МІНІСТЕРСТВО ОСВ ІТИ І НАУКИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК СЕКЦІЯ ІТП

ВИПУСКНА РОБОТА

на тему: **«Інформаційне та програмне забезпечення пошуку рисунків за допомогою телеграм бота»**

Завідувач випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Шаповалов С .П.**

Студента групи ІН – 61 **Сердюк Ю. О.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

СЕКЦІЯ ІКТ

Затверджую_____

Зав. кафедрою Довбиш А.С.

“_____” _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-61 спеціальності “Інформатика” денної форми навчання Сердюка Юрія Олександровича.

Тема: “**Інформаційне та програмне забезпечення пошуку рисунків за допомогою телеграм бота**”

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) огляд існуючих рішень; 2) постановка завдання й формування завдань дослідження; 3) програмна реалізація та її опис; 4) тестування; 5) висновки.

Дата видачі завдання “_____”_____2020 р.

Керівник випускної роботи_____ Шаповалов С.П.

Завдання прийняв до виконання _____ Сердюк Ю.О.

РЕФЕРАТ

Записка: 42 стор., 40 рис., 1 додаток, 14 джерел

Об’єкт дослідження – Telegram Bot.

Мета роботи – Розроблення телеграм бота для пошуку рисунків.

Методи дослідження – Отримання та парсинг сторінки рисунків у Google за запитом. Застосування мови програмування Python для розробки телеграм бота.

Результати – Розроблено телеграм бота який здійснює пошук рисунків за запитом користувача, і повертає в відповідь таку ккількість яку обире користувач. Підтримує всі мови, фрази з декількох слів, та смайли.

TELEGRAM MOBILE, TELEGRAM WEB, TELEGRAM DESKTOP, PYTHON

ЗМІСТ

ВСТУП

Telegram – багатоплатформовий месенджер, який дозволяє обмінюватися повідомленнями і медіафайлами багатьох форматів. Використовується пропріетарна серверна частина с закритим кодом, що працює на потужностях декількох компаній США і Німеччини, що фінансуються Павлом Дуровим в обсязі близько 13 млн. доларів США щорічно, і кілька клієнтів з відкритим вихідним кодом, у тому числі під ліцензією GNU GPL [3].

Павло Дуров є засновником соціальної мережі «ВКонтакте». В інтерв'ю The New York Times Павло розповів, що початкова ідея створення проєкту прийшла йому ще в 2011 році, коли до його дверей приходили спецназівці. Коли останні все-таки пішли, П. Дуров відразу ж написав своєму братові Миколі. Тоді ж Павло й усвідомив, що у нього немає безпечного способу комунікації з братом. Сервіс побудований на технології шифрування листування MTProto, розробленої братом Павла - Миколою. Сам «телеграм» спочатку був експериментом, який належитьав Павлу, компанії Digital Fortress з метою протестувати MTProto на великих навантаженнях.

У жовтні 2013 року Telegram мав 100 000 активних користувачів (РисВ.1),

Рисунок В.1 – Кількість користувачів Telegram станом в період 2013-2018рр.

а в березні 2018 року аудиторія Telegram зросла до 200 мільйонів активних користувачів.

Станом на 24 квітня 2020 року Telegram має більш ніж 400 мільйонів активних користувачів (РисВ.2). Про це повідомив сам Telegram кожному користувачу особисто.

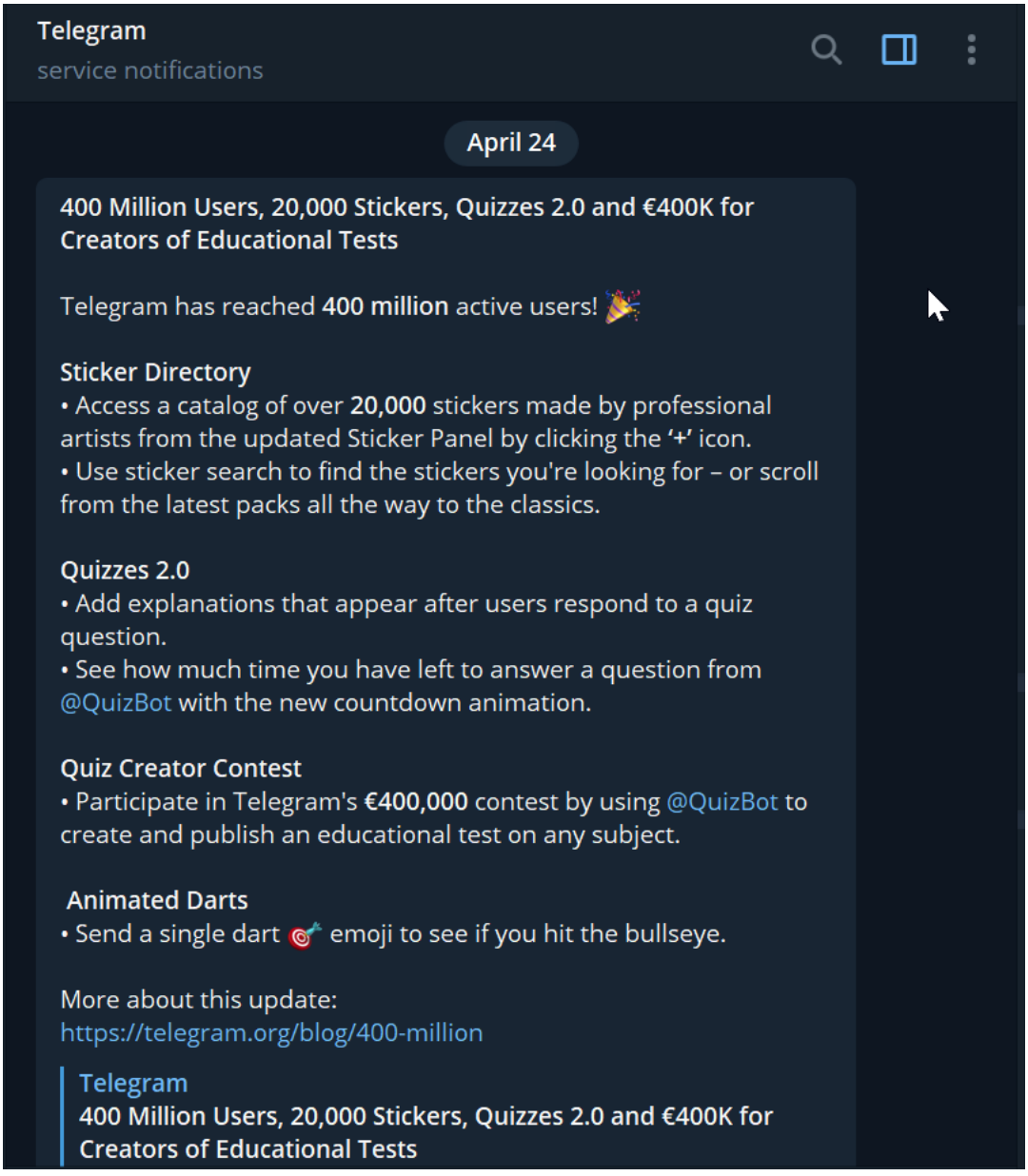

Рисунок В.2 – Кількість користувачів у квітні 2020 року

Telegram – хмарний месенджер, його можна використовувати одночасно на декількох пристроях, і всі чати і файли (за винятком Секретних чатів) будуть доступні на цих пристроях. У всіх чатах можна використовувати голосові повідомлення, відеоповідомлення, прикріплення файлів, стікери, gif-анімації і

емодзі; є відмітка про те, що співрозмовник прочитав повідомлення і т. п. А серед основних особливостей месенджера можна виділити наступні:

- Звичайні чати – це хмарні чати між двома користувачами.
- «Секретні чати» (Secret Chats) – окремий вид чатів, які використовують наскрізне шифрування. Історія повідомлень в таких чатах не зберігається в хмарі, а тільки на пристроях відправника і одержувача. З «Секретних чатів» Ви не зможете надсилати повідомлення.
- Усі потрібні програми можна зберегти в окремий діалог. Також туди можна завантажити необмежену кількість файлів, тобто месенджер надає нескінченне хмарне сховище.
- Окрім звичайних і «Секретних чатів» в Telegram є групи і канали.
- Анонімність: Telegram не надає нікому, окрім самих адміністраторів каналу, інформацію про те, хто веде канал і хто на нього підписаний.
- Боти: за допомогою спеціального API сторонні розробники можуть створювати «ботів», спеціальні акаунти, керовані програмами.
- З травня 2015 року користувачі програми можуть створювати власні стікери. Стікери – це зображення, які можуть допомогти висловити емоції краще emoji.
- Telegram дозволяє шукати інших користувачів не тільки за номером телефону, а також використовувати публічні імена (username).
- Фоторедактор і створення gif-файлів.
- Голосові дзвінки – використовують закінчене шифрування, як і «Секретні чати». Коли це можливо, дзвінки будуть проходити через однорангове з’єднання, використовуючи аудіокодеки для економії трафіку.
- У Telegram вбудований повноцінний фоторедактор. На відміну від інших месенджерів, на фотографію можна не тільки додати стікери,

але і повністю її відретушувати – змінити яскравість, контрастність, насиченість кольору та інше.

Станом на березень 2020 року офіційні клієнти для Telegram включають у себе:

- Мобільні додатки для Android і iOS / iPadOS;
- Додатки для Windows, Linux і macOS;
- Веб-додаток, веб-додатки для Chrome app, веб-додаток для React.

Telegram переведений і продовжує переводитися на наступні мови:

- версія для Windows: білоруський, чеська, французька, польська, українська, турецька та російська;
- для Android: азербайджанський, білоруський, чеська, французька, польська, українська, турецька, узбецький і російська;
- для iOS (iPhone і iPad): білоруський, чеська, польська, українська, турецька та російська;
- для OS X: білоруська, польська і російська.

1. ОГЛЯД ВІДОМИХ РІШЕНЬ

1.1 Види ботів та їх порівняльна оцінка

- Інтеграція з іншими службами. Боти можуть збагатити чати Telegram вмістом із зовнішніх служб: Gmail Bot, GIF бот, IMDB бот, Wiki bot, Музичний бот, Youtube bot, GitHub бот[9]
- Приймання платежі від користувачів Telegram. Бот може пропонувати платні послуги або працювати як віртуальна вітрина.
- Спеціальні боти для сповіщення: прогнози погоди, переклади, форматування або інші послуги.
- Бот Markdown, бот наклейки, бот для голосування, як бот
- Боти з одиночними та багатокористувацькими іграми. Можуть запропонувати багатий досвід HTML5, від простих аркад та пазлів до 3D-стрілок та стратегічних ігор у режимі реального часу. Наприклад GameBot, Gamee
- Боти можуть з'єднати людей, які шукають партнерів для спілкування на основі спільних інтересів або близькості.

У роботі за конкрурента було взято @ImageSearchBot (Рис 1.1) [14]. Єдиним призначенням цього Телеграм бота є моментальний пошук картинок з будь-яким текстом або зображенням яке здатне замінити цей текст. Ці картинки можна відправляти своїм співрозмовникам у різних групах і чатах, замість текстових повідомлень, щоб здивувати їх і урізноманітнити бесіду.

Рисунок 1.1 Логотип ImageSearchBot

У ході роботи бот був протестований. Як він нам повідомляє при запуску, що відправивши йому якийсь текст, буде повернуто зображення яке відповідає цьому тексту. Також додавши бота в групову бесіду можно за його допомогою шукати зображення по группі. Демонстрація роботи (Рис 1.2):

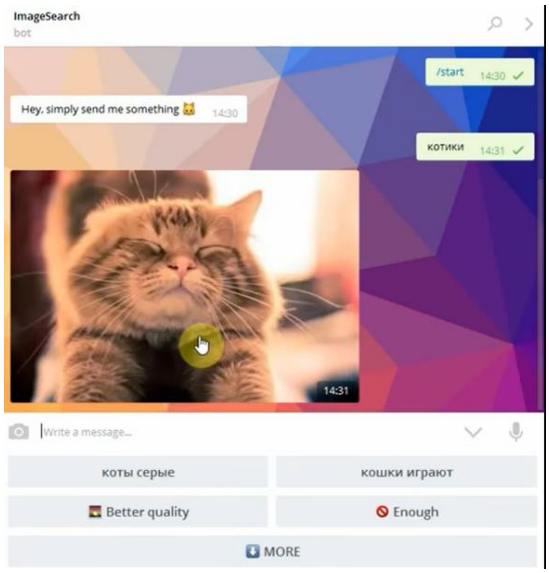

Рисунок 1.2 – Демонстрація роботи пошуку рисунку ImageSearchBot

Відправляємо боту запит «котики», і бачимо результат. Бот повернув нам зображення відповідно до запиту який було відправлено. Також є можливість отримати більшь детальне зображення на подібну тематику. Наприклад, «котики граються» (Рис 1.3).

