ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Чат-бот для пошуку фільмів на доступних веб-ресурсах, за допомогою мови програмування Python»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр ПЕРЕПАДЯ
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр. КНД-42
Олександр ПЕРЕПАДЯ

Керівник: Володимир ВАСИЛЕНКО
науковий ступінь, вчене звання к.т.н., доцент

Рецензент: _____
науковий ступінь, вчене звання _____
(Ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Перепаді Олександру Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Чат-бот для пошуку фільмів на доступних веб-ресурсах, за допомогою мови програмування Python»

керівник кваліфікаційної роботи Василенко В.В. доцент кафедри
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» 02.2025 р. № 56

2. Строк подання кваліфікаційної роботи «31» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, мова програмування Python, платформа Telegram, параметри для створення інформаційної системи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз середовища розробки, опис проектування чат-боту, опис розробки боту
5. Перелік графічного матеріалу: *презентація(назви основних слайдів)*
1. Об`єкт, Предмет та Мета Дослідження
2. Технічне завдання

3. Програмне забезпечення
4. Діаграма прецедентів
5. Огляд чат-боту
6. Висновок

6. Дата видачі завдання «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	24.02-09.03.25	Виконано
2	Розробка концепції та архітектури додатку	09.03-16.03.25	Виконано
3	Розробка концепції та архітектури додатку Текст аналітичної частини (1-го розділу)	17.03-23.03.25	Виконано
4	Вибір засобів для реалізації проекту (2-ий розділ)	24.03-29.03.25	Виконано
5	Реалізація програмної частини проекту	01.04-07.04.25	Виконано
6	Текст частини з розробкою додатку (3-й розділ)	08.04-24.04.25	Виконано
7	Вступ, висновок, реферат,	25.04-02.05.25	Виконано
8	Презентація	03.05-08.05.25	Виконано

Здобувач вищої освіти _____ Олександр ПЕРЕПАДЯ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Володимир ВАСИЛЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)'*

РЕФЕРАТ

Дипломна робота присвячена розробці та опису чат-боту створеного на платформі Telegram, який використовуватиметься для полегшеного пошуку фільмів .

Дипломна робота знаходиться на 59 сторінках і включає 24 картинки та 11 бібліографічних посилань.

Вона зроблена з відповідних розділів: вступ, 3 розділи провідної частини, висновки та перелік посилань.

Актуальність обраної теми полягає в тому що у сучасному цифровому суспільстві месенджер відіграє важливу роль у щоденному спілкуванні, роботі та розвагах мільйонів користувачів. Telegram є одним з найпопулярніших месенджерів, не тільки надаючи обмін повідомленнями, але і потужні інструменти автоматизації завдяки ботам. Розробка телеграм-ботів відкриває різноманітні можливості для створення інтерактивних сервісів, особливо в сфері розваг.

Метою роботи є розробка функціонального Telegram-бота для автоматизованого пошуку фільмів, який забезпечує користувачеві зручний інтерфейс для отримання інформації про кінофільми за допомогою інтеграції з відкритими кіно-базами даних (API), зокрема TMDb. Досягнення поставленої мети передбачає створення програмного продукту, який реалізує пошук фільмів рейтином, жанром та інших параметрів через інтерактивний інтерфейс у месенджері Telegram.

Об'єктом дослідження є інформаційні системи автоматизованого надання даних користувачеві використовуючи середовище месенджерів, зокрема Telegram-боти як інструмент взаємодії людини з інформаційними сервісами.

Предметом дослідження технології розробки Telegram-ботів, способи інтеграції з відкритими API баз даних фільмів та програмні засоби реалізації пошукових і діалогових функцій у месенджерах.

Ключові слова: Telegram-бот, чат-бот, API, пошук фільмів , месенджер.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 9
ВСТУП 10
1 АНАЛІТИЧНА ЧАСТИНА 12
1.1 Поняття чат-боту 12
1.2 Чат-боти в Телеграм для пошуку фільмів 14
1.3 Огляд існуючих рішень 16
1.4 Що таке Телеграм 19
1.5 Діаграма прецедентів 21
1.6 Обґрунтування теми проекту, актуальність 22
2 ЗАСОБИ РЕАЛІЗАЦІЇ ПРОЕКТУ 25
2.1 Середовище розробки PyCharm 25
2.2 Python 26
2.3 Бібліотеки для написання телеграм-бота 27
2.3.1 Бібліотека logging 28
2.3.2 Бібліотека requests 28
2.3.3 Бібліотека python-telegram-bot 29
3 ОПИС РОЗРОБКИ ЧАТ-БОТУ 30
3.1 Процес підготовки до розробки чат-боту 30
3.2 Процес розробки чат-боту 34
3.2.1 Підключення бібліотек 36
3.2.2 Конфігурація API 37
3.2.3 Налаштування логування 39
3.2.4 Обробник команди /start 40
3.2.5 Отримання трендових фільмів 40
3.2.6 Робота з жанрами 42
3.2.7 Головна функція 45
3.2.8 Метод Long Polling 47
ВИСНОВКИ 49

ЛІТЕРАТУРА 50
ДОДАТОК А 51
ДОДАТОК Б 56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (*Application Programming Interface*) — інтерфейс прикладного програмування; набір засобів для взаємодії між програмним забезпеченням і зовнішніми сервісами. У роботі використовується API ресурсу TMDb для отримання даних про фільми.

TMDb (*The Movie Database*) — онлайн-база даних про фільми, телесеріали, акторів та інший мультимедійний контент. Надання даних здійснюється через відкритий API.

URL (*Uniform Resource Locator*) — уніфікований ідентифікатор ресурсу, що вказує на розташування документа чи сервісу в Інтернеті.

Python — мова програмування високого рівня, що застосовується для розробки чат-ботів, веб-скрейпінгу, обробки даних тощо.

Telegram Bot API — інтерфейс програмування для створення ботів у месенджері Telegram, що дозволяє надсилати повідомлення, отримувати події, керувати діалогом тощо.

async / await — ключові слова в Python, що використовуються для асинхронного програмування; дозволяють не блокувати виконання програми під час очікування відповіді від зовнішніх ресурсів.

requests — бібліотека Python для надсилання HTTP-запитів і отримання відповідей від веб-серверів.

logging — модуль Python для реєстрації інформації про хід виконання програми, налагодження та обробки помилок.

ВСТУП

Чат-боти - це сучасні програмні рішення, які імітують розмови з користувачами в реальному часі. Вони доступні в різних областях, зокрема електронної комерції, обслуговування клієнтів, охорони здоров'я, освіти та розваг. Використовуючи чат-боти для пошуку розважального контенту, ви можете автоматизувати рутинні дії, швидко отримати доступ до актуальної інформації та створити зручний інтерактивний інтерфейс для користувачів.

У рамках цієї роботи був розроблений чат-бот, який має інтелектуального помічника для пошуку відео на доступних веб-ресурсах. Основною мовою програмування для реалізації рішення є Python, загальна мова високого рівня, широко використовується в області штучного інтелекту, обробки даних, веб-розробки та автоматизації. Завдяки великій кількості бібліотек (таких як запити, BeautifulSoup, telebot, aiogram, re і т.д.) Python забезпечує гнучкість та ефективність у створенні веб-роботів та інструментів для обробки інформації з Інтернету.

Мета роботи - створити функціонального чат-бота, який зможе шукати фільми за ключовими словами, типом, роком випуску, актором або іншими критеріями на основі наявних інтернет-ресурсів. Роботи повинні мати простий інтерфейс, який швидко реагує на запити користувачів, надає структуровану інформацію про фільм (назва, короткий опис, рік, клас, посилання на перегляд тощо) і забезпечує простоту використання. У процесі розробки також розглядаються захоплення мережі (автоматичний збір інформації з веб-сторінок), побудова логіки діалогу, обробка запитів природними мовами та інтеграція з програмами миттєвих повідомлень, такими як Telegram. Це дозволяє оцінити практичні аспекти створення сучасної інформаційної системи на базі Python і продемонструвати її практичні можливості застосування при вирішенні прикладних завдань.

Актуальність даної теми обумовлена зростаючим попитом на інтелектуальних цифрових помічників і необхідністю пошуку користувачами контенту зручно, швидко і ефективно. Запропоноване рішення може поліпшити

взаємодію з інформаційними службами і полегшити розробку систем автоматизації в області цифрових технологій.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Поняття чат-боту

Чат-боти - це інтерактивні програми, які імітують спілкування з людьми за допомогою текстових або голосових повідомлень. Вони дозволяють автоматизувати взаємодію між користувачами та комп'ютерами на основі заздалегідь визначених сценаріїв або за допомогою методів штучного інтелекту. Чат-боти можуть самостійно вести прості розмови з користувачами, відповідати на запити, виконувати команди та надавати корисну інформацію.

Перший приклад такої програми з'явився в 1966 році. Найвідомішим раннім проектом став віртуальний співрозмовник ELIZA, який симулював розмови з психотерапевтами. У той час робототехніка, хоча і дуже обмежена в можливостях, заклала основу для подальшого розвитку системи діалогу.

З 2010 року, зі зростанням популярності мобільних месенджерів, почався новий етап розвитку чат-ботів. Їх починають активно використовувати на таких платформах, як Telegram, Facebook Messenger, Viber, Skype, Slack і т.д. В даний час чат-боти можуть існувати як автономні додатки або вбудовуватися в існуючі сервіси і пошукові системи.

Чат-боти широко використовуються в наступних галузях:

- інтернет-банкінг (оплата, баланс повідомлень тощо);
- Навчання (навчальні тести, мовні тренажери);
- Готельно-ресторанний бізнес (бронювання столиків, замовлення);
- Інтернет-магазин (підбір товару, консультація);
- Індустрія розваг (рекомендації щодо фільмів, ігор, музики).

Більшість сучасних роботів зосереджуються на вузькоспеціалізованих завданнях і рідко охоплюють широкий спектр функцій. Чат-боти мають різні класифікації в літературі та практиці, але в цілому їх можна класифікувати на ділові та технічні класифікації.

Бізнес-класифікація чат-ботів (Рисунок 1.1)

1.Чат-боти для розмови - Для будь-яких конкретних цілей, для загального спілкування, жартувати або імітувати розмови в реальному часі.

2.Chat Assistant - Функції, які чітко визначені, наприклад, допомога у виборі продукту, збір інформації або усунення несправностей.

3.Q & a

Рисунок 1.1 – Бізнес-класифікація чат-бот

Технічна класифікація чат-ботів (рис. 1.2)

1. Робототехніка на основі бізнес-правил

2. Робота за запланованою програмою. Користувач вибирає одну з відповідей. Такі роботи не розуміють вільного тексту і працюють як системи сценаріїв або «роботи сценаріїв».

3. Роботи на основі штучного інтелекту

4. Використання NLP (обробка природної мови), NLU (розуміння природної мови), нейронних мереж. Вони самостійно розробляють відповіді, вивчаючи контексти та аналізуючи великі обсяги даних розмов. Перевагою є гнучкість, а недоліком - необхідність значних ресурсів для навчання.

5. Гібридні чат-боти

6. Об'єднати перші два типи елементів. Зазвичай існують попередньо написані сценарії для стандартних запитів і обмежена підтримка AI для обробки більш складних випадків. Це найпоширеніший тип робота на даному етапі.

