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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Чат-бот Telegram на мові програмування Python для підбору книг»

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

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

_____ Віталій КОЛЯДЮК
(підпис) (Ім'я, ПРІЗВИЩЕ здобувача)

Виконав: здобувач вищої освіти гр. КНД-43
Віталій КОЛЯДЮК

Керівник: Олена ШИКУЛА
науковий ступінь, вчене звання: д.ф.-м.н., професор

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

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Колядюку Віталію Вікторовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Чат-бот Telegram на мові програмування Python для підбору книг

керівник кваліфікаційної роботи Олена ШИКУЛА д.ф.-м.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. №56

2. Строк подання кваліфікаційної роботи «27» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи:
текстові запити користувача (жанр, автор, настрій, мова);
перелік жанрів та ключових слів для NLP;
структура відповіді з Google Books API (JSON-формат).

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження сучасних підходів до реалізації чат-ботів.
Розробка модульної архітектури бота для пошуку книжкової інформації.
Реалізація інтеграції з Google Books API для отримання метаданих про книги.

Створення локального кешу на базі SQLite для збереження результатів і зменшення навантаження на API.

5. Перелік графічного матеріалу: презентація

1) Тема дипломної роботи
2) Мета роботи. Предмет дослідження.
3) Актуальність роботи.
4) Аналіз існуючих рішень
5) Архітектура Telegram-бота

6. Дата видачі завдання 25 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та аналогів, підбір джерел	23.02-09.03.25	виконане
2	Поставновка задач, вибір інструментів, технічне проєктування	09.03-16.03.25	виконане
3	Розробка модулів: NLP, API, кеш, форматування	17.03- 23.03.25	виконане
4	Побудова Telegram-інтерфейсу, інтеграція компонентів	24.03- 29.03.25	виконане
5	Тестування роботи бота, демонстрація функціоналу	29.03-10.04.25	виконане
6	Оформлення пояснювальної записки та додатків	10.04-20.04.25	виконане
7	Написання вступу, висновків, узагальнення результатів	20.04-05.05.25	виконане
8	Підготовка презентації, перевірка, здача роботи	05.05-15.05.25	виконане

Здобувач вищої освіти _____ Віталій КОЛЯДЮК
(підпис) (Ім'я, ПРІЗВИЩЕ)

Керівник
кваліфікаційної роботи _____ Олена ШИКУЛА
(підпис) (Ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 55 стор., 22 рис., 6 табл., 20 джерел.

Наукове завдання – Розробка програмного засобу для автоматизованого підбору книжкової інформації на основі текстових запитів у Telegram з використанням зовнішніх джерел та інструментів обробки природньої мови.

Мета роботи – Створити Telegram-бота, що дозволяє здійснювати швидкий та персоналізований пошук книжкової літератури шляхом інтеграції Google Books API та реалізації базового модуля NLP для інтерпретації запитів.

Об’єкт дослідження – Чат-боти в месенджерах як інструменти автоматизації інформаційного пошуку.

Предмет дослідження – Процес проєктування, розробки та реалізації Telegram-бота для підбору книжкової інформації на основі запитів користувача.

Короткий зміст роботи:

У роботі досліджено сучасні рішення в галузі створення чат-ботів, проаналізовано можливості мови програмування Python та бібліотек aiogram, requests, nlkt, python dotenv, pytest для роботи з Telegram API і Google Books API. Розроблено архітектуру Telegram-бота, реалізовано обробку запитів користувача з використанням засобів попередньої обробки природньої мови. Інтегровано запити до Google Books API та реалізовано кешування відповідей у базі даних SQLite для підвищення швидкодії. Надано приклади роботи бота, структуру відповідей, а також запропоновано напрямки для подальшого розвитку проєкту.

Ключові слова: Telegram-бот, Python, Google Books API, NLP, SQLite, чат-бот, Інформаційний пошук, Aiogram.

ЗМІСТ

ВСТУП

В сучасному світі стрімко розвиваються цифрові технології та інформаційні сервіси. Одним із важливих напрямів є автоматизація процесів пошуку та надання інформації користувачам через месенджери. Найбільшу популярність серед таких платформ здобув Telegram завдяки своїй відкритій архітектурі, гнучкому API та високій швидкості роботи.

В наш час спостерігається зростання попиту на ефективні інструменти для пошуку книг серед широкого кола користувачів: студентів, книголюбів, науковців. Традиційні способи пошуку книжкової інформації часто є незручними або вимагають значних витрат часу через інформаційне перевантаження. У цьому контексті інтеграція Telegram-бота з Google Books API відкриває нові можливості для швидкого та персоналізованого пошуку книг за допомогою простого та інтуїтивного інтерфейсу.

Таким чином, створення Telegram-бота для підбору книг із використанням Python та зовнішніх API є актуальним завданням, яке відповідає сучасним потребам інформаційного суспільства.

Метою даної роботи є розробка Telegram-бота нам мові програмування Python для ефективного підбору книг, із залученням можливостей Google Books API. Для досягнення поставленої мети необхідно вирішити наступні завдання:

- Провести аналіз предметної області та огляд існуючих рішень у сфері чат-ботів для підбору книг;
- Обґрунтувати вибір інструментів для розробки чат-бота;
- Спроєктувати архітектуру чат-бота для підбору книжок;
- Реалізувати інтеграцію Google Books API та організувати обробку запитів користувачів;
- Провести тестування роботи чат-бота і оцінити його ефективність;
- Визначити напрями подальшого розвитку розробленої системи.

Об'єктом дослідження є чат-боти в месенджерах як засіб автоматизації інформаційного пошуку.

Предметом дослідження є процес створення чат-бота в Telegram, який забезпечує підбір книжкової продукції на основі інтеграції з Google Books API.

У процесі виконання роботи використовувалися методи аналізу літературних джерел та існуючих рішень, методи програмування, тестування програмного забезпечення, а також методи аналізу ефективності інформаційних систем.

Наукова новизна полягає у поєднанні можливостей Natural Language Processing(NLP) із функціоналом Google Books API для побудови персоналізованих рекомендацій у Telegram-боті. Такий підхід дозволяє покращити якість і точність підбору книжок відповідно до запитів користувачів.

Практична цінність роботи. Розроблений Telegram-бот може бути використаний у бібліотеках, книжкових спільнотах, освітніх установах та електронних книжкових магазинах для автоматизації процесу підбору книг. Система може слугувати базою для подальшого розвитку інформаційних сервісів.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ TELEGRAM-БОТА ДЛЯ ПІДБОРУ КНИГ

1.1 Чат-боти в месенджерах: сучасний стан і перспективи

У сучасних умовах розвитку інформаційних технологій чат-боти стали важливим елементом автоматизації комунікацій у різноманітних сферах діяльності: бізнесі, освіті, медицині, обслуговуванні клієнтів тощо.

Чат-боти – це програми, що імітують спілкування з людиною в тестовому або голосовому режимі за допомогою певних алгоритмів та інструментів штучного інтелекту.

Telegram є однією з найбільших популярних платформ для створення чат-ботів завдяки наявності спеціального Telegram Bot API, який забезпечує розробникам можливості швидкої інтеграції ботів у месенджери без необхідності створення окремого клієнта. Ключовими перевагами Telegram як платформи належать відкритий API та проста авторизація, широкий набір інтеграційних можливостей, включаючи обробку повідомлень, інтерактивних кнопок, inline-режиму та платежів. Також важливою характеристикою є масштабованість і висока продуктивність, що дозволяє обслуговувати тисячі користувачів одночасно, а безпека даних користувачів забезпечується завдяки наскрізному шифруванню.

За даними досліджень, чат-боти в месенджерах активно використовуються для автоматизації довідкових служб, підтримки користувачів, онлайн-консультування, бронювання послуг і товарів, а також у сфері електронної комерції.

Перспективи розвитку чат-ботів у найближчі роки пов’язані з такими тенденціями, як використання елементів штучного інтелекту (AI) та обробки природної мови (NLP) для підвищення якості спілкування, інтеграція з великими базами знань та сторонніми сервісами через API, наприклад, бібліотечні каталоги, книжкові сервіси. Також набирає популярності використання гібридних моделей,

що поєднують автоматичну обробку запитів із можливістю залучення людини на складних етапах обробки інформації. Окрему увагу привертає поширення персоналізованих ботів, що аналізують дані користувача для підбору індивідуальних рекомендацій.

Таким чином, розвиток чат-ботів в екосистемах месенджерів, зокрема в Telegram, є надзвичайно перспективним напрямком, що відповідає сучасним запитам користувачів на оперативність, доступність та індивідуалізацію послуг.

1.2 Особливості пошуку книг: потреби користувачів Telegram

У контексті розвитку цифрових технологій пошук книжкової інформації став суттєвою складовою задоволення інтелектуальних та освітніх потреб користувачів. Telegram, як популярний месенджер із підтримкою бот-платформи, пропонує широкий спектр можливостей для створення автоматизованих систем рекомендацій книг, орієнтованих на потреби цільової аудиторії.

1.2.1 Основні категорії користувачів

Аналіз цільової аудиторії Telegram свідчить про наявність кількох основних груп користувачів, зацікавлених у книжкових сервісах. До них належать студенти та школярі які шукають навчальну літературу, підручники, довідники, а також науковці та дослідники, що мають в доступі до наукових монографій, академічних праць. Крім того, вагому частку аудиторії становлять книголюби, котрі прагнуть знайти новинки художньої літератури, бестселери або рекомендації відповідно до жанру або настрою. Не менш активними користувачами є батьки й освітяни, зацікавлені в дитячій літературі для різних вікових груп.

Таким чином, функціонал чат-бота має бути гнучким і враховувати різноманітність запитів користувачів.

1.2.2 Типові запити користувачів

Аналіз реальних сценаріїв використання книжкових ботів у Telegram показує, що користувачі найчастіше формулюють свої запити, спираючись на чотири основні критерії. Поширеними є звернення за жанром, наприклад: “Підкажи фантастику”, “Хочу детектив”. Не менш часто зустрічаються запити, орієнтовані на конкретного автора, як-от: “Що написав Стівен Кінг?”. Частина користувачів цікавиться новинками книжкового ринку, ставлячи питання на кшталт: “Що популярного вийшло у 2025 році?”. Окрему категорію становлять запити, сформульовані відповідно до емоційного стану або настрою читача, як, наприклад: “Хочу щось легке і надихаюче” або “Книга для мотивації”.

Оскільки такі запити часто є нечіткими або емоційно забарвленими, ефективна робота чат-бота вимагає впровадження механізмів попередньої обробки природньої мови. Саме за допомогою технологій NLP система здатна точніше розпізнати намір користувача і сформувати релевантну відповідь.

1.2.3 Проблеми пошуку книг у Telegram

Попри численні переваги, пошук книг у Telegram за допомогою існуючих ботів супроводжується низкою характерних проблем. Однією з основних є інформаційне перевантаження, коли користувач отримує велику кількість нерелевантних результатів через недостатньо точні або погано налаштовані пошукові запити. Ще одним суттєвим недоліком є відсутність персоналізації: багато ботів не враховують уподобання користувача при наступних пошуках, що знижує ефективність взаємодії. Крім того, деякі системи працюють із вузькими та технічно обмеженими каталогами, не охоплюючи новинку чи рідкісну літературу, що зменшує цінність таких сервісів для вимогливих користувачів.

Для усунення зазначених недоліків доцільним є використання потужних відкритих джерел даних, зокрема Google Books API, який забезпечує доступ до мільйонів книжкових позицій. Така інтеграція дозволяє отримувати метадані про

книги, включаючи авторів, жанрову класифікацію, рейтинги та короткі анотації, що значно покращує якість рекомендацій.

1.2.4 Вимоги до функціоналу бота

На основі аналізу потреб користувачів можна сформувати базові вимоги до функціоналу чат-боту для підбору книг у Telegram. Передусім, система повинна підтримувати запити, сформульовані за жанром, автором, рівнем популярності або настроєм користувача. Результати пошуку мають бути релевантними, а також сортуватись відповідно до рейтингу. Важливо щоб бот зберігав історію запитів і рекомендацій, що дозволить забезпечити елементи персоналізації під час наступних звернень. Не менш істотною є мінімізація часу відповіді, оскільки оперативність є однією з ключових переваг месенджерів як платформи. Інтерфейс взаємодії з ботом повинне залишатись інтуїтивно зрозумілим, з короткими текстовими відповідями та можливістю уточнення запиту за допомогою інтерактивних кнопок.

Таким чином, побудова ефективного чат-бота для підбору книг у Telegram має бути орієнтована на задоволення реальних потреб користувачів шляхом забезпечення швидкого, точного та персоналізованого пошуку книжкової інформації.

1.3 Огляд аналогів і обґрунтування необхідності розробки нового рішення

Аналіз існуючих рішень у сфері автоматизованого підбору книжкової літератури через меседжери показує, що сьогодні Telegram пропонує в основному ботів з обмеженим функціоналом. Серед таких рішень переважають боти, що дозволяють здійснювати пошук книг за назвою або автором, іноді з можливістю завантаження електронної копії книги. Однак повноцінних систем, орієнтованих на

персоналізований підбір книги за настроєм, жанром, рекомендаційною системою на основі аналізу запитів користувача, у Telegram на даний момент немає.

1.3.1 Поточний стан ринку Telegram-ботів для пошуку книжної літератури

Під час проведення власного аналізу екосистеми Telegram-ботів було виявлено, що більшість наявних сервісів мають характерні обмеження, які значно знижують якість пошуку книжкової інформації. Насамперед, пошук часто реалізовано лише за прямими запитами, наприклад, за назвою книги або іменем автора, без можливості фільтрації результатів за жанром, настроєм або іншими характеристиками. Крім того, такі сервіси не передбачають персоналізованого підходу, оскільки історія запитів користувача не аналізується, а рекомендації не формуються відповідно до його вподобань. Ще однією проблемою є відсутність інтеграції з великими офіційними базами книжкових даних, зокрема з Google Books API або іншими бібліотечними каталогами. Варто також зазначити, що більшість ботів мають інтерфейс, орієнтований переважно на англомовну аудиторію, що ускладнює використання такими сервісами україномовними користувачами. Майже повністю відсутні функції розширеного пошуку, які б дозволили б враховувати емоційних стан, стиль або тематику літератури, як це властиво сучасним рекомендаційним системам.

