Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ДОСЛІДЖЕННЯ МЕТОДІВ РОЗРОБКИ ЧАТ-БОТІВ

Текстова частина до курсової роботи

за спеціальністю «Комп'ютерні науки» 122

Керівник курсової роботи

ст.в. Салата К. В.

_____

(підпис)

Виконала студентка 3 курсу

Сидорова Є.О.

«____»_____2023 р.

Київ 2023

Зміст

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри інформатики,

к.ф.-м.н., доц. Гороховський С.С

_____

(підпис)

«___» _____ 2023 р

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентці 3-го курсу, факультету інформатики Сидоровій Єлізавєті

Тема: Дослідження методів розробки чат-ботів

Вихідні дані:

Зміст ТЧ до курсової роботи:

Зміст

Вступ

1.Загальна інформація про чат-боти

2.Дослідження

3.Реалізація на основі дослідження

Висновок

Список використаної літератури

Дата видачі «____»_____ 2023 р. Керівник _____

(підпис)

Завдання отримала _____

(підпис)

Календарний план виконання курсової роботи

Тема: Дослідження методів розробки чат-ботів:

№	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	11.11.2022	
2.	Аналіз матеріалів за темою курсової роботи	25.12.2022	
3.	Розробка та реалізація алгоритму створення чат-боту	08.01.2023	
4.	Написання текстової частини	15.02.2023	
5.	Надання роботи на перевірку керівником	12.04.2023	
6.	Коригування на основі результатів перевірки	01.05.2023	
7.	Остаточне оформлення роботи та презентації	09.05.2023	
8.	Захист курсової роботи	23.05.2023	

Сидорова Є.О. _____

Салата К.В. _____

«____» _____ _____р.

Вступ

Зі стрімкою світовою діджиталізацією людство стикається з постійними оновленнями, зокрема у галузі ІТ. Сфера послуг, пошук інформації, навчальні процеси, спілкування та навіть транспорт переходить на цифрове керування. Саме через такі оновлення суспільству необхідно навчатись комунікації з більшістю новоутворень, які спрощують життя. Кожного дня ми стикаємось з безліччю технологій, пов’язаних з онлайн-ресурсами: веб-сторінки, розмови по відео зв’язку, соцмережі, онлайн-банкінг, шопінг або перегляд відео. А з часом таких послуг буде в рази більше.

Відомі компанії будують інтуїтивно зрозумілі інтерфейси, надають документації та поради для всіх клієнтів, щоб процес використання цифрових помічників ставав якомога простішим. Однією з головних переваг їх використання, у тому числі й чат-ботів, є відсутність необхідності у використанні додаткового програмного забезпечення. Широкий спектр функціональності та технологій при створенні таких застосунків скорочує час як розробникам, так і клієнтам.

Кожен з нас може спробувати себе на практиці тестувальника засобів обробки природньої мови, штучного інтелекту, нейронних мереж та машинного навчання. Тим більше й ознайомитись з історією їх створення, розбором їх можливостей і застосуванню своїх вмінь під час їх імплементації у власні застосунки.

Основною метою цього дослідження є вивчення теоретичних відомостей про чат-ботів, практичне застосування набутих знань та аналіз усіх наданих матеріалів. Об’єктом розробки курсової роботи є безпосередньо чат-бот для спрощення пошуку інформації в Інтернеті.

1. Загальна інформація про чат-боти

1.1. Поняття чат-боту

Чат-бот – комп'ютерна програмна система, яка обробляє запити користувача та дає персоналізовану відповідь на них. Зазвичай складність вимог споживача визначає швидкість та наявність відповіді програми. Чат-бот, який розроблено у великих компаніях, після задання занадто важкого запиту може перенаправити вас до іншого помічника-асистента [1].

Прикладами взаємодії таких програм за допомогою команд користувача можуть бути отримання потрібної пошукової інформації (Forbes у Telegram), зв'язок з банківськими системами (Ощадбанк у Viber), замовлення сервісів доставки (Ресторан Жовте Море у Telegram) та інші.

1.2. Історія створення чат-ботів

Найпершим у світі чат-ботом вважається "ELIZA", який був розроблений у 1960-тих роках Джозефом Вайзенбаумом з технологічного інституту Массачусетса. Головною метою створення цієї програми було імітування звичайної людської розмови на основі прописаних можливих сценаріїв для відповідей. Із популяризацією винаходу люди висловлювали свої враження, спілкуючись з програмою, як із психотерапевтом. [2]

Сьогодні можна спробувати роботу цього чат-боту на практиці за допомогою веб-застосунків. Одним із прикладів наведений на рисунку 1.1.

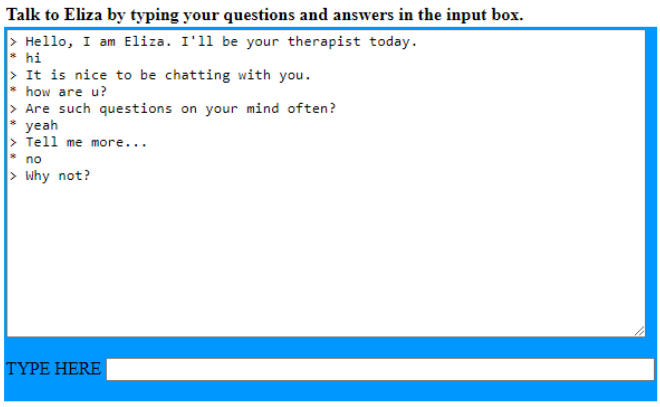

Рисунок 1.1 – Чат-бот "ELIZA" сьогодні.

Джерело: [5]

Кеннет Марк Колбі в 1972 році створив чат-бот “PARRY” на основі своїх психіатричних досягнень. Програмна система імітувала психічнохворого із діагнозом шизофренії, а також перевірялась за допомогою тесту Тюрінга. [2]

Сучасні чат-боти, звісно ж, перемагають своїх попередників новітніми технологіями, спрощеннями в розробці та постійними вдосконаленням. Інтелектуальні помічники Siri, Google Assistant, Cortana й інші швидко вирішують низку завдань за допомогою голосової обробки на пристроях.

1.3. Типізація чат-ботів

Типізація чат-ботів може бути створена на основі різних критеріїв: методи розробки, платформи для роботи, способи взаємодії з користувачем тощо. Загально цей взаємозв’язок можна представити таким чином:

1. Одним із найбільш розповсюджених типів у використанні в повсякденному житті є *голосові боти*. Звертаючись до асистентів цього виду, ми відтворюємо зв’язок «мовлення в текст» і «текст у мовлення». Штучний інтелект із розумінням природньої мови ідентифікує ключові слова та видає правильну команду із супроводом аудіо відповіді помічника. [3]
2. *Гібридні чат-боти* створені для автоматизованої персоналізованої відповіді на запит користувача, проте занадто складні питання будуть вирішуватися агентом зі служб підтримки. Представники компанії в будь-який момент можуть втрутитися до «проблемної задачі», що поставлена програмі в певний час. [3]

   Гібридні чат-боти можуть бути створені на різних типах взаємодії з агентом: веб-чат, обмін повідомленнями та обмін повідомленнями в чаті. Перший тип характеризується співпрацею відвідувачів сайту, які мають питання, через онлайн-сервіс із включенням в режимі реального часу. Другий тип включає в себе месенджери для обміну у «швидкому асинхронному» темпі, а третій – програму для кооперації компанії та клієнтів у додатку. [4]
3. *Чат-боти на основі меню* розроблені за принципом фіксованого дерева

рішень, яке демонструється користувачеві у вигляді карти вибору з кнопок. Обираючи придатні варіанти, врешті-решт людина отримує остаточне рішення від програми. [3]

Гарним прикладом такого чат-боту є KMAScheduler у Telegram. Принцип його роботи ґрунтується на записі до потрібних дисциплін студентів за допомогою кнопок. Рисунок 1.2 та 1.3 показують взаємодію з ним на практиці.

