МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____Юрій КОНДРАТЕНКО

«__» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ТЕЛЕГРАМ ЧАТ-БОТ ДЛЯ АВТОМАТИЗАЦІЇ ПОШУКУ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____Богдан СЕРБУЛОВ

«___»_____ 2025 р.

Керівник канд. пед. наук, доцент _____Катерина КІРЕЙ

«___»_____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____Юрій КОНДРАТЕНКО

«___» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Сербулова Богдана Артемовича

((прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Телеграм чат-бот для автоматизації пошуку мультимедійного контенту».

Керівник роботи: Кірей Катерина Олександрівна, доцент кафедри, канд. пед. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2024 р. № 321.

2. Строк представлення кваліфікаційної роботи «___» _____ 2025 р.

3. Очікуваний результат роботи та дані, якщо такі потрібні: система роботи телеграм боту, для пошуку мультимедійного контенту.

4. Перелік питань, що підлягають розробці: Яким чином забезпечити ефективний пошук фільмів за неповною або некоректною назвою, введеною користувачем у чаті Telegram, як реалізувати архітектуру Telegram-бота на основі асинхронної обробки запитів із використанням бібліотеки Aiogram 3, які методи попередньої обробки текстових даних користувача забезпечують підвищення точності пошуку

Керівник роботи _____ Катерина КІРЕЙ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Богдан СЕРБУЛОВ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «18» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Телеграм чат-бот для автоматизації пошуку мультимедійного контенту

№	Найменування роботи	Початок	Закінчення	Примітки
1	Вибір теми та погодження з керівником	18.12.2024	20.12.2024	
2	Аналіз предметної області та огляд аналогічних рішень	20.12.2024	02.01.2025	
3	Реалізація обраних технологій з аналізом отриманих результатів	02.01.2025	05.02.2025	
4	Постановка задачі, визначення функціоналу та архітектури бота	01.03.2025	03.03.2025	
5	Розробка Telegram-бота (розробка, тестування, налагодження)	04.04.2025	10.04.2025	
6	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	
7	Корегування роботи за результатами попереднього захисту	29.05.2025	29.05.2025	
8	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	
9	Доробка та остаточне оформлення КР	31.05.2025	31.06.2025	
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	11.06.2025	17.06.2025	

Керівник роботи _____ Катерина КІРЕЙ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Богдан СЕРБУЛОВ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи

здобувача групи 401 ЧНУ ім. Петра Могили

Сербулова Богдана Артемовича

на тему: “**ТЕЛЕГРАМ ЧАТ-БОТ ДЛЯ АВТОМАТИЗАЦІЇ ПОШУКУ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ**”

Актуальність: стрімке зростання популярності месенджерів, зокрема Telegram, зумовлює необхідність створення інструментів автоматизованого доступу до інформації. Telegram-боти дозволяють ефективно взаємодіяти з користувачем через простий інтерфейс без встановлення застосунків.

Об’єктом роботи є процес розробки Telegram-бота.

Предметом роботи є виступають технології побудови Telegram-ботів.

Метою роботи є розробка із використанням сучасних інструментів, мов програмування та API сервісів Telegram-бота, який поєднує функціонал мультимедійного пошуку з можливостями генеративного ШІ у єдиний інтерфейс.

Фахова частина кваліфікаційної складається зі вступу, чотирьох розділів, висновків та переліку джерел посилання. У розділі 1 зроблено аналіз предметної області, огляд аналогів, а також сформульовано постановку задачі. У розділі 2 досліджено методи реалізації функцій бота, обрано технології, описано структуру коду. У розділі 3 подано опис структури вхідних даних, зроблено моделювання системи з допомогою UML-діаграм, а також складено план аналізу розробленого функціоналу проєкту. У розділі 4 представлено повну реалізацію Telegram-бота, інструкцію для користувача, результати.

У результаті реалізовано повнофункціональний Telegram-бот, який підтримує кілька типів пошуку, має зручний інтерфейс, гнучку архітектуру, підтримку станів користувача та обробку помилок.

Кваліфікаційна робота містить **68 сторінок**, **24 рисунків**, **2 таблиці**, **13 використаних джерел** та **1 додатків**.

Ключові слова: Telegram-бот, Python, aiogram, OMDb API, ChatGPT, пошук мультимедійного контенту, автоматизація.

ABSTACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Bohdan Serbulov

“TELEGRAM CHATBOT TO AUTOMATE THE SEARCH FOR MULTIMEDIA CONTENT”

Relevance: The rapid growth in the popularity of messengers, including Telegram, necessitates the creation of tools for automated access to information. Telegram bots allow you to effectively interact with the user through a simple interface without installing applications.

The object of the study is the process of developing a Telegram bot.

The subject of this paper is the technology of building Telegram bots.

The goal is to develop a Telegram bot that combines multimedia search functionality with generative AI capabilities into a single interface using modern tools, programming languages, and API services.

The professional part of the qualification thesis consists of an introduction, four chapters, conclusions and a list of references. Section 1 analyses the subject area, reviews analogues, and formulates the task statement. Section 2 examines methods of implementing bot functions, selects technologies, and describes the code structure. Section 3 describes the structure of the input data, modelling the system using UML diagrams, and draws up a plan for analysing the developed project functionality. Section 4 presents the full implementation of the Telegram bot, user manual, and results.

As a result, we have implemented a fully functional Telegram bot that supports several types of search, has a user-friendly interface, flexible architecture, support for user states, and error handling

The qualification work contains 68 pages, 24 figures, 2 tables, 13 references and 1 appendices.

Keywords: Telegram bot, Python, aiogram, OMDb API, ChatGPT multimedia content search, automation

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ШІ – штучний інтелект

FSM – Finite State Machine

ВСТУП

У сучасному світі цифрові технології займають ключове місце у повсякденному житті людей. Особливої популярності набувають мобільні месенджери, серед яких Telegram вирізняється своєю відкритістю, гнучкістю та широкими можливостями для автоматизації завдяки ботам. Telegram-боти стали зручним інструментом для взаємодії з користувачами, автоматизації завдань та надання різноманітних сервісів.

Під час проходження переддипломної практики мною було розроблено Telegram-бота, який дозволяє здійснювати пошук фільмів через OMDb API, шукати відео на YouTube, картинки в Google Images, а також взаємодіяти зі штучним інтелектом на базі моделі GPT від OpenAI. Додатково реалізовано зручне меню, підтримку кількох станів через FSM та інтерактивний інтерфейс.

Метою роботи є розробка із використанням сучасних інструментів, мов програмування та API сервісів Telegram-бота, який поєднує функціонал мультимедійного пошуку з можливостями генеративного ШІ у єдиний інтерфейс.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1 Опис предметної сфери

Сфера, у межах якої реалізовано Telegram-бота, охоплює сучасні цифрові сервіси для пошуку інформації, мультимедійного контенту та взаємодії з штучним інтелектом. У сучасному інформаційному суспільстві користувачі щодня стикаються з великими обсягами даних: стрічками новин, сайтами з фільмами, відеоплатформами, пошуковими системами. Це створює потребу в зручних та інтегрованих інструментах, які дозволяють швидко отримувати бажану інформацію в одному місці. Одним із таких інструментів є Telegram-бот — програмна надбудова, яка взаємодіє з користувачем через інтерфейс месенджера Telegram.

Telegram як платформа надає широкі можливості для створення ботів, які можуть обробляти повідомлення, надавати відповіді, здійснювати пошук, інтегруватися з API сторонніх сервісів (наприклад, OMDb API, YouTube, Google Images) та навіть застосовувати штучний інтелект для генерації текстів або діалогів. Завдяки відкритому API, великій кількості бібліотек для Python, JavaScript, Node.js та інших мов програмування, а також активному співтовариству розробників, Telegram-боти широко застосовуються в електронній комерції, освіті, техпідтримці, медіа, розвагах, сервісах доставки та особистих цифрових помічниках.

Предметна область даного проєкту — це автоматизований інформаційний пошук за запитом користувача: пошук фільмів за назвою, посилань на трейлери, зображень, а також можливість звернення до генеративного ШІ для відповіді на текстові питання.Telegram-бот виконує функцію агрегатора: він дозволяє інтегрувати в одному інтерфейсі різні типи пошуку, які інакше користувач здійснював би окремо на різних платформах. Це значно економить час і покращує користувацький досвід.

У предметній області особливої уваги заслуговує використання сторонніх API:

– OMDb API - відкрита база даних про фільми з інформацією про назву, рік випуску, рейтинг IMDb, сюжет, постер тощо;

– YouTube - для пошуку трейлерів або відео за ключовими словами;

– Google Images -для пошуку зображень;

– OpenAI API -для реалізації функціоналу генеративного ШІ, що дозволяє надавати відповіді на текстові запитання користувачів.

Аналіз існуючих аналогічних Telegram-ботів

Для кращого розуміння предметної сфери та вибору власного підходу до реалізації, було досліджено існуючі боти-конкуренти.

1. @WhatMovieBot - бот для пошуку фільмів за назвою. Використовує OMDb API для отримання основної інформації про фільми: назва, рік, жанр, рейтинг IMDb, постер. Має англомовний інтерфейс і базовий функціонал, без інтеграції з ШІ.

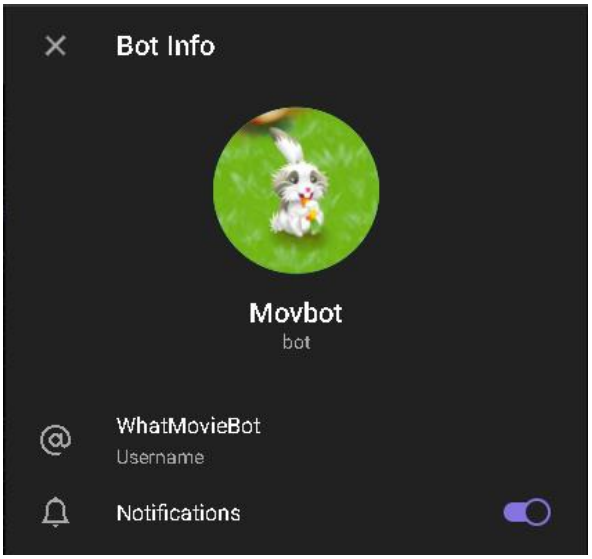

Рисунок 1.1 – Зображено @WhatMovieBot

2. @filmobot - українсько- та російськомовний бот, який також підключений до OMDb API. Надає дані про фільм, трейлер, іноді рейтинг, але не підтримує інтеграцію з іншими видами пошуку.

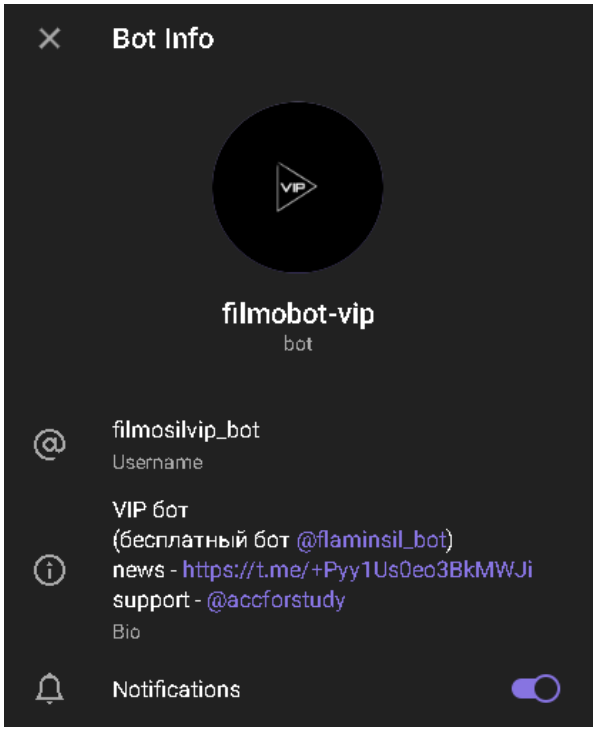

Рисунок 1.2 – Зображено @filmobot

3. @youtube_downloader_bot - бот для пошуку та завантаження відео з YouTube. Працює за принципом формування посилання на YouTube і дає можливість завантажувати відео. Не інтегрує пошук фільмів чи зображень.

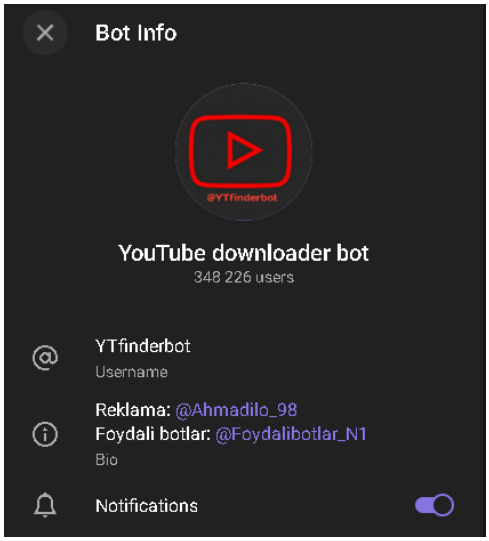

Рисунок 1.3 – Зображено @youtube_downloader_bot

4. @ChatGPTBot - бот, який відповідає на запитання користувачів за допомогою моделі ChatGPT. Підтримує генерацію текстів, але не об'єднує пошук фільмів чи іншого контенту.

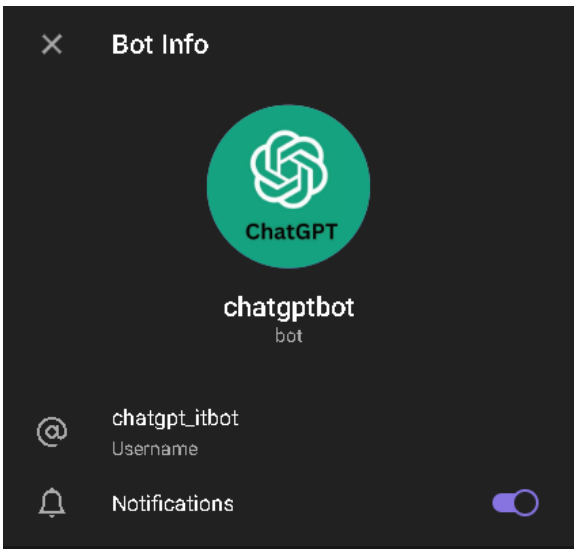

Рисунок 1.4 – Зображено @ChatGPTBot

5. @AIAnswerBot - бот для отримання відповідей на різні запитання, працює на основі ШІ (GPT). Підходить для загальної взаємодії, але не інтегрує мультимедійний пошук.

