МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

Нейромережева система рекомендації музичних творів

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН-41

Кучерук Владислав Денисович

(прізвище та ініціали)

Керівник Федусенко О.В.

(прізвище та ініціали)

кандидат технічних наук, доцент

(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № від р.
зав. кафедри_____ доц. Іларіонов О.Є.Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
завідувача кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2023 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кучеруку Владиславу Денисовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Нейромережева система рекомендації музичних творів, затверджена протоколом засідання кафедри від « » 2023 р. №3

2. Термін здачі студентом закінченого проекту (роботи): 31 травня 2023 року

3. Вихідні дані до проекту (роботи)

Дані про пісні; Історія прослуховувань користувачів; дані про користувачів

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1. Аналітичний огляд літератури з побудови рекомендаційних систем. Аналіз існуючих систем, що використовують рекомендаційні системи. Постановка задачі.

2. Проектні рішення нейромережевої рекомендаційної система на основі алгоритму колаборативного фільтрування. Аналіз основних функцій та проектування архітектури системи.

3. Реалізація та тестування розробленої системи.

4. Висновки.

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій):

1. Аналітичний огляд: актуальність поставленої задачі, проблематика та поточні дослідження.

2. Постановка задачі: функціональне моделювання та розробка вимог до системи.

3. Проектні рішення: колаборативне фільтрування, основні функції та архітектура системи.

4. Програмна реалізація та огляд процесу тестування: інформаційне забезпечення, структурна схема підсистем, результати тестування.

5. Висновки

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання __ _____ 2023 року

Керівник _____ _____ / /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Вивчення навчальної, довідкової, спеціальної літератури, стану питання.	20.02-02.02	
2.	Підготовка матеріалів першого розділу. Аналіз існуючих рішень, формування постановки задачі та концепції роботи.	10.02-05.03	
3.	Підготовка матеріалів другого розділу. Аналіз методів моніторингу полів.	06.03-15.03	
4.	Підготовка матеріалів другого розділу. Аналіз основних функцій та розробка архітектури системи.	16.03-10.04	
5.	Підготовка матеріалів третього розділу. Програмна реалізація модулів системи та їх тестування.	11.04-03.05	
6.	Аналіз отриманих результатів. Написання висновків.	04.05-10.05	
7.	Завершення роботи, її оформлення та подання на рецензію.	11.05-15.05	
8.	Створення мультимедійної презентації захисту.	15.05-19.05	

Студент-дипломник _____ / /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ _____ / /
(підпис) (ПІБ)

АНОТАЦІЯ

Кучерук Владислав Денисович виконав випускну кваліфікаційну роботу на тему «Нейромережева система рекомендації музичних творів» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналітичний огляд літератури з побудови рекомендаційних систем, розглянуто архітектуру системи, розроблено математичне, інформаційне та програмне забезпечення, яке забезпечує створення персоніфікованих списків рекомендацій з піснями для користувачів.

Ключові слова: нейромережа, навчання, колаборативна фільтрація, рекомендація, прогнозування.

ANNOTATION

The degree project: «Neural Network-based Music Recommendation System» was completed by Kucheruk Vladyslav specialty 122 – «Computer Sciences».

The graduation qualification work includes an analytical literature review on the construction of recommendation systems, an examination of the system architecture, and the development of mathematical, informational, and software solutions that enable the creation of personalized recommendation lists for users, specifically focusing on music recommendations.

Keywords: neural network, learning, collaborative filtering, recommendation, prediction.

ЗМІСТ

ВСТУП

В сучасному світі, де щодня в мережу вивантажується безліч фільмів, телешоу, книг, музичних творів, рецензій і багато іншого, звичайній людині стає все складніше знайти потрібний контент. Час, витрачений на безперервний пошук, стає великою проблемою. Люди стають більш прискіпливими у виборі того, що споживають, і відсікають сервіси, які не задовольняють їхні потреби.

Разом із стрімким розвитком алгоритмів машинного навчання з'явилися платформи, що пропонують користувачам контент різного виду. Такі компанії, як Amazon, Netflix, Spotify, Meta, Google та інші, активно працюють над розробкою та вдосконаленням рекомендаційних систем. Кожна з цих систем має свої переваги та недоліки, і пошук найкращих алгоритмів для створення рекомендаційних систем залишається актуальною проблемою. В інтернеті часто проводяться конкурси від IT-компаній, в яких розробники з усього світу змагаються у створенні найбільш ефективної та точної нейронної мережі чи алгоритму, що надає рекомендації на основі запропонованих наборів даних.

Метою дослідження є аналіз, розробка та застосування алгоритму прогнозування рекомендацій для користувача на основі нейронних мереж. Для вирішення поставленої задачі необхідно визначити ефективний метод, що використовується при побудові рекомендаційних систем та запрограмувати за допомогою доступних інструментів.

Об’єктом дослідження є вивчення колаборативних підходів при реалізації рекомендаційних систем.

Предметом дослідження є нейромережі для прогнозування рекомендацій з алгоритмом колаборативного фільтрування.

Завданням даної роботи є розробка веб-додатку, який використовує нейронну мережу для забезпечення рекомендацій засобом колаборативного фільтрування.

В ході даної роботи буде розглянуто основні види рекомендаційних систем та задачі, що перед ними ставляться. Будуть проаналізовані переваги та недоліки найбільш поширених систем. Також буде розроблений алгоритм колаборативного фільтрування на основі нейронної мережі. Для цього буде створена серверна та клієнтська частини, які будуть взаємодіяти між собою, щоб забезпечити ефективний потік даних для найбільш точного прогнозування рекомендацій за допомогою нейромережного модулю. На основі цього буде створено повноцінний програмний застосунок з базою даних та інтерфейсом користувача, який буде дозволяти зручно управляти рекомендаційною системою.

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ НА ОСНОВІ НЕЙРОМЕРЕЖ

1.1. Аналітичний огляд літератури з побудови музичних рекомендаційних систем

Рекомендаційні системи - системи фільтрації контенту, які базуються на алгоритмах (машинного навчання або інших методах), що ставлять за основну мету формування списку рекомендації відповідно до вподобань. Тобто, коли користувач шукає щось, чи то в мобільному застосунку, чи в будь-якій пошуковій системі, рекомендаційна система проводить аналіз запиту та проводить відбір даних, що потенційно можуть його цікавити, при цьому надається перевага саме тому контенту, що більше всього підходить даному користувачу (часто враховуючи попередній досвід - історію прослуховувань тощо). Як правило, використовуються певні класи алгоритмів, що допомагають робити це з високою точністю.

Всього виділяють два основні види рекомендаційних систем[2]:

- колаборативне фільтрування (collaborative filtering) – виконується пошук користувачів, що мають схожі уподобання та здійснюється аналіз контенту, який міститься в їх історії прослуховувань[11]. За базове правило беруть тенденцію, що користувачів можна умовно розділити на деяку кількість кластерів та поширювати контент, що може сподобатися користувачам в межах одного кластеру. Наприклад, є ті, кому подобається рок-музика, таких користувачів можна виділити в кластер, але серед них теж можна знайти деякі групи за улюбленим виконавцем і т. п. Такий розподіл призводить до того, що новий користувач, під час формування списку рекомендацій, отримує в результаті композиції, що сподобалися іншим користувачам з кластеру, до якого його було віднесено (рис. 1.1). Слід зауважити, що в кінцевий список рекомендацій будуть потрапляти

композиції, що новий користувач раніше не слухав, але інші, в межах його кластеру так[4].

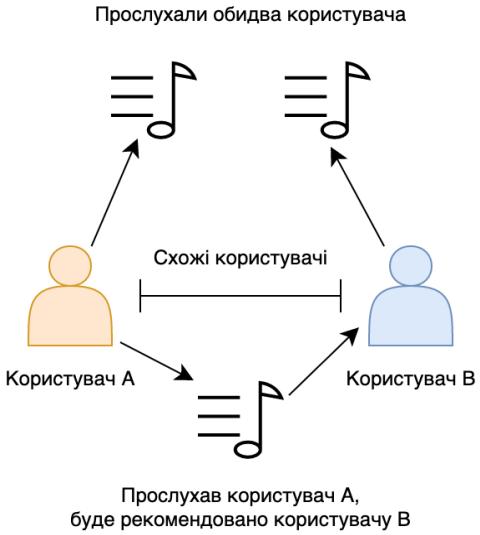

Рисунок 1.1 – Алгоритм колаборативного фільтрування

- на основі вмісту (content based) – рекомендації відбираються у відповідності до атрибутів (вимірюваних), якими характеризується композиції, попередньо високо оцінені користувачем (чи які просто входять до історії прослуховувань тощо)[3]. Наприклад, в історію прослуховувань було збережено 5 композицій, що мають певний жанр, схожих виконавців і т. п., тоді список рекомендацій буде сформовано таким чином: система аналізує характеристики (жанр, кількість прослуховувань, тривалість, ім’я виконавця тощо) та виконує пошук композицій в БД, що мають схожі параметри, використовуючи деякий алгоритм (brute force чи будь який інший) (рис. 1.2).

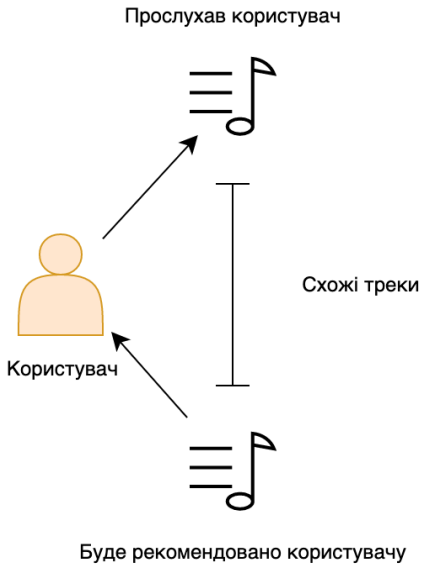

Рисунок 1.2 – Алгоритм фільтрування на основі вмісту

Історично склалося, що алгоритми пошуку рекомендацій використовували обчислювальні методи. Одним з перших та найбільш відомих є K найближчих сусідів, що базується на пошуку схожих користувачів та аналізі їх оцінок.

Весь алгоритм роботи можна узагальнити до декількох кроків:

Крок 1 - Завантажити датасет

Крок 2 - Обрати ціле значення змінної K, для позначення кількості найближчих сусідів

Крок 3 - Для кожної точки даних для тестування повторювати:

Крок 3.1 - Розрахувати відстані тестових даних до тренувальних за допомогою одного з доступних методів: Евклідова, манхетенська (лінійна) відстань, Хеммінга і т. д.

Крок 3.2 - Відсортувати обраховані відстані в порядку зростання

Крок 3.3 - Обрати K перших рядків з найменшими значеннями

Крок 3.4 - Елемент, для якого буде здійснюватися прогнозування буде віднесено до того класу, який найбільше зустрічається серед його сусідів

Крок 4 − Кінець роботи

Знаходження відстаней реалізується багатьма різними способами, нижче представлені найбільш поширені:

- Евклідова відстань:

$$\square_{\square\square} = \sqrt{\sum_{\square=1}^{\square} (\square_{\square\square} - \square_{\square\square})^2}$$

(1.1)

- Манхеттенська відстань:

$$\square_1(\square, \square) = \|\square - \square\|_1 = \sum_{\square=1}^{\square} |\square_{\square} - \square_{\square}|,$$

(1.2)

де $\square = (\square_1, \square_2, \dots, \square_{\square})$ *i* $\square = (\square_1, \square_2, \dots, \square_{\square})$ - вектори

- Відстань Хеммінга – число позицій, у яких цифри, представлені в бінарному форматі однакової довжини, різні.

Іншим розповсюдженим типом рекомендаційних систем є системи, що використовують інтелектуальні моделі. Їх видів існує дуже багато, вони мають як недоліки, так і переваги. Частіше зустрічаються Байесівські мережі (наївний класифікатор), латентні моделі, розподіл Діріхле, марківські процеси і т. д.

В загальному, алгоритм оцінює рейтинги для записів, що не знаходяться в Y, які використовується для ранжування елементів.

$$\widehat{\square}_{\square\square} = \square(\square, \square|\square),$$

(1.3)

де y(u,i) – прогнозована оцінка взаємодії користувача та елемента;

θ – параметри моделі;

f() – перетворює параметри моделі в прогнозовану кількість очок.

- Пошукова модель – це нейромережа, що покликана відібрати з великої кількості кандидатів тих, що найбільше підпадають під вподобання користувача. У випадках, коли БД містить 100 000 або більше записів, ця модель відіграє важливу роль – відсікає більшість кандидатів та залишить лише 10-100 тих, хто найбільше підпадає під пошукові критерії.
- Рейтингова модель – це нейромережа, що найчастіше бере дані з виходу пошукової моделі та проводить додаткові налаштування, щоб надалі виконувати більш точне прогнозування.

В більшості проектів основна увага приділяється саме першій моделі, оскільки вона здатна незалежно функціонувати та робити більш-менш точні прогнози. Друга використовується для покращення роботи системи рекомендацій та для персоніфікації.

Пошукова модель створюється з двох під моделей:

- Модель запиту (query model) – обчислює представлення пошукового запиту, як правило у вигляді вектора фіксованого розміру, використовуючи його особливості.
- Модель-кандидат (candidate model) – обчислює представлення кандидату та його особливості.

Виходи двох підмоделей формують оцінку спорідненості шляхом множення, де більший результат відповідає кращій відповідності запита до відібраного кандидата.

Рейтингова модель так само використовує query model та candidate model, але додатково ще містить декілька складених Dense шарів. Така архітектура є найбільш поширеною, але є велика кількість модифікацій. Тут, на вхід може подаватися кандидат та назва композиції (може бути будь яка інша характеристика) та на виході отримується прогнозована оцінка користувача (кандидата). Музичні твори з найбільшою потенційною оцінкою потрапляють до списку рекомендацій.

Схема побудови нейромережі представлена на рис. 1.3. Слід зауважити, що вхідні та шари колаборативного фільтрування можуть видозмінюватися. Як правило, це робиться для того, щоб покращити роботи моделі в цілому та отримувати більш точні прогнози.

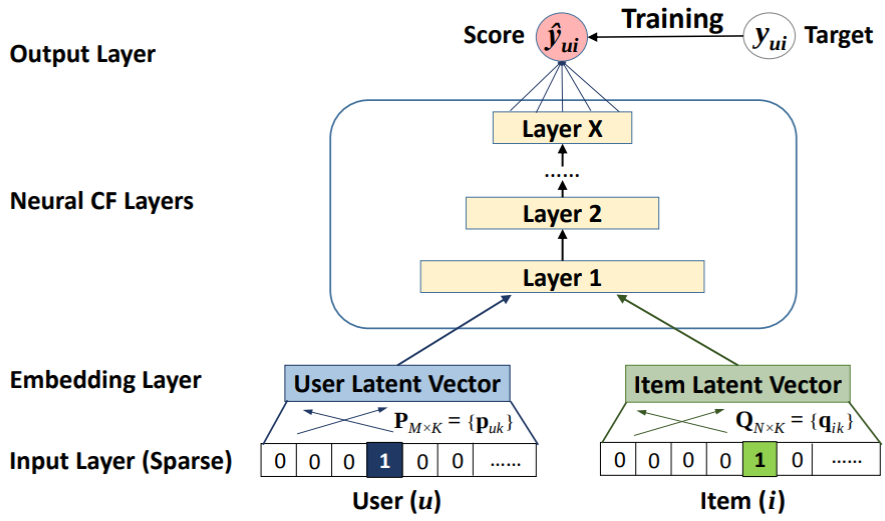

Рисунок 1.3 – Структура Neural Collaborative Filtering (NCF)

Схематичний принцип роботи можна представити на рис. 1.4. Він виглядає нескладно, але всередині захована логіка, що вимагає значних обчислювальних ресурсів.

Вхідний прошарок перетворює в бінарне представлення розріджений вектор для ідентифікації користувача та елемента, де:

- Item (i): 1 означає, що користувач взаємодіяв з Item(i)
- User (u): для ідентифікації користувача

Шар вбудовування (Embeddings) — це повнозв’язаний шар, який проектує розріджене представлення на щільний вектор. Отримані вбудовування користувача/елемента є прихованими векторами користувача/елемента.

Нейронні шари CF використовують багаторівневу нейронну архітектуру. Остаточний вихідний рівень повертає прогнозовану оцінку мінімізуючи точкові/парні втрати.

NCF модифікує рівняння 3, наступним чином:

$$\hat{\square}_{\square\square} = \square(\square^{\square}\square^{\square}_{\square}, \square^{\square}\square^{\square}_{\square}|\square, \square, \square_{\square}),$$

(1.4)

де P – матриця латентних факторів для користувачів (Розмір = M * K);

Q – матриця латентних факторів для елементів (Розмір = N * K);

$\theta_{\square}$ – параметри моделі.

