ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Telegram-бот для пошуку музики з використанням алгоритмів пошуку по відрізку аудіо»

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-21
Олександр КРАСНОВ
Керівник: старший викладач кафедри
комп’ютерних наук
Олег ЛУКУТІН

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

Краснов Олександр Миколайович

Тема роботи	Телеграм бот для пошуку музики з використанням алгоритмів пошуку по відрізку аудіо
Номер та дата наказу про затвердження теми	№121-7 від 24 грудня 2024 року
Коротка постановка завдання	Створити Telegram-бота, який зможе автоматично розпізнавати музичні треки за допомогою аналізу відрізків аудіо, отриманих із відео, аудіофайлів або голосових повідомлень, та надавати користувачам інформацію про знайдені треки з посиланням на їх прослуховування у Spotify. https://telegrambots.github.io/book/
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	A guide to Telegram.Bot library URL:. https://telegrambots.github.io/book/ 1. Кузьменко, І. М. Теорія графів: навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки» / І. М. Кузьменко ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,25 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2020. – 71 с. URL: https://ela.kpi.ua/handle/123456789/35854 (дата звернення: 30.3.2025).
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має містити теоретичне, системотехнічне або експериментальне дослідження за темою роботи, яку слід розглядати як складне спеціалізоване завдання або практичну проблему в галузі комп'ютерних наук, яка характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 27 грудня 2024 р.

Керівник Олег ЛУКУТІН

Здобувач освітнього ступеня бакалавра Олександр КРАСНОВ

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 Олег ЛУКУТІН

Здобувач освітнього ступеня бакалавра Олександр КРАСНОВ

Краснов О.М Телеграм бот для пошуку музики з використанням алгоритмів пошуку по відрізку аудіо.

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 розробки телеграм ботів на прикладі розробки боту "MusicFi", з використанням Spotify API та Shazam API.

Ключові слова: Telegram, бот, розробка Telegram боту, музика

Krasnov O.M. Telegram-bot for music search using audio segment search algorithms

Project explanatory note by specialty 122 – Computer science. – «KROK» University, Educational and Scientific Institute of information and communication technologies, Department of Computer Science, Kyiv, 2025.

The paper examines the challenges of developing Telegram bots using the example of creating the "MusicFi" bot, which utilizes Spotify API and Shazam API.

Keywords: Telegram, bot, Telegram bot development

ЗМІСТ

ВСТУП **6**
РОЗДІЛ 1 ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ **7**
1.1 Предметна область 7
1.2 Огляд Аналогів 11
1.3 Постановка задачі 15
Висновки до розділу 17
РОЗДІЛ 2 ПРОЄКТУВАННЯ **18**
2.1 Функціональні вимоги 18
2.2 Нефункціональні вимоги 19
2.3 Проєктування інтерфейсу 20
Висновки до розділу 23
РОЗДІЛ 3 РЕАЛІЗАЦІЯ **24**
3.1 Способи реалізації 24
3.1.1 Використання сторонніх API-сервісів для розпізнавання музики *24*
3.1.2 Реалізація на основі бібліотек локального або відкритого типу *25*
3.1.3 Комбінований підхід *27*
3.1.4 Інтеграція зі Spotify API *28*
3.1.5 Альтернативи *28*
3.1.6 Огляд способів *29*
3.2 Конструювання 31
3.2.1 Отримання та ініціалізація токенів *31*
3.2.2 Імпорт бібліотек *33*
3.2.3 Авторизація Telegram-бота та ініціалізація Spotify *35*
3.2.4 Асинхронна функція для розпізнавання музики *36*
3.2.5 Створення функції конвертації та пошуку у Spotify *36*
3.2.6 Реалізація функції завантаження по посиланню *38*
3.2.7 Взаємодія з користувачем *39*
3.2.8 Розробка основної функції обробки *40*
Висновки до розділу 41
ВИСНОВКИ **42**
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ **43**

ВСТУП

Актуальність теми. У сучасному світі музика є важливою складовою повсякденного життя людини. З розвитком цифрових технологій та платформ для потокового відтворення музики (Spotify, Apple Music, YouTube Music тощо) зросла потреба в інструментах для автоматичного визначення музичних треків за аудіофрагментами. Водночас, багато користувачів стикаються з труднощами під час пошуку назви композиції, почутої у відео, голосовому повідомленні або аудіофайлі.

Метою даної роботи є розробка Telegram-бота для пошуку музики, який дозволяє ідентифікувати аудіотреки за допомогою алгоритмів розпізнавання аудіофрагментів, автоматично обробляти отримані дані та надавати користувачам посилання на відповідні треки в Spotify.

Об'єктом дослідження є процес пошуку та ідентифікації музичних треків за аудіофрагментами, отриманими з різних джерел (відео, аудіо, голосові повідомлення).

Предмет дослідження є Telegram-бот для пошуку музики з використанням алгоритмів розпізнавання аудіофрагментів та інтеграції зі Spotify.

Методами дослідження є аналіз існуючих рішень і технологій, алгоритми обробки аудіофайлів, використання API (Shazam та Spotify), а також сучасні бібліотеки Python для роботи з Telegram та обробки медіафайлів.

Практичне значення. Результати дослідження мають практичне значення, оскільки розроблений Telegram бот може бути використаний для ідентифікації музики у повсякденному спілкуванні, забезпечуючи швидкий доступ до інформації про трек та його прослуховування в Spotify.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку посилань (11 найменувань). Пояснювальна записка містить 2 таблиці, 19 рисунків та 1 графік. Загальний обсяг пояснювальної записки складає 44 сторінки, основний зміст викладено на 43 сторінках.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ

1.1 Предметна область

Telegram "зародився" в 2013 році як дослідний проєкт Павла Дурова (рис 1.1). Він задумав перевірити технологію зашифрованого листування – MTProto, розроблену його братом Миколою Дуровим. Дурову було цікаво, як технологія витримає велике навантаження. Так з'явилася перша версія програми, яка до нині доступна майже на всіх платформах.

Рисунок 1.1 – Павло Дуров, автор Telegram

Джерело: [1]

У грудні 2013 року Дуров оголосив про нагороду в 200 тисяч доларів у біткоїнах тому, хто розшифрує його листування з братом. Фрагмент шифру виклали на сайті, проте зламати його ніхто не зміг. Пізніше премію збільшили до 300 тисяч доларів за розшифровку листування двох ботів, Пола та Ніка. Розшифрувати повідомлення, захищені месенджером, так і не вдалося, тому Telegram набув репутації найнадійнішого месенджера.

Визначною рисою Telegram у порівнянні з іншими месенджерами є боти. З їхньою допомогою можна відшукати майже будь-що, що вас цікавить, слідкувати

за новинами різних ЗМІ та інших інформаційних ресурсів, грати в ігри, розв'язувати буденні завдання та багато іншого.

По суті, це робить месенджер корисним навіть якщо у вас немає жодного контакту, адже окрім спілкування з друзями, Telegram пропонує широкий вибір функцій, що не завжди стосуються спілкування. Telegram-боти є ефективним засобом автоматизації завдань, що дозволяє інтегрувати різні алгоритми розпізнавання аудіо у зручний для користувача інтерфейс.

Боти активно застосовуються для автоматизації рутинних завдань у різних сферах. Наприклад, вони можуть виконувати функції віртуальних асистентів, допомагати з плануванням справ, надавати актуальну інформацію про погоду, курс валют або новини.

У сфері освіти Telegram-боти стали інструментом для тестування знань, проведення опитувань та надання навчальних матеріалів. Вони можуть працювати як персональні репетитори, надаючи відповіді на запитання, генеруючи інтерактивні завдання або навіть моделюючи діалог з користувачем для покращення мовних навичок.

Також великого поширення набули фінансові боти, що допомагають відстежувати витрати, налаштовувати бюджети та навіть здійснювати грошові перекази. Завдяки інтеграції з платіжними системами такі боти можуть слугувати зручним інструментом для онлайн-оплат та контролю фінансових операцій.

Ще однією цікавою нішею є Telegram-боти для електронної комерції. Вони дозволяють автоматизувати обробку замовлень, надавати клієнтам персоналізовані рекомендації та навіть забезпечувати підтримку через чат, використовуючи алгоритми обробки природної мови.