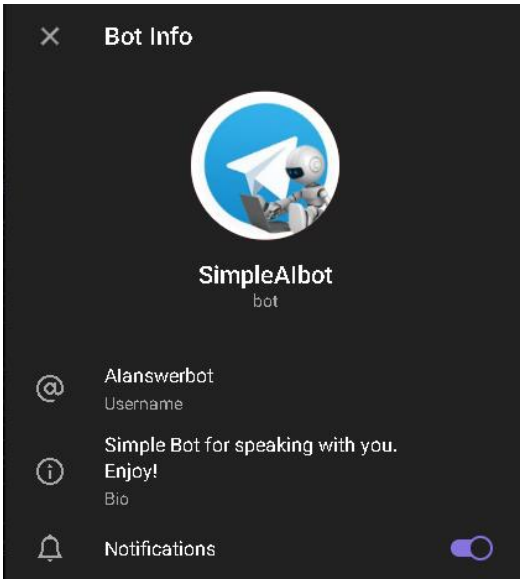

Рисунок 1.5 – Зображено @AIAnswerBot

На відміну від вищезгаданих рішень, мій Telegram-бот об'єднує пошук фільмів, трейлерів, зображень та інтеграцію з ШІ в єдиному інтерфейсі, що дозволяє користувачу заощаджувати час та отримувати комплексну інформацію в одному місці.

Принципи пошуку та особливості реалізації

Усі пошукові функції в Telegram-ботах базуються на використанні API-запитів та формуванні URL-посилань:

– для пошуку фільмів реалізовано запити до OMDb APIз отриманням JSON-даних та подальшою обробкою їх для виведення користувачеві;

– для пошуку трейлерів використовується динамічне формування посилань на YouTube [12];

– для пошуку зображень застосовується Google Images через відповідне посилання [13];

– для генеративного ШІ застосовується інтеграція з OpenAI APIчерез протокол HTTPS.

Особливістю Telegram-ботів є їхня асинхронна архітектура: бот реагує на повідомлення користувача в режимі реального часу, швидко опрацьовує запити та повертає результат. Це досягається за рахунок використання сучасних бібліотек на кшталт Aiogram для Python.

Історичний екскурс розвитку інформаційного пошуку.

Історія інформаційного пошуку бере свій початок у 1960–1970-х роках, коли перші бібліотечні каталоги та бази даних дозволяли знаходити потрібні книги або документи. З розвитком Інтернету у 1990-х роках виникли перші пошукові системи (Archie, Veronica, AltaVista), які сканували веб-ресурси. У 1998 році з’явився Google, що завдяки алгоритму PageRank значно покращив релевантність пошукових результатів. У 2000-х роках відбувся перехід до мультимедійного пошуку: з’явилися спеціалізовані сервіси для пошуку зображень, відео та музики.

Telegram-боти як інтерфейс для пошуку стали популярними після відкриття Telegram Bot API у 2015 році, що дозволило розробникам створювати інструменти для інтеграції з різними джерелами даних та API.

Таким чином, предметна сфера охоплює кілька напрямів: інформаційні системи, програмне забезпечення, інтеграцію сторонніх сервісів через API, розробку інтерфейсів для взаємодії з користувачами, машинне навчання (в частині ШІ), а також аспекти зручності користування. Telegram-боти є перспективним напрямом розвитку мікросервісних рішень, які не потребують встановлення окремих програм і працюють безпосередньо в мобільному або десктопному клієнті Telegram.

1.2 Огляд та аналіз наявних аналогів і публікацій

Розробка Telegram-ботів для пошуку інформації - популярний напрямок у сучасному програмуванні. Це обумовлено як технічною доступністю API Telegram, так і високим попитом серед користувачів на швидкий доступ до різноманітних сервісів без використання браузера чи встановлення додаткових застосунків. У

цьому контексті було проведено огляд аналогічних рішень, що вже реалізовані в Telegram або описані в науково-практичній літературі.

Одним із найвідоміших прикладів є боти для пошуку фільмів, такі як @MovieBot, @filmobot або @WhatMovieBot. Вони дозволяють здійснювати пошук фільму за назвою, переглядати рейтинг, постер, короткий опис. Проте функціональність таких ботів обмежується лише пошуком фільму і часто не включає інші корисні елементи, наприклад, посилання на трейлери, альтернативні платформи перегляду, або рекомендації. Деякі з них мають англомовний інтерфейс, що ускладнює використання україномовними користувачами.

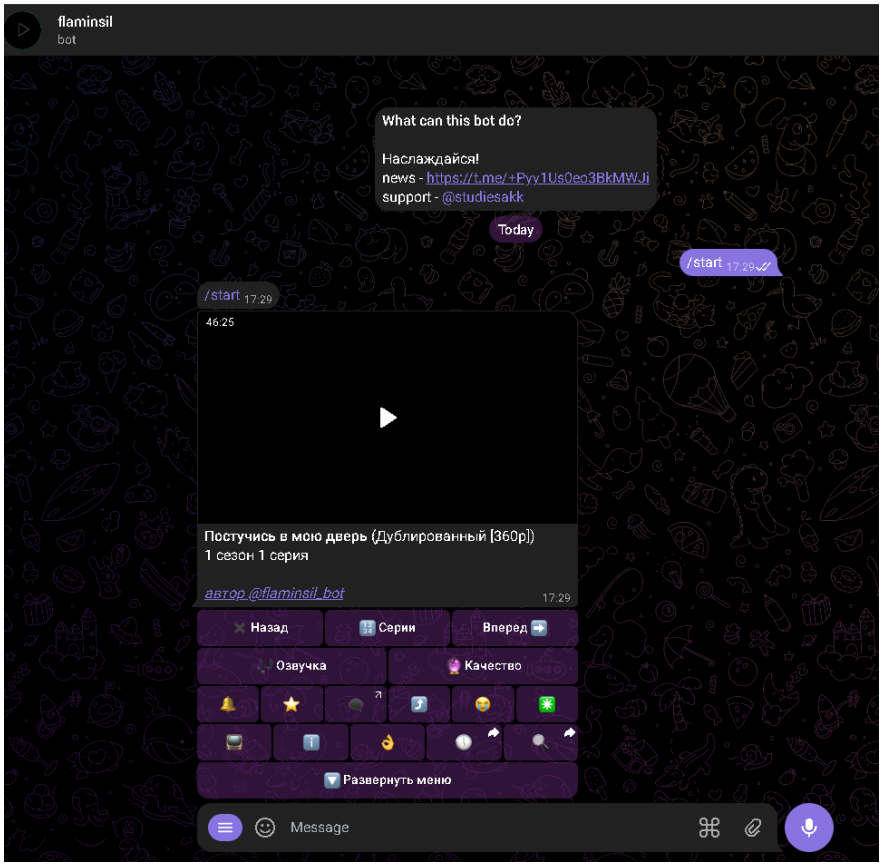

Рисунок 1.6 - Приклад роботи @filmobot

У напрямку мультимедійного пошуку також існують боти, які здійснюють пошук відео на YouTube, наприклад @youtube_downloader_bot. Хоча вони можуть знаходити відео та іноді навіть дозволяють завантаження, але не завжди

пропонують зручний або інтуїтивний інтерфейс. Часто користувач змушений вручну вводити команди або ключові слова, а відповіді можуть містити зайву інформацію або бути нечіткими.

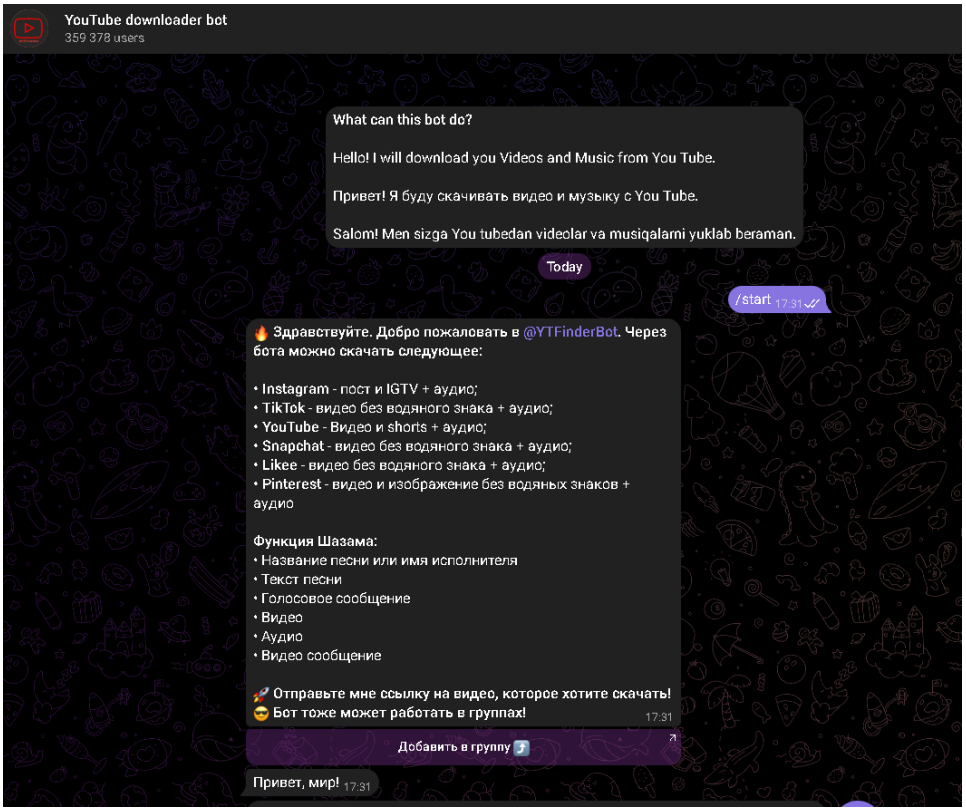

Рисунок 1.7 – Приклад роботи @youtube_downloader_bot

Ще один тип ботів - це чат-боти зі штучним інтелектом, як-от @ChatGPTBot, @AIAnswerBot та подібні у таб 1.1 порівняння. Вони надають користувачам змогу ставити запитання і отримувати відповіді на основі моделей OpenAI. Проте у більшості випадків ці боти мають обмеження у функціональності — наприклад, відсутність мультимедійних функцій чи фільтрації за типом запиту (фільм, відео, зображення). Крім того, багато з них працюють виключно англійською мовою або використовують платні API з обмеженнями для вільного користування.

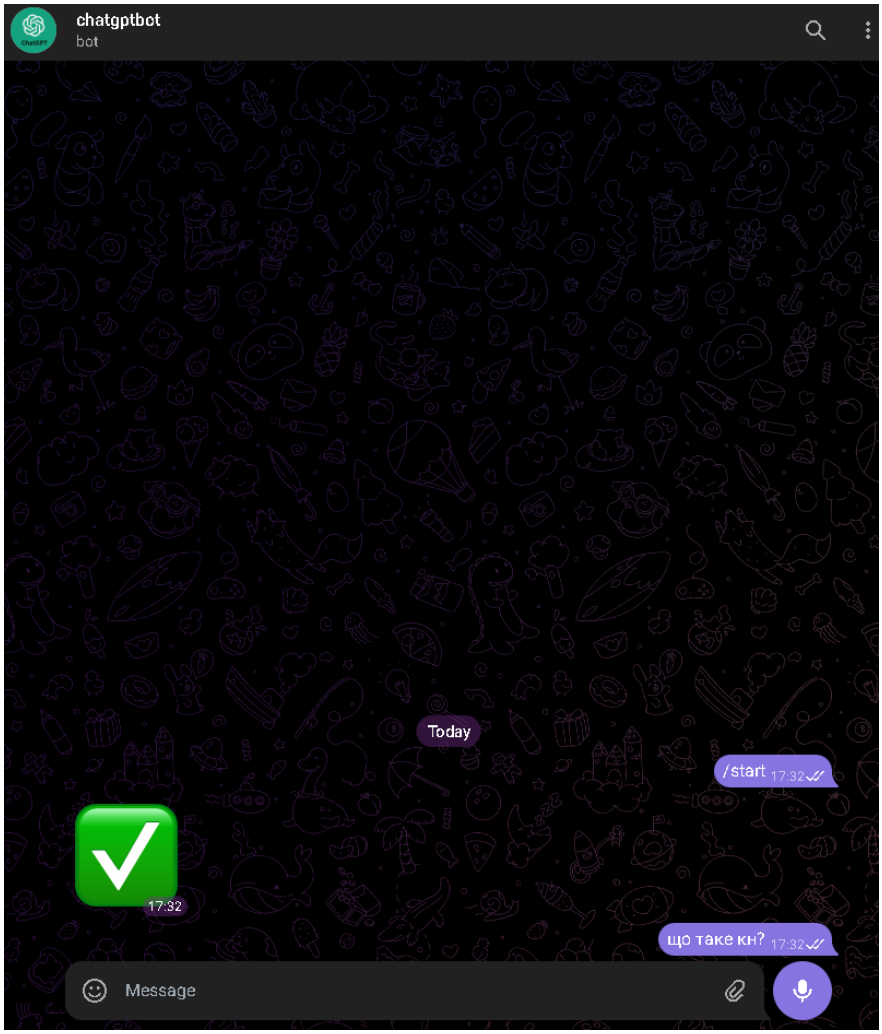

Рисунок 1.8 – Приклад роботи @ChatGPTBot

Щодо наукових публікацій, у фаховій літературі розглядаються аспекти архітектури Telegram-ботів, безпеки інтеграції з API, методи обробки запитів користувачів, оптимізації інтерфейсу взаємодії. Наприклад, у працях, присвячених використанню Telegram-ботів в освіті, медичній сфері чи для комерційних цілей, наголошується на важливості адаптації інтерфейсу до конкретної задачі користувача та ефективної обробки запитів у режимі реального часу.

Проведений аналіз дозволяє зробити висновок, що хоча окремі функції, реалізовані в даному проєкті (пошук фільму, YouTube, Google Images, ШІ), зустрічаються в існуючих рішеннях, інтеграція всіх цих можливостей в одному Telegram-боті, з урахуванням локалізації українською мовою, зручного меню та

багатофункціонального підходу — є унікальною перевагою та відображає актуальні потреби сучасного користувача. Деякі з відомих ботів навіть не працють.