Оскільки f - багаторівневий персептрон, то рівняння може бути представлене наступним чином:

$$\square(\square^{\square}\square^{\square}_{\square}, \square^{\square}\square^{\square}_{\square}) =$$

$$\square_{\square\square\square}(\square_{\square}(\dots\square_{2}(\square_{1}(\square^{\square}\square^{\square}_{\square}, \square^{\square}\square^{\square}_{\square}))\dots)), \quad (1.5)$$

де $\square_{\square\square}(\square\square\square)$ – функція відображення для вихідного шару

$\square_{\square\square}(\square)$ – функція відображення для $\square_{\square}$ шару NCF шару

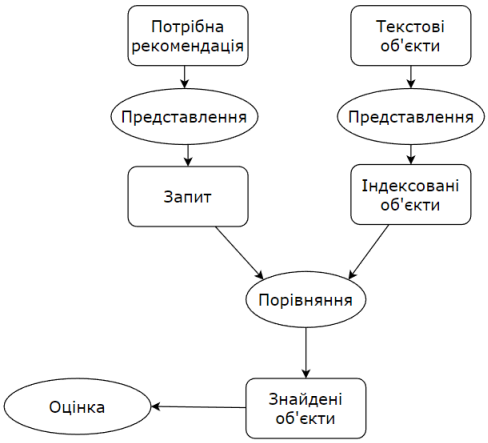

Рисунок 1.4 – Схема роботи NCF

Слід зауважити, що для завантаження вхідних даних необхідно створити прошарок, що буде об’єднувати дві частини моделі. Для вирішення цієї задачі можна використати BruteForce. Це модель, що використовується для репрезентації query моделі.

1.2. Аналіз існуючих інформаційних систем для рекомендації музики

1.2.1. Spotify

Аудіо стрімінгова платформа, одна з найбільших у світі, що з’явилася в 2006 році. Її аудиторія нараховує майже півмільярда активних користувачів, з яких майже половина використовує щомісячну підписку на сервіс[13].

Пропонує велику базу музичних композицій та подкастів, що налічує близько 82 мільйонів позицій від різних лейблів та медіа компаній. Сервіс пропонує безкоштовний функціонал, але він є обмеженим, містить рекламу і т. п. В той же час, доступ за підпискою, що включає багато додаткових можливостей, таких як оффлайн доступ, можливість створення власних плейлистів і багато іншого.

Spotify доступний в більшій частині Європи, Африки, США та Азії. Доступ за підпискою використовують 53% користувачів з Європи та штатів - це складає 67% доходів сервісу. На китайському ринку Spotify програє місцевому гіганту QQ Music. Сервіс доступний на різних операційний системах: Android, IOS, Windows, Linux.

1.2.2. iTunes

Програмний застосунок, що позиціонує себе як медіаплеєр, медіа бібліотека, утилита для менеджменту на мобільному пристрої та клієнтський застосунок для iTunes Store. Розроблений компанією Apple Inc. Використовується для придбання, прослуховування, завантаження та менеджменту мультимедійних файлів на персональних комп’ютерах з системою macOS або Windows (рідше)[14].

Була анонсована та розроблена в 2001 році за ініціативою Стіва Джобса з ціллю надати користувачам зручне сховище медіа даних з можливістю створювати персональні колекції (плейлисти). Після першого

релізу через декілька років було додано можливість придбати та завантажити цифрову музику. Це відкрило новий ринок та подарувало життя такому пристрою як iPod. З наступними версіями в застосунку почали з’являтися різні цифрових файлів - відео, подкасти, електронні книжки та мобільні застосунки, що були придбані через iOS App Store. Але це потягнуло багато критики в бік компаній, оскільки застосунок втратив свій основний фокус на аудіо контент. Тому врешті-решт було прийняте рішення створити різні застосунки та визначити основну мету для кожного, не перетворюючи його на сховище для всіх файлів.

1.2.3. YouTube Music

Музичний стрімінговий сервіс, розроблений Google, що пропонує користувачам зручну платформу з аудіо контентом, інтерфейс якої, схожий до YouTube, але з чіткою орієнтацією на предметну область. Він дозволяє користувачам шукати треки за жанром, плейлистами та рекомендаціями. Останні є доволі точними, в порівнянні з тими, що пропонуються іншими платформами, оскільки Google має можливість навчати свої рекомендаційні нейронні мережі на досить великих об’ємах даних[15].

Сервіс пропонує своїм користувачам платний доступ за підпискою, яка прибирає рекламу, дозволяє завантажувати музику для офлайн прослуховування тощо. Також цей доступ можна отримати тим, хто має YouTube Premium. Але знову таки, тільки ті користувачі, що використовують екосистему Google цілком, отримують від цього найбільше переваг.

1.2.4. Pandora

Музичний стрімінговий сервіс, належить Sirius XM Holdings з головним офісом в Окленді, Каліфорнія, Сполучені Штати. Надає користувачам платний доступ до величезній бібліотеці композицій та

ставить за основну мету формування найбільш персоніфікованих рекомендацій своїм користувачам (в основу покладено Music Genome Project)[16].

Компанія було заснована в 2000 році орієнтована на B2B сегмент та позиціонувала себе як рекомендаційна платформа. Основний фокус був на створенні інтернет-радіо продукта. Сервіс надає обмежений функціонал в безкоштовному режимі та багато можливостей для тих, хто платить за підписку, наприклад, офлайн режим, вища якість звуку, завантаження тощо.

Варто зазначити, що Pandora недоступна в Україні, хоча має багато користувачів з усього світу.

1.2.5. Порівняння існуючих систем для рекомендації музичних творів

Кожна з систем, що були розглянуті раніше, має свої переваги та недоліки, в табл. представлений порівняльний аналіз ключових функцій, що присутні в більшості сучасних платформ для прослуховування музики.

Таблиця 1.1. Порівняльний аналіз існуючих рішень

Сервіс / Критерій	Spotify	iTunes	YouTube Music	Pandora
Зручний інтерфейс	-	-	-	-
Відкритий код	-	-	-	-
Кросплатформеність	+	-	+	+
Рекомендаці	+	+	+	+

йні системи				
Безкоштовність	-	-	-	-
Доступність в Україні	+	+	+	-

З порівняльної таблиці видно, що існуючі системи мають свої недоліки, вони з’являються в більшості через те, що комерційна складова та ринок вимагає постійного впровадження нового функціоналу, це спричиняє перехід від сервісу з чітко визначеною функцією в розряд супер платформи (Super App). Такі зміни не завжди дають позитивні результати, вони вимагають створення екосистеми, бо в іншому випадку стає неможливим утримувати користувача, як результат - зменшення прибутку та в подальшому закриття проектів.

1.3. Аналіз основних процесів предметного середовища

Предметне середовище містить наступні суб’єкти:

- користувач (слухач, адміністратор, виконавець)
- авторизаційний сервіс
- база даних (містить музичні композиції та інформацію, пов’язану з користувачами)

Кожен суб’єкт має унікальні можливості, але є ключові процеси, що їх об’єднують. Основним є процес прослуховування музичних творів, який складається з декількох важливих частин, який є передумовою для отримання рекомендацій, що будуть розглянуті нижче. Контекстна діаграма представлена на рис. 1.5:

Рисунок 1.5 – Контекстна діаграма IDEF0 процесу прослуховування музичних творів

Вимоги до захисту персональних даних висуваються з метою запобігання отримання доступу до аккаунту користувача стороннім особам, найпоширенішим прикладом, є захист за допомогою надійного пароля.

Під бізнес правилами мається на увазі набір зобов’язань, що висувається як до системи в цілому, так і до окремих процесів в ній. Наприклад, процес прослуховування пісень обов’язково супроводжується й наданням відповідних інтерфейсів керування користувачу.

Процес прослуховування можна декомпозувати на 3 основні підпроцеси, представлені на рис. 1.6:

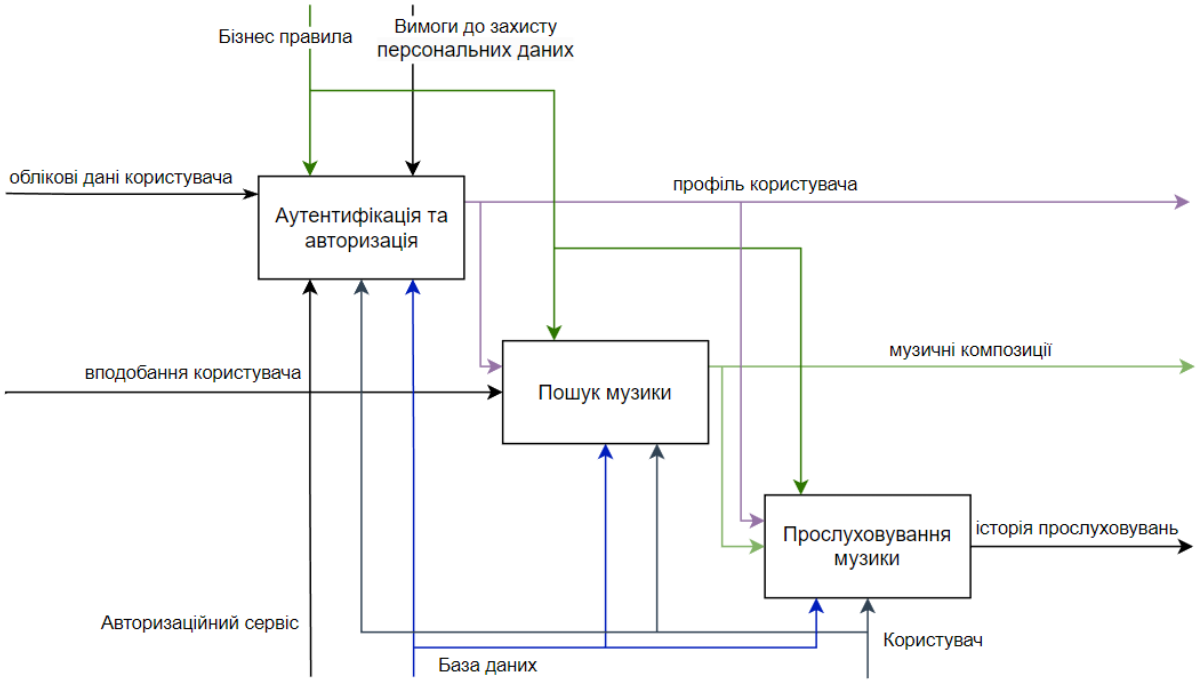

Рисунок 1.6 – Декомпозиція процесу прослуховування музичних творів ЯК-Є

- Автентифікація та авторизація (необхідна для candidate model) - процес, що необхідний для ідентифікації користувачів системи та для надання більш персоніфікованих рекомендацій. Так для кожного профіля буде створюватися історія прослуховувань, яка в подальшому буде використовуватися для навчання моделі.

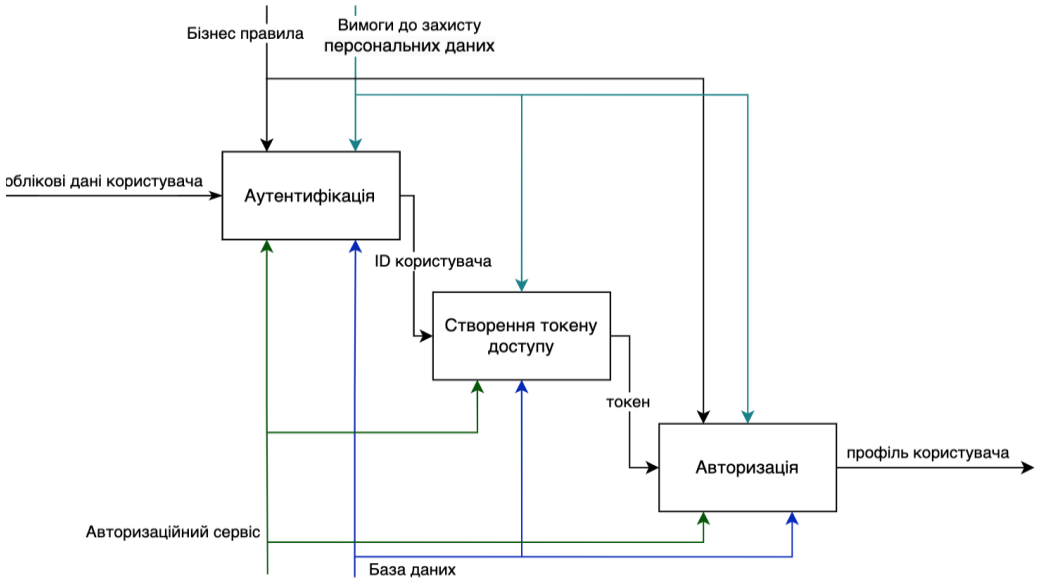

Рисунок 1.7 – Декомпозиція процесу “Автентифікація та авторизація ЯК-Є”

- пошук музики (необхідна для query model) - процес, що необхідно оптимізовувати в будь якій системі, що надає доступ до контенту та обробляє велику кількість даних. Пошук стає невід’ємною частиною програмного застосунку, що впливає на UX.

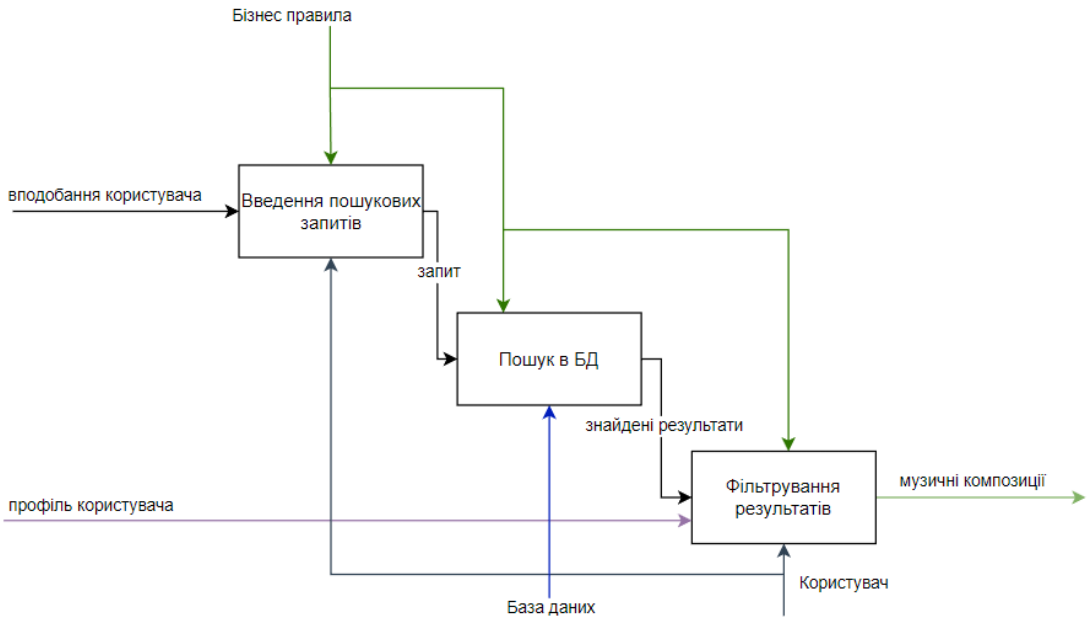

Рисунок 1.8 – Декомпозиція процесу “Пошук музики ЯК-Є”

- формування рекомендацій - основний процес в будь якій рекомендаційній системі, він має бути максимально швидким та простим для кінцевого користувача.