Серед небагатьох рішень, що частково реалізують функціонал пошуку книжкової інформації в Telegram, можна згадати бота **@GoogleBooksSearchBot**. Цей інструмент надає користувачу змогу здійснювати пошук книг за ключовими словами, іменем автора або назвою твору безпосередньо в межах месенджера Telegram. Отримані результати містять повідомлення, у яких вказується назва книги та її автор. Важливо, що бот забезпечує швидке реагування на текстовий запит і не вимагає переходу на зовнішні веб-сайти чи виконання додаткових дій. Інтеграція з Telegram дозволяє користувачеві взаємодіяти із сервісом у звичному середовищі мобільного або десктопного застосунку. Інтерфейс бота є простим і не

потребує авторизації чи реєстрації, а швидкість відповіді та стабільність роботи справляють позитивне враження.

Разом із тим, попри зручність використання, бот має істотні обмеження та ризики. Зокрема, у відповідь на запити користувач отримує посилання, структура яких не дозволяє однозначно визначити джерело файлу, що створює проблему прозорості та довіри до ресурсу (див. приклад на рис. 1.1). Ще серйознішою є проблема порушення авторських прав: деякі результати містять прямі посилання на PDF-файли книг, які можна безкоштовно завантажити. Такий спосіб розповсюдження суперечить принципам захисту інтелектуальної власності та позбавляє авторів можливості отримувати винагороду за свою працю. Крім того, функціонал бота не враховує історії запитів, не підтримує фільтрацію за жанром або настроєм і не надає персоналізованих рекомендацій. Варто також зазначити, що, попри назву, бот не інтегрований з офіційною базою Google Books API, що ставить під сумнів коректність джерел інформації.

Таким чином, хоча GoogleBooksSearchBot формально виконує базову функцію пошуку книжок у Telegram, його реалізація не відповідає вимогам до якісного, етичного та персоналізованого сервісу. Відсутність офіційної інтеграції з Google Books API, недотримання авторських прав та обмежений функціонал підкреслюють актуальність розробки нового Telegram-бота. Такий агент має бути побудований на основі відкритих і легальних джерел, підтримувати сучасні рекомендаційні механізми та забезпечувати високий рівень зручності й етичності доступу до літературного контенту.

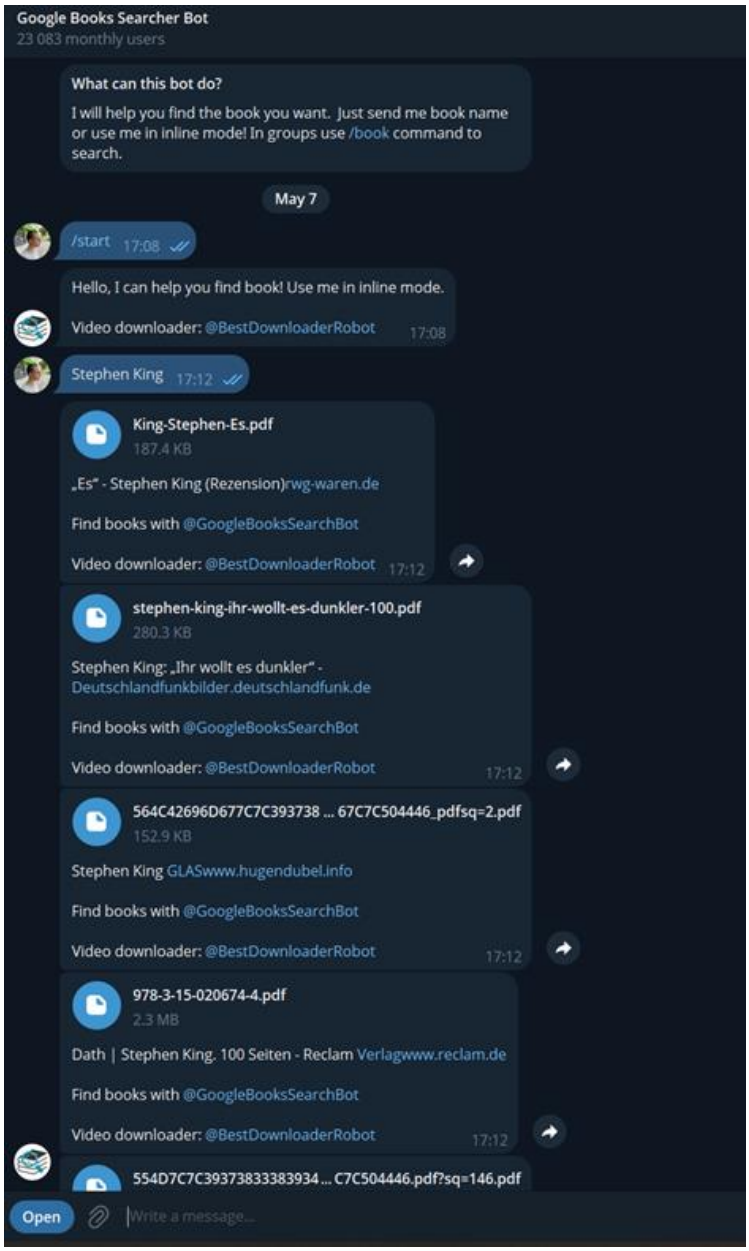


Рисунок 1.1 – Приклад відповіді бота @GoogleBooksSearchBot

1.3.2 Обґрунтування необхідності створення нового рішення

У зв’язку з відсутністю в Telegram повноцінних рішень для інтелектуального та персоналізованого підбору книжкової літератури, постає об’єктивна необхідність створення нового Telegram-бота, який здатен враховувати ключові потреби сучасного користувача. Однією з основних вимог є можливість надання персоналізованих рекомендацій, що базуватимуться на введених запитах, жанрових уподобаннях та історії попередніх пошуків користувача. Важливою складовою такого рішення має стати інтеграція з Google Books API, що забезпечить доступ до актуальної інформації про книги, авторів, жанрову класифікацію, рейтинги та рецензії.

Особливу увагу слід приділити створенню інтерфейсу, який буде інтуїтивно зрозумілим, україномовним та не вимагатиме від користувача виконання складних дій або команд. Бот повинен також підтримувати пошукові запити, сформульовані не лише за змістовими характеристиками, але й за емоційним критерієм, наприклад: «щось надихаюче», «новинки фантастики» чи «мотиваційна література». Для забезпечення швидкодії система має реалізовувати механізм локального кешування популярних запитів, що дозволить зменшити затримки у формуванні відповіді.

Запропоноване рішення істотно відрізняється від наявних аналогів своєю багатофункціональністю, гнучкістю та орієнтацією на покращення користувацького досвіду. Такий бот не лише автоматизуватиме процес пошуку книжної інформації, а й сприятиме популяризації читання серед різних категорій користувачів за допомогою доступного і зручного інструменту.

1.3.3 Порівняння можливостей існуючих рішень і пропонованого проекту

Для кращого розуміння переваг пропонованого Telegram-бота доцільно подати порівняння характеристик (див. табл. 1.1):

Таблиця 1.1 – Порівняння характеристик бота.

Критерії	**Існуючі боти**	**Пропоноване рішення**
Пошук за назвою книги	Так	Так
Пошук за автором	Так	Так
Пошук за жанром	Ні	Так
Пошук за настроєм	Ні	Так
Пошук за настроєм	Ні	Так
Інтеграція з базами даних	Ні	Так
Історія запитів і персоналізація	Ні	Так
Україномовний інтерфейс	Ні	Так
Підтримка рекомендаційних алгоритмів	Ні	Так

Згідно з порівняльною таблицею, розробка нового Telegram-бота дозволить суттєво розширити функціональність пошуку книжкової літератури у месенджері, забезпечивши користувачів більш релевантними і персоналізованими рекомендаціями.

1.4 Формування мети та завдань розробки чат-бота

Аналіз існуючих рішень показав, що на даний момент у Telegram відсутні ефективні інструменти для персоналізованого підбору книжкової літератури за жанром, настроєм чи іншими індивідуальними характеристиками користувачів. Виявлений дефіцит рішень такого типу обґрунтовує необхідність розробки нового

програмного продукту, що буде відповідати сучасним вимогам до автоматизації пошуку книжкової інформації.

1.4.1 Мета розробки

Метою даної роботи є створення програмного агента (Telegram-бота) на мові програмування Python, який забезпечуватиме ефективний персоналізований підбір книжкової літератури з використанням можливостей Google Books API та технологій обробки природної мови (NLP).

Бот має забезпечити користувачам такі можливості:

- Швидкий доступ до релевантної книжкової інформації.
- Пошук книг за жанрами, настроєм, автором або ключовими словами.
- Персоналізовані рекомендації на основі історії запитів.
- Інтуїтивно зрозумілий і зручний україномовний інтерфейс.

1.4.2 Завдання розробки

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз предметної області та визначити особливості пошуку книг через месенджери.
2. Вибрати найбільш відповідний технологічний стек для розробки чат-бота, включаючи вибір бібліотек з Telegram API та інструментів обробки природної мови.
3. Спроєктувати архітектуру чат-бота з урахуванням вимог до масштабованості, надійності та зручності використання.
4. Розробити алгоритми обробки запитів користувачів, фільтрації та ранжування результатів відповідно до заданих параметрів.
5. Реалізувати механізми інтеграції з Google Books API для отримання інформації про книжкові ресурси.

6. Побудувати систему кешування відповідей для оптимізації швидкості обробки запитів та зменшення навантаження на зовнішній API.

7. Реалізувати персоналізацію рекомендацій на основі збереженої історії взаємодій користувача з ботом.

8. Провести тестування чат-бота для виявлення та усунення можливих помилок у роботі, оцінити ефективність системи рекомендацій.

9. Розробити пропозиції щодо подальшого вдосконалення системи, у тому числі шляхом розширення функціоналу за рахунок додаткових джерел даних або застосування методів машинного навчання.

1.4.3 Очікувані результати

У результаті даної роботи буде створено програмний агент для месенджера Telegram (Telegram-бот), який надаватиме користувачам зручний сервіс для індивідуального пошуку та підбору книжкової літератури, що базуватиметься на інтеграції з авторитетною базою Google Books, використані NLP та принципах персоналізації інформаційних запитів.

2 ТЕХНОЛОГІЧНІ ЗАСОБИ РОЗРОБКИ ЧАТ-БОТА

2.1 Мова програмування Python та її переваги

Python – це високорівнева мова програмування загального призначення, яка завдяки своїй простоті, читабельності та великому екосистемному оточенню стала одним із найпопулярніших інструментів для створення програмного забезпечення, зокрема веб-сервісів, ботів і систем штучного інтелекту (ШІ).

У межах даної роботи мову Python обрано як основний інструмент розробки Telegram-агента. Такий вибір зумовлений низкою факторів. По-перше, мова має простий та лаконічний синтаксис, що пришвидшує розробку та спрощує підтримку коду. По-друге, екосистема Python охоплює велику кількість бібліотек і модулів, які дають змогу швидко інтегрувати Telegram API, працювати з HTTP-запитами, обробляти JSON, реалізовувати NLP(Обробку природної мови) та взаємодіяти з базами даних. Важливим чинником є й активна спільнота, що забезпечує наявність великої кількості документації, навчальних матеріалів та готових рішень які знижують поріг входу для нових розробників. Також слід зазначити гнучкість і кросплатформеність, що дозволяє запускати бот як локально, так і в хмарному середовищі.

У рамках реалізації Telegram-агента Python буде використано для взаємодії з Telegram API через бібліотеки на зразок python-telegram-bot або aiogram, для надсилання HTTP-запитів до Google Books API з використанням бібліотек requests або google-api-python-client, а також для обробки текстових запитів користувача та реалізації логіки пошуку книг. Крім цього, Python застосовується для збереження даних у базі SQLite за допомогою модуля sqlite3, а також для реалізації NLP(Natural Language Processing) де можуть бути використані бібліотеки spaCy, nltk або transformers.

Для розробки бота використовується Python версії 3.11 або новішої, оскільки ця версія має низку покращень. Зокрема, вона забезпечує вищу продуктивність,

розширену підтримку сучасних структур даних та вдосконалену реалізацію асинхронного програмування, що є критично важливим у випадках, коли бот має обробляти запити від великої кількості користувачів одночасно.

Хоча Python є оптимальним вибором для реалізації даного проєкту, варто коротко розглянути і можливі альтернативи. Однією з таких є JavaScript у середовищі Node.js. Ця мова має розвинуту екосистему пакетів npm і підтримку бібліотек для Telegram, зокрема node-telegram-bot-api. Асинхронна модель виконання також є корисною у випадку паралельної обробки запитів. Проте ця технологія має низку обмежень, серед яких менш зручний синтаксис для швидкого прототипування, обмежена підтримка бібліотек для NLP, а також необхідність у детальнішому ручному налаштуванні роботи з HTTP-запитами та форматами даних.

Ще однією альтернативою є Java. Вона вирізняється високою продуктивністю та стабільністю, а також добре підходить для реалізації масштабних корпоративних систем. Однак її застосування у проєктах типу Telegram-бота вимагає більших витрат ресурсів на налаштування середовища та підтримку коду, що робить Python кращим вибором з погляду гнучкості й швидкості реалізації. Крім того, реалізація простої логіки в Java зазвичай супроводжується більшим обсягом коду, що уповільнює цикл розробки. Екосистема бібліотек для створення ботів у цій мові є менш динамічною, а можливості інтеграції з модулями NLP обмежені.

Інша потенційна альтернатива — мова Go (або Golang), яка демонструє високу швидкість виконання та ефективну підтримку паралелізму. Ці характеристики роблять її привабливою для сервісів, що працюють у режимі реального часу. Проте, з огляду на обмежену кількість бібліотек для Telegram та майже повну відсутність засобів для обробки природної мови, її використання для створення функціонального бот-рішення видається недоцільним. Крім того, статична типізація Go обмежує гнучкість при реалізації логіки обробки користувацьких запитів.

PHP також розглядається як популярний інструмент веб-розробки, однак його застосування у сфері розробки Telegram-ботів є обмеженим. Попри широку підтримку на більшості хостингів, мова не має належної інтеграції з Telegram API, а також сучасних бібліотек для роботи з текстом і реалізації NLP-модулів. Додаткові труднощі виникають під час організації асинхронної обробки запитів, що є критично важливим для забезпечення швидкодії системи. Через зазначені фактори PHP не можна вважати ефективною альтернативою Python у рамках реалізації даного проєкту.

В таблиці 2.1 можна спостерігати коротку порівняльну характеристику всіх описаних мов програмування.