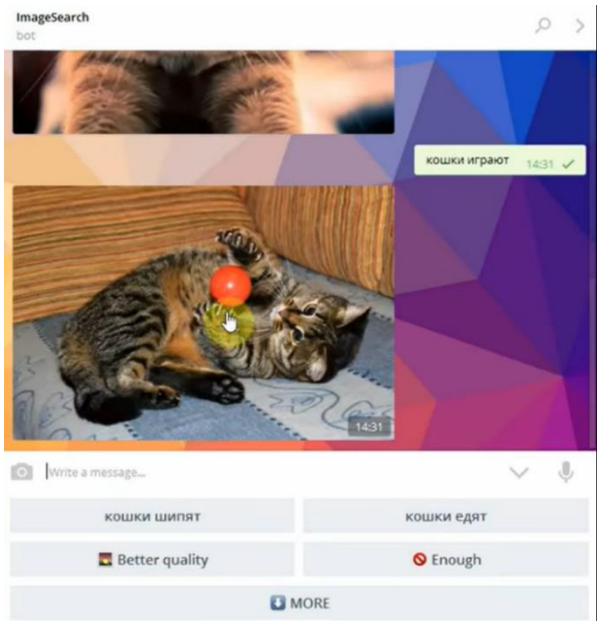

Рисунок 1.3 – Демонстрація роботи пошуку рисунку за декількома словами в ImageSearchBot

Також бот надає нам можливість отримати зображення кращої якості, або ж завершити сесію роботи (Рис 1.4).

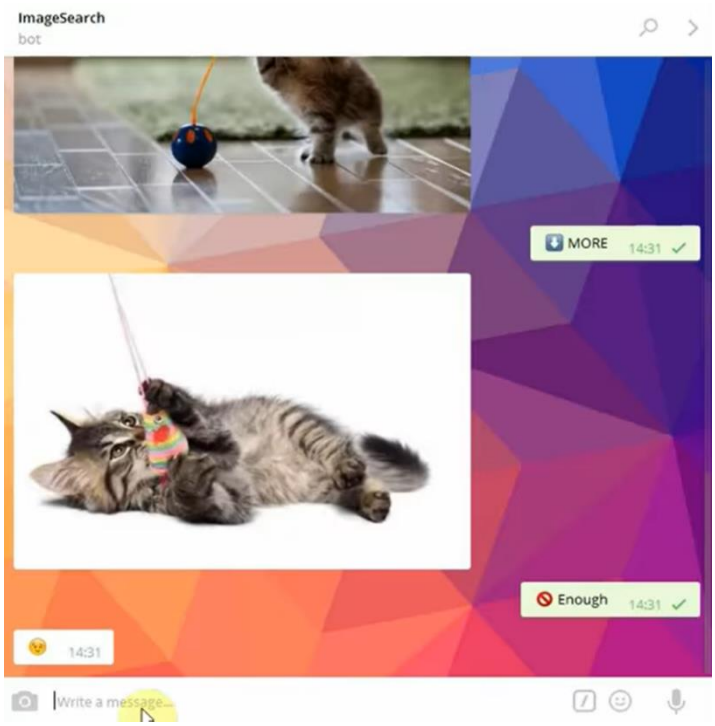

Рисунок 1.4 – Демонстрація роботи пошуку більшої кількості рисунків в ImageSearchBot

Також, є можливість додати бота в груповий чат (Рис 1.5), де бот показує гарну працездатність.

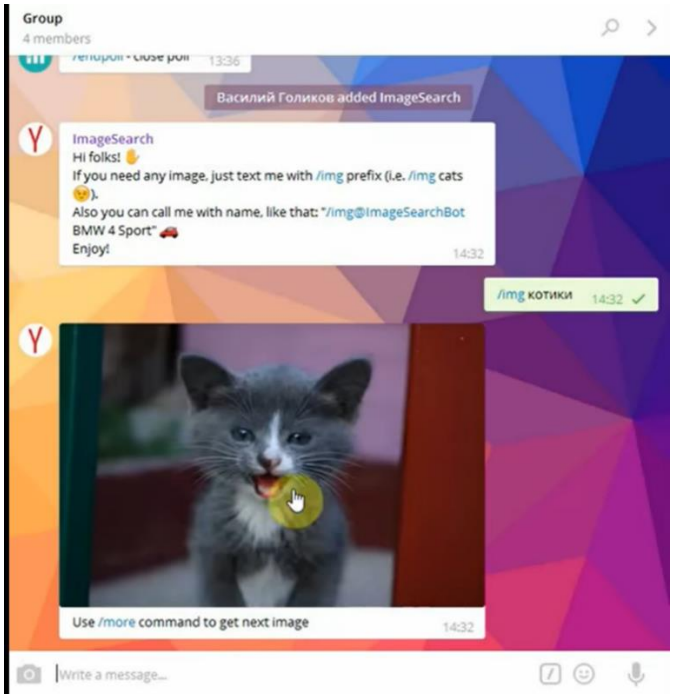

Рисунок 1.5 – Демонстрація роботи додавання ImageSearchBot в груповий чат

Було встановлено, що бот відповідає на запити досить шкидко, приблизно 0.5-1 секунди. Це є досить прийнятно. Проте є мінус у роботі цього бота – це те, що він здійснює пошук в Яндекс, який, нажаль, не працює без підключення до VPN з'єднаяння.

Також було протестовано @ImageBot. Як повідомляє розробник про цього бота: «…найпростіший бот для пошуку зображень. Для того щоб робот почав роботу, вам необхідно задати йому команду «get» і назву картинки. Необхідний файл незабаром до вас прийде. Зараз дуже популярні анімовані картинки Gif. Щоб попросити бота знайти для вас такі зображення, вам слід в запиті вказати «getgif».

У ході тесту бот не відповідав на жоден запит, ні українською ні англійскою мовами. Причини по яких бот не відповідав не було встановлено.

З нашим припущенням, це пов'язано с тим що він здійснює пошук по Якдексу і не зовсім коректний для використання у нашому регіоні (Рис 1.6).

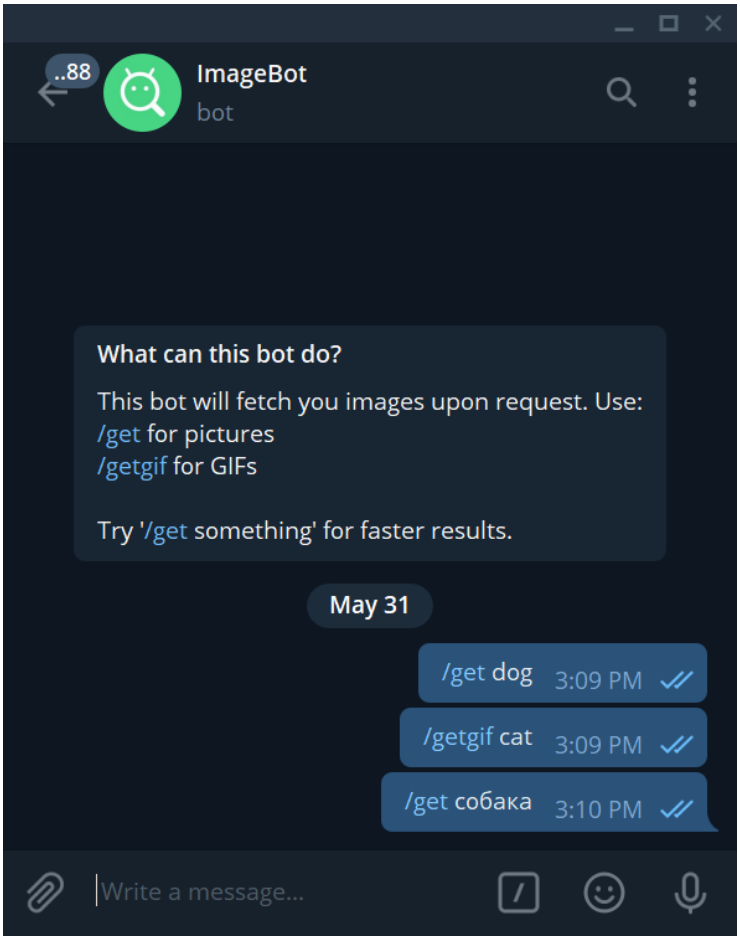

Рисунок 1.6 – Демонстрація роботи додавання ImageBot

Таблиця 1.2 Порівняльна таблиця ImageSearchBot та ImageBot

	ImageSearchBot	ImageBot
Пошук рисунків	Так	Так
Підтримка роботи в груповому чаті	Так	Ні
Підтримка різніх мов	Ні	Ні
Підтримка фраз	Так	Ні
Підтримка пошуку смайликів	Ні	Ні
Пошук гіф-зображень	Ні	Так

В приведеній вище таблиці можна зробити висновок що представлені конкуренти не мают такої можливості як:

- підтримка пошуку смайликів
- підтримка різніх мов
- Пошук гіф-зображень
- Підтримка роботи в груповому чаті

1.2 Прототип боту, як конкурент

У роботі буде реалізовано бота (Рис1.7) який повинен підтримувати пошук зображень на різних мовах, який підтримує фрази та спроможний шукати

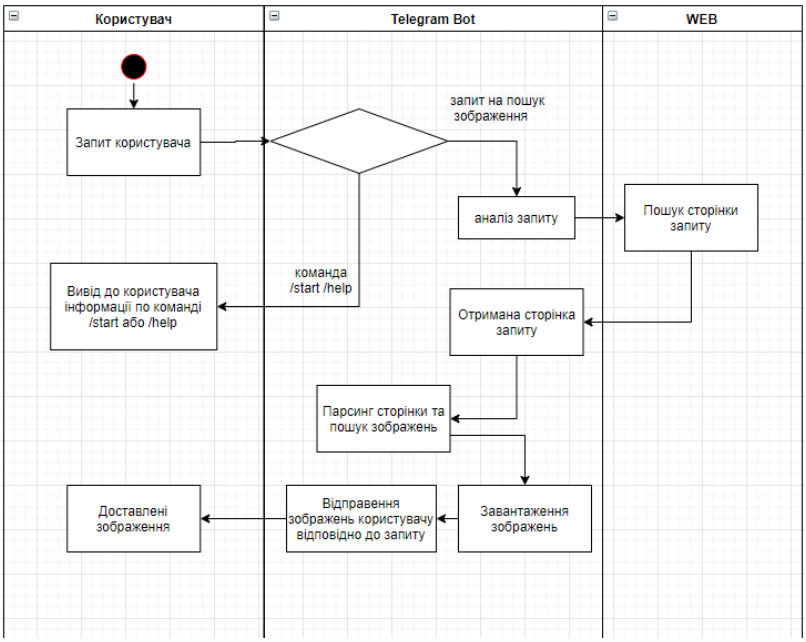

Рисунок 1.7 – Діаграма послідовності роботи бота

смайлики. Подана діаграма демонструє роботу бота та його взаємодію з користувачем за всесвітньою мережою.

1.3 Постановка задачі

За завдання було взято розробку телеграм бота який повертає зображення по запиту користувача. І відправляє запрошувану кількість зображень. Підтримує Google. Підтримує звичайні зображення, такі як JPG. Вміє працювати зі смайлами.

Команди які повинні бути реалізовані:

- / help, показує довідкове повідомлення з інформацією про бота та його використання;
- / start: команда привітання бота з користувачем та вхідної інформації.

2 ВИБІР МЕТОДУ РІШЕННЯ

2.1 Інструментарії та програмне забезпечення

Телеграм бот написаний за допомогою мови Python версії 3.8.2. Поширене застосування мови програмування - одноразові скрипти для маніпуляцій з даними, конвертери і інші речі. Гарний в застосуванні до таких задач як: «прочитати 200 Мб даних, відфільтрувати записи і залити їх в базу даних, а заодно порахувати базову статистику і намалювати графік» або «відфільтрувати файли логів за датою створення.

Для написання було вікористано середовище розробки Microsoft Visual Studio [15]. Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня.

