ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Telegram-бот пошуку та підбору пісень з використанням рекомендаційної системи»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп’ютерні науки»
Освітня програма «Комп’ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав:здобувач 4 курсу
групи КН-21
Артем ДУДНЯК
Керівник: викладач кафедри
комп’ютерних наук
Богдан БОЙКО

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2025 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри
комп'ютерних наук
_____Сергій МІЧКІВСЬКИЙ
«_____»_____20__р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дудняк Артем Олександрович

Тема роботи	Telegram-бот пошуку та підбору пісень з використанням рекомендаційної системи
Номер та дата наказу про затвердження теми	№121-7 від 24 грудня 2024 року
Коротка постановка завдання	Розробити Telegram-бота для пошуку та підбору пісень на основі рекомендаційних систем. Розробити та реалізувати систему збереження та пошуку музичних треків, створити зручний інтерфейс взаємодії з користувачем через Telegram та реалізувати алгоритм обробки запитів для пошуку й пропонування пісень.
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1.Як підключити Python до баз даних SQL // SharpCoder URL: https://uk.sharpcoderblog.com/blog/how-to-connect-python-to-sql-databases 2.Як створити телеграм-бота на Python// SpaceLab – URL: https://spacelab.ua/articles/yak-stvoriti-telegram-bota-na-python/
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має передбачити теоретичне, системотехнічне або експериментальне дослідження складного спеціалізованого завдання або практичної проблеми в галузі комп'ютерних наук, яке характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 27 грудня 2024 р.

Керівник Богдан БОЙКО

Здобувач освітнього ступеня бакалавра Артем ДУДНЯК

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
	Підготовчий етап		
1	Вибір напряму дослідження	02.12.2024 р.	*виконано*
2	Формування теми та призначення керівника	16.12.2024 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	23.12.2024 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	27.12.2024 р.	*виконано*
	Основний етап		
5	Розробка концепції кваліфікаційної роботи	13.01.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	20.01.2025 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	10.03.2025 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	24.03.2025 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	31.03-04.04.2025 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	07.04.2025 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	14.04.2025 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	21.04.2025 р.	*виконано*
	Завершальний етап		
13	Представлення рукопису для перевірки на плагіат	28.04-04.05.2025 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	05.05-11.05.2025 р.	*виконано*
15	Передзахист кваліфікаційної роботи	12.05-16.05.2025 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	19.05-06.06.2025 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	09.06-15.06.2025 р.	*виконано*
18	Доопрацювання доповіді та презентації для захисту	09.06-15.06.2025 р.	*виконано*
19	Захист кваліфікаційної роботи	16.06-22.06.2025 р.	*виконано*

Керівник Богдан БОЙКО

Здобувач освітнього ступеня бакалавра Артем ДУДНЯК

Дудняк А.О. Telegram-бот пошуку та підбору пісень з використанням рекомендаційної системи

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 – Комп'ютерні науки (освітня програма – Комп'ютерні науки) СО Бакалавр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2025.

Розглянуто проблеми автоматизованого пошуку та рекомендацій музичних треків на прикладі розробки Telegram-бота. Реалізовано Telegram-бот для пошуку та підбору пісень на основі бази даних, з використанням алгоритмів збереження та пошуку музичних треків. Створено зручний інтерфейс взаємодії з користувачем через Telegram, а також розроблено алгоритм обробки запитів для ефективного пошуку та пропонування пісень.

Ключові слова: Telegram-бот, база даних, рекомендаційна система, пошук музичних треків, обробка запитів.

Dudnyak A.O. Telegram bot for searching and selecting songs using a recommendation system

Explanatory note to the qualification work in the specialty 122 – Computer Science (educational programme – Computer Science) SO Bachelor. – Higher Educational Institution 'University of Economics and Law 'KROK", Educational and Scientific Institute of Information and Communication Technologies, Department of Computer Science, Kyiv, 2025.

The problems of automated search and recommendations of music tracks are considered using the example of developing a Telegram bot. A Telegram bot was implemented to search and select songs based on a database, using algorithms for storing and searching music tracks. A user-friendly interface was created for interaction with users via Telegram, and an algorithm was developed for processing requests for effective song search and recommendation.

Translated with DeepL.com (free version)Keywords: Telegram bot, database, music track search, recommendation system, query processing.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний світовий музичний ринок представлений великою кількістю композицій, що ускладнює процес пошуку відповідної музики для користувачів. Рекомендаційні системи стали важливим інструментом для персоналізації музичних уподобань, і багато провідних компаній, таких як Spotify, Apple Music та YouTube Music, активно використовують алгоритми машинного навчання для вдосконалення рекомендацій. Дослідження в цій галузі проводять як українські, так і закордонні вчені, серед яких слід відзначити роботи F. Ricci, L. Rokach, B. Shapira, присвячені аналізу та побудові рекомендаційних систем. В Україні дослідження алгоритмів рекомендаційних систем виконували О. Г. Береза, О. В. Сидоренко та інші.Проте, незважаючи на значний прогрес у цій сфері, існує проблема відсутності простих та доступних інструментів для швидкого пошуку та підбору музики у звичних месенджерах. Більшість існуючих сервісів є окремими додатками, які потребують окремого встановлення та налаштувань. Використання Telegram-бота як платформи для реалізації музичної рекомендаційної системи дозволить спростити доступ до персоналізованого підбору музики, що підвищує актуальність даного дослідження.

Мета дослідження – розробка Telegram-бота, що реалізує систему автоматичного підбору музики на основі рекомендаційних алгоритмів.

Завдання дослідження – вивчити існуючі методи та алгоритми музичних рекомендаційних систем:

- спроєктувати Telegram-бота з урахуванням принципів взаємодії з користувачем;
- реалізувати Telegram-бот із вбудованою системою рекомендацій;
- протестувати та оцінити ефективність роботи створеної системи;
- розробити пропозиції щодо подальшого вдосконалення бота.

Об'єкт дослідження – процес автоматизованого підбору музичних композицій з використанням рекомендаційних систем.

Предмет дослідження – Telegram-бот, що використовує алгоритми машинного навчання та API музичних сервісів для пошуку та рекомендації музики.

Методи дослідження. У роботі використовуються методи аналізу алгоритмів рекомендаційних систем, методи машинного навчання, розробка та інтеграція API, методи програмування чат-ботів на платформі Telegram.

Практичне значення. Запропоноване рішення дозволить користувачам швидко знаходити нову музику у зручному форматі без необхідності використання окремих додатків. Telegram-бот може використовуватись як самостійний сервіс або інтегруватися в інші музичні платформи.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. Загальний обсяг роботи складає 36 сторінок, основний зміст викладено на 35 сторінках.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ З РОЗРОБКИ TELEGRAM-БОТА ПОШУКУ ТА ПІДБОРУ ПІСЕНЬ

1.1 Опис предметної області

Розроблений Telegram-бот для пошуку та підбору пісень є інноваційним інструментом, що дозволяє користувачам зручно і швидко знаходити музику відповідно до їхніх смаків, використовуючи рекомендаційні алгоритми. Головна мета цього проєкту - спростити доступ до нових музичних композицій та створити персоналізований досвід для кожного користувача, не обмежуючи його можливості платними підписками чи обмеженнями на кількість прослуховуваних пісень.

Напрямки застосування:

- зручний пошук музики – користувач може швидко знаходити пісні за ключовими словами та виконавцем, що значно спрощує доступ до нового контенту;
- відсутність обмежень та повна безкоштовність – бот не містить платних функцій або обмежень у використанні, що робить його доступним для всіх охочих;
- соціальна взаємодія – користувачі можуть ділитися своїми улюбленими треками та плейлистами з друзями безпосередньо у Telegram;
- масштабованість - оскільки бот працює на платформі Telegram, користувачі можуть без проблем підключатися до нього на будь-якому пристрої, чи то смартфон, чи комп’ютер. Це робить його доступним для широкої аудиторії без необхідності додаткових налаштувань;
- інтуїтивно зрозумілий інтерфейс - інтерфейс бота розроблений так, щоб користувач міг легко здійснювати пошук пісень.

