

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
МЕТОДІВ ПІДБОРУ МУЗИКИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
MUSIC SELECTION METHODS

Виконав: студент(ка) 2 курсу, групи КНТ-111м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Хабло Д.О.

(прізвище та ініціали)

Керівник Корнієнко С.К.

(прізвище та ініціали)

Рецензент Терещенко Е.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Хабла Дениса Олексійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів підбору музики

Research and Software Implementation of Music Selection Methods

керівник проєкту (роботи) Корнієнко Сергій Костянтинович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “4” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) грудень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Корнієнко С.К., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 7 ” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Хабло Д.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Корнієнко С.К.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 87 с., 26 рис., 3 табл., 3 дод., 16 джерел.

МУЗИЧНА КОМПОЗИЦІЯ, ЕМОЦІЇ, ПІДБІР МУЗИКИ, МАШИННЕ НАВЧАННЯ, ВЕБЗАСТОСУНОК.

Об'єкт роботи – процес пошуку нової музики. Предмет роботи – методи підбору музики. Мета роботи – розробити програмне забезпечення підбору музики для забезпечення потреб слухачів щодо пошуку нової музики.

Матеріали, методи та технічні засоби: мова програмування Python, фреймворк веброзробки Django, бібліотека Keras.

Результати. Основний результат роботи виражений в її науковій новизні, що полягає в запропонованому методі підбору музики, заснованому на розбитті даного процесу на етапи виділення емоцій за голосом людини та підбору музичних композицій за виділеними емоціями, що дозволяє формувати перелік композицій для прослуховування з різним ступенем новизни в залежності від типу емоцій.

Висновки. Розроблений програмний продукт призначений для відтворення музичних композицій з обмеженого каталогу з наданням можливостей підбору музики з цього каталогу для розширення простору її відтворення. Програма надає інформацію про статистику прослуховування музики за країнами, про найкращі композиції, виділені за статистикою відтворення музики слухачами, дозволяє формувати власні переліки композицій, альбомів у різному вигляді. Програма може використовуватися авторизованими і неавторизованими користувачами.

Галузь використання – сервіси програвання музики онлайн для окремих компаній-правовласників.

ABSTRACT

Explanatory note to the qualifying work of the master: 87 pages, 26 figures, 3 tables, 3 appendixes, 16 sources.

MUSIC COMPOSITION, EMOTIONS, MUSIC SELECTION, MACHINE LEARNING, WEB APPLICATION.

The object of the work is the process of finding new music. Subject of work is methods of music selection. The purpose of the work is to develop music selection software to meet the needs of students to find new music.

Materials, methods and technical means: Python programming language, Django web development framework, Keras library.

Results. The main result of the work is expressed in its scientific novelty, which is in the proposed method of music selection, based on the division of this process into stages of selection of emotions by human voice and selection of musical compositions by selected emotions, which enables to form a list according to the type of emotions.

Conclusions. The developed software product is designed to play music songs from a limited catalog with the ability to select music from this directory to expand the space of its playback. The program provides information on the statistics of listening to music by country, the best songs, selected by the statistics of music playback listeners, allows users to create their own lists of songs, albums in various forms. The program can be used by authorized and unauthorized users.

Field of use is online music playback services for separate companies of rights holders.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Аналіз існуючих програмних засобів потокового відтворення музики	11
1.2 Висновки за розділом 1	16
2 Матеріали і методи.....	17
2.1 Методи класифікації даних для підбору музики	17
2.2 Метод підбору музики	19
2.3 Висновки за розділом 2	21
3 Проєктування програмного забезпечення	22
3.1 Вибір засобів веброзробки	22
3.2 Функціональні вимоги до програмного забезпечення	22
3.3 Проєктування бази даних	26
3.4 Структура програмного забезпечення	33
3.5 Висновки за розділом 3	34
4 Основні рішення щодо реалізації компонентів системи.....	35
4.1 Алгоритм функціонування програми.....	35
4.2 Опис основних розроблених класів	37
4.3 Висновки за розділом 4	47
5 Експлуатація, тестування та експериментальне дослідження програми	48
5.1 Призначення програми	48
5.2 Умови виконання програми	48
5.3 Робота з програмою	48
5.4 Експериментальне дослідження	51
5.5 Висновки за розділом 5	52
Висновки	53
Перелік джерел посилання	54
Додаток А Технічне завдання	56

Додаток Б Текст програми	60
Додаток В Слайди презентації.....	80

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Прослуховування музики є однією з базових розваг у сучасному світі. Тому важливість будь-яких засобів, які зможуть вплинути на зручність виконання процесів у даній сфері, важко переоцінити.

Музична індустрія поступово відновлювалась з 2021 року, що відбувалось зі збільшення доходів від продажів компакт-дисків і вінілових платівок, які в 2020 році постраждали від примусового закриття роздрібних магазинів через пандемію, і зі збільшенням потокових сервісів, які в цей період взагалі стали основним джерелом доходу для артистів та звукозаписних компаній на тлі пандемії коронавірусу [1].

Промислова торгова група Американської асоціації індустрії звукозапису повідомила, що доходи від роздрібної торгівлі музичними продуктами в США зросли на 27% у першому півріччі 2021 року до 7,1 мільярда доларів проти 5,6 мільярдів доларів минулого року: більшість музичних доходів за останній рік надходило від потокового передавання (84%), за ним йдуть продажі фізичних компакт-дисків і вінілових платівок (10%), цифрові завантаження (5%) і ліцензії на авторські права на музику (2%) [1]-[3].

Отож, зважаючи на активний розвиток потокових сервісів протягом останнього десятиліття, активне зростання протягом останнього року, слід вказати на актуальність даної сфери. Одночасно з наданням можливості відтворення музики, що звичайно є базовим завданням, потрібно забезпечити можливості знаходження нової музики, використовуючи інтелектуальні засоби для цього. Це дозволяє користувачу постійно знаходитися в потоці музики, що відповідно сприяє збільшенню прибутків галузі.

Розвиток саме інтелектуальних засобів для пошуку музичних композицій слухачами є дуже актуальним завданням на даний момент. Сучасні потокові сервіси вирішують це завдання по-різному, а універсального засобу звичайно не існує. Тому конкуренція призводить до

необхідності постійного оновлення таких засобів, щоб відповідати вимогам конкурентного ринку.

Метою даної кваліфікаційної роботи було визначено розроблення програмного забезпечення (ПЗ) підбору музики для забезпечення потреб слухачів щодо пошуку нової музики.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз існуючих програмних засобів потокового відтворення музики

Spotify (рис. 1.1) є одним з найвідоміших сервісів для потокового відтворення музики [4].

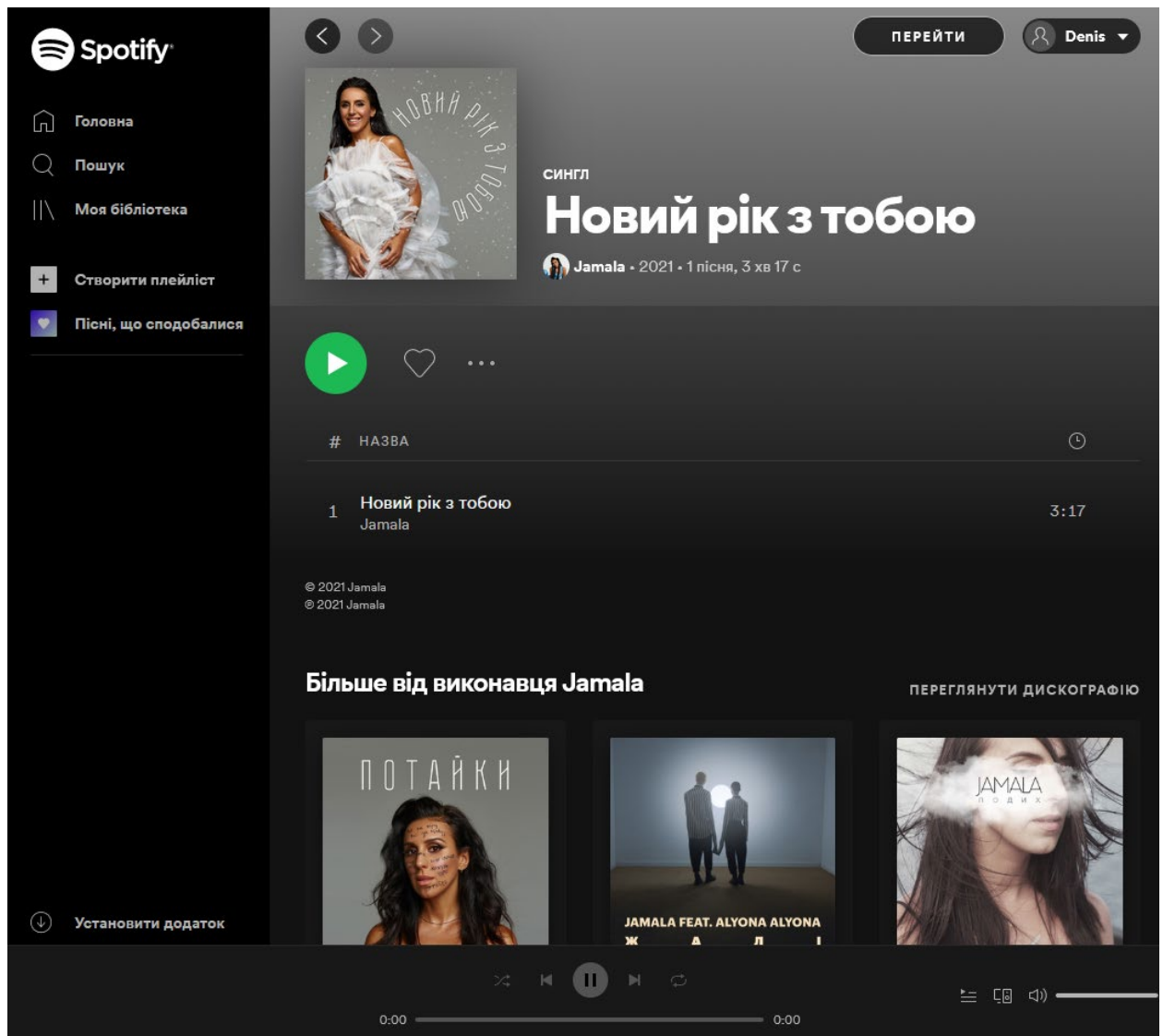


Рисунок 1.1 – Перегляд сторінки виконавця в Spotify

Spotify є піонером у потоковій передачі музики і, можливо, найвідомішим сервісом, відносно якого можна відзначити наступне:

- пропонує низку підібраних переліків музичних композицій, включаючи найвідоміший з них, – перелік відтворення Discover Weekly, при цьому постійно впроваджуються нові;
- сервіс Spotify дозволяє транслювати Spotify Connect на різні пристрої через Wi-Fi;
- надає доступ до понад 70 мільйонів пісень, 3.2 мільйонів подкастів незалежно від того, чи перебуває користувач на рівні безкоштовного чи преміум-класу;
- надає можливість легко створювати власні переліки відтворення та синхронізувати їх для прослуховування в автономному режимі;
- дозволяє стежити за виконавцями та отримувати сповіщення, коли вони випускають нову музику або оголошують про майбутнє шоу;
- нав'язливість реклами у безкоштовному сервісі;
- не можна слухати конкретні пісні на безкоштовному рівні, лише деякий перелік на основі запитаної музики;
- відсутність можливості відтворення музики в якості без втрат [5]-[7].

Найновішою платформою серед найвідоміших платформ потокового відтворення музики є YouTube Music (рис. 1.2) [8].

YouTube Music має інтегровані засоби, які дозволяють не тільки слухати музику, але і переглядати відео деяких композицій.

Безкоштовний рівень дозволяє отримати доступ до всього вмісту, але відтворення переривається рекламою, дані не можуть бути завантажені, і для того, щоб почути композиції, потрібно, щоб екран був увімкнений, перехід до рівня Premium знімає ці обмеження, при цьому значним недоліком сервісу є те, що потокова передача обмежена лише 256 кбіт/с, навіть при використанні платних рівнів [5]-[6].

Загалом наявність платних підписок і обмежена якість відтворення є недоліками або обмеженнями обох сервісів.

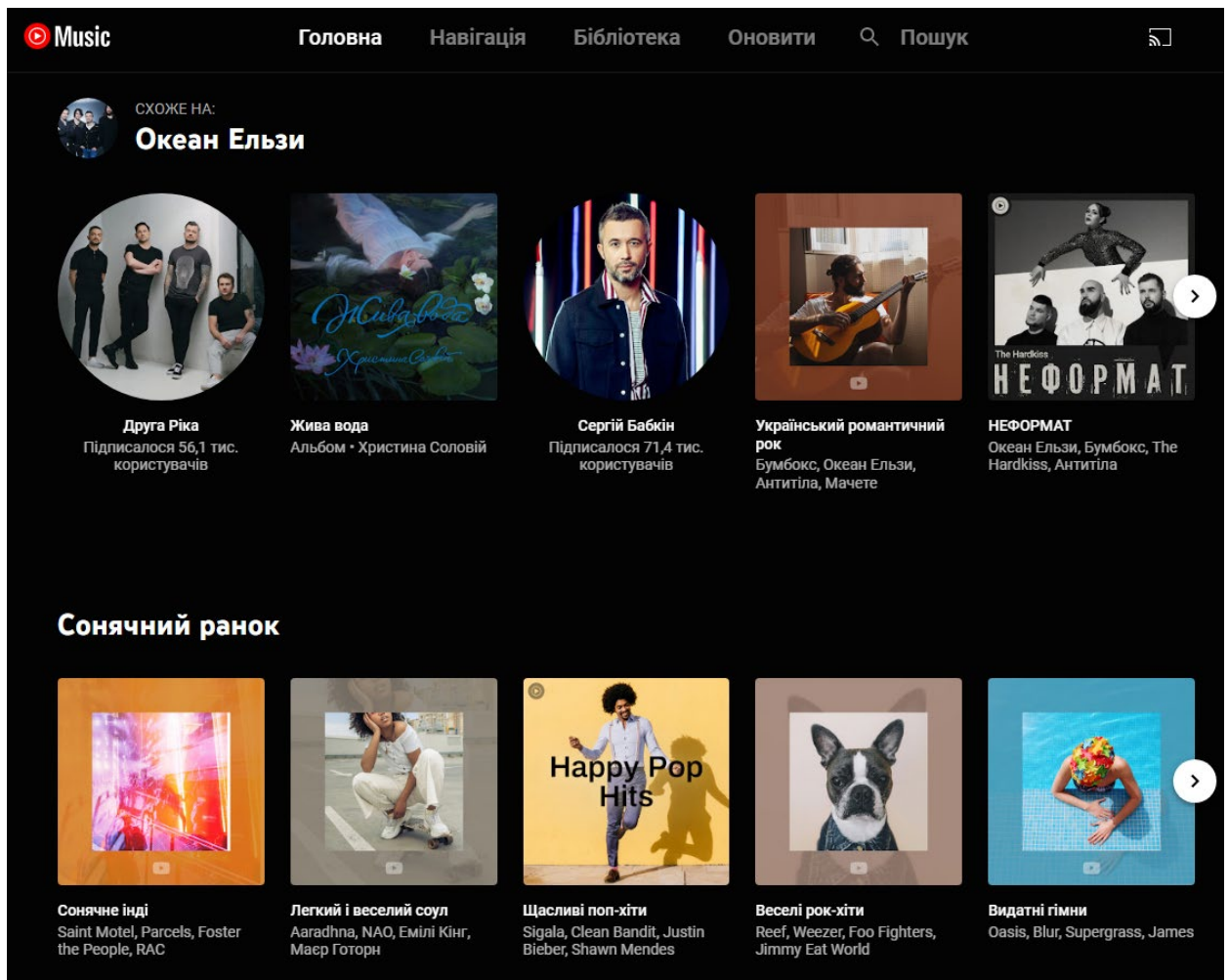


Рисунок 1.2 – Переліки музичних композицій в YouTube Music

Окрім даних сервісів існують також програми, які підтримують окремо від них процес пошуку нової музики, в тому числі процес формування переліків композицій, що відповідають певним критеріям.

У якості прикладу такого ПЗ використаємо Chosic (рис. 1.3) [9].

Chosic надає можливість формувати переліки композицій на основі різних музичних сервісів, надаючи наступні можливості:

- пошук подібних музичних композицій (рис. 1.3): потрібно задати назву композиції, можна уточнити кількість композицій у переліку, а також певні її характеристики (танцювальність, емоційну забарвленість тощо);
- пошук музичних композицій за жанром;
- пошук виконавців, подібних на заданого;
- перегляд статистики прослуховувань на Spotify;

- формування переліків композицій випадковим чином;
- формування переліку нових композицій тощо.