2.2 Застосування відомих бібліотек та компонетів

У роботі були використані наступні бібліотеки (Рис 2.1) мови Python :

```
import telebot
from telebot import types
from urllib.request import  urlretrieve
import requests
import logging
from bs4 import BeautifulSoup
```

Рисунок 2.1 – Використані бібліотеки мови Python у роботі

Сторонні бібліотеки були встановлені за допомогою наступних команд:

- pip3 install pyTelegramBotAPI
- pip3 install bs4
- pip3 install requests

Їх актуальні версії

bs4 **0.0.1**

requests 2.7.0

pyTelegramBotAPI 3.7.1

2.2.1 Бібліотека pyTelegramBotAPI

Бібліотека ботів Telegram[12], з декораторами маршрутів, яка дозволяє розробнику створювати телеграм-ботів на мові Python використовуючи стандартні методи бібліотеки. Для початку роботи потрібно отримати Access Token. Для його створення потрібно поговорити з @BotFather.

BotFather – найпростіший спосіб для реєстрації, настройки та управління іншими telegram-ботами. За допомогою BotFather можна зареєструвати необмежену кількість нових спамерських пошукових роботів. Єдиною умовою для реєстрації нового бота – є його унікальний username і виконати кілька простих кроків [4]:

1. придумати ім’я бота – яке буде відображуватися в контактах користувачів (Рис 2.2);
2. придумати *username* – ім’я повинно бути унікальним і закінчуватися на «bot». Допускаються букви латинського алфавіту, цифри і символ підкреслення. Загальна кількість символів не менше 5 і не більше 32;
3. якщо все в порядку, то у відповідь ми отримаємо повідомлення з Access Token. Токен необхідний для роботи з Bot API за допомогою

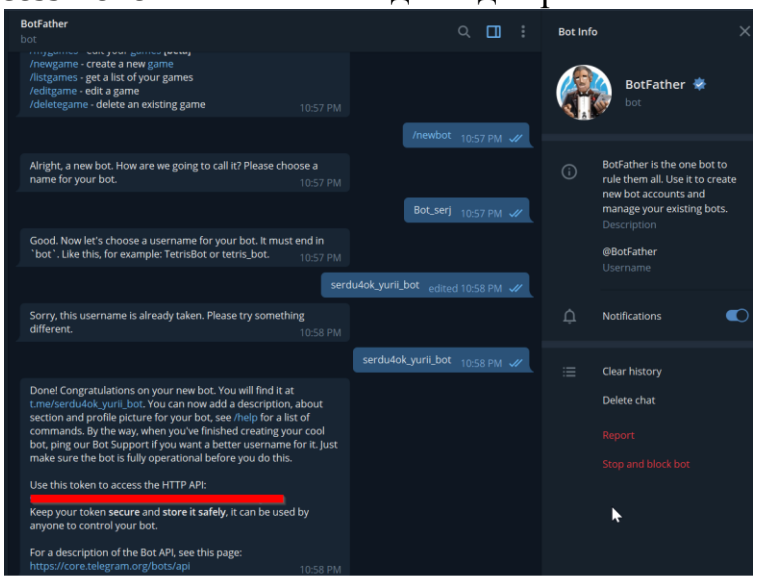

Рисунок 2.2 – Створення бота у інтерфейсі BotFather

http-протоколу. Не можна передавати його іншим і бажано не втрачати.

2.2.2 Бібліотека bs4

Beautiful Soup – це бібліотека Python для отримання даних з файлів HTML і XML [7]. Вона зазвичай економить програмістам години і дні роботи. Може перетворити навіть неправильну розмітку в дерево синтаксичного розбору. Підтримує прості і природні способи навігації, пошуку та модифікації дерева синтаксичного розбору.

Для роботи конструктору Beautiful Soup потрібно документ XML або HTML у вигляді рядка (або відкритого файлоподобного об'єкта). Він зробить синтаксичний розбір і створить у пам’яті структури даних, відповідні документи. Якщо обробити за допомогою Beautiful Soup добре оформлений документ, то розібрана структура буде виглядати також як і вихідний документ. Але, якщо його розмітка буде містити помилки, то Beautiful Soup використовує евристичні методи для побудови найбільш відповідної структури даних [8].

Деякі теги можуть бути вкладеними (<BLOCKQUOTE>), а деякі – ні (<P>). Таблиці і списки тегів мають природний порядок вкладеності. Наприклад, теги <TD> з'являються тільки в обрамленні тегів <TR> і ніяк інакше. Вміст тега <SCRIPT> не братиме участі в розборі HTML. Тег <META> може визначати кодування документа. Бібліотека обчислює найбільш ймовірні місця для закриваючих тегів, навіть якщо вони відсутні у вихідному документі.

Одним із загальновідомих недоліків BeautifulStoneSoup є те, що він не знає про самозакривних тегах. У HTML є фіксований набір самозакривних тегів, але у випадку з XML усе залежить від того, що записано в DTD. Можна повідомити BeautifulStoneSoup, що певні теги є такими, що зачиняються, передавши їх імена через аргумент конструктора selfClosingTags:

```
print BeautifulStoneSoup(xml, selfClosingTags=['selfclosing']).prettify()
# <tag>
#  Text 1
#  <selfclosing />
#  Text 2
# </tag>
```

Документ Beautiful Soup можна перетворити в рядок за допомогою функції str або методів prettify або renderContents.

2.2.3 Бібліотека requests

Бібліотека requests є стандартним інструментом для складання HTTP-запитів у Python [11]. Простий і акуратний API значно полегшує трудомісткий процес створення запитів. Таким чином, можна зосередитися на взаємодії зі службами і використанні даних у додатку. Дозволяє надсилати HTTP / 1.1 запити надзвичайно легко. Не потрібно вручну додавати рядки запитів до ваших URL-адрес або формувати-кодувати ваші POST-дані. Що вміє бібліотека:

- Безліч методів http аутентифікації.
- Сесії з куками.
- Повноцінна підтримка SSL.
- Різні методи, наприклад .json (), які повернуть дані в потрібному форматі.
- Проксінг.
- Грамотна і логічна робота з винятками.

2.2.4 Бібліотека urllib

Цей модуль забезпечує інтерфейс високого рівня для отримання даних у всесвітній павутині. Зокрема, функція urlopen () схожа на вбудовану функцію open (), але приймає універсальні локатори ресурсів (URL) замість іменних файлів. Діють деякі обмеження – він може відкривати лише URL-адреси для читання, а операції пошуку не доступні. Модуль urllib був розділений на частини і перейменований в Python 3 на urllib.request, urllib.parse та urllib.error [5].

У роботі було вікористано метод бібліотеки urllib – *urllib.urlretrieve(url[, filename[, reporthook[, data[, context]]]]).* Потрібно скопіювати мережевий об’єкт, позначений URL-адресою, у локальний файл. Якщо URL-адреса вказує на локальний файл або існує дійсна кешована копія об'єкта, об'єкт не копіюється. Повернути кортеж (ім'я файлу, заголовки), де ім'я файлу – це

локальне ім'я файлу, під яким можна знайти об'єкт, а заголовки – це будь-який метод інформації (), повернутий urlopen (), повернутий (для віддаленого об'єкта, можливо, кешований). Винятки такі ж, як і для urlopen ().

Другий аргумент, якщо він присутній, вказує розташування файлу для копіювання (якщо воно відсутнє, то розташування буде тимчасовим файлом із згенерованим іменем). Третій аргумент, якщо він присутній, – це дзвінок, який буде викликаний один раз при встановленні мережевого з'єднання та один раз після кожного блоку, прочитаного далі. Виклику буде передано три аргументи; кількість блоків, переданих дотепер, розмір блоку в байтах та загальний розмір файлу. Третій аргумент може бути -1 на старих серверах FTP, які не повертають розмір файлу у відповідь на запит на пошук.

Якщо URL використовує ідентифікатор схеми http: схема, необов'язковий аргумент даних може бути наданий для вказівки POST-запиту (як правило, тип запиту GET). Аргумент даних повинен бути у стандартному застосуванні / x-www-form-urlencoded-форматі. Параметр контексту може бути встановлений на екземпляр ssl.SSLContext для налаштування параметрів SSL, які використовуються, якщо urlretrieve () здійснює з'єднання HTTPS.

Довжина вмісту трактується як нижня межа: якщо є більше даних для читання, urlretrieve () читає більше даних, але якщо є менше даних, це збільшує виняток. У цьому випадку ви все ще можете завантажити завантажені дані, вони зберігаються в атрибуті вмісту екземпляра винятку. Якщо не було надано заголовок "Довжина вмісту", urlretrieve () не може перевірити розмір завантажених даних, а просто поверне його. У цьому випадку ви просто повинні припустити, що завантаження було успішним.

2.2.5 Бібліотека logging

Цей модуль визначає функції та класи, які реалізують гнучку систему реєстрації подій для додатків та бібліотек. Ключовою перевагою наявності API журналу, що надається стандартним модулем бібліотеки, є те, що всі модулі Python можуть брати участь у реєстрації, тому журнал додатків може включати

власні повідомлення, інтегровані з повідомленнями сторонніх модулів. Модуль забезпечує багато функціональності та гнучкості [2].

Основні класи, визначені модулем, разом із їх функціями:

- Логери відкривають інтерфейс, який безпосередньо використовує код програми.
- Обробники надсилають записи журналів (створені реєстраторами) до відповідного пункту призначення.
- Фільтри забезпечують тоншу зернистість для визначення, які журнали записів слід виводити.
- Форматори задають макет записів журналів у кінцевому висновку.

Loggers не повинні створюватися миттєво безпосередньо, але завжди через функцію рівня модуля logging.getLogger (ім'я). Кілька викликів до getLogger () з тим самим іменем завжди повертатимуть посилання на той самий об’єкт Logger.

setLevel встановлює поріг для реєстратора на рівень. Повідомлення, які мають менш суворий рівень, будуть ігноруватися. Повідомлення журналу, які мають рівень вираженості або вище, будуть випромінюватись тим, хто обробник чи обробник обслуговує цей реєстратор, якщо тільки рівень обробника не буде встановлений на більш високий рівень серйозності. Коли створюється logger, рівень встановлюється на NOTSET (що спричиняє обробку всіх повідомлень, коли logger є кореневим реєстратором, або делегування до батьківського, коли журнал – це некореневий реєстратор). Термін "делегування батькові" означає, що якщо в реєстраторі є рівень NOTSET, його ланцюг реєстраторів предків проходить доти, доки не буде знайдено предка з рівнем, відмінним від NOTSET, або не буде досягнуто кореня. Якщо предка знайдено з рівнем, відмінним від NOTSET, то рівень цього предка трактується як ефективний рівень реєстратора, на якому почався пошук предка, і використовується для визначення способу обробки події реєстрації. Якщо корінь буде досягнуто, і він має рівень NOTSET, то всі повідомлення будуть

оброблені. В іншому випадку рівень кореня використовуватиметься як ефективний рівень.

Чисельні значення рівнів журналу наведені в наступній таблиці (Рис 2.3).

Level	Numeric value
CRITICAL	50
ERROR	40
WARNING	30
INFO	20
DEBUG	10
NOTSET	0

Рисунок 2.3 – Рівні журналу для логування

Об'єкти форматування відповідають за перетворення LogRecord у (зазвичай) рядок, який може бути інтерпретований або людиною, або зовнішньою системою. Базовий Форматтер дозволяє задавати рядок форматування. Якщо нічого не надається, використовується значення за замовчуванням '% (message) s', яке просто включає повідомлення в дзвінок реєстрації. Форматор можна ініціалізувати за допомогою рядка формату, який використовує знання атрибутів LogRecord – таких як значення за замовчуванням, використовуючи той факт, що повідомлення та аргументи користувача попередньо відформатовані в атрибут повідомлення LogRecord. Цей рядок формату містить стандартні клавіші відображення Python% -style.

У LogRecord є ряд атрибутів, більшість з яких походить від параметрів до конструктора.

Attribute name	Format	Description
args	You shouldn't need to format this yourself.	The tuple of arguments merged into `msg` to produce `message`, or a dict whose values are used for the merge (when there is only one argument, and it is a dictionary).
asctime	`%(asctime)s`	Human-readable time when the `LogRecord` was created. By default this is of the form '2003-07-08 16:49:45,896' (the numbers after the comma are millisecond portion of the time).
created	`%(created)f`	Time when the `LogRecord` was created (as returned by `time.time()`).
exc_info	You shouldn't need to format this yourself.	Exception tuple (à la `sys.exc_info`) or, if no exception has occurred, `None`.
filename	`%(filename)s`	Filename portion of `pathname`.
funcName	`%(funcName)s`	Name of function containing the logging call.
levelname	`%(levelname)s`	Text logging level for the message (`'DEBUG'`, `'INFO'`, `'WARNING'`, `'ERROR'`, `'CRITICAL'`).
levelno	`%(levelno)s`	Numeric logging level for the message (`DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`).
lineno	`%(lineno)d`	Source line number where the logging call was issued (if available).
message	`%(message)s`	The logged message, computed as `msg % args`. This is set when `Formatter.format()` is invoked.
module	`%(module)s`	Module (name portion of `filename`).
msecs	`%(msecs)d`	Millisecond portion of the time when the `LogRecord` was created.
msg	You shouldn't need to format this yourself.	The format string passed in the original logging call. Merged with `args` to produce `message`, or an arbitrary object (see Using arbitrary objects as messages).
name	`%(name)s`	Name of the logger used to log the call.
pathname	`%(pathname)s`	Full pathname of the source file where the logging call was issued (if available).
process	`%(process)d`	Process ID (if available).
processName	`%(processName)s`	Process name (if available).
relativeCreated	`%(relativeCreated)d`	Time in milliseconds when the LogRecord was created, relative to the time the logging module was loaded.
stack_info	You shouldn't need to format this yourself.	Stack frame information (where available) from the bottom of the stack in the current thread, up to and including the stack frame of the logging call which resulted in the creation of this record.
thread	`%(thread)d`	Thread ID (if available).
threadName	`%(threadName)s`	Thread name (if available).

