Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА БОТА В TELEGRAM ДЛЯ ПІДБОРУ ФІЛЬМІВ ІЗ ВИКОРИСТАННЯМ GPT-МОДЕЛІ ВІД OPENAI
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-3
Шералієва Б. Д.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О. В.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Шералієвої Бібігул Давлаталіївної
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка бота в Telegram для підбору фільмів із використанням GPT-моделі від OpenAI

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 26 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мовна модель GPT з відкритим кодом OpenAPI, бібліотека Telegram.Bot, дані про фільми з TMDb API, мова програмування C#, база даних MongoDB, середовищє розробки Visual Studio Community 2022.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області та огляд існуючих рішень.
2. Проєктування, вибір формату та технологій застосунку для підбору фільмів.
3. Розробка бота в Telegram для підбору фільмів із використанням GPT-моделі від OpenAI.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми персоналізованих рекомендацій фільмів із використанням великих мовних моделей, постановка задачі, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-15.04.25	
4	Аналіз технічних засобів	16.04.25-20.04.25	
5	Проєктування структири застосунку	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____ доц. Тітова О. В._____
(підпис) (посада, прізвище, ініціали

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 70 с., 1 табл., 40 рис., 32 джерело.

ПІДБІР ФІЛЬМІВ, TELEGRAM-БОТ, ВЕЛИКІ МОВНІ МОДЕЛІ, OPENAI API, GPT, API TMDB, .NET, C#, MONGODB.

Об’єктом роботи є процес зручного надання користувачам рекомендацій щодо вибору фільмів.

Метою роботи є розробка бота, що здійснює зручний підбір фільмів відповідно до запитів користувача, його вподобань та додаткових фільтрів.

У ході роботи було розглянуто принципи функціонування великих мовних моделей та їх можливості для генерації відповідей на запити у природній мові. Досліджено структуру Telegram Bot API на C# та особливості взаємодії з OpenAI API. Реалізовано підключення до TMDb API для отримання інформації про фільми. Розроблено логіку взаємодії з користувачем у вигляді діалогу, з урахуванням різних сценаріїв пошуку.

У результаті виконаної роботи розроблено функціонального бота в Telegram, який взаємодіє з користувачем у зручному інтерфейсі месенджера, дозволяє шукати фільми за різними критеріями, формувати особисті списки, а також отримувати рекомендації на основі запиту користувача.

MOVIE SELECTION, TELEGRAM BOT, LARGE LANGUAGE MODELS, OPENAI API, GPT, API TMDB, .NET, C#, MONGODB.

The object of the work is the process of conveniently providing users with recommendations for choosing movies.

The aim of the work is to develop a bot that provides convenient movie selection based on user queries, preferences and additional filters.

During the work, the principles of functioning of large language models and their capabilities for generating responses to queries in natural language were considered. The structure of the Telegram Bot API in C# and the features of interaction with the OpenAI API were studied. Connection to the TMDb API was implemented to obtain information about movies. The logic of interaction with the user in the form of a dialogue was developed, taking into account various search scenarios.

As a result of the work, a functional bot was created that interacts with the user in a convenient interface of the messenger, allowing you to search for films based on other criteria, create lists and make recommendations based on queries.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів.....6
Вступ.....7
1 Аналіз предметної області та огляд існуючих рішень.....8
1.1 Боти як інструмент взаємодії з користувачем.....8
1.2 Великі мовні моделі та їх сучасне застосування.....9
1.3 Огляд існуючих рішень для підбору фільмів.....12
1.4 Аналіз літератури щодо застосування LLM в системах персоналізованих рекомендацій.....17
1.5 Постановка задачі.....20
2 Проєктування бота в Telegram для підбору фільмів із використанням GPT-моделі від OpenAI.....21
2.1 Визначення вимог до функціоналу бота.....21
2.2 Обґрунтування вибору платформи для реалізації чату.....22
2.3 Архітектура та структура застосунку.....24
2.4 Отримання даних про фільми через TMDb API.....27
2.5 Опис роботи моделі GPT від OpenAI API.....28
2.6 Проєктування структури бази даних для зберігання списків фільмів.....30
2.7 Алгоритм процесу відповіді GPT на запит користувача.....33
3 Розробка бота в Telegram для підбору фільмів.....36
3.1 Обґрунтування вибору середовища програмної реалізації.....36
3.2 Створення та налаштування бота в Telegram.....37
3.3 Отримання доступу до TMDb API.....39
3.4 Отримання доступу до OpenAI API.....42
3.5 Створення промпту.....43
3.6 Налаштування та програмна реалізація застосунку.....45
3.7 Ілюстрація роботи бота та його тестування.....49
3.8 Перспективи подальшої роботи.....63
Висновки.....65
Перелік джерел посилання.....67

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

LLM – Large Language Models (великі мовні моделі)

GPT – Generative Pre-Trained Transformer (генеративний попередньо тренований трансформер)

TMDb – The Movie Database (база даних фільмів)

API – Application Programming Interface (прикладний програмний інтерфейс)

СУБД – система управління базами даних

БД – бази даних

JSON – JavaScript Object Notation

URL – Uniform Resource Locator (єдиний вказівник на ресурс)

HTTP – Hypertext Transfer Protocol (протокол передачі гіпертексту)

ВСТУП

Сьогодні ми живемо в час інформаційного надлишку, де постійно спостерігається зростання кількості даних. Основними джерелами цього потоку даних є соціальні мережі, новинні платформи, різноманітні сервіси, які завжди пропонують користувачам новий контент. Особливо це стосується сфери кіноіндустрії, де щодня з'являються нові фільми, серіали та трейлери. З одного боку, це створює необмежений вибір, де бажання кожного можуть бути задовольними, але з іншого боку, унаслідок цього можна відчути інформаційне перевантаження, в якому легко загубитися. Тому часто замість того, щоб просто насолоджуватися переглядом, користувачам доводиться витрачати багато часу на пошуки підходящого фільму, який би відповідав його поточним інтересам, бажанню або настрою.

Попри існування ряд онлайн-сервісів для підбору фільмів, які зазвичай застосовують алгоритми підбору, що спирається на колаборативної та контентної фільтрації, жанрових уподобань, рейтингу або популярності фільмів, ці підходи мають певні обмеження. Головна проблема закладається в тому, що вони здебільшого ігнорують емоційний стан користувача та контекст його запиту. Часто це сильно впливає на кінцевий вибір.

Однак останні часи принесли значні зміни у цифрових технологій. Ми спостерігаємо швидкий розвиток штучного інтелекту, особливо великих мовних моделей (LLM). Ці моделі можуть обробляти запити у вільній формі, розуміти деталі мови, контекст і, іноді, навіть настрій користувача. Такі особливості сучасних технологій відкривають нові можливості, які можна застосувати у системах рекомендацій для генерації адаптивних рекомендацій.

Таким чином, виникає актуальна задача у розробці зручного помічника, який допомагатиме знаходити фільми відповідно до індивідуальних побажань з використанням можливостей штучного інтелекту. Таке рішення може дозволити заощадити час користувачам і зробити їх перегляд фільмів більш осмисленим і персоналізованим.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Боти як інструмент взаємодії з користувачем

Боти – це програмні застосунки, які імітують спілкування з людиною через текстові або голосові\відео повідомлення та полегшують роботу з повсякденними завданнями. Вони автоматизують взаємодію між системою та користувачем, а саме можуть відповідати на питання, допомагати з навігацією, виконувати певні дії або надавати персоналізовані поради.

Першим відомим прикладом такого рішення є чат-бот ELIZA, який був створений у 1966 році в Массачусетському технічному університеті. Він був розроблений для імітації лікаря-психотерапевта та відповідав на питання пацієнтів з допомогою готових та заздалегідь прописаних шаблонів [1].

З того часу боти еволюціонували від простих сценаріїв до складних систем. Залежно від алгоритмів, на яких базується їхня робота, боти поділяють на:

– сценарні (обмежені) – працюють за заданим сценарієм, відповідають лише на передбачені запити;

– інтелектуальні (самонавчальні) – базуються на основі штучного інтелекту, які використовують методи машинного навчання та великі мовні моделі і можуть аналізувати контекст розмови, підлаштовуватись до нових обставин та ставати кращими з досвідом.

Треба зазначити, що сучасні боти можуть бути вжиті в різні платформи, такі як вебсайти, мобільні програми, соціальні мережі або месенджери. І тому їх активно використовують у різних галузях, зокрема:

– електронна комерція – допомагають у виборі товарів, з оформленням замовлень, надають консультації в режимі реального часу;

– банківська справа – повідомляють про баланс, здійснюють перекази, відповідають на типові фінансові запитання;

– охорона здоров’я – запис на прийом, первинне опитування симптомів та надання порад щодо лікування, пошук інформації про ціни на ліки;

– освіта – пояснення складного матеріалу, проведення тестів, організація навчального процесу;

– туризм і подорожі – бронювання квитків, пошук готелів, надання інформації про рейси, поради щодо подорожей, автоматичне інформування про затримки;

– розваги – пропонують музику, фільми, новини, інтерактивні ігри.

За даними [2], до 2025 року близько 88% користувачів у світі хоча б раз взаємодіяли з чат-ботом, а 34% споживачів надають перевагу чат-ботам перед традиційною підтримкою через дзвінки або електронну пошту.

У порівнянні зі звичайними сайтами чи мобільними застосунками, боти мають ряд важливих переваг [3-5]. По-перше, вони не потребують встановлення, користувачі можуть спілкуватися з ботом у знайомому для них місті, наприклад, у месенджерах, які люди використовують щодня. Це робить використання зручне і доступне для багатьох. По-друге, інтерфейс бота простий і не переповнений зайвими елементами. Не потрібно витрачати додаткові кошти на його створення, що дуже підходить для стартапу. Також, їх просто масштабувати, розширювати функціональність і обробляти багато запитів одночасно. Саме через ці переваги боти часто використовують для втілення ідей в бізнесі, сервісах та стартапах.

1.2 Великі мовні моделі та їх сучасне застосування

Великі мовні моделі (Large Language Model, LLM) – це сучасна технологія штучного інтелекту, яка працює з мовою за допомогою методів глибокого навчання. Вони здатні ефективно працювати з текстовою інформацією та створювати зв’язний мовний контент на основі великої кількості текстових даних [6].

Однією з головних переваг LLM є їхня здатність працювати з мовою, яка подібна до людського. Завдяки архітектурі трансформерів, що лежить в основі більшості LLM, вони можуть з високою точністю виконувати різноманітні мовні завдання. Це дозволяє використовувати їх у різних галузях.

Серед основних завдань можна виділити наступні:

– розуміння та генерація тексту (LLM «розуміють» та аналізують контекст, зміст і цілі користувача, здатні створювати цілісний текст у відповідь на введений запит або коротких підказок. Це корисно для написання статей, постів у соціальні мережі, кодів, творчих текстів тощо);

– машинний переклад (LLM можуть сприймати та створювати тексти, а також правильно й точно перекладати між різними мовами);

– аналіз настроїв (моделі можуть визначати емоційний тон повідомлень (позитивний, нейтральний або негативний), це допомагає, наприклад, бізнесам краще розуміти клієнтів і відповідно реагувати на їхні прохання);

– системи відповідей на запитання (LLM здатні давати змістовні відповіді на різні запитання – від простих до складних. Їх використовують у навчальних платформах, пошукових системах і довідкових сервісах, щоб швидко знаходити потрібну інформацію);

– чат-боти та віртуальні помічники (завдяки здатності вести діалог природною мовою, LLM застосовуються в розмовних системах, які імітують живе спілкування та допомагають користувачів у різних ситуаціях. На практиці їхній потенціал досліджується, зокрема в роботі [8], де автори представили консультативну систему для абітурієнтів університету на базі моделі GPT із використанням навчальних матеріалів університету як джерела відповідей);

– генерація нових ідей (LLM можна також застосувати для розробки креативних рішень. Наприклад, вони здатні пропонувати ідеї для нових

продуктів або послуг, а також генерувати оригінальний контент у галузі мистецтва та розваг).

Однією з найвідоміших реалізацій великих мовних моделей стала ChatGPT від компанії OpenAI. У порівнянні з іншими популярними платформами, у листопаді 2022 року він зміг набрати понад мільйон користувачів лише за п’ять днів після запуску (рис. 1.1).

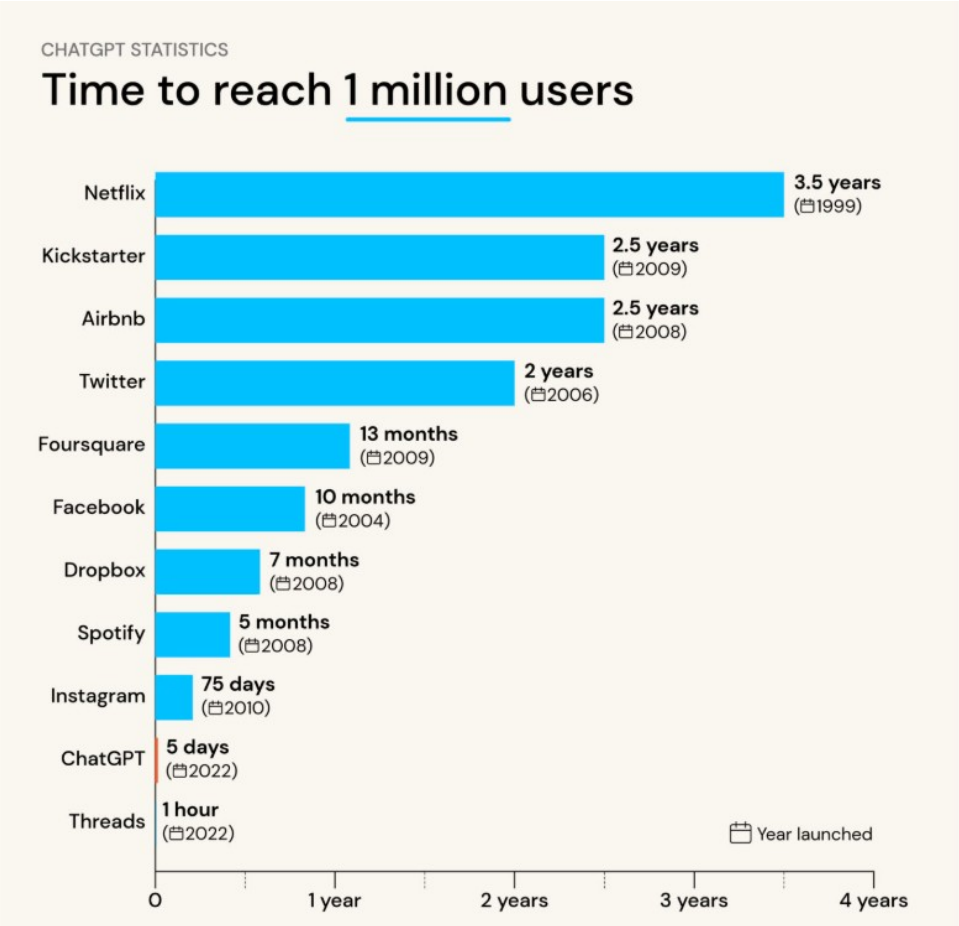

Рисунок 1.1 – Зростання темпів ChatGPT у порівнянні з іншими платформами [9]

Наразі існує п’ять провідних мовних моделей: GPT-4o, Клод 3.7 Сонет, Gemini 2.0 Flash, Грок 3 та DeepSeek R-1. Вони вирізняються потужною продуктивністю, підтримкою мультимодального вводу, багатомовністю та здатністю працювати з великими обсягами даних. Це робить їх ефективними для різних завдань [10].

1.3 Огляд існуючих рішень для підбору фільмів

На сьогоднішній день є багато застосунків, які допомагають людям обирати фільм для перегляду. У цьому підрозділі розглянемо декілька з них, які відрізняються за підходом і призначенням.

Спочатку розглянемо один з найвідоміших у світі стрімінгових сервісів – Netflix (рис. 1.2). Компанія була заснована у 1997 році як сервіс поштової доставки DVD, яка пізніше перетворилася на платформу для перегляду відео онлайн [11]. Netflix має одну з ключових функцій, яка відрізається від інших, це потужна рекомендаційна система. Вона працює завдяки спеціальним алгоритмам – колаборативній фільтрації, контентному аналізу та методом машинного навчання. Це допомагає Netflix підбирати фільми та серіали, які можуть сподобатися конкретному користувачу.

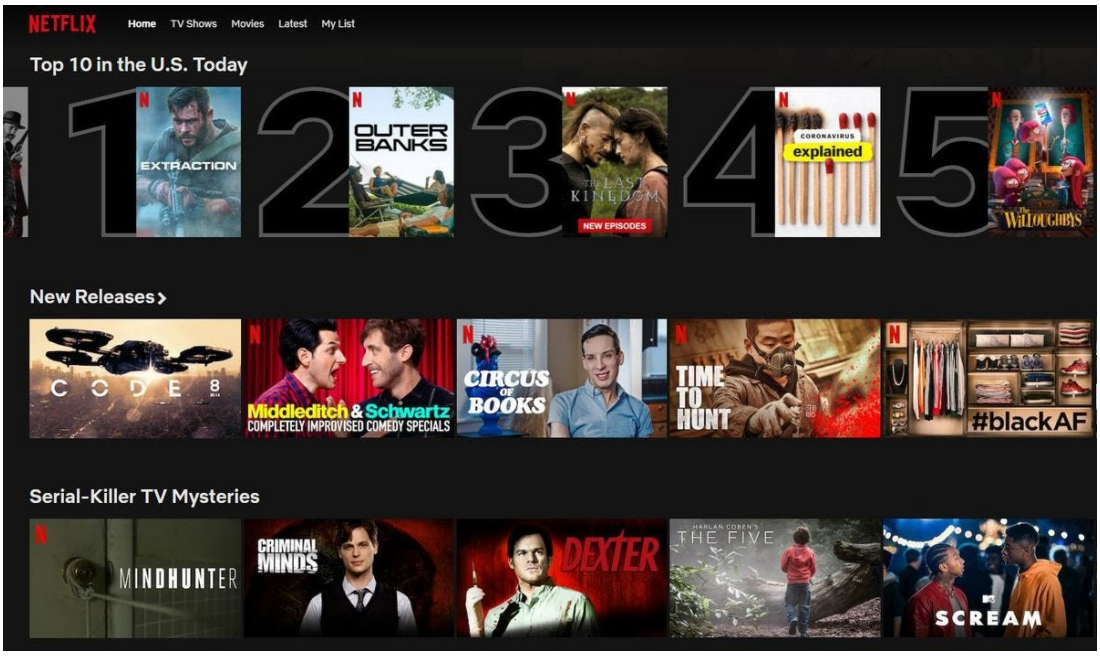

Рисунок 1.2 – Зовнішній вигляд інтерфейса сайта Netflix [12]

Переваги Netflix:

– пропонує широкий вибір фільмів, серіалів, документальних проєктів та власного контенту, які можна переглядати безпосередньо на самому сервісі;

– добре підлаштовується під інтереси користувача після ретельного аналізу історії переглядів, вподобання та взаємодію з контентом;

– користувач може оцінювати контент, що йому сподобалося або не сподобалося, і таким чином формувати особисті списки;

– фільми на платформі згруповані в тематичні добірки не тільки за жанром, а й за настроєм чи основним сюжетом. Серед прикладів: «сильні жіночі ролі», «похмурі трилери», «драми про підлітків»;

– зручний і легкий у зрозумінні інтерфейс для мобільних і десктопних пристроїв.

Серед недоліків варто відзначити:

– присутня проблема «холодного старту», тобто новим користувачам платформа ще не знає, що запропонувати;

– у разі частого перегляду фільмів одного жанру, система може почати пропонувати лише схожий контент;

– для користування платформою потрібна платна підписка, причому з кожним роком її вартість зростає;

– доступність контенту залежить від регіону, тому дуже часто в українській бібліотеці відсутні деякі популярні проєкти;

– деякі фільми та серіали не мають перекладу або озвучку українською мовою.