Рисунок 1.2 – Технічна класифікація чат-ботів

Операційна архітектура чат-бота

Загальна процедура взаємодії користувача з чат-ботом така:

1. Користувачі відправляють запити через один з доступних каналів (месенджер, голосовий помічник, веб-інтерфейс).

2. Якщо запит є голосовим запитом, використовуйте методи ASR (автоматичного розпізнавання мовлення) та TTS (тексту в голос) для перетворення в текст.

3. Запит доступу до платформи обміну повідомленнями для ідентифікації намірів користувача за допомогою технології NLU.

4. Крім того, запит обробляється DialogManager, який керує скриптами, зберігає контексти та надсилає запити зовнішнім інформаційним системам, якщо це необхідно.

5. Після обробки формується відповідь - текст, мова або дія.

6. Якщо інформації недостатньо, робот уточнює запит від користувача.

7. Заповнені відповіді відправляються назад користувачеві в зручній формі.

В результаті чат-боти є комплексною системою, яка забезпечує автоматизовану обробку інформації в реальному часі, значно підвищуючи ефективність цифрових взаємодій.

1.2 Чат-боти в Телеграм для пошуку фільмів

Пошукові боти фільмів - це спеціалізовані програми (в основному в миттєвих повідомленнях, таких як Telegram, Discord або соціальні мережі), які допомагають

користувачам швидко знаходити інформацію про відеоконтент, такий як фільми, телешоу та серіалів. Вони інтегровані з базами даних фільмів, такими як TMDb, IMDb, Kinoshuk і забезпечують зручний інтерфейс пошуку.

Основні функції кіно-ботів:

- Пошук фільмів за назвою - можливість пошуку фільмів за ключовим словом.
- Рекомендації - вибір за типом, оцінкою та популярністю.
- Афіші та новинки - Інформація про прем'єри, тренди в кіноіндустрії.
- Рейтинги та відгуки - Рейтинги для IMDb, TMDb, Kinoshuku.
- Посилання на перегляд - іноді інтегрована з законними потоковими медіа-платформами.
- Зручний інтерфейс в Messenger - немає необхідності відкривати інші сайти.

Боти бувають поділені за типом платформи:

- Telegram-боти (наприклад, @MovieSearchBot, @KinoUABot)
- Discord-боти (наприклад, MovieNight, FilmFinder)
- Чат-боти у соцмережах (Facebook Messenger, Viber)
- Голосові помічники (Alexa, Google Assistant з інтеграцією кіносервісів)
- Та за функціоналом:
- Універсальні – пошук за назвою, жанром, актором.
- Тематичні – тільки аніме, тільки класика кіно, тільки українські фільми.
- Агрегатори – показують, де дивитися фільм (Netflix, Disney+, MEGOGO тощо).

Як працюють кіно-боти?

Користувач пише команду або натискає кнопку (наприклад, "/top").

Бот робить запит до API кінобази (TMDb, IMDb, Кінопошук).

Отримує дані, обробляє їх і відправляє у вигляді повідомлення.

Додає посилання, рейтинги, опис тощо.

Переваги кіно-ботів перед звичайними сайтами:

- Швидкість – не треба відкривати браузер і шукати вручну.
- Зручність у месенджері – все в одному місці (Telegram, Discord).
- Менше реклами – багато кіносайтів перевантажені баннерами.
- Персоналізація – деякі боти запам’ятовують уподобання.

Недоліки ботів для пошуку фільмів:

- Обмежений функціонал – не всі мають детальні фільтрі.
- Залежність від API – якщо TMDb/IMDb не працює, бот теж "лагає".
- Не всі мають українську локалізацію.

Найпопулярніші API для кіно-ботів:

- TMDb (The Movie Database) – безкоштовний, багато мов, включаючи українську.
- IMDb API – менш доступний, але дуже точний.
- Кінопошук API – український сервіс, але обмежений функціонал.

Висновок: чи варто користуватися кіно-ботами?

- Так, якщо потрібен швидкий пошук без зайвих кліків.
- Так, якщо ви хочете отримувати рекомендації в одному додатку (наприклад, Telegram).
- Ні, якщо вам потрібні дуже детальні фільтри або архівні дані.

1.3 Огляд існуючих рішень

Cinemaking_Bot (Рисунок 1.3) – це інтерактивний кіно-помічник у Telegram, який допомагає швидко знаходити інформацію про фільми, отримувати

персоналізовані рекомендації та дізнаватися про новинки кіноіндустрії. Він ідеально підходить для кіноманів, критиків і звичайних глядачів, які хочуть отримувати точні дані без зайвих кліків.

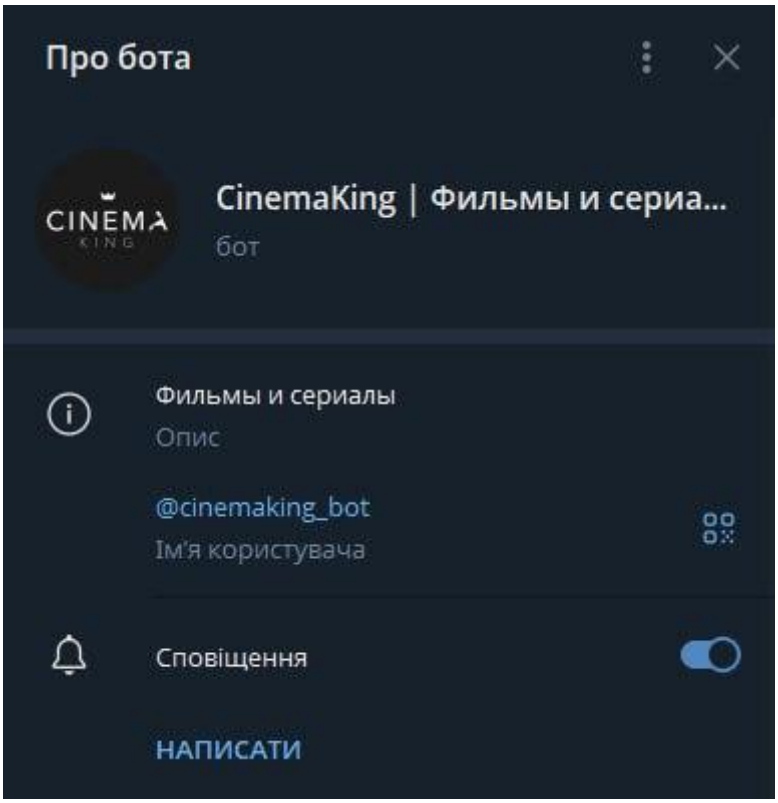


Рисунок 1.3 - Cinemaking_Bot

KinoSerialBot (Рисунок 1.4) - це інтелектуальний бот для пошуку фільмів, серіалів та аніме, який допомагає швидко знаходити цікавий контент, отримувати детальну інформацію та персоналізовані рекомендації. Він ідеально підходить як для звичайних глядачів, так і для справжніх кіноманів.

Розроблений мною Telegram-бот для пошуку фільмів має низку переваг порівняно з іншими подібними сервісами. Ось основні конкурентні переваги:

- Локалізація та зручність для українських користувачів.
- Повністю україномовний інтерфейс (на відміну від більшості ботів, які працюють лише англійською або російською)

- Фільми з українськими перекладами (використовується LANGUAGE=uk-UA для TMDb API)
- Зрозумілий UX – простий вибір через кнопки, а не складні текстові команди

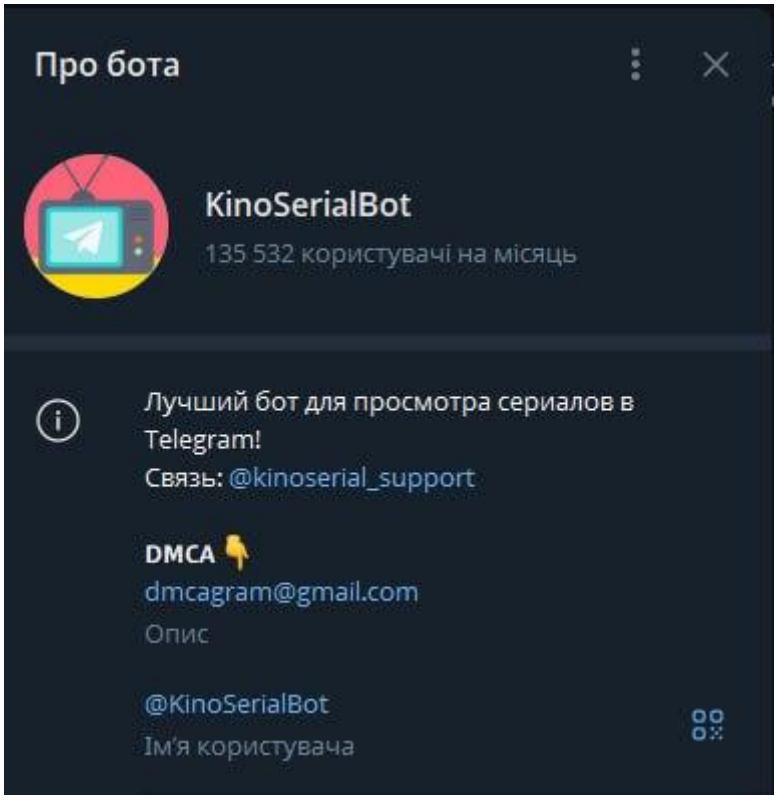


Рисунок 1.4 - KinoSerialBo

2. Актуальність та якість рекомендацій:

- Трендові фільми дня – оновлюваний список на основі популярності на TMDb
- Підбір за жанром – точні рекомендації (не просто випадкові фільми, а топ-5 за популярністю)
- Рейтинги від TMDb – реальні оцінки глядачів, а не рекламні підбірки

3. Швидкість та мінімалістичний дизайн:

- Миттєвий пошук – дані завантажуються прямо з TMDb без зайвих посередників

- Оптимізовано для мобільних пристроїв – гарна читабельність у Telegram
- Посилання на офіційні сторінки TMDb – можливість переглянути деталі фільму

4. Відсутність реклами та платних функцій:

- Немає нав'язливої реклами (на відміну від багатьох онлайн-кіноджерел)
- Повністю безкоштовний – не вимагає підписок або донатів
- Не збирає дані користувачів (немає реєстрації, трекінгу тощо)

Висновок: чому варто користуватись саме цим ботом?

Для українських користувачів – це один з небагатьох повністю локалізованих кіно-ботів.

Для тих, кому потрібен простий і швидкий пошук – без реєстрації, реклами та зайвих кліків.

Для фанатів кіно – тільки актуальні та перевірені підбірки на основі TMDb.

Якщо ви хочете отримувати найкращі фільми дня за 1 клік – цей бот ідеальний варіант!

1.4 Що таке Телеграм

Telegram - це багатоплатформовий месенджер, орієнтований на безпечні, швидкі та зручні повідомлення. Створений у 2013 році Павлом Дуровим після виходу з проекту соціальної мережі «ВКонтакте». Потужні цифрові засоби зв'язку, особливо в Східній Європі, Азії та пострадянському просторі.

Telegram - це хмарний месенджер - всі повідомлення, файли, діалогові вікна та мультимедійні дані зберігаються в хмарному сховищі Telegram і синхронізуються між пристроями користувачів.