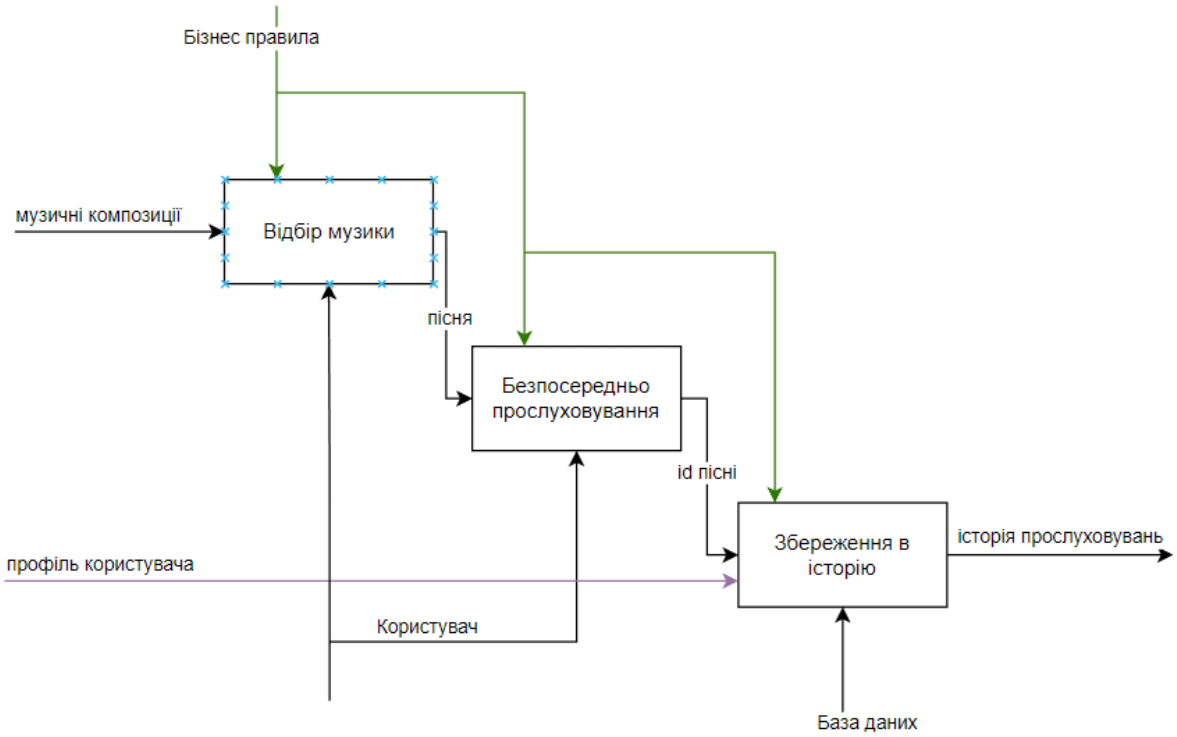

Рисунок 1.9 – Декомпозиція процесу “Прослуховування музики ЯК-Є”

1.4. Постановка задачі на розробку системи

Інтелектуальний програмний продукт має бути представлений як готова до використання система, основна мета якої - надати доступ до БД з музичними творами та запропонувати користувачеві персоніфіковані музичні рекомендації.

Функціональні та нефункціональні вимоги.

Таблиця 1.2. Функціональні вимоги

Вимога	Опис

Авторизація користувачів	Система має надавати рекомендації за вподобаннями користувачів, тому необхідно для кожного створити профіль, доступ в який має бути захищений паролем
Реєстрація та вхід в систему	Клієнт може отримати доступ до застосунку за своїми обліковими даними та токеном (за умови його дійсності)
Доступ з JWT токеном	Для того, щоб використовувати бекенд, кожний endpoint має бути захищений AuthGuard, що буде перевіряти актуальний токен клієнта. Доступ має підтримуватися протягом 24 годин.
Реалізація CRUD операцій над даними про користувачів системи та музичними творами	Застосунок має бути динамічним реагувати на додавання / видалення / редагування записів в БД, оскільки на даному етапі це лише модуль, то при подальшому масштабуванні для його підтримання необхідно тільки оновлювати таблиці з користувачами та музичними творами, будь які інші сутності в БД можуть існувати незалежно
Кількість музичних творів необхідних	Для того, щоб отримати список

для побудови рекомендації має бути обмеженою	рекомендацій музичних творів, необхідно попередньо прослухати як мінімум декілька музичних композицій, для того щоб сформувати релевантні рекомендації. Дані щодо прослуховування необхідно зберігати в БД
Пошук пісень за ключовими словами	Система має надати можливість шукати пісні за назвою (чи іншими ключовими словами, що її характеризують)
Прослуховування музики	Має бути реалізовано можливість слухати музику. Це окрема сторінка для кожної композиції. Вона має бути доступна за унікальним URL (наприклад, …/track/1, де 1 - ID пісні)
Додавання коментарів	До кожної пісні маю бути додана секція з коментарями від користувачів, оновлення має відбуватися за REST правилами

Таблиця 1.3. Нефункціональні вимоги

Вимога	Опис
Реалізувати зручний UI	Створити користувацький інтерфейс, що має відображати: 1. Сторінка авторизації (sign-in/up)

	2. Сторінки для пошуку музичних композицій та функцією autocomplete 3. Сторінку з списком рекомендацій (якщо користувач має історію прослуховувань) 4. Головну сторінку з доступними музичними композиціями 5. Сторінку для кожної пісні окремо (плеєр), для того, щоб була можливість виконувати певні дії (пауза, додавання в улюблені і т. п.) 6. Необхідно відображати плеєр з основними керуючими елементами (прогрес, гучність, пауза)
Реалізувати документацію для backend	За допомогою засобів Swagger API, необхідно описати всі існуючі інтерфейси взаємодії з бекендом. Передбачити інформацію по моделям в БД та всім об’єктам передачі даних (dto)
Валідація даних для надання рекомендацій	Якщо кількість елементів, що надходять від клієнта менша за мінімально необхідну, то треба повертати відповідні помилки з

	серверного застосунку (та/або не надсилати запит з FE). Передбачити валідацію на клієнті (якщо історія має більшу кількість прослуханих композицій, то необхідно брати до уваги останні та відправляти в запиті на пошук рекомендацій тільки нещодавно додані до історії).

1.5. Висновки до першого розділу

Музичні сервіси дають величезні прибутки, тому їх кількість постійно збільшується. Це призводить до того, що необхідно не тільки намагатися отримати нових користувачів, а якимось чином їх утримати. Єдиний спосіб, що дозволяє вирішити цю проблему, це персоніфікація контенту. Більшість компаній це розуміють, тому збільшується попит на створення рекомендаційних систем з високим відсотком персоніфікації.

Рекомендаційні модулі можуть мати різну архітектуру, використовувати різні підходи для оцінки уподобань і т. п., але найбільш перспективними є ті, що базуються на колаборативному фільтруванні на основі нейромереж. Вони досить ефективні при роботі з величезними базами даних та точними у порівнянні з традиційними методами, такими як, наприклад, метод K найближчих сусідів.

Отже, музична рекомендаційна система необхідна, якщо:

- Треба збільшити задоволеність і залученість користувачів
- Створити максильмано персоніфіковану платформу
- Покращити UX сервісу спростивши пошук нових пісень

- Збільшити дохід за рахунок кількості підписок на сервіс
- Отримати корисну інформацію про звички користувачів системи для подальшого розширення та побудови стратегії розвитку

РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ МУЗИЧНОГО ЗАСТОСУНКУ З СИСТЕМОЮ РЕКОМЕНДАЦІЙ НА ОСНОВІ НЕЙРОМЕРЕЖ

2.1 Розробка архітектури

2.1.1 Функціональний аналіз музичного застосунку з системою рекомендацій

Додаток для прослуховування музики з нейромережним модулем для рекомендацій на основі колаборативного фільтрування є дуже вимогливим до функціоналу, оскільки складається з кількох підсистем, що мають взаємодіяти як одна з одною, так й існувати окремо. Такий застосунок може бути створений за допомогою наступних функцій:

- Перегляд списку пісень:
    - Функція, що завантажує список пісень з сервера.
    - Функція, яка відображає список пісень на сторінці клієнта.
- Пошук пісень:
    - Функція, яка дозволяє користувачеві ввести назву пісні або виконавця.
    - Функція, яка шукає пісні в базі даних на сервері за назвою або виконавцем.
    - Функція, яка відображає результати пошуку на сторінці клієнта.
- Програвання пісень:
    - Функція, яка дозволяє користувачеві вибрати пісню зі списку або з результатів пошуку.
    - Функція, яка починає відтворення вибраної пісні на клієнті.
    - Функція, яка відображає назву та виконавця пісні на сторінці клієнта.
- Рекомендації пісень:

- Функція, яка збирає дані про прослуховування пісень користувачем.
- Функція, яка передає дані про прослуховування на сервер для аналізу.
- Функція, яка використовує натреновану модель TensorFlow для генерації рекомендацій на основі колаборативного фільтрування.
- Функція, яка відображає рекомендації пісень на сторінці клієнта.

- Авторизація та реєстрація:
  - Функція, яка дозволяє користувачеві створити обліковий запис.
  - Функція, яка дозволяє користувачеві увійти до свого облікового запису.
  - Функція, яка перевіряє дані авторизації та реєстрації на сервері.

Умовно весь функціонал можна поділити на три основні гілки, що містять клієнтські, серверні та функції нейромережного модуля. На рис. 2. зображені всі основні функції, що мають бути реалізовані в додатку. Враховуючи, що для вирішення подібної задачі, а саме побудови музичного додатку, найчастіше використовуються клієнт-серверна архітектура, тому деякі функції, що знаються подібними реалізуються одразу в двох підсистемах, але при цьому вони працюють по-різному. Окремо необхідно виділити й нейромережний модуль, що має свої специфічні функції, але здебільшого вони можуть бути об’єднані в дві ключові групи - навчання та передбачення.

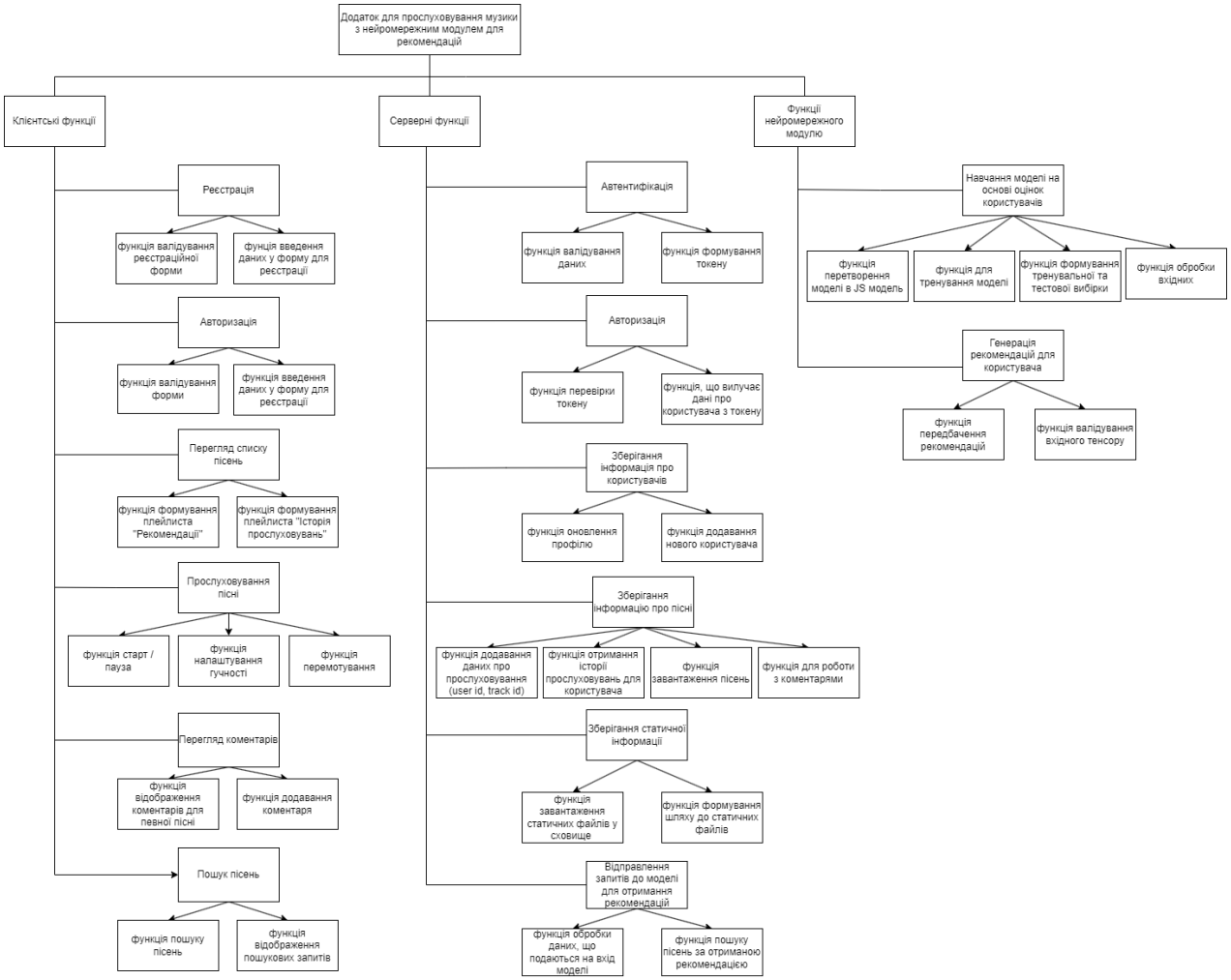

Рисунок 2.1 – Дерево функцій застосунку для прослуховування музики

Щоб більш детально проаналізувати ключові процеси, які відбуваються в додатку з рекомендаціями, можна зобразити їх у вигляді EPC діаграм (рис. 2.2 - 2.3). Такими процесами є:

- п
ошук пісень в застосунку
- ф
ормування рекомендацій

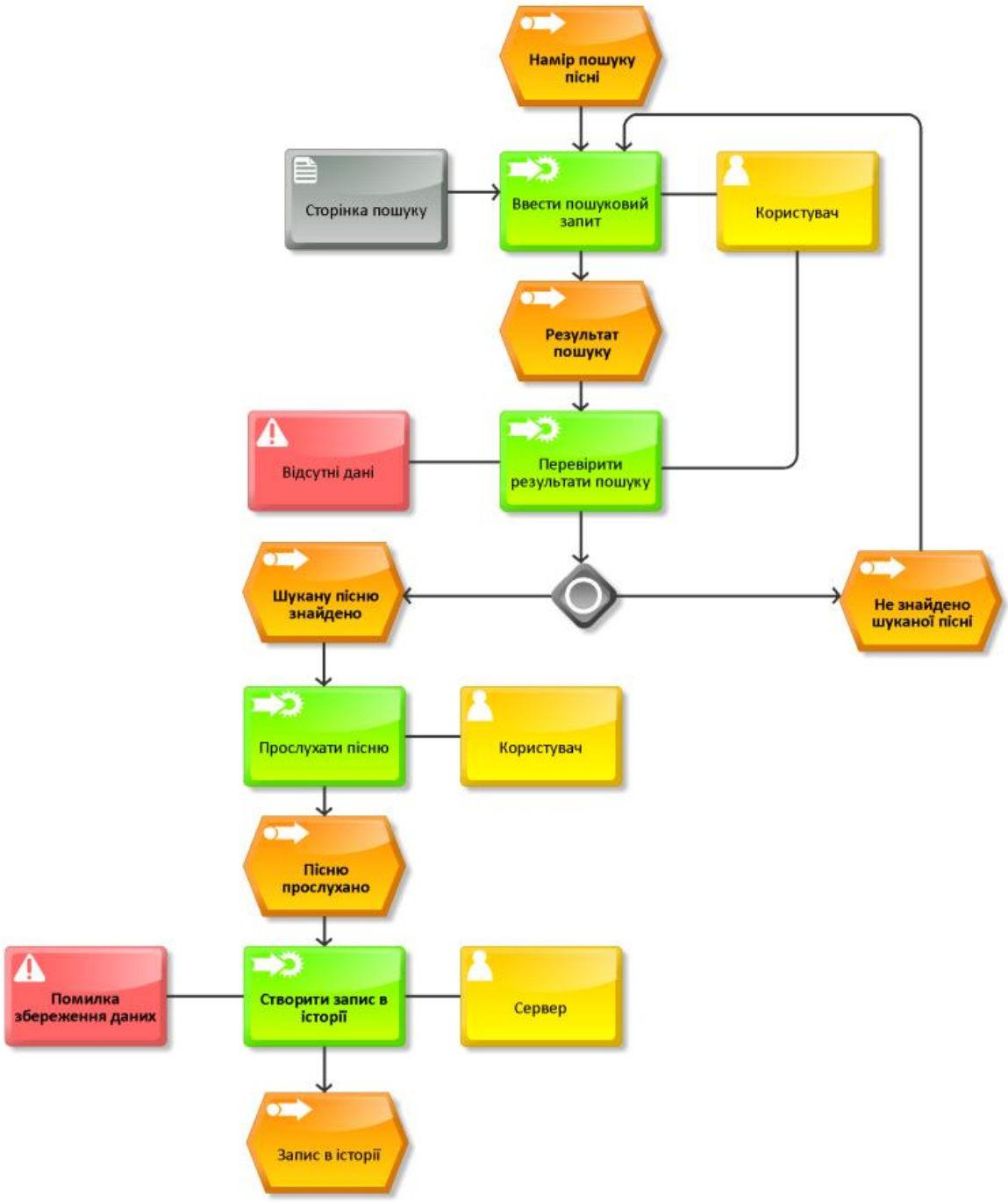

Рисунок 2.2 – Процес пошуку музики в застосунку

Користувач є ключовим учасником процесу пошуку в музичному застосунку, а система відповідає за відпрацювання запитів та зберігання їх в історії. Важливо зазначити, що користувач може здійснювати пошук багаторазово, доки не отримає задовільний результат. Система, в свою чергу, фіксує проблеми, що можуть виникати під час взаємодії з

користувачем і т. п., головна мета полягає в тому, щоб забезпечити користувачеві те, що він шукає та зібрати звіти щодо активності - таких як прослуховування.