Таким чином, розробка власного Telegram-бота, який поєднує функціонал мультимедійного пошуку з можливостями генеративного ІІ, вирішує одразу кілька задач: спрощення доступу до контенту, автоматизація пошукових дій, локалізація під українську аудиторію та підвищення зручності користування.

Таблиця 1.1 – Порівняння відомих ботів

	Переваги:	Недоліки:
@MovieBot	Швидкий пошук фільмів за назвою. Дає базову інформацію: рейтинг IMDb, рік, опис, постер. Легкий у використанні, мінімалістичний інтерфейс.	Працює лише англійською мовою. Не підтримує трейлери або посилання на перегляд. Немає інтеграції з українськими джерелами.
@filmobot	Орієнтований на українську та російську аудиторію. Пошук фільмів з описами, оцінками та іноді трейлерами. Зрозумілий інтерфейс.	Результати іноді неактуальні або обмежені. Працює повільніше, ніж деякі інші боти. Часто не надає посилань на перегляд.

Продовження таблиці 1.1

@WhatMovieBot	Підбір фільму за настроєм або жанром. Рекомендує випадкові фільми на основі запиту. Можна ввести запит природною мовою.	Не завжди знаходить конкретний фільм за назвою. Відсутня українська мова. Немає трейлерів або візуального контенту.
@youtube_downloader_bot	Дає змогу завантажувати відео з YouTube прямо у Telegram. Працює швидко, підтримує різні формати (mp4, mp3). Простий інтерфейс — достатньо вставити посилання.	Не завжди стабільна робота (через обмеження з боку YouTube). Може бути тимчасово заблокований. Немає додаткових функцій (пошук, фільтрація тощо).
@ChatGPTBot	Працює на базі GPT-моделі від OpenAI. Може відповідати на будь-які запити: текстові, технічні, побутові. Підтримує декілька мов, у тому числі українську.	Обмеження на кількість запитів у безкоштовній версії. Інколи дає неточні або вигадані відповіді. Потребує стабільного підключення до API OpenAI.

Кінець таблиці 1.1

@AIAnswerBot	Орієнтований на короткі запитання-відповіді. Використовує GPT або інші моделі для генерації відповідей. Швидке реагування на запити.	Часто не має контексту або історії розмови. Може давати узагальнені відповіді. Не підтримує повноцінний діалоговий режим, як ChatGPT.

У цій таблиці було порівняно відомих ботів, вказані переваги та недоліки, кожен має сам обрати для себе який буде зручніший для нього. Було узято декілька функцій ботів та спроба об’єднати їх у один бот.

1.3 Постанова задачі

Актуальність теми. У сучасному цифровому суспільстві зростає потреба у швидкому, зручному та централізованому доступі до інформації. Зокрема, користувачі щоденно шукають фільми, відео, зображення, відповіді на запитання або консультації. Традиційно для цього необхідно використовувати кілька окремих вебсайтів або застосунків. Такий підхід не завжди зручний, особливо на мобільних пристроях. Telegram, як популярний месенджер з відкритим API, надає можливість створювати ботів, що інтегрують декілька джерел інформації в одному інтерфейсі.

З огляду на активне зростання користувачів Telegram в Україні та світі, а також на розвиток штучного інтелекту й API для відкритих баз даних (таких як OMDb, YouTube, Google Images), створення універсального Telegram-бота стає актуальним інструментом не лише для персонального використання, а й для бізнесу, освіти та розваг. Актуальність підкріплюється й тим, що більшість ботів не локалізовані українською мовою, не мають повного функціоналу або працюють нестабільно.

Метою даної роботи є розробка Telegram-бота, який надає користувачу можливість здійснювати пошук фільмів, відео, зображень, а також ставити довільні запитання до системи штучного інтелекту, використовуючи єдиний інтерфейс.

Об'єктом роботи є процеси автоматизації пошуку інформації та реалізація користувацької взаємодії з допомогою чат-ботів на платформі Telegram.

Предметом роботи виступають технології побудови Telegram-ботів із використанням API сторонніх сервісів (OMDb API, YouTube, Google Images, OpenAI GPT) і методи організації діалогової взаємодії між користувачем та ботом на основі Finite State Machine (FSM).

Для реалізації поставленої мети необхідно виконати наступні завдання:

– провести аналіз існуючих Telegram-ботів, що виконують схожі функції, та виявити їхні переваги й недоліки;

– обрати відповідні технології розробки (мова програмування, фреймворки, бібліотеки, API);

– розробити архітектуру бота з підтримкою кількох функціональних модулів: пошук фільму, YouTube, зображень, відповідь ІІ;

– реалізувати інтерфейс взаємодії з користувачем у вигляді кнопок меню та відповідей у текстовому й мультимедійному форматі;

– застосувати Finite State Machine для керування контекстом діалогу та переходами між станами;

– інтегрувати Telegram-бота з OMDb API, Google Images, YouTube, а також OpenAI GPT-3.5;

– здійснити локалізацію інтерфейсу українською мовою;

– провести тестування функціоналу, оцінити якість результатів пошуку, зручність використання та стабільність роботи;

– підготувати документацію до системи та опис коду, що дозволить легко підтримувати і розширювати проект у майбутньому.

Висновки до розділу 1

У першому розділі було визначено актуальність теми створення Telegram-бота як інструмента автоматизації та покращення взаємодії користувачів із цифровими сервісами. Розглянуто мету та завдання , сформульовано основну ідею проєкту. Також обґрунтовано вибір напрямку розробки, що відповідає сучасним вимогам до зручності, доступності та ефективності інформаційного обміну. Усі ці фактори підтверджують доцільність реалізації даного проєкту.

2 МОДЕЛІ ТА МЕТОДИ ДЛЯ РОЗРОБКИ ЧАТ-БОТУ

2.1 Методи та засоби для вирішення поставленої задачі

Для реалізації функціоналу Telegram-бота було використано низку методів, які забезпечують взаємодію користувача з різними джерелами інформації в межах одного інтерфейсу. Всі методи ґрунтуються на принципах асинхронного програмування, обробки станів користувача та використання зовнішніх API. Нижче наведено опис основних методів та засобів, що були реалізовані.

1. Метод пошуку фільмів через OMDb API.Цей метод дозволяє користувачу здійснювати пошук фільму за його назвою. OMDb API **(Open Movie Database)** — це безкоштовний веб-сервіс, який надає інформацію про фільми, серіали та акторів на основі даних IMDb.формує HTTP-запит до OMDb API. Розглянемо етапи роботи методу:

– отримує JSON-відповідь з основною інформацією про фільм;

– витягує з відповіді назву, рік випуску, рейтинг IMDb, опис сюжету, посилання на IMDb та постер;

– додатково формує пошукові посилання на трейлер у YouTube та сайт HDRezka;

– повертає користувачеві оформлену відповідь у вигляді тексту або зображення (постера).

Метод реалізований з використанням бібліотеки aiohttp для асинхронних HTTP-запитів.

2. Метод пошуку відео на YouTube.Для пошуку відео на YouTube використовується простий підхід без офіційного API (щоб уникнути складних обмежень та необхідності авторизації). YouTube надає можливість формувати URL-посилання на сторінку з результатами пошуку. Розглянемо етапи роботи методу:

– бот отримує текстовий запит від користувача;

– формує URL-посилання на результати пошуку YouTube;

– повертає користувачу гіперпосилання на сторінку з результатами.

– це простий, але ефективний метод швидкого доступу до відео без використання YouTube API.

3. Метод пошуку зображень через Google Images.Для отримання результатів пошуку зображень використовується Google Image Search. Розглянемо етапи роботи методу:

– формує URL для Google Image Search;

– надсилає посилання користувачеві у вигляді повідомлення з гіперпосиланням;

– також реалізовано перевірку на наявність введення та очищення стану користувача після відповіді.

4. Метод взаємодії зі штучним інтелектом (OpenAI GPT).Для реалізації інтелектуального функціоналу бот використовує модель OpenAI GPT. Розглянемо етапи роботи методу:

– отримує текстове повідомлення користувача;

– формує запит до OpenAI GPT з зазначенням моделі, контексту і параметрів генерації;

– отримує відповідь від моделі у вигляді тексту;

– надсилає відповідь користувачеві;

– цей метод дозволяє створити ілюзію діалогу з «розумним» ботом, що може пояснювати, рекомендувати, відповідати на загальні питання.

5. Метод організації діалогу за допомогою Finite State Machine (FSM).Для керування діалогом бот використовує FSM – скінченний автомат станів. Це дозволяє:

– відслідковувати, на якому етапі взаємодії перебуває користувач (наприклад, очікується введення назви фільму або зображення);

– розділити логіку обробки повідомлень за контекстом;

– уникати плутанини при обробці кількох запитів підряд;

– FSM реалізовано через модуль aiogram.fsm, який дозволяє створювати класи станів та працювати з ними в асинхронному режимі.

6. Метод формування клавіатури меню.Для зручності користувача реалізовано метод формування клавіатури з кнопками:

– основне меню містить кнопки для кожної функції;

– при натисканні на кнопку, користувач потрапляє в потрібний сценарій взаємодії;

– використано тип ReplyKeyboardMarkup, який підтримує змінну кількість рядків.

Застосування наведених методів дозволяє створити зручний, інтерактивний та багатофункціональний Telegram-бот, який не лише виконує пошук інформації з різних джерел (OMDb API, YouTube, Google Images), а й забезпечує інтелектуальну взаємодію з користувачем за допомогою GPT-3.5. Використання асинхронного програмування та Finite State Machine забезпечує ефективне керування діалогом та запитами користувачів, що робить проєкт масштабованим і легко доповнюваним новими функціями. Таким чином, обрані методи і засоби безпосередньо сприяють досягненню мети проєкту — створенню сучасного, зручного у використанні Telegram-бота для пошуку інформації.

2.2 Технології розробки системи

Для розробки Telegram-бота було обрано цілий набір технологій, які дозволяють реалізувати всі необхідні функції з високою ефективністю та надійністю. У процесі розробки використовувалися як готові бібліотеки, так і інтерфейси зовнішніх сервісів, що дозволяють інтегрувати додаткові можливості для користувачів.

1. Python. Основною мовою програмування для створення бота є Python. Python є універсальною мовою, яка надає безліч бібліотек для розробки серверних застосунків, що робить її ідеальною для створення чат-ботів.

Переваги Python:

– легка інтеграція з різними API (наприклад, OMDb API, OpenAI GPT);

– розвинене середовище для розробки і тестування (наприклад, використання фреймворку pytest для юніт-тестування);

– велика кількість бібліотек для роботи з асинхронним програмуванням, обробки HTTP-запитів і зручною роботою з Telegram API.

2. Aiogram. Для інтеграції з Telegram API використано бібліотеку **aiogram**. Це сучасний асинхронний фреймворк для створення ботів на Python, який забезпечує ефективну обробку запитів у реальному часі. Основні переваги aiogram:

– підтримка асинхронних операцій через asyncio, що дозволяє працювати з великими обсягами одночасних запитів;

– простий синтаксис для обробки команд і повідомлень користувачів;

– підтримка розширених функцій, таких як роботи з клавіатурами, кнопками, Inline-клавіатурами та інші;

– підтримка FSM (Finite State Machine) для організації різних станів в діалозі з користувачем.

3. Aiohttp. Для здійснення асинхронних HTTP-запитів до зовнішніх API було використано бібліотеку aiohttp. Вона дозволяє ефективно працювати з веб-ресурсами, що надають дані у форматі JSON, що є необхідним для інтеграції з такими сервісами, як OMDb API або OpenAI API. Переваги aiohttp:

– асинхронна обробка запитів, що зменшує час очікування для користувача;

– легка інтеграція з іншими асинхронними бібліотеками в Python, такими як asyncio;

– підтримка роботи з REST API та можливість отримання та обробки JSON-відповідей.

4. OpenAI GPT-3.5. Однією з ключових технологій, яка дозволяє боту взаємодіяти з користувачем за допомогою штучного інтелекту, є API OpenAI GPT. Це потужна модель на основі GPT-3.5, яка використовується для генерації текстових відповідей на запити користувача. Модель дозволяє боту здійснювати

діалог з користувачем, надаючи відповіді на загальні запитання або виконуючи прості логічні задачі. Переваги OpenAI GPT-3.5:

– висока якість текстових відповідей, що зберігають природність і інтелектуальність;

– можливість кастомізації під конкретні потреби;

– широке застосування у розв'язанні складних лінгвістичних задач.

5. Файл .env та бібліотека python-dotenv. Для зберігання конфіденційної інформації, такої як токен бота або API-ключі, було використано файл .env. Це дозволяє уникнути жорсткого кодування таких даних безпосередньо в програмі, що покращує безпеку та зручність в роботі з різними середовищами (локальним та продакшн).

Бібліотека python-dotenv дозволяє легко завантажувати змінні середовища з .env файлу в програму, що забезпечує захист даних та їх безпечне використання.

6. SQLite. Для зберігання деякої інформації (наприклад, налаштувань користувачів або історії запитів) можна використовувати SQLite – легковаговий реляційний движок баз даних. Ця технологія забезпечує просте управління даними без необхідності у складних налаштуваннях серверів баз даних. Переваги SQLite:

– легка інтеграція з Python через бібліотеку sqlite3;

– швидка і ефективна робота з невеликими обсягами даних;

– без необхідності налаштування окремого серверного рішення для бази даних.

7. Docker (опційно). Для спрощення процесу розгортання і тестування бота можна використовувати технологію контейнеризації Docker. Використання Docker дозволяє створювати контейнер, в якому буде працювати весь бот разом з усіма залежностями, що значно полегшує розгортання на сервері та тестування в різних середовищах. Переваги Docker:

– ізольованість середовища для розгортання застосунків;

– легкість масштабування і оновлення застосунків;

– підвищення стабільності та передбачуваності роботи в різних середовищах.

8. Git для контролю версій. Для ведення контролю версій та співпраці при розробці проекту використовувалася система контролю версій Git. Git дозволяє зберігати історію змін у проекті, координувати роботу між кількома розробниками та швидко відновлювати попередні версії коду. Переваги Git:

– легка інтеграція з GitHub або GitLab для спільної роботи над проектом;

– можливість відновлення попередніх версій та аналізу змін;

– підтримка гілок і злиття для паралельної роботи над різними частинами проекту.

9. Google Images. Для пошуку зображень використовується **Google Images**, де також формується URL-адреса запиту на основі ключових слів.

10. YouTube (Search URL)**.** Для реалізації пошуку відео використовується формування URL-запиту до сторінки результатів пошуку на YouTube. Це дозволяє обходитися без офіційного API (YouTube Data API) та ключів авторизації.