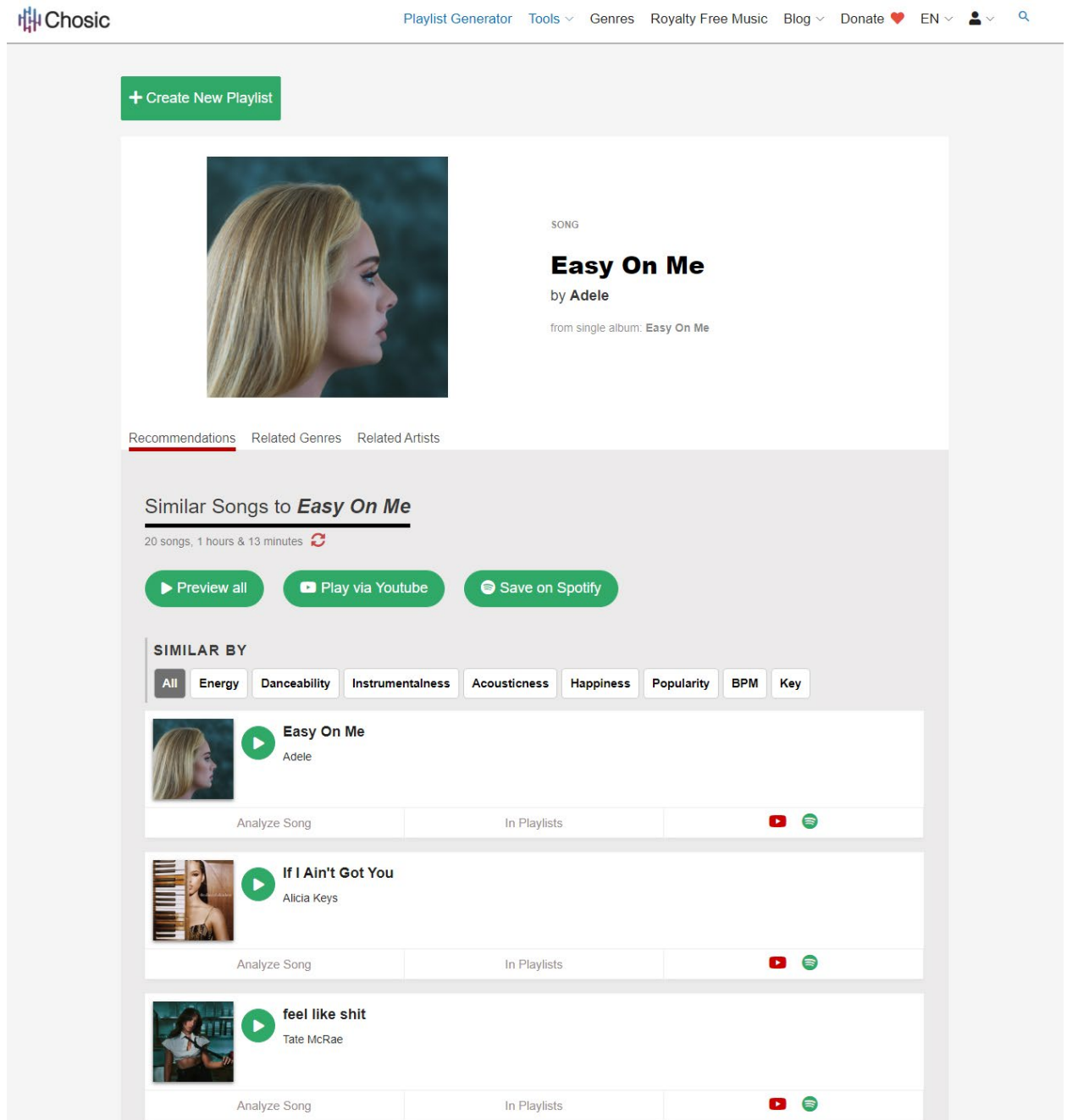


Рисунок 1.3 – Перелік подібних музичних композицій у Chosic

Серед наявних програм виділено Spotify і YouTube Music для проведення порівняння програми, яку потрібно розробити, з аналогами.

Розробка, що ведеться в кваліфікаційній роботі, не буде мати платних версій, хоча в подальшому це може за потреби бути змінено. Сферою

використання програми може бути підтримка відтворення музичних композицій з певного обмеженого каталогу. Це може бути один з продюсерських центрів, який бажає виділити свій каталог музичних композицій серед загальнодоступних, не надаючи власний каталог у розпорядження великих сервісів потокового відтворення музики. Такий каталог потребує засобів пошуку нових композицій, для того щоб привернути увагу користувачів серед усіх наявних композицій. Це дозволить створити певний нішевий засіб, але при тому залишатися в загальній тенденції автоматичного знаходження музики для слухача, що важливо в сучасному світі.

Результати аналізу предметної області у вигляді порівняння аналогів з розробкою, яка виконується, представлені в табл. 1.1.

Таблиця 1.1 – Результати аналізу предметної області

Критерій	Spotify	YouTube Music	Розробка
Платна версія	Є	Є	Немає, але можливо в подальшому
Відтворення нішевого каталогу музики	Усі власники прав в одному каталозі	Усі власники прав в одному каталозі	Так
Пошук серед каталогу для неавторизованих користувачів	Немає	Немає	Є
Відтворення музики в якості без втрат	Немає	Немає	Є
Пошук музики на основі емоцій	Немає	Немає	Є

Однією з важливих особливостей для програми, яка розробляється, є можливість пошуку серед каталогу для неавторизованих користувачів. У такому випадку користувач зможе ознайомитись з доступним каталогом, але для відтворення музики йому потрібно буде авторизуватися. У той же час в Spotify або в YouTube Music користувачу для цього потрібно одразу авторизуватися, інакше доступ до каталогу неможливий.

1.2 Висновки за розділом 1

Завдання дипломної кваліфікаційної роботи для досягнення цілей роботи:

- виконати аналіз предметної області;
- виконати дослідження методів підбору музики;
- виконати проєктування ПЗ;
- виконати реалізацію ПЗ.

У першому розділі виконаний аналіз програмних аналогів, включаючи зокрема Spotify, YouTube Music, Chosic, дозволив сформулювати ідею ПЗ, яке створюється, виділити основні функціональні особливості такої розробки. На відміну від великих потокових сервісів передбачається використання розробки для відтворення музики на основі нішевого каталогу для забезпечення діяльності окремої компанії, яка розповсюджує авторські права деякої групи правовласників і не бажає працювати з великими компаніями.

2 МАТЕРІАЛИ І МЕТОДИ

Під час виконання підбору музики потрібно визначити з множини доступних композицій ті, які відповідають певним критеріям. У даному випадку таким критерієм є відповідність емоцій слухача та підібраних композицій.

2.1 Методи класифікації даних для підбору музики

Згорткові нейронні мережі відрізняються від інших нейронних мереж високою продуктивністю при введенні зображення або аудіосигналу, вони мають три основні типи шарів: у той час як за згортковими шарами можуть слідувати додаткові згорткові шари або шари об'єднання, повністю пов'язаний шар є останнім шаром, з кожним шаром згорткова нейронна мережа збільшується у своїй складності, ідентифікуючи більші частини зображення, більш ранні шари зосереджені на простих елементах, таких як кольори [10].

Згортковий шар є першим шаром згорткової мережі:

- коли дані зображення просуваються через шари мережі, вона починає розпізнавати більші елементи або форми об'єкта, поки нарешті не ідентифікує передбачуваний об'єкт;

- цей шар є основним будівельним блоком згорткової нейронної мережі, і саме там відбувається більшість обчислень, для цього потрібно декілька компонентів (вхідні дані, фільтр і карта ознак);

- детектор або карта ознак, також відома як ядро або фільтр, буде переміщатися по сприйнятливих полях зображення (цей процес відомий як згортка);

- фільтр – це двовимірний масив ваг, який представляє частину зображення (зазвичай карта є матрицею 3×3 , що також визначає розмір рецептивного поля);

- під час застосування фільтру до області зображення між вхідними пікселями та фільтром обчислюється крапковий добуток, цей крапковий добуток потім подається у вихідний масив, після цього фільтр зміщується на один крок, повторюючи процес, поки ядро не охопить усе зображення;
- остаточний вихід із ряду точкових добутків із входу та фільтра відомий як карта ознак, карта активації або згорнута функція;
- після кожної операції згортки мережа застосовує перетворення Rectified Linear Unit до карти об'єктів, вносячи нелінійність у модель;
- інший шар згортки може слідувати за початковим шаром згортки: у такому випадку структура мережі може стати ієрархічною, оскільки пізні рівні можуть бачити пікселі в сприйнятливих полях попередніх шарів [10]-[11].

Об'єднуючий шар зменшує розмірність, зменшуючи кількість параметрів у вхідних даних:

- подібно до згорткового шару, операція об'єднання об'єднує фільтр по всьому входу, але різниця в тому, що цей фільтр не має жодних ваг: замість цього ядро застосовує функцію агрегації до значень у сприйнятливому полі, заповнюючи вихідний масив;
- існує два основних типи об'єднання;
- максимальне об'єднання – під час переміщення фільтра по входу він вибирає піксель з максимальним значенням для відправки на вихідний масив, крім того цей підхід, як правило, використовується частіше в порівнянні з середнім об'єднанням;
- середнє об'єднання: коли фільтр переміщується по вхідному сигналу, він обчислює середнє значення в полі сприйняття для відправлення до вихідного масиву [10]-[11].

Повнозв'язний шар виконує завдання класифікації на основі ознак, витягнутих через попередні шари: значення пікселів вхідного зображення не пов'язані безпосередньо з вихідним шаром у частково з'єднаних шарах, однак у повністю зв'язному шарі кожен вузол вихідного шару підключається

безпосередньо до вузла попереднього шару, у той час як рівні згортки та об'єднання, як правило, використовують функції Rectified Linear Unit, цей рівень зазвичай використовує функцію активації softmax для належної класифікації вхідних даних, створюючи ймовірність від 0 до 1 [10].

2.2 Метод підбору музики

Метод підбору музики (рис. 2.1) повинен виконати розділення музичних композицій за емоціями, яким вони відповідають, та виконати виділення емоцій людини, щоб співставити ці два джерела і виділити загальне рішення.

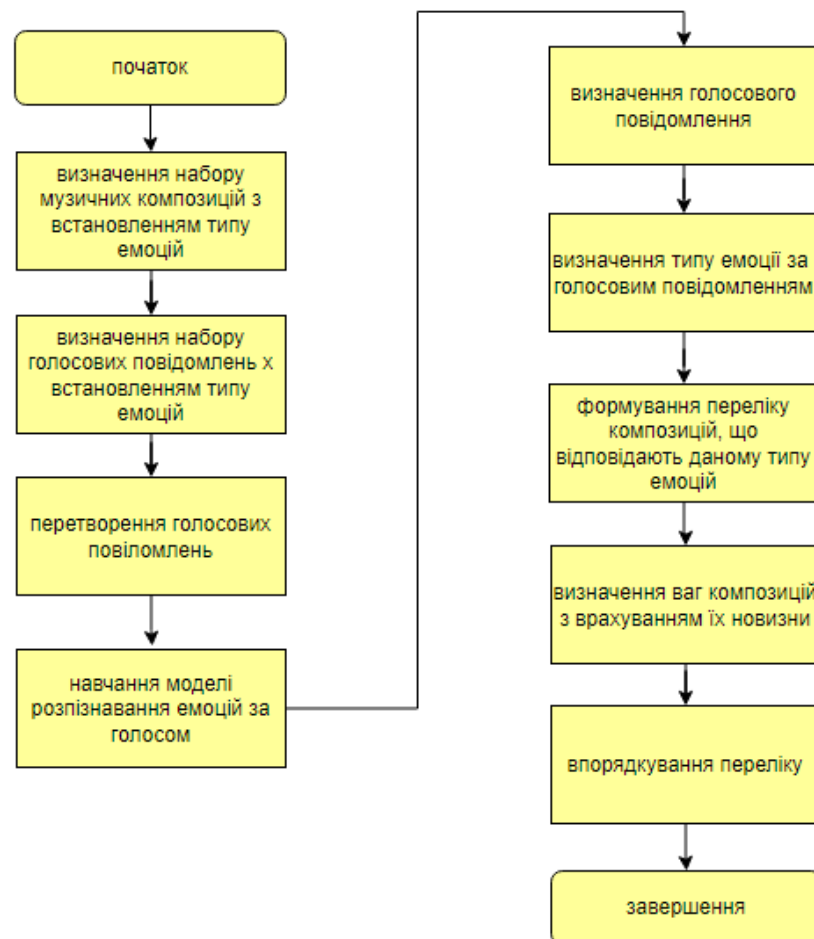


Рисунок 2.1 – Схема роботи методу підбору музики

Спочатку мають бути виділені емоції, яким мають відповідати

музичні композиції. Тобто реалізується накопичення музичних композицій і кожна з них має бути розмічена відповідним типом емоцій.

Після цього має бути сформовано набір даних, який складатиметься з голосових повідомлень з віднесенням для кожного з них типу емоцій.

Для того щоб виділити емоції людини, вирішено було використовувати голосове повідомлення від слухача. В ньому він має відповісти на привітальне повідомлення, створене програмою шляхом відтворення раніше записаного повідомлення, а далі сформулювати якесь побажання або просто проговорити якусь коротку фразу. Саме цей фрагмент використовується методом для подальшої роботи.

На наступному етапі реалізується перетворення голосових повідомлень до графічної форми. Для цього кожен фрагмент розділяється до мінімального розміру і створюється спектрограма відповідного фрагмента. Кожній спектрограмі встановлюється відповідний тип емоції на основі результатів виконання другого етапу.

Далі реалізується створення і навчання моделі розпізнавання емоцій за голосом людини. Для цього використовується згортоква нейронна мережа, яка складається з двох згорткових шарів зі слідуєчими за ними об'єднуючими шарами і застосуванням повнозв'язного шару на виході.

Модель навчається на наборі даних, який був отриманий у обробленому вигляді на попередньому етапі.

Після цього користувач має задати нове голосове повідомлення, а навчена нейронна мережа має класифікувати його, визначивши при цьому мітку, яка відповідає типу емоцій. За встановленою міткою на наступному етапі визначається перелік композицій, які відповідають тому самому типу емоцій.

На наступному етапі реалізується обчислення ваг для кожної композиції з переліку, сформованого на попередньому етапі. Для цього має бути встановлено для кожного типу емоцій відповідне число для улюблених композицій користувача і для нових композицій. Далі кожна з композицій

множиться на відповідне для даного типу емоцій число. Початковою вагою композиції є 1. В залежності від того, чи є ця композиція улюбленою для користувача, чи є вона новою (була опублікована протягом тижня), вона отримує відповідне значення ваги.

Далі за отриманими значеннями ваг всі композиції впорядковуються і таким чином формується перелік композицій за результатами відбору музики.

2.3 Висновки за розділом 2

Розглянуто принципи побудови згорткових нейронних мереж для розв'язання задачі класифікації. Запропоновано метод підбору музики, заснований на розбитті даного процесу на етапи виділення емоцій за голосом людини та підбору музичних композицій за виділеними емоціями, що дозволяє формувати перелік композицій для прослуховування з різним ступенем новизни в залежності від типу емоцій.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів веброзробки

Під час першого рішення щодо засобів, за допомогою яких має вестися розробка, було вирішено використовувати мову програмування Python [12]. Для реалізації вебзастосунку, адже вона є зручною для швидкої розробки застосунків, придатною для веброзробки і має велику кількість бібліотек для роботи з даними для підтримки функціональності представленого методу.

Вибір фреймворку веброзробки (табл. 3.1) був зведений до порівняння за засобами безпеки (можливість захисту від різного роду атак, від внесення даних, які можуть спотворити систему), засобами роботи над керуванням даними (важлива наявність адміністративної панелі для пришвидшення розробки), спільнотою розробників (простіше знайти відповіді на проблемні запитання), кількістю вбудованих засобів, а найкращим за даними показниками є фреймворк Django [13], тож його і було обрано.