1.2 Визначення потенційних конкурентних переваг

Spotify (рис. 1.1) – це один із найбільш популярних музичних стрімінгових сервісів. Він пропонує користувачам доступ до мільйонів пісень, підкастів та відео, створюючи персоналізовані рекомендації за допомогою складних алгоритмів на основі вподобань користувача [3].

Переваги:

– великий каталог музики - доступ до мільйонів пісень, що охоплюють практично всі жанри музики;
– інтелектуальна рекомендаційна система - алгоритм пропонує нові пісні на основі ваших вподобань;
– зручність використання на різних пристроях - Spotify доступний на смартфонах, ПК, планшетах та розумних колонках.

Недоліки:

– платний доступ - більшість функцій доступні тільки для преміум-користувачів (без реклами, офлайн-доступ);
– обмеження на безкоштовному плані - реклама, обмеження на пропускання треків.

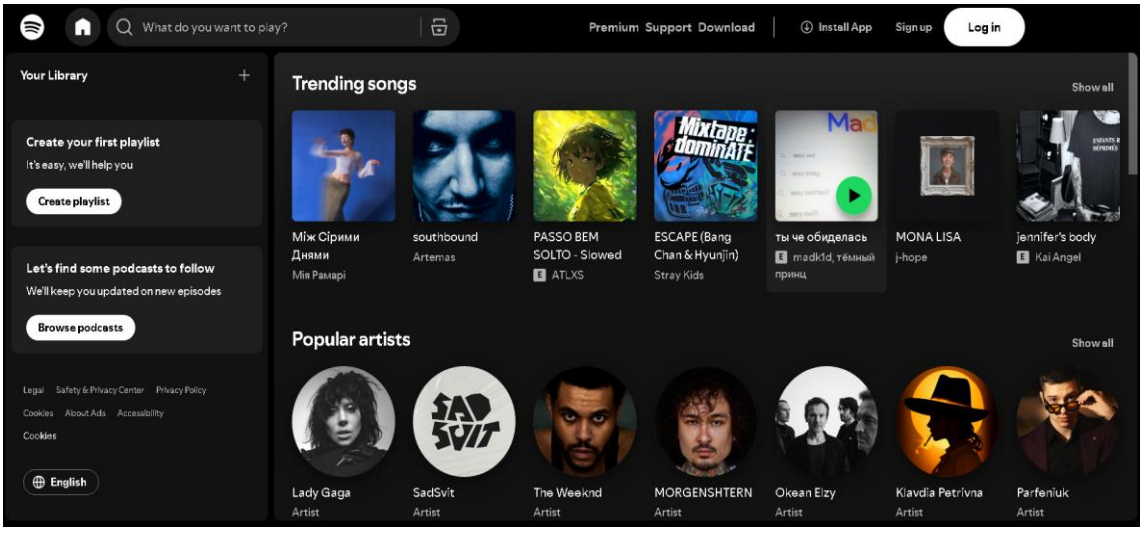

Рисунок 1.1 – Spotify

Джерело [3]

Apple Music (рис. 1.2) - сервіс від Apple, який пропонує користувачам величезний каталог пісень, відео та підкастів. Основною перевагою є висока інтеграція з іншими продуктами Apple, такими як iPhone, iPad, Mac та Apple Watch.

Переваги:

– висока якість звуку - підтримка безвтратного аудіо для слухачів з високими вимогами до якості;
– інтеграція з екосистемою Apple - сервіс безперешкодно працює з іншими пристроями Apple;
– індивідуальні рекомендації - система аналізує ваші вподобання і пропонує нові треки.

Недоліки:

– висока вартість підписки - для доступу до більшості функцій потрібно підписатися на платний план;
– малий безкоштовний доступ - в порівнянні з іншими сервісами;
– Apple Music має обмежений доступ для безкоштовних користувачів.

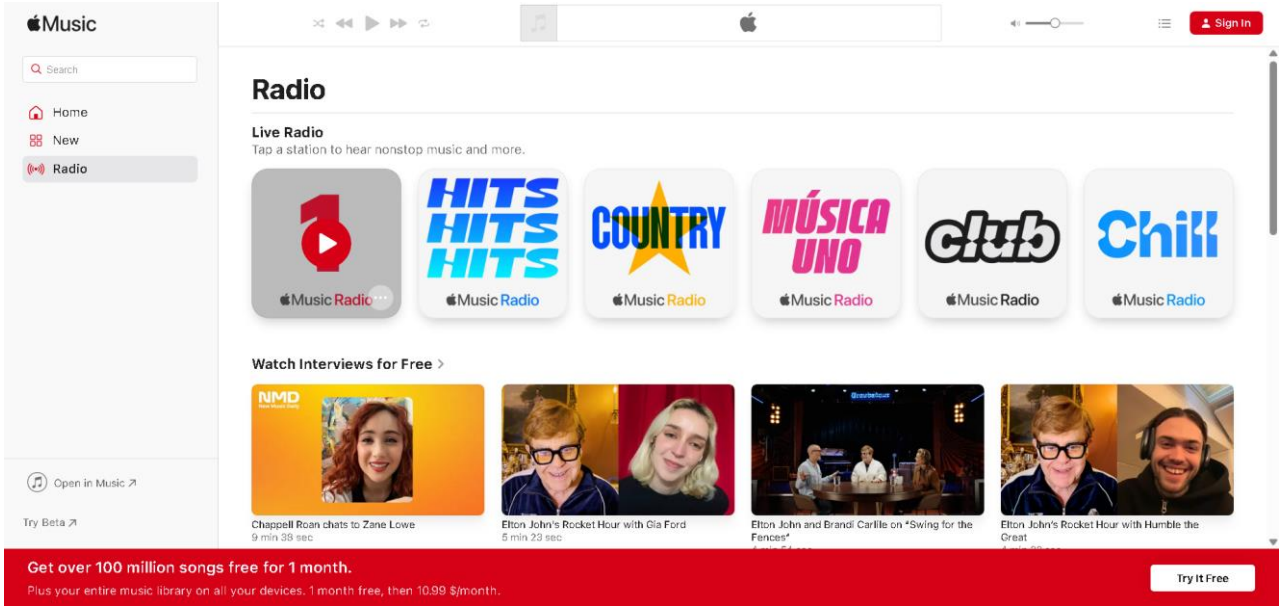

Рисунок 1.2 – Apple Music

Джерело [4]

YouTube Music (рис. 1.3) - це ще один потужний конкурент на ринку музичних сервісів. Сервіс пропонує доступ до великої кількості пісень і відео, а також інтеграцію з YouTube, що дозволяє знаходити кліпи на популярні треки та живі виступи.

Переваги:

– великий вибір відеокліпів - доступ до музичних відео та виступів в живу;
– інтеграція з YouTube - можливість переходити до відео-версій пісень безпосередньо з додатку;
– персоналізовані рекомендації - система пропонує контент відповідно до вподобань.

Недоліки:

– реклама в безкоштовній версії - безкоштовна версія містить рекламу, що може бути незручно для користувачів;
– обмежений доступ без підписки - для користування всіма функціями необхідно підписатися на преміум план.