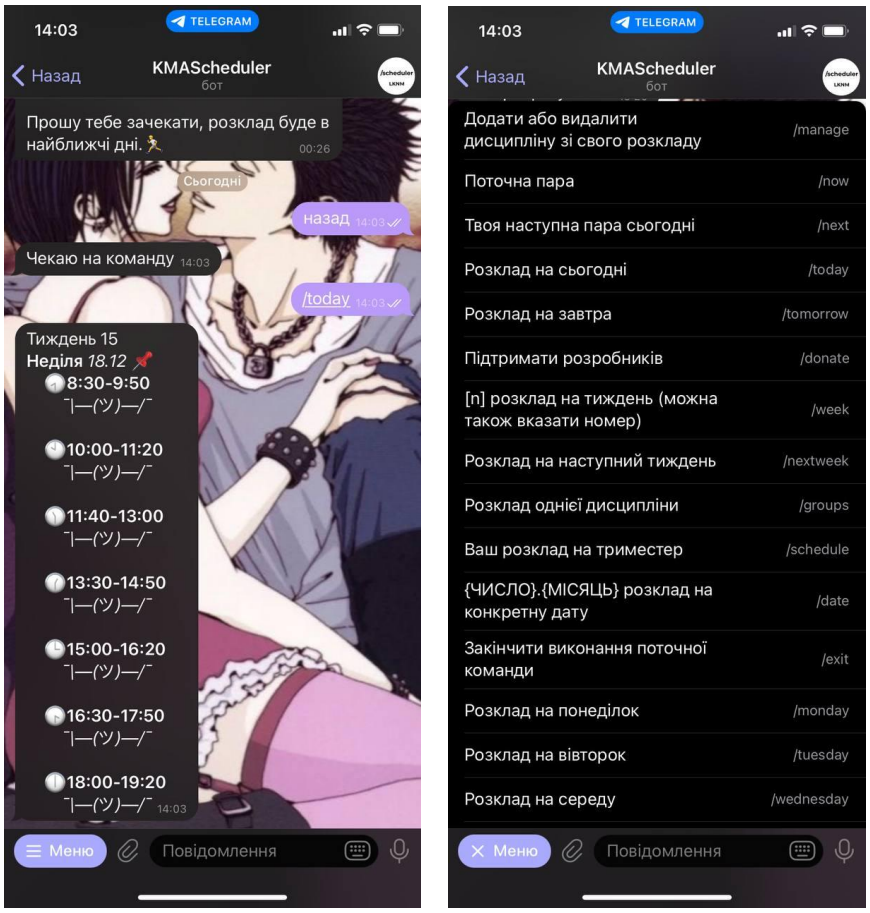

Рисунок 1.1-1.2 – Чат-бот на основі меню KMAScheduler у Telegram

4. *Чат-бот на основі навичок* компонує в собі певний набір завдань та способи його розширення на основі вже створеного програмного забезпечення. За допомогою можливості підключення фреймворків такого типу до систем розумного будинку користувачі можуть керувати своїми приладами на основі вже визначених команд.

Чат-бот на основі правил, створений за логікою конструкцій if/then, так само

виконує запити клієнтів за визначеними параметрами розробників. Насамперед, слід зауважити про повне передбачення всіх можливих конструкцій для запиту, адже, програма має зрозуміти чітке завдання користувача. [3]

5. *Чат-бот на основі ключових слів* працюють на знаходженні слів-тригерів у запиті та NLP. Проте технології фатичного діалогу не є досконалими, адже не всі запити можуть бути зрозумілими чи правильно виконаними в результаті їх обробки. У такому разі доречним є комбінування, наприклад, з чат-ботом на основі меню, де користувач зможе сам зможе обрати чітку команду в разі неправильного виконання. [3]
6. *Контекстний чат-бот на основі штучного інтелекту* відрізняється запам'ятовуванням інформації про користувача, з яким він взаємодіє, та його попередніх дій. Підключення до бази даних клієнтів допомагає утримувати, оновлювати та додавати реляції, а основі таких запитів ШІ забезпечує коректність контексту розмови, використовуючи крос-платформний пошук. [3]
7. *Чат-боти підтримки* характеризуються однією загальною ціллю – допомогти клієнту. Вони є гарним прикладом у використанні системи самообслуговування чи онлайн-навчанні для розуміння інтерфейсу. *Трансакційні чат-боти* зосередженні на простоті та оптимізації переказу коштів без особливого втручання користувача на основі швидкого каналу та вузького кола доступних команд. [3]

1.4. Переваги використання чат-ботів

З переходом до ери новітніх технологій бізнес-індустрія намагається зробити все можливе для спрощення у наданні послуг своїм клієнтам та центру підтримки їх користувачів. Чат-боти через свою технологічність, зручність у використанні та широкий спектр функціональності є гарними помічниками у сфері обслуговування.

По-перше, слід зазначити про цілодобову доступність. Оскільки оператори

не завжди мають змогу надавати послуги 24/7, чат-боти є універсальними помічниками для підтримки клієнтів. [6]

Особливості часового поясу для компаній, які працюють на міжнародному рівні, мають бути передбачені задля більшої аудиторії клієнтів. До того ж, багатомовність є гарним доповненням для такого бізнесу, тому чат-боти з таким функціоналом є універсальними для комунікації з іноземними користувачами.

Згідно з дослідженням Tidio [7] 62 % споживачів надають перевагу чат-ботам, аніж очікуванню допомоги консультанта. Також приблизно 53% респондентів вважають довгу відсутність вільних операторів значним мінусом у наданні послуг компаніями.

По-друге, багатопоточний підхід надає можливість обслуговувати одночасно кількох клієнтів. Це дозволяє звільнити операторів для вирішення більш пріоритетних завдань, які чат-боти не в змозі опрацювати. Надання швидкого персоналізованого підходу сприяє зручності в обробці легких завдань: відстеження за номером, пошук інформації про товар, безпосереднє замовлення тощо. Більш того, людський фактор не притаманний програмному забезпеченню, тобто зміна настрою консультанта чи неякісне обслуговування під час розмови з оператором неможливе під час комунікації з ботом. [6]

За опитування American Express [8] під час виникання якихось проблем близько 23 % людей шукають особистої офлайн-взаємодії з компанією, а також 40% обирають розмову по телефону з консультантом. Чат-боти у свою чергу звільняють персонал від такого типу завдань, надаючи клієнтам можливості вирішення проблеми в більш зручний для них спосіб.

По-третє, збирання та надання інформації є важливим, як для компанії, так і для користувачів. Підключення чат-ботів до баз даних сприяє створенню звітів та аналітики, що у свою чергу допомагає якомога краще зрозуміти процес роботи компанії. [6]

За результатами дослідження Tidio [7] 74% власників бізнес індустрії задоволені роботою чат-ботів, де приблизно 1 з 10 клієнтів не буде задоволений

комунікацією з чат-ботом.

Отже, чат-боти вміло пораються з поставленою задачею для спрощення операцій у сфері обслуговування та позитивно впливають на розвиток бізнес-індустрії. За даними Tidio [7] 62% компанії планують інтегрувати собі таку можливість у майбутньому, до того ж згідно з дослідженням Statista [9] до 2027 року ринок чат-ботів оцінюватиметься у 454,8 млн доларів.

2. Дослідження

2.1. Методи розробки

В основі створення чат-ботів закладені три важливі технології: штучний інтелект, обробка природної мови та машинне навчання. [10]

Машинне навчання допомагає в узгодженні вхідних даних запиту, контекстного аналізу та прогнозуванні. Алгоритми чат-ботів з використанням ШІ використовують дані фактичних відповідей користувачів задля вдосконалення системи власної роботи. Штучний інтелект дозволяє виконувати завдання, які зазвичай виконує людина, проте NLP є ключовою складовою таких можливостей. [12]

Обробка природної мови – комп'ютерна технологія для обробки людських мов у корисний спосіб. Існує два типи двигунів NLP: хмарні та внутрішні. Перший характеризується постійним оновленням даних, простотою та доступною ціною. Другий тип відрізняється підвищеним рівнем конфіденційності, в особливості для фінансових компаній створюють власний NLP для підтримання цілісності даних. Від правильного вибору двигуна залежить успіх роботи чат-боту.

В основі архітектури NLP займають місце класифікатор намірів та екстрактор сутностей. Класифікатор розподіляє дані за допомогою зіставлення передбачених шаблонів, алгоритмів машинного навчання у багатокласовому поділі або нейронних мереж. Останні два передбачають векторні просторові моделі даних для їх числового представлення. Екстрактор сутностей обирає лише ключову інформацію із запиту для його вирішення. Наприклад, особисті дані клієнту чи тип проблеми, з яким він стикнувся. [11]

Розглянемо детальніше класифікацію архітектури створення чат-ботів. Стандартною структурою зіставлення шаблонів є мова розмітки ШІ, яка називається «AIML». Чат-боти такого типу використовують базу знань, що характеризується взаємозв'язком <pattern> та <template>. Відбувається класифікація запиту, під час якої слова чи речення підпадають до певних тегів визначених категорій. [13]

У результаті генерується кінцева відповідь програми, рисунок 2.1:

```
#PATTERN MATCHING
<category>
    <pattern>What is the capital of \*</pattern>

    <template>
        <srai>The capital of <star/></srai>
    </template>

</category>
-----------------------------------------------
#PRE_STORED PATTERNS
<category>
    <pattern>The capital of the United States of America?</pattern>
    <template>The capital of the United States of America is Washington, D.C.</template>
</category>
<category>
    <pattern>The capital of India?</pattern>
    <template>The capital of India is New Delhi.</template>
</category>
```

Рисунок 2.1 – Зіставлення шаблонів

Джерело: [13]

Алгоритми у свою чергу допомагають аналізувати великі набори даних через ієрархічну структуру, що спрощує роботу з шаблонізацією. «Редукціоністський» підхід характеризується зменшенням проблеми для спрощення рішення. Мультиноміальний наївний алгоритм Байєса є класичним алгоритмом для класифікації текстових даних та NLP, за формулою якого здійснюється підрахунок балів на спільність та приналежність окремих слів до визначених класів. Таким чином за результатом підрахунків найбільш ймовірним класом буде той, у кого найбільший рейтинговий бал. [14]