Основні можливості Telegram:

Висока безпека та конфіденційність. Telegram використовує свій протокол шифрування MTProto для захисту даних під час передачі. Функція Secret Chat доступна за допомогою наскрізного шифрування, де повідомлення зберігаються тільки на пристроях учасників розмови. Можливість налаштування самознищення повідомлень через певну кількість часу.

Масштабованість і швидкість. Telegram оптимізований для швидкого обміну повідомленнями, навіть якщо підключення до Інтернету слабке. Серверна інфраструктура розподілена по всьому світу, зменшуючи затримки в доставці даних.

Підтримка великих спільнот та каналів. Telegram дозволяє створювати групи до 200 000 учасників і канали з необмеженими номерами користувачів, що робить його ефективною платформою для масової комунікації, інформації та маркетингу. Канал служить одностороннім потоком інформації (новини, анонси, блоги тощо).

Боти та автоматизація. Telegram надає спеціальний інтерфейс для Bot API, який дозволяє створювати чат-ботів для виконання різних функцій: обслуговування клієнтів, ігор, замовлень, опитувань, пошуку інформації тощо. Боти можуть взаємодіяти з користувачами через команди, кнопки, онлайн-запити та веб-хуки.

Підтримка мультимедіа та документів. Telegram дозволяє відправляти файли практично в будь-якому форматі, до 2 ГБ (у безкоштовній версії), включаючи текстові документи, відео, аудіо, архіви та зображення. Всі файли зберігаються в хмарі і можуть бути переглянуті або завантажені.

Крос-платформний. Telegram має клієнтські програми для всіх основних операційних систем: Android, iOS, Windows, MacOS, Linux та веб-версії (Telegram Web, WebK). Дані синхронізуються відразу на всіх пристроях користувача.

Телеграма в контексті розвитку чат-ботів. Однією з найважливіших і привабливих функцій Telegram є простота створення та інтеграції ботів.

Прийом і обробка текстових і голосових повідомлень;

Надсилання зображень, відео, кнопок, форм;

Підтримка інтерактивних меню та швидких відповідей

Взаємодія з зовнішніми API, базами даних та серверами.

Для розробників Telegram надає детальну документацію та SDK, а також підтримку обробки Web hook, онлайн-режим, ліцензування через Telegram Passport тощо.Messenger також не блокує сторонніх ботів, на відміну від деяких конкурентів, що робить його особливо привабливим для проектів, пов'язаних з автоматизованою взаємодією користувачів.

Підсумовуючи все вищевказане можу виділити, що основними перевагами Telegram є відкритість, доступність API, можливість створювати ботів, великі групи і канали, крос-платформи, підтримка великих файлів і більш високий рівень конфіденційності.Таким чином, Telegram – це не тільки зручний засіб спілкування, але й потужна платформа для побудови автоматизованих систем, зокрема, чат-ботів.

1.5 Діаграма прецедентів

Після ретельного аналізу вимог користувачів ми розробили діаграму прецедентів (Рисунок 1.5) для візуалізації функціональних вимог нашої системи. Ця діаграма є важливим інструментом для визначення взаємодії користувачів із системою.

Діаграма прецедентів - це графічне представлення в UML, де:

- Відображає взаємодію між учасником (користувачем/системою) і функцією системи
- Показати важливі випадки використання
- Визначення меж системи

UML (Unified Modeling Language) - стандартизована мова моделювання. Візуалізація, проектування та документація для програмних систем. Забезпечує єдиний спосіб опису архітектури системи. Містить графічні символи для різних аспектів дизайну

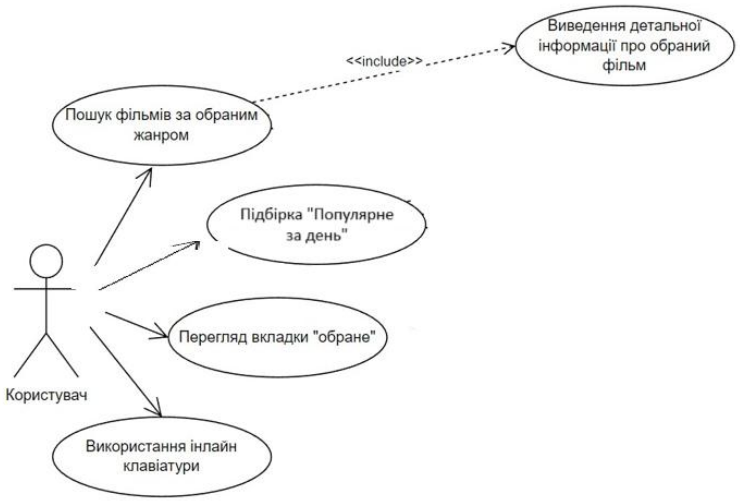


Рисунок 1.5 - Діаграма прецедентів телеграм-боту MovieBot

У нашому випадку малюнок вище особливо корисний для:

- Чітко визначена функція бота
- Поділ регулярних ролей користувача та адміністратора
- Визначення зовнішніх точок контакту системи TMDb API
- Планування майбутніх удосконалень

Цей інструмент дозволяє нам візуалізувати всі використання системи та забезпечити, щоб роботи, розроблені повністю відповідають очікуванням кінцевого користувача.

1.6 Обґрунтування теми проекту, актуальність

В рамках цього проекту стоїть завдання розробити чат-бота для пошуку фільмів в месенджері Telegram. Важливість цього рішення полягає в тому, щоб надати користувачам зручний інструмент для пошуку відповідних відео без необхідності витрачати багато часу на перегляд численних онлайн-ресурсів. Месенджер став невід'ємною частиною повсякденного життя, а чат-боти завоювали велику популярність як інструмент автоматизації взаємодії. Чат-боти дозволяють вирішувати безліч завдань, економлячи час і сили користувачів.

Навіщо спілкуватися в Telegram?

Telegram є одним з найпопулярніших месенджерів і активно використовується для спілкування і взаємодії з різними сервісами через роботів. Створення чат-бота за допомогою цього месенджера - розумний вибір, оскільки він надає широкий спектр можливостей для розробки роботів і велику аудиторію користувачів для широкого охоплення.

Головне завдання чат-бота - допомогти користувачам знайти фільми, які вони зараз дивляться. Користувачі можуть вводити запити в чат-бот і отримувати відповіді за допомогою зручного інтерфейсу. Алгоритми чат-ботів швидко аналізують запити, знаходять потрібні відео та надають інформацію, яку можна використовувати для перегляду відео на потрібних платформах.

Опис функції чат-бота:

Пошук фільмів за типом: Користувачі можуть вибрати, які типи фільмів їх цікавлять. Роботи шукають фільми за вказаним жанром, що дозволяє користувачам швидко знаходити фільми для перегляду.

Пошук за допомогою вбудованої клавіатури: Для зручності користувача робот використовує вбудовану клавіатуру для швидкого пошуку фільмів, жанрів або інших параметрів. Це забезпечує інтуїтивно зрозумілий процес взаємодії з роботами.

Щоб додати або видалити фільм з улюбленого: Користувачі можуть додати фільм до списку обраних для подальшого перегляду та видалити його, якщо він передумає або знайде інший фільм.

Користувачі можуть отримати пряме посилання на сайт.

Цей чат-бот поєднує в собі кілька унікальних функцій, які відрізняють його від інших подібних сервісів. Він дозволяє легко і швидко знаходити відео на різних платформах без ручного перегляду сайтів. Користувачі отримують всю необхідну інформацію безпосередньо в месенджері, зменшуючи потребу в сторонніх сервісах або інших додатках.

Чат-бот доступний на всіх поточних операційних системах, що дозволяє використовувати його на будь-якому пристрої, незалежно від платформи. Це

забезпечує зручність і доступність для користувачів, які використовують різні гаджети.

Стрімінгові сервіси набирають популярності: сьогодні велика кількість людей використовують потокові сервіси для перегляду фільмів і серіалів. Однак пошук фільмів на багатьох платформах може бути трудомістким процесом. Як інструмент автоматизації цього процесу, чат-боти полегшують життя користувачам.

Поліпшення доступності технологій: Використання миттєвих повідомлень і чат-ботів стало повсякденним явищем. Додавання нових функцій до цього популярного інструменту робить його більш корисним і доступним для широкої аудиторії.

Необхідність зручних інструментів для пошуку фільмів: користувачі часто стикаються з проблемою вибору фільмів з широкого спектру контенту на різних платформах. Чат-бот, який допоможе автоматично знайти тип фільму або інші параметри, буде високо оцінений завдяки зручності та економії часу.

В результаті проект не тільки має актуальність, але і має значний потенціал для поліпшення користувацького досвіду при виборі фільмів для перегляду в потокових сервісах.

2 ЗАСОБИ РЕАЛІЗАЦІЇ ПРОЕКТУ

2.1 Середовище розробки PyCharm

PyCharm – це кросплатформне, потужне і розумне інтегроване середовище розробки, створене компанією JetBrains. Цей інструмент надає все, що вам може знадобитися для комфортної роботи з мовою Python, починаючи від автодоповнення коду і закінчуючи потужним відладчиком та інструментами рефакторингу.

Чому ми обрали PyCharm для розробки Telegram-бота:

Інтелектуальне автодоповнення коду. PyCharm надає підказки при написанні коду (автокомпліт), що значно прискорює роботу. Наприклад, під час імпорту бібліотек або виклику методів з telegram.ext ви швидко бачите доступні функції та їх опис.

Зручна навігація по проєкту. Можна швидко переходити між файлами, класами, функціями, що дуже корисно, коли код стає більшим. Це економить час і зменшує кількість помилок.

Вбудований відлагоджувач (debugger). PyCharm має зручний інтерфейс для запуску коду в режимі налагодження — це дозволяє «крок за кроком» переглядати, що відбувається в програмі, перевіряти змінні, ловити помилки.

Підтримка віртуальних середовищ. Із PyCharm легко створити або підключити venv, що дозволяє ізольовано встановлювати потрібні бібліотеки (наприклад, python-telegram-bot, requests). Це захищає проєкт від конфліктів з іншими проєктами.

Інтеграція з Git та GitHub. Для збереження проєкту в репозиторії, спільної роботи чи резервного копіювання — PyCharm має вбудовану підтримку Git, що дозволяє коміти, пуші, перегляд історії змін прямо з середовища розробки.

Вбудована підтримка Python. PyCharm створений спеціально для Python. Він одразу підтримує перевірку синтаксису, підсвітку помилок, типізацію, форматування PEP8, підказки щодо кращих практик програмування.

Можливість встановлення плагінів. PyCharm дозволяє розширювати функціональність (наприклад, встановити плагіни для підтримки Markdown, JSON viewer, Docker, API tester тощо).

Зручне тестування коду. Із PyCharm ви можете запускати частини коду, окремі функції або повні тести без зайвих складнощів.

Як саме це допомогло в Telegram-боті:

- Швидка інтеграція бібліотеки python-telegram-bot.
- Комфортна робота з мережевими запитами до TMDb (requests).
- Швидка перевірка результатів відповіді API через інтерактивний режим або debug.
- Швидка обробка регулярних виразів (filters.Regex) завдяки підсвітці.
- Зручна робота з багатьма callback-функціями, що типово для бота.