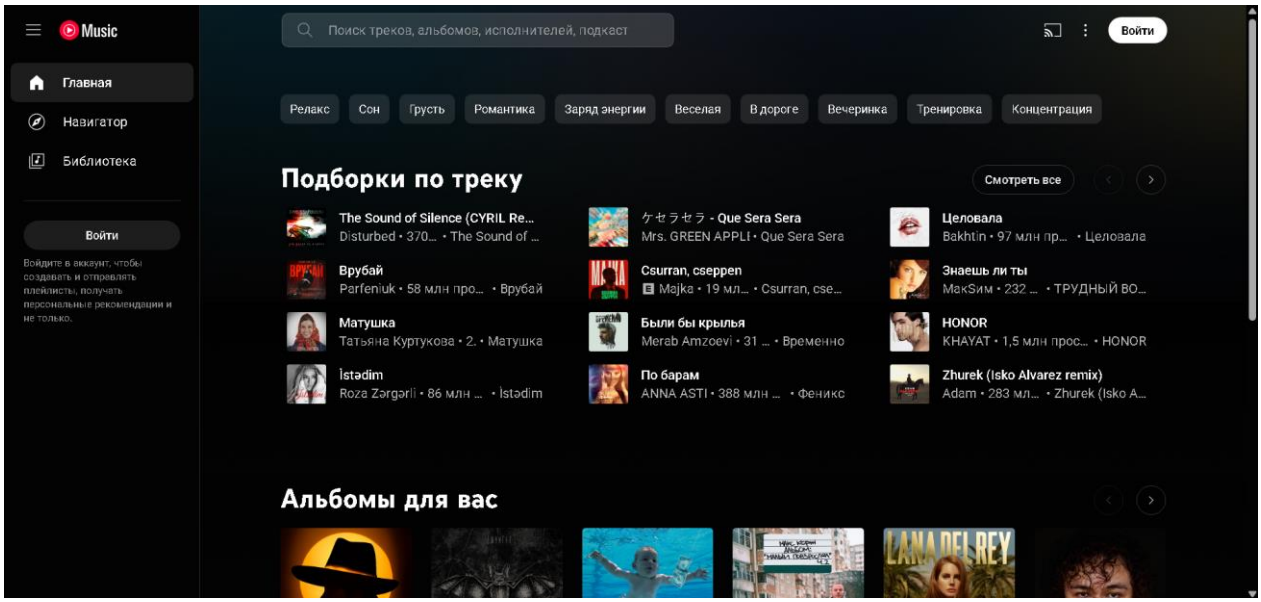

Рисунок 1.3 – YouTube Music

Джерело [5]

1.3 Постановка задачі

Аналіз і визначення функціональних вимог:

- визначити, які основні функції має виконувати бот (наприклад, пошук пісень за назвою, виконавцем, настроєм, історією прослуховувань);
- визначити методи інтеграції з музичними API, які дозволять отримувати інформацію про треки;
- впровадити систему рекомендацій, яка буде аналізувати вподобання користувача.

Розробка архітектури бота:

- спроєктувати логіку роботи бота, включаючи процеси обробки запитів, взаємодії з API та формування відповідей;
- вибрати серверні технології та базу даних для збереження історії пошуку та вподобань користувачів;
- забезпечити швидкодію бота та його стабільну роботу в умовах навантаження.

Розробка рекомендаційної системи:

- дослідити та реалізувати алгоритми машинного навчання або інші методи аналізу музичних уподобань;
- налаштувати персоналізовані рекомендації на основі історії запитів користувача;
- оптимізувати систему рекомендацій для підвищення точності підбору пісень.

Створення інтерфейсу взаємодії з користувачем:

- розробити зручний та інтуїтивно зрозумілий інтерфейс Telegram-бота;
- реалізувати систему команд та інтерактивних кнопок для швидкого вибору дій;
- впровадити підтримку мультимедійного контенту (наприклад, відображення обкладинок альбомів, посилань на прослуховування).

Тестування та оптимізація:

- перевірити коректність роботи бота на різних пристроях та версіях Telegram;
- виправити можливі баги та покращити продуктивність;
- оптимізувати запити до API, щоб зменшити затримку у видачі результатів.

Реліз та підтримка:

- запуск бота у відкритий доступ та збір зворотного зв’язку від користувачів;
- додавання нових функцій на основі відгуків;
- регулярне оновлення рекомендаційної системи та підтримка стабільної роботи.

Висновки до розділу 1

У цьому розділі було проведено аналіз предметної області та визначено основні аспекти, пов’язані з пошуком та підбором музики за допомогою рекомендаційних систем. Було розглянуто існуючі сервіси та їхні особливості, а також виокремлено конкурентні переваги майбутнього Telegram-бота.

Проаналізувавши потенційні можливості бота, визначено, що основними перевагами є безкоштовний доступ до функціоналу, відсутність обмежень на використання, простота та швидкість взаємодії. Також сформульовано ключові завдання розробки, включаючи інтеграцію з музичними API, створення рекомендаційної системи та забезпечення зручного інтерфейсу користувача.

Розробка такого бота є актуальною задачею, оскільки вона дозволяє спростити процес пошуку музики та зробити його більш персоналізованим без необхідності використання складних алгоритмів самостійно. Подальші розділи роботи будуть присвячені технічній реалізації проєкту та детальному опису процесу розробки.

РОЗДІЛ 2
ПРОЄКТУВАННЯ

2.1 Проєктування структури

Нефункціональні вимоги:

- надійність: Telegram-бот повинен функціонувати без збоїв, стабільно відповідати на запити користувача та коректно обробляти помилки;
- зручність використання - інтерфейс взаємодії повинен бути інтуїтивно зрозумілим, команди повинні бути простими, із підказками при введенні;
- швидкодія: бот повинен швидко реагувати на запити, забезпечуючи комфортну взаємодію;
- масштабованість: структура має дозволяти додавання нових функцій, таких як збереження історії.

Функціональні вимоги:

- пошук пісень: реалізувати механізм пошуку пісень за назвою, виконавцем або фрагментом тексту;
- рекомендації: впровадити рекомендаційну систему, яка на основі історії взаємодій або переваг користувача пропонує нові треки;
- інтерфейс користувача: реалізувати зручний telegram-інтерфейс через кнопки та текстові команди;
- отримання метаданих: бот повинен вміти отримувати дані про трек (назва, виконавець, жанр, тощо);
- можливість збереження пісень на рис. 2.1 представлено діаграму прецедентів (use case diagram).

Умовні позначення:

- користувач - <<actor>> - користувач Telegram-бота;
- кол - функції бота: пошук пісні, отримання рекомендацій, збереження до списку, перегляд історії;
- include включення обов’язкової функціональності у складну дію;

- extend - розширення базового функціоналу за умовою;

--> ініціація запиту від користувача;

- <-- відповідь системи.