Прикладом роботи є рисунки 2.2 та 2.3:

```
class: weather
    "is it nice outside?"
    "how is it outside?"
    "is the weather nice?"

class: greeting
    "how are you?"
    "hello there"
    "how is it going?"
```

```
input: "Hi there"
 term: "hi" (no matches)
 term: "there" (class: greeting)
 classification: greeting (score=1)

input: "What's it like outside?"
 term: "it" (class: weather (2), greeting)
 term: "outside (class: weather (2) )
 classification: weather (score=4)
```

Рисунки 2.2-2.3 – Класифікація вхідного тексту

Джерело: [14]

Тут слід зазначити й про компоненти самої NLP: розуміння та генерування природної мови. NLU перетворює неструктуровані дані у структурне представлення задля їх оптимальної обробки. Цей процес починається з лексичного аналізу: методом поділу тексту від фрагментів до слів програма переходить синтаксичного розбору. Перевірка граматичних аспектів розпізнає контекст вхідних даних, а семантичний аналіз передає їхню змістовність. Інтеграція дискурсу та прагматичний аналіз створюють фінальну стадію обробки та інтерпретують початкове повідомлення. NLG через планування тексту у лінгвістично правильній змістовній формі генерує відповідь на запит користувача. [10]

Нейронні мережі складаються з взаємопов’язаних вузлів, які компонуються в елементи для додаткової обробки інформації за допомогою динамічних реакції станів на запит. За допомогою синапсів, що створилися на основі повторюваних ітерацій, кінцевий результат видає дані з найбільш низьким рівнем помилок. Класифікація подібна до робити алгоритмів, проте повторне обчислення на всіх рівнях порівнюється між собою. Продуктивна нейронна мережа міститиме меншу кількість коду, ніж алгоритмічний аналог, проте й довшого періоду виконання. [14]

2.2. Дослідження платформ для чат-ботів

Платформи для розробки чат-ботів є безпосереднім інструментом для їх створення, а платформи публікації – для їх використання. Розглянемо детальніше представників обидвох категорій.

2.2.1 Платформи для розробки

Botpress – open-source розмовне програмне забезпечення ШІ з підтримкою NLU (OpenBook від Botpress). Створення чат-ботів відбувається за допомогою візуальних потоків та невеликої кількості даних, розділених на сутності. Наявність візуального контролю бесід, емулятору перевірки розмови та широка інтеграція на платформи публікації дозволяє керувати системою на належному

високому рівні.

Microsoft Bot Framework базується на кодуванні, надає деталізований контроль над процесом створення чат-ботів та має широкий спектр вибору інструментів. NLU механізм є не найзручнішим, оскільки Luis є окремим програмний забезпеченням. *Botkit* є частиною інструментів Microsoft Bot Framework з більшою увагою до інтерфейсу, який також надає доступ до різних чат-платформ.

BotMan – безкоштовний фреймворк на основі PHP, який є найпопулярнішим за своїм типом. Проектуючи логіку чат-боту, BotMan дозволяє підключити його одразу на багато платформ для публікацій, не залежить від фреймворку та має потужний синтаксис. [15]

2.2.2 Платформи для використання

У 2021 році українці пройшли опитування про найбільш використовувані месенджери. Лідерами респондентів стали Viber(73,6%), Facebook(42,7%) та Telegram (31,6%). [16]

Було вирішено провести опитування серед 62 респондентів щодо використання цих месенджерів загалом та роботою чат-ботів у цих додатках. Під час дослідження брали участь люди різного віку, проте більшості з них від 18 років до 21 (91,9%).

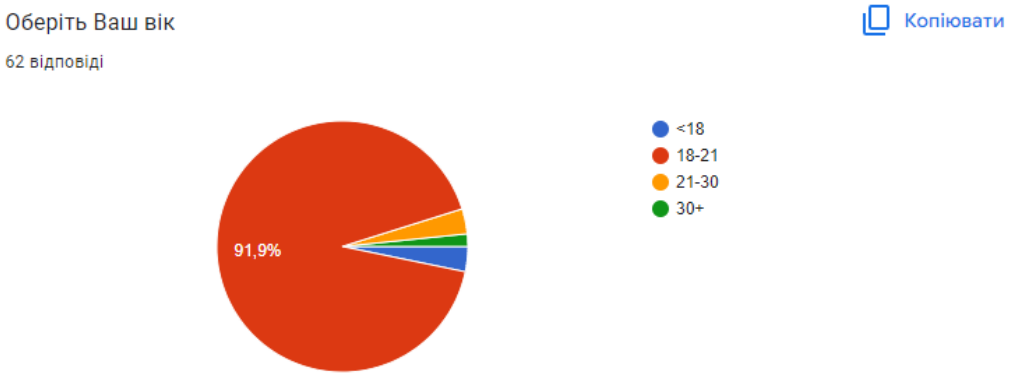

Рисунок 2.4 – Результати опитування(вік респондентів)

Слід зазначити, що лідером серед трьох відомих месенджерів виявився саме

Telegram. Більш того, найкраще оцінюють свій досвід у використанні цих додатків також під час роботи з Telegram.

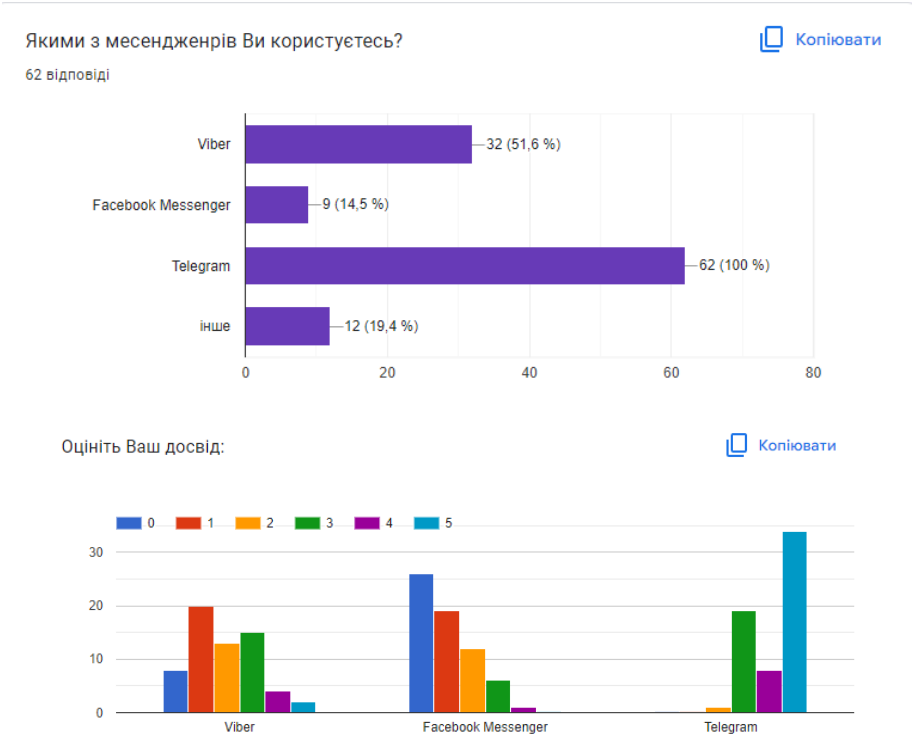

Рисунок 2.5-2.6 – Результати опитування(топ месенджерів)

Щодо використання чат-ботів: 41 людина використовує чат-боти, з яких 95,8% надає перевагу так само Telegram.

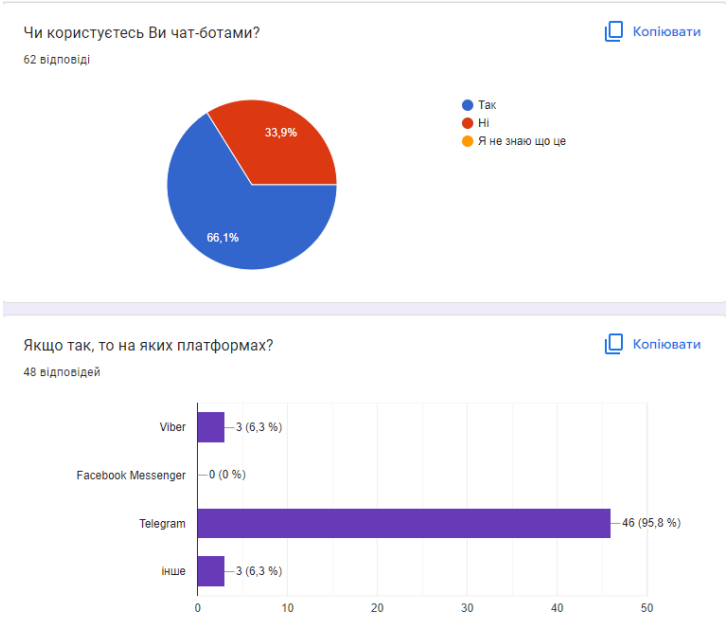

Рисунок 2.7 – Результати опитування(використання чат-ботів)

Отже, судячи з проведеного дослідження, соціальна мережа Telegram є актуальною серед молоді та є найпопулярнішою та найбільш зручною для використання чат-ботів.

