouTube Music збирає дані про історію прослуховування користувачів, включаючи пісні, які вони відтворювали, альбоми, на які підписувалися, та виконавців, яких вони переглядали. Ці дані допомагають побудувати профіль музичних смаків та уподобань користувача.

— **Неявний та явний відгук:** Аналогічно до інших платформ, YouTube Music враховує як неявний відгук (історія прослуховування, пропуски, кількість переглядів), так і явний відгук (лайки, дизлайки) для кращого розуміння музичних вподобань користувачів.

— **Колаборативна фільтрація:** Ймовірно, YouTube Music використовує техніки колаборативної фільтрації для рекомендацій музики на основі уподобань інших користувачів з схожими музичними смаками. Це може включати пропозиції для пісень та виконавців, які сподобалися іншим користувачам з подібними прослуховуваннями.

— **Фільтрація за вмістом:** Платформа може аналізувати аудіохарактеристики пісень (такі як жанр, темп, настрій), відеоролики та тексти пісень для рекомендацій музики зі схожим вмістом.

— **Аналіз підписок та бібліотеки користувача:** Рекомендаційна система YouTube Music може також брати до уваги відеокліпи, пісні та альбоми,

на які користувачі підписалися або додали до своїх бібліотек. Це надає додатковий контекст щодо їхніх інтересів.

— **Користувацькі плейлисти та створення контенту:** YouTube Music може використовувати інформацію про плейлисти, які створили користувачі, а також враховувати власний створений контент.

— **Завантаження відео та взаємодія зі спільнотою:** Інформація про відео, яке користувачі завантажили або переглядали, а також їхні взаємодії з коментарями та лайками, може також вплинути на рекомендації.

— **Машинне навчання та аналіз даних:** YouTube Music використовує методи машинного навчання та аналізу даних для виявлення закономірностей та зв'язків у масиві інформації, що зібрана від користувачів.

— **Аналіз трендів та популярності:** YouTube Music може аналізувати тенденції та популярність пісень, виконавців та жанрів для забезпечення актуальних рекомендацій.

1.3. Аналіз існуючих методів розробки музичної рекомендаційної системи

Концепція систем рекомендацій зазнала значних змін протягом останніх декількох десятиліть. Наприклад, в роботі Резніка та Варіана [8], опублікованій у 1997 році, система рекомендацій характеризується наступним чином: "Традиційна система рекомендацій включає процес, де користувачі подають рекомендації як вхідні дані. Ці дані потім система збирає та перенаправляє до відповідних одержувачів. Залежно від контексту, ключова зміна може відбуватися на етапі збору цих даних, або ж унікальність системи полягає в здатності точно підібрати рекомендатора та того, хто шукає рекомендацію".

Згодом визначення систем рекомендацій було розширено, щоб охопити ширший спектр систем, які надають рекомендації незалежно від їхнього механізму створення. Це може бути сформульовано так: "Будь-яка система, що виробляє індивідуалізовані рекомендації як результат своєї роботи, або сприяє

користувачеві у персоналізованому виборі об'єктів серед великої кількості можливостей" [9]. Далі більш універсальне визначення було запропоновано Адомавічіусом та Тужилінім: «Нехай C – множина всіх користувачів, а S – множина можливих елементів, які можуть бути рекомендовані. Нехай u – функція корисності, яка визначає корисність предмету s для користувача c : $C \times S \Rightarrow R$, де R – повністю впорядкований список (наприклад, беззнакові цілі або дійсні числа в певному проміжку). Тоді для кожного користувача $c \in C$, нам необхідно обрати елемент $s' \in S$, що максимізує корисність користувача»[10].

З аналізу попередніх визначень стає ясно, що в структурі рекомендаційних систем основними компонентами є *користувачі* та *елементи*. Користувач в даному контексті – це особа, яка взаємодіє з системою, надаючи свої враження та оцінки різних об'єктів (елементів). Водночас, рекомендаційна система відповідає на це, пропонуючи користувачу нові елементи на основі його попередніх вподобань та відгуків.

Вхідні дані для рекомендаційної системи залежать від типу використовуваного алгоритму. Загалом, ці дані можуть належати до однієї з наступних категорій:

— Вхідні дані для рекомендаційних систем часто включають в себе *оцінки*, які відображають думку користувача стосовно певного об'єкта. Ці оцінки можуть мати різні форми. Наприклад, на багатьох музичних сайтах використовується система оцінювання від 0 до 100, де 0 - це найнижча оцінка, а 100 - найвища. Також широко поширена п'ятизіркова система оцінювання, де 0 зірок означає найгірший рейтинг, а 5 зірок - найкращий. Використовується також бінарна система оцінювання, де 0 вказує на негативне ставлення, а 1 - на позитивне. Окрім того, оцінки можуть генеруватися на основі історії покупок користувача, перегляду посилань, а також інших шаблонів взаємодії з інформацією.

— *Демографічні дані* також є важливою категорією вхідних даних для рекомендаційних систем. Це включає інформацію, таку як вік, стать та рівень

освіти користувачів. Збір цих даних може бути складним, оскільки вони зазвичай отримуються безпосередньо від користувачів. Ця інформація може використовуватися для більш точного визначення потреб та інтересів користувачів, забезпечуючи більш персоналізовані рекомендації.

— *Контент-орієнтовані дані* є ще однією ключовою категорією вхідних даних для рекомендаційних систем. Ці дані базуються на аналізі тексту документів, пов'язаних з предметами, які користувач оцінював. Особливості, отримані з такого аналізу, використовуються як вхідні дані для фільтраційних алгоритмів з метою визначення профілю користувача. Це може включати аналіз ключових слів, тем, стилів та інших атрибутів тексту, що дозволяє системі розуміти та враховувати інтереси та уподобання користувача більш точно.

Ціллю рекомендаційної системи є згенерувати пропозиції по новим предметам чи спрогнозувати корисність певного елемента для користувача.

Вихідними даними рекомендаційної системи може бути *передбачення* або *рекомендація*:

— *Передбачення*, яке виражається числовим значенням, представляє очікувану думку користувача щодо певного предмета. Таке передбачення має бути в тому ж числовому діапазоні, що і вхідні дані, щоб забезпечити консистентність і точність. Цей підхід у рекомендаційних системах відомий як використання "Індивідуального рахунку" або персоналізованого оцінювання. Він дозволяє системі прогнозувати, як користувач може відреагувати на певний продукт або контент, базуючись на його попередніх взаємодіях та перевагах.

— Ще однією ключовою характеристикою рекомендаційних систем є створення *рекомендацій*, які представляють собою список з n елементів, вибраних з метою максимального відповідання інтересам користувача. Основна умова такого списку полягає в тому, що він повинен містити тільки ті елементи, з якими користувач ще не стикався - тобто те, що він ще не купував, не слухав, не читав, не дивився тощо. Цей метод в рекомендаційних системах відомий як Топ-N рекомендації або Ранговий рахунок. Це підхід допомагає користувачам

відкривати нові продукти або контент, які відповідають їхнім перевагам і водночас є новими для них.

Матриця оцінок користувача. У найзагальнішій формі, дослідження рекомендаційних систем часто представлено у вигляді таблиці або схеми, як це зазначено в таблиці 1.

Таблиця 1.1

Вигляд матриці оцінок користувача

		Елементи					
		1	2	...	i	...	m
Користувачі	1	87		15	48		
	2	39		78			19
	...		49			16	
	u			19	100		
	...		51				33
	n	62			48		
a		73	?		51	98	

Відомі вподобання користувача показані матрицею з n користувачів та m предметів, де кожна клітинка $r_{u,i}$ відповідає оцінці наданій предмету i користувачем u . Матриця оцінок користувачів у рекомендаційних системах часто буває розрідженою, оскільки користувачі зазвичай не оцінюють більшість доступних елементів. Основною задачею рекомендаційної системи є прогнозування оцінок, які користувач міг би дати тим елементам, які він ще не оцінював. Для цього система зазвичай визначає рейтинги для елементів, які користувач ще не переглядав, і ті предмети, які отримують найвищі прогнозовані оцінки, виносяться на розгляд користувача у вигляді рекомендацій.

Види алгоритмів фільтрації. Різні алгоритми фільтрації рекомендаційних систем можна категоризувати в 3 типи (див. рис. 1):

— *Колаборативна фільтрація.* У системах, що базуються на цьому алгоритмі, рекомендації для користувача формуються на основі оцінок, наданих усіма користувачами.

— *Фільтрація за вмістом*. Цей підхід характеризується рекомендацією користувачеві елементів, які за вмістом або характеристиками схожі на ті предмети, які раніше йому подобалися.

— *Гібридна фільтрація*. Цей алгоритм інтегрує разом два основних підходи: колаборативну фільтрацію та фільтрацію за вмістом.

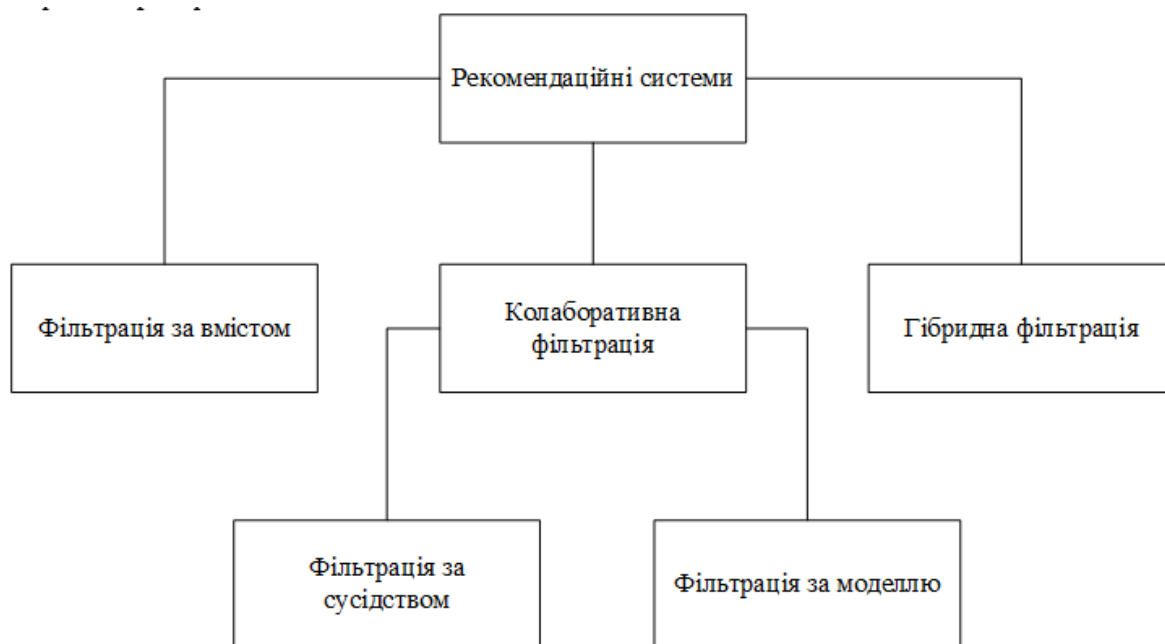


Рис. 1.6. Види фільтраційних алгоритмів рекомендаційних систем

Колаборативна фільтрація. Алгоритми колаборативної фільтрації функціонують, збираючи відгуки користувачів у вигляді рейтингів для різних елементів. Вони використовують ці дані для аналізу спільних тенденцій оцінювання серед різних користувачів з метою визначення, які елементи можна було б рекомендувати.

Колаборативна фільтрація може бути поділена на два основних підвиди, як це показано на рисунку 1: *фільтрація за сусідством* і *фільтрація за моделлю*. Розглянемо ці алгоритми.

Колаборативна фільтрація за сусідством. У підходах, що базуються на сусідстві (neighborhood-based approaches), процес відбору рекомендацій включає визначення підмножини користувачів, які мають схожі вподобання або оцінки з поточним користувачем. Потім, зважена комбінація оцінок цих схожих користувачів використовується для формування передбачень щодо потенційних

інтересів заданого користувача. Основні кроки цього алгоритму можна узагальнити так:

1. На першому етапі кожному користувачу присвоюється вага, яка визначається на основі ступеня його подібності до поточного користувача.
2. Далі вибираються k користувачів, які мають найбільше спільного з поточним користувачем, тобто найближчі сусіди.
3. Використовуючи ваги, визначені на першому кроці, розраховуються прогнозовані оцінки на основі оцінок, наданих вибраними сусідами.

У кроці 1 вага $w_{a,u}$ являється мірою спільності між користувачем u та поточним користувачем a . Найбільш поширеною мірою спільності є коефіцієнт кореляції Пірсона. Він записується такою формулою:

$$w_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2 \sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}}, \quad (1)$$

де I – множина елементів оцінена обома користувачами, $r_{u,i}$ – оцінка надана елементу i користувачем u і $\bar{r}_a$ – середня оцінка надана користувачем u [10].

У кроці 3 передбачення розраховуються як зважене середнє значення відхилень від середнього значення сусіда:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \bar{r}_u) \times w_{a,u}}{\sum_{u \in K} w_{a,u}}, \quad (2)$$

де $p_{a,i}$ – передбачення для поточного користувача a для елемента i , $w_{a,u}$ – подібність між користувачами a та u і K – це сусіди або множина найбільш подібних користувачів. [7].

Подібність на основі кореляції Пірсона визначає ступінь лінійної залежності між двома змінними. Це означає, що оцінки двох користувачів можна розглядати як вектори в m -вимірному просторі, і для визначення подібності можна використовувати косинус кута між цими векторами. Цей підхід відображає, наскільки близько орієнтовані вектори оцінок користувачів, де вектори, що спрямовані у схожих напрямках, вказують на вищу ступінь подібності між перевагами користувачів:

$$w_{a,u} = \cos(\vec{r}_a, \vec{r}_u) = \frac{\vec{r}_a \cdot \vec{r}_u}{\|\vec{r}_a\|_2 \times \|\vec{r}_u\|_2} = \frac{\sum_{i=1}^m r_{a,i} r_{u,i}}{\sqrt{\sum_{i=1}^m r_{a,i}^2} \sqrt{\sum_{i=1}^m r_{u,i}^2}}, \quad (3)$$

При розрахунку косинусної подібності важливо враховувати, що рейтинги не можуть бути від'ємними, і всі неоцінені елементи повинні мати рейтинг 0 [7].

Розширення до колаборативної фільтрації за сусідством. Фільтрація за елементами є одним з розширень методу колаборативної фільтрації за сусідством. У цьому підході фокус зміщується з аналізу схожості між користувачами на аналіз схожості між самими елементами.

Коли алгоритм колаборативної фільтрації за сусідством застосовується до великої кількості користувачів та елементів, він дійсно може зіткнутися з проблемами масштабування через високу розрахункову складність пошуку подібних користувачів. У відповідь на цю проблему, Грег Лінден запропонував метод колаборативної фільтрації елемент-до-елемента, який зосереджується на зіставленні оцінених користувачем елементів з подібними елементами, а не на пошуку подібних користувачів.

В цьому підході, подібності між парами елементів i та j розраховуються використовуючи кореляцію Пірсона:

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}, \quad (4)$$

де U – множина елементів користувачів, які оцінили обидва елементи i та j ; $r_{u,i}$ – оцінка надана елементу i користувачем u і $\bar{r}_i$ – середня оцінка i -того елемента серед користувачів.[7].

Тоді рейтинг для i -того елемента для користувача a може бути спрогнозований використовуючи просте зважене середнє значення:

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} \cdot w_{i,j}}{\sum_{j \in K} |w_{i,j}|}, \quad (5)$$

де K – множина сусідів k -тових елементів оцінені користувачем a , що найбільш подібні до i -того елемента [7].

Для колаборативної фільтрації за елементами може бути застосована альтернативні метрики, наприклад, скорегована косинусова подібність.

Колаборативна фільтрація за моделлю. Розвиток моделей машинного навчання та алгоритмів добування даних відіграє важливу роль у вдосконаленні рекомендаційних систем. Ці технології дозволяють системам виявляти складні шаблони в тренувальних даних, що наступно використовується для роблення обґрунтованих прогнозів на тестових або реальних даних. Алгоритми колаборативної фільтрації за моделлю, як наприклад Баєсові моделі та кластерні моделі, розроблялися для усунення деяких недоліків традиційних методів колаборативної фільтрації. [11].

Баєсовий алгоритм колаборативної фільтрації за моделлю. Баєсова мережа – це орієнтований, ациклічний граф з триплетом $\langle N, A, \Theta \rangle$, де кожне ребро $n \in N$ представляє випадкову змінну, а кожна направлена дуга $a \in A$ між ребрами це імовірнісний зв'язок між змінними і Θ – це таблиця умовної ймовірності, що розраховує наскільки ребро залежить від його батьків. Баєсові мережі часто використовуються для завдань класифікації.

Алгоритм колаборативної фільтрації на основі наївного Байєса використовує метод наївного Байєса для виконання прогнозів. Він базується на припущенні, що різні характеристики (або ознаки) незалежні в рамках кожного класу. Ймовірність того, що конкретний клас відповідає всім наявним ознакам, обчислюється, а клас з найвищою обчисленою ймовірністю вибирається як прогнозований. У випадках з неповними даними обчислення ймовірності та класифікація здійснюються на основі доступних спостережень [14]:

$$\text{class} = \arg \max p(\text{class}_j) \prod_0 P(X_0 = x_0 | \text{class}_j), \quad (6)$$

Оцінювач Лапласа використовується для розгладження ймовірності розрахунків і уникнення умовної ймовірності 0:

$$P(X_i = x_i | Y = y) = \frac{\#(X_i = x_i, Y = y) + 1}{\#(Y = y) + |X_i|}, \quad (7)$$

де $|X_i|$ – це розмір множини класу $\{X_i\}$ [11].