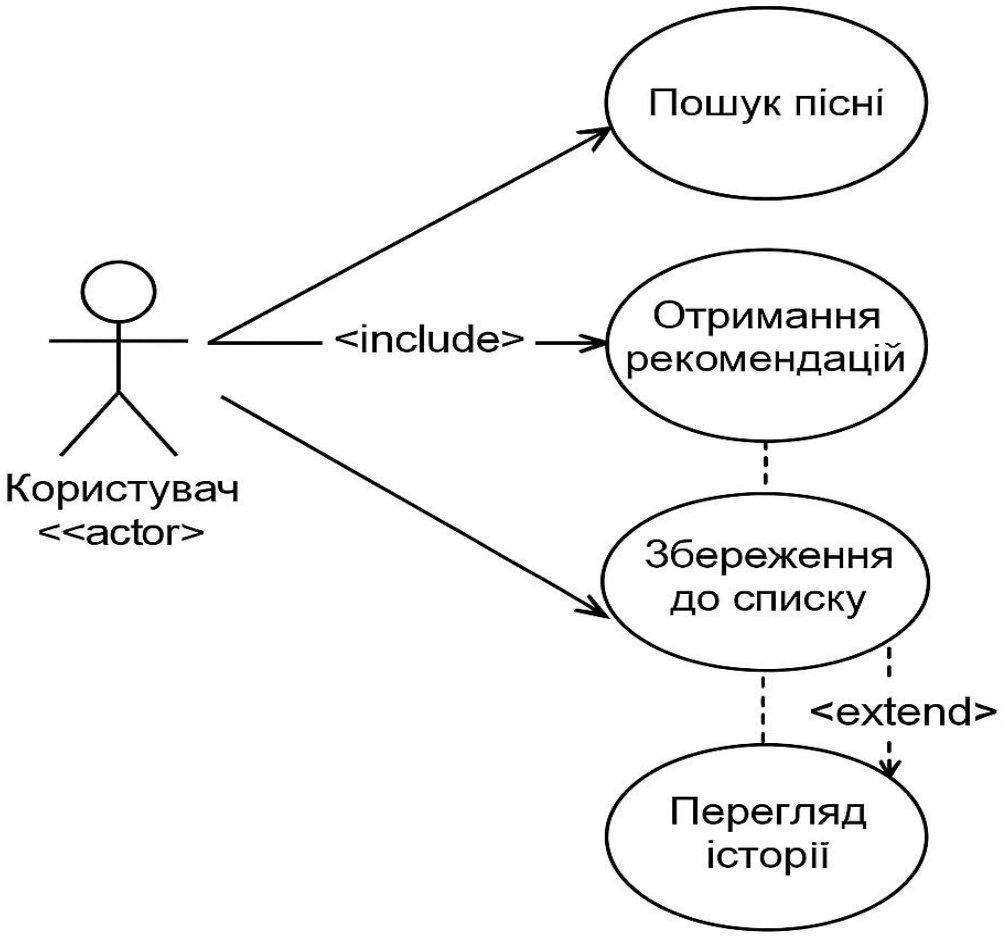

Рисунок2.1 - Діаграма прецедентів

Джерело: розроблено автором

Користувач запускає бот, далі обирає одну з функцій: пошук пісні, перегляд рекомендацій, перегляд збережених пісень тощо. При пошуку, бот надсилає запит до музичної бази даних або API, а у випадку з рекомендаціями - викликає алгоритм рекомендаційної системи. Відповіді формуються у вигляді повідомлень з результатами або списками.

На рис. 2.2 представлено діаграму послідовності (Sequence Diagram).

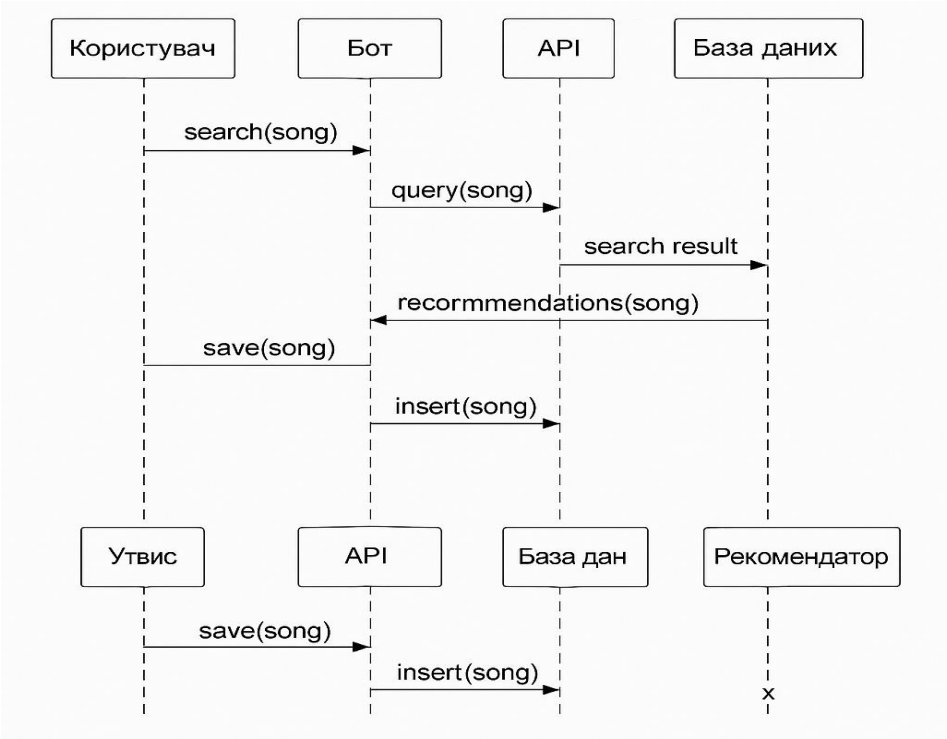

Рисунок 2.2 -Діаграма послідовності

Джерело: розроблено автором.

Умовні позначення:

- прямокутник об'єкт, який надсилає або приймає повідомлення (користувач, бот, API, база даних, рекомендатор)
- лінія життя - тривалість активності об’єкта;
- довгий прямокутник - тривалість активності об’єкта;
- < пересилання інформації;
- x - завершення процесу

Пояснення діаграми Користувач надсилає команду до бота, наприклад, "Знайди пісню". Бот надсилає запит до відповідного API, отримує список пісень, форматує відповідь і надсилає її користувачу. При запиті рекомендацій - викликається підсистема, яка аналізує дані користувача (історію запитів, збережені треки) і формує персоналізовану відповідь.

Для моделювання взаємозв’язків у системі Telegram-бота було розроблено ER-діаграму, яка відображає логічну структуру бази даних та

основні сутності, необхідні для збереження інформації про користувачів і рекомендовані музичні треки.

У системі передбачено три основні сутності: songs, users та user_history. Сутність songs зберігає відомості про аудіотреки, зокрема їх назви, унікальні URL-адреси джерел та шляхи до локальних копій файлів. Сутність users описує користувачів Telegram-бота за допомогою їх унікального ідентифікатора чату та налаштувань мови інтерфейсу, що дає змогу реалізувати персоналізовану взаємодію. Сутність user_history відповідає за збереження історії запитів користувачів: вона пов’язана з користувачем через ідентифікатор чату та фіксує запитані аудіотреки разом із міткою часу звернення.

Зв’язки між сутностями забезпечують узгодженість даних: один користувач може мати багато записів в історії запитів, що відображає його попередні взаємодії з ботом. Унікальність кожного треку за URL дозволяє уникати дублювання та спрощує механізми повторного доступу до вже опрацьованих ресурсів. У сукупності така модель підтримує ключові функції Telegram-бота - від збереження користувацьких уподобань до оптимізації обробки музичного контенту. Таким чином, ER-діаграма (рис 2.3 відображає зв’язок типу "один до багатьох" між користувачами та музичними треками, де кожен користувач може мати декілька пов’язаних композицій. Така модель забезпечує зручне зберігання історії взаємодії з ботом, у тому числі для подальшого аналізу вподобань або покращення рекомендаційного алгоритму.

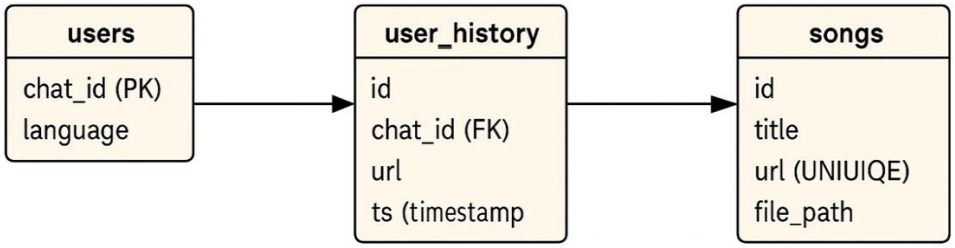

Рисунок 2.3 – ER-Діаграма

Джерело: розроблено автором

2.2 Проєктування дизайну та інтерфейсу

2.2.1 Принципи побудови текстового інтерфейсу

Telegram-бот як інтерфейсний об’єкт функціонує в рамках обмеженого текстового середовища, тому під час проєктування враховувалися такі ключові фактори:

1. Простота інтерфейсу

Telegram не дозволяє створювати візуально насичені інтерфейси з вікнами, панелями або шрифтами. Вся взаємодія здійснюється у вигляді:

- коротких повідомлень від бота;
- простих користувацьких запитів;
- кнопок без складних елементів керування.