Наступним одним із популярних сервісів для введення особистої фільмотеки та пошуку фільмів є застосунок Letterboxd (рис. 1.3). Це соціальна платформа для кіноглядачів, які хочуть зберігати свої враження від переглядів, оцінювати, ділитися думками та обговорювати переглянуте. Вона з’явилася у 2011 році й з того часу здобула широку аудиторію серед кіноманів у всьому світі.

На відміну від багатьох інших застосунків для підбору фільмів, основна увага Letterboxd зосереджена не лише технічній інформації про фільм (актори, жанр, рейтинг тощо), а й на особистому досвіді реальних людей. Таким чином, формується живі спільнота, де рекомендації

формується не лише за допомогою алгоритмів, а й завдяки особистим смакам користувачів.

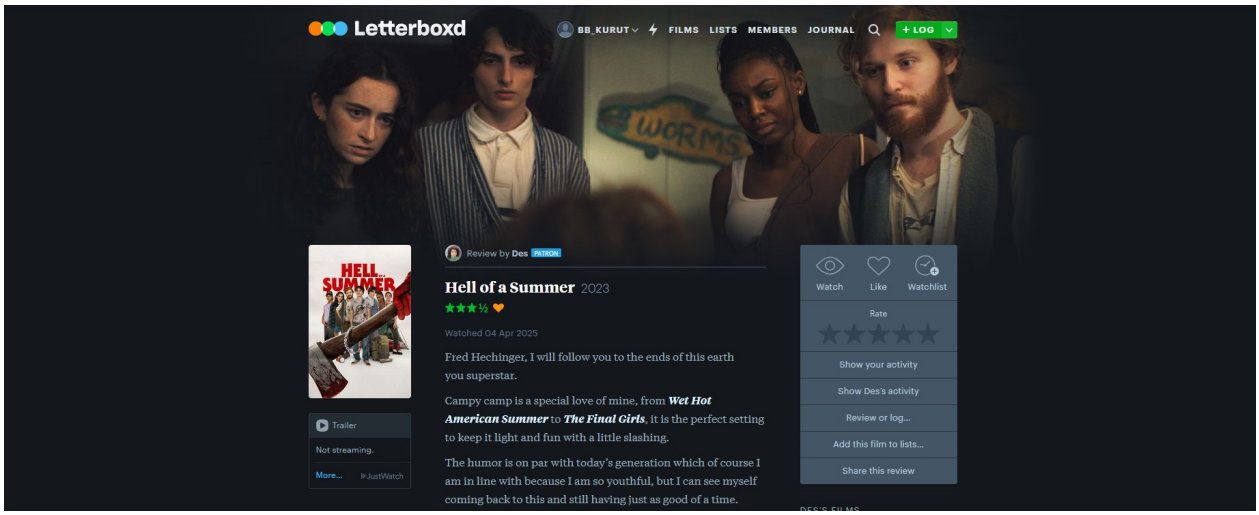

Рисунок 1.3 – Фрагмент інтерфейсу Letterboxd із прикладом рецензії іншого користувача на фільм

До основних переваг платформи Letterboxd належать:

– можна створювати та вести щоденник переглядів, де можна відзначити дату та час перегляду фільму;

– є можливість ставити оцінку фільмам за п’ятизірковою шкалою;

– писати текстові розгорнуті рецензії, з якими можна ділитися з іншими;

– створювати власні списки за темами, жанрами чи своїми критеріями, наприклад, «мої улюблені фільми жахів» чи «на перегляд з друзями»;

– соціальна взаємодія, яка проявляється у вигляді коментування, підписування на користувачів, ставлення вподобайок тощо.

Серед недоліків варто відзначити:

– через відсутність глибокого автоматичного підбору фільмів через алгоритми рекомендації підбираються не за індивідуальними смаками, а на активності інших користувачів;

– присутня велика кількість реклами, що дуже часто заважає у процесі функціонування з платформою;

– платформа доступна лише англійською мовою, це може ускладнювати користування для тих, хто нею не володіє.

В Україні також існують власні онлайн-застосунки, які орієнтовані на підбір фільмів. Серед них можна згадати такі платформи, як стрімінгові сервіси MEGOGO та Sweet.tv, які пропонують не лише кіно, а й телебачення, спортивні трансляції тощо, як Fander – застосунок із механікою свайпів для вибору фільмів, або DzygaMDB – українську базу даних про кіно. Проте серед усіх українських проєктів окремо виділимо Джарвіс, який відрізняється підходом до рекомендацій.

Джарвіс – це інтерактивний асистент, який допомагає користувачам у виборі фільмів відповідно до їхнього настрою [13]. Сервіс зосереджується на фільмах з українською аудіодоріжкою. Це робить його особливо актуальним і зручним для україномовної аудиторії (рис. 1.4).

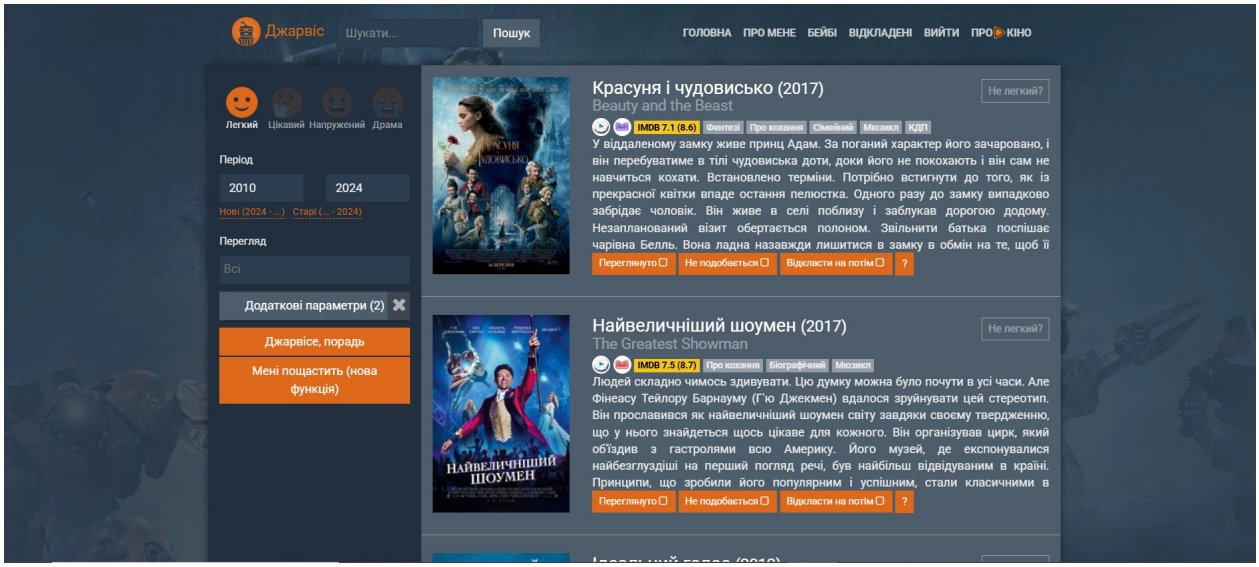

Рисунок 1.4 – Зовнішній інтерфейс Джарвіс

Взаємодія з вебплатформою відбувається у формі короткого опитування, після якого користувач отримує персоналізовану добірку

фільмів. Додатково можна відфільтрувати результати за жанром, роком випуску чи платформою перегляду.

Переваги Джарвіс:

– усі запропоновані фільми мають українську озвучку та дубляж;

– має легкий у зрозумінні інтерфейс та простий у використанні;

– присутні спеціальні категорії «КДП» і «КДП>13» в параметрах пошуку, вони допомагають підібрати фільми для перегляду з дітьми, з урахуванням вікових обмежень та тематики;

– інтегрований з такими легальними сервісами, як MEGOGO та Sweet.tv, на які одразу можна перейти до перегляду.

Недоліки Джарвіс:

– обмежена база фільмів, оскільки сервіс працює лише з фільмами, що мають українську доріжку;

– підбір фільмів відбувається за спрощеним принципом, без глибокого аналізу вподобань користувача, тому рекомендації іноді здаються поверхневими та не адаптуються з часом;

– часто можна не знайти менш відомі або нещодавно випущені фільми.

Окрім повноцінних вебплатформ та онлайн-сервісі, було також оглянуто стан ботів в месенджері Telegram для підбору фільмів. Але під час огляду подібних рішень було встановлено, що більшість із них втратили актуальність або не функціонують взагалі. Проте, вдалося виявити декілька активних ботів, які надають базовий функціонал, наприклад, пошук фільму за назвою, жанром чи випадковим чином, надання дуже короткого опису, постера, рейтингу, а також посилань на зовнішні ресурси для перегляду.

Такі боти можуть бути корисними для швидкого ознайомлення з фільмами, однак рівень персоналізації та інтелектуальної обробки запитів у них обмежений.

Отже, на основі проведеного аналізу можна зробити висновок, що більшість сучасних застосунків не враховують емоційний стан користувача

та не підтримують запити у вільній формі. Це підкреслює актуальність розробки більш адаптивних рішень, які б поєднували гнучкість великих мовних моделей і перевірені джерела кіноконтенту.

1.4 Аналіз літератури щодо застосування LLM в системах персоналізованих рекомендацій

Під час аналізу літератури було оглянуто ряд робот, присвячених використанню великих мовних моделей у системах персоналізованих рекомендацій. Ці роботи демонструють, як LLM можуть покращити якість рекомендацій завдяки глибшому розумінню природної мови, контексту запиту та емоцій користувача, що є важливою складовою користувацького досвіду та прийняття рішень [14, 15].

У першій публікації [16] автор розглядає одну з основних проблем сучасних рекомендаційних систем, яка називається проблема «холодного старту». Вона виникає, коли система не має достатньо інформації про нового користувача або новий продукт, і через це не може надати точних рекомендацій. Автор підкреслює, що традиційні алгоритми, зокрема колаборативна фільтрація, сильно залежать від історії взаємодії користувача з системою (перегляди, оцінки, покупки тощо). Коли така історія відсутня система не може сформувати достатньо персоналізовану рекомендацію. Тому погіршується досвід користувача.

Як варіант рішення, автор пропонує використання великих мовних моделей, що здатні аналізувати текстові запити – описи, відгуки або природну мову, якою користувач висловлює свої вподобання. Це дозволяє створити базовий профіль навіть без попередньої взаємодії з системо.

До переваг такого підходу віднесено:

– глибше розуміння контексту запиту;

– роботу з описовими характеристиками контенту (сюжет, жанр, стиль, тональність);

– адаптацію до користувача на основі кількох текстових повідомлень без анкети чи історії переглядів.

Ці теоретичні міркування підтверджуються у більш точному дослідженні [17], де автори на практиці перевірили ефективність LLM в умовах, схожих до «холодного старту». У роботі показано, що навіть без попереднього навчання (у форматі zero-shot або few-shot), LLM здатні формувати персоналізовані рекомендації на основі текстових описів або мінімальних уподобань. Результати порівняли з класичними методами колаборативної фільтрації підбору фільмів. LLM показали хорошу якість, навіть якщо це були відкриті моделі середнього розміру.

Третє дослідження [18] зосереджено на здатності LLM аналізувати запити користувачів, які написані у вільній формі. Автори стверджують, що більшість людей обирають фільми не лише за жанром зі списку, а й часто потребують в описі своїх вподобань чи побажань у формі розгорнутих текстів. Це важко зробити в традиційних системах. Наприклад, замість «Фільм жахів» користувач може описати: «Я хочу подивитися трилер із захопливим сюжетом, як «Острів проклятих». Саме такі запити є викликом для класичних рекомендаційних систем.

Для цього автори протестували 38 моделей LLM (включаючи GPT-4o, Llama 3, Gemini та інші) та використали реальні приклади запитів з Reddit. Ключові результати дослідження:

– LLM показали вищу точність у рекомендаціях, ніж класичні методи (наприклад, doc2vec);

– великі моделі, як GPT-4o, продемонстрували найкращі результати, але й відкриті моделі середнього розміру, як Gemma 2 9B, працюють також достатньо добре;

– різні стратегії формулювання запитів (zero-shot та few-shot) не дали суттєвої різниці в результатах.

В наступній роботі [19] автори розглядають можливості використання великих мовних моделей як інструмент для рекомендації фільмів у форматі діалогу. Основна увага приділяється взаємодії GPT-подібної моделі з користувачами в реальному часі через чат-інтерфейс. В цьому чаті учасники формулювали запити у вільній формі, а модель відповідала як персональний асистент.

В експерименті взяли участь 160 активних користувачів онлайн-сервісу рекомендацій фільмів. Їм пропонувалося три сценарії: пошук нішевих фільмів, підбір фільмів для перегляду під час подорожі та створення добірки на день народження разом з друзями. Було використано промпти типів zero-shot, one-shot і few-shot.

Оцінювання ефективності проводили шляхом анкетування та аналізу діалогів, з урахуванням таких метрик, як персоналізація, новизна, різноманітність, довіра, зрозумілість і загальна корисність рекомендацій.

Основні результати дослідження такі:

– користувачі позитивно оцінили здатність моделі пояснювати свої рекомендації, що покращувало сприйняття відповідей;

– діалоговий формат спілкування також був сприйнятий позитивно, особливо можливість надавати додаткову інформацію про свої вподобання;

– водночас деякі учасники висловили своє невдоволення щодо недостатнього рівня персоналізації, оскільки модель часто пропонувала популярні фільми, які вже бачили;

– учасники, які мали меншу історію переглядів, були більш задоволені, а досвідчені глядачі оцінювали їх критичніше;

– коли користувачі надавали певний контекст, наприклад, улюблені фільми або жанри, якість відповідей значно покращувалась, а загальні або тестові запити без контексту давали менш задовільні результати;

– найкращі оцінки отримав сценарій з пошуку нішевих фільмів, за новизну та відповідність очікуванням;

– незалежно від типу промпту користувачі оцінювали якість рекомендацій приблизно однакова.

Аналіз наведених досліджень підтверджує значний потенціал великих мовних моделей у сфері персоналізованих рекомендацій фільмів. Вони вирішують проблему «холодного старту» та краще розуміють запити користувача. Це робить їх корисним інструментом для сервісів, що враховують інтереси та емоції людей.

1.5 Постановка задачі

Таким чином, створення бота для підбору фільмів із використанням великих мовних моделей є актуальним завданням, що поєднує сучасні підходи в галузі персоналізованих рекомендацій, штучного інтелекту та взаємодії з користувачем у зручному форматі.

Об’єктом роботи є процес зручного надання користувачам рекомендацій щодо вибору фільмів.

Метою роботи є розробка бота, що здійснює зручний підбір фільмів відповідно до запитів користувача, його вподобань та додаткових фільтрів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– дослідити підходи до реалізації персоналізованих рекомендацій, зокрема із застосуванням великих мовних моделей;

– ознайомитися з Telegram API та інструменти для реалізації бота мовою програмування C#;

– реалізувати підключення до TMDb API для отримання інформації про фільми;

– реалізувати інтеграцію з OpenAI API для обробки текстових запитів користувача та генерації рекомендацій;

– розробити інтерфейс бота та логіку взаємодії з користувачем;

– провести тестування бота та оцінити його ефективність.

2 ПРОЄКТУВАННЯ БОТА В TELEGRAM ДЛЯ ПІДБОРУ ФІЛЬМІВ ІЗ ВИКОРИСТАННЯМ GPT-МОДЕЛІ ВІД OPENAI

2.1 Визначення вимог до функціоналу бота

На етапі проєктування системи спочатку необхідно чітко визначити вимоги до його функціональних можливостей. Це упорядкує логіку роботи, продумає взаємодію з користувачем і сформує технічне завдання, якому буде відповідати кінцевий продукт.

Головна мета створення бота – це забезпечити зручний, швидкий і особистий підбір фільмів згідно з проханнями користувача. Для цього бот має поєднати класичні фільтри, таких як жанр, ключові слова, рейтинг тощо, з інтелектуальними методами створення порад на основі природної мови, які реалізовані через LLM.

У результаті аналізу предметної області та вивчення подібних сервісів було сформовано наступні ключові вимоги до функціоналу бота:

– надавати можливість здійснювати пошук фільмів за назвою або ключовими словами;

– забезпечувати пошук фільмів за жанром, популярністю, рейтингом або датою випуску;

– виводити детальну інформацію про фільм (постер, опис, жанр, рейтинг, дата релізу, трейлер);

– генерувати персоналізовані рекомендації на основі уподобань користувача;

– реалізувати можливість випадкового вибору фільму;

– дозволяти додавати фільми до списку «Цікаві» та «Переглянуті»;

– надавати можливість оцінювання переглянутих фільмів за 10-бальною шкалою;

– забезпечувати перегляд персональних списків користувача;

– забезпечувати оновлення персональних списків користувача;

– створювати можливість ділитися списком зацікавлених та переглянутих фільмів з іншими користувачами.

Для кращого розуміння взаємодії користувача з ботом побудовано діаграму прецедентів, яка є графічним відображенням взаємодії між акторами та системою. Вона допомагає візуалізувати функціональність системи з точки зору користувача. У нашому випадку, всі функції зосереджені навколо єдиного актора – звичайного користувача (рис. 2.1). Під користувачем треба розуміти будь-яку особу, яка взаємодіє з ботом для отримання рекомендацій щодо фільмів та управління власними добірками.

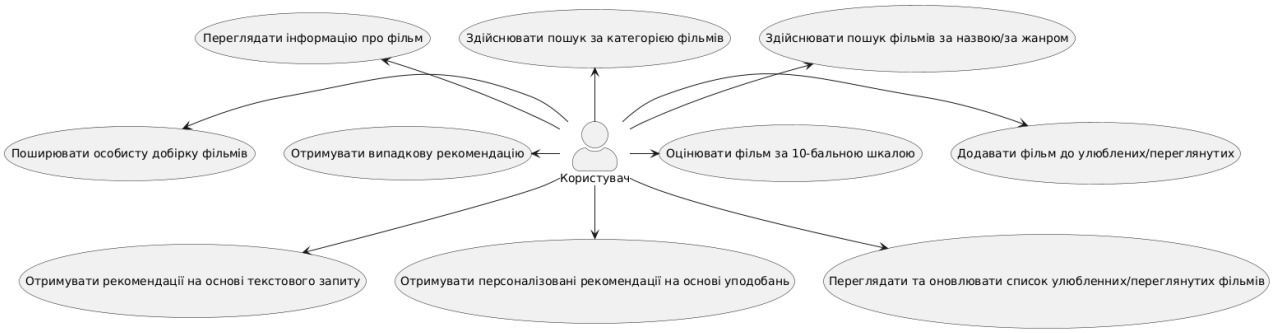

Рисунок 2.1 – Головна діаграма прецедентів для боту з підбору фільмів

2.2 Обґрунтування вибору платформи для реалізації чату

Далі у процесі проєктування системи важливим кроком є вибір платформи, через яку здійснюватиметься взаємодія користувача з ботом. У даному проєкті було вирішено реалізувати бота на базі месенджера Telegram. Такий вибір обумовлений декількома причинами.

По-перше, Telegram є одним із найпопулярніших месенджерів в Україні, особливо після початку повномасштабного вторгнення у 2022 році. Дана платформа стала джерелом миттєвих новин, обміну інформацією та комунікації між людьми. Зараз більш ніж 70% українців регулярно

користуються Telegram. Вона активно застосовується серед молодої аудиторії для спілкування [20].

По-друге, Telegram надає офіційний відкритий Bot API, який добре задокументований з прикладами запитів, параметрів, відповідей, з описом використання тощо. BotFather процес запуску нового бота є швидким і простим, який займає кілька хвилин. Без реєстрації бізнесу або подачі заявки, як це вимагає WhatsApp або Viber. Крім того, треба зазначити, що Telegram має наявність великої кількості доступних бібліотек для різних мов програмування. Це робить його універсальним для всіх розробників.

По-третє, Telegram відрізняється від інших месенджерів тим, що робить спілкування між ботом та користувачем легким через зручний та привабливий інтерфейс. Оскільки він підтримує великий набір UI-компонентів, таких як inline-кнопки, меню, клавіатуру і мультимедіа. Також на противагу WhatsApp, Telegram можна одночасно гнучко використовувати на кількох пристроях, бо всі дані зберігаються в хмарі і автоматично синхронізуються.