При застосуванні наївного Байєсового алгоритму до колаборативної фільтрації, мультикласові дані зазвичай переводяться в формат бінарного класу, після чого трансформуються в булеві вектори ознак на основі рейтингової

матриці. Цей процес спрощує використання наївного Байєса для задач колаборативної фільтрації, але також веде до певних викликів, таких як проблеми масштабування та втрата інформації, яка властива мультикласовим даним.

Що стосується точності і масштабування, то було виявлено, що хоча простий Байєсовий алгоритм може мати гіршу точність порівняно з методами, заснованими на кореляції Пірсона, він часто виявляється кращим з точки зору масштабованості. Це тому, що прогнози на основі кореляції Пірсона базуються на спостережуваних рейтингах, що може вимагати більше часу на обчислення, в той час як наївний Байєс може швидше створювати прогнози, особливо у випадках з великими наборами даних.

Кластерний алгоритм колаборативної фільтрації за моделлю. Кластер є групою об'єктів даних, які є схожими між собою всередині одного кластера, але відрізняються від об'єктів в інших кластерах. Схожість між об'єктами в межах кластера може бути виміряна за допомогою різних метрик, серед яких метрика Мінковського та кореляція Пірсона є поширеними варіантами.

Для двох об'єктів даних, $X = (x_1, x_2, \dots, x_n)$ та $Y = (y_1, y_2, \dots, y_n)$, метрика Мінковського визначається так:

$$d(X, Y) = \sqrt[q]{\sum_{i=1}^n |x_i - y_i|^q}, \quad (8)$$

де n – це розмірність об'єкта, x_i та y_i – значення i -того простору об'єкта X та Y відповідно і q – додатне ціле число (коли $q = 1$ – це *відстань Манхеттена*, коли $q = 2$ – це *відстань Евкліда*)[11].

Кластерні методи аналізу даних включають розподільні методи, як-от k -середніх, що відрізняються ефективністю та легкістю впровадження, методи на основі щільності, такі як DBSCAN та OPTICS, які шукають щільні згруповані об'єкти, та ієрархічні методи, з яких відомий BIRCH. Ці методи створюють різні рівні розподілу даних за певним критерієм. Кластеризація в колаборативній фільтрації може використовуватися для розділення даних на групи перед застосуванням інших методів, таких як фільтрація на основі кореляції Пірсона, для підвищення ефективності прогнозування в кожному кластері.

Фільтрація за вмістом. Чисті рекомендаційні системи, які базуються на колаборативній фільтрації, використовують виключно матриці оцінок, наданих користувачами. У такому підході користувачі та елементи розглядаються як основні одиниці, а прогнозування здійснюється без урахування унікальних характеристик окремих користувачів чи елементів. Проте, можливості індивідуальних рекомендацій можуть бути покращені шляхом включення додаткової інформації, такої як демографічні дані користувачів або специфіка елементів, наприклад, інформація про виробника чи жанр музичного альбому. З іншого боку, рекомендаційні системи, орієнтовані на вміст, забезпечують рекомендації шляхом порівняння контенту, який описує елемент, з інтересами та перевагами користувача.

У сфері рекомендаційних систем значна частина досліджень зосереджена на рекомендації об'єктів із текстовим контентом, як-от вебсторінки, альбоми, книги чи фільми. Для вебсторінок особливо цінними є асоційовані тексти, такі як описи чи відгуки користувачів. Деякі підходи розглядають це як завдання Отримання Інформації (OI), де контент асоціюється з інтересами користувача і розглядається як запит. У цьому контексті, документи, які ще не були оцінені користувачем, аналізуються на предмет їх актуальності або схожості з цим запитом. Такий підхід дозволяє рекомендаційним системам забезпечувати більш персоналізовані та релевантні пропозиції, враховуючи індивідуальні переваги користувачів.

Альтернативний підхід до Отримання Інформації (OI) полягає у розгляді рекомендацій як завдання класифікації. В цьому контексті кожен об'єкт, який потребує рекомендації, розглядається як приклад, який містить вміст елемента.

Гібридна фільтрація. Для оптимізації ефективності рекомендаційних систем були запропоновані різні гібридні підходи, які поєднують елементи фільтрації за вмістом та колаборативної фільтрації. Одна з базових стратегій гібридизації полягає у створенні окремих рейтингових списків за допомогою обох методів і подальшому їх об'єднанні в один кінцевий список рекомендацій. Цей підхід дозволяє використовувати переваги обох систем: детальне розуміння

вмісту і контексту елементів від фільтрації за вмістом і велике охоплення та врахування поведінки користувачів від колаборативної фільтрації. Таким чином, гібридні системи можуть пропонувати більш точні та релевантні рекомендації.

Було запропоновано загальний фреймворк для поєднання колаборативної фільтрації з фільтрацією за вмістом, який включає в себе використання контент-орієнтованих прогнозів для перетворення розрідженої матриці оцінок користувачів у повністю заповнену матрицю. У цьому підході спочатку використовується класифікатор на основі наївного Байєса, натренований на документах, що описують елементи, оцінені кожним користувачем. Цей класифікатор потім використовується для заміни неоцінених елементів на передбачені оцінки.

Після створення повністю заповненої матриці псевдооцінок, застосовується колаборативна фільтрація для знаходження користувачів, схожих на поточного користувача, і виконання прогнозів. Для цього використовується кореляція Пірсона, але з правильно зваженими коефіцієнтами, щоб врахувати спільно оцінені елементи та передбачення на основі контенту для конкретного користувача.

Цей гібридний алгоритм показує кращі результати, ніж чисто колаборативна фільтрація, фільтрація за вмістом, або їх проста лінійна комбінація, завдяки більш комплексному підходу до врахування різних джерел інформації.

Існують деякі інші гібридні алгоритми, що ґрунтуються на колаборативній фільтрації, але також зберігають фільтрацію за вмістом для кожного користувача. У таких підходах фільтрація за вмістом використовується не для прямої рекомендації елементів, а для визначення схожих користувачів. Наприклад, профіль кожного користувача може бути представлений як вектор зважених слів, отриманих з успішних прикладів під час тренування, використовуючи алгоритм Віноу.

Замість застосування колаборативної фільтрації до традиційної матриці оцінок користувачів, цей підхід використовує матрицю профілів користувачів.

Це дозволяє системі виявляти схожих користувачів на основі їхніх інтересів і переваг, які виражаються через зважені слова в їх профілях, і потім виконувати прогнози за допомогою колаборативної фільтрації на цій основі. Таким чином, рекомендації можуть стати більш точними і релевантними для кожного індивідуального користувача.

Власна рекомендаційна система. Оскільки в поточній системі наявна система оцінювання різних музичних проєктів, найкраще музичну рекомендаційну систему реалізувати з використанням матричної факторизації, а тобто – колаборативної фільтрації.

Але, в поточний час, рекомендаційні системи стають все складнішими і, для отримання високої точності результатів, однієї моделі вже недостатньо. Задля отримання найкращих рекомендацій, було вирішено скористатися нейронними мережами. Вони добре адаптуються під різні дані, що дозволить отримати найліпші результати.

Для того, щоб створити власну нейронну мережу, потрібно спочатку проаналізувати вже існуючі.

ANN (Artificial Neural Networks):

ANN[12,13] – це широкий термін, що охоплює різні архітектури нейронних мереж. Вони складаються зі штучних нейронів, які передають і обробляють інформацію.

MLP (Multilayer Perceptron) – один з популярних типів ANN, використовується для розпізнавання жанру. Він має один або кілька прихованих шарів, що дозволяє моделі виявляти складні залежності між вхідними характеристиками музичного треку та жанром.

Можна використовувати різні функції активації, такі як ReLU (Rectified Linear Unit), Sigmoid або Softmax, залежно від вимог завдання розпізнавання жанру.

RNN (Recurrent Neural Networks):

RNN[19-21] мають здатність зберігати попередні стани, що дозволяє їм моделювати послідовні дані залежно від контексту. Вони часто використовуються для обробки послідовних даних, таких як мова або музика.

Один з популярних підходів – LSTM (Long Short-Term Memory), який дозволяє моделювати довготривалі залежності. LSTM має механізми затримки, забування та оновлення, що допомагають ефективно працювати з послідовними даними.

RNN можна використовувати для аналізу аудіосигналу треку шляхом поділу його на послідовні часові вікна та використання RNN для визначення стилістичних особливостей жанру.

CNN (Convolutional Neural Networks):

CNN[16-18] використовується для обробки візуальних та аудіоданих. Вони ефективно виявляють локальні особливості та шаблони шляхом застосування згорткових шарів.

CNN може використовуватись для обробки спектрограм аудіосигналів. Спектрограма візуалізує спектральні характеристики треку та може містити інформацію про ритмічні шаблони, акорди, мелодійні структури тощо.

CNN використовуються для здійснення згортки (convolution) та пулінгу (pooling) з метою виявлення важливих ознак жанру в спектрограмі.

Оскільки останні дві нейронні мережі мають нішові використання, для заданої рекомендаційної системи було вирішено скористатись звичайною нейронною мережею ANN.

1.4. Інструментальні засоби реалізації програмного продукту

Для того, щоб розробити вебплатформу, потрібно обрати засоби розробки. Для цього оберемо:

— **React** — це безкоштовна та відкрита бібліотека JavaScript, яка використовується для створення користувацьких інтерфейсів. Вона базується на компонентах, що дозволяє розробникам легко створювати взаємодіючі UI

елементи. React підтримується компанією Meta (раніше відома як Facebook) та великою спільнотою розробників.

— **Redux Toolkit** – це відкрита бібліотека JavaScript, призначена для ефективного управління станом у додатках. Ця бібліотека часто використовується разом із бібліотеками для створення користувацьких інтерфейсів, такими як React. Redux Toolkit надає набір інструментів та функціональностей, які спрощують процес роботи з Redux, відомою бібліотекою для управління станом додатків.

— **RTK Query** , являючись частиною Redux Toolkit, є потужним інструментом для отримання та кешування даних у вебдодатках. Цей інструмент створений для спрощення процесу завантаження даних, автоматизуючи багато звичних задач, пов'язаних із запитами на сервер та управлінням кешем.

— **Node.js** – це відкрите кросплатформне середовище виконання JavaScript, яке використовує двигун V8 і дозволяє виконувати JavaScript-код поза веббраузером. Використовуючи Node.js, розробники можуть створювати інструменти командного рядка та серверні сценарії. Це означає, що вони можуть виконувати код на серверному боці для генерації динамічного вмісту вебсторінок перед їх відправленням на клієнтський веббраузер користувача. Node.js впроваджує концепцію "JavaScript скрізь", що дозволяє розробникам використовувати одну мову програмування для обох сторін, серверного і клієнтського коду, спрощуючи розробку вебдодатків.

— **PostgreSQL**, також відома як Postgres, є безкоштовною системою керування реляційними базами даних з відкритим вихідним кодом (RDBMS). Вона славиться своєю розширюваністю і відповідністю стандартам мови SQL. PostgreSQL підтримує транзакції з властивостями атомарності, узгодженості, ізоляції і довговічності (ACID), а також має ряд корисних функцій, таких як автоматично розширені подання, матеріалізовані перегляди, тригери, зовнішні ключі та збережені процедури. Ця система призначена для обробки різних завдань, від роботи на одній машині до використання великих сховищ даних або вебсервісів з великою кількістю одночасних користувачів.

— **SASS** – це скриптова мова, яка інтерпретується у каскадних таблицях стилів (CSS). Головна мета SASS - спростити процес розробки CSS і роботи з CSS-файлами. SASS розширює можливості CSS, надаючи кілька механізмів, які зазвичай доступні в більш традиційних мовах програмування, але відсутні у CSS. Інтерпретатор SASS перетворює код на мові SassScript у звичайні CSS-правила. З іншого боку, SASS можна розглядати як синтаксичний цукор або зручний спосіб писати CSS код.

— **Webpack** – це безкоштовний пакет модулів з відкритим вихідним кодом для JavaScript. Основна функція Webpack - збирати і обробляти різні модулі та залежності в JavaScript-проєкті. Цей інструмент призначений в основному для роботи з JavaScript, але він також може обробляти зовнішні ресурси, такі як HTML, CSS та зображення, якщо встановлені відповідні завантажувачі (loaders). Webpack бере на вхід модулі та їх залежності і генерує статичні асети, які представляють ці модулі, допомагаючи забезпечити ефективну організацію та завантаження ресурсів для вебдодатків.

Для того, щоб розробити музичну рекомендаційну систему, було обрано такі засоби розробки:

— **Python** – це високорівнева мова програмування з простим та зрозумілим синтаксисом, що робить її дуже популярною серед розробників. Вона підходить для багатьох задач, включаючи веброзробку, наукові обчислення, автоматизацію завдань і, звісно, для реалізації алгоритмів штучного інтелекту. Python має велику та активну спільноту розробників, що робить його ідеальним вибором для великої кількості проєктів.

— **Flask** – це легкий мікрофреймворк для розробки вебдодатків. Він надає базовий набір інструментів для створення вебсерверів та додатків, дозволяючи розробникам фокусуватися на реалізації функціональності, без зайвої складності.

— **Pandas** – це потужна бібліотека для обробки та аналізу даних в Python. Вона надає структури даних, такі як DataFrame, які дозволяють легко

виконувати операції з даними, включаючи фільтрацію, групування і об'єднання таблиць.

— **NumPy** – це бібліотека для обчислювань з використанням масивів та матриць. Вона дозволяє виконувати чисельні обчислення швидко і ефективно, що робить її ідеальним інструментом для наукових обчислень та обробки даних.

— **Scikit-learn** – це бібліотека для машинного навчання в Python. Вона має велику кількість реалізованих алгоритмів для класифікації, регресії, кластеризації та інших завдань машинного навчання. Scikit-learn також надає інструменти для оцінки та валідації моделей[14].

— **TensorFlow** – це бібліотека для реалізації глибокого навчання та створення нейронних мереж. Вона має велику популярність в галузі штучного інтелекту і дозволяє розробникам створювати складні нейронні мережі для різних завдань, включаючи розпізнавання зображень та обробку природної мови. TensorFlow також має активну спільноту та багато ресурсів для навчання[15].

Висновки до першого розділу

Отже, в першому розділі було проаналізовано методи та засоби розробки вебплатформи для надання музичних відгуків та музичної рекомендаційної системи.

Було проведено аналіз існуючих аналогічних вебсистем, при цьому відібрані були два вебсайти – Metacritic і RateYourMusic. Аналізуючи ці ресурси, було виявлено, що для нашого вебсайту має бути такі функціональності: можливість оцінювання музичних творів користувачами; розрахунок загальної оцінки музичних творів; використання системи кольорових позначок для оцінки; створення зручного та зрозумілого дизайну сайту.

Для досягнення поставлених цілей, зокрема створення ефективної музичної рекомендаційної системи, було проведено аналіз популярних платформ – Spotify та YouTube Music. Обидві ці платформи відзначаються використанням сучасних методів машинного навчання для персоналізації рекомендацій та

покращення користувацького досвіду. Вони надають можливість користувачам вибирати з різноманітних жанрів, виконавців та треків, враховуючи їхні уподобання та історію прослуховування.

Для створення музичної рекомендаційної системи було використано метод колаборативної фільтрації за моделлю. Цей метод базується на аналізі історії прослуховування та поведінки користувачів для знаходження схожих музичних уподобань. Він дозволяє рекомендувати треки, які можуть бути цікавими для конкретного користувача, на основі його спільності у смаках з іншими користувачами. Це доповнює підхід персоналізації за жанрами та виконавцями та забезпечує більш глибоке та різноманітне рекомендування музичних треків. Також, додатково, було вирішено створити нейронну мережу.

Для реалізації музичної вебплатформи в цьому розділі було обрано такі інструменти: React, Redux Toolkit, RTK Query, Node.js, PostgreSQL, SASS, Webpack.

З ціллю розробити музичну рекомендаційну систему, було вирішено використати такі засоби: Python, Flask, Pandas, NumPy, Scikit-learn, TensorFlow.

РОЗДІЛ 2.ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ТА МОДЕЛІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

2.1. Проєктування бази даних вебплатформи для надання музичних відгуків

Для зберігання відгуків, музичних творів, виконавців, повідомлень, користувачів та лайків у музичні вебплатформі була створена база даних. В даному випадку використовувалася система керування базами даних PostgreSQL. Вона має численні переваги, такі як високопродуктивні механізми транзакцій, надійність, вбудовану підтримку низькоструктурованих даних у форматі JSON і можливість створення нових типів даних. На рисунку 2.1 показана структура бази даних, яка була спроектована для цієї вебсистеми.