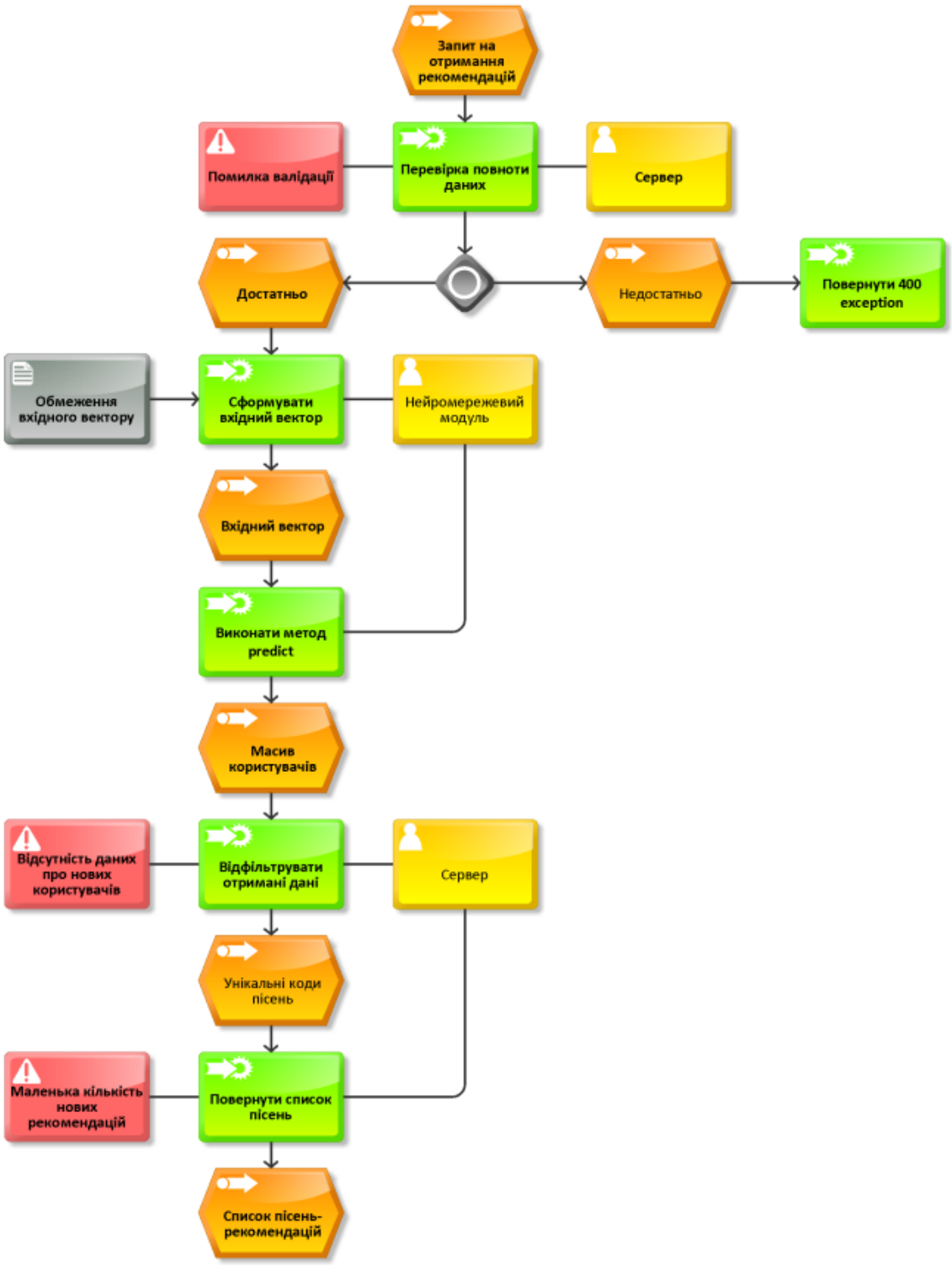

Рисунок 2.3 – Процес створення рекомендацій

Надання персоніфікованих рекомендацій (рис. 2.3) відбувається тільки за запитом авторизованого користувача із врахуванням його історії прослуховувань. Це необхідно для того, щоб список рекомендацій відображав різноманітність музичних смаків користувача. Тут основні ролі отримають сервер та нейромережевий модуль, які тісно взаємодіють між собою, щоб згенерувати та дешифрувати (predict повертає рекомендованих користувачів, а не одразу пісні) виведення. Варто зазначити, що останній етап можна реалізовувати багатьма способами, все залежить від того, наскільки адекватним буде результат роботи нейромережі. Наприклад, може статися випадок, коли рекомендованих пісень немає, або їх замала кількість (навіть менше за вхідний вектор), тоді слід розглядати кількох споріднених користувачів, а не одного.

2.1.2 Аналіз процесу прослуховування музичних творів

Процес прослуховування музики вже був раніше представлений, але при створенні музичного застосунку необхідно врахувати, що без рекомендаційної системи, він не може утримувати користувачів, оскільки немає персоніфікованих плейлистів та пошукових асистентів тощо. Нижче представлена діаграма процесу прослуховування музики, але з запровадженою рекомендаційною системою (рис. 2.):

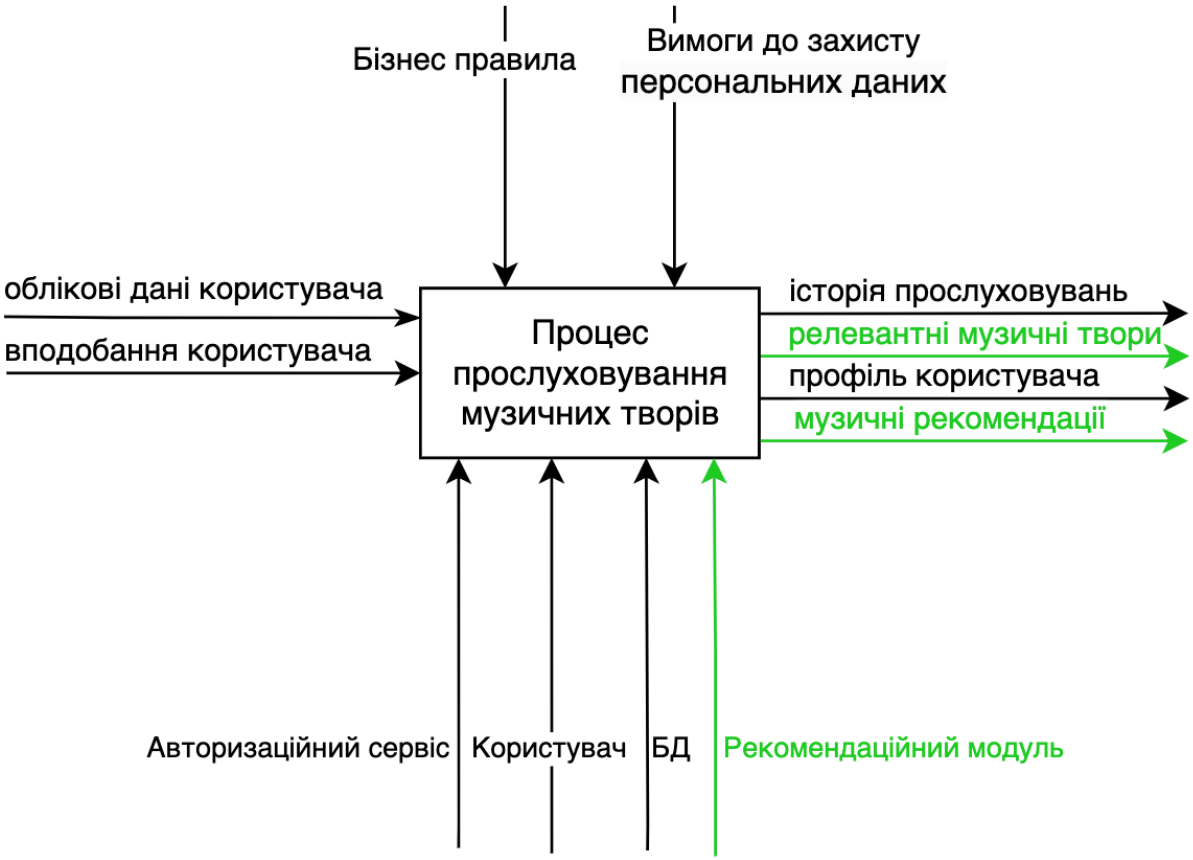

Рисунок 2.1 – Контекстна діаграма IDEF0 процесу прослуховування музичних творів ЯК-БУДЕ

Разом з рекомендаційним модулем з’явився один додаткових вихід - музичні рекомендації, музичні твори ж змінилися на релевантні музичні твори. Такі зміни достатньо сильно впливають на те, як надалі може бути використаний музичний застосунок,оскільки він тепер наділений персоніфікованістю.

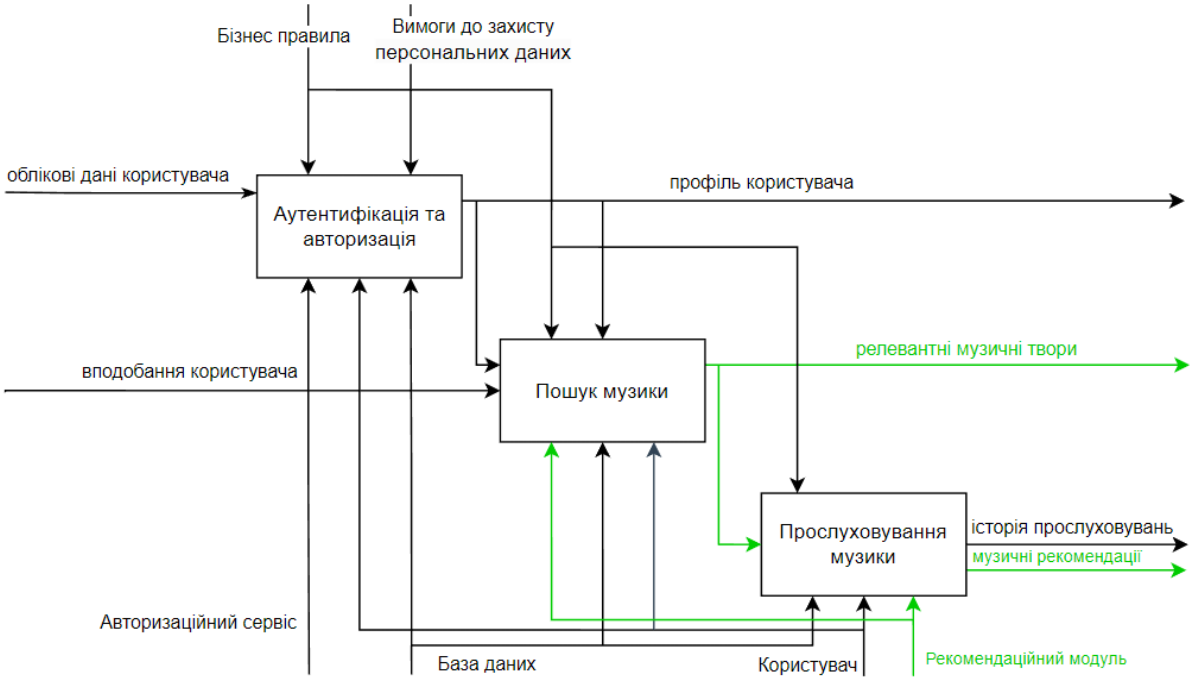

Рисунок 2.2 – Декомпозиція процесу прослуховування музичних творів ЯК-БУДЕ

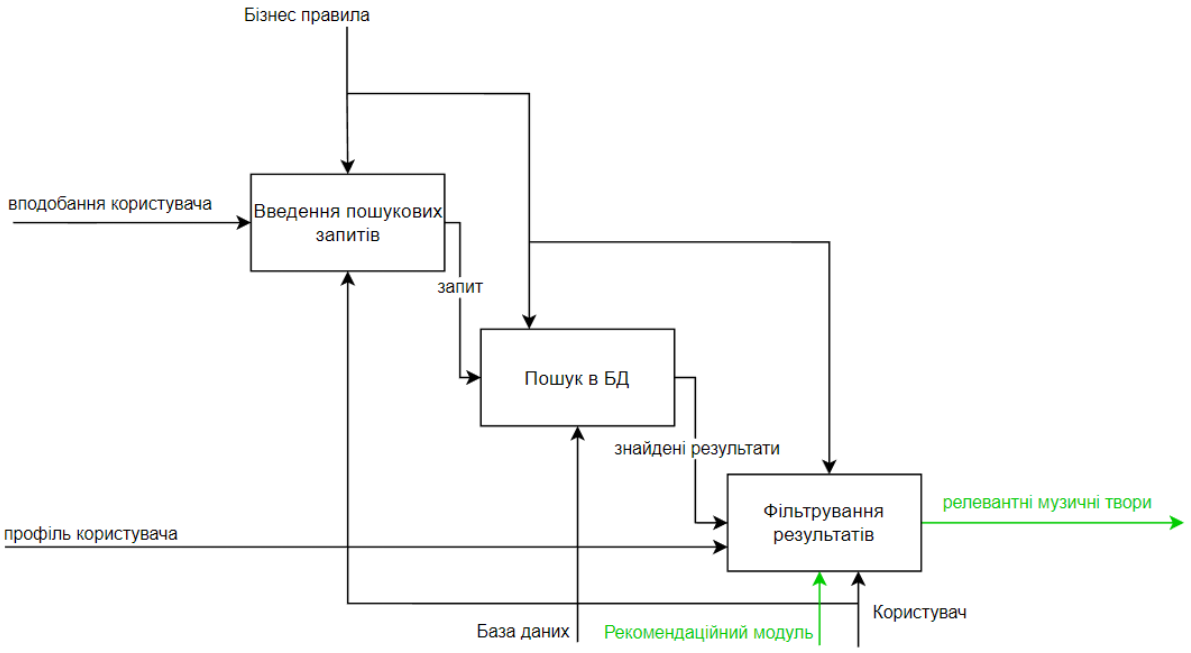

Рисунок 2.3 – Декомпозиція процесу “Пошук музики ЯК БУДЕ”

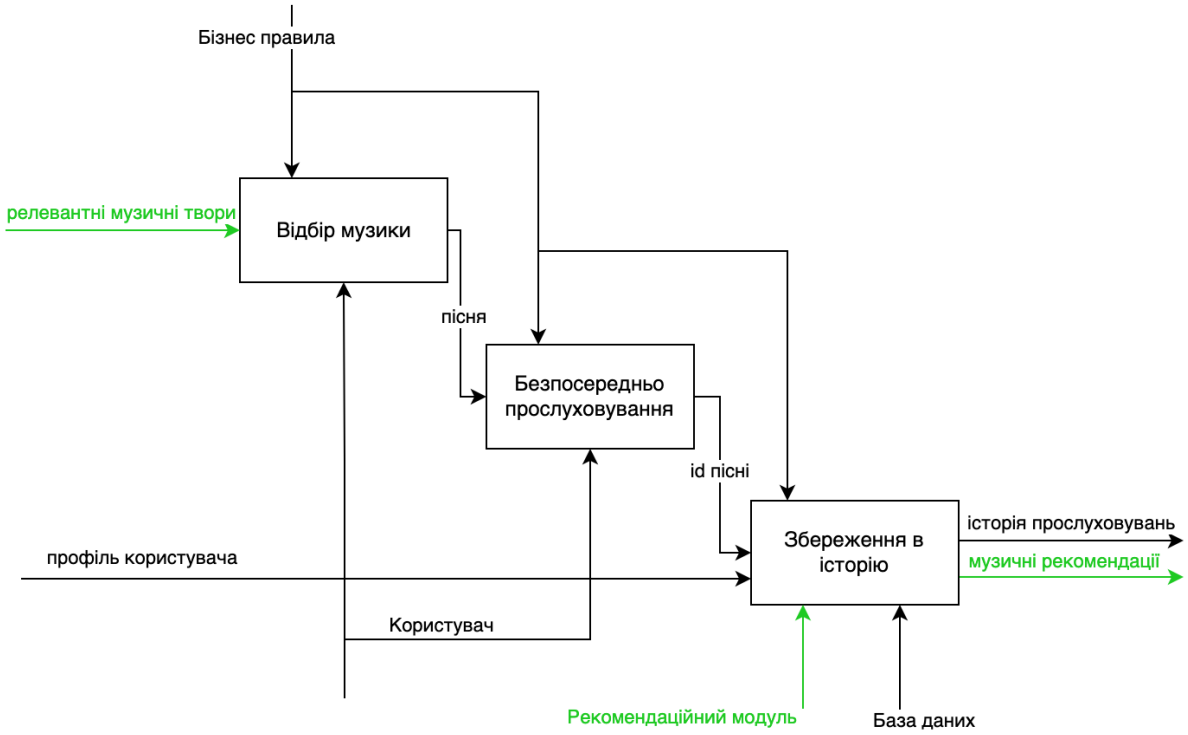

Рисунок 2.4 – Декомпозиція процесу “Прослуховування музики ЯК БУДЕ”

2.1.3 Архітектура інформаційної системи музичного застосунку

Архітектура музичного застосунку має складатися з кількох основних частин:

- Клієнтський інтерфейс – веб застосунок, який надає користувачам доступ до музичних творів, формує плейлисти, дозволяє створювати власні профілі, прослуховувати контент та отримувати рекомендації на основі їх уподобань.
- Сервер – набір контролерів та сервісів, що забезпечують взаємодію з БД, надають доступ до аналітики – модуль забезпечує збір статистичних даних про користувачів, їхні відгуки та поведінку в додатку, що дозволяє аналізувати та вдосконалювати процес надання рекомендацій. Містить документацію, для того щоб надати можливість інтеграції з клієнтським застосунком.