Завдяки відкритому API та розширеним можливостям кастомізації Telegram залишається одним із найгнучкіших месенджерів для створення розумних ботів. Його екосистема продовжує зростати, що видно на графіку (рис

1.2) пропонуючи користувачам ще більше функцій, які можуть значно спростити їхнє життя та покращити взаємодію з цифровим світом.

Рисунок 1.2 - Графік зростання кількості Telegram-ботів

Джерело: [2]

Одним із перспективних напрямів є використання Telegram-ботів для інтеграції цих технологій у повсякденне життя. Такі боти можуть розпізнавати пісні з голосових повідомлень, аудіофайлів або навіть уривків відео, що відкриває широкі можливості для користувачів, які хочуть швидко знайти назву треку.

Розпізнавання музики ґрунтується на методах цифрової обробки сигналів (DSP), що включають спектральний аналіз, перетворення Фур'є, вейвлет-аналіз та методи нейронних мереж. Штучний інтелект значно покращує ці алгоритми, дозволяючи ідентифікувати музику навіть у складних акустичних умовах.

Додатково впровадження API-сервісів, таких як Whisper для розпізнавання напівпродиктованого тексту, Shazam або Audd.io для пошуку оригінальних

композицій, а також баз даних на зразок Spotify чи Genius для ідентифікації пісень за текстом, дозволяє створити потужний інструмент для аудіопошуку

Завдяки інтеграції з хмарними обчисленнями та мобільними додатками можливе створення ще більш зручних та доступних рішень. У перспективі можна очікувати появу інтелектуальних помічників, які не лише ідентифікуватимуть музику, а й аналізуватимуть її структуру, рекомендуватимуть схожі треки або навіть створюватимуть нові мелодії на основі вподобань користувачів.

Таким чином, Telegram-боти не лише виконують роль інформаційних помічників, але й стають повноцінними мультимедійними інструментами, що можуть конкурувати з традиційними мобільними додатками. Завдяки зростаючим можливостям інтерфейсу Telegram, зокрема підтримці надсилання аудіо, відео та голосових повідомлень, відкриваються нові перспективи для інтеграції алгоритмів розпізнавання звуку без необхідності встановлення додаткового ПЗ на пристрої користувача.

Використання таких ботів особливо зручне у випадках, коли потрібно швидко ідентифікувати музику «на ходу» – без реклами, реєстрації чи повного доступу до потокових сервісів. До прикладу, отримавши голосове повідомлення або аудіофайл у чаті, бот може автоматично розпізнати композицію, знайти її в каталозі Spotify або YouTube та надати пряме посилання для прослуховування. Це значно підвищує інтерактивність і корисність месенджера в очах користувачів.

З технічної точки зору, такий функціонал базується на комбінації технологій: попередня обробка аудіосигналу (обрізання, конвертація, нормалізація), його передача до сервісу розпізнавання, а далі – пошук відповідного треку в музичних базах даних. Додатково можна реалізувати підтримку текстового пошуку, коли користувач вводить уривок слів з пісні, а бот здійснює пошук за текстами пісень через такі сервіси, як Genius чи Musixmatch.

У довгостроковій перспективі боти, здатні розпізнавати аудіо, можуть стати складовою частиною персоналізованих рекомендаційних систем. На основі історії пошуків та улюблених жанрів можна формувати добірки пісень, створювати автоматичні плейлисти або пропонувати концертні події поблизу. Це дозволить перетворити простого Telegram-бота на повноцінного музичного помічника, адаптованого до потреб кожного користувача.

1.2 Огляд Аналогів

Данний Telegram бот не є єдиним в своєму жанрі пошуку музики в телеграмі, тому необхідно переглянути і порівняти між собою його аналоги. Порівнюючи наш бот з іншими, можна виявити якісні відмінності, що дозволяють збагатити його новими ідеями або удосконаленнями.

Також, порівнюючи з іншими, можна виявити потенційні переваги та недоліки, що стануть основою для подальших вдосконалень та розвитку боту Таким чином, порівняння мого боту MusicFi з аналогами допоможе не лише збагатити розуміння ассортименту ботів, але й виявити шляхи для його поліпшення.

Тема пошуку музики досить розпоширена в телеграмі, та в ній в основному домінують боти спрямовані на пошук музики по назві або виконавцю, та подальшому їх відправленню у вигляді аудофайлу напряму, мій ж Telegram бот буде знаходити музику з відео, посилань, голосових повідомлень, назві пісні та слів з неї напряму.

Для прикладу візьмемо два популярних бота по пошуку аудіофрагментів та музики в Telegram які різняться по своєму функціоналу : VK Music Bot та Shazam music bot.

VK Music Bot (рис. 1.3) – на мою думку найпопулярніших з ботів по пошуку музики в Telegram, він використовує алгоритми пошуку по назві або виконавцю, з можливістю окремо відфільтрувати пошук по необхідному параметрі (або

пошук виключно по назві, або виключно по виконавцю), має можливість зміни мови а також налаштування інтерфейсу, з важливих це кількість знайдених пісень в одному стовпці та показ бітрейту.

Бот має доволі інтуїтивне управління та не навантаженний інтерфейс та потужний пошуковий потенціал. З мінусів можу відокремити це й ж самий пошуковий потенціал – при спробі знайти одну конкретну пісню, бот видає десятки варіацій, серед яких немало каверів, низькоякісних аудіодоріжок, або навіть відрізків або ‘порожніх’ пісень.

Як можемо бачити на рис. 1.4 – бот без проблем знайшов цілих 286 варіацій необхідної нам пісні, але видав безліч варіацій та реміксів. На мою думку подібна деталь являється одночасно як і величезним мінусом, так і величезним плюсом, з однієї сторони у користувача є можливість знайти безліч варіантів улюбленої пісні, з іншої подібне нагромадження різноманітностей може завадити знайти необхідний оригінал.

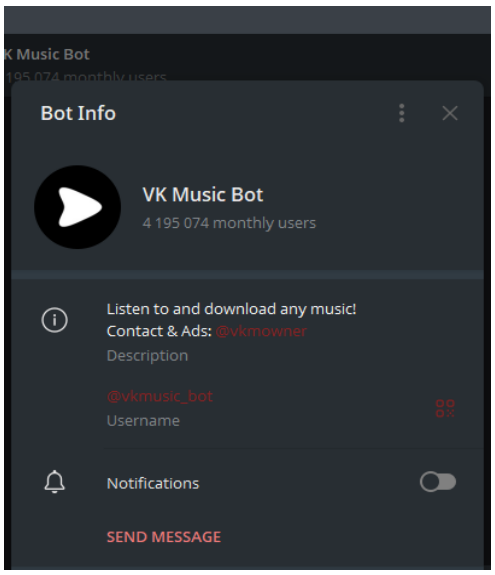

Рисунок 1.3 – Опис,аватар та юзернейм VK Music Bot

Джерело: [2]

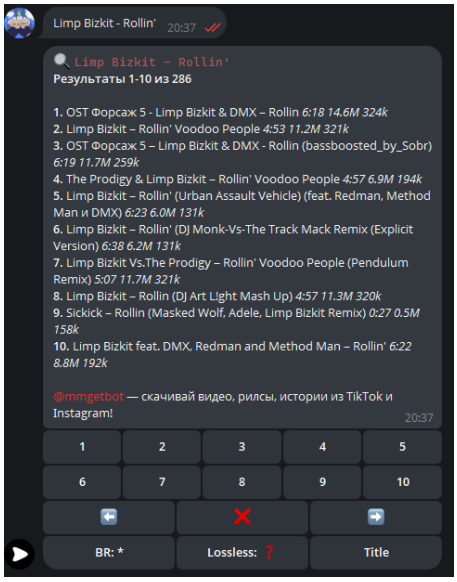

Рисунок 1.4 – Результат пошуку популярної пісні в VK Music Bot

Джерело: [2]

Shazam music bot (рис 1.5) – має куди більш розлогий функціонал, він має можливість знаходити пісні як за допомогою назви або ж виконавця, так і за рахунок інших методів – відправки тексту з пісень, голосових повідомленнях в якому лунає текст або ж мелодія пісні, знаходити музику з відправлених відео-файлів, а також знаходити пісні за допомогою посилань на відео в TikTok, YouTube та Instagram.