Рис 2.4 – Ряд атрибутів у LogRecord

Атрибути (Рис2.4) можна використовувати для об'єднання даних із запису в рядок формату. У таблиці подано (в алфавітному порядку) назви атрибутів, їх значення та відповідний заповнювач у рядку формату% -style.

Якщо використовувати {}-форматування (str.format ()), можна використовувати {attrname} як заповнювач у рядку формату. Якщо використовувати $ -форматування (string.Template), використовуйте форму $ {attrname}.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОЕКТУ

3.1 Як все буде працювати

На комп'ютері працює інтерпретатор Python, а всередині інтерпретатора крутиться програма на Python. Вона відповідає за весь контент: в неї закладені всі шаблони тексту, вся логіка, вся поведінка. Усередині програми на Python працює бібліотека, яка відповідає за спілкування із сервером Телеграма. У бібліотеку ми вшили секретний ключ, щоб сервер Телеграма розумів, що наша програма пов'язана з певним ботом. Коли клієнт з Телеграма робить запит, запит приходить на сервер, а сервер відправляє його на наш комп'ютер. Запит обробляється програмою на Python, відповідь йде на сервер Телеграма, сервер віддає відповідь клієнту (Рис 3.1).

Рисунок 3.1 – Діаграма взаємодії клієнт-сервер

Бот буде працювати тільки тоді, коли включений комп'ютер і на ньому запущена програма на Python. Якщо комп'ютер вимкнеться, пропаде інтернет або буде вимкнутий інтерпретатор, то бот працювати перестане: запити будуть приходити, але ніхто на них не відповість.

Спочатку здійснюється налаштування зі сторони самого Телеграма. Після того, як ми зареєстрували свого бота, потрібно дати йому опис,

зображення та стандартні команди для вікористання.

Було реалізовано дві команди за допомогою /setcommands в интерфейсі BotFather (Рис 3.2):

- start - start Bot
- help - description

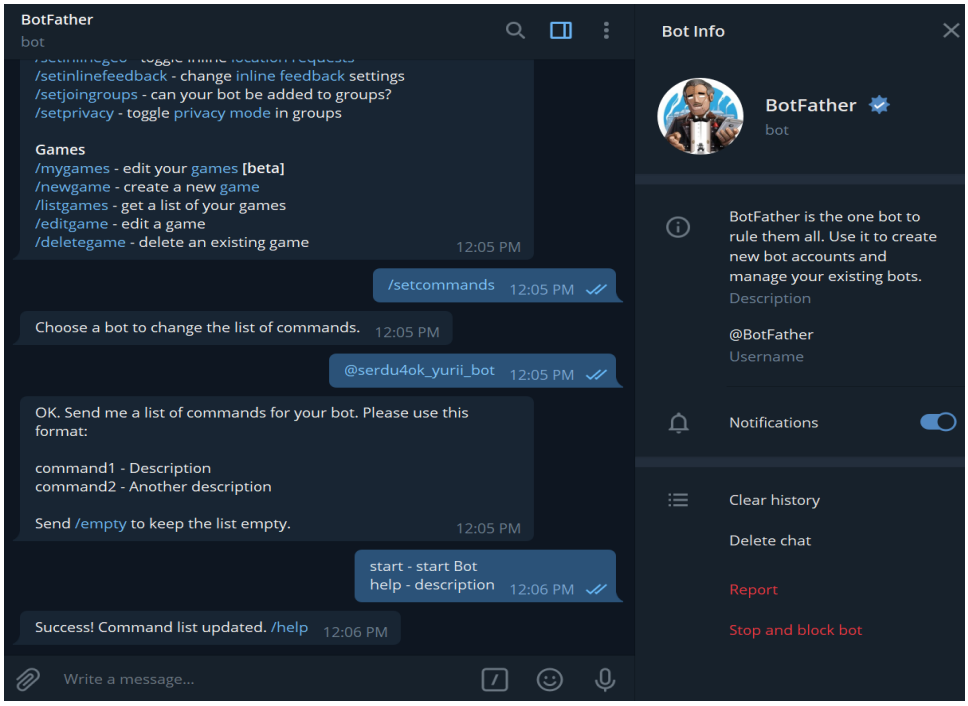

Рисунок 3.2 – Встановлення команд для бота у інтерфейсі BotFather

Також додано опис бота за допомогою команди /setdescription (Рис 3.2): A simple bot to search for any pictures you want

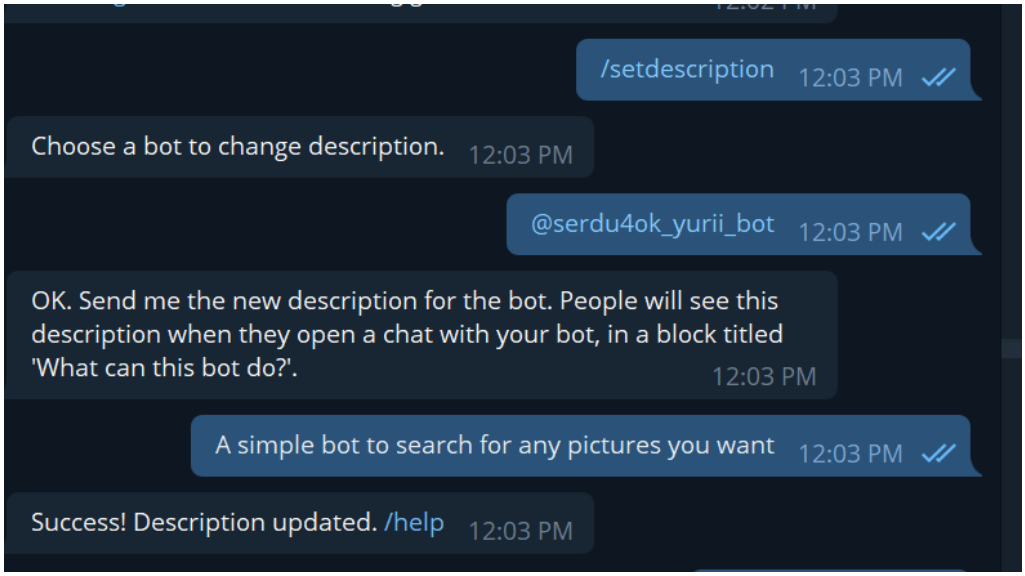

Рисунок 3.2 – Добавання опису для бота у інтерфейсі BotFather

Необхідна умова для бота це встановлення зображення за допомогою команди /setuserpic (Рис 3.3) [13].

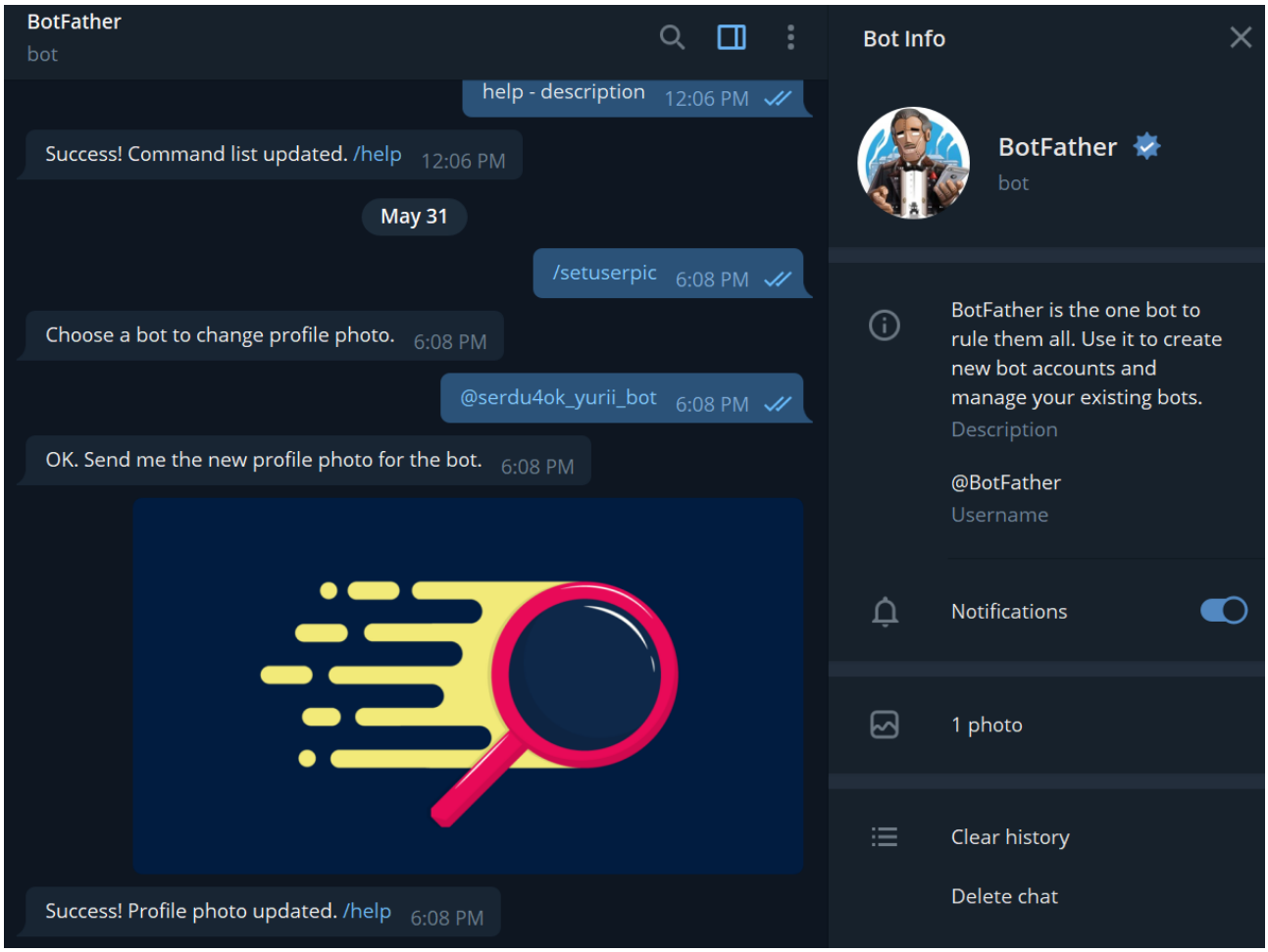

Рисунок 3.3 – Встановлення зображення для бота у інтерфейсі BotFather

Тепер бот має команди, зображення яке будуть бачити користувачі, та опис при першому вікористанні.

3.2 Програмна реалізація

Для того, щоб програма на Python вміла управляти телеграм-ботами, необхідно на самому початоку коду додати рядок (Рис 3.4) [6]:

```
bot = telebot.TeleBot('911708696:AAHPZU4JB2oshlOTx-IET3D0ZSPavE-Q2PA')
```

Рисунок 3.4 – Встановлення токену для взаємодії програми з телеграм ботом

Після цієї декларації потрібно зареєструвати кілька так званих обробників повідомлень. Обробники повідомлень визначають фільтри, через які повідомлення повинно пройти. Якщо повідомлення передає фільтр,

викликається оформлена функція, а вхідне повідомлення передається як аргумент.

Було створено три обробника повідомлень від користувачаю [10]. Це функція, прикрашена декоратором message_handler екземпляру TeleBot. Обробники повідомлень складаються з одного або декількох фільтрів. Кожен фільтр повертає значення True для певного повідомлення, щоб обробник повідомлень мав право на обробку цього повідомлення.

