КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-застосунок для підбору музичних композицій із урахуванням вподобань»

Виконав _____
(Підпис)
Скляров Андрій Олександрович
(прізвище, ім'я, по батькові)

Керівник к.т.н. Силантьєв Сергій Олексійович
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали)

(Дата)

Київ – 2021

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Веб-застосунок для підбору музичних композицій із урахуванням вподобань»

Освітня програма: Прикладне програмування
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Скляров Андрій Олександрович	

Назва роботи українською та англійською мовами

Веб-застосунок для підбору музичних композицій із урахуванням вподобань

Web application for the selection of musical compositions based on preferences

Мета бакалаврської кваліфікаційної роботи, завдання

Мета роботи: Підвищення ефективності пошуку музичних композицій

План роботи:
1. Сучасні підходи до розроблення і впровадження веб-застосунків
2. Проектування системи веб-застосунку для підбору музичних композицій
3. Тестування та програмна реалізація веб-застосунку

ПІБ, ступінь, звання наукового керівника роботи:

_____

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

оме р	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	виконано
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	05.03.2021	виконано
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	09.04.2021	виконано
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	07.05.2021	виконано
9.	Подання роботи у першому варіанті	11.05.2021	виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	виконано
3.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврську роботу)	11.06.2021	виконано
14.	Захист кваліфікаційної роботи бакалавра	23.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	2
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	13
2	16
3	18
Висновки	1
Перелік використаних джерел	3
Додатки	0

				ДП XXXX 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.				Відомість дипломної роботи	Лист	Листів
Керівн.						
Н/контр.						
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

До бакалаврської дипломної роботи Склярова Андрія Олександровича на тему «Веб-застосунок для підбору музичних композицій із урахуванням вподобань»

Дана дипломна робота присвячена створенню динамічного веб-застосунку для підбору музичних композицій із урахуванням вподобань.

Для розробки застосунку було проведено дослідження доцільності створення, проаналізовано подібні застосунки, що можуть надавати музичні рекомендації. Після дослідження було проведено порівняння існуючих систем і поставлено задачі на розробку.

Спроектовану систему виконано на фреймворці Flask. Це зумовлено його простотою, зручністю та здатністю до легкого масштабування проекту. У роботі приведені результати аналізу щодо вибору технологічного стеку та детальний опис процесу розробки застосунку. Розроблений застосунок розміщено на хмарному сервері та протестовано на предмет продуктивності, зручності використання, способів надання вподобань та доступності. Представлено інструкцію використання та аналіз результату роботи в порівнянні з існуючими аналогами.

Загальний обсяг роботи: 62 сторінки, 25 рисунків, 2 таблиці, 30 посилань.

Ключові слова: музичні композиції, веб-застосунок, вподобання, підбір музичних композицій, рекомендації.

ANNOTATION (ABSTRACT)

To the bachelor's thesis of Skliarov Andrii Oleksandrovich on the topic “Web application for the selection of musical compositions based on preferences”

This thesis is devoted to creating a dynamic web application for the selection of musical compositions based on preferences.

To develop the application, a study was conducted on the feasibility of creation, analyzed similar applications that can provide musical recommendations.

After the research, a comparison of existing systems was made and development tasks were set.

The designed system is made on the Flask framework. This is due to its simplicity, convenience and ability to easily scale the project. The paper presents the results of the analysis of the choice of technological stack and a detailed description of the application development process.

The developed application is hosted on a cloud server and tested for performance, usability, preferences and availability. The instruction of use and the analysis of result of work in comparison with existing analogues are presented.

Total volume of work: 62 pages, 25 figures, 2 tables, 30 links.

Keywords: musical compositions, web application, preferences, selection of musical compositions, recommendations.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS - JavaScript

URL - Uniform Resource Locator

SPA – Single Page Application

SQL – Structured Query Language

JIT – Just-In-Time

WSGI – Web Server Gateway Interface

BSD - Berkeley Software Distribution

W3C - World Wide Web Consortium

DOM - Document Object Model

FTP – File Transport Protocol

Amazon EC2 - Amazon Elastic Compute Cloud

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 7

ВСТУП 10

РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ ДО РОЗРОБЛЕННЯ ТА ВПРОВАДЖЕННЯ ВЕБ-ЗАСТОСУНКІВ 12

1.1 Огляд проблеми надлишкового вибору в сегменті музичного напрямку 12

1.2 Огляд існуючих застосунків та порівняння підбору музичних композицій з урахуванням вподобань 17

1.3 Оцінка вимог та постановка задачі 21

ВИСНОВКИ ПО РОЗДІЛУ 1 24

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПІДБОРУ МУЗИЧНИХ КОМПОЗИЦІЙ 25

2.1 Огляд способів вдосконалення підбору музичних композицій з урахуванням вподобань 25

2.2 Аналіз технологій для реалізації підбору музичних композицій у вигляді веб-застосунку 29

2.3 Пошук програмного рішення для підбору музичних композицій37

ВИСНОВКИ ПО РОЗДІЛУ 2 40

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ 41

3.1 Проектування та розробка веб-застосунку для підбору музичних композицій 41

3.2 Інструкція користувача 51

3.3 Тестування та аналіз отриманих результатів 55

ВИСНОВКИ ПО РОЗДІЛУ 3 59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60

ВСТУП

З розвитком інформаційних технологій створення і поширення мультимедійної інформації стало просте як ніколи. Разом з цим з’являється проблема пошуку потрібного контенту у гігантському інформаційному просторі. Інтернет наповнений настільки великою кількістю контенту, що користувач може легко в ньому заплутатись, тому є просто необхідною структуризація та кластеризація інформації. Якщо користувачу складно розібратись у тому, що пропонує інтернет-ресурс, то це не завжди означає, що в ньому занадто багато інформації – зазвичай, вона просто погано структурована. Це особливо відчувається, коли мова йдеться про музичні композиції, адже на відміну від текстової та графічної інформації, для аналізу людиною аудіоінформації потрібна безпосередня участь у процесі. Однією із причин, чому система рекомендацій знаходить популярність у сучасному суспільстві – економія часу і надання людям можливості простого та переважно якісного вибору.

Але, нажаль, жоден існуючий рекомендаційний сервіс не може повністю задовольнити потреби користувачів. Навіть штучний інтелект не завжди влучно пропонує той контент, який користувач хотів би бачити. Це зумовлено тим, що при прослуховуванні певної композиції, користувачам можуть сподобатися різні елементи звукоряду (барабани, головний мотив, вокал тощо) і підібрані рекомендації не охоплять вподобання всіх користувачів. Підвищити якість надання рекомендацій можна шляхом створення більшої кількості застосунків, що використовують різні способи підбору композицій. Таке рішення дозволить покрити більшу площу попиту на послуги рекомендацій музики і, відповідно, буде задовольняти більшу кількість користувачів. Тому створення веб-застосунку для підбору музичних композицій є **актуальним науково-прикладним завданням.**

Метою дипломної роботи є створення веб-застосунку, що спеціалізується на пропонуванні подібних композицій на основі вибору користувача та підвищення якості надання подібних композицій.

Завдання дослідження*:*

- аналіз предметної області;
- ознайомлення з подібними застосунками;
- синтез нового способу підбору музичних композицій на основі результатів порівняння існуючих подібних застосунків;
- розробка веб-застосунку для підбору музичних композицій з урахуванням вподобань.

Об’єктом даної роботи є веб-застосунок підбору музичного контенту.

Предметом даної роботи є використання сучасних інформаційних технологій для розробки веб-застосунку для підбору музичних композицій із урахуванням вподобань.

Методи дослідження. Аналіз впливу надлишкового вибору на процес підбору музичних композицій, узагальнення інформації про можливості сучасних технологій у процесі розробки веб-застосунків, реалізація та тестування веб-застосунку для підбору музичних композицій із урахуванням вподобань.

Практичне значення отриманих результатів. Розроблений веб-застосунок для підбору музичних композицій з урахуванням вподобань покликаний покращити процес вибору музичних композицій в умовах надлишковості інформації.

РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ ДО РОЗРОБЛЕННЯ ТА ВПРОВАДЖЕННЯ ВЕБ-ЗАСТОСУНКІВ

1.1 Огляд проблеми надлишкового вибору в сегменті музичного напрямку

Явище надлишкового вибору виникає, коли доступно багато еквівалентних варіантів [1]. Прийняття рішення стає переважним через безліч потенційних результатів та ризиків, які можуть виникнути в результаті неправильного вибору. Маючи занадто багато приблизно однаково варіантів вибору, розумово виснажує, тому що кожен варіант повинен бути зважений з альтернативами, щоб вибрати найкращий. Спочатку більший вибір призводить до більшого задоволення, але по мірі збільшення кількості варіантів вибір стає дедалі складнішим, і люди, як правило, відчувають більший тиск, розгубленість та потенційну незадоволеність своїм вибором. Хоча більший діапазон можливостей вибору музичних композицій може спочатку бути привабливими, менші набори призводять до збільшення задоволеності.

Ще одна складова надлишкового вибору - сприйняття часу. Вибір з широкого діапазону варіантів може здатися ще складнішим за обмеженого часу. У XXI столітті шалена швидкість розвитку технологій і великі обсяги інформації перешкоджають концентрації уваги. Для опису цієї культурної проблеми все частіше застосовують термін «синдром інформаційної втоми». Інформаційна втома є причиною різкої зміни настрою людини, викликає необгрунтовані рішення і вчинки. Мозок перенасичений інформацією, він перевантажений і знаходиться в стані тривоги, що приводить до нездатності адекватно сприймати і переробляти необхідний обсяг інформації, що надходить. Часто занадто великий обсяг інформації призводить до «аналітичного паралічу». Девід Льюїс характеризував це явище як поступальний процес, якому притаманні такі риси:

– постійна тяга до нової інформації і до її потенційних джерел;

– хронічне безсоння і тривога за новий день (людина починає розмірковувати, чи все він зробив правильно протягом дня, постійне аналізування);
– зниження здатності приймати обдумані рішення (немає часу на те, щоб обміркувати інформацію, що надійшла).

Професор фінського університету Тампери, Рейджо Саволайнен, визначає фільтрування та вилучення як загальну відповідь на інформацію. Фільтрування передбачає швидке з'ясування того, чи можна ігнорувати певну інформацію, у даному випадку музичні композиції, на основі певних критеріїв – вподобань людини. Вилучення відноситься до обмеження кількості джерел інформації, з якими взаємодіє людина. Він розрізняє джерела інформації, що «тягнуть» і «штовхають», причому джерелом, що «тягне» є те, де хтось шукає відповідну інформацію, а джерелом, що «штовхає», коли інші вирішують, яка інформація може бути цікавою. Вони зазначають, що використовуючи перший вид джерела, можливо уникнути перевантаження інформації, але ви ризикуєте втратити важливу інформацію. [2]

Було запропоновано багато рішень щодо зменшення перевантаження інформацією. На основі визначення інформаційного перевантаження існує два загальних підходи до боротьби з ним: [3]

– Зменшити кількість вхідної інформації – слідкувати за потоком вхідної інформації і обмежити введення-виведення, видаливши зайві джерела інформації.
– Покращити здатність до обробки інформації – вирішальне значення має те, як людина записує, формує та зберігає інформацію.

Для зменшення потоку вхідної інформації, застосування фільтрування базується на критеріях фільтрації інформації. У даний спосіб, кількість поступаючої інформації зменшується, що полегшує можливість вибору для людини. Для визначення вподобань у процесі вибору музичних композицій проводиться аналіз структури та інформаційного наповнення твору. Серед структурних елементів, які можуть певною мірою відноситись до вподобань є:

1. Автор музичної композиції
2. Жанр виконання
3. Емоційне забарвлення музичної композиції

Вибір музичних композицій наданням переваги конкретному автору, зумовлено існуючою схильністю людини до прослуховування музики у стилі, притаманному лише цьому автору. Так, наприклад, існують композитори та співаки, що виробляють музичний контент у стилі, який важко класифікувати за існуючими методами класифікації музичних творів. Вони мають фанатську аудиторію, що цікавиться їх напрямом та стилем творчості.

Рисунок 1.1 – Топ 10 музичних виконавців у світі станом на I квартал 2020 року

За даними сайту Avyanna, побудовано гістограму, яка визначає рівень вподобання користувачами виходячи з прибутку, який отримує виконавець [4]. Кореляція рівня вподобань та суми статку виконавця пояснюється кількістю прослуховувань композицій на різних музичних стрімінгових сервісах, відвідуваністю культурних заходів таких як концерти та шоу-програми, купівлі брендованої продукції виконавця тощо. Тобто, рівень статку автора напряму залежить від кількості фанатів.

Музичний жанр - це загальноприйнята категорія, яка визначає деякі музичні твори як такі, що належать до спільної традиції або набору умов. [5]

Його слід відрізняти від музичної форми та музичного стилю, хоча на практиці ці терміни іноді використовуються як взаємозамінні. [6] Музику можна розділити на жанри різними способами, такі як популярна музика та художня музика, або релігійна музика та світська музика. Художня природа музики означає, що ці класифікації часто є суб'єктивними та суперечливими, а деякі жанри можуть накладатися одне на одного.