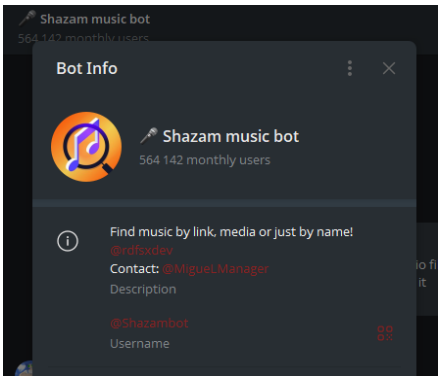

Рисунок 1.5 – Опис,аватар та юзернейм Shazam music bot

Джерело: [3]

В свою чергу, цей бот має куди більше плюсів, але й також куди більше мінусів. З плюсів можна виділити величезну варіацію пошуку, бот приймає варіанти як напряму через назву, так і розпізнає музику за допомогою інших способів, будь то посилання, відеофайл чи навіть голосовое повідомленняа також точний пошук оригінальних варіацій пісень (рис 1.6).

Мінуси цього боту дають про себе знати відразу ж після першого використання – відсутність вибору мови, реклама, можливість налаштування відображення результату а також перевантажений інтерфес.

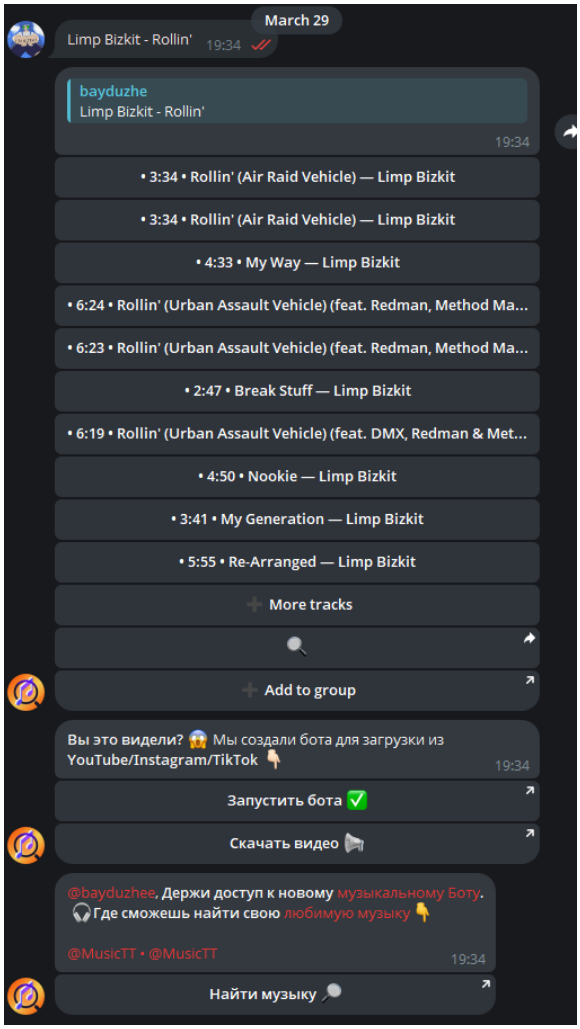

Рисунок 1.6 – Результат пошуку популярної пісні в Shazam music bot

Джерело: [3]

Обидва боти мають як величезні плюси один перед одним, так і величезні мінуси, в табл. 1.1 вони будуть наглядно наведені.

Таблиця 1.1 - Недоліки та переваги аналогів

Назва Бота	*Лаконічний пошук*	*Можливість поушку через безліч варіантів*	*Нав'язування реклами*	*Копмпактний інтерфейс*	*Налаштування*
Shazam music bot	*так*	*так*	*так*	*ні*	*ні*
VK Music Bot	*ні*	*ні*	*ні*	*так*	*так*

Завдяки цій таблиці мій телеграм бот буде створенний з урахуванням усіх плюсів та мінусів найпопулярніших ботів, заради створення комфортного середовища для користувача з можливістю варіативного та лаконічного пошуку музики.

1.3 Постановка задачі

Бот орієнтований на широке коло звичайних користувачів месенджера Telegram, які прагнуть швидко, зручно та без зайвих дій ідентифікувати музичну композицію, що їх зацікавила. Основна мета полягає у наданні користувачеві назви треку та прямого посилання на нього у популярному потоковому сервісі Spotify. Такий підхід дозволяє не лише знайти пісню, а й одразу додати її до власної бібліотеки або прослухати в декілька кліків, що відповідає сучасним очікуванням аудиторії щодо простоти доступу до контенту.

Хоча технічно можливо реалізувати функцію надсилання пісень у форматі аудіофайлів, така опція свідомо не розглядається у межах даного проєкту. Це пояснюється бажанням дотримуватись чинного законодавства у сфері інтелектуальної власності та уникнути ризиків, пов’язаних із несанкціонованим розповсюдженням авторського контенту. Розповсюдження музики у вигляді файлів без відповідної ліцензії кваліфікується як порушення авторських прав, тому реалізація легального сценарію використання є пріоритетом розробки.

Завдання, які необхідно вирішити під час створення Telegram-бота, охоплюють низку етапів як технічного, так і користувацького характеру:- отримання токену Telegram бота;

- Отримання токену Telegram-бота – реєстрація через BotFather з метою отримання авторизаційного ключа, необхідного для подальшої інтеграції з Telegram API;

- Отримання токену Spotify – налаштування облікового запису розробника на Spotify Developer Dashboard, створення додатку та отримання Client ID / Client Secret для доступу до Spotify Web API. Цей доступ дає змогу шукати треки, отримувати метадані, альбоми та інші пов’язані дані;

- Створення мінімалістичного та інтуїтивно зрозумілого інтерфейсу – бот повинен взаємодіяти з користувачем у максимально простій формі: відповідати на текстові запити, обробляти аудіо й голосові повідомлення, надавати кнопки для повторного пошуку чи зміни мови;

- Велика варіативність методів пошуку – користувач матиме змогу шукати пісню не лише за аудіофрагментом, а й за уривком тексту, голосовим повідомленням чи навіть вручну введеною назвою. Це робить бот універсальним помічником у пошуку музики.

Висновки до розділу

Telegram був створений у 2013 році Павлом Дуровим з метою тестування технології шифрування MTProto. Завдяки високому рівню безпеки він здобув репутацію одного з найбільш надійних месенджерів. Однією з ключових особливостей Telegram є боти, які дозволяють автоматизувати рутинні завдання, покращити взаємодію користувачів із цифровим середовищем та застосовуються у сфері освіти, фінансів та електронної комерції.

Боти для пошуку музики в Telegram відіграють важливу роль у доступі користувачів до аудіоконтенту. Аналіз аналогів, таких як VK Music Bot та Shazam Music Bot, дозволив визначити їхні переваги та недоліки. VK Music Bot має потужний пошуковий потенціал, але генерує занадто багато варіантів, що може ускладнити пошук оригінальної композиції. Shazam Music Bot володіє розширеними можливостями пошуку, включаючи розпізнавання музики з відео та голосових повідомлень, проте його інтерфейс перевантажений, а реклама може заважати користувачеві.

На основі проведеного аналізу було сформульовано основні завдання для розробки нового Telegram-бота MusicFi, який об'єднає найкращі функції конкурентів та запропонує користувачам:

- лаконічний та точний пошук музики;
- підтримку кількох варіантів пошуку (по назві, тексту, голосовим повідомленням, відео та посиланням);
- мінімалістичний та інтуїтивний інтерфейс;
- відсутність нав'язливої реклами;
- інтеграцію з Spotify для отримання офіційних посилань на треки;
- можливість зміни мови інтерфейсу;

РОЗДІЛ 2
ПРОЄКТУВАННЯ

2.1 Функціональні вимоги

Ефективність роботи Telegram-бота для розпізнавання музики оцінюється через глибокий аналіз його функціональних та нефункціональних характеристик. Такий підхід дозволяє не лише чітко окреслити можливості системи, а й передбачити необхідні ресурси та процеси, які слід реалізувати для забезпечення її повноцінного функціонування. Визначення вимог також створює основу для подальшого тестування, удосконалення та масштабування програмного продукту..