2.2.3 Актуальність Telegram

До переваг месенджеру Telegram можна віднести багато функцій: доступність на різних оперативних системах, реакції, статичні та нестатичні стікери, самознищення чатів і повідомлень, великий розмір файлів для відправки. Цікавим доповненням є боти Telegram, які допомагають користувачам легше та простіше створювати власних чат-ботів.

BotFather – офіційний чат-бот від Telegram, який є ключовим помічником при написанні більшості чат-ботів. З його допомогою можна керувати всіма застосунками, створеними через обліковий запис їх власника. Як і будь-який чат-бот, BotFather має перелік власних команд: для початку використання – /start, для виводу всіх функцій – /help, тощо. Створення власного застосунку відбувається шляхом надання унікального посилання на бота та створенням секретного токену, який потрібен для підключення під час написання коду. Також BotFather допомагає для налаштування опису й зображення профілю, списку команд та їх редагування. [17]

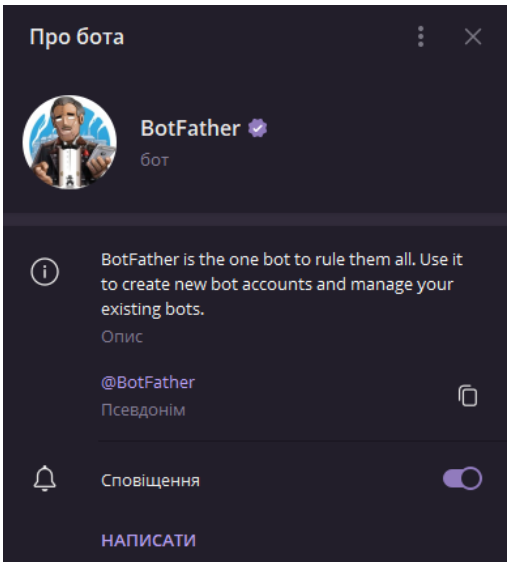

Рисунок 2.8 – Чат-бот BotFather у Telegram

3. Реалізація на основі дослідження

3.1. Опис майбутнього чат-боту

Опис майбутнього чат-боту: програма, яка допомагає користувачеві знати певну інформацію та ілюстрації за допомогою його запиту. Тематикою застосунку є аніме та його персонажі.

Вхідні параметри початково базуються на назві анімації, а надалі передаються за допомогою кнопок з меню чат-боту. Програма надає користувачеві можливість дізнатися основну інформацію, опис, рейтинг та пошук ілюстрацій персонажів, які фігурують у зазначеному запиті. У відповідях програми закладений принцип фатичного діалогу та обробки природної мови користувача.

Перевагою чат-боту є його приналежність до месенджеру, який є доступний на більшості з оперативних систем.

3.2. Вибір засобів для розробки

Мова програмування високого рівня Python відрізняється з-поміж інших динамічною семантикою, простим синтаксисом, підвищеною продуктивністю та широким вибором бібліотек для імпортування. [18]

Середовище розробки PyCharm – спеціальна кросплатформна IDE для роботи з Python, що надає доступ до багатьох інструментів під час написання коду. Він є дуже зручним для перевірки синтаксису, автоматичним доповненням коду та розумними пропозиціями на основі аналізу дій користувача. [19]

Платформа для використання чат-боту – Telegram. За вище наведеними дослідженнями даного месенджеру можна стверджувати про його актуальність, популяризацію та широкий спектр функціоналу.

Офіційний чат-бот Telegram – BotFather, який описано в попередньому розділі, слугує зручним інструментом для створення власного бота.

Список модулів Python, які були використані в програмі:

1. random – генератор псевдовипадкових чисел на визначеному проміжку або вибір елементу із заданого переліку. [20]

2. urllib (urllib.request, urllib.error) – модулі для взаємодії з URL – адресами, а саме з роботою запитів [21], обробкою та пошуком помилок і винятків при їх роботі [22].
3. re – робота з регулярними виразами, їх синтаксисом та операціями зіставлення. [23]
4. telebot (telebot, types) – модуль для API при створенні чат-боту у Telegram, його налаштуванням та управлінням. [24]
5. textblob (TextBlob) – обробка текстових даних, включаючи базові завдання роботи з NLP. [25]
6. bs4(BeautifulSoup) – забезпечує роботу з парсингом веб-сторінок та викачуванням даних. [26]
7. bing_image_urls(bing_image_urls) – викачування та обробка картинок за доомогою Bing. [27]
8. json – модуль для роботи з файлами типу json, їх декодуванням та роботою з API. [28]

Більш детальний огляд на роботу цих модулів буде надано безпосередньо під час розбору коду.

3.3. Огляд методів та тестування

Перш за все слід розглянути створення чат-боту через @BotFather у Telegram: За допомогою команд /newbot, /setdescription та /setuserpic створюється назва майбутнього чат-боту, посилання, HTTP API токен, встановлюється опис та фото його профіля.

Отримавши токен, користувачеві надається можливість налаштовувати підключення до чат-боту за допомогою модуля telebot в файлі botToken.py:

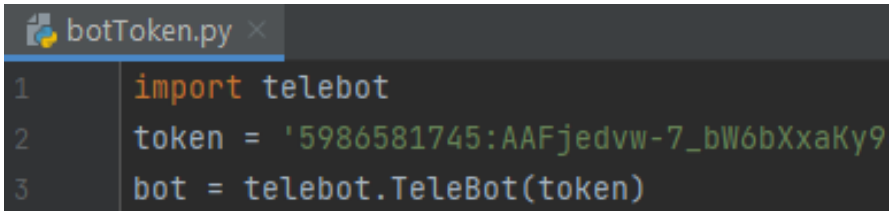

Рисунок 3.1 – Підключення чат-боту в коді

Модуль main.py відповідає за налаштування функціональності та першочергової обробки запитів користувача. Робота цього модуля працює таким чином:

1. Він зберігає всю інформацію про користувачів, які працюють з чат-ботом під час цього сеансу. Це створено для того, щоб при отриманні запиту, який стосується вже введеної назви аніме в авторизованого користувача, не потрібно було вводити її кожного разу, а також щоб керувати багатопоточністю, яка створюється одночасним використанням чат-боту в різних тестувальників. При аналогічній ситуації для неавторизованого користувача програма перепитає назву та збереже його дані в пам’яті. Якщо дізнається інформація вже за новим найменуванням, тоді дані автоматично оновлюються і нові запити будуть оброблені згідно з цим ім’ям. У коді за збереження відповідає словник user_info, де ключами є ID користувачів, а значеннями – одномірні масиви з інформацією по їх останніх заданих аніме.
2. Декоратори @bot.message_handler(), які відповідають за отримання різних вхідних даних від користувача та вказують, що буде робити наступна функція після їх отримання. @bot.message_handler(commands=['start']) налаштовує початкове кнопкове меню та логіку команди /start, а @bot.message_handler(content_types=['text']) – будь-які наступні запити та їх відповідні кнопкові меню. До того ж, bot.polling(none_stop=True) допомагає створювати нескінченний цикл отримання повідомлень від користувачів з Telegram API.

Детальніше про @bot.message_handler(commands=['start']): функція start() бере на вхід параметр message, за допомогою якого отримується вся потрібна інформація про користувача: його ID, ім’я та ID чату з ботом. Програма записує його дані в пам’ять, вітається з ним і пропонує обрати одну з 2 початкових кнопок для взаємодії.

```python
import urllib.error
from telebot import types
from analysingMessage import analyse
from fatic import talking
from botToken import bot
from checking import *
from animeInfo import choose_character, anime_info, for_request, make_request

user_info = {}


# command /start + menu bar
@bot.message_handler(commands=['start'])
def start(message):
    id_user = message.from_user.id
    if user_info.get(id_user) is None:
        user_info[id_user] = ['anime_link', 'characters', 'name_anime']
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1, one_time_keyboard=True)
    button1 = types.KeyboardButton('Let`s talk about anime')
    button2 = types.KeyboardButton('Let`s discover some arts')
    markup.add(button1, button2)
    answer = f'Welcome, *{message.from_user.first_name}* ! \n' + talking("greeting")
    bot.send_message(message.chat.id, answer, reply_markup=markup, parse_mode='Markdown')

```

Рисунок 3.2 – main.py @bot.message_handler(commands=['start'])

Розглянемо, як це працює на практиці під час процесу взаємодії з цим чат-ботом у Telegram:

Рисунок 3.3-3.4 – Тестування початкового меню та команди /start

Детальніше про @bot.message_handler(content_types=['text']): функція get_text() бере на вхід параметр message, за допомогою якого отримується вся потрібна інформація про користувача та його запит. Програма перевіряє чи закріплені за теперішнім користувачем дані певного аніме, якщо ні, то масив за його ID заповниться значеннями за замовчуванням. У залежності від того, який текст прийшов на вхід (команди з кнопок чи будь-який інший текст), функція get_text() буде викликати потрібні їй методи з інших модулів, повертати певну відповідь та будувати нові меню. Будь-який текст, який не відповідає ні одній з кнопок чат-боту сприймається, як назва нового аніме.

main.py

```
# text
@bot.message_handler(content_types=['text'])
def get_text(message):
    id_user = message.from_user.id
    if id_user not in user_info.keys():
        user_info[id_user] = ['anime_link', 'characters', 'name_anime']
        get_text(message)
    elif message.text == 'Let`s talk about anime':
        bot.send_message(message.chat.id, talking("talking"), parse_mode='html')
    elif message.text == "Let`s discover some arts" or message.text == "Arts":
        if check_anime(id_user, user_info):
            user_info[id_user][1] = choose_character(id_user, user_info)
            ch = (user_info[id_user])[1]
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=3, one_time_keyboard=True)
            b = types.KeyboardButton('Back')
            for c in ch:
                button = types.KeyboardButton(c)
                markup.add(button)
            markup.add(b)
            bot.send_message(message.chat.id, talking("characters"), reply_markup=markup)
        else:
            bot.send_message(message.chat.id, talking("anime miss"), parse_mode='html')
    elif message.text == "Description" or message.text == "Rating" or message.text == "Characters" or \
            message.text == "Information":
        if check_anime(id_user, user_info):
            text = anime_info(message.text.lower(), id_user, user_info)
            bot.send_message(message.chat.id, text, parse_mode='html')
        else:
            bot.send_message(message.chat.id, talking("anime miss"), parse_mode='html')
    elif message.text == "Back to main menu":
        start(message)
    elif message.text == "Back":
        message.text = 'Yes, that`s my anime'
        get_text(message)
    elif message.text == "Yes, that`s my anime":
        text = 'Choose what do you want:'
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2, one_time_keyboard=True)
        button1 = types.KeyboardButton('Description')
```

Рисунок 3.5.1 – main.py @bot.message_handler(content_types=['text'])

```
        button2 = types.KeyboardButton('Rating')
        button3 = types.KeyboardButton('Characters')
        button4 = types.KeyboardButton('Information')
        button5 = types.KeyboardButton('Back to main menu')
        button6 = types.KeyboardButton('Arts')
        markup.add(button6, button1, button2, button3, button4, button5)
        bot.send_message(message.chat.id, text, reply_markup=markup)
    elif message.text == "No":
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1, one_time_keyboard=True)
        button1 = types.KeyboardButton('Back to main menu')
        markup.add(button1)
        bot.send_message(message.chat.id, talking("problems"), reply_markup=markup)
    elif check_characters(id_user, message.text, user_info):
        bot.send_message(message.chat.id, talking("waiting"), parse_mode='html')
        url = make_request(message.text, '', 'character', id_user, user_info)
        try:
            bot.send_photo(message.chat.id, url)
        except urllib.error.HTTPError:
            url = make_request(message.text, '', 'character', id_user, user_info)
            bot.send_photo(message.chat.id, url)
        bot.send_message(message.chat.id, "Pick another one or go to main menu", parse_mode='html')
    else:
        user_info[id_user][2] = 'name_anime'
        bot.send_message(message.chat.id, talking("waiting"), parse_mode='html')
        res = (for_request(analyse(message.text), 'anime', id_user, user_info))
        if res == "language" or user_info[id_user][2] == 'name_anime':
            bot.send_message(message.chat.id, talking("language"), parse_mode='html')
        else:
            text = 'Is it anime ' + res + '?'
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1, one_time_keyboard=True)
            button1 = types.KeyboardButton('Yes, that`s my anime')
            button2 = types.KeyboardButton('No')
            markup.add(button1, button2)
            bot.send_message(message.chat.id, text, reply_markup=markup, parse_mode='html')

            |
bot.polling(none_stop=True)
```

Рисунок 3.5.2 – main.py @bot.message_handler(content_types=['text'])

Кнопкові меню передбачаються 3 типами: початкове(рисунок 3.3), меню для вибору потрібної інформації (рисунок 3.5.3) та перелік персонажів для пошуку картинок(рисунок 3.5.4), який надає можливість скролу.

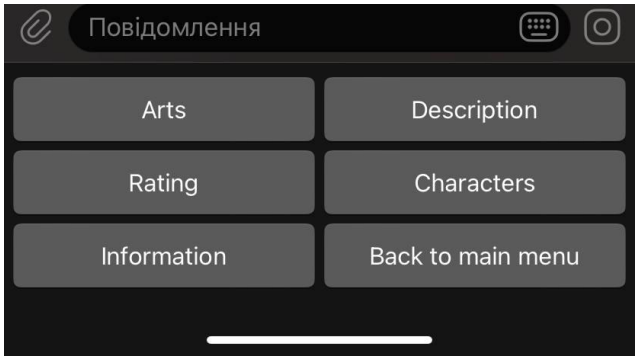

Рисунок 3.5.3 – Тестування меню для потрібної інформації

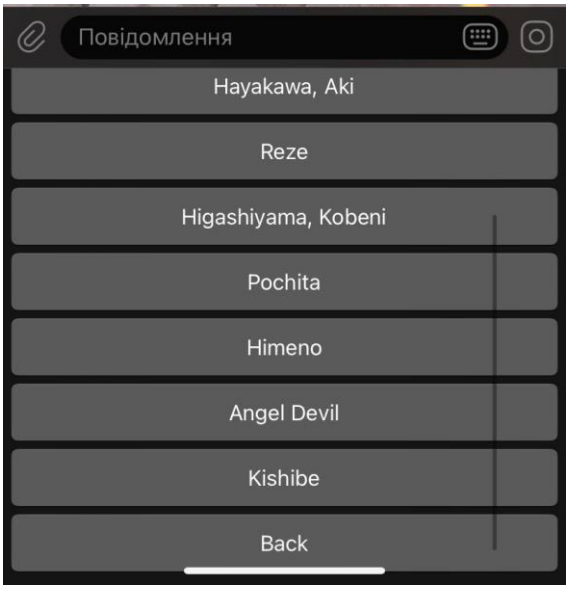

Рисунок 3.5.4 – Тестування меню вибору конкретного персонажа для пошуку зображень

Для перевірки наявності інформації про кожного користувача в пам’яті існує модуль checking з функціями check_anime(user_id, user_info) та check_characters(user_id, name, user_info). Його основною задачею є те, щоб вдосконалитись, що значення в словнику user_info за ID теперішнього користувача є не дефолтними. Обидва методи передбачають, що за відсутності інформації користувача потрібно перепитати його запит, а також не відображати кнопки з переліком персонажів для пошуку зображень.

checking.py

```
def check_anime(user_id, user_info):
    anime_link = (user_info[user_id])[0]
    if anime_link == 'anime_link':
        return False
    else:
        return True


def check_characters(user_id, name, user_info):
    characters = (user_info[user_id])[1]
    if characters == 'characters' or name not in characters:
        return False
    else:
        return True
```

Рисунок 3.6 – checking.py

За більшість текстових повідомлень чат-боту відповідає модуль fatic з

функцією talking, яка приймає на вхід параметр topic(тематика ситуації). Завдяки файлу answers.json метод за визначений тегом знаходить варіанти відповідей чат-боту та вибирає випадкову з них. Цей модуль викликається в багатьох інших модулях програми.

fatic.py

```python
import random
import json

with open('answers.json') as file:
    data_answers = json.load(file)


def talking(topic):
    for tags in data_answers['answers']:
        topic2 = tags["tag"]
        if topic == topic2:
            answers = tags["message"]
            return random.choice(answers)
```