Таблиця 3.1 – Вибір фреймворку веброзробки

Критерій	Flask	Django
Сучасні засоби безпеки	В основному не підтримуються самим фреймворком	Є
Засоби роботи над керуванням даними	Немає	Є
Спільнота розробників	Середня	Велика
Кількість вбудованих засобів	Невелика	Достатньо велика

3.2 Функціональні вимоги до програмного забезпечення

Реалізуючи всі вимоги до виконуваної розробки для забезпечення її

особливостей порівняно з існуючими програмами та основні вимоги до подібного типу програмних систем, необхідно забезпечити виконання наступних функціональних вимог:

- відтворення музичних композицій;
- пошук музичних композицій за назвою;
- пошук альбомів музичних композицій;
- формування переліку нових композицій, які викликають інтерес у інших слухачів;
- створення власних переліків музичних композицій і подальше їх редагування;
- знаходження переліків музичних композицій, створених іншими слухачами;
- підбір музичних композицій за настроєм;
- вподобання музичної композиції;
- зазначення антипатії до музичної композиції;
- перегляд переліку композицій з музичного альбому;
- перегляд переліку альбомів за музичним виконавцем;
- підписування на музичного виконавця;
- формування переліку нових композицій серед виконавців, на які підписаний слухач;
- додавання музичних альбомів до власної бібліотеки;
- формування переліку найкращих композицій за рік;
- формування переліку найкращих композицій за країнами;
- формування переліку жанрів, які найбільш часто прослуховуються у країні;
- формування переліку композицій виконавця з найбільшою увагою слухачів;
- керування музичними виконавцями, альбомами та композиціями;
- авторизація слухача і адміністратора системи;
- реєстрація слухача.

Усі функціональні вимоги були доведені до сценаріїв використання ПЗ різними користувачами. Основними користувачами програми є слухачі та неавторизовані користувачі, яких бажано перетворити на слухачів.

Неавторизовані користувачі можуть тільки ознайомитися з даними музичних композицій. Їх сценарії використання приведені на рис. 3.1. Сценарій пошуку даних узагальнює сценарії пошуку виконавців, альбомів та музичних композицій. Сценарій реєстрації розширює сценарій авторизації.



Рисунок 3.1 – Діаграма варіантів використання ПЗ неавторизованим користувачем

Слухач (рис. 3.2) має повний доступ до всіх інструментів роботи з музичними композиціями. У випадку платної підписки достатньо перевіряти її під час авторизації, оскільки саме ці дві ролі є відмінністю між платним і безкоштовним відтворенням музики.

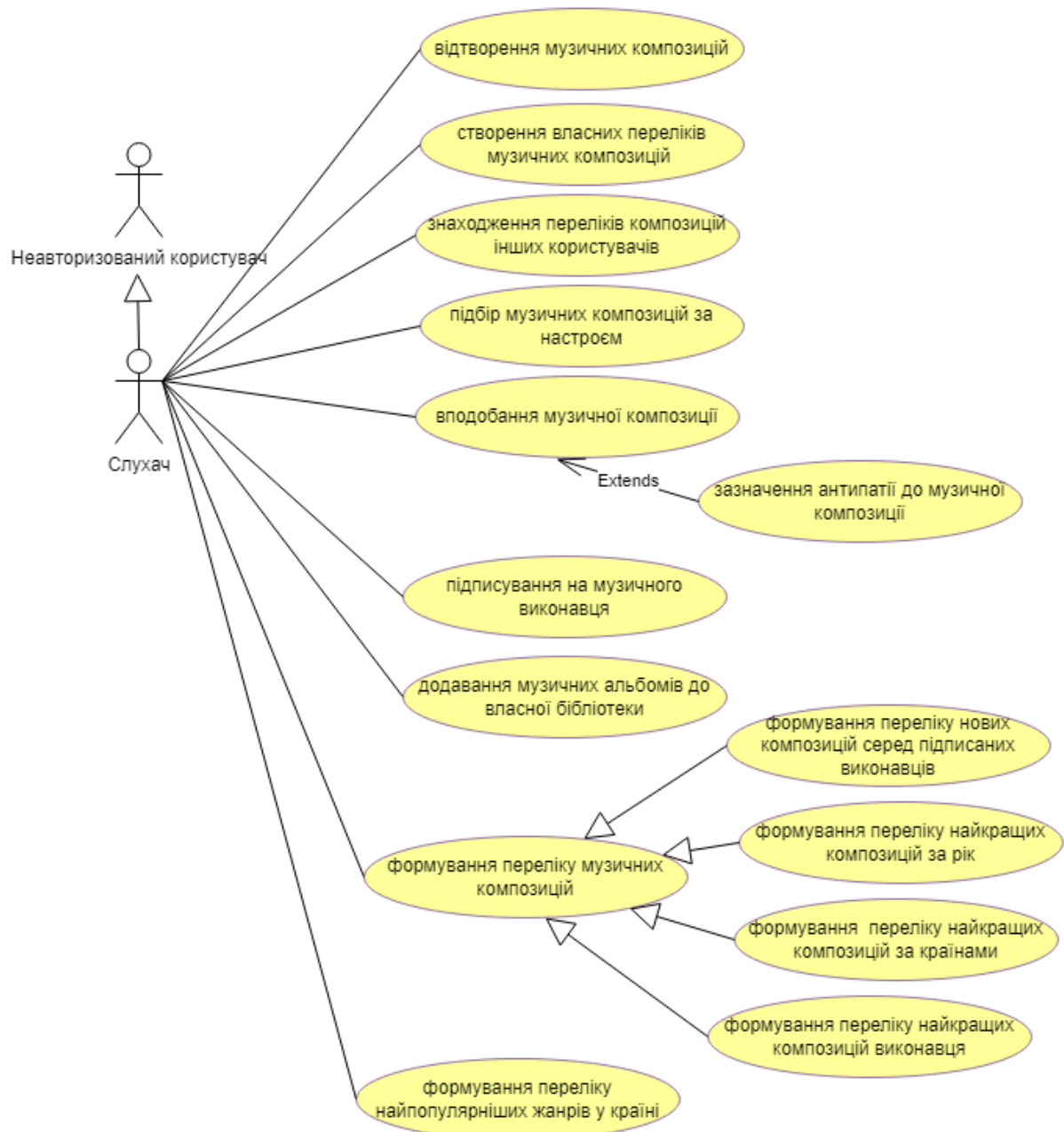


Рисунок 3.2 – Діаграма варіантів використання ПЗ авторизованим користувачем

Ще одним можливим типом користувачів є адміністратор. Він забезпечує тільки загальне керування даними, з якими працюють інші користувачі. До таких даних належать дані музичних виконавців, альбомів та композицій. Дана функціональність має бути забезпечена через стандартну адміністративну панель Django.

3.3 Проектування бази даних

Для підтримки роботи визначених функцій програми необхідно було забезпечити роботу зі створення сховища даних, представленого базою даних (БД). У результаті її було визначено у формі реляційної БД з представленням таблиць БД і зв'язків між ними. Для підтримки роботи з БД використано систему керування БД MySQL як одну з найбільш широко використовуваних, стабільних і безкоштовних систем керування БД.

Таблиця MusicAuthor представляє виконавця музичних композицій на основі наступних полів:

- id – ідентифікатор виконавця музичних композицій;
- artist – назва виконавця;
- bandflag – поле, що визначає, чи є виконавець гуртом, чи сольним виконавцем;
- story – короткий опис виконавця;
- site – вебсайт виконавця;
- photo – фотографія виконавця;
- subscribers – кількість підписників на виконавця.

Таблиця MusicAlbum представляє альбом виконавця музичних композицій на основі наступних полів:

- id – ідентифікатор альбому;
- name – назва альбому;
- moment – дата, коли був виданий альбом;
- photo – фотографія обкладинки музичного альбому;
- musicauthor – виконавець, який створив альбом: представлений зовнішнім ключем;
- inlibraries – кількість внесень музичного альбому слухачами до своєї бібліотеки.

Між таблицями MusicAuthor і MusicAlbum зв'язок «багато до багатьох»: один виконавець може мати від нуля до великої кількості різних

музичних альбомів, а кожен альбом може мати хоча би одного, а може і більше авторів, які є його основними виконавцями, зазначеними на обкладинці.

Зв'язок «багато до багатьох» реалізовано створенням проміжної таблиці MusicAlbumAuthors, що складається з ідентифікатора альбому та ідентифікатора виконавця, що формує складений ключ.

Таблиця MusicComposition представляє музичну композицію на основі наступних полів:

- id – ідентифікатор музичної композиції;
- title – назва музичної композиції;
- photo – фотографія, яка використовується для ілюстрації музичної композиції;
- musicalbum – музичний альбом, до якого входить композиція, визначений зовнішнім ключем (може не мати ніякого значення, якщо пісня не входить в альбом, хоча стандартним сценарієм роботи є внесення композиції до якогось альбому);
- musictext – текст музичної композиції;
- listens – кількість разів, яку було прослухано музичну композицію;
- likes – кількість уподобань музичної композиції;
- dislikes – кількість негативних реакцій на музичну композицію;
- stream – файл музичної композиції (шлях до файлу або адреса файлу в мережі).

Між таблицями MusicAuthor і MusicComposition зв'язок «багато до багатьох»: один виконавець може мати від нуля до великої кількості різних музичних композицій, а кожна композиція може мати хоча би одного, а може і більше виконавців.

Зв'язок «багато до багатьох» реалізовано створенням проміжної таблиці MusicCompositionAuthors, що складається з ідентифікатора композиції та ідентифікатора виконавця, що формує складений ключ.

Вводяться відповідні проміжні таблиці між таблицями MusicComposition і MusicAuthors та між таблицями MusicAlbum і MusicAuthor при наявності зв'язку між таблицями MusicComposition і MusicAlbum тому, що альбом і композиція з нього можуть мати різний склад виконавців. Наприклад, альбом може виконувати пара виконавців, тобто загалом робота визначена їх спільною, а окрему композицію при цьому може виконувати лише один з виконавців.

Таблиця Subscribe представляє підписку слухача на музичного виконавця на основі наступних полів:

- id – ідентифікатор підписки;
- musicauthor – виконавець, на якого оформлено підписку, визначений зовнішнім ключем;
- listener – слухач, який оформив підписку, визначений зовнішнім ключем;
- moment – дата і час, коли було оформлено підписку.

Між таблицями MusicAuthor і Subscribe зв'язок «один до багатьох»: один виконавець може мати від нуля до великої кількості різних підписок слухачів, а кожна підписка оформлена тільки на одного виконавця.

Між таблицями Listener і Subscribe зв'язок «один до багатьох»: один слухач може мати від нуля до великої кількості різних підписок, а кожна підписка оформлена одним слухачем.

Таблиця Library представляє внесення музичного альбому в бібліотеку слухача на основі наступних полів:

- id – ідентифікатор музичної композиції;
- musicalalbum – музичний альбом, визначений зовнішнім ключем;
- listener – слухач, який вніс музичний альбом у бібліотеку, визначений зовнішнім ключем;
- moment – дата і час внесення музичного альбому в бібліотеку слухачем;

– listens – кількість прослуховувань слухачем, який вніс цей альбом в свою бібліотеку, даного альбому.

Між таблицями MusicAlbum і Library зв'язок «один до багатьох»: один альбом може входити в бібліотеки різних слухачів (від нуля до великої кількості), а кожна бібліотека вміщує один альбом тільки один раз.

Між таблицями Listener і Library зв'язок «один до багатьох»: один слухач може мати від нуля до великої кількості різних альбомів у бібліотеці слухача, але кожний запис у бібліотеці належить тільки одному слухачу.

Таблиця Listener представляє слухача на основі наступних полів:

- id – ідентифікатор слухача;
- username – ім'я користувача, який є слухачем;
- password – пароль слухача;
- birthdate – дата народження слухача;
- email – електронна пошта слухача;
- photo – фотографія слухача;
- country – країна, в якій проживає слухач., визначений зовнішнім ключем.

Між таблицями Country і Listener зв'язок «один до багатьох»: одна країна може мати від нуля до великої кількості різних слухачів у програмі, але кожний слухач може бути зазначений проживаючим тільки в одній країні.

Таблиця Like представляє уподобання слухачем музичної композиції на основі наступних полів:

- id – ідентифікатор уподобання;
- musiccomposition – музична композиція, яку було уподобано, визначена зовнішнім ключем;
- listener – слухач, який уподобав музичну композицію, визначений зовнішнім ключем;
- type – тип уподобання (уподобання або антипатія).

Між таблицями MusicComposition і Like зв'язок «один до багатьох»: одна музична композиція може мати від нуля до великої кількості уподобань, але кожне уподобання відповідає тільки одній музичній композиції.

Між таблицями Listener і Like зв'язок «один до багатьох»: один слухач може мати від нуля до великої кількості уподобань, але кожне уподобання створюється тільки одним конкретним слухачем.

Таблиця Listen представляє прослуховування слухачем музичної композиції на основі наступних полів:

- id – ідентифікатор прослуховування;
- musiccomposition – музична композиція, яку прослухав слухач, визначена зовнішнім ключем;
- listener – слухач, визначений зовнішнім ключем;
- moment – дата і час прослуховування.

Між таблицями MusicComposition і Listen зв'язок «один до багатьох»: одна музична композиція може мати від нуля до великої кількості прослуховувань, але кожне прослуховування відповідає тільки одній музичній композиції.

Між таблицями Listener і Listen зв'язок «один до багатьох»: один слухач може мати від нуля до великої кількості прослуховувань, але кожне прослуховування реалізується тільки одним конкретним слухачем.

Таблиця Genre представляє жанр на основі наступних полів:

- id – ідентифікатор жанру;
- name – назва жанру.

Між таблицями Genre і MusicComposition зв'язок «багато до багатьох»: один жанр може бути представлений від нуля до великої кількості різних музичних композицій, а кожна композиція може мати від нуля до великої кількості жанрів.

Зв'язок «багато до багатьох» реалізовано створенням проміжної таблиці MusicCompositionGenre, що складається з ідентифікатора композиції та ідентифікатора жанру, що формує складений ключ.

Таблиця MusicList (рис. 3.3) представляє перелік музичних композицій, створений слухачем, на основі наступних полів:

- id – ідентифікатор переліку музичних композицій;
- listener – слухач, який створив перелік, визначений зовнішнім ключем;
- name – назва переліку;
- privatelevel – рівень доступу (до приватних переліків не можна отримати доступ іншим користувачам).

Між таблицями Listener і MusicList зв'язок «один до багатьох»: один слухач може мати від нуля до великої кількості різних переліків музичних композицій, а кожен перелік належить тільки одному слухачу.

Між таблицями MusicList і Composition зв'язок «багато до багатьох»: один перелік може мати від нуля до великої кількості різних музичних композицій, а кожна композиція може входити до великої кількості жанрів (або в жодний).

Зв'язок «багато до багатьох» реалізовано створенням проміжної таблиці MusicListComposition, що складається з ідентифікатора композиції та ідентифікатора переліку, що формує складений ключ.

Таблиця Tag представляє тег на основі наступних полів:

- id – ідентифікатор тегу;
- name – назва тегу.

Між таблицями MusicList і Tag зв'язок «багато до багатьох»: один перелік може мати від нуля до великої кількості різних тегів, а кожен тег може характеризувати від нуля до великої кількості переліків.

Зв'язок «багато до багатьох» реалізовано створенням проміжної таблиці MusicListTag, що складається з ідентифікатора композиції та ідентифікатора тегу, що формує складений ключ.

Таблиця Country представляє країну на основі наступних полів:

- id – ідентифікатор країни;
- name – назва країни.