Функціональні вимоги :

- лаконічність пошуку – бот повинен точно визначати музичний контент у файлі або відео, яке надіслав користувач. Навіть якщо у відео використано варіацію оригінальної пісні (наприклад, ремікс або короткий фрагмент), система повинна мати здатність ідентифікувати першоджерело треку;

- універсальність та багатоканальність – однією з ключових вимог до функціональності є здатність працювати з контентом із різних популярних соціальних платформ. Зокрема, бот повинен підтримувати обробку посилань з TikTok, YouTube та Instagram – платформ, які є основними джерелами коротких відео з використанням музики;

- надання прямого посилання на прослуховування – після успішного розпізнавання треку бот має формувати пряме посилання на пісню у сервісі Spotify. Це спрощує доступ користувача до повного треку та забезпечує зручність подальшого використання;

- збереження історії пошуку – бот має зберігати пошуки користувача, задля комфорту та можливості повернутись до минулих результатів;

- кометування дій – Важливо, щоб бот інформував користувача про кожен етап своєї роботи. Наприклад, надсилання повідомлення про початок завантаження, хід розпізнавання, успіх або невдачу операції. Такий підхід підвищує прозорість функціонування сервісу, формує довіру до його результатів і покращує взаємодію з користувачем.

2.2 Нефункціональні вимоги

Нефункціональні вимоги зосереджують увагу не стільки на тому, що робить програмний продукт, скільки на тому, як саме він це реалізує. Ці вимоги стосуються якості роботи системи в умовах реального використання та впливають на зручність, ефективність і стабільність взаємодії з користувачем. Нефункціональні вимоги:

- швидкодія – бот має реагувати на дії користувача швидко та без затримок. Час від отримання запиту до відповіді не повинен перевищувати 2–3 секунд у стандартних умовах. Це забезпечує комфортну взаємодію та не змушує користувача чекати довго

- доступність – Сервіс повинен працювати безперервно, бути доступним 24/7 із мінімальним часом простою. Надійна інфраструктура й автоматичне відновлення у разі збоїв є ключовими умовами для безперебійної роботи

- масштабованість – Система має бути готова до обробки великої кількості одночасних запитів. Бот повинен витримувати високі навантаження, що виникають при пікових періодах активності користувачів. Це вимагає використання асинхронної обробки, хмарних сервісів або балансування навантаження

- зручність в користуванні – Інтерфейс бота має бути інтуїтивно зрозумілим. Мінімалістичний дизайн, короткі текстові команди та мінімальна кількість кроків до отримання результату дозволяють користувачеві

зосередитися на основній цілі – пошуку музики, – без відволікання на технічні нюанси

- локалізація – Бот повинен підтримувати як мінімум дві мови інтерфейсу: українську та англійську. Це розширює аудиторію користувачів і робить сервіс доступним для неукраїномовних користувачів, що особливо важливо у міжнародному контексті використання

2.3 Проєктування інтерфейсу

Інтерфейс реалізовано через Telegram як платформу, що не вимагає окремої розробки UI. Комунікація ведеться через команди та текстові повідомлення. Взаємодія користувача з інтерфейсом боту буде являтись максимально мінімалістичною, при старті боту користувач отримає коротеньке повідомлення з інформаційним вікном в котрому буде вітальне повідомлення яке також описує як взаємодіяти з ботом.(рис 2.1)

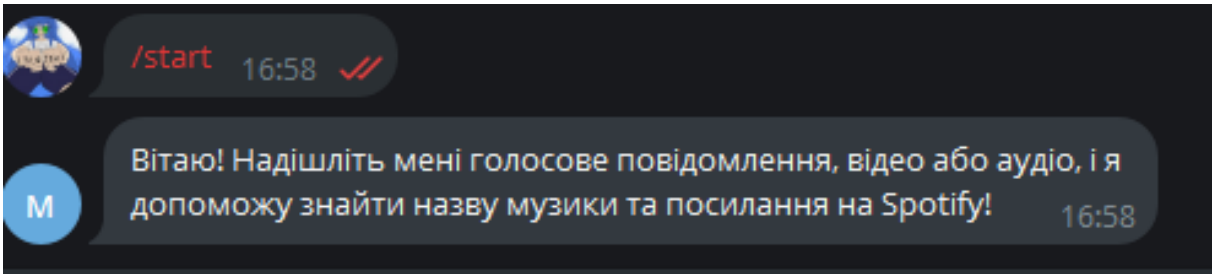

Рисунок 2.1 – Макет вітального повідомлення боту MusicFi

Джерело [4]

Ключовим інтерфейсним рішенням є команда /language, яка дозволяє змінити мову взаємодії між українською та англійською. Для зручності користувача використано клавіатуру вибору мови з двома кнопками (рис 2.2)

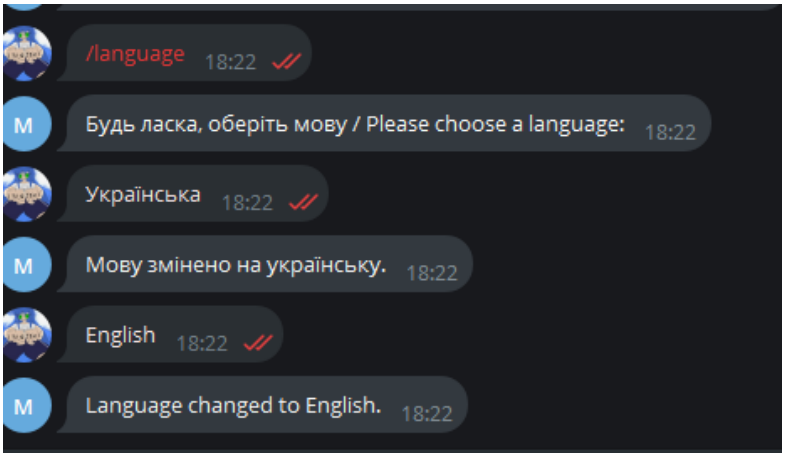

Рисунок 2.2 – Макет інтерфейсу зі зміною мови в боті MusicFi

Джерело [4]

Функціональний процес взаємодії користувача з Telegram-ботом є логічно послідовним і реалізується через низку чітко визначених етапів. Кожен етап виконує свою роль у загальному алгоритмі обробки запиту користувача – від ініціалізації до надання результату у вигляді назви пісні та посилання на неї в сервісі Spotify. Нижче наведено покроковий опис цього процесу::

- Ініціалізація взаємодії користувача з ботом – команда /start - На першому етапі користувач запускає бот за допомогою стандартної Telegram-команди /start. У відповідь бот надсилає привітальне повідомлення, яке містить коротку інструкцію щодо подальших дій

- вибір мови інтерфейсу – команда /language - Одразу після запуску користувач має можливість змінити мову взаємодії з ботом за допомогою команди /language. Бот пропонує вибір між доступними мовами (українська та англійська), що забезпечує зручність користування для широкого кола користувачів;

- користувач відправляє посилання/відео боту;

- завантаження відеоконтенту ботом - Бот автоматично аналізує отриману інформацію;

- конвертація відео у формат аудіо (формат аудіофайлу) - Після отримання відео бот здійснює його обробку – конвертацію у формат MP3. Це необхідний крок для подальшого аналізу аудіотреку;

- розпізнавання музики - аудіофайл передається до сервісів розпізнавання через Python-бібліотеку яка виконує розпізнавання композиції – визначає назву треку та виконавця;

- додатковий пошук у Spotify за назвою пісні - Отриману інформацію (назву треку та виконавця) бот передає у бібліотеку, в якій виконується додатковий пошук У відповідь отримується унікальне посилання на пісню в Spotify;

- надання результатів користувачу - на завершальному етапі бот формує повідомлення, яке містить знайдену назву пісні, ім’я виконавця та активне посилання на трек у Spotify. Таким чином, користувач може миттєво перейти до прослуховування музики.

Приклад функціональної відопвіді бота (рис 2.3)

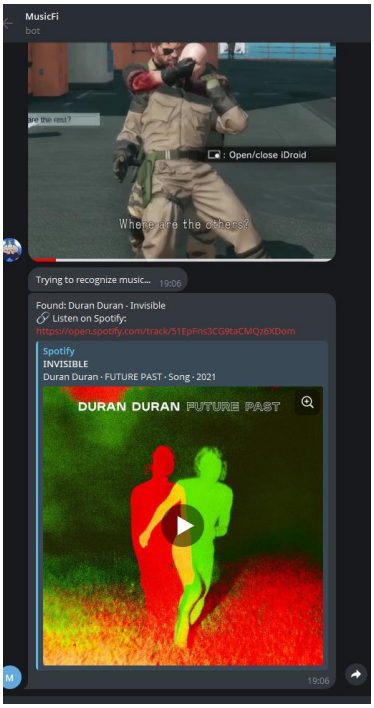

Рисунок 2.3 – Макет функціоналу бота MusicFi

Джерело [4]

Висновки до розділу

Розглянуто функціональні та нефункціональні вимоги.