Це змушує концентруватися на якісному формулюванні запитів і відповідей, що дозволяє користувачеві інтуїтивно розуміти, які можливості надає бот.

2. Послідовність стилю

Щоб зробити взаємодію впізнаваною, у всіх відповідях використовується уніфікований шаблон:

- однакова побудова повідомлень;
- використання емодзі для виділення типів дій або відповідей;
- короткі заголовки і чіткий розділ між частинами відповіді (наприклад, пісня – виконавець – кнопки).

Це забезпечує швидке візуальне сприйняття інформації навіть на мобільному пристрої.

3. Орієнтація на зручність (UX-фокус)

Усі елементи проєктувалися з урахуванням:

- мінімізації зусиль користувача - наприклад, щоб зберегти пісню, достатньо натиснути кнопку, не вводити команду;
- зменшення кількості повідомлень - бот надсилає стислі повідомлення, але з максимальним змістом;

– автоматичних підказок - бот після кожної відповіді пропонує, що можна зробити далі.

4. Підтримка кирилиці та мультимовність

Оскільки цільова аудиторія - україномовні користувачі, усі повідомлення мають коректну граматику, адаптовану лексику, а також можливість додати англійську версію в майбутньому.

2.2.2 Принципи взаємодії через кнопки та команди

Telegram API підтримує обмежену, але достатню для зручної взаємодії, систему елементів керування.

1. Команди. Команди - це основа текстової навігації в боті:

– вони зручні для досвідчених користувачів, які не потребують підказок;

– кожна команда відповідає за чітку дію (наприклад, /recommend – повертає рекомендовані пісні).

До кожної команди додається автоматичне пояснення у стартовому повідомленні, щоб користувач знав, як ними користуватись.

2. Reply-кнопки. Це панель із постійно доступними діями, наприклад:

css

[Пошук пісні] [Мій список] [Рекомендації]

Reply - кнопки показуються автоматично при старті або у важливих етапах взаємодії. Вони дозволяють діяти без введення тексту - одне натискання = запит.

3. Inline-кнопки. Найзручніший інструмент для взаємодії з результатами.

Наприклад, після пошуку пісні, бот надсилає варіант з кнопками:

css

[Слухати] [Зберегти] [Наступна пісня]

Inline - кнопки дозволяють зберігати пісні, прослуховувати, гортати списки без повторного введення.

4. Обробка повідомлень вручну. Користувач може самостійно вводити текстовий запит (назву треку або виконавця) - бот має бути достатньо "розумним", щоби:

- розпізнати помилки або варіанти введення (наприклад, "LadyGaga" = "Lady Gaga");
- надати корисну відповідь навіть у разі неоднозначного вводу.

2.2.3 Створення прототипів сценаріїв взаємодії

Прототипування Telegram-бота - це створення структурованих діалогових гілок (user flows), які імітують можливі шляхи користувача.

1. Основні сценарії наведені у табл 2.1.

Таблиця 2.1 – Основні сценарії телеграм-боту

№	Назва	Суть
1	Пошук пісні	Користувач вводить назву → бот надсилає результати з кнопками
2	Рекомендації	Бот пропонує треки на основі жанрів або вподобань
3	Перегляд списку	Користувач бачить збережені треки і може їх прослухати або видалити

2. Діалогові карти

Для кожного сценарію були створені мапи переходів, які містили:

- точку входу (запуск команди, натискання кнопки);
- кроки взаємодії (введення тексту, натискання кнопок);
- реакції бота (вивід інформації, збереження);
- альтернативи (у разі помилки або неуспіху запиту).

Це дозволило протестувати логіку та запобігти «глухим кутам» у спілкуванні з ботом.

3. Помилки та винятки

Бот також має реагувати на непередбачені ситуації. Відсутність результатів: за вашим запитом нічого не знайдено. Спробуйте ввести іншу назву.

Проблеми з API. Сервер тимчасово недоступний. Повторіть пізніше.

4. Адаптація до мобільного формату

Оскільки Telegram-боти переважно використовуються зі смартфонів:

- усі повідомлення мають бути короткими (до 3–5 рядків);
- кнопки розміщуються у 1–2 рядки;
- розмір вмісту не перевищує екрану, щоб не доводилось гортати.

Висновки до розділу 2

Спроєктований інтерфейс Telegram-бота забезпечує інтуїтивну, зрозумілу та приємну взаємодію, що базується на принципах мінімалізму та простоти, характерних для платформ типу месенджерів. Він відповідає функціональним можливостям Telegram, використовуючи inline-кнопки, callback-запити та повідомлення, які адаптуються до дій користувача. Завдяки цьому користувач може здійснювати пошук пісень, отримувати рекомендації та зберігати улюблені треки всього в кілька кліків, без потреби вводити складні команди або завантажувати окремі застосунки.

Інтерфейс побудований з урахуванням потреб сучасного користувача, який очікує швидкого та зручного доступу до функціоналу. Бот реагує миттєво, показує результати у зручному форматі з кнопками для прослуховування та збереження, а також зберігає історію взаємодії, що дозволяє легко повернутися до обраних треків. Такий підхід не тільки покращує юзер-досвід, але й підвищує залучення та задоволеність користувачів від взаємодії з ботом.

Таким чином, інтерфейс Telegram-бота не просто реалізує технічну функцію, а виступає ключовим елементом загальної системи взаємодії, роблячи пошук і рекомендацію музики максимально доступними, адаптивними та зручними в межах платформи Telegram.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ

3.1 Особливості реалізації

Розробка Telegram-бота має свої особливості (рис 3.1), пов’язані з обмеженнями середовища Telegram, форматом спілкування та технічними вимогами до обробки повідомлень у реальному часі. У цьому підрозділі розглянемо інструменти, які були використані для реалізації бота, а також методи, що забезпечують його функціонування.

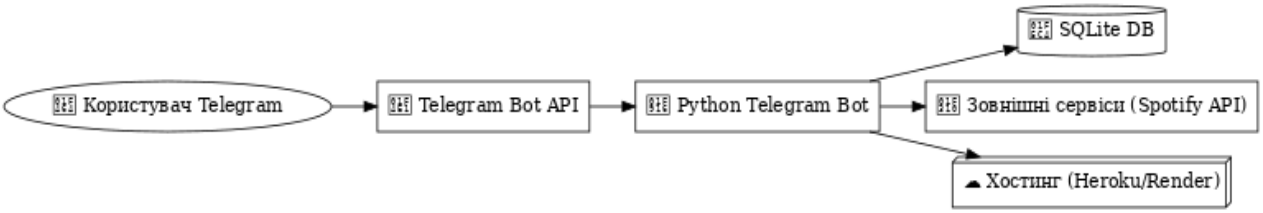

Рисунок 3.1 -Діаграма особливостей реалізації

Джерело: розроблено автором.

На діаграмі представлена логічна архітектура Telegram-бота для пошуку та рекомендації музики. Вона ілюструє основні компоненти системи та взаємодію між ними:

- користувач Telegram взаємодіє з ботом через Telegram Bot API;
- основна логіка реалізована на сервері Python, який обробляє повідомлення користувача;
- для зберігання інформації про пісні використовується SQLite база даних;
- для отримання музичних треків або рекомендацій система може інтегруватися з зовнішніми сервісами (наприклад, Spotify API);
- бот може бути розгорнутий на хостингу типу Heroku або Render, що забезпечує доступність через Інтернет.