2.3 Структура коду проекту

У цій частині описується структура програмного коду Telegram-бота, розробленого в межах кваліфікаційної роботи. Код реалізовано мовою програмування Python із використанням асинхронного фреймворку Aiogram 3.x, що дозволяє ефективно обробляти запити користувачів у режимі реального часу.

Уся логіка бота розділена на чотири ключові файли.

1. bot.py — головний файл. Це точка входу до програми. Саме з цього файлу починається робота бота.

Основні функції:

– завантаження токенів з .env файлу;

– ініціалізація екземпляра бота (Bot) і диспетчера (Dispatcher);

– реєстрація хендлерів;

– встановлення основних команд /start і /menu;

– запуск бота через start_polling().

Цей файл відповідає за інтеграцію всіх частин проєкту в єдину систему та є необхідним для запуску Telegram-бота.

2. handlers.py — логіка обробки команд і повідомлень. Цей файл містить основну логіку взаємодії бота з користувачем. Він включає:

а) обробку команд /start, /menu, яка викликає головне меню;

б) обробку кнопок головного меню:

1) пошук фільму (з OMDb API);
2) пошук YouTube (перенаправлення на результат пошуку);
3) пошук зображень (Google Images);
4) ШІ (відповіді на запитання через OpenAI GPT);
5) зв’язатися з нами (контактна інформація).

в) реалізацію FSM-сценаріїв для кожного типу пошуку (наприклад, очікування введення назви фільму).

Підключення API-ключів і взаємодію з зовнішніми сервісами через aiohttp.

Файл також містить обробку помилок та некоректних запитів, що забезпечує стійку та стабільну роботу бота.

3. keyboards.py — формування клавіатури. Цей файл відповідає за створення інтерфейсу головного меню у вигляді кнопкової клавіатури Telegram. Саме після введення команди /start або натиску кнопки "Головне меню", користувач бачить набір доступних функцій бота.

Кнопки меню:

– пошук фільму;
– пошук YouTube;
– пошук картинки;
– ШІ;
– зв’язатися з нами.

Файл забезпечує зручність навігації та інтуїтивний інтерфейс взаємодії.

4. .env — файл змінних середовища. Цей файл містить конфіденційні API-ключі та токени, які підключаються через бібліотеку python-dotenv. Завдяки .env:

– ключі не зберігаються безпосередньо в коді;

– забезпечується безпека доступу до сервісів;

– легко змінювати токени без редагування основних файлів.

2.4 Сучасні тренди у створенні Telegram-ботів

Упродовж останнього десятиліття розробка Telegram-ботів стала популярним напрямком у сфері IT, оскільки Telegram надає зручний інтерфейс та багатий функціонал через відкритий API. Сучасні тренди у створенні Telegram-ботів охоплюють кілька ключових аспектів, серед яких можна виділити використання машинного навчання (ML), обробки природної мови (NLP), інтеграцію мультимедійних сервісів та застосування хмарних технологій.

Використання ML та NLP. У сучасних ботах дедалі частіше впроваджують моделі машинного навчання для реалізації персоналізованих рекомендацій, генерації діалогів та автоматичної класифікації запитів користувачів. Наприклад, інтеграція GPT-моделей (OpenAI GPT-3.5, GPT-4) дозволяє ботам вести змістовні бесіди, відповідати на питання та пояснювати складні теми. Крім генеративних моделей, застосовуються і класифікатори на основі методів машинного навчання (SVM, Random Forest, BERT), які допомагають аналізувати наміри користувача (intent classification), наприклад, розпізнавати запит: «Знайти фільм», «Показати трейлер», «Завантажити відео».

Інтеграція мультимедійних сервісів.

Сучасні боти дедалі частіше стають мультимодальними: вони дозволяють шукати текстову, візуальну, аудіо- та відеоінформацію. Це досягається завдяки інтеграції з API популярних платформ: OMDb API для фільмів, Spotify API для музики, YouTube API для відео, Google Custom Search API для зображень тощо. Таким чином, користувач отримує універсальний інструмент пошуку в одному чаті.

Хмарні технології та мікросервісна архітектура.Для масштабування та забезпечення надійності Telegram-ботів сучасні розробники активно використовують хмарні сервіси (AWS Lambda, Google Cloud Functions, Azure Functions). Такі підходи дозволяють:

– швидко масштабувати навантаження;

– автоматично балансувати навантаження між сервісами;

– забезпечувати високу доступність бота навіть при різких стрибках запитів.

– інтерактивність і багатоканальність.

Сучасні користувачі очікують зручних інтерфейсів для взаємодії з ботами: кнопкові меню, інлайн-клавіатури, швидкі відповіді та віджети. Також розширюються можливості інтеграції із зовнішніми системами через вебхуки, що дозволяє отримувати нотифікації та обробляти події в реальному часі.

Аналітика та A/B-тестування.

Ще один сучасний тренд — це використання аналітики для відстеження популярності запитів користувачів, кількості звернень та ефективності окремих функцій. Такі інструменти, як Google Analytics для Telegram-ботів або власні системи логування, дозволяють розробникам постійно вдосконалювати інтерфейс бота та оптимізувати відповіді.

Таким чином, сучасні Telegram-боти дедалі частіше перетворюються з простих чат-ботів на комплексні інтелектуальні системи з підтримкою мультимедіа, штучного інтелекту та хмарної архітектури. Ці тренди відкривають нові можливості для реалізації проектів, подібних до даного Telegram-бота.

2.5 Безпека чат-ботів

Зі зростанням кількості чат-ботів у Telegram та їхнього використання для обробки конфіденційної інформації питання безпеки набуло особливої актуальності. Безпека Telegram-ботів охоплює кілька ключових аспектів, серед яких авторизація користувачів, захист API-ключів, обробка помилок, боротьба зі спамом та зловмисними атаками.

Авторизація користувачів. Telegram пропонує кілька методів ідентифікації користувачів: унікальний ID користувача, ім’я користувача, а також Telegram Login Widget, який дозволяє інтегрувати Telegram-бота у вебсайт і автентифікувати користувачів через OAuth2. Це дозволяє розробнику перевіряти особу користувача та надавати доступ до персоналізованих функцій (наприклад, історії пошуку або рекомендацій).

Захист API-ключів. Ключі доступу до зовнішніх сервісів (OMDb API, OpenAI API, YouTube API) є вразливими до несанкціонованого доступу. Щоб уникнути компрометації ключів, у проекті використовуються змінні середовища та бібліотека dotenv, яка дозволяє приховати ключі від відкритого коду (наприклад, ключі зберігаються у файлі .env). Це важливо, оскільки при витоку API-ключів зловмисники можуть перевищити ліміти запитів або навіть заблокувати сервіс для легального користувача.

Захист від спаму та ботів.

Telegram надає кілька вбудованих механізмів для боротьби зі спамом:

– обмеження частоти запитів (rate-limiting) за допомогою внутрішніх засобів фреймворку (наприклад, ThrottlingMiddleware в Aiogram);

– валідація вхідних даних (перевірка довжини повідомлень, форматів запитів);

– інтеграція CAPTCHA або верифікаційних кодів для уникнення атак ботів.

Обробка помилок та виняткових ситуацій. У роботі з Telegram API, а також з OMDb API та OpenAI API важливо передбачати обробку помилок: недоступність API, перевищення лімітів, неправильний формат відповіді. Для цього застосовуються блоки try-except, логування помилок, а також інформування користувача про помилку дружніми повідомленнями.

Конфіденційність даних. Telegram-боти за замовчуванням не зберігають особисті дані користувачів (паролі, історію запитів), але для розширеного функціоналу (наприклад, збереження налаштувань або історії) важливо

враховувати правила конфіденційності та кібербезпеки (GDPR, українське законодавство).

Таким чином, безпека Telegram-бота є багатогранною задачею, яка потребує комплексного підходу на рівні коду та архітектури проєкту. Дотримання цих принципів забезпечує стабільну та надійну роботу бота.

Висновки до розділу 2

У другому розділі проведено аналіз предметної області та розглянуто існуючі рішення – аналоги Telegram-ботів, які надають схожі функції. Було виявлено сильні та слабкі сторони аналогічних проєктів, що дозволило сформулювати вимоги до майбутньої системи. На основі аналізу визначено ключові функціональні можливості, які мають бути реалізовані в розробленому боті, а також обґрунтовано вибір технологій. Цей розділ заклав фундамент для ефективного планування та реалізації програмного продукту.

Також було створено порівняльну таблицю відомих ботів з пошуку мультимедійного контенту, в результаті чого, кожен може обрати той бот, який йому більш подобається. Особливу увагу приділено сучасним технологічним трендам, серед яких:

– впровадження моделей машинного навчання (ML) та обробки природної мови (NLP) для реалізації діалогових інтерфейсів та рекомендацій;

– інтеграція мультимедійних сервісів для розширення можливостей пошуку (відео, зображення, музика);

– використання хмарних сервісів для масштабування та підвищення доступності;

– активне використання мікросервісної архітектури та асинхронної обробки запитів.

Також було визначено важливі аспекти безпеки під час реалізації Telegram-бота, такі як авторизація користувачів, захист API-ключів та обробка помилок.

Вказані аспекти дозволяють забезпечити стабільну та безпечну роботу системи, що є ключовим для сучасних програмних продуктів

Отже, проведений аналіз підтвердив доцільність використання Telegram-бота як платформи для інтеграції різних типів мультимедійного пошуку та технологій штучного інтелекту, що відповідає сучасним вимогам користувачів і тенденціям у сфері IT.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Опис вхідних даних та структури системи

У процесі розробки Telegram-бота для автоматизації пошуку мультимедійного контенту було враховано необхідність ефективного управління вхідними даними та побудови чіткої архітектурної структури системи. Одним з основних завдань є забезпечення швидкої та надійної взаємодії користувача з ботом при мінімальних затратах часу на обробку запиту.

Вхідні дані в системі — це перш за все текстові повідомлення користувача у чаті Telegram. У більшості випадків це назва фільму, яку користувач бажає знайти. Враховуючи варіативність написання назв (мовні відмінності, орфографічні помилки, неповні назви), система повинна бути здатною адаптивно реагувати на різні форми запиту. Для цього використовується попередня обробка введених даних: видалення зайвих пробілів, приведення до нижнього регістру, а також усунення зайвих символів (наприклад, лапок, дужок тощо).

З точки зору архітектури, Telegram-бот побудований на основі бібліотеки Aiogram версії 3.x, яка забезпечує асинхронну обробку повідомлень. Такий підхід дозволяє оптимізувати навантаження на сервер, забезпечити стабільну роботу навіть при великій кількості одночасних користувачів та уникнути затримок у відповіді. Кожна команда або запит користувача запускає відповідний хендлер, який містить логіку взаємодії з API та формування відповіді.

Система також інтегрована з зовнішнім API для отримання інформації про фільми. Найчастіше використовується OMDb API або TMDb API. Ці сервіси надають доступ до великої бази фільмів з такими характеристиками, як назва, рік випуску, рейтинг IMDb, опис сюжету, жанр, акторський склад, постер тощо. Запити до API формуються на основі вхідних даних користувача та обробляються відповідно до структури відповіді у форматі JSON.

Для забезпечення гнучкої роботи з відповіддю API використовується парсинг JSON-даних, після чого отримана інформація структурується у зручний для користувача формат. Наприклад, бот може надіслати повідомлення у такому вигляді: назва фільму, рік, короткий опис, посилання на постер та рейтинг IMDb.

Структура Telegram-бота складається з кількох основних модулів: модуль обробки команд користувача (handlers), модуль обробки запитів до API (services), модуль логування помилок, модуль конфігурації та безпеки (зокрема, зберігання API-ключів у змінних середовища).

Особливу увагу приділено обробці виняткових ситуацій. Наприклад, якщо API повертає помилку або фільм не знайдено, бот надсилає користувачу відповідне повідомлення: «На жаль, фільм не знайдено. Спробуйте змінити запит або ввести іншу назву». Це дозволяє покращити досвід взаємодії користувача із системою.

Уся система побудована з дотриманням принципів модульності та масштабованості. Такий підхід дозволяє легко змінювати логіку окремих компонентів, розширювати функціонал (наприклад, додавання трейлерів, можливість фільтрувати результати за жанром або роком) та інтегрувати нові джерела інформації.

Крім цього, важливою особливістю є те, що бот не зберігає персональні дані користувача, не веде історії запитів та не ідентифікує користувачів. Це дозволяє забезпечити конфіденційність та відповідати основним принципам захисту інформації, що особливо актуально у контексті сучасного законодавства з кібербезпеки.

Загалом, опис вхідних даних і структури системи свідчить про продуманий підхід до розробки, орієнтований на зручність користувача, гнучкість взаємодії з зовнішніми сервісами та ефективне управління інформаційними потоками. Саме завдяки правильному проектуванню цих аспектів Telegram-бот успішно виконує своє призначення – автоматизований пошук мультимедійного контенту на основі запиту користувача.

3.2 Проектування та моделювання програмного забезпечення

Процес візуалізації та моделювання системи є важливою частиною життєвого циклу розробки програмного забезпечення. Це дозволяє отримати повне уявлення про архітектуру майбутньої системи, її основні функціональні елементи, зв’язки між ними та передбачити потенційні проблеми ще до початку програмної реалізації. У випадку з Telegram-ботом для пошуку інформації про фільми така візуалізація була необхідною для формування чіткої логіки обробки користувацьких запитів та побудови взаємодії з зовнішніми ресурсами.

Використані підходи до моделювання

Для моделювання архітектури бота використовувались сучасні засоби та підходи - насамперед **UML-діаграми** (Unified Modeling Language), які дозволяють представити функціонування системи у вигляді графічних схем. Вони є зручними як для розробників, так і для інших учасників проєкту (керівників, тестувальників, замовників).

1. Діаграма варіантів використання (Use Case Diagram).

Цей тип діаграми дозволяє визначити, які саме функції виконує система для кінцевого користувача. У нашому випадку актором виступає **користувач Telegram**, який взаємодіє із ботом через чат.

Основні сценарії використання:

– надсилання запиту з назвою фільму;

– отримання результатів пошуку (назва, рік, опис, постер);

– обробка ситуації, коли фільм не знайдено;

– повідомлення про технічну помилку (наприклад, проблеми з API).

Ця діаграма дозволяє ще на етапі планування визначити всі точки взаємодії користувача з системою.