Таблиця 2.1 – Порівняльна характеристика мов програмування для створення Telegram-ботів

Критерій	**Python**	**JavaScript**	**Java**	**Go**	**PHP**
Простота синтаксису	Висока	Середня	Низька	Середня	Низька
Швидкість розробки	Висока	Висока	Низька	Середня	Низька
Підтримка Telegram API	Повна	Повна	Часткова	Часткова	Обмежена
Наявність бібліотек для NLP	Розвинена	Обмежена	Обмежена	Практично відсутня	Відсутня
Інтеграція з API	Зручна	Задовільна	Складна	Обмежена	Складна
Паралельна обробка запитів	Добра	Добра	Складна	Вбудована	Обмежена
Документація та навчальні ресурси	Дуже багато	Дуже багато	Багато	Обмежено	Обмежено
Придатність до створення чат-ботів	Висока	Висока	Низька	Середня	Низька

Наведене порівняння свідчить, що жодна з альтернатив не забезпечує одночасно швидкість розробки продукту, не має такого простого синтаксису як Python та немає такої кількості якісних бібліотек для роботи з Telegram API, HTTP,

NLP. Python, у свою чергу, поєднує всі ці переваги, що робить його найкращим вибором для реалізації Telegram-бота в рамках цієї роботи.

2.2 Бібліотеки для роботи з Telegram і Google Books API

Ефективна реалізація Telegram-агента вимагає використання перевірених та зручних у застосуванні бібліотек. У межах даної роботи обрано низку Python-бібліотек, які забезпечують стабільну взаємодію з Telegram Bot API, Google Books API, а також реалізують обробку запитів, форматування відповідей та зберігання інформації.

2.2.1 Бібліотеки для взаємодії з Telegram

Aiogram – це сучасна асинхронна бібліотека для роботи з Telegram Bot API, яка реалізована на основі модуля asyncio. ЇЇ основною перевагою є підтримка асинхронного виконання, що дозволяє ефективно обробляти одночасно велику кількість запитів користувачів без втрати продуктивності. Завдяки чіткому розділенню логіки обробки повідомлень за допомогою хендлерів забезпечується зручність сортування коду, а вбудована підтримка FSM (Finite State Machine) дозволяє реалізовувати повноцінну діалогову взаємодію з користувачем. Додатковою перевагою є активна спільнота розробників, а також наявність актуальної документації, що робить бібліотеку зручною як для початківців, так і для досвідчених фахівців.

Альтернативним рішенням є бібліотека python-telegram-bot, яка базується на синхронній моделі виконання. Вона вирізняється простотою у використанні та досить швидким стартом, що робить її зручною для невеликих проєктів або навчальних цілей. Проте через відсутність вбудованої асинхронності така бібліотека має обмеження щодо масштабованості, особливо у випадках, коли бот має взаємодіяти з великою кількістю користувачів одночасно. Зважаючи на це, для

реалізації системи, яка має обслуговувати багато паралельних запитів із високою швидкістю, перевагу доцільно надати саме асинхронній бібліотеці aiogram.

2.2.2 Бібліотеки для доступу до Google Books API

Для взаємодій з сервісом Google Books API у Python існує декілька зручних інструментів. У межах цього проєкту було розглянуто два основні підходи: використання бібліотеки requests та офіційного клієнта Google – google-api-python-client.

Бібліотека requests є базовим інструментом для здійснення HTTP-запитів у Python. Вона дозволяє надіслати запити типу GET або POST до зовнішніх API, працювати з параметрами запуску, заголовками та кодами відповіді, а також легко обробляти відповіді у форматі JSON. Її головною перевагою є простота використання, читабельність коду та мінімальні вимоги до конфігурації. У межах цього проєкту request використовується як основний засіб безпосередньої взаємодії з API Google Books, що є доцільним у контексті роботи з відкритим інтерфейсом, який не потребує авторизації, окрім передачі API-ключа.

Другим варіантом є офіційна бібліотека google-api-python-client, що надається компанією Google. Вона є універсальним інструментом для роботи з великою кількістю сервісів Google, включаючи Google Books API. Попри свою багатофункціональність, ця бібліотека є більш складною у налаштуванні, передбачає авторизацію через OAuth2 і використовує об’єктну модель представлення відповіді, що дещо ускладнює роботи при реалізації простих запитів. Проте вона пропонує високий рівень абстракції, зручні інструменти для обробки структурованих даних і підтримку розширених можливостей.

З огляду на поставлені завдання та відсутність потреби в авторизації, у межах даної роботи було прийнято рішення використовувати саме бібліотеку requests, яка повністю задовольняє вимогу інтеграції з Google Books API.

2.2.3 Додаткові бібліотеки

Для реалізації функціоналу Telegram-бота, окрім основних бібліотек для роботи з Telegram API та Google Books API, використовується також додаткові інструменти, які забезпечують обробку природньої мови, роботу з базами даних і форматами обміну даними.

Однією з ключових бібліотек є Natural Language Toolkit (NLTK), яка широко використовується для базової обробки природньої мови в Python. У межах цього проєкту вона застосовується для токенізації запитів користувача, виділення ключових слів (зокрема жанру або імені автору), а також для попередньої фільтрації вхідного тексту з метою точнішого формулювання пошукового запиту.

Для збереження даних у проєкті використовується стандартна бібліотека sqlite3, яка забезпечує взаємодію з вбудованою реляційною базою даних. Зокрема, вона дозволяє зберігати історію запитів користувачів, кешувати результати пошуку книг для пришвидшення відповіді на повторні звернення, а також підтримує ведення бази авторів для полегшеного доступу до вже обробленої інформації.

Окрему роль відіграє бібліотека json, яка використовується для серіалізації та десеріалізації відповідей API. Це дає змогу зручно працювати з форматованими даними, які надходять від Google Books, і перетворювати їх у структури, придатні для обробки в коді.

Поєднання зазначених бібліотек дозволяє реалізувати всі необхідні компоненти для створення повноцінного інтелектуального Telegram-бота. Завдяки такому технологічному стеку забезпечується ефективна обробка користувацьких запитів, стабільна інтеграція з зовнішніми сервісами, а також надійне збереження й повторне використання даних. Це створює основу для побудови масштабованої та функціонально гнучкої системи рекомендацій книжкової літератури.

2.3 Методи обробки запитів і рекомендацій

Одним із основних функціональних елементів Telegram-бота є здатність коректно інтерпретувати запити користувача та надавати відповіді книжкові рекомендації. Цей процес складається з кількох послідовних етапів: обробки текстового запиту, вилучення ключових параметрів, формування запиту до зовнішнього API (у нашому випадку – Google Books API), обробки відповіді та фільтрації результатів.

2.3.1 Попередня обробка запиту

Після отримання повідомлення від користувача бот здійснює попередню обробку тексту. Для цього використовуються методи Natural Language Processing (NLP) з використанням бібліотек nltk або spaCy.

Основні кроки:

- Токенізація – розбиття тексту на окремі слова або фрази.
- Нормалізація – зведення слів до базової форми (лематизація або стемінг).
- Виділення ключових слів – пошук фраз, які вказують на жанр, ім’я автора, рік, настрій тощо.

Приклад:

Запит: «Порадь мені щось мотиваційне від українського автора»

Ключові слова: “мотиваційне”, “український автор”.

2.3.2 Формування параметрів пошуку

Після обробки запиту бот формує параметри HTTP-запиту до Google Books API, наприклад:

GET https://www.googleapis.com/books/v1/volumes?q=subject:motivation+inauthor:ukrainian&key=API_KEY

Таке формування дозволяє:

- Звузити результати до книг, які відповідають жанру або стилю;
- Фільтрувати за автором або за ключовими словами в назві;
- Обмежити кількість результатів або впорядкувати їх за релевантністю.

2.3.3 Обробка відповіді API

API повертає відповідь у форматі JSON, яка містить список книжкових записів. За допомогою модуля json бот аналізує:

- Назву книги;
- Автора(ів);
- Опис;
- Категорії;
- Рейтинг, кількість сторінок, рік публікації тощо.

Бот відбирає з результатів лише релевантні елементи й формує зручний текстовий блок для Telegram.

Приклад повідомлення користувачу:

Назва: The Power of Purpose

Автор: Richard Leider

Категорія: Self-Help

Рейтинг: 4.2/5

Посилання на книгу в Google Books

2.3.4 Кешування і повторна видача

Щоб уникнути дублювання запитів та знизити навантаження на API, результати пошуку зберігаються у локальній базі даних SQLite разом із параметрами пошуку. При повторному зверненні до аналогічного запиту бот витягує інформацію з кешу.

2.3.5 Принципи побудови рекомендацій

Хоча на даному етапі реалізується базований підбір на основі запиту, проєкт передбачає можливість подальшого вдосконалення механізму рекомендацій. У майбутньому функціонал може бути доповнено механізмами фільтрації результатів за рейтингом, наявністю рецензій або розгорнутістю опису книги. Це дозволить забезпечити більш якісну та структуровану видачу.

Крім того, потенційно можливо впровадження алгоритмів ранжування книжкових позицій відповідно до семантичної близькості запиту, що значно підвищить релевантність результатів. Не менш важливим напрямом розвитку є персоналізація, яка передбачає врахування історії запитів конкретного користувача, його уподобань, обраних жанрів та мови видання.

Реалізація таких функцій може здійснюватися на основі векторного аналізу текстових описів або шляхом використання бібліотек машинного навчання, які здатні виявляти приховані зв’язки між контентом і поведінкою користувача.

Можна зазначити, що застосовані методи обробки запитів у Telegram-боті поєднують класичні підходи до роботи з природною мовою (NLP) та інтеграцією з Google Book API. Така комбінація дозволяє досягти високої точності в надані рекомендацій, забезпечити швидку реакцію системи на звернення та створити основу для подальшого впровадження механізмів персоналізації.

2.4 Інструменти розробки, тестування та розгортання

2.4.1 Середовище розробки: Visual Studio Code (VS Code)

Для написання коду Telegram-бота обрано Visual Studio Code (VS Code) – безкоштовне, кросплатформенне середовище розробки від компанії Microsoft. Воно є одним із найпопулярніших інструментів серед Python-розробників завдяки гнучкості, розширюваності та інтуїтивному інтерфейсу.

Перевагами використання VS Code є підтримка Python через офіційне розширення з автодоповненням, інтегрованим відладчиком, запуском скриптів у терміналі. Не менш важливим є можливість налаштувати розширення для роботи з API, JSON, SQLite, Markdown, Git та Docker. Вбудований термінал для управління проєктом без переключання до зовнішньої консолі дає можливість зручно виконувати всі дії в межах одного застосунку. Зручним доповненням є інтеграція з Git, що дозволяє зручно виконувати коміти, змінювати гілки, переглядати історію прямо з редактора.

Порівняння з альтернативами (див. табл. 2.2):

Таблиця 2.2 – Порівняння середовищ розробки

Середовище розробки	**Переваги**	**Недоліки**
VS Code	Легка вага, розширення, інтеграція з Git, підтримка Python, безкоштовне	Потрібна ручна установка деяких модулів
PyCharm	Потужний IDE з глибокою інтеграцією Python	Важчий, деякі функції доступні лише у платній версії
Sublime Text	Швидкість, мінімалізм	Слабша інтеграція з Python і Git, платний
JupyterLab	Ідеальний для data science	Менш зручний для розробки повноцінних Telegram-ботів

VS Code обрано як найбільш збалансоване, гнучке й практичне середовище для розробки Python-бота.

2.4.2 Система контролю версій Git + GitHub

Для контролю змін у коді, організації колективної роботи та публікації проекту було використано систему контролю версій Git у поєднанні з платформою GitHub. Такий підхід забезпечує ефективне керування історією змін і дозволяє розробнику відстежувати будь-які редагування, коментувати конкретні коміти, а також повертатися до попередніх версій у разі потреби. Однією з важливих переваг Git є можливість створення паралельних гілок, що дає змогу працювати над різними модулями або функціональністю проєкту без ризику порушити стабільність основної гілки розробки.

Використання GitHub додатково розширює можливості керування проєктом. Зокрема, ця платформа дозволяє зручно розміщувати код у відкритому або приватному доступі, співпрацювати з іншими учасниками команди, а також використовувати інструменти безперервної інтеграції та розгортання. Завдяки цьому процес тестування, збирання та публікації змін може бути частково або повністю автоматизовані, що позитивно впливає на якість програмного забезпечення. Крім того, GitHub надає зручні засоби для документації проєкту, зокрема через README-файли, які можуть містити опис функціоналу, інструкції з запуску, ліцензійні умови та іншу важливу інформацію.

Порівняння з аналогами (див. табл. 2.3):

Таблиця 2.3 – Порівняння систем контролю версій та аналогів

Система	Переваги	Недоліки
Git + GitHub	Популярність, гнучкість, інтеграції, підтримка IDE	Потребує базового вивчення команд
Git + GitLab	Подібна функціональність, приватні репозиторії	Менша популярність у Python-спільноті
Mercurial (Bitbucket)	Простий синтаксис	Менш гнучка система гілок і злиття
Dropbox / Google Drive	Простота для новачків	Відсутність контролю версій, злиття, історії

Git + GitHub – це стандарт сучасної розробки. Він обраний через підтримку автоматизації, публічну документацію та широку спільноту.

2.4.3 Інструменти для розгортання

Для забезпечення стабільної роботи Telegram-бота перед його розгортанням у середовищі експлуатації було застосовано інструменти тестування, зокрема бібліотеку pytest. Вона використовується для перевірки працездатності окремих модулів, серед яких функції взаємодії з Google Books API, обробка відповідей у форматі JSON, а також розбір користувацьких запитів. Основною перевагою pytest є простота синтаксису, що дає змогу швидко створювати тести навіть на початковому етапі проєкту. Бібліотека дозволяє також здійснювати тестове покриття коду, виявляючи критичні помилки до того, як продукт буде розгорнуто або переданий користувачеві.

У межах реалізації проєкту було проведено модульне тестування (unit-тести), яке охоплює окремі функції логіки обробки запитів, перевірку правильності відповіді, отриманої з API, а також коректність роботи кешу, реалізованого на базі SQLite. Це дало змогу ізолювати потенційні джерела помилок і підвищити надійність окремих компонентів системи до моменту її інтеграції в цілісне рішення.