3.1.1 Опис інструментів

1. Мова програмування. Python 3.11 - гнучка, проста у використанні мова, що підтримується спільнотою Telegram. Python має бібліотеки для роботи з Telegram API, HTTP-запитами, базами даних і машинним навчанням.

2. Telegram API. Для взаємодії з Telegram використано python-telegram-bot - це популярна бібліотека з підтримкою обробки повідомлень, inline-кнопок, callback-запитів, webhook-режиму.

3. Flask. Міні-фреймворк для створення окремих API, які дозволяють відокремити рекомендаційну систему у вигляді мікросервісу. Це дає можливість масштабувати систему без зміни ядра бота.

4. API для пісень. Spotify API та Deezer API надають доступ до даних про пісні: назви, жанри, альбоми, популярність, уривки для прослуховування. Через авторизовані запити бот отримує відповідну інформацію.

5. База даних. Для зберігання історії пошуків, збережених треків та вподобань обрано PostgreSQL, як масштабоване рішення з підтримкою реляційних зв’язків та JSON-полів. У прототипі може бути використана SQLite для спрощення.

6. Хостинг. Деплой бота виконаний на платформі Render.com (або аналогах: Heroku, Railway), яка дозволяє налаштувати Webhook, отримувати HTTPS-запити від Telegram та підтримувати стабільну роботу 24/7.

7. Інструменти розробки. Visual Studio Code, Postman (для перевірки API), GitHub (для контролю версій і командної роботи).
У процесі реалізації Telegram-бота для пошуку та рекомендації музичних композицій було інтегровано бібліотеку Scikit-learn, що забезпечила побудову базової моделі рекомендаційної системи на основі контентного аналізу.

Зокрема, Scikit-learn використовується для векторизації текстових запитів користувачів і назв пісень. Текстовий контент перетворюється у числові вектори з використанням таких методів, як:

–TF-IDF (Term Frequency – Inverse Document Frequency) - дозволяє визначити інформативність термінів у множині запитів;

–CountVectorizer - класичний підхід до представлення тексту як «мішка слів».

Таке представлення дозволяє подати музичні треки та запити користувача як багатовимірні вектори у просторі ознак.

На основі отриманих векторних представлень реалізовано алгоритм пошуку схожих пісень за допомогою моделі Nearest Neighbors, що також надається Scikit-learn. Механізм функціонує за принципом обчислення метрик схожості (наприклад, косинусної подібності або евклідової відстані) між векторами запиту та доступними треками.

Таким чином, Scikit-learn виступає інструментом реалізації контентно-орієнтованої системи рекомендацій, яка дозволяє формувати більш релевантні результати пошуку та підвищує якість персоналізованої взаємодії з користувачем.

3.1.2 Опис методів

Основні методи реалізації Telegram-бота включають:

- командні обробники: кожна команда користувача (/start, /search, /recommend, /list) викликає певну функцію;
- inline-кнопки та callback-запити: після кожної дії бот надсилає кнопки, які запускають відповідну логіку (наприклад, рекомендації або збереження);
- асинхронна обробка повідомлень: використання async def для забезпечення високої продуктивності та одночасної обробки кількох запитів;
- фільтрація та машинне навчання: для створення рекомендацій може бути використано простий алгоритм на основі жанрів або рейтингу, або модель на основі схожості запитів (наприклад, cosine similarity);
- запити до API: бот надсилає REST-запити до сторонніх API та обробляє JSON-відповіді, витягуючи потрібну інформацію;

– сесійна логіка: кожен користувач ідентифікується за Telegram ID, що дозволяє вести історію запитів та рекомендацій індивідуально.

3.2 Лістинг

У цьому пункті наведено фрагменти коду, які реалізують основну функціональність бота.

1. Стартове повідомлення

python

```
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    keyboard = [
        [InlineKeyboardButton("Пошук пісні", callback_data="search")],
        [InlineKeyboardButton("Рекомендації", callback_data="recommend")],
        [InlineKeyboardButton("Мій список", callback_data="list")]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await update.message.reply_text("Вітаю у музичному боті! Оберіть дію:",
reply_markup=reply_markup)
```

2. Запит пісень

python

```
def search_songs(query):
    response =
requests.get(f"https://api.spotify.com/v1/search?q={query}&type=track&limit=5",
headers=headers)
    results = response.json()["tracks"]["items"]
    return [{
        "title": track["name"],
        "artist": track["artists"][0]["name"],
        "url": track["external_urls"]["spotify"]
    } for track in results]
```

3. Відповідь з кнопками

```
python
async def show_song(update: Update, context: ContextTypes.DEFAULT_TYPE, song):
    buttons = [
        [InlineKeyboardButton("Слухати", url=song["url"])],
        [InlineKeyboardButton("Зберегти", callback_data=f"save|{song['title']}|{song['artist']}")]
    ]
    await update.message.reply_text(f"{song['title']} - {song['artist']}", reply_markup=InlineKeyboardMarkup(buttons))
```

3.3 Створення моделей

Telegram-бот не має візуального інтерфейсу як у мобільних додатків, однак структура повідомлень і кнопок - це своєрідний UI у текстовому форматі.

База даних. Відповідно до ER-діаграми було розроблено sql-запити для створення таблиць.

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    telegram_id VARCHAR(50) NOT NULL,
    preferences TEXT
);

CREATE TABLE tracks (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    title TEXT,
    artist TEXT,
    added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Формат відповіді у Telegram

less

Назва: Imagine

Виконавець: John Lennon

[Слухати] [Зберегти] [Наступна]

3.4 Конструювання

Для наочної демонстрації плану дій основних етапів реалізації Telegram-бота було використано метод структурної декомпозиції проєкту (Work Breakdown Structure, WBS) (рис. 3.2), що дозволяє представити загальний процес розробки у вигляді логічно впорядкованих підетапів.

Цей підхід відображено на діаграмі особливостей реалізації, яка структурно охоплює ключові компоненти проєкту, зокрема:

- налаштування середовища: встановлення Python, необхідних бібліотек, створення та конфігурація віртуального середовища (venv);
- підключення API: робота з Telegram Bot API та інтеграція із зовнішніми музичними сервісами;
- реалізація логіки: розробка модулів обробки запитів, пошуку, збереження обраних композицій та формування персоналізованих рекомендацій;
- інтерфейс користувача: реалізація інтерактивних елементів взаємодії через команди та кнопки Telegram;
- тестування: перевірка функціональної логіки, обробки винятків та збереження даних;
- розгортання (деплой): публікація Telegram-бота на хмарному хостингу (Heroku або Render), налаштування змінних середовища та конфігураційних параметрів.