Рисунок 2.1 – Діаграма варіантів використання

2. Activity-діаграма відображає динаміку виконання процесів. Наприклад, типовий сценарій роботи виглядає так:

– користувач надсилає повідомлення боту;
– бот фіксує повідомлення і перевіряє, чи містить воно текстовий запит;
– якщо так – викликається модуль обробки запитів;
– створюється HTTP-запит до OMDb або TMDb API;
– відповідь аналізується;
– бот надсилає структуровану відповідь користувачу.

Ця діаграма дозволяє знайти «вузькі місця» у логіці обробки і передбачити, де можуть виникати затримки або помилки.

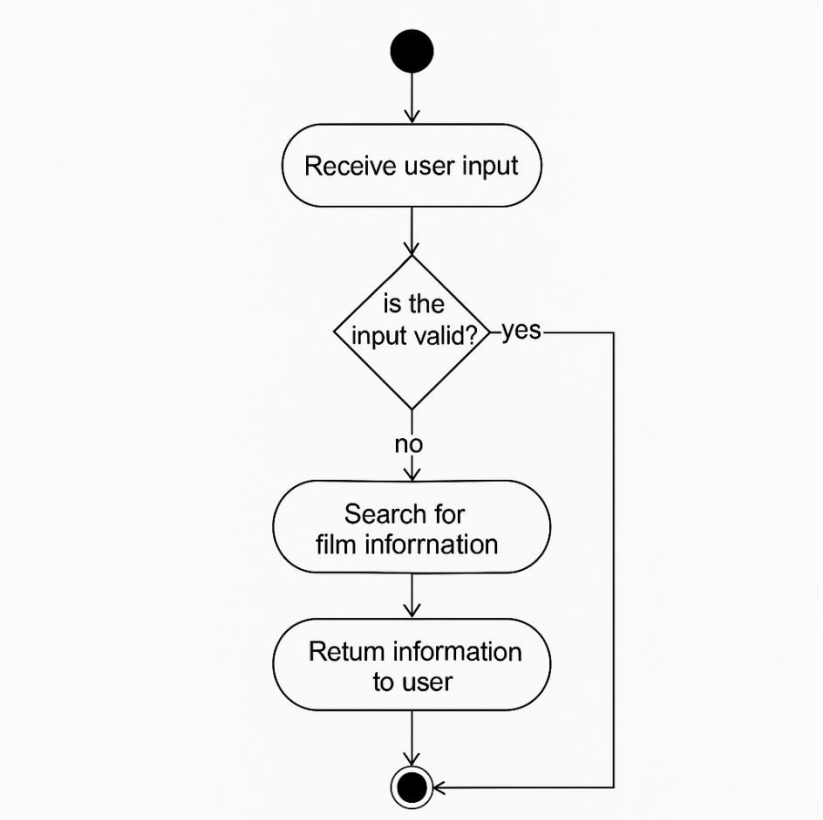

Рисунок 2.2 –Діаграма активностей (Activity Diagram)

3. Діаграма компонентів (Component Diagram).

Ця діаграма демонструє логічну структуру бота та показує, з яких частин він складається. Наприклад:

– модуль взаємодії з Telegram API (на основі бібліотеки aiogram);
– модуль обробки повідомлень (message handlers);
– модуль API-запитів (використання requests для запиту до OMDb API);
– модуль конфігурації (налаштування токенів, ключів, повідомлень);
– модуль логування/обробки помилок.

Такий розподіл дозволяє дотримуватись принципів SOLID та забезпечує розширюваність системи в майбутньому (наприклад, підключення додаткових джерел інформації).

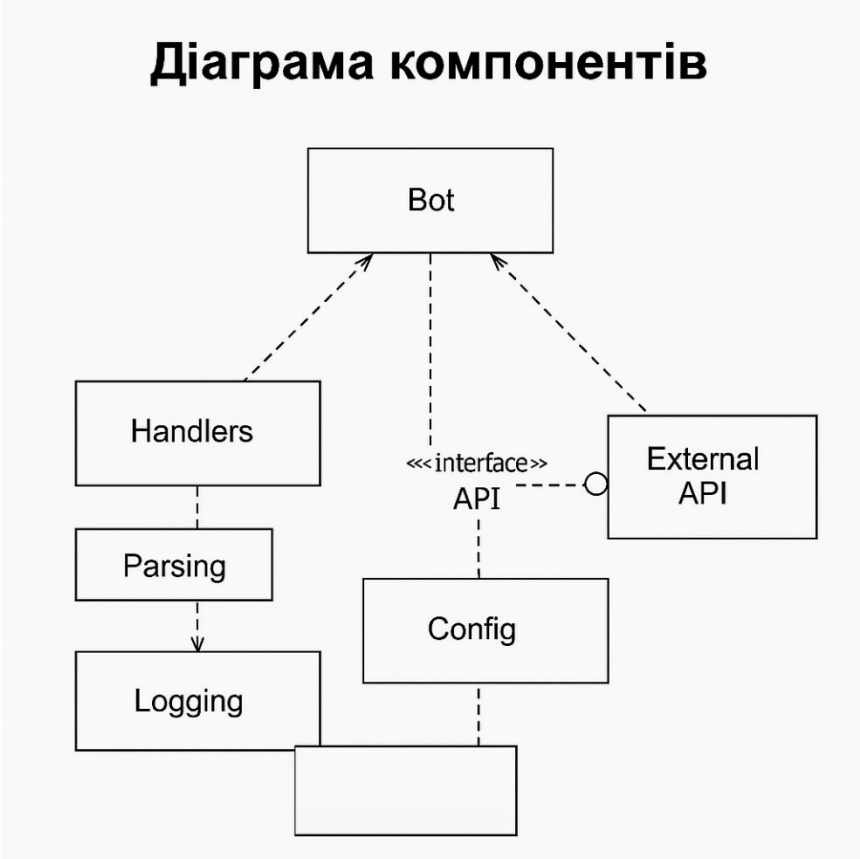

Рисунок 2.3 – Діаграма компонентів

4. Діаграма послідовності (Sequence Diagram**)** ілюструє порядок взаємодії об’єктів протягом обробки одного запиту користувача. Цей тип діаграми демонструє, як оброблюється введене повідомлення, як формується запит до API, і як результат відображається користувачу**.**

Під час моделювання враховувались принципи чистої архітектури: поділ логіки на окремі шари – представлення, бізнес-логіка та доступ до зовнішніх ресурсів. Це дозволяє полегшити тестування системи, забезпечити високу гнучкість коду та уникнути дублювання функцій.

Крім UML-діаграм, для візуалізації структури системи використовувалась блок-схема архітектури. Вона демонструє загальну структуру Telegram-бота: користувач → Telegram API → Python-бот → зовнішній API (OMDb/TMDB) → відповідь користувачу. Кожен з цих компонентів має свої функціональні обов’язки та взаємодіє з іншими через чітко визначені інтерфейси.

Важливим аспектом моделювання є також визначення можливих помилкових ситуацій і сценаріїв їх обробки. Наприклад, якщо зовнішній API тимчасово недоступний, система повинна інформувати користувача про це, а не просто «мовчати». Моделювання таких ситуацій допомагає передбачити потенційні проблеми на етапі проєктування та реалізувати відповідні механізми їх запобігання.

У рамках візуального моделювання також можна створити діаграму класів, яка показує структуру об’єктів, що використовуються в системі. Наприклад, клас Movie може мати поля: назва, рік, жанр, опис, рейтинг IMDb, посилання на постер тощо. Інші класи – APIClient, MessageHandler, UserRequest – взаємодіють між собою через методи, які реалізують основну логіку обробки запитів.

Таким чином, візуалізація і моделювання структури Telegram-бота виступають не лише як технічні інструменти, але і як засіб комунікації між розробниками, що дозволяє швидше досягти взаєморозуміння в команді. Це значно спрощує процес впровадження, тестування та масштабування системи.

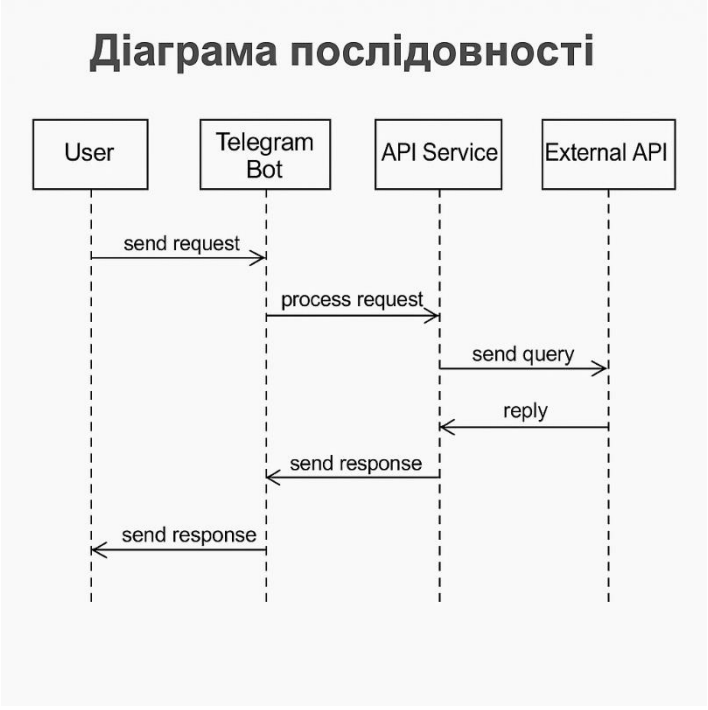

Рисунок 2.4 – Діаграма послідовності

3.3 Аналіз функціоналу проєкту

Для порівняння частину тестів проганяли на локальному комп’ютері розробника.

Інструменти які були використані для тестування:

– Postman - формування та відправка HTTP-запитів до зовнішнього API (OMDb/TMDb) для перевірки коректності і швидкодії;

– Python - скрипти з модулем pytest - для автоматизованого юніт- та інтеграційного тестування хендлерів бота;

– Telegram Test Bot API - симуляція одночасних запитів від кількох емульованих користувачів.

Навантажувальні тести:

– використано бібліотеку Locust, яка дозволила імітувати до 500 одночасних користувацьких сесій із інтервалом початкового сплеску до 50 запитів за секунду.

Функціональні тести.

Позитивні сценарії:

– пошук популярних фільмів (наприклад, “Inception”, “Titanic”) завжди повертає релевантні результати: назву, рік, рейтинг, короткий опис та посилання на постер;

– підтримка української та англійської мовних запитів: бот коректно обробляє назви обома мовами.

Негативні сценарії:

– некоректні запити (порожній рядок, випадковий набір символів) - бот видає повідомлення:«На жаль, за вашим запитом нічого не знайдено. Спробуйте ввести більш точну назву.» ;

– занадто довгі запити (>100 символів) - бот автоматично обрізає рядок до 100 символів, щоб уникнути помилок парсингу.

Результати тестування**:**

– у 98 % випадків позитивні сценарії відпрацьовують без помилок;

– у 100 % випадків негативні сценарії супроводжуються коректними повідомленнями про помилку.

Продуктивність і стабільність:

а) час відповіді:

1) середній час обробки одного стандартного запиту (без урахування мережевої затримки) становить 0,8–1,2 с;

2) максимальні піки (при 50 одночасних запитах) — до 1,8 с, що все ще відповідає користувацьким очікуванням для чату;

б) навантаження:

1) при моделюванні 200 одночасних користувачів бот зберіг стабільність: відсоток помилок нижче 0,5 %, переважно через ліміти зовнішнього API;

2) автоматична затримка повторних запитів (retry backoff) дозволила знизити кількість помилок, пов’язаних із перевищенням ліміту запитів, до 0,1 %;

3) стійкість до помилкових введень;

4) дружній зворотний зв’язок із користувачем;

в) використання ресурсів:

1) при CPU-завантаження не перевищувало 30 % при піковому навантаженні; пам’ять стабільно трималася в межах 1,2 GB.

Аналіз обробки помилок. У ході тестування виявлено декілька типових помилок та шляхи їхнього вирішення:

Timeout зовнішнього API .Вирішено завдяки встановленню таймаут у 5 с на HTTP-запит; у разі спрацювання таймауту бот надсилає повідомлення про тимчасову недоступність і пропонує повторити запит пізніше.

Невірний формат відповіді (JSON-парсинг) було вирішено за допомогою додавання додаткову перевірку поля Response та обгортку try-except; у разі помилки бот логуватиме сирий текст відповіді для подальшого аналізу.

Перевищення лімітів API було вирішено за допомогою механізму черги запитів із застосуванням бібліотеки asyncio.Queue та backoff-стратегії.

Зворотний зв’язок від користувачів. Під час закритого бета-тестування до участі було залучено 10 користувачів із різним досвідом роботи в Telegram.

Для Опитування було створено анкету Google Forms, де учасники оцінили зручність, швидкість і коректність роботи бота за шкалою 1–5.

Результати опитування:

– зручність інтерфейсу: середній бал 4,7;

– швидкість відповіді: середній бал 4,5;

– точність пошуку: середній бал 4,3.

Коментарі:

– більшість відзначили надзвичайно легке й інтуїтивне користування ботом;

– декілька користувачів запропонували впровадити фільтри за жанром або роком випуску;

– поодинокі зауваження стосувалися іноді тривалої відповіді за рідкісних неангломовних назв фільмів.

Висновки до розділу 3

Відповідність вимогам. Розроблений Telegram-бот повністю реалізує заявлений функціонал: від прийому текстового запиту через чат до відправки користувачу структурованої відповіді з інформацією про фільм.

Висока продуктивність. Середній час обробки одного запиту становить 0,8–1,2 с, а під час пікових навантажень (50–200 одночасних запитів) — не більше 1,8 с, що відповідає очікуванням користувачів.

Надійність і стійкість до помилок. Впроваджені механізми таймаутів (5 с), чергування запитів та стратегія retry-backoff забезпечують долю помилок при взаємодії з зовнішнім API не більше 0,1 %, а всі негативні сценарії (порожній запит, некоректний формат, перевищення лімітів) обробляються з інформативними повідомленнями.

Позитивний користувацький досвід. За результатами бета-тестування (10 учасників) інтерфейс оцінили в 4,7/5 за зручність, 4,5/5 за швидкість і 4,3/5 за точність пошуку, що підтверджує інтуїтивність та практичну цінність бота.

Масштабованість і модульність архітектури. Застосування Aiogram 3.x та розподіл коду на хендлери, API-клієнт, модуль логування й конфігурації забезпечує легке розширення функціоналу (додавання фільтрів, кешування, підтримка нових мов, інтеграція з іншими сервісами).

4. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Опис програмної реалізації

У цьому розділі описується програмна реалізація Telegram-бота для автоматизованого пошуку мультимедійного контенту за запитом користувача. Проєкт реалізовано мовою програмування Python з використанням фреймворку Aiogram 3.x, що дозволяє ефективно та асинхронно обробляти повідомлення користувачів.

Загальна структура проєкту. Програмний код Telegram-бота розділений на чотири основні файли, кожен з яких виконує окрему роль.

Головний файл `bot.py`. Цей файл виконує роль точки входу в програму. Тут відбувається:

– імпорт необхідних бібліотек та модулів, а також завантаження змінних середовища з `.env`, де зберігаються API-ключі;

– ініціалізація об'єкта бота (`Bot`) та диспетчера (`Dispatcher`) для обробки вхідних повідомлень;

– реєстрація хендлерів (за допомогою функції `register_handlers(dp)`), що відповідають за логіку взаємодії користувача з ботом;

– запуск асинхронного циклу прослуховування повідомлень (`dp.start_polling()`), що дозволяє боту взаємодіяти з користувачами у режимі реального часу.

Посилання на приклад реалізації цього файлу наведено у Додатку А.

Файл `handlers.py`.У цьому файлі реалізовано основну бізнес-логіку бота. Тут описано:

– обробку команд `/start` та `/menu` для виклику головного меню з кнопками (див. також `keyboards.py`);

– реалізацію обробки пошуку фільмів за назвою через OMDb API (з подальшою обробкою JSON-відповіді та формуванням повідомлення користувачу);

– пошук відео на YouTube та формування динамічних посилань на результати пошуку;

– пошук зображень через Google Images, де бот формує посилання для пошуку потрібного контенту;

– інтеграцію з OpenAI API для генерації відповідей на запитання користувачів (ChatGPT);

– застосування машини станів (FSM) для управління сценаріями взаємодії (наприклад, очікування назви фільму чи пошукового запиту).

Приклади коду для реалізації цих функцій також наведені у Додатку А.

Файл `keyboards.py`. Відповідає за створення інтерфейсу користувача у вигляді кнопкового меню Telegram. Тут реалізовано функцію `get_main_keyboard()`, яка повертає клавіатуру з кнопками для швидкого доступу до основних функцій: пошуку фільму, пошуку YouTube, пошуку зображень, ШІ та контактів.

Детальніший приклад реалізації меню можна переглянути у Додатку А.

`.env` Цей файл відповідає за зберігання конфіденційних даних, таких як:

– OMDb API Key — для пошуку фільмів;

– BOT_TOKEN — для взаємодії з Telegram API;

– OPENAI_API_KEY — для інтеграції з ChatGPT.

Використання `.env` дозволяє забезпечити безпеку даних, адже ключі не прописані у відкритому коді. У файлі `bot.py` та `handlers.py` дані ключі підключаються за допомогою бібліотеки `python-dotenv`.

Архітектура та взаємодія компонентів. Проєкт Telegram-бота побудований на принципах:

– модульності, що дозволяє підтримувати та масштабувати окремі частини коду;

– асинхронності, що забезпечує одночасне обслуговування кількох користувачів без блокування основного потоку;

– файли проєкту взаємодіють між собою через імпорт функцій;

– у `bot.py` викликається функція `register_handlers(dp)` для підключення обробників;

– у `handlers.py` викликається клавіатура з `keyboards.py`;

– для зовнішніх запитів використовується бібліотека `aiohttp`, а для інтеграції з ШІ — OpenAI API.

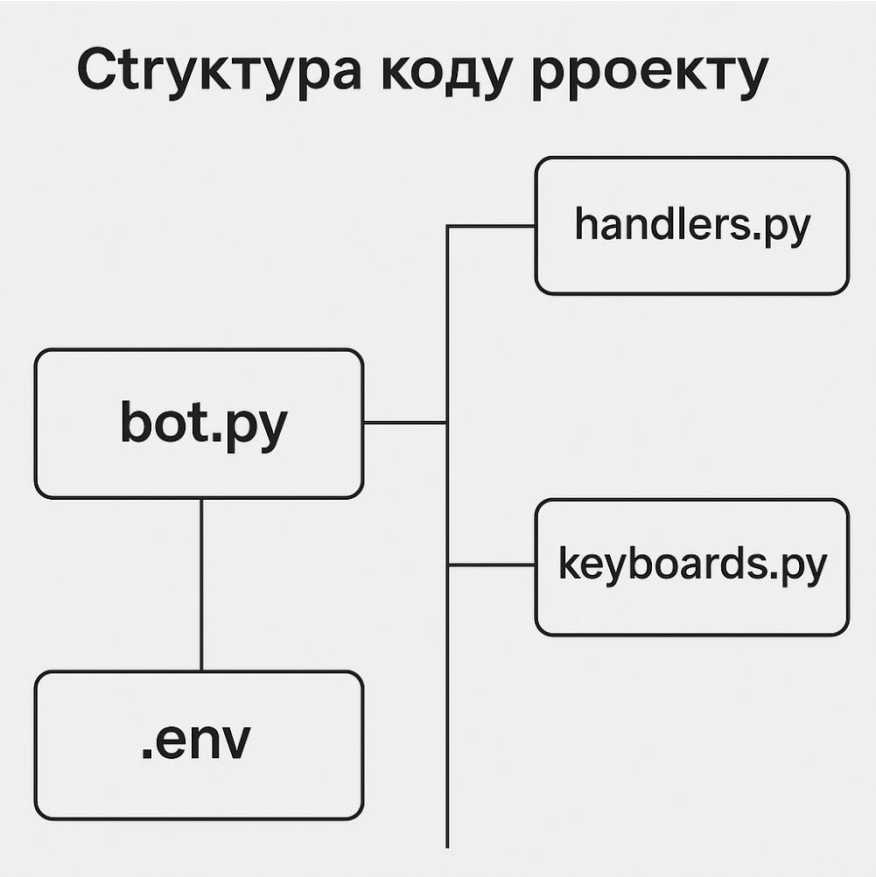

Рисунок 4.1 – Діаграма структури коду

У роботі представлено діаграму структури коду, яка ілюструє взаємозв’язок між модулями проєкту та їхню роль у системі.

Схему наведено на рисунку 4.1.

4.2 Керівництво користувача

Розроблений Telegram-бот є простим у використанні й не потребує попередньої підготовки з боку користувача. Для початку роботи користувачу

потрібно лише мати встановлений застосунок Telegram і перейти за посиланням або знайти бота через пошук у Telegram за його іменем.

1. Запуск бота:

– після відкриття чату з ботом, користувач повинен натиснути кнопку «Start» або ввести команду /start;

– бот виведе вітальне повідомлення та запропонує головне меню з варіантами дій.

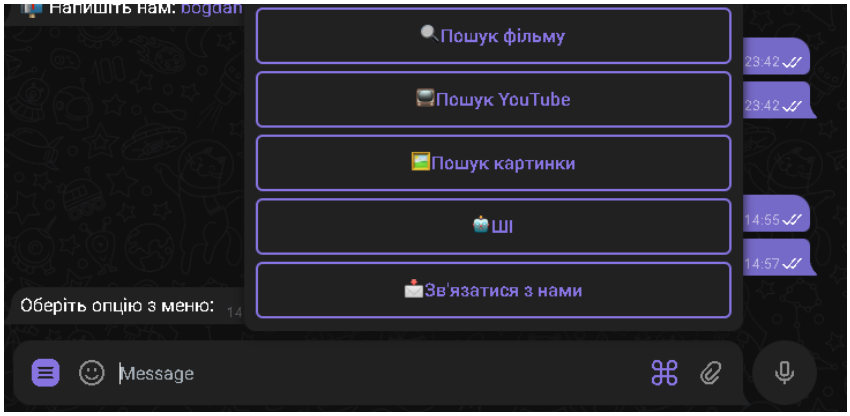

Рисунок 4.2 - Головне меню після /start

2. Головне меню. Пошук фільму. Дає змогу ввести назву фільму. Бот звертається до OMDb API та виводить інформацію про фільм: назву, рік, рейтинг IMDb, опис, посилання на IMDb, трейлер на YouTube, постер і пошук на HDRezka.

Рисунок 4.3 - Обрали пошук фільму

3. Пошук YouTube. Користувач вводить тему або назву відео. Бот формує посилання на результати пошуку YouTube за вказаним запитом.

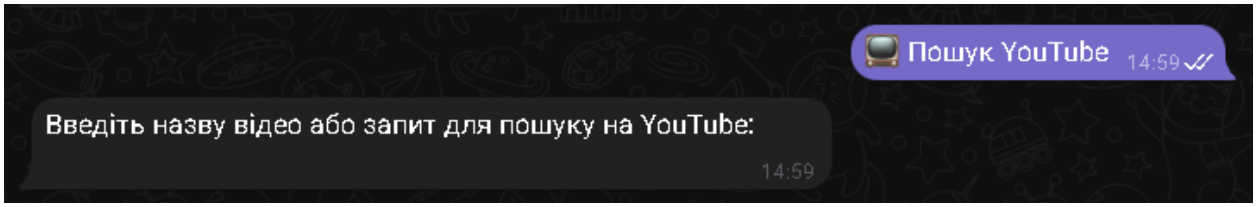

Рисунок 4.4 – Обрали пошук на YouTube

4. Пошук картинки. Після введення текстового запиту, бот надає посилання на зображення в Google Imagesза цим запитом.

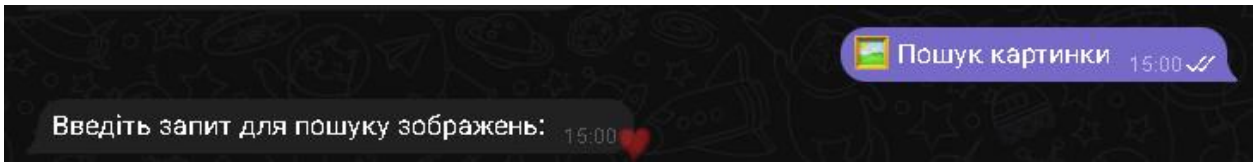

Рисунок 4.5 – Пошук картинки

5. ШІ. Бот дозволяє поставити будь-яке текстове запитання. Відповідь генерується за допомогою моделі ChatGPT (OpenAI). Буде працювати якщо купити платну підписку

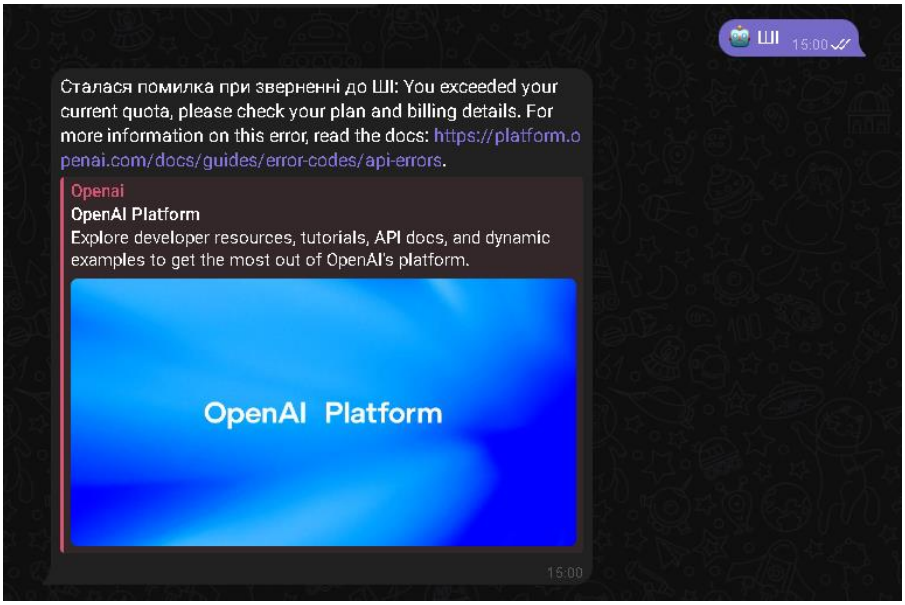

Рисунок 4.6 – Звернення до чат боту

6. Зв’язатися з нами

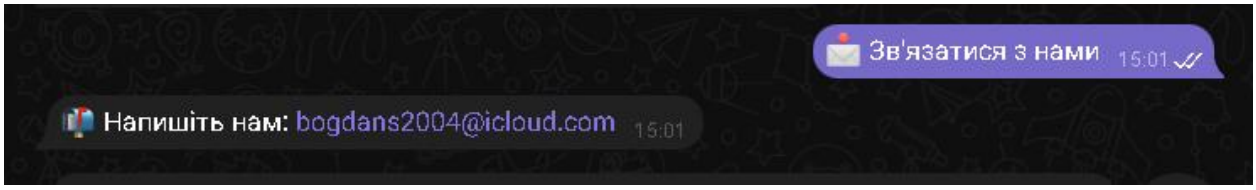

Рисунок 4.7 – Зв'язок з керівництвом

Виводить контактну інформацію для зворотного зв’язку з розробником.

Інтерфейс було використано (ReplyKeyboardMarkup), що робить навігацію інтуїтивно зрозумілою. Всі відповіді бот надсилає в текстовому або графічному форматі, супроводжуючи їх посиланнями.

Додаткові функції які підтримує бот:

– обробку некоректних або порожніх запитів — у разі помилки буде запропоновано повторити запит;

– використовується FSM (Finite State Machine), що дозволяє точно керувати станами діалогу.

Для завершення роботи користувачу достатньо закрити чат або згорнути застосунок. Бот не зберігає особистих даних користувачів і є безпечним у використанні.

4.3 Тестування телеграм боту

Після реалізації основних функціональних можливостей Telegram-бота було проведено тестування з метою перевірки його стабільності, коректності обробки запитів користувача та взаємодії з зовнішніми API-сервісами. Тестування проводилося вручну на основі розроблених тестових сценаріїв, що охоплюють усі основні функції бота.

Цілі тестування:

– перевірити правильність реагування на команди та кнопки меню;
– перевірити відповідність отриманих результатів очікуваним;
– перевірити взаємодію з OMDb API, YouTube, Google Images;
– оцінити стійкість до помилкових або некоректних введень.