2.4.4 Розгортання Telegram-бота

Telegram-агент має функціонувати у цілодобовому режимі, бути доступним із будь-якого пристрою та забезпечувати безперебійну обробку запитів користувачів. Зважаючи на ці вимоги, для розгортання системи було обрано хмарну платформу Railway. Вона виявилась оптимальним рішенням для невеликих проектів завдяки простоті використання, безкоштовному тарифу на базовому рівні, а також спеціальній адаптації проєктів, реалізованих мовою Python.

Однією з ключових переваг Railway є вбудована інтеграція з GitHub, що дозволяє здійснювати автоматичне розгортання застосунку після кожного оновлення коду в репозиторії. Крім того, платформа підтримує налаштування змінних середовища, що забезпечує безпечне збереження API-ключів, токенів доступу до Telegram Bot API та інших параметрів конфігурації. Railway також надає можливість логування у реальному часі, що дає змогу відстежувати вхідні запити, внутрішні події та потенційні помилки. Окрему увагу варто звернути на зручність налаштування: для запуску проєкту немає потреби встановлювати додаткове програмне забезпечення або вручну конфігурувати середовище виконання.

Порівняння з аналогами (див. табл. 2.4):

Таблиця 2.4 – Порівняння хмарних сервісів та аналогів

Платформа	**Переваги**	**Недоліки**
Railway	Простота, автоматичний деплой, інтеграція з Git	Менша кількість гайдлайнів, ніж у Heroku
Heroku	Відомість, гарна документація, GUI	З 2022 року безкоштовні плани обмежено, сплячі процеси
PythonAnywhere	Спеціалізовано на Python	Обмежена підтримка Telegram API, ручне налаштування
Render	Зручний веб-інтерфейс	Менш оптимізований для Telegram-ботів, довший деплой
VPS	Повний контроль, кастомізація	Вимагає Linux-адміністрування, не підходить новачкам

Таким чином, Railway дозволяє забезпечити ефективне, доступне та безперервне розгортання Telegram-бота, що відповідає вимогам до хмарних застосунків.

2.4.5 Додаткові інструменти

Для тестування та візуалізації роботи Telegram-бота під час розробки було використано низку додаткових інструментів, які суттєво полегшують роботу з API, базами даних і сценаріями тестування.

Одним із таких інструментів є Postman — десктопний застосунок, що дозволяє надсилати тестові запити до сторонніх API, зокрема до Google Books API, без потреби запуску всього застосунку. Серед основних переваг варто відзначити наявність зручного візуального інтерфейсу для формування запитів типу GET або POST, миттєвий перегляд структури відповіді у форматі JSON, можливість імітації запитів незалежно від коду бота, а також функцію збереження колекцій і сценаріїв тестування для повторного використання. Існують також альтернативи, зокрема curl, HTTPie та Insomnia, однак їх основним недоліком є потреба у володінні

командним рядком, а також відсутність зручного графічного інтерфейсу, що знижує доступність для початківців.

Ще одним корисним інструментом є SQLiteStudio — графічне середовище для роботи з локальними базами даних SQLite. У цьому проєкті воно використовується для візуального аналізу таблиць, пов'язаних з історією запитів користувача, а також для налагодження структури сховища. Інструмент дозволяє виконувати SQL-запити безпосередньо у графічному вікні, переглядати вміст таблиць, редагувати записи й перевіряти схему бази без потреби запуску зовнішніх скриптів. Його перевагами є простота використання, відсутність необхідності встановлення додаткових серверів, зручна візуалізація структури БД та легка інтеграція з файлами .db, що зберігаються в межах проєкту. Альтернативою може бути CLI-інструмент SQLite Shell, однак у порівнянні з ним SQLiteStudio є значно зручнішим завдяки інтуїтивному інтерфейсу й підтримці ручного налагодження.

Підбір усіх інструментів здійснювався з урахуванням простоти, функціональності та доступності для швидкої інтеграції у процес розробки. Вибрані платформи повністю покривають життєвий цикл бота — від написання коду й тестування окремих модулів до розгортання та супроводу, забезпечуючи ефективну розробку в сучасному технологічному середовищі.

2.5 Обґрунтування вибору технологічного стеку

Вибір відповідного стеку є ключовим етапом при розробці будь-якого програмного забезпечення, особливо якщо мова йде про інтерактивного Telegram-бота. У межах цього проєкту ставилось завдання створити систему, яка забезпечує стабільну взаємодію з користувачем, високу швидкість обробки запитів, доступ до книжкових даних через Google Books API, а також можливість персоналізованого надання рекомендацій на основі параметрів, що вводяться користувачем.

З огляду на зазначені цілі, вибір технологій здійснювався з урахуванням кількох критично важливих критеріїв. Одним із таких була наявність сучасних

бібліотек для взаємодії з Telegram API та зовнішніми веб-сервісами, які могли б забезпечити просту й ефективну інтеграцію. Додатково важливим чинником була підтримка засобів для обробки природної мови (NLP), оскільки розуміння тексту запитів є основним компонентом логіки бота.

Також враховувалася простота розгортання, тестування й обслуговування програмного продукту, що дозволяє зменшити витрати на підтримку в майбутньому. Не менш значущою виявилася гнучкість обраного середовища при масштабуванні системи, що дає змогу адаптуватися до зростаючої кількості користувачів. Останнім, але не менш важливим фактором, була активність спільноти розробників і наявність актуальної документації, яка суттєво полегшує процес розробки, дозволяє швидше знаходити рішення на типові проблеми та інтегрувати нові функції без зайвих витрат часу.

На основі порівняльного аналізу(див. п. 2.4) сформовано стек технологій, який буде використано в дипломному проєкті (див. табл. 2.5):

Таблиця 2.5 – Обрані компоненти технологічного стеку

Компонент	**Обраний інструмент**	**Обґрунтування вибору**
Мова програмування	Python	Простий синтаксис, гнучкість, велика кількість бібліотек для NLP, API, тестування.
Telegram API	Aiogram	Асинхронна модель, FSM, масштабованість, розвинена документація.
API-клієнт	Requests	Простота формування запитів до Google Books API, контроль параметрів.
NLP	Nltk	Мінімальні вимоги, швидкий старт, базова обробка запитів користувача.
Сховище даних	SQLite	Простота використання, відсутність серверної частини, збереження кешу та історії.
Середовище розробки	VS Code	Легка вага, підтримка Python, інтеграція з Git, розширення для SQLite, JSON, API.
Контроль версій	Git + GitHub	Гнучкий контроль історії змін, підтримка деплою через GitHub, можливість CI/CD.
Тестування	Pytest	Гнучка структура тестів, проста конфігурація.
Розгортання	Railway	Автоматичне деплоювання, підтримка Python, безкоштовний тариф для невеликих проєктів.
Допоміжні інструменти	Postman, SQLiteStudio	Зручна перевірка API-запитів, візуальний аналіз БД.

Аргументи на користь обраного стеку:

1. Інструменти адаптовані під Telegram – вибрані бібліотеки (aiogram, requests) тісно інтегруються з Telegram Bot API і забезпечують швидкій обмін даними.

2. Підтримка NLP – nltk надає мінімальний набір інструментів для

синтаксичного розбору, що відповідає поставленим задачам.

3. Простота розгортання – Railway дозволяє миттєво публікувати бота, а GitHub інтегрує контроль версій із хостингом.

4. Мінімальні апаратні вимоги – весь стек легко працює на локальній машині або у хмарному сховищі без додаткових витрат.

Обраний стек технологій є збалансованим та оптимальним для реалізації дипломного проєкту. Він забезпечує швидку розробку, стабільну роботу, прозору структуру й можливість подальшого вдосконалення. Усі компоненти мають відкриту ліцензію, активну спільноту користувачів та велику кількість готових рішень, що робить їх придатними для як навчальних, так і комерційних проєктів.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ TELEGRAM-БОТА

3.1 Проєктування архітектури чат-бота

Перед початком реалізації Telegram-бота необхідно визначити архітектуру системи, її ключові модулі, схему взаємодій між компонентами та потоки обробки даних. Грамотно побудована архітектура є запорукою стабільності, розширюваності та зручності в підтримці проєкту.

3.1.1 Функціональні вимоги до системи

На основі проведеного аналізу цільової аудиторії та сформованих у першому розділі завдань, Telegram-бот має реалізовувати набір функціональностей, які забезпечують його ефективну роботу як системи пошуку та рекомендацій книжкової літератури. Передусім, бот повинен уміти обробляти текстові запити користувачів, здійснюючи попередній аналіз вмісту повідомлень.

У межах цієї обробки система має виявляти ключові параметри запиту, такі як жанр, автор або емоційна складова (наприклад, настрій читача). На основі виявленої інформації формуються запити до Google Books API, після чого бот здійснює обробку та фільтрацію отриманих результатів відповідно до релевантності й змістовного наповнення.

Бот повинен надавати користувачу персоналізовану рекомендацію, яка максимально відповідає заданим параметрам. Для цього реалізується збереження історії запитів у локальній базі даних, що дозволяє надалі враховувати уподобання користувача. Крім того, система повинна кешувати відповіді від API, що знижує навантаження на зовнішній сервіс і пришвидшує повторні звернення користувача.

Оскільки Telegram-бот має бути доступним у будь-який час, його архітектура передбачає роботу в цілодобовому режимі після розгортання, без необхідності технічного втручання з боку адміністратора.

3.1.2 Загальна архітектура бота

Бот реалізується за модульною структурою, що дозволяє розділити логіку за окремими відповідальностями:

- Модуль взаємодії з Telegram API (через aiogram);
- Модуль обробки тексту (NLP-функції);
- Модуль API-запитів до Google Books;
- Модуль форматування та видачі результатів;
- Модуль роботи з базою даних (SQLite);
- Модуль кешування запитів;
- Модуль налаштувань і конфігурацій (ключі, токени, константи).

3.1.3 Потік даних(логіка запиту)

1. Користувач надсилає повідомлення боту (наприклад: “Підкажи новий детектив”).
2. Модуль Telegram (aiogram) отримує повідомлення.
3. NLP-модуль виділяє ключові параметри – жанр детектив.
4. Бот перевіряє, чи існує відповідь на подібний запит у кеші (SQLite база даних).
    - - Якщо відповідь знайдено – бот формує відповідь на основі збережених даних і переходить до пункту 9.
    - Якщо відповідь відсутня – виконується запит до Google Books API.
5. Формується HTTP-запит до Google Books API на основі параметрів запиту користувача.
6. API повертає список книг у форматі JSON.
7. Бот обробляє відповідь: перевіряється наявність назви, авторів, жанру, рейтингу, опису.
8. Зібрані результати зберігаються у кеш SQLite, разом із параметрами

пошуку.

9. Користувачу надсилається структурована відповідь із результатами пошуку.

На рисунку 3.1 можна спостерігати графічне відображення потоку даних.

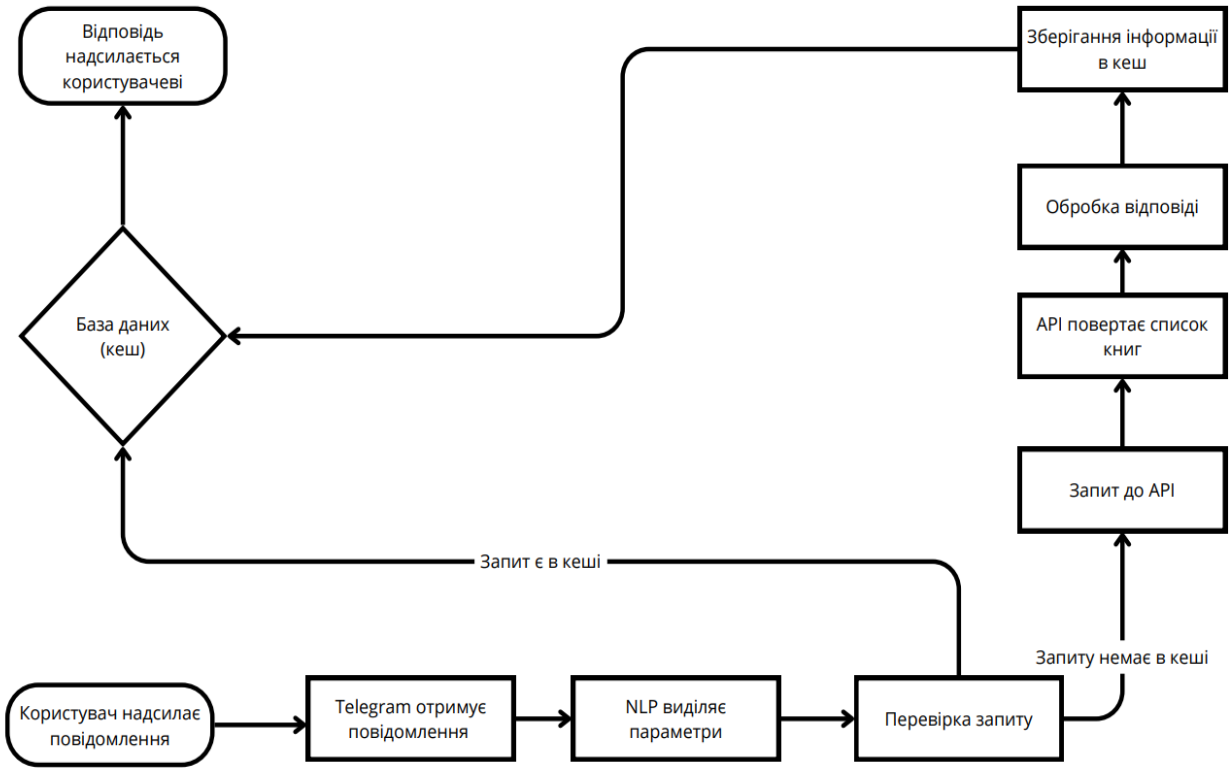


Рисунок 3.1 – Діаграма потоку даних

3.1.4 Компонента діаграма

Для візуалізації архітектури системи можна виділити такі ключові компоненти:

- Telegram Handler: отримує і відправляє повідомлення користувачу.
- Text Processor (NLP): аналізує повідомлення, вилучає запит.
- API Client: формує запит до Google Books API та обробляє JSON-відповідь.
- Response Builder: створює повідомлення у форматі, зручному для

Telegram.

- SQLite Manager: зберігає історію запитів, реалізує кеш.
- Config Loader: містить змінні середовища (ключі, токени).
- Cache Controller: Перевіряє, чи є результат у базі, та витягує його.