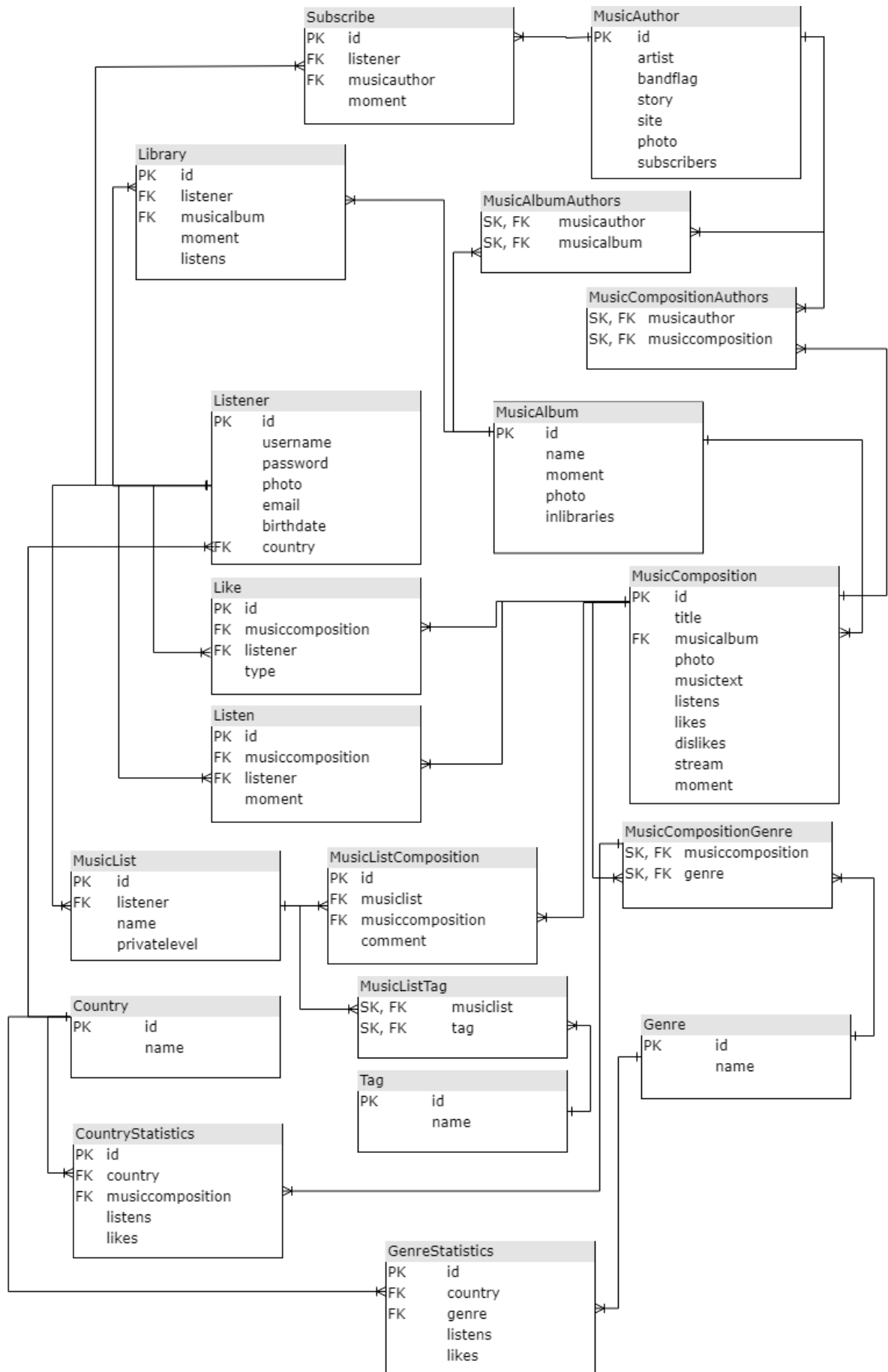


Рисунок 3.3 – Схема БД

Таблиця CountryStatistics представляє статистику прослуховування музичних композицій за країнами на основі наступних полів:

- id – ідентифікатор запису;
- country – країна, статистика за якою задана, визначена зовнішнім ключем;
- musiccomposition – музична композиція, статистика за якою задана, визначена зовнішнім ключем;
- listens – кількість прослуховувань цієї музичної композиції слухачами з цієї країни;
- likes – кількість уподобань цієї музичної композиції слухачами з цієї країни.

Таблиця GenreStatistics представляє статистику прослуховування жанрів музичних композицій на основі наступних полів:

- id – ідентифікатор запису;
- country – країна, статистика за якою задана, визначена зовнішнім ключем;
- genre – жанр, статистика за яким задана, визначений зовнішнім ключем;
- listens – кількість прослуховувань музичних композицій цього жанру слухачами з цієї країни;
- likes – кількість уподобань музичних композицій цього жанру слухачами з цієї країни.

3.4 Структура програмного забезпечення

Програмна система відтворення музики складається з двох основних підсистем, які в процесі своєї взаємодії визначають можливості пошуку музики для слухача та її відтворення.

Підсистема підбору музики за емоціями слухачів реалізує метод, представлений в розділі 2, і використовує в якості вхідних даних голос користувача, який відповідає на привітальні повідомлення системи.

Підсистема відтворення музики є вебзастосунком, реалізованим на основі фреймворку Django. Вона забезпечує формування переліків композицій в інтерфейсі користувача, забезпечує будь-яку комунікацію з ним, відтворює музику, а безпосередньо за засобами визначення переліку композицій за емоціями звертається до першої підсистеми.

Результуюча структура системи представлена на рис. 3.4.



Рисунок 3.4 – Схема структури програми

3.5 Висновки за розділом 3

У третьому розділі було визначено мову Python для розробки програмного застосунку, виконано порівняння фреймворків розробки вебзастосунків для Python – Django та Flask. Для реалізації цієї системи було обрано Django. Виконано проєктування роботи з даними. Визначено схему структури програми з розбиттям програмної системи на дві підсистеми.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування програми

Алгоритм функціонування програми побудований на основі необхідності розподілу функцій між авторизованими користувачами і неавторизованими та на створенні меню для вибору режимів роботи. Режим роботи є вхідними точками у послідовність реалізації функціональності у подальшому. Обираючи певний альбом або виконавця, користувач далі може перейти до потрібної композиції, прослухати її.

Під час початку роботи користувач отримує інформацію про програму загалом, а далі може обрати перегляд нових композицій або виконати пошук даних. Пошук даних об'єднує в собі пошук виконавців, альбомів та композицій за їх назвою. Для виконання всіх інших дій йому потрібно спочатку авторизуватися, що і є ще одним можливим режимом роботи програми у початковий момент.

Коли слухач виконав авторизацію, він обирає варіант роботи з розширеного переліку, що охоплює зокрема перегляд статистики прослуховувань, підбір музики за емоціями слухача, перегляд і керування власними підписками на музичних виконавців, перегляд і керування власними переліками музичних композицій. У будь-який момент слухач може обрати потрібну йому функцію з меню програми, після чого повернутися до цього вибору заново. Цей процес продовжується до того моменту, поки слухач не обере вихід з облікового запису або не завершить роботу програми.

Створена в результаті схема функціонування програми приведена на рис. 4.1.

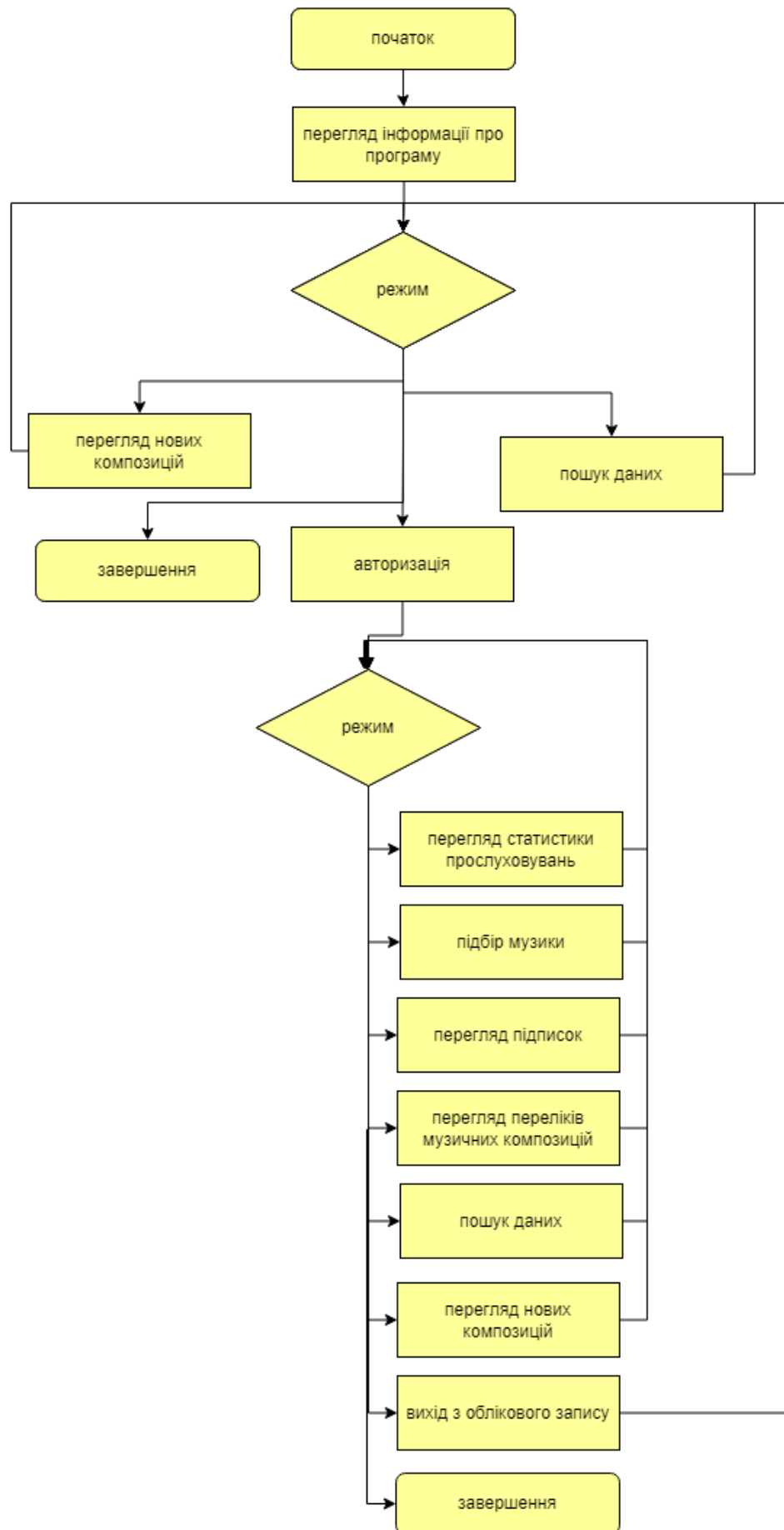


Рисунок 4.1 – Схема функціонування програми

4.2 Опис основних розроблених класів

Основні розроблені в роботі класи охоплюють класи представлень, які формують дані для заповнення шаблонів ними, та класи моделей, які визначають структуру самих даних.

Клас представлення `MusicCompositionView` виконує заповнення шаблону `music.html` даними музичної композиції з номером, переданим через параметри методу, додаючи перелік усіх уподобань даної музичної композиції. Клас створений наслідуванням класу `DetailView`, забезпечуючи деталізацію музичної композиції.

Клас представлення `AddMusicCompositionView` виконує підтримку додавання нової музичної композиції, забезпечуючи для цього збереження даних, отриманих через поля форми, на основі шаблону `addmusic.html`, додаючи при цьому ініціалізуючі нульові значення кількості уподобань, негативних реакцій та прослуховувань. Клас створений наслідуванням класу `CreateView` та класу авторизації `LoginRequiredMixin`.

Клас представлення `UpdateMusicCompositionView` виконує підтримку редагування музичної композиції, забезпечуючи для цього збереження даних, отриманих через поля форми, на основі шаблону `updatemusic.html`. Клас створений наслідуванням класу `UpdateView` та класу авторизації `LoginRequiredMixin`.

Клас `MostListenedAuthorMusicCompositionsView` забезпечує підтримку представлення перегляду переліку найкращих музичних композицій виконавця, номер якого передається через параметр `author` GET-запиту. Визначення найкращих композицій відбувається за кількістю прослуховувань, повертаються 10 найкращих композицій за цим критерієм. Клас створений наслідуванням класу `ListView`. Дані заповнюють шаблон `bestlistenedauthormusic.html`.

Наступні класи також організовані наслідуванням класу `ListView`.

Клас `AlbumMusicView` забезпечує підтримку представлення перегляду музичних композицій альбому, номер якого передається через параметр `album` GET-запиту. Дані заповнюють шаблон `albummusic.html`.

Клас `MostAnticipatedAuthorMusicCompositionsView` забезпечує підтримку представлення перегляду переліку музичних композицій виконавця, номер якого передається через параметр `author` GET-запиту, які мають найбільшу кількість негативних реакцій у слухачів. Повертаються 10 композицій з найбільшим значенням за цим критерієм. Дані заповнюють шаблон `bestdislikedauthormusic.html`.

Клас `MostLikedAuthorMusicCompositionsView` забезпечує підтримку представлення перегляду переліку найкращих музичних композицій виконавця, номер якого передається через параметр `author` GET-запиту. Визначення найкращих композицій відбувається за кількістю уподобань, повертаються 10 найкращих композицій за цим критерієм. Дані заповнюють шаблон `bestlikedauthormusic.html`.

Клас `YearPopularMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 музичних композицій, випущених протягом останнього року, які отримали найбільшу кількість уподобань від слухачів, заповнюючи даними шаблон `yearpopularnewmusic.html`.

Клас `MostListenedMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 найкращих музичних композицій, які було прослухано найчастіше, заповнюючи даними шаблон `bestlistenedmusic.html`.

Клас `MostAnticipatedMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 музичних композицій, які викликали негативні емоції в найбільшій кількості слухачів, заповнюючи даними шаблон `bestdislikedmusic.html`.

Клас `MostLikedMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 музичних композицій, які було

уподобано найбільшою кількістю слухачів, заповнюючи даними шаблон `bestlikedmusic.html`.

Клас `MostLikedCountryMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 музичних композицій, які було уподобано найбільшою кількістю слухачів у заданій через параметр `country` GET-запиту країні, заповнюючи даними шаблон `bestlikedcountrymusic.html`.

Клас `MostDisLikedCountryMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 музичних композицій, які отримали негативні оцінки від найбільшої кількості слухачів у заданій через параметр `country` GET-запиту країні, заповнюючи даними шаблон `bestdislikedcountrymusic.html`.

Клас `MostListenedCountryMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 музичних композицій, які було найбільше прослухані слухачами у заданій через параметр `country` GET-запиту країні, заповнюючи даними шаблон `bestlistenedcountrymusic.html`.

Клас `NewPopularMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 музичних композицій, які були опубліковані за останній тиждень і мали найбільшу кількість уподобань, заповнюючи даними шаблон `popularnewmusic.html`. Усі шаблони розроблені на основі шаблону [14].

Клас `NewPopularSubscribedMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 100 музичних композицій, які були опубліковані за останній тиждень виконавцями, на які підписався авторизований у даний момент користувач, заповнюючи даними шаблон `popularsubsmusic.html`.

Реалізація даного відбору композицій реалізується наступним чином.

Спочатку виконується визначення композицій, які були опубліковані протягом останнього тижня, сортуючи їх за кількістю уподобань:

```
musiccontext = super().get_queryset(*args, **kwargs)
```

```

endnewdate = datetime.today() - timedelta(days=7)
compos = MusicComposition.objects.filter(moment__gte = endnewdate)
compos = compos.order_by('-likes')

```

Після цього відбувається перевірка для композицій того, чи є хоча би один з виконавців композиції таким, на якого підписався авторизований у даний момент слухач (якщо так, то композиція залишається, якщо ні, то пропускається, продовжуючи, поки не буде знайдено 100 таких композицій):

```

mus = {}
for music in compos:
    ms = music.musicauthors
    sub = Subscribe.objects.filter(listener_id = self.request.user.id)
    flag = False
    for s in sub:
        if s.musicauthor in ms:
            flag = True
    if flag:
        mus.add(music)
    if mus.count() == 100:
        break

```

Клас `ListenerLikedMusicCompositionsView` забезпечує підтримку представлення перегляду переліку музичних композицій, які були уподобані користувачем, авторизованим у даний момент, заповнюючи даними шаблон `likedmusic.html`.

Клас `MostListenedAuthorMusicCompositionsView` забезпечує підтримку представлення перегляду переліку 10 музичних композицій заданого через параметр `author` GET-запиту виконавця, які були прослухані

найбільшу кількість разів, заповнюючи даними шаблон `bestlistenedauthormusic.html`.

Клас `SearchMusicCompositionsView` забезпечує підтримку представлення перегляду переліку музичних композицій, які містять в назві `title`, значення якого передано параметром GET-запиту, заповнюючи даними шаблон `searchmusic.html`.

Клас `AuthorCompositionsView` забезпечує підтримку представлення перегляду переліку всіх музичних композицій виконавця, номер якого було передано через значення параметра `author` GET-запиту, заповнюючи даними шаблон `authormusic.html`.

Аналогічно влаштовані інші класи представлень.

Клас `MusicAuthor` є класом моделі виконавця, що організований наслідуванням базового класу моделі Django Model і включає наступні атрибути, що відповідають структурі таблиці:

- `artist` – поле символьного типу, створене на основі класу `CharField`, обмежене довжиною в 255 символів;
- `bandflag` – поле логічного типу, створене на основі класу `BooleanField`;
- `story` – поле символьного типу, створене на основі класу `CharField`;
- `site` – поле символьного типу, створене на основі класу `CharField`, обмежене 255 символами;
- `photo` – поле символьного типу, створене на основі класу `CharField`, обмежене 255 символами;
- `subscribers` – поле цілочисельного типу, створене на основі класу `IntegerField`.