Наступна провідна перевага – це конфіденційність та високий рівень захисту. Telegram використовує власний протокол шифрування MTProto, який дозволяє захищати дані під час передавання. Також ідентифікація користувача відбуваєтсья через «chat_id», бот не знає ніякої його особистої інформації. Надійність захисту повідомленнь користувачів була підтверджена часом та є однією з найкращих на ринку застосунків-мессенджерів.

Крім того, Telegram добре пристосований у застосуванні сторонніх API. Він дозволяє легко підключити зовнішні API, наприклад, для отримання даних про фільми або генерації рекомендацій через OpenAI. У Telegram немає сильних обмежень, і все працює швидко. В інших месенджерах, наприклад, WhatsApp чи Facebook Messenger, присутні додаткові вимогливі правила і складніша процедура підключення інших сервісів.

В альтернативу можна було б зробити окремий вебсайт або мобільний додаток для роботи з ботом. Але створення повного чату з нуля потребує більше ресурсів (розробка UI-дизайну, реєстрація користувачів, налаштування серверної частини та система зберігання). Це значно ускладнює реалізацію і збільшує обсяг роботи. У той час, Telegram вже має весь необхідний функціонал для створення та підтримки бота. Такий підхід дозволяє зосередитись на логіці рекомендацій, а не на основній функціональності чату.

Отже, Telegram є оптимальним рішенням для реалізації бота в межах нашого проєкту.

2.3 Архітектура та структура застосунку

Згідно з вимогами проєкту, інтерфейсом користувача є Telegram-бот. Уся комунікація між користувачем і системою проходить через Telegram Bot API, який має набір інструментів, що дає можливість програмно надсилати повідомлення та отримувати відповіді за протоколом HTTP, тобто через вебмережу. Таким чином, застосунок реалізує клієнт-серверну архітектуру, де Telegram-користувач виступає клієнтом, який надсилає запити, а серверною частиною є програмна логіка, яка їх обробляє і повертає відповідь.

Оскільки Telegram-бот для пошуку та рекомендацій фільмів є відносно невеликим застосунком, то було вирішено обрати монолітну архітектуру для серверної частини системи. Монолітна архітектура передбачає створення єдиної цілісної програми, в межах якої вся логіка, всі компоненти об’єднані в одну велику програму (рис. 2.2).

Основна перевага такої архітектури є простота розробки, тестування та запуск. Уся логіка, включно з обробкою даних, збереженням і відображенням, реалізується в одному місці. Завдяки цьому легше

підтримувати єдине середовище та зменшується час на розробку та його налагодження.

Рисунок 2.2 – Загальна модель монолітної архітектури

Серед недоліків потрібно зазначити, що, коли система починає збільшуватися за рахунок нових функцій, то це може призвести до навантаження системи і вона може стати важче підтримуваною. Компоненти починають тісно залежати один від одного, тому будь-які зміни або оновлення в одній частині застосунку можуть впливати на роботу інших. Також можливе виникнення критичних помилок, які зупиняють роботу всієї програми, а не окремого її фрагмента.

Архітектура застосунку зображена на рисунку 2.3.

«TelegramClient» – інтерфейс, через який взаємодіє користувач з ботом у Telegram.

«TelegramBotModul» відповідає за зв’язок між користувачем і ботом, прийом й обробку повідомлень, надсилання відповідей, взаємодію з іншими модулями.

«ChatGptModul» виконує функцію генерації текстових відповідей на основі користувацьких вільних текстових запитів. Він звертається до GPT-моделі, яка інтерпретує запит користувача та повертає відповідну рекомендацію.

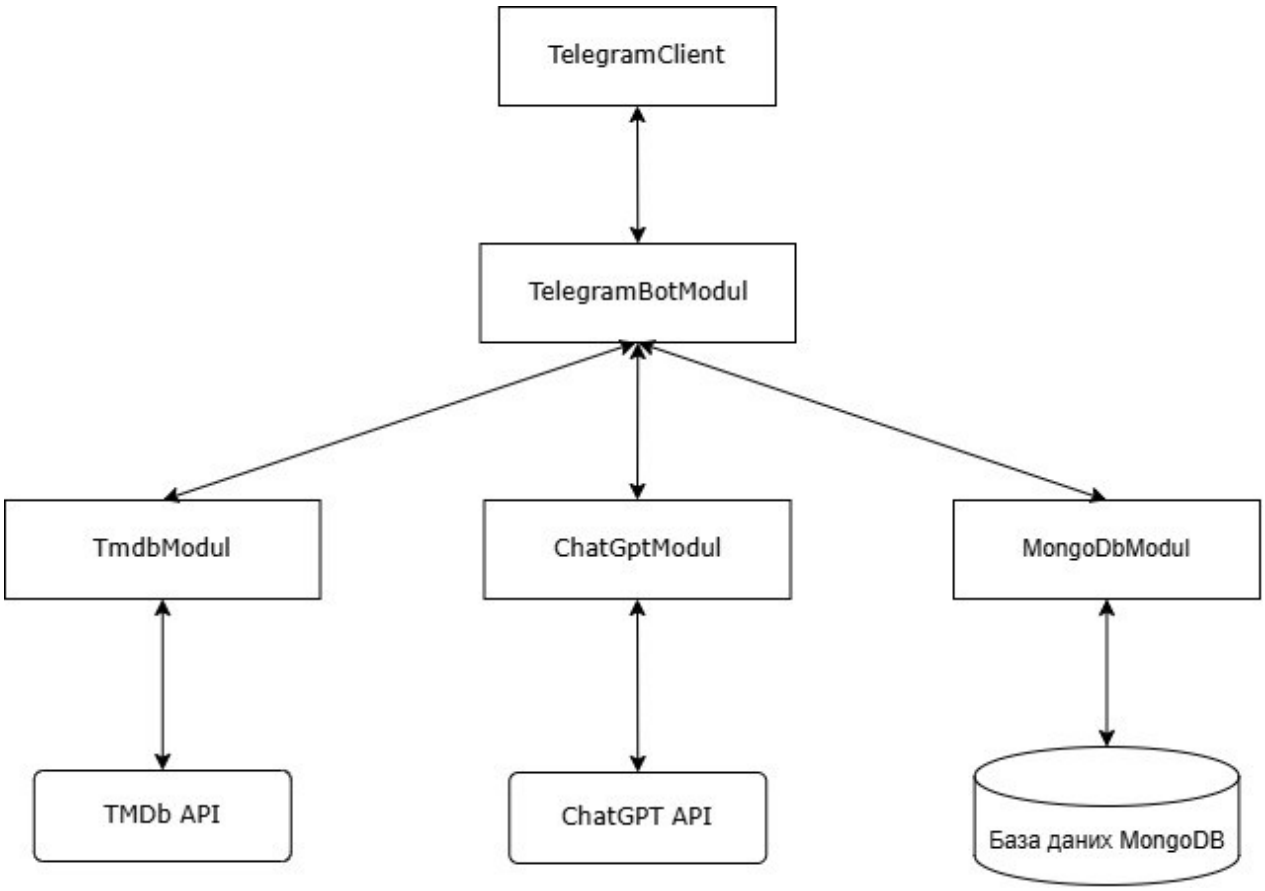

Рисунок 2.3 – Діаграма архітектури застосунку

«TmdbModul» реалізує пошук фільмів та отримання повної інформації про них із бази TMDb, включаючи постери, назву, описи, жанри, рейтинги, дату випуска та посилання на трейлер.

«MongoDbModul» забезпечує збереження та отримання особистих списків фільмів (зацікавлених і переглянутих) в базі даних MongoDB, реалізує CRUD-операції над колекціями.

Також, як видно на зображенні 2.3, для створення боту необхідно використовувати відповідні зовнішні API, які забезпечать отриманню акутальних й достовірних метаданих про фільми та застосуванню можливостей штучного інтелекту. Розглянемо більш детально ці два компоненти далі.

2.4 Отримання даних про фільми через TMDb API

Оскільки бот спрямований на рекомендації фільмів, тому дуже важливо обрати правильне рішення для отримання даних про фільми. Існує два основні підходи дял вирішення цього завдання: створити власну базу даних фільмів або підключитися до готових рішень. Створення власної бази займає багато часу, необхідно регулярно його оновлювати і вимагає великих ресурсів. Тому було прийнято рішення використовувати зовнішній API, який вже має всю потрібну інформацію.

Серед популярних варіантів джерел інформації про фільми є TMDb (The Movie Database) API. Він надає доступ до великої бази даних фільмів, серіалів, акторів та багато іншого. Платформа забезпечує можливість отримання актуальної інформації про фільми, включно з назвами, описами, рейтингами, постерами, трейлерами та іншими метаданими. Дана API постійно оновлюється спільнотою та редакторами, тому можна бути впевненим у забезпеченності достовірності даних. Також вона надає безкоштовний базовий доступ для використання.

Також треба зазначити, що TMDb API побудований на принципах REST, що робить сервіс дуже універсальним. Оскільки це дозволяє працювати з ним практично з будь-якої мови програмування, яка вміє надсилати HTTP-запити. На рисунку 2.4 показано, як формується шаблон URL-запиту.

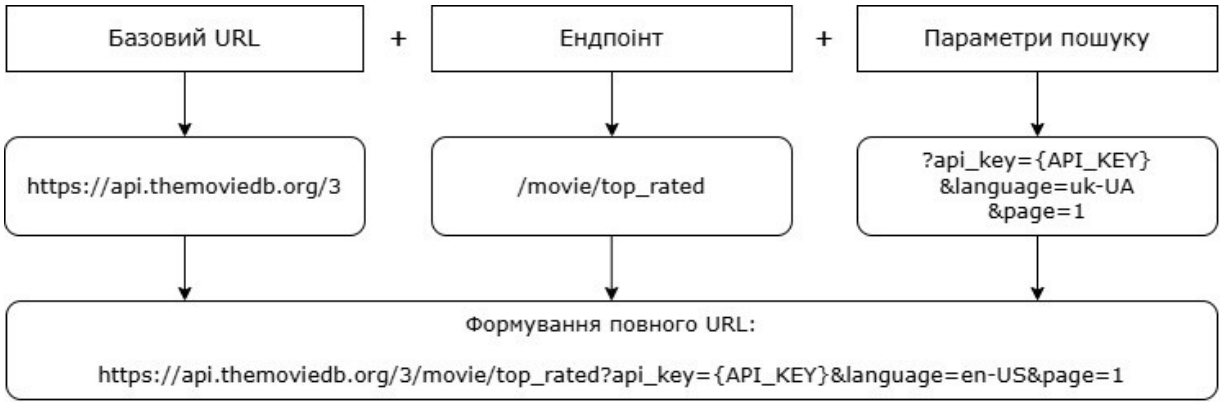

Рисунок 2.4 – Формування запиту до TMDb API

Детально розглянемо структуру запиту:

– базовий URL – це основна адреса сервера TMDb API, через яку відбувається вся взаємодія. Цей стандартний шлях використовується для всіх запитів;

– ендпоїнт – конкретна частина API, яка відповідає за певну дію або набір даних. Наприклад, «/movie/popular» використовується для отримання списку популярних фільмів;

– параметри пошуку – це додаткові налаштування запиту, які передаються через URL, щоб змінити результати. До основних параметрів належать language (встановлює мову результатів), page (номер сторінки результатів), а також обов'язковий унікальний api_key для авторизації запиту. Без ключа запит буде невалідним.

Коли ми надсилаємо запит до TMDb API, то у відповідь отримаємо JSON-об'єкт. Він містить масив фільмів з їхніми основними характеристиками (ID, назва, опис, рейтинг та інше). Крім самих фільмів, також повертається інформація про кількість сторінок та загальну кількість результатів.

2.5 Опис роботи моделі GPT від OpenAI API

Оскільки однією з вимог бота є згенерувати рекомендацію користувачу на основі текстового запиту та списків фільмів, то для цього було вирішено використати одну з сучасних великих мовних моделей, яка здатна аналізувати та генерувати текст на основі заданого запиту. Було обрано модель GPT від OpenAI API.

Ілюстрація нижче схематично зображує загальний принцип функціонування мовної моделі (рис. 2.5). Усе починається з введення тексту користувача, який проходить через попередню обробку – його ділять на частини (токени), а потім кожен токен перетворюють у числові вектори. Далі

ці вектори передаються безпосередньо у мовну модель, яка на основі всіх слів спочатку «вгадує», що логічно має бути далі. На виході модель створює ймовірнісний розподіл можливих відповідей, серед яких обирається найкращий варіант. Після цього відбувається декодування результату назад у текст – це і є відповідь, яку бачить користувач.

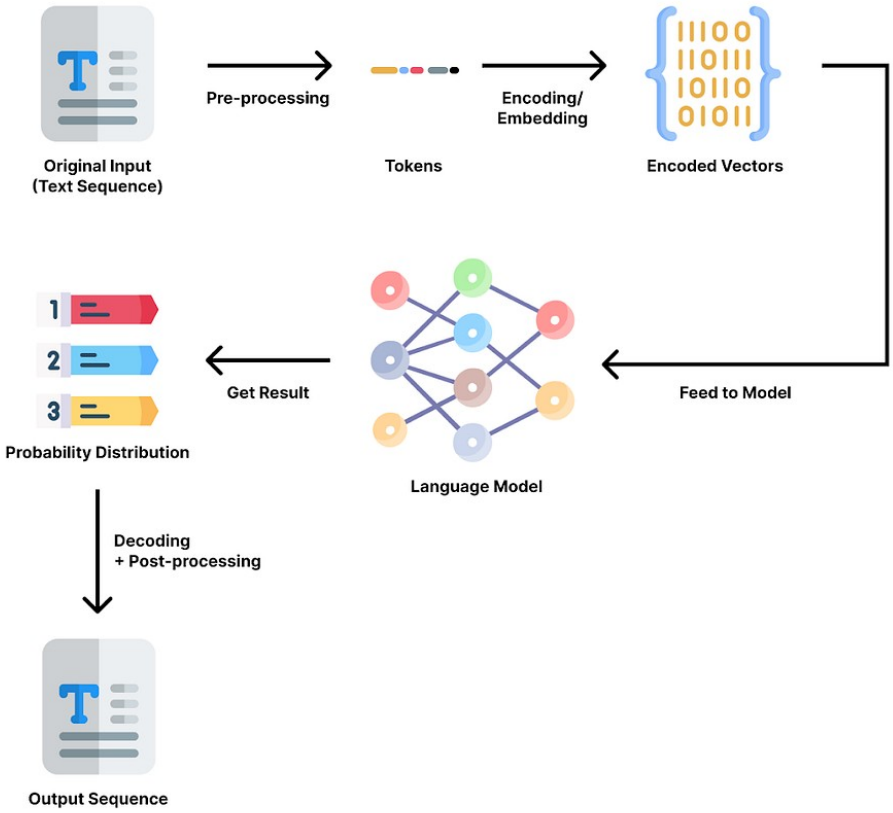

Рисунок 2.5 – Загальний принцип роботи моделі GPT [21]

Архітектура трансформера є основою GPT. Вона є типом нейронної мережі, яка схожа на нейрони у нашому людському мозку і використовується всередині цієї мовної моделі. Саме завдяки їй модель здатна одночасно враховувати контекст усього речення. В основі трансформера лежить механізм самоуваги (self-attention), який дозволяє кожному слову визначати, які слова в реченні найбільш важливі для розуміння змісту. Це дає змогу враховувати не лише окремі слова, а й їхній контекст у межах всього запиту.

Крім того, для взаємодії з моделлю GPT існують такі три ролі:

– система (system) визначає загальні умови та правила поведінки моделі. Це своєрідна інструкція або контекст, що задає стиль, тон, формат відповіді та роль, яку виконує модель;

– користувач (user) містить запит або повідомлення від користувача. Це може бути в формі питань, команд або введення даних, які потребують обробки або відповіді від моделі;

– помічник (assistant) – це є відповідь, згенерованою моделю на запит користувача відповідно до заданого контексту та інструкцій.

2.6 Проєктування структури бази даних для зберігання списків фільмів

У сучасних системах бази даних відіграють ключову роль, адже саме вони відповідають за організоване зберігання, доступ і обробку необхідної інформації [22-24].

У випадку з Telegram-ботом для підбору фільмів виникає потреба в підключенні постійного сховища для забезпечення збереження даних користувача, зокрема списків улюблених і переглянутих фільмів. Саме для цього використовується база даних, вона дає змогу зберігати улюблені фільми, дату додавання, відмічати переглянуте, залишати оцінки, а також отримувати доступ до збереженої інформації в будь-який момент під час взаємодії з ботом.

У якості СУБД планується використовувати MongoDB. MongoDB є документоорієнтованою NoSQL-базою даною, яка дозволяє зберігати інформацію у форматі BSON/JSON документів.

MongoDB була обрана через наступні переваги:

– гнучкість структури – MongoDB не вимагає зазделегідь визначати структуру таблиць, як у SQL, можно легко додавати нові поля без зміни всієї структури бази;

– підтримка вкладених масивів і об’єктів – MongoDB дозволяє зберігати всередині документа масиви й вкладені об’єкти без потреби у додаткових таблицях або зв’язках, таким чином, можна уникати складних операцій JOIN [25];

– зручність інтеграції – MongoDB легко інтегрується з мовами програмування через відповідні бібліотеки;

– масштабованість – завдяки вбудованому механізму масштабування (sharding) MongoDB дозволяє з легкістю працювати з великим обсягом запитів одночасно;

– мінімальні витрати на розробку схеми – завдяки своїй простоті й гнучкості можна одразу працювати з даними без попереднього проєктування схеми, що прискорює розробку MVP чи бета-версій;

– вбудована підтримка реплікацій – MongoDB забезпечує збереження даних навіть при збоях сервера.

Таким чином, було вирішено, що всі дані обраних фільмів користувачів зберігатимуться в одній колекції, де кожен документ відповідатиме одному користувачу бота. У документі є поле з Telegram Chat ID, який є унікальним та дозволяє легко співвіднести користувача з його списками фільмів, а також два основні списки-масиви: Favorites та Watched. Оскільки повну інформацію про фільми ми будемо отримувати із зовнішнього джерела, з API TMDb, то немає потреби зберігати всю додаткову інформацію у власній базі даних. Достатньо зберігати лише унікальний ідентифікатор фільму (TMDb ID), за яким у будь-який момент можна звернутися до зовнішнього API та отримати актуальні дані (опис, жанр, постер, рейтинг тощо). Назву фільму зберігається для зручного пошуку, а дата додавання допомагає відслідковувати, коли саме фільм потрапив до списка. Треба також зазначити про потребу зберігання

назв фільмів в оригіналі, тобто англійською мовою. Це необхідно для коректної роботи функції генерації персоналізованих рекомендацій.

Щоб краще зрозуміти логіку побудови бази даних, у таблиці 2.1 подано короткий опис ключових полів. Додатково, в лістингу 2.1 наведена типова структура документа в колекції.

Таблиця 2.1 – Опис основних полів БД

Назва поля	**Тип даних**	**Опис**
ChatId	Number (Long)	Унікальний ідентифікатор користувача в Telegram
Favorites	Array of Object	Масив об’єктів з зацікавленими фільмами
Film_id	Number (Int)	Унікальний ідентифікатор фільму з TMDB
Title	String	Назва фільму на українській мові
OriginalTitle	String	Назва фільму на англійській мові
Favorites.DateAdded	Date	Дата додавання фільму до списку улюблених
Watched	Array of Object	Масив об’єктів з переглянутими фільмами
Watched.Rating	Number (Int)	Оцінка користувача за переглянутий фільм (від 1 до 10)
Watched.DateWatched	Date	Дата, коли фільм було відзначено як переглянутий

Лістинг 2.1 Загальна структура документа в MongoDB:

```
{
 "ChatId": 609130192,
 "Favorites": [
```

```
    {
     " Film_id": 12445,
     "Title": "Гаррі Поттер та смертельні реліквії: Частина 2",
     "OriginalTitle": "Harry Potter and the Deathly Hallows: Part 2",
     "DateAdded": "2025-04-11T16:12:36Z"
    }
  ],
  "Watched": [
    {
     " Film_id": 1010581,
     "Title": "Моя провина",
     "OriginalTitle": "My Fault",
     "Rating": 10,
     "DateWatched": "2025-04-11T12:50:12.965Z"
     }
  ]
}
```


2.7 Алгоритм процесу відповіді GPT на запит користувача

Процес формування відповіді GPT на запит користувача щодо персоналізованих рекомендацій реалізується через послідовну взаємодію між кількома модулями системи Telegram-бота. Основна мета цього процесу є надати доречні рекомендації фільмів, що враховують індивідуальні вподобання користувача.