3.1.5 Технологічні взаємозв'язки

- Telegram Handler ↔ Text Processor
- Text Processor ➞ SQLite Manager (перевірка кешу)
- SQLite Manager ➞ API Client (якщо кеш відсутній)
- API Client ➞ Response Builder
- Response Builder ➞ Telegram Handler
- API Client ➞ SQLite Manager (запис нового кешу)
- Config Loader використовується всередині всіх вищезазначених модулів
- Pytest тестує: Text Processor, API Client, SQLite Manger

На рисунку 3.2 можна спостерігати діаграму технологічних зв'язків.

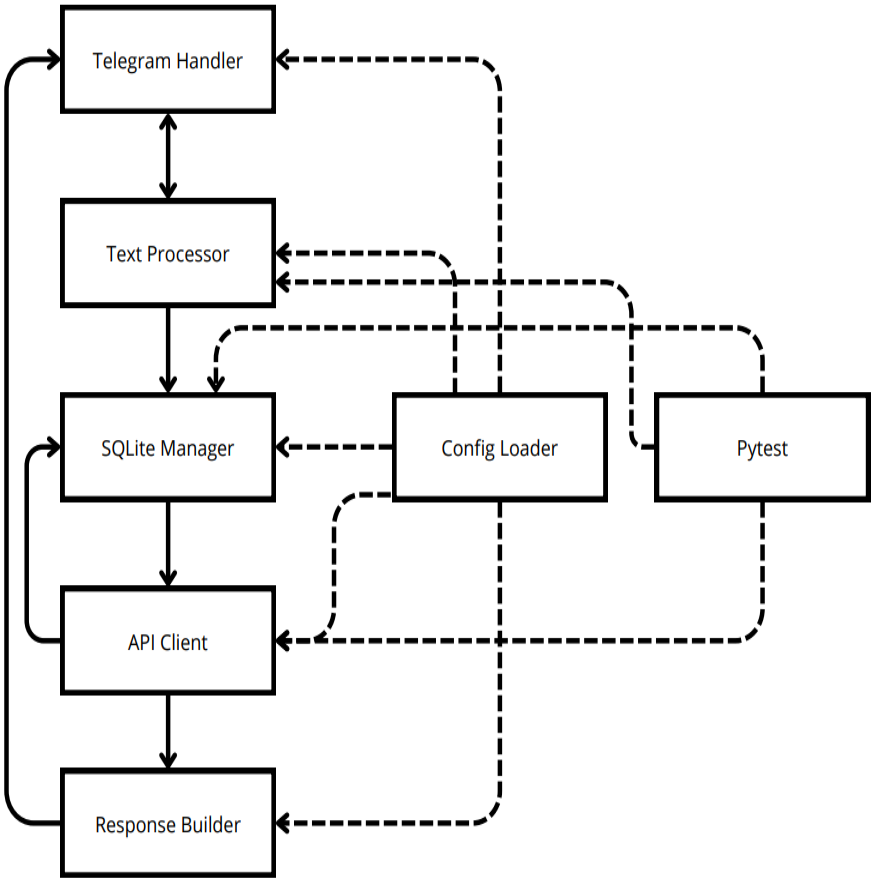


Рисунок 3.2 – Діаграма технологічних зв’язків

3.1.6 Переваги обраної архітектури

Архітектура Telegram-бота, розроблена в межах даного проєкту, базується на принципах модульності, простоти розгортання, масштабованості та оптимізації запитів. Однією з ключових переваг є її модульна структура, яка дозволяє тестувати та змінювати окремі компоненти незалежно один від одного. Це спрощує як початкову розробку, так і подальше обслуговування системи.

Архітектура проєкту добре масштабована: до неї легко додати нові функції, зокрема інтеграцію з іншими API або розширення функціоналу бота без порушення існуючої логіки. Такий підхід дозволяє забезпечити гнучкість при розвитку системи й адаптацію до нових вимог користувача.

Окрему увагу приділено питанню підтримки. Завдяки зрозумілій структурі коду, з чітким розмежуванням логіки між модулями, проєкт може бути швидко адаптований або доповнений навіть стороннім розробником без потреби вивчення всієї системи з нуля.

З погляду безпеки, бот реалізований із дотриманням основних рекомендацій: усі токени, API-ключі та конфіденційні параметри зберігаються у змінних середовища, що виключає ризик випадкового витоку даних через публічний код.

Узагальнюючи, можна сказати, що обрана архітектура забезпечує не лише надійність і стабільність роботи Telegram-бота, а й створює міцну основу для його подальшого вдосконалення. Такий підхід дозволяє у перспективі реалізувати персоналізовані рекомендації, інтеграцію з іншими сервісами.

3.2 Інтеграція з Google Books API

3.2.1 Отримання API-ключа Google Books API

Для забезпечення зв’язку Telegram-бота з Google Books API необхідно спочатку отримати індивідуальний API-ключ. Цей ключ є обов’язковим параметром усіх запитів до API та дає змогу здійснювати авторизовані звернення до сервісу.

Алгоритм отримання API ключа:

1. Увійти до облікового запису Google і перейти на офіційний портал Google Cloud console.
2. Створити новий проєкт (див. рис. 3.3).
3. У меню навігації перейти до розділу “API & Services – Library” та знайти сервіс Google Books API (див. рис. 3.4).
4. Натиснути “Enable” для активації API у межах поточного проєкту.
5. Перейти до розділу “Credentials” й натиснути “Create credentials” – “API

key” (див. рис. 3.5).

6. Скопіювати згенерований ключ і зберегти його у файл .env щоб уникнути витоку чутливої інформації (див. рис. 3.6).

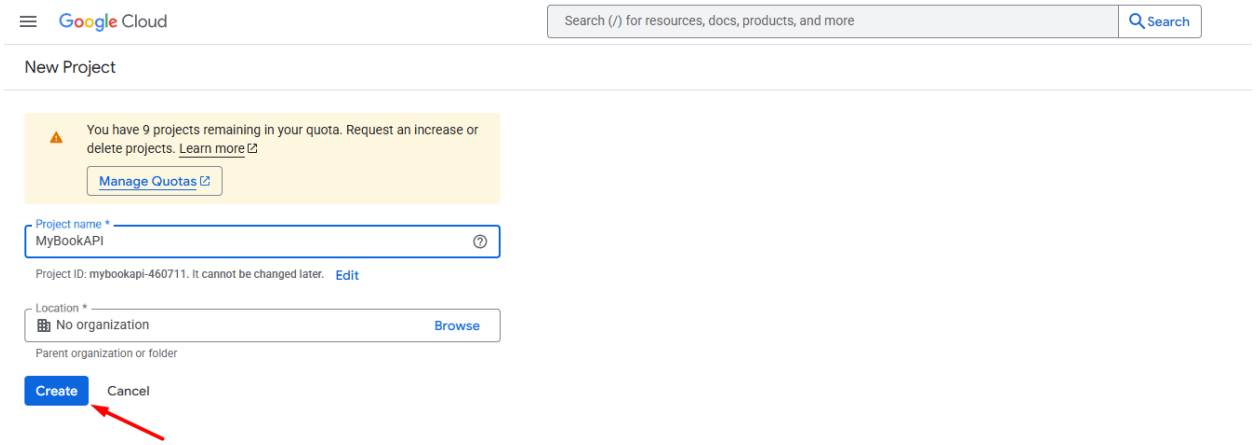


Рисунок 3.3 – Створення проєкту

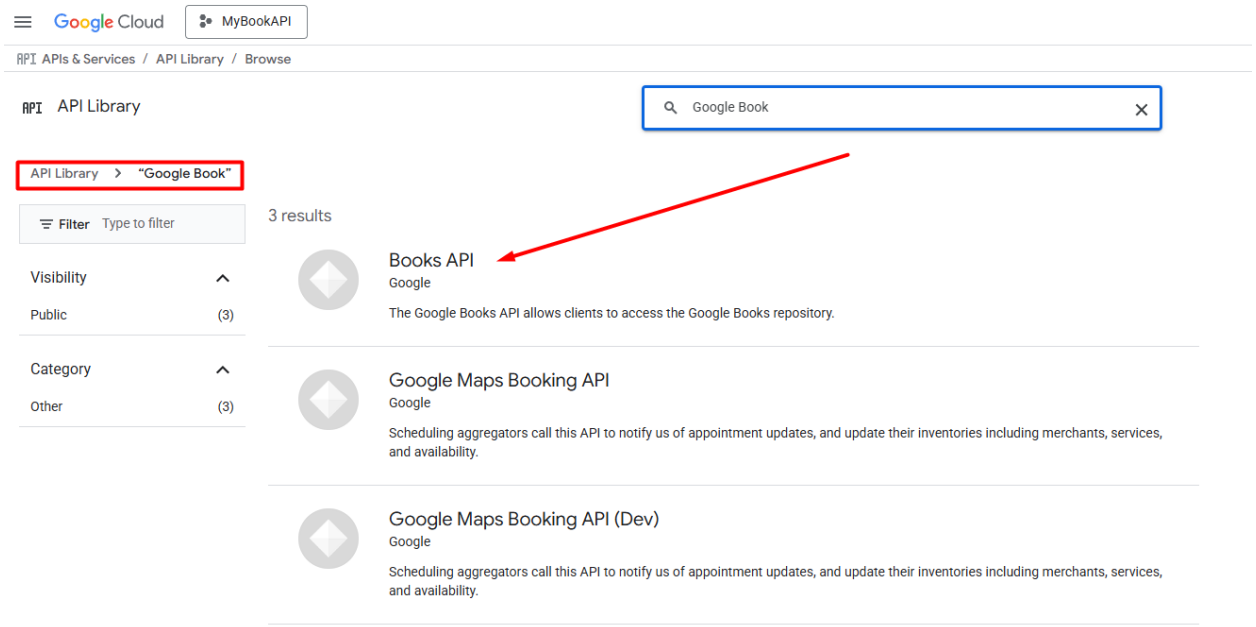


Рисунок 3.4 – Перехід до Google Books API

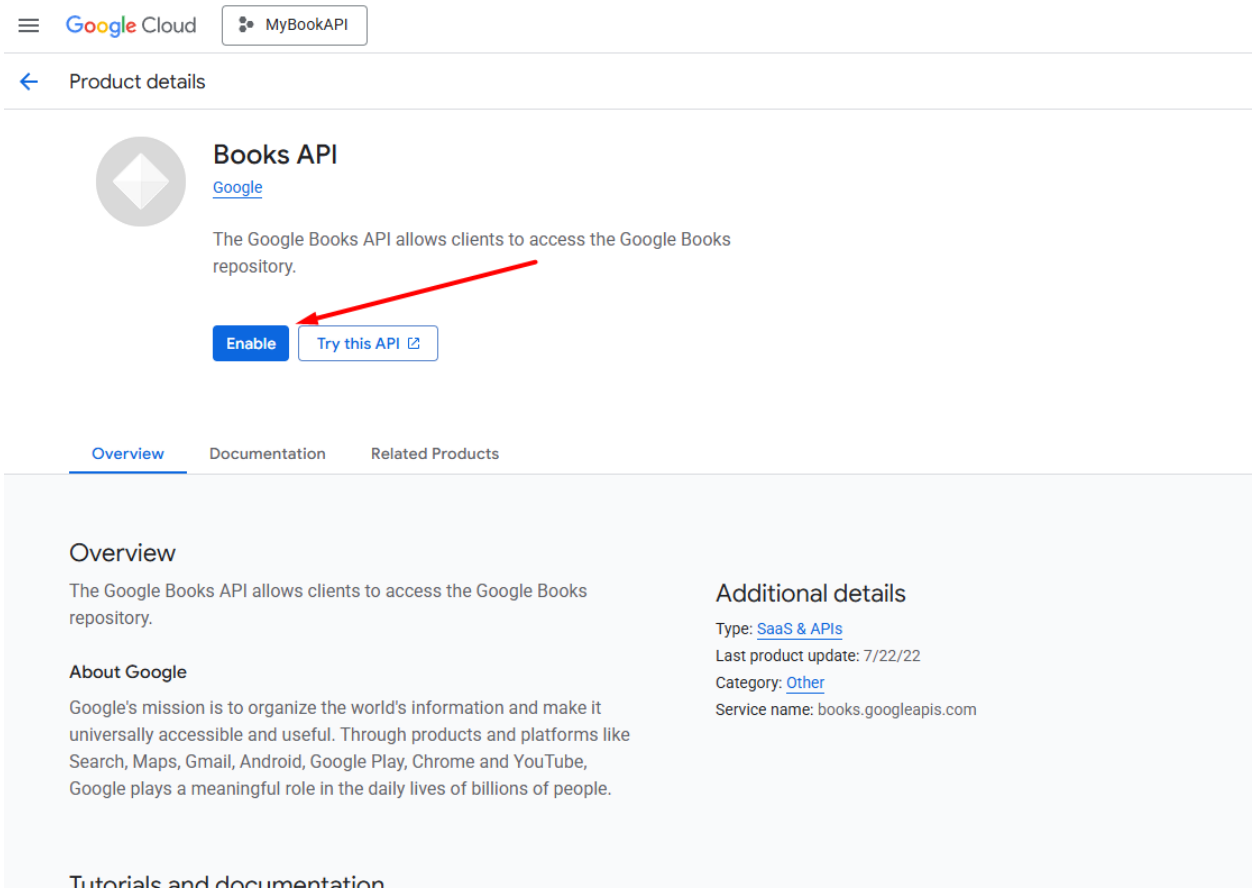


Рисунок 3.5– Активація Google Books API

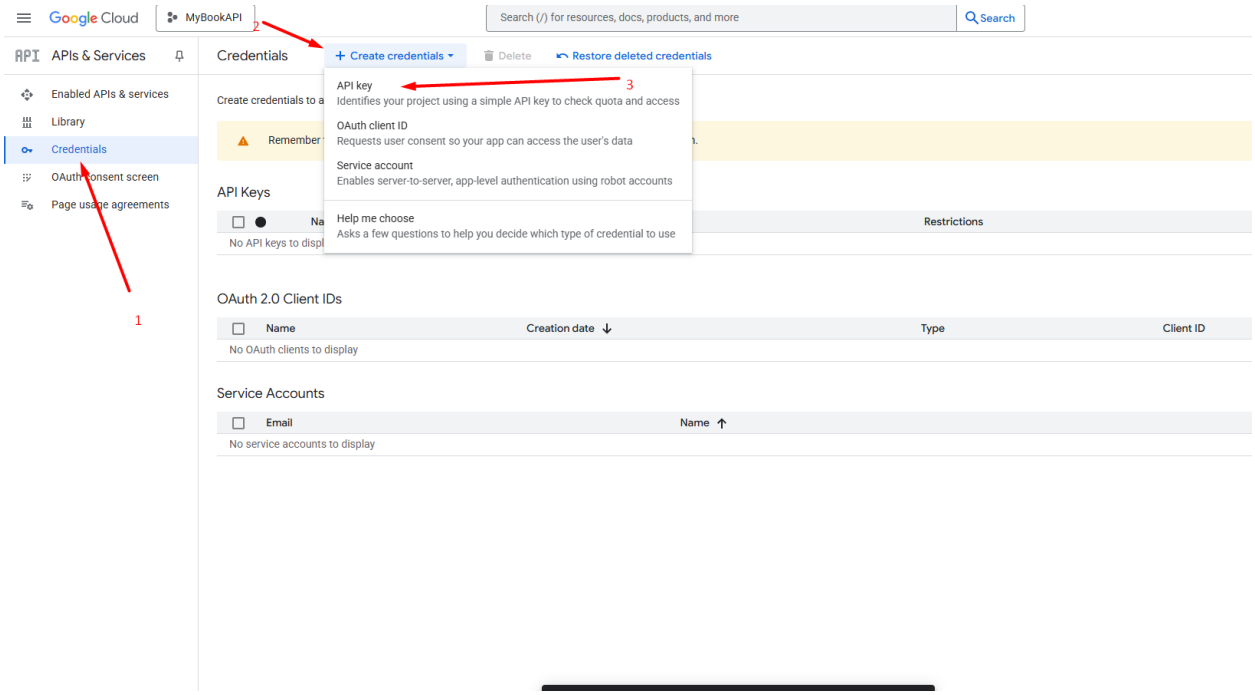


Рисунок 3.6 – Створення API ключа

Приклад збереження API ключа в .env (див. рис. 3.7):

```
.env
1    GOOGLE_BOOKS_API_KEY=AIzaSy....token....
```

