КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня магістра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

РОЗРОБКА ІНТЕЛЕКТУАЛЬНИХ ЧАТ-БОТІВ ДЛЯ РІЗНИХ МЕСЕНДЖЕРІВ З ВИКОРИСТАННЯМ СЕРВІСІВ OPENAI

Виконав студент 2-го курсу магістратури
Станіслав ДЗУНДЗА

(підпис)

Науковий керівник:

асистент, кандидат фіз.-мат. наук
Костянтин ЖЕРЕБ

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«__» травня 2023р.,
протокол №
Завідувач кафедри
О. ПРОВОТАР _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 80 сторінок, 35 ілюстрацій, 46 джерел посилань.

API, FLASK, OPENAI, PYTHON, SQLITE, TELEGRAM, UML, VIBER, ЧАТ-БОТ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об'єктом розробки є процес обробки вхідних повідомлень від користувача, взаємодії з різними сервісами компанії OpenAI, та передачі отриманого результату кінцевим користувачам.

Метою кваліфікаційної роботи є створення інтелектуальних чат-ботів для месенджерів Telegram та Viber з функціоналом ведення діалогу з користувачем, генерації зображень та транскрибування аудіо- та відеофайлів.

Для розробки даної програми використовувалась мова програмування Python та набір різноманітних бібліотек для створення чат-ботів, веб-серверів та взаємодії з OpenAI API і базою даних. Перед безпосереднім створенням програми був розроблений її користувацький інтерфейс та архітектура за допомогою UML діаграм.

Результат роботи: проведено аналіз найпопулярніших месенджерів на даний момент: Viber та Telegram, проаналізовано принципи створення чат-ботів для них, проаналізовано найпопулярніші сервіси компанії OpenAI, які в подальшому будуть використовуватись в імплементації проєктів, розроблено користувацький інтерфейс та архітектуру чат-ботів, а також створено самі чат-боти, які в даний момент доступні для використання. Проведено тестування розроблених систем.

Створені програмні продукти можуть використовуватись будь-якими користувачами месенджерів Telegram або Viber з метою отримання відповідей на різноманітні запитання чи просто ведення діалогу з інтелектуальним віртуальним помічником на бажані теми. За допомогою розроблених чат-ботів можна створювати зображення за поданим описом, навіть якщо таких досі не існує в мережі інтернет, та робити транскрибування аудіо- чи відеофайлів.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

AI – Artificial Intelligence, штучний інтелект;

API – Application Programming Interface, прикладний програмний інтерфейс;

CPU – Central Processing Unit, центральний процесор;

HTTP – HyperText Transfer Protocol, протокол передачі гіпертексту;

HTTPS – HyperText Transfer Protocol Secure, безпечний протокол передачі гіпертексту;

JSON – JavaScript Object Notation;

UI – User Interface, користувацький інтерфейс;

UML – Unified Modeling Language, уніфікована мова моделювання;

URL – Uniform Resource Locator, єдиний вказівник на ресурс;

UUID – Unique User Identifier, унікальний ідентифікатор користувача;

ПЗ – Програмне забезпечення.

ВСТУП

Оцінка сучасного стану об’єкта розробки. З впровадженням мобільних пристроїв та невід'ємної ролі інтернету в повсякденному житті, найпопулярніші месенджери, такі як Viber та Telegram, стали невід'ємною частиною комунікації для мільйонів людей. Розвиток технологій штучного інтелекту та машинного навчання створює безліч можливостей для розробки віртуальних помічників – інтелектуальних чат-ботів, які можуть адаптуватися до потреб користувачів і швидко надавати більш точну та корисну інформацію, ніж вони могли б знайти в інтернеті.

Актуальність роботи та підстави для її виконання. З огляду на швидкий розвиток технологій у галузі штучного інтелекту та машинного навчання, актуальність даної роботи полягає у дослідженні та аналізі передових сервісів, таких як сервіси компанії OpenAI, та створенні чат-ботів з високим рівнем інтелектуальних здібностей та різноманітним і корисним функціоналом з використанням цих сервісів. Чат-боти можуть сприяти ефективності бізнес-процесів шляхом автоматизації рутинних завдань, таких як обслуговування клієнтів, проведення маркетингових кампаній та управління проєктами. Вони допомагають зекономити час, ресурси та покращити задоволення клієнтів. Окрім того вони ще й можуть служити віртуальними помічниками або супровідниками, що надають соціальну та емоційну підтримку користувачам, особливо в ситуаціях, коли доступ до психологічної допомоги обмежений або неможливий. Тому інтелектуальні чат-боти для одних з найпопулярніших на даний момент месенджерів: Viber та Telegram, створені з використанням технологій OpenAI, повинні користуватися величезним попитом. Поки що таких систем не дуже багато, оскільки ця галузь лише набуває свого росту, тому створення даних чат-ботів є задачею актуальною та практично необхідною.

Мета й завдання роботи. Метою роботи є розробка інтелектуальних чат-ботів для месенджерів Viber та Telegram з використанням сервісів компанії OpenAI.

Для досягнення мети потрібно вирішити такі задачі:

- Проаналізувати вплив месенджерів та їхніх чат-ботів на повсякденне життя користувачів;
- Проаналізувати принципи розробки чат-ботів для месенджерів Viber та Telegram;
- Проаналізувати сервіси, створені компанією OpenAI, та їхній API;
- Визначити, які із сервісів будуть інтегровані в систему. Сформувати вимоги до розроблюваних чат-ботів;
- Провести проєктування архітектури розроблюваних систем;
- Спроєктувати зручний та інтуїтивний користувацький інтерфейс;
- Розробити інтелектуальні чат-боти для месенджерів Viber та Telegram, які будуть використовувати деякі з сервісів компанії OpenAI;
- Провести мануальне тестування готової системи.

Об'єкт, методи й засоби розроблення. Об’єктом дослідження є процес обробки вхідних повідомлень від користувача, взаємодії з сервісами компанії OpenAI, та передачі отриманого результату кінцевим користувачам. Предметом дослідження є інтелектуальний чат-бот для месенджерів Viber та Telegram. Методи дослідження: структурний аналіз та проєктування інформаційних систем, моделювання з використанням мови UML, методи об’єктно-орієнтованого аналізу, проєктування і програмування.

Можливі сфери застосування. Практична значущість роботи полягає в розробці зручних інтелектуальних чат-ботів, що можуть використовуватись споживачами в надзвичайно різноманітних цілях, оскільки вони засновані на технології штучного інтелекту, що може допомогти відповісти на різноманітні запитання, забезпечити допомогу у розв'язанні завдань та багато іншого. Він може бути корисним для бізнесу та клієнтського сервісу, а також для особистих потреб, таких як пошук інформації або навіть просто для розваг.

РОЗДІЛ 1. РОЛЬ МЕСЕНДЖЕРІВ ТА ЧАТ-БОТІВ У СУЧАСНОМУ СУСПІЛЬСТВІ

1.1 Що являють собою месенджери та яка їхня роль у повсякденному житті людей

Месенджер – це програмне забезпечення, яке надає можливість обмінюватися приватними повідомленнями між двома чи більше користувачами. З кожним днем з’являється все більше і більше додатків для обміну повідомленнями, і ця технологія швидко стає найпопулярнішим способом комунікації, замінюючи SMS і MMS. Месенджери відіграють важливу роль в повсякденному житті людей, оскільки стають мостами між різними країнами, культурами та індивідами, що сприяє розвитку комунікаційних можливостей та зближенню людей. Завдяки месенджерам можна легко та швидко встановлювати зв'язок з друзями, родичами або колегами, незалежно від відстані чи часових поясів. Вони дозволяють підтримувати контакт з близькими, поділятися думками, ідеями, емоціями, а також відсвяткувати особливі події разом, навіть якщо фізично не можуть бути поряд [1].

У сучасному суспільстві месенджери відіграють дуже значну роль. У бізнесі вони сприяють співпраці між співробітниками, полегшують координацію роботи, управління проєктами та обмін інформацією. Це дозволяє компаніям працювати ефективніше, підвищувати продуктивність та забезпечувати кращу комунікацію між різними відділами та командами. Месенджери також можуть бути використані для підтримки особистого розвитку та добробуту. Через спеціалізовані чат-боти та платформи, люди можуть отримувати поради зі здоров'я, фітнесу, медитації, духовності або навіть кар'єри. Це допомагає забезпечити підтримку та мотивацію на шляху до досягнення особистих цілей та покращення якості життя. Ще дані програми

використовуються для навчання та освіти, дозволяючи студентам та викладачам спілкуватися, проводити онлайн-заняття та обговорювати навчальний матеріал. Вони стали особливо корисними, коли люди перейшли на дистанційне навчання та роботу в 2020 році у зв’язку з пандемією [2].

Загалом, месенджери стали невід’ємною частиною нашого повсякденного життя, оскільки вони пропонують безліч можливостей для спілкування, співпраці, навчання, розваг та особистого розвитку. Ці програми допомагають нам зберігати зв'язок з нашими близькими, розвивати наші професійні та особисті навички, а також відкривати нові можливості для розвитку та зростання.

1.2 Що таке чат-боти і для чого вони потрібні

Чат-боти – це окремі облікові записи користувача в певній соціальній мережі чи месенджері, які керуються програмою, розміщеною локально чи на якомусь сервері. Вони стали важливим елементом сучасного суспільства, оскільки допомагають полегшити доступ до інформації, автоматизувати рутинні задачі та вдосконалювати сервіси і продукти. В основному боти запрограмовані виконувати одну конкретну функцію, наприклад: робити розсилку всім підписаним на нього користувачам, здійснювати грошові перекази, шукати потрібну інформацію і багато інших речей. Також є боти, які запрограмовані на звичайне спілкування з людьми, для створення яких використовується технологія машинного навчання. Часом їх буває важко відрізнити від звичайного користувача на перший погляд, оскільки вони можуть надсилати повідомлення, вести діалог і допомагати в багатьох питаннях, як звичайні люди [3]. На рис. 1.1 зображений приклад спілкування з ботом Зоряною, яка доступна в Viber, Telegram і Facebook. І, дійсно, її відповіді дуже схожі на людські.

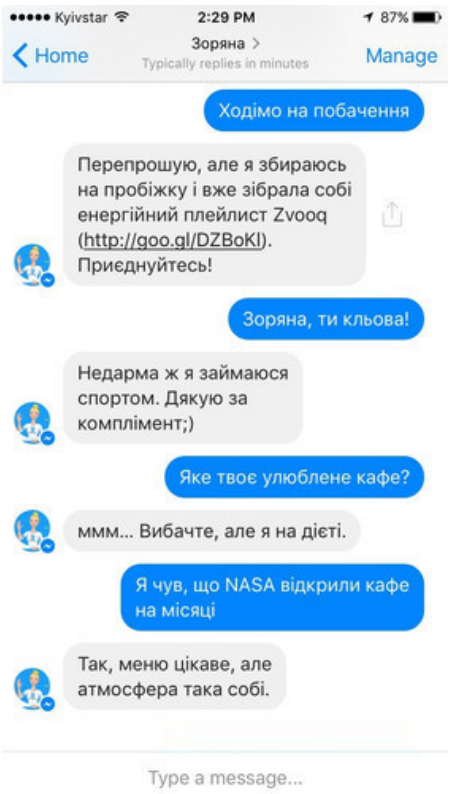

Рисунок 1.1 – Приклад спілкування із ботом Зоряною у Viber [4]

Одним з ключових аспектів чат-ботів є їхня здатність взаємодіяти з користувачами через різні платформи, такі як месенджери, соціальні мережі, веб-сайти та інші застосунки. Це робить їх універсальним інструментом для різних сфер життя та діяльності. Наприклад, чат-боти можуть служити віртуальними помічниками, допомагаючи користувачам знаходити потрібну інформацію, відповідаючи на запитання та надаючи рекомендації. Крім того, чат-боти активно використовуються в бізнесі для підтримки клієнтів, оскільки вони можуть обробляти велику кількість запитів одночасно. Спираючись на сучасні реалії, ми можемо бачити, що боти знімають до 50% загрузки з онлайн консультантів. Їхня перевага в тому, що вони одночасно можуть спілкуватись із багатьма користувачами і можуть працювати цілодобово. Бот здатен дати моментальну відповідь на поставлене питання, якщо ж у нього немає цієї відповіді – він може з’єднати людину з реальним консультантом. Ці асистенти допомагають компаніям вирішувати різноманітні проблеми клієнтів, економлячи час і ресурси. Чат-боти також можуть виступати в ролі

маркетингових інструментів, просування продуктів та послуг, залучення нових клієнтів та підтримка зв'язку з ними. У сфері освіти чат-боти також виявляються корисними, допомагаючи студентам з засвоєнням навчального матеріалу, проведенням тестів та контролю за їхнім прогресом [5].

Популярною і водночас помилковою є думка, що боти з’явились в XXI ст. Насправді – це не так. Першим ботом, який імітував спілкування з людиною була програма ELIZA, створена у 1966 році. Якщо говорити про виконання якихось задач, то першим був бот під назвою Бармен(BARTENDER), який пропонував користувачу випадковий напій [6].

З кожним днем чат-боти стають все популярніші й популярніші. Ще нещодавно всі створювали різноманітні мобільні додатки для смартфонів, але тренди змінюються. Люди надають перевагу ботам, оскільки їхнє створення простіше, займає менше часу і ресурсів. Загалом, чат-боти мають великий потенціал для різних сфер сучасного суспільства, допомагаючи автоматизувати задачі та покращувати якість сервісів. Вони розвиваються та вдосконалюються завдяки новим технологіям, таким як штучний інтелект та машинне навчання, що дає їм можливість стати ще більш адаптивними та корисними для користувачів [5].