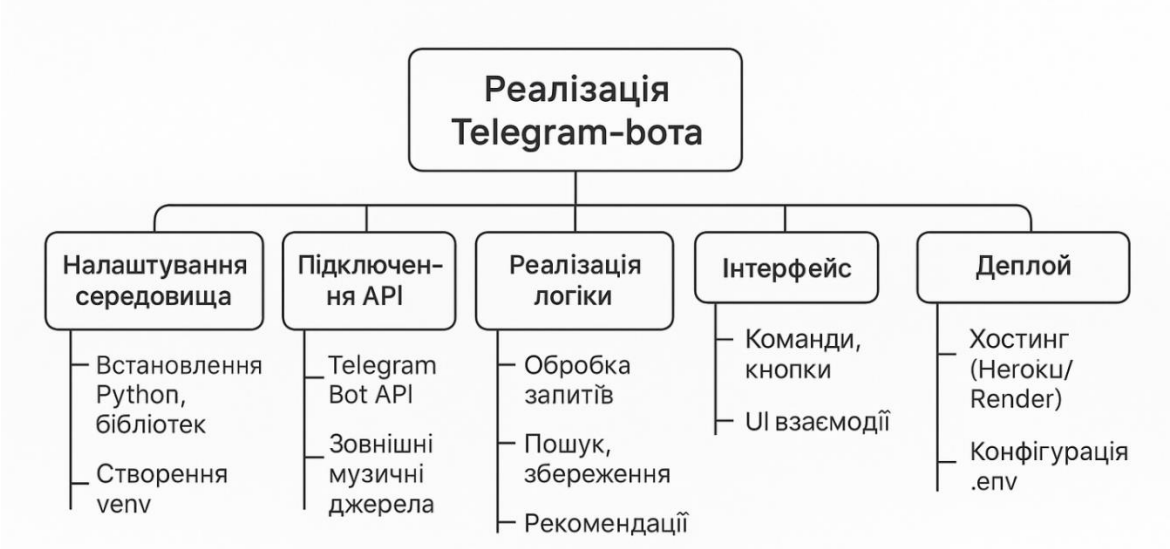

Рисунок 3.2 -Діаграма особливостей реалізації

Джерело: розроблено автором.

Структурована діаграма етапів реалізації Telegram-бота. Вона показує Work Breakdown Structure (WBS) - розбиття проєкту на ключові етапи:

- налаштування середовища - встановлення Python, бібліотек, створення venv;
- підключення API - робота з Telegram Bot API, підключення зовнішніх джерел музики;
- реалізація логіки - написання коду обробки запитів, пошуку, збереження, рекомендацій;
- інтерфейс - реалізація команд, кнопок та UI взаємодії в Telegram;
- тестування - перевірка логіки, збереження даних, обробки помилок;
- деплой - розгортання на хостингу (Heroku/Render), конфігурація .env.

Призначення. Ця діаграма демонструє поетапне планування та реалізацію компонентів бота, що дозволяє ефективно координувати розробку.

1. Архітектура додатку

- handler Layer: відповідає за обробку команд і повідомлень;
- service Layer: реалізує логіку (наприклад, пошук, збереження, рекомендації);
- data Layer: робота з БД, запис результатів, отримання історії.

2. Модульність

Кожна функція винесена в окремий файл:

- handlers/start.py
- handlers/search.py
- services/api.py
- db/models.py

3. Сценарії використання

Користувач вводить запит → бот надсилає варіанти → користувач зберігає обрану пісню → бот додає її до БД та надсилає підтвердження.

3.5 Тестування

Для відображення життєвого циклу обробки запиту користувача у Telegram-боті розроблено діаграму на рис 3.3.

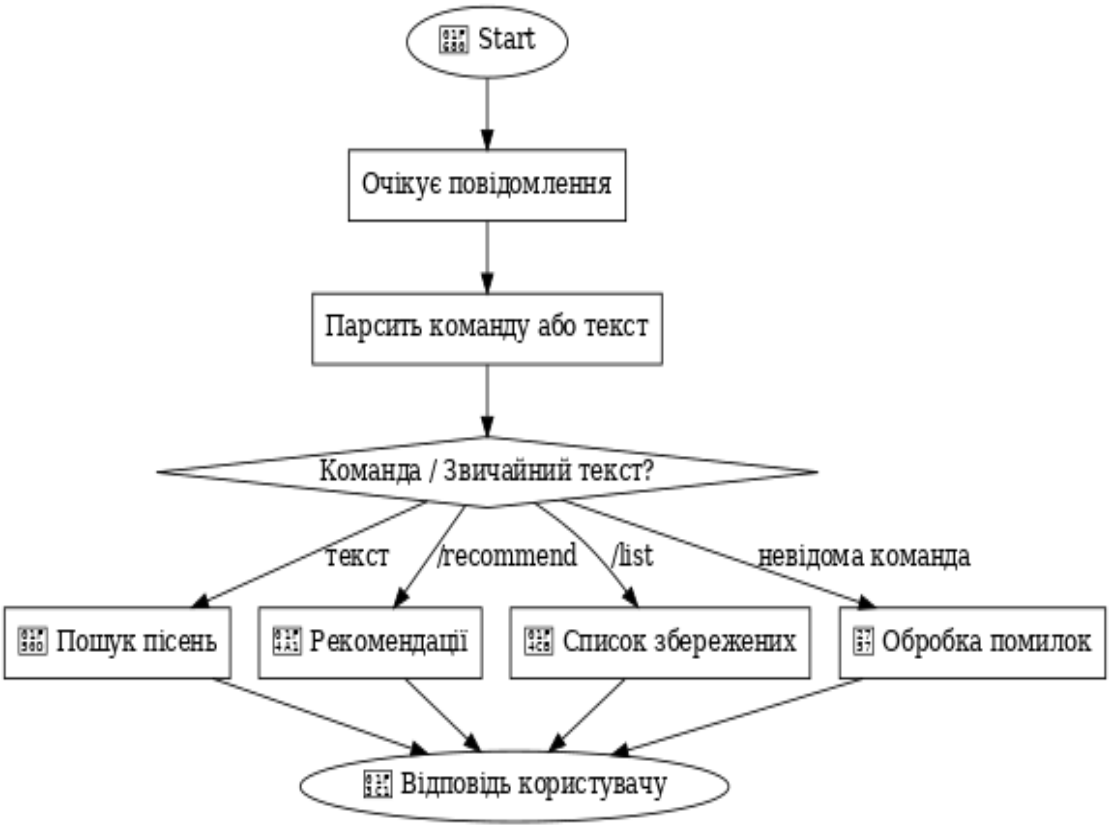

Рисунок 3.3 -Діаграма тестування

Джерело: розроблено автором.

Процес починається зі старту бота. Далі бот очікує повідомлення від користувача. В залежності від типу повідомлення (команда чи звичайний текст), бот переходить до відповідного блоку:

- пошук пісень - при текстовому запиті;
- рекомендації - при команді /recommend;
- список пісень - при команді /list.

Якщо команда не розпізнана - відбувається обробка помилки. Після виконання запиту бот формує відповідь користувачу. Ця діаграма використовується для візуалізації логіки обробки запитів під час тестування, дозволяє виявити точки відмови або помилки у логіці.

1. Юніт-тести

Юніт-тестування охопило ключові компоненти функціональності системи, зокрема:

– проведено тестування логіки пошуку музичних композицій за вхідними текстовими запитами користувача. Перевірено коректність обробки запитів, включаючи ситуації з некоректним або порожнім вводом, а також відповідність результатів очікуваним даним, що повертаються з API;

– оцінено стабільність та правильність взаємодії з зовнішніми музичними сервісами. Особливу увагу приділено обробці виняткових ситуацій, таких як відсутність з'єднання, некоректні ключі доступу або затримки у відповіді сервера. Перевірено механізми повторного надсилання запитів та обробки повідомлень про помилки;

– тестовано коректність запису інформації про користувачів та треки до реляційної бази даних. Окрему увагу приділено перевірці цілісності даних, наявності необхідних зв'язків між таблицями (наприклад, між користувачами та доданими ними треками), а також обробці можливих винятків при доступі до бази.

Таким чином, юніт-тестування підтвердило функціональну спроможність основних модулів системи та їхню готовність до подальшої експлуатації в реальних умовах.

2. Інтеграційне тестування

Відправляється справжній запит у Telegram → перевіряється реакція бота на кнопки, команди, збереження треків.

3. Обробка винятків

У процесі розробки програмного забезпечення особливе значення має обробка виняткових ситуацій, що можуть виникати під час виконання коду. Однією з типових ситуацій є помилки, пов’язані із взаємодією із зовнішніми веб-сервісами, зокрема через нестабільність мережевого з’єднання, недоступність API або інші технічні збої. Ігнорування таких ситуацій може призводити до некоректної поведінки застосунку або його аварійного завершення.