Клас містить метод для визначення рядкового представлення об'єкта виконавця. Також клас містить метод створення прямих посилань на об'єкт `get_absolute_url`.

Клас `MusicAlbum` є класом моделі музичного альбому, що організований наслідуванням базового класу моделі `Model` і включає наступні атрибути, що відповідають структурі таблиці:

- `artist` – поле символьного типу, створене на основі класу `CharField`, обмежене довжиною в 255 символів;
- `name` – поле символьного типу, створене на основі класу `CharField`, обмежене довжиною в 255 символів;
- `moment` – поле дати, створене на основі класу `DateField`;
- `photo` – поле символьного типу, створене на основі класу `CharField`, обмежене довжиною в 255 символів;
- `inlibraries` – цілочисельне поле, створене на основі класу `IntegerField`;
- `musicauthors` – поле зв'язку «багато до багатьох», створене на основі класу `ManyToManyField`, визначає зв'язок з моделлю `MusicAuthor` і зберігає перелік всіх виконавців музичного альбому.

Клас містить метод для визначення рядкового представлення об'єкта альбому. Також клас містить метод створення прямих посилань на об'єкт `get_absolute_url`.

Клас `MusicComposition` є класом моделі музичної композиції, що організований наслідуванням базового класу моделі `Model` і включає наступні атрибути, що відповідають структурі таблиці:

- `title` – поле символьного типу, створене на основі класу `CharField`, обмежене довжиною в 255 символів;
- `photo` – поле символьного типу, створене на основі класу `CharField`, обмежене довжиною в 255 символів;
- `musicalbum` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `MusicAlbum`;
- `moment` – поле дати, створене на основі класу `DateField`;
- `musictext` – текстове поле, створене на основі класу `TextField`;
- `listens` – цілочисельне поле, створене на основі класу `IntegerField`;
- `likes` – цілочисельне поле, створене на основі класу `IntegerField`;

- dislikes – цілочисельне поле, створене на основі класу IntegerField;
- stream – поле символьного типу, створене на основі класу CharField, обмежене довжиною в 255 символів;
- genres – поле зв'язку «багато до багатьох», створене на основі класу ManyToManyField, визначає зв'язок з моделлю Genre і зберігає перелік всіх жанрів музичної композиції;
- musicauthors – поле зв'язку «багато до багатьох», створене на основі класу ManyToManyField, визначає зв'язок з моделлю MusicAuthor і зберігає перелік всіх виконавців музичної композиції.

Клас містить метод для визначення рядкового представлення об'єкта композиції. Також клас містить метод створення прямих посилань на об'єкт `get_absolute_url`.

Клас `Subscribe` є класом моделі підписки, що організований наслідуванням базового класу моделі `Model` і включає наступні атрибути, що відповідають структурі таблиці:

- moment – поле дати, створене на основі класу DateField;
- listener – поле зовнішнього ключа, створене на основі класу ForeignKey для зв'язку з моделлю Listener;
- musicauthor – поле зовнішнього ключа, створене на основі класу ForeignKey для зв'язку з моделлю MusicAuthor.

Клас `Library` є класом моделі бібліотеки музичних альбомів, що організований наслідуванням базового класу моделі `Model` і включає наступні атрибути, що відповідають структурі таблиці:

- moment – поле дати, створене на основі класу DateField;
- listener – поле зовнішнього ключа, створене на основі класу ForeignKey для зв'язку з моделлю Listener;
- musicalbum – поле зовнішнього ключа, створене на основі класу ForeignKey для зв'язку з моделлю MusicAlbum;
- listens – цілочисельне поле, створене на основі класу IntegerField.

Клас `Listener` є класом моделі слухача, що організований наслідуванням класу `AbstractUser` з отриманням відповідних атрибутів і включає додатково наступні атрибути, що відповідають структурі таблиці:

- `photo` – поле символьного типу, створене на основі класу `CharField`, обмежене довжиною в 255 символів;
- `email` – поле символьного типу, створене на основі класу `CharField`, обмежене довжиною в 255 символів;
- `country` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `Country`;
- `birthdate` – поле дати, створене на основі класу `DateField`.

Клас `Like` є класом моделі уподобання слухачем музичної композиції, що організований наслідуванням базового класу моделі `Model` і включає наступні атрибути, що відповідають структурі таблиці:

- `moment` – поле дати, створене на основі класу `DateField`;
- `listener` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `Listener`;
- `musiccomposition` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `MusicComposition`;
- `type` – поле логічного типу, створене на основі класу `BooleanField`.

Клас `Listen` є класом моделі прослуховування слухачем музичної композиції, що організований наслідуванням базового класу моделі `Model` і включає наступні атрибути, що відповідають структурі таблиці:

- `moment` – поле дати, створене на основі класу `DateField`;
- `listener` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `Listener`;
- `musiccomposition` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `MusicComposition`.

Клас `Genre` є класом моделі жанру, що організований наслідуванням базового класу моделі `Model` і включає атрибут `name`, який є полем

символьного типу, створеним на основі класу CharField, обмеженим довжиною в 255 символів.

Клас Country є класом моделі країни, що організований наслідуванням базового класу моделі Model і включає атрибут name, який є полем символьного типу, створеним на основі класу CharField, обмеженим довжиною в 255 символів.

Клас Tag є класом моделі тегу, що організований наслідуванням базового класу моделі Model і включає атрибут name, який є полем символьного типу, створеним на основі класу CharField, обмеженим довжиною в 255 символів.

Клас MusicList є класом моделі переліку музичних композицій, що організований наслідуванням базового класу моделі Model і включає наступні атрибути, що відповідають структурі таблиці:

- moment – поле дати, створене на основі класу DateField;
- listener – поле зовнішнього ключа, створене на основі класу ForeignKey для зв'язку з моделлю Listener;
- name – поле символьного типу, створене на основі класу CharField, обмежене довжиною в 255 символів;
- privatelevel – поле логічного типу, створене на основі класу BooleanField;
- tags – поле зв'язку «багато до багатьох», створене на основі класу ManyToManyField, визначає зв'язок з моделлю Tag і зберігає перелік всіх тегів переліку музичних композицій.

Клас MusicListComposition є класом моделі деталізації складу переліку музичних композицій, що організований наслідуванням базового класу моделі Model і включає наступні атрибути, що відповідають структурі таблиці:

- comment – поле символьного типу, створене на основі класу CharField, обмежене довжиною в 255 символів;

- `musiccomposition` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `MusicComposition`;
- `musiclist` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `MusicList`.

Клас `CountryStatistics` є класом статистики прослуховування музичних композицій у певній країні, що організований наслідуванням базового класу моделі `Model` і включає наступні атрибути, що відповідають структурі таблиці:

- `listens` – цілочисельне поле, створене на основі класу `IntegerField`;
- `likes` – цілочисельне поле, створене на основі класу `IntegerField`;
- `musiccomposition` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `MusicComposition`;
- `country` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `Country`.

Клас містить метод для визначення рядкового представлення об'єкта статистики прослуховування музичних композицій у країні. Також клас містить метод створення прямих посилань на об'єкт `get_absolute_url`.

Клас `GenreStatistics` є класом статистики прослуховування жанрів музичних композицій у певній країні, що організований наслідуванням базового класу моделі `Model` і включає наступні атрибути:

- `listens` – цілочисельне поле, створене на основі класу `IntegerField`;
- `likes` – цілочисельне поле, створене на основі класу `IntegerField`;
- `genre` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `Genre`;
- `country` – поле зовнішнього ключа, створене на основі класу `ForeignKey` для зв'язку з моделлю `Country`.

Клас містить метод для визначення рядкового представлення об'єкта статистики прослуховування жанрів музичних композицій у країні. Також клас містить метод створення прямих посилань на об'єкт `get_absolute_url`.

4.3 Висновки за розділом 4

У четвертому розділі описано логіку роботи програми на основі виділення меню для вибору функцій та розділення доступу авторизованих і неавторизованих користувачів до функцій програми, описано створені класи програмної системи, їх призначення та основні принципи створення і роботи цих класів.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для відтворення музичних композицій з обмеженого (нішевого) каталогу з наданням можливостей підбору музики з цього каталогу для розширення простору відтворення музики. Програма повинна використовуватися авторизованими і неавторизованими користувачами без надання останнім можливості відтворювати музичні композиції.

5.2 Умови виконання програми

Для початкової роботи програми на сервері потрібно забезпечити наступні вимоги:

- чотириядерний процесор з частотою від 2 ГГц;
- мінімальний обсяг оперативної пам'яті – 8 Гб;
- вільний простір у 10 Гб на жорсткому диску.

Умовами експлуатації є запуск вебсервера, сервера БД на основі MySQL, встановлений інтерпретатор мови програмування Python.

5.3 Робота з програмою

Робота з програмою реалізується через стандартну адміністративну панель для адміністратора системи і через вебінтерфейс, який надає варіанти роботи для слухача і для неавторизованого користувача.

Для неавторизованого користувача одним з базових елементів роботи є перегляд нових композицій (рис. 5.1). На цій сторінці можна також перемкнутися на перелік композицій, які найбільше уподобали слухачі, які

найбільше прослухали слухачі або які викликали найбільшу антипатію у слухачів.

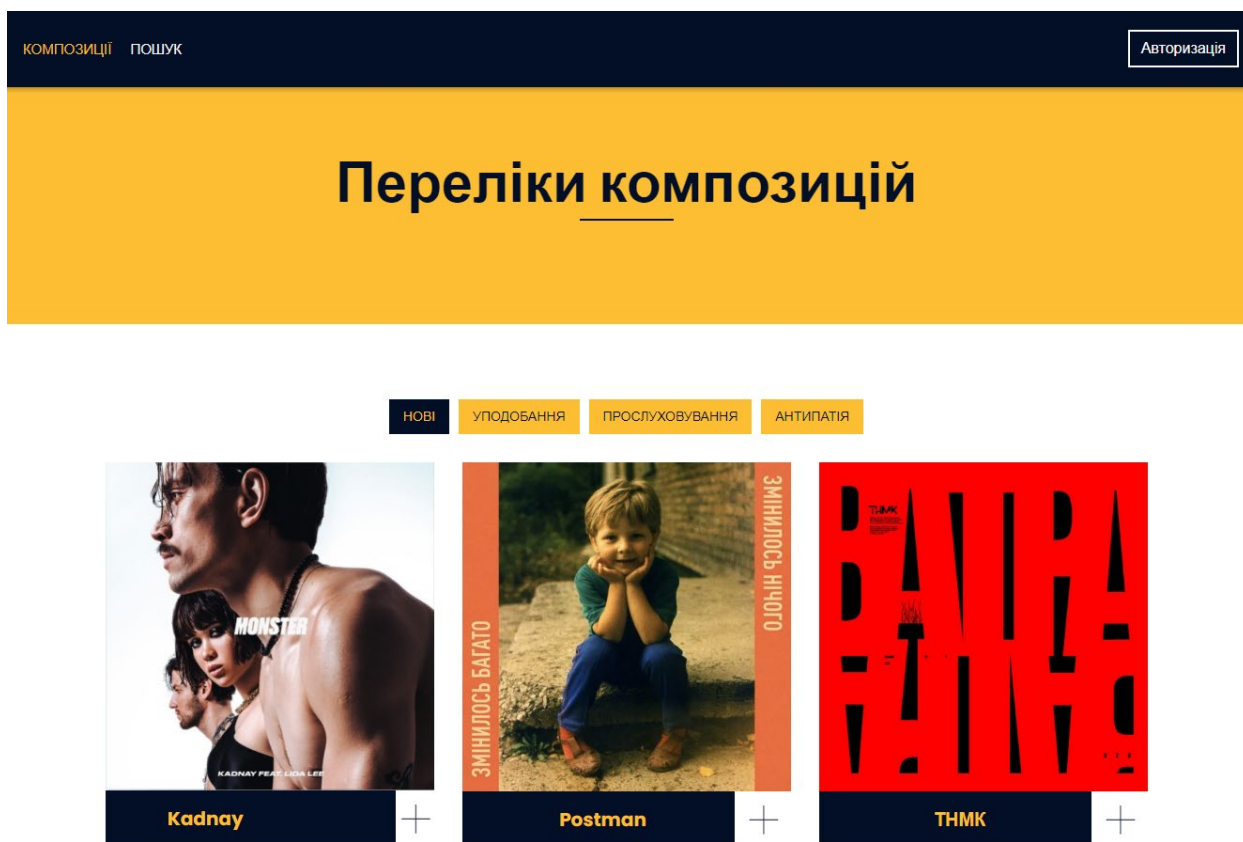
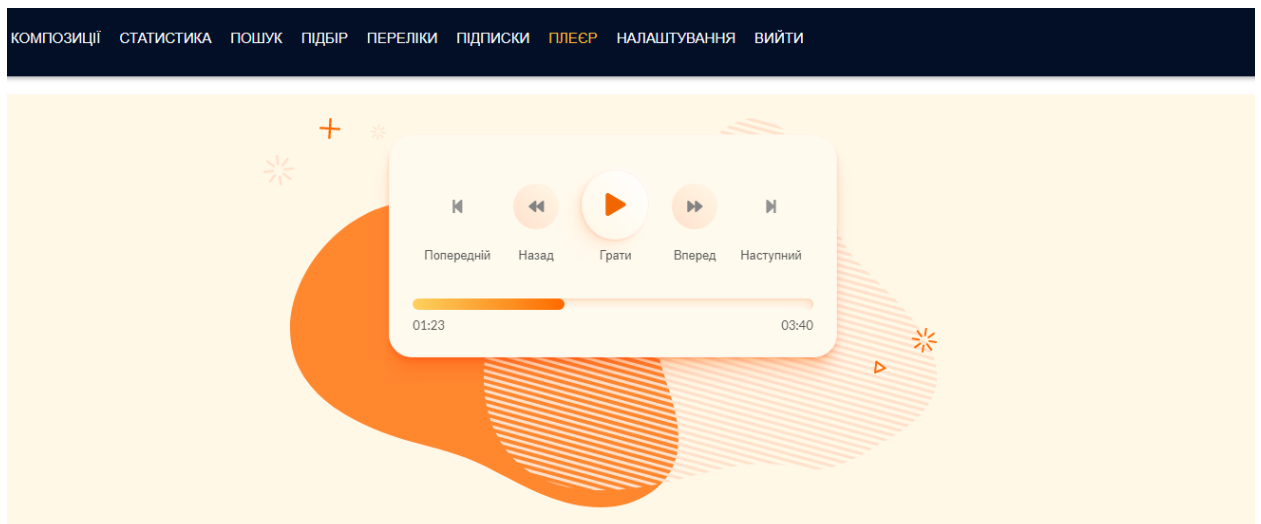


Рисунок 5.1 – Перелік нових композицій у програмі

Кожну композицію можна відтворити, натиснувши на зображення плюса. Можна перейти на окрему сторінку цієї композиції, натиснувши на її назву. Після додавання композиції до відтворення, вона одразу починає програватися. У даному випадку неавторизований користувач не може виконати цю дію, при спробі відбудеться перенаправлення на сторінку авторизації.

Для авторизованого користувача, слухача, доступний весь набір функцій. Зокрема він може програвати композиції (рис. 5.2), отримувати доступ до плеєра з переїком композицій (для цього призначений пункт меню «Плеєр» меню). Звідси можна відтворювати композицію, вилучити з переліку композицій непотрібні.



Перелік композицій

Jamala - Новий рік з тобою	+
Kadnay - Monster	+
Archive - Super8	+

Рисунок 5.2 – Відтворення музики у програмі

Окрім доступу до плеєра слухач отримає можливість так само як і неавторизований слухач отримувати доступ до переліку композицій, до пошуку, звідки можна обрати, пошук чого саме виконується (у випадкаючому переліку задати виконавця, альбом або композицію).

Статистика прослуховування композицій, зокрема за країнами, доступна з пункту меню «Статистика».

Переліки композицій, створені слухачем, з можливістю пошуку за тегами доступні через пункт меню «Переліки».

Виконавці, на які підписався слухач, доступні через пункт меню «Підписки». Щоб підписатися на виконавця, потрібно зайти на його сторінку, звідти доступна відповідна кнопка.

Налаштування облікового запису доступні через пункт меню «Налаштування».