Перший обробник – команда / start, повідомлення в якому відбувається привітання та старт бота.

```
@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, "Добро пожаловать, {0.first_name}!\nЯ - {1.first_name}, бот созданный чтобы быть помогать искать картинки, отправьь что ты хочешь найти.".format(message.from_user, bot.get_me()))
```

Другий обробник – команда / help, повідомлення користувачу в якому повідомляється принцип роботи доступною мовою.

```
@bot.message_handler(commands=['help'])
def help_message(message):
    bot.send_message(message.chat.id, "Этот бот создан для поиска любых картинок. Просто напиши /start и то что хочешь найти\nИ после укажи количество картинок которые ты хочешь что бы я тебе отправил".format(message.from_user, bot.get_me()))
```

Третій обробник призначений для аналізу вхідного повідомлення від користувача, та обробку основного функціоналу.

```
@bot.message_handler(content_types=['text'])
def lalala(message):
    if message.chat.type == 'private':
```

Після входу в интерфейс бота та використання команд користувач робить запит на пошук зображення. У ході дослідження було встановленно що пошук рисунків в системі Google здійснюється за певним шаблоном, який буде використаний:

'https://www.google.co.in/search?q='+ word+'&source=lnms&tbm=isch'

У програмі *Word* буде дорівнювати тому що користувач надішле боту. Тобто його запит на зображення. Чи то будет одне слово, фраза, чи смайлик. Після того як бот передав слово, программа за допомогою методу get бібліотеки requests отримає веб сторінку.

```
r = requests.get('https://www.google.co.in/search?q='+ word+'&source=lnms&tbm=isch')
```

Після чого було отримано **Response** об’єкт названий r. Щоб далі обробляти дані, об’єкт r перетворено в `r.text` та відпаршено за допомою бібліотеки bs4 в тип `BeautifulSoup` для подальшого пошуку тегів зображень.

```
soup = BeautifulSoup(r.text, 'html.parser') # parce request to suop
```

Бібліотека надає можливість здійснити пошук за тегами та додатковими атрибутами. У ході досліджень було встановлено, що в будь якій HTML сторінці зображення знаходяться під тегом **< img >** та з конкретним класом **class = 't0fcAb'**

```
results = str (soup.find_all('img', attrs = {'class' : 't0fcAb'})) #find image and convert to str
```

Далі відбувається процес завантаження (Рис 3.5) зображень на локальне середовище за домомогою методу *urlretrieveurlretrieve* де було вказано в якому форматі проводити завантаження: шлях та тип файлу.

```
for  i in range (1,6):
    url =  str(b[i])[:-5] # convert list to string for download
    print(url)
    w = str(i)
    full_path = 'images/' + w + '.jpg'
    urlretrieve(url,full_path)
```

Рисунок 3.5 – Програмна реалізація завантаження рисунків

Після заданого алгоритму бот повертає користувачу три кнопки (Рис 3.6), які надають зробити вибір скільки зображень відправити.

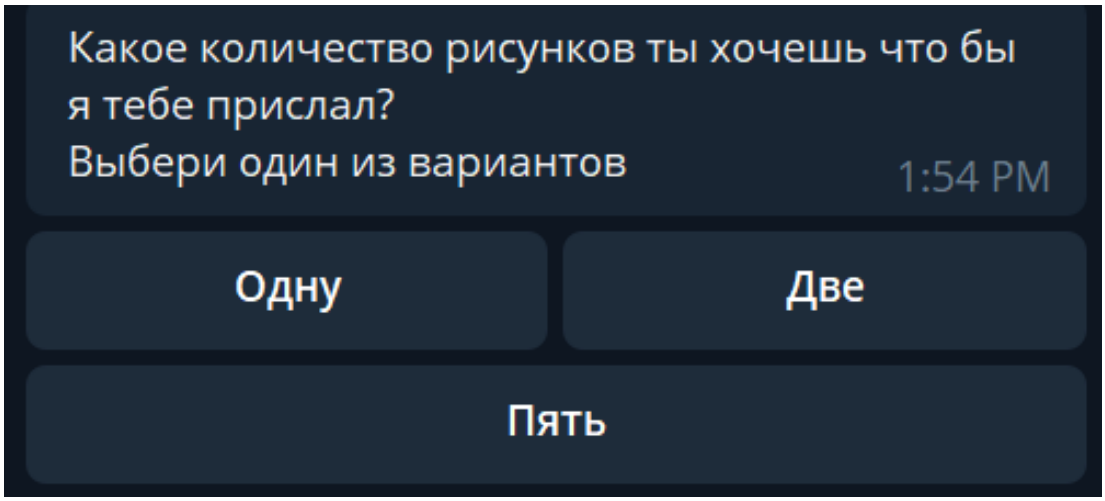

Рисунок 3.6 – Інтерфейс користувача при доставленні кнопок

Кнопки реалізовані за допомогою *InlineKeyboardMarkup.* Який має два параметра, назву кнопки та callback_data. Другий параметр буде передано далі, після натискання кнопки користувачем.

```
markup = types.InlineKeyboardMarkup(row_width=2)
            item1 = types.InlineKeyboardButton("Одну",
callback_data='one')
            item2 = types.InlineKeyboardButton("Две",
callback_data='two')
            item3 = types.InlineKeyboardButton("Пять",
callback_data='five')

            markup.add(item1, item2, item3)
            bot.send_message(message.chat.id,'Какое
количество рисунков ты хочешь что бы я тебе
```

```
прислал?\nВыбери один из вариантов',
reply_markup=markup)
```

Відправлення зображень відбувається за допомогою функціїї callback_inline(call), яка надсилає ту кількість яку було запрошено: одну, дві, чи п'ять.

```
if call.data == 'one':
                bot.send_message(call.message.chat.id,
'Держи одину картинку')
                uis_pdf = open('images/' +  '1.jpg',
'rb')
                bot.send_photo(call.message.chat.id,
uis_pdf)
                uis_pdf.close()
            elif call.data == 'two':
                bot.send_message(call.message.chat.id,
'Держи две картинки')
                uis_pdf = open('images/' +  '1.jpg',
'rb')
                bot.send_photo(call.message.chat.id,
uis_pdf)
                uis_pdf.close()
                uis_pdf = open('images/' +  '2.jpg',
'rb')
                bot.send_photo(call.message.chat.id,
uis_pdf)
                uis_pdf.close()
            elif call.data == 'five':
                bot.send_message(call.message.chat.id,
'Держи держи пять картинок')
                for i in range(1,6):
```

```
                    q = str(i)
                    uis_pdf = open('images/' + q +
'.jpg', 'rb')
                    bot.send_photo(call.message.chat.id,
uis_pdf)
                    uis_pdf.close()
```

Щоб кнопки далі не заважали, відразу після доставки вони зникають, за подомогою реалізаціїї *edit_message_text*, що дозволяє прибрати їх. Також реалізовано *answer_callback_query*, що повідомляє про успішну доставку рисунків (Рис 3.7).

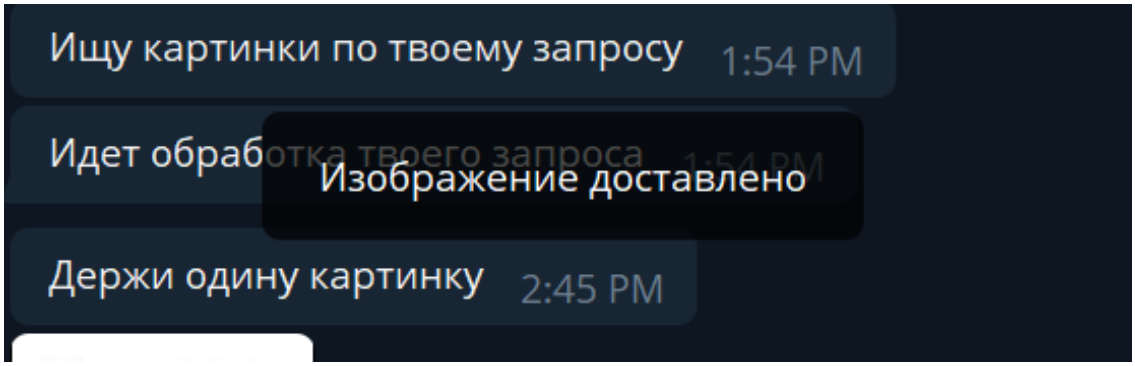

Рисунок 3.7 – Інтерфейс користувача при доставці рисунків

Завдяки реалізації логування, можно легко відслудкувати роботу бота, а саме:

- точний час запитів (Рис 3.8);

```
%D0%B5%D0%BC%D1%83+%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D1%83 HTTP/1.1" 200 427
[2020-05-29 17:07:44,547 INFO connectionpool]: Starting new HTTPS connection (1): www.google.co.in
[2020-05-29 17:07:45,163 DEBUG connectionpool]: "GET /search?q=%D0%BD%D0%B0%D1%81%D1%82%D1%8F&source=lnms&tbm=isch HTTP/1.1" 200
None
[2020-05-29 17:07:47,824 DEBUG connectionpool]: "POST /bot911708696:AAHPZU4JB2oshlOTx-IET3D0ZSPavE-Q2PA/sendMessage?
chat_id=347679758&text=%D0%9A%D0%B0%D0%BA%D0%BE%D0%B5+%D0%BA%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE+
%D1%80%D0%B8%D1%81%D1%83%D0%BD%D0%BA%D0%BE%D0%B2+%D1%82%D1%8B+%D1%85%D0%BE%D1%87%D0%B5%D1%88%D1%8C+%D1%87%D1%82%D0%BE+
%D0%B1%D1%8B+%D1%8F+%D1%82%D0%B5%D0%B1%D0%B5+%D0%BF%D1%80%D0%B8%D1%81%D0%BB%D0%B0%D0%BB%3F%0A%D0%92%D1%8B
%D0%B1%D0%B5%D1%80%D0%B8+%D0%BE%D0%B4%D0%B8%D0%BD+%D0%B8%D0%B7+%D0%B2%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82%D0%BE
%D0%B2&reply_markup=%7B%22inline_keyboard%22%3A+%5B%5B%7B%22text%22%3A+%22%5Cu041e%5Cu0434%5Cu043d%22%2C+%22callback_data%22%3A+
%22one%22%7D%2C+%7B%22text%22%3A+%22%5Cu0414%5Cu0432%5Cu0430%22%2C+%22callback_data%22%3A+%22two%22%7D%5D%2C+%5B%7B%22text%22%3A
+%22%5Cu041f%5Cu044f%5Cu0442%5Cu044c%22%2C+%22callback_data%22%3A+%22five%22%7D%5D%5D%7D HTTP/1.1" 200 900
[2020-05-29 17:07:49,670 DEBUG connectionpool]: "GET /bot911708696:AAHPZU4JB2oshlOTx-IET3D0ZSPavE-Q2PA/getUpdates?
offset=568022342&timeout=20 HTTP/1.1" 200 1160
```

Рисунок 3.8 – Відображення часу запитів в логах

- успішне доставлення зображень (Рис 3.9);

```
2020-05-30 12:40:15,353 DEBUG connectionpool]: "POST /bot911708696:AAHPZU4JB2oshlOTx-IET3D0ZSPavE-Q2PA/editMessageText?text=
D0%98%D0%B4%D0%B5%D1%82+%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0+%D1%82%D0%B2%D0%BE%D0%B5%D0%B3%D0%BE+
D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B0&chat_id=347679758&message_id=362 HTTP/1.1" 200 449
2020-05-30 12:40:15,416 DEBUG connectionpool]: "POST /bot911708696:AAHPZU4JB2oshlOTx-IET3D0ZSPavE-Q2PA/answerCallbackQuery?
allback_query_id=1493273191079794431&text=%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5+%D0%B4%D0%BE
D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%BE HTTP/1.1" 200 25
```

Рисунок 3.9 – Відображення доставлення зображень в логах

- причину падіння програми (Рис 3.10);

```
Traceback (most recent call last):
  File "C:\Users\yuse0619\AppData\Local\Programs\Python\Python38-32\lib\site-packages\requests\adapters.py", line 360, in send
    resp = conn.urlopen(
  File "C:\Users\yuse0619\AppData\Local\Programs\Python\Python38-32\lib\site-packages\requests\packages
\urllib3\connectionpool.py", line 596, in urlopen
    retries = retries.increment(method, url, error=e, _pool=self,
  File "C:\Users\yuse0619\AppData\Local\Programs\Python\Python38-32\lib\site-packages\requests\packages\urllib3\util\retry.py",
line 245, in increment
    raise six.reraise(type(error), error, _stacktrace)
  File "C:\Users\yuse0619\AppData\Local\Programs\Python\Python38-32\lib\site-packages\requests\packages\urllib3\packages\six.py",
line 309, in reraise
    raise value.with_traceback(tb)
  File "C:\Users\yuse0619\AppData\Local\Programs\Python\Python38-32\lib\site-packages\requests\packages
\urllib3\connectionpool.py", line 542, in urlopen
    httplib_response = self._make_request(conn, method, url,
  File "C:\Users\yuse0619\AppData\Local\Programs\Python\Python38-32\lib\site-packages\requests\packages
\urllib3\connectionpool.py", line 374, in _make_request
    httplib_response = conn.getresponse()
```

Рисунок 3.10 – Відображення падіння програми в логах

- втрати зєднання з інтернетом чи пошуковим сервером (Рис 3.11);

```
offset=568022338&timeout=20 HTTP/1.1" 200 23
[2020-05-28 20:15:05,589 ERROR util]: ConnectionError occurred, args=(ProtocolError('Connection aborted.', ConnectionResetError
(10054, 'An existing connection was forcibly closed by the remote host', None, 10054, None)),)
Traceback (most recent call last):
  File "C:\Users\yuse0619\AppData\Local\Programs\Python\Python38-32\lib\site-packages\requests\packages
\urllib3\connectionpool.py", line 372, in _make_request
    httplib_response = conn.getresponse(buffering=True)
TypeError: getresponse() got an unexpected keyword argument 'buffering'

During handling of the above exception, another exception occurred:
```

Рисунок 3.11 – Відображення втратіи зєднання в логах

3.3 Тестування

3.3.1 Тестування Telegram Desktop

Тестування проведено в Telegram Desktop версіїї 2.1.6. Відразу після входу до бота, можна побачити його опис та зображення, які булі додані раніше (Рис 3.12).