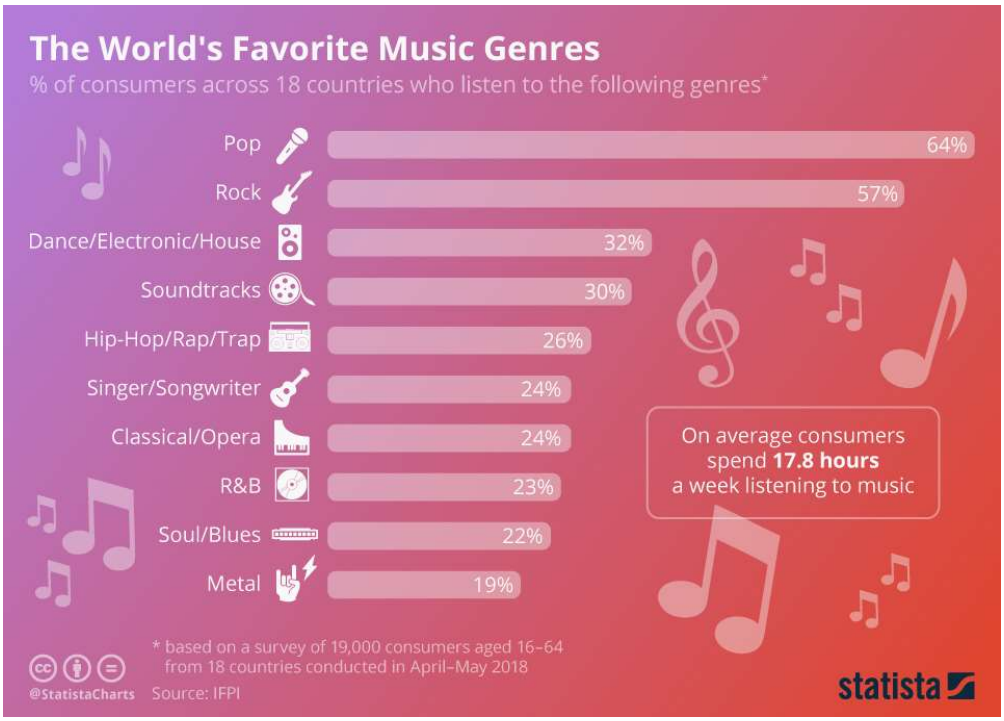

Рисунок 1.2 – Статистика популярності музичних жанрів станом на 2018 рік

За даними сервісу Statista, було проведено опитування 19000 людей з 18 країн, віком від 16 до 64 років на рахунок вподобань у прослуховуванні певних музичних жанрів [7]. На графіку можна бачити, що найбільш популярними є поп та рок, що становлять 64% та 57% відповідно. На третьому місці знаходиться електронна музика з 32% вподобань і замикають п’ятірку лідерів жанри саундтреків до фільмів та ігор з 30% та хіп-хоп з 26%.

Фільтрація вподобань за критерієм емоційного забарвлення музичної композиції зумовлена самопочуттям та поточним настроєм людини. Вчені UC Berkeley опитали понад 2500 людей у США та Китаї щодо їх емоційних реакцій на ці та тисячі інших пісень із жанрів, включаючи рок, фолк, джаз, класику, оркестр, експериментальний та хеві-метал [8].

З початку, волонтери відсканували тисячі відео на YouTube на наявність музики, що викликала різноманітні емоції. З них дослідники створили колекцію аудіокліпів для використання в своїх експериментах. Потім майже 2000 учасників дослідження у Сполучених Штатах та Китаї оцінили близько 40 музичних зразків на основі 28 різних категорій емоцій, а також за шкалою позитиву та негативу і шкалою рівнів збудження.

Використовуючи статистичний аналіз, дослідники дійшли до 13 загальних категорій досвіду, які збереглися в різних культурах та виявили, що вони відповідають конкретним почуттям, таким як «гнітюче» або «мрійливе».

Щоб забезпечити точність цих висновків у другому експерименті, майже 1000 людей із США та Китаю оцінили понад 300 додаткових західних та традиційних китайських зразків музики, спеціально призначених для того, щоб викликати зміни у валентності та збудженні. Їхні відповіді допомогли чітко виявити 13 категорій.

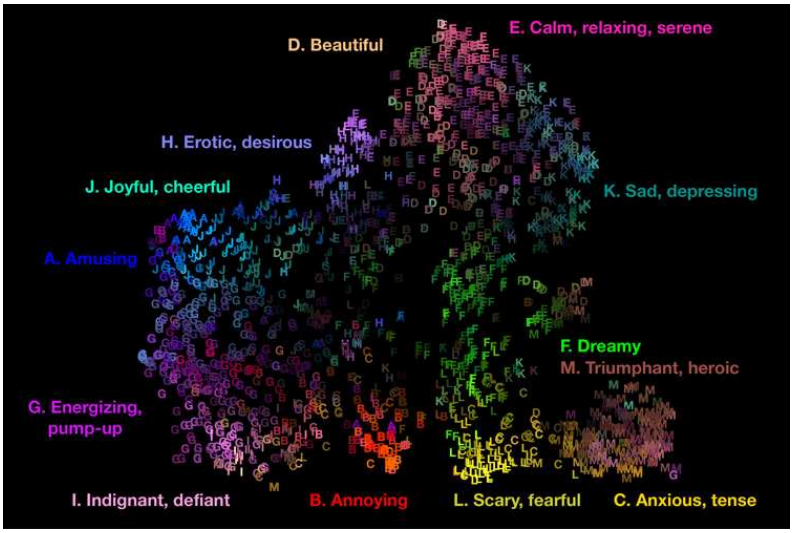

Рисунок 1.3 – Результати дослідження з виявлення категорій емоційного забарвлення музичних композицій

«Чотири пори року» Вівальді змушували людей відчувати енергію. Накачав їх «Rock the Casbah» від Clash. «Let’s Stay Together» Ела Гріна викликав чуттєвість, а «Somewhere over the Rainbow» Із Камакавівуоле викликав радість. Тим часом важкий метал широко розглядався як зухвалий і,

як задумав його композитор. Оцінка музичного супроводу сцени з душу в фільму «Psycho» викликала страх.

Дослідники визнають, що деякі з цих асоціацій можуть базуватися на контексті, в якому учасники дослідження раніше чули певний музичний твір, наприклад, у фільмі чи відео на YouTube. Але це менш вірогідно у випадку з традиційною китайською музикою, за допомогою яких проводився контроль адекватності висновків.

1.2 Огляд існуючих застосунків та порівняння підбору музичних композицій з урахуванням вподобань

Для порівняння існуючих рішень в даній предметній області, на просторах Інтернету потрібно знайти застосунки, що спеціалізуються на рекомендаціях музичного контенту із урахуванням вподобань. Для виокремлення видимих переваг чи недоліків сервісів, насамперед, треба визначити критерії оцінювання, за якими буде проводитися порівняння і постановка задачі.

Gnoosic – веб-застосунок, що надає користувачу музичні рекомендації за критерієм авторства [9]. У якості джерела інформації та обчислювальних алгоритмів використовується веб-ресурс GNOD (Global Network of Discovery). Застосунок має лише дві активні сторінки. На першій потрібно ввести назви трьох музичних груп або виконавців, які вам подобаються. Після введення ваших уподобань застосунок почне пропонувати нових виконавців. Кожного можна вподобати, не вподобати або залишитися байдужим, поставивши «Я не знаю». На підставі ваших відповідей будуть підбиратися наступні варіанти. Список є скінченним, і його завершенню можна розпочати новий підбір або відповісти на коротке опитування щодо якості наданих рекомендацій для покращення роботи застосунку. Послухати чи подивитися інформацію про групу не можна. Також відсутня інформація про популярні музичні твори тих чи інших виконавців. Доведеться після кожного назви відкривати сторонній музичний стрімінговий сервіс щоб дізнатися, чи вартий виконавець того, щоб ви додали його в свою бібліотеку музики.

Magic Playlist - це веб-застосунок, що надає користувачу музичні рекомендації у вигляді плейлисту за критеріями емоційного забарвлення, виконавцем або назвою композиції. Він розроблений за допомогою API Spotify [10]. На головній сторінці розташовано поле текстового пошуку, за допомогою якого можна знайти вподобання відносно введеної назви композиції, кнопки пошуку за настроєм генерованого плейлисту та пошук за окремими акторами. Обравши вид вподобань, алгоритм визначає основних виконавців і створює список відтворення на основі їх рейтингових композицій. Наповнення плейлисту можна регулювати шляхом вибору сумарної тривалості всіх композицій, що варіюється від 1 до 3 годин. Ви можете переглянути кожну пісню, прослухати її фрагмент та в разі невподобання видалити зі списку. Оскільки сервіс побудований на Spotify API, для повноцінного користування потрібно мати підписку на стрімінгову платформу Spotify. У разі відсутності підписки, користувач не зможе скористатися опцією збереження плейлисту до своєї музичної бібліотеки та буде вимушений витрачати час на пошук отриманих вподобань на інших стрімінгових платформах.

Itcher – веб-застосунок, доступний також на iOS та Android, що пропонує різний мультимедійний контент, зокрема й музику [11]. Для користування застосунком, потрібно створити обліковий запис, який буде зберігати дані про історію наданих рекомендацій. На головній сторінці знаходиться меню вибору напрямку надання рекомендацій та стрічка з рекомендаціями на основі попередніх використань. Натиснувши на кнопку пошуку музичних композицій, з’являється фільтр. Вподобання можна налаштувати у декілька способів: можна обрати 5 улюблених виконавців або обрати жанр та приблизний рік створення музичної композиції і отримати рекомендації щодо схожих авторів. Застосунок не підтримує відтворення музичних фрагментів та цілої композиції, однак має посилання на музичний стрімінговий сервіс Apple Music. У кожної композиції є свій рейтинг, за допомогою якого визначається рівень правдивості вподобання. Також присутні коментарі інших користувачів, щодо якості наданої їм рекомендації.

Для порівняння характеристик застосунків, у розрахунок будуть братися наступні критерії оцінки:

1. Швидкість підбору музичних композицій із урахуванням вподобань – дані базуються на середньому значенні 10 замірів продуктивності.
2. Зручність використання – наскільки застосунком зручно користуватися та який зовнішній вигляд він має.
3. Фільтри вподобань – способи, за якими користувач може визначити свої вподобання всередині застосунку.
4. Доступність – наскільки даний застосунок є доступним для користувача.
5. Релевантність пропозицій – співвідношення отриманих рекомендацій до вподобань користувача. Ця оцінка є суб’єктивною і базується на думці автора роботи.
6. Технологія отримання даних – спосіб, у який веб-застосунок отримує дані для представлення користувачу.

В таблиці 1.1 приведено порівняння веб-застосунків Gnoosic, Magic Playlist та Itcher.

Таблиця 1.1. Порівняння застосунків підбору музичних композицій з урахуванням вподобань

Застосунок / Критерій оцінювання	Gnoosic	Magic Playlist	Itcher
Швидкість надання рекомендацій	≈ 0,838 мс	≈ 1126 мс	≈ 8791 мс

Продовження таблиці 1.1

Застосунок / Критерій оцінювання	Gnoosic	Magic Playlist	Itcher
Зручність використання	Мінімалістичний та зручний, інтуїтивно зрозумілий та простий у використанні	Зручний у використанні, має адаптивний привітливий дизайн	Недопрацьований дизайн, низька якість графічних матеріалів, відсутня адаптивність
Способи пошуку	Пошук за автором	Пошук за автором, назвою та емоційним забарвленням	Пошук на основі рейтингу; пошук за жанрами та роком
Доступність	Безкоштовний застосунок, відсутня реєстрація	Для розблокування повного списку можливостей, необхідна підписка	Безкоштовний застосунок, присутня реєстрація
Релевантність вподобань	Частково співпадають	Добре співпадають	Частково співпадають
Технологія отримання даних	API	API	API

З отриманих даних слідує, що обрані застосунки є досить різними у своїх характеристиках і принципах роботи. На даному етапі можна визначити переваги та недоліки цих застосунків, базуючись на даних дослідження.

Перевагами Gnoosic є висока швидкість роботи застосунку, доступність та простота використання. Застосунок безкоштовний та не потребує наявності підписки, тому він є доступним для всіх користувачів, містить мінімальний набір елементів для взаємодії з користувачем, що спрощує процес експлуатації. Серед недоліків слід виокремити малу кількість наданої інформації на запит у вигляді лише назв виконавців, відсутність пошуку за жанрами та іншими фільтрами, неможливість попередньо перевірити релевантність наданих рекомендацій.

Для застосунку Magic Playlist перевагами являється невелика затримка відповіді серверу, приємний дизайн та наявність достатньої кількості фільтрів для урахування вподобань користувача. Є можливість попереднього ознайомлення з композиціями шляхом прослуховування фрагментів, завдяки чому можна перевірити якість рекомендацій, та опція вибору тривалості плейлисту. Недоліками застосунку є відсутність підбору музичних композицій за жанрами та урізаний функціонал безкоштовної версії. Аби зберегти результат підбору, потрібно мати платну підписку на стрімінговий сервіс Spotify.

Застосунок Itcher пропонує користувачу пошук за жанрами та роком, а також відносно вподобаних виконавців. Для кожного наданого вподобання є розділ з коментарями інших користувачів, наскільки правильною є рекомендація. Можна самому оцінювати рейтинг композиції чи її автора. Застосунок безкоштовний, але вимагає створення облікового запису. Серед проблем застосунку, найбільшою являється висока затримка відповіді сервера. Наявні проблеми з відображенням контенту на сторінках: це прослідковується у відсутності адаптивного дизайну та низькою якістю графічного супроводу композицій, таких як обкладинка чи фото автора.

Веб-застосунки використовують програмний інтерфейс (API) для отримання результатів пошуку вподобань. Такий спосіб спрощує взаємодію між клієнтською та серверною частиною.