PyCharm — ідеальне середовище для Python-проєктів будь-якої складності. У випадку створення Telegram-бота воно забезпечує:

- стабільну роботу,
- швидку розробку,
- мінімізацію помилок,
- комфорт під час налагодження.

Це дозволило нам ефективно створити функціонального Telegram-бота з інтеграцією до TMDb API, з дотриманням усіх стандартів Python-розробки.

2.2 Python

Python - це динамічна, інтерпретована, об'єктно-орієнтована мова програмування зі строгими динамічними типами. Заснована в 1990 році голландським програмістом Гвідо ван Россумом. Python - це універсальна мова, яка дозволяє писати зрозумілий і добре структурований код. Через свою простоту програми Python часто набагато коротші, ніж аналоги, написані на інших мовах програмування.

Python - це багатоплатформна мова, де програми можуть працювати на різних операційних системах без зміни коду. Іншою перевагою мови є розширена бібліотека за замовчуванням, встановлена на Python. Він містить нестандартні модулі для операційних систем, мереж, баз даних, файлів у різних форматах, створення графічних інтерфейсів тощо.

Python підходить для написання невеликих скриптів і створення складних програмних систем.

Можливим недоліком мови є відносно низька швидкість виконання коду. Оскільки Python не є компільованою мовою, його код спочатку перетворюється на байт-код, а потім виконується інтерпретатором. Через це програму Python можна порівняти з програмою, подібною до C. Аналогії, написані на таких мовах, працюють повільно, однак через потужність сучасних комп'ютерів це рідко стає ключовою проблемою. Крім того, модулі, написані на C або C++, можуть бути підключені до Python, що забезпечує високу продуктивність в ресурсоємних частинах програми.

2.3 Бібліотеки для написання телеграм-бота

Бібліотеки в Python в икористовуються для спрощення розробки додатків та проектів. Також вони дозволяють значно пришвидшити процес розробки застосунків. В бібліотеках знаходяться вже готові рішення тих чи інших задач. Адже написання всього коду з нуля без використання бібліотек займе велику кількісьть людського ресурсу. Бібліотеки в свою чегру поділяться на три типи:

- стандартні;
- сторонні;
- власні.

В цьому проекті я використаю лише стандартні та сторонні бібліотеки .

2.3.1 Бібліотека logging

Модуль logging є частиною стандартної бібліотеки Python і призначений для логування (запису) подій, повідомлень та помилок під час виконання програми. Він допомагає відстежувати роботу програми, діагностувати проблеми та аналізувати поведінку коду.

Ключові можливості:

- Рівні логування. Підтримує різні рівні важливості повідомлень: DEBUG, INFO, WARNING, ERROR, CRITICAL.
- Форматування. Можливість налаштувати формат виводу (дата, час, рівень, повідомлення тощо).
- Обробники (Handlers). Логи можна записувати у різні місця: консоль, файли, мережеві ресурси, email тощо.
- Фільтрація. Можна фільтрувати повідомлення за рівнем або іншими критеріями.
- Гнучкість. Підтримує ієрархію логерів, що дозволяє налаштовувати логування для окремих модулів.

2.3.2 Бібліотека requests

Бібліотека requests є популярним інструментом для відправлення HTTP-запитів у Python. Вона спрощує роботу з веб-API, завантаження даних із мережі та взаємодію з серверами.

Ключові можливості:

- Простий синтаксис. Інтуїтивно зрозумілі методи для роботи з HTTP (GET, POST, PUT, DELETE тощо).
- Підтримка параметрів запиту. Додавання параметрів URL, заголовків (headers), даних форми (form data), JSON тощо.

- Обробка відповідей. Легкий доступ до статус-коду, заголовків, вмісту відповіді (текст, JSON, бінарні дані).
- Сесії та cookies. Підтримка збереження сесій для ефективного управління cookies та авторизацією.
- Обробка помилок. Вбудовані винятки для мережевих проблем (наприклад, ConnectionError, Timeout).

2.3.3 Бібліотека python-telegram-bot

Бібліотека python-telegram-bot (також відома як telegram або python-telegram-bot) призначена для створення ботів у месенджері Telegram. Вона надає API для взаємодії з Telegram Bot API.

Ключові можливості:

- Асинхронна підтримка. Версія 20.x+ повністю асинхронна (на основі asyncio).
- Різноманітні компоненти. Підтримка клавіатур (InlineKeyboard, ReplyKeyboard), команди, обробники повідомлень, CallbackQuery тощо.
- Робота з типами контенту. Обробка текстів, фото, аудіо, документів, голосових повідомлень, стикерів.
- Розширені функції. Webhook-підтримка, пагінація, чат-менеджмент (бан, кік, адміністрування).
- Документація та спільнота. Детальні приклади та активна спільнота.

3 ОПИС РОЗРОБКИ ЧАТ-БОТУ

3.1 Процес підготовки до розробки чат-боту

Першим і найголовнішим етапом розробки мого проекту є створення власного токена для ботів в телеграм, через вбудований інструмент @BotFather (Рисунок 3.1) .

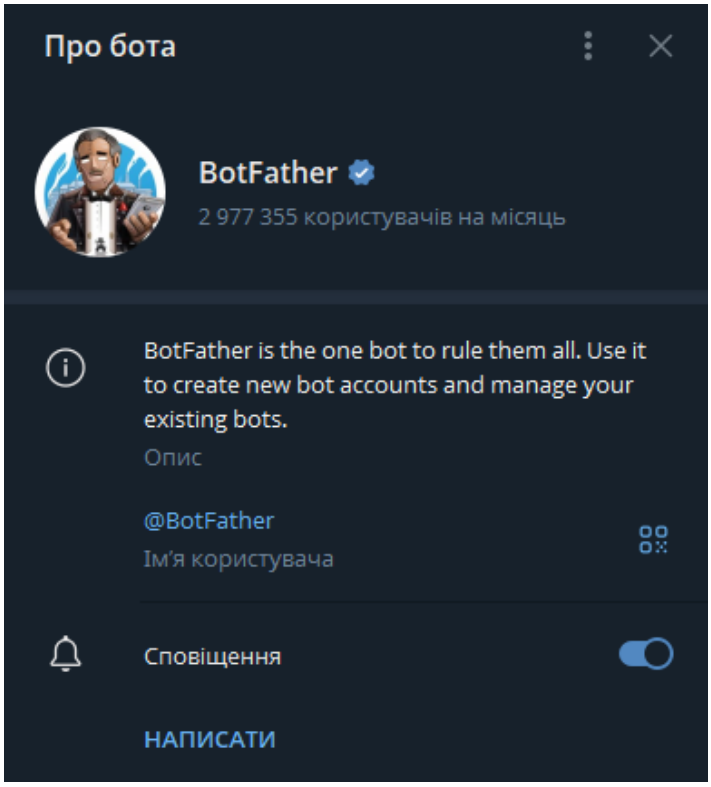


Рисунок 3.1 – BotFather

Це спеціальний майстер у Messenger, який допомагає користувачам створювати, налаштовувати та керувати власними віртуальними помічниками. Він забезпечує зручний інтерфейс та набір інструментів, які навіть ті, хто не має глибоких технічних знань, можуть легко запускати та змінювати роботів відповідно до їхніх потреб. Його можна метафорично назвати «батьком» всіх ботів Telegram, адже саме через нього відбувається більша частина процесу створення і управління ботами в цьому месенджері.

Для того щоб створити токен через BotFather я спочатку запустив додаток Telegram (можна як на смартфоні, так і на комп’ютері). У пошуку ввів @BotFather і вибрав відповідного користувача з синім значком перевірки.

Далі треба натиснути кнопку Start або ввести команду /start, щоб активувати діалог із BotFather. (Рисунок 3.2)

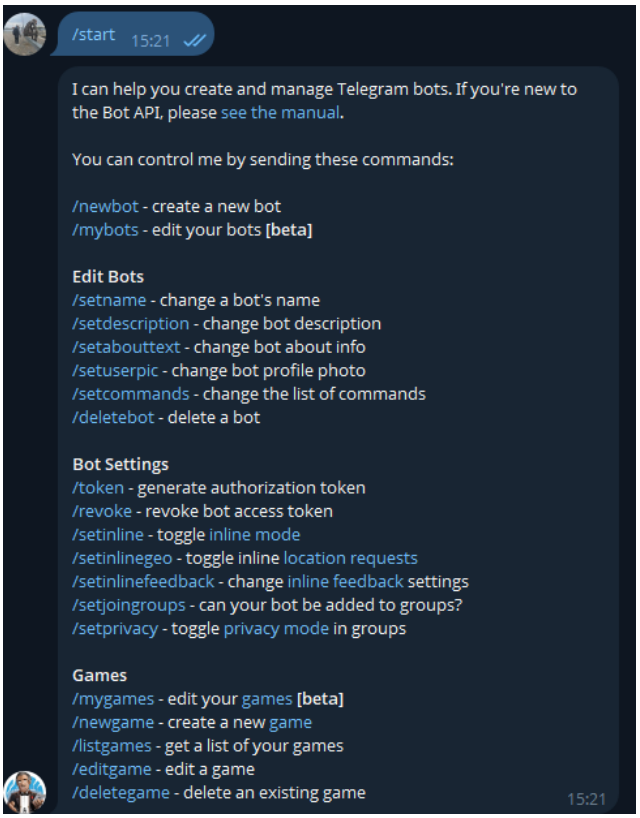


Рисунок 3.2 – Початок роботи з асистентом

Користуючись інструкцією яку надіслав асистент я використав функцію /newbot, яка відповідає за створення нового бота. (Рисунок 3.3)

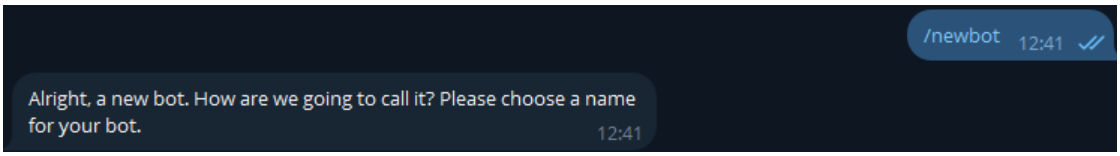


Рисунок 3.3 – Введення команди /newbot

Після цього мені потрібно було вказати назву мого боту. BotFather відповідає проханням вказати ім’я бота (name) — це повне ім’я, яке буде відображатися у профілі бота. (Рисунок 3.4)

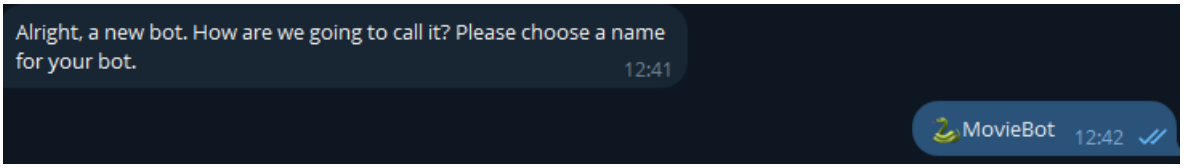


Рисунок 3.4 – Присвоєння назви

Далі слідкуючи інструкції мені потрібно було створити коротке імя боту. В подальшому воно буде використуватись для пошуку розробленого мною бота серед всіх доступних ботів в систему телеграму. Коли асистент попросить вказати користувацьке ім’я (username) для бота — воно повинне бути унікальним, завершуватися на bot і складатися лише з латинських літер. (Рисунок 3.5)