- Рекомендаційний модуль (модуль обробки рекомендацій) – аналізує інформацію про інтереси та уподобання користувача і, використовуючи метод колаборативного фільтрування, видає рекомендації щодо музичних композицій, які ймовірно сподобаються користувачу.

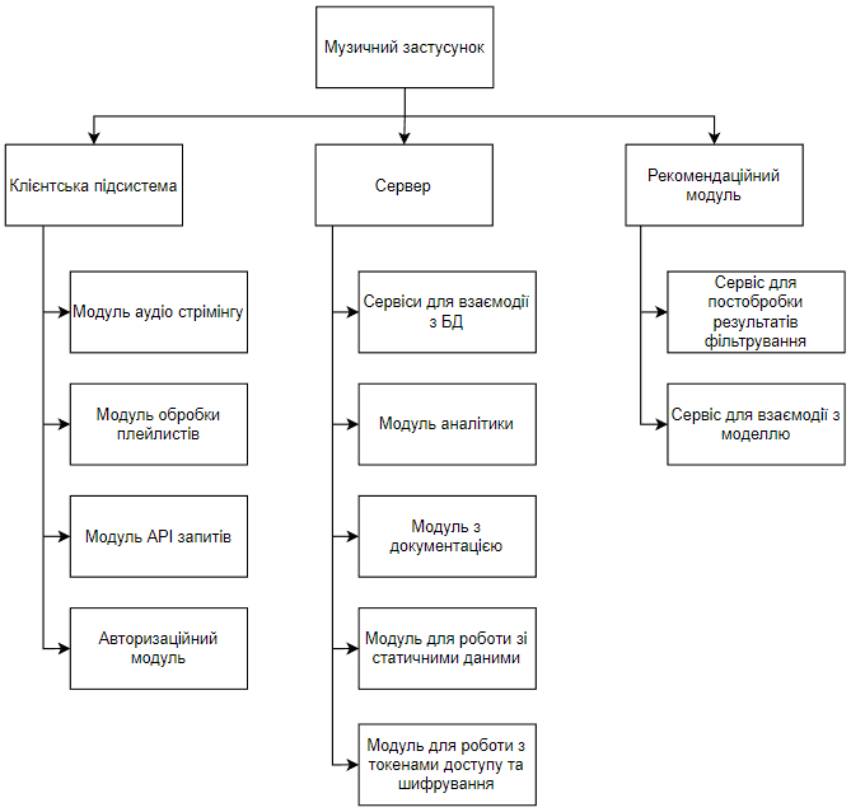

Рисунок 2.5 – Інформаційна система музичного застосунку

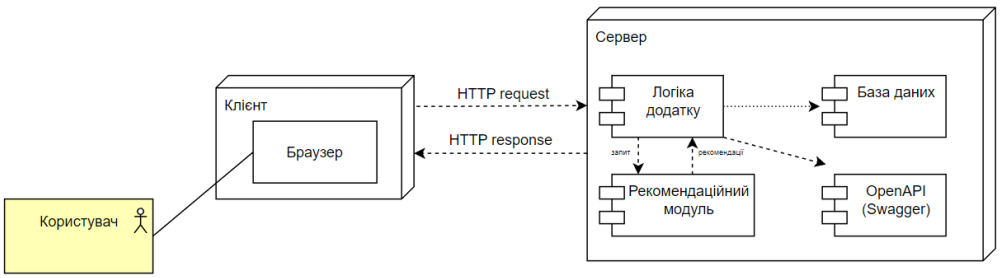

Рисунок 2.6 – Архітектура прикладної програми для прослуховування музики з нейромережним модулем

2.2 Інформаційне забезпечення музичного застосунку з нейромережним модулем

2.2.1 Розробка інформаційного забезпечення

За допомогою колаборативного фільтрування збирається інформацію про користувачів та їхні уподобання щодо музики, а потім на основі цих даних створюється рекомендації для кожного користувача. Для зберігання цих даних потрібно використовувати базу даних, яка допоможе легко зберігати й так само отримувати інформацію про користувачів, пісні, прослуховування та їх відношення між собою. В табл. 2.1 наведено опис сутностей та в табл. 2.2 опис їх взаємодії.

Таблиця 2.1. Опис сутностей інфологічної моделі

№ п/п	Назва атрибуту	Ідентифікатор	Характеристики атрибуту		
			Тип значення (довжина поля)	Обмеження на довжину/ значення	Признак ключового атрибуту
Сутність «Song»					
1	Код пісні	song_id	Ціле	1..n	PK
2	Назва	song	Символьний	1…10000	-
3	Пошуковий запит	search	Символьний	1…10000	-
4	Виконавець	atrist	Символьний	1…10000	-
5	Аудіо	audio	Символьний	0…10000	-
	Зображення	image	Символьний	0…10000	-
Сутність «User»					
1	Код користувача	user_id	Ціле	1…n	PK

2	Ім’я	name	Символьний	1…10000	-
3	Пошта	email	Символьний	1…10000	-
Сутність «Comment»					
1	Код коментаря	comment_id	Ціле	1…n	PK
2	Текст	text	Символьний	1...100	-
3	Код користувача	user_id	Ціле	1…n	FK
4	Код пісні	song_id	Ціле	1…n	FK

Також необхідно зазначити всі зв’язки, що є між сутностями та надати пояснення природи їх існування (табл. 2.2):

Таблиця 2.2. Опис зв’язків між сутностями інфологічної моделі

№ п/п	Сутності, що утворюють зв’язок	Тип зв’язку	Пояснення
1	Song – User	n : m	Кожен користувач може прослухати багато пісень, так само як і кожну пісню можуть прослухати багато користувачів
2	Song - Comment	1 : 0..n	Кожен коментар належить тільки одній пісні, але пісня може мати декілька коментарів
3	Comment – User	0..n : 1	Кожний коментар може бути створений тільки одним користувачем, але кожен користувач може залишати багато коментарів

На рис. 2. наведено концептуальну модель такої БД:

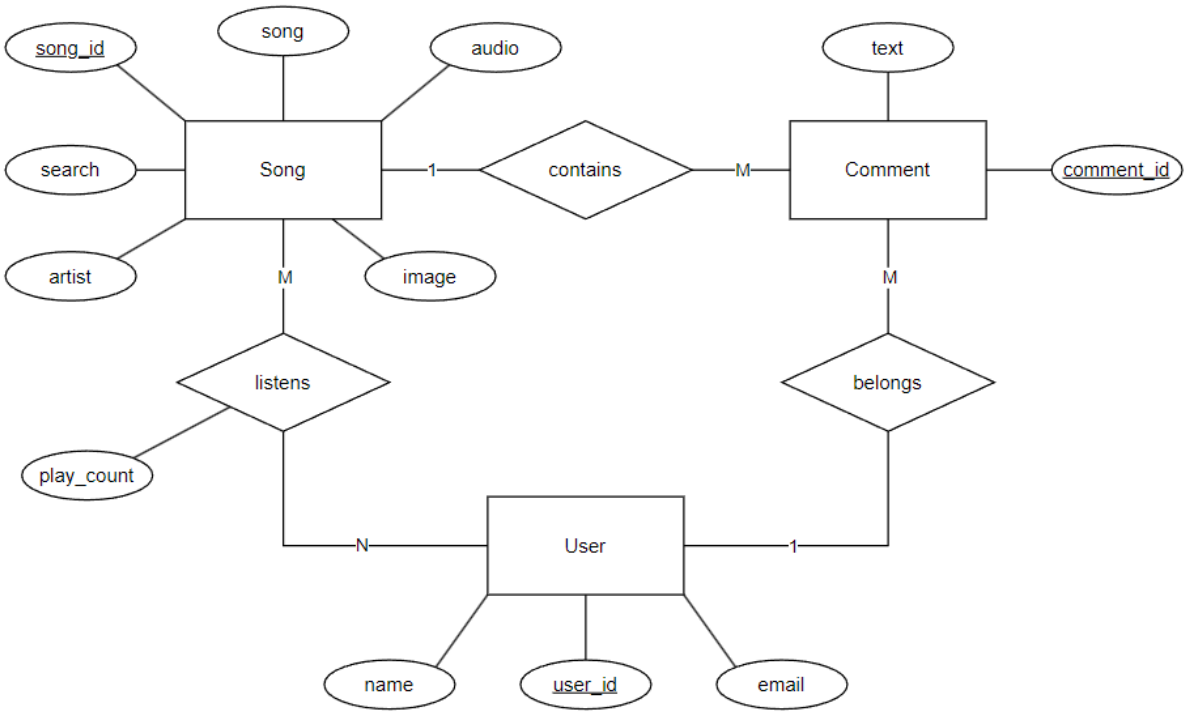

Рисунок 2.9 – Концептуальна модель БД

Для створення концептуальної моделі було використано програму Power Designer, яка надає можливість згенерувати логічну (рис. 2.10) та фізичну моделі (рис. 2.11) при цьому розв'язавши проблему багато до багатьох, що виникає між деякими сутностями.

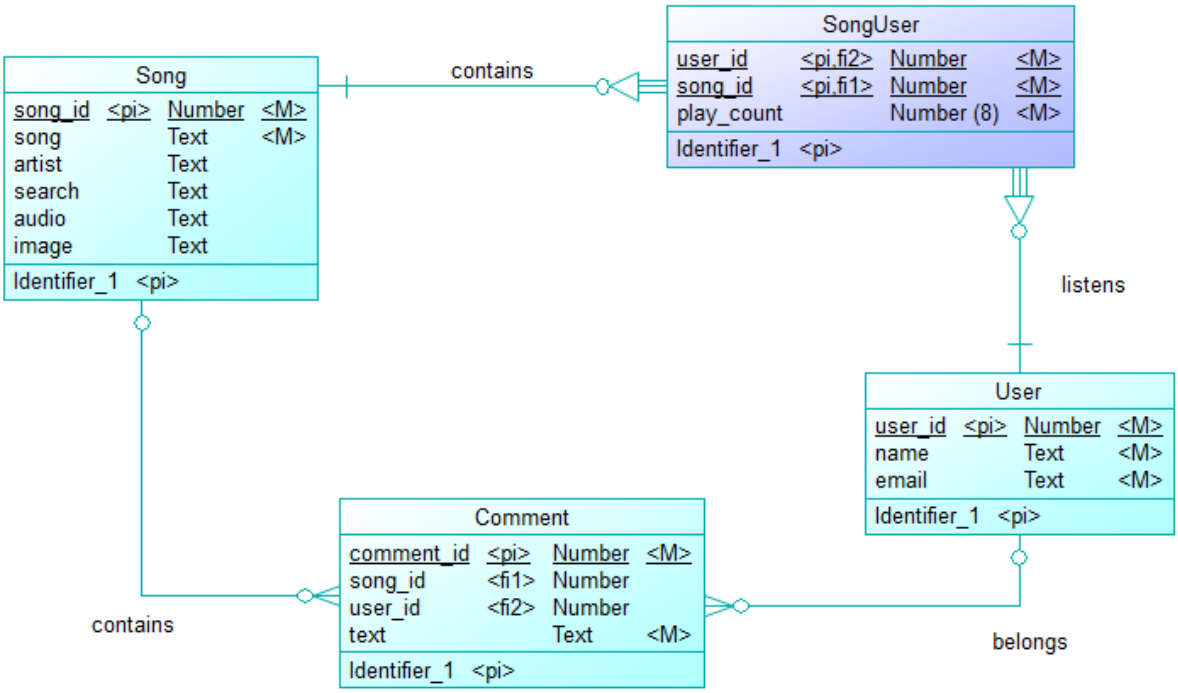

Рисунок 2.10 – Логічна модель БД

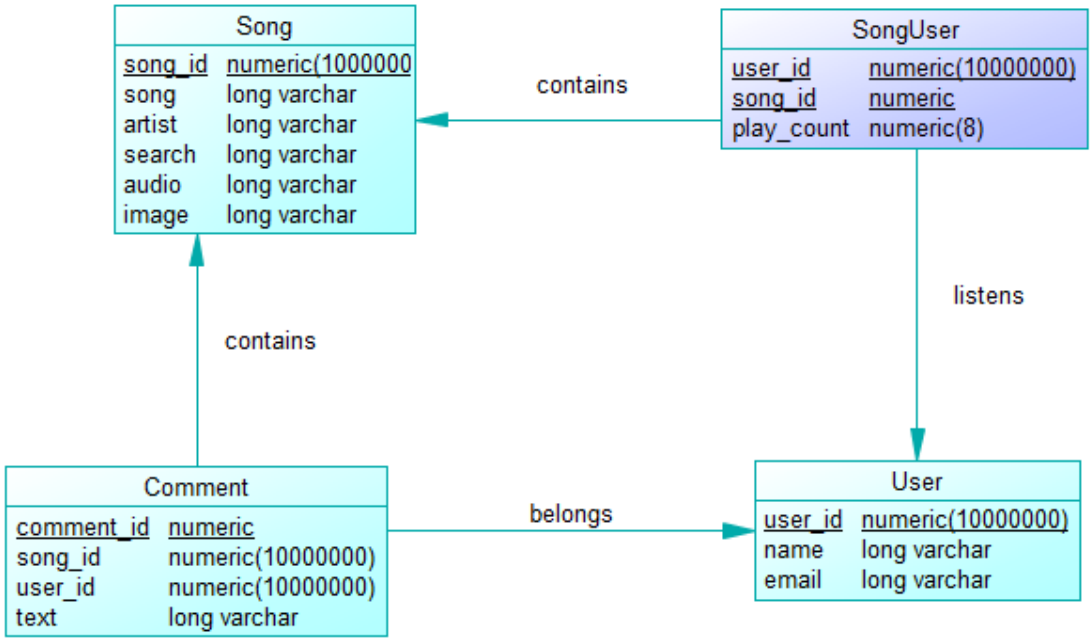

Рисунок 2.11 – Фізична модель БД

2.3 Математичне забезпечення для розробки нейромережі з колаборативним фільтруванням

Архітектура нейромережі для побудови рекомендацій будується наступним чином. Спочатку визначаються параметри моделі, такі як розмірність вкладення (embedding) для треків (n_embedding_dimensions) та кількість пісень (n_songs_in) та користувачів (n_users). Далі створюються дві моделі - QUERY (tracks) та CANDIDATE (user)[9].

Модель QUERY отримує на вхід вектор з n_songs_in значень, що представляє попередньо відібрані користувачем треки. Вектор проходить через вкладення розмірності n_embedding_dimensions, після чого застосовується згладжування та середнє пулінгу. Ця модель має на виході вектор з розмірністю (n_embedding_dimensions,).

Модель CANDIDATE отримує на вхід числове значення, що представляє ID користувача. Значення також проходить через вкладення розмірності n_embedding_dimensions та згладжується до вектора розмірністю (n_embedding_dimensions,).

Далі моделі QUERY та CANDIDATE використовуються для побудови повної моделі з допомогою TFRS. Завданням моделі є рекомендація треків користувачеві. Для цього використовується метрика FactorizedTopK, яка вимірює точність рекомендацій.

У кінці, модель компілюється з оптимізатором Adagrad, навчається на датасеті та зберігається. Потім використовуючи метод BruteForce з TFRS, створюється індекс, який дозволяє швидко знаходити найбільш відповідних користувачів для відповідних треків.