РОЗДІЛ 2. МЕСЕНДЖЕР ТЕЛЕГРАМ ТА ПРИНЦИПИ СТВОРЕННЯ БОТІВ ДЛЯ НЬОГО

2.1 Загальний огляд месенджеру Telegram

Месенджер **Telegram** – це сучасний сервіс для обміну повідомленнями, який був анонсований 14 серпня 2013 року та досі продовжує розвиватись стрімкими темпами. Від початку свого існування, Telegram зосередився на принципах безпеки, конфіденційності та швидкості роботи. Завдяки цим властивостям, він здобув велику популярність у користувачів по всьому світу. Сьогодні, як повідомляється, він має понад 550 мільйонів активних користувачів щомісяця, що є на 175% більше, аніж це було в 2018 році. А за кількістю завантажень він входить до десятки найпопулярніших соціальних мереж світу і є сьомим додатком за кількістю завантажень на iOS та Android. Водночас, Telegram обійшов такі популярні месенджери, такі як Viber, Line та Signal, проте він досі поступається лідерам ринку, таким як WhatsApp та Facebook Messenger, які налічують кілька мільярдів користувачів кожен [7].

Telegram надає своїм користувачам можливість обмінюватися текстовими повідомленнями, фотографіями, відео, аудіо, стікерами та документами різного формату. Також за допомогою нього можна створювати групові чати та канали для широкого розповсюдження інформації, а також здійснювати голосові та відеодзвінки.

Однією з найцікавіших функцій Telegram є можливість створення ботів, які можуть автоматизувати різні процеси та спрощувати взаємодію між користувачами. Боти можуть надсилати повідомлення, відповідати на запитання, виконувати різні завдання та надавати корисну інформацію.

Основною відмінністю Telegram від інших месенджерів є його надійне шифрування, яке забезпечує безпеку та анонімність користувачів.

Використовуючи так званий протокол MTProto, месенджер забезпечує конфіденційність переписки та захист від несанкціонованого доступу до облікових записів клієнтів. Цей протокол використовує симетричне шифрування, тобто це означає, що за допомогою ключа, за яким дані шифрувалися, можна їх розшифрувати. Алгоритм побудований на базі 256-бітного **AES**, 2048-бітного **RSA** алгоритмів шифрування. Обмін криптографічними ключами між користувачами відбувається на базі протоколу **Діффі-Гелмана**, який надає змогу безпечно обмінюватись ключами через публічний канал [8].

Варто відзначити, що Telegram є кросплатформним додатком і доступний для різних платформ, включаючи **Android**, **iOS**, **Windows**, **macOS**, **Linux**. Також він доступний у вигляді **Web-**додатку. Це дозволяє користувачам легко синхронізувати свої повідомлення та файли між різними пристроями і не залежати від якоїсь однієї операційної системи.

Підсумовуючи, можна сказати, що Telegram – це інноваційний месенджер, який забезпечує швидкість, безпеку та конфіденційність всім своїм користувачам. Його розробники постійно працюють над вдосконаленням додатку, враховуючи потреби користувачів та різноманітні глобальні виклики. Завдяки своїм перевагам та активному розвитку, Telegram продовжує збільшувати свою аудиторію та зміцнювати свою позицію на ринку месенджерів.

2.2 Принципи створення чат-ботів для Telegram

Бот в Telegram – це окремий обліковий запис до якого не потрібно прив’язувати номер мобільного телефону. Користувач може взаємодіяти з ним двома шляхами:

- Надсилати повідомлення і команди боту, використовуючи діалог із ним або добавляючи його до різноманітних чатів;

- Посилати запити напряму через поле вводу повідомлень, вказуючи ім’я бота в форматі **<@ім’я_бота>** і текст запиту до нього (див рис. 2.1). Це дозволяє отримувати відповідь від нього у будь-якому чаті. Такий спосіб взаємодії з ботом має назву «**inline**».

Відрізнити бота від людини в Telegram дуже просто:

- Боти не мають мережевого статусу. Замість нього на цьому місці знаходиться напис «**bot**»;
- Також боти не починають спілкування із користувачем першими.

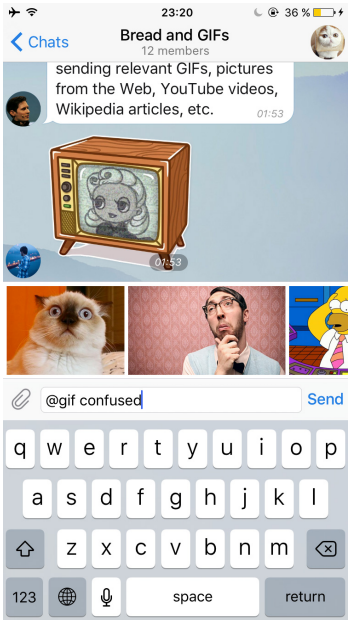

Рисунок 2.1 – inline спосіб взаємодії з ботом в Telegram [9]

Створити обліковий запис для чат-бота в Telegram дуже просто. Це можна зробити за допомогою такого ж бота, створеного розробниками месенджеру, який має назву **BotFather**. Спілкування з ним відбувається за принципом надсилання команд та отримання подальших вказівок. Насамперед, йому треба надіслати команду «/**newbot**», у відповідь на яку BotFather попросить вказати ім'я бота та унікальне ім'я користувача для нього. Ім'я бота буде відображатись у списку контактів користувачів, а за допомогою унікального імені можна буде звертатись до нього з будь-якого місця. Також унікальне ім'я чат-бота повинно мати закінчення «**bot**», наприклад: «smart_assistant_bot», «SmartAssistantBot». Після завершення початкових налаштувань буде отримано унікальний токен

(token), що являє собою рядок із символів, за допомогою якого можна буде керувати ботом зі своєї програми. Він має зберігатись тільки у творця бота і бути доступним тільки йому, оскільки є ключем доступу до бота [10].

Окрім створення ботів, BotFather надає широкий спектр команд для їх налаштування та редагування. Розглянемо найважливіші з них.

1) Команди для редагування:
    a) «/**setname**» – зміна імені бота;
    b) «/**setdescription**» – зміна опису функціональності для бота. Цей опис користувачі зможуть побачити при першій взаємодії з чат-ботом;
    c) «/**setuserpic**» – зміна зображення профілю;
    d) «/**setcommands**» – встановлення списку команд, які підтримуються ботом. Розробники Telegram рекомендують користувачам, які займаються створенням ботів, надати для нього наступний список команд: «/**start**», «/**description**». Це робиться для того, щоб всі чат-боти були схожими в користуванні, і в людей не виникало ніяких труднощів при освоєнні функціоналу;
    e) «/**deletebot**» – команда, за допомогою якої можна видалити обліковий запис створеного бота.
2) Команди для налаштування:
    a) «/**setjoingroups**» – команда, яка надає чи забирає в бота можливість приєднуватись до багатокористувацьких чатів;
    b) «/**setprivacy**» – встановлює тип повідомлень, які буде отримувати бот при приєднанні до чатів з багатьма користувачами;
    c) «/**setinline**» – встановлює чи скидає режим користування ботом в будь-якому з чатів Telegram.

Після всіх налаштувань можна приступати до написання програми, яка буде взаємодіяти з новоствореним ботом. Для цього Telegram надає стороннім розробникам, які мають намір розробляти якогось бота, доступ до свого **API** (Application Programming Interface). API, або ж як його називають українською –

прикладний програмний інтерфейс – це набір способів за допомогою яких одна програма може взаємодіяти з іншою. API виділяє для нас певну функціональність, якою ми можемо користуватися не знаючи як вона реалізована [11]. У нашому випадку Telegram надає користувачам набір доступних запитів, які можна надсилати на сервер, за допомогою яких відправники можуть отримати зворотню відповідь. Всі запити до серверу мають бути певної форми, яка має наступний вигляд: **https://api.telegram.org/bot<token>/METHOD_NAME** , де **<token>** – це унікальний текстовий рядок, який ми отримали при створенні бота, а **METHOD_NAME** – це один із доступних методів, які містяться в API. Telegram API підтримує надсилання **GET** і **POST** запитів на сервер. GET і POST – це два різні типи **HTTP**-запитів. В загальному GET використовується для отримання якоїсь інформації без її зміни і включає всю необхідну інформацію в **URL** (Uniform resource locator), який в свою чергу є стандартизованою адресою певного ресурсу та є визначником місцеперебування сайту в мережі. URL складається з домену, шляху до сторінки та імені її файлу. POST запит – навпаки, використовується для надсилання чи зміни певної інформації. Він вимагає надсилання на сервер додаткових даних в певному форматі, які зберігаються в тілі повідомлення [12]. Також Telegram API підтримує чотири шляхи передачі параметрів у запитах:

1) URL query string (рядок запиту);
2) application/x-www-form-urlencoded;
3) application/json (не використовується для надсилання файлів);
4) multipart/form-data (використовується для надсилання файлів).

Розглянемо тільки два з них, які використовуються найчастіше.

Query string є частиною URL, яка виконує роль призначення значень для заданих параметрів. Query string включає в себе поля, які були добавлені до базового URL вручну за допомогою веб-браузера чи автоматично якоюсь програмою. В залежності від надісланих параметрів, веб сервер буде повертати різні результати. На рис. 2.2 ми можемо побачити, що URL умовно поділена на

дві частини розділювачем «**?**». До знаку питання знаходиться звичайна адреса сайту, після нього рядок запиту, який складається з двох параметрів: title та action зі значеннями Query_string та edit відповідно, які в свою чергу розділені між собою символом «**&**». Таким чином можна використовувати стільки параметрів, скільки потрібно. Цей метод передачі даних може бути використаний для надсилання текстових повідомлень користувачу та отримання відповіді від нього [13].

Рисунок 2.2 – Приклад простого рядку запиту

Multipart/form-data – це складовий тип вмісту повідомлення, який використовується для надсилання POST запитів з вкладеними даними. Він вказується в полі заголовка **Content-Type** (тип вмісту). Повідомлення типу multipart/form-data може містити декілька частин , і це означає, що ми можемо одночасно відправляти декілька типів даних. Прикладом використання цієї технології є лист, який надсилається через електронну пошту. При надсиланні такого листа формується повідомлення типу **multipart/form-data**, де заголовок, тема, адреса отримувача, текст та вкладені файли є окремими частинами. Кожна частина обов’язково має містити атрибут «**name**». Цей тип повідомлень може бути використаний для надсилання файлів користувачу, який робить певні запити до чат-бота [12].

Відповідь на запит нашої програми до сервера Telegram завжди приходить у форматі **JSON** об’єкта. JSON є абревіатурою, яка розшифровується як JavaScript Object Notation. Він являє собою певний текстовий формат обміну даними, де інформація розташовується компактно, завдяки чому її легко прочитати одержувачу, навіть якщо цей текст читатиме людина. JSON зберігає дані об’єкта у вигляді множини «**name**»:«**value**», де «name» – це рядок, а «value» може набувати різних форм, таких як число, рядок, масив, інший об’єкт, логічний літерал. Сам JSON об’єкт починається з символа «{» та закінчується

символом «}». Не дивлячись на те, що цей формат утворився від мови програмування JavaScript, він є незалежним і може використовуватись практично будь-якою мовою програмування [14].

Рисунок 2.3 – Форма запису JSON об’єктів [14]

Одним з найважливіших методів Telegram API є **getUpdates**. За допомогою нього можна отримувати всі повідомлення, які надійшли від користувачів до створеного чат-бота. Цей метод повертає JSON об’єкт **Result**, який в свою чергу містить масив JSON об’єктів **Update**. Всі вони містять інформацію про надіслані повідомлення, їх формат та інші додаткові дані, в залежності від яких чат-бот буде надавати відповідь користувачам [15].

Насамкінець, варто відмітити, що існують безліч бібліотек для різних мов програмування, які являють собою зручну обгортку для взаємодії з Telegram API та надають чудовий інтерфейс для роботи з ним у вигляді функцій чи методів. Саме за допомогою цих бібліотек найчастіше і створюють чат-боти для месенджеру Telegram.

РОЗДІЛ 3. МЕСЕНДЖЕР VIBER ТА ПРИНЦИПИ СТВОРЕННЯ БОТІВ ДЛЯ НЬОГО

3.1 Загальний огляд месенджеру Viber

Месенджер **Viber** є одним з найвідоміших сервісів обміну повідомленнями та здійснення голосових та відеодзвінків на світовому ринку. Він був заснований у 2010 році ізраїльськими розробниками: Талмоном Марко та Ігорем Магазінником. З того часу Viber став одним з провідних месенджерів, завдяки своїй простоті, зручності та можливості здійснювати безкоштовні дзвінки, маючи тільки підключення до інтернету та смартфон [16].

Основні функції Viber включають обмін текстовими повідомленнями, фотографіями, відео, аудіо, стікерами та документами різного формату. Користувачі мають можливість створювати групові чати та канали, за допомогою яких можна розсилати інформацію будь-якого типу величезній аудиторії. Крім цього, Viber пропонує голосові та відеодзвінки як для двох користувачів, так і в групах. Також Viber дозволяє здійснювати безкоштовні міжнародні дзвінки на телефонні номери за допомогою функції Viber Out, якщо підключення до інтернету відсутнє.