1.3 Оцінка вимог та постановка задачі

Веб-застосунок для підбору музичних композицій із урахуванням вподобань має бути побудований таким чином, щоб взаємодія з користувачем займала якомога менше часу, при цьому рекомендуючи контент максимально схожий на вподобання користувача. Взаємодія має відбуватися через простий та зрозумілий інтерфейс, аби не відволікати користувача непотрібною інформацією. Правила підбору мають основуватися на максимально великій кількості фільтрів для збільшення можливостей вираження користувачем своїх вподобань, але не настільки, щоб викликати у користувача негативні

емоції. Усі процеси слід оптимізувати, для підвищення продуктивності застосунку підбору музичних композицій. Для цього, веб-застосунок, який буде основою для реалізації проекту повинен бути розроблений за допомогою сучасних технологій, що мінімізують витрати на підтримку рішення та максимізують його показники продуктивності. Також важливим є дизайн та структура застосунку, оскільки взаємодія з користувачем відбувається через графічний інтерфейс. Мобільність застосунку дозволить користувачам використовувати застосунок на будь якому десктопному та мобільному пристрої, що має доступ до Інтернету.

Однак, ґрунтуючись на даних порівняння, існуючі застосунки та сервіси мають недоліки, які впливають на якість підбору музичних композицій за вподобаннями. Відсутність достатньої кількості фільтрів спричиняє зменшення кількості множин пошуку вподобань і негативно впливає на результат. Без можливості прослуховування фрагментів отриманих рекомендацій, користувач не може швидко визначити релевантність наданої рекомендації відносно свого музичного смаку. Кількість користувачів, які можуть використовувати застосунок відноситься до рівня доступності цього застосунку. Багато людей просто не готові платити гроші за користування сервісами, тому порівняно з іншими, не отримають доступу до повноцінного функціоналу або будуть вимушені переглядати рекламу. Проблема відображення контенту може звести нанівець усі переваги застосунку, якщо ним буде не зручно користуватися.

З точки зору користувача, застосунок не повинен викликати проблем використання через великий асортимент функціоналу або його відсутності. Простий та приємний інтерфейс має розміщувати на собі елементи, які безпосередньо впливають на процес підбору музичних композицій з урахуванням вподобань. Також, можливість у будь-який момент змінити критерії підбору і відсутність впливу попередніх рекомендацій буде плюсом у зручності.

За допомогою комбінування основних переваг застосунків, можна добитися результатів, що у перспективі мають бути кращими за показники аналогічних застосунків. Це дозволить представити користувачу ширший діапазон інструментів опису своїх вподобань, зручний інтуїтивно-зрозумілий інтерфейс та підвищити швидкість роботи веб-застосунку для підбору музичних композицій з урахуванням вподобань.

На основі оцінки стану проблеми, поставлено задачі щодо реалізації рішення:

- Вдосконалити процес підбору музичних композицій із урахуванням вподобань як жанр, назва композиції чи ім’я автора.
- Визначити функціональні та технічні вимоги для проектування та розробки веб-застосунку.
- Протестувати та порівняти результати з існуючими застосунками підбору музики.

ВИСНОВКИ ПО РОЗДІЛУ 1

Збільшення використання Інтернету серед компаній та приватних осіб вплинуло на спосіб ведення бізнесу. Це призвело до стрімкого росту кількості інформації, що спричинило появу такого феномену як надлишковість інформації та «синдром інформаційної втоми». Це особливо помітно у сегменті мультимедійної інформації, а саме музичних композицій. Люди слухають музику, щоб отримати певні емоції та поринути у свої думки.

Одним із шляхів вирішення цієї проблеми є системи підбору музичного контенту виходячи з вподобань людини. Метою цих систем є надання релевантного контенту на основі вподобань та музичного смаку користувача. Такі системи є досить адаптивними і можуть застосовуватися у багатьох сферах, що потребують фільтрації контенту за специфічними критеріями. Це значить, що даний тип систем може бути використаний для підбору фільмів, зображень, книг, телевізійних шоу тощо.

На сьогодні існує багато веб-застосунків, що реалізують підбір музичних композицій із урахуванням вподобань користувача. Вони використовують різні технології та алгоритми роботи, що дозволяє помічати тенденції схожості композицій швидше, ніж людина і оперують набагато більшою кількістю інформації в момент часу. Таким чином, для виявлення способів створення нового веб-застосунку для підбору музичних композицій було проведено порівняння існуючих аналогів та виокремлено основні переваги та недоліки, на основі аналізу яких сформовано тезисну постановку задачі.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПІДБОРУ МУЗИЧНИХ КОМПОЗИЦІЙ

2.1 Огляд способів вдосконалення підбору музичних композицій з урахуванням вподобань

У даному параграфі йде врахування усіх аспектів функціонування аналогів та опис процесу моделювання вдосконаленого рішення. Базуючись на результатах аналізу, потрібно синтезувати нове рішення, яке модернізує процес підбору музичних композицій з урахуванням вподобань.

Першим і найголовнішим аспектом формування задумки до реалізації є сам механізм отримання рекомендацій на основі вподобань. Необхідно визначити переваги та недоліки методів отримання рекомендацій з точки зору зручності розробки, використання та швидкодії для збільшення показників продуктивності застосунку. З порівняльної характеристики проаналізованих застосунків відомо, що всі вони використовують таку технологію отримання даних як REST API. Це зумовлено зручністю генерації запитів до обчислювального центру, в ролі якого виступає веб-сервер та поверненням структурованих даних у зручному для обробки клієнтом вигляді. API це лише програмний інтерфейс, який стандартизує процес передачі інформації, та основним критерієм вибору являється характеристика системи, яка лежить на серверній частині застосунку. Нажаль визначити достовірно тип алгоритму надання рекомендацій неможливо в зв’язку конфіденційності інформації веб-застосунку. Розробники не оприлюднюють дану інформацію публічно аби запобігти розвитку конкурентів, що можуть використати чужі напрацювання з метою отримання прибутку. Тому на основі вимірювань швидкості надання відповіді та суб’єктивної релевантності наданих вподобань можна визначити, яку саме систему підбору музичних композицій слід використати з метою покращення веб-застосунку.

Gnoosic використовує GNOD API, що є програмним інтерфейсом для нейронної мережі GNOD, яка одночасно обслуговує декілька проектів надання рекомендацій за вподобаннями [12]. У зв’язку з простотою структури відповіді

сервера, клієнт отримує відповідь швидко. Проблемою використання даного програмного інтерфейсу є мала кількість інформації, що передається для представлення користувачу, її банально недостатньо для підтвердження релевантності наданих вподобань. Також

Itcher використовує власний Itcher API для реалізації підбору музичних композицій і доступу до інформації щодо принципів роботи не надається. Веб-застосунок надає великий обсяг інформації для характеристики підібраних музичних композицій, такий як ім’я автора, назва композиції, опис контенту та відгуки користувачів. Проте час, який витрачається на очікування завершення обробки запиту дуже великий в порівнянні з аналогами і наданий супроводжуючий графічний матеріал низької якості.

Для реалізації функціонування веб-застосунку Magic Playlist в якості джерела обчислювальних потужностей було взято Spotify API [13]. Spotify API – це програмний інтерфейс компанії Spotify, який був створений спеціально для розробників музичних веб-застосунків з метою збільшення кількості дочірніх сервісів. Це API досить громіздке та має велику кількість функцій, саме тому на основі нього зараз базується понад десяток інших застосунків. Перевагами використання розробок провідної музичної стрімінгової платформи, надає розробникам можливість імплементації якісного програмного забезпечення у своїх проектах, технічну підтримку у питаннях функціоналу й режиму роботи. Застосунки на основі даного API показують середні по часу відгуку результати, що є допустимою нормою, тому це можна вважати за перевагу. Кількість наданої інформації про підборі музичні композиції є вичерпною для користувача і дає змогу оцінити релевантність наданих вподобань.

Вимоги щодо дизайну представлення інформації користувачу основані на результатах порівняння зовнішнього вигляду аналогів. Для задоволення потреб користувачів, що користуються різними гаджетами для виходу в Інтернет, потрібно максимально привести веб-застосунок до поняття адаптивності дизайну та зручності використання. Це найскладніший вид

верстки, при якому сторінка підлаштовується під різні екрани [дж]. При цьому зовнішній вигляд сайту може змінюватися в залежності від розмірів екрану пристрою, на якому його переглядають, ну функціонал залишається незмінним. Головна перевага адаптивної верстки - орієнтація на мобільні технології. Все більше користувачів використовують для виходу в Інтернет смартфони. Адаптивна верстка дає можливість показувати таким користувачам більш полегшену версію сторінки веб-застосунку. До того ж адаптивна верстка краще сприймається пошуковими системами. Адаптивна верстка має і свої недоліки. Це, в першу чергу, її складність. Другий недолік випливає з першого. Оскільки адаптивна верстка має на увазі велику кількість витрачених сил і часу, її вартість досить висока. Найкращим себе показав застосунок Magic Playlist, оскільки він має простий структурований дизайн та може змінювати структуру будови сторінки в залежності від розмірів екрану девайса користувача. Також інтерфейс витримано у мінімалістичному стилі, що спрощує розуміння призначення елементів інтерфейсу і не захламлює його.

Кількість критеріїв оцінки вподобань користувача має базуватися на ідеї «золотої середини», тобто не мало і не занадто багато. Способи визначення вподобань застосунку Gnoosic є недостатніми та потребують наближення до кількості фільтрів, як у Magic Playlist та Itcher. Ще одним важливим параметром є тип фільтру пошуку вподобань, серед яких фільтри жанру та текстовий пошук музичної композиції чи автора покривають найбільшу площу музичних композицій, що можуть бути вподобані. На основі цього, слідує доцільним визначити для користувача фільтрування вподобань за популярними жанрами.

Для забезпечення доступності веб-застосунку, умови використання для всіх користувачів мають бути однаковими та не залежати від підписки чи наявності облікового запису. Застосунок Gnoosic надає можливість експлуатації безкоштовно та без необхідності наявності персоналізованого аккаунту користувача. Це робить його ненав’язливим, на відміну від Magic Playlist та Itcher. Пропозиція безкоштовно надавати користувачу послугу

рекомендаційного характеру на основі його вподобань привертає увагу та може забезпечити застосунок популярністю.

Для реалізації проекту веб-застосунку з підбору музичних композицій на основі вподобань потрібно розробити детальний план розробки та чіткі вимоги на основі досліджень з першого розділу роботи. За своїм представленням, застосунок повинен мати зручну навігацію по сторінкам, адаптивний та привітливий дизайн, сторінки з описом помилок, що можуть виникнути під час сесії. Оскільки в наш час є актуальною тема захисту даних користувачів та безпека в інтернеті, такому відносно малому сервісу не має нагоди збирати персональні дані користувачів, це лише додасть зайвих клопотів як власнику, так і користувачеві [14].

Функціонал веб-застосунку має бути спрямований на пошук музичних композицій з урахуванням вподобань, перелік якого подано нижче:

- Можливість пошуку автору та назви композиції через поле пошуку.
- Можливість пошуку композицій за жанрами.
- Перегляд композицій певного автора.
- Пошук подібних композицій до обраної користувачем.
- Наявність супроводжуючого графічного контенту (обкладинка композиції / зображення автора).
- Можливість прослуховувати фрагменти шуканих композицій для кращого сприйняття користувачем та зручнішого пошуку подібних творів.
- Можливість перегляду композицій на сторонніх ресурсах за посиланнями.

На стартовій сторінці має знаходитись привітання користувача та елементи пошуку контенту, через які можна перейти до інших сторінок в залежності від вибору. При повнотекстовому пошуку, має виконуватися навігація на сторінку з результатами пошуку або на сторінку, що відповідає за

обробку помилок. При натисканні на назву автора має виконуватися перехід на сторінку з композиціями цього автора, а при натисканні на назву пісні чи обкладинку композиції має виконуватись пошук подібних творів. Кожна композиція / автор має розташовуватись у своєму окремому контейнері та містити графічне зображення, що характеризує композицію / автора відповідно.

2.2 Аналіз технологій для реалізації підбору музичних композицій у вигляді веб-застосунку

Веб-застосунок – клієнт-серверний застосунок, в якому клієнт взаємодіє з веб-сервером за допомогою браузера. [15] Логіка веб-застосунку розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі. Незалежно від платформи на якій відкрито застосунок, контент має відображатися вірно і сайт повинен вчасно реагувати на дії користувача.

Головною відмінністю веб-застосунку від веб-сайту є принцип подачі контенту. В той час, коли звичайний веб-сайт надає користувачу можливість споглядати статичні сторінки з інформацією, веб-застосунок відображає динамічні зміни на сторінці в залежності від дій та потреб користувача, надаючи можливість взаємодіяти з функціональними аспектами сторінки. Прикладами веб-сайтів є новинні портали, інтернет-видання, інтернет-форуми, тощо. А яскравим прикладом веб-застосунків є онлайн-магазини, соціальні мережі та розважальні портали.

Принципи побудови веб-сайту та веб-застосунку відрізняються тим, що у першому випадку від користувача не вимагається прямої взаємодії. Як тільки користувач виконую дію на сторінці, створюється запит, на який слідує відповідь з боку серверу, на якому розташований ресурс. Всі веб-застосунки можуть бути веб-сайтами, проте не всі веб-сайти є веб-застосунками. Цікавий той факт, що веб-застосунок потребує не лише звичних інструментів для створення веб-сторінок, таких як HTML, CSS, JavaScript (яких вже достатньо для реалізації сторінки), а й додаткових ресурсів у вигляді бібліотек,

фреймворків, баз даних. Це дозволяє створювати більш гнучкі та масштабовані проекти, відкриті для вдосконалення.

Порівняно з нативними застосунками, веб-застосунок працює на всіх пристроях, незалежно від типу операційної системи, архітектури процесора пристрою, тощо. [16] Веб-застосунки, як правило, не мають специфічних вимог до апаратного забезпечення та платформи, на якій будуть розгортатися та не потребують встановлення на пристрій – для їх використання потрібен лише браузер.