Під час тестування, для першого користувача з датасету виводиться вектор треків та найбільш рекомендовані користувачі для цих треків.

Тут важливо розуміти, що модель не порівнює вектор запиту з кожним вектором користувача в базі даних, оскільки це вимагало значний обчислювальних ресурсів та часу. Замість цього, модель використовує

техніку ранжування, щоб ефективно знайти найкращі відповіді. Кожен вектор користувача оцінюється за допомогою певної метрики близькості, яка використовується для порівняння з вектором запиту. Модель потім повертає найкращі результати відповідно до метрики близькості.

Для ранжування використовується метрика FactorizedTopK, яка ранжує кандидатів за відстанню до запиту в просторі вбудовувань. Кожен користувач та кожна пісня представляються у вигляді вектора вбудовувань, а відстань між двома векторами обчислюється за допомогою косинусної схожості[10].

Процес ранжування полягає в обчисленні вектора вбудовування для запиту (пісень, які слухає користувач) та для кожного кандидата (користувачів, які можуть зацікавитися певними піснями), а потім обчисленні косинусної схожості між запитом та кожним кандидатом. Кандидати ранжуються за зростанням відстані, тобто ті, що мають меншу відстань, отримують вищий рейтинг. Найкращі k кандидатів за рейтингом вважаються успішними рекомендаціями і повертаються моделлю.

Вектор вбудування (embedding vector) – це числовий вектор, який відображає категоріальні ознаки (такі як назва треку, виконавець, жанр тощо) у простір з числовими координатами.

Цей метод є досить ефективним для задач побудови рекомендацій, оскільки дозволяє ефективно працювати з великими наборами даних та знаходити персоналізовані рекомендації для кожного користувача.

Замість використання ID пісень безпосередньо в моделі, використовується їх векторне представлення. Таким чином, модель може знаходити схожість між піснями на основі їх векторів вбудування. Прикладом для ID=15, може бути наступне представлення: [0.9, -0.2, 0.5, 0.1, ..., 0.6].

Косинусна схожість – це міра схожості між двома векторами, яка визначається косинусом кута між ними. Ця міра може бути використана для

порівняння двох векторів в багатьох задачах, включаючи формування рекомендацій.

Щоб обрахувати косинусну схожість між двома векторами, спочатку необхідно визначити їхню довжину (або норму). Для вектора a це можна зробити за допомогою формули:

$$||□|| = \sqrt{□_1^2 + □_2^2 + \ldots + □_□^2},$$

(2.1)

де $□_1, □_2, \ldots, □_□$ - координати вектора a.

Потім необхідно обчислити добуток скалярів двох векторів. Наприклад, для векторів a і b це буде:

$$□ * □ = □_1 * □_1 + □_2 * □_2 + \ldots + □_□ □_□$$

(2.2)

Косинусну схожість між двома векторами можна обчислити за допомогою наступної формули:

$$□□□□□□□□□□□□□□(□, □) = \frac{□ * □}{||□|| * ||□||}$$

(2.3)

Отже, якщо у нас є два вектори, наприклад, a = [1, 2, 3] і b = [4, 5, 6], то спочатку необхідно обчислити їхні норми:

$$||□|| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$$

$$||□|| = \sqrt{4^2 + 5^2 + 6^2} = \sqrt{77}$$

Потім знайти добуток скалярів:

$$□ * □ = 1 * 4 + 2 * 5 + 3 * 6 = 32$$

І нарешті, можна обчислити косинусну схожість:

$$□□□□□□□□□□□□□□(□, □) = \frac{32}{\sqrt{14}\sqrt{77}} = 0.9746$$

Отриманий результат майже дорівнює одиниці. Це означає, що вектори a і b мають досить високу схожість.

2.4 Висновки до другого розділу

У результаті розробки архітектури прикладної програми для прослуховування музики з системою рекомендацій на основі колаборативного фільтрування з використанням нейромережі було проведено функціональний аналіз додатку, розроблено IDEF0 процес прослуховування музичних творів, проаналізовано інформаційні потоки та розроблено концептуальну, логічну та фізичну моделі бази даних.

Результатом функціонального аналізу є дерево функцій, яке відображає основні функції, що виконує додаток. Було виділено функції пошуку та відтворення пісень, додавання їх до списку прослуховування та отримання рекомендацій музичних творів на основі колаборативного фільтрування з використанням нейромережі.

Для процесу прослуховування музичних творів було розроблено IDEF0 діаграму, яка показує послідовність дій користувача та інформаційних потоків між користувачем та системою.

Було розроблено концептуальну, логічну та фізичну моделі бази даних, які відображають структуру та зв'язки між таблицями, а також типи даних, індекси та зв'язки між ними.

Таким чином, в результаті розробки архітектури було визначено основні функції додатку, описано їх послідовність та взаємодію з іншими процесами, а також розроблено базу даних для зберігання музичних творів, користувачів та їх історію прослуховувань.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МУЗИЧНОГО ЗАСТОСУНКУ З СИСТЕМОЮ РЕКОМЕНДАЦІЙ НА ОСНОВІ НЕЙРОМЕРЕЖ

3.1 Обґрунтування вибору програмних засобів

При виборі програмних засобів для розробки, краще звертати увагу на популярні та ефективні в розробці веб-додатків та машинного навчання рішення. Це необхідно для швидкого аналізу помилок і пошуку оптимальних рішень загальновідомих проблем. Також варто враховувати можливості масштабування, які надає фреймворк чи бібліотека, щоб впровадження нового функціоналу не вимагало величезних зусиль та не призводило до переписування цілих модулів.

Для бекенду було обрано Nest JS – це фреймворк, який дозволяє легко та швидко створювати масштабовані та гнучкі серверні додатки на Node.js мовою Typescript. Він пропонує широкі можливості для взаємодії з базами даних та забезпечує безпеку та захист від зловмисників. Надає можливість будувати модульну та/або мікросервісну архітектуру, яку легко підтримувати та розвивати в майбутньому. Швидкість роботи такого бекенду буде досить високою, оскільки Node JS використовує механізм асинхронної взаємодії з довготривалими операціями та не блокує інших клієнтів. Окрім цього, немає необхідності вирішувати проблеми з багатопоточністю, бо архітектура є однопотокова[7].

Для зберігання даних було обрано PostgreSQL, оскільки це одна з найпопулярніших та надійних реляційних баз даних. PostgreSQL забезпечує високий рівень безпеки, масштабованості, швидкості при роботі з даними, а також має багато корисних інструментів для роботи з ними. Тут також представлені деякі типи даних, що покращують досвід розробника, наприклад, money, xml, json.

Звичайно, що власноруч робити SQL запити немає сенсу, оскільки в більшості випадків вони не будуть оптимізованими. Для вирішення цієї задачі можна використати одну з ORM систем. Для застосунку побудови застосунку можна використати TypeORM. Це дуже популярне рішення, що без проблем взаємодіє з будь якою реляційною БД та до цього ж, має типізацію, яку необхідно підтримувати при використанні Typescript.

Для того, щоб зібрати бекенд застосунок, краще всього використати Docker compose, бо необхідно налаштувати запуск не тільки застосунку, а й, наприклад, для postgres. При подальшому масштабуванні може виникнути необхідність додати Redis, чи розбити архітектуру на мікросервіси. Для цього потрібно буде описати взаємодію між портами, щоб не виникало конфліктів.

Для розробки фронтенду було обрано React та Next JS. Хоча React є лише бібліотекою інструментів, використання його в поєднанні з фреймворком Next JS дозволяє максимально реалізувати її потенціал. Next JS забезпечує можливості для створення рішень зі статичним та серверним рендерингом, що дозволяє досягти високої продуктивності та оптимізувати роботу з сервером[8].

Ці технології дозволяють створювати динамічні та інтерактивні інтерфейси користувача швидко та легко. React має широку підтримку та активну спільноту розробників, що дозволяє легко знайти відповіді на питання та знайти рішення на випадок проблем. За допомогою Next JS можна створювати високопродуктивні додатки з високою швидкістю завантаження сторінок, що є важливим для забезпечення задоволення користувачів.

Завдання полягає у створенні нейромережі за допомогою бібліотеки Tensorflow на мові Python, а потім конвертування моделі в JS з використанням інструментів Tensorflow JS. Це необхідно для того, щоб

модель можна було використовувати на Node JS бекенді без використання gRPC чи будь яких інших засобів комунікації між мікросервісами, написаними різними мовами програмування.

Отже, такі рішення дозволяють створювати програмні застосунки з покращеними можливостями розширення та забезпечення високої швидкості роботи. При використанні популярних рішень, можна швидко знайти відповідну документацію та знайти рішення загальновідомих проблем.

3.2 Структура програмного забезпечення музичного застосунку з нейромережним модулем для рекомендацій

Для того, щоб показати взаємодію внутрішніх компонентів системи, варто відобразити її на структурних схемах (рис. 3.1) та навести специфікацію (табл. 3.1).

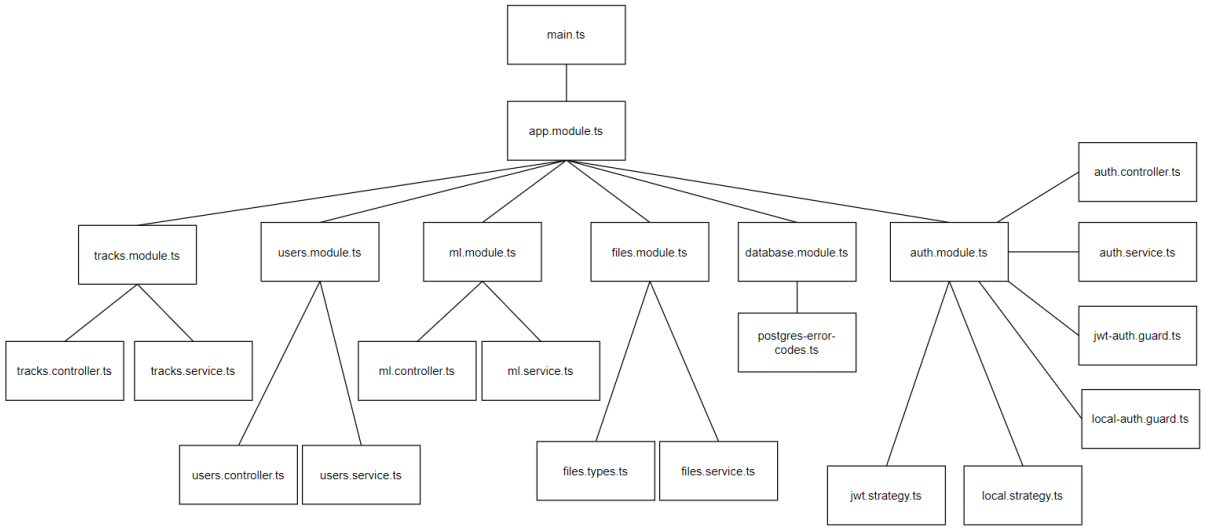

Рисунок 3.1 – Структурна схема програмних модулів сервера

Таблиця 3.1. Специфікація програмних модулів серверного застосунку

Модуль	Опис
main.ts	Головний модуль, що запускає, всі

	модулі, ініціалізує документацію та викликає глобальні middleware. Вхідна інформація: - змінні середовища - конфігурація Вихідна інформація: - запущений екземпляр сервера Вид доступу: відкритий
app.module.ts	Початковий модуль, інтегрує всі модулі, налаштовує роботу зі статичними даними (файлами). Вхідна інформація - Всі модулі - Валідація змінних середовища Вихідна інформація: - AppModule Вид доступу: відкритий
tracks.module.ts	Модуль для роботи з треками. Вхідна інформація - Моделі Song, UsersSongs, Comment, User - FilesModule Вихідна інформація: - TracksModule Вид доступу: відкритий
tracks.controller.ts	Контроллер, який містить кінцеві

	точки для обробки інформації, що стосується треків. Вхідна інформація - Сервіс для роботи з треками - типи та DTO Вихідна інформація: - TracksController Вид доступу: містить захищені кінцеві точки
tracks.service.ts	Сервіс для роботи з треками. Вхідна інформація - Репозиторії: Comment, Song, UsersSongs - Сервіси: Users, Files Вихідна інформація - TracksService Вид доступу: відкритий
users.module.ts	Модуль для роботи з користувачами. Вхідна інформація - Моделі: User - FilesModule Вихідна інформація: - UsersModule Вид доступу: відкритий
users.controller.ts	Контроллер, який містить кінцеві точки для обробки інформації, що

	стосується користувачів. Вхідна інформація - Сервіс для роботи з користувачами - типи та DTO - інтерсептори Вихідна інформація: - TracksController Вид доступу: містить захищені кінцеві точки
users.service.ts	Сервіс для роботи з користувачами. Вхідна інформація - Репозиторії: Users - Сервіси: Files Вихідна інформація - UsersService Вид доступу: відкритий
ml.module.ts	Модуль для роботи з інтелектуальною моделлю. Вхідна інформація - TracksModule Вихідна інформація: - MLModule Вид доступу: відкритий
ml.controller.ts	Контроллер, що працює з моделлю: передає вхідні дані та повертає результати прогнозування.

	Вхідна інформація - Сервіс для роботи з інтелектуальною моделлю - типи та DTO Вихідна інформація: - MLController Вид доступу: містить захищені кінцеві точки
ml.service.ts	Сервіс для роботи з користувачами. Вхідна інформація - Репозиторії: Users - Сервіси: Files Вихідна інформація - UsersService Вид доступу: відкритий
files.module.ts	Модуль для роботи з файлами. Вихідна інформація: - FilesModule Вид доступу: відкритий
files.service.ts	Сервіс для роботи з користувачами. Вхідна інформація - Репозиторії: Users - Сервіси: Files Вихідна інформація - UsersService Вид доступу: відкритий

database.module.ts	Модуль для роботи з БД. Вхідна інформація - TypeOrmModule Вихідна інформація: - DatabaseModule Вид доступу: відкритий
auth.module.ts	Модуль для роботи з треками. Вхідна інформація - UsersModule, PassportModule, JwtModule, ConfigModule Вихідна інформація: - AuthModule Вид доступу: відкритий
auth.controller.ts	Контроллер, який містить кінцеві точки для здійснення аутентифікації користувачів. Вхідна інформація - Сервіс аутентифікації - типи та DTO - guards Вихідна інформація: - AuthController Вид доступу: містить захищені кінцеві точки
jwt-auth.guard.ts	Клас для ініціалізації та налаштування JWT

local-auth.guard.ts	Клас для ініціалізації та налаштування аутентифікації за допомогою email та password
jwt.strategy.ts	Клас для валідації користувачів за їх токенами
local.strategy.ts	Клас для валідації користувачів за їх email та password

Так само можна зобразити й клієнтську частину (рис. 3.2) та навести специфікацію (табл. 3.2).

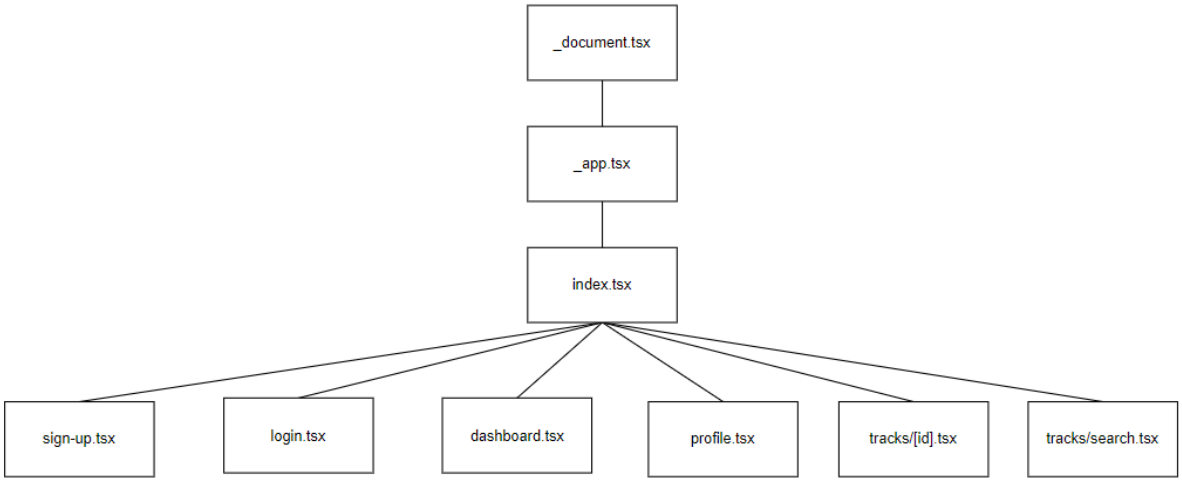

Рисунок 3.2 – Структурна схема програмних модулів клієнта

Таблиця 3.2. Специфікація програмних модулів клієнтського застосунку

Модуль	Опис
_document.tsx	Головний вхідний файл застосунку, що містить основні налаштування для його існування та базові залежності (шрифти, стилі, скрипти).