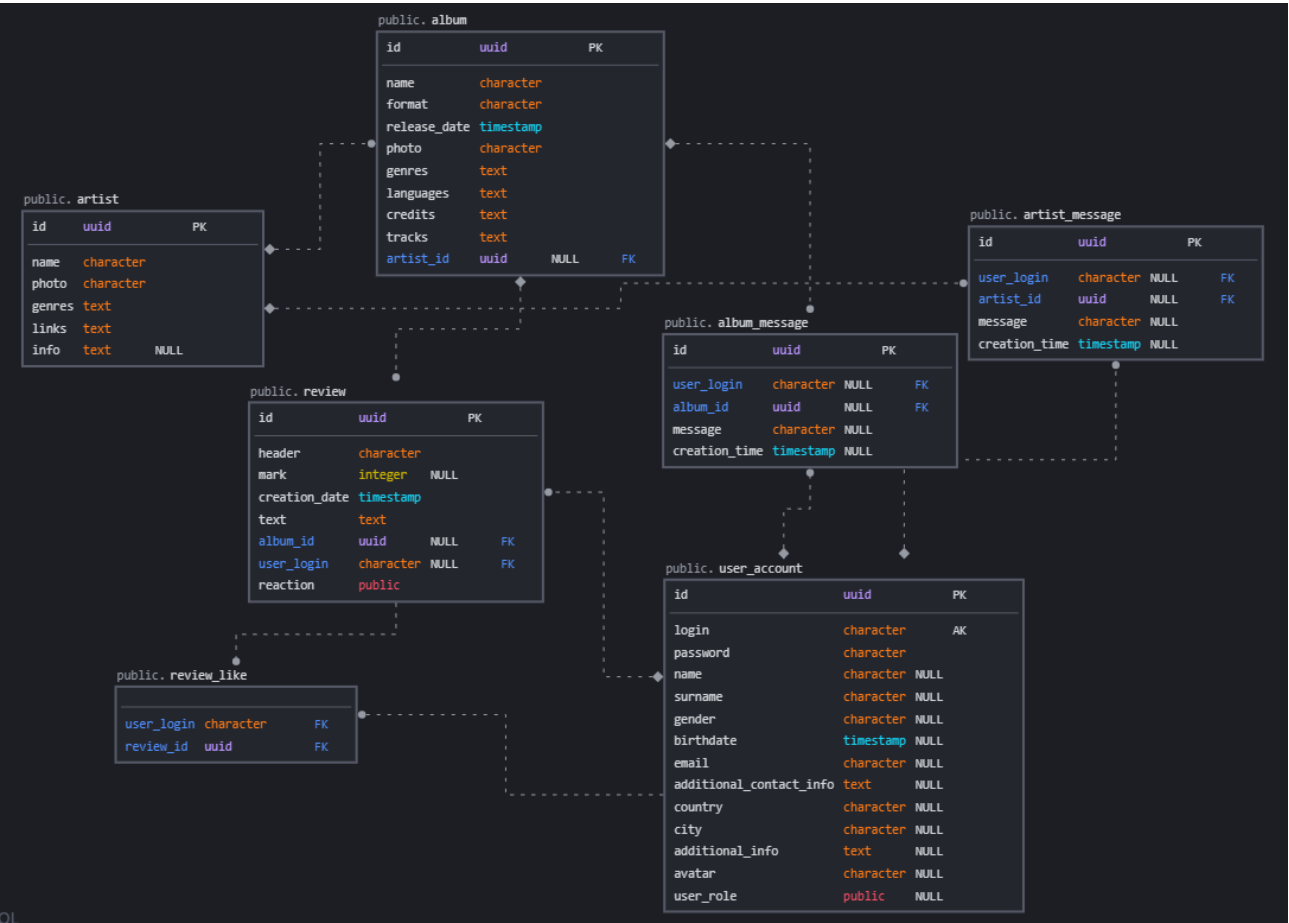


Рис. 2.1. База даних для соціальної мережі для меломанів

Для збереження даних користувачів в базі даних використовується таблиця з назвою "user_account". У цій таблиці є поле "user_role", яке використовує перелік (enum) ролей з двома можливими значеннями: "user" та "moderator". Структура таблиці представлена в таблиці 2.1.

Таблиця 2.1

Таблиця «user_account»

<i>Назва</i>	<i>Тип</i>	<i>ПК</i>	<i>ЗК</i>	<i>Опис поля</i>
id	uuid	+	-	Унікальний ідентифікатор користувача
login	character varying(20)	-	-	Логін користувача, який використовується для авторизації
name	character varying(50)	-	-	Ім'я
surname	character varying(50)	-	-	Прізвище
gender	character varying(10)	-	-	Стать
password	character varying(20)	-	-	Пароль користувача для авторизації
birthdate	timestamp	-	-	Дата народження
email	character varying(50)	-	-	Пошта
additional_contact_info	text	-	-	Додаткова контактна інформація

Продовження таблиці 2.1

country	character varying(50)	-	-	Країна проживання
city	character varying(50)	-	-	Населений пункт
additional_info	text	-	-	Додаткова інформація
avatar	character varying(20)	-	-	Ім'я файлу з фото профілю користувача
user_role	role	-	-	Роль користувача, яка може мати значення "user" або "moderator"

Для збереження даних відгуків в базі даних створено таблицю з назвою "review". У цій таблиці існує поле "reaction", яке використовує перелік (enum) реакцій з трьома можливими значеннями: "good" (позитивна реакція), "meh" (нейтральна реакція) та "bad" (негативна реакція). Деталі структури таблиці наведені в таблиці 2.2.

Таблиця 2.2

Таблиця «review»

Назва	Тип	ПК	ЗК	Опис поля
id	uuid	+	-	Унікальний ідентифікатор відгуку
header	character varying(20)	-	-	Заголовок
mark	integer	-	-	Оцінка

Продовження таблиці 2.2

creation_date	timestamp	-	-	Дата та час створення відгуку
user_login	character varying(20)	-	+	Логін користувача
reaction	reaction	-	-	Враження від музичного твору, яке може мати значення "good", "meh" або "bad"
album_id	uuid	-	+	Унікальний ідентифікатор музичного твору

Для збереження даних про музичні твори в базі даних створена таблиця з назвою "album". Структура цієї таблиці наведена в таблиці 2.3.

Таблиця 2.3

Таблиця «album»

Назва	Тип	ПК	ЗК	Опис поля
id	uuid	+	-	Унікальний ідентифікатор музичного твору
name	character varying(20)	-	-	Назва
format	character varying(20)	-	-	Формат

Продовження таблиці 2.3

release_date	timestamp	-	-	Дата виходу
genres	text	-	-	Жанри
languages	text	-	-	Мови
credits	text	-	-	Інформація про людей, що брали участь у створенні музичного проєкту
photo	character varying(20)	-	-	Ім'я файлу з фотографією музичного твору
tracks	text	-	-	Треки
artist_id	uuid	-	+	Унікальний ідентифікатор виконавця

Для збереження даних про виконавців музичних творів в базі даних створена таблиця з назвою "artist". Структура цієї таблиці наведена в таблиці 2.4.

Таблиця 2.4

Таблиця «artist»

Назва	Тип	ПК	ЗК	Опис поля
id	uuid	+	-	Унікальний ідентифікатор виконавця
name	character varying(20)	-	-	Назва

Продовження таблиці 2.4

photo	character varying(20)	-	-	Ім'я файлу з фотографією виконавця
links	text	-	-	Посилання
info	text	-	-	Опис
genres	text	-	-	Жанри

Для збереження даних про повідомлення на сторінці виконавця музичного твору створена таблиця з назвою "artist_message". Структура цієї таблиці наведена в таблиці 2.5.

Таблиця 2.5

Таблиця «artist_message»

Назва	Тип	ПК	ЗК	Опис поля
id	uuid	+	-	Унікальний ідентифікатор повідомлення у форматі UUID
artist_id	uuid	-	+	Унікальний ідентифікатор виконавця
user_login	character varying(20)	-	+	Логін користувача, який створив це повідомлення
message	text	-	-	Текстове поле, яке містить текст повідомлення

Продовження таблиці 2.5

creation_date	timestamp	-	-	Дата та час створення повідомлення
---------------	-----------	---	---	------------------------------------

Для збереження даних про повідомлення на сторінці музичного твору створена таблиця з назвою "album_message". Структура цієї таблиці наведена в таблиці 2.6.

Таблиця 2.6

Таблиця «album_message»

Назва	Тип	ПК	ЗК	Опис поля
id	uuid	+	-	Унікальний ідентифікатор повідомлення
user_login	character varying(20)	-	+	Логін користувача, який створив це повідомлення
message	text	-	-	Текст повідомлення
creation_date	timestamp	-	-	Дата створення повідомлення
album_id	uuid	-	+	Унікальний ідентифікатор музичного твору

Для збереження даних про вподобайки на відгуках створена таблиця з назвою "review_like". Структура цієї таблиці наведена в таблиці 2.7.

Таблиця 2.7

Таблиця «review_like»

Назва	Тип	ПК	ЗК	Опис поля
review_id	uuid	-	+	Код відгуку
user_login	character varying(20)	-	+	Логін користувача

2.2. Архітектура та узагальнена структура музичної вебплатформи

Уніфікована мова моделювання (UML) діаграма компонентів використовується для візуалізації того, як компоненти програмної системи об'єднуються разом для створення більших компонентів або складних програмних систем. Ця діаграма допомагає ілюструвати структуру складних систем.

На рисунку 2.2 показана фронтенд-частина програмної системи у вигляді діаграми компонентів. Ця діаграма включає наступні елементи:

Source:

- *Pages*: Компоненти, які представляють окремі сторінки вебсайту.
- *Components*: Різні компоненти, які можуть використовуватися на різних сторінках сайту.
- *App.jsx*: Головний компонент вебсайту, який є точкою входу в додаток.
- *Index.jsx*: Головний вхідний компонент додатку.

Public:

- *css*: Каталог, де зберігаються стилі вебсайту.
 - *mixins*: Файли з повторюваними частинами стилів.
 - *pages*: Стилi, які використовуються на сторінках.
 - *components*: Стилi для різних компонентів.
 - *fonts.scss*: Файл, що містить налаштування шрифтів для сайту.
 - *variables.scss*: Файл, який містить змінні та налаштування для

стилів сайту.

— *app.scss*: Стилі для головного компонента вебсайту.

— *font*: Каталог, де зберігаються шрифти сайту.

— *images*: Каталог, де знаходяться зображення, які використовуються на вебсайті.

— *js*: Каталог, де розміщуються JavaScript-файли для фронтенду додатку.

— *index.html*: Головний файл HTML для вебсайту.

— *index.scss*: Головний файл стилів для всього додатку.

Store:

— *apis*: Каталог, де знаходяться файли для взаємодії з API.

— *slices*: Слайси Redux для керування станом користувача.

— *store.js*: Файл з конфігурацією Redux Store для керування станом додатку.

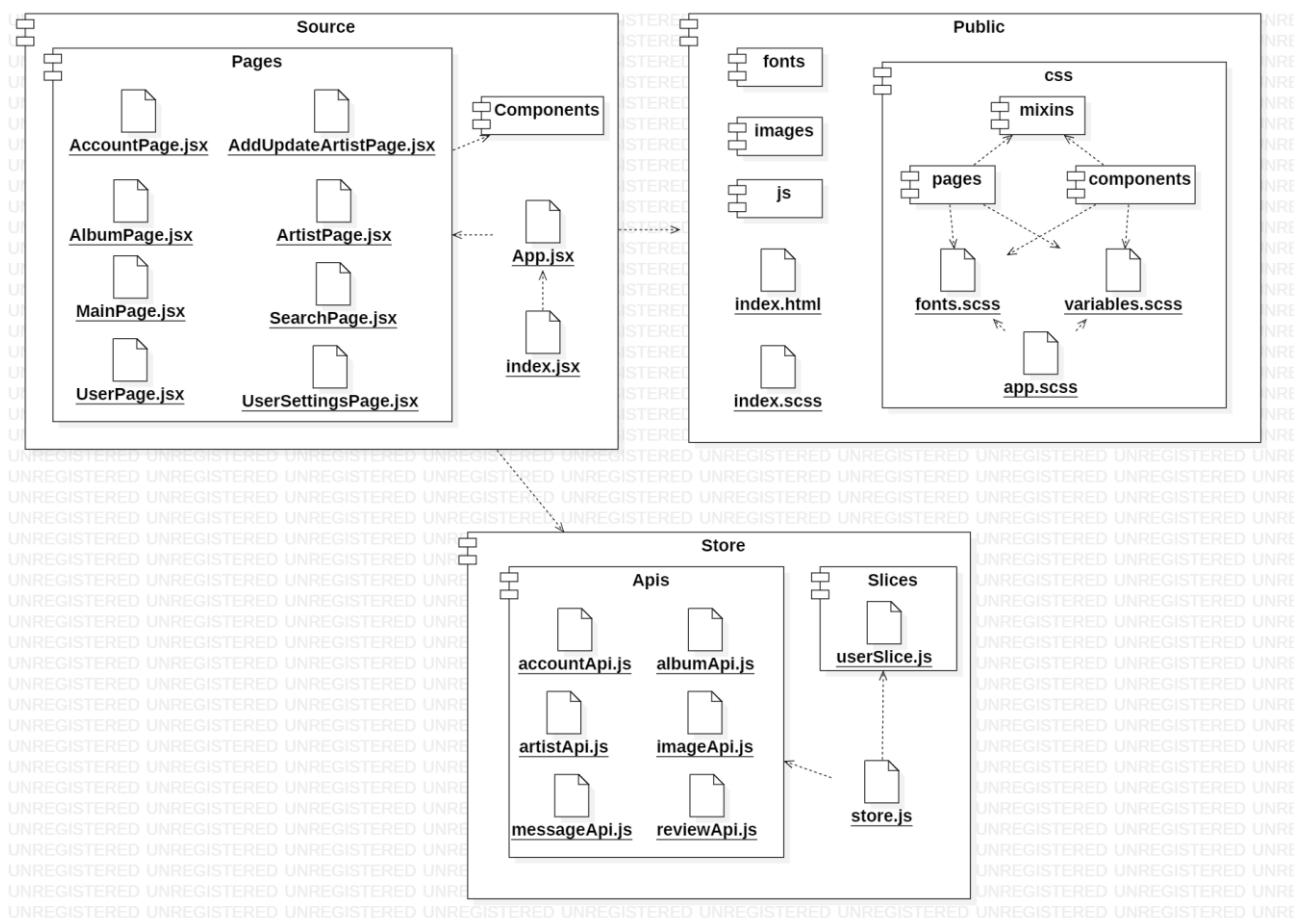


Рис. 2.2. Діаграма компонентів для фронтенд-частини

На рисунку 2.3 можна побачити структуру бекенд-частини у формі діаграми компонентів, де присутні такі складові елементи:

Models:

- *Controllers* відповідають за логіку керування моделями та обробку запитів від користувачів.
- *Routes* визначають шляхи, за якими можна звертатися до різних частин додатку.
- *Config* містить конфігураційні файли, що регулюють різні аспекти додатку, такі як налаштування сервера бази даних, завантаження файлів на сервер, а також налаштування чату.
- *App.js* є головним файлом програми, який об'єднує всі компоненти та визначає основну логіку додатку.

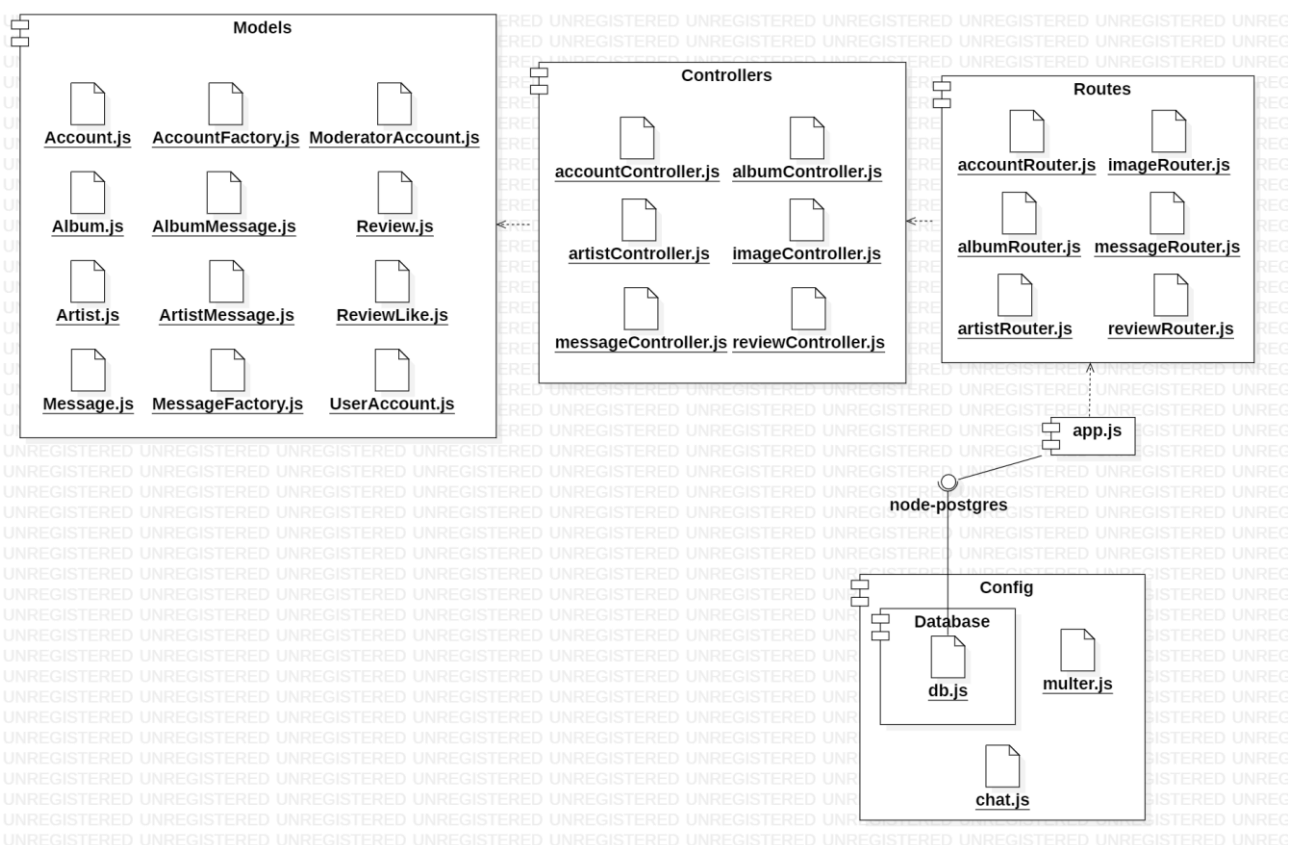


Рис. 2.3. Діаграма компонентів для бекенд-частини

Для відображення структури системи використаємо діаграму розгортання. Ця діаграма відтворює фізичний розташунок компонентів програмного та апаратного забезпечення на окремих вузлах. В контексті вебсайту, наприклад,

діаграма розгортання вказує, які апаратні компоненти (такі як вебсервер, сервер додатків та сервер бази даних) існують, які програмні компоненти (наприклад, вебдодаток, база даних) запускаються на кожному вузлі, і як вони взаємодіють між собою для забезпечення роботи системи. Діаграма розгортання (див. рис. 2.4) має наступну структуру:

- *Client*: Це включає в себе статичні файли публічної частини (Public), розроблені компоненти React (JSX-Components) та компонент для зберігання стану вебсистеми – Redux-store.
- *API Server*: В цей компонент входять моделі вебдодатку (Models), контролери (Controllers) та шляхи доступу до API (Routes). Він відповідає за обробку запитів користувачів і взаємодіє з базою даних та фронтенд-частиною.
- *Database*: Використовується PostgreSQL для зберігання та управління даними, необхідними для функціонування вебсистеми.