Рисунок 3.12 – Привітання з користувачем при першому запуску бота

Перевірено додані команди /start і /help та їх працездатність. Бот вітається з кожним користувачем особисто (Рис 3.13).

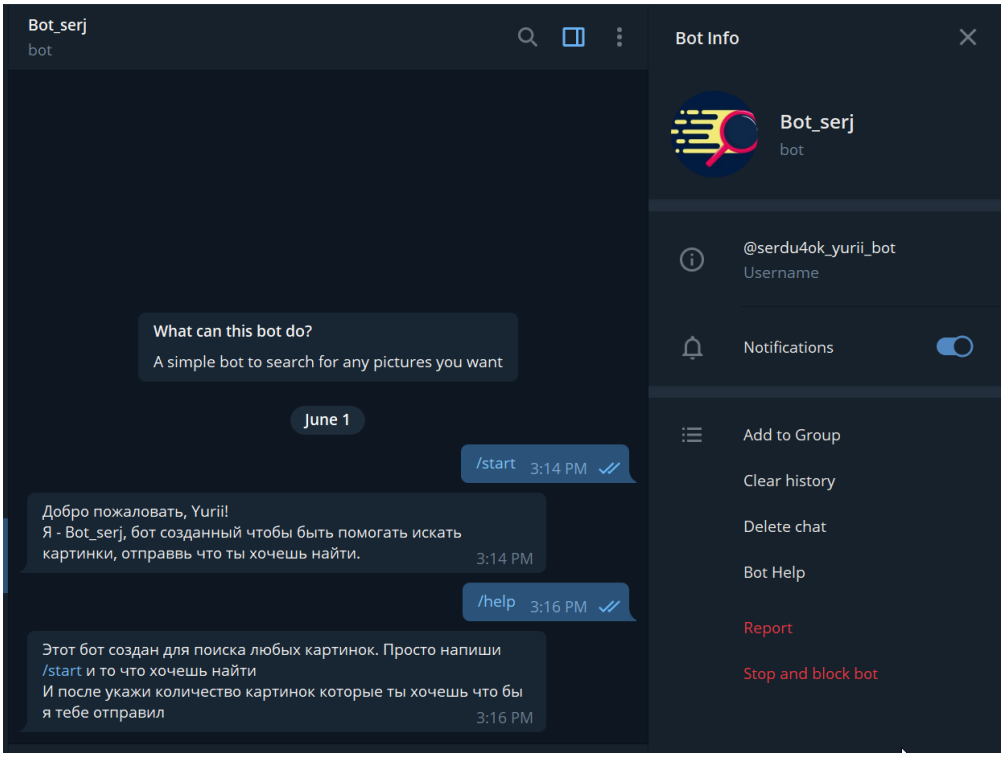

Рисунок 3.13 – Тестування роботи кнопок /start та /help

Перевірено здатність обробляти запити користувача і як бот відповідає на них (Рис 3.14).

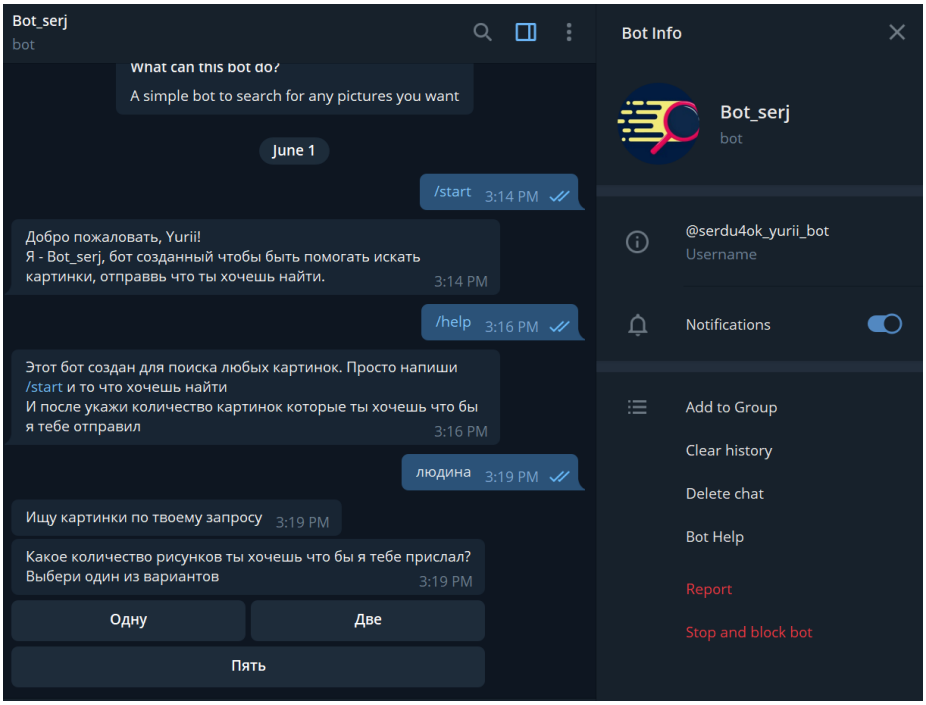

Рисунок 3.14 – Тестування роботи запиту на рисунок

Як можна побачити бот успішно опрацював запит і готовий надіслати зображеня. Обираємо «два» і дивимось на результат (Рис 3.15):

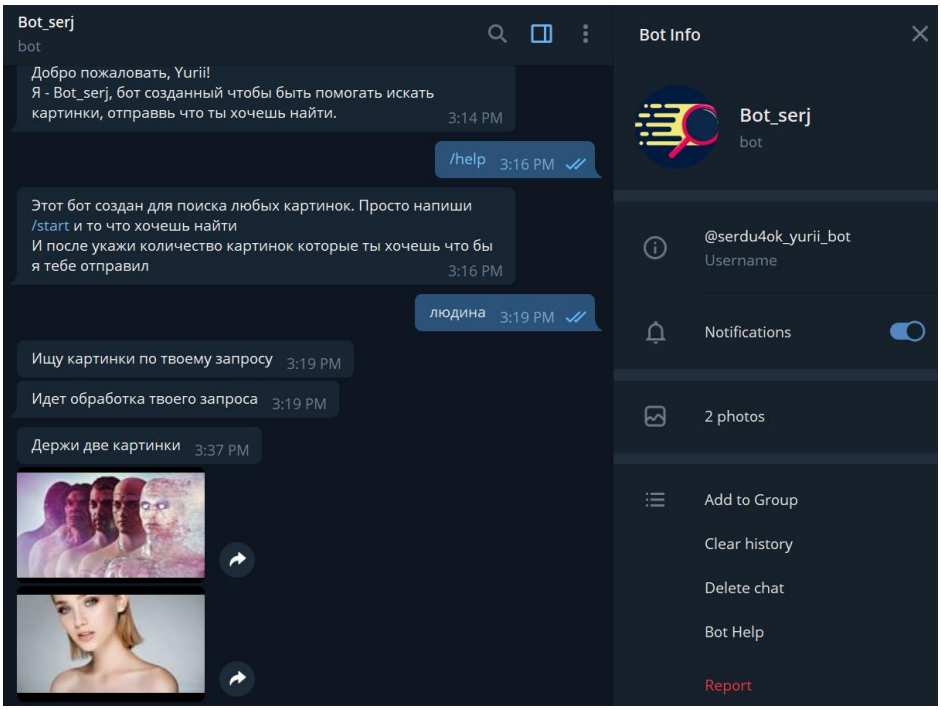

Рисунок 3.15 – Тестування роботи доставлення рисунків ботом

Так як бот робить пошук в Google, зробивши той же самий запит в пошуковій системі можна побачити ці самі зображення (Рис 3.16)

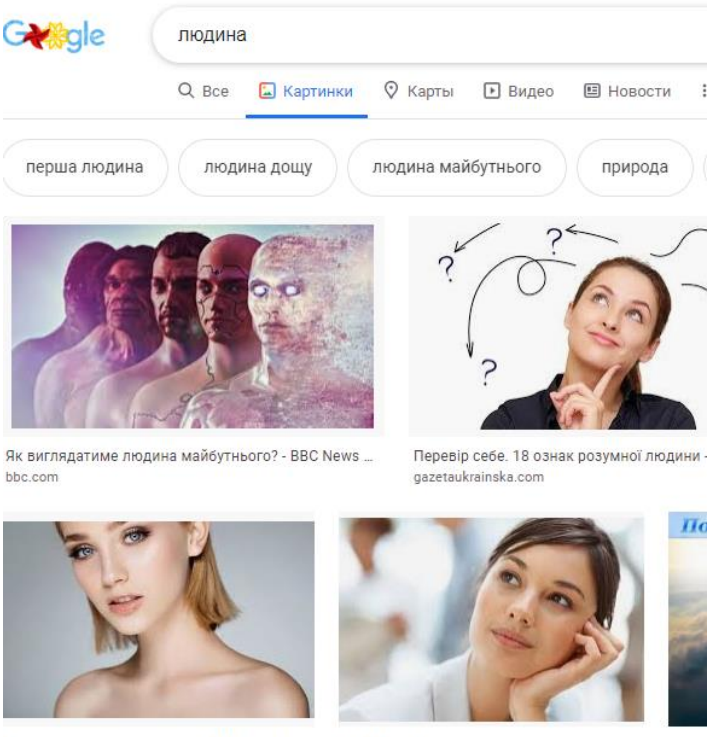

Рисунок 3.16 – Перевірка схожості рисунків з пошуком в інтернеті

Перевірено працездатність кнопки на одне зображення (Рис 3.17)

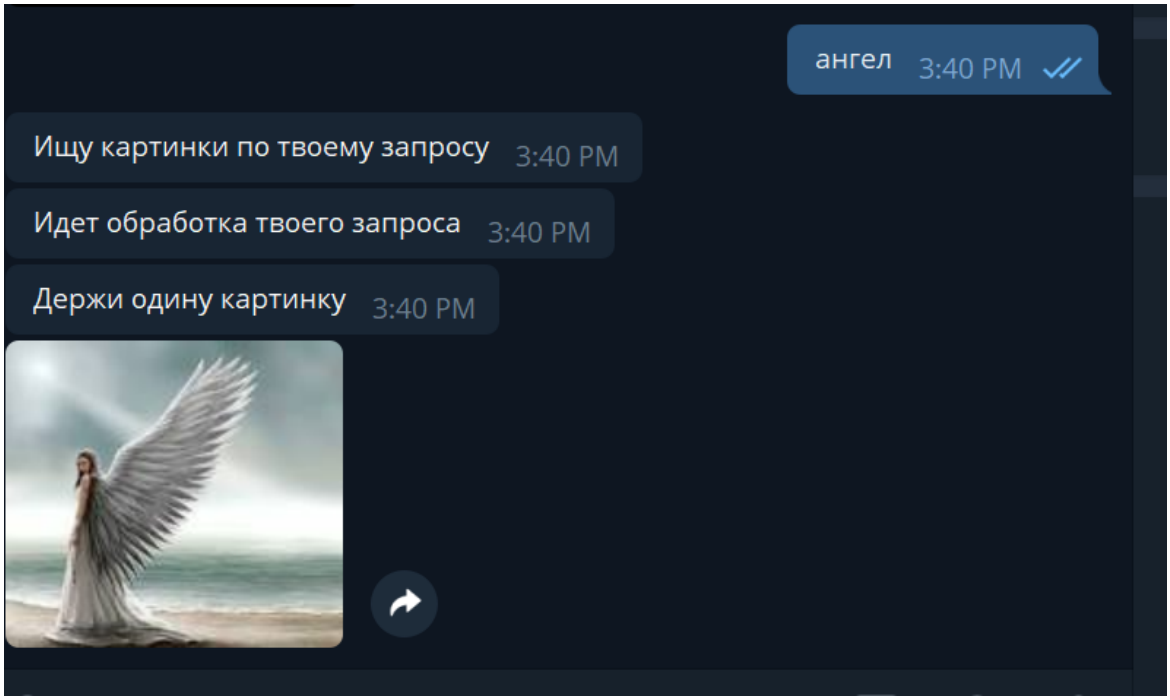

Рисунок 3.17 – Тестування роботи доставлення одного рисунка ботом

Та на всі доступні п'ять (Рис 3.18)