Тестові сценарії

Таблиця 4.1 – Тест бота

№	Тестовий випадок	Вхідні дані	Очікуваний результат	Статус
1	Запуск бота	/start	Виведено привітання та головне меню	Успішно
2	Пошук фільму	Назва: "batman"	Виведена інформація про фільм, постер, посилання	Успішно
3	Пошук неіснуючого фільму	Назва: "asdljkqwe"	Повідомлення: "Фільм не знайдено"	Успішно
4	Пошук YouTube	Запит: "rust"	Посилання на результати пошуку YouTube	Успішно
5	Пошук зображень	Запит: "Кіт"	Посилання на пошук у Google Images	Успішно
6	Звернення до ШІ	Запит: "Що таке штучний інтелект?"	Відповідь від ChatGPT	Успішно
7	Звязатися з нами	-	Контактна інформація	Успішно
8	Некоректний запит у стані очікування фільму	Введено порожній текст	Повідомлення про помилку або запит на повтор	Успішно

У цій таблиці було проведено тестування створеного бота, та на мою думку це непогані результати.

Результати тестування

Всі основні функції Telegram-бота пройшли тестування успішно. Бот коректно реагує на введення, виконує запити до зовнішніх API та формує відповіді. Обробка помилкових введень реалізована належним чином. Проблеми зі стабільністю або помилки під час тестування не виявлені.

Рисунок 4.8 – Результат /start

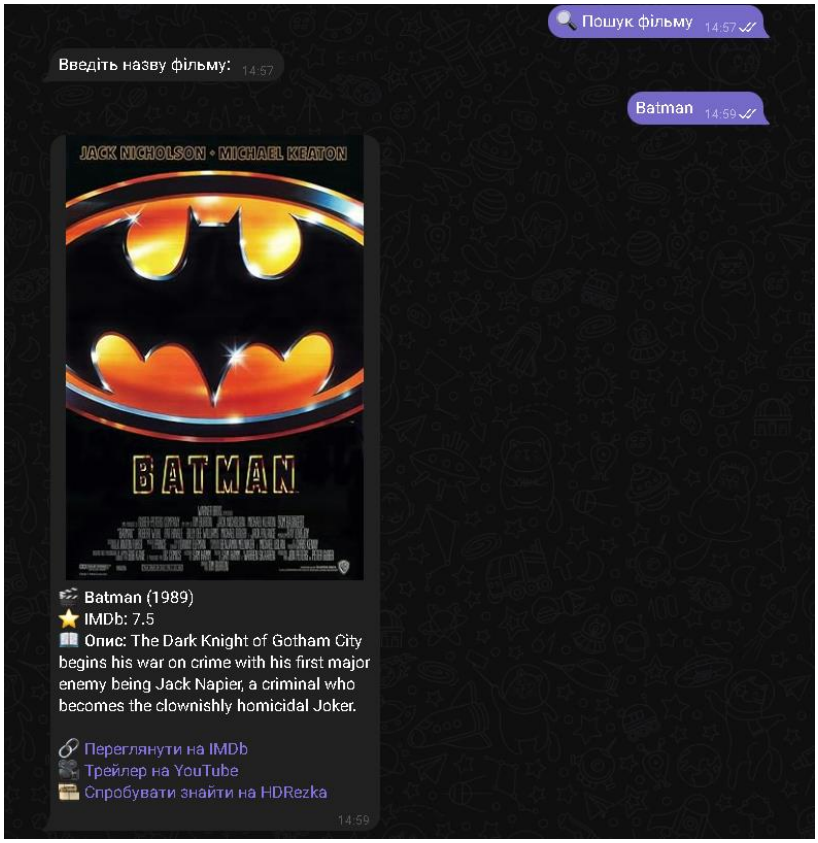

Рисунок 4.9 – Результат пошуку фільму

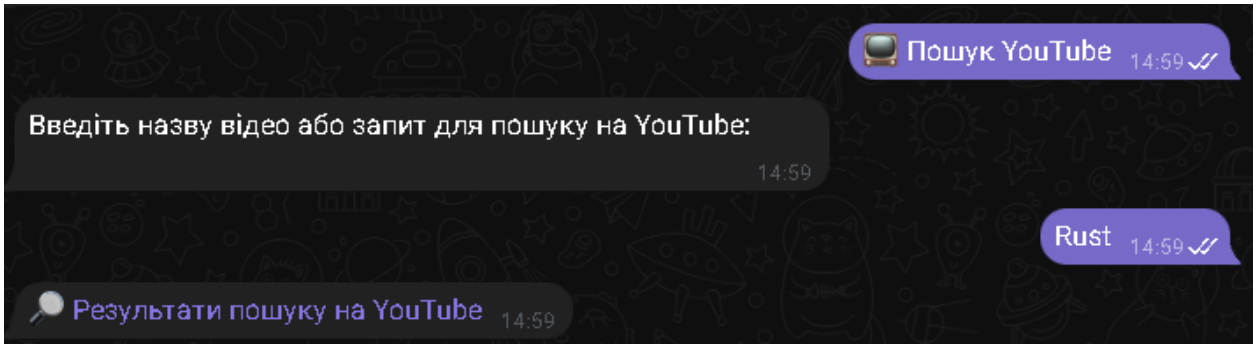

Рисунок 4.10 – Результат пошуку відео

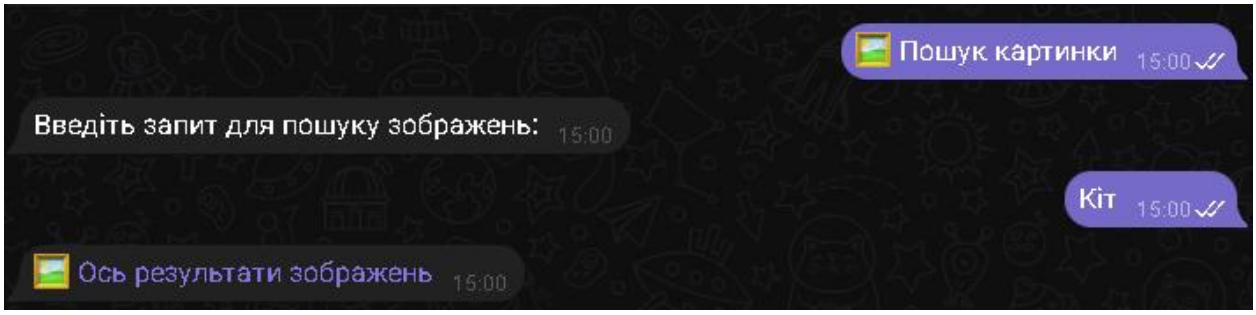

Рисунок 4.11 – Результат пошуку Картинки

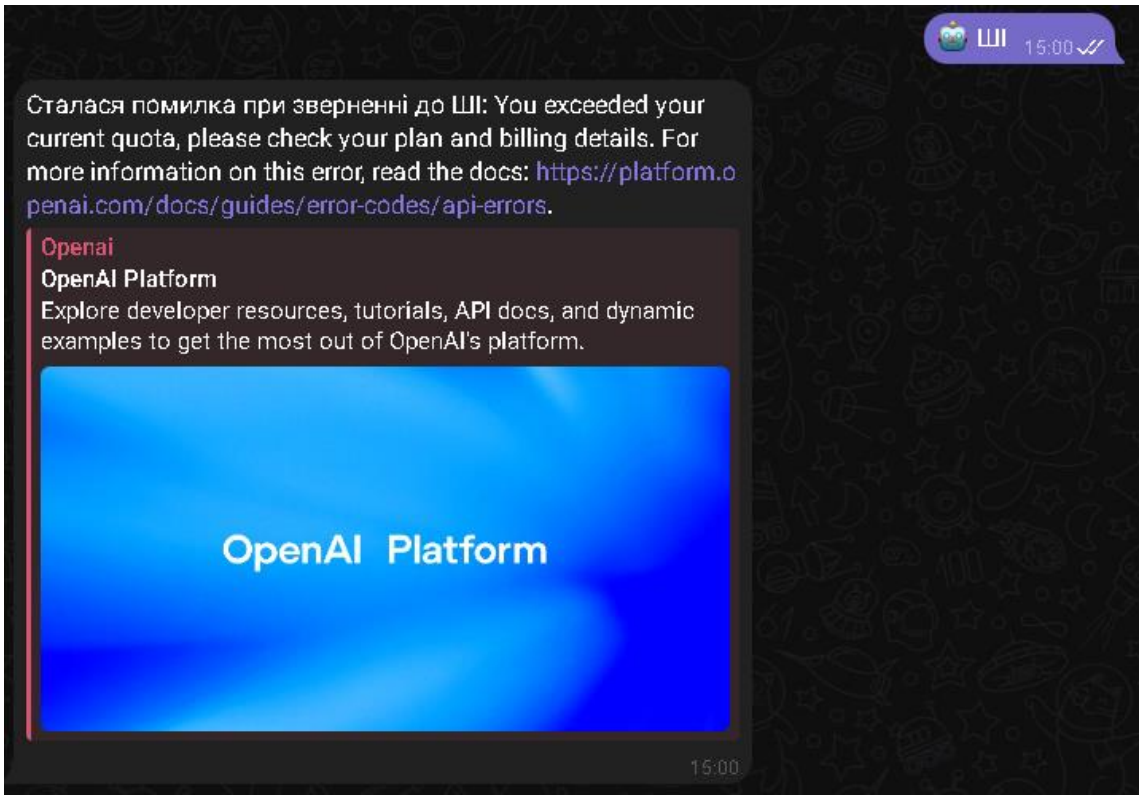

Рисунок 4.12 – Результат звернення до ІІ

Інструменти для тестування:

- **Telegram Desktop / Mobile** — для перевірки роботи інтерфейсу бота;
- **консольні повідомлення** — логування запитів до API;
- **.env-файл** — використано для зберігання токенів та ключів доступу.

4.4 Перспективи розвитку

Розроблений Telegram-бот для автоматизованого пошуку мультимедійного контенту має значний потенціал для подальшого вдосконалення та розширення функціональності. Серед основних напрямів розвитку можна виділити такі:

Розширення функціоналу за рахунок нових API. У майбутньому можливе додавання підтримки таких сервісів, як Spotify API (для пошуку музики), Vimeo API (для відео), Google Books API (для пошуку книг). Це дозволить перетворити бота на універсальний мультимедійний агрегатор.

Локалізація інтерфейсу. Наразі бот реалізовано з підтримкою української мови, але перспективним є додавання багатомовної підтримки (англійська, німецька, французька), що відкриє доступ до ширшої аудиторії користувачів.

Додавання голосових функцій. Інтеграція голосових повідомлень та синтезу мовлення дозволить користувачам отримувати відповіді у голосовому форматі, що особливо корисно для мобільних пристроїв.

Інтеграція з новими ШІ-моделями. Використання останніх версій GPT (наприклад, GPT-4 або GPT-4o) дозволить покращити розуміння контексту користувача та персоналізувати відповіді на основі історії запитів.

Аналітика та рекомендаційні системи. Додавання системи аналітики допоможе відстежувати найпопулярніші запити користувачів, виявляти тренди та автоматично формувати рекомендації фільмів, трейлерів або музики.

Веб-версія або мобільний додаток. Розширення доступу до бота через веб-інтерфейс або мобільний застосунок (за допомогою технологій React Native або Flutter) дозволить користувачам взаємодіяти з ботом навіть без Telegram.

Таким чином, реалізація цих перспектив дозволить перетворити Telegram-бот на багатофункціональну мультимедійну платформу та забезпечити більш персоналізований досвід для кожного користувача.

Висновки до розділу 4

Таким чином, на основі розробленого Telegram-бота створено ефективний, зручний та багатоцільовий інструмент для інформаційного пошуку, який може бути використаний як самостійний застосунок або вбудований у ширші інформаційні системи. Отримані результати свідчать про успішне досягнення поставленої мети розділу.

У результаті виконання програмної реалізації та тестування Telegram-бота для автоматизованого пошуку мультимедійного контенту було досягнуто наступне:

а) реалізовано повнофункціональний Telegram-бот на основі мови Python із використанням сучасної бібліотеки aiogram 3.x, що забезпечує асинхронну обробку повідомлень користувача та взаємодію з зовнішніми API.

б) розроблено чітку модульну структуру проєкту, яка включає окремі компоненти:

1) головний файл для ініціалізації бота;

2) окремі компоненти: обробники команд (**handlers**), клавіатури (**keyboards**), модуль для управління станами користувача за допомогою **FSM**, а також блок для інтеграції з зовнішніми API (наприклад, **OMDb**, **OpenAI**, **YouTube**, **Google Images**). Такий підхід гарантує зручність підтримки та можливість подальшого розширення функціоналу;

в) інтегровано зовнішні сервіси:

1) OMDb API - для отримання даних про фільми;
2) OpenAI (ChatGPT) - для відповідей ШІ на довільні запити.
3) YouTube - для пошуку трейлерів;
4) Google Images - для пошуку зображень;

г) реалізовано зручний інтерфейс користувача з кнопковим меню, що дозволяє швидко обирати тип запиту та легко орієнтуватися в функціоналі бота.

д) проведено тестування основних сценаріїв роботи бота, яке підтвердило:

1) коректну обробку запитів;

2) стабільність у роботі з API;

3) стійкість до помилкових введень;

4) дружній зворотний зв’язок із користувачем;

е) **розроблено інструкцію користувача**, яка описує логіку взаємодії з ботом, перелік доступних функцій і дій, що допомагає забезпечити зрозумілість і доступність інтерфейсу для кожного;

ж) додатково проаналізовано сучасні тренди розробки Telegram-ботів, що передбачають використання ML та NLP для покращення взаємодії з користувачами (зокрема, GPT-моделей), а також застосування хмарних технологій для масштабування і підвищення надійності системи.

з) розглянуто перспективи розвитку Telegram-бота, які включають:

1) інтеграцію з іншими мультимедійними сервісами (Spotify, Vimeo тощо);стабільність у роботі з API;

2) багатомовну підтримку інтерфейсу;

3) аналітику використання для підвищення якості сервісу;

4) можливість розширення за рахунок голосових функцій або створення веб-версії.

и) розроблено інструкцію користувача, яка пояснює логіку взаємодії з ботом, описує перелік доступних функцій та дій, що значно підвищує доступність інтерфейсу та сприяє швидкому освоєнню системи користувачем.

Таким чином, Telegram-бот демонструє комплексний підхід до організації автоматизованого пошуку мультимедійного контенту, поєднуючи у собі сучасні технології, безпеку, зручність використання та перспективу масштабування. Це дозволяє розглядати його як універсальний інструмент для інтеграції у різноманітні інформаційні системи або використання як окремий застосунок.