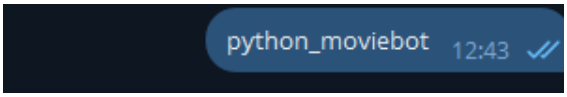


Рисунок 3.5 – Присвоєння id

Після успішного виконання дій які були згадані раніше я дібрався до завершального етапу створення бота. BotFather надіслав мені повідомлення (Рисунок 3.6) з підтвердженням, яке містило:

- Назву бота
- Користувацьке ім’я
- API токен (HTTP API Token)

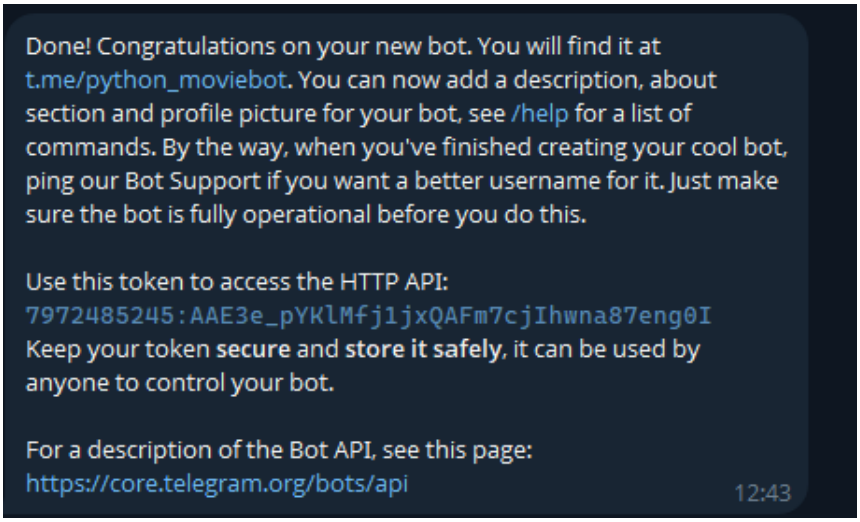


Рисунок 3.6 – Отриманий токен

Токен (від англ. token - «токен», «маркер», «знак») - це унікальний рядок символів, який діє як ключ для доступу до певного ресурсу, послуги або функції. У контексті програмування та веб-технологій токен є засобом аутентифікації та авторизації.

Іншими словами, токен — це "цифровий пропуск", який повідомляє системі: "Я маю дозвіл на доступ".

Токени можна використовувати в багатьох сферах програмування, таких як:

- Telegram-ботах;
- веб-розробці (токени доступу, авторизації);
- мобільних додатках.

У веб-розробці токени найчастіше використовують для авторизації користувачів. Наприклад, коли користувач входить на сайт, сервер генерує JWT (JSON Web Token) і передає його клієнту. Потім цей токен надсилається в заголовку кожного HTTP-запиту. Це використвується для того щоб підтвердити що користувач має доступ до певних налаштувань (наприклад до зміни даних профілю та налаштувань профілю відповідно).

Також ми використовуємо токени для роботи з API. При використанні зовнішнього API(наприклад коли ми використовуємо Google Maps, OpenAI,

YouTube Data API), ми завжди використовуємо API-токен. Таким чином сервіс має можливівсть відстежити, хто саме робить ті чи інші запити, скільки саме запитів було надіслано та чи взагалі користувач має право на такі дії.

Також токени використовуються у мобільних додатках. Насамперед це використвується для забезпечення нормальної авторизації користувачів. А точніше щоб користувачі мали змогу залишатися в системі наввіть після того як додаток було закрито. Тобто користувачам не потрібно буде постійно вводити логін і пароль для повторної авторизації.

Також можна використувати для забезпечення більш безпечних умов передавання даних мід клієнтом та сервером.

При розробці дипломної роботи я також використовував токен. Який у свою чергу я отримав через телеграм асистента BotFather. Токен для телеграм ботів має вигляд унікального довгого рядку з використанням символів (літер, цифр та знаків). Детальніше можна розглянути на рисунку 3.6.

За допомогою цього токена ми отримали доступ до Telegram Bot API. Цей дозвіл надає доступ до надсилання повідомлень та взаємодіяти з ботом. Таож тепер розроблений бот матиме змогу виконувати певні функції, такі як, обробляти команди та відповідати на запити та інші.

В подальшому я використовуватиму цей токен в коді бота. Це було зроблено для того щоб бот був постійно підключений до одного акаунту. Адже в подальшому буде створено єдиний головний акаунт який буде адмініструвати і вносити зміни в роботу розробленого мною бота.

Важливим моментом також є те що цей токен нівякому разі не можна передавати третім особам. Адже таким таким та особа яка його отримає матитме можливість використовувати бота з повним пакетом адміністрування. Тобто вносити зміни в роботу розробленого бота.

3.2 Процес розробки чат-боту

Розробку чат-боту (рисунок 3.7) можна назвати цікавим та багатоетапним

процесом. В ньому поєднано багато різних по складності і уяві процесів. Тут одночасно поєднано і програмування і дизайн, для того щоб зробти інтерфейс з яким користувач буде працювати насамперед. Ачого варта лише інтеграція зі сторонніми сервісами.

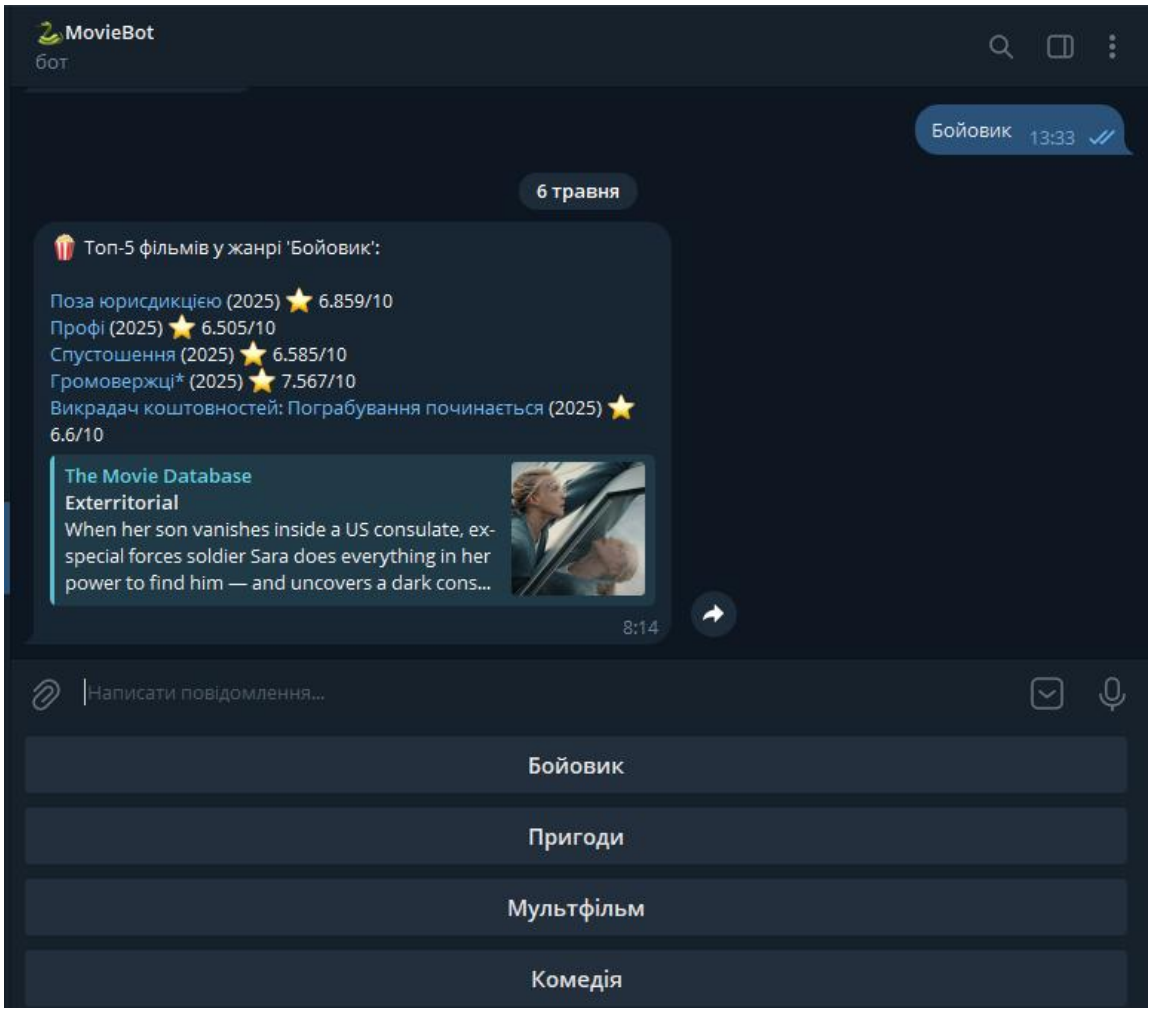


Рисунок 3.7 – Готовий чат-бот

Розглянемо покроковий план згідно якого я працював над проектом.

Визначаємо цілі та функціонал чат-боту. Основна мета боту ? Цільова адиторія та на якій платформі він буде функціонувати.

Обираємо технології та інструменти які допоможуть у розробці даного боту. Потрібно визначитись із мовою програмування. В моєму випадку я використовуватиму python. Після визначення мови я знайшов бібліотеки які допоможуть мені у використання даного проекту.

Проектування архітектури займає також значну нішу в роботі з проектом.

Аже для правильної роботи кожної функції нам потрібно схема згідно якої весь наш проект буде працювати. Для цього використовуються блок-схеми. За допомогою цього етапу я склав процедури які відбуватимуться при певних технічних та людських факторах. Наприклад коли користувач введе неправильно якусь певну команду чи надасть запит який чат бот не зможе обробити.

І основний етап написання самого коду. В ньому обєднуються всі попередні етати розробки. Адже саме тут починається сам процес розробки проекту. Саме на цьому етапі потрібнно об’єднати ідею і інструменти які необхідні для реалізації чат-боту.

Тестування також важливий момент який одразу ж починається після написання основного коду. Адже саме на цьому етапі починається налагодження всіх функцій для їх коретної роботи . Впровадження нових функцій також відбуввається саме після цього етапу .

І завершальний етап це deployment – збірка всього проекту в одне ціле і зпуск його до роботи.

3.2.1 Підключення бібліотек

Для того щоб розробити даний проект треба використати певні бібліотеки. Кожна з яких виконуватиме важливу і значну роль в роботі всього проекту . (Рисунок 3.7)

Бібліотеки котрі були використані при розробці даного проекту:

- logging - для відображення подій бота
- requests - для HTTP-запитів до TMDb API
- telegram та telegram.ext - основні бібліотеки для створення Telegram-бота

Для роботи з цими бібліотеками потрібно їх встановити. Щоб це зробити нам потрібно встановити пакети з бібліотеками через командний рядок. Для цього треба

використати команду pip install. Ця команда допомагає нам встановити пакет із розширенням тієї бібліотеки яка нам потрібна.