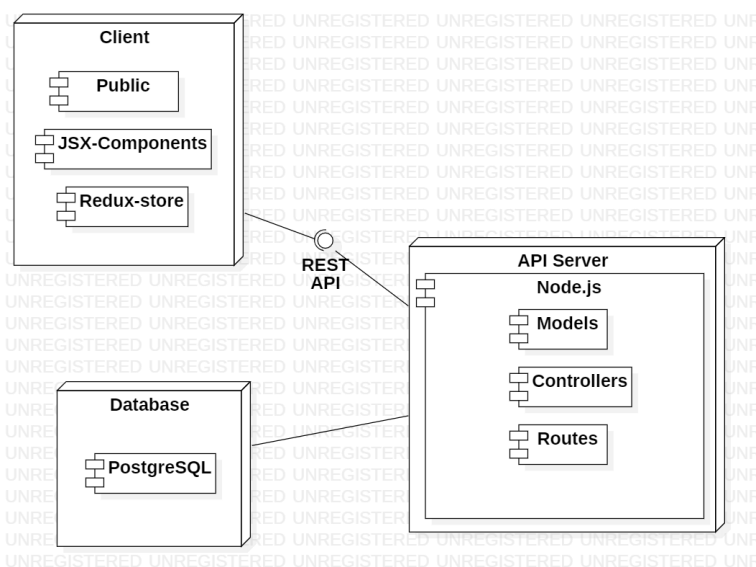


Рис. 2.4. Діаграма розгортання

2.3. Розробка музичної рекомендаційної системи

Для того, щоб розробити музичну рекомендаційну систему, потрібно отримати дані з бази даних. Спершу, напишемо клас для отримання даних з БД:

```
from dotenv import load_dotenv
from sqlalchemy import create_engine
```

```
load_dotenv()

class Database:
    def __init__(self, user: str, password: str, host: str, db_name: str, port: int = 5432):
        db_string = f"postgresql+psycopg2://{user}:{password}@{host}:{port}/{db_name}"
        self.engine = create_engine(db_string)

    def __del__(self):
        self.engine.dispose()
```

Цей код реалізує клас для з'єднання з базою даних PostgreSQL за допомогою SQLAlchemy, починаючи з імпорту необхідних модулів. Перш за все, він імпортує `load\_dotenv` з модуля `dotenv`, що дозволяє завантажити змінні оточення з файлу `.env`. Також імпортується `create\_engine` з SQLAlchemy, що використовується для створення з'єднання з базою даних.

Після імпорту, код викликає `load\_dotenv()`, щоб завантажити змінні оточення. Далі йде визначення класу `Database`, який приймає параметри, необхідні для підключення до бази даних, включаючи ім'я користувача, пароль, адресу хоста, назву бази даних та порт (зі стандартним значенням 5432 для PostgreSQL).

У конструкторі класу (`\_\_init\_\_`) формується рядок з'єднання, який включає всі передані параметри і вказує на використання psycopg2 як драйвера для PostgreSQL.

На завершення, у класі визначено метод деструктора (`\_\_del\_\_`), який викликається при знищенні об'єкта класу. У цьому методі використовується `self.engine.dispose()`, що допомагає коректно закрити з'єднання з базою даних та звільнити ресурси, пов'язані з двигуном.

Після цього, можна перейти до створення музичної рекомендаційної системи. Для цього створимо спеціальний клас MusicRecommenderSystem.

```
class MusicRecommenderSystem:

    def __init__(self, db_engine: Engine):
        self.result_precision = 0.5
        self.album_ratings_df = pd.read_sql('SELECT mark, user_login, album_id FROM review', db_engine)
        self.album_artist_df = pd.read_sql('SELECT album.id AS album_id, artist.id AS artist_id FROM album \
```

```

                                JOIN artist ON album.artist_id =
artist.id', db_engine)
    self.album_ratings_df = self.album_ratings_df.dropna()
    self.combine_rating = pd.merge(self.album_ratings_df, self.album_artist_df,
on='album_id')

    self.user_ids =
self.combine_rating['user_login'].astype("category").cat.codes.values
    self.ratings = self.album_ratings_df['mark'].values

```

В першу чергу, потрібно задати рекомендаційній системі дані для аналізу та навчання. Конструктор приймає як параметр об'єкт Engine з SQLAlchemy, який забезпечує зв'язок з базою даних. Цей зв'язок використовується для витягування даних, необхідних для системи рекомендацій.

У класі встановлюється параметр `result_precision` зі значенням 0.5, що буде використано пізніше для визначення точності рекомендацій.

З бази даних завантажуються два набори даних: оцінки альбомів (включаючи інформацію про користувачів та ідентифікатори альбомів) та відносини між альбомами та виконавцями (ідентифікатори альбомів та виконавців). Завантажені дані очищаються від пропущених значень.

Після завантаження ці два набори даних об'єднуються за допомогою ідентифікатора альбому.

Далі, логіни користувачів перетворюються у числові коди, оскільки більшість алгоритмів машинного навчання працює ефективніше з числовими даними. Цей процес допомагає у подальшому аналізі та генерації рекомендацій.

На завершення, зберігається масив оцінок альбомів, який буде використовуватися в системі рекомендацій.

Далі перейдемо до розробки першої моделі рекомендаційної системи, що ґрунтується на матричній факторизації.

```

def __total_reviews_count_by_album(self):
    self.combine_rating = self.combine_rating.dropna(axis=0, subset=['album_id'])

    album_ratingCount = (self.combine_rating
        .groupby(by=['album_id'])['mark']
        .count()
        .reset_index()
        .rename(columns={'mark': 'totalRatingCount'}))
)

```

```
rating_with_totalRatingCount = self.combine_rating.merge(
    album_ratingCount, left_on = 'album_id', right_on = 'album_id', how = 'left'
)
self.user_rating =
rating_with_totalRatingCount.drop_duplicates(['user_login', 'album_id'])
```

Метод `__total_reviews_count_by_album` спочатку очищає об'єднаний набір даних `combine_rating` від будь-яких пропущених значень у стовпці `album_id`. Після цього він групує дані за ідентифікаторами альбомів і підраховує кількість відгуків для кожного альбому. Ця інформація далі об'єднується з початковим набором даних, надаючи кожному запису відомості про загальну кількість відгуків на альбом. На заключному етапі метод видаляє дублікати записів, залишаючи унікальні комбінації користувачів та альбомів.

```
def __create_album_matrix(self):
    self.album_user_rating_pivot = self.user_rating.pivot(index = 'user_login',
columns = 'album_id', values = 'mark').fillna(0)
    self.X = self.album_user_rating_pivot.values.T
```

Метод `__create_album_matrix` створює матрицю, де кожен рядок представляє користувача, а кожен стовпець - окремий альбом. В клітинах матриці знаходяться оцінки, які користувачі дали цим альбомам. Цей процес починається з перетворення вихідного DataFrame `user_rating` у pivot table, де індексами є ідентифікатори користувачів, а стовпцями - ідентифікатори альбомів. Пропущені значення в цій таблиці заповнюються нулями, оскільки відсутність оцінки може бути інтерпретована як нейтральна позиція. На завершення, виконується транспонування матриці, так що кожен рядок відповідає альбому, а стовпці - різним користувачам.

```
def __fit_model(self):
    SVD = TruncatedSVD(n_components=12, random_state=17)
    matrix = SVD.fit_transform(self.X)
    self.corr = np.corrcoef(matrix)
```

Метод `__fit_model` в контексті системи рекомендацій музики використовує алгоритм зменшення розмірності даних, зокрема Truncated SVD (Singular Value Decomposition), для створення моделі схожості між різними елементами, такими як альбоми або виконавці.

Спочатку метод створює екземпляр класу Truncated SVD з бібліотеки `scikit-learn`, налаштовуючи його на зменшення даних до 12 вимірів. Це важливо,

оскільки вихідні дані, як правило, можуть бути високовимірними і складними для аналізу. Заданий параметр `random_state` гарантує, що процес зменшення розмірності буде відтворюваним.

Після ініціалізації SVD, метод застосовує цей алгоритм до підготовленої матриці `self.X`, яка містить дані про взаємодії користувачів з музичними творами. Процес `fit_transform` перетворює цю матрицю, виділяючи ключові складові, які найкраще відображають взаємозв'язки у вихідних даних.

На завершальному етапі, метод обчислює коефіцієнти кореляції для отриманої матриці, використовуючи функцію `np.corrcoef` з NumPy. Результатом є матриця кореляцій, де кожен елемент показує, наскільки сильно одна складова пов'язана з іншою. Ця матриця є цінною для системи рекомендацій, оскільки вона допомагає визначити, які елементи (альбоми або виконавці) є схожими один на одного, і використовується для генерації точних рекомендацій.

```
def __check_album_results(self, album_id: str) -> list:
    album_ids = self.album_user_rating_pivot.columns
    album_id_list = list(map(str, album_ids))
    album_index = album_id_list.index(album_id)
    corr_album_index = self.corr[album_index]
    return list(map(str, album_ids[corr_album_index >= self.result_precision]))
```

У класі системи рекомендацій музики, два методи `__check_album_results` і `__check_artist_results` використовуються для виявлення схожих альбомів.

Метод `__check_album_results` починається з визначення усіх існуючих ідентифікаторів альбомів. Він шукає конкретний альбом у цьому списку та використовує його індекс для отримання відповідного рядка у матриці кореляцій. Кожен елемент у цьому рядку відображає кореляцію між заданим альбомом та іншими альбомами. Метод вибирає альбоми, чия кореляція з заданим альбомом перевищує певний поріг, встановлений як міра схожості. В результаті, він повертає список ідентифікаторів цих схожих альбомів.

Аналогічний ряд методів створений для визначення рекомендацій для виконавців.

Далі перейдемо до створення моделі нейронної мережі.

```
def __create_neural_network_model(self, type):
    embedding_size = 50
```

```

        if type == 'album':
            item_id = self.combine_rating.album_id
            self.item_ids =
self.combine_rating['album_id'].astype("category").cat.codes.values
        else:
            item_id = self.combine_rating.artist_id
            self.item_ids =
self.combine_rating['artist_id'].astype("category").cat.codes.values

        user_input = Input(shape=[1], name="User-Input")
        user_embedding =
Embedding(input_dim=self.combine_rating_df.user_login.nunique(),
            output_dim=embedding_size,
            name="User-Embedding")(user_input)
        user_vec = Flatten(name="Flatten-Users")(user_embedding)
        item_input = Input(shape=[1], name="Item-Input")
        item_embedding = Embedding(input_dim=item_id.nunique(),
            output_dim=embedding_size,
            name="Item-Embedding")(item_input)
        item_vec = Flatten(name="Flatten-Items")(item_embedding)

        conc = Concatenate()([user_vec, item_vec])

        fc1 = Dense(256, activation='relu')(conc)
        drop1 = Dropout(0.3)(fc1)
        fc2 = Dense(128, activation='relu')(drop1)
        drop2 = Dropout(0.3)(fc2)
        fc3 = Dense(64, activation='relu')(drop2)
        out = Dense(1)(fc3)

        self.nn_model = Model([user_input, item_input], out)
        self.nn_model.compile('adam', 'mean_squared_error')

```

Метод `__create_neural_network_model` у класі, призначеному для системи рекомендацій музики, створює нейронну мережу для прогнозування оцінок, які користувачі можуть дати альбомам або виконавцям. Цей метод використовує техніку вбудовування (embedding), яка дозволяє перетворити ідентифікатори користувачів і музичних елементів на щільні вектори нижчої розмірності.

Спочатку метод визначає, чи буде модель працювати з альбомами чи виконавцями, в залежності від переданого параметра `type`. На основі цього вибору ідентифікатори альбомів або виконавців конвертуються в числові коди для подальшого використання в моделі.

Далі створюються два вбудовані шари: один для користувачів і один для елементів (альбомів або виконавців). Кожен користувач і кожен елемент представляються вектором фіксованої розмірності (у даному випадку 50). Ці вектори потім вирівнюються до одновимірного вигляду.

Після цього вбудовані вектори користувачів та елементів об'єднуються, формуючи основу для подальших шарів нейронної мережі. Мережа містить кілька повнозв'язних шарів із активаційною функцією ReLU, а також шари Dropout, які допомагають запобігти перенавчанню моделі.

В кінці, останній шар мережі виводить єдине числове значення, яке є прогнозованою оцінкою користувача для даного елемента. Модель компілюється з використанням оптимізатора Adam і функції втрати mean squared error, що є стандартним вибором для задач регресії.

```
def __train_neural_network(self, epochs=5):
    self.nn_model.fit([self.user_ids, self.item_ids], self.ratings, epochs=epochs)
```

Метод `__train_neural_network` у класі системи рекомендацій музики відповідає за тренування нейронної мережі, створеної у попередньому методі. Цей процес включає подачу даних у мережу та її навчання на цих даних протягом заданої кількості епох.

```
def get_nn_recommendations(self, item_id: str, type: str, top_n: int = 10):
    self.__create_neural_network_model(type)
    self.__train_neural_network()
    item_index = self.combine_rating[self.combine_rating[type + '_id'] ==
item_id][type + '_id'].astype("category").cat.codes.values[0]
    item_array = np.array([item_index] * len(self.user_ids))
    predictions = self.nn_model.predict([item_array, self.item_ids])

    top_items = np.argsort(predictions[:,0])[::-1][:top_n]
    recommended_item_ids = self.combine_rating.iloc[top_items][type + '_id'].values
    return recommended_item_ids.tolist()
```

Метод `get_nn_recommendations` у класі системи рекомендацій музики використовує навчену нейронну мережу для генерації рекомендацій. Цей процес включає декілька кроків, щоб визначити, які музичні елементи (альбоми або виконавці) найбільше можуть зацікавити користувача на основі заданого елемента.

Спочатку метод створює та тренує нейронну мережу, використовуючи існуючі дані оцінок. Він викликає `__create_neural_network_model` з параметром `type`, який вказує, чи повинні рекомендації базуватися на альбомах або виконавцях, а потім тренує модель за допомогою `__train_neural_network`.

Далі метод визначає позицію заданого елемента (альбому або виконавця) у навчальному наборі даних. Це робиться шляхом знаходження індексу заданого ідентифікатора елемента (`item_id`) серед відповідних ідентифікаторів у наборі даних.

На наступному кроці метод генерує масив, що містить індекс цього елемента, повторений стільки разів, скільки є користувачів у системі. Цей масив використовується разом з ідентифікаторами користувачів для виконання прогнозування за допомогою нейронної мережі.

Мережа прогнозує оцінки, які кожен користувач міг би дати цьому конкретному елементу. Після отримання прогнозів, метод вибирає топ-N елементів з найвищими прогнозованими оцінками. Це робиться за допомогою сортування прогнозів та вибору ідентифікаторів елементів, які відповідають найвищим прогнозованим оцінкам.

На заключному кроці метод повертає список ідентифікаторів цих топ-N елементів як рекомендовані.

```
def get_recommendations_by_album(self, album_id: str) -> list:
    self.__total_reviews_count_by_album()
    self.__create_album_matrix()
    self.__fit_model()

    svd_recommendations = set(self.__check_album_results(album_id))
    nn_recommendations = set(self.get_nn_recommendations(album_id, 'album'))
    recommendations = svd_recommendations.intersection(nn_recommendations)

    return recommendations
```

Метод `get_recommendations_by_album` у класі системи рекомендацій музики забезпечує виведення рекомендацій на основі конкретного альбому. Цей метод поєднує дві техніки для знаходження рекомендацій: матричну факторизацію (SVD) та прогнозування за допомогою нейронної мережі.

Спочатку метод виконує ряд підготовчих кроків, які включають обчислення загальної кількості оцінок для кожного альбому, створення матриці користувач-альбом і застосування SVD для виявлення латентних факторів в цих даних.

Потім, використовуючи результати SVD, метод визначає набір альбомів, які кореляційно схожі на заданий альбом. Паралельно, метод також застосовує нейронну мережу для генерації прогнозованих рекомендацій, використовуючи інформацію про взаємодії між користувачами та альбомами.

На останньому етапі метод знаходить перетин між рекомендаціями, отриманими від SVD та нейронної мережі. Цей перетин включає альбоми, які рекомендовані обома методами, забезпечуючи таким чином більш надійні та точні рекомендації.