З точки зору безпеки, Viber використовує шифрування **end-to-end**, яке забезпечує захист даних користувачів від стороннього перехоплення. У багатьох службах обміну повідомленнями треті сторони зберігають дані, які шифруються лише під час передачі. Цей метод шифрування на стороні сервера захищає дані лише від неавторизованих користувачів. Але завдяки цьому методу відправник також може переглядати інформацію, що може бути небажаним у випадках, коли необхідна конфіденційність даних у всіх кінцях. У випадку end-to-end шифрування, зашифровані дані можуть переглядати лише ті, хто має ключі дешифрування. Іншими словами, end-to-end шифрування запобігає доступу до

даних стороннім особам, у тому числі третім сторонам. Це створює додатковий рівень конфіденційності та безпеки для користувачів сервісу [17].

Стосовно популярності, Viber є одним з найвідоміших месенджерів у світі, хоча й поступається таким гігантам, як: WhatsApp, Facebook Messenger та Telegram. На 2021 рік, кількість активних користувачів Viber склала близько 1 мільярд осіб. Хоча його популярність у порівнянні з найбільшими гравцями на ринку месенджерів дещо нижча, Viber відрізняється своєю великою поширеністю у певних регіонах, зокрема у Східній Європі та деяких азіатських країнах [18].

Однією з причин популярності Viber є його функціональність, яка постійно розвивається та оновлюється. Наприклад, сервіс запровадив підтримку чат-ботів, які можуть автоматизувати різні процеси та спрощувати взаємодію між користувачами, а також різноманітні стікери та інтеграцію з іншими популярними сервісами.

На даний момент Viber доступний для різних платформ: Android, iOS, Windows, macOS та Linux, що є також дуже великою його перевагою.

Підсумовуючи, можна сказати, що Viber є дуже відомим і популярним месенджером, який пропонує широкий спектр функцій і можливостей для своїх користувачів. Незважаючи на те, що Viber поступається деяким світовим лідерам ринку месенджерів, він продовжує активно розвиватися та збільшувати свою аудиторію, зокрема завдяки фокусу на безкоштовних дзвінках та розширенню функціональності.

3.2 Принципи створення чат-ботів для Viber

Для створення користувацьких чат-ботів Viber надає непоганий REST API. Проте, перш ніж почати ним користуватись, потрібно створити обліковий запис користувача Viber та прив’язати його до активного мобільного номеру телефону. Після цього можна приступити, власне, до створення чат-бота. Це

можна зробити, перейшовши за наступним посиланням https://partners.viber.com/account/create-bot-account. На даній сторінці відображені різні поля вводу інформації, про чат-бот, які необхідно заповнити.

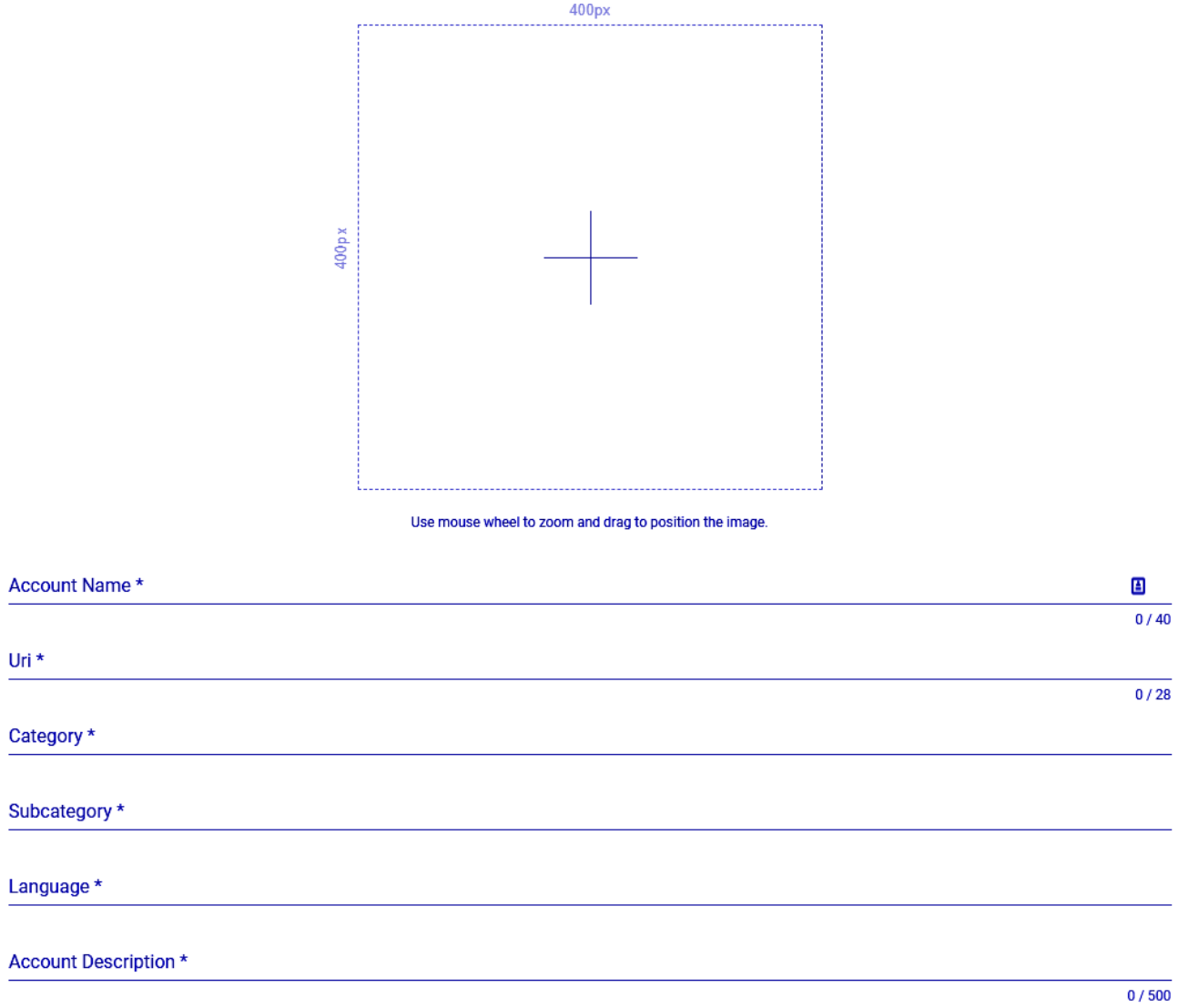

Рисунок 3.1 – Сторінка створення Viber чат-бота

За допомогою кнопки «плюс» посередині можна встановити зображення для облікового запису чат-бота, а в полі «**Account name**» необхідно написати його ім’я, яке буде відображатись всім користувачам, при взаємодії з ним. На місці «**Uri**» треба вказати унікальний ідентифікатор чат-бота, за яким він може бути знайдений клієнтами в подальшому. Цей ідентифікатор повинен містити максимум 28 символів. В полях «**Category**» та «**Subcategory**» необхідно зазначити до якої сфери належить даний чат-бот. При натисканні на них,

користувацький інтерфейс запропонує обрати певний вид діяльності із наперед заготовленого списку. Також потрібно обрати мову, яка буде основною для асистента. Ще однією важливою інформацією про чат-бот є опис його функціональності. Він повинен бути зазначений в полі «**Account Description**». Після вказання всіх необхідних даних та погодження з політиками Viber, можна натиснути на кнопку «**Create**», яка створить наш бот і відобразить сторінку з його активною інформацією.

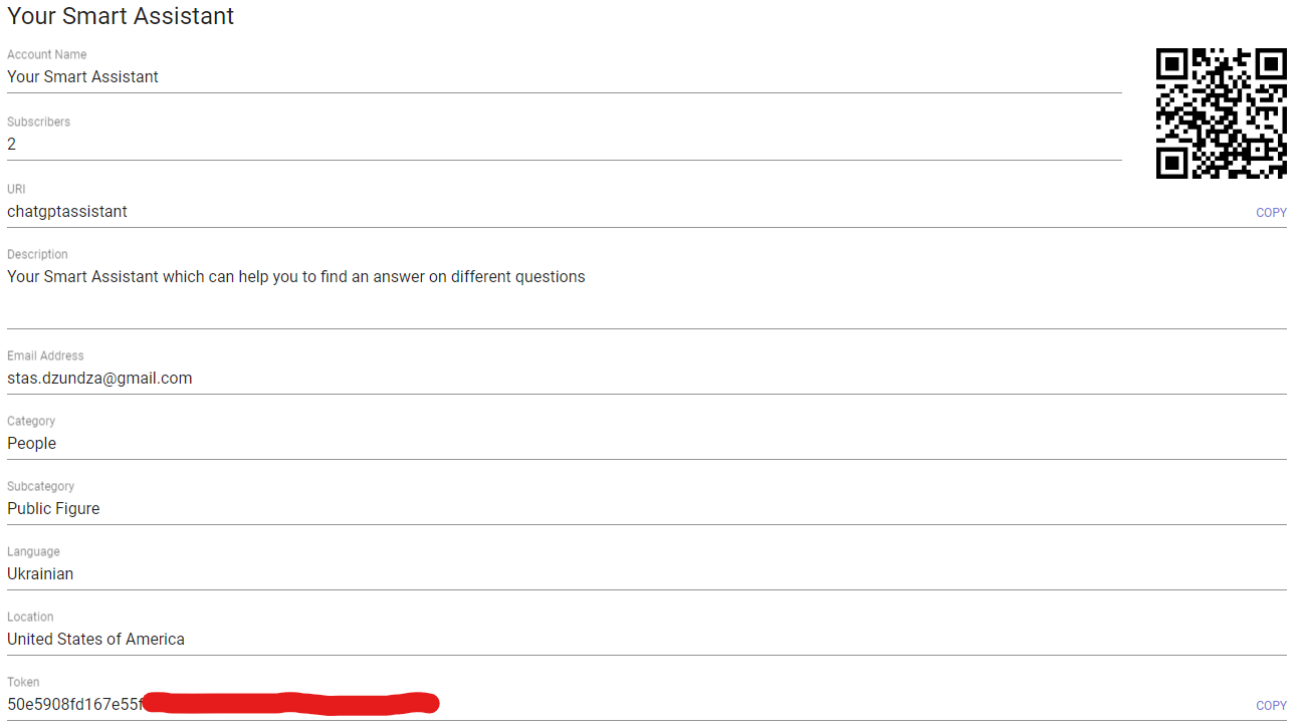

Рисунок 3.2 – Сторінка інформації про створеного чат-бота

Варто зазначити, що власником та адміністратором Viber чат-бота є користувач, з облікового запису якого він був створений. Наразі боти підтримуються на пристроях iOS і Android, на яких працює Viber версії 6.5 і вище, а на настільних комп'ютерах – починаючи з версії 6.5.3 [19].

В результаті створення чат-бота буде отримано його унікальний токен, який знаходиться на сторінці інформації про чат-бот в останньому полі – «**Token**» (див. рис. 3.2). Він використовується для автентифікації запиту до Viber API і для запобігає надсиланню запитів від імені бота неавторизованими

особами. Кожен запит до API має містити HTTP-заголовок під назвою **X-Viber-Auth-Token**, що містить токен чат-бота.

Всі запити до API мають бути певної форми, яка виглядає наступним чином: **https://chatapi.viber.com/pa/METHOD_NAME**, де **METHOD_NAME** – це одна із доступних функцій, які містяться в API. Розглянемо деякі з цих методів детальніше [19].

Після отримання токена чат-бота, потрібно встановити для нього **webhook**. Webhook – це метод, який дозволяє одному веб-серверу надсилати автоматичні повідомлення іншому серверу, коли відбувається певна подія. Це може бути використано для реагування на різні події в реальному часі, такі як: оновлення даних, надходження нових повідомлень тощо. В контексті Viber чат-ботів, webhook використовується для отримання повідомлень від користувачів та відповіді на них відповідно до певних інструкцій. Коли користувач надсилає повідомлення чат-боту, Viber сповіщає про дану подію сервер, на якому запущений бот, за допомогою webhook, передаючи інформацію про повідомлення та користувача [19].