Рисунок 3.7 – Приклад збереження API ключа

А потім – зчитування ключа в коді з config.py (див. рис. 3.8):

```
config.py > ...
1    # config.py
2    import os
3    from dotenv import load_dotenv
4
5
6    load_dotenv()
7    GOOGLE_BOOKS_API_KEY = os.getenv("GOOGLE_BOOKS_API_KEY")
```

Рисунок 3.8 – Зчитування ключа API в коді.

Такий підхід дозволяє безпечно працювати з API без ризику витоку даних та легко змінювати ключ у майбутньому.

3.2.2 Формування запитів

Формування HTTP-запитів реалізовано у модулі google_books.py. У функції fetch_books(query, language='en') виконується запит до API з параметрами (див. рис. 3.9):

- Пошукова фраза (q) – створюється з ключових слів (наприклад, жанр, автор);
- Обмеження за мовою (langRestrict);
- Сортування (orderBy='relevance');
- Кількість результатів (maxResults=5);
- API-ключ.

```python
# google_books.py
def fetch_books(query, language="en", max_results=5):
    url = "https://www.googleapis.com/books/v1/volumes"
    params = {
        "q": query,
        "langRestrict": language,
        "orderBy": "relevance",
        "maxResults": max_results,
        "key": API_KEY
    }
    response = requests.get(url, params=params)
    ...
```

Рисунок 3.9 – Приклад звернення до API Google Book

Цей підхід дозволяє гнучко формувати запити на основі введених користувачем даних.

3.2.3 Обробка відповіді

Отримання з API результати мають формат JSON. Бот аналізує об’єкти масиву items та видобуває з кожного:

- Назву книги (title);
- Список авторів (authors);
- Короткий опис (description);
- Жанри (categories);
- Рейтинг (averageRating);
- Обкладинку (imageLinks.thumbnail);
- Посилання на Google Gooks (infoLink);

Усе це реалізовано у тій же функції fetch_books() (рис. 3.9). Для кожної книги формується словник із ключовими даними, які потім передаються далі в модуль форматування.

3.2.4 Форматування результату

Для зручності сприйняття результатів, інформація про книгу оформлюється у вигляді форматованого повідомлення з використанням Markdown. Це реалізовано у модулі response_formatter.py, функція format_book_response(book: dict) (див. рис. 3.10).

```
# response_formatter.py
def format_book_response(book: dict) -> str:
    ...
    return (
        f"*📖 Назва:* {title}\n"
        f"*✍️ Автор(и):* {authors}\n"
        f"*📚 Жанр:* {categories}\n"
        f"*⭐ Рейтинг:* {rating}\n"
        f"[🔗 Відкрити книгу]({info_link})"
    )
```

Рисунок 3.10 – Приклад форматування відповіді

Такий підхід дозволяє отримати стилістично зрозумілу та зручну відповідь у Telegram, не виходячи за межі звичного формату месенджера.

Узагальнюючи, Інтеграція Telegram-бота реалізована через спеціалізований модуль google_books.py, який виконує HTTP-запити, обробляє відповіді та передає дані у вигляді структурованих словників. API-ключ безпечно зберігається в .env файлі, а обробка даних дозволяє швидко й ефективно отримати інформацію про книги на основі користувацьких запитів.

3.3 Організація локального сховища даних

Для забезпечення швидкої роботи Telegram-бота та зменшення кількості повторних звернень до API до зовнішнього Google Books API реалізовано локальне

кешування запитів і відповідей. У якості сховища використано вбудовану реляційну базу даних SQLite, яка повністю інтегрована в Python через стандартний модуль sqlite3.

3.3.1 Структура бази даних

Уся інформація зберігається в таблиці cache. Створення бази відбувається автоматично при запуску бота через модуль init_db.py. Таблиця має таку структуру (див. рис. 3.11):

```sql
CREATE TABLE IF NOT EXISTS cache (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    query TEXT NOT NULL,
    response TEXT NOT NULL,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Рисунок 3.11 – Приклад структури бази даних

3.3.2 Реалізація кешування

Логіка кешування реалізована в модулі cache_manager.py, який виконує дві основні функції:

Отримання кешу (див. рис. 3.12):

```python
def get_cached_response(query):
    ...
    cursor.execute("SELECT response FROM cache WHERE query = ?", (query,))
    ...
```

Рисунок 3.12 – Приклад отримання кешу

Запис у кеш (див. рис. 3.13):

```
def save_to_cache(query, response):
    ...
    cursor.execute("INSERT INTO cache (query, response) VALUES (?, ?)", (query, response))
    ...
```

Рисунок 3.13 – Приклад запису кешу

Усі функції працюють через стандартне підключення до sqlite3 із використанням DB_PATH, що зчитується з config.py.

3.3.3 Послідовність роботи з кешем у боті

1. Коли користувач надсилає запит – обробка починається в модулі message_handler.py.
2. Запит уніфікується (очищається від зайвих пробілів, регістру, тощо) та передається до get_cached_response().
3. Якщо результат у кеші є – він одразу повертається користувачу.
4. Якщо в кеші нічого не знайдено – викликається fetch_books() із модуля google_books.py.
5. Отримана відповідь з API зберігається у кеш через save_to_cache() для подальшого використання.

Реалізація локального кешування дозволила підвищити швидкодію Telegram-бота, зменшити кількість звернень до API та забезпечити збереження вже отриманих результатів. Така архітектура з використанням SQLite є зручною, мінімалістичною та ефективною для невеликих Telegram-застосунків.

3.4 Розробка алгоритмів обробки запитів і рекомендацій

Ефективна розробка Telegram-бота неможлива без правильної інтерпретації користувачем запитів. Для цього реалізовано механізм обробки природньої мови (NLP), який дозволяє виявити ключові слова, що стосуються жанру, автора,

тематики або настрою книги. На основі ключів формуються параметри для запиту до Google Books API.

3.4.1 Попередня обробка запиту (NLP)

Обробка тексту здійснюється в модулі text_processor.py (див. рис. 3.14). Основна функція extract_keywords(text) виконує:

- Приведення всього тексту до нижнього регістру;
- Видалення пунктуації;
- Токенізацію (розбиття на слова);
- Фільтрацію стоп-слів (наприклад: «порадь, «мені», «щось», «будь ласка»);
- Визначення жанру або категорії (на основі словника жанрів).

```
def extract_keywords(text: str) -> dict:
    ...
    return {
        "genre": detected_genre,
        "author": author_name,
        "language": language_code
    }
```

Рисунок 3.14 – Приклад структури функції extract_keywords()

Бібліотека nltk використовується для токенізації та базової фільтрації, а список стоп-слів і жанрів формує словник усередині коду.

3.4.2 Виділення параметрів запиту

Після обробки користувацького тексту функція повертає словник з ключовими параметрами.

Наприклад:

“Порадь мені щось з української фантастики”

➔ `{“genre”: “fantasy”, “language”: “uk”}`

Якщо не виявлено жодного ключа – бот повертає відповідь-заглушку з проханням уточнити запит.

3.4.3 Формування запитів до API

Після успішного вилучення ключових слів параметрів функція fetch_books() з модуля google_books.py отримує сформований рядок для запиту, наприклад:

subject:fantasy+inauthor:Tolkien

Інші параметри запиту – laingRescrict, orderBy, maxResults, key – задаються за замовчуванням.

Таким чином, весь ланцюжок виглядає так:

1. Користувач: “Підбери щось із мотивації українською”
2. Extract_keywords() – {“genre”: motivation”, “language”: “uk”}
3. Сформований запит – q=subject:motivation&langResctict=uk
4. Запит надсилається до Google Books API
5. Отриманий список книг передається на форматування

3.4.4 Форматування відповіді та рекодація

Після отримання результатів запиту з Goolge Books API список обробляється функцією format_book_response() (модуль response_formatter.py), яка повертає зручне для Telegram повідомлення у Markdown-форматі (див. рис. 3.15):

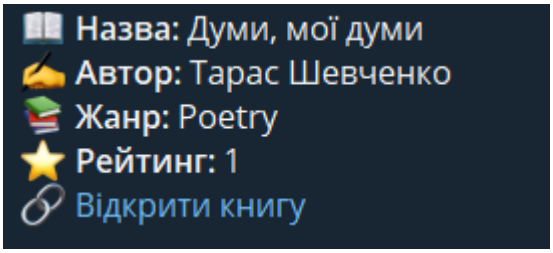


Рисунок 3.15 – Приклад оформлення відповіді ботом

Цей блок одразу передається користувачеві.

У цьому підрозділі реалізовано логіку, що поєднує NLP-аналіз запиту, форматування для зовнішнього API та створення результату у форматі, зручному для кінцевого користувача. Такий підхід дозволяє досягати персоналізованих рекомендацій на основі навіть простих запитів у природній мові.

3.5 Реалізація Telegram-бота

3.5.1 Налаштування @BotFather

Telegram-боти створюються через офіційного бота @BotFather – інструмент, що дозволяє зареєструвати нового бота, налаштувати його параметри та отримати унікальний токен для програмної взаємодії через Telegram Bot API.

Покрокова інструкція реєстрації бота

1. У Telegram відкрити бот @BotFather та натиснути кнопку Start.
2. Ввести команду /newbot або обрати її в меню для створення бота.
3. Ввести назву бота (наприклад: SearchBook)
4. Ввести унікальне ім’я користувача, що закінчується на bot (наприклад: BookAdvisorBot).
5. Після успішної реєстрації @BotFather надає токен доступу, який виглядає так: 12…9:ABC….xyZ (див. рис. 3.16).

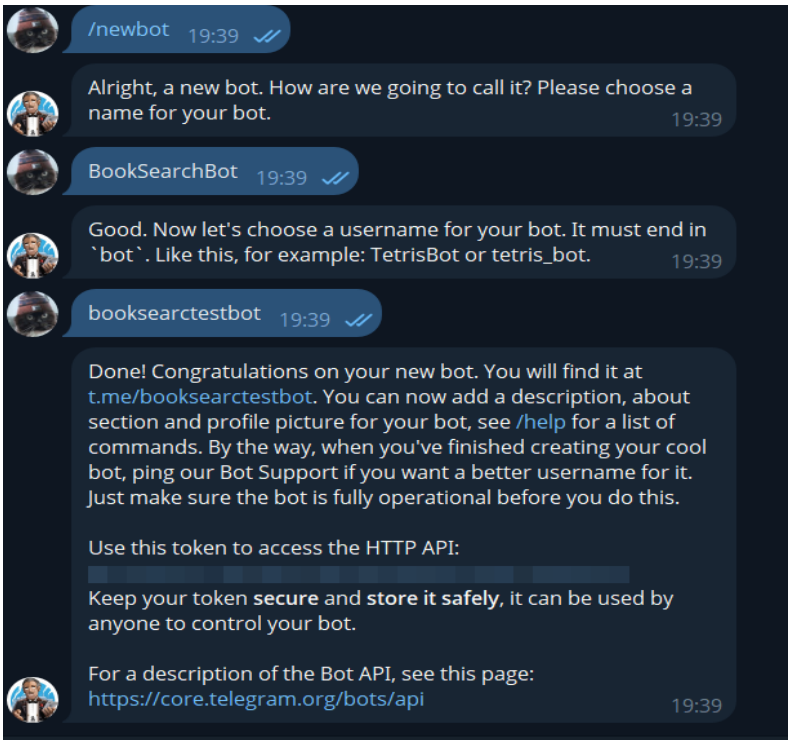


Рисунок 3.16– Реєстрація бота

Застосування токена в коді

Отриманий токен не зберігається напряму у коді, а виноситься у .env файл (див. рис. 3.17):

Рисунок 3.17 – Винесення токену в .env файл

Далі токен зчитується через модуль config.py за допомогою бібліотеки dotenv (див. рис. 3.18):

```python
# config.py
from dotenv import load_dotenv
import os

load_dotenv()
BOT_TOKEN = os.getenv("TELEGRAM_BOT_TOKEN")
```

Рисунок 3.18 – Зчитування токену для використання в коді

Це дозволяє уникнути витоку чутливої інформації при розгортанні або публікації проєкту у відкритому репозиторії.

Висновок: Реєстрація Telegram-бота через @BotFather є обов'язковим етапом для інтеграції Telegram API. Отриманий токен є унікальним і використовується у програмному коді для авторизації. Дотримання принципів безпеки (зберігання токена в .env) дозволяє підтримувати чисту архітектуру та уникнути витоку конфіденційних даних.

3.5.2 Реєстрація хендлерів у бібліотеці aiogram

Telegram-бот реалізований із використанням асинхронної бібліотеки aiogram, яка забезпечує ефективну обробку великої кількості одночасних запитів. Усі повідомлення від користувачів обробляються через спеціальні функції – хендлери, зареєстровані у Dispatcher.

Ініціалізація бота та диспетчера

У файлі main.py створюються об'єкти Bot та Dispatcher, а також обробник повідомлень (див. рис 3.19):

```python
from aiogram import Bot, Dispatcher
from aiogram.types import Message
from message_handler import register_handlers
from config import BOT_TOKEN

bot = Bot(token=BOT_TOKEN)
dp = Dispatcher()

register_handlers(dp)

dp.run_polling(bot)
```

Рисунок 3.19 – Приклад коду ініціалізації бота та диспетчера

Функція register_handlers(dp) імпортується з модуля message_handler.py і реєструє всі обробники бот-команд і звичайних повідомлень.

Для реєстрації обробка повідомлень у message_handler.py використовується декоратор @dp.message() для реєстрації функції, яка реагує на текстові повідомлення від користувача (див. рис. 3.20).