Пункт меню «Підбір» реалізує доступ до використання запропонованого методу, що і є основною особливістю роботи. На цій

сторінці потрібно спочатку натиснути на зображення мікрофона, записати повідомлення, а далі відкриється наступні сторінка з результатами підбору композицій (рис. 5.3). На ній доступна можливість відтворити потрібну композицію. Додати композицію в улюблені або навпаки висловити антипатію можна зі сторінки, яка відкривається натисканням на назву внизу під зображенням (відображається назва виконавця, а при наведенні відображається поверх зображення назва виконавця та назва композиції).

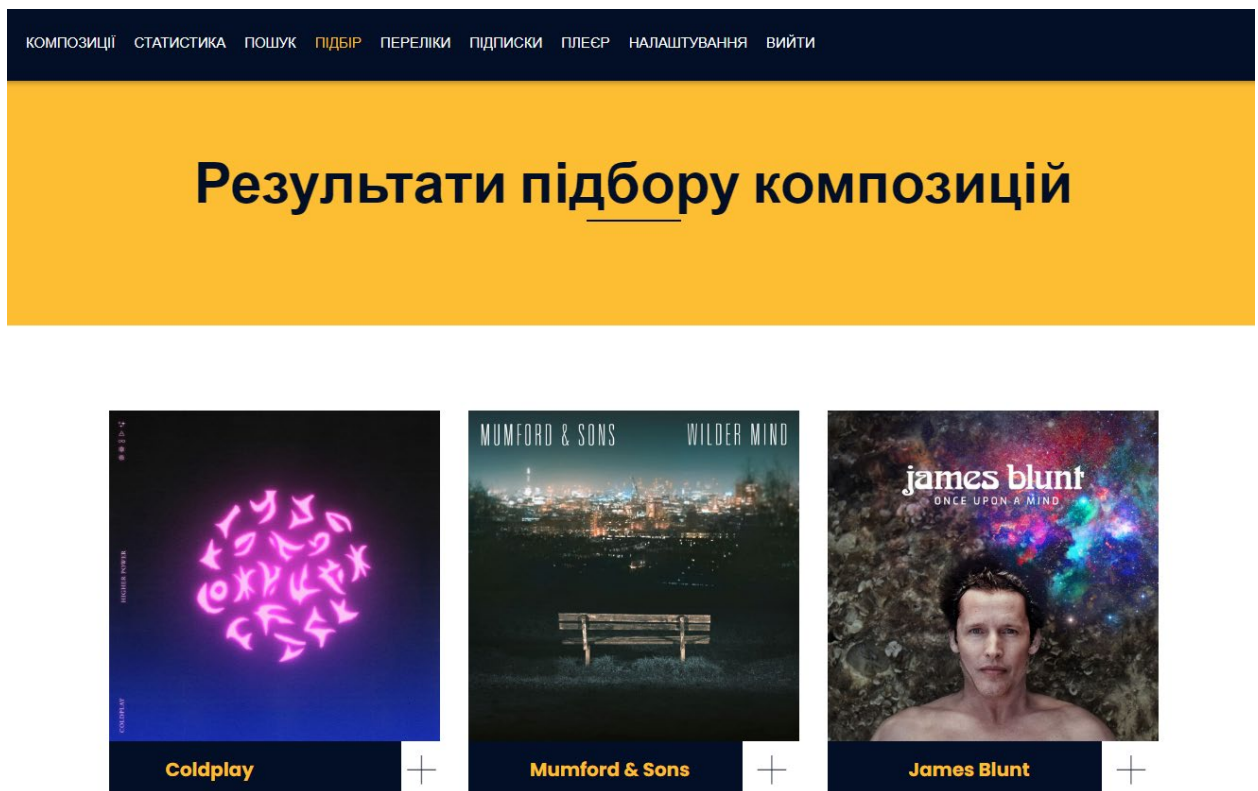


Рисунок 5.3 – Перелік підібраних композицій у програмі

5.4 Експериментальне дослідження

Експериментальне дослідження проводилось щодо розв'язання задачі класифікації голосових повідомлень за типом емоцій. Тому для підтримки роботи запропонованого методу використовувалось дві вибірки: вибірка даних, що характеризують типи емоцій, визначені для композицій, представлені вибіркою [15], та вибірка даних, що характеризують типи

емоцій для голосових повідомлень. Останню вибірку було сформовано на основі даних проєкту [16].

Представлений у розділі 2 метод було порівняно з методом опорних векторів. Результати експериментального дослідження щодо точності класифікації голосових повідомлень за типом емоцій представлені в табл. 5.1.

Таблиця 5.1 – Результати дослідження

Метод	Точність
Метод опорних векторів	0,67
Представлений метод	0,83

5.5 Висновки за розділом 5

У даному розділі розглянуто виконання основних функцій програми з представленням відповідних інтерфейсів користувача, проведено експериментальне дослідження проблеми класифікації голосових повідомлень за типом емоцій.

ВИСНОВКИ

Виконаний аналіз програмних аналогів, включаючи зокрема Spotify, YouTube Music, Chosic, дозволив сформулювати ідею ПЗ, виділити його основні функціональні особливості. На відміну від великих потокових сервісів передбачається використання розробки для відтворення музики на основі нішевого каталогу для забезпечення діяльності окремої компанії підтримки інтересів правовласників.

Виконано аналіз інтелектуальних методів, які можуть бути використані для розв'язання проблеми, і запропоновано власний метод підбору музики.

Визначено мову Python для розробки програмного застосунку, виконано порівняння фреймворків розробки вебзастосунків для Python – Django та Flask. Для реалізації вебзастосунку було обрано Django. Виконано проєктування роботи з даними. Визначено схему структури програми з розбиттям програмної системи на дві підсистеми.

Розроблено програмну систему, що призначена для відтворення музичних композицій з обмеженого каталогу з наданням можливостей підбору музики з цього каталогу для розширення простору її відтворення. Програма надає інформацію про статистику прослуховування музики за країнами, про найкращі композиції, виділеними за статистикою відтворення музики слухачами, дозволяє формувати власні переліки композицій, альбомів у різному вигляді.

Усі поставлені завдання на роботу виконано.

Наукова новизна роботи полягає в запропонованому методі підбору музики, заснованому на розбитті даного процесу на етапи виділення емоцій за голосом людини та підбору музичних композицій за виділеними емоціями, що дозволяє формувати перелік композицій для прослуховування з різним ступенем новизни в залежності від типу емоцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Speakman, K. Music Streaming Subscriptions Surpass 80 Million Amid Streaming Boom, RIAA Reports [Electronic resource] / K. Speakman. – Access mode : <https://www.forbes.com/sites/kimberleespeakman/2021/09/13/music-streaming-subscriptions-hit-46-million-amid-streaming-boom-riaa-reports/?sh=337145d91fd8>.
2. Musical trends and predictability of success in contemporary songs in and out of the top charts [Electronic resource] / Myra Interiano, Kamyar Kazemi, Lijia Wang et al. – Access mode : <https://royalsocietypublishing.org/doi/10.1098/rsos.171274>.
3. Saragih, H. S. Predicting which song will be popular, can we do that? [Electronic resource] / H. S. Saragih. – Access mode : <https://www.thefineryreport.com/opinionarticles/predicting-which-song-will-be-popular-can-we-do-that>.
4. Listening is everything. Spotify [Electronic resource]. – Access mode : <https://www.spotify.com/ua-en/>.
5. O’Boyle, B. Which is the best music streaming service in the UK? [Electronic resource] / B. O’Boyle. – Access mode : <https://www.pocket-lint.com/apps/buyers-guides/126892-best-music-streaming-services-apps-and-sites-spotify-vs-apple-music-and-more>.
6. Casserly, M. The best music streaming services 2021 [Electronic resource] / M. Casserly. – Access mode : <https://www.techadvisor.com/feature/entertainment/best-music-streaming-services-3695632/>.
7. Pendlebury T. Best music streaming service for 2021 [Electronic resource] / T. Pendlebury. – Access mode : <https://www.cnet.com/tech/services-and-software/best-music-streaming-service/>.
8. YouTube Music [Electronic resource]. – Access mode : <https://music.youtube.com/>.
9. Discover New Music – Chosic [Electronic resource]. – Access mode :

<https://www.chosic.com/>.

10. What are Convolutional Neural Networks? IBM [Electronic resource]. – Access mode : <https://www.ibm.com/id-en/topics/convolutional-neural-networks>.

11. Goodfellow, I. Deep Learning [Text] / Ian Goodfellow, Yoshua Bengio, Aaron Courville. – Cambridge : MIT Press, 2016. – 326 p.

12. Васильєв О. Програмування мовою Python . [Текст] /О. Васильєв, Київ. : Богдан-видавництво, 2019. – 508 с.

13. The web framework for perfectionists with deadlines. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/>.

14. Builderz Free Website Template. Free CSS Templates. Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page266/builderz>.

15. Spotify Music data to identify the moods. Kaggle [Electronic resource]. – Access mode : <https://www.kaggle.com/musicblogger/spotify-music-data-to-identify-the-moods>.

16. IEMOCAP Database [Electronic resource]. – Access mode : <https://sail.usc.edu/iemocap/>.

ДОДАТОК А
Технічне завдання

Вступ

На даний момент програмні засоби, які забезпечують потокове відтворення музики, направлені на глобалізацію процесів, тому на них представлені композиції різних компаній-правовласників з різних країн. Проте раз за разом виникають випадки, коли правовласників не влаштовують ті умови, які надаються такими сервісами. Тоді вони мають необхідність розірвання договорів і використання інших програмних продуктів для відтворення своєї музики. Саме дана задача і розв'язується в роботі. ПЗ має враховувати необхідність відтворення деякого нішевого каталогу, тобто каталогу певної компанії-правовласника. Для цього окрім засобів відтворення мають надаватися сучасні засоби підбору музики.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів підбору музики», яку затверджено наказом № 435 від 5 листопада 2022 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для відтворення музичних композицій з обмеженого (нішевого) каталогу з наданням можливостей підбору музики з цього каталогу для розширення простору відтворення музики. Програма повинна використовуватися авторизованими і неавторизованими користувачами без надання останнім можливості відтворювати музичні композиції.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Функціональні характеристики, що мають надаватися програмою:

- відтворення музичних композицій;
- пошук музичних композицій за назвою;
- пошук альбомів музичних композицій;
- формування переліку нових композицій, які викликають інтерес у інших слухачів;
- створення власних переліків музичних композицій і подальше їх редагування;
- знаходження переліків музичних композицій, створених іншими слухачами;
- підбір музичних композицій за настроєм;
- вподобання музичної композиції;
- зазначення антипатії до музичної композиції;
- перегляд переліку композицій з музичного альбому;
- перегляд переліку альбомів за музичним виконавцем;
- підписування на музичного виконавця;
- формування переліку нових композицій серед виконавців, на які підписаний слухач;
- додавання музичних альбомів до власної бібліотеки;
- формування переліку найкращих композицій за рік;
- формування переліку найкращих композицій за країнами;
- формування переліку жанрів, які найбільш часто прослуховуються у країні;
- формування переліку композицій виконавця з найбільшою увагою слухачів;
- керування музичними виконавцями, альбомами та композиціями;
- авторизація слухача і адміністратора системи;

– реєстрація слухача.

A.3.2 Вимоги до складу та параметрів технічних засобів

Для початкової роботи програми на сервері потрібно забезпечити наступні вимоги:

- чотириядерний процесор з частотою від 2 ГГц;
- мінімальний обсяг оперативної пам'яті – 8 Гб;
- вільний простір у 10 Гб на жорсткому диску.

A.3.3 Умови експлуатації

Умовами експлуатації є запуск вебсервера, сервера БД, встановлений інтерпретатор мови програмування Python.

A.3.4 Вимоги до транспортування та збереження

Програма має розповсюджуватися через вебсервер, тому передбачається виділення сервера і віддалене розташування програми на ньому без необхідності фізичного розповсюдження копій програми.

A.4 Порядок контролю та приймання

Розробка реалізується під час виконання кваліфікаційної роботи магістра, тому потребує дотримання строків розробки у відповідності з завданням і календарним планом, визначеними на роботу.

ДОДАТОК Б
Текст програми

Б.1 Текст файла /views/musiccompositionviews.py

```

from django.views.generic.detail import DetailView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView
from django.views.generic.edit import UpdateView
import musicadviserapp.models
from datetime import datetime, timedelta

class MusicCompositionView (DetailView):

    model = MusicComposition
    template_name = 'music.html'

    def get_context_data(self, **kwargs):
        musiccontext = super().get_context_data(**kwargs)
        muscomp =
MusicComposition.objects.get(id=self.kwargs['pk'])
        musiccontext ['mc'] = muscomp
        complikes = Like.objects.filter(musiccomposition_id
= self.kwargs['pk'])
        return musiccontext

class AddMusicCompositionView (LoginRequiredMixin,
CreateView):

    template_name = 'addmusic.html'
    model = MusicComposition

    fields = [
        'title',
        'photo',

```

```

        'musictext',
        'stream',
        'genres',
        'musicauthors'
    ]

```

```

login_url = 'loginview'
success_url = 'musiccompositionview'

```

```

def form_valid(self, form):
    form.likes = 0
    form.dislikes = 0
    form.listens = 0
    return super().form_valid(form)

```

```

class UpdateMusicCompositionView (LoginRequiredMixin,
UpdateView):

```

```

    template_name = 'updatemusic.html'
    model = MusicComposition

```

```

    fields = [
        'title',
        'photo',
        'musicalbum',
        'musictext',
        'stream',
        'genres',
        'musicauthors'
    ]

```

```

login_url = 'loginview'

```

```

success_url = 'musiccompositionview'

class MostListenedAuthorMusicCompositionsView (ListView):

    model = MusicComposition
    template_name = 'bestlistenedauthormusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        author = self.request.GET.get('author')
        compos =
MusicComposition.objects.filter(authors__id = <author>)
        compos = compos.order_by('-listens')
        musiccontext ['music'] = compos [0:10]
        return musiccontext

class AlbumMusicView (ListView):

    model = MusicComposition
    template_name = 'albummusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        alb = self.request.GET.get('album')
        compos =
MusicComposition.objects.filter(musicalbum_id = alb)
        compos = compos.order_by('id')
        musiccontext ['music'] = compos
        return musiccontext

```

```

class MostAnticipatedAuthorMusicCompositionsView
(ListView):

    model = MusicComposition
    template_name = 'bestdislikedauthormusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        author = self.request.GET.get('author')
        compos =
MusicComposition.objects.filter(authors__id = <author>)
        compos = compos.order_by('-dislikes')
        musiccontext ['music'] = compos [0:10]
        return musiccontext

class MostLikedAuthorMusicCompositionsView (ListView):

    model = MusicComposition
    template_name = 'bestlikedauthormusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        author = self.request.GET.get('author')
        compos =
MusicComposition.objects.filter(authors__id = <author>)
        compos = compos.order_by('-likes')
        musiccontext ['music'] = compos [0:10]
        return musiccontext

```

```
class YearPopularMusicCompositionsView (LoginRequiredMixin,
ListView):
```

```
    model = MusicComposition
    template_name = 'yearpopularnewmusic.html'
    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        endnewdate = datetime.today() - timedelta(years=1)
        compos =
MusicComposition.objects.filter(moment__gte = endnewdate)
        compos = compos.order_by('-likes')
        musiccontext ['music'] = compos [0:100]
        return musiccontext
```

```
class MostListenedMusicCompositionsView ( ListView):
```

```
    model = MusicComposition
    template_name = 'bestlistenedmusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        compos = MusicComposition.objects.all()
        compos = compos.order_by('-listens')
        musiccontext ['music'] = compos [0:100]
        return musiccontext
```

```
class MostAnticipatedMusicCompositionsView (ListView):
```

```
    model = MusicComposition
    template_name = 'bestdislikedmusic.html'
```

```

def get_queryset(self, *args, **kwargs):
    musiccontext = super().get_queryset(*args,
**kwargs)

    compos = MusicComposition.objects.all()
    compos = compos.order_by('-dislikes')
    musiccontext ['music'] = compos [0:100]
    return musiccontext

```

```

class MostLikedMusicCompositionsView (ListView):

```

```

    model = MusicComposition
    template_name = 'bestlikedmusic.html'

```

```

def get_queryset(self, *args, **kwargs):
    musiccontext = super().get_queryset(*args,
**kwargs)

    compos = MusicComposition.objects.all()
    compos = compos.order_by('-likes')
    musiccontext ['music'] = compos [0:100]
    return musiccontext

```

```

class MostLikedCountryMusicCompositionsView (ListView):