У межах реалізації Telegram-бота, що взаємодіє з музичними API (наприклад, Spotify або Deezer), було реалізовано механізм обробки винятків для забезпечення стійкості системи. Зокрема, для обробки помилок при виконанні HTTP-запитів використано конструкцію try–except, що дозволяє перехопити винятки, пов’язані з мережею, які виникають під час виклику requests.get(...).

При виявленні винятку, пов’язаного з класом requests.exceptions.RequestException, бот не завершує виконання, а замість цього інформує користувача про тимчасову недоступність сервісу. Таким чином, реалізовано базову форму зворотного зв’язку, яка підвищує рівень користувацького досвіду за рахунок передбачуваної та безпечної реакції системи на непередбачувані ситуації фрагмент коду на мові python:

```
try:
    response = requests.get(...)
except requests.exceptions.RequestException:
    await update.message.reply_text("Проблема з пошуком. Спробуйте пізніше.")
```

Цей підхід дозволяє мінімізувати вплив зовнішніх чинників на стабільність роботи Telegram-бота, забезпечуючи кращу надійність та адаптивність системи у реальних умовах її експлуатації. У подальшому

обробку винятків доцільно розширити, передбачивши логування подібних випадків для аналізу та вдосконалення функціонування сервісу.

3.6 Використання

1. Розгортання:

- бот завантажено на Render.com;
- створено .env з токеном Telegram;
- налаштовано Webhook;
- налаштовано таймер автоматичного перезапуску при збоях.

2. Покрокова інструкція для користувача:

- запустити бота, натиснувши /start;
- обрати « Пошук пісні» або « Рекомендації»;
- отримати список пісень;
- прослухати або зберегти;
- перейти до « Мій список», щоб переглянути обране.

3. Зручність

Максимальна відповідність очікуванням користувача: швидка відповідь, мінімум дій, автоматичне збереження історії.

Висновки до розділу 3

У третьому розділі було детально розглянуто процес реалізації Telegram-бота для пошуку та рекомендації музичних треків. Основна увага приділена архітектурі рішення, вибору інструментів та методів програмної реалізації, що забезпечують ефективну взаємодію між користувачем і системою.

Було обґрунтовано вибір мови програмування Python та бібліотеки python-telegram-bot, які дозволили реалізувати асинхронну обробку запитів у реальному часі. В якості основи для рекомендаційної системи застосовано прості алгоритми фільтрації, що можуть бути доповнені більш складними моделями в майбутньому. Для зберігання даних обрано гнучке рішення - PostgreSQL (на етапі прототипування - SQLite).

Інтеграція з зовнішніми API (Spotify, Deezer) надала можливість отримувати актуальні музичні дані, а мікросервісна архітектура через Flask забезпечила масштабованість системи. Візуальна складова Telegram-бота реалізована у вигляді текстового UI з кнопками, що забезпечує зручну взаємодію для користувача.

Було також виконано модульну побудову проєкту, створено структуровану базу даних, реалізовано основні сценарії використання, протестовано функціональність та реалізовано розгортання на хмарному хостингу Render.com.

Таким чином, розроблений Telegram-бот демонструє ефективну інтеграцію декількох технологій для реалізації персоналізованої системи пошуку та рекомендації музики, що відповідає сучасним вимогам до взаємодії у месенджерах.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було реалізовано повний цикл дослідження, проектування та впровадження Telegram-бота, що здійснює автоматизований підбір музичних композицій на основі рекомендаційних алгоритмів. Робота охопила вивчення теоретичних засад побудови рекомендаційних систем, аналіз сучасних технологічних рішень та практичну реалізацію функціонального прототипу.

Проведений аналіз предметної області дозволив виявити суттєвий розрив між потенціалом сучасних рекомендаційних систем і зручністю їх використання для кінцевих користувачів, особливо у межах популярних месенджерів. Визначено, що Telegram-бот є ефективною платформою для реалізації інтелектуальної системи підбору музики, оскільки не потребує встановлення додаткового програмного забезпечення, забезпечує швидкий доступ до функцій та підтримує інтуїтивну взаємодію.

В межах практичної частини було розроблено Telegram-бота, що поєднує такі ключові компоненти:

- інтерфейс взаємодії з користувачем, побудований із використанням inline-кнопок та callback-запитів;
- модулі обробки даних з використанням Python та бібліотеки python-telegram-bot;
- інтеграція з API музичних сервісів (Spotify, Deezer) для отримання актуальної інформації про треки;
- реалізація базових рекомендаційних алгоритмів фільтрації;
- гнучка система зберігання даних із застосуванням PostgreSQL;
- модульне структурування проєкту та використання мікросервісної архітектури (Flask), що забезпечує масштабованість і можливість подальшого розвитку.

Важливим результатом дослідження є демонстрація того, що Telegram-боти можуть слугувати ефективною альтернативою традиційним музичним

застосункам, надаючи користувачам персоналізований досвід без перевантаження інтерфейсом та без потреби в складній інсталяції.

До основних практичних результатів роботи можна віднести:

- впровадження зручного та мінімалістичного інтерфейсу, адаптованого під швидку взаємодію в месенджері;
- забезпечення стабільної роботи сервісу у хмарному середовищі Render.com;
- створення основи для подальшого впровадження складніших моделей рекомендацій на основі машинного навчання (наприклад, колаборативної фільтрації, кластеризації або нейронних мереж);
- розробку архітектури, що може бути масштабована або легко адаптована для інтеграції з іншими сервісами.

Запропоноване рішення апробовано під час тестування на реальних сценаріях використання. Результати показали високий рівень стабільності роботи бота, що підтверджує доцільність обраної архітектури.

Перспективи подальших досліджень передбачають:

- удосконалення алгоритмів персоналізації через впровадження моделей машинного навчання;
- інтеграцію з додатковими музичними платформами;
- розширення функціоналу бота за рахунок аналізу емоційного стану користувача, часу доби, активності тощо;
- створення веб-інтерфейсу або мобільної версії як доповнення до Telegram-бота.

Загалом, результати проведеного дослідження підтверджують можливість створення зручного, масштабованого та адаптивного інструменту для персоналізованого підбору музики на основі сучасних рекомендаційних технологій у межах платформи Telegram.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Як підключити Python до баз даних SQL. SharpCoder. URL: https://uk.sharpcoderblog.com/blog/how-to-connect-python-to-sql-databases
2. Як створити телеграм-бота на Python. SpaceLab. URL: https://spacelab.ua/articles/yak-stvoriti-telegram-bota-na-python/
3. Spotify. URL: https://open.spotify.com/
4. Apple Music. URL: https://music.apple.com/us/radio
5. YouTube Music. URL: https://music.youtube.com/
6. Telegram Bot API. URL: https://core.telegram.org/bots/api
7. Python Telegram Bot Documentation. URL: https://docs.python-telegram-bot.org
8. Flask Documentation. URL: https://flask.palletsprojects.com
9. Spotify for Developers. URL: https://developer.spotify.com/documentation/web-api
10. Deezer API. URL: https://developers.deezer.com/api
11. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/
12. Render Hosting Platform. URL: https://render.com
13. SQLite Documentation. URL: https://www.sqlite.org/docs.html
14. GitHub (для керування версіями та спільної розробки). URL: https://github.com
15. Requests: HTTP for Humans. URL: https://requests.readthedocs.io/en/latest/
16. Scikit-learn Documentation (для моделей подібності). URL: https://scikit-learn.org/stable/documentation.html
17. Visual Studio Code. URL: https://code.visualstudio.com
18. Postman (інструмент для тестування API). URL: https://www.postman.com
19. Python 3.11 Documentation. URL: https://docs.python.org/3.11/