Було розглянуто та розроблено інтерфейс бота, а також функціонуючий прототип з можливістю зміни мови а повним запланованим функціоналом.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ

3.1 Способи реалізації

У рамках проєкту було поставлено задачу розробити Telegram-бота, який дозволяє здійснювати пошук музики за аудіофрагментами, що надсилаються користувачем. Основні функціональні та нефункціональні було розглянуто, прийшов час для розглянення можливих методів реалізації. Для досягнення цієї мети розглядалося кілька можливих способів, які можна умовно поділити на три категорії: використання сторонніх API-сервісів, застосування локальних або напівавтономних бібліотек для аудіорозпізнавання, а також варіанти на основі комбінованих рішень.

3.1.1 Використання сторонніх API-сервісів для розпізнавання музики

Один із найпростіших та швидких способів реалізації функції розпізнавання музики полягає у використанні вже готових хмарних сервісів, які надають доступ до спеціалізованих аудіофінгерпринтингових систем через API. Ці сервіси мають попередньо навчені моделі, розподілені обчислювальні потужності та величезні бази даних пісень. Завдяки цьому, розробник не має потреби в реалізації власного алгоритму розпізнавання чи формування бази з мільйонів аудіотреків.

ACRCloud[5] – це один із лідерів ринку аудіоідентифікації. Сервіс дозволяє проводити розпізнавання за допомогою REST API або клієнтських SDK. Він підтримує широкий спектр форматів, працює з аудіофрагментами довжиною від 1 до 12 секунд та повертає вичерпну інформацію про трек: назву, виконавця, альбом, а також ідентифікатори для Spotify, Apple Music, YouTube, тощо.

Переваги ACRCloud:

- висока точність та швидкість розпізнавання;

- широке покриття міжнародної музики;
- інтеграція з декількома музичними платформами;
- документація для різних мов програмування.

Недоліки:

- обмеження безкоштовного тарифу (до 1000 запитів на місяць);
- необхідність реєстрації та зберігання ключів API;
- відсутність повного контролю над процесом розпізнавання.

Переваги підходу:

- висока точність і мінімальні зусилля з боку розробника;
- не потребує навчання моделей чи створення баз даних;

Обмеження:

- обмежена кількість безкоштовних запитів;
- залежність від стабільності сервісу та API-політик;
- необхідність роботи з конфіденційними токенами;
- неможливість працювати в офлайн-режимі.

Цей підхід підходить для швидкого запуску мінімально життєздатного продукту або коли пріоритетом є швидкість розробки, а не повна автономність системи.

3.1.2 Реалізація на основі бібліотек локального або відкритого типу

Для досягнення більшої автономності, гнучкості та контролю над обробкою запитів у рамках даного проєкту також було переглянуто реалізацію Telegram-бота на основі відкритих бібліотек, які не потребують обов’язкового підключення до зовнішніх API. Основною перевагою такого підходу є можливість виконувати ключові операції на стороні сервера, без надмірної залежності від сторонніх сервісів.

Telebot[6] (pyTelegramBotAPI) – бібліотека для синхронної роботи з Telegram Bot API. Дозволяє легко організувати маршрутизацію повідомлень,

обробку вкладених аудіо/голосових повідомлень, відповіді та командну логіку. Хоча вона є синхронною, її простота і стабільність роблять її зручною для невеликих проєктів.

Shazamio[7] – асинхронна Python-бібліотека, яка працює через неофіційне API Shazam. Вона дозволяє ідентифікувати пісню за аудіофрагментом та повертає розширену інформацію: виконавець, назва треку, жанр, обкладинка, дата релізу тощо.

Asyncio [8] – стандартна бібліотека для асинхронного програмування в Python. Необхідна для коректного виклику методів shazamio. Використовується або через asyncio.run(), або через створення подій у поточному потоці (loop.run_until_complete()).

Ffmpeg [9] – засоби для конвертації та обробки аудіо. Наприклад, Telegram надсилає голосові повідомлення у форматі .ogg , який не підтримується більшістю аудіоаналізаторів. Через ffmpeg файл перетворюється на .mp3 або .wav перед подачею до shazamio.

Цей підхід забезпечує більшу автономність, дозволяє уникнути залежності від платних API, забезпечує хорошу точність розпізнавання та можливість гнучкого керування логікою роботи бота.

Переваги підходу:

- Безкоштовна розробка без тарифних обмежень;
- Повний контроль над усіма етапами обробки;
- Можливість подальшого розширення та інтеграції з іншими системами;
- Легкість у розгортанні на власному сервері.

Недоліки:

- Потребує налаштування серверного середовища (ffmpeg, python-ffmpeg, libopus);
- Потребує обробки асинхронного коду у синхронному контексті (telebot);

Цей підхід забезпечує більшу автономність, дозволяє уникнути залежності від платних API, забезпечує хорошу точність розпізнавання та можливість гнучкого керування логікою роботи бота.

3.1.3 Комбінований підхід

Ще одним варіантом реалізації є створення гібридної (комбінованої) архітектури, яка об’єднує переваги локальних бібліотек, зовнішніх сервісів і внутрішньої обробки даних. Такий підхід особливо доцільний у випадках, коли потрібно досягти балансу між точністю, швидкістю відповіді, автономністю та масштабованістю проєкту.

Основні ідеї комбінованого підходу:

Попереднє локальне фільтрування та підготовка перед відправкою аудіофрагменту на розпізнавання – використовується ffmpeg та pydub для нормалізації гучності, усунення шумів, обрізки до оптимальної довжини.

Основне розпізнавання проходить за допомогою локальних бібліотека(наприклад, shazamio), якщо розпізнавання не дало точного результату (або повернуло низьку впевненість), бот може автоматично зробити другий запит до зовнішнього API, наприклад ACRCloud або AudD.

Це дозволяє зменшити витрати на сторонні сервіси, використовуючи їх лише тоді, коли локальні методи не спрацювали.

Для зниження навантаження на обчислення та API-сервіси результати розпізнавання можуть зберігатися в локальній базі даних (наприклад, SQLite або PostgreSQL).

Якщо той самий аудіофрагмент або схожий вже був оброблений раніше - бот одразу повертає збережений результат.

Переваги такого підходу:

- Гнучкість у виборі методу розпізнавання;

- Вища точність завдяки резервному пошуку через зовнішні API;

- Оптимізація витрат та ресурсів;

Недоліки:

- Більша складність реалізації логіки та обробки помилок;
- Потреба в більшій кількості зовнішніх залежностей;
- Необхідність ретельного керування правами доступу до сторонніх API.

3.1.4 Інтеграція зі Spotify API

Окрім розпізнавання пісень, важливою частиною функціоналу є пошук знайденого треку в каталозі Spotify для надання користувачу повної інформації про нього – назви, обкладинки альбому, виконавця, дати релізу, а також прямого посилання на відтворення.

Для цього може використовуватись:

spotipy – офіційна Python-бібліотека для роботи з Spotify Web API. Дозволяє здійснювати пошук треків, отримувати альбоми, створювати або редагувати плейлисти (при наявності авторизації).

Альтернативно можна надсилати запити напряму до Spotify API, використовуючи бібліотеки requests, httpx або aiohttp.

Інтеграція зі Spotify забезпечує завершеність користувацького сценарію: від розпізнавання аудіофрагменту – до можливості відразу прослухати пісню на популярному стримінговому сервісі.

3.1.5 Альтернативи

У межах аналізу також розглядалися інші варіанти реалізації.

aiogram – асинхронна альтернатива telebot, яка дозволяє будувати Telegram-ботів повністю в асинхронному середовищі. Це могло б полегшити інтеграцію з shazamio, але потребує повного переписування логіки на асинхронний підхід.

dejavu – open-source реалізація алгоритму аудіофінгерпринтингу, аналогічного Shazam. Дозволяє розпізнавати музику на основі попередньо зібраної бази треків. Підходить для великих автономних систем, але вимагає значного обсягу підготовки даних та серверних ресурсів.

librosa, pydub – бібліотеки для аналізу аудіо, які можуть бути використані для попередньої обробки звукових сигналів або екстракції фіч (мел-спектрограм, частот тощо), однак не забезпечують повноцінного розпізнавання композицій.