```

```

    model = MusicComposition
    template_name = 'bestlikedcountrymusic.html'

```

```

def get_queryset(self, *args, **kwargs):
    musiccontext = super().get_queryset(*args,
**kwargs)

    country = self.request.GET.get('country')

```

```

        stat = CountryStatistics.objects.filter(country__id
= countryp)
        stat = stat.order_by('-likes')
        stat = stat [0:100]
        mus = {}
        for music in stat:
            comp = MusicComposition.objects.get(id =
music.musiccomposition.id)
            mus.add(comp)
        musiccontext ['music'] = mus
        return musiccontext

```

```

class MostDisLikedCountryMusicCompositionsView (ListView):

    model = MusicComposition
    template_name = 'bestdislikedcountrymusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        countryp = self.request.GET.get('country')
        stat = CountryStatistics.objects.filter(country__id
= countryp)
        stat = stat.order_by('-dislikes')
        stat = stat [0:100]
        mus = {}
        for music in stat:
            comp = MusicComposition.objects.get(id =
music.musiccomposition.id)
            mus.add(comp)
        musiccontext ['music'] = mus
        return musiccontext

```

```

class MostListenedCountryMusicCompositionsView (ListView):

    model = MusicComposition
    template_name = 'bestlistenedcountrymusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        countryp = self.request.GET.get('country')
        stat = CountryStatistics.objects.filter(country__id
= countryp)

        stat = stat.order_by('-listenes')
        stat = stat [0:100]
        mus = {}
        for music in stat:
            comp = MusicComposition.objects.get(id =
music.musiccomposition.id)
            mus.add(comp)
        musiccontext ['music'] = mus
        return musiccontext

class NewPopularMusicCompositionsView (ListView):

    model = MusicComposition
    template_name = 'popularnewmusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        endnewdate = datetime.today() - timedelta(days=7)

```

```

        compos =
MusicComposition.objects.filter(moment__gte = endnewdate)
        compos = compos.order_by('-likes')
        musiccontext ['music'] = compos [0:100]
        return musiccontext

```

```

class NewPopularSubscribedMusicCompositionsView
(LoginRequiredMixin, ListView):

```

```

        model = MusicComposition
        template_name = 'popularsubsmusic.html'

        def get_queryset(self, *args, **kwargs):
            musiccontext = super().get_queryset(*args,
**kwargs)

            endnewdate = datetime.today() - timedelta(days=7)
            compos =
MusicComposition.objects.filter(moment__gte = endnewdate)
            compos = compos.order_by('-likes')
            mus = {}
            for music in compos:
                ms = music.musicauthors
                sub = Subscribe.objects.filter(listener_id =
self.request.user.id)
                flag = False
                for s in sub:
                    if s.musicauthor in ms:
                        flag = True
            if flag:
                mus.add(music)
            if mus.count() == 100:
                break

```

```

musiccontext ['music'] = mus
return musiccontext

```

```

class ListenerLikedMusicCompositionsView
(LoginRequiredMixin, ListView):

    model = MusicComposition
    template_name = 'likedmusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        listener = self.request.user
        lis = Listener.objects.get(id = listener.id)
        likes = Like.objects.filter(listener = lis)
        likes = likes.order_by('-id')
        mus = {}
        for like in likes:
            m = MusicComposition.get(id =
like.musiccomposition)
            mus.add(m)
        musiccontext ['music'] = compos [0:100]
        return musiccontext

class MostListenedAuthorMusicCompositionsView (ListView):

    model = MusicComposition
    template_name = 'bestlistenedauthormusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

```

```

        author = self.request.GET.get('author')
        compos =
MusicComposition.objects.filter(musicauthors__id = <author>)
        compos = compos.order_by('-listens')
        musiccontext ['music'] = compos [0:10]
        return musiccontext

class SearchMusicCompositionsView (ListView):

    model = MusicComposition
    template_name = 'searchmusic.html'
    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        searchtitle = self.request.GET.get('title')
        compos =
MusicComposition.objects.filter(title__contains = searchtitle)
        musiccontext ['music'] = compos
        return musiccontext

class AuthorCompositionsView (ListView):

    model = MusicComposition
    template_name = 'authormusic.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        auth = self.request.GET.get('author')
        compos =
MusicComposition.objects.filter(musicauthors__id = <auth>)
        musiccontext ['music'] = compos

```

```
return musiccontext
```

Б.2 Текст файла /models/models.py

```
from django.contrib.auth.models import AbstractUser
from django.db import models
from django.urls import reverse
from .managers import CustomUserManager

class MusicAuthor (models.Model):
    artist = models.CharField(max_length=255)
    bandflag = models.BooleanField()
    story = models.CharField(max_length=255)
    site = models.CharField(max_length=255)
    photo = models.CharField(max_length=255)
    subscribers = models.IntegerField()

    def __str__(self):
        return self.artist

    def get_absolute_url(self):
        return reverse('musicauthor', args=[self.id])

class MusicAlbum (models.Model):
    name = models.CharField(max_length=255)
    moment = models.DateField()
    photo = models.CharField(max_length=255)
    inlibraries = models.IntegerField()
    musicauthors = models.ManyToManyField(MusicAuthor)

    def __str__(self):
        return self.name
```

```

def get_absolute_url(self):
    return reverse('musicalbum', args=[self.id])

class MusicComposition (models.Model):
    title = models.CharField(max_length=255)
    photo = models.CharField(max_length=255)
    musicalbum = models.ForeignKey (MusicAlbum,
on_delete=models.CASCADE)
    moment = models.DateField()
    musictext = models.TextField()
    listens = models.IntegerField()
    likes = models.IntegerField()
    dislikes = models.IntegerField()
    stream = models.CharField(max_length=255)
    genres = models.ManyToManyField(Genre)
    musicauthors = models.ManyToManyField(MusicAuthor)

def __str__(self):
    return self.title

def get_absolute_url(self):
    return reverse('musiccomposition', args=[self.id])

class Subscribe (models.Model):
    musicauthor = models.ForeignKey (MusicAuthor,
on_delete=models.CASCADE)
    listener = models.ForeignKey (Listener,
on_delete=models.CASCADE)
    moment = models.DateTimeField()

class Library (models.Model):

```

```

        musicalbum = models.ForeignKey (MusicAlbum,
on_delete=models.CASCADE)
        listener = models.ForeignKey (Listener,
on_delete=models.CASCADE)
        moment = models.DateTimeField()
        listens = models.IntegerField()

class Listener (AbstractUser):
    birthdate = models.DateField()
    email = models.CharField(max_length=255)
    photo = models.CharField(max_length=255)
    country = models.ForeignKey (Country,
on_delete=models.CASCADE)

class Like (models.Model):
    musiccomposition = models.ForeignKey (MusicComposition,
on_delete=models.CASCADE)
    listener = models.ForeignKey (Listener,
on_delete=models.CASCADE)
    type = models.BooleanField()

class Listen (models.Model):
    musiccomposition = models.ForeignKey (MusicComposition,
on_delete=models.CASCADE)
    listener = models.ForeignKey (Listener,
on_delete=models.CASCADE)
    moment = models.DateTimeField()

class Genre (models.Model):
    name = models.CharField(max_length=255)

class MusicList (models.Model):

```

```

        listener = models.ForeignKey (Listener,
on_delete=models.CASCADE)
        name = models.CharField(max_length=255)
        privatelevel = models.BooleanField()
        tags = models.ManyToManyField(Tag)

class MusicListComposition (models.Model):
    musiccomposition = models.ForeignKey (MusicComposition,
on_delete=models.CASCADE)
    musiclist = models.ForeignKey (MusicList,
on_delete=models.CASCADE)
    comment = models.CharField(max_length=255)

class Country (models.Model):
    name = models.CharField(max_length=100)

class Tag (models.Model):
    name = models.CharField(max_length=30)

class CountryStatistics (models.Model):
    musiccomposition = models.ForeignKey (MusicComposition,
on_delete=models.CASCADE)
    country = models.ForeignKey (Country,
on_delete=models.CASCADE)
    listens = models.IntegerField()
    likes = models.IntegerField()

    def __str__(self):
        return self.country.name + ' ' +
self.musiccomposition.title + ' - ' + self.listens

    def get_absolute_url(self):

```

```

        return reverse('genrestatistics', args=[self.id])

class GenreStatistics (models.Model):
    genre = models.ForeignKey (Genre,
on_delete=models.CASCADE)
    country = models.ForeignKey (Country,
on_delete=models.CASCADE)
    listens = models.IntegerField()
    likes = models.IntegerField()

    def __str__(self):
        return self.country.name + ' ' + self.genre.name +
' - ' + self.listens
    def get_absolute_url(self):
        return reverse('genrestatistics', args=[self.id])

```

Б.3 Текст файла /views/musicalbumviews.py

```

from django.views.generic.detail import DetailView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView
from django.views.generic.edit import UpdateView
import musicadviserapp.models
from datetime import datetime, timedelta

class MusicAlbumView (DetailView):

    model = MusicAlbum
    template_name = 'musicalbum.html'

    def get_context_data(self, **kwargs):
        musiccontext = super().get_context_data(**kwargs)

```

```

        musalb =
MusicAlbum.objects.get(id=self.kwargs['pk'])
        musiccontext ['ma'] = musalb
        compositions =
MusicComposition.objects.filter(musicalbum_id =
self.kwargs['pk'])
        return musiccontext

```

```

class AddMusicAlbumView (LoginRequiredMixin, CreateView):

```

```

    template_name = 'addalbum.html'
    model = MusicAlbum

```

```

    fields = [
        'name',
        'photo',
        'moment',
        'musicauthor'
    ]

```

```

    login_url = 'loginview'
    success_url = 'musicalbumview'

```

```

    def form_valid(self, form):
        form.inlibraries = 0
        return super().form_valid(form)

```

```

class UpdateMusicAlbumView (LoginRequiredMixin,
UpdateView):

```

```

    template_name = 'updatealbum.html'
    model = MusicAlbum

```

```

fields = [
    'name',
    'photo',
    'moment',
    'musicauthor'
]

login_url = 'loginview'
success_url = 'musicalbumview'

class AuthorAlbumsView (ListView):
    model = MusicAlbum
    template_name = 'authoralbums.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        author = self.request.GET.get('author')
        albums = MusicAlbum.objects.filter(musicauthors__id
= <author>)

        albums = albums.order_by('-moment')
        musiccontext ['albums'] = albums
        return musiccontext

class NewAlbumsView (ListView):

    model = MusicAlbum
    template_name = 'newalbums.html'

    def get_queryset(self, *args, **kwargs):