3.2.2 Конфігурація API

Бот API надає можливість підключити бота до системи Telegram. Telegram-боти — це звичайні облікові записи, але для їх налаштування не потрібен додатковий номер телефону. Керувати цими ботами можна за допомогою HTTPS-запитів. Однак існує два методи отримання та надсилання даних.(Рисунок 3.8)

```
import logging
import requests
from telegram import Update
from telegram.ext import (
    Application,
    CommandHandler,
    MessageHandler,
    ContextTypes,
    filters
)
```

Рисунок 3.8 – Імпортовані бібліотеки

Long Polling та Webhook — це методи, які дозволяють отримувати оновлення від бота:

- Перший метод (Long Polling) періодично запитує сервери Telegram на наявність оновлень через певні проміжки часу.
- Другий метод (Webhook) передбачає, що сервер сам сповіщає додаток про нові оновлення.

Щоб керувати ботом за допомогою Telegram Bot API, потрібно отримати спеціальний токен, який є унікальним для кожного бота. Його можна отримати, зареєструвавши нового бота в особливому чат-боті «BotFather».

Приклад діаграми згідно якої працює зв'язок між користувачем і результатом який прагне побачти користувач через API можна побачти на (рисунку 3.9).

```
# Налаштування | API запитів  з сайту TMDb
TMDB_API_KEY = "524173086b8f652025c99d8d4136dbdc"
LANGUAGE = "uk-UA"  # Українська мова
```

Рисунок 3.9 – API

Для того щоб наш бот міг надсилати та отримувати дані або запити на сайт ми повинні отримати ключ API. Цей ключ можна отримати з сайтів де розміщені відеоматеріали або будь-яка інша інформація яка нас цікавить. В даному випадку ми повинні оримати з сайту TMDb повний перелік фільмів що були завантажені на нього. Схему його роботи ми можемо побачити на діаграмі (Рисунок 3.10)

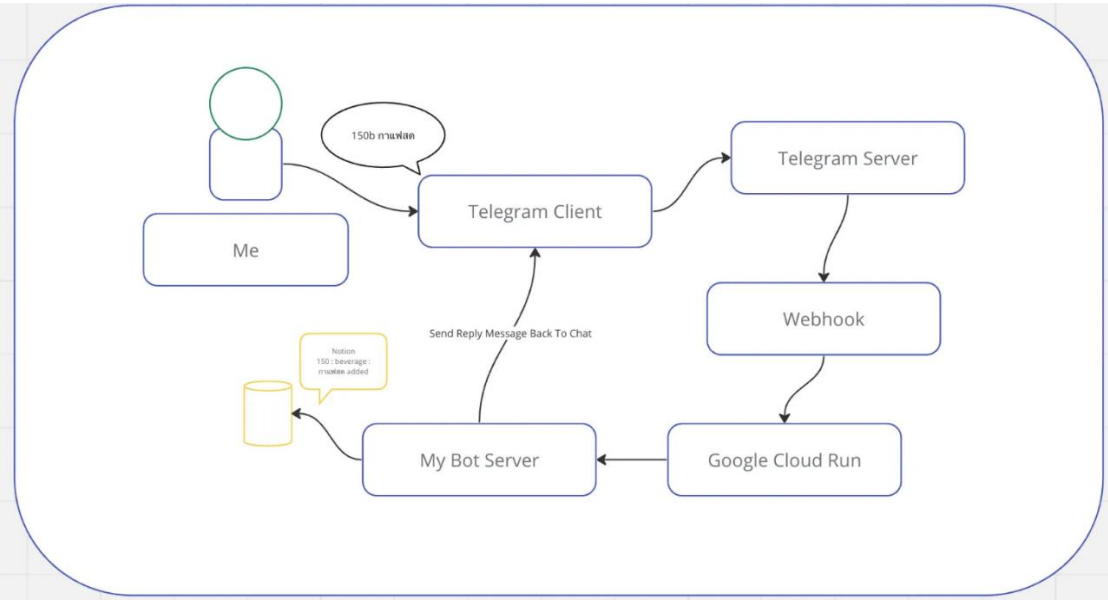


Рисунок 3.10 – Діаграма Bot API

Також за допомогою функції адміністрування ми визначили мову інтерфейсу. В даному випадку нас цікавить україномовна частина сайту. Тобто це означає що в

подальшому ми будемо працювати лиже з українською мовою на сайті і при ориманні запитів в нашому телеграм боті.

3.2.3 Налаштування логування

Для відстеження нормальної роботи потрібно налаштувати логування. Тобто режим де відображатиметься правильність роботи нашого боту (Рисунок 3.11).

```
# Налаштування логування для запитів для роботи  з нашим сайтом
logging.basicConfig(
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s", level=logging.INFO
)
logger = logging.getLogger(__name__)
```

Рисунок 3.11 – Логування

Це використовують для декількох важливих цілей:

Для того щоб налагодити та виявити помилки. Фіксація помилок що виникли під час роботи боту. Та зберігання всіх дій які виконав користувач. Робиться це насамперед для того щоб ми могли проаналізувати у випадку виникнення якоїсь проблеми, внаслідок чого вона виникла і на якому етапі.

Моніторинг активності користувачів. Тобто ми можемо бачити перелік найпопулярніших запитів які відправляють користувачі. І в подальшому використати цю інформацію для налаштування більш клієнтоорієнтованого інтерфейсу .

Аналітика поведінки користувачів. Тобто ми можемо збирати якусь статистику про команди які найчастіше використовуються нашими користувачами. І в разі чого ми одразу бачимо де виникають труднощі і маємо більше шансів якнайшвидше виправити помилки. Також завдяки цим даним ми можемо покращувати нашого бота.

Безпека. Це найголовніше при роботі з сторонніми ресурсами. При виявлені підозрілих дій, це одразу фіксується і ми може виконувати певні дії. Це актуально

коли користувач може проводити певні атаки на бота, розсилати спам та вводити некоректні запити. Також ми можемо у випадку виявлення незрозумілої активності для притягнення до відповідальності некоректних користувачів використати їхню IP-адресу яка також буде зберігатись в наших логах.

При збоях можна відновити роботу. Адже логи нам допомагають зрозуміти що сталося перед тим як наш бот ліг.

3.2.4 Обробник команди /start

Команда /start - це одна з основних частин логіки чат-бота, яка створює перше враження, налаштовує структуру навігації та забезпечує зручну взаємодію з користувачем з перших хвилин використання чат-бота.

Проект реалізує обробник команди /start, який служить точкою введення для взаємодії користувачів з чат-ботом. Основна мета цієї команди - привітати користувача і надати йому інтуїтивно зрозуміле головне меню для подальшої навігації. При додаванні функціоналу бота було запроваджено асинхронну функцію start(), яка є обробником команди /start. Ця команда викликається по замовчуванню, щойно користувач вперше запускає бота.

3.2.5 Отримання трендових фільмів

У межах проєкту з розробки чат-бота для пошуку фільмів було реалізовано окрему асинхронну функцію fetch_trending_movies(), яка відповідає за отримання списку найпопулярніших фільмів за поточний день.

Ця функція є критично важливою для формування актуального та привабливого контенту для користувача. Використання функції fetch_trending_movies, ця функція зазвичай використовується для:

Автоматичного отримання даних трендових фільмів дня за допомогою зовнішнього API (The Movie Database-TMDb).

Автоматичного оновлення актуального контенту у бота, без потреби стороннього втручання.

Надання користувачам найактуальніших рекомендацій фільмів, які зараз користуються найбільшим попитом.

Після ознайомлення з метою функції fetch_trending_movies(), потрібно перейти до не менш важливого кроку – сформування правильного URL-адресу запиту. Який в свою чергу буде містити в собі всю потрібну інформацію для авторизації та фільтрації результатів. Від цього пункту залежить, чи зможе бот отримувати найактуальнішу інформацію про фільми.

Тому наступним кроком у створенні чат-боту є формування повної адреси запиту до TMDb API, яку в свою чергу буде передано у функцію requests.get()(Рисунок 3.12).

```
url = f"https://api.themoviedb.org/3/trending/movie/day?api_key={TMDB_API_KEY}&language={LANGUAGE}"
try:
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()["results"][:10]  # Перші 5 фільмів
```

Рисунок 3.12 – Функція requests.get()

Тож перейдемо до покрокового формування URL-запиту до TMDb API:

Використаємо офіційну кіноплатформу The Movie Database (TMDb), яка надає API для доступу до великої і актуальної бази даних фільмів.

Адреса запиту створюється динамічно з використанням декількох змінних.

TMDB_API_KEY – це унікальний ключ API, який надається розробнику після реєстрації на сайті TMDb.

LANGUAGE – це мова, якою буде повертатися інформація отримана на платформі TMDb (наприклад, "uk-UA" як в нашому випадку, або "en-US" для англійської).(Рисунок 3.8)

Після створення коректної URL-адреси нам потрібно зробити наступний крок. Виконання HTTP-запиту (метод GET) – це один основний метод протоколу

HTTP, який застосовується зазвичай для отримання даних із веб-сервера. У нашому випадку GET-запит використовується для взаємодії з TMDb API.

В моєму боті ця взаємодія відбувається за допомогою бібліотеки requests.

Після виконання всіх вище вказаних кроків нам потрібно перевірити статус відповіді, якщо відповідь має статус-код 200, це означає, що запит був виконаний успішно, і в такому разі нам повернеться лише перші 5 результатів з отриманого списку трендових фільмів. Але у розробці з використанням сторонніх API надзвичайно важливо передбачити різні ситуації, наприклад коли щось йде не за планом (сервер не доступний, відсутність інтернету, API-ключ не дійсний, або неочікуваний формат відповіді від сервера).

Саме через це я використав конструкцію try-except. У випадку помилки, або не успішної відповіді, функція повертає None, що в свою чергу дозволяє коректно обробити дану ситуацію ще на рівні функції.(Рисунок 3.13)

```
try:
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()["results"][:10]  # Перші 5 фільмів
except Exception as e:
    logger.error(f"Помилка при отриманні топу фільмів: {e}")
return None
```

Рисунок 3.13 – Функція try-except

3.2.6 Робота з жанрами

Крім огляду трендових фільмів, користувачам бота надаєаться можливість сортувати фільми за жанром. Це дозволяє максимально зручно підібрати стрічку відповідну власним вподобанням (комедія, драма, трилер, бойовик тощо.)