Після розгляду різних варіантів реалізації було обрано оптимальний підхід, який поєднує бібліотеки telebot, asyncio, ffmpeg та shazamio. Такий стек дозволяє розробити Telegram-бота з мінімальними залежностями від зовнішніх сервісів, забезпечити достатню гнучкість, стабільність та масштабованість рішення. Інтеграція зі Spotify API дозволяє доповнити функціонал і зробити результат максимально корисним для користувача. У майбутньому структура проєкту дозволяє легко розширити можливості, наприклад, додавши підтримку пошуку текстів пісень або створення персоналізованих рекомендацій.

3.1.6 Огляд способів

Розглянувши усі доступні способи реалізації поставленої задачі, я дійшов висновку, що доцільно систематизувати отриману інформацію у вигляді таблиці (табл. 3.1), яка наочно демонструє ключові етапи реалізації кожного з варіантів, їх функціональні особливості, переваги та потенційні обмеження. Такий підхід дозволяє не лише зручно порівняти можливі шляхи реалізації, але й обґрунтовано обрати оптимальний із них відповідно до поставлених цілей проєкту.

На основі порівняння можливих підходів(табл 3.1) було обрано рішення реалізувати Telegram-бота з використанням локальних та відкритих бібліотек, зокрема shazamio, ffmpeg, telebot та asyncio. Цей підхід дозволяє забезпечити повний контроль над обробкою аудіофайлів, не залежати від сторонніх API-сервісів та уникнути обмежень, пов'язаних із платними тарифами або політиками

використання сторонніх платформ. Крім того, обрана реалізація легко розширюється і адаптується до нових вимог, що дозволяє інтегрувати додаткові джерела пошуку, кешування та навіть резервні API в майбутньому. Таким чином, було обрано найбільш стабільну, автономну та економічно доцільну архітектуру для поставленої задачі.

Таблиця 3.1 – Огляд варіантів

Підхід	*Опис*	*Основні бібліотеки/ сервіси*	*Переваги*	*Недоліки*
Сторонні API-сервіси	*Використання готових хмарних рішень для розпізнавання аудіо та інтеграції з музичними платформами через REST API.*	*ACRCloud, AudD API, Gracenote, Musixmatch API*	*Висока точність, простота реалізації, не потребує навчання моделей або створення баз треків*	*Обмеження безкоштовного тарифу, залежність від сторонніх сервісів, потрібні API-ключі*
Локальні та відкриті бібліотеки (використано в проєкті)	*Робота з аудіофайлом виконується локально, розпізнавання відбувається через публічно доступні інструменти, не прив'язані до платних API.*	*shazamio, telebot, ffmpeg, asyncio, pydub*	*Повна автономність, безкоштовність, гнучкість, легке розгортання*	*Потрібна робота з асинхронним кодом, складніша обробка форматів*
Комбіновані рішення	*Поєднання локальної обробки, резервного використання API*	*shazamio, ACRCloud / AudD, ffmpeg, pydub, кешування з SQLite/PostgreSQL*	*Баланс між точністю і автономністю, гнучкість*	*Вища складність логіки, потреба в грамотному управлінні ресурсами*

3.2 Конструювання

3.2.1 Отримання та ініціалізація токенів

Для отримання токену Telegram-бота необхідно скористуватись ще одним ботом, а саме офіційним ботом розробленим спеціально для створення, адміністрування та редагування властних ботів – BotFather [10] (рис 3.1). Цей бот є центральним інструментом Telegram для розробників, і через нього можна створювати нових ботів, змінювати їхні назви, описи, зображення, налаштовувати команди та інші параметри взаємодії.

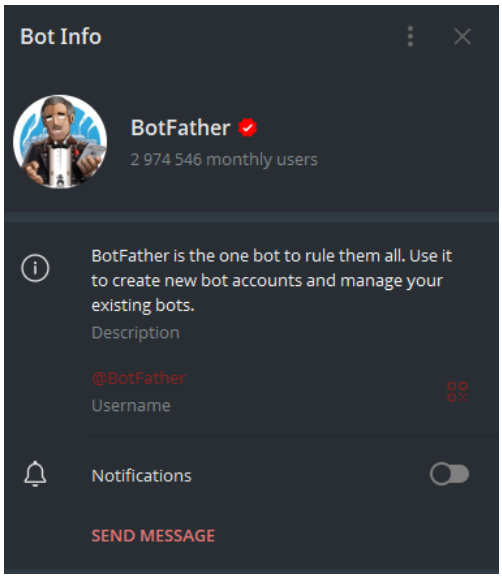

Рисунок 3.1 – BotFather

Джерело: [10]

Отримання токену власного Telegram-бота виконується за допомогою єдиної команди, після якої слід ввести назву майбутнього бота, а також його унікальне посилання в мережі Telegram. Після завершення цих дій система автоматично згенерує токен доступу – це спеціальний рядок символів, що дозволяє програмі ідентифікувати себе як конкретного бота в API Telegram (рис. 3.2).

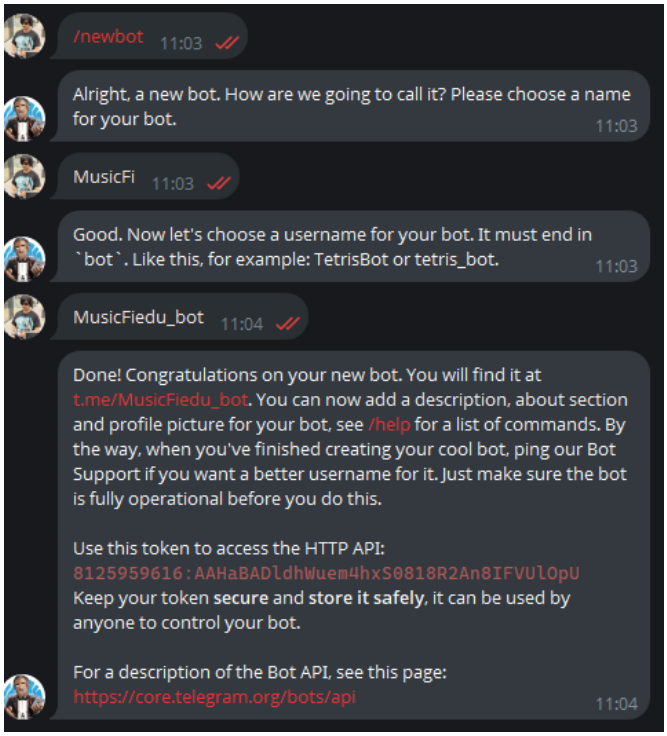

Рисунок 3.2 – Отримання токену Telegram-бота

Джерело: [10]

Коли Telegram-токен уже наявний, наступним кроком є отримання токену для інтеграції зі Spotify. Для цього необхідно зареєструватися як розробник у сервісі Spotify for Developers Dashboard [11], створити новий додаток на сайті, після чого буде згенеровано пару значень – Client ID та Client Secret, які й використовуються як токен доступу до Spotify API (рис. 3.3)

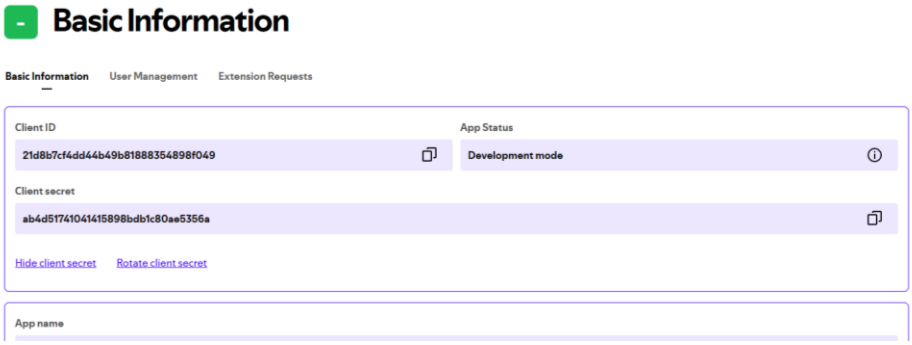

Рисунок 3.3 – Токен Spotify

Джерело: [11]

Ці облікові дані дозволяють додатку взаємодіяти з відкритими даними платформи Spotify – зокрема здійснювати пошук треків, переглядати альбоми, отримувати прямі посилання на композиції тощо.