Рисунок 3.7.1 – fatic.py

answers.json

```json
{"answers": [
        {"tag": "greeting",
         "message": ["I am bot AnimeTalker, what would you like to do?", "I am bot AnimeTalker, choose what you want to do",
           "I am AnimeTalker, so... Would you like to talk about the anime or discover some arts", "Nice to see you there, I am bot AnimeTalker, choose what you want to do",
           "Let`s begin with anime you want to talk about or discover arts of your favourite characters"]
        },
        {"tag": "talking",
         "message": ["It`s a great idea, so let`s start from your favourite anime!", "I am crazy about anime, my favourite is Nana, what about you?",
           "So... What anime do you want to discuss?", "What is the name of anime you want to talk about?", "Let`s begin with anime you want to talk about",
           "Tell me please the name of anime", "Let`s start with the name of your anime"]
        },
        {"tag": "characters",
         "message": ["Please choose your character", "Choose the character from the list",
           "Here are characters, please choose one from the list", "Here are the buttons with characters, choose the right one",
            "You need to choose at least one character from the given list"]
        },
        {"tag": "waiting",
         "message": ["Just wait... It will take some time", "Just wait... I am searching for it",
           "Please wait while I am searching", "Nice choice, please wait", "Please wait, I am searching"]
        },
        {"tag": "problems",
         "message": ["Sorry, your request is not correct to me, try again", "Just try again... I can`t work with your request.",
           "Sorry, I can`t find any information to your request. Please, can you say it in other words"]
        },
        {"tag": "language",
         "message": ["Sorry, your request is not correct to me, try to say it in English, please", "Just try again, but write in English.",
           "Sorry, I can`t find any information to your request. Please, can you say it in English, please"]
        },
        {"tag": "anime miss",
         "message": ["Sorry, you need to choose anime at first and then try again", "Just try again, but, please, choose your anime at first",
           "Sorry, you have not chosen any anime, write its name name and try again"]
        },
        {"tag": "info miss",
         "message": ["Sorry, this information is absent", "My apologizes, this information is absent",
           "Sorry, I tried to find this info, but it`s absent"]
        }
    ]
}
```

Рисунок 3.7.2 – answers.json

Оскільки повідомлення чат-боту може складатися з шаблону питання, наприклад, «I am crazy about anime, my favourite is Nana, what about you?», де

відповідь користувача може складатися не лише з назви, або може мати друкарську помилку, потрібно передбачати таку ситуацію. Модуль analysingMessage, використовуючи бібліотеку textblob, розділяє вхідне повідомлення на n-грами в методі analyse для знаходження конкретної назви аніме в запиті. Якщо користувач напише «my favorite anime is Chainsaw man», де «Chainsaw man» - це найменування, то n-грами будуть такі:

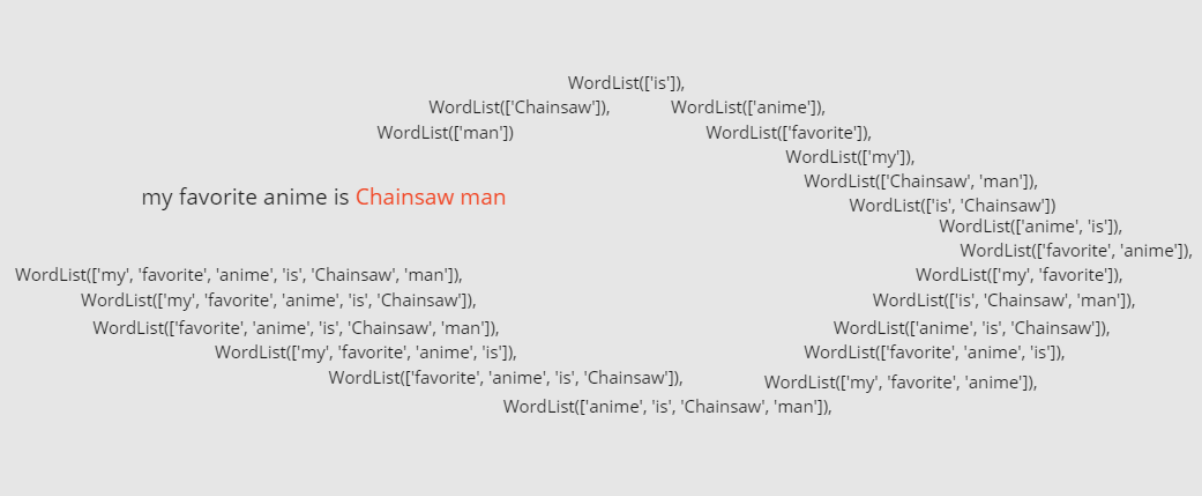

Рисунок 3.8.1 – n-грами для запиту

Отож, метод analyse(message) поверне список з елементів типу Wordlist, які надалі оброблятимуться в іншому модулі.

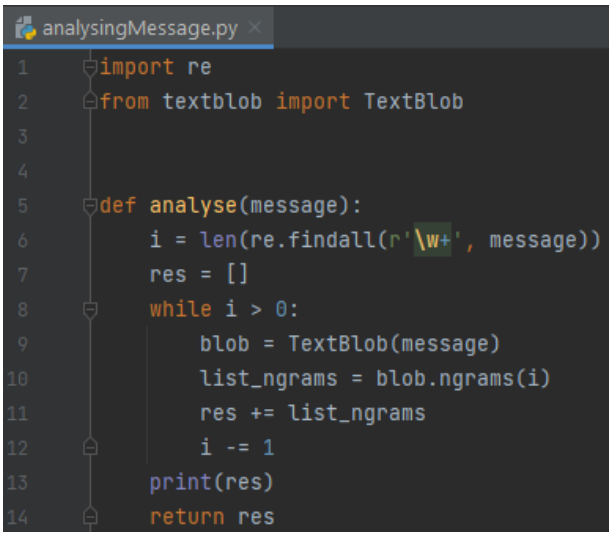

Рисунок 3.8.2 – analysingMessage.py

За надання інформації на запит за конкретним аніме відповідає модуль animeInfo. У ньому наявно 4 функції:

1. make_request – відповідає за пошук конкретної назви аніме та його посилання на сайті «https://myanimelist.net», а також за пошук картинок за ім’ям персонажа на платформі Bing. Знаходження найменування відбувається через парсинг сторінки за вхідним запитом: якщо знайдено співпадіння, яке абсолютно відповідає параметру, програма поверне значення "exact", в іншому випадку – "maybe". У процесі опрацювання відбувається зміна даних користувача у пам’яті, тобто присвоюється назва аніме та посилання на нього, якщо наявні хоча б якісь результати пошуку, і дані в програмі ще дефолтні. При наявності проблем з мовою повідомлення користувача, буде повертатиметься "language", яке в main.py буде опрацьовано як тема незрозумілої мови для програми. Модуль bing_image_urls допомагає знайти 1000 різних картинок за ім’ям персонажа і повертає посилання на випадкову з них.

```
def make_request(request, name, typeR, user_id, user_info):
    if typeR == 'anime':

        url = 'https://myanimelist.net/anime.php?q=' + request + '&cat=' + typeR
        try:
            html = urlopen(url).read()
        except:
            return "language"

        soup = BeautifulSoup(html, 'html.parser')
        res = soup.find_all('strong')
        for el in res:
            anime_name = el.string
            for link in soup.find_all("a", {"class": "hoverinfo_trigger fw-b fl-l"}):
                u = link.get('href')
                if u is not None and link.string == anime_name:
                    symbols = ":", "!", "'", "(", ")", ".", "/", "+", ";"
                    for s in symbols:
                        anime_name = anime_name.replace(s, "")
                        name = name.replace(s, "")
                    if name.lower() == anime_name.lower() or (el == res[0] and (user_info[user_id])[2] == "name_anime"):
                        (user_info[user_id])[0] = u
                        (user_info[user_id])[2] = anime_name
                        if name.lower() == anime_name.lower():
                            return "exact"
        return "maybe"
    elif typeR == 'character':

        name_anime = (user_info[user_id])[2]
        url = bing_image_urls(request + " " + name_anime, limit=1000)
        rand_url = random.choice(url)
        while rand_url:
            try:
                urlopen(rand_url)
                return rand_url
            except:
                rand_url = random.choice(url)
```

Рисунок 3.9.1 – animeInfo.py, make_request(request, name, typeR, user_id, user_info)

2. for_request – відповідає за опрацювання списку n-грам, щоб знайти назву найбільш ймовірного найменування аніме з запиту користувача. За допомогою застосування методу make_request на кожну n-граму

відбувається пошук можливих варіантів і повертається результат. Аналогічно з роботою попередньої функції: при знаходженні точного співпадіння програма зупиниться і відправить саме його, в інших випадках – все залежить від попередньої та подальшої роботи make_request.

```
def for_request(list1, typeR, user_id, user_info):
    request = ''
    i = 0
    while i < len(list1):

        words = list1[i]
        k = 0

        while k < len(words):
            if k != 0:
                if typeR == 'anime':
                    request += '%20'
                else:
                    request += '+'
            request += words[k]
            k += 1
        names = [''.join(w) for w in words]
        name = ''
        n = 0
        while n < len(names):
            if n != len(names) - 1:
                name += names[n] + ' '
            else:
                name += names[n]
            n += 1
        calling = (make_request(request, name, typeR, user_id, user_info))
        if calling == "exact":
            make_request(request, name, typeR, user_id, user_info)
            return (user_info[user_id])[2]
        else:
            request = ''
            i += 1
    return (user_info[user_id])[2]

```

Рисунок 3.9.2 – animeInfo.py, for_request(list1, typeR, user_id, user_info)