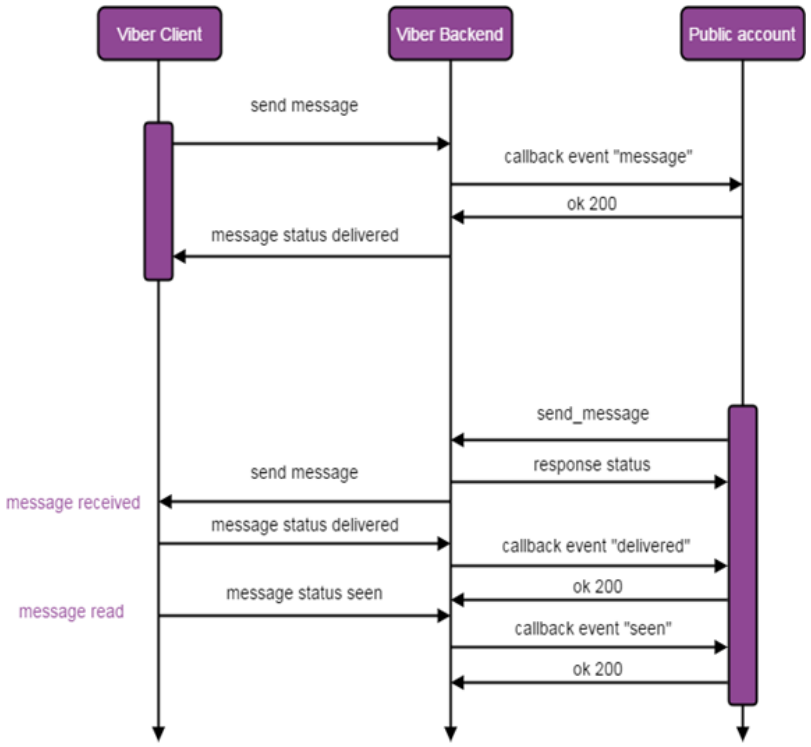

Рисунок 3.3 – Sequence діаграма обміну повідомлень між чат-ботом та користувачем [19]

Сервер, може обробити отриману інформацію, згенерувати відповідь і відправити її назад до Viber, який, в свою чергу, відправить відповідь користувачеві. Встановлення webhook може бути виконано шляхом виклику API POST методу **set_webhook** із дійсною та сертифікованою URL-адресою, за якою знаходиться розроблений чат-бот. З міркувань безпеки дозволені лише URL-адреси з дійсним і офіційним сертифікатом SSL [19] від надійного центру сертифікації, який має бути в списку надійних кореневих сертифікатів Sun Java. set_webhook API метод приймає на вхід такі параметри як:

1) **url –** унікальна URL-адреса веб-серверу, на якому розміщений чат-бот;
2) **event_types –** типи повідомлень, які будуть оброблятися чат-ботом;
3) **send_name** та **send_photo –** прапорці, які вказують, чи буде чат-бот отримувати імена відправників та їхні фото відповідно.

```
{
   "url":"https://my.host.com",
   "event_types":[
      "delivered",
      "seen",
      "failed",
      "subscribed",
      "unsubscribed",
      "conversation_started"
   ],
   "send_name": true,
   "send_photo": true
}
```

Рисунок 3.4 – Приклад параметрів для set_webhook API [19]

Наступним ключовим методом Viber API є **send_message**. Це POST метод, який дозволяє чат-боту надсилати повідомлення користувачам Viber. Варто відзначити, що надсилати повідомлення можна тільки тим користувачам, які підписані на даного чат-бота. **send_message** підтримує різні типи повідомлень: текстові, зображення, відео, файли, розташування, наклейки, контакти, користувацькі клавіатури тощо [19].

```
{
   "receiver":"01234567890A=",
   "min_api_version":1,
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "tracking_data":"tracking data",
   "type":"text",
   "text":"Hello world!"
}
```


Рисунок 3.5 – Приклад параметрів для надсилання текстового повідомлення через send_message API [19]

У відповідь на запит до даного API, Viber повертає JSON-рядок в якому вказано статус надсилання та унікальний ідентифікатор повідомлення. Якщо статус дорівнює нулю, то це означає, що повідомлення було надіслано успішно і доставлено адресату.


```
{
"status":0,
"status_message":"ok",
"message_token":5741311803571721087,
"chat_hostname":"SN-CHAT-05_",
"billing_status":1
}
```


Рисунок 3.6 – Приклад відповіді від Viber API [19]

Останнім важливим методом Viber API, який буде розглянуто, є **broadcast_message**. Він дозволяє чат-ботам надсилати повідомлення кільком користувачам Viber, які на нього підписані, одночасно. Це можна зробити, вказавши список отримувачів у поле **broadcast_list**, яке є параметром даного API методу (див. рис. 3.7) [19].

Так само, як і для Telegram, не зовсім зручно здійснювати виклики розглянутих API напряму, це доцільно робити використовуючи якісь обгортки над ним. На щастя, для Viber чат-ботів існує офіційна бібліотека для мови

програмування Python – «**viberbot**», яка надає зручний інтерфейс для взаємодії з Viber API [19].

```
{
   "broadcast_list":[
      "ABB102akPCRKFaqxWnafEIA==",
      "ABB102akPCRKFaqxWna111==",
      "ABB102akPCRKFaqxWnaf222=="
   ]
}
```

Рисунок 3.7 – Приклад вхідних параметрів для broadcast_message [19]

РОЗДІЛ 4. ОГЛЯД СЕРВІСІВ OPENAI

4.1 Що таке OpenAI

OpenAI – це створена в Америці дослідницька лабораторія штучного інтелекту (AI), що складається з безприбуткової гілки OpenAI Incorporated та її прибуткової дочірньої корпорації OpenAI Limited Partnership. OpenAI проводить дослідження в галузі AI з метою просування та розвитку дружнього штучного інтелекту. Лабораторія відкрилась у 2015 в Сан-Франциско і мала за мету створити «загальний штучний інтелект» або AGI. Її засновниками були такі видатні особистості, як: Ілон Маск, Сем Альтман, Водзіч Зарек та інші. Компанія хотіла захиститися від майбутнього, в якому великі технологічні гіганти, такі як Google, зможуть досконало опанувати технологію AI та монополізувати її. Головна ціль даної організації полягала в тому, щоб створювати програмне забезпечення штучного інтелекту прозорим способом і зробити свої продукти відкритими, для того щоб світ міг отримати від них всі можливі переваги. Одним з ключових принципів OpenAI є співпраця з іншими дослідницькими та освітніми установами, а також з промисловістю. Це сприяє обміну знаннями та досвідом, що робить AI-технології доступнішими для розробників та користувачів по всьому світу [20]. Першою відкритою розробкою компанії став OpenAI Gym. Він являв собою середовище для тренування, у якому користувачі могли створювати свої алгоритми, а згодом розміщувати їх у різних віртуальних просторах, де вони проходили тестування. Дана задумка згодом дала свій ефект – з’явився алгоритм OpenAI Five – п'ять віртуальних гравців, створених OpenAI для відеогри Dota 2, які у квітні 2019 року перемогли команду OG, що була чинним чемпіоном світу в даній дисципліні на той час. Після цього OpenAI зосередився на загальніших дослідженнях і розробках в галузі AI [21].

У 2019 році OpenAI отримала один мільярд доларів фінансування від Microsoft, який погодився ліцензувати та комерціалізувати деякі технології компанії. А вже у 2020 році OpenAI створила GPT-3. Дана мережа мала 45 ТБ текстових даних, перетворених у 175 млрд параметрів [21].

У березні 2023 року відбувся офіційний анонс GPT-4 – нової мовної моделі, створеної OpenAI, яка може генерувати текст, схожий на людську мову. Вона вдосконалює технологію, що використовується в GPT-3. Згідно з OpenAI, ця мовна модель нового покоління є більш досконалою у трьох ключових сферах: креативність, візуальне введення та довший контекст. Що стосується креативності, OpenAI стверджує, що GPT-4 набагато краще підходить як для створення, так і для співпраці з користувачами над творчими проєктами. Довший контекст також відіграє важливу роль. GPT-4 тепер може обробляти до 25 000 слів тексту від користувача. Системи OpenAI працюють на суперкомп'ютерній платформі на базі Azure від Microsoft [22].

4.2 Популярні OpenAI сервіси та огляд їхніх API

Розглянемо тепер три найпопулярніші AI моделі: ChatGPT, DALL-E та Whisper, які в подальшому будуть використовуватись для створення чат-ботів.

Почнемо з **ChatGPT** – чат-бота зі штучним інтелектом на основі GPT-3.5, представленого у листопаді 2022 року, який набув надзвичайної популярності, оскільки міг виконувати будь-які завдання з текстом: писати твори, генерувати ідеї для у різних сферах діяльності, писати код для програм, складати вірші, давати поради тощо. ChatGPT збирає дані, створені людьми, з Інтернету, і використовує обчислювальні прогнози для відповідей на повідомлення, що надходять від користувача. Головна особливість ChatGPT – це генерація відповідей, подібних до людських. Тому він чудово підходить для розробки чат-ботів та віртуальних помічників. На даний момент є дві найактуальніші

моделі ChatGPT: gpt-3.5-turbo та gpt-4, проте остання ще досі знаходиться в бета-стадії і доступна тільки для відносно малого кола осіб [23].

DALL-E – це революційний сервіс від компанії OpenAI, який дозволяє генерувати зображення на основі текстових описів. Він використовує технологію GPT для створення зображень, що відповідають текстовому опису. DALL-E може створювати реалістичні зображення різних об'єктів, персонажів, сцен та абстрактних композицій. Навіть якщо відповідне зображення неможливо знайти в реальному світі, дана модель може згенерувати унікальні малюнки, що відповідають заданому користувачем опису. Наприклад, можна надати DALL-E запит «футуристичне місто на воді», і модель створить унікальне зображення, що відповідає отриманому опису. Цей сервіс може бути корисним для графічних дизайнерів, митців та інших, хто потребує візуалізації концепцій на основі тексту. Його інтеграція в чат-боти для різних месенджерів виглядає чудовою ідеєю [24].

Whisper – це ще один інноваційний сервіс від OpenAI, що зосереджений на розумінні та обробці звукових сигналів, зокрема мовлення. Він базується на моделі GPT-3, але був адаптований для роботи з аудіоінформацією замість текстової. Основна мета Whisper – перетворення звукових даних у текстове представлення. Даний сервіс відрізняється від інших подібних тим, що він навчався на величезному наборі даних з аудіозаписами з різних мов та акцентів. Це дозволяє системі краще справлятися з різноманітністю мовлення, розуміти складні аудіоматеріали та працювати в різних середовищах. Whisper може бути корисним для розробників програмного забезпечення, що хочуть створити програми та сервіси з голосовим керуванням, а також для інших сфер, де важливо швидко перетворювати голосові дані на текст. В розроблюваних чат-ботах він може бути використаний для транскрибування аудіо- та відеофайлів, а також для розпізнавання голосових повідомлень користувача [25].

Варто відзначити, що OpenAI надає розробникам доступ до своїх моделей через API. Це дозволяє інтегрувати потужні AI-технології у різноманітні

застосунки, такі як чат-боти, перекладачі, автоматичне генерування тексту тощо. OpenAI API сервіси є чудовими інструментами для створення інноваційних продуктів та розв'язання складних проблем. Також для надсилання запитів до OpenAI API потрібно мати API-ключ, який який можна отримати на офіційному сайті компанії OpenAI в налаштуваннях особистого кабінету користувача. Цей ключ являє собою текстовий рядок у наступному форматі:

sk-xxxxxxxxxxxxxxxxxxxxxxxx

, де замість символів *x* знаходиться певна цифра цифр чи літера латиниці.

Розглянемо детальніше API для ChatGPT. Його URL виглядає наступним чимном – «*https://api.openai.com/v1/chat/completions*». Це POST запит, який може приймати дуже багато параметрів, що можуть бути як обов’язковими, так і опціональними. Зупинимось спочатку на першій категорії параметрів, на даний момент їх існує всього 2: **model** та **messages**. В полі model вказується ідентифікатор моделі, яка повинна використовуватись. Це може бути: gpt-3.5-turbo, gpt-3.5, gpt-3 тощо. В параметрі messages має знаходитись список повідомлень, які були створені в рамках діалогу користувача з ChatGPT. Тобто, там повинна зберігатись історія чату. Кожне повідомлення – це JSON об’єкт, який містить поле **role**, яке вказує на його відправника, та поле **content**, де зберігається, власне, вміст повідомлення. role може набувати трьох значень: **system**, **user** чи **assistant**. За допомогою system можна встановити роль асистента. user та assistant використовуються для маркувань повідомлень користувача та чат-бота відповідно. Також повідомлення може містити ім’я автора повідомлення, проте його вказувати не обов’язково. Розглянемо ще декілька необов’язкових, але важливих параметрів даного API. Один з них – це **temperature**, який впливає на різноманітність результату. За допомогою нього можна вказати, яку температуру вибірки використовувати. Дане поле приймає значення від 0 до 2. Вищі значення, як-от 0.8, зроблять результат більш випадковим, тоді як нижчі значення, навпаки, будуть надавати більш детерміновані результати. Ще одним важливим параметром являється **n**, який

визначає скільки результатів потрібно згенерувати на запит від користувача. За замовчуванням він дорівнює одиниці. Останнім параметром цього API, який буде розглянуто є **max_tokens**. За допомогою нього можна встановити максимальну кількість токенів, яку буде містити результат надісланого POST запиту. Як стверджує OpenAI, один токен – це приблизно чотири символи. Стандартне значення цього поля рівне ліміту токенів, які може згенерувати за один раз модель, що використовується у запиті [23].

```
curl https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
    "model": "gpt-3.5-turbo",
    "messages": [{"role": "user", "content": "Hello!"}]
  }'
```

Рисунок 4.1 – Приклад запиту до OpenAI API використовуючи «curl»

Тепер перейдемо до огляду DALL-E API, яке доступне за наступним URL – «*https://api.openai.com/v1/images/generations*». Воно приймає значно менше параметрів ніж ChatGPT API, і тільки один з них є обов'язковим – це поле **prompt**, в якому потрібно вказувати опис того зображення, яке користувач має намір згенерувати. Дане API також має параметр **n**, який визначає, скільки зображень повинно створитись. Межі значень цього параметра знаходяться в діапазоні від 1 до 10 включно. Опція **size** встановлюється з метою задати розмір вихідних зображень. На даний момент DALL-E API підтримує тільки три розміри: 256x256, 512x512 (встановлений за замовчуванням) та 1024x1024. Як результат POST запиту повертається список з посилань на згенеровані зображення. Якщо є необхідність змінити тип значення, що повертається, на b64_json, то це можна зробити змінивши параметр **response_format** [23].

Насамкінець, оглянемо API сервісу Whisper та параметри запитів, які він може приймати. За допомогою параметру **file** треба передавати аудіо- чи відеофайл у форматі mp3, mp4, mpeg, mpga, m4a, wav, чи webm, який потрібно транскрибувати. Поле **model** слугує для вибору AI моделі, яка буде виконувати

задачу транскрибування. На даний момент існує тільки одна така модель – **whisper-1**. Знову ж таки, якщо потрібно змінити тип результату, що повертається, на json, text, str, verbose_json чи vvt, то це можна зробити встановивши відповідне значення параметру **response_format**. Останнім полем, яке варто відмітити, є **language**. За допомогою нього можна встановити мову у форматі ISO-639-1, на якій записана аудіодоріжка. Цей параметр є необов’язковим, як і response_format, але коректно його вказавши, можна значно збільшити точність результату, який буде згенерований [23].