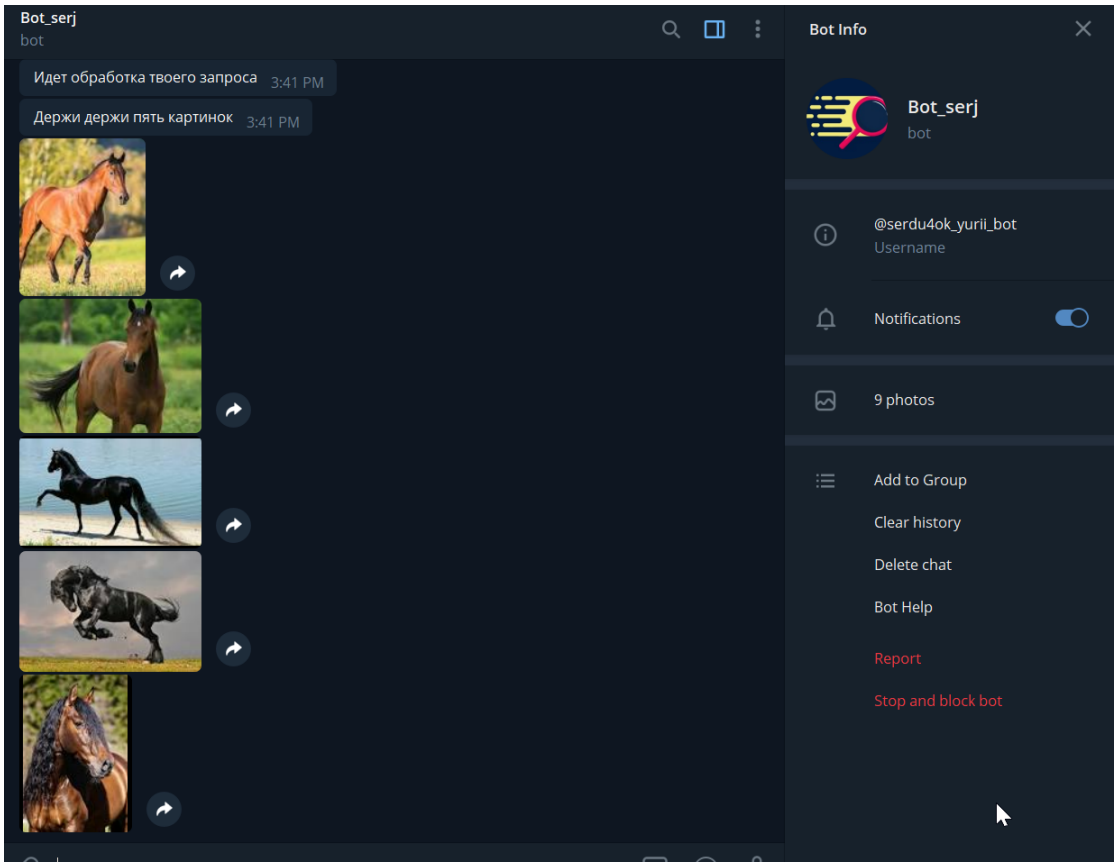

Рисунок 3.18 – Тестування роботи доставлення п’яти рисунків ботом

Проведено тест на розуміння програми фраз з декількох слів(Рис 3.19)

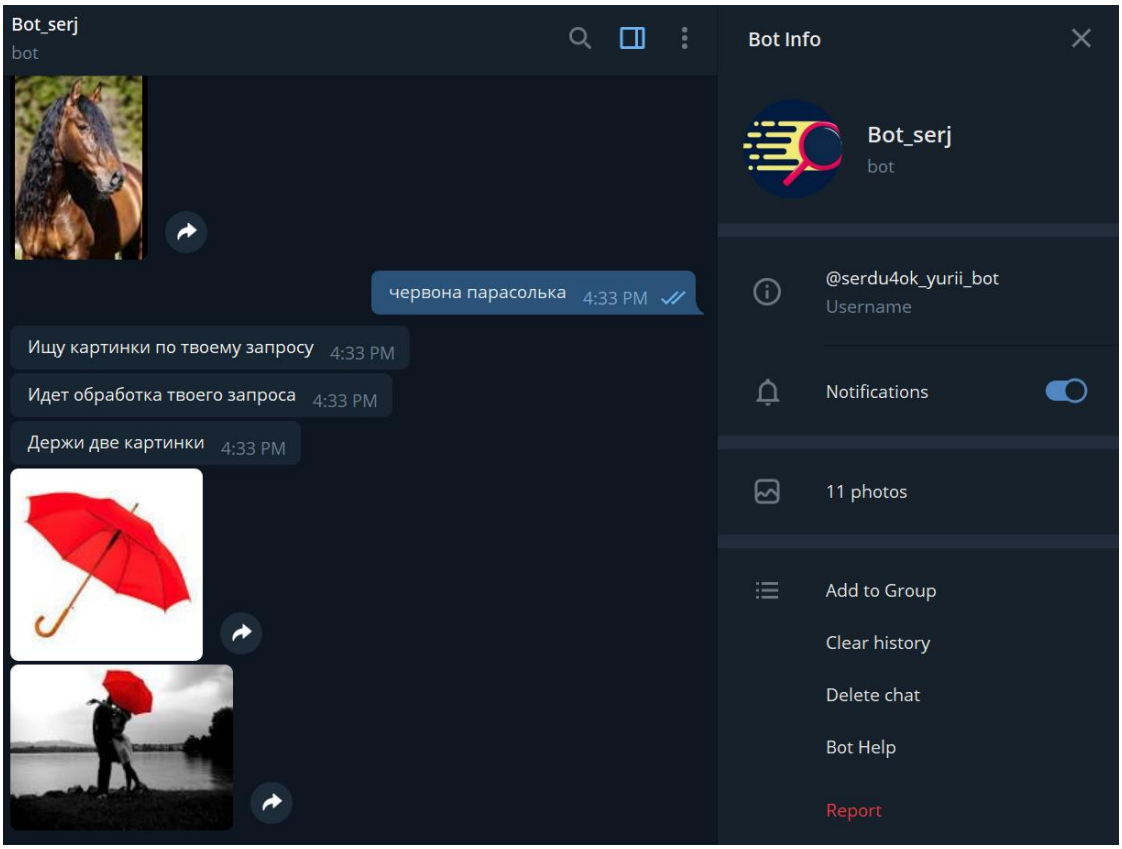

Рисунок 3.19 – Тестування роботи бота з фразами з декількох слів

Та на різних мовах (Рис 3.20)

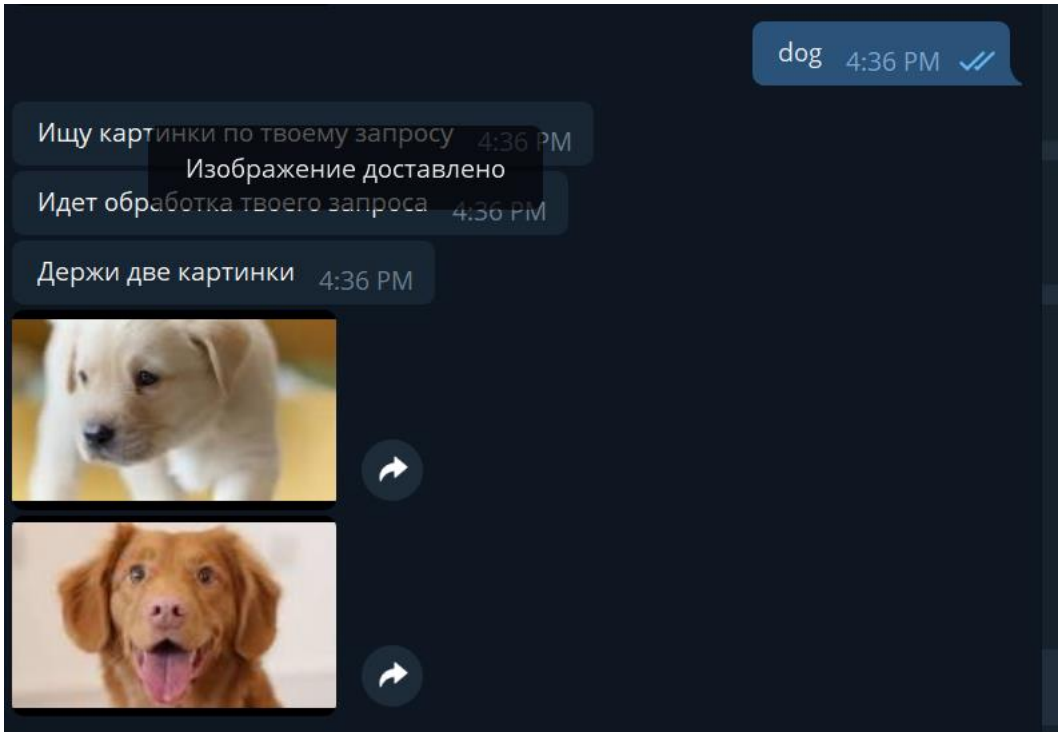

Рисунок 3.20 – Тестування роботи на англійській мові

Проведена перевірка розуміння ботом смайликів(Рис 3.21)

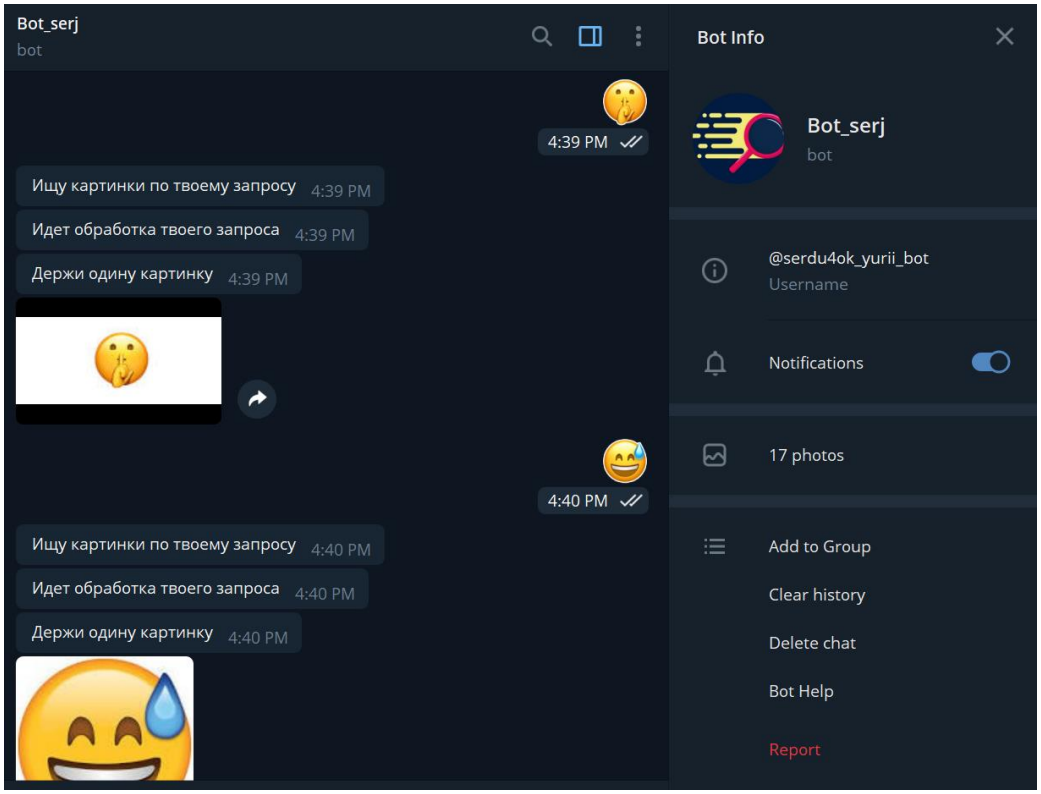

Рисунок 3.21 – Тестування роботи бота на роботу з смайлами

3.3.2 Тестування Telegram Mobile (Android)

Також проведено тестування різними користувачами на різних Мобільних платформах (Рис 3.22).