_app.tsx	Файл, що використовується для налаштування зовнішній провайдерів (обгорток) для всіх модулів в застосунку.
index.tsx	Сторінка, що відкривається при використанні базового шляху: /. Містить логіку для захисту застосунку від неавторизованого доступу.
dashboard.tsx	Сторінка з панеллю, що містить наступні компоненти: - Логотип - Навігаційна панель, що дозволяє перейти на сторінку профілю чи вийти з акаунту - Основні плейлисти - Кнопка для навігації на сторінку пошуку Вид доступу: захищена від неавторизованих користувачів
sign-up.tsx	Сторінка реєстрації, що вимагає від користувача ім’я, електронну пошту та пароль. Вид доступу: відкрита тільки для неавторизованих користувачів
login.tsx	Сторінка логіну, для аутентифікації

	потрібно ввести електронну пошту та пароль. Вид доступу: відкрита тільки для неавторизованих користувачів
tracks/search.tsx	Доступна за шляхом: /tracks/search. Сторінка пошуку пісень за назвою Вид доступу: захищена від неавторизованих користувачів
tracks/[id].tsx	Доступна за шляхом: /tracks/[id].tsx. Сторінка, що відображає детальну інформацію про пісню та містить секцію з коментарями. Вид доступу: захищена від неавторизованих користувачів
profile.tsx	Сторінка профілю. Містить в собі функціонал для оновлення аватару та інформації про користувача. Вид доступу: захищена від неавторизованих користувачів

Для більш детального розуміння того, як відбувається взаємодія користувача з системою та, безпосередньо, процесу генерації рекомендацій, є сенс представити графи переходів для цього (рис. 3.3):

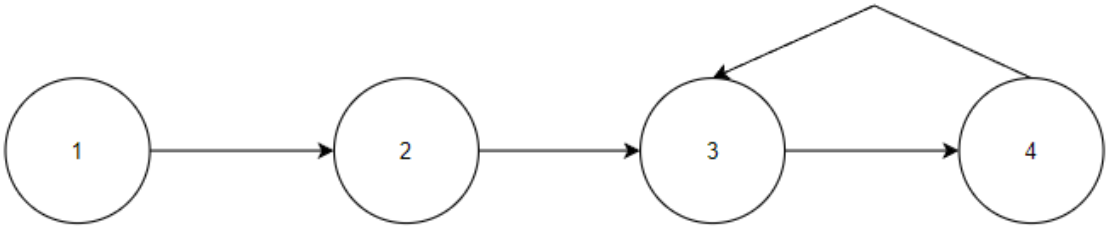

Рисунок 3.3 – Граф переходів між сторінками на клієнті, що відображає процес формування рекомендацій для нового користувача

В табл. 3.3 наведено опис графу переходів, що представляє собою назви основних сторінок, що має відвідати користувач для того, щоб отримати рекомендації. Варто зазначити, що розглядається випадок для нових користувачів, що не мають історії прослуховування. Для вже активних слухачів більшість дій мають сенс для отримання нових рекомендацій.

Таблиця 3.3. Опис графу переходів між сторінками на клієнті при формуванні рекомендацій для нового користувача

Номер	Опис
1	Сторінка sign-up.tsx, яка відображає форму реєстрації.
2	Сторінка login.tsx, яка відображає форму авторизації.
3	Сторінка dashboard.tsx, яка відображає пусті плейлісти з прослуховуваннями та рекомендаціями.

4	Сторінка search.tsx, яка відображає текстове поле для пошуку пісень та результати пошуку.

3.3 Керівництво користувача музичного застосунку

При першому завантаженні застосунку користувачу буде запропоновано створити новий обліковий запис, після якого він одразу потрапляє на сторінку з логіном (рис. 3.5).

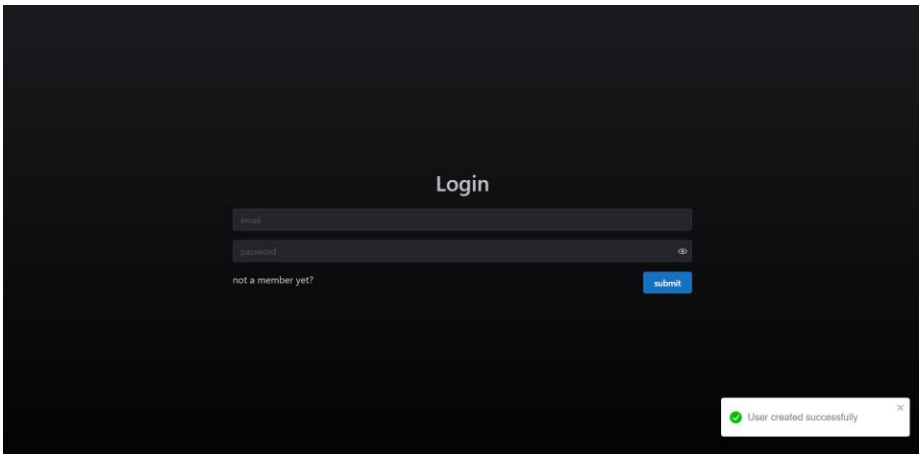

Рисунок 3.5 – Успішна реєстрація нового облікового запису

- Вхід – після реєстрації з’являється можливість увійти в систему, натиснувши кнопку "Submit". Але потрібно ввести електронну пошту та пароль, який який був вказаний при реєстрації. Після успішного входу користувач отримує доступ до усіх функцій музичного застосунку (рис. 3.6).

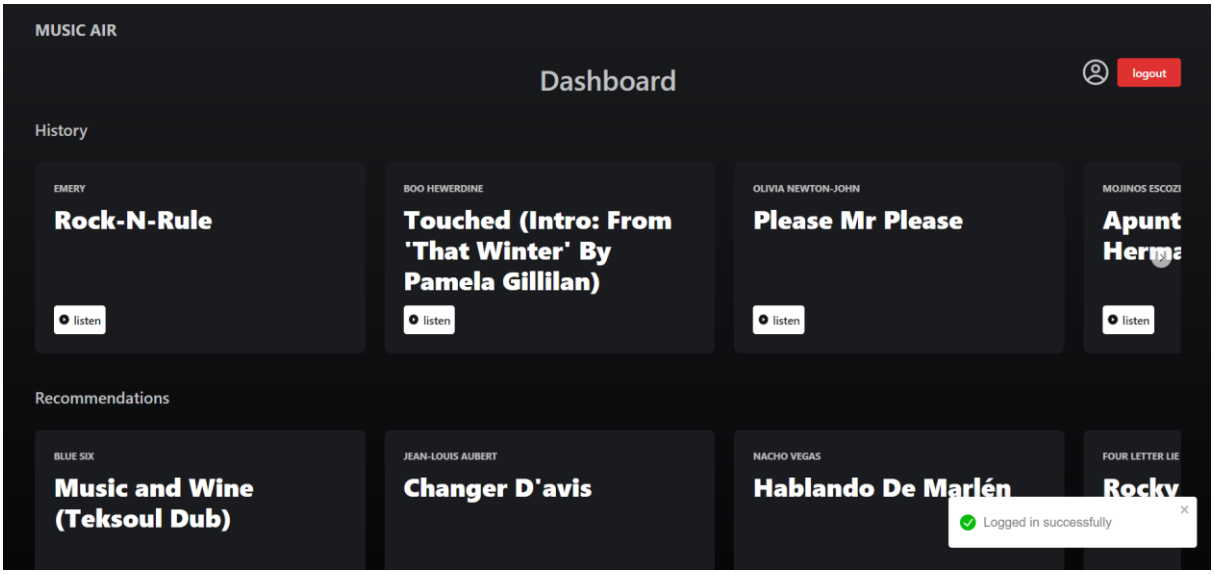

Рисунок 3.6 – Сторінка з головною панеллю застосунку

- Пошук та прослуховування музики – у застосунку користувач може прослуховувати музику з різних жанрів та виконавців. Для цього достатньо знайти потрібну пісню та натисніть на кнопку "listen". Меню плеєра дозволяє ставити на паузу, перемотувати треки та налаштувати гучність з допомогою відповідних кнопок (рис. 3.7).

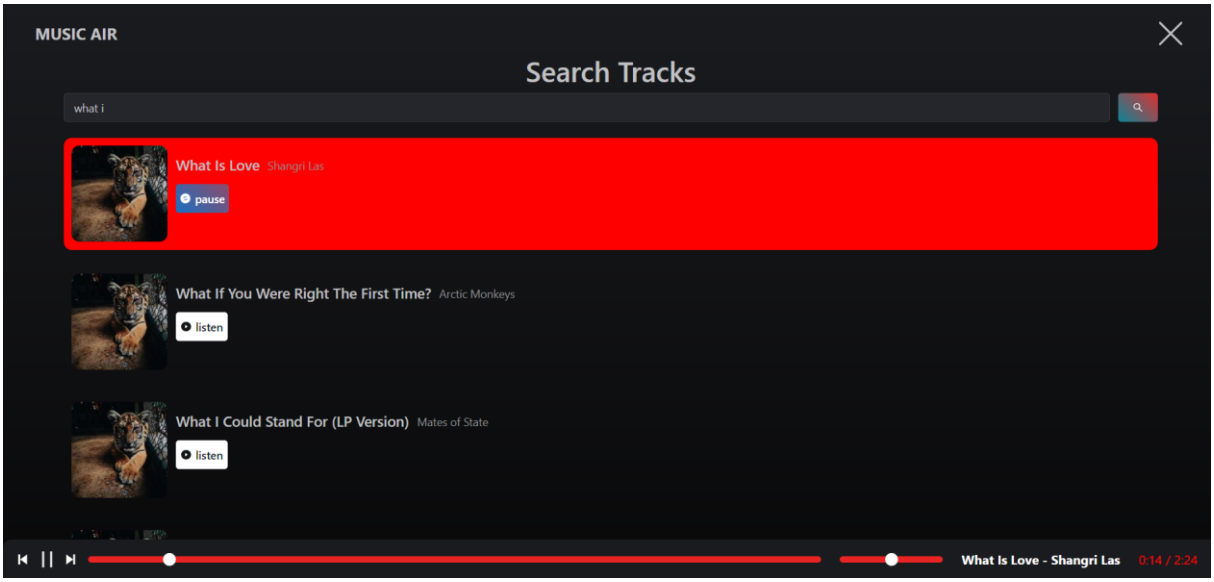

Рисунок 3.7 – Музичний плеєр

- Додавання коментарів – у застосунку реалізовано можливість додавати коментарі до будь-якої пісні. Для цього достатньо перейти

до відповідної сторінки пісні, додати у поле для введення коментар та натиснути на кнопку "Post" (рис. 3.9).

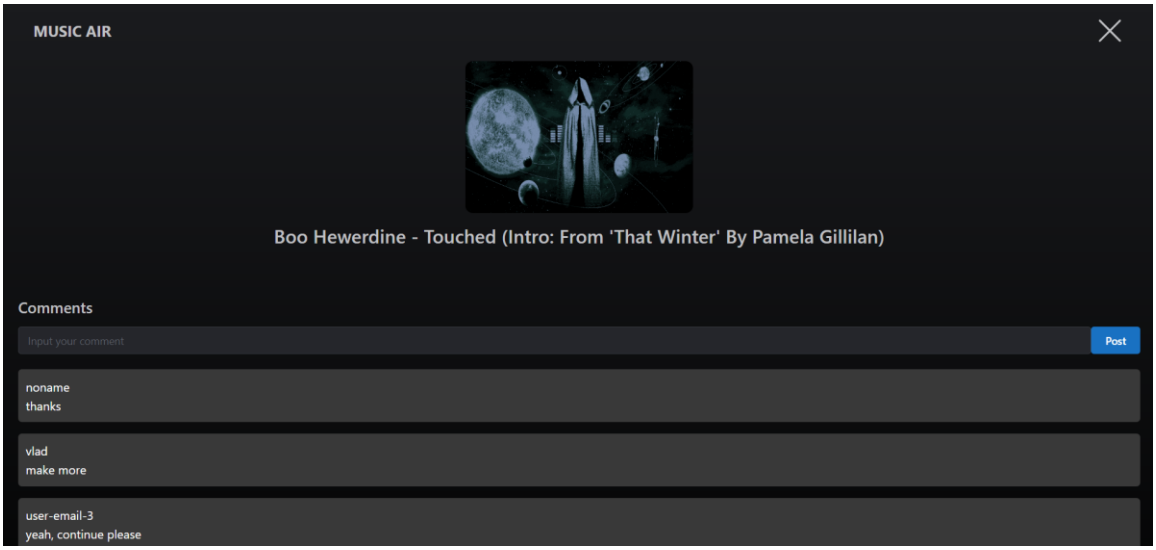

Рисунок 3.8 – Інформація про пісню та секція з коментарями

- Оновлення профілю – кожен зареєстрований користувач може оновити свої особисті дані та інформацію про профіль. Для цього потрібно з головного меню (dashboard) перейти у профіль, натиснувши на іконку користувача справа від кнопки для виходу з системи. Далі можна відредагувати свої дані за допомогою відповідних полів та зберегти зміни (рис. 3.9).

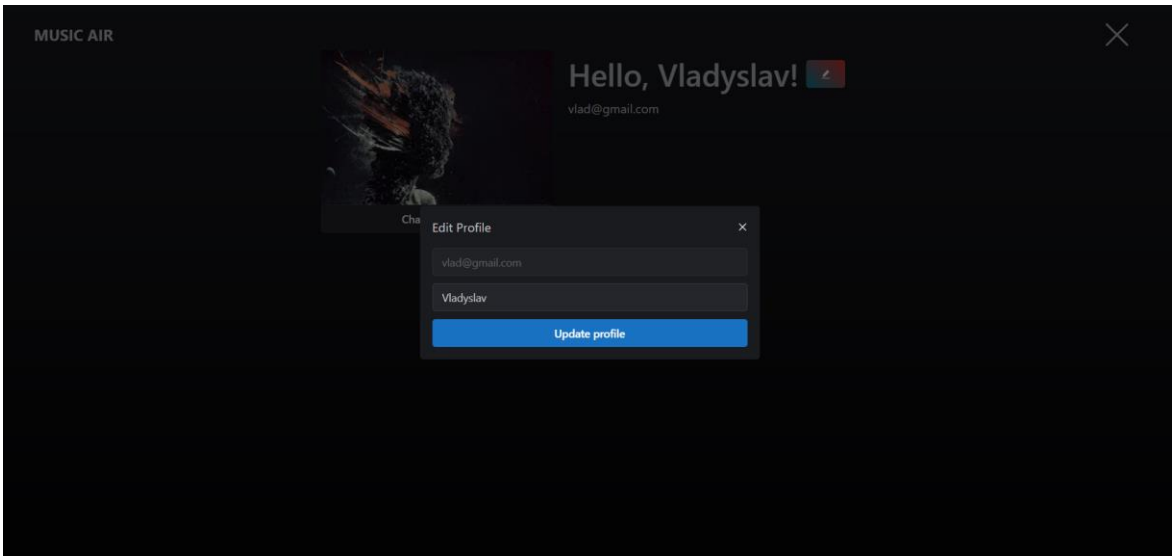

Рисунок 3.9 – Редагування профілю

3.4 Огляд процесу тестування музичного застосунку з системою рекомендацій

Після налаштування взаємодії клієнта та сервера, необхідно провести E2E тестування для перевірки правильності роботи застосунку. Для цього є сенс створити табл. 3.4, що відображає базові сценарії та отримані результати.