РОЗДІЛ 5. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНИХ ЧАТ-БОТІВ ДЛЯ МЕСЕНДЖЕРІВ TELEGRAM ТА VIBER

5.1 Формування специфікації для чат-ботів

Специфікація програмного продукту є важливим документом в процесі його розробки. Вона містить детальний опис вимог до програмного продукту, його функціональності, характеристик і властивостей. Специфікація допомагає замовнику та розробнику зрозуміти, що потрібно розробити і як це повинно працювати перед тим як переходити власне до реалізації. Вона також забезпечує зрозумілість вимог та дозволяє уникнути непорозумінь та помилок у процесі розробки. Специфікація до ПЗ допомагає зосередитися на результаті та підвищує якість продукту, оскільки дозволяє визначити всі вимоги та параметри, які повинні бути виконані. Крім того, вона дозволяє визначити трудомісткість та вартість розробки програмного продукту, що є важливим для планування бюджету та термінів розробки, хоча це не є суттєво для даної кваліфікаційної роботи [26].

У специфікації описуються всі вимоги до програмного продукту, включаючи його функціональні та нефункціональні вимоги. Функціональні та нефункціональні вимоги – це дві ключові категорії вимог, які визначають очікувані характеристики системи. Вони відрізняються за природою та метою [27].

Функціональні вимоги описують конкретні функції або можливості системи, які необхідно реалізувати для задоволення потреб користувачів. Вони зазвичай включають дії, які система має виконувати, її поведінку та взаємодію з користувачами або іншими системами. Ці вимоги можуть включати:

- авторизацію та аутентифікацію користувачів;
- управління даними та їх обробку;

- пошук та фільтрацію інформації;
- інтеграцію з іншими системами тощо.

Нефункціональні вимоги відносяться до атрибутів системи, які характеризують її якість, ефективність або властивості, що впливають на користувацький досвід та відповідність вимогам різних стейкхолдерів. Нефункціональні вимоги можуть включати:

- швидкість відгуку системи, пропускну здатність тощо;
- частоту та тривалість збоїв, можливість відновлення після них;
- здатність справлятися зі збільшенням навантаження;
- захист від несанкціонованого доступу, зберігання даних тощо;
- здатність системи працювати з різними пристроями, операційними системами, браузерами.

Тож, для того, щоб мати чітке розуміння, який функціонал має бути в чат-ботах, та як вони мають працювати і взаємодіяти з користувачем, опишемо для них функціональні та нефункціональні вимоги. Розроблені в рамках кваліфікаційної роботи чат-боти повинні задовольняти наступним функціональним вимогам:

- Чат-боти повинні мати головне меню, через яке користувачі зможуть з ним взаємодіяти. Меню повинно містити наступні можливості: «Почати діалог з віртуальним асистентом», «Згенерувати зображення», «Зробити транскрипцію медіа-файлу», «Встановити OpenAI API-ключ», «Отримати допомогу по користуванню чат-ботом»;
- Користувач повинен надати свій OpenAI API-ключ перед тим, як отримати доступ до функціоналу системи;
- Користувач повинен мати можливість обирати роль асистента, з яким планує спілкуватись. Бот з відповідною роллю має якісніше відповідати на запитання, які пов’язані зі сферою його діяльності. Наприклад, чат-бот з роллю «Кухар» повинен без проблем надавати рекомендації для страв, які можна приготувати з певного набору продуктів;

- Система повинна запам’ятовувати історію повідомлень з користувачем в рамках відкритого діалогу і могти давати відповіді зважаючи на контекст розмови;
- Система повинна надавати можливість в будь-який момент завершити діалог та повернути користувача до головного меню чат-бота;
- Чат-бот повинен відповідати на повідомлення користувача, навіть якщо той знаходиться в головному меню програми, але у такому випадку не вимагається замап’ятовування історії діалогу та надання відповідей в залежності від її контексту;
- Telegram чат-бот повинен розпізнавати як текстові, так і голосові повідомлення від користувача. Для Viber бота достатньо тільки сприйняття текстових повідомлень (дане обмеження буде аргументоване в подальшому);
- Чат-боти мають надавати можливість згенерувати від одного до чотирьох зображень малого, середнього та великого розмірів за описом, який написав користувач;
- Чат-боти повинні мати функціонал транскрибування аудіо- та відеофайлів, які їм буде надсилати користувач;
- Чат-боти повинні надавати можливість отримати розгорнуту інструкцію про те, як ними користуватись;
- З будь-якого місця чат-бота має бути можливість повернутись в його головне меню.

Описані функціональні вимоги стосуються як Telegram, так і Viber чат-ботів з однією відмінністю у пункті про тип повідомлень, на які повинен реагувати бот. Viber асистент повинен відповідати тільки на текстові повідомлення, оскільки Viber заблокував можливість надсилати голосові повідомлення чат-ботам і це технічно неможливо імплементувати.

Нефункціональні вимоги, яким повинні задовольняти розроблені чат-боти:

- API-ключі користувачів повинні бути зашифровані та зберігатися в базі даних;
- Інтерфейс чат-ботів повинен бути зручним, зрозумілим та доступним для користувачів;
- Код системи повинен бути структурований та розділений на класи та модулі для подальшого легкого розширення та реалізації нових функцій;
- Чат-боти повинні працювати з різними версіями месенджерів та пристроями користувачів;
- Чат-боти повинні бути запущена на сервері і бути доступними цілодобово;
- Система повинна дотримуватися політик використання та правил OpenAI API.

5.2 Створення користувацького інтерфейсу

Користувацький інтерфейс (UI) – це спосіб взаємодії користувача з програмним продуктом, який включає в себе елементи інтерфейсу, такі як: кнопки, меню, поля вводу, таблиці, діалогові вікна тощо. Користувацький інтерфейс дозволяє користувачам взаємодіяти з програмним забезпеченням і виконувати його функції.

Важливість проєктування користувацького інтерфейсу перед розробкою полягає в ряді факторів, які спільно впливають на успішність та прийняття продукту. По-перше, добре спроєктований користувацький інтерфейс забезпечує зручність використання. Це означає, що користувачі можуть легко навчитися користуватися програмою, а це в свою чергу сприяє задоволенню користувачів. Незручний або нелогічний інтерфейс може призвести до відмови користувачів від продукту. Ефективність також є ключовою аспектом в проєктуванні UI. Чим більш ефективно користувач може виконувати свої завдання та досягати своїх цілей, тим більше йому буде подобатись продукт.

Також важливо щоб користувацький інтерфейс був привабливим і зрозумілим. Розробляючи UI, важливо враховувати очікування користувача щодо візуальної естетики та простоти використання, що може забезпечити приємний досвід від користування продуктом та сприяти його подальшій популярності [28].

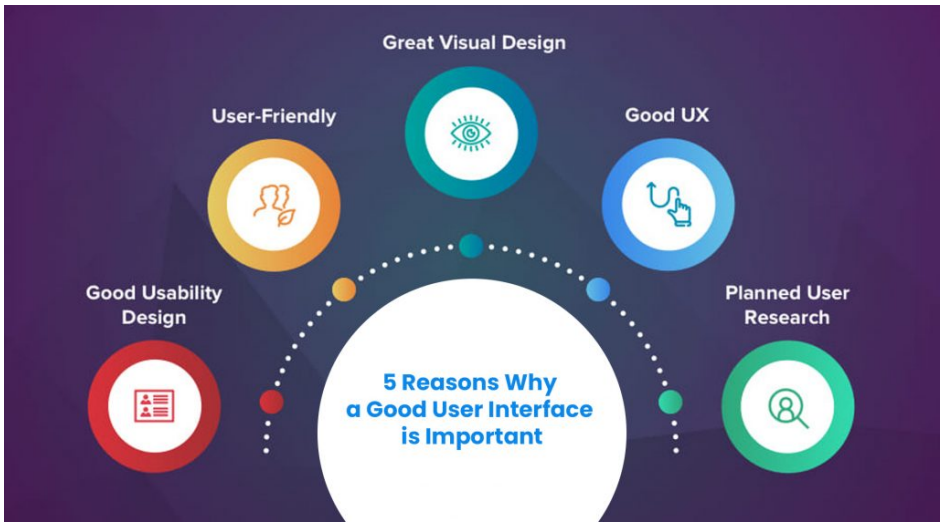

Рисунок 5.1 – 5 ознак хорошого користувацького інтерфейсу [29]

Тож, враховуючи важливість гарного та зручного користувацького інтерфейсу, перейдемо тепер до його створення для розроблюваних інтелектуальних чат-ботів.

Інтерфейс користувача в наших чат-ботах буде виконаний англійською мовою для того, щоб охопити значно більшу кількість аудиторії та складатиметься з різних станів, про які ми зараз поговоримо детальніше. Варто зазначити, що в рамках спілкування з віртуальним помічником, чат-бот може відповідати користувачу на різних мовах, в залежності від мови, на якій було вхідне повідомлення. Новому користувачу, який щойно натрапив на чат-бот і підписався на нього, завжди буде відображатись початкова сторінка з привітанням. У Telegram боті додатково буде інформація про опис доступних функціональних можливостей програми та кнопка «Розпочати», яка дозволяє почати користування чат-ботом (див. рис 5.2). Зазначимо, що на подальших

рисунках інтерфейс користувача Telegram чат-бота буде відображатись зліва, тоді як UI Viber чат-бота буде розташований справа.

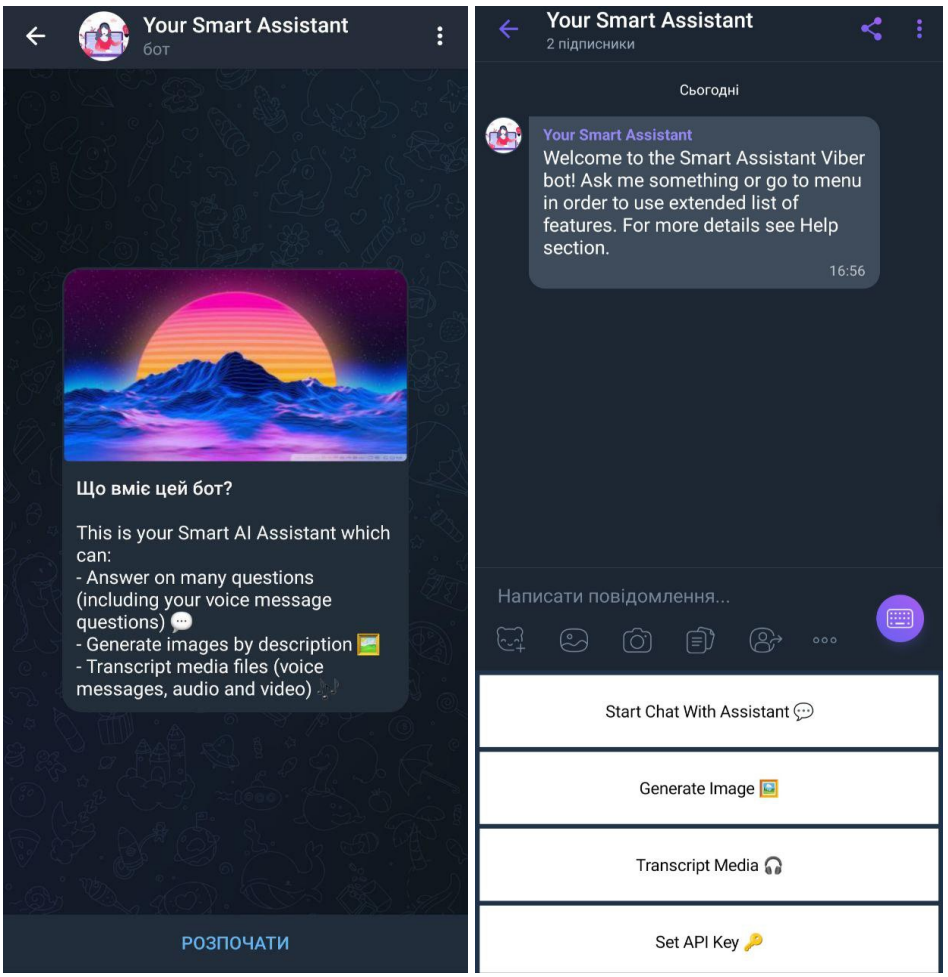

Рисунок 5.2 – Початкова сторінка чат-ботів

Після натискання на кнопку «Розпочати» у Telegram, чи на кнопку «Set Api Key» у Viber, користувачу буде запропоновано ввести свій OpenAI API-ключ, який необхідний для продовження користування системою.