```

```

        musiccontext = super().get_queryset(*args,
**kwargs)

        endnewdate = datetime.today() - timedelta(days=7)
        albums = MusicAlbum.objects.filter(moment__gte =
endnewdate)

        albums = albums.order_by('-inlibraries')
        musiccontext ['albums'] = albums
        return musiccontext

class SearchMusicAlbumsView (ListView):

    model = MusicAlbum
    template_name = 'searchalbum.html'

    def get_queryset(self, *args, **kwargs):
        musiccontext = super().get_queryset(*args,
**kwargs)

        searchtitle = self.request.GET.get('title')
        albums = MusicAlbum.objects.filter(title__contains
= searchtitle)

        musiccontext ['music'] = albums
        return musiccontext

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
МЕТОДІВ ПІДБОРУ МУЗИКИ
Кваліфікаційна робота магістра

Виконав студент групи КНТ-111м

Д.О. Хабло

Керівник: к.т.н., доцент

С.К. Корнієнко

2022

Рисунок В.1 – Слайд 1

Об'єкт, предмет, мета роботи

Об'єкт роботи – процес пошуку нової музики.

Предмет роботи – методи підбору музики.

Мета роботи – розробити програмне забезпечення підбору музики для забезпечення потреб слухачів щодо пошуку нової музики.

Завдання роботи:

- виконати аналіз предметної області;
- виконати дослідження методів підбору музики;
- виконати проектування програмного забезпечення;
- виконати реалізацію програмного забезпечення.

2

Рисунок В.2 – Слайд 2

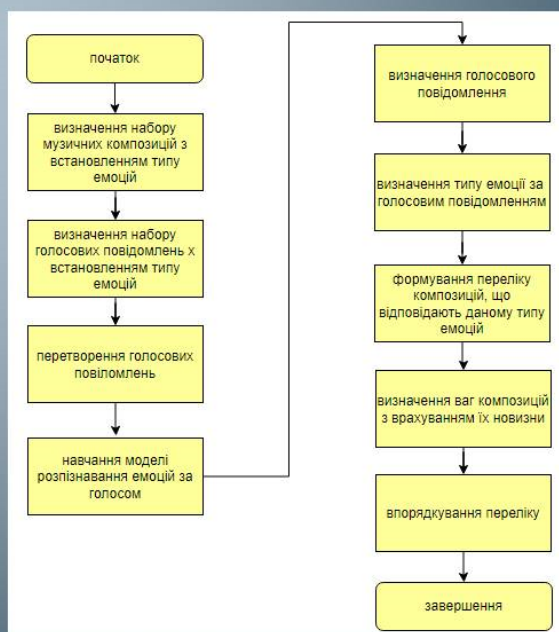
Порівняння програмних аналогів

Критерій	Spotify	YouTube Music	Розробка
Платна версія	Є	Є	Немає, але можливо в подальшому
Відтворення нішевого каталогу музики	Усі власники прав в одному каталозі	Усі власники прав в одному каталозі	Так
Пошук серед каталогу для неавторизованих користувачів	Немає	Немає	Є
Відтворення музики в якості без втрат	Немає	Немає	Є
Пошук музики на основі емоцій	Немає	Немає	Є

3

Рисунок В.3 – Слайд 3

Метод підбору музики



4

Рисунок В.4 – Слайд 4

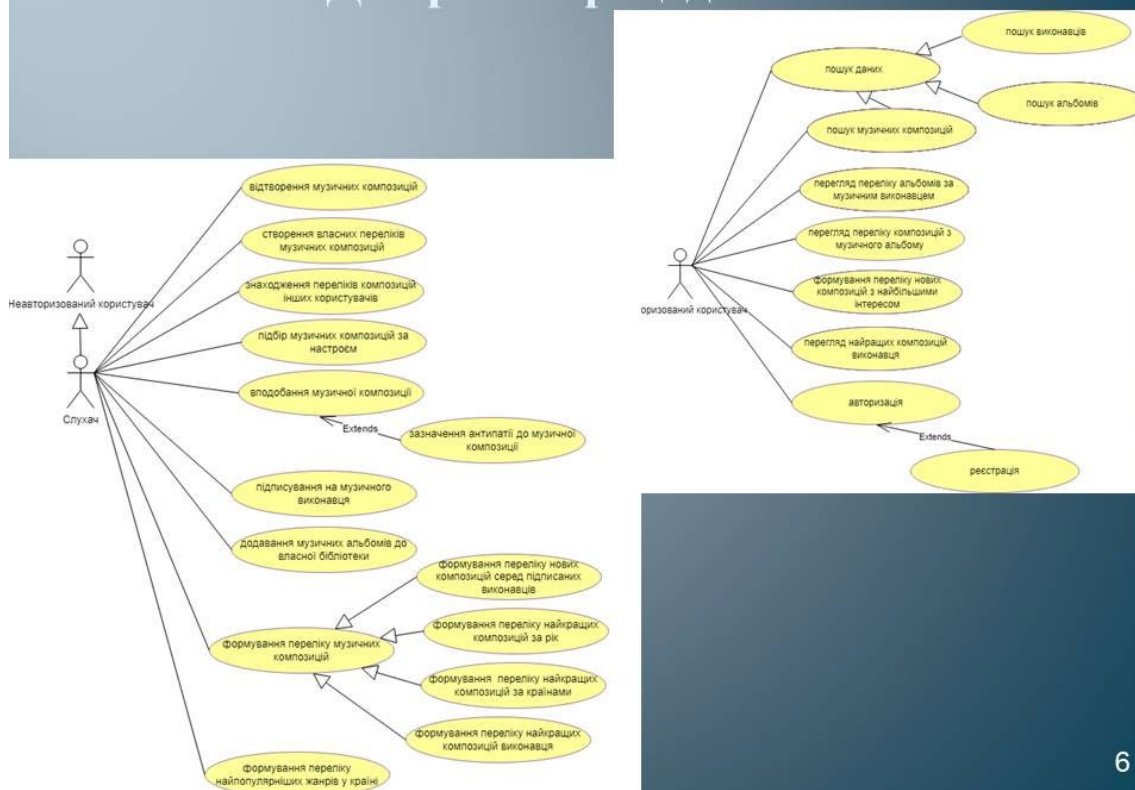
Вибір фреймворку веброзробки

Критерій	Flask	Django
Сучасні засоби безпеки	В основному не підтримуються самим фреймворком	Є
Засоби роботи над керуванням даними	Немає	Є
Спільнота розробників	Середня	Велика
Кількість вбудованих засобів	Невелика	Достатньо велика

5

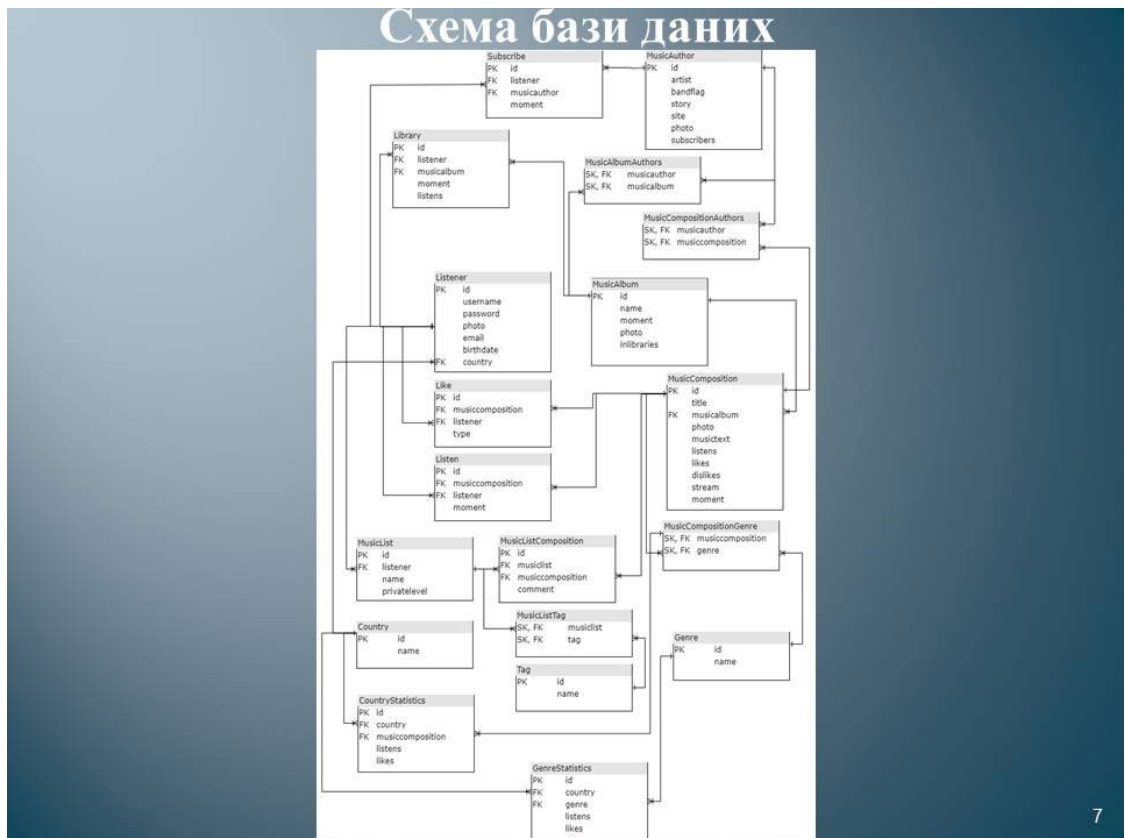
Рисунок В.5 – Слайд 5

Діаграми прецедентів



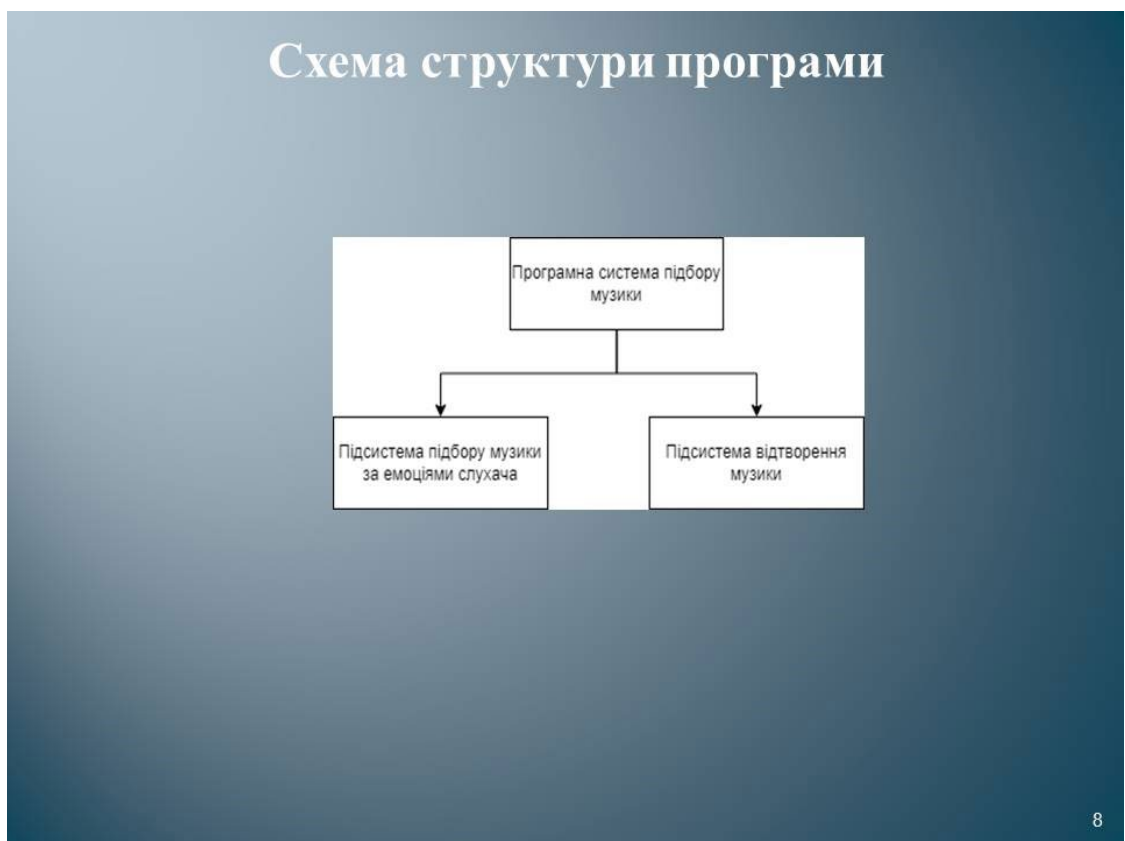
6

Рисунок В.6 – Слайд 6



7

Рисунок В.7 – Слайд 7



8

Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9

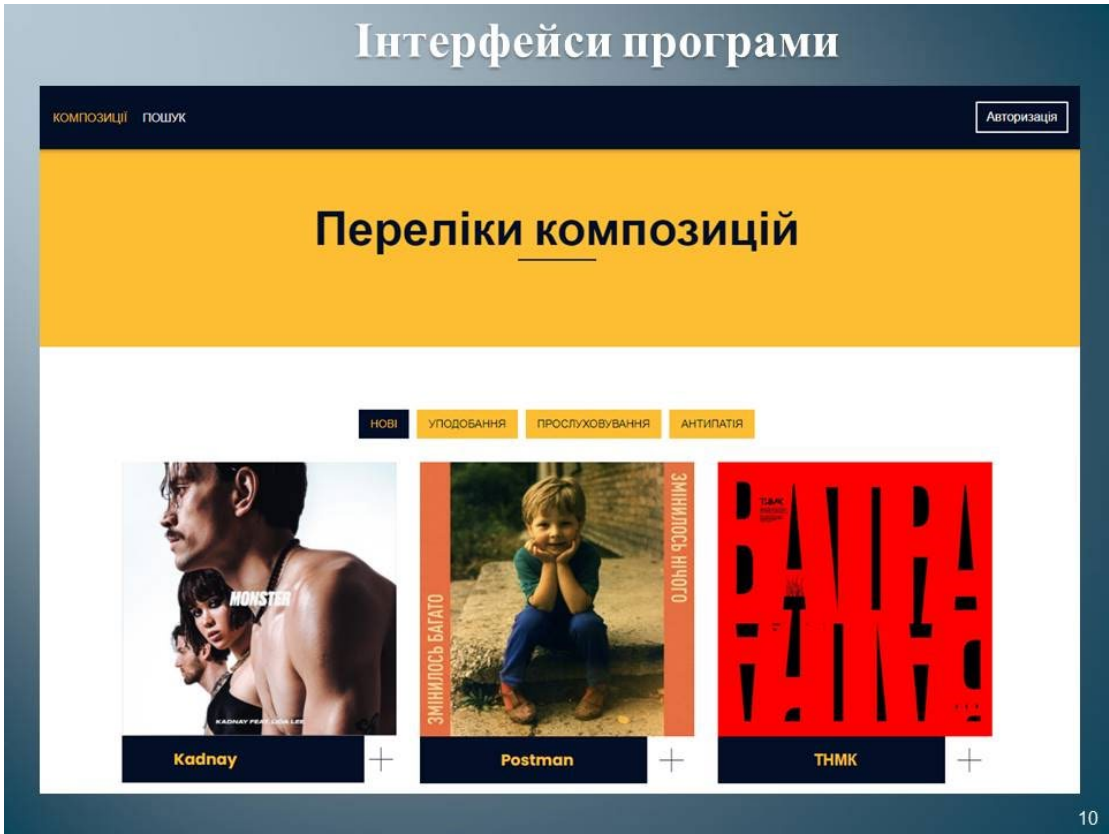


Рисунок В.10 – Слайд 10

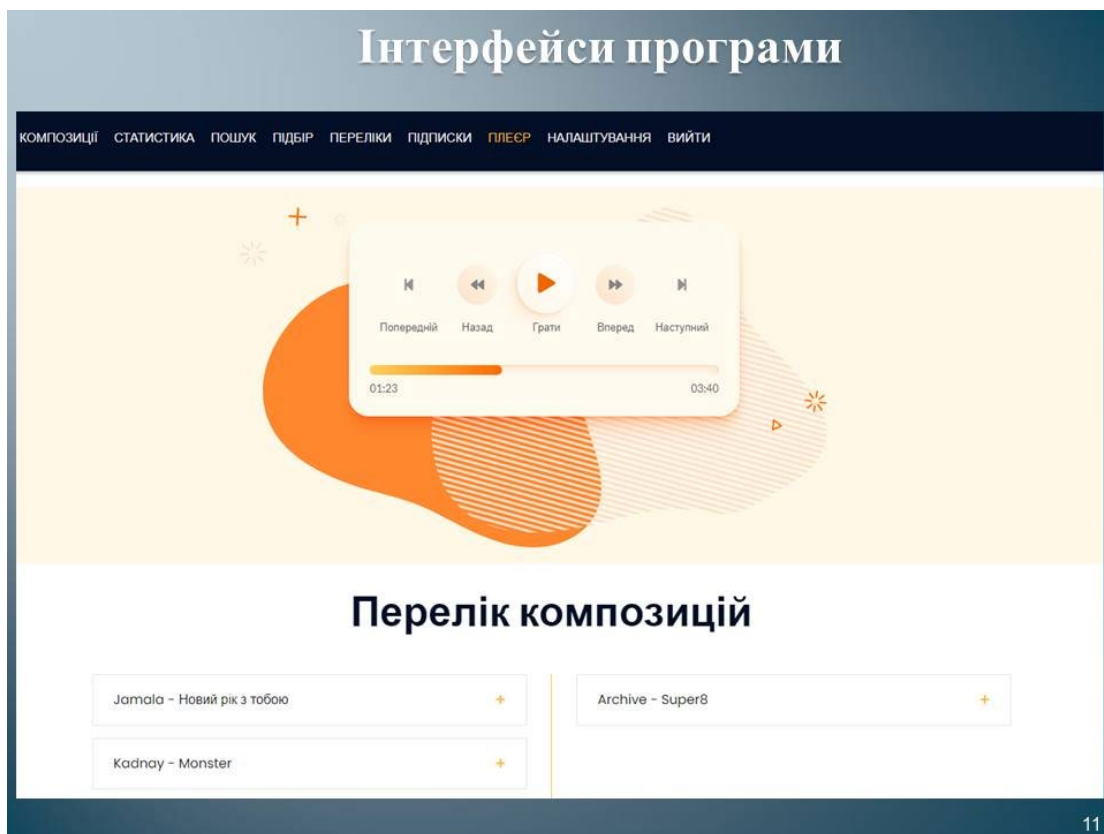


Рисунок В.11 – Слайд 11

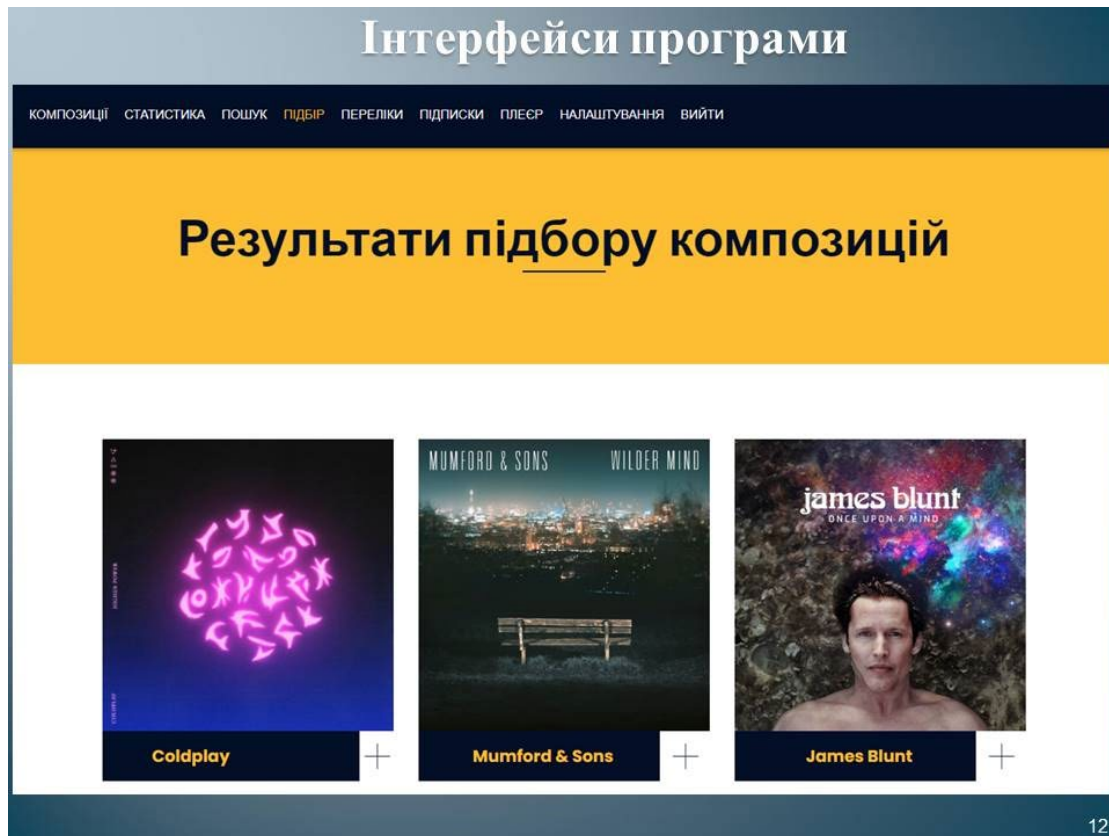


Рисунок В.12 – Слайд 12

Експериментальне дослідження точності класифікації голосових повідомлень за типом емоцій

Метод	Точність класифікації
Метод опорних векторів	0,67
Представлений метод	0,83

13

Рисунок В.13 – Слайд 13

Висновки

Розроблено програмну систему, що призначена для відтворення музичних композицій з обмеженого каталогу з наданням можливостей підбору музики з цього каталогу для розширення простору її відтворення. Програма надає інформацію про статистику прослуховування музики за країнами, про найкращі композиції, виділеними за статистикою відтворення музики слухачами, дозволяє формувати власні переліки композицій, альбомів у різному вигляді.

Наукова новизна роботи полягає в запропонованому методі підбору музики, заснованому на розбитті даного процесу на етапи виділення емоцій за голосом людини та підбору музичних композицій за виділеними емоціями, що дозволяє формувати перелік композицій для прослуховування з різним ступенем новизни в залежності від типу емоцій.

14

Рисунок В.14 – Слайд 14