```python
def register_handlers(dp: Dispatcher):
@dp.message()
async def handle_message(message: Message):
    ...
```

Рисунок 3.20 – Приклад коду з використанням @dp_message()

Після отримання текстового повідомлення від користувача Telegram-бота передає його на обробку до NLP-модуля, який аналізує вхідний текст і виділяє ключові параметри запиту. Далі система перевіряє, чи існує відповідь на подібний запит у локальному кеші. Якщо результат уже збережено, бот витягує його без звернення до зовнішніх сервісів. У протилежному випадку відбувається формування запиту до Google Books API, після чого отримана відповідь обробляється та надсилається користувачеві.

Застосований підхід має кілька суттєвих переваг. Насамперед, уся взаємодія побудована на основі асинхронного виконання за допомогою конструкцій async def, що забезпечує обробку запитів у неблокуючому режимі. Це дозволяє боту підтримувати високу швидкодію навіть при великій кількості одночасних звернень. Крім того, логіка роботи системи чітко розділена за модулями: окремо реалізовано блок обробки тексту, компонент для кешування відповідей, а також модуль для взаємодії з API. Така модульність не лише підвищує зручність супроводу коду, а й сприяє його масштабуванню. У майбутньому до проєкту можна легко додати додаткові обробники команд, інтерактивні меню або кнопки без потреби в суттєвому переписуванні основної архітектури.

Використання бібліотеки aiogram у поєднанні з модульною структурою дозволяє реалізувати надійного, швидкого та масштабованого Telegram-бота. Усі повідомлення користувачів централізовано обробляються через Dispatcher, що спрощує підтримку та подальший розвиток проєекту.

3.5.3 Структура діалогу з користувачем

Telegram-бот, реалізований у межах даного проєкту, має просту, лаконічну та інтуїтивну структуру взаємодії з користувачем. Основна мета- дозволити користувачу швидко надіслати запит у вільній формі і отримати релевантну рекомендацію книги.

Типовий сценарій взаємодії

1. Користувач відкриває бота й вводить текстовий запит у природній мові – «Порадь щось з фантастики українською».
2. Бот виділяє ключові параметри запиту (genre, language).
3. Відбувається перевірка кешу - запит до Google API – форматування результату.
4. Користувачу повертається повідомлення у зручному вигляді (див. рис. 3.21):

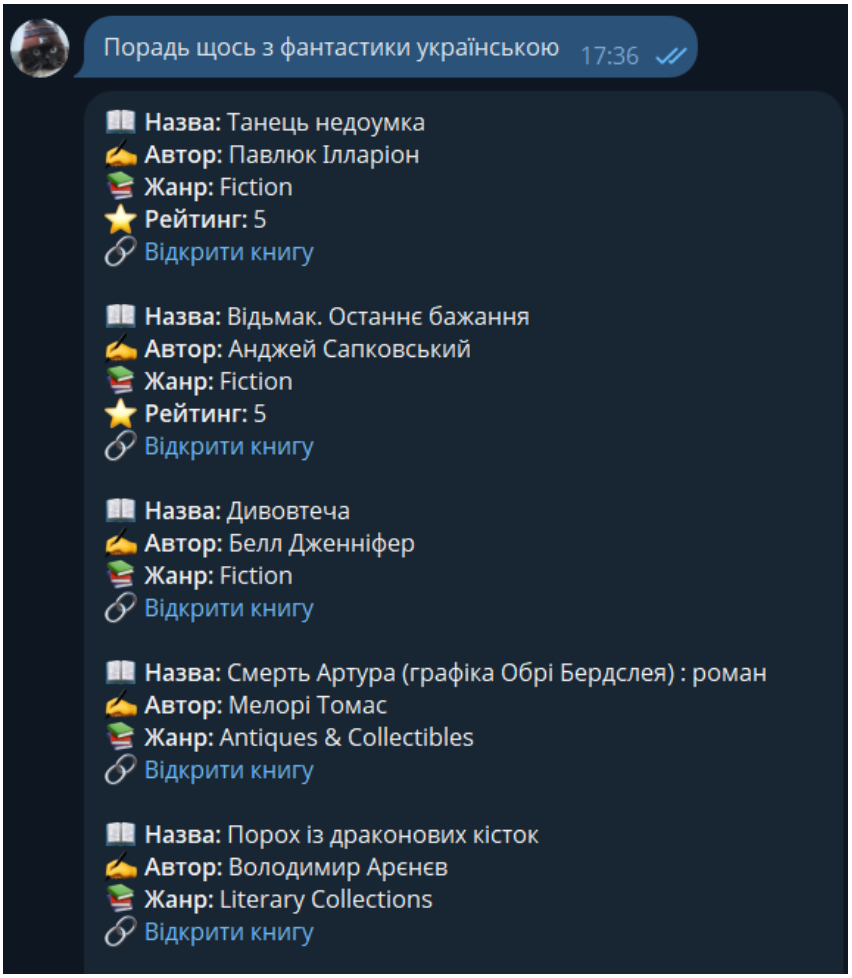


Рисунок 3.21 – Приклад типового запиту користувача

Поведінка у випадку помилки

Якщо запит порожній або не розпізнано ключових слів:

Бот відповідає фразою заглушкою (див. рис. 3.22):

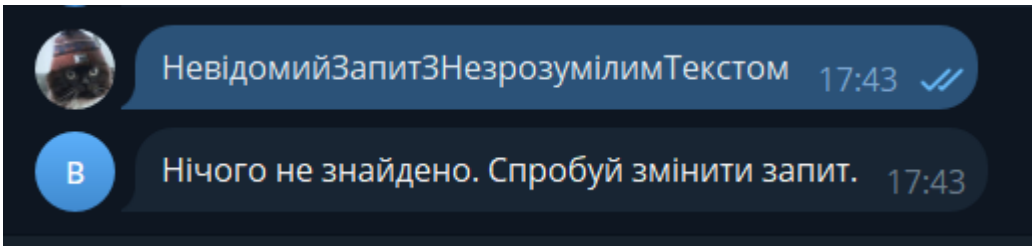


Рисунок 3.22 – Поведінка боту у випадку помилки

Структура діалогу між Telegram-ботом і користувачем побудована з урахуванням принципів простоти, гнучкості та швидкодії. Однією з головних

переваг є мінімалізм взаємодії: користувач не стикається зі складними меню або формами реєстрації, а весь процес зводиться до введення текстового запиту. Такий підхід знижує бар’єр входу та робить сервіс доступним для широкої аудиторії.

Водночас бот демонструє високу гнучкість у сприйняті запитів, оскільки здатен коректно обробляти повідомлення з різною структурою та формулюваннями. Це дозволяє користувачеві взаємодіяти із системою у вільній формі, не обмежуючись суворо визначеними командами або шаблонами.

Окремо слід визначити швидкість відповіді, яка в більшості випадків не перевищує кількох секунд. Така оперативність забезпечується за рахунок ефективної реалізації логіки обробки, кешування даних і асинхронної архітектури, що дає змогу надавати результат практично миттєво після запиту.

ВИСНОВКИ

У ході виконання роботи було реалізовано Telegram-бота, який забезпечує користувачам зручний і швидкий підбір книжкової літератури відповідно до заданих параметрів. Проєкт охопив повний цикл створення програмного забезпечення — від аналізу предметної області та постановки задач до безпосередньої реалізації системи та демонстрації її функціональних можливостей.

У межах роботи було виконано низку наукових і прикладних завдань. Зокрема, розроблено чат-бота на мові програмування Python із використанням асинхронної бібліотеки aiogram. Реалізовано інтеграцію з сервісом Google Books API, що дозволило отримувати доступ до великої бази книжкових метаданих. Створено модуль попередньої обробки запитів користувача з використанням бібліотеки nltk, який забезпечує виявлення жанру, мови та інших релевантних параметрів. Для підвищення продуктивності системи реалізовано кешування результатів у локальній базі даних SQLite. Було також розроблено інтуїтивну модель взаємодії з користувачем — без складних команд або навігаційних меню. Уся логіка проєкту структурована за модулями, що значно полегшує супровід і забезпечує гнучке масштабування системи в майбутньому.

Практична цінність створеного Telegram-бота полягає у його здатності бути інтегрованим у реальні комунікаційні середовища. Зокрема, він може бути використаний у книжкових Telegram-спільнотах як засіб для рекомендації літератури, в освітніх каналах як інструмент для пошуку навчальних або художніх текстів, а також у бібліотеках як інтерфейс до електронного каталогу. Крім того, проєкт може слугувати основою для подальших комерційних або дослідницьких ініціатив, пов’язаних з інтелектуальними пошуковими сервісами.

У перспективі передбачається низка удосконалень. Серед них — покращення модуля NLP шляхом впровадження моделей машинного навчання на основі бібліотек spaCy або transformers, підтримка рекомендацій, що враховують історію запитів користувача, розширення функціоналу шляхом додавання інтерактивних

елементів, системи статистики та інтеграції з додатковими API. Також можливе створення веб-інтерфейсу або десктопної версії застосунку на основі ядра існуючого бота.

Загалом, мета роботи була досягнута. Створений Telegram-бот є прикладом ефективного використання сучасних засобів Python-розробки та зовнішніх API, демонструючи здатність поєднувати технологічні рішення з реальними запитами кінцевого користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1. Telegram. Telegram Bot API Documentation [Електронний ресурс]. – URL: https://core.telegram.org/bots/api
2. Google Developers. Google Books API Documentation [Електронний ресурс]. – URL: https://developers.google.com/books/docs/v1/using
3. Aiogram Documentation. Modern and Fully Asynchronous Telegram Bot Framework [Електронний ресурс]. – URL: https://docs.aiogram.dev/
4. Python Developer's Guide [Електронний ресурс]. – URL: https://devguide.python.org/
5. Requests: HTTP for Humans™ – Requests 2.32.3 documentation [Електронний ресурс]. – URL: https://requests.readthedocs.io/en/latest/
6. SQLite Documentation [Електронний ресурс]. – https://sqlite.org/docs.html
7. Accenture. Technology Vision 2023: When Atoms Meet Bits [Електронний ресурс]. – URL: https://www.accenture.com/us-en/insights/technology/technology-trends-2023
8. Pytest. Python testing framework documentation [Електронний ресурс]. – URL: https://docs.pytest.org/
9. Jurafsky D., Martin J.H. Speech and Language Processing. – 3rd ed. – Prentice Hall, 2020. – 1024 с – URL: https://web.stanford.edu/~jurafsky/slp3/
10. 2023, International Journal of Advanced Research in Arts, Science, Engineering & Management – URL: https://www.academia.edu/128276801/Implementation_of_Chatbot_in_the_Field_of_Education
11. Топ-5 кращих IDE для Python розробки у 2024 [Електронний ресурс]. – URL: https://university.sigma.software/best-python-ide-and-code-editors/
12. Git, Gitlab та GitHub: Особливості систем контролю версій та їх відмінності [Електронний ресурс]. – URL: https://pnn.com.ua/ua/blog/detail/git-gitlab-and-

github-difference-and-peculiarities-of-version-control-systems

13. Порівняння СУБД MySQL, POSTGRESQL та MS SQL SERVER [Електронний ресурс]. – URL: https://data-b-i.com/uk/article/porivnyannya-subd-mysql-postgresql-mssqlserver.html
14. Чат-боти: Плюсі і мінуси [Електронний ресурс]. – URL: https://ukr-bot.com/plyusi-ta-minusi-chat-botiv/
15. Вступ до NLP. Як розробити діалогову систему [Електронний ресурс]. – URL: https://dou.ua/lenta/columns/introduction-to-nlp/
16. Що таке UNIT тести і як їх писати [Електронний ресурс]. – URL: https://foxminded.ua/yunit-testy/
17. Що таке парсинг Python і де його використовують [Електронний ресурс]. – URL: https://foxminded.ua/parsynh-python/
18. Яку мову програмування обрати для створення чат-бота: Python чи JavaScript [Електронний ресурс]. – URL: https://it-club.com.ua/which-language-to-choose-to-create-a-telegram-bot/
19. Як створити телеграм-бота на Python [Електронний ресурс]. – URL: https://spacelab.ua/articles/yak-stvoriti-telegram-bota-na-python/
20. Повне вивчення Python: Поради та трюки для просунутих програмістів [Електронний ресурс]. – URL: https://itproger.com/ua/news/polnoe-izuchenie-python-soveti-i-tryuki-dlya-prodvinutih-programmistov

ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

Main.py

```
import asyncio
import logging
from aiogram import Bot, Dispatcher
from config import TELEGRAM_TOKEN
from handlers.message_handler import register_message_handlers
from db.init_db import init_db

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

async def main():
    logger.info("Запуск бота...")
    init_db()

    bot = Bot(token=TELEGRAM_TOKEN)
    dp = Dispatcher()

    register_message_handlers(dp)
    await dp.start_polling(bot)