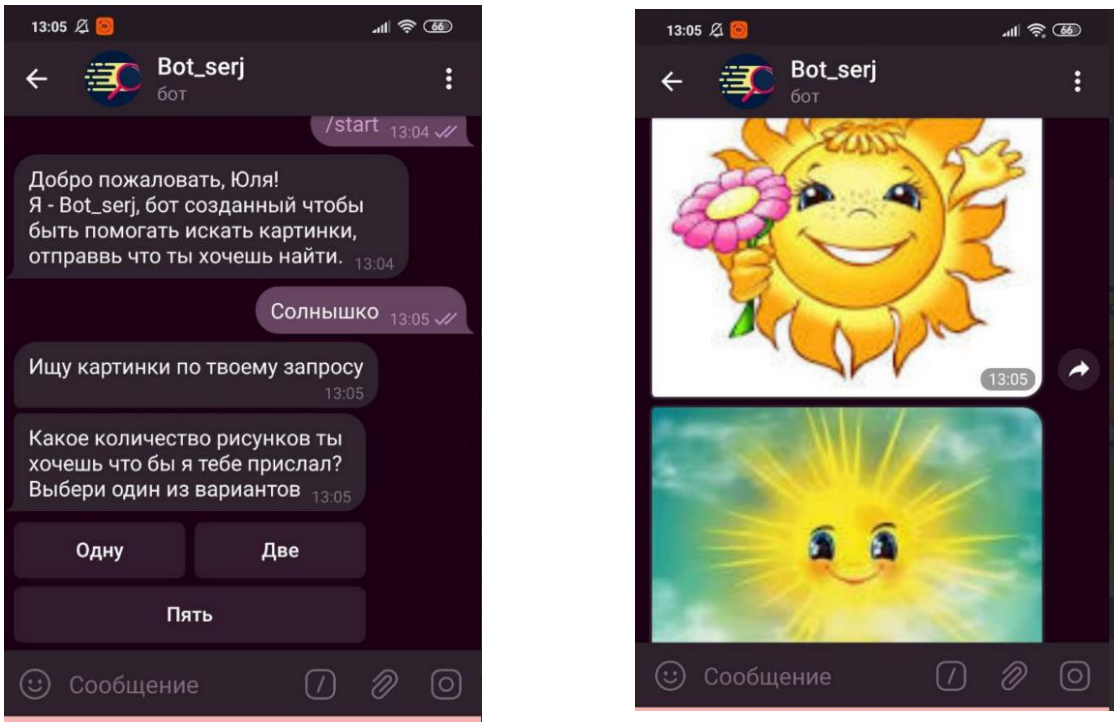

Рисунок 3.22 – Тестування роботи бота на платформі Android

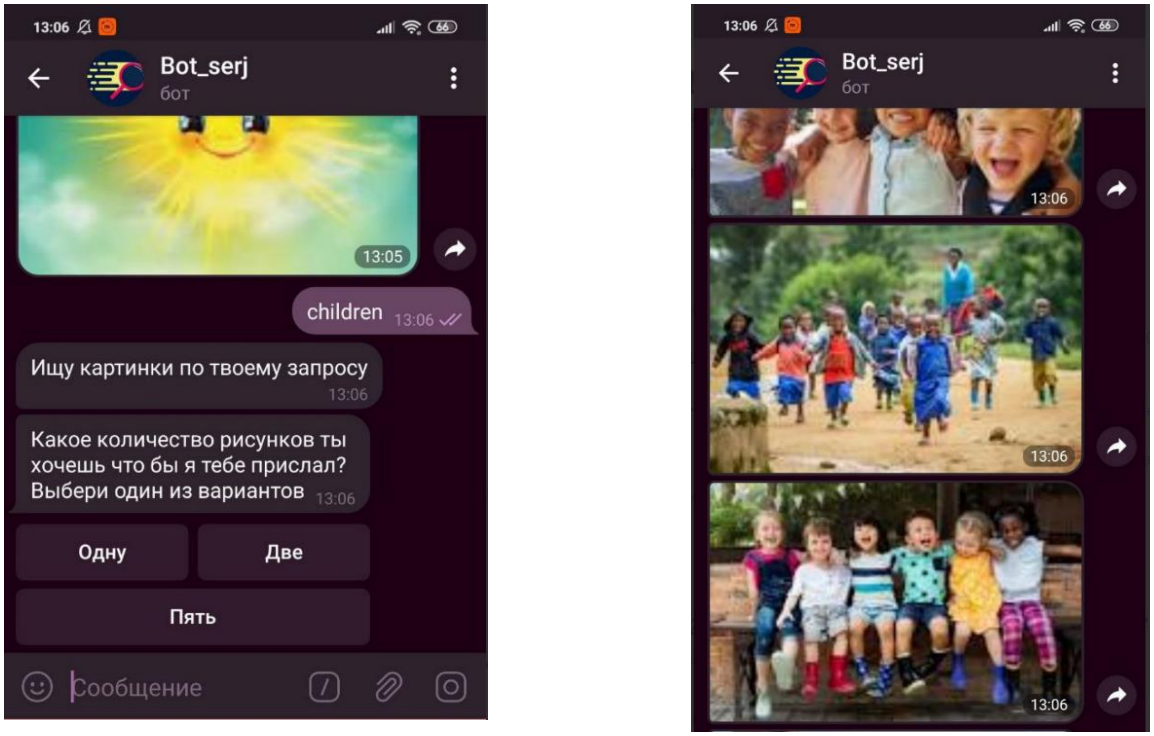

Рисунок 3.23 – Тестування роботи бота на платформі Android на англійськькій мові

3.3.3 Тестування Telegram Mobile(IPhone)

Тестування на платформі Iphone

Рисунок 3.23 –Тестування роботи бота на платформі IPhone

ВИСНОВКИ

У ході роботи було розроблено телеграм бота який здійснює пошук рисунків за запитом користувача, і повертає в відповідь таку кількість яку обирае користувач. Підтримує всі мови, фрази з декількох слів, та смайли. Здійснює пошук за допомогою API в Google.

Проведено аналіз за допомогою якого Google здійснює пошук зображень в інтернеті, і на основі цього здійснюється пошук. Проведено тестування на декількох платформах, а саме:

- Telegram Desktop
- Telegram Mobile(Android)
- Telegram Mobile(IPhone)

СПИСОК ЛІТЕРАТУРИ

1. A Byte of Python – Russian, 26. – 158 с. – (Sphinx).
2. logging – Logging facility for Python [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/logging.html.
3. Телеграм [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Telegram.
4. Боты: информация для разработчиков [Електронний ресурс] – Режим доступу до ресурсу: https://tlgrm.ru/docs/bots.
5. urllib.request – Extensible library for opening URLs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/urllib.request.html.
6. pyTelegramBotAPI library [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/pyTelegramBotAPI/.
7. BeautifulSoup library [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/BeautifulSoup/.
8. Beautiful Soup documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.crummy.com/software/BeautifulSoup/bs3/.
9. 15 TELEGRAM-БОТОВ [Електронний ресурс] – Режим доступу до ресурсу: https://grintern.ru/blog/141-15-telegram-botov-kotoryh-vy-polyubite.
10. Телеграм-бот на Python [Електронний ресурс] – Режим доступу до ресурсу: https://thecode.media/python-bot/.
11. Make a Request [Електронний ресурс] – Режим доступу до ресурсу: https://requests.readthedocs.io/en/master/user/quickstart/#passing-parameters-in-urls.
12. pyTelegramBotAPI documentation [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/eternnoir/pyTelegramBotAPI.
13. Зображення бота [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/search?q=search&rlz=1C1GCEA_enUA887UA891

&sxsrf=ALeKk01D3OtzCXUDmMy3s_wO5_9Nr52UDQ:1590937697651&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiR3M_isN7pAhUki8MKHS1NBAEQ_AUoAXoECBMQAw&biw=1280&bih=881#imgrc=Xx5ZUR9QXxhHBM.

14. Telegram бот на PYTHON. Практические примеры [Електронний ресурс] – Режим доступу до ресурсу: https://winkomp.ru/telegram-bot-python-pytelegrambotapi.
15. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio.

ДОДАТОК

```
import telebot
from telebot import types
from urllib.request import  urlretrieve
import requests
import logging
from bs4 import BeautifulSoup


def configure_logging():
    logger = logging.getLogger()
    logger.setLevel(logging.DEBUG)
    handler = logging.StreamHandler()
    handler.setFormatter(
        logging.Formatter('[%(asctime)s %(levelname)s %(module)s]:
%(message)s'))
    logger.addHandler(handler)
    Filehandler =
logging.FileHandler(r'C:\Users\yuse0619\D\Diplom\video\youtube\log.txt') #Path
to your LOG FILE.
    Filehandler.setFormatter(
        logging.Formatter('[%(asctime)s %(levelname)s %(module)s]:
%(message)s'))
    logger.addHandler(Filehandler)

    return logger

logger = configure_logging()


bot = telebot.TeleBot(TOKEN)

@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, "Добро пожаловать, {0.first_name}!\nЯ -
{1.first_name}, бот созданный чтобы быть помогать искать картинки, отправвь что
ты хочешь найти.".format(message.from_user, bot.get_me()))

@bot.message_handler(commands=['help'])
def help_message(message):
```

```
    bot.send_message(message.chat.id, "Этот бот создан для поиска любых
картинок. Просто напиши /start и то что хочешь найти\nИ после укажи количество
картинок которые ты хочешь что бы я тебе отправил".format(message.from_user,
bot.get_me()))


@bot.message_handler(content_types=['text'])
def lalala(message):
    if message.chat.type == 'private':

            word = message.text

            print('Dowload   ' + word+ '   has been started')
            bot.send_message(message.chat.id, 'Ищу картинки по твоему запросу')
            r =  requests.get('https://www.google.co.in/search?q='+
word+'&source=lnms&tbm=isch')
            soup = BeautifulSoup(r.text, 'html.parser') # parce request to suop
            results = str (soup.find_all('img', attrs = {'class' : 't0fcAb'}))
#find image and convert to str
            b =  results.split('<img alt="" class="t0fcAb" src="')  # split for
list
            del b[0]

            for  i in range (1,6):
                url =  str(b[i])[:-5] # convert list to string for download
                print(url)
                w = str(i)
                full_path = 'images/' + w + '.jpg'
                urlretrieve(url,full_path)

            markup = types.InlineKeyboardMarkup(row_width=2)
            item1 = types.InlineKeyboardButton("Одну", callback_data='one')
            item2 = types.InlineKeyboardButton("Две", callback_data='two')
            item3 = types.InlineKeyboardButton("Пять", callback_data='five')

            markup.add(item1, item2, item3)
            bot.send_message(message.chat.id,'Какое количество рисунков ты
хочешь что бы я тебе прислал?\nВыбери один из вариантов',  reply_markup=markup)


@bot.callback_query_handler(func=lambda call: True)
def callback_inline(call):
```

```
    try:
        if call.message:
            if call.data == 'one':
                bot.send_message(call.message.chat.id, 'Держи одину картинку')
                uis_pdf = open('images/' +  '1.jpg', 'rb')
                bot.send_photo(call.message.chat.id, uis_pdf)
                uis_pdf.close()
            elif call.data == 'two':
                bot.send_message(call.message.chat.id, 'Держи две картинки')
                uis_pdf = open('images/' +  '1.jpg', 'rb')
                bot.send_photo(call.message.chat.id, uis_pdf)
                uis_pdf.close()
                uis_pdf = open('images/' +  '2.jpg', 'rb')
                bot.send_photo(call.message.chat.id, uis_pdf)
                uis_pdf.close()
            elif call.data == 'five':
                bot.send_message(call.message.chat.id, 'Держи держи пять
картинок')
                for i in range(1,6):
                    q = str(i)
                    uis_pdf = open('images/' + q + '.jpg', 'rb')
                    bot.send_photo(call.message.chat.id, uis_pdf)
                    uis_pdf.close()
            # remove inline buttons
            bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id, text="Идет обработка твоего запроса",
                reply_markup=None)

            # show alert
            bot.answer_callback_query(callback_query_id=call.id,
show_alert=False,
                text="Изображение доставлено")

    except Exception as e:
        print(repr(e))

bot.polling()
```