Для реалізації цієї функції в моєму проєкті було створено декілька основних компонентів, одним з яких є обробник show_genres(), він відображає список у вигляді зручного меню. (Рисунок 3.14) У відповідь TMDb повертає список об’єктів.

```python
async def show_genres(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:  1 usage
    """Показати список жанрів  """
    genres = await fetch_genres()
    if genres:
        genre_names = [[genre["name"]] for genre in genres]
        await update.message.reply_text(
            "Оберіть жанр:",
            reply_markup={"keyboard": genre_names, "resize_keyboard": True}
        )
    else:
        await update.message.reply_text("На жаль, не вдалося завантажити жанри. Спробуйте пізніше.")
```

Рисунок 3.14 – Функція show_genres()

Наступним компонентом є функція fetch_genres(), вона формує URL-запит до TMDb API, за використанням endpoint genre/movie/list, що в свою чергу повертає повний список жанрів фільмів. (Рисунок 3.15)

```python
async def fetch_genres():  2 usages
    """Отримати список жанрів з TMDb"""
    url = f"https://api.themoviedb.org/3/genre/movie/list?api_key={TMDB_API_KEY}&language={LANGUAGE}"
    try:
        response = requests.get(url)
        if response.status_code == 200:
            return response.json()["genres"]
    except Exception as e:
        logger.error(f"Помилка при отриманні жанрів: {e}")
    return None
```

Рисунок 3.15 – Функція fetch_genres()

Також мною було використано метод fetch_movies_by_genre() .(Рисунок 3.16)

Це асинхронний метод який призначений для того щоб ми могли отримати фільми за заданим жанром з сайту TMDb.

Це реалізовано наступним чином:

- Спочатку потрібно викликати метод fetch_genres(). Ця функція повертає нам список жанрів які доступні на сайті TMDb. Адже на сайті кожен жанр є словником, тобто в нього є ключі з якимим ми можемо працювати id та name.

- Потім потрібно переіврити чи взягалі існують такі жанри на сайті. Якщо genres_data порожній або None функція одразу повертає None, що означає неможливість продовжити пошук без даних про жанри.
- Далі відбувається пошук ідентифікатора жанру (genre_id). За допомогою циклу for перебираються всі жанри зі списку genres_data. Якщо ім’я жанру (genre["name"]) збігається (незалежно від регістру) з введеним параметром genre_name, то зберігається його id у змінну genre_id. Після цього цикл переривається.

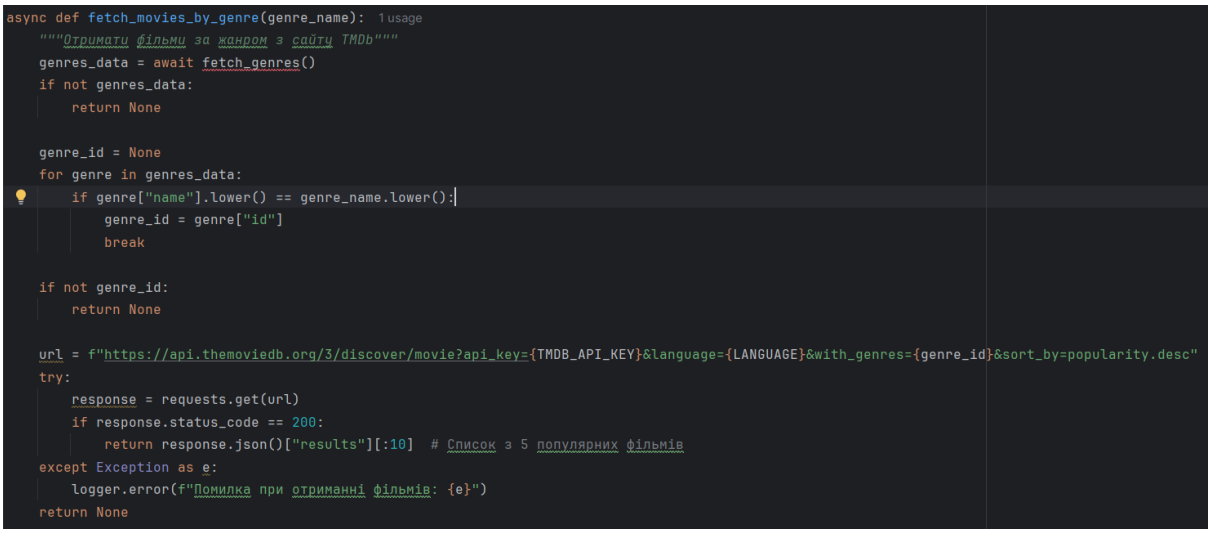


Рисунок 3.16 – Функція fetch_movies_by_genre()

Перевірка знайденого жанру Якщо genre_id так і не був знайдений (залишився None), функція повертає None. Це означає, що введений жанр відсутній у базі сайту TMDb.

Далі в нас формується запит API до сайту TMDb. Для цього в нас формується URL до сайту TMDb API, який вже в свою чергу фільтрує фільми за:

- обраним жанром (with_genres)
- мовою (language)
- сортуванням за популярністю (sort_by=popularity.desc)

Для реалізації цієї дії нам потрібні змінні: TMDB_API_KEY (ключ API) та LANGUAGE (наприклад, 'uk-UA' або 'en-US').

Наступний крок це відправка самого запиту та обробка відповіді від сайту. Якщо запит проходить успішно то нам видається список з популярних фільмів по жанру

У випадку якщо в нас виникла помилка ми запускаємо процес логування описаний раніше. (Рисунок 3.17)

```
except Exception as e:
    logger.error(f"Помилка при отриманні фільмів: {e}")
return None
```

Рисунок 3.17 – Помилка при отриманні фільмів

3.2.7 Головна функція

Одним з головних елементів коду який я написав для чат-боту є головна функція main(). Дана функція є головною точкою початку програми. Тобто вона організовує роботу всього різновиду функцій, методів та об’єктів що присутні в коді.

Основна задача цієї функції це організація простого інтерфейсу бота. Який в свою чергу має можливість реагувати на певні команди та текстові повідомлення що були написані користувачем.

Розглянемо детальніше принцип роботи даної функції:

- Запуск бота для подальшого його використання (Рисунок 3.18):
- Створення бота за допомогою функції Application.builder()
- Використовує наданий нам токен боту ("TELEGRAM_TOKEN") для авторизації його в системі телеграм.
- За допомогою методу build() відбувається завершальний етап налаштування конфігурації боту.

```
"""Запуск нашого боту """
application = Application.builder().token("7972485245:AAE3e_pYKlMfj1jxQAFm7cjIhwna87engOI").build()
```

Рисунок 3.18 – Ініціалізація боту

Система яка оброблятиме запити та повідомлення від користувачів(Рисунок 3.19):

```
application.add_handler(CommandHandler("start", start))
application.add_handler(MessageHandler(filters.Regex("^Топ фільмів дня$"), send_trending_movies))
application.add_handler(MessageHandler(filters.Regex("^Обрати за жанром$"), show_genres))
application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, send_movie_recommendations))
```

Рисунок 3.19 – Запуск обробників запитів

Для початку роботи з ботом потрібно використати команду /start. Дана команда просто запустить функціонал боту.

send_trending_movies () дана функція додає в наш боту кнопку «Топ фільмів дя» та реагує на її натискання користувачем. При натисканні звертається за допомогою API запиту на сайт TMDb і виводить список найпопулярніших фільмів на сьогодні.

show_genres() дана функція обробляє запит на вибір фільмів по жанрах. Показує список з доступних жанрів.

send_movie_recommendations() ця функція повинна обробляти всі інші текстові запити від користувачів які не є командами. Наприклад, жанр що обрав користувач. І після його обрання надсилає відповідні рекомендації.

Метод application.run_polling() є основним механізмом, який забезпечує нормальну роботу Telegram-бота. Цей метод забезпечує взаємодію з Telegram API. Основний принцип його роботи полягає в тому що бот постійно з періодом в декілька секунд надсилає запити до серверів Телеграму. Називається цей метод (Long Polling). На відміну від коротких інтервалів опитування, цей метод зменшує навантаження на сервер. В свою ж чергу Телеграм надає відповідь лише коли є якась оновлена інформація.

3.2.8 Метод Long Polling

Long Polling — метод який допомагає отримувати оновлення від сервера. Яка є чимось середнім між класичним методом Polling, який робить періодичні запити до сервера. Та WebSockets Який працює по принципу постійного з’єднання з

сервером. Ця технологія дуже часто використовується в Телеграм-ботах. А також у веб-розробці.

Розглянемо детальніше як працює даний метод. А точніше його принципи роботи:

Користувач за допомогою бота надсилає певний запити до сервера Телеграму і чекає на відповідь.

Сервер в свою чергу отримує цей запит, але не відповідає миттєво. Запит висить на сервері і залишається відкритим доти поки не з’явиться нове оновлення яке стосується саме його.

Якщо оновлення з’явилось сервер одразу передає відповідь боту .

Якщо ж оновлення, тобто відповіді на запит не має протягом певного часу то сервер поверне порожній запит і тоді доведеться надсилати новий запит.

Чим відрізняється від звичайного методу Polling? В звичайному методі Polling користувач повинен періодично запитувати в сервера. А при використаному мною методі Long Polling користувач повинен надіслати запит і чекати поки сервер не матиме певні дані для надання правильної відповіді на цей запит. В свою чергу звичайний метод таким чином перевантажує сервер через велику кількість запитів. Що в свою чергу створює велику затримку між відповідями. Але використаний мною Long Polling надсилає менше запитів на сервер, адже сервер відповідає лише в тому випадку коли в нього є відповідь а не кожен раз як застарілий метод Polling. Що і призводить до кращої роботи над запитами і відповідно меншої затримки на відповідь (Рисунок 3.20).

```
def main() -> None:  1 usage
    """Запуск нашого  боту  """
    application = Application.builder().token("7972485245:AAE3e_pYKlMfj1jxQAFm7cjIhwna87eng0I").build()

    application.add_handler(CommandHandler("start", start))
    application.add_handler(MessageHandler(filters.Regex("^Топ фільмів дня$"), send_trending_movies))
    application.add_handler(MessageHandler(filters.Regex("^Обрати за жанром$"), show_genres))
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, send_movie_recommendations))

    application.run_polling()
```

Рисунок 3.20 – Запуск бота

Long Polling — це простий і надійний спосіб отримувати оновлення в Telegram ботах без необхідності налаштовувати сервер. Він ідеально підходить для розробки та невеликих проектів.

ВИСНОВКИ

У результаті виконання роботи було розроблено телеграм-бота для пошуку фільмів. Розглянуто та проаналізовано існуючі рішення та підходи до розробки чат-ботів, виділено основні особливості їх розробки, а також технології та прийоми їх роботи.

Опрацьовано велику кількість інформації від передових видань у сфері розробок чат-ботів, надано загальну інформацію щодо розробки ботів для пошуку фільмів та наукових проектів, пов’язаних з даним типом роботи.

На основі проаналізованої інформації обрано оптимальний спосіб розробки. Дано розгорнуте порівняння та опис переваг саме такого підходу для створення чат-боту. Обрано платформу Telegram та технології для створення чат боту. Розроблено систему для пошуку фільмів яка має має зв'язок з сервером сайту TMDb.

Telegram-бот для пошуку фільмів є ефективним рішенням, що поєднує функціональність сучасних API (наприклад, The Movie Database або IMDb) із доступністю у повсякденному використанні. Такий бот може стати альтернативою складним вебсайтам чи додаткам, особливо для користувачів, які віддають перевагу мінімалістичному інтерфейсу та швидкій взаємодії.

З технічної точки зору, створення Telegram-бота дає змогу реалізувати навички роботи з API, асинхронного програмування, обробки запитів, логіки діалогів та баз даних.

Отже, розробка Telegram-бота для пошуку фільмів є актуальним напрямом, який поєднує потреби сучасного користувача, тенденції автоматизації сервісів, розвиток технологій бот-платформ, а також сприяє удосконаленню навичок у сфері програмної інженерії.