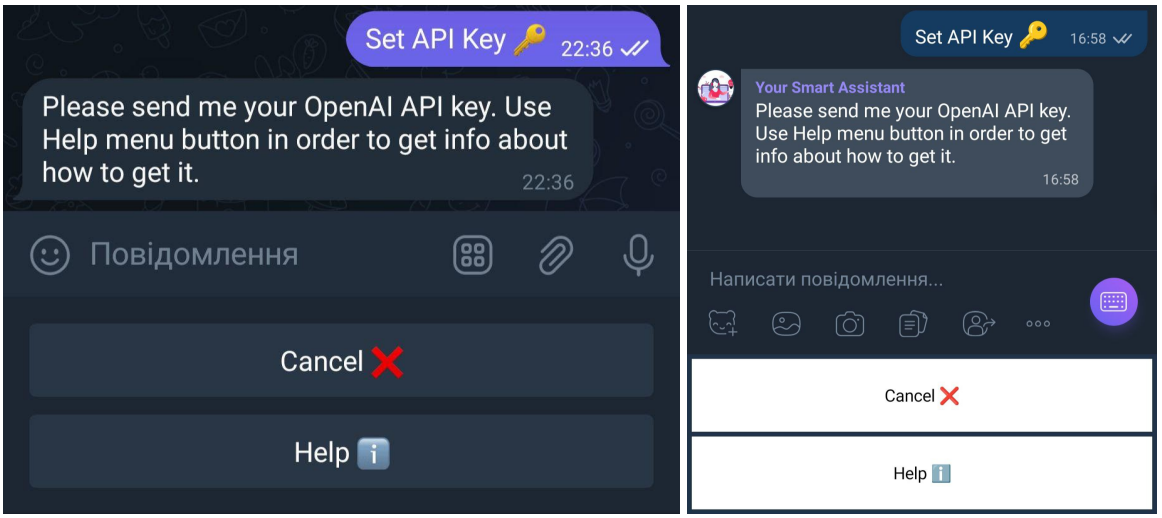

Рисунок 5.3 – Сторінка встановлення API-ключа користувача

Якщо ж користувач не знає, що таке API-ключ чи де його дістати – він завжди зможе скористатись допомогою. Після натискання на кнопку «Help» система надасть йому всю необхідну інформацію, яка може знадобитись у рамках користування чат-ботом. Приклад частини допоміжної інформації можна побачити на рис. 5.4.

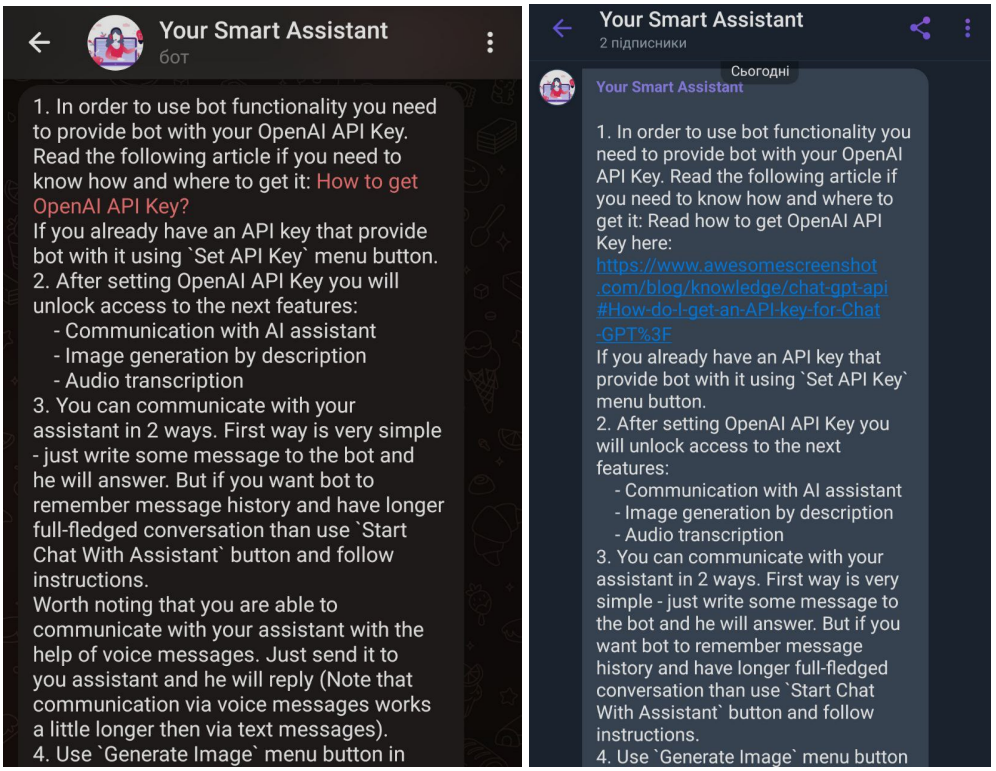

Рисунок 5.4 – Приклад допоміжної інформації для користувача

Після того, як користувач надасть системі свій API-ключ, йому відобразиться головне меню чат-бота (рис. 5.5), яке складається з наступних кнопок: «Start Chat With Assistant», «Generate Image», «Transcript Media», «Set OpenAI API-key», «Help» і стане доступний повний функціонал системи. Кнопка «Set OpenAI API-key» повинна бути присутня навіть після того, як користувач надав цей ключ на початку спілкування з чат-ботом, оскільки даний ключ може бути некоректно введеним і потребувати заміни, або ж користувач просто схоче використати якийсь інший ключ, наприклад, від свого іншого OpenAI облікового запису. Якщо якісь кнопки не поміщаються в області видимості меню – до них має бути можливість дістатись за допомогою прогортування вниз цього ж меню.

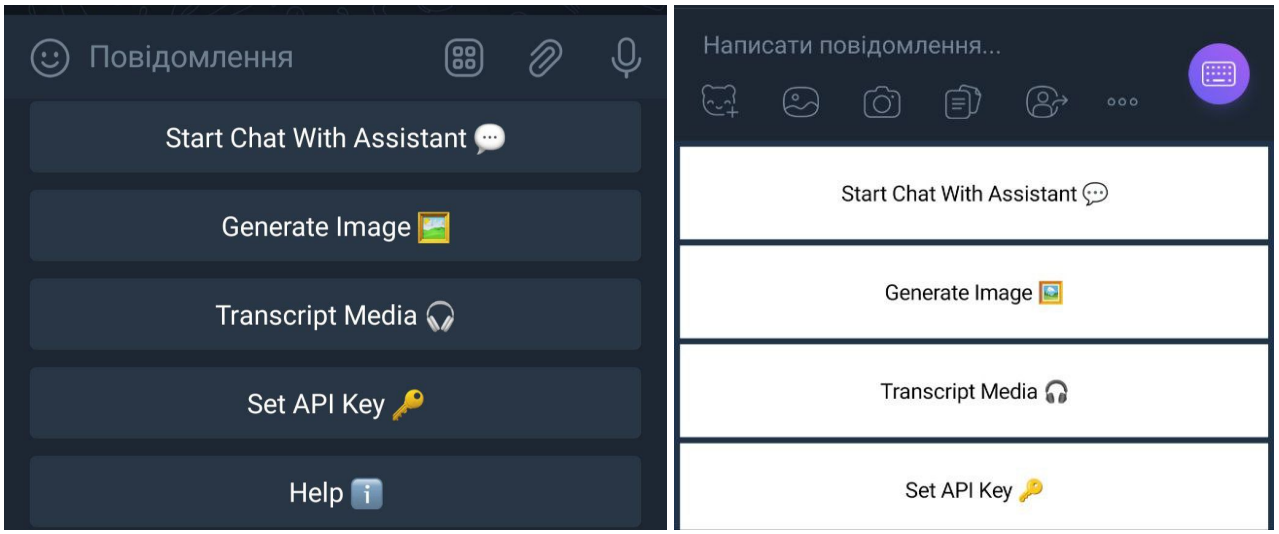

Рисунок 5.5 – Головне меню чат-ботів

Натискання на кнопку «Start Chat With Assistant» призведе до відкриття меню з вибором ролі віртуального помічника (рис. 5.6), з яким користувач має намір почати спілкування. На даний момент система підтримує наступні ролі: «чат-бот», «кухар», «лікар» «професійний спортсмен», «науковець» та «жартівник». Специфікація ролі повинна допомогти користувачу отримувати більш якісні відповіді на питання з тих чи інших галузей.

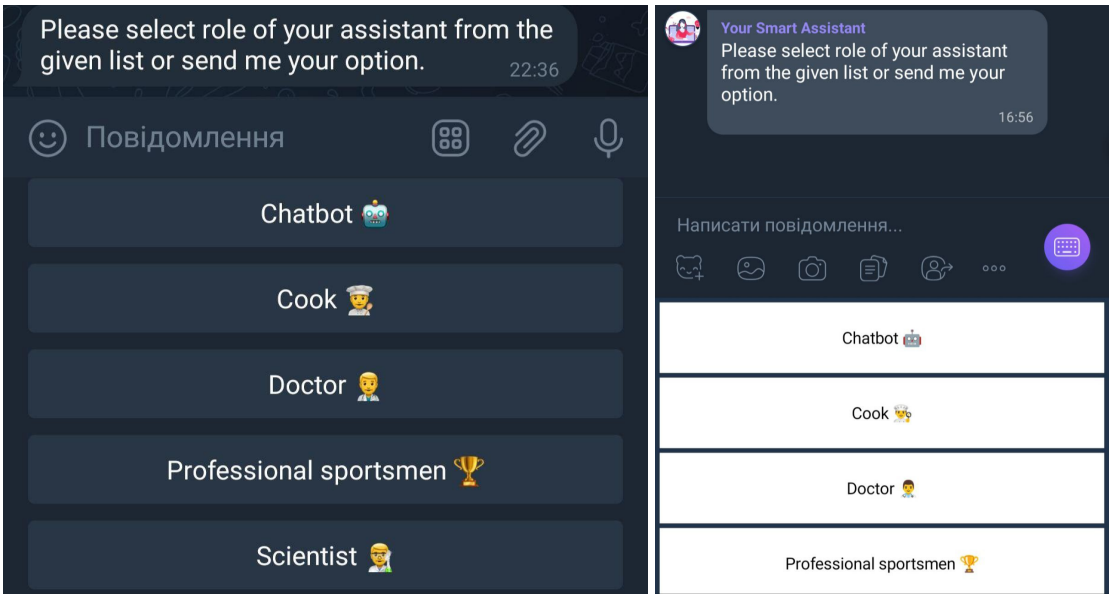

Рисунок 5.6 – Меню вибору ролі чат-бота

Після вибору ролі – чат з AI асистентом розпочнеться і користувачу буде запропоновано задавати будь-які питання, які його цікавлять. Для того щоб написати щось асистенту і отримати відповідь, достатньо просто відправити йому повідомлення і зачекати на відповідь (рис. 5.7). Спочатку чат-бот повинен надіслати повідомлення, у якому буде сказано, що вхідні дані від користувача отримані і асистент генерує на них відповідь, щоб не склалось враження, що бот завис чи не працює. Після генерування відповіді повідомлення про очікування повинно видалитись з діалогу, щоб не заповнювати його непотрібними речами, а відповідь надіслана людині, яка задала асистенту питання чи просто щось написала.

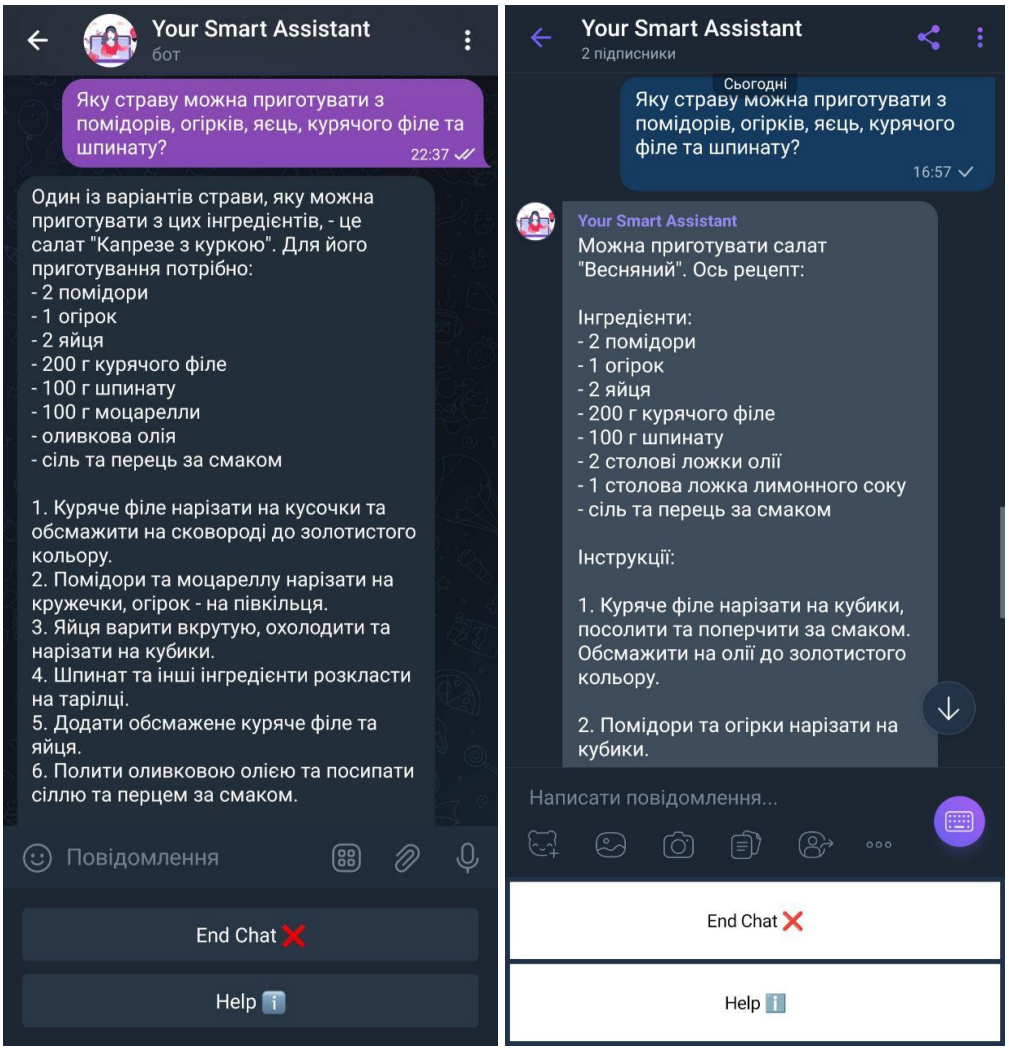

Рисунок 5.7 – Демонстрація діалогу з чат-ботами

Варто зазначити, що користувачі Telegram чат-бота повинні мати можливість спілкуватись зі своїм віртуальним асистентом за допомогою голосових повідомлень в рамках відкритого діалогу. Для цього достатньо просто записати голосове повідомлення і надіслати його асистенту. Формат відповіді має бути такий же, як і на текстове повідомлення (рис. 5.8).