Також на відміну від нативних застосунків, веб-застосунки не мають такої проблеми як оновлення до актуальної версії. У випадку з браузерним застосунком, існує лише одна працююча версія, якою користуються усі користувачі. При оновленні версії усі користувачі автоматично переходять на неї, коли у нативних застосунків можуть виникати проблеми при завантаженні та оновленні до актуальної версії.

Оскільки веб-застосунки не повинні бути схвалені ринком застосунків, вони можуть бути випущені в будь-який час і в будь-якій формі відповідно до бажань розробника.

Основним плюсом веб-застосунків є їх мобільність та доступність. Для доступу потрібно мати URL-адресу та вихід в мережу Інтернет. Користувач може зберегти результат своєї роботи на сервері і потім повернутися до своїх напрацювань у будь-який час та з будь-якого місця, де є вихід в мережу Інтернет. Хоча веб-застосунок може використовуватись на різних пристроях, але якщо це не адаптивний веб-сайт, то користувач буде спостерігати проблеми з відображенням контенту.

Наявність підключення до Інтернету буде необхідним для доступу до застосунку, інакше користувач не зможе ніяк з ним взаємодіяти. Це особливо стосується людей, що живуть у віддалених регіонах або перебувають у подорожі, так як якість з’єднання з мережею може бути нестабільною і ускладнюватиме роботу з застосунком.

При оновленні сторінки виникає помітна затримка відповіді сервера, що зумовлено необхідністю встановлення HTTP-з’єднання, обробки запиту на серверній стороні та повернення оновленого стану в клієнт.

Логіка веб-застосунку розподілена між сервером та клієнтом таким чином, що основні функції та маніпуляції з даними виконуються на стороні серверу, а задачі, що можуть обмежитись потужностями браузеру – на стороні клієнта [17]. Дані користувачів у своїй більшості зберігаються на сервері, хоча деяка їх частина може міститись у cookie-файлах браузера.

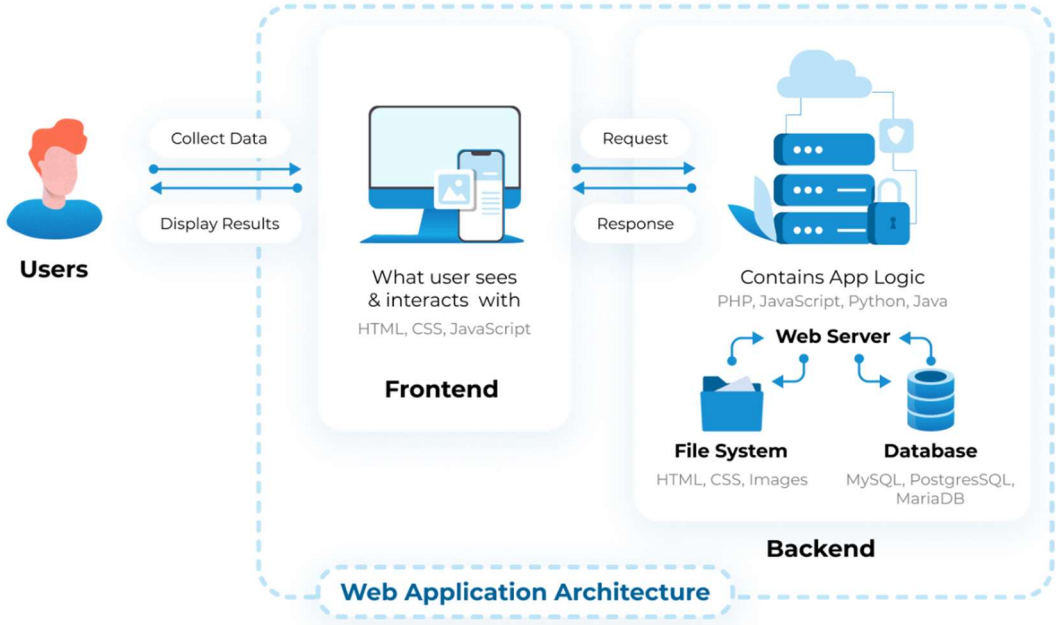

Рисунок 1.4 – Схема клієнт-серверної архітектури

Клієнт

Клієнтська сторона програми - це частина, яка є видимою для користувачів і представляє "обличчя" або "фасад" програми. [18] З цієї причини його зазвичай називають "інтерфейсом". Технологій веб-стека, пов’язаних з цим рівнем розробки, не так багато:

– Мова розмітки гіпертексту (HTML) контролює структуру інформації, що відображається користувачеві у веб-браузері.
– Каскадні таблиці стилів (CSS) визначають стиль відображуваних даних, керуючи такими параметрами, як шрифти, макети та кольори тексту та фону.

– JavaScript керує інтерактивними веб-функціями програми.

Всі ці технології є потужними інструментами в руках кваліфікованих фахівців. Однак їх поєднання надає абсолютно новий рівень можливостей розробникам веб-застосунків. Хоча JavaScript, HTML та CSS розглядаються як звичайні стандарти в цій галузі, ви можете замінити їх такими засобами розробки, як Apache Flex та іншими.

Використання фреймворків є широко розповсюдженою практикою, яка полегшує процес розробки веб-застосунків. Популярні приклади інтерфейсних фреймворків та бібліотек включають:

– Angular – основа, розроблена Google для проектування динамічної структури даних для веб-застосунків.
– React – бібліотека від Facebook Inc., орієнтована на розробку інтерфейсів SPA, що означає, що програма може відображати різні дані на одній сторінці, не перезавантажуючи їх. Використовуючи JSX (синтаксичне розширення React для Javascript) розробники можуть легко створювати масштабовані інтерактивні рішення.
– Vue.js – легкий фреймворк JavaScript для створення адаптивних користувацьких інтерфейсів для SPA, який швидко набирає популярність серед розробників з моменту появи в 2014 році.

Що стосується веб-застосунків, сучасна розробка інтерфейсу вимагає поєднання кількох компонентів: бібліотек та фреймворків, оскільки лише чогось одного недостатньо для задоволення вимог клієнта та користувачів. З цієї причини набирають популярності спеціалізовані набори веб-розробок. Найпопулярніші їх приклади:

– Bootstrap – колекція шаблонів CSS та JavaScript для проектування компонентів інтерфейсу. Її основною метою є підвищення швидкості реагування та підтримка принципу веб-розробки «перш за все для мобільних пристроїв». Варто зауважити, що основною метою цієї основи є веб-сторінки, а не веб-програми.

– Foundation – набір з трьох спеціалізованих фреймворків для веб-сайтів, застосунків та електронної пошти. Це економія часу для front-end розробників завдяки включеному набору ефективних компонентів для HTML, JS та CSS, орієнтованих на швидкість реагування та підхід «перш за все для мобільних пристроїв».

Кожен із вищезазначених інструментів має набір переваг та недоліків, спільноту учасників та ряд найбільш підходящих сценаріїв впровадження.

Сервер

Серверна сторона (back-end) веб-програми прихована від користувачів і включає всі компоненти, необхідні для запуску програми. До компонентів на стороні сервера належать:

– Мова програмування для написання коду веб-програми. Як і при розробці на стороні клієнта, існує кілька мов та численні рамки для спрощення та вдосконалення процесу кодування. Серед найпопулярніших інструментів для внутрішнього програмування ми можемо назвати такі мови, як Python, PHP, Ruby, Java та Scala та їх фреймворки: Django, Flask, Laravel, Ruby on Rails, Spring та Play, відповідно.

– База даних для зберігання даних програми. Ці об'єкти можуть бути реляційними або нереляційними, залежно від моделі управління даними. Реляційні бази даних використовують у своїй роботі структуровану мову запитів (SQL), тоді як нереляційні (NO-SQL) бази даних використовують інші моделі для зберігання та пошуку даних. За даними ресурсу DB-Engines, 5 найкращих баз даних: Oracle, MySQL, Microsoft SQL Server, PostgreSQL та MongoDB. Примітно, що всі вони, крім останнього, є відносними. Однак у контексті розробки веб-застосунків уподобання дещо відрізняються.

– Сервер для обробки запитів, що надходять з боку клієнта, тобто від користувачів програми. Тут вибір досить обмежений: майже всі веб-

програми розробляються з використанням Nginx або Apache як серверів застосунків.

Якщо з технологіями створення графічного інтерфейсу веб-застосунку можна використати базові мови та широковживані фреймворки, то для серверної частини потрібно приділити більше уваги, щоб обрати технологію яка в перспективі має покращити продуктивність веб-застосунку.

За даними опитування розробників серверної частини веб-застосунків у 2020 році, найпопулярнішим вибором розробників є мова Python.

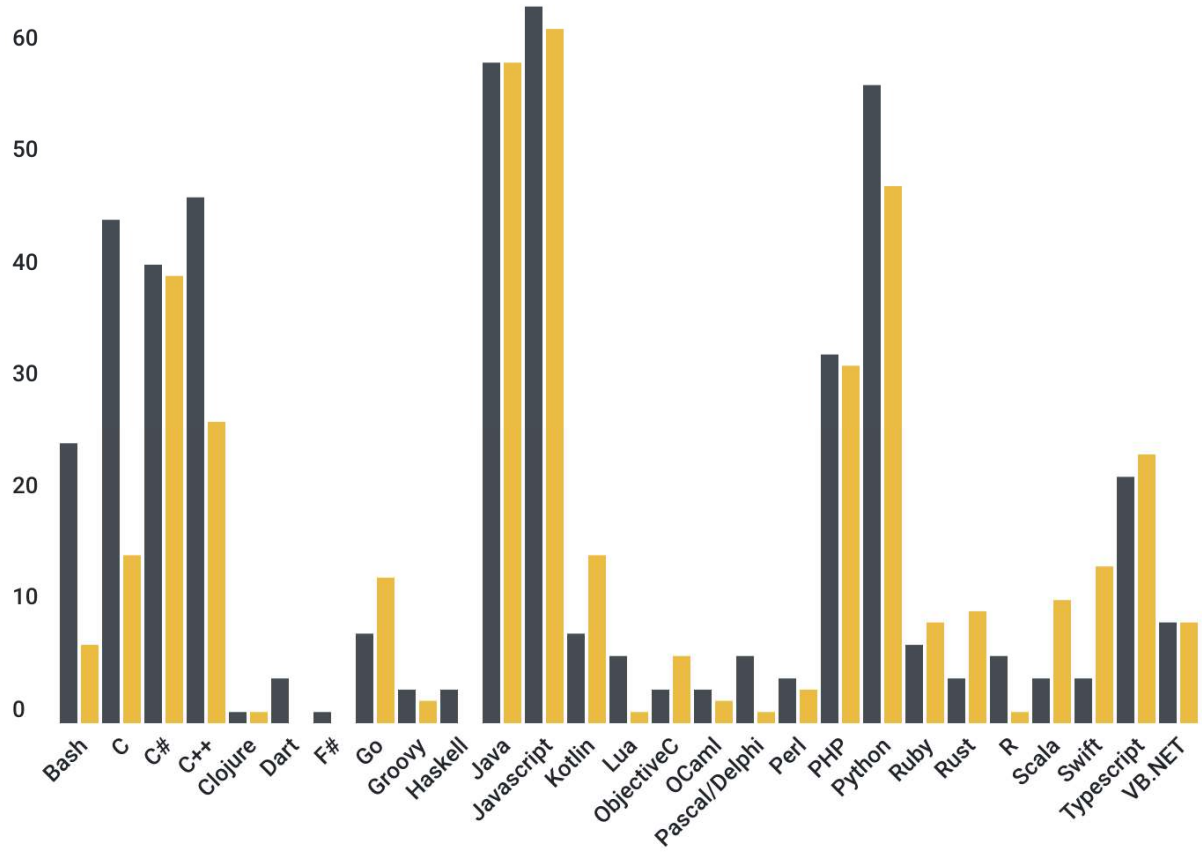

Рисунок 2.1 – Графік популярності мов для розробки серверної частини у відсотках

Для початку, Python надзвичайно добре справляється з рейтингами популярності спільноти [19]. Він був нагороджений (вчетверте) мовою програмування TIOBE 2020 року, отримавши найвищий приріст популярності за один рік (2,01%). Індекс TIOBE - це діаграма мов програмування, заснована на кількості професійних інженерів у всьому світі, що використовують ту чи іншу мову та кількості доступних навчальних курсів. Як показує опитування

Stack Overflow 2020, Python також посідає третє місце як найбільш улюблена мова програмування (66,7% респондентів заявляють, що це їх найулюбленіший вибір) [20].

Щоб розпочати свій проект з розробки програмного забезпечення, популярність, безумовно, є одним із факторів, про який потрібно пам'ятати. Незалежно від того, чи хочете ви зібрати кваліфіковану власну команду або знайти надійного технічного партнера, який підтримає вас у створенні вашого продукту, перехід на широко поширену технологію, безсумнівно, допоможе набагато швидше розпочати процес розробки.

Однією з причин такої кількості програмістів, які працюють з Python, є його простота. Синтаксис Python нагадує синтаксис англійської мови, що робить його простим і стислим. Це також не вимагає від розробників використання низькорівневих функцій, таких як управління пам'яттю, для виконання більш досконалої роботи. Наприклад, якби ми кодували ту саму функціональність за допомогою Java, це зайняло б у три-п'ять разів більше часу, ніж кодування її за допомогою Python.