Аналогічно визначаються рекомендації і для виконавців.

Повний код програми, що визначає рекомендації, можете переглянути в додатках А, Б та В.

Висновки до другого розділу

У цьому розділі було реалізовано базу даних для вебсистеми. Структура бази даних включає сім таблиць, призначених для зберігання даних про користувачів, відгуки, музичні твори, виконавців, лайки, а також повідомлення на сторінках окремих музичних творів та виконавців.

Після цього етапу було зосереджено увагу на розробці архітектури вебдодатку. Використання діаграм компонентів та діаграм розгортання сприяло детальному плануванню структури фронтенду та бекенду вебплатформи. Ці візуальні засоби спростили процес визначення загальної архітектури сайту.

Також було успішно реалізована музичну рекомендаційну систему. Система інтегрує різноманітні методи обробки даних та алгоритми машинного навчання, включаючи матричну факторизацію та нейронні мережі, що дозволяє виявляти складні взаємозв'язки між користувачами та музичними творами.

Основні етапи реалізації включали збір та попередню обробку даних, розробку моделі прогнозування на основі нейронної мережі, а також використання методів зменшення розмірності для підвищення ефективності обчислень. Значною перевагою системи є її здатність адаптуватися до індивідуальних уподобань користувачів, пропонуючи їм високоякісні та відповідні музичні рекомендації.

Отже, реалізована музична рекомендаційна система є свідченням потенціалу сучасних методів машинного навчання та аналізу даних у створенні персоналізованого користувацького досвіду. Її подальше вдосконалення та адаптація можуть відкрити нові горизонти у сфері музичних сервісів та цифрових рекомендацій.

РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ ВЕБПЛАТФОРМИ ДЛЯ НАДАННЯ МУЗИЧНИХ ВІДГУКІВ

3.1. Порядок роботи вебплатформи для надання музичних відгуків

Для доступу до повного функціоналу вебсайту, користувачам необхідно пройти процес реєстрації. При першому відвідуванні сайту вони зустрінуться з інтерфейсом, який пропонує дві опції: вхід у систему для існуючих користувачів та створення нового облікового запису для новачків. Цей інтерфейс представлений у вигляді форми, яка є зручною для навігації. Як приклад, на рисунку 3.1 можна побачити зображення форми входу, спеціально розробленої для соціальної мережі, призначеної для любителів музики.

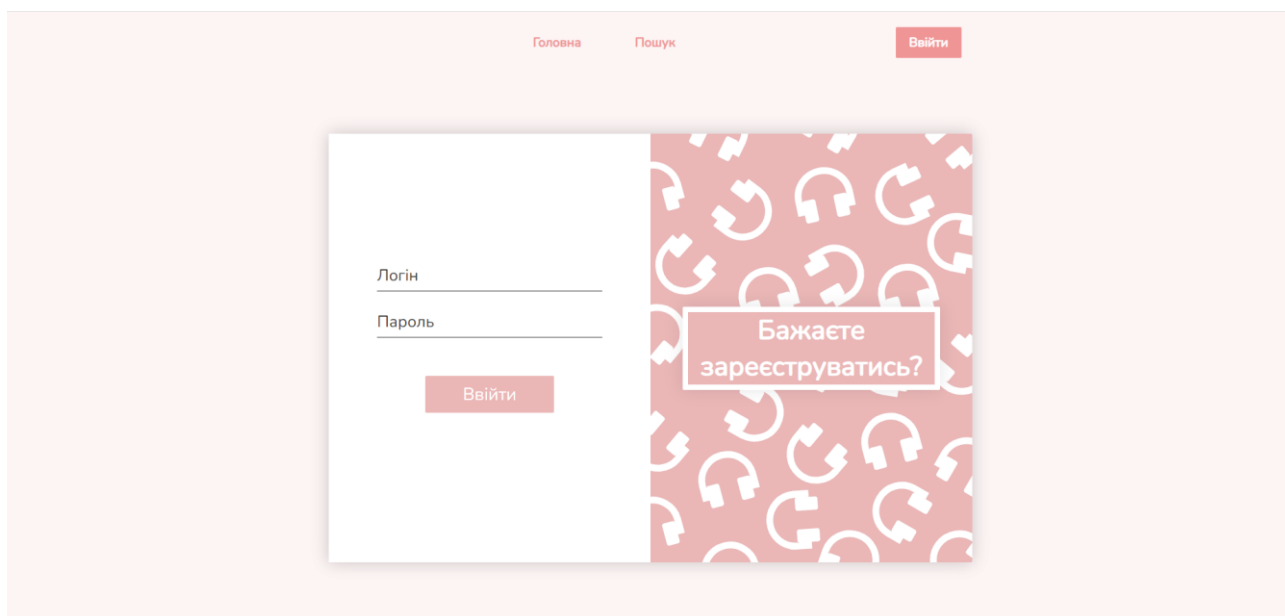


Рис. 3.1. Форма входу на сайт

Після вибору опції "Бажаєте зареєструватись", користувач перенаправляється до процесу реєстрації. На першому етапі з'являється форма для введення базових даних облікового запису. Користувач вводить необхідну інформацію, а потім може або перейти до наступного кроку, натиснувши "Далі", або повернутися до попередньої сторінки за допомогою кнопки "Повернутись". Після переходу на наступний крок з'являється форма для введення особистої інформації. Цей процес продовжується через декілька етапів, після чого

реєстрація завершується. Для ілюстрації, рисунок 3.2 демонструє приклад однієї з форм реєстрації.

Рис. 3.2. Форма з персональними даними

Після завершення процесу реєстрації користувачі отримують можливість увійти на свою персональну сторінку, як це ілюстровано на рисунку 3.3. На цій сторінці відображаються фотографія профілю, інформація про користувача, а також його відгуки. Щоб переглядати деталі профілю, можна використовувати стрілки для навігації вправо або вліво.

В верхній частині сторінки розташована панель навігації, яка дозволяє перейти на головну сторінку користувача, відкрити налаштування профілю або вийти з акаунта. Крім того, на сторінці користувача є спеціальний розділ, що надає функціональність для фільтрації та сортування відгуків. Всі ці елементи роблять навігацію по сайту інтуїтивно зрозумілою та зручною.

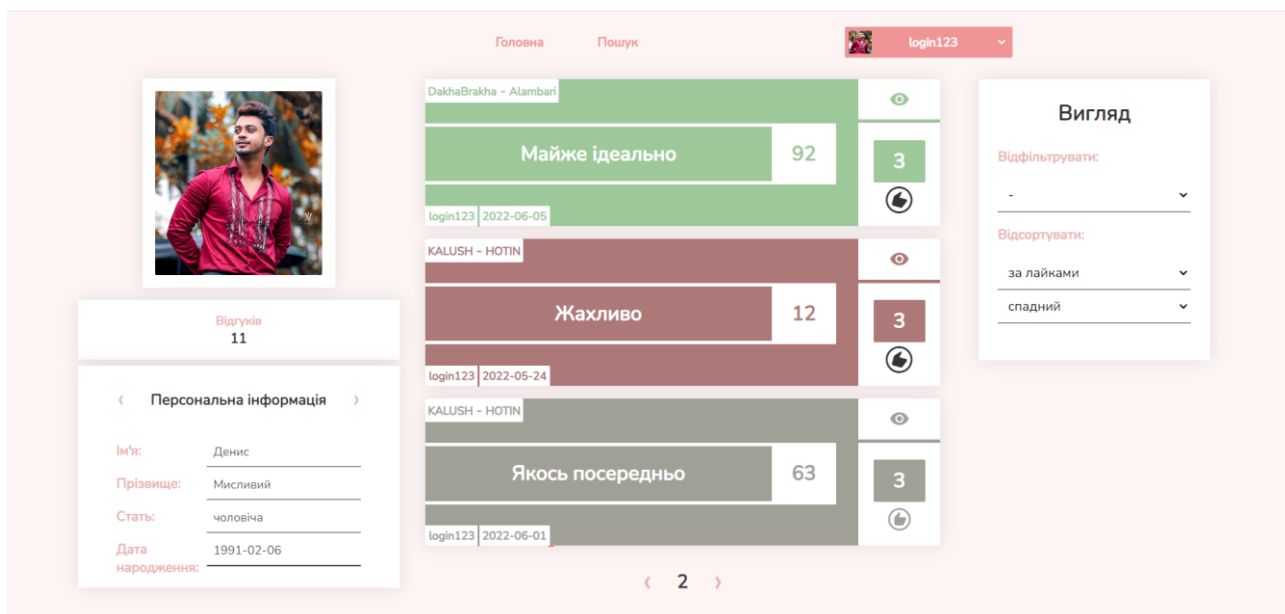


Рис. 3.3. Сторінка користувача

Щоб написати відгук на музичний твір, спершу необхідно його знайти на сайті. Для цього користувачі можуть скористатися панеллю навігації, яка дозволяє перейти на спеціальну сторінку пошуку. На цій сторінці розміщена пошукова стрічка, а також ряд критеріїв для пошуку, серед яких можна вибрати музичний твір, виконавця, або інших користувачів. Пошук музичних творів та виконавців відбувається за їх назвами, тоді як для знаходження користувачів використовується логін.

Для пошуку достатньо ввести запит у пошукову стрічку та натиснути клавішу Enter або скористатися кнопкою пошуку. Після обробки запиту, сайт відображає результати, серед яких можна вибрати і перейти на сторінку обраного музичного твору. Цей процес та інтерфейс сторінки пошуку демонструється на рисунку 3.4.

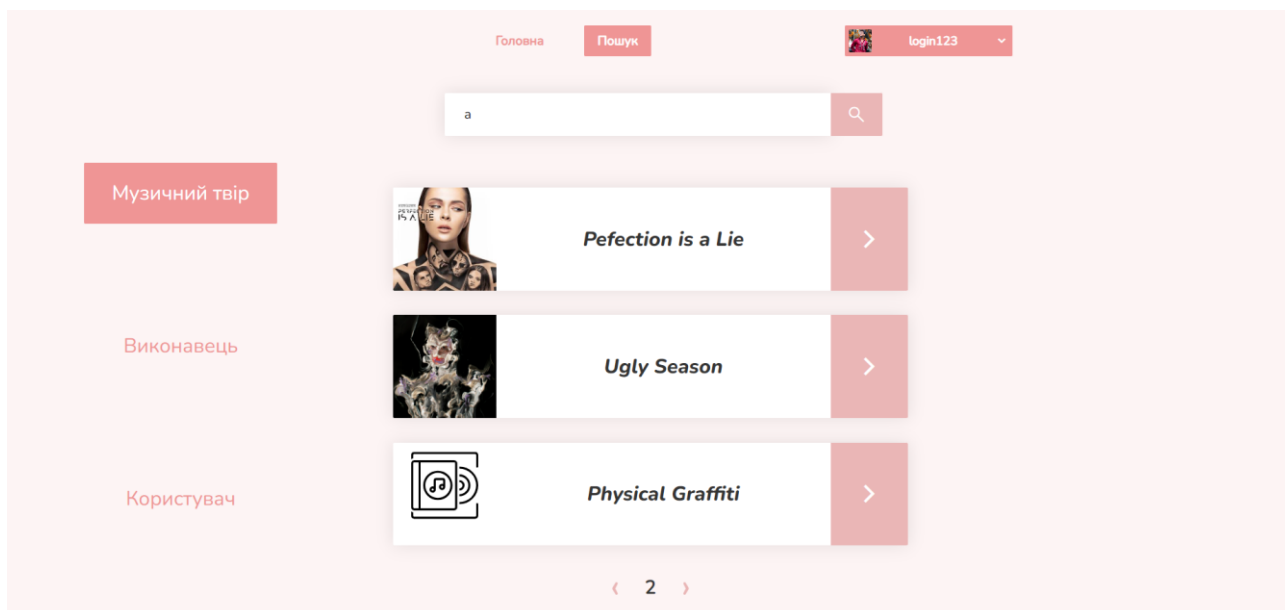


Рис. 3.4. Сторінка пошуку

Після переходу на сторінку конкретного музичного твору, користувачам відкривається інформація про цей твір, включно з деталями та описом. На цій сторінці також доступна кнопка, яка дозволяє додати відгук на музичний твір. Для зручності користувачів, сторінка оснащена навігаційними елементами, що полегшують перехід між різними секціями сайту. Загальний вигляд та розміщення елементів на сторінці музичного твору можна побачити на рисунку 3.5.

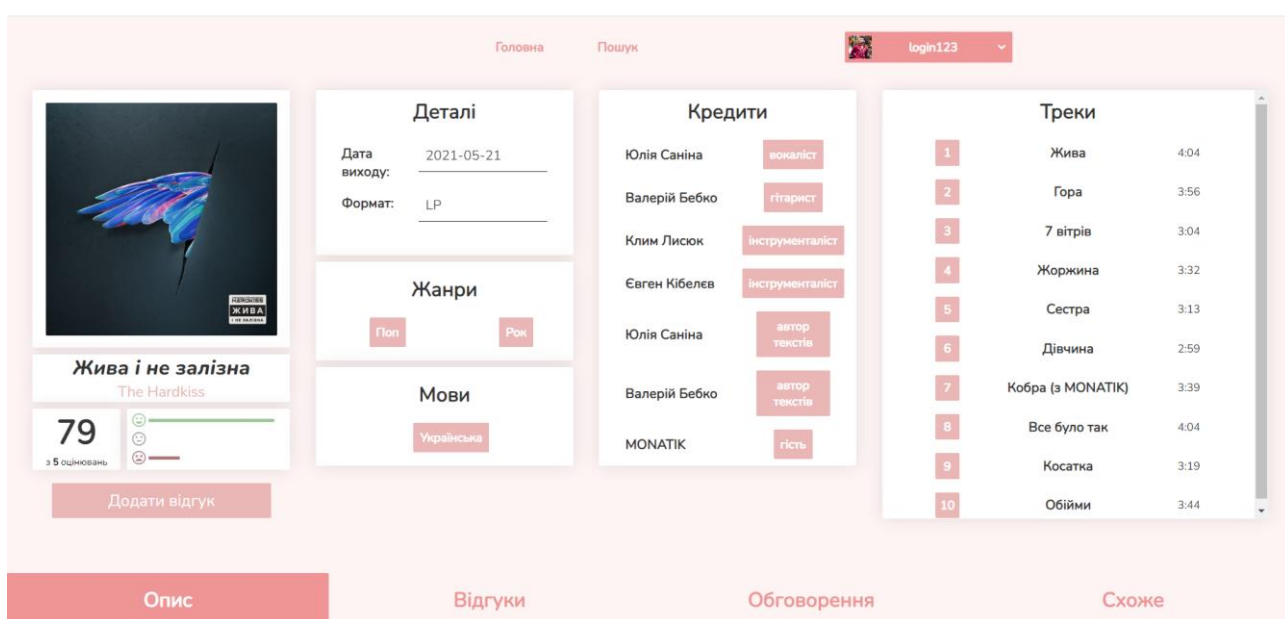


Рис. 3.5. Сторінка музичного твору

Для того щоб залишити відгук на музичний твір, користувачам потрібно заповнити кілька полів. По-перше, ввести заголовок відгуку, який коротко висвітлює його суть. По-друге, оцінити твір, вказавши свою оцінку. По-третє, описати свої враження, які ви відчули під час прослуховування. І нарешті, надати детальний текст відгуку, де можна виразити свою думку та враження від твору.

Структура та інтерфейс сторінки для написання відгуку демонструються на рисунку 3.6. Це забезпечує користувачам зручний та інтуїтивно зрозумілий спосіб поділитися своїми думками та оцінками щодо музичних творів.

Рис. 3.6. Сторінка додавання відгуку

Після успішного додавання відгуку користувачі можуть перейти на сторінку, яка відображає всі відгуки на даний музичний твір. На цій сторінці зазвичай відображаються кілька відгуків, серед яких знайдеться і той, що тільки що був доданий. Користувачі мають можливість "лайкати" відгуки, що є своєрідною формою вираження своєї підтримки чи згоди з думками інших.

Також передбачена можливість перегляду деталей кожного відгуку, що може включати текст відгуку, загальну оцінку твору та відомості про автора відгуку. Ці елементи забезпечують більш глибокий та інформативний досвід перегляду сторінки з відгуками. Загальний вигляд та інтерфейс сторінки з відгуками музичного твору можна побачити на рисунку 3.7.