3. anime_info – метод, який за допомогою парсингу на збереженому за ID користувачем сайті знаходить інформацію про опис, рейтинг, загальну інформацію та список персонажів в залежності від теперішнього запиту. Якщо якийсь з критеріїв відсутній, чат-бот поверне повідомлення за темою "info miss", використовуючи вже розглянутий модуль fatic.py та його функцію talking.

```python
def anime_info(task, user_id, user_info):
    try:
        anime_link = (user_info[user_id])[0]
        html = urlopen(anime_link).read()
    except urllib.error.HTTPError:
        return 'error'

    soup = BeautifulSoup(html, 'html.parser')
    result = ''
    if task == 'description':

        res = soup.find("p", {"itemprop": "description"})
        for e in res:
            if e.string is not None:
                result += e.string
        if result == '':
            result = talking("info miss")
        return result

    elif task == 'rating':
        answer = ''
        res = soup.find_all("div", {"class": "score-label score-8"})
        if res:
            for e in res:
                answer = 'Rating is ' + ''.join(e.string.strip())
        else:
            answer = talking("info miss")
        return answer

    elif task == 'information':
        result = " "
        res = soup.find_all('h2')
        for e in res:
            if e.string == 'Alternative Titles':
                res2 = (e.find_all_next("div", {"class": "spaceit_pad"}))
                i = 0
                while i < len(res2):
                    a = res2[i]
                    t = a.find_next('span').string
                    if t == 'Score:':
                        return result
                    result += t
                    if i == 0:
                        t = ((e.find_next("div", {"class": "spaceit_pad"})).find_all(text=True, recursive=False)[1])
                        t = t.replace('\n', '')
                        result += ' ' + t.lstrip() + '\n'
                    else:
                        a = res2[i - 1]
                        t = ((a.find_next("div", {"class": "spaceit_pad"})).find_all(text=True, recursive=False)[1])
                        if t == '\n':
                            a = res2[i]
                            t = (a.find_next("a"))
                        m = t.string.replace('\n', '')
                        result += ' ' + m.lstrip() + '\n'
                    i += 1
        if result == '':
            return talking("info miss")

    elif task == 'characters':
        result = 'Characters are:' + '\n'
        res = soup.find("div", {"class": "detail-characters-list clearfix"})
        for e in res:
            res2 = (e.find_all("h3", {"class": "h3_characters_voice_actors"}))

            for c in res2:
                result += c.find_next('a').string + '\n'
        if result == '':
            result = talking("info miss")
        return result
```

Рисунок 3.10.1-3.10.2 – animeInfo.py, anime_info(task, user_id, user_info)

4. choose_character – функція, яка допомагає отримати список персонажів заданого аніме у коректному вигляді для меню. Ці дані так само зберігаються в пам’яті задля того, щоб користувач зміг бачити та вибирати кнопки (рисунок 3.5.4). Для пошуку даних цей метод використовує вже описану функцію anime_info за назвою 'characters' для параметру task.

```
def choose_character(id_user, user_info):
    characters = anime_info('characters', id_user, user_info)
    characterArr = characters.split('\n')
    if len(characterArr) > 0:
        del characterArr[0]
        del characterArr[len(characterArr) - 1]
    return characterArr
```

Рисунок 3.10.1-3.10.2 – animeInfo.py, anime_info(task, user_id, user_info)

Перейдемо до тестування багатопоточності й функціональності при одночасному використанні з трьох різних акаунтів.

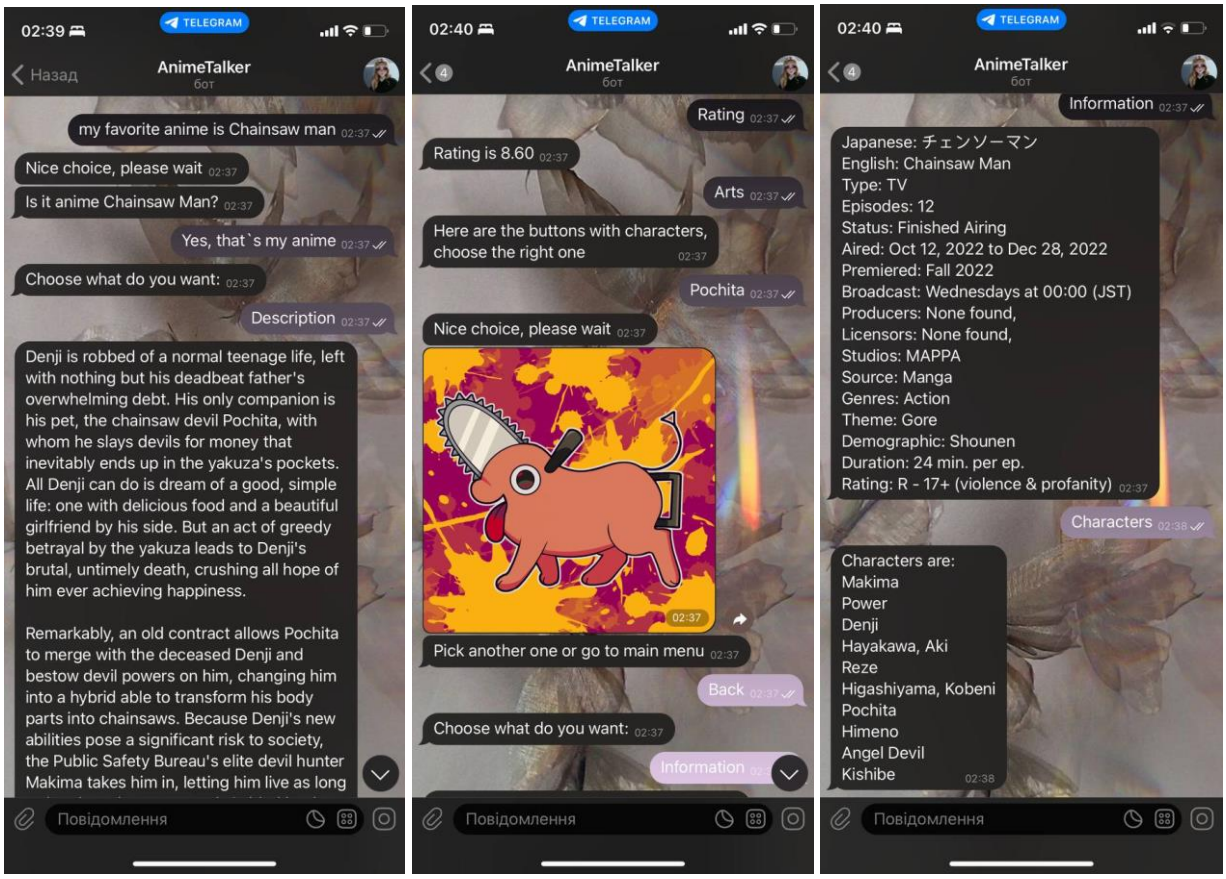

Рисунок 3.11.1-3.11.3 – Тестування першого користувача за запитом схеми на рисунку 3.8.1

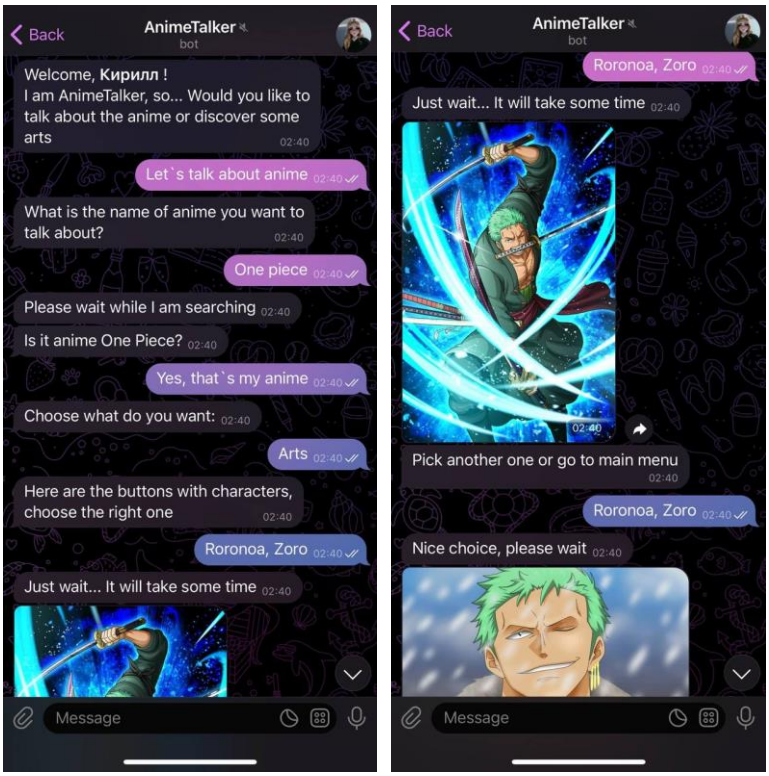

Рисунок 3.12.1-3.12.2 – Тестування другого користувача, включаючи запит на пошук різних картинок на одного персонажа