Тепер, коли усі необхідні токени отримано та налаштовано, можна переходити безпосередньо до етапу написання програмного коду, що реалізуватиме логіку Telegram-бота та його взаємодію з користувачем і зовнішніми сервісами.

3.2.2 Імпорт бібліотек

Для початку написання коду нам необхідно імпортувати усі необхідні бібліотеки.(рис 3.4)

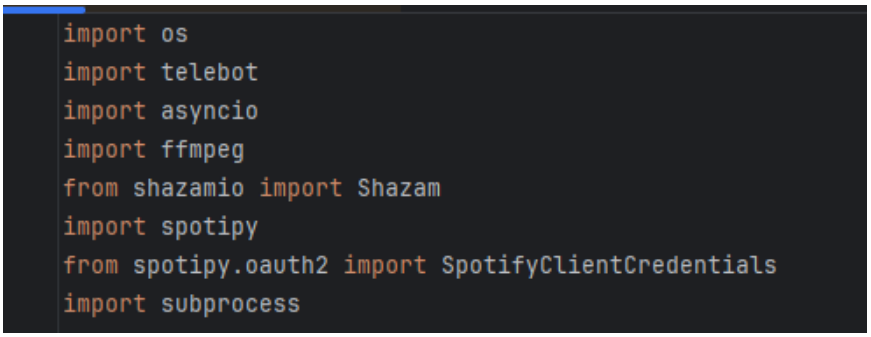

Рисунок 3.4 – Імпортовані бібліотеки

Джерело: розроблено автором

Модуль os – стандартна бібліотека Python, яка надає функції для роботи з операційною системою. Це дозволяє динамічно керувати файловою структурою під час обробки медіа.

Наша основна бібліотека, яка забезпечує роботу Telegram-бота – telebot (відомий також як PyTelegramBotAPI). Він дає змогу ініціалізувати бота за допомогою токена, прослуховувати вхідні повідомлення, обробляти текст, аудіо, відео, команди (в нашому випадку це /start, /language) та надсилати відповіді.

Бібліотека є синхронною, що в цьому випадку спрощує логіку обробки повідомлень.

shazamio – це асинхронна Python-бібліотека, яка реалізує неофіційний доступ до Shazam API. У коді створюється об’єкт Shazam, за допомогою якого викликається метод recognize(file_path), що повертає інформацію про знайдену композицію (назва, виконавець, обкладинка тощо). Бібліотека дозволяє працювати без ключів API, що значно спрощує розгортання проєкту.

asyncio – стандартна бібліотека для керування асинхронними процесами у Python. Вона потрібна для роботи з бібліотекою shazamio, яка є асинхронною. Оскільки telebot – синхронна, asyncio використовується для створення окремих подій, які дозволяють запускати асинхронні функції всередині синхронного контексту. Це дає змогу уникнути конфліктів між синхронною логікою Telegram і асинхронними запитами до Shazam.

Також імпортується Python-інтерфейс для утиліти FFMPEG, яка використовується для обробки медіафайлів. Зокрема, у коді вона застосовується для конвертації голосових повідомлень Telegram (.ogg) або витягнутих з відео аудіофрагментів у формат .mp3, необхідний для розпізнавання музики.

Конвертація відбувається без виведення логів, з налаштуванням параметрів якості, що забезпечує стабільність і продуктивність.

spotipy – офіційна Python-бібліотека для взаємодії з Spotify Web API. Ця бібліотека готує інструментарій для реалізації логіки пошуку знайденого треку у Spotify.

from spotipy.oauth2 import SpotifyClientCredentials - це конкретний клас з модуля spotipy, який дозволяє отримати доступ до Spotify API без авторизації конкретного користувача. Цей механізм підходить для сценаріїв, коли потрібно лише здійснювати пошук, без створення або редагування плейлистів.

subprocess – стандартний модуль Python, що дозволяє виконувати команди операційної системи. У даному коді він використовується для запуску утиліти yt-

dlp, яка завантажує відео за URL з YouTube, TikTok або Instagram. Це рішення дозволяє розширити функціонал бота без складних API-інтеграцій з кожною з платформ

3.2.3 Авторизація Telegram-бота та ініціалізація Spotify

Після підключення всіх необхідних бібліотек, наступним етапом у реалізації бота стало створення ключових змінних, об’єктів та допоміжних структур, необхідних для коректної роботи з API Telegram, Spotify та медіаконтентом користувача.

Створюємо змінну TOKEN (рис3.5), у яку записується токен доступу, отриманий через BotFather. Потім створюється об'єкт bot, який представляє Telegram-бота і слугує точкою входу для всіх подальших команд та обробки повідомлень користувачів.

```
TOKEN = "8125959616:AAHaBADldhWuem4hxS0818R2An8IFVUlOpU"
bot = telebot.TeleBot(TOKEN)

SPOTIFY_CLIENT_ID = "21d8b7cf4dd44b49b81888354898f049"
SPOTIFY_CLIENT_SECRET = "ab4d51741041415898bdb1c80ae5356a"
sp = spotipy.Spotify(auth_manager=SpotifyClientCredentials(
    client_id=SPOTIFY_CLIENT_ID,
    client_secret=SPOTIFY_CLIENT_SECRET
))
```

Рисунок 3.5 – Токени

Джерело: розроблено автором

Оголошуємо дві нові змінні – SPOTIFY_CLIENT_ID і SPOTIFY_CLIENT_SECRET(рис 3.5). Їх значення генеруються у панелі розробника Spotify після створення додатку. Далі ці дані передаються у SpotifyClientCredentials, який використовується для отримання токену доступу до публічного Spotify API.

Створений на основі цього токену об'єкт sp є клієнтом Spotify, через якого надалі відбуватиметься пошук треків за результатами розпізнавання.

3.2.4 Асинхронна функція для розпізнавання музики

Створюємо асинхронну функцію recognize_music(file_path), яка реалізує логіку взаємодії з бібліотекою shazamio (рис 3.6). Ця функція приймає шлях до mp3-файлу, обробляє його, та повертає результат розпізнавання – тобто структуру з назвою треку, виконавцем та іншими метаданими. Для обробки можливих винятків передбачено except, завдяки якому бот не завершує роботу у разі помилки.

```
async def recognize_music(file_path):
    shazam = Shazam()
    try:
        result = await shazam.recognize(file_path)
        return result
    except Exception as e:
        print(f"Помилка при розпізнаванні музики: {e}")
        return None
```

Рисунок 3.6 – Асинхронна функція

Джерело: розроблено автором

Ця функція є критичним елементом системи, оскільки саме вона відповідає за основну функцію бота – розпізнавання композиції за аудіофрагментом. Її асинхронність забезпечує продуктивну роботу системи навіть при великій кількості запитів

3.2.5 Створення функції конвертації та пошуку у Spotify

Однією з обов’язкових умов для успішної обробки аудіофайлу через бібліотеку shazamio є формат mp3. Telegram надсилає голосові повідомлення у

форматі .ogg, а відео, завантажені з YouTube або інших платформ, мають свої специфікації звуку. Тому перед передачею в Shazam API, аудіо необхідно стандартизувати.

Для цього у проєкті створено окрему функцію (рис 3.7), яка здійснює конвертацію будь-якого аудіофайлу у формат .mp3 із заданими параметрами якості. Конвертація відбувається за допомогою бібліотеки ffmpeg, яка викликається у режимі мовчазної обробки, без виведення логів або повідомлень у консоль.

```
def convert_to_mp3(input_file, output_file):
    try:
        ffmpeg.input(input_file).output(output_file, format='mp3', audio_bitrate='128k').run(overwrite_output=True, quiet=True)
        return True
    except Exception as e:
        print(f"Помилка при конвертації: {e}")
        return False
```

Рисунок 3.7 – Функція конвертації

Джерело: Розроблено автором

Після того як трек було успішно сконвертовано та розпізнано, виникає необхідність надати користувачу не лише назву пісні та виконавця, а й пряме посилання на її прослуховування. Для цього використовується функція, яка реалізує пошук треку в Spotify через об’єкт sp, ініціалізований раніше (рис 3.8).

```
def get_spotify_link(artist, track):
    query = f"{artist} {track}"
    results = sp.search(q=query, limit=1, type='track')
    if results['tracks']['items']:
        return results['tracks']['items'][0]['external_urls']['spotify']
    return None
```

Рисунок 3.8 – Функція пошуку посилання в Spotify

Джерело: Розроблено автором

Функція працює наступним чином:

- приймає два вхідні параметри: назву пісні та ім’я виконавця;
- формує пошуковий запит, що складається з обох цих значень;
- передає запит до Spotify Web API через метод search;

У разі успішного знаходження результату – повертає посилання на трек, якщо результат відсутній – повертає повідомлення про неможливість знайти композицію.