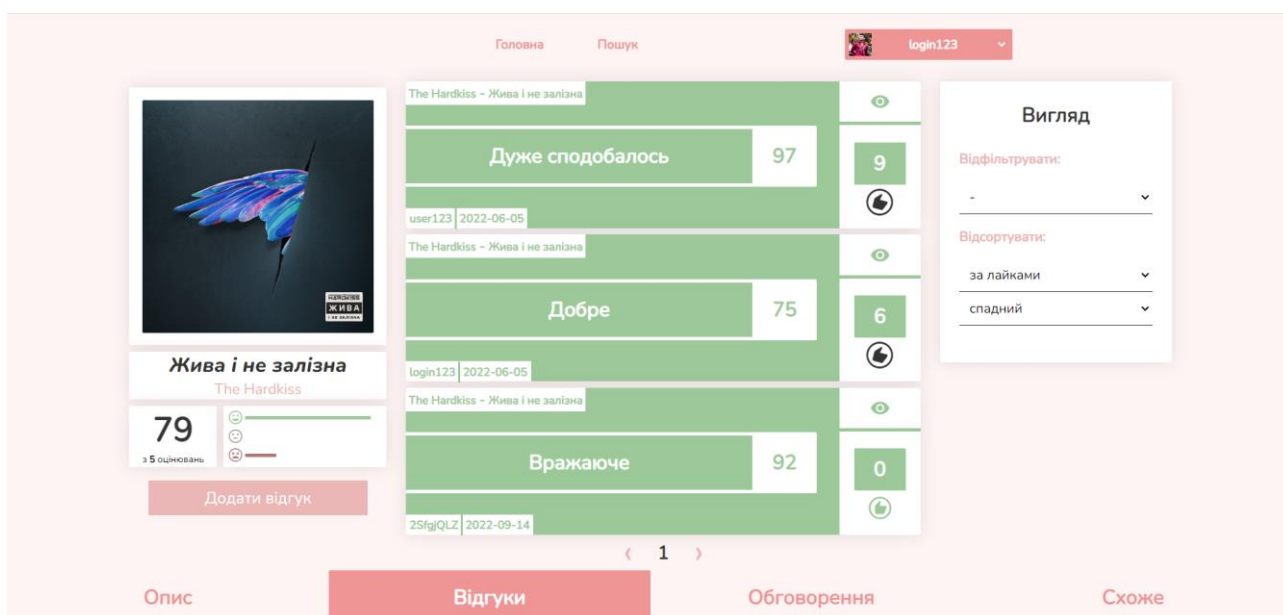


Рис. 3.7. Сторінка із відгуками музичного твору

На сторінці конкретного відгуку представлена детальна інформація, яка включає текст самого відгуку, його заголовок, а також ім'я автора відгуку. Це дає можливість іншим користувачам отримати уявлення про враження та думки людини щодо музичного твору.

Крім того, якщо користувач, який переглядає сторінку, є автором цього відгуку, він має додаткові опції для управління своїм відгуком. Зокрема, існує можливість редагувати відгук, якщо потрібно внести зміни або оновити інформацію, а також видаляти відгук, якщо це вважається необхідним. Ці функції забезпечують більш гнучке управління вмістом, який користувачі публікують на сайті.

Вигляд та розташування цих елементів на сторінці відгуку можна побачити на рисунку 3.8, що демонструє інтерфейс та доступність цих функцій.

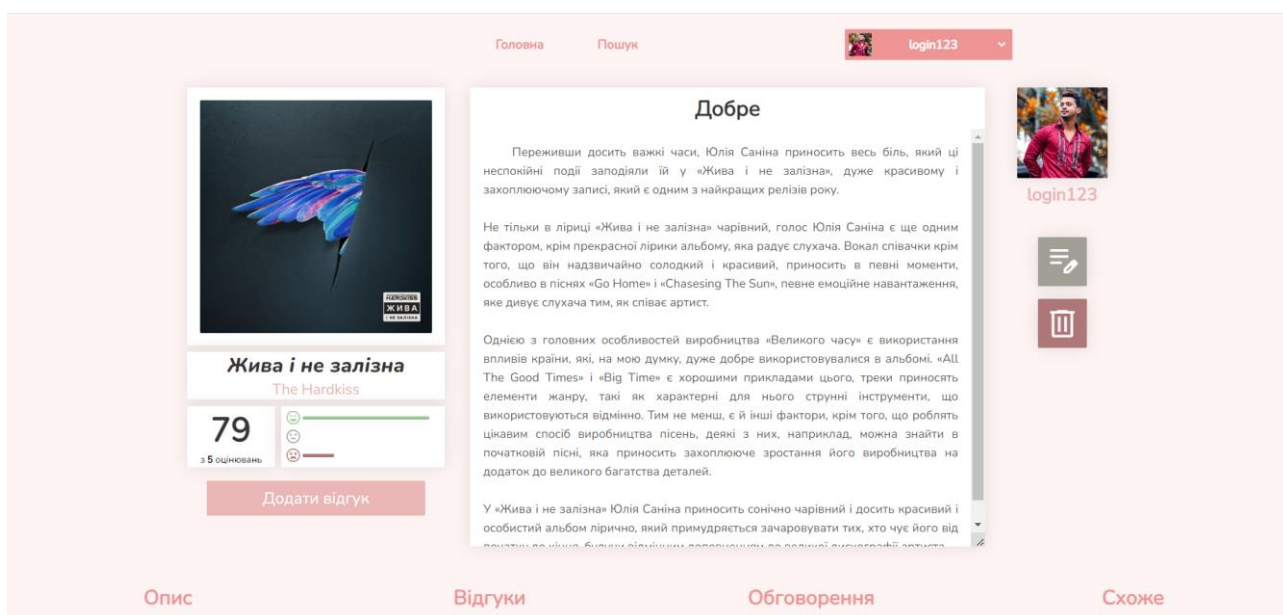


Рис. 3.8. Сторінка відгуку

Крім перегляду відгуків, користувачі мають можливість перейти на спеціальну сторінку для обговорення, де розташований груповий чат. Цей чат дозволяє користувачам вести дискусії та обмінюватися думками у вільній формі. Це може бути корисним для обговорення різних аспектів музичних творів, висловлення думок та спілкування з іншими любителями музики.

На сторінці обговорення, зазвичай, можна знайти інтерфейс групового чату, який є інтуїтивно зрозумілим та легким у використанні. Користувачі можуть бачити повідомлення від інших учасників, а також надсилати свої власні коментарі.

Для наочного представлення, як виглядає сторінка обговорення з груповим чатом, можна звернутися до рисунку 3.9. Цей рисунок ілюструє інтерфейс сторінки та як вона виглядає для користувачів, які беруть участь у обговоренні.

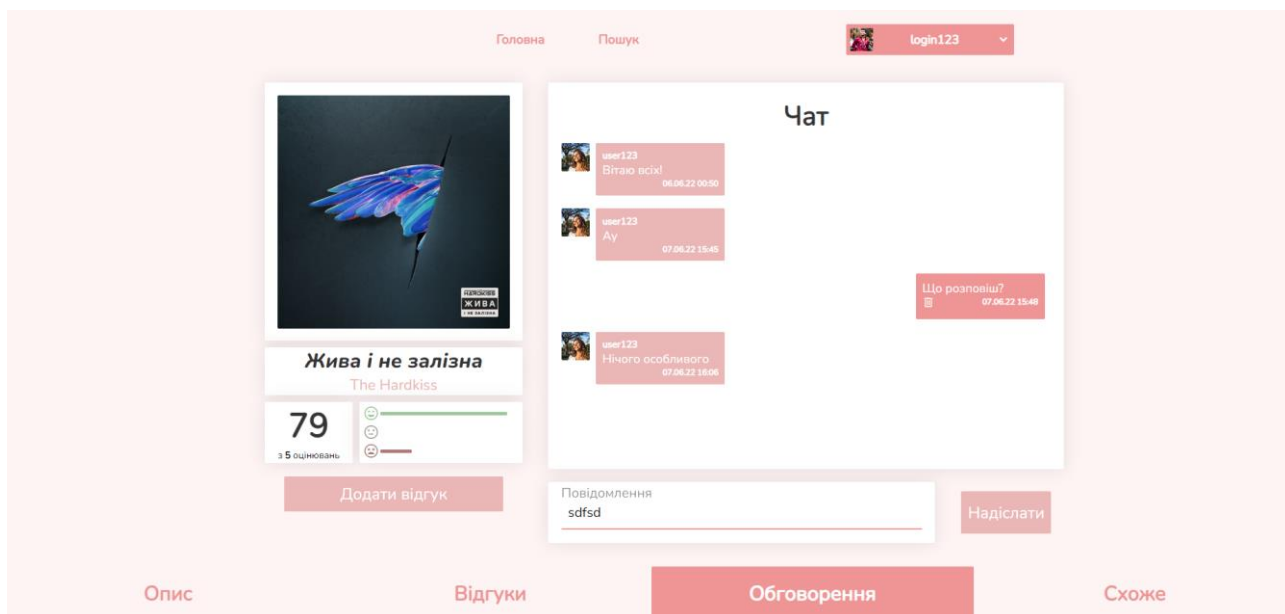


Рис. 3.9. Сторінка обговорення

Перейшовши на сторінку «Схоже», можемо побачити рекомендації схожих музичних проєктів. Це можна побачити на рисунку 3.10

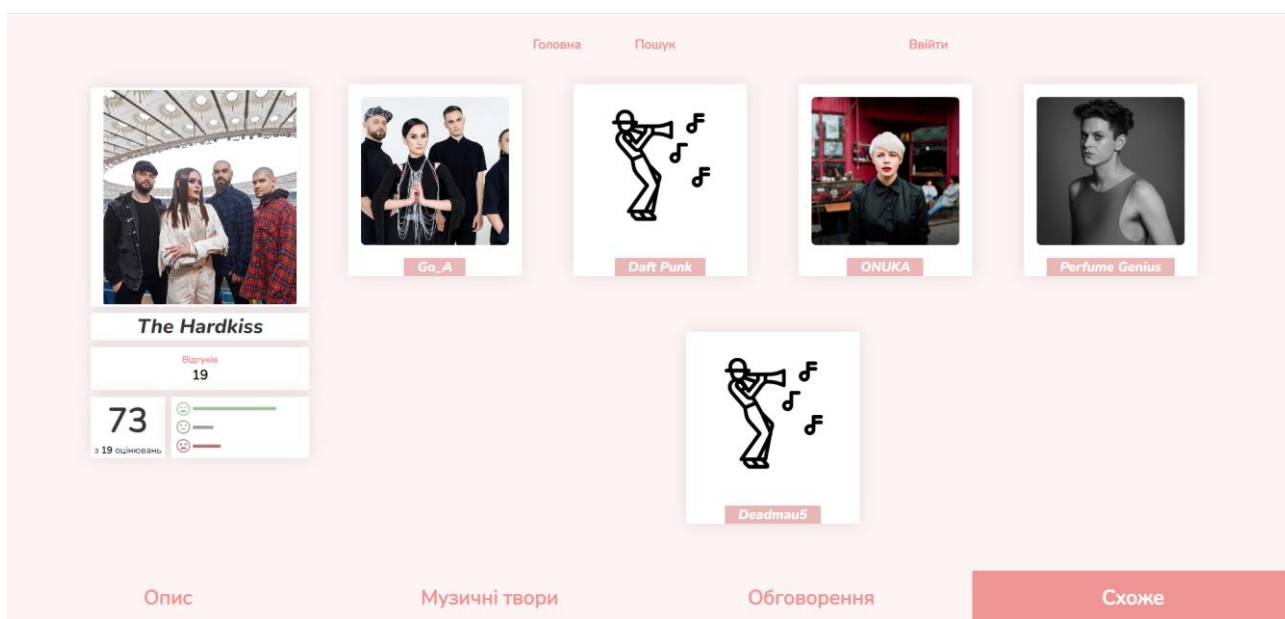


Рис. 3.10. Сторінка схожих музичних проєктів

На сайті також присутні спеціальні сторінки, присвячені виконавцям. Кожна з таких сторінок містить детальну інформацію про конкретного виконавця. Ця інформація може включати біографію, дискографію, важливі досягнення, фотографії та інші відомості, які допомагають глибше зрозуміти творчість та кар'єру виконавця.

Структура сторінки з описом виконавця є інтуїтивно зрозумілою, надаючи користувачам легкий доступ до всієї необхідної інформації. Це дозволяє шанувальникам глибше зануритися в світ улюбленого артиста та дізнатися більше про його музику та життєвий шлях.

Для прикладу, як виглядає така сторінка, можна звернутися до рисунку 3.11, де представлено візуальне зображення інтерфейсу сторінки з описом виконавця.

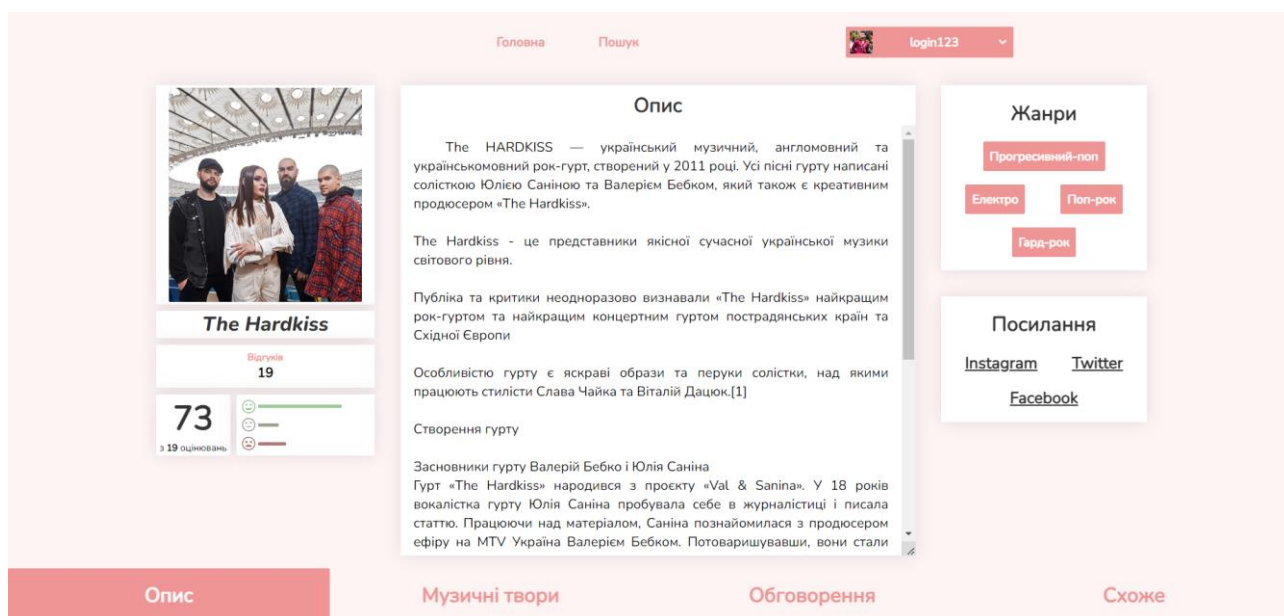


Рис. 3.11. Сторінка з описом виконавця

На сайті передбачена можливість не лише ознайомитися з інформацією про виконавця, але й переглянути список його музичних творів. Користувачі можуть перейти до конкретного твору, щоб дізнатися про нього більше, а також взяти участь у обговореннях, пов'язаних з цим твором або виконавцем.

Сторінка обговорення виконавця має схожу структуру та функціонал до сторінки обговорення музичного твору, що забезпечує послідовність та зручність користування сайтом. Користувачі можуть вільно обмінюватися думками, коментарями та враженнями, створюючи активну спільноту навколо виконавців та їхніх творів.

Щодо перегляду музичних творів виконавця, сторінка, де представлений такий список, демонструється на рисунку 3.12. На ній користувачі можуть

знайти перелік усіх творів виконавця, ознайомитися з назвами, роками випуску та іншою важливою інформацією, яка може бути цікавою для шанувальників.

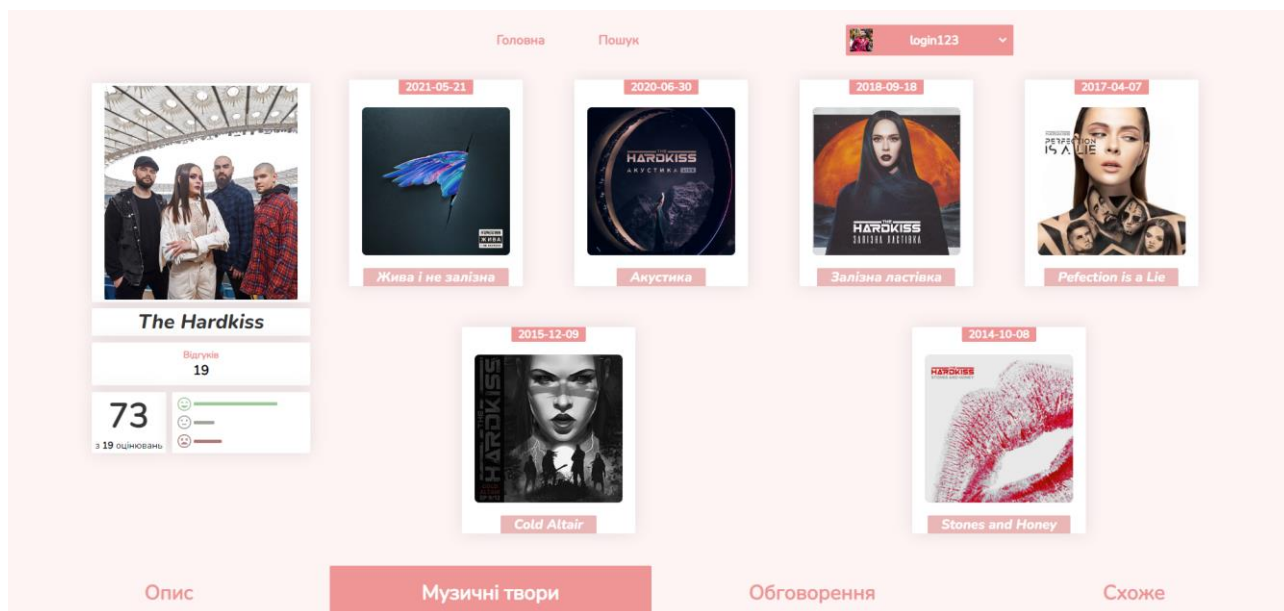


Рис. 3.12. Сторінка з музичними творами виконавця

На сайті також присутня головна сторінка, яка служить як центральний вузол для швидкого доступу до різних секцій та функцій. На головній сторінці користувачі можуть знайти різноманітну інформацію, яка включає перелік популярних або очікуваних музичних творів, а також список популярних виконавців. Це дозволяє відвідувачам сайту бути в курсі актуальних тенденцій та новинок у світі музики.