ПЕРЕЛІК ПОСИЛАНЬ

1. Telegram Bot API. Офіційна документація. – https://core.telegram.org/bots/api

2. Python Telegram Bot Library. Документація. – https://docs.python-telegram-bot.org

3.The Movie Database (TMDb) API Documentation –https:/ /developer.themoviedb.org/docs

4. RapidAPI. Платформа для доступу до сторонніх API (IMDb, Netflix і подібні) – https://rapidapi.com

5. Python 3. Офіційна документація. – https://docs.python.org/3/

6. Heroku. Хостинг Telegram-ботів. Офіційна документація – https://devcenter.heroku.com/categories/reference

7. GitHub. Платформа для зберігання та спільної роботи над кодом – https://github.com

8. Stack Overflow. Спільнота програмістів для вирішення технічних проблем – https://stackoverflow.com

9. Бондаренко С.В. Розробка Telegram-бота засобами Python // Молодий вчений. — 2022. — №5. — С. 33–37.

10. Куліш О.І. Telegram-боти як засіб автоматизації сервісів // Інформаційні технології і засоби навчання. — 2021. — №3 (83). — С. 119–125.

11. YouTube: Python Telegram Bot Tutorial (серія навчальних відео) – https://www.youtube.com/results?search_query=python+telegram+bot+tutorial

ДОДАТОК А

Вихідний код застосунку

```
import logging
import requests
from telegram import Update
from telegram.ext import (
    Application,
    CommandHandler,
    MessageHandler,
    ContextTypes,
    filters
)

# Налаштування  API запитів  з сайту TMDb
TMDB_API_KEY = "524173086b8f652025c99d8d4136dbdc"
LANGUAGE = "uk-UA"  # Українська мова

# Налаштування логування для запитів для роботи  з нашим сайтом
logging.basicConfig(
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
level=logging.INFO
)
logger = logging.getLogger(__name__)


async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    """використання команди старт /start"""
    menu_options = [["Топ фільмів дня"], ["Обрати за жанром"]]
    await update.message.reply_text(
        " 🎬 Вітаю у кіно-боті!\nОберіть опцію:",
        reply_markup={"keyboard": menu_options, "resize_keyboard": True}

    )


async def fetch_trending_movies():
    """створення підбірки фільмів популярних за день з сайту TMDb"""
    url =
f"https://api.themoviedb.org/3/trending/movie/day?api_key={TMDB_API_KEY}&lang
```

```
uage={LANGUAGE}"
    try:
        response = requests.get(url)
        if response.status_code == 200:
            return response.json()["results"][:10]  # Перші 5 фільмів
    except Exception as e:
        logger.error(f"Помилка при отриманні топу фільмів: {e}")
    return None


async def send_trending_movies(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> None:
    """Надсилає топ фільмів дня"""
    movies = await fetch_trending_movies()

    if not movies:
        await update.message.reply_text("На жаль, не вдалося завантажити топ фільмів.
Спробуйте пізніше.")
        return

    response = " 🍿 <b>Топ-5 фільмів сьогодні:</b>\n\n"

    for movie in movies:
        title = movie.get("title", "Невідома назва")
        year = movie.get("release_date", "")[:4]
        rating = movie.get("vote_average", 0)
        movie_id = movie.get("id", "")
        url = f"https://www.themoviedb.org/movie/{movie_id}"

        response += f"<a href='{url}'>{title}</a> ({year}) ⭐ {rating}/10\n"

    await update.message.reply_text(
        response,
        parse_mode='HTML',
        disable_web_page_preview=False
    )


async def fetch_genres():
    """Отримати список жанрів з TMDb"""
    url =
```

```
f"https://api.themoviedb.org/3/genre/movie/list?api_key={TMDB_API_KEY}&langua
ge={LANGUAGE}"
    try:
        response = requests.get(url)
        if response.status_code == 200:
            return response.json()["genres"]
    except Exception as e:
        logger.error(f"Помилка при отриманні жанрів: {e}")
    return None



async def show_genres(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
None:
    """Показати список жанрів  """
    genres = await fetch_genres()
    if genres:
        genre_names = [[genre["name"]] for genre in genres]
        await update.message.reply_text(
            "Оберіть жанр:",
            reply_markup={"keyboard": genre_names, "resize_keyboard": True}
        )
    else:
        await update.message.reply_text("На жаль, не вдалося завантажити жанри.
Спробуйте пізніше.")


async def fetch_genres():
    pass


async def fetch_movies_by_genre(genre_name):
    """Отримати фільми за жанром з сайту TMDb"""
    genres_data = await fetch_genres()
    if not genres_data:
        return None

    genre_id = None
    for genre in genres_data:
        if genre["name"].lower() == genre_name.lower():
            genre_id = genre["id"]
```

```
            break

    if not genre_id:
        return None

    url =
f"https://api.themoviedb.org/3/discover/movie?api_key={TMDB_API_KEY}&languag
e={LANGUAGE}&with_genres={genre_id}&sort_by=popularity.desc"
    try:
        response = requests.get(url)
        if response.status_code == 200:
            return response.json()["results"][:10]  # Список з 5 популярних фільмів
    except Exception as e:
        logger.error(f"Помилка при отриманні фільмів: {e}")
    return None


async def send_movie_recommendations(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> None:
    """Надсилає підбірку фільмів за жанром  популярних  на даний момент"""
    genre = update.message.text
    movies = await fetch_movies_by_genre(genre)

    if not movies:
        await update.message.reply_text("Не вдалося знайти фільми цього жанру.
Спробуйте інший жанр.")
        return

    response = f" 🍿  Топ-5 фільмів у жанрі '{genre}':\n\n"

    for movie in movies:
        title = movie.get("title", "Невідома назва")
        year = movie.get("release_date", "")[:4]
        rating = movie.get("vote_average", 0)
        movie_id = movie.get("id", "")
        url = f"https://www.themoviedb.org/movie/{movie_id}"

        response += f"<a href='{url}'>{title}</a> ({year}) ⭐ {rating}/10\n"

    await update.message.reply_text(
        response,
```

```
        parse_mode='HTML',
        disable_web_page_preview=False
    )


def main() -> None:
    """Запуск нашого  боту  """
    application =
Application.builder().token("7972485245:AAE3e_pYKlMfj1jxQAFm7cjIhwna87eng0I
").build()

    application.add_handler(CommandHandler("start", start))
    application.add_handler(MessageHandler(filters.Regex("^Топ фільмів дня$"),
send_trending_movies))
    application.add_handler(MessageHandler(filters.Regex("^Обрати за жанром$"),
show_genres))
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
send_movie_recommendations))

    application.run_polling()


if __name__ == "__main__":
    main()
```

ДОДАТОК Б

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально науковий інститут інформаційних технологій
Кафедра комп'ютерних наук

«Чат-бот для пошуку фільмів на доступних веб-ресурсах, за допомогою мови програ-мування Python»

Виконав: студент 4 курсу,групи КНД–42
Перепадя Олександр Сергійович
Керівник роботи: доцент кафедри,
кандидат технічних наук Василенко В.В.

Об`єкт, Предмет та Мета Дослідження

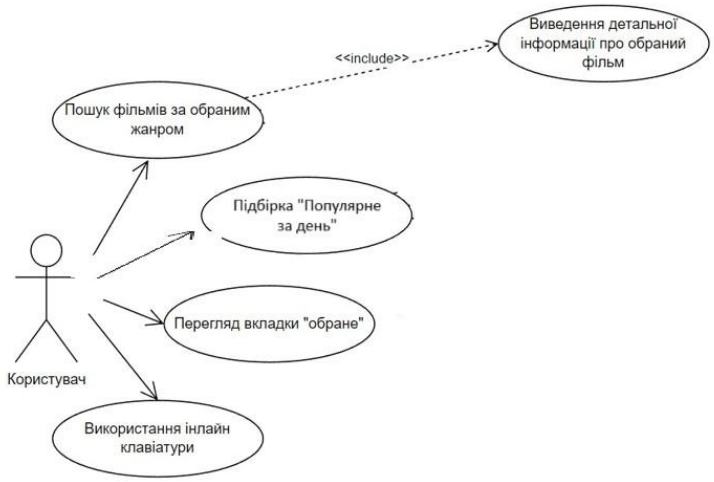
- ***Об`єкт дослідження*** – інформаційні системи автоматизованого надання даних користувачеві використовуючи середовище месенджерів, зокрема Telegram-боти як інструмент взаємодії людини з інформаційними сервісами
- ***Предмет дослідження*** – технології розробки Telegram-ботів, способи інтеграції з відкритими API баз даних та програмні засоби реалізації пошукових і діалогових функцій у месенджерах.
- ***Мета дослідження***– є розробка функціонального Telegram-бота для автоматизованого пошуку фільмів.

Технічне завдання

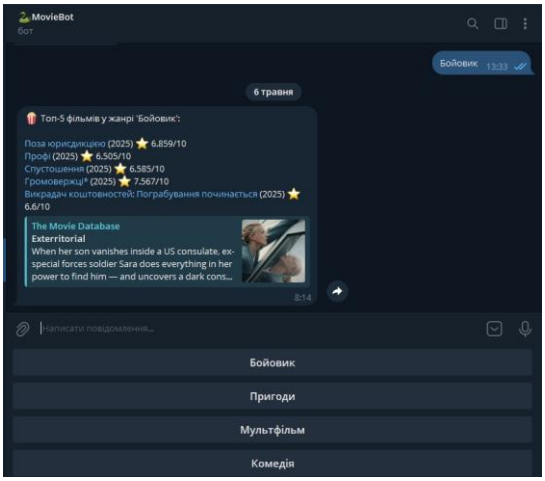
1. Україномовний інтерфейс.
2. Відображення доступних команд.
3. Зручний інтерфейс.
4. Показ популярних фільмів за день.
5. Пошук фільмів за жанром.
6. Доступ до бази даних фільмів за допомогою API запиту.
7. Посилання повинні бути функціональними .

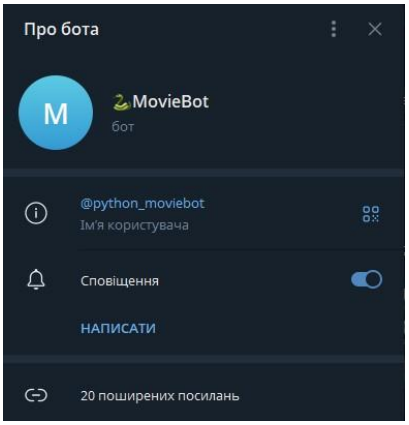
Програмне забезпечення

Діаграма прецедентів



Огляд чат-боту





Висновок

1. Було розроблено телеграм-бота для пошуку фільмів. Розглянуто та проаналізовано існуючі рішення та підходи до розробки чат-ботів, виділено основні особливості а також технології та прийоми їх роботи. Внаслідок чого було і розроблено цей проект враховуючи всі плюси і недоліки подібних проектів.
2. Даний проект розроблено чітко по завданню з дотриманям всіх вимог. В результаті чого було отримано додаткові знання про в сфері розробки чат-ботів на платформі Telegram , а також додатковий досвід розробки на Python.