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

api/google_books.py

```
import requests
from config import GOOGLE_BOOKS_API_KEY
from nltk.corpus import stopwords
import re

def is_garbage(word):
    return len(word) > 20 and not re.search(r"[\s\d]", word)

def fetch_books(keywords):
    if not keywords:
```

```
        return {"error": "empty"}

    language = None
    genre = None
    rest = []

    for word in keywords:
        w = word.lower()
        if w in LANGUAGE_MAP:
            language = LANGUAGE_MAP[w]
        elif w in GENRE_MAP:
            genre = GENRE_MAP[w]
        elif w not in UA_STOPWORDS:
            rest.append(w)

    if not genre and (not rest or all(is_garbage(w) for w in rest)):
        return {"error": "unrecognized"}

    if genre:
        query = f"subject:{genre}"
    else:
        query = "+".join(rest)

    order = "newest" if genre else "relevance"

    params = {
        "q": query,
        "orderBy": order,
        "maxResults": 40,
        "key": GOOGLE_BOOKS_API_KEY
    }
    if language:
        params["langRestrict"] = language

    url = "https://www.googleapis.com/books/v1/volumes"
    response = requests.get(url, params=params)
    if response.status_code != 200:
        return {"items": []}
```

db/cache_manager.py

```
import sqlite3
from config import DB_PATH

def get_cached_response(query):
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    cursor.execute("SELECT response FROM cache WHERE query = ?", (query,))
    row = cursor.fetchone()
    conn.close()
    return row[0] if row else None

def save_cached_response(query, response):
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    cursor.execute("INSERT OR REPLACE INTO cache (query, response) VALUES (?,
?)", (query, response))
    conn.commit()
    conn.close()
```

config.py

```
import os
from dotenv import load_dotenv
load_dotenv()
TELEGRAM_TOKEN = os.getenv("TELEGRAM_TOKEN")
GOOGLE_BOOKS_API_KEY = os.getenv("GOOGLE_BOOKS_API_KEY")
DB_PATH = os.getenv("DB_PATH", default="book_bot_cache.db")
```

db/init_db.py

```
import sqlite3
from config import DB_PATH

def init_db():
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS cache (
            query TEXT PRIMARY KEY,
            response TEXT
        )
    ''')
    conn.commit()
```

```
    conn.close()
```

handlers/message_handler.py

```
from aiogram import Router, types, F
from aiogram.types import Message
from nlp.text_processor import extract_keywords
from api.google_books import fetch_books
from utils.response_formatter import format_books_response
from db.cache_manager import get_cached_response, save_cached_response

router = Router()

@router.message(F.text.startswith("/start"))
async def handle_start(message: Message):
    guide = (
        "Привіт! 👋\n"
        "Я бот для підбору книжок за твоїм запитом.\n\n"
        "📌 Просто напиши мені, що саме ти хочеш прочитати. Наприклад:\n"
        "- Порадь мотиваційну книгу\n"
        "- Щось українське історичне\n"
        "- Книга про бізнес або саморозвиток\n\n"
        "🔄 Я оброблю твій запит, звернуся до Google Books і надам добірку книжок.\n"
        "⚡ Якщо запит уже був — отримаєш швидку відповідь з кешу.\n\n"
        "Почни просто зараз! 📚"
    )
    await message.answer(guide)

@router.message()
async def handle_user_message(message: Message):
    user_query = message.text.strip().lower()

    if user_query.startswith("/start") or user_query.startswith("/help"):
        return

    cached = get_cached_response(user_query)
    if cached:
        await message.answer(cached, parse_mode="Markdown")
        return
```

```
    keywords = extract_keywords(user_query)
    books_data = fetch_books(keywords)
    response_text = format_books_response(books_data)
    save_cached_response(user_query, response_text)

    await message.answer(response_text, parse_mode="Markdown")

def register_message_handlers(dp):
    dp.include_router(router)
```

ДОДАТОК Б. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерних наук

ДИПЛОМНА РОБОТА
на ступінь вищої освіти бакалавр
із спеціальності 122 Комп'ютерні науки
Чат-бот Telegram на мові програмування Python для підбору книг

Виконав: студент 4 курсу, групи КНД-43
Колядюк Віталій Вікторович
Керівник: професор кафедри Комп'ютерних наук
д.ф.-м.н., професор Шикула О.М.

Київ - 2025

1

Актуальність роботи

На сьогодні Telegram є одним із найпопулярніших месенджерів, який активно використовується не лише для спілкування, але й для реалізації різних сервісів через чат-ботів. Попри це, у сфері пошуку літератури спостерігається відсутність якісних Telegram-рішень, зокрема таких, що враховують мову, жанрові вподобання чи емоційні запити користувача.

Google Books API надає доступ до великої кількості книжкових джерел, але ця інформація залишається недоступною безпосередньо в Telegram. Тому створення інтелектуального бота, який поєднує зручність месенджера з можливостями зовнішніх API є актуальним і відповідає сучасним інформаційним потребам користувачів.

2

Мета, предмет та об’єкт дослідження

Об’єкт дослідження: Чат-боти в месенджерах як інструмент автоматизованого інформаційного пошуку

Мета роботи: Розробити Telegram-бота для підбору літератури з використанням Google Books API та технологій обробки природньої мови.

Наукове завдання: Реалізувати персоналізований підбір книг у Telegram із застосуванням зовнішніх API та NLP-засобів на мові Python.

Методи дослідження:

- Аналіз предметної області та існуючих рішень;
- структурне проєктування;
- Розробка й тестування програмного забезпечення
- Оцінка ефективності взаємодії з користувачем

Чат-боти як інструмент

Чат-боти активно використовуються в освіті, бізнесі, техпідтримці.

Telegram став популярною платформою завдяки відкритому API, гнучкості та високій продуктивності.

Перспективи – інтеграція з AI, персоналізація, автоматизація пошуку.

Пошук книжок у Telegram

Типові запити – жанр, автор, настрій, новинки.

Запити часто неструктуровані або емоційно забарвлені.

Існує потреба в обробці природньої мови (NLP).

Аналоги та обґрунтування нового рішення

Більшість ботів не підтримують персоналізацію або фільтрацію.

Результати часто неетичні – прямі PDF-посилання на літературу без дотримання авторських прав.

Висновок: Потрібен новий бот із інтеграцією до великої бази даних літератури та етичною подачею.

Вибір мови програмування

Обрано Python 3.11 – оптимальний для швидкої розробки та підтримки.

Переваги: простий синтаксис, велика кількість бібліотек, активна спільнота.

Підтримує асинхронність(важливо для одночасної обробки запитів)

Критерій	Python	JavaScript	Java	Go	PHP
Простота синтаксису	Висока	Середня	Низька	Середня	Низька
Швидкість розробки	Висока	Висока	Низька	Середня	Низька
Підтримка Telegram API	Повна	Повна	Часткова	Часткова	Обмежена
Наявність бібліотек для NLP	Розвинена	Обмежена	Обмежена	Практично відсутня	Відсутня
Інтеграція з API	Зручна	Задовільна	Складна	Обмежена	Складна
Паралельна обробка запитів	Добра	Добра	Складна	Вбудована	Обмежена
Документація та навчальні ресурси	Дуже багато	Дуже багато	Багато	Обмежено	Обмежено
Придатність до створення чат-ботів	Висока	Висока	Низька	Середня	Низька

Бібліотеки та інструменти

Telegram API: Бібліотека *aiogram* (асинхронна, FSM, модульність).

Google Books API: *requests, json* – для роботи з HTTP-запитами.

NLP: *nltk* – для аналізу тексту, токенізації, виділення параметрів.

База даних: *sqlite3* – для кешування запитів та історії.

Розгортання і тестування

Railway – хмарна платформа для цілодобової роботи бота.

Git + GitHub – контроль версій, автоматичний деплой.

Середовище розробки	Переваги	Недоліки
VS Code	Легка вага, розширення, інтеграція з Git, підтримка Python, безкоштовне	Потрібна ручна установка деяких модулів
PyCharm	Потужний IDE з глибокою інтеграцією Python	Важчий, деякі функції доступні лише у платній версії
Sublime Text	Швидкість, мінімалізм	Слабша інтеграція з Python і Git, платний
JupyterLab	Ідеальний для data science	Менш зручний для розробки повноцінних Telegram-ботів

Система	Переваги	Недоліки
Git + GitHub	Популярність, гнучкість, інтеграції, підтримка IDE	Потребує базового вивчення команд
Git + GitLab	Подібна функціональність, приватні репозиторії	Менша популярність у Python-спільноті
Mercurial (Bitbucket)	Простий синтаксис	Менш гнучка система гілок і злиття
Dropbox / Google Drive	Простота для новачків	Відсутність контролю версій, злиття, історії

Pytest – для перевірки модулів (NLP, кеш, API).

Обґрунтування вибору стеку

Висока швидкість розробки та масштабованість.

Простота інтеграції з API, Telegram та NLP-бібліотеками.

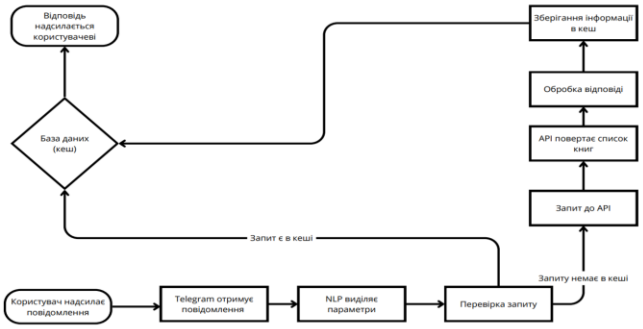
Низький поріг входу + підтримка української мови.

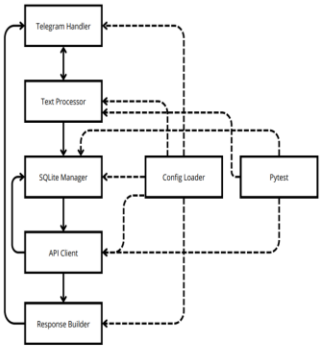
Компонент	Обраний інструмент	Обґрунтування вибору
Мова програмування	Python	Простий синтаксис, гнучкість, велика кількість бібліотек для NLP, API, тестування.
Telegram API	aiogram	Асинхронна модель, FSM, масштабованість, розвинена документація.
API-клієнт	requests	Простота формування запитів до Google Books API, контроль параметрів.
NLP	nltk	Мінімальні вимоги, швидкий старт, базова обробка запитів користувача.
Сховище даних	SQLite	Простота використання, відсутність серверної частини, збереження кешу та історії.
Середовище розробки	VS Code	Легка вага, підтримка Python, інтеграція з Git, розширення для SQLite, JSON, API.
Контроль версій	Git + GitHub	Гнучкий контроль історії змін, підтримка деплою через GitHub, можливість CI/CD.
Тестування	pytest	Гнучка структура тестів, проста конфігурація.
Розгортання	Railway	Автоматичне деплоювання, підтримка Python, безкоштовний тариф для невеликих проєктів.
Допоміжні інструменти	Postman, SQLiteStudio	Зручна перевірка API-запитів, візуальний аналіз БД.

Архітектура Telegram-бота

Модульна структура: окремо – текст, кеш, API, відповідь.

Telegram <-> Text Processor -> (cache або API) -> Response -> Користувач





Взаємодія через асинхронні функції (*async def*).

Взаємодія з Google Books API

Запит формується на основі ключових слів (жанр, автор).

Використовується бібліотека requests та API-ключ.

Отримані дані парсяться й перетворюються у зручний формат.

Кешування та база даних

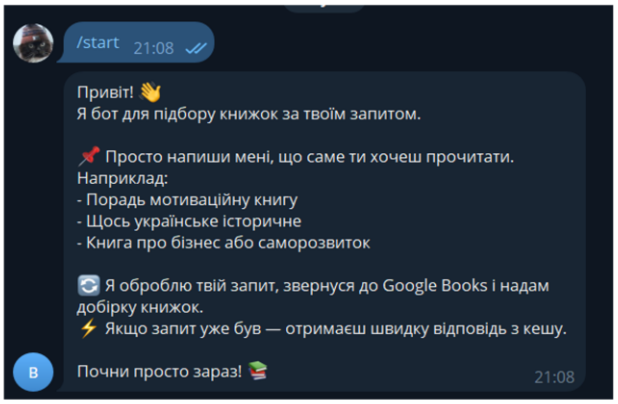
База даних: SQLite.

Зберігається історія запитів, відповіді, автори.

Кеш дає змогу зменшити кількість запитів до API.

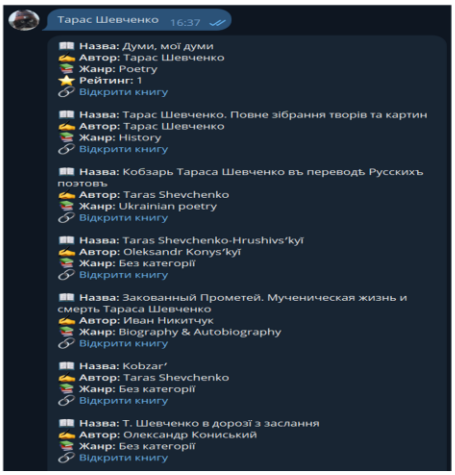
Інтерфейс та функціональність

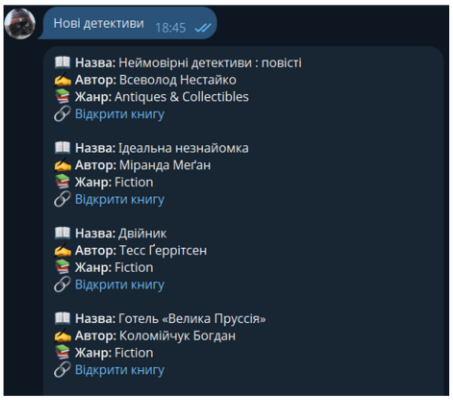
Мінімалістичний текстовий інтерфейс – без команд і меню.



Інтерфейс та функціональність

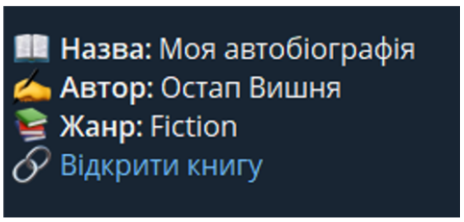
Бот підтримує запити за жанром, автором, настроєм.





Інтерфейс та функціональність

Відповідь містить назву, автора, жанр та посилання на літературу.



Висновки

1. Реалізовано Telegram-бота для пошуку книг із використанням Google Books API.
2. Забезпечено обробку запитів, кешування та простий інтерфейс.
3. Використано Python, aiogram, SQLite, nltk.
4. Система легко масштабована й адаптована під персоналізовані рекомендації.
5. Проєкт має практичну цінність і потенціал для подальшого розвитку