Алгоритм складається з наступних ключових кроків:

Крок 1. Даний запит ініціалізується після натискання користувачем кнопку «Персоналізовані рекомендації» в Telegram-інтерфейсі. Запит передається «TelegramBotModul» для подальшої обробки.

Крок 2. На основі унікального ідентифікатора користувача бот звертається до бази даних MongoDB, щоб отримати два списки:

– фільми, які користувач додав до розділу «Зацікавлені»;

– фільми з розділу «Переглянуті» разом з їхніми оцінками.

Крок 3. На основі зібраних даних компонент «ChatGptModul» формує структурований текстовий запит до моделі GPT. У ньому чітко вказується роль моделі, завдання, опис формату відповіді з прикладом, а також перелік фільмів, які слід аналізувати для майбутніх рекомендацій.

Крок 4. Сформований промпт передається через API до GPT-моделі від OpenAI. Модель аналізує надану інформацію, жанри, сюжети та атмосферу вказаних фільмів і генерує список нових рекомендацій (назви фільмів англійською мовою з роком випуску).

Крок 5. Згенерований список передається до «TmdbModul», який звертається до TMDb API для отримання локалізованої інформації українською мовою (назва, опис, жанр, рік випуску, рейтинг, трейлер, постер, посилання на трейлер).

Крок 6. Бот формує компактне повідомлення зі списком рекомендованих фільмів, де кожна позиція представлена у вигляді інлайн-кнопки з назвою і роком випуску.

Крок 7. Після того як користувач натискає на одну з кнопок, бот надсилає повну картку обраного фільму з усією необхідною інформацією.

Крок 8. Якщо користувач додасть фільм до «Зацікавлених» або «Переглянутих», ці зміни фіксуються в базі даних.

Треба додатково зауважити, що було вирішено виконувати пошук фільмів з відповіді GPT-моделі не лише за назвою, а й за роком випуску. Пошук тільки за назвою може бути неточним, оскільки існують фільми з однаковими або схожими назвами. Крім того, деякі фільми можуть мати кілька версій, знятих у різні роки. Тому рік допомагає точніше визначити потрібний фільм.

Додатково цей процес відображено на діаграмі послідовностей, яка чітко показує взаємодію між усіма компонентами, як виконуються дії, які повідомлення надсилаються і коли (рис. 2.6).

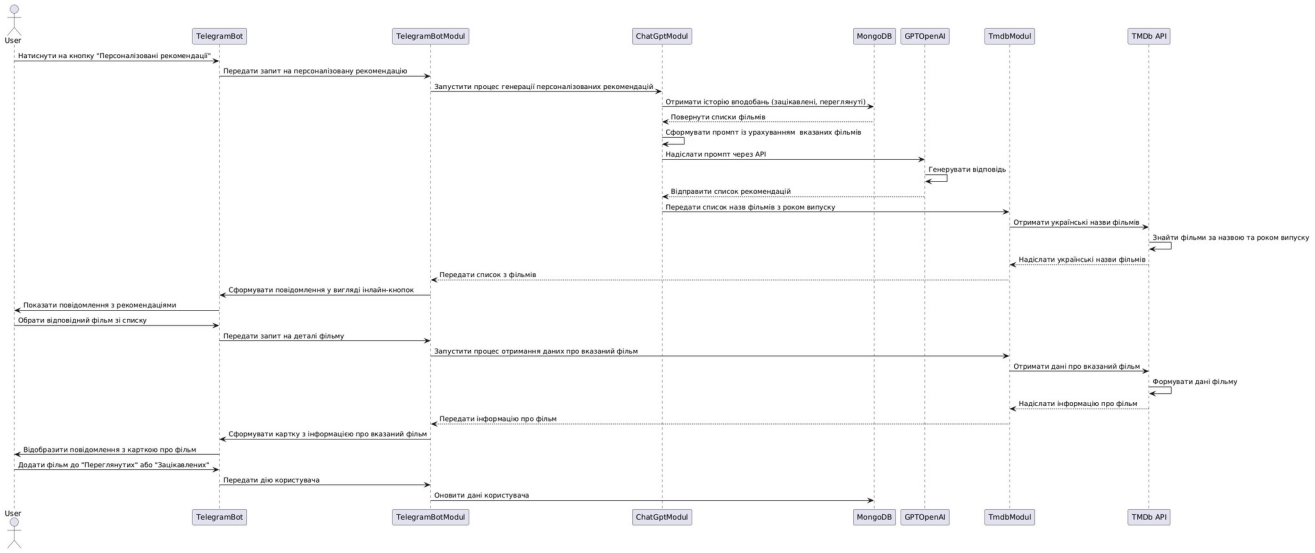

Рисунок 2.6 – Діаграма послідовності отримання рекомендацій фільмів від моделі GPT

3 РОЗРОБКА БОТА В TELEGRAM ДЛЯ ПІДБОРУ ФІЛЬМІВ

3.1 Обґрунтування вибору середовища програмної реалізації

Одним із ключових етапів реалізації даного проєкту є вибір відповідних інструментів, мов програмування та технологій, які дозволять ефективно реалізувати поставлену задачу. У рамках кваліфікаційної роботи було прийнято рішення використовувати середовище Visual Studio Community 2022 та мову програмування C#.

Visual Studio Community 2022 - це безкоштовне інтегроване середовище розробки програмного забезпечення від Microsoft (рис. 3.1). За допомогою Visual Studio можна писати код на різних мовах, в тому числі C#, проводити тестування та налагоджувати програмний код взагалі. У нього є всі необхідні інструменти для комфортної роботи з проектами на платформі .NET, включаючи редактор коду, відлагоджувач, термінал, візуальне керування проєктом і інтеграція з іншими інструментами розробки (Git. Azure).

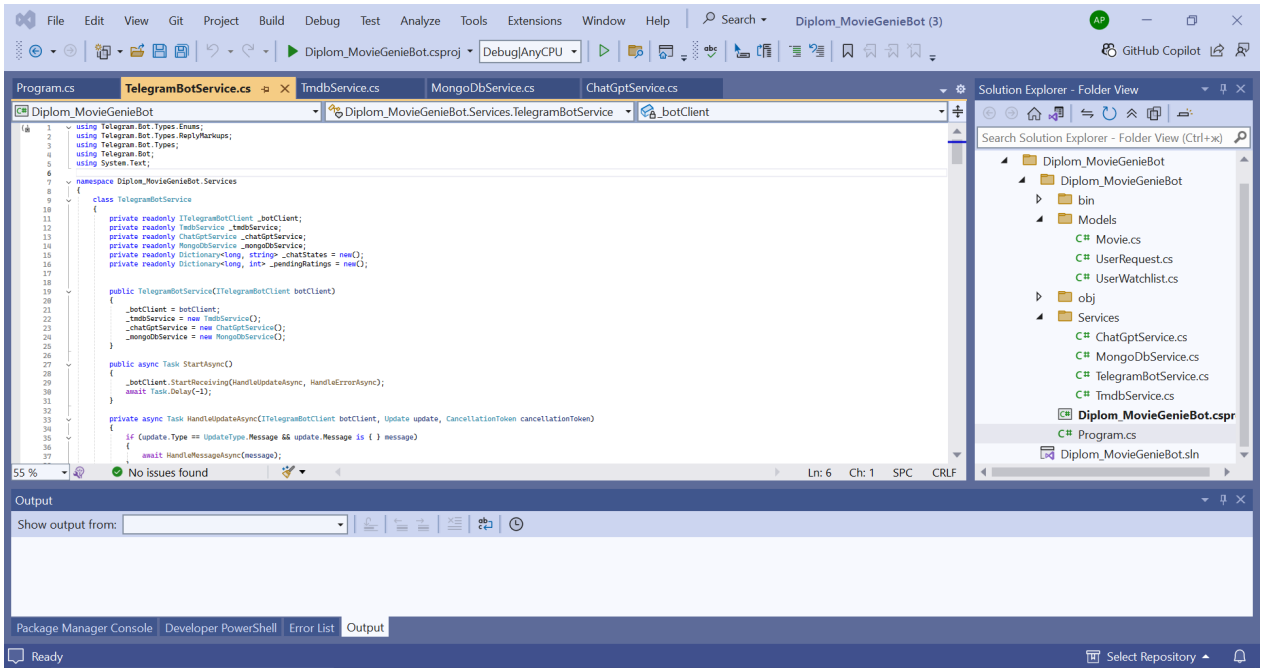

Рисунок 3.1 – Інтерфейс Visual Studio 2022

Мова програмування C# було вибрано як головну мову програмування через її стабільність, сувору типізацію та підтримку необідних бібліотек. Вона має великі можливості для роботи з HTTP-запитами, базами даних і інтеграцій з зовнішніми сервісами. Всі ці аспекти важливі для даного застосунку. Через те, що C# підтримує асинхронність і паралельні процеси, що важливо в час обробки одночасних запитів користувачів, то вона дуже підходить для реалізації логіки роботи ботів.

Також, як було зазначено раніше, в застосунок були інтегровані декілька зовнішніх сервісів. Наприклад, для отримання відповідних рекомендацій фільмів було використано API ChatGPT (gpt-4.1) від OpenAI через HTTP-запити. Щоб зберігати персональні списки користувачів була обрана нереляційна база даних MongoDB, яка працює з JSON-подібними документами. У коді це реалізовано через бібліотеку MongoDB.Driver, яка забезпечує просту роботу з колекціями та працює з документами у форматі BSON. Також застосовується інтеграція з TMDb API, через яку можна отримувати детальну інформацію про фільми.

Використовується основну бібліотеку TelegramBot, яка надає всю підтримку можливостей Telegram Bot API, для формування взаємодії з користувачами через месенджер. Вона дозволяє обробляти повідомлення, кнопки, команди, inline-клавіатури, обробку медіа та багато іншого. Завдяки їй була реалізована логіка діалогу між ботом і користувачем.

3.2 Створення та налаштування бота в Telegram

Для створення власного бота необхідно отримати токен для авторизації та підключення через API. Робиться це за допомогою службового бота, який називається BotFather. Він надає можливість створювати, налаштовувати та керувати ботами в Telegram.

Щоб почати працювати з BotFather, потрібно спершу знайти його в пошуку Telegram за іменем користувача «@BotFather», відкрити з ним чат і запустити діалог, надіславши команду «/start». Після цього бот відразу відповідає списком доступних команд, які допомагають користувачам створювати та налаштовувати ботів.

Процес створення нового бота через BotFather дуже простий і включає лише три основні кроки (рис. 3.2):

Крок 1. Вводимо команду «/newbot», щоб запустити процес реєстрації, після чого BotFather попросить ввести ім’я для бота. Це може бути будь-яке зручне та зрозуміле ім’я, яке буде відображатися в заголовку чату. Його можна змінити у будь-який момент.

Крок 2. Далі необхідно придумати та вказати скорочену назву для посилань для бота, яке має бути унікальним та закінчуватися на «_bot». У випадку, якщо назва зайнята, то потрібно обрати іншу.

Крок 3. У відповідь BotFather надсилає спеціальне посилання для відкриття чату з новим ботом, а також унікальний токен доступу, який є ключем у вигляді рядку символів. Цей токен є важливим, оскільки саме він використовується в коді для підключення до Telegram API, щоб керувати програмно ботом.

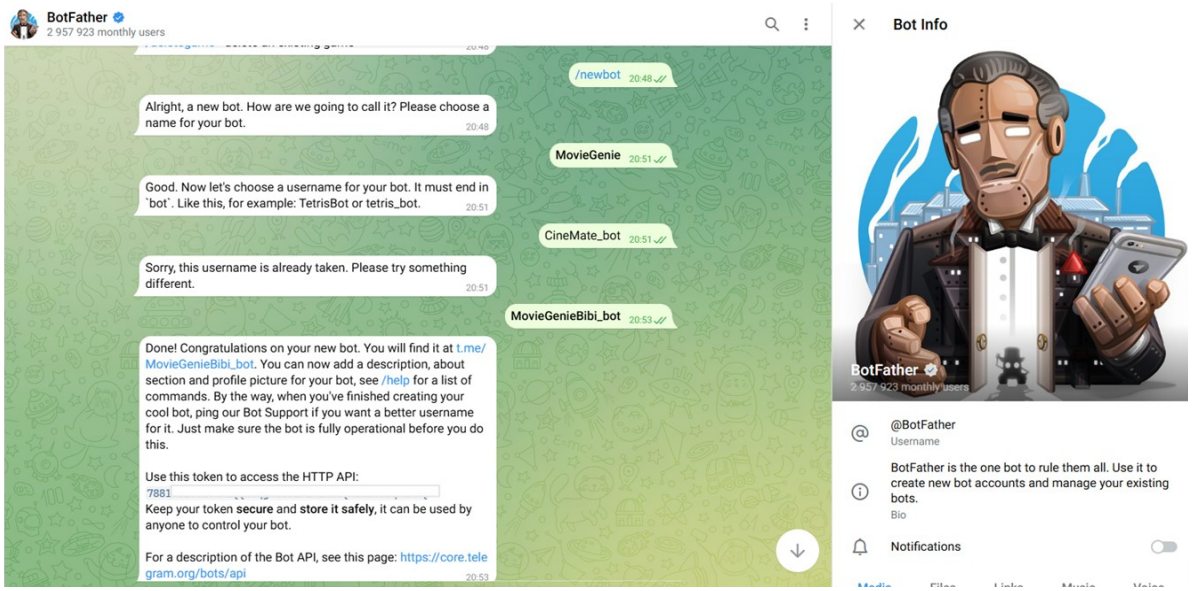

Рисунок 3.2 – Приклад створення бота через BotFather

Після успішного створення бота варто одразу перейти до його базового налаштування. Сервіс BotFather дозволяє змінювати ім’я, опис, коротку інформацію, фотографію, політику конфіденційності та інше. Ці параметри формують загальне враження про бота для користувача. Для цього необхідно надіслати команду «/mybots» та вибрати відповідного бота зі списку. На екрані з’явиться меню налаштувань, де можна провести необхідні поправки (рис. 3.3).

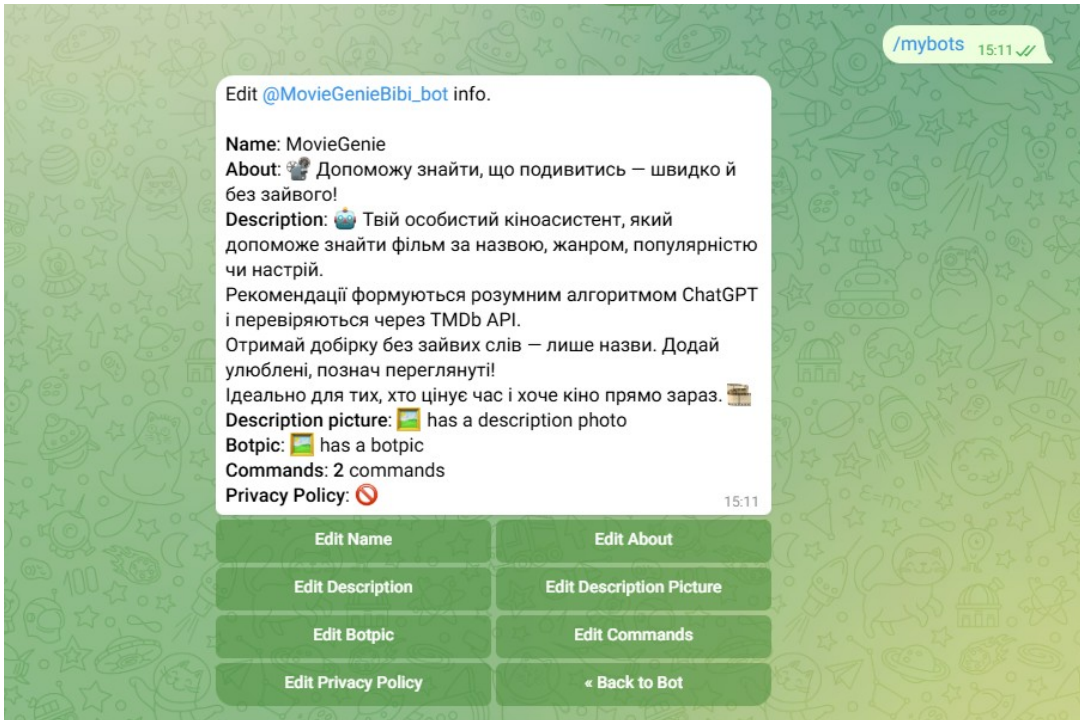

Рисунок 3.3 – Налаштування створеного бота через BotFather

3.3 Отримання доступу до TMDb API

Для отримання доступу до актуальної бази даних з фільмами через TMDb API необхідно мати спеціальний API-ключ, який дозволяє інтегрувати цю платформу у власні проєкти. Цей ключ довзволить нам надсилати запити до відкритих ресурсів TMDb та отримувати детальну інформацію про фільми.

Процес отримання ключа складається з наступних послідовних етапів:

– створити обліковий запис на офіційному сайті TMDb. Для реєстрації необхідно вказати ім’я користувача, пароль та електронну пошту;

– заповнити невелику анкету, вказавши всю необхідну інформацію про застосунок (рис. 3.4), а також обов’язково підтвердити, що API буде використовуватися не в комерційних цілях для отримання безкоштовного доступу;

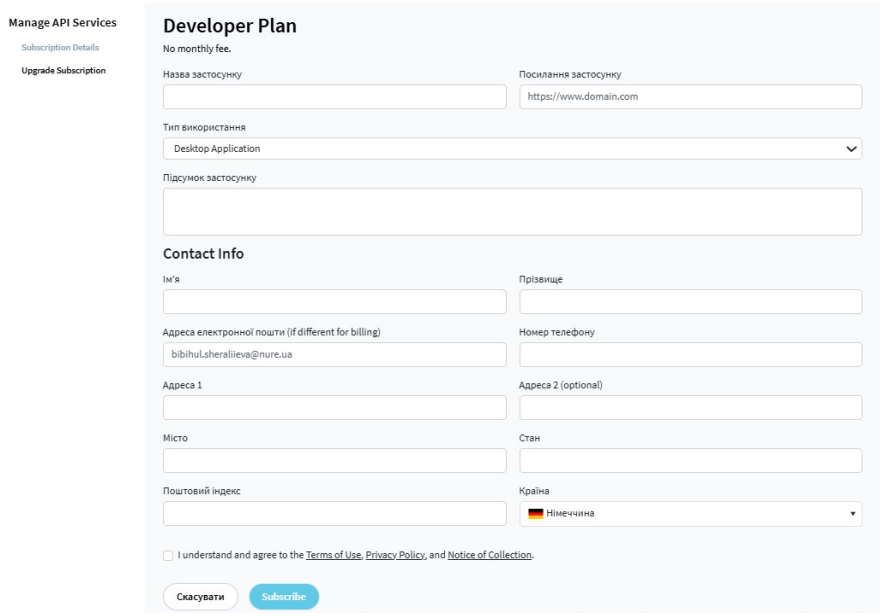

Рисунок 3.4 – Анкета для отримання API-ключа в TMDb API

– після миттєвої обробки анкети користувач отримує унікальний API-ключ. Його можна побачити в особистому кабінеті у розділі «API» (рис. 3.5). Саме цей ключ потрібно використовувати у коді застосунку при формуванні HTTP-запитів до платформи.

Крім цього, в особистому кабінеті можна відстежувати та аналізувати статистику використання TMDb API-ключа. Вона відображає загальну кількість запитів та дату викликів. Такі графіки дозволяють проглядати активність інтеграції, своєчасно виявляти перевищення лімітів або навантаження. На рисунку 3.6 наведена статистика використання доступу до TMDb, яка отримана під час розробки нашого боту в Telegram.