Третя причина, чому вибір Python для внутрішньої розробки в 2021 році здається гарною ідеєю - це безліч фреймворків та бібліотек, які дозволяють розробникам автоматизувати реалізацію певних завдань і, як результат, приділяють більше уваги бізнес-логіці програми. Оскільки кожен фреймворк відповідає різним видам потреб, його вибір слід робити, виходячи з вимог проекту. На сьогодні, для розробки веб-застосунків, найпопулярнішими є два фреймворки: Django та Flask [21].

Випущений у 2005 році, Django має наступні переваги:

– Довгий час існування, що перетворюється на більш широку підтримку ком'юніті розробників та надійну документацію.
– Вибираючи цей веб-фреймворк, ви отримуєте повний пакет: адміністративну панель, об'єктно-реляційне відображення (ORM), інтерфейс бази даних, структура каталогів для додатків. Це

призводить до відсутності необхідності використовувати додаткові сторонні інструменти та бібліотеки.

- Django-admin полегшує обробку адміністративної частини загальних проектів завдяки готовій до використання та повністю функціональній структурі адміністратора.
- Економія часу завдяки вбудованому механізму шаблонів.
- Завдяки вбудованому інструменту відлагоджування, Django дозволяє розробникам Python створювати веб-програми без зовнішнього введення та створювати нові програми в рамках проекту.
- Система ORM дозволяє розробникам легко працювати з базами даних (MySQL, SQLite, Oracle тощо).
- Django надає пріоритет безпеці та робить акцент на запобіганні таким проблемам, як ін'єкція SQL, міжсайтовий сценарій або клацання.
- Більше того, система аутентифікації користувачів гарантує безпеку та надійність даних користувачів.

Однак стільки переваг повинно мати свою ціну. У випадку з Django це монолітна структура, яка може бути громіздкою (особливо для розробників з меншим досвідом) і може містити занадто багато функцій, непотрібних для функціонування веб-застосунку для підбору музичних композицій з урахуванням вподобань.

Список переваг Flask сильно відрізняється від Django, через переслідування інших цілей:

- Будучи мікрофреймворком, що пропонує основні функції веб-програми, такі як маршрутизація URL-адрес або обробка помилок, Flask є легкою та гнучкою.
- Оскільки Flask є простим рішенням, новачкові-інженеру легше звикнути, а досвідчений розробник Python, який використовує

його, отримує більшу гнучкість, коли йдеться про додавання функціональних можливостей, таких як автентифікація, обробка форм чи робота з API.

– Вище згадані переваги означають, що Flask ідеально підходить для менш масштабних проектів, які потрібно швидко запускати.

Тим не менш, подібно до Django, Flask не є ідеальним рішенням: його мікроструктура робить його менш потужним, ніж Django, і він має меншу підтримку спільноти. Однак ще раз, ці недоліки являються зовсім незначними для розробки веб-застосунку для підбору музичних композицій із урахуванням вподобань.

2.3 Пошук програмного рішення для підбору музичних композицій

Виходячи з порівняння аналогів, найкращим варіантом для підбору музичних композицій буде використання існуючих API, що спеціалізуються на підборі музичних композицій з урахуванням вподобань користувачів. Такі програмні рішення можна знайти у спеціалізованих API-маркетплейсах або використати розробки однієї з компаній, що активно розвивається та зарекомендувала себе як вибір користувачів впродовж років використання. Вибір має переважати в сторону організацій, що розпоряджаються великими ресурсами і вкладають більше часу та матеріалів для підтримки та вдосконалення своїх розробок. Однак, оцінити якість усіх пропозицій дуже складно, тому можна припустити, що всі API, розроблені великими організаціями, працюють досить добре та надають вичерпний функціонал для імплементації його до проекту.

Серед існуючих сервісів, що дозволяють розробникам створювати нові застосунки на основі їх API було обрано програмний інтерфейс Shazam. Shazam – безкоштовний кросплатформний проект, що дозволяє користувачеві визначити, що за пісня грає в даний момент (раніше був доступний тільки комерційний проект і тільки для мобільних пристроїв) [22].

Користувач Shazam використовує мікрофон пристрою для запису фрагмента музики, яка грає де-небудь. Потім програма порівнює фрагмент з центральною базою даних і при успішному зіставленні видає інформацію про трек. На даний момент сервіс надає інформацію про більш ніж 11 млн треків. Shazam може ідентифікувати записані звуки, які передаються з будь-яких джерел, таких, як радіо- або телетрансляція, музика в кінофільмі чи клубі, за умови, що рівень фонового шуму не надто високий. Shazam зберігає каталог аудіо, упізнаних за допомогою програми, даючи прямі посилання на дані треки на Apple Music, якщо такі там є.

На відміну від Spotify та інших стрімінгових платформ, Shazam нею не являється – у Shazam не можна створювати плейлисти або використовувати у якості плеєра, він здебільшого використовується як генератор переходів до стрімінгових платформ для подальшого прослуховування музики. Протягом 2018 року у Shazam було 478 мільйонів активних користувачів на рік, порівняно з 400 мільйонами, зафіксованими в 2017 році. [23] Деякі джерела пояснюють це стрімке зростання нещодавнім придбанням Shazam технічним гігантом Apple, за 400 мільйонів доларів США. [24] Apple придбала Shazam наприкінці 2018 року; це був крок, який деякі вважають суперечливим через вплив, який придбання може здійснити на конкурентів. Обговорення щодо придбання розпочалося в 2017 році, і було зазначено, що Shazam більше не буде перенаправляти користувачів на Spotify або Google Play Music, а замість цього буде обслуговувати користувачів Apple Music.

Shazam до моменту придбання використовував Spotify Web API, але після придбання сервісу, зв’язки зі Spotify були розірвані. Після цього, в API маркетплейсах з'явився Shazam Core API. [25] Рекомендації не є основною функцією застосунку, більша частина уваги розробників присвячена саме алгоритмам розпізнавання композицій. Нові можливості платформи хоч і зробили її зручнішою, проте так і залишилися в тіні основного призначення сервісу.

Доступ до цього програмного інтерфейсу надає API-провайдер Tipsters CO. Видимими перевагами є можливість безкоштовного використання з метою ознайомлення та тестування, наявність документації, низька затримка відповіді, що становить ≈573 мс.

ВИСНОВКИ ПО РОЗДІЛУ 2

За результатами порівняння, що були викладені у першому розділі, проведено систематизований аналіз технології створення подібних застосунків. Детальний аналіз особливостей аналогів допоміг виявити покращений набір характеристик розроблюваного веб-застосунку. Врахування переваг та недоліків існуючих систем допомогло визначити основні аспекти, на які слід звернути увагу, а саме кількість пропонованих фільтрів пошуку вподобань, зручність та доступність веб-застосунку, найоптимальніший варіант розробки серверної частини для якомога вищого показнику швидкодії.

Використання розробниками технологій гігантів індустрії породжує рух модернізації та вдосконалення, міксування особливостей для створення нового бачення на певні речі. Розбивка на мікросервіси та окремі застосунки спрощує процес користування, відокремлюючи непотрібні користувачу функції та вдосконалення роботи чи презентабельності окремих можливостей.

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

3.1 Проектування та розробка веб-застосунку для підбору музичних композицій

У цьому параграфі описується процес розробки веб-застосунку для підбору музичних композицій із урахуванням вподобань. Від початку розробки до розгортання проекту на сервері буде використано різні інструменти та технології. Важливою частиною кожного проекту є архітектура програмного забезпечення. Потрібно насамперед визначити яким чином буде працювати застосунок та яка послідовність дій приведе до очікуваного результату. Для цього необхідно побудувати діаграму послідовностей веб-застосунку, яка характеризує поведінку системи відносно часу. Об’єкти взаємодіють між собою шляхом передачі інформації в рамках сценарію.

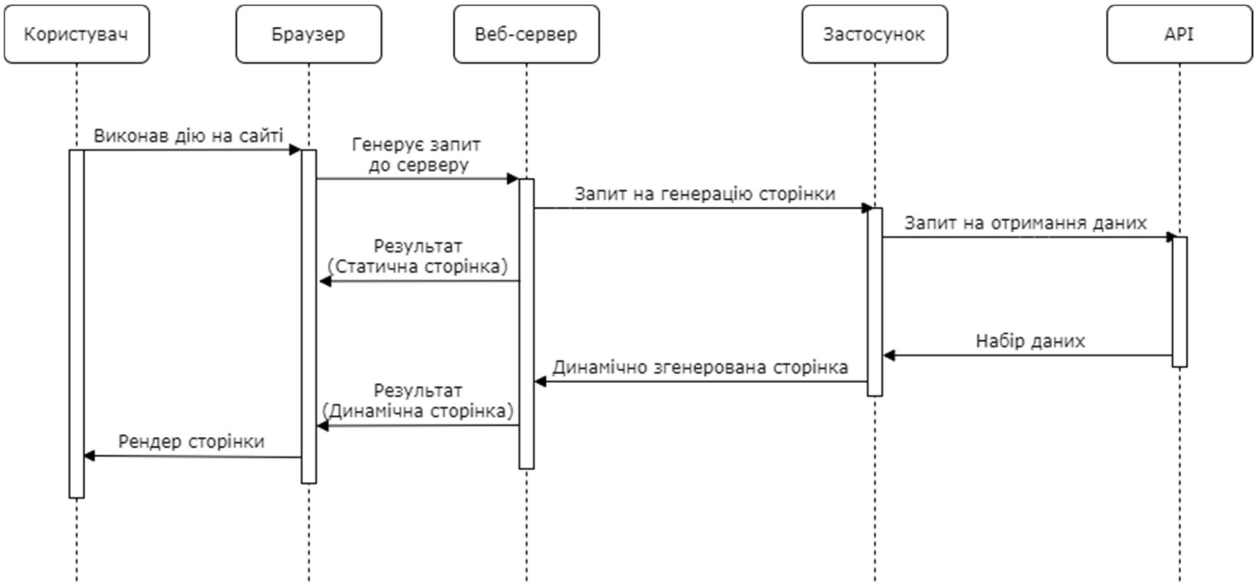

Рисунок 3.1 – Діаграма послідовностей

Користувач взаємодіє з елементами керування на сторінці. Після реєстрації дій користувача, браузер формує запит до серверу для виконання певної дії. Сервер направляє запит до адресату, в даному випадку веб-застосунку, та в залежності від наповнення запиту, застосунок створює запит на отримання даних від API. Після отримання відповіді від API, застосунок

визначає послідовність та спосіб генерації структури сторінки на основі отриманих даних та передає сформований файл розмітки до браузеру через шлюз веб-серверу. Браузер отримує відповідь від серверу та представляє користувачу результат його запиту.

Сучасні веб-застосунки мають прості та інформативні URL адреси. Це допомагає людям запам'ятати ці URL, особливо зручно для використання з мобільних пристроїв або пристроїв з повільним мережевим з'єднанням. Якщо користувач може перейти відразу на бажану сторінку, без попереднього відвідування початкової сторінки, він з більшою ймовірністю повернеться на цю сторінку і в наступний раз. [26] Для цього використовують декоратор route(), за допомогою якого можна створювати статичні та динамічні адреси.

```python
@app.route('/')
def home():
    return render_template('home.html', title='home')


@app.route('/results')
def results():
    result = request.form['request']
    return render_template('results.html', result=result)


@app.route('/recommendations/<string:trackid>', methods=['GET'])
def resultofclick(trackid):
    result = trackid
    data = getrecommendations(result)
    song = getsongdetails(result)["share"]["subject"]
    return render_template('songs-page.html', result=data, title='Results of searching', songName=song)
```

Рисунок 3.2 – Використання декоратору для створення шляхів

Правила для URL, що працюють в Flask, засновані на модулі маршрутизації Werkzeug. Цей модуль реалізований у відповідність з ідеєю забезпечення гарних і унікальних URL-адрес.

До кожного декоратора прив’язаний особистий метод, що виконується при переході по вказаному маршруту.

```
Endpoint             Methods    Rule
-------------------- ---------  --------------------------------
artistpage           GET        /artist/<string:artistid>/songs
bygenre              GET, POST  /by-genre
favicon              GET        /favicon.ico
home                 GET        /
resultofclick        GET        /recommendations/<string:trackid>
results              GET        /results
searchresult         GET, POST  /search-result
searchresultfromrecs GET        /search-result/<string:query>
static               GET        /static/<path:filename>
```

Рисунок 3.3 – Список маршрутів веб-застосунку

Всередині кожного такого методу знаходиться код, який виконує свій сценарій в залежності від призначення. Даний метод повинен повертати значення у вигляді html-сторінки або виконувати переадресацію на другий маршрут. Приклади такої переадресації можна спостерігати при авторизації на різних сайтах, де на невеликий час адресний рядок містить складне для розуміння посилання з різних символів, а потім змінюється на просте.

Кожне API має список кінцевих точок (endpoints), звернувшись до яких, програмний інтерфейс поверне результат, що відповідає запиту. Взаємодія з API винесена у окремий файл проекту. Всередині знаходяться методи, що виконують звернення до кінцевих точок та повертають оброблені дані у вигляді JSON. Після цього, результат додатково обробляється сервером та генерується веб-сторінка, яка відправляється до браузера для подальшого рендеру.

```python
def getartistsongs(artistid):
    url = "https://shazam-core.p.rapidapi.com/v1/artists/tracks"
    querystring = {"artist_id": artistid, "limit": "100"}
    headers = {
        'x-rapidapi-key': apikey,
        'x-rapidapi-host': "shazam-core.p.rapidapi.com"
    }
    response = requests.request(
        "GET", url, headers=headers, params=querystring)
    res = response.json()
    return res
```

Рисунок 3.4 – Метод реалізації запиту

Для можливості використовувати ендпоінти, потрібно правильно сформулювати запит. У заголовку (header) знаходиться інформація, яка описує

тип запиту, дані для автентифікації, режим роботи тощо. У тілі запиту (body) містяться параметри, які є вказівками для API в якому вигляді треба надати результат.