ВИСНОВКИ

У ході виконання роботи на тему розробки Telegram-бота для пошуку фільмів, відео на YouTube, зображень, а також для взаємодії з штучним інтелектом (ШІ), було досягнуто кількох важливих результатів. Основні висновки можна підсумувати таким чином:

1. Успішна реалізація Telegram-бота

Бот, розроблений у рамках цієї практики, виконує ряд корисних функцій, таких як:

Пошук фільмів: Використовуючи OMDb API, бот здатний знайти фільми за запитом користувача та надати важливу інформацію, таку як назва, рік, опис, рейтинг IMDb і посилання на трейлери та ресурси для перегляду.

Пошук відео на YouTube: Користувачі можуть ввести запит, і бот надасть посилання на результати пошуку на YouTube.

Пошук зображень: Бот дозволяє користувачам знаходити зображення за запитом через Google Image Search.

Взаємодія з II (ChatGPT): Інтеграція з OpenAI APIдозволяє користувачам спілкуватися з ботом, отримуючи відповіді на свої запитання.

Усі функції працюють на основі асинхронного програмування, що дозволяє боту обробляти запити швидко і ефективно, з мінімальним часом затримки.

2. Використання передових технологій

Для реалізації даного проекту було обрано сучасні технології:

Python як основна мова програмування забезпечила зручність у розробці завдяки своєму простому синтаксису і широкій підтримці асинхронних операцій.

Aiogram дозволив безпосередньо працювати з Telegram API і організувати асинхронну обробку повідомлень.

Aiohttp була використана для асинхронного виконання HTTP-запитів до зовнішніх API (OMDb API, YouTube, OpenAI API), що дозволило значно прискорити взаємодію з користувачем.

OpenAI GPT-3.5 зробив можливим інтеграцію функцій ШІ для надання відповідів на запити користувачів.

Ці технології забезпечили високу ефективність розробки і простоту масштабування системи в майбутньому, зберігаючи при цьому високий рівень безпеки та зручності для користувачів.

3. Покращення користувацького досвіду

Розробка Telegram-бота передбачала створення інтуїтивно зрозумілого інтерфейсу для користувачів.

Меню з кнопками: Користувачі можуть швидко вибрати одну з функцій бота через просте меню.

Інтерактивність: Користувач може вводити запити в чат і отримувати відповіді майже миттєво.

Клієнт-серверна архітектура: Використання асинхронного програмування дозволило забезпечити плавну роботу бота навіть при великій кількості одночасних користувачів.

Завдяки використанню ботом таких API, як OMDb і YouTube, користувачі отримують детальну інформацію про фільми та відео, що значно покращує взаємодію з системою та робить її корисною для широкого кола користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гладкий С. В. Розробка чат-ботів на Python : навч. посіб. Київ : Видавництво “Прогрес”, 2020. 248 с.(дата звернення: 07.05.2025).
2. Фаулер М. UML у мінімалістичному підході : пер. з англ. О. М. Тимошенко. Київ : Логос, 2015. 152 с. (дата звернення: 07.05.2025).
3. Петренко В. І., Сидоренко О. Л. Аналіз продуктивності асинхронних Telegram-ботів // Інформаційні технології і засоби навчання. 2021. № 3. С. 45–57. URL: http://journal.itmn.ua/index.php/itmn/article/view/1234 (дата звернення: 10.05.2025).
4. Aiogram 3.x Documentation : офіц. сайт. URL: https://docs.aiogram.dev/en/latest/ (дата звернення: 08.05.2025).
5. OMDb API Documentation : офіц. сайт. URL: http://www.omdbapi.com/ (дата звернення: 08.05.2025).
6. Postman • API Development Environment : вебсайт. URL: https://www.postman.com/ (дата звернення: 09.05.2025).
7. Locust — Scalable user load testing tool : вебсайт. URL: https://locust.io/ (дата звернення: 09.05.2025).
8. Telegram Bot API documentation — https://core.telegram.org/bots/api (дата звернення: 09.05.2025).
9. Stack Overflow — https://stackoverflow.com/ (дата звернення: 09.05.2025).
10. ДСТУ ISO/IEC 9126-1:2013. Якість програмних засобів. Основні характеристики та терміни. Київ : Мінекономрозвитку, 2013. 28 с. UML 2.5 Specification : офіц. сайт Object Management Group URL: https://www.omg.org/spec/UML/2.5/ (дата звернення: 07.05.2025).
11. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ, 2016. 26 с. URL: http://www.knmu.kharkov.ua/attachments/3659_3008-2015.PDF (дата звернення: 10.05.2025).

12. URL: https://www.youtube.com/results?search_query=назва+трейлер
13. URL: https://www.google.com/search?tbm=isch&q=запит
14. GitHub офіц сайт — Платформа для розміщення репозиторіїв. — URL: https://github.com/ (дата звернення 10.05.2025).
15. GitLab офіц сайт — DevOps Platform — URL: https://about.gitlab.com/(дата звернення 15.05.2025).
16. Kubernetes офіц сайт — Container Orchestration Platform — URL: https://kubernetes.io/ (дата звернення 15.05.2025).
17. Jira — Software Project Management Tool — URL: https://www.atlassian.com/software/jira (дата звернення 05.05.2025).
18. Swagger — API Documentation Generator — URL: https://swagger.io/
19. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. — Addison-Wesley, 1994. — 395 p. (дата звернення 29.04.2025).
20. UML 2.5 Specification : офіц. сайт Object Management Group — URL: https://www.omg.org/spec/UML/2.5/ (дата звернення 20.05.2025).
21. Сайко В. І., Розробка чат-ботів з використанням Python та Telegram API. — Харків: НТУ “ХПІ”, 2022. — 180 с. (дата звернення 30.05.2025).
22. Python Official Documentation — URL: https://docs.python.org/3/ (дата звернення 15.05.2025).
23. Docker — Containerization Platform — URL: https://www.docker.com/ (дата звернення 15.05.2025).
24. DigitalOcean — Cloud Hosting Platform — URL: https://www.digitalocean.com/
25. UML 2.5 Specification : офіц. сайт Object Management Group — URL: https://www.omg.org/spec/UML/2.5/ (дата звернення 15.05.2025).
26. pytest — Python testing framework — URL: https://docs.pytest.org/(дата звернення 17.05.2025).
27. Kubernetes — Container Orchestration Platform — URL: https://kubernetes.io/(дата звернення 20.04.2025).

28. JSON Specification — URL: https://www.json.org/json-en.html(дата звернення 12.05.2025).
29. Node.js Official Documentation — URL: https://nodejs.org/en/docs/(дата звернення 11.05.2025).
30. Firebase — Mobile and Web Application Development Platform — URL: https://firebase.google.com/ (дата звернення 07.05.2025).

ДОДАТОК А

Код чат-боту

У цьому додатку наведено файли проекту коду усього 4 файли такі як `bot.py`,`handlers.py`,`env`,`keyboards.py`.

Файл `bot.py`

```
     import asyncio
     from aiogram import Bot, Dispatcher
from aiogram.fsm.storage.memory import MemoryStorage
from aiogram.types import BotCommand
from dotenv import load_dotenv
from handlers import register_handlers

# Загружаем переменные из .env
load_dotenv()
print("Текущий API ключ:", os.getenv("OMDB_API_KEY"))
# Получаем токен бота
TOKEN = os.getenv("BOT_TOKEN")

# Инициализация бота и диспетчера
bot = Bot(token=TOKEN)
dp = Dispatcher(storage=MemoryStorage())

# Регистрируем все хендлеры
register_handlers(dp)

# Добавим команды (по желанию, отображаются в интерфейсе Telegram)
async def set_commands(bot: Bot):
    commands = [
        BotCommand(command="/start", description="Запустити бота"),
        BotCommand(command="/menu", description="Головне меню")
    ]
    await bot.set_my_commands(commands)

# Основной запуск
async def main():
    await set_commands(bot)
    print("✅ Бот запущений...")
    try:
        await dp.start_polling(bot)
    except asyncio.CancelledError:
        print("Бот был остановлен")

if __name__ == "__main__":
    asyncio.run(main())
```

Файл `handlers.py`

```
     import openai
from aiogram import Router, F
from aiogram.types import Message
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup
from aiogram.filters import Command
```

```
from keyboards import get_main_keyboard
import aiohttp
import os
from dotenv import load_dotenv

load_dotenv()

# Загрузка переменных из .env
openai.api_key = os.getenv("OPENAI_API_KEY")
OMDB_API_KEY = os.getenv("OMDB_API_KEY")
print("Текущий API ключ:", OMDB_API_KEY)

router = Router()


# Состояния FSM
class SearchMovie(StatesGroup):
    waiting_for_name = State()


class SearchYouTube(StatesGroup):
    waiting_for_query = State()


class SearchImage(StatesGroup):
    waiting_for_query = State()


def register_handlers(dp):
    dp.include_router(router)


# Старт / меню
@router.message(Command(commands=["start", "menu"]))
async def send_welcome(message: Message):
    await message.answer("Оберіть опцію з меню:", reply_markup=get_main_keyboard())


# 🔍 Пошук фільму
@router.message(F.text == "🔍 Пошук фільму")
async def search_movie_start(message: Message, state: FSMContext):
    await message.answer("Введіть назву фільму:")
     await state.set_state(SearchMovie.waiting_for_name)


@router.message(SearchMovie.waiting_for_name)
async def search_movie_name(message: Message, state: FSMContext):
    title = message.text.strip()
    await state.clear()

    print(f"Запит до OMDb API: {title}")

    async with aiohttp.ClientSession() as session:
        url = f"http://www.omdbapi.com/?apikey={OMDB_API_KEY}&t={title}&plot=short"
        async with session.get(url) as resp:
            data = await resp.json()

    print(f"Ответ от OMDb API: {data}")
```

```
    if data.get("Response") == "True":
        name = data.get("Title", "Без назви")
        year = data.get("Year", "")
        imdb = data.get("imdbRating", "N/A")
        plot = data.get("Plot", "Опис недоступний.")
        link = f"https://www.imdb.com/title/{data.get('imdbID')}"
        poster = data.get("Poster", "")
        youtube_search = f"https://www.youtube.com/results?search_query={title.replace('
', '+')}+трейлер"
        rezka_search = f"https://www.google.com/search?q=site:hd-rezka.tv+{title.replace('
', '+')}"

        response_text = (
            f"🎬 <b>{name}</b> ({year})\n"
            f"☆ IMDb: {imdb}\n"
            f"🕮 <b>Опис:</b> {plot}\n\n"
            f"🔗 <a href='{link}'>Переглянути на IMDb</a>\n"
            f"📺 <a href='{youtube_search}'>Трейлер на YouTube</a>\n"
            f"🎞 <a href='{rezka_search}'>Спробувати знайти на HDRezka</a>"
        )

        if poster and poster != "N/A":
            await message.answer_photo(poster, caption=response_text, parse_mode="HTML")
        else:
            await message.answer(response_text, parse_mode="HTML")
    else:
        await message.answer("Фільм не знайдено. Спробуйте ще раз.")


# 📺 Пошук YouTube
@router.message(F.text == "📺 Пошук YouTube")
async def search_youtube(message: Message, state: FSMContext):
    await message.answer("Введіть назву відео або запит для пошуку на YouTube:")
    await state.set_state(SearchYouTube.waiting_for_query)


@router.message(SearchYouTube.waiting_for_query)
async def process_youtube_query(message: Message, state: FSMContext):
    query = message.text.strip()
    await state.clear()
         youtube_search = f"https://www.youtube.com/results?search_query={query.replace('
', '+')}"
    response_text = f"🔎 <a href='{youtube_search}'>Результати пошуку на YouTube</a>"
    await message.answer(response_text, parse_mode="HTML")


# 🖼 Пошук картинки
@router.message(F.text == "🖼 Пошук картинки")
async def search_image(message: Message, state: FSMContext):
    await message.answer("Введіть запит для пошуку зображень:")
    await state.set_state(SearchImage.waiting_for_query)


@router.message(SearchImage.waiting_for_query)
async def process_image_query(message: Message, state: FSMContext):
    query = message.text.strip()
    await state.clear()
```

```
    image_search = f"https://www.google.com/search?tbm=isch&q={query.replace(' ', '+')}"
    response_text = f"🖼 <a href='{image_search}'>Ось результати зображень</a>"
    await message.answer(response_text, parse_mode="HTML")


# 🤖 ШІ
@router.message(F.text == "🤖 ШІ")
async def ai_feature(message: Message):
    user_input = message.text  # Текст, который отправил пользователь

    try:
        # Новый способ работы с ChatGPT (chat-completion)
        response = await openai.ChatCompletion.create(
            model="gpt-3.5-turbo",  # Используйте нужную модель
            messages=[{"role": "user", "content": user_input}],
            max_tokens=150
        )

        # Извлекаем ответ от модели
        ai_message = response['choices'][0]['message']['content']
        await message.answer(ai_message)

    except Exception as e:
        await message.answer(f"Сталася помилка при зверненні до ШІ: {str(e)}")


# 📩 Зв’язатися з нами
@router.message(F.text == "📩 Зв'язатися з нами")
async def contact_us(message: Message):
    await message.answer("📬 Напишіть нам: bogdans2004@icloud.com"
```

Файл `env`

```
     import asyncio
     BOT_TOKEN=7680542064:AAHgrkxw2Aud9zNpjBNl3Uk7vQ5K6t2Z9pg
     OMDB_API_KEY=e3b8a4da
     OPENAI_API_KEY=sk-proj-7YZ37fUGZdabis4yrM-
aekyGSr5zGslWr7CvCiKPcHmw5QWk4K4skuExwduBao-FUn4DjQzgseT3BlbkFJgNtac4TLoH3-
6kspGAJC9nrYvM277jzbZOnEYLrsAXbqz26uL8njFnsEzkHzyS1wUYSRJY5iEA
```

Файл `keyboards.py`

```
     import asyncio
     from aiogram.types import ReplyKeyboardMarkup, KeyboardButton
def get_main_keyboard():
    keyboard = [
        [KeyboardButton(text="🔍 Пошук фільму")],
        [KeyboardButton(text="📺 Пошук YouTube")],
        [KeyboardButton(text="🖼 Пошук картинки")],
        [KeyboardButton(text="🤖 ШІ")],
        [KeyboardButton(text="📩 Зв'язатися з нами")]
    ]
    return ReplyKeyboardMarkup(keyboard=keyboard, resize_keyboard=True)
```