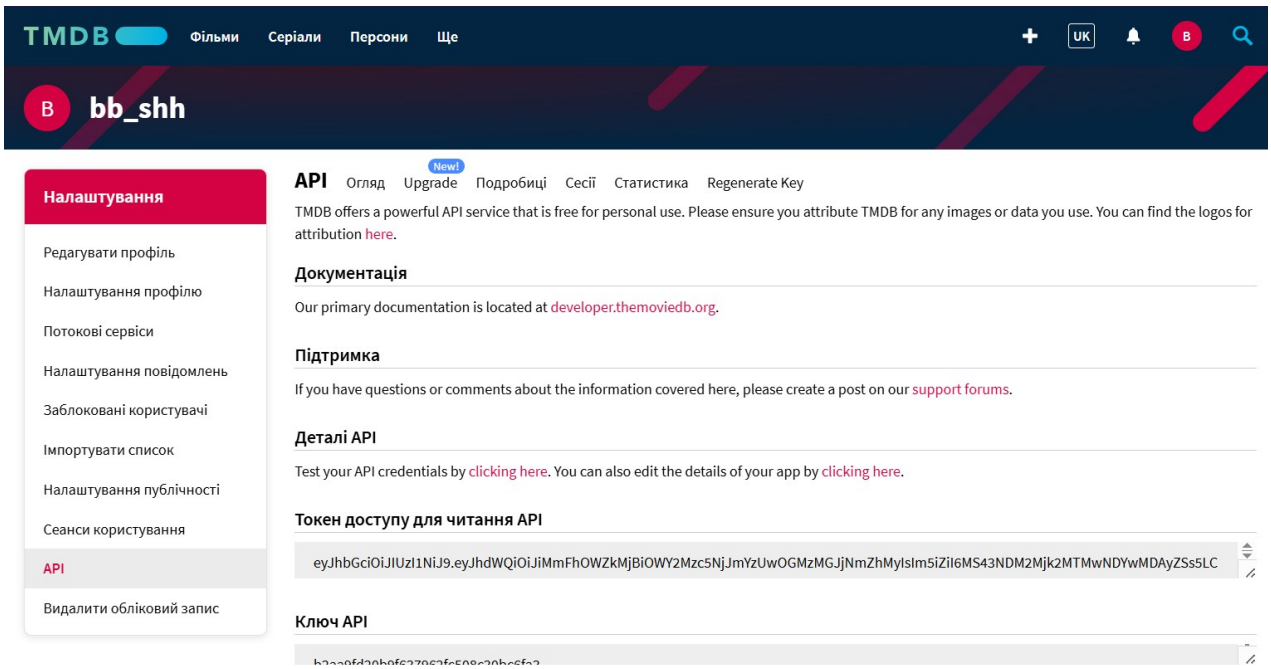

Рисунок 3.5 – Особистий кабінет у розділі «API» з генерованим ключом

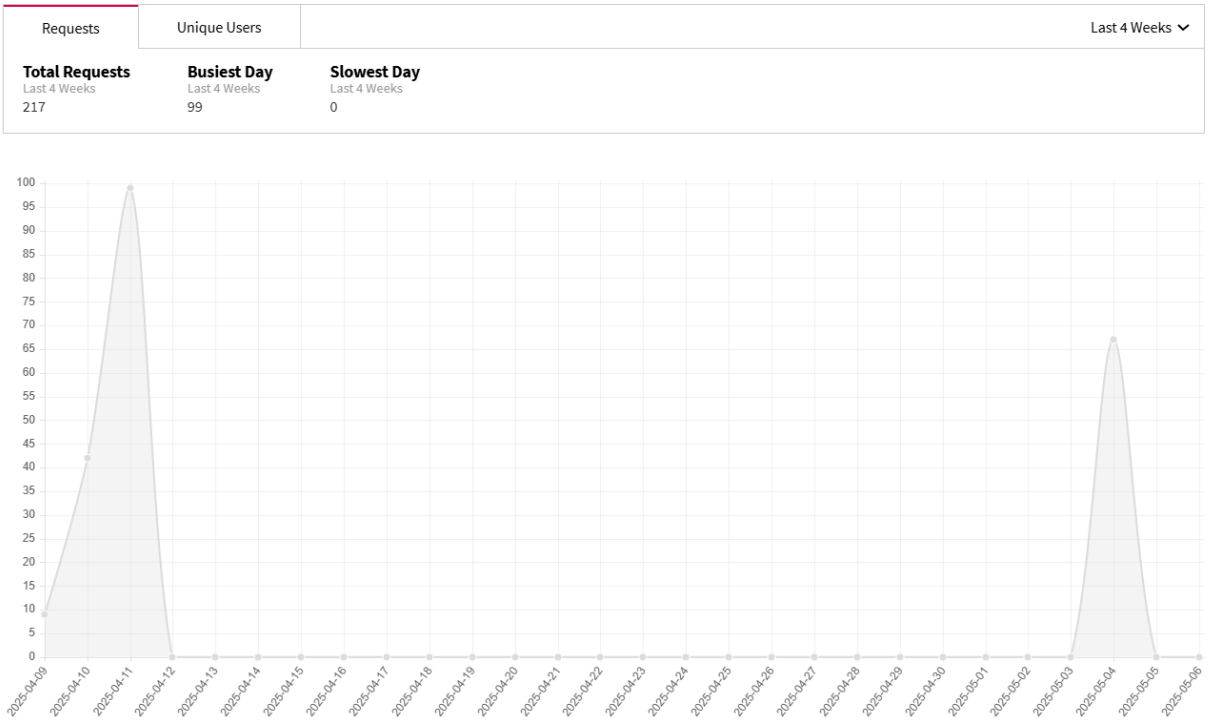

Рисунок 3.6 – Статистика використання API під час розробки Telegram-бота

3.4 Отримання доступу до OpenAI API

Щоб підключити GPT-модель через API від OpenAI, розробнику потрібно отримати спеціальний доступний ключ. Для цього необхідно виконати кілька простих кроків:

Крок 1. Зареєструватися на сайті OpenAI. Для цього потрібно створити обліковий запис, вказавши необхідну інформацію про себе.

Крок 2. Отримати API-ключ. Після реєстрації та підтвердження акаунта отримується унікальний ключ, який надалі використовується для підключення до сервісу (рис. 3.7).

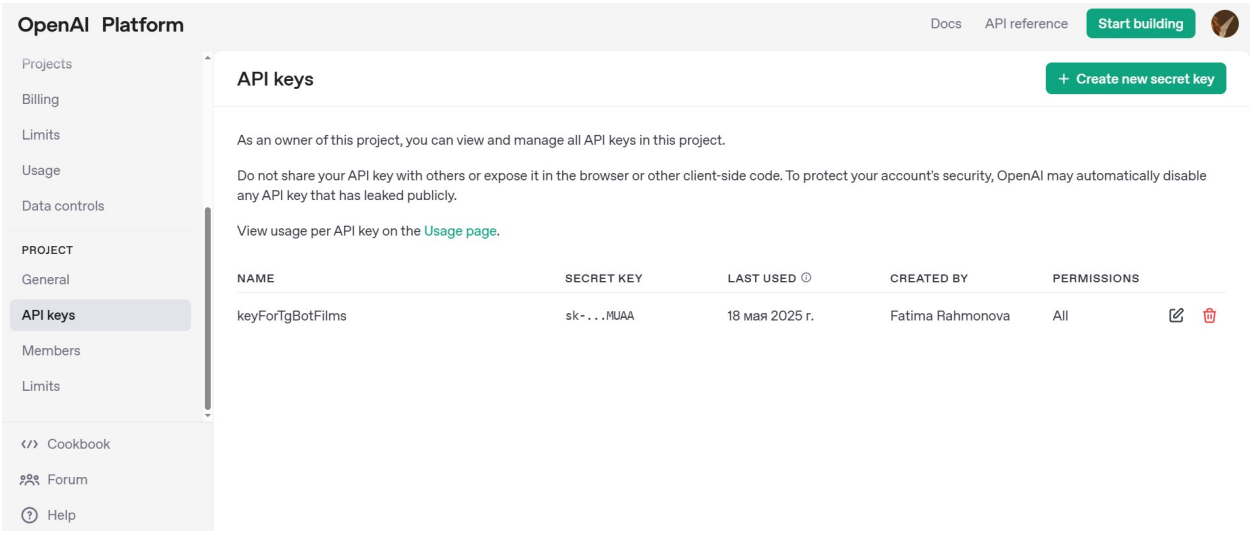

Рисунок 3.7 – Сторінка створення ключів доступу до OpenAI API

Крок 3. Поповнити баланс. Передбачено, що розробник повинен оплатити за використання. Тому необхідно внести кошти відповідно до вибраного тарифу, а згодом надсилати запити.

Після цього API-ключ можна вставити у свій застосунок і почати використовувати можливості GPT у своєму проєкті.

3.5 Створення промпту

Особливу увагу треба приділити на формування спеціального інструктажу (промпт) для того, щоб модель давала якісні рекомендації, які б могли задовільнити бажання користувача. У нашому випадку, це текстовий запит до GPT, де вказується, що вона має має зробити, що саме очікується від відповіді, якого формату вона має бути, який стиль слід зберігати, які дані та обмеження враховувати. Відповідно, успішність генерації залежить переважно від якості промпта: чим точніше, конкретніше та логічніше сформульоване завдання, тим релевантнішими будуть результати. Для створення ефективного промпту, слід дотримуватися структурованого запиту за шаблоном: [Роль] + [Завдання] + [Деталі / Формат / Обмеження].

Спочатку вказується, ким має бути модель, далі що вона повинна зробити, і нарешті як саме має виглядати відповідь. Краще формулювати промпти англійською мовою, оскільки більшість LLM, зокрема GPT, краще інтерпретують і виконують запити саме англійською.

Ще одна корисна техніка – це ланцюжок думок, яка змушує модель міркувати поступово. Для цього варто використовувати фразу «думай крок за кроком» або її еквіваленти. Також варто повторювати ключові інструкції у межах промпта кілька разів, щоб модель чітко засвоїла обмеження.

У рамках розробки Telegram-бота для підбору фільмів, де модель GPT від OpenAI API використовується для генерації рекомендацій, було передбачено два основні сценарії:

– текстовий запит користувача у довільній формі, наприклад, «Хочу подивитися щось пригодницьке. Порадь, щось атмосферне та естетичне з подорожами у часі»;

– генерація рекомендацій на основі історії вподобань користувача, які зберігаються в базі даних (список зацікавлених та переглянутих фільмів).

Для кожного з цих сценаріїв були розроблені кінцеві промпти після декількох коригувань, які продемонстровані відповідно у лістингах 3.1 та 3.2.

Лістинг 3.1 Промпт для генерації на основі текстового запиту:

You are an intelligent and emotionally aware movie assistant that recommends films based on user preferences, moods, interests, or examples of previously enjoyed movies. Your task is to carefully analyze the user's input and suggest exactly 15 relevant movie titles (in English). Avoid overused or overly popular suggestions unless they are a perfect fit. Do not make up any movies, they must be found in The Movie Database (TMDb). Do not add any descriptions, comments, or emojis. Exclude movies specifically mentioned in the user's request. Each entry should be in the format: "Number. Original Title (Year of Release)". Help the user find their next favorite movie.

Лістинг 3.2 Промпт на основі визначених фільмів користувача:

You are an intelligent and emotionally aware movie assistant.

Your task is to recommend exactly 20 movies (original English titles) that the user will like, based on the movies he liked and disliked. You must analyze each movie from his lists step by step, taking into account the following aspects: genre, main themes, plot complexity, emotional tone, pacing, and visual or stylistic appeal of the movies. Use this analysis to create a psychological profile of the user's tastes and generate a list of recommendations, providing genre, stylistic, and emotional diversity within their preferences.

Do not invent any movies, they must be found in The Movie Database (TMDb). It is strictly forbidden to include movies that are already in the user's lists in a new list!! Do not add any descriptions, comments, or emojis. Each entry must be in format: "Number. English Title (Year of Release)".

Movies that the user is interested in but has not yet watched:

{LIST_OF_INTERESTED_FILMS}.

User’s watched movies with ratings (10 being the highest):

{LIST_OF_WATCHED_FILMS}.

3.6 Налаштування та програмна реалізація застосунку

Для створення затосунку було обрано шаблон Console Application на базі .NET Core 8. Загальна структура проєкту у вигляді дерева файлів зображено на рисунку 3.8.

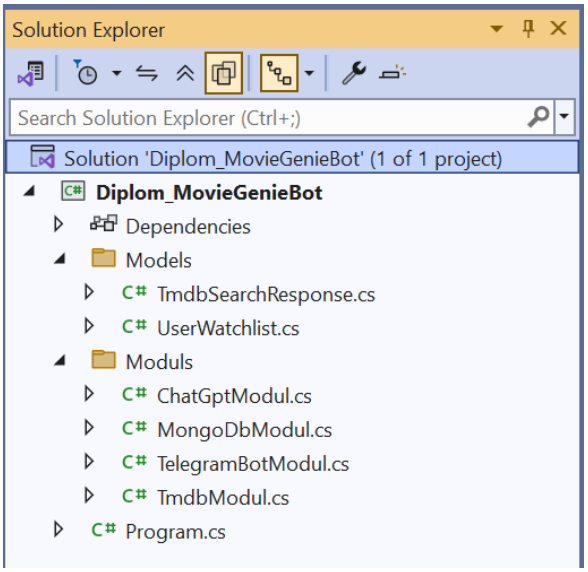

Рисунок 3.8 – Загальна структура проекту в Visual Studio

Як бачимо, проєкт складається з наступних логічних блоків:

– Program.cs – головний вхідний файл програми, у якому запускається та налаштовується бот;

– Models – містить класи-моделі, що описують дані, з якими працює застосунок;

– Modules – містить основні функціональні модулі, які відповідають за взаємодію з Telegram API, базою даних MongoDB, сервісом TMDb та API ChatGPT.

Далі коротко буде описано кожну з них.

Головний файл «Program.cs» виконує ініціалізацію Telegram-бота, задає токен доступу до Bot API, запускає нескінченний цикл обробки запитів через метод StartAsync. Після запуску всі події обробляються через делегати.

Клас «TmdbSearchResponse» слугує моделлю, що містить набір фільмів, отриманих у відповідь на запит до TMDb API. Він містить список об’єктів «TmdbMovieDetails», які, в свою чергу, включають всю необхідну інформацію: назву, опис, дату виходу, жанри, рейтинг, постер тощо. Для коректного зіставлення JSON-властивостей використовуються атрибути JsonProperty (рис. 3.9).

```
9     public class TmdbSearchResponse
10    {
11        public required List<TmdbMovieDetails> Results { get; set; }
12    }
13
14    public class TmdbMovieDetails
15    {
16        public int Id { get; set; }
17        public required string Title { get; set; }
18        [JsonProperty("original_title")]
19        public required string OriginalTitle { get; set; }
20        [JsonProperty("release_date")]
21        public required string ReleaseDate { get; set; }
22        public required List<TmdbGenre> Genres { get; set; }
23        public required string Overview { get; set; }
24        [JsonProperty("poster_path")]
25        public required string PosterPath { get; set; }
26        [JsonProperty("vote_average")]
27        public double VoteAverage { get; set; }
28    }
29
30    public class TmdbGenre
31    {
32        public int Id { get; set; }
33        public required string Name { get; set; }
34    }
35}
```

Рисунок 3.9 – Обробка відповіді від TMDb API

Клас «UserWatchlist» представляє собою модель користувача, який взаємодіє з ботом. В цій моделі зберігається список улюблених фільмів користувача та його chat_id. Для зручності роботи з MongoDB, використовуються атрибути BsonElement, BsonId і BsonDateTimeOptions, які дозволяють керувати зберіганням у BSON-форматі.

Клас «TelegramBotModul» містить центральну логіку Telegram-бота. Він відповідає за прийом і обробку повідомлень від користувача, реагування на натискання кнопок, взаємодію з іншими класами. Його основні функції:

– відображає головне меню бота з кнопками для основних дій через класи ReplyKeyboardMarkup та InlineKeyboardMarkup;

– відповідає за формування і надсилання списку фільмів (наприклад, популярні, рекомендовані або збережені користувачем);

– через метод HandleMessageAsync оброблює вхідні повідомлення користувача, тобто визначає тип запиту користувача (наприклад, пошук фільму, запит рекомендації, перегляд списку) та викликає відповідні методи;

– через метод HandleCallbackQueryAsync аналізує дані callback-подій та виконує відповідну логіку (наприклад, надсилання детального опису фільму, додавання або видалення зі списку, оновлення перегляду тощо);

– через словник _chatStates зберігається, в якому стані перебуває чат. Наприклад, очікування назви фільму або оцінки, очікування введення опису для генерації рекомендації.

«MongoDbModul» відповідає за зберігання та обробку користувацьких списків фільмів у БД MongoDB. Він забезпечує додавання фільмів до зацікавлених або переглянутих, отримання списків, видалення фільмів, перевірку наявності фільму в списках. Дані зберігаються у колекції «UserWatchlist», яка містить ChatId користувача та відповідні списки. При ініціалізації створюється підключення до БД та унікальний індекс за ChatId.

Клас «TmdbModul» забезпечує пошук фільмів через зовнішнє TMDb API. Робити пошук фільмів за назвою, жанром або категорією та отримувати розширену інформацію про конкретний фільм, включно з описом, постером, рейтингом, жанрами та трейлером (рис. 3.10) – це є його основні завдання.