У процесі використання може статися неочікуваний збій у роботі застосунку, розрив з'єднання з сервером чи відмова у відповіді від серверу. На такий випадок, щоб застосунок не змінив свого вигляду та забезпечив користувача інформацією стосовно проблеми, треба реалізувати методи, які будуть відповідати за роботу застосунку у разі появи помилок. Поширеними помилками для веб-простору є помилки, код яких виглядає як 4.x.x або 5.x.x, де 4.x.x – помилка на стороні клієнта, 5.x.x – помилка на стороні сервера. [27]

```
@app.errorhandler(Exception)
def handle_exception(e):
    return render_template("error500.html", e="I cannot find anything by this request."), 500


@app.errorhandler(404)
def handle_404(error):
    return render_template("error404.html",e=error), 404
```

Рисунок 3.5 – Декоратор та методи обробки помилок

Для реалізації було використано декоратор errorhandler(), який може приймати в себе аргумент типу помилки. Якщо під час користування виникає певний тип помилки, викликається метод, що прив'язаний до декоратора обробника цього типу помилки, і повертає користувачеві сторінку з поясненням виникнення помилки і пропонує перейти на головну сторінку або продовжити пошук. У проекті створена папка «templates», всередині якої знаходяться усі html файли застосунку.

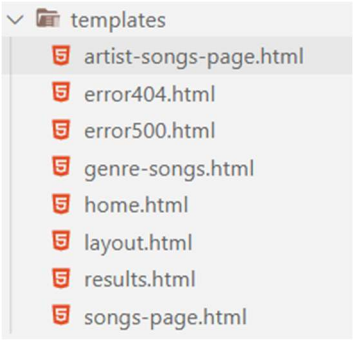

Рисунок 3.6 – Структура html-сторінок веб-застосунку

Кожна з сторінок має свою функцію, наприклад, home.html відповідає за розмітку головної сторінки, songs-page.html є шаблоном для відображення рекомендованих музичних композицій тощо. Особливу увагу приділено сторінці layout.html, яка є батьківською для усіх інших сторінок. Вона має базову структуру, підключені стилі та скрипти, які потрібні на кожній сторінці. За допомогою фреймворка-шаблонізатора Jinja у процесі розробки використано спеціальну мову розмітки, щоб отримать можливість підключати та включати шаблони сторінок одне до одного. Розробка розмітки стає динамічною, шаблони можна використовувати для рендеру різних сторінок, більше не потрібно дублювати код. За допомогою специфічних тегів {%extends ‘назва шаблону’%} та {%include ‘назва шаблону’%} можна вказати шаблону, який файл розмітки потрібно внаслідувати. Також можна перевизначити блоки через теги {% block ‘назва блоку’%}{% endblock%}.

```
<main role="main" class="container">
    <div id="particles-js"></div>
    <div id="page-wrapper">
        <div class="row">
            <div class="col">
                {% block content %}{% endblock %}
            </div>
        </div>
    </div>
</main>
```

Рисунок 3.7 – Реалізація тегу {%block%} в layout.html.

Розробимо сторінку для видачі результатів пошуку results.html. Вона буде унаслідувати структурні теги, підключені скрипти та стилі від базового шаблону. Для цього нам потрібно:

1. Створити файл розмітки та прописати успадкування від базового шаблону.
2. Визначити контент шаблону всередині тегу {%block%}.
3. Використовуючи бібліотеку стилів Bootstrap, розробити дизайн сторінки та кожного елементу, який буде передано з серверу.
4. Передати дані для генерації динамічного контенту.

На рисунку 3.8 показано фрагмент коду сторінки results.html.

```
{% extends "layout.html" %}
{% block content%}

<div class="row">
    <h2>Results for searching: {{query}}</h2>
</div>
</br>
<div class="row">
    <div class="col">
        <h3>Artists</h3>
        </br>
        {% for artist in artists %}
        <div class="card mb-3" style="max-width: 100%;">
            <div class="row no-gutters">
                <div class="col-md-3">
                    <a href="/artist/{{artist.id}}/songs">
                        <img src="{{artist.avatar.default}}" class="card-img" alt="Artist picture"
                            onerror="this.src='/static/images/placeholder.jpg';">
                    </a>
                </div>
                <div class="col-md">
                    <div class="card-body">
                        <h5 class="card-title">{{artist.name}}</h5>
                        <a class="card-text" href="/artist/{{artist.id}}/songs">Go to artist songs</a>
                    </div>
                </div>
            </div>
        </div>
        {% endfor %}
    </div>
```

Рисунок 3.8 – results.html

Аналогічним методом створено інші шаблони розмітки, опис призначення яких наведено у списку:

1. home.html – Головна сторінка яку бачить користувач при першому візиті до сайту. Містить у собі текстове поле для пошуку та одноіменні кнопки для пошуку музики за жанрами.
2. layout.html – Базова сторінка від якої інші шаблони успадковують стиль оформлення, структуру та поведінку.
3. results.html – Показує результат виконання пошуку за запитом. На екрані зображується дві колонки, які виводять результати пошуку по авторам та по композиціям відповідно. Інформація подана у вигляді окремих карток для кожного елементу.
4. genre-songs.html – Відображає результати пошуку музики за жанрами. Кожна композиція представляється у окремій ноді та надає користувачу таку інформацію як обкладинка композиції, назва автора, назва композиції та музичний фрагмент. Також кожна нода містить кнопки з посиланнями на цю композицію на інших платформах.

5. songs-page.html – Відображає підібрані композиції на основі вибору користувача. Кожна композиція представляється у окремій ноді та надає користувачу таку інформацію як обкладинка композиції, назва автора, назва композиції та музичний фрагмент.
6. artist-songs-page.html – Відображає композиції певного автора. Кожна композиція представляється у окремій ноді та надає користувачу таку інформацію як обкладинка композиції, назва автора, назва композиції та музичний фрагмент. Також кожна нода містить кнопки з посиланнями на цю композицію на інших платформах.

Для верстки дизайну можуть використовуватися різні принципи розмітки елементів. В даному проекті реалізовано адаптивний дизайн та верстку блоками. Адаптивна верстка – дизайн, який підлаштовується (адаптується) під розмір екрану, в тому числі може відбуватися перебудова блоків з одного місця на інше, або їх заміна блоками відображеними тільки при певному вирішенні. [28] Адаптивна верстка прийшла на зміну ідеї створення спеціальних мобільних версій сайту, розташованих на окремих піддоменів. Наприклад, на просторах Інтернету можна зустріти такі піддомени як «m.site.com».

Блоки являють собою структурні елементи, які можна розміщувати на веб-сторінці шляхом накладення їх один на одного з точністю до пікселя. В HTML і XHTML блок - це елемент веб-сторінки, створений за допомогою тега , до якого застосовується стильове оформлення.

При цьому дотримуються наступних принципів:

– Розділення вмісту і оформлення.
– Активне застосування тега .
– Таблиці застосовуються тільки для представлення табличних даних.

Рисунок 3.9 – Реалізація блочної верстки сторінки

Блоки можна розміщувати у вікні браузера з точністю до пікселя. Положення блока задається двома координатами щодо будь-якого кута вікна браузера, батьківського елемента або документа. Скрипти дозволяють змінювати параметри блоку динамічно. Це дає можливість створювати на сторінці різні ефекти, такі як випадаючі меню, банери, плаваючі вікна та інше.

Так як застосунок може відправляти POST-запити на сервер, для уникнення помилок відповіді потрібно проводити валідацію форм. Це правило стосується полів пошуку, так як через них формуються запити до серверу. У даному випадку, проблема може виникнути при відправці порожнього запиту, на що сервер поверне помилку з кодом 400 (Bad Request). Помилка вказує на те, що сервер не зміг обробити (зрозуміти) запит, надісланий клієнтом, через неправильний синтаксис, неправильне оформлення повідомлення запиту або оманливу маршрутизацію запиту.

Цю проблему вирішено шляхом відключення кнопки пошуку при порожньому полі. Кожній формі та полю пошуку було призначено унікальне Id, що можна бачити на рис. 3.10.

```
    <div class="row justify-content-center">
        <form class="form-inline" method="POST" action="/search-result" id="form_home">
            <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search" id="searchbar_home"
                name="searchquery">
            <button class="btn btn-outline-danger my-2 my-sm-0" type="submit">Search</button>
        </form>
    </div>
</div>
```

Рисунок 3.10 – Налаштування форми пошуку

Селектор відслідковує стани цих елементів і відключає взаємодію з формою у разі пустого поля пошуку, що не дозволяє відправити порожній запит на сервер, як показано на рис. 3.11.

```
$(document).ready(function () {
  $('#form_home').submit(function() {
    if ($.trim($("#searchbar_home").val()) === "") {
      return false;
    }
});
```

Рисунок 3.11 – Селектор форми

Окрім цього, в ході попереднього тестування функціоналу було виявлено проблему з програванням фрагментів композицій. При натисканні на кнопку запуску іншого фрагменту, попередній продовжував грати та створювати перешкоди для прослуховування користувачем.

```
$(function(){
  $("audio").on("play", function() {
      $("audio").not(this).each(function(index, audio) {
        audio.pause();
      });
  });
});
```

Рисунок 3.12 – Селектор для тегу <audio>

Для цього було створено селектор для тегу <audio> та встановлено опцію можливості роботи лише одного такого елементу в один момент часу. Таким чином, при натисканні на кнопку запуску іншого музичного фрагменту, попередній автоматично завершувався.

Щоб налаштувати проект для розгортання на хмарній платформі Heroku, потрібно завантажити інтерфейс командного рядку на комп'ютер та встановити його. Далі авторизуємося в своєму обліковому записі Heroku через команду “heroku login” у терміналі. Після успішної авторизації, у кореневій

папці проекту потрібно створити файл з назвою “Procfile” без розширення. Всередині потрібно вказати тип веб-серверу, який потрібно запустити для нашого застосунку та файл, з якого буде починатися компіляція програми. Наприклад, “web: gunicorn app:app”, де web:gunicorn – інструкція, що вказує тип веб-сервера gunicorn, app:app – посилання на ім’я головного файлу застосунку.

Далі за допомогою команди “pip3 freeze > requierements.txt” нам треба створити текстовий файл requirements.txt який буде містити інформацію про версії встановлених модулів проекту.

Ініціалізуємо гіт репозиторій та реєструємо усі зміни в системі:

1. git init – ініціалізує систему контролю версій для проекту
2. git add . – додає усі файли до списку відстежування змін
3. git commit -m “message” – реєструє зміни у всіх відстежуваних файлах у вигляді стану файлів.

Для початку завантаження файлів до платформи, потрібно в терміналі виконати команду “heroku create music-seeker”, де «music-seeker» - назва веб-застосунку всередині Heroku. Після виконання команди, Heroku Git створює віддалений репозиторій, куди нам потрібно копіювати файли з локального репозиторію за допомогою команди “git push heroku master” в терміналі.

```
(venv) F:\Music recommender>git push heroku master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 385 bytes | 192.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the same version as the last build: python-3.9.5
remote:        To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.9.5
remote: -----> Installing pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 65.2M
remote: -----> Launching...
remote:        Released v17
remote:        https://music-seeker.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/music-seeker.git
   82eaabf..aa7ca1c  master -> master
```

Рисунок 3.13 – Успішний результат компіляції проекту в хмарному сервері

Після завершення переносу файлів, веб-сервер автоматично розпізнає файл Procfile та збирає застосунок на сервері. В разі успішного завершення збірки, у консолі термінала з’явиться посилання на веб-застосунок в Інтернеті.

3.2 Інструкція користувача

Даний веб-застосунок призначений для пошуку музичних композицій та підбору музичних композицій на основі вподобань користувача. Щоб розпочати роботу, потрібно перейти за посиланням https://music-seeker.herokuapp.com/ , після чого користувач опиниться на головній сторінці веб-застосунку.

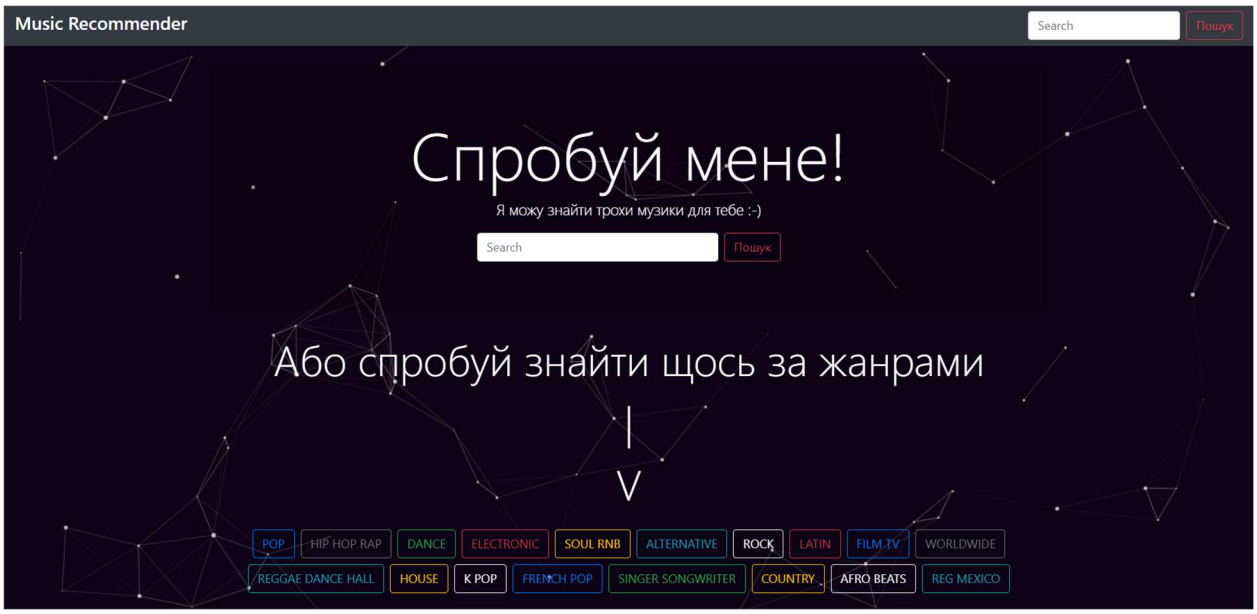

Рисунок 3.14 – Головна сторінка веб-застосунку

На головній сторінці розташовані елементи, які можна використати для пошуку композиції у два способи: пошук за текстом та пошук за жанрами. Для пошуку за текстом, користувач має увести свій запит в поле пошуку та натиснути кнопку «Search». Застосунок переносить користувача на сторінку, на якій відображаються результати пошуку, якщо вони є. Ця сторінка має дві колони: у лівій розташовані знайдені автори, а у правій композиції.