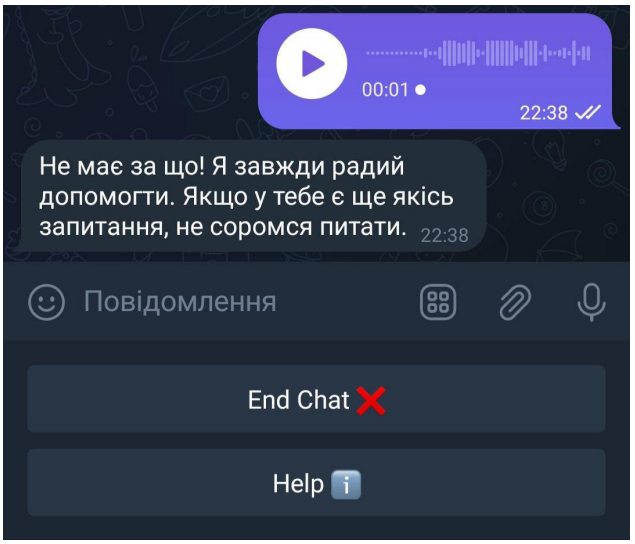

Рисунок 5.8 – Демонстрація відповіді Telegram бота на голосове повідомлення «Дякую» від користувача

Користувач може або продовжити спілкування з асистентом, або завершити його, натиснувши на кнопку «End Chat», яка має бути завжди доступною в рамках відкритого діалогу. Завершення чату поверне користувача до головного меню чат-бота (рис 5.5).

Тепер розглянемо наступний важливий функціонал розроблюваних чат-ботів – це створення зображень. Щоб почати цей процес, користувач повинен натиснути на кнопку «Generate Image» в головному меню, в результаті чого йому буде запропоновано ввести опис зображення, яке він хоче, щоб асистент намалював (рис 5.9).

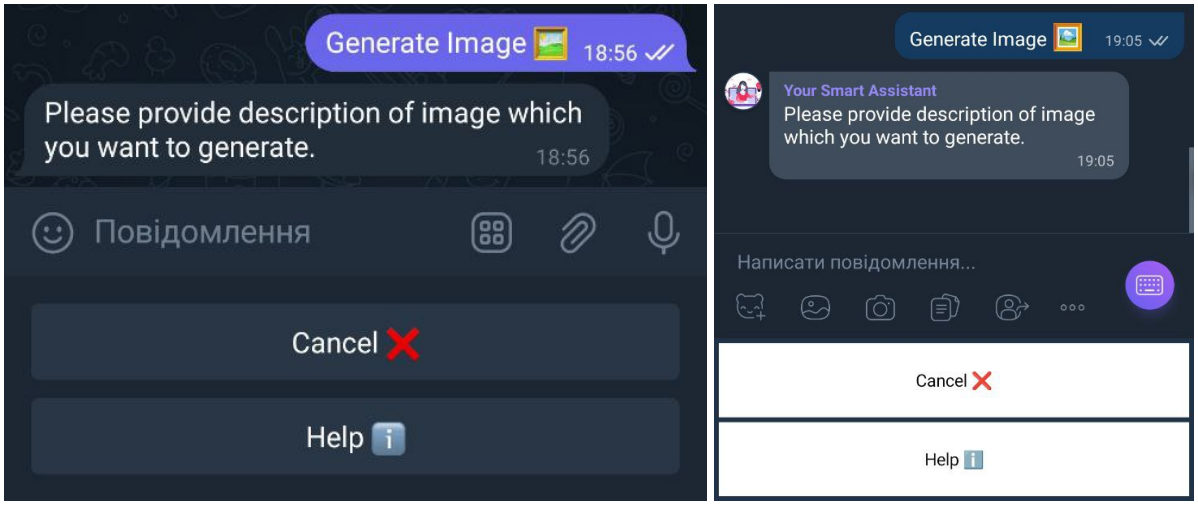

Рисунок 5.9 – Меню вводу опису зображення

Як тільки чат-бот отримав та зберіг опис, він одразу ж повинен відобразити меню з вибором кількості зображень, які користувач бажає створити (рис 5.10). У меню повинні бути опції вибору від одного до чотирьох зображень. Цей ліміт обумовлений сервісом DALL-E, який, власне, і відповідає за створення цих малюнків.

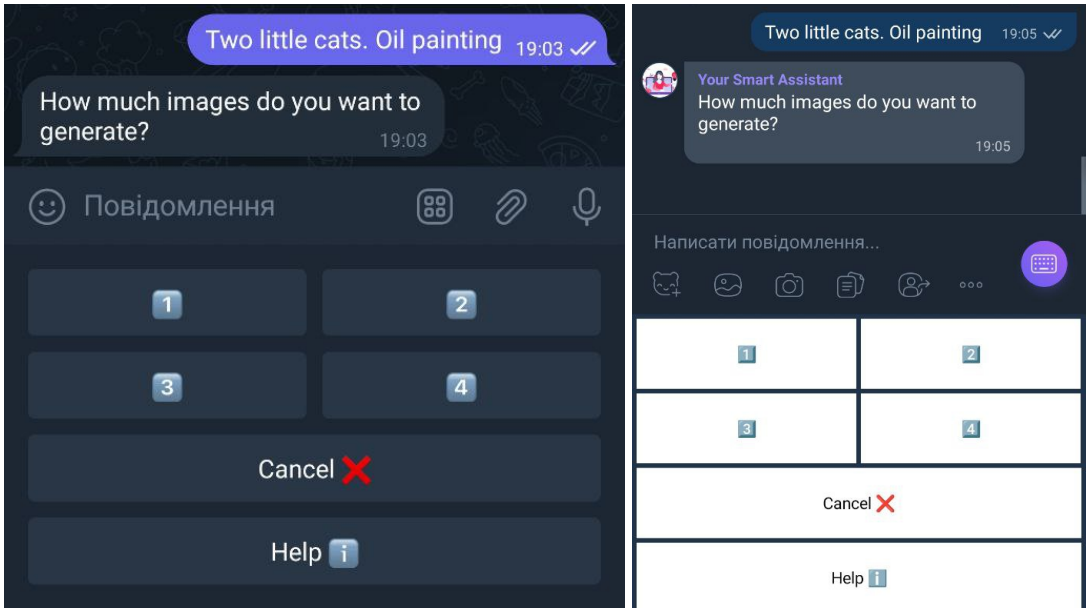

Рисунок 5.10 – Меню вибору кількості зображень

Наступним кроком після вибору кількості картинок має бути вибір їх розміру (рис. 5.11). Користувачу має бути запропонована можливість обрати, яке зображення він бажає згенерувати: мале, середнє чи велике.

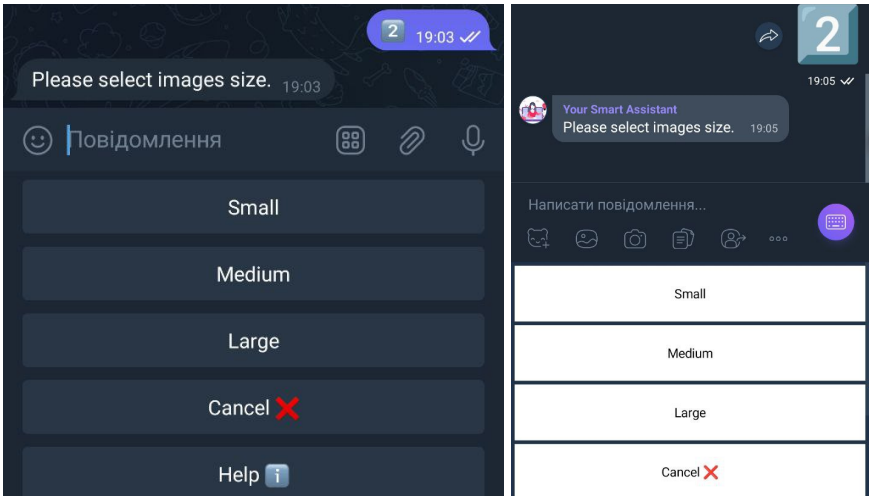

Рисунок 5.11 – Меню вибору розміру зображень

Ці три розміри містять під собою наступні відповідні піксельні значення: 256x256, 512x512, 1024x1024. Дані цифри знову ж таки є обмеженнями сервісу DALL-E. І, насамкінець, після того як користувач обрав розмір зображення, він повинен отримати сповіщення про те, що зображення генерується і потрібно трохи зачекати. Як тільки чат-бот буде мати ці малюнки – він повинен зразу надіслати їх користувачу і повернути його в головне меню (рис 5.12). Відмітимо, що якщо користувач вирішив створити більше ніж одну картинку, то Telegram бот повинен надіслати йому групу згенерованих зображень, в той час як Viber бот – кожне по окремості, оскільки – це обмеження даного месенджера.

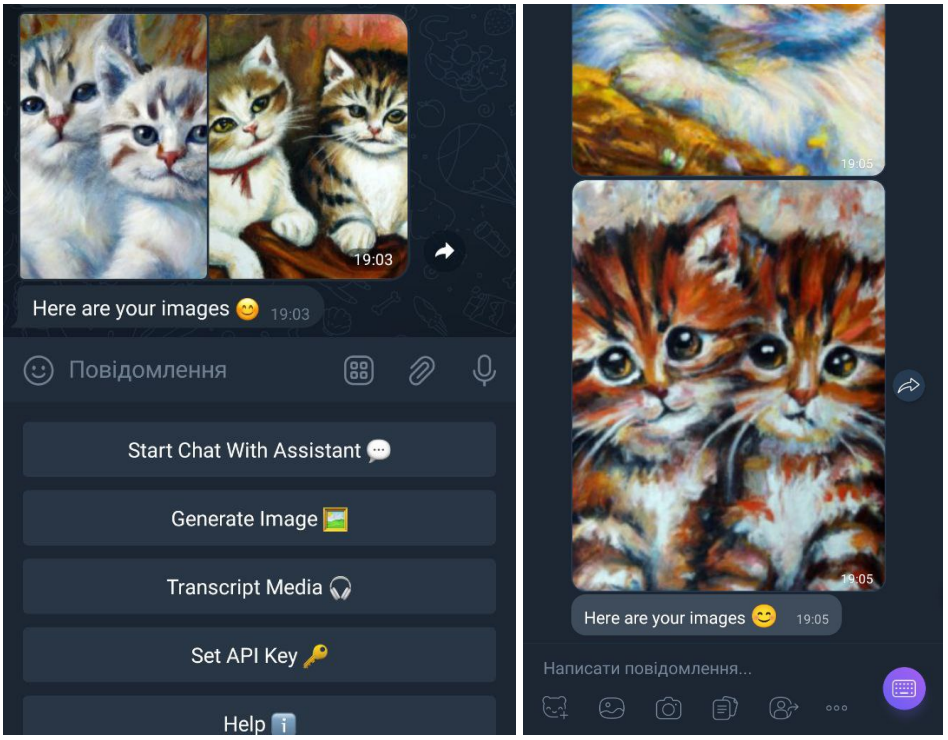

Рисунок 5.12 – Результат створення зображень розроблюваними чат-ботами

Ну і наостанок розглянемо користувацький інтерфейс для функціоналу, який дозволяє здійснювати транскрипцію аудіо та відеофайлів. За це відповідає кнопка «Transcript Media» в головному меню обох чат-ботів. Натискаючи на неї, користувачу буде запропоновано надіслати медіафайл, який він бажає транскрибувати (рис. 5.13). Система додатково повинна уточнити формати файлів, які підтримуються, щоб не вводити в оману користувача.

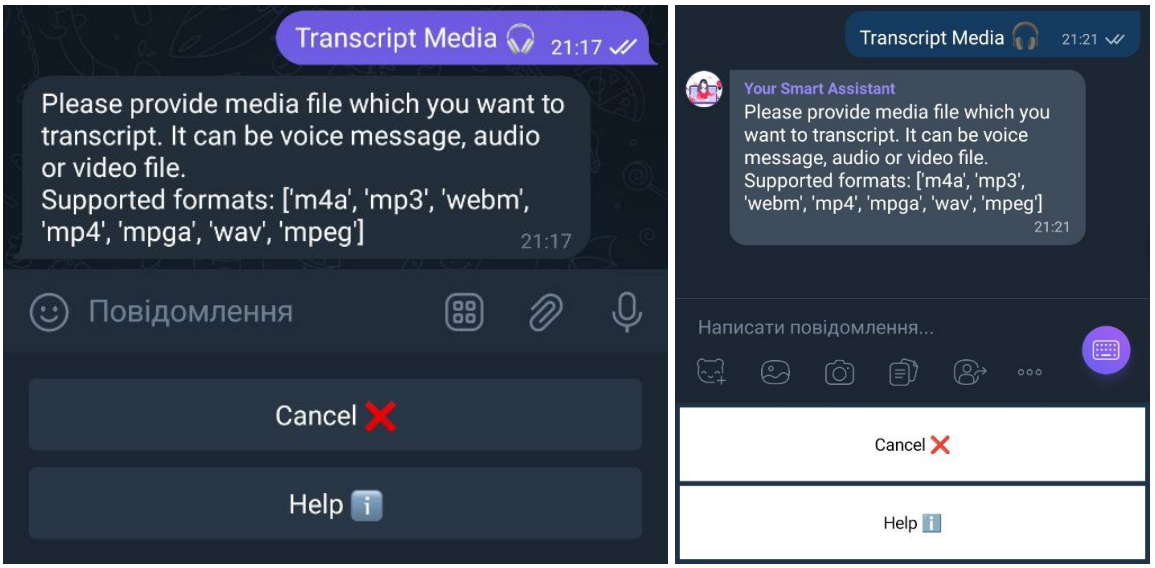

Рисунок 5.13 – Меню надсилання медіафайлу для транскрибування

Після того, як користувач надішле цей файл, система повідомить його, що потрібно трішки зачекати і як тільки результат транскрибування буде готовий – одразу ж надішле його у відповідь. Приклад транскрибування голосового повідомлення до Telegram бота та записаного аудіофайлу для Viber бота з однаковим вмістом, можна побачити на рис. 5.14.