Клас також обробляє результати, які отримані від ChatGPT. Оскільки відповідь є нумерованим списком назв фільмів англійською мовою та їх року випуску, нам спочатку потрібно витягти їх з тексту за допомогою регулярного виразу. Результатом є список кортежів типу (назва, рік). Далі асинхронно робиться пошук кожного фільму через API TMDb за назвою та роком, або, за необхідності, лише за назвою. Таким чином, користувач отримує локалізовані рекомендації на основі відповідей GPT (рис. 3.11).

```
public async Task SendMovieDetailsAsync(ITelegramBotClient botClient, long chatId, int movieId, bool fromWatchlist = false)
{
    string detailsUrl = $"{BaseUrl}/movie/{movieId}?api_key={ApiKey}&language=uk-UA";
    string videosUrl = $"{BaseUrl}/movie/{movieId}/videos?api_key={ApiKey}&language=uk-UA";

    var movie = await FetchFromApi<TmdbMovieDetails>(detailsUrl);
    if (movie == null)
    {
        await botClient.SendMessage(chatId, "Не вдалося отримати інформацію про фільм 😕");
        return;
    }

    var videos = await FetchFromApi<TmdbVideoResponse>(videosUrl);
    if (videos?.Results == null || videos.Results.Count == 0)
    {
        string fallbackUrl = $"{BaseUrl}/movie/{movieId}/videos?api_key={ApiKey}&language=en-US";
        videos = await FetchFromApi<TmdbVideoResponse>(fallbackUrl);
    }

    var trailer = videos?.Results?.FirstOrDefault(v => v.Type == "Trailer" && v.Site == "YouTube");
    string trailerUrl = trailer != null ? $"https://www.youtube.com/watch?v={trailer.Key}" : "Трейлер не знайдено 😕";

    string releaseYear = !string.IsNullOrEmpty(movie.ReleaseDate)
        ? movie.ReleaseDate.Split('-')[0]
        : "N/A";

    string rating = movie.VoteAverage > 0
        ? $"{movie.VoteAverage:F1}/10"
        : "Немає рейтингу";

    string genresText = movie.Genres != null && movie.Genres.Any()
        ? string.Join(", ", movie.Genres.Select(g => g.Name))
        : "Невідомо";

    string? posterUrl = !string.IsNullOrEmpty(movie.PosterPath)
        ? $"https://image.tmdb.org/t/p/w500{movie.PosterPath}"
        : null;

    string messageText = $"🎬 <b>{movie.Title} ({releaseYear})</b>\n\n" +
                         $"⭐ Рейтинг IMDb: {rating}\n\n" +
                         $"🎭 Жанр: {genresText}\n\n" +
                         $"📖 <i>{movie.Overview}</i>\n\n" +
                         $"▶ <a href=\"{trailerUrl}\">Дивитися трейлер</a>";
```

Рисунок 3.10 – Формування повної інформації про фільм з TMDb API

```
private List<(string Title, int Year)> ParseMovieTitlesFromGptResponse(string gptResponse)
{
    var movies = new List<(string, int)>();
    var regex = new Regex(@"^\d+\.\s*(.+?)\s*\((\d{4})\)", RegexOptions.Multiline);

    foreach (Match match in regex.Matches(gptResponse))
    {
        if (match.Success && int.TryParse(match.Groups[2].Value, out int year))
        {
            movies.Add((match.Groups[1].Value.Trim(), year));
        }
    }

    return movies;
}

private async Task<TmdbMovieDetails> FindMovieWithUkrainianTitle(string englishTitle, int year)
{
    string searchUrl = $"{BaseUrl}/search/movie?api_key={ApiKey}" +
                       $"&language=en-US&query={HttpUtility.UrlEncode(englishTitle)}" +
                       $"&year={year}";

    var searchResult = await FetchFromApi<TmdbSearchResponse>(searchUrl);

    // Якщо не знайдено - шукаємо без року
    if (searchResult?.Results == null || !searchResult.Results.Any())
    {
        searchUrl = $"{BaseUrl}/search/movie?api_key={ApiKey}" +
                    $"&language=en-US&query={HttpUtility.UrlEncode(englishTitle)}";
        searchResult = await FetchFromApi<TmdbSearchResponse>(searchUrl);
    }

    if (searchResult?.Results == null || !searchResult.Results.Any())
        return null;

    // Беремо найточніший результат (перший у списку)
    var movieId = searchResult.Results[0].Id;

    // Отримуємо деталі українською
    string detailsUrl = $"{BaseUrl}/movie/{movieId}?api_key={ApiKey}&language=uk-UA";
    return await FetchFromApi<TmdbMovieDetails>(detailsUrl);
}
```

Рисунок 3.11 – Методи обробки відповіді від мовної моделі

Клас «ChatGptModul» містить логіку звернення до GPT-моделі від API OpenAI. Його основна функція – це формувати рекомендації фільмів на основі побажань користувача у текстовій вільній формі або з урахуванням його вподобань щодо фільмів із бази даних MongoDB. Залежно від обраного сценарію (промпту), клас генерує відповідні запити та передає їх до моделі через HTTP-запит (рис. 3.12).

```
98      private async Task<string> SendPromptToGptAsync(string systemPrompt, string userInput)
99      {
100         try
101         {
102             using var client = new HttpClient();
103             client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", _apiKey);
104
105             var requestData = new
106             {
107                 model = "gpt-4.1-2025-04-14",
108                 messages = new[]
109                 {
110                     new { role = "system", content = systemPrompt },
111                     new { role = "user", content = userInput }
112                 }
113             };
114
115             var content = new StringContent(JsonConvert.SerializeObject(requestData), Encoding.UTF8, "application/json");
116             var response = await client.PostAsync(_OpenAiUrl, content);
117
118             if (!response.IsSuccessStatusCode)
119             {
120                 Console.WriteLine("Статус код: " + response.StatusCode);
121                 var err = await response.Content.ReadAsStringAsync();
122                 Console.WriteLine("Помилка GPT: " + err);
123                 return "На жаль, не вдалося отримати рекомендацію 😕";
124             }
125
126             var responseString = await response.Content.ReadAsStringAsync();
127             dynamic jsonResponse = JsonConvert.DeserializeObject<dynamic>(responseString);
128             string reply = jsonResponse.choices[0].message.content.ToString();
129
130             return CleanResponse(reply);
131         }
132         catch (Exception ex)
133         {
134             Console.WriteLine("Виняток при запиті до GPT: " + ex.Message);
135             return "На жаль, не вдалося отримати рекомендацію 😕";
136         }
137     }
```

Рисунок 3.12 – Формування та надсилання HTTP-запиту до GPT-моделі

3.7 Ілюстрація роботи бота та його тестування

Для взаємодії з ботом в Telegram достатньо мати встановлений застосунок Telegram на мобільному пристрої або можна скористатися вебверсією месенджера.

Щоб знайти бота, необхідно спочатку ввести його ім'я користувача у вікні пошуку. Реалізований бот має назву «MovieGenieBibi_bot». Після

вибору відповідного боту у списку результатів відкривається діалогове вікно з описом про себе (рис. 3.13).

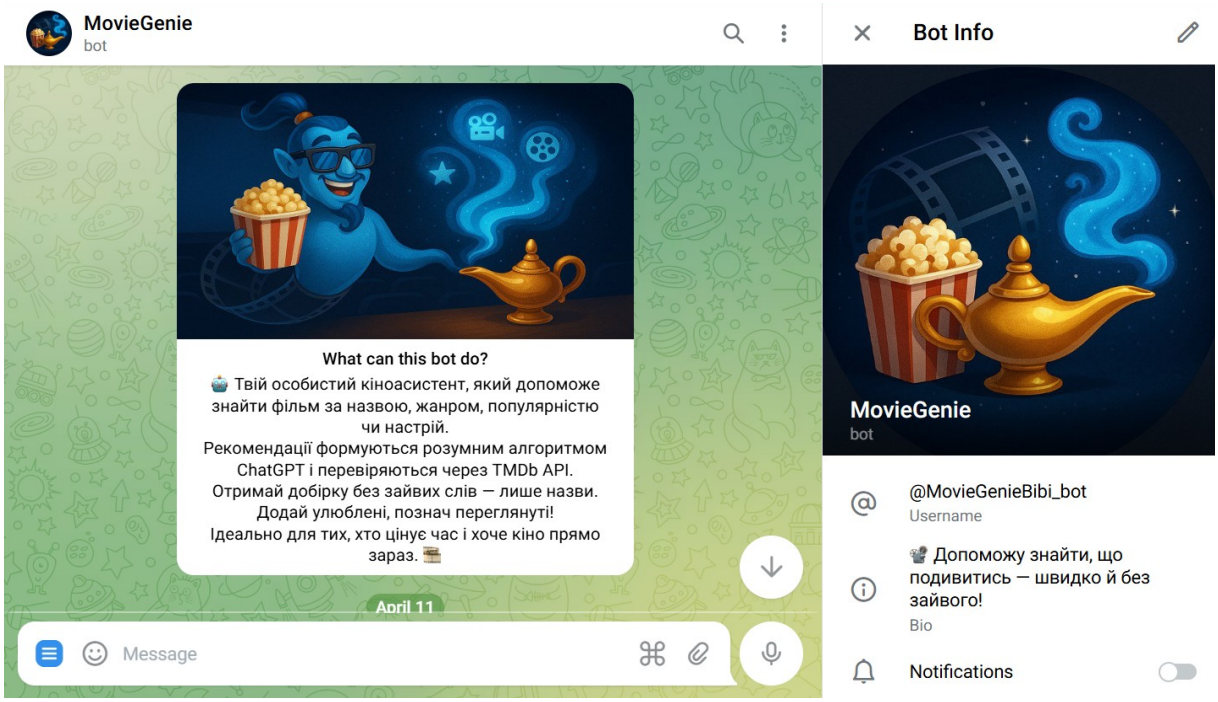

Рисунок 3.13 – Діалогове вікно бота з особистою інформацією

Щоб почати працювати з ботом, необхідно його запустити. Для цього необхідно відправити команду «/start» (рис. 3.14). Одразу відправляється повідомлення з привітанням і після з'являється кнопка «Меню» з основними командами:

– знайти фільм;
– випадковий фільм;
– рекомендації від ШІ;
– мій список.

Бот використовує інлайн-клавіатуру та callback-обробку для реалізації зручної взаємодії користувача з ботом. Якщо користувач обираю опцію «Знайти фільм», то йому пропонується три способи пошуку у вигляді інлайн-кнопок: за назвою, за жанром та за категоріями (рис. 3.15).

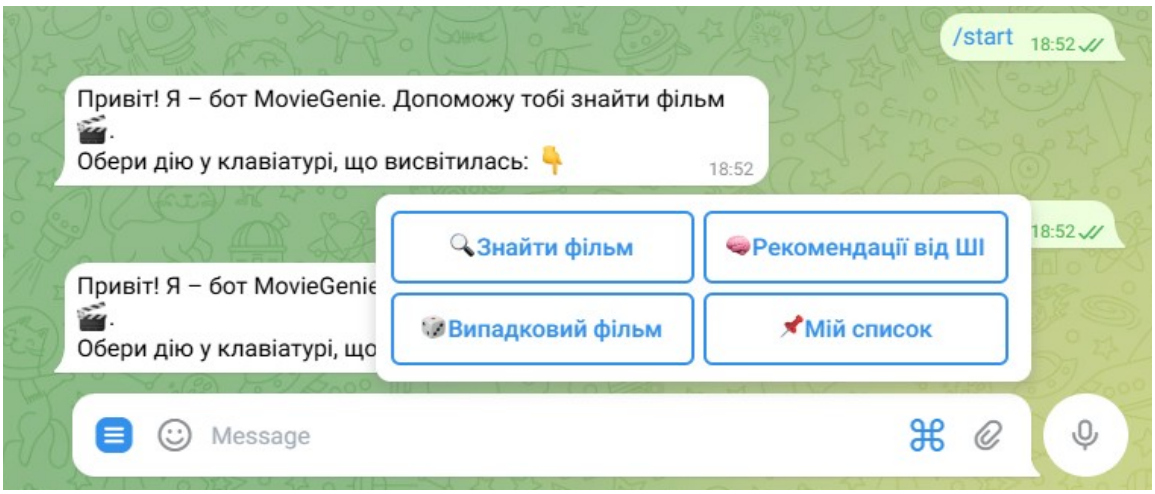

Рисунок 3.14 – Перша взаємодія з ботом

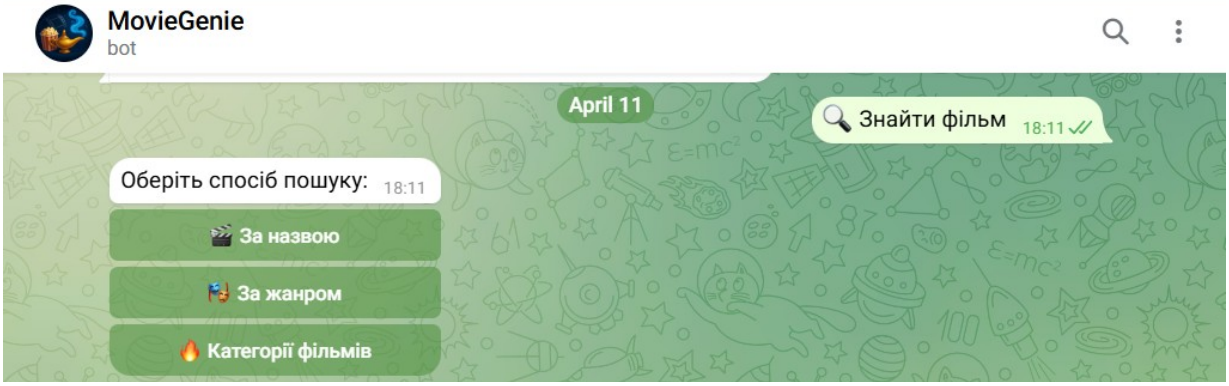

Рисунок 3.15 – Способи пошуку фільмів

У випадку вибору кнопки «За назвою» бот надсилає повідомлення, де просить ввести вручну повну або часткову назву фільму. Після введення, наприклад, «Гаррі Поттер», бот виконує запит до TMDb API та формує список фільмів, які відповідають цій назві (рис. 3.16). Такий приклад обрано не випадково, адже існує кілька різних фільмів із цією назвою.

Список формується у вигляді вбудованих кнопок із назвою фільму та роком його виходу. Користувач може натиснути на будь-яку з них, щоб переглянути повну інформацію про фільм. Якщо бот не зміг знайти необхідний фільм або дані неповні, то він виводить відповідне повідомлення (рис. 3.17).

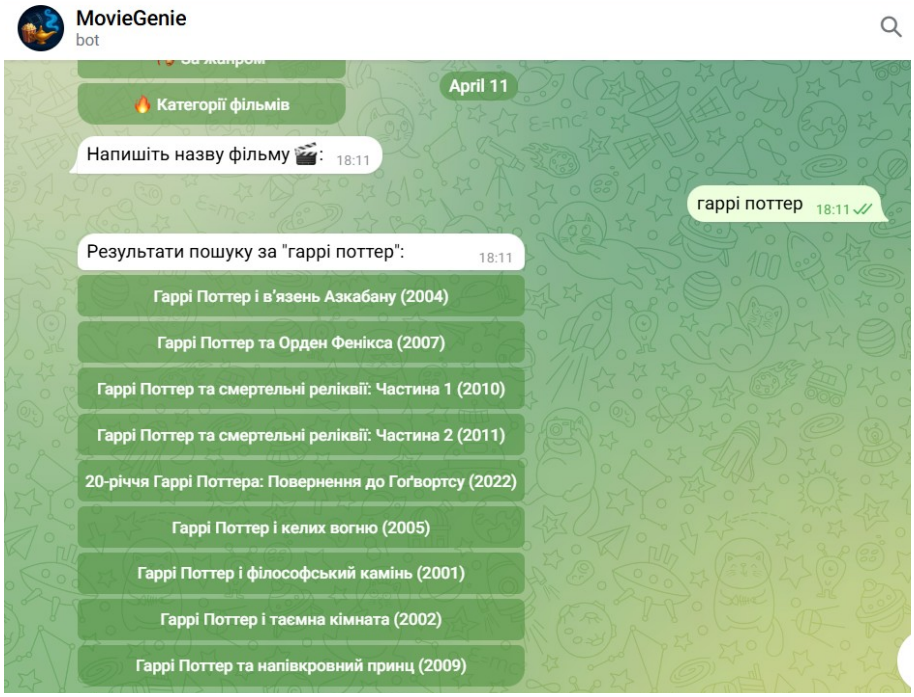

Рисунок 3.16 – Успішний пошук за назвою фільму та видача знайдених варіантів

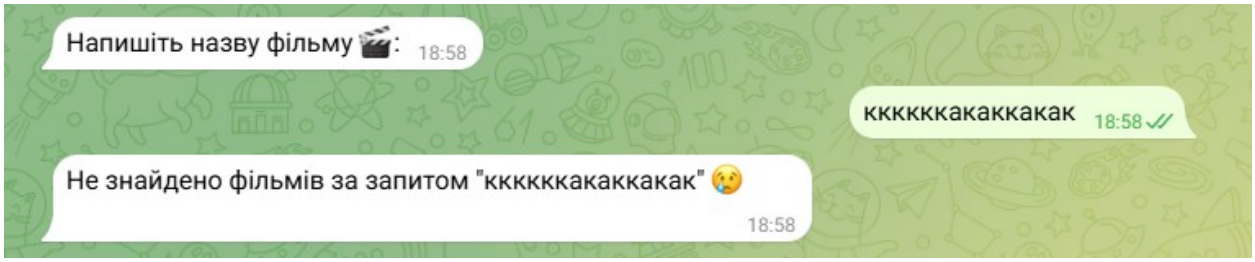

Рисунок 3.17 – Невдалий пошук за назвою фільму

Якщо користувач хоче отримати добірку фільмів одного жанру, то необхідно обрати функцію «За жанром». У відповідь бот просто надає список типових популярних жанрів, серед яких треба визначити один, наприклад, «Історичний». Виконується запит до бази, формується добірка з 20 фільмів відповідної тематики та виводиться так само у вигляді вбудованих кнопок, щоб швидко переглянути варіанти, не гублячись в надміру інформаії (рис. 3.18).

Коли користувач не має ясного запиту, а просто хоче обрати щось актуальне на сьогодні, то він може скористатися пошуком «За категоріями», завдяки якому можна швидко знайти фільми з певного тематичного набору.

Наприклад, найпопулярніші, найбільш рейтингови, нещодавно додані або які повинні скоро з'явитися (рис. 3.19).

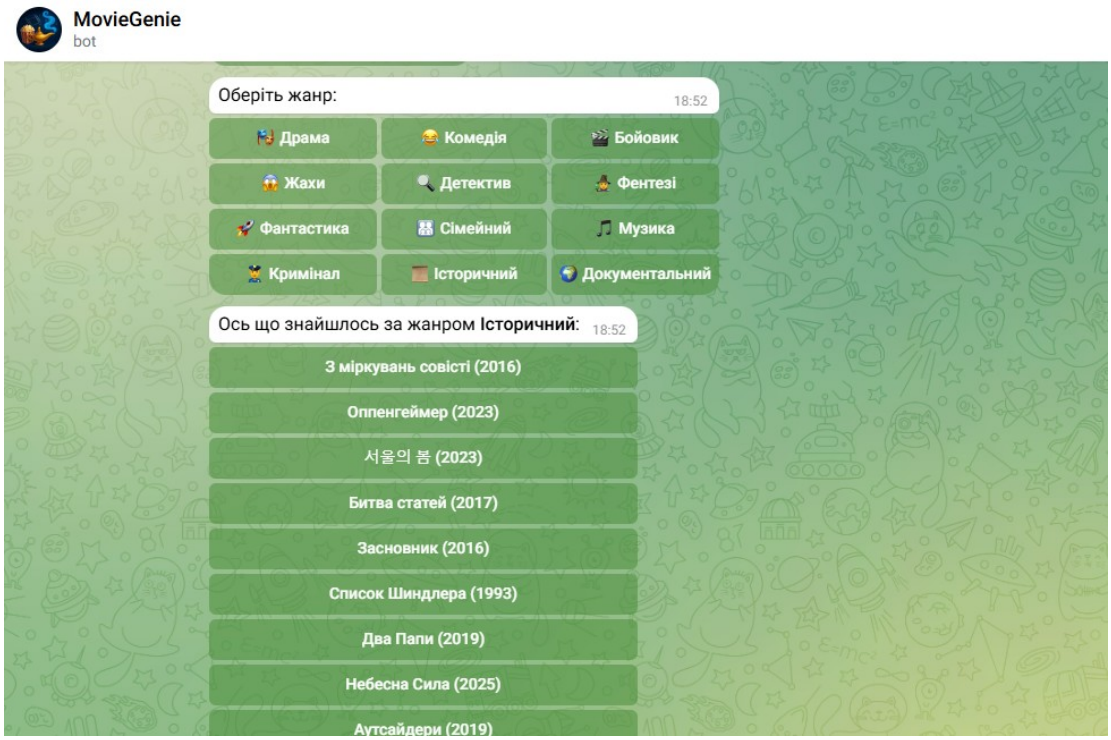

Рисунок 3.18 – Результат пошуку фільмів за жанром «Історичний»

Рисунок 3.19 – Пошук популярних фільмів сьогодні

Як було продемонстровано вище, майже кожен результат – це окрема кнопка з назвою фільму. Користувач може натиснути на будь-яку з них, щоб отримати картку з повною інформацією про фільм, що містить постер, назву українською (за наявності), рік випуску, рейтинг за версією IMDb, жанр, короткий опис і посилання на трейлер, а також кнопки для додавання до списку зацікавлених або переглянутих (рис. 3.20). Це дає змогу швидко оцінити, чи підходить обраний фільм для перегляду, та вести особистий список вподобань для перегляду.