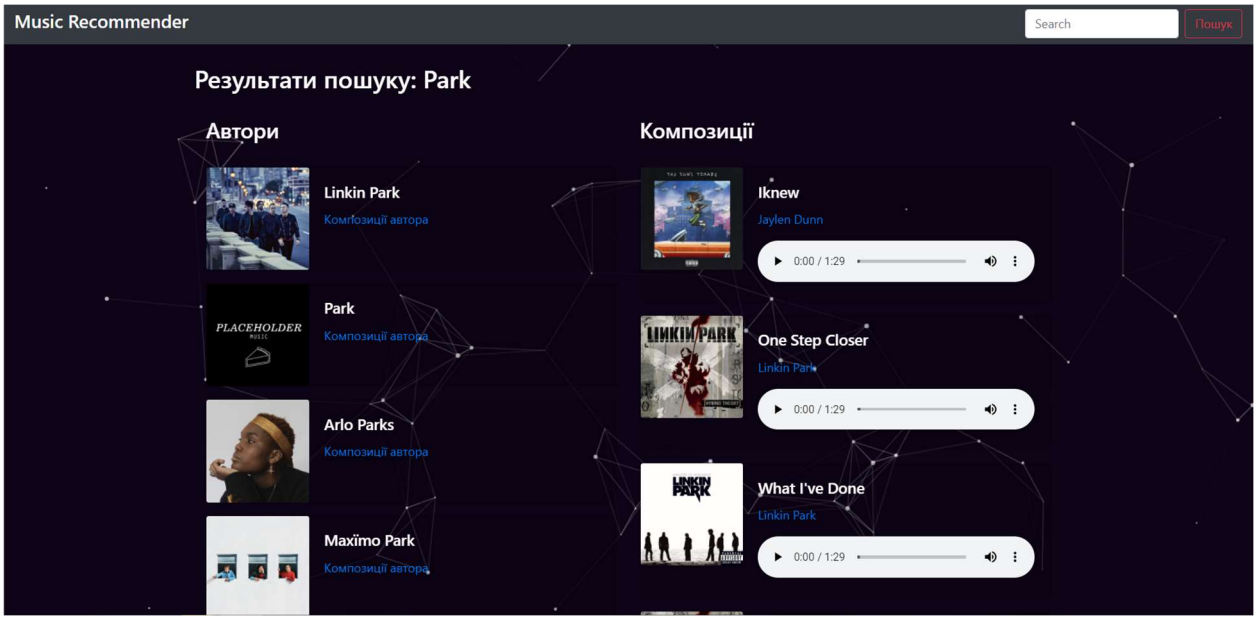

Рисунок 3.15 – Результати пошуку за запитом

У противному випадку, користувач побачить сторінку на рис. 3.6, яка надає пояснення щодо її появи.

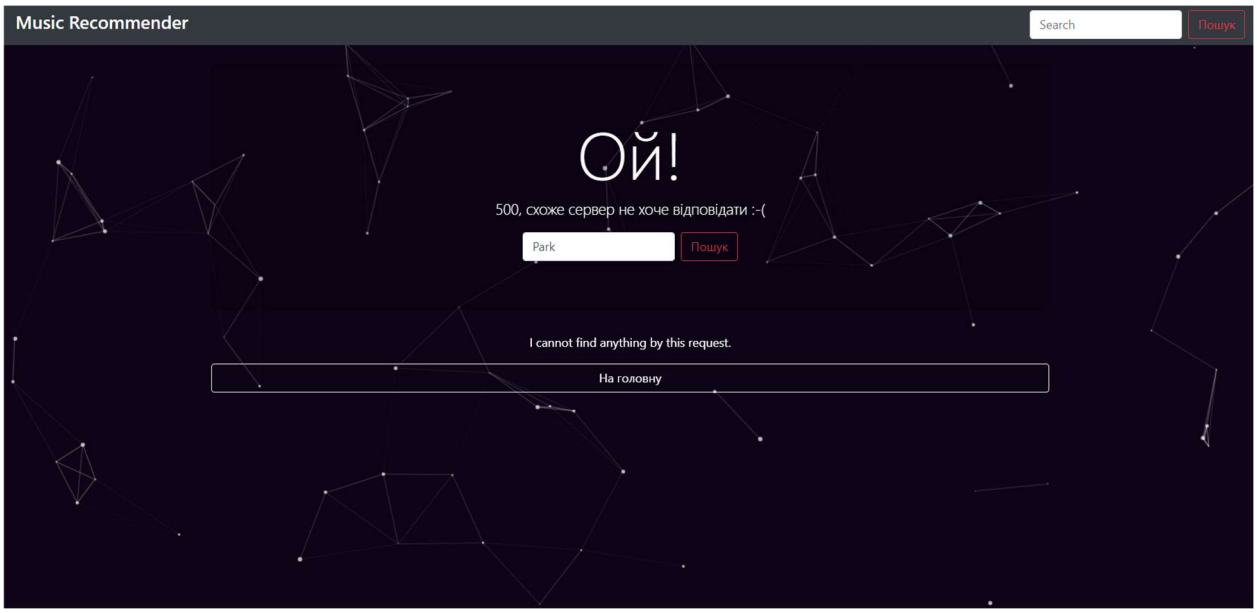

Рисунок 3.16 – Сторінка помилкового запиту

Знайдені композиції можна прослухати щоб переконатися, чи є композиція тією, що шукав користувач. При наведенні курсору миші на зображення, з'являється підказка з текстом «Search related».

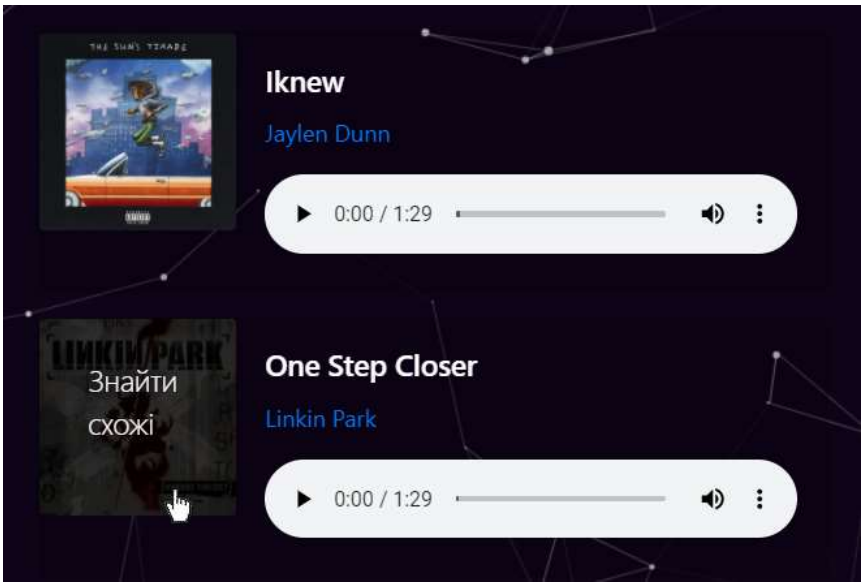

Рисунок 3.17 – Вигляд підказки

При натисканні на зображення відбувається перехід на сторінку зі схожими музичними творами.

Сторінка з рекомендованими творами містить окремі картки, на кожній з яких розміщена інформація, що стосується окремої композиції. Користувач може бачити назву твору, ім’я чи псевдонім автора, обкладинку, програвач музичного фрагменту та кнопки для переходу на сторонні платформи. При натисканні на обкладинку, знову з’являється підказка, що зображення є посиланням на сторінку зі схожими композиціями і виконує оновлення поточної сторінки, але заповнюючи новими даними.

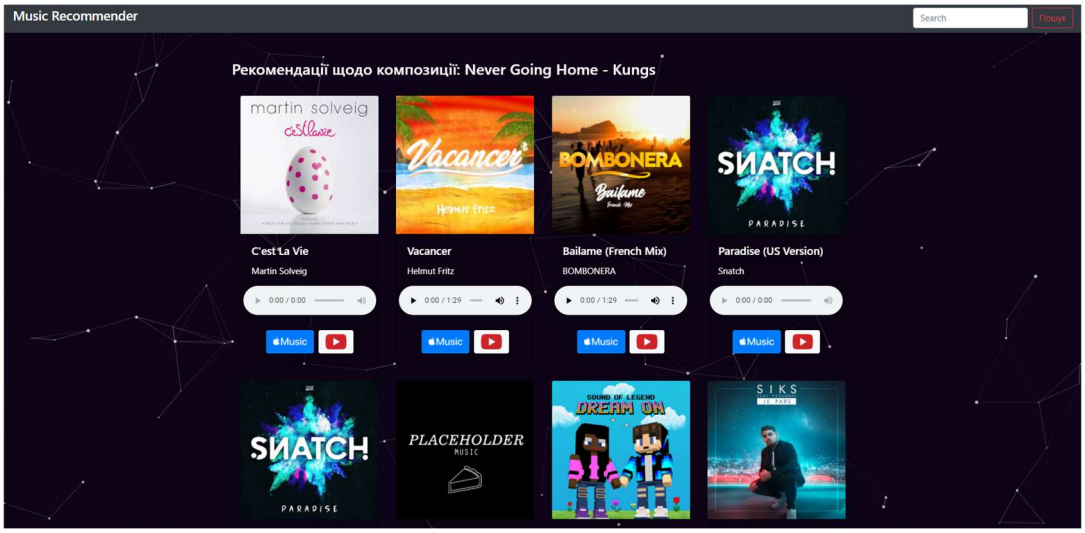

Рисунок 3.18 – Сторінка з рекомендаціями

Повернемося назад до сторінки з результатами пошуку та оберемо елемент з колонки виконавців. Для перегляду композицій цього автора, Потрібно натиснути на зображення автора або його ім’я. Користувачу відкриється сторінка з переліком композицій вибраного автора. Аналогічним чином, при наведенні курсору на обкладинку композиції з’являється підказка і по кліку виконується перехід на сторінку з рекомендаціями.

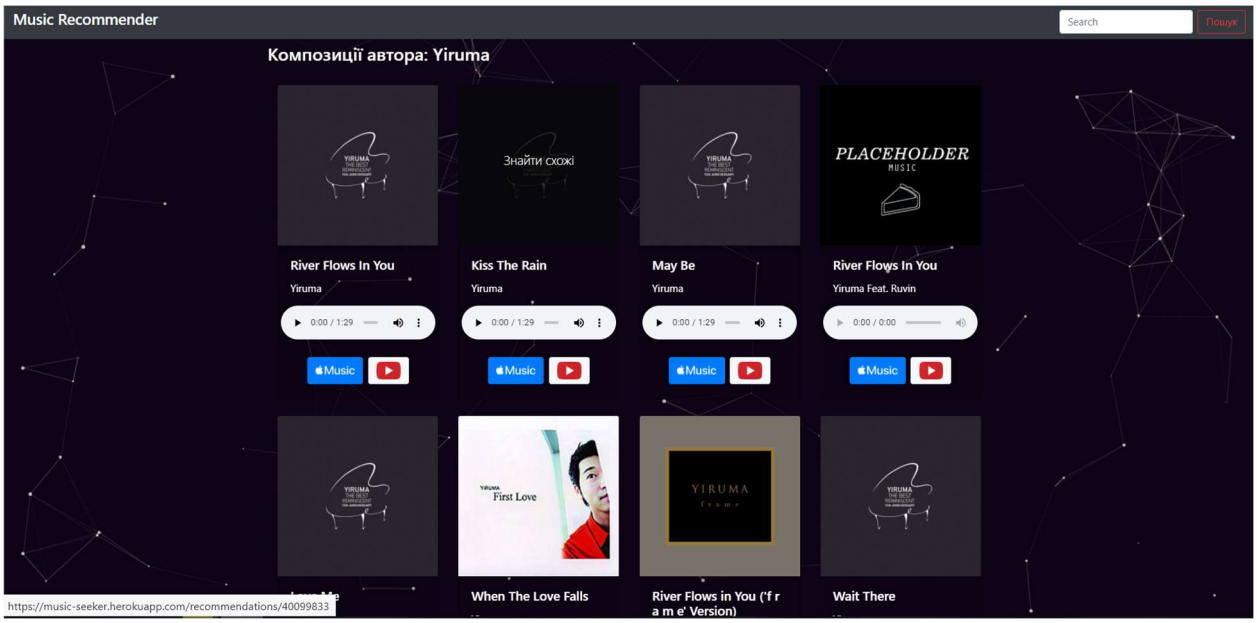

Рисунок 3.19 – Сторінка композицій автора

Для пошуку музики за жанрами, потрібно прогорнути до нижньої частини сторінки, куди вказує стрілочка, і натиснути на кнопку з назвою жанру.