Головна сторінка, таким чином, виступає як швидкий путівник по сайту, де можна відразу ж знайти найбільш актуальну та запитувану інформацію. Завдяки цьому, користувачі мають можливість легко орієнтуватися у широкому спектрі музичних матеріалів, представлених на сайті.

Вигляд та розташування елементів на головній сторінці можна побачити на рисунку 3.13.

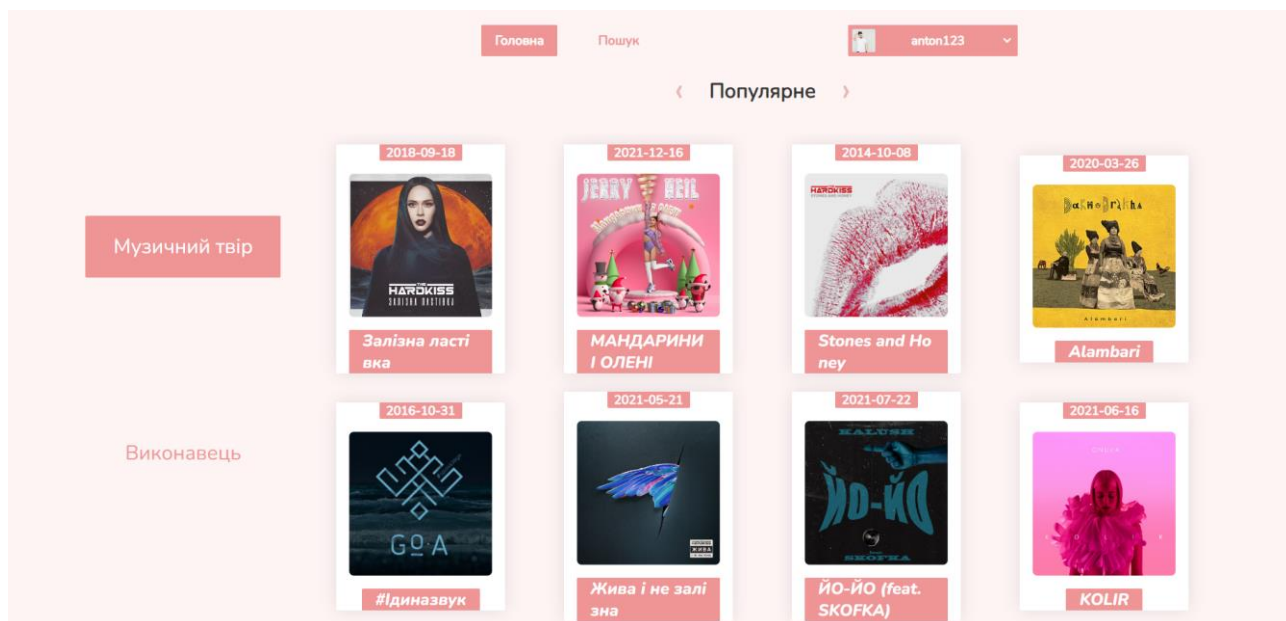
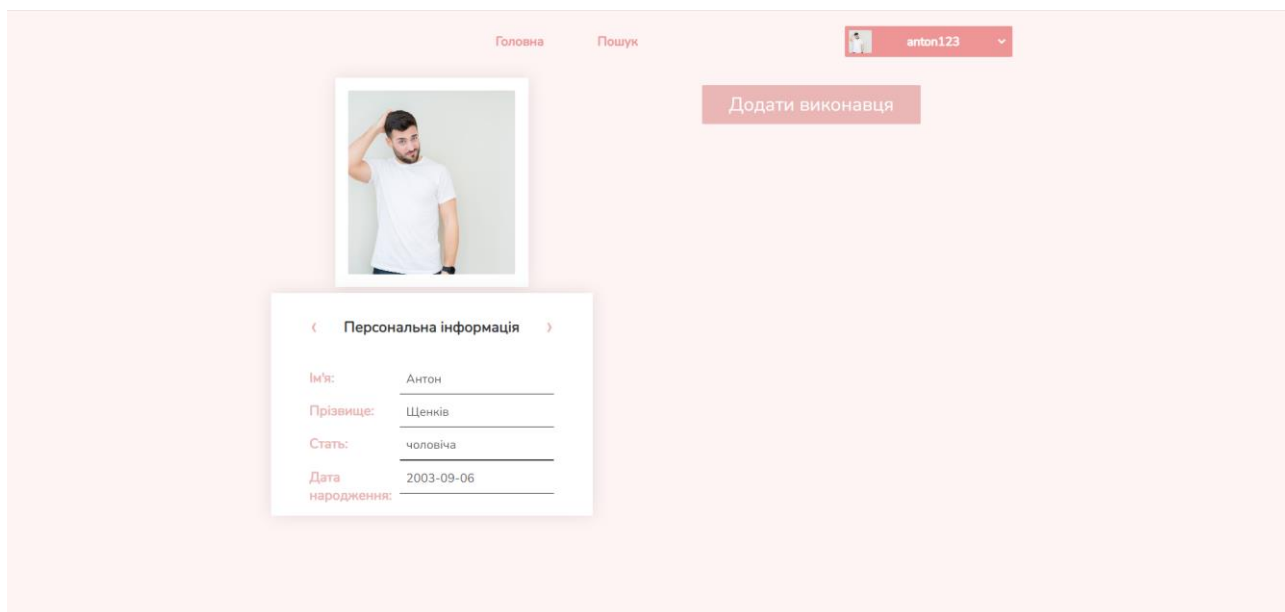


Рис. 3.13. Головна сторінка

На сайті передбачена також спеціальна роль модератора, яка наділяє користувачів додатковими повноваженнями та функціоналом. Коли користувач увійшов на сайт як модератор, інтерфейс сторінки змінюється, відображаючи інструменти та можливості, що специфічні для цієї ролі.

Однією з ключових особливостей сторінки модератора є можливість додавання нових виконавців на сайт. Це реалізується через спеціальну кнопку, яка дозволяє внести інформацію про нового артиста, забезпечуючи оновлення та розширення контенту сайту. Це важливий інструмент, що допомагає підтримувати актуальність і різноманітність музичної бази даних.

Вигляд і функціонал сторінки модератора, включно з кнопкою для додавання виконавців, можна побачити на рисунку 3.13.



Головна Пошук anton123

Додати виконавця

Персональна інформація

Ім'я: Антон

Прізвище: Щенків

Стать: чоловіча

Дата народження: 2003-09-06

Рис. 3.14. Сторінка модератора

Після натискання на кнопку "Додати виконавця" на сторінці модератора, користувач перенаправляється на спеціальну сторінку, призначену для додавання нового виконавця. На цій сторінці представлена форма з рядом полів, які необхідно заповнити для створення профілю виконавця. Ці поля можуть включати інформацію, таку як ім'я виконавця, біографію, жанри, в яких він спеціалізується, фотографії, та інші важливі відомості.

Заповнення цих полів дозволяє модераторам ефективно додавати нових виконавців до бази даних сайту, забезпечуючи регулярне оновлення та розширення контенту. Ця функція є важливою частиною управління ресурсом, оскільки вона допомагає підтримувати сайт актуальним та цікавим для користувачів.

Вигляд сторінки для додавання виконавця, включаючи всі необхідні поля, можна побачити на рисунку 3.15.

Рис. 3.15. Сторінка з додаванням виконавця

Адміністратор сайту має розширені можливості управління контентом, які включають зміну та видалення інформації про музичні твори та виконавців. На сторінці кожного музичного твору чи виконавця адміністратор може знайти відповідні кнопки для редагування або видалення. Це дозволяє адміністратору швидко та ефективно вносити необхідні зміни в базу даних сайту, забезпечуючи її актуальність та точність.

Крім того, на сторінці виконавця адміністратору доступна ще одна важлива функція – можливість додавання нових музичних творів для цього виконавця. Це розширює можливості управління контентом, дозволяючи адміністратору доповнювати профілі виконавців новими творами.

Щоб зрозуміти, як виглядають ці функції на практиці, можна звернутися до рисунку 3.16, де показано інтерфейс сторінки виконавця з точки зору адміністратора.

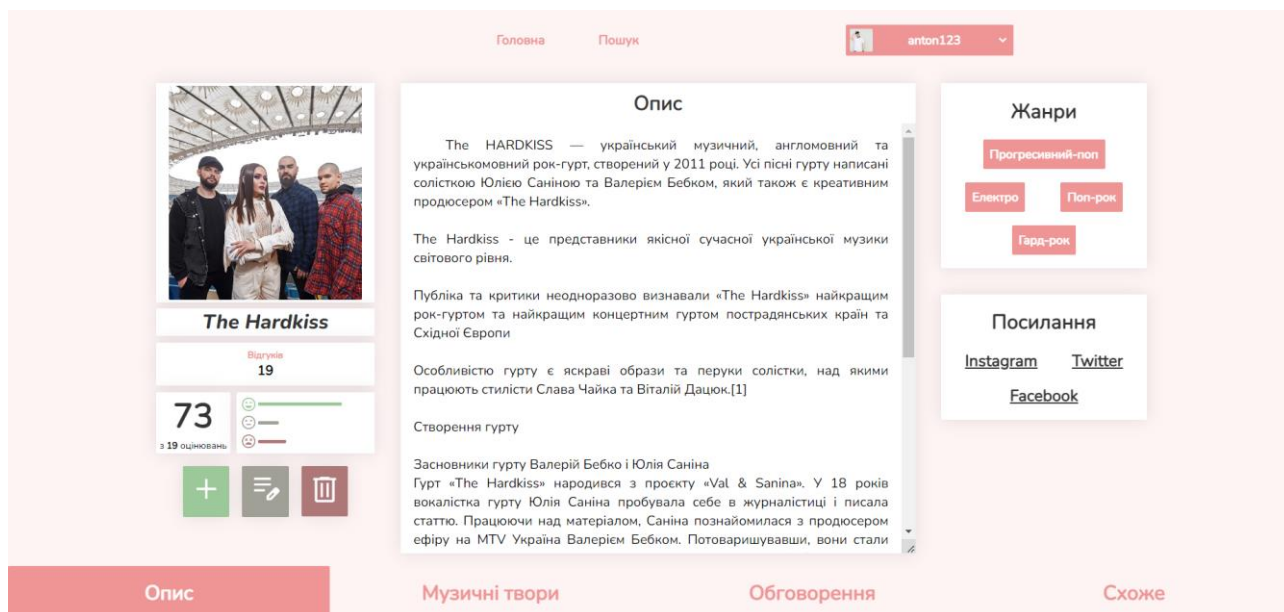


Рис. 3.16. Сторінка виконавця зі сторони модератора

Кожен зареєстрований користувач на сайті має доступ до власної сторінки налаштувань. Ця сторінка дозволяє користувачам вносити зміни до своїх персональних даних та керувати своїм профілем. Можливості, які включають налаштування, зазвичай охоплюють оновлення контактної інформації, зміну паролю, налаштування конфіденційності, а також зміну інших персональних даних, які користувач надав при реєстрації.

Окрім цього, на сторінці налаштувань також є опція для видалення облікового запису. Ця функція дозволяє користувачам повністю видалити свій профіль з сайту, якщо вони більше не бажають користуватися послугами або з якоїсь іншої причини хочуть припинити свою участь.

Для наочного представлення, як виглядає сторінка налаштувань та які функції вона включає, можна звернутися до рисунку 3.17.

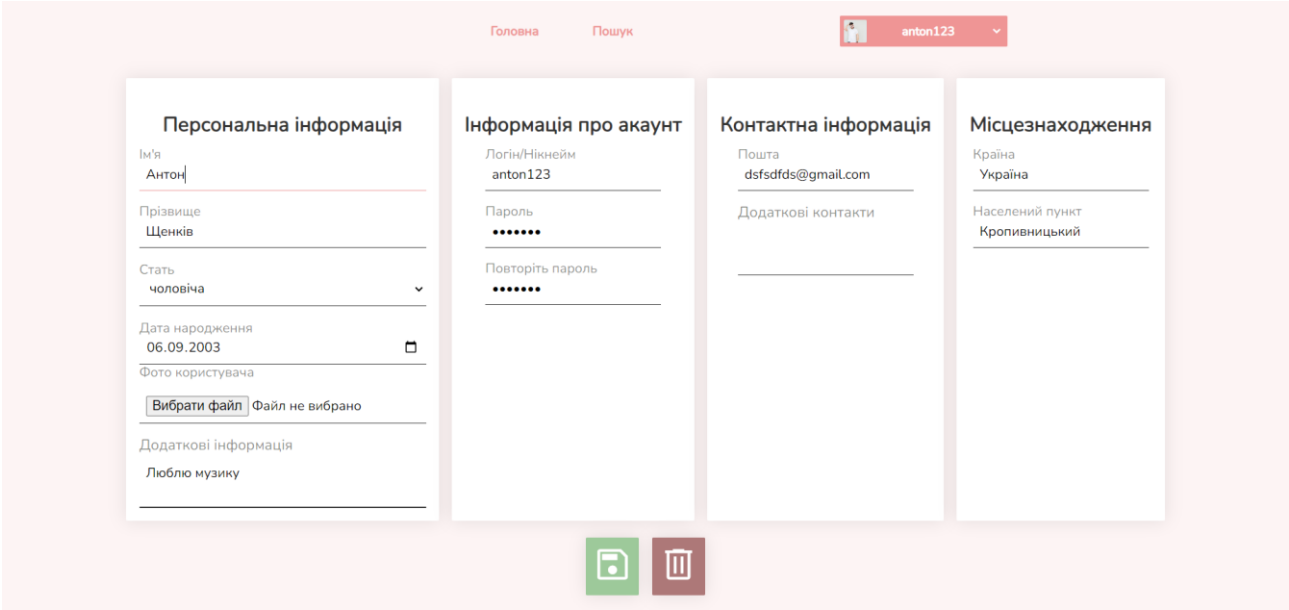


Рис. 3.17. Сторінка із налаштуваннями

3.2. Тестування музичної вебплатформи

Ініціюємо процес тестування програми. Почнемо з аналізу функціоналу форми для створення облікового запису:

Таблиця 3.1

Тестування форми створення облікового запису

Номер тесту	Очікуваний результат	Фактичний результат	Пройдено/не пройдено (P/F)
1	Всі введені дані, за винятком додаткової контактної інформації, додаткових відомостей та зображення профілю, повинні бути визнані заповненими	Введені дані, крім додаткових контактних даних, додаткової інформації та фотографії профілю, були визнані заповненими	Р
2	Система не визнає логін, якщо він вже існує в базі даних	Логін не було визнано, оскільки він вже існує	Р
3	Поля паролів мають бути однаковими для їх визнання	Введені паролі, які не співпадали, не були визнані	Р

Продовження таблиці 3.1

4	Дата народження не повинна перевищувати поточну дату	Дата народження, яка була пізніша за сьогоднішню, не була визнана	P
5	Файл для фото профілю повинен бути зображенням та не перевищувати 2 Мб	Файл, який не був зображенням або мав розмір більший за 2 Мб, не був прийнятий	P
6	Всі поля, крім поля з оцінкою, не приймають дані, якщо довжина перевищує встановлений максимум	Введені дані в полях, за винятком поля з оцінкою, які перевищували максимальну довжину, не були прийняті	P

Наступним кроком буде перевірка форми для входу в систему.

Таблиця 3.2

Тестування форми для входу в систему

<i>Номер тесту</i>	<i>Очікуваний результат</i>	<i>Фактичний результат</i>	<i>Пройдено/не пройдено (P/F)</i>
1	Незаповнені поля не будуть враховані системою	Незаповнені поля не були враховані при обробці	P
2	При відсутності логіна система повинна відображати відповідне повідомлення	При спробі використання неіснуючого логіна з'явилося повідомлення про його відсутність	P
3	У разі помилки при введенні пароля має з'явитися повідомлення про помилку	Під час введення неправильного пароля система вивела повідомлення про помилку	P

Потім перейдемо до тестування функцій форми для написання відгуків.

Таблиця 3.3

Тестування форми для написання відгуків

<i>Номер тесту</i>	<i>Очікуваний результат</i>	<i>Фактичний результат</i>	<i>Пройдено/не пройдено (P/F)</i>
1	Незаповнені поля, за винятком поля для введення оцінки, не приймаються системою	Поля без введених даних, крім поля з оцінкою, не були враховані	P
2	Поле оцінки приймає тільки значення в межах від 0 до 100	Поле для оцінки відхиляло числа, які не лежали в діапазоні від 0 до 100	P
3	Поля, крім поля з оцінкою, не приймають введення тексту, якщо його довжина перевищує заданий максимум	Введені дані в полях, крім поля для оцінки, що перевищували максимальну довжину, не були прийняті	P

Далі ми візьмемося за тестування форми, призначеної для виконавців.

Таблиця 3.4

Тестування форми для заповнення виконавців

<i>Номер тесту</i>	<i>Очікуваний результат</i>	<i>Фактичний результат</i>	<i>Пройдено/не пройдено (P/F)</i>
1	Незаповнені поля не будуть враховані системою	Незаповнені поля не були враховані при обробці	P
2	Поле з одинарним символом цитати буде прийнято системою	Поле, що містило одинарний символ цитати, було успішно прийняте	P
3	Введення в поля, крім поля з оцінкою, що перевищує визначений максимум, не буде прийнято	При введенні тексту, що перевищує дозволenu довжину, з'являлося повідомлення про неправильно введений пароль	P

Продовження таблиці 3.4

4	Файл для фото профілю повинен бути зображенням та не перевищувати 2 Мб	Файл, який не був зображенням або мав розмір більший за 2 Мб, не був прийнятий	Р
---	--	--	----------

Після цього ми зосередимося на перевірці форми для заповнення музичних композицій.