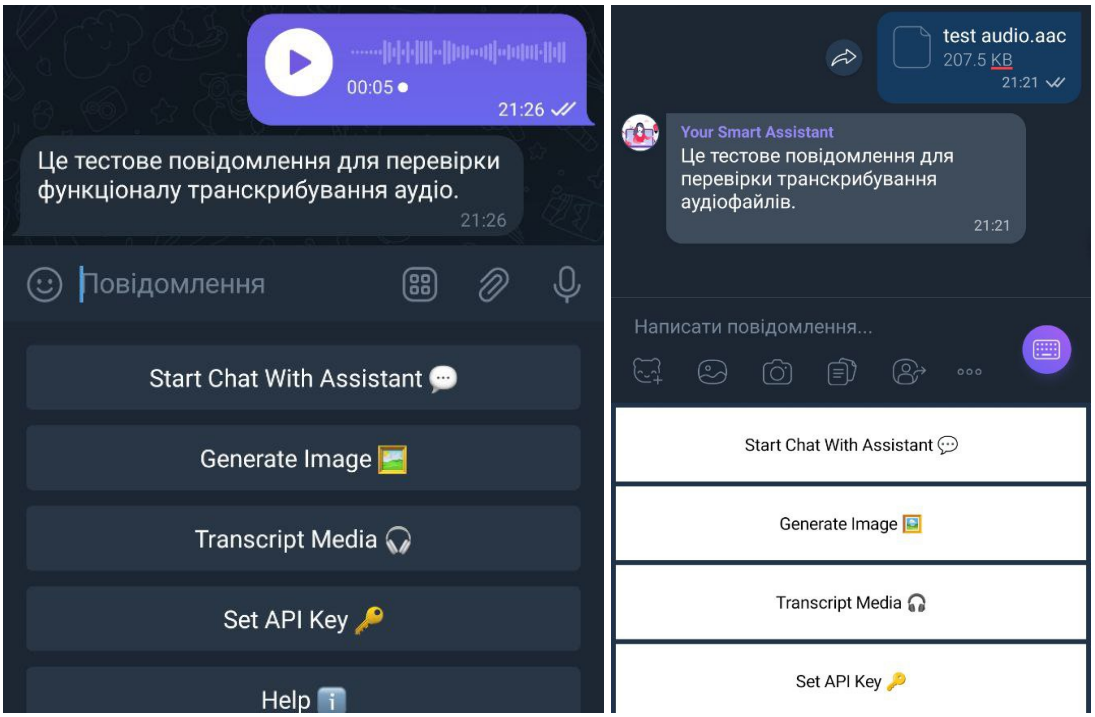

Рисунок 5.14 – Приклад результату транскрибування аудіофайлів

Також одразу після отримання результатів, користувач буде повернений в головне меню системи.

Варто зазначити, що в будь-який момент користувачу повинна бути доступна можливість отримати інформацію про те, як користуватись чат-ботами. Для цього кнопка «Help» повинна бути присутня у всіх можливих меню. Аналогічно у всіх меню (крім головного) повинна бути доступна кнопка «Cancel», натискаючи на яку, завжди можна буде повернутись до початкової сторінки чат-ботів, незважаючи на те, в якому стані вони зараз знаходиться. Також біля тексту на кожній кнопці має бути присутнє зображення, яке асоціюється з дією, яку виконує ця кнопка. Наприклад, червоний хрест біля кнопки «Cancel» чи «End Chat» або значок інформації біля кнопки «Help». Це допоможе зробити користувацький інтерфейс інтуїтивно зрозумілішим і візуально приємнішим.

5.3 Вибір технологій для розроблювальних систем та їхня характеристика

Після проєктування UI для чат-ботів, саме час обрати технології, за допомогою яких вони будуть створюватись. Технологічний стек для IT-проєкту – це наче будівельні матеріали для будинку. Дуже важливо правильно обрати технології для розробки програмного забезпечення, оскільки коректно підібрані технології можуть полегшити процес розробки та забезпечити високу продуктивність роботи команди розробників. Використання підходящих мов програмування, бібліотек та інструментів може прискорити розробку, зменшити кількість помилок, що можуть виникати в її процесі, спростити внесення змін та поліпшити якість створюваного коду. Крім того, правильний вибір стеку технологій сприяє стабільності та надійності розроблюваного продукту [30].

Мовою програмування для імплементації даних чат-ботів була обрана **Python**, оскільки – це популярна мова програмування, яка широко

використовується в наукових обчисленнях, аналізі даних і машинному навчанні. Це універсальна мова, яку можна використовувати для широкого кола завдань, включаючи розробку чат-ботів [6]. Існує кілька причин, чому ця мова є найкращим вибором для даної мети. По-перше, Python має простий та зрозумілий синтаксис, який дозволяє швидко та легко створювати програми. Завдяки своїй структурі та синтаксису, навіть новачки можуть легко зрозуміти код та розширювати функціонал чат-ботів. Другим важливим аспектом є наявність великої кількості бібліотек та фреймворків для Python, які спрощують розробку чат-ботів для різних платформ [31]. Враховуючи дані переваги, було обрано саме цю мову програмування. За допомогою неї можна реалізувати всю логіку чат-ботів та взаємодію з OpenAI API.

Оскільки чат-боти повинні зберігати деякі дані користувачів (наприклад, OpenAI API-ключі), то варто також обрати базу даних, яка буде використовуватись для цієї мети. Для даних проєктів було обрано **SQLite –** легку і компактну СУБД, яка зберігає дані в локальному файлі, підтримує стандартні SQL запити та має багато корисних функцій, що дозволяють працювати з даними ефективно та безпечно. Вона не вимагає спеціальної конфігурації чи налаштувань, і може працювати на різних операційних системах, таких як Windows, Linux, та Mac OS. Крім того, SQLite не вимагає окремого сервера баз даних, що робить її ідеальним вибором для невеликих проєктів, такі як чат-боти. Ще одним плюсом SQLite є її швидкість та ефективність. Оскільки вона зберігає дані в локальному файлі, запити до бази даних оброблюються швидше, ніж у серверних базах даних. Крім того, SQLite підтримує багато корисних функцій, які дозволяють працювати з даними ефективно та безпечно. Наприклад, вона має вбудований механізм захисту даних та підтримує транзакції, що дозволяє забезпечити цілісність даних та уникнути помилок при виконанні запитів. Ну і насамкінець, варто відзначити, що SQLite має інтеграцію з більшістю мов програмування і чудово працює з Python, що дозволяє використовувати її в різних проєктах та середовищах розробки [32].

Одна з описаних раніше нефункціональних вимог до систем полягає у тому, що OpenAI API-ключі користувачів повинні зберігатись у базі даних у зашифрованому вигляді, з метою збільшення безпеки даних користувачів і запобіганню їх витоку. Для цієї цілі був обраний модуль для шифрування та розшифрування даних в Python – **Fernet** [33]. Саме він використовуватиметься для захисту API-ключів користувачів під час зберігання в базі даних SQLite.

Також, зважаючи на функціональні вимоги до розроблюваних чат-ботів, вони повинні володіти функціоналом ведення діалогу з користувачем, генерації зображень по заданому опису та транскрибування аудіо- та відеофайлів. Саме для цих цілей будуть використовуватись відповідні сервіси компанії OpenAI, такі як: **ChatGPT**, **DALL-E** та **Whisper**. Детальніший їх опис та огляд може бути знайдений в розділі 4 даної кваліфікаційної роботи.

Ще однією невід'ємною частиною чат-ботів стане **FFmpeg –** відкрита програма, яка дозволяє записувати, конвертувати та обробляти аудіо- та відеозаписи. FFmpeg використовуватиметься у даній системі для перетворення голосових повідомлень чи інших медіафайлів від користувача у підтримуваний формат перед транскрибуванням за допомогою Whisper API від OpenAI. Як ми вже знаємо, сервіс Whisper здатен обробляти тільки файли наступних форматів: m4a, mp3, webm, mp4, mpga, wav, mpeg, тому важливо було знайти технологію, яка забезпечує сумісність різних типів мультимедійних файлів, що надсилаються і, тим самим, зробить чат-боти більш зручними у користуванні [34].

Розглянуті вище технології є спільними як для Telegram, так і для Viber чат-ботів, але в силу того, що принцип їх створення для цих месенджерів значно відрізняється, потрібно буде використовувати для цього різні методи.

Для створення Telegram ботів з використанням мови програмування Python існує безліч бібліотек, кожна з яких має свої переваги та недоліки. Після детального аналізу багатьох з них, вибір для розроблюваного чат-бота зупинився на «**python-telegram-bot**» версії 20.2, оскільки вона є однією з найбільш популярних та ефективних бібліотек, які використовуються для цієї

мети. «python-telegram-bot» надає розробникам різноманітні інструменти для обробки повідомлень, команд та інших подій, що надходять від користувачів. А завдяки високій абстракції, дана бібліотека дозволяє зосередитися на бізнес-логіці та взаємодії з користувачем, замість поглиблення в технічні аспекти роботи з Telegram API. Більше того, ця технологія дуже добре задокументована і знаходиться у відкритому доступі, що значно спрощує її освоєння та дозволить створити Telegram бот з бажаним функціоналом швидко і якісно [35].

Для створення Viber ботів немає такого різноманіття інструментів, як для Telegram, тому вибір впав на офіційну бібліотеку «**viberbot**» версії 1.0.11, яку рекомендує використовувати сама компанія Viber у своїх матеріалах. Вона також знаходиться у відкритому доступі, є безкоштовною і має непогану документацію інтерфейсу [19]. Оскільки чат-бот для месенджеру Viber повинен бути розроблений у вигляді веб-сервера, варто ще обрати інструмент для створення веб-додатків. Після детального аналізу наявних варіантів, вибір зупинився на веб-фреймворку для Python – **Flask**. Цей модуль надає інструменти, бібліотеки та технології, які дозволяють з легкістю створювати веб-додатки різної складності та масштабу. Фреймворк Flask належить до категорій мікрофреймворків – фреймворків, які майже не залежить від будь-яких зовнішніх бібліотек. Це, звичайно, має як свої переваги, так і недоліки. Переваги полягають у тому, що фреймворк є легковаговим, у нього існує тільки невелика залежність від оновлення та спостереження за помилками безпеки. Недоліком в свою чергу є те, що інколи доводиться виконувати більше роботи самостійно щоб розширювати список потрібних залежностей, додаючи плагіни [36]. Але, зважаючи на те, що для Viber чат-боту ніяких додаткових залежностей і не потрібно, Flask стає ідеальним вибором для цієї задачі.

Безсумнівно, кожна з цих технологій відіграє важливу роль у розробці системи та дозволяє створити надійні, безпечні та ефективні чат-боти, які будуть відповідати розробленим функціональним і нефункціональним вимогам

і, найголовніше, потребам користувачів.

5.4 Створення архітектури чат-ботів

Перед, власне, переходом до написання коду чат-ботів, варто розробити їхню архітектуру для того, щоб отримати загальні уявлення про те, як все має бути влаштовано, і, щоб уникнути переписування певних модулів безпосередньо під час розробки, через те, що вони були погано сплановані. Архітектура програмного забезпечення – це певний план, який описує основні компоненти системи, їх взаємозв'язки та взаємодію з оточенням, таким як бази даних тощо. Архітектура відіграє дуже важливу роль у процесі розробки кожного продукту, оскільки визначає всі рішення щодо дизайну, стандартів, шаблонів та інших аспектів, які безпосередньо впливають на функціональність, стабільність, масштабованість, майбутню підтримку та успішність створюваної системи [37].

Для виконання цієї задачі скористаємося засобами UML. UML розшифровується як Unified Modeling Language і є сучасним підходом для моделювання та документування програмного забезпечення. Їхня мета – створити візуальне представлення системи для того, щоб краще її розуміти та спростити подальшу підтримки навіть для тих розробників, які не знайомі з кодом продукту. Принцип візуалізації полягає у створенні діаграм різних типів (в загальному їх є 14), кожен з яких представляє різний рівень абстракції [38]. Для даних чат ботів більш ніж достатнім буде створення трьох типів діаграм: Use Case, Class та Sequence. Про них ми й поговоримо далі.

Use Case діаграма (див. рис. 5.15) представляє найвищий рівень абстракції і показує, яку функціональність надає сервіс для користувача. Користувач в даній діаграмі позначається спеціальним символом у вигляді людини, яка називається «Actor». Сама ж функціональність описується текстом в фігурі овальної форми. Користувачі з’єднуються лініями з цими фігурами [38].