Щоб уникнути перевантаження результатів, обмежено максимальну кількість результатів до одного. Після чого з результату береться URL для зовнішнього доступу до треку в Spotify.

3.2.6 Реалізація функції завантаження по посиланню

Функція download_video_audio(url) (рис 3.9) відповідає за завантаження відео з онлайн-ресурсів (TikTok, YouTube, Instagram) за вказаним користувачем посиланням. Для цього використовується утиліта yt-dlp, яка запускається як системна команда через subprocess.run(). Відео зберігається у тимчасову директорію downloads з шаблонною назвою. Після завершення завантаження функція перевіряє, чи файл з'явився, і повертає повний шлях до нього. У разі помилки або відсутності результату функція повертає None.

```python
def download_video_audio(url):
    try:
        output_path = "downloads/downloaded_video.%(ext)s"
        command = f'yt-dlp -o "{output_path}" {url}'
        subprocess.run(command, shell=True, check=True)

        downloaded_files = [file for file in os.listdir("downloads") if file.startswith("downloaded_video")]
        if downloaded_files:
            return os.path.join("downloads", downloaded_files[0])
        else:
            return None
    except Exception as e:
        print(f"Помилка при завантаженні відео: {e}")
        return None
```

Рисунок 3.9 – Функція завантаження

Джерело: Розроблено автором

Цей компонент дозволяє реалізувати універсальну обробку відеоконтенту без використання окремих API для кожної платформи, забезпечуючи гнучкість і масштабованість системи.

3.2.7 Взаємодія з користувачем

На данному етапі реалізовано базову взаємодію Telegram-бота з користувачем. Після надсилання команди /start бот ініціює вітальне повідомлення та пропонує вибір мови інтерфейсу (рис 3.10). Для зручності реалізовано клавіатуру з мовними варіантами, після вибору яких інформація зберігається у словнику user_languages, що забезпечує персоналізацію подальшої комунікації.

Крім цього, передбачено можливість зміни мови на будь-якому етапі роботи бота за допомогою команди /language, що підвищує гнучкість і зручність використання.

Рисунок 3.10 – Вітальне меню бота MusicFi

Джерело: [5]

Додатково обробляються текстові повідомлення. Якщо бот виявляє в тексті гіперпосилання (наприклад, на відео з YouTube або TikTok), він сприймає це як запит на завантаження мультимедійного контенту й ініціює відповідну процедуру обробки. У разі, якщо повідомлення не містить очікуваного формату, користувач отримує підказку щодо допустимих варіантів взаємодії. Окрім цього

бот також інформує користувача про кожну свою дію та помилку, за допомогою чого доволі просто відслідковувати етапи на яких виникли труднощі.

Цей блок забезпечує інтуїтивно зрозумілий механізм старту та первинної навігації, адаптований під користувача з мінімальним досвідом роботи з ботами, і є основою для подальшого функціонування системи.

3.2.8 Розробка основної функції обробки

Ця функція об’єднує в собі весь повний ланцюг обробки: завантаження, конвертація, розпізнавання, пошук у Spotify, формування та надсилання відповіді(рис 3.11).

```
def process_audio_video(file_path, chat_id, lang='ua'):
    mp3_file = f"{file_path}.mp3"
    if convert_to_mp3(file_path, mp3_file):
        loop = asyncio.new_event_loop()
        asyncio.set_event_loop(loop)
        result = loop.run_until_complete(recognize_music(mp3_file))

        if result and 'track' in result:
            track = result['track']['title']
            artist = result['track']['subtitle']
            spotify_link = get_spotify_link(artist, track)

            if spotify_link:
                bot.send_message(chat_id, f"Found: {artist} - {track}\n🔗 Listen on Spotify: {spotify_link}" if lang == 'en' else f"Знайдено: {artist} - {track}\n🔗 Слухати у Spotify: {spotify_link}")
            else:
                bot.send_message(chat_id, f"Found: {artist} - {track}\nSpotify link not found." if lang == 'en' else f"Знайдено: {artist} - {track}\nПосилання на Spotify не знайдено.")
        else:
            bot.send_message(chat_id, "Could not recognize music." if lang == 'en' else "Не вдалося розпізнати музику.")
    else:
        bot.send_message(chat_id, "File processing error." if lang == 'en' else "Помилка при обробці файлу.")

    os.remove(file_path)
```

Рисунок 3.11 – Основна функція обробки

Джерело: Розроблено автором

Структура її роботи виглядає наступним чином :

Бот завантажує та конвертує файл користувача/відео або аудіофайл з посилання у формат .mp3 за допомогою раніше створеної функції, асинхронне розпізнає композиції з використанням shazamio та asyncio та обробляє результат.

Якщо композицію знайдено – формується повідомлення з назвою треку, виконавцем та посиланням Spotify, якщо ні – надсилається ввічливе повідомлення про відсутність результату.

В кінці обробки виконується видалення тимчасових файлів після завершення операції – як mp3, так і оригінального завантаженого файлу.

Висновки до розділу

У межах розглянутого функціонального блоку було реалізовано повноцінну логіку взаємодії Telegram-бота з користувачем. Забезпечено інтуїтивно зрозумілий старт, підтримку мультимовності, автоматичне розпізнавання текстових запитів та інтеграцію з популярними відеоплатформами. Це дозволяє ефективно направляти користувача до основної функціональності сервісу, формуючи позитивний досвід взаємодії з ботом і закладаючи основу для подальшої обробки мультимедійного контенту.

ВИСНОВКИ

Виконана кваліфікаційна робота присвячена розробці Telegram-боту "MusicFi". У ході проекту було досліджено та впроваджено всі етапи створення боту, починаючи з проектування концепції та закінчуючи написанням функціонуючого продукту.

Проведено аналіз існуючих аналогів боту, визначено основні вимоги як функціональні так і не функціональні, розроблено інтерфейс, створено функціонал для пошуку з популярних платформ та впроваджено можливості конвертація яка відбувається без виведення логів, з налаштуванням параметрів якості, що забезпечує стабільність і продуктивність.

У процесі розробки було використано сучасні інструменти та технології, зокрема середовище Telegram, що дозволило забезпечити високий рівень стабільності та відзивчивості бота. Також було застосовано asyncio що дало змогу запускати асинхронні функції всередині синхронного контексту. Це дає дозволило уникнути конфліктів між синхронною логікою Telegram і асинхронними запитами до Shazam

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Стаття Telegram у Wikipedia URL: https://uk.wikipedia.org/wiki/Telegram *(дата звернення 27.03.2025).*
2. Графік зростання кількості Telegram-ботів URL : https://whop.com/blog/telegram-statistics/?utm_source=chatgpt.com *(дата звернення 27.03.2025).*
3. VK music bot URL : https://t.me/vkmusic_bot *(дата звернення 27.03.2025).*
4. Shazam music bot URL : https://t.me/Shazambot *(дата звернення 27.03.2025).*
5. MusicFi bot URL : https://t.me/MusicFiedu_bot *(дата звернення 03.04.2025)*
6. ARCcloud URL : https://www.acrcloud.com *(дата звернення 19.04.2025)*
7. TeleBot URL : https://pypi.org/project/pyTelegramBotAPI/ *(дата звернення 19.04.2025)*
8. Shazamio URL : https://github.com/shazamio/ShazamIO *(дата звернення 19.04.2025)*
9. Ffmpeg URL : https://www.ffmpeg.org *(дата звернення 19.04.2025)*
10. BotFather URL : https://t.me/BotFather *(дата звернення 22.04.2025)*
11. Spotify for Developers Dashboard. URL : https://developer.spotify.com *(дата звернення 22.04.2025)*