Таблиця 3.5

Тестування форми для заповнення музичних композицій

<i>Номер тесту</i>	<i>Очікуваний результат</i>	<i>Фактичний результат</i>	<i>Пройдено/не пройдено (P/F)</i>
1	Пусті поля не зчитуються	Пусті поля не зчитуються	Р
2	Поле з одинарним символом цитати зчитуються	Поле з одинарним символом цитати зчитуються	Р
3	Поле для вибору фото профіля не зчитується, якщо обрано файл, який не є картинкою або розмір файлу перевищує 2 Мб	Поле для вибору фото профіля не зчитується, якщо обрано файл, який не є картинкою або розмір файлу перевищує 2 Мб	Р

І на завершення, ми сконцентруємося на тестуванні безпеки шляхів доступу та загального функціоналу вебсайту.

Таблиця 3.6

Тестування безпеки шляхів та функціоналу вебсайту

<i>Номер тесту</i>	<i>Очікуваний результат</i>	<i>Фактичний результат</i>	<i>Пройдено/не пройдено (P/F)</i>
1	Доступ до власної сторінки модератора дозволений лише для користувачів з роллю модератора	Доступ до особистої сторінки можливий тільки для користувачів з правами модератора	Р

Продовження таблиці 3.6

2	Сторінка для додавання музичних творів доступна тільки для модераторів	Лише модератори мали змогу перейти на сторінку додавання музичних творів	Р
3	Можливість редагування музичних творів відкрита тільки для модераторів	Тільки модератори мали доступ до сторінки зміни музичних творів	Р
4	Функція видалення музичних творів обмежена правами модератора	Видалення музичних творів було можливим лише для користувачів з роллю модератора	Р
5	Сторінка додавання виконавця доступна виключно для модераторів	Доступ до сторінки додавання виконавця мав лише модератор	Р
6	Тільки модератор може перейти на сторінку редагування виконавця	Сторінка редагування виконавця була доступна тільки для модераторів	Р
7	Функція видалення виконавців обмежена правами модератора	Видалення виконавців було можливим лише для модераторів	Р
8	Пошук користувачів не включає в себе модератора	Пошук користувачів не показує в пошуку адміністратора	Р
10	Тільки зареєстровані користувачі мають право додавати відгуки	Функція додавання відгуків була доступна лише зареєстрованим користувачам	Р
11	Редагування власних відгуків дозволено тільки для користувачів	Користувачі могли змінювати лише свої власні відгуки	Р
12	Видалення відгуків можливе тільки для їх авторів	Користувачі мали можливість видаляти лише свої відгуки	Р
13	Редагування особистих даних доступне лише для користувачів	Редагування особистих даних доступне лише для користувачів	Р

Продовження таблиці 3.6

14	Тільки користувач може видалити свою сторінку профілю	Функція видалення профілю була доступна тільки для її власника	Р
15	Написання повідомлень дозволено тільки зареєстрованим користувачам	Тільки зареєстровані користувачі могли надсилати повідомлення	Р

Висновки до третього розділу

У цьому розділі детально описуються основні функціональні можливості програми, які спрямовані на забезпечення зручного та інтерактивного користувацького досвіду. Починаючи з процесу реєстрації, користувачі мають можливість створити акаунт через докладний, але зрозумілий етапний процес. Після входу в систему, користувачі потрапляють на свою персональну сторінку, де вони можуть налаштовувати та оновлювати свою персональну інформацію, що дозволяє їм краще висловлювати свою індивідуальність у спільноті.

Однією з унікальних функцій є можливість користувачів додавати власні відгуки, редагувати їх або видаляти, що надає їм активну роль у формуванні спільноти. Участь у дискусіях на платформі також сприяє створенню жвавого спілкування між користувачами.

Функція перегляду схожих музичних проєктів та виконавців є особливо цікавою, оскільки вона дозволяє користувачам відкривати нові музичні жанри та артистів, які відповідають їхнім музичним смакам та перевагам. Ця функція використовує алгоритми рекомендацій, щоб надати користувачам персоналізований досвід.

На головній сторінці, музичні твори та виконавці впорядковані за категоріями, що полегшує навігацію та дозволяє користувачам легко знайти музику, що їх цікавить.

Крім того, в вебдодатку присутня роль модератора. Особи, які займають цю посаду, відповідають за управління та контроль контенту на сайті. Вони

забезпечують дотримання правил та стандартів, встановлених на платформі, тим самим підтримуючи високу якість та релевантність матеріалів, доступних для користувачів.

В рамках тестування вебпроєкту були випробувані наступні елементи: форма входу в систему, процедура реєстрації користувачів, система відправлення відгуків, інструментарій для додавання музичних творів, форма для заповнення виконавців і перевірка на безпеку шляхів доступу та загальної функціональності вебдодатку. В результаті, вебплатформа для надання музичних відгуків демонструє стабільну роботу і виконує всі поставлені завдання.

ВИСНОВКИ

За час роботи над випускною магістерською роботою було розроблено та детально досліджено вебплатформу для надання музичних відгуків та рекомендацій. Проєкт охоплює три ключові аспекти – аналіз та розробку вебплатформи, створення бази даних та архітектури вебдодатку, а також реалізацію користувацького інтерфейсу та функціональностей.

У першому розділі проєкту було здійснено глибокий аналіз існуючих вебсистем, таких як Metacritic і RateYourMusic, для ідентифікації ключових функцій, необхідних для нашого сайту. В результаті було виявлено потребу в реалізації функцій оцінювання музичних творів, використання кольорових позначок для оцінок, а також створення інтуїтивно зрозумілого дизайну. Разом з тим, особлива увага була приділена розробці ефективної музичної рекомендаційної системи, для чого були використані аналітичні дані з популярних платформ, таких як Spotify та YouTube Music. Використання методу колаборативної фільтрації та сучасних технік машинного навчання дозволило створити систему, яка здатна адаптуватися до музичних переваг кожного користувача.

Другий розділ зосереджувався на розробці бази даних і архітектури вебдодатку. Було створено структуру бази даних, що включає таблиці для зберігання інформації про користувачів, відгуки, музичні твори, виконавців та інші важливі елементи. Використання діаграм компонентів і розгортання сприяло чіткому визначенню фронтенду і бекенду вебплатформи. Реалізація музичної рекомендаційної системи з використанням матричної факторизації та нейронних мереж дозволила виявити складні взаємозв'язки між користувачами і музичними творами та надавати персоналізовані рекомендації.

Третій розділ роботи був присвячений реалізації функціональностей програми, спрямованих на забезпечення оптимального користувацького досвіду. Було розроблено інтуїтивний інтерфейс, що дозволяє користувачам легко здійснювати навігацію по сайту, реєструватися, вносити зміни в особисту

інформацію та брати активну участь в обговоренні через додавання відгуків та участь у дискусіях. Особлива увага була приділена функції перегляду схожих музичних проєктів та виконавців, що забезпечує користувачам різноманітність вибору та нові музичні відкриття. Додатково, важливою частиною було тестування функціональності вебплатформи. Це включало перевірку правильної роботи всіх ключових компонентів: форм входу та реєстрації, форми відгуків, функцій перегляду та пошуку музичних творів і виконавців. Було надано значимості забезпеченню безпеки даних та надійності шляхів доступу до функціоналу вебдодатку, щоб забезпечити стабільну та безпечну роботу платформи для всіх користувачів.

Отже, реалізована вебплатформа є вагомим внеском у сферу музичних сервісів, демонструючи як інноваційні технології машинного навчання та аналізу даних можуть бути використані для створення глибоко персоналізованого та зручного користувацького досвіду. Система показує великий потенціал для подальшого розвитку та адаптації, відкриваючи нові можливості для музичних рекомендацій та цифрових взаємодій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аналіз існуючих алгоритмів музичних рекомендаційних систем / Гордєєв Р.С., Граф М.С. // Стаття з наукового журналу «Технічна інженерія» .– 19 грудня 2022. – № 2 (90) 2022. – С. 86-93.
2. Розробка соціальної мережі для меломанів / Гордєєв Р.С., Вакалюк Т.А. // Тези доповідей III Всеукраїнської науково-практичної інтернет-конференції "СУЧАСНІ КОМП'ЮТЕРНІ ТА ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ". – 12-19 грудня 2022. – С. 346-348.
3. IFPI GLOBAL MUSIC REPORT 2022 [Електронний ресурс]. – Режим доступу: <https://globalmusicreport.ifpi.org/>.
4. A systematic review and research perspective on recommender systems / Діпджоті Рой, Мала Дата // Journal Of Big Data. – № 59 (2022). – 3 травня 2022
5. The million dollar programming prize / Р.М. Белл, Дж. Беннетт, Й. Корен та К. Волінський // IEEE Spectrum. – №46. – 2 травня 2009. – С. 28-33
6. 'Reinforcing the Blockbuster Nature of Media': The Impact of Online Recommenders / Ден Фледер, Картік Хосанагар // Knowledge at Wharton Staff. – №74(2). – 3 жовтня 2007. – С. 133-148
7. Recommender Systems. In: Sammut, C., Webb, G.I. / Прем Мелвіль, Вікас Сіндгвані // Encyclopedia of Machine Learning and Data Mining. – Springer, Boston, MA. – №2. – 14 квітня 2017. – С. 1056-1066
8. Recommender systems. Commun ACM / Пол Резнік, Хал Р. Варіан // Communications of the ACM. – №40. – березень 1997. – С. 56-58
9. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction / Робін Бурке // Springer Nature. – №12. – листопад 2002. – С. 331-370
10. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions / Г. Адомавічіус, А. Тужилін // IEEE Transactions on Knowledge and Data Engineering. – №17(6). – липень 2005. – С. 734-749

11. Survey of Collaborative Filtering Techniques / Шяююан Су, Тагі М.. Хошгофтаар // Advances in Artificial Intelligence. – vol. 2009. – січень 2009
12. Artificial Neural Network / Стівен Вальчак, Нарцисо Черпа // Academic Press. – №3. – грудень 2003. – С. 631-645
13. Neural Networks and Deep Learning: A Textbook / Charu C. Aggarwal // Springer Cham. – №2. – 13 вересня 2018.
14. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow / Aurélien Géron // O'Reilly. – №2. – вересень 2019.
15. Deep Learning with TensorFlow and Keras / Amita Kapoor, Antonio Gulli, Sujit Pal, Francois Chollet // Packt Publishing. – №3. – 6 жовтня 2022.
16. Convolutional Neural Networks in Visual Computing / Ragav Venkatesan, Baoxin Li // CRC Press. – №3. – 8 вересня 2017.
17. Hands-On Convolutional Neural Networks with TensorFlow / Iffat Zafar, Giounona Tzanidou, Richard Burton, Nimesh Patel // Packt Publishing. – №1. – 28 серпня 2018.
18. Convolutional neural network / Gerardus Blokdyk // 5STARCooks. – №2. – 24 лютого 2022.
19. Neural Networks with R / Giuseppe Ciaburro, Balaji Venkateswaran // Packt Publishing. – №1. – 27 вересня 2017.
20. Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability / Danilo Mandic, Jonathon Chambers // Wiley. – №1. – 7 серпня 2001.
21. Recurrent Neural Network / Kassahun Tesfaye // LAP LAMBERT Academic Publishing. – №1. – 30 червня 2023.

ДОДАТКИ


```

        .count()
        .reset_index()
        .rename(columns={'mark': 'totalRatingCount'})
    )
    rating_with_totalRatingCount = self.combine_rating.merge(
        artist_ratingCount, left_on = 'artist_id', right_on = 'artist_id', how =
'left'
    )
    self.user_rating =
rating_with_totalRatingCount.drop_duplicates(['user_login', 'artist_id'])

    def __create_album_matrix(self):
        self.album_user_rating_pivot = self.user_rating.pivot(index = 'user_login',
columns = 'album_id', values = 'mark').fillna(0)
        self.X = self.album_user_rating_pivot.values.T

    def __create_artist_matrix(self):
        self.artist_user_rating_pivot = self.user_rating.pivot(index = 'user_login',
columns = 'artist_id', values = 'mark').fillna(0)
        self.X = self.artist_user_rating_pivot.values.T

    def __fit_model(self):
        SVD = TruncatedSVD(n_components=12, random_state=17)
        matrix = SVD.fit_transform(self.X)
        self.corr = np.corrcoef(matrix)

    def __check_album_results(self, album_id: str) -> list:
        album_ids = self.album_user_rating_pivot.columns
        album_id_list = list(map(str, album_ids))
        album_index = album_id_list.index(album_id)
        corr_album_index = self.corr[album_index]
        return list(map(str, album_ids[corr_album_index >= self.result_precision]))

    def __check_artist_results(self, artist_id: str) -> list:
        artist_ids = self.artist_user_rating_pivot.columns
        artist_id_list = list(map(str, artist_ids))
        artist_index = artist_id_list.index(artist_id)
        corr_artist_index = self.corr[artist_index]
        return list(map(str, artist_ids[corr_artist_index >= self.result_precision]))

    def __create_neural_network_model(self, type):
        embedding_size = 50

        if type == 'album':
            item_id = self.combine_rating.album_id
            self.item_ids =
self.combine_rating['album_id'].astype("category").cat.codes.values
        else:
            item_id = self.combine_rating.artist_id

```

```

        self.item_ids =
self.combine_rating['artist_id'].astype("category").cat.codes.values

        user_input = Input(shape=[1], name="User-Input")
        user_embedding =
Embedding(input_dim=self.combine_rating_df.user_login.nunique(),
          output_dim=embedding_size,
          name="User-Embedding")(user_input)
        user_vec = Flatten(name="Flatten-Users")(user_embedding)
        item_input = Input(shape=[1], name="Item-Input")
        item_embedding = Embedding(input_dim=item_id.nunique(),
          output_dim=embedding_size,
          name="Item-Embedding")(item_input)
        item_vec = Flatten(name="Flatten-Items")(item_embedding)

        conc = Concatenate()([user_vec, item_vec])

        fc1 = Dense(256, activation='relu')(conc)
        drop1 = Dropout(0.3)(fc1)
        fc2 = Dense(128, activation='relu')(drop1)
        drop2 = Dropout(0.3)(fc2)
        fc3 = Dense(64, activation='relu')(drop2)
        out = Dense(1)(fc3)

        self.nn_model = Model([user_input, item_input], out)
        self.nn_model.compile('adam', 'mean_squared_error')

    def __train_neural_network(self, epochs=5):
        self.nn_model.fit([self.user_ids, self.item_ids], self.ratings, epochs=epochs)

    def get_nn_recommendations(self, item_id: str, type: str, top_n: int = 10):
        self.__create_neural_network_model(type)
        self.__train_neural_network()
        item_index = self.combine_rating[self.combine_rating[type + '_id'] ==
item_id][type + '_id'].astype("category").cat.codes.values[0]
        item_array = np.array([item_index] * len(self.user_ids))
        predictions = self.nn_model.predict([item_array, self.item_ids])

        top_items = np.argsort(predictions[:,0])[::-1][:top_n]
        recommended_item_ids = self.combine_rating.iloc[top_items][type + '_id'].values
        return recommended_item_ids.tolist()

    def get_recommendations_by_album(self, album_id: str) -> list:
        self.__total_reviews_count_by_album()
        self.__create_album_matrix()
        self.__fit_model()

        svd_recommendations = set(self.__check_album_results(album_id))
        nn_recommendations = set(self.get_nn_recommendations(album_id, 'album'))
        recommendations = svd_recommendations.intersection(nn_recommendations)

```

```
    return recommendations

def get_recommendations_by_artist(self, artist_id: str) -> list:
    self.__total_reviews_count_by_artist()
    self.__create_artist_matrix()
    self.__fit_model()

    svd_recommendations = set(self.__check_artist_results(artist_id))
    nn_recommendations = set(self.get_nn_recommendations(artist_id, 'artist'))
    recommendations = svd_recommendations.intersection(nn_recommendations)

    return recommendations
```

Клас для роботи із базою даних:

```
from dotenv import load_dotenv
from sqlalchemy import create_engine

load_dotenv()

class Database:
    def __init__(self, user: str, password: str, host: str, db_name: str, port: int = 5432):
        db_string = f"postgresql+psycopg2://{user}:{password}@{host}:{port}/{db_name}"
        self.engine = create_engine(db_string)

    def __del__(self):
        self.engine.dispose()
```

Сервер, що обробляє запити на отримання рекомендацій:

```
from flask import Flask
from flask_cors import CORS, cross_origin

from mrs_library.mrs_class import MusicRecommenderSystem
from data.database import Database
import os
import warnings
from dotenv import load_dotenv

warnings.filterwarnings("ignore",category = RuntimeWarning)
load_dotenv()

db = Database(os.getenv('DB_USER'), os.getenv('DB_PASSWORD'), os.getenv('DB_HOST'),
os.getenv('DB_NAME'))
mrs_ms_model = MusicRecommenderSystem(db.engine)

app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'

## 6d693116-fca8-4973-b549-e87d785ac418
@app.route('/album-recommendations/<album>')
@cross_origin()
def get_album_recommendations(album):
    return mrs_ms_model.get_recommendations_by_album(album)

## 74fc5995-ed26-4fd2-8e31-9824b40f95ff
@app.route('/artist-recommendations/<artist>')
@cross_origin()
def get_artist_recommendations(artist):
    return mrs_ms_model.get_recommendations_by_artist(artist)

if __name__ == '__main__':
    app.run(debug=True)
```