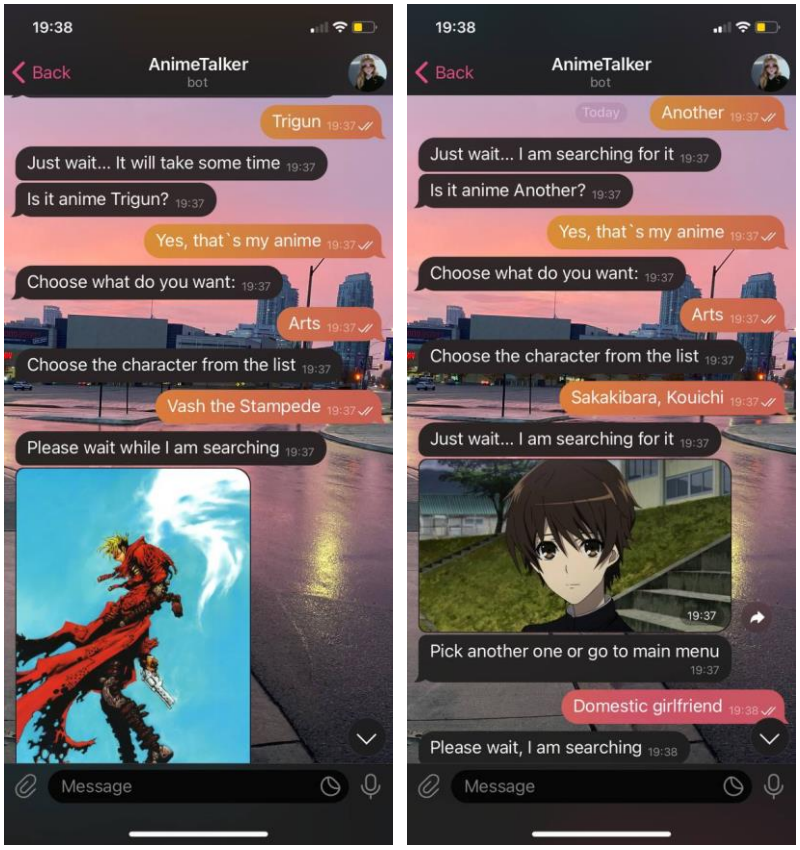

Рисунок 3.13.1-3.13.2 – Тестування третього користувача, включаючи запит на пошук різних аніме та їх персонажів

Висновок

Завдяки цій курсовій роботі було опановано теоретичні матеріали про історичний екскурс створення чат-ботів, їх класифікації, технологій при їх розробці, відмінності в методах та обширними можливостями для їх розробки. Дослідження надало змогу дізнатися про актуальність таких програм серед різних користувачів та їхніх вимог.

Результатом курсової роботи став чат-боту у месенджері Telegram, написаний мовою програмування Python, який використовує базові технології NLP та низку модулів для роботи з API сервісами. Вивчено основи роботи з проектуванням власних технологій онлайн-помічників, описано логіку визначеного проекту для пошуку інформації за потрібним запитом.

Додатково подано детальну структуру та код програми, а також тестування декількох різних користувачів для наочної демонстрації різноманітності запитів.

Отже, завдяки цьому дослідженню вдалось розробити власного чат-бота для спрощення пошуку інформації в інтернеті, який можна й надалі вдосконалювати та впроваджувати на постійне онлайн використання.

Список використаної літератури

1. [Електронний ресурс] What is a chatbot? https://www.oracle.com/chatbots/what-is-a-chatbot/
   (Перевірка на доступ до ресурсу - 05.05.2023)
2. [Електронний ресурс] The history of Chatbots – From ELIZA to ALEXA https://onlim.com/en/the-history-of-chatbots/
   (Перевірка на доступ до ресурсу - 05.05.2023)
3. [Електронний ресурс] What Is a Chatbot? Meaning, Working, Types, and Examples https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-chatbot/
   (Перевірка на доступ до ресурсу - 05.05.2023)
4. [Електронний ресурс] Hybrid chatbot: how to make humans and robots work together https://www.ringcentral.com/us/en/blog/integration-agents-chatbots-botmind/
   (Перевірка на доступ до ресурсу - 05.05.2023)
5. [Електронний ресурс] ELIZA: a very basic Rogerian psychotherapist chatbot https://web.njit.edu/~ronkowit/eliza.html
   (Перевірка на доступ до ресурсу - 05.05.2023)
6. [Електронний ресурс] Why use chatbots in your business. Top 9 reasons https://www.ideta.io/blog-posts-english/top-9-reasons-why-you-should-use-a-chatbot-in-your-business
   (Перевірка на доступ до ресурсу - 05.05.2023)
7. [Електронний ресурс] The Future of Chatbots: 80+ Chatbot Statistics for 2023 https://www.tidio.com/blog/chatbot-statistics/
   (Перевірка на доступ до ресурсу - 05.05.2023)
8. [Електронний ресурс] American Express Study Shows Rising Consumer Expectations for Good Customer Service https://www.inc.com/peter-roesler/american-express-study-shows-rising-consumer-expectations-for-good-customer-service.html
   (Перевірка на доступ до ресурсу - 05.05.2023)

9. [Електронний ресурс] Chatbot market revenue worldwide from 2018 to 2027 https://www.statista.com/statistics/1007392/worldwide-chatbot-market-size/ (Перевірка на доступ до ресурсу - 05.05.2023)
10. [Електронний ресурс] Skills and Technologies Driving Chatbot Innovation https://jasoren.com/skills-and-technologies-driving-chatbot-innovation/ (Перевірка на доступ до ресурсу - 05.05.2023)
11. [Електронний ресурс] Which NLP Engine to Use In Chatbot Development https://blog.vsoftconsulting.com/blog/which-nlp-engine-to-use-in-chatbot-development (Перевірка на доступ до ресурсу - 05.05.2023)
12. [Електронний ресурс] HOW TO DEVELOP A CHATBOT https://vilmate.com/blog/how-to-develop-a-chatbot/ (Перевірка на доступ до ресурсу - 05.05.2023)
13. [Електронний ресурс] How do chatbots work? What is the Chatbot Architecture 101? https://www.senseforth.ai/conversational-ai/how-do-chatbots-work/ (Перевірка на доступ до ресурсу - 05.05.2023)
14. [Електронний ресурс] Soul of the Machine: How Chatbots Work https://medium.com/@gk_/how-chat-bots-work-dfff656a35e2 (Перевірка на доступ до ресурсу - 05.05.2023)
15. [Електронний ресурс] 13 Best Open Source Chatbot Platforms to Use in 2022 https://botpress.com/blog/open-source-chatbots (Перевірка на доступ до ресурсу - 05.05.2023)
16. [Електронний ресурс] Найпопулярніші месенджери в Україні, – опитування https://konkurent.ua/publication/84235/naypopulyarnishi-mesendzheri-v-ukraini-opituvannya/ (Перевірка на доступ до ресурсу - 05.05.2023)
17. [Електронний ресурс] How to set up your Telegram Bot using BotFather https://blog.devgenius.io/how-to-set-up-your-telegram-bot-using-botfather-fd1896d68c02 (Перевірка на доступ до ресурсу - 05.05.2023)

18. [Електронний ресурс] What is Python? Executive Summary https://www.python.org/doc/essays/blurb/ (Перевірка на доступ до ресурсу - 05.05.2023)
19. [Електронний ресурс] Get started with PyCharm https://www.jetbrains.com/help/pycharm/quick-start-guide.html#code-assistance (Перевірка на доступ до ресурсу - 05.05.2023)
20. [Електронний ресурс] random — Generate pseudo-random numbers https://docs.python.org/3/library/random.html# (Перевірка на доступ до ресурсу - 05.05.2023)
21. [Електронний ресурс] urllib.request — Extensible library for opening URLs https://docs.python.org/3/library/urllib.request.html#module-urllib.request (Перевірка на доступ до ресурсу - 05.05.2023)
22. [Електронний ресурс] urllib.error — Exception classes raised by urllib.request https://docs.python.org/3/library/urllib.error.html#module-urllib.error (Перевірка на доступ до ресурсу - 05.05.2023)
23. [Електронний ресурс] re — Regular expression operations https://docs.python.org/3/library/re.html (Перевірка на доступ до ресурсу - 05.05.2023)
24. [Електронний ресурс] Welcome to pyTelegramBotAPI’s documentation! https://pytba.readthedocs.io/en/latest/index.html (Перевірка на доступ до ресурсу - 05.05.2023)
25. [Електронний ресурс] TextBlob: Simplified Text Processing https://textblob.readthedocs.io/en/dev/ (Перевірка на доступ до ресурсу - 05.05.2023)
26. [Електронний ресурс] beautifulsoup4 4.12.2 https://pypi.org/project/beautifulsoup4/ (Перевірка на доступ до ресурсу - 05.05.2023)
27. [Електронний ресурс] bing-image-urls 0.1.5 https://pypi.org/project/bing-image-urls/ (Перевірка на доступ до ресурсу - 05.05.2023)

28. [Електронний ресурс] json — JSON encoder and decoder
https://docs.python.org/3/library/json.html
(Перевірка на доступ до ресурсу - 05.05.2023)