Таблиця 3.4. Тестові сценарії та результати

Назва	Вхідні дані	Очікуваний результат	Отриманий результат
Відображення плейлиста з історією прослуховувань			
Відсутня історія прослуховувань	Авторизаційний токен з даними про користувача	Порожній масив	[]
Історія прослуховувань налічує декілька пісень	Авторизаційний токен з даними про користувача	Масив об’єктів, кожен з яких містить інформацію, про прослухану пісню	[{id: 54420, ...}, {id: 85202, ...}, {id: 53068, …}, {id: 46873, …} …]
Формування плейлисту з рекомендаціями			
Відсутня історія прослуховувань або налічує < 3 пісень	[] / [54420] / [54420, 85202]	Запит на отримання рекомендацій не має відправлятися	Запит ігнорується за відсутності необхідної кількості

			вхідних даних
Історія прослуховувань налічує > 3 пісень	[54420, 85202, 53068]	Масив рекомендацій, без повторів, що містить основні відомості про пісні	[{id: 10441, song: "Music and Wine (Teksoul Dub)"...}, {id: 64295, song: "Changer D'avis", … }, {id: 81680, song": "Hablando De Marlén", …}, {id: 124679, song: "Rocky Loves Emily (Album Version)", …}, {id: 41551, song: "The Banks Workout", … }, {id: 76781, song: "Scandal", …}]
Пошук пісень			
Пошук існуючих в БД пісень	what is love	Масив пісень, що мають назву “what is love”	[{id: 123414, song: "What Is Love 2K9", …}, {id: 51766, song:

			"What Is Love (12\" Mix)", …}, {id: 68459, song: "What Is Love", …}, {id: 118692, song: "What Is Love?", …}, {id: 24922, song: "What Is Love", …}]
Пошук пісень, що відсутні в БД	unknown_song_title	Порожній масив	[]
Оновлення даних профілю			
Оновлення зображення користувача	Клік на кнопку “Change avatar” та вибір image/format файлу	Зображення профілю має змінитися на нове	Зображення профілю змінюється на нове
Оновлення ім’я	Клік на іконку для внесення змін, введення нового імені в полі для введення та клік	Має відбутися оновлення інформації	Ім’я користувача оновлюється на відображається миттєво

	на “Update profile”		
Прослуховування пісні			
Перемикання пісень	Клік на кнопку для завантаження наступної пісні	На сторінці пошуку має завантажити наступну пісню, що йде в списку результатів	Завантажується наступна пісня, в істрою прослуховувань попередня не записується
Пауза на плеєрі	Клік на кнопку для паузи	Має зупинити пісню, але має залишатися можливість продовжити прослуховуванн я	Пісня зупиняється, при натисканні на кнопку для відновлення, продовжує грати
Перемотування пісні за допомогою контролеру	Перемотування пісні	Пісня має перемотуватися відповідно до шкали	Пісня перемотується відповідно до шкали

Окремо необхідно провести тестування роботи нейромережі. Перед перевіркою на тестувальній вибірці, потрібно налаштувати метрики для тренувальної. Датасет, що використовується для навчання - Million Song Dataset[17]. Для того, щоб отримати метрики, достатньо використати FactorizedTopK, що представлений в пакеті TensorFlow. Ця метрика приймає на вхід один обов’язковий аргумент - список кандидатів (рис. 3.10):

```
candidates = dataset.batch(128).map(lambda x: candidate_model(x['user']))
metrics = tfrs.metrics.FactorizedTopK(candidates=candidates)
```

Рисунок 3.10 – Програмний код для налаштування метрики

Для тренування моделі також використовується й інший важливий компонент - втрати. Оскільки в алгоритмі використовується Retrieval Task, то достатньо тільки вказати метрику - втрати будуть включені до обчислення автоматично (рис. 3.11).

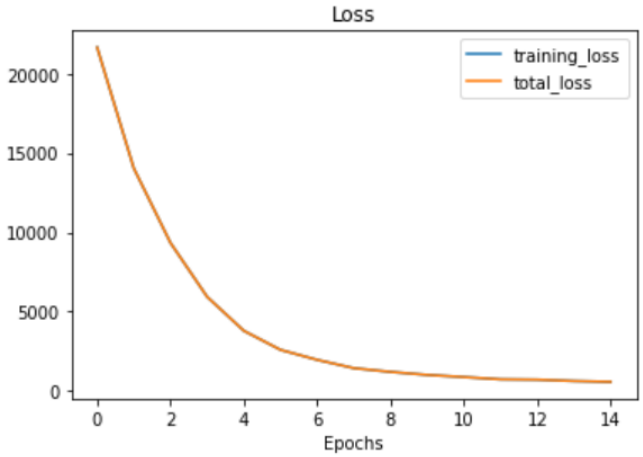

Рисунок 3.11 – Втрати під час навчання

Додатково можна побачити графік точності роботи для топ X елементів, що будується на основі FactorizedTopK метрики, описаної вище (рис. 3.12).

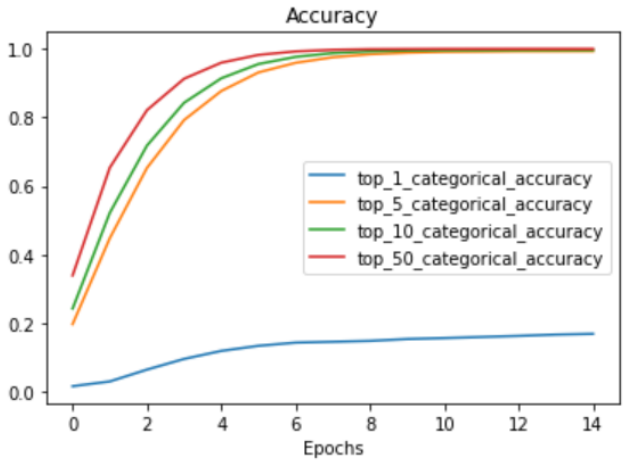

Рисунок 3.12 – Точність моделі під час тренування

В табл. 3.4 описані вхідні дані, взяті з реального застосунку та результати взаємодії всіх компонентів застосунку.

Таблиця 3.4. Тестові сценарії роботи нейромережі та результати

Вхідні дані	Отриманий результат
[1, 2, 3]	[{id: 2, song: "The Bells Sketch"…}, {id: 20834, song: "Una Rafaga De Amor"…}, {id: 59532, song: "Scorpio"…}]
[54420, 85202, 53068]	[{id: 10441, song: "Music and Wine (Teksoul Dub)"...}, {id: 64295, song: "Changer D'avis", … }, {id: 81680, song": "Hablando De Marlén", …}, {id: 124679, song: "Rocky Loves Emily (Album Version)", …}, {id: 41551, song: "The Banks Workout", … }, {id: 76781, song: "Scandal", …}]
[10441, 10441, 10441]	[{id: 11240, song: "Coisa Da Antiga", ...}, {id: 63223, song: "Lago En El Cielo", ...}, {id: 10441, song: "Music and Wine (Teksoul Dub)", ...}, {id: 65382, song: "String Quartet No. 13 in A minor D.804 (\"Rosamunde\"): IV. Allegro moderato", ...}, {id: 41442, song: "Matching Weight", ...}, {id: 83827, song: "Represent", ...}, {id: 44491, song: "Juhlat", ...}, {id: 41797, song: "Love Pig", ...}, {id: 57901, song: "Vuelvo Al Sur", ...},

	{id: 52585, song: "Mustang Sally", ...}, {id: 119339, song: "Cali Nights", ...}, {id: 43932, song: "Namaste", ...}]
[61316, 6987, 57133]	[{id: 61316, song: "Welcome 2 Detroit", ...}, {id: 6987, song: "Sarasa", ...}, {id: 57133, song: "Widow Weeds", ...}, {id: 70960, song: "Everything Has Its Point", ...}, {id: 70741, song: "Number Three_ Never Forgot (Album Version)", ...}]

Результати роботи можна проаналізувати наступним чином, якщо в результаті ми отримуємо в списку рекомендацій пісні, що дублюють вхідні дані, це означає, що вектор вбудування для User1 (користувач, для якого формуються рекомендації) побудований максимально близько до вектора вбудування User2 (користувач з найбільш схожими вподобаннями).

Майже в усіх випадках ми отримали вихідний вектор, що містить одну чи більше композицій, що містяться у вхідному векторі, а це означає, що вподобання користувача, для якого формуються рекомендації, максимально подібні до вподобань знайденого в БД користувача.

3.5 Висновки до третього розділу

У даному розділі дипломної роботи було розглянуто створення музичного застосунку, який базується на використанні різних технологій, таких як Nest JS, React, Next JS та Tensorflow. Було описано архітектуру системи та проаналізовані основні складові її реалізації.

Окрему увагу було приділено інтелектуальній моделі, що використовує нейромережі для створення колаборативної фільтрації, яка

дає можливість рекомендувати користувачам музичні треки на основі їхніх попередніх уподобань та інтересів.

Отже, можна зробити висновок, що розроблений музичний застосунок є складною технічною системою, яка базується на використанні різних технологій та інструментів. Використання інтелектуальної моделі, що ґрунтується на нейромережах, дозволяє покращити якість рекомендацій та забезпечити більш точне врахування особистих уподобань користувачів. Результати даної роботи можуть бути корисні для подальшого розвитку музичних застосунків та дослідження в галузі машинного навчання.

ВИСНОВКИ

В даній дипломній роботі було створено повноцінний веб-застосунок з рекомендаційною системою, що використовує нейронну мережу. Рекомендаційні системи стають все більш важливими для компаній, оскільки допомагають полегшити пошук та сприяють розвитку продуктів та сервісів. У роботі були розглянуті різні види рекомендаційних систем та проведено аналіз їх переваг та недоліків.

Основним алгоритмом, який було реалізовано, є колаборативне фільтрування з використанням нейромережі за допомогою фреймворку Tensorflow. Для адаптації моделі в реальному середовищі було створено клієнтську та серверну частини, де серверна частина напряму взаємодіє з інтелектуальною моделлю та базою даних. Документація API та приклади запитів були надані для зручного користування системою.

Результати тестування показали, що нейромережа ефективно працює та забезпечує достатню кількість рекомендацій на вхідних даних. Вона досить швидко та ефективно працює й для нових користувачів. Однак, для подальшого вдосконалення системи можна розглянути використання моделей, що фільтрують за змістом, для покращення точності та надійності прогнозів, а також для подальшого врахування того, що система має динамічно реагувати на додавання нових пісень.

У цілому, розроблений веб-застосунок з рекомендаційною системою є значним кроком у поліпшенні пошуку та забезпеченні персоналізованого контенту для користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 1. Stafford SA. Music in the Digital Age: The Emergence of Digital Music and Its Repercussions on the Music Industry. The Elon Journal of Undergraduate Research in Communications Vol. 1 No. 2. 2010.
2. Aggarwal CC. Recommender Systems: The Textbook. 1st ed.: Springer; 2016.
3. Oord Avd, Dieleman S, Schrauwen B. Deep content-based music recommendation. Advances in Neural Information Processing Systems 26 (NIPS 2013). 2013.
4. O'Bryant J. A survey of music recommendation and possible improvements. In ; 2017
5. Гайна, Г.А. (2005). Основи проектування баз даних: Навчальний посібник. Київ: Київський національний університет будівництва і архітектури. с. 204.
6. Chen, Peter Pin-Shan (1976). The entity-relationship model—toward a unified view of data. ACM Transactions on Database Systems 1 (1). с. 9–36.
7. Документація фреймворку для розробки серверного застосунку Nest JS. [Електронний ресурс]. Nest JS документація. Режим доступу до ресурсу – https://nestjs.com/
8. Документація фреймворку для розробки клієнтської Next JS. [Електронний ресурс]. Next JS документація. Режим доступу до ресурсу – https://nextjs.org/
9. Документація по застосуванню retrieval моделі. [Електронний ресурс]. Tensorflow документація. Режим доступу до ресурсу – https://www.tensorflow.org/recommenders/examples/basic_retrieval

10. Документація по застосуванню basic ranking моделі. [Електронний ресурс]. Tensorflow документація. Режим доступу до ресурсу – https://www.tensorflow.org/recommenders/examples/basic_ranking

11. D. Goldberg et al., “Using Collaborative Filtering to Weave an Information Tapestry,” Comm. ACM, vol. 35, 1992, pp. 61-70.

12. B.M. Sarwar et al., “Application of Dimensionality Reduction in Recommender System—A Case Study,” Proc. KDD Workshop on Web Mining for e-Commerce: Challenges and Opportunities (WebKDD), ACM Press, 2000.

13. Офіційний веб сайт Spotify. [Електронний ресурс]. Режим доступу до ресурсу – https://open.spotify.com/

14. Офіційний веб сайт ITunes. [Електронний ресурс]. Режим доступу до ресурсу – https://www.apple.com/itunes/

15. Офіційний веб сайт YouTube Music. [Електронний ресурс]. Режим доступу до ресурсу – https://music.youtube.com/

16. Офіційний веб сайт Pandora Music. [Електронний ресурс]. Режим доступу до ресурсу – https://www.pandora.com/

17. Веб Сайт для завантаження датасету - Million Song Dataset. [Електронний ресурс]. Режим доступу до ресурсу – http://millionsongdataset.com/tasteprofile/

Додаток А

Програмний код для створення інтелектуальної моделі

```
import csv
import numpy as np
import tensorflow as tf
import tensorflow_recommenders as tfrs
n_songs_in = 3

base_dir = '/data/music-recommender'
weights_dir = base_dir +'/{}-weights'
index_model_dir = base_dir + '/model'
tfjs_model_dir = base_dir + '/tfjs-model'
users_songs = {}
with open('./datasets/clean/users_songs.csv') as file:
    reader = csv.reader(file)
    next(reader)
    for line in reader:
        user = int(line[0])
        song = int(line[1])
        try:
            users_songs[user].append(song)
        except:
            users_songs[user] = [song]
user_ids = {}
with open('./datasets/clean/user_ids.csv') as file:
    reader = csv.reader(file)
    next(reader)
    for line in reader:
        user_ids[int(line[1])] = line[0]
songs_ids = {}
with open('./datasets/clean/song_ids.csv') as file:
    reader = csv.reader(file)
    next(reader)
    for line in reader:
        songs_ids[int(line[1])] = line[0]
n_users = len(user_ids)
n_songs = len(songs_ids)

print(f'Dataset contains: {n_users} users, {n_songs} songs')
dataset = {'tracks': [], 'user': []}

for user, tracks in users_songs.items():
    selected_tracks = np.random.choice(tracks, n_songs_in)
    dataset['user'].append(user)
    dataset['tracks'].append(selected_tracks)
permutation = np.random.permutation(len(dataset['tracks']))
```

```
dataset = {
   'tracks': np.asarray(dataset['tracks'])[permutation],
   'user': np.asarray(dataset['user'])[permutation]
}
dataset = tf.data.Dataset.from_tensor_slices(dataset)

print('len_dataset', permutation.shape[0])

for row in dataset.take(1):
   print(row)
n_embedding_dimensions = 24

# create two separate models: QUERY and CANDIDATE

# QUERY (tracks)

tracks_input = tf.keras.Input(shape=(n_songs_in))
tracks_embedded = tf.keras.layers.Embedding(n_songs + 1, n_embedding_dimensions)(tracks_input)
tracks_embedded_avg = tf.keras.layers.AveragePooling1D(pool_size=3)(tracks_embedded)
query = tf.keras.layers.Flatten()(tracks_embedded_avg)
query_model = tf.keras.Model(inputs=tracks_input, outputs=query)

# CANDIDATE (user)

user_input = tf.keras.Input(shape=(1))
user_embedded = tf.keras.layers.Embedding(n_users+1, n_embedding_dimensions)(user_input)
candidate = tf.keras.layers.Flatten()(user_embedded)
candidate_model = tf.keras.Model(inputs=user_input, outputs=candidate)
candidates = dataset.batch(128).map(lambda x: candidate_model(x['user']))
metrics = tfrs.metrics.FactorizedTopK(candidates=candidates)
task = tfrs.tasks.Retrieval(metrics=metrics)
class Model(tfrs.Model):
   def __init__(self, query_model, candidate_model):
      super().__init__()
      self._query_model = query_model
      self._candidate_model = candidate_model
      self._task = task

   def compute_loss(self, features, training=False):
      query_embedding = self._query_model(features['tracks'])
      candidate_embedding = self._candidate_model(features['user'])
      return self._task(query_embedding, candidate_embedding)
## COMPILE AND TRAIN MODEL
model = Model(query_model, candidate_model)
# load model weights - this is to resume training
# model._query_model.load_weights(weights_dir.format('query'))
# model._candidate_model.load_weights(weights_dir.format('candidate'))
```

```
model.compile(optimizer=tf.keras.optimizers.Adagrad(learning_rate=0.1))
model.fit(dataset.repeat().shuffle(300_000).batch(4096), steps_per_epoch=25, epochs=10, verbose=1)
model._query_model.save_weights('./data/model/query-weights')
model._candidate_model.save_weights('./data/model/candidate-weights')
index = tfrs.layers.factorized_top_k.BruteForce(model._query_model)
index.index_from_dataset(
    tf.data.Dataset.zip((
      dataset.map(lambda x: x['user']).batch(100),
      dataset.batch(100).map(lambda x: model._candidate_model(x['user']))
    ))
)
for features in dataset.shuffle(2000).batch(1).take(1):
    print('tracks', features['tracks'])
    scores, users = index(features['tracks'])
    print('recommended users', users)
index.save('./data/model')
import subprocess
cmd = [
    'tensorflowjs_converter',
    '--input_format=tf_saved_model',
    '--output_format=tfjs_graph_model',
    './data/model',
    './data/tfjs-model'
]
subprocess.run(cmd)
```