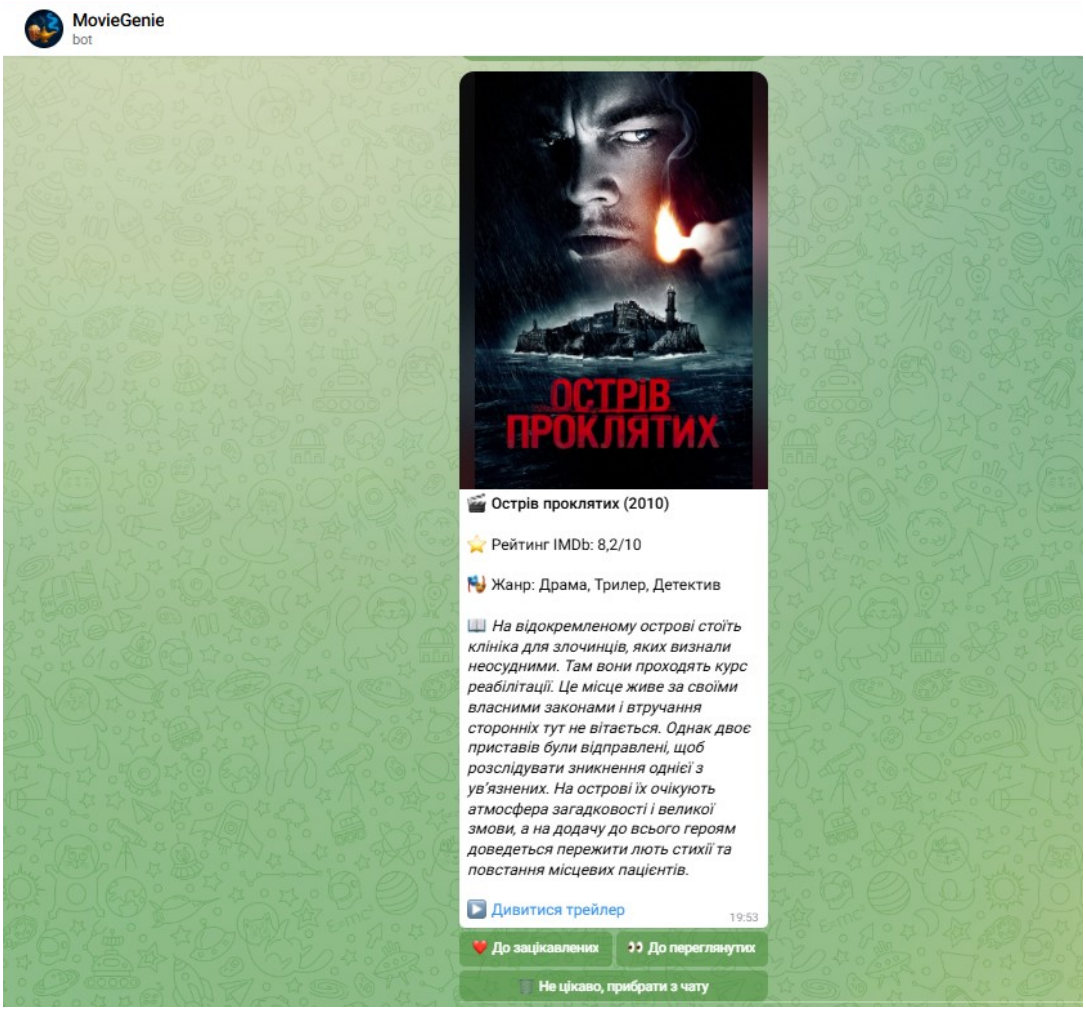

Рисунок 3.20 – Пошук популярних фільмів сьогодні

Окрім основних функціональних кнопок, додано ще одне зручне доповнення – кнопку «Не цікаво, прибрати з чату». Її призначення суто утилітарне, тобто ця кнопка забезпечує користувачу самостійно очищати чат

від фільмів, які не викликали жодного інтересу. Це дає змогу тримати інтерфейс у чистоті та зосередитись лише на тих варіантах, які сподобалися.

Також варто зазначити, що під час формування посилання на трейлер пріоритет надається українській мові. Якщо для фільму доступний україномовний трейлер, бот обирає саме його, у разі її відсутності надається англомовна версія. У прикладі присутнє посилання на трейлер на YouTube, тому можна його переглянути (рис. 3.21).

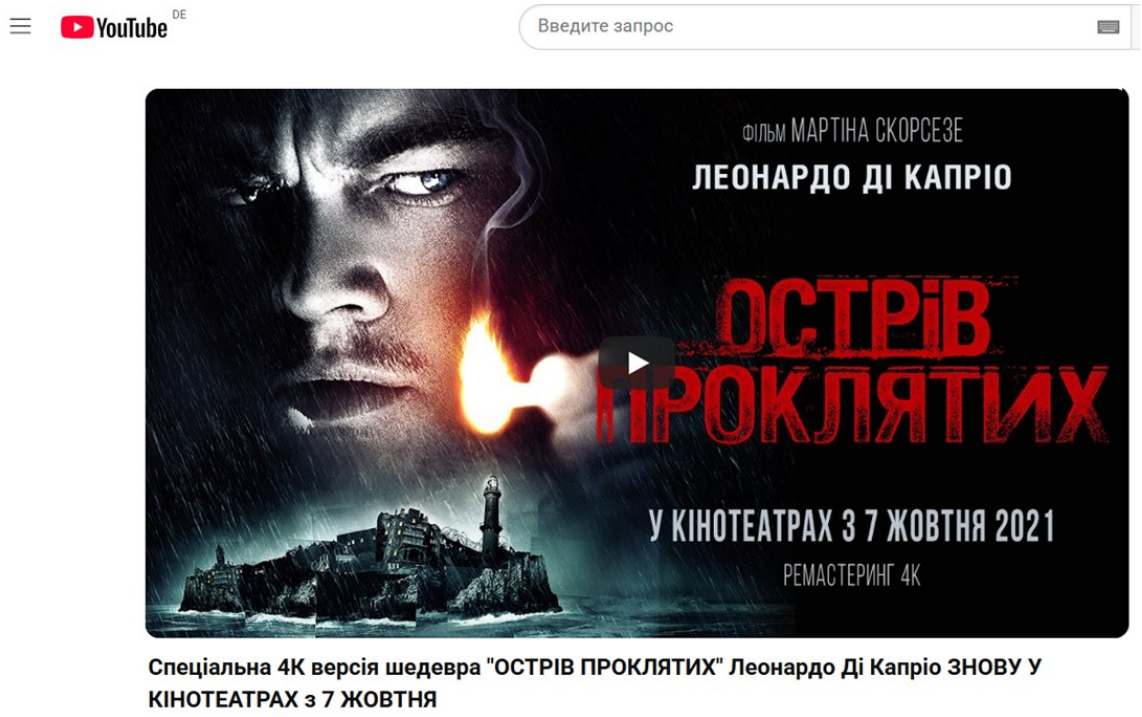

Рисунок 3.21 – Трейлер фільму на сайті YouTube

Якщо користувач не має ніяких конкретних побажань, то для таких випадків передбачена опція «Випадковий фільм». Вона дозволяє отримати рекомендацію абсолютно рандомного фільму. Для цього бот звертається до API TMDb з параметром сортування за популярністю та випадково вибраною сторінкою результатів (від 1 до 500). І зразу надсилає користувачу повну картку з описом (рис. 3.22).

Однією з цікавих функцій бота є опція «Рекомендації від ШІ», яка реалізована за допомогою моделі GPT від OpenAI. В цьому режимі користувач може написати абсолютно довільне текстове повідомлення,

наприклад, «Хочу подивитися фільм, який схожий на «Щоденник пам'яті», про кохання, але не з хеппі-ендом» (рис. 3.23).

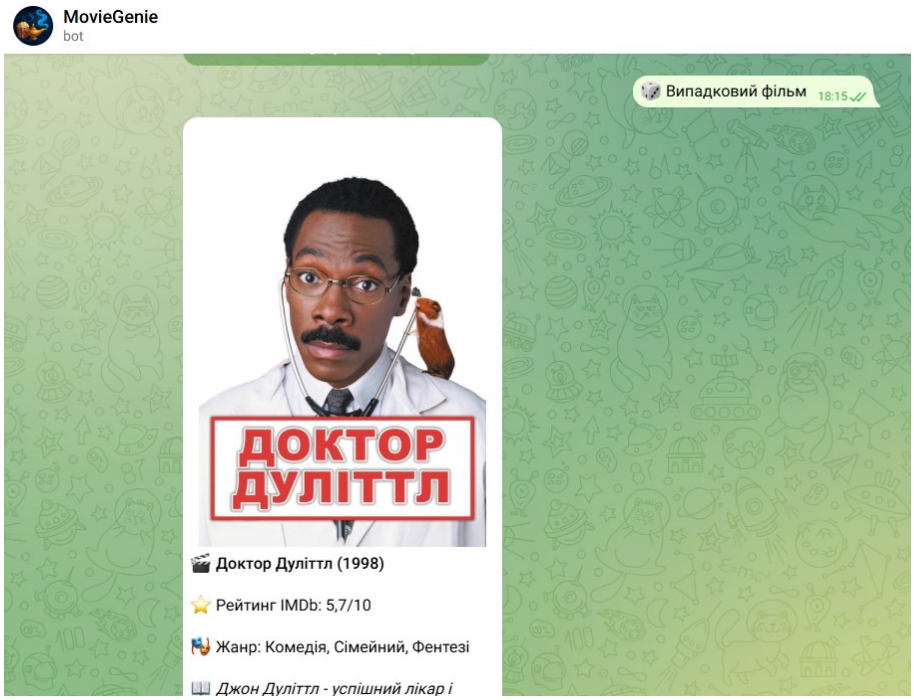

Рисунок 3.22 – Рекомендований випадковий фільм, згенерований ботом

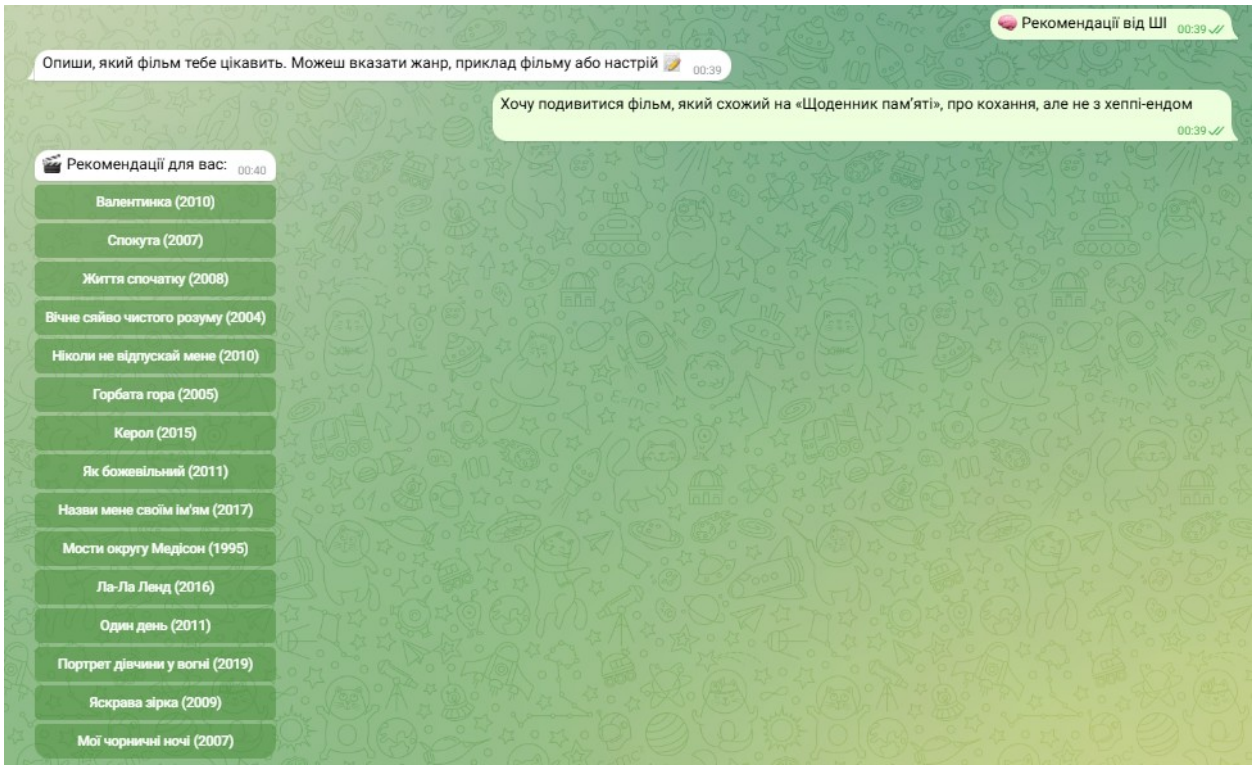

Рисунок 3.23 – Вивід результату рекомендацій GPT-моделі на основі текстового запиту

Після чого бот надсилає відповідний запит до GPT. Модель аналізує її та повертає список рекомендованих фільмів, які виводяться у зручному вигляді. У випадку з запитом про драму без щасливого фіналу, бот підібрав дуже вдалі фільми, які справді відповідають бажаному контексту. Усі фільми були проаналізовані та можна дійсно затвердити, що вони викликають сильні емоції, мають схожу атмосферу та несподівану кінцівку.

З цього випливає, що такий підхід дозволяє враховувати не лише конкретні фільми, а й настрій або контекст запиту. Особливо корисною вона є для людей, яким легше сформулювати текстово свої певні бажання або настрій.

Як зазначалось раніше, кожен фільм можна додати до особистого списку фільмів «Зацікавлені» або «Переглянуті». Уся інформація зберігається в базі даних MongoDB, що дозволяє надалі аналізувати вподобання користувача для ще точніших рекомендацій. Якщо фільм зацікавив користувача, йому спободався опис, то він натискає кнопку «До зацікавлених», після чого бот зберігає дану інформацію та відправляє підтверджувальне повідомлення. Коли фільм уже переглянуто, то користувач повинен обрати кнопку «До переглянутих». Після цього бот надсилає повідомлення з проханням надати йому оцінку від 1 до 10, де 1 – найнижча, а 10 – найвища (рис. 3.24). Потім ці оцінки будуть мати вплив на формування персональних рекомендацій.

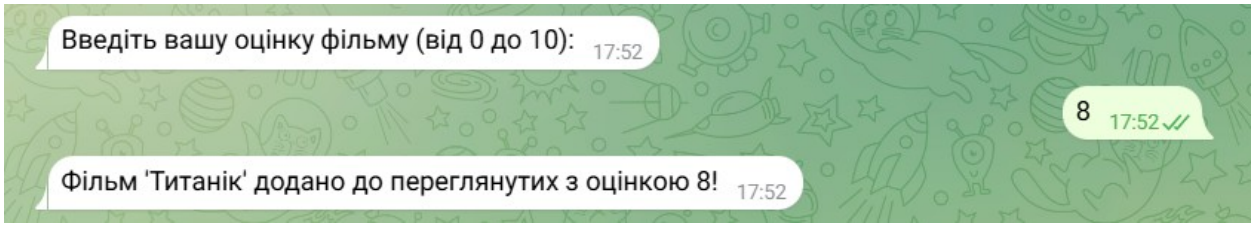

Рисунок 3.24 – Надання оцінки до переглянутого фільму

Варто помітити, що після додавання фільму до списку, під час наступного звернення до нього картка з детальною інформацією буде мати

лише одну кнопку, яка повідомлятиме, що даний фільм додано до зацікавлених із зазначенням дати або переглянутих з відповідною оцінкою (рис. 3.25).

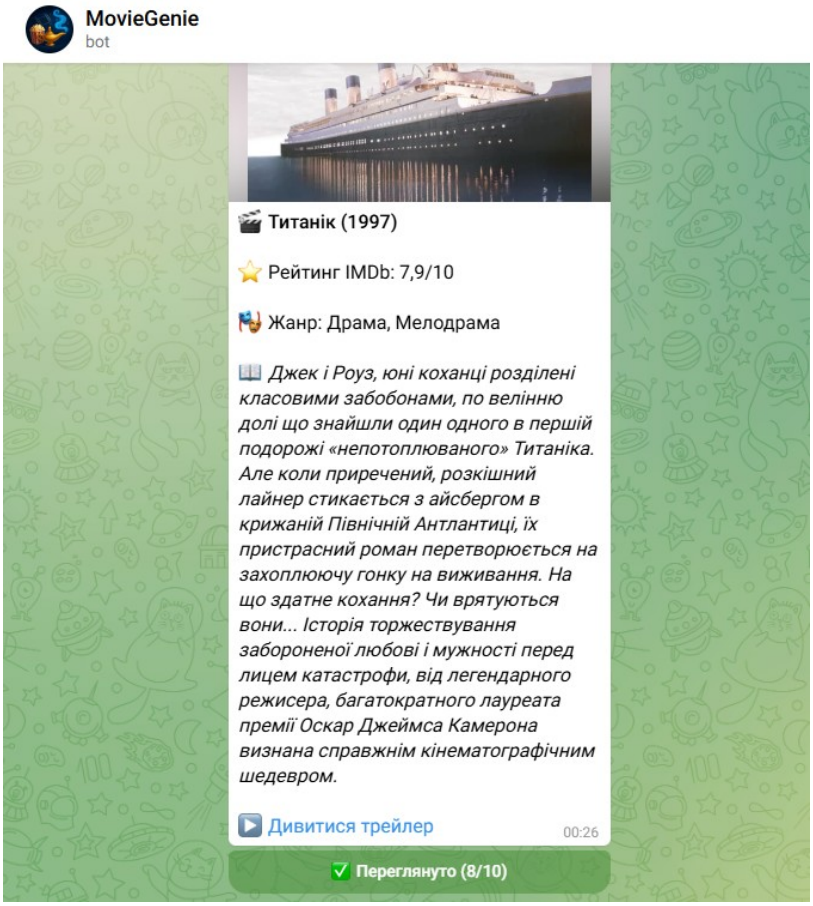

Рисунок 3.25 – Інформаційна картка фільму після додавання до «Переглянутих»

Опція «Мій список» на головному меню бота відкриває додатковий функціонал, який дозволяє користувачам переглядати й управляти своїми вподобаннями. Після натискання на цю кнопку з'являється чотири додаткові можливості (рис. 3.26).

Розділ «Зацікавлені» містить усі фільми, які користувач додав до відповідного списку. Вони відображаються у вигляді кнопок з назвою фільму та датою додавання. Натиснувши на будь-яку з них, користувач отримує розширену інформацію про фільм, аналогічну тій, що надається під час першого пошуку.

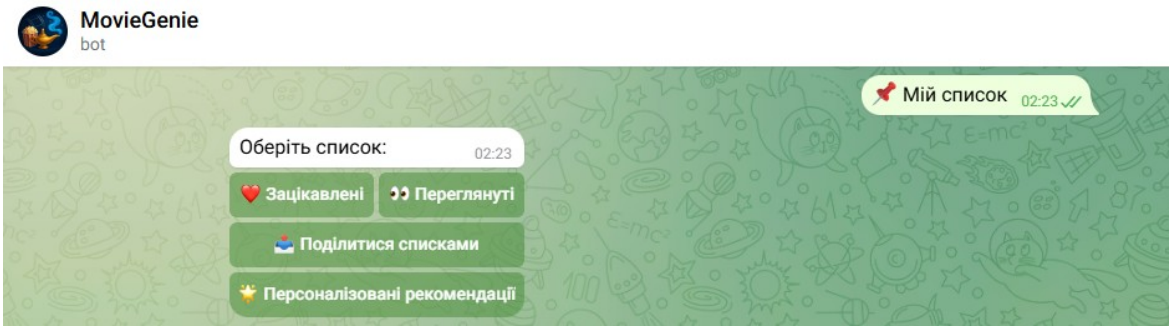

Рисунок 3.26 – Інтерфейс опції «Мій список» з додатковими можливостями

У випадку, коли фільм зі списку вже переглянуто, то його можна перемістити в інший список. Якщо користувач більше не зацікавлений у перегляді конкретного фільму, він може видалити його зі списку за допомогою відповідної кнопки (рис. 3.27).

У розділі «Переглянуті» відображаються всі фільми, які користувач уже переглянув і оцінив. Як і у випадку з «Зацікавленими», фільми представлені у вигляді інтерактивних кнопок. При натисканні відкривається картка з усією інформацією про фільм, а також зазначеною раніше оцінкою. Це дозволяє користувачеві швидко переглянути історію переглядів та, за бажанням, видалити фільм із цього списку (рис. 3.27).