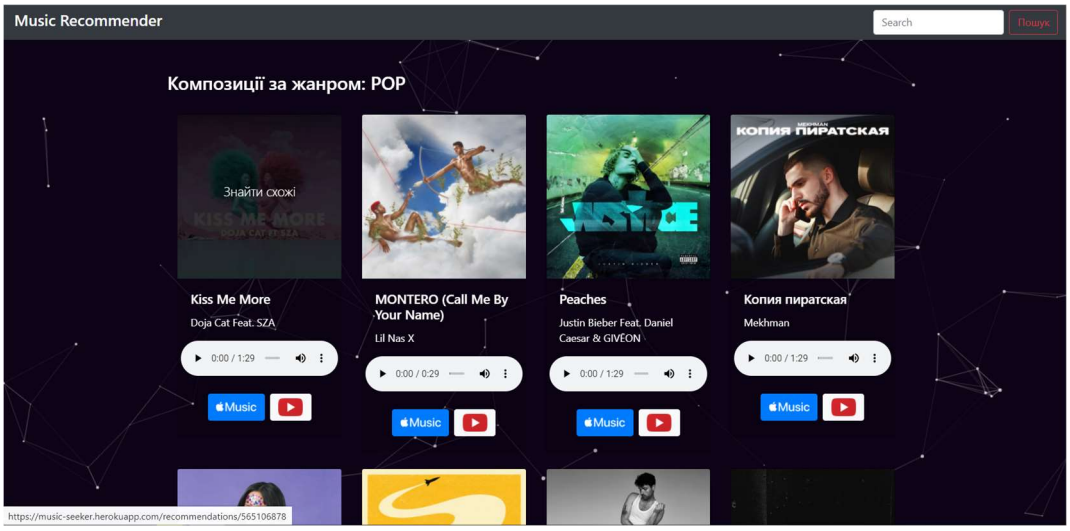

Рисунок 3.20 – Результат пошуку за жанром «Pop»

Користувача направляє на сторінку з композиціями, відібраними по обраному критерію. Аналогічним чином, при наведенні курсору на обкладинку композиції з'являється підказка і по кліку виконується перехід на сторінку з рекомендаціями.

3.3 Тестування та аналіз отриманих результатів

У цьому параграфі проведено тестування веб-застосунку для підбору музичних композицій з урахуванням та аналіз одержаних результатів та їх порівняння з поставленими задачами на розробку веб-застосунку.

Технічні та функціональні вимоги для проектування та розробки веб-застосунку базуються на сучасних методологіях проектування та розробки програмного забезпечення. У процесі реалізації проекту максимальну кількість часу було уділено саме проектуванню системи та підбору оптимального набору інструментів та технологій розробки.

Для порівняння отриманого результату, потрібно провести оцінку роботи веб-застосунку та його продуктивність аналогічно до тестування існуючих веб-застосунків. Показники оцінювання наведено нижче в табл. 3.1.

Таблиця 3.1 Результати тестування веб-застосунку Music Recommender

Критерій	Показники
Швидкість надання рекомендацій	≈724 мс
Зручність використання	Мінімалістичний та зручний, інтуїтивно зрозумілий та простий у використанні, має адаптивний привітливий дизайн
Способи пошуку вподобань	Пошук за жанрами, ім'ям автору чи назвою композиції
Доступність	Повністю безкоштовний для користувача, відсутній механізм обов'язкової авторизації, відразу доступний повний функціонал
Релевантність вподобань	Є можливість переконатися у релевантності наданих рекомендацій шляхом прослуховування фрагменту композиції

Вимірювання швидкості проводилося інструментами браузеру Opera. Було виконано 10 запитів до серверу у різні проміжки часу та обраховано середній час витрат на генерацію сторінки з вподобаннями. Дані показники є трохи кращими аніж у аналогу Gnoosic (≈838 мс), навіть дивлячись на те, що розроблений застосунок повертає набагато більше інформації для користувача та дозволяє виконувати більше взаємодій. Оскільки Gnoosic показав себе найшвидшим з аналогів, то можна зробити висновок про кращу продуктивність розробленого застосунку відносно всіх аналогів. Таким чином, Music recommender працює швидше за Gnoosic на 15,7%, майже на 56,5% швидше за Magic Playlist та з дуже великою різницею швидше за Itcher (більше ніж в 12 разів).

Тестування зручності використання направлене виявити наскільки об’єкт тестування відповідає критеріям ергономічності, лаконічності та зручності. [29]

Дизайн застосунку виконаний у темних кольорах, що комфортно при перегляді. Білі відтінки можуть занадто напружувати людське око, особливо при високому показнику яскравості екрану девайсу, що буде спричиняти більший дискомфорт при перегляді і відштовхне користувача від тривалого використання.

Застосунок локалізований на англійській мові, що робить його зручним для багатьох країн. Усі елементи притримуються мінімалістичного стилю та інтуїтивно зрозумілі для користувача. В лівому верхньому кутку знаходиться назва застосунку, яка водночас є активним посиланням на головну сторінку. Так зроблено для того, щоб не допустити потрапляння користувача на сторінку з якої не можна вибратися. Всі кнопки застосунку містять інформацію щодо дій, які виконують. Якщо результат пошуку відсутній, застосунок повертає повідомлення, що нічого не знайшов та пропонує пошукати ще раз. Музичні фрагменти можна відрегулювати по гучності та поставити на паузу.

Для людей, які мають вади зору, додано можливість масштабування сторінки без порушення структури розмітки та альтернативний опис графічних елементів на сторінках.

Основним джерелом даних веб-застосунку для підбору музичних композицій є програмний інтерфейс платформи Shazam. У проекті реалізовано необхідні функції для нормальної презентації застосунку, як вузьконаправленого сервісу з підбору композицій на основі вподобань. Реалізація програмного інтерфейсу має нюанс у вигляді плати провайдеру за користування API. Для реалізації проекту було достатньо і базового пакету можливостей, але для повномасштабного запуску застосунку потрібно отримати більш розширений список можливостей від провайдера. Веб-застосунок, в теорії, може давати прибуток, якщо розміщувати в ньому рекламу.

Порівняно зі своїми аналогами Gnoosic, Itcher та Magic Playlist, перевагами Music Recommender є:

1. Безкоштовність використання веб-застосунку та відсутність обов’язкової авторизації користувача.
2. Релевантність підібраних вподобань користувач може перевірити шляхом прослуховування музичного фрагменту композиції.
3. Достатня кількість варіантів визначення вподобань у вигляді пошуку за жанром, пошуку актора та окремої композиції.
4. Адаптивний дизайн, що дозволяє користуватися веб-застосунком з різних пристроїв без недоліків відображення контенту на сторінках.

Серед недоліків застосунку, слід виокремити роботу самого API через некоректну роботу на деякі запити та специфічну структуру деяких відповідей. Не усі повернені результати мають у своєму описі посилання на зображення артиста чи обкладинки композиції та музичні фрагменти, але така проблема також прослідковується у Magic Playlist та Itcher. Це впливає на результати відображення контенту на сторінках та взаємодії з користувачем.

У якості прогнозу на основі аналізу, можна додати, що застосунок має потенціал для розвитку та залучення інвестицій. Вдосконалення та розширення можливостей застосунку зможуть підвищити його продуктивність та корисність.

ВИСНОВКИ ПО РОЗДІЛУ 3

На основі вихідних даних аналізу першої та другої частини, спроектовано покрокову стратегію розробки та підібрано актуальний та потужний технологічний стек розробки. Роботу над застосунком було розділено на розробку серверної та клієнтської частини окремо, а потім зведення їх до цілісного вигляду, детально описавши процес програмування кожної частини. Застосунок розміщено на хмарній платформі, що дозволяє користувачам отримати доступ з усіх куточків світу.

Створено коротку інструкцію користувача по використанню веб-застосунку. Детальний розпис інструкції дозволяє користувачу краще зрозуміти можливості експлуатації застосунку для переслідування особистих цілей.

По закінченню робіт над проектом, проведено тестування даного застосунку з метою виявлення показників відповідності до поставленої задачі та функціональних вимог. За результатами тестування виконано порівняння продуктивності розроблюваного застосунку відносно аналогів, способів пошуку вподобань, зручність використання застосунку, доступність та можливість оцінки релевантності вподобань. На момент написання висновку, веб-застосунок Music Recommender працює за посиланням https://music-seeker.herokuapp.com/.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lepper, Mark R. "When choice is demotivating: Can one desire too much of a good thing?". *Journal of Personality and Social Psychology*. 2000. Vol.79 (6) (дата звернення: 26.05.2021).

2. Reijo Savolainen. "Filtering and withdrawing: strategies for coping with information overload in everyday contexts". *Journal of Information Science.* 2007. Vol 33. (дата звернення: 26.05.2021).

3. Soucek and Moser. "Coping with Information Overload in Email Communications: Evaluation of A Training Intervention". *Computers in Human Behavior*. 2010. Vol.26 (6) (дата звернення: 26.05.2021).

4. Top 10 artists 1990–2000. *Avyanna* : веб-сайт. URL: https://avyanna.net/en/blog/ (дата звернення: 26.05.2021).

5. Music genre. *Wikipedia :* веб-сайт. URL: https://en.wikipedia.org/wiki/Music_genre#cite_note-2 (дата звернення: 26.05.2021).

6. Roger B. Dannenberg. Style in Music. *Springer-Verlag.* 2010. Vol.1 (дата звернення: 26.05.2021).

7. The World's Favorite Music Genres. *Statista* : веб-сайт. URL: https://www.statista.com/chart/15763/most-popular-music-genres-worldwide/ (дата звернення: 26.05.2021).

8. Music evokes at least 13 emotions. Scientists have mapped them. *Berkeley News* : веб-сайт. URL: https://news.berkeley.edu/2020/01/06/music-evokes-13-emotions/ (дата звернення: 26.05.2021).

9. Gnoosic. *Gnoosic* : веб-сайт. URL: https://www.gnoosic.com/ (дата звернення: 26.05.2021).

10. Magic Playlist. *Spotify* : веб-сайт. URL: https://developer.spotify.com/community/showcase/magic-playlist/ (дата звернення: 19.03.2021).

11. Itcher. *Itcher* : веб-сайт. URL: http://itcher.com/my/recommendations/ (дата звернення: 04.04.2021).

12. GNOD API. *Gnod* : веб-сайт. URL: https://www.gnod.com/ (дата звернення: 04.04.2021).

13. *Spotify for Developers* : веб-сайт. URL: https://developer.spotify.com/ (дата звернення: 04.04.2021).

14. Захист інформації у сучасному світі. *РемОнлайн* : веб-сайт. URL: https://remonline.ua/blog/protection-of-information-in-modern-world/ (дата звернення: 04.04.2021).

15. What is a Web Application? *Stackpath* : веб-сайт. URL: https://blog.stackpath.com/web-application/ (дата звернення: 15.03.2021).

16. Web Apps vs. Native Apps: Advantages & Disadvantages. *Jacapps* : веб-сайт. URL: https://jacapps.com/web-apps-vs-native-apps-advantages-disadvantages/ (дата звернення: 15.03.2021).

17. Modern Web Application Architecture Explained: Components, Best Practices and More. *Listlink* : веб-сайт. URL: https://litslink.com/blog/web-application-architecture (дата звернення: 15.03.2021).

18. Choosing a technology stack for web application development. *Light-it* : веб-сайт. URL: https://light-it.net/blog/choosing-a-technology-stack-for-web-application-development/ (дата звернення: 15.03.2021).

19. Backend development choice #1: Python. *MerixStudio* : веб-сайт. URL: https://www.merixstudio.com/blog/backend-development/ (дата звернення: 15.05.2021).

20. Stack Overflow Developer Survey 2020.*Stack Overflow* : веб-сайт. URL: https://insights.stackoverflow.com/survey/2020#most-popular-technologies (дата звернення: 15.05.2021).

21. Flask vs Django: choosing Python framework. *MerixStudio* : веб-сайт. URL: https://www.merixstudio.com/blog/flask-vs-django-choosing-python-framework/ (дата звернення: 15.05.2021).

22. About Shazam. *Shazam* : веб-сайт. URL: https://www.shazam.com/company (дата звернення: 19.03.2021).

23. Shazam Statistics and Facts (2021). *DMR* : веб-сайт. URL: https://expandedramblings.com/index.php/shazam-statistics/ (дата звернення: 19.03.2021).

24. It's official: Apple is buying Shazam. *Insider* : веб-сайт. URL: https://www.businessinsider.com/its-official-apple-is-buying-shazam-2017-12 (дата звернення: 19.03.2021).

25. Shazam Core API. *RapidApi* : веб-сайт. URL: https://rapidapi.com/tipsters/api/shazam-core/details (дата звернення: 19.03.2021).

26. Flask Routing. *Palletsprojects* : веб-сайт. URL: https://flask.palletsprojects.com/en/1.1.x/quickstart/ (дата звернення: 26.04.2021).

27. How to Fix a 400 Bad Request Error [Causes and Fixes]. *Kinsta* : веб-сайт. URL: https://kinsta.com/knowledgebase/400-bad-request/#causes (дата звернення: 07.05.2021).

28. Блочная вёрстка. *Htmlbook* : веб-сайт. URL: http://htmlbook.ru/samlayout/blochnaya-verstka (дата звернення: 07.05.2021).

29. Usability Testing: the Key to Design Validation. *Moodup* : веб-сайт. URL: https://moodup.team/blog/usability-testing-the-key-to-design-validation/ (дата звернення: 12.05.2021).