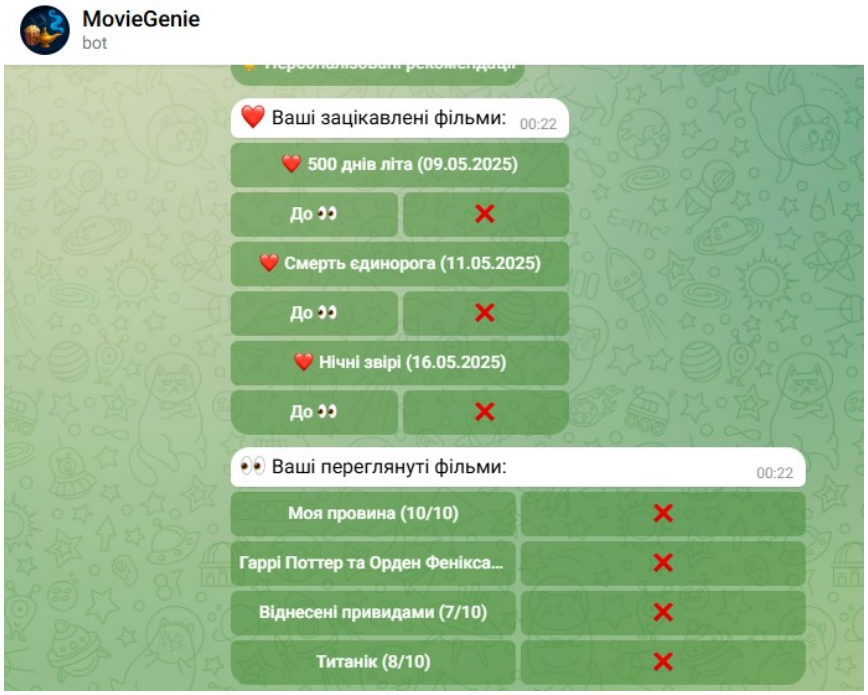

Рисунок 3.27 – Вигляд списків «Зацікавлені» та «Переглянуті»

Навпроти, функція «Поділитися списками» дозволяє надіслати свої зацікавлені та переглянуті списки іншій людині. Коли користувач натискає цю кнопку, бот створює спеціальне посилання у вигляді команди формату «/shared_list_ID», де ID – це унікальний ідентифікатор чату користувача. Цю команду можна скопіювати й надіслати іншій особі, яка в свою чергу може в будь-який час відправити цю команду у чат з ботом та отримати список фільмів власника у вигляді звичайного тексту з нумерацією (рис. 3.28). Це дуже зручна фунція, бо, коли користувач хоче порадити фільм або обговорити з кимось свої вподобання, йому не треба робити скриншоти чи вручну виписувати назви.

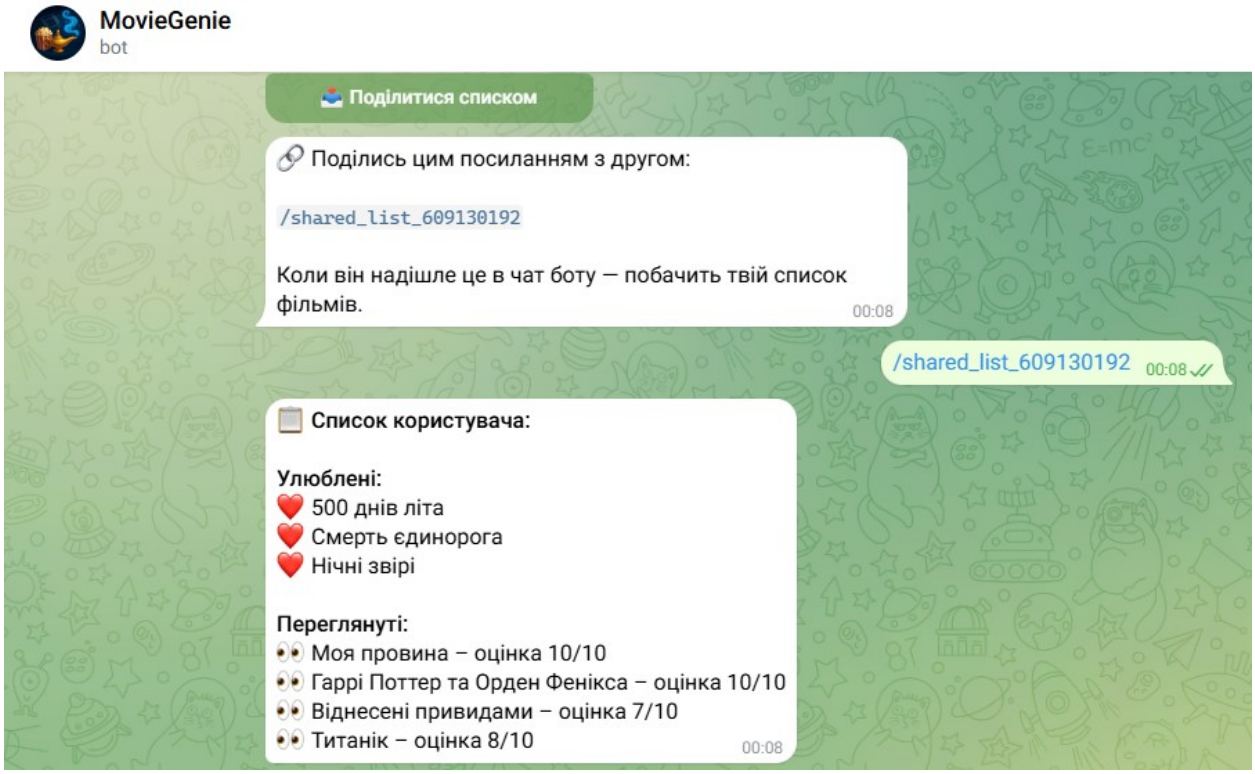

Рисунок 3.28 – Приклад згенерованого посилання для доступу до списків фільмів

Щоб отримати унікальний список фільмів, який сформований на основі індивідуальних вподобань користувача, то треба використовувати опцію «Персоналізовані рекомендації». Після того як ініціалізували запит бот аналізує списки «Переглянуті» та «Зацікавлені» через GPT-модель, яка

враховує декілька факторів: жанр, сюжетну структуру, емоційний тон, візуальну стилістику та ключові теми кожного фільму. Завдяки такому аналізу створюється психологічний профіль користувача, який бот використовує для створення списку рекомендацій.

Щоб перевірити ефективність цієї функції, було проведено тестування на основі заданого списку переглянутих та зацікавлених фільмів (рис. 3.28).

Результати були детально проаналізовані. Вони показали, що згенеровані рекомендації добре відображають смаки користувача, оскільки в них переважають емоційні драми, нестандартні романтичні історії, стилістичні фільми про кохання, втрату та спогади. У списку є різні жанри, але всі підходять під інтереси користувача (від мелодрам до психологічного фентезі). Жоден з фільмів не повторює ті, що вже були в списках (рис. 3.29).

Рисунок 3.29 – Приклад персоналізованих рекомендацій на основі вподобань користувача

У випадку, якщо користувач не має ще збережених даних про його вподобання щодо фільмів, то бот виводить відповідне повідомлення про неможливість створення рекомендацій (рис. 3.30).

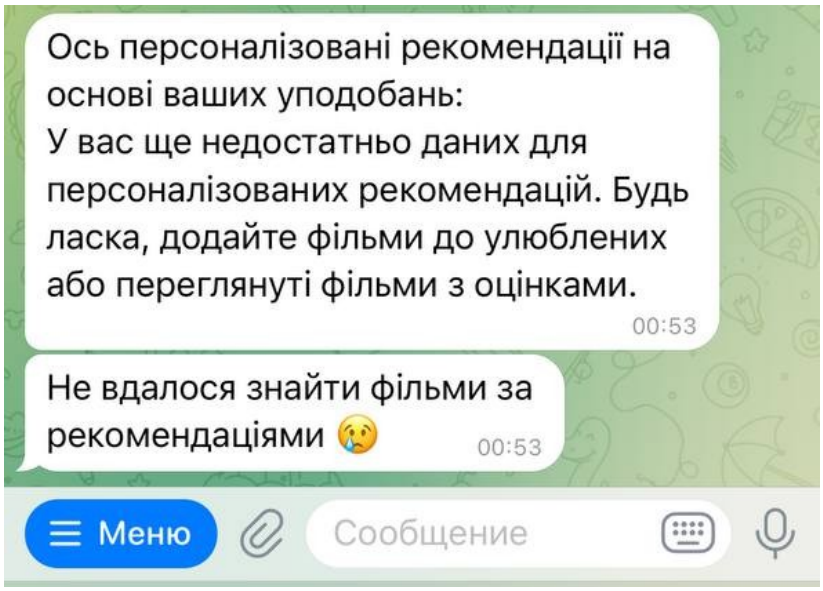

Рисунок 3.30 – Повідомлення від бота про відсутність даних для генерації рекомендацій

Щоб користувач зміг звернутися до будь-якого з фільмів зі списку рекомендацій у зручний час, не повторюючи більше запит, результати пошуку, які надсилає бот, залишаються доступними в історії чату.

3.8 Перспективи подальшої роботи

Розроблений бот у Telegram, який надає рекомендації фільмів за допомогою мовної моделі GPT, є актуальним прикладом застосування штучного інтелекту у повсякденних задачах. Він вже демонструє генерацію персоналізованих відповідей на основі текстових запитів користувача та його історії вподобання. Однак, ніколи не існує кінцевої точки досконалості при розробці застосунків, особливо на початковому етапі запуску ідей. Тому проєкт має значний потенціал для подальшого вдосконалення.

По-перше, у найближчому майбутньому планується запуск бота на хостинг. Це зробити його публічним для всіх, без потреби запуску додаткових програм. Таким чином, кожен бажаючий зможе протестувати функціонал бота у будь-який момент. І, додатково, можна впровадити функцію надання оцінки якості рекомендацій бота, наприклад, коротке опитування після кожної взаємодії або рейтингову шкалу. Такий зворотній зв'язок від користувачів стануть цінним джерелом для наступник удосконалень та змін системи.

По-друге, можливим напрямком є вдосконалення системи персоналізації. На даний момент аналіз та генерація рекомендацій відбувається за рахунок зацікавлених та переглянутих фільмів користувача. Було б також корисно зберігати інформацію щодо нецікавих фільмів користувача. Це допоможе ще точніші підібрати фільми, відсіваючи непотрібні варіанти. До цього ж, як варіант отримання більшої інформації про теперішній стан користувача, його бажань, можна запровадити опитування з декількох уточнювальних питань або повноцінний діалог з ботом. Додавання фільтрації, де можна вказувати акторів, роки випуску фільмів, країну виробництва тощо, також сприяє покращенню.

Наступний напрямок, який зможе покращити функціональність системи, це створення особистих користувацьких списків. Окрім стандартних категорій («Цікаві», «Переглянуті»), доцільно дозволити користувачам

створювати власні унікальні списки з будь-якою назвою. Наприклад, це можуть бути такі списки, як «Фільми на перегляд з друзями», «Щоб подивитися в дощовий день» або «Коли мені буде сумно». Така функція точно покращить користувацький досвід.

Четвертий напрямок - це інтеграція з потоковими сервісами. Зараз бот надає лише посилання на трейлер фільму на YouTube. Це інформативно, але не достатньо для повного користувацького досвіду, тому у майбутньому можна додати прямі посилання для перегляду фільму в таких сервісах, як Netflix, Megogo чи SWEET.TV. До того ж, така співпраця зі стрімінговими сервісами зможе принести фінансовий прибуток.

Також доцільною перспективою є розширення мовної підтримки бота, щоб він був доступним не лише для україномовних користувачів, а й для іноземної аудиторії також, то треба додати варіанти перекладу інтерфейсу на різні мови.

У майбутньому також можна розглянути використання мультимодальних моделей, які працюють не тільки з текстом, а й з зображеннями. Наприклад, аналізуючи постери фільмів або обкладинки, бот зможе краще зрозуміти візуальний стиль, який подобається користувачу. У цьому напрямі можуть стати корисними наукові роботи з обробки зображень [26-31].

Отже, система має широкий потенціал для подальшого розвитку. Реалізація запропонованих ідей покращить якість сервісу та зробить його придатним для виходу на ринок.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований бот в Telegram, який допомагає користувачам зручно підібрати фільми для перегляду на основі різних критеріїв. Ключовою особливістю проєкту стало використання мовної моделі GPT від OpenAI, яка забезпечує інтелектуальну генерацію рекомендацій до текстових запитів і вподобань користувача.

На початковому етапі було проаналізовано існуючі рішення у кіно-рекомендаційних систем, вивчено можливості LLM, досліджено особливості побудови ботів, а також обґрунтовано доцільність використання Telegram як платформи для створення інтерактивного асистента. Було сформульовано вимоги до системи, розроблено архітектуру бота, обрано відповідні інструменти розробки – мову програмування C#, бібліотеку Telegram.Bot, базу даних MongoDB, TMDB API для отримання інформації про фільми, а також інтеграція з OpenAI GPT API.

Під час виконання роботи було реалізовано:

– функціональну взаємодію з користувачем через Telegram завдяки використанню Telegram Bot API;

– інтеграцію з API OpenAI для обробки вільних запитів і генерації списку фільмів відповідно до запиту користувача;

– підключення до TMDb API для отримання детальної та актуальної інформації про фільми;

– зберігання персональних списків користувача в базі даних MongoDB для забезпечення індивідуального досвіду та зручного керування «Зацікавленими» і «Переглянутими» фільмами, а також для поширення власними добірками;

– реалізацію основного функціоналу: пошук за назвою, жанром, категоріями, випадковий вибір, рекомендації на основі штучного інтелекту.

Система була протестована, усі функції перевірено на практиці та продемонстровано у вигляді прикладів і скриншотів.

Отримані результати показують, що GPT-моделі мають великий потенціал в області рекомендацій і можуть добре розуміти запити користувачів у природній мові. Запропоноване рішення містить простоту використання, адаптивність, інтерактивность та персоналізацію.

Результати роботи апробовано у вигляді тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Berry, D. M. (2023). The Limits of Computation: Joseph Weizenbaum and the ELIZA Chatbot. Weizenbaum Journal of the Digital Society, 3(3).

2. 65 Chatbot Statistics for 2025 – New Data Released. DemandSage. URL: https://www.demandsage.com/chatbot-statistics/ (дата звернення 12.04.2025).

3. Що таке чат-боти, для чого вони потрібні, де застосовуються та які мають переваги. MC.today, Media for Creators. URL: https://mc.today/uk/shho-take-chat-boti/ (дата звернення 12.04.2025).

4. Шатаєва Т. Чат-бот для бізнесу: як створити та сценарії використання. Блог HelpCrunch. URL: https://helpcrunch.com/blog/uk/chat-bot-dla-biznesu/ (дата звернення 12.04.2025).

5. Тітов С. В., & Тітова О. В. (2015). Оцінка юзабіліті освітніх сайтів: методи і технології. Вісник Харківської державної академії культури. Серія: Соціальні комунікації, (47), 127-134.

6. Що таке велика мовна модель (Large Language Model, LLM)? | TheTransmitted. TheTransmitted.URL: https://thetransmitted.com/adlucem/shho-take-velyka-movna-model-large-language-model-llm/ (дата звернення 13.04.2025).

7. Тардіф А. Розкриття потужності великих мовних моделей (LLM). Unite.AI. URL: https://www.unite.ai/uk/великі-мовні-моделі/ (дата звернення 13.04.2025).

8. Yakovleva O., Nebeský L., Kirichenko A. (2023). Using the GPT models for responses based on custom content to develop neural consultant for university applicants. Abstracts of V International Scientific and Practical Conference «The world of modern technologies and inventions» Madrid, Spain. Pp. 172-178.

9. Brandl R., Ellis C. ChatGPT Statistics and User Numbers 2025 - OpenAI Chatbot. Tooltester. URL: https://www.tooltester.com/en/blog/chatgpt-statistics/ (дата звернення 13.04.2025).

10. МакФарланд А. 5 найкращих великих мовних моделей (LLM) у квітні 2025 року. Unite.AI. URL: https://www.unite.ai/uk/найкращі-великі-мовні-моделі-llms/ (дата звернення 13.04.2025).

11. Jain S. History of Netflix- Founding, Model, Timeline, Milestones (2025). *VdoCipher Blog*. URL: https://www.vdocipher.com/blog/2017/06/netflix-revolution-part-1-history/ (дата звернення 14.04.2025).

12. PhD S. L. Netflix vs. decision fatigue: How to solve the paradox of choice. Medium. URL: https://uxdesign.cc/netflix-vs-decision-fatigue-how-to-solve-the-paradox-of-choice-888ca56db4b (дата звернення 14.04.2025).

13. Джарвіс - ваш асистент з вибору фільмів. URL: https://jarvis.net.ua/ (дата звернення 17.04.2025).

14. Bielozorov A., Bezbradica M., Helfert M. The Role of User Emotions for Content Personalization in e-Commerce: Literature Review. HCI in Business, Government and Organizations. eCommerce and Consumer Behavior. Cham, 2019. P. 177–193.

15. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks, Int. J. Acad. Inf. Syst. Res., vol. 7, no. 7, pp. 25-36.

16. Rupani B. Solving the Cold Start Problem: Leveraging Large Language Models for Effective Recommendations. Medium. URL: https://medium.com/@Bhawna_Rupani/solving-the-cold-start-problem-leveraging-large-language-models-for-effective-recommendations-a170991189b3 (дата звернення 18.04.2025).

17. Sanner, S., Balog, K., Radlinski, F., Wedin, B., & Dixon, L. (2023, September). Large language models are competitive near cold-start recommenders for language-and item-based preferences. In Proceedings of the 17th ACM conference on recommender systems (pp. 890-896).

18. Eberhard, L., Ruprechter, T., & Helic, D. (2024). Large Language Models as Narrative-Driven Recommenders. arXiv preprint arXiv:2410.13604.

19. Sun, R., Li, X., Akella, A., & Konstan, J. A. (2024). Large language models as conversational movie recommenders: A user study. arXiv preprint arXiv:2404.19093.

20. Ukrinform. Telegram, YouTube чи TikTok: звідки українці дізнаються новини. Укрінформ - актуальні новини України та світу. URL: https://www.ukrinform.ua/rubric-society/3890827-telegram-youtube-ci-tiktok-zvidki-ukrainci-diznautsa-novini.html#google_vignette (дата звернення 28.04.2025).

21. Zhao G. (. How ChatGPT really works, explained for non-technical people. Medium. URL: https://medium.com/design-bootcamp/how-chatgpt-really-works-explained-for-non-technical-people-71efb078a5c9 (дата звернення 01.05.2025).

22. Тітова, О. В. (2018). Основи інформаційного забезпечення управління

23. Chorna, O., Didyk, P., Titov, S., & Titova, O. (2024). Usage of Clustering Algorithms for Automating Route Planning in Transportation Routing Tasks // *Системи обробки інформації*. № 1 (176) 2024, p. 115-123.

24. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, International Journal of Engineering and Information Systems, 6(2), pp. 40–48.

25. Chauhan, A. (2019). A review on various aspects of MongoDB databases. *International Journal of Engineering Research & Technology (IJERT)*, *8*(05), 90-92.

26. Кобилін, О., Вечірська, І., & Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, pp. 63-76.

27. Gorokhovatskyi V., and Tvoroshenko I. (2024) An effective method for transforming an image description into a compact vector for classification. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / Ministry of Education and Science of Ukraine, Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor). – Kyiv: Publishing House «Caravela», pp. 25-28.

28. Кобилін, О.А., & Творошенко, І.С. (2021). Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ, 124 с.

29. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, vol. 33, no. 1, pp. 113-125.

30. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, IEEE Access, 10, pp. 124738-124746.

31. Бібігул, Ш., & Вечірська, І. (2023). Використання методу знаходження n-ого лінійного логічного перетворення для розв’язання задачі знаходження гіпотетично зв’язаних об’єктів. Матеріали конференцій МНЛ, (22 грудня 2023 р., м. Чернівці), 267-269.

32. Шералієва Б.Д. (2025) Використання GPT-моделей для персоналізованого вибору фільмів у Telegram-боті. *Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.)*. Харків: ХНУРЕ, 2025. Т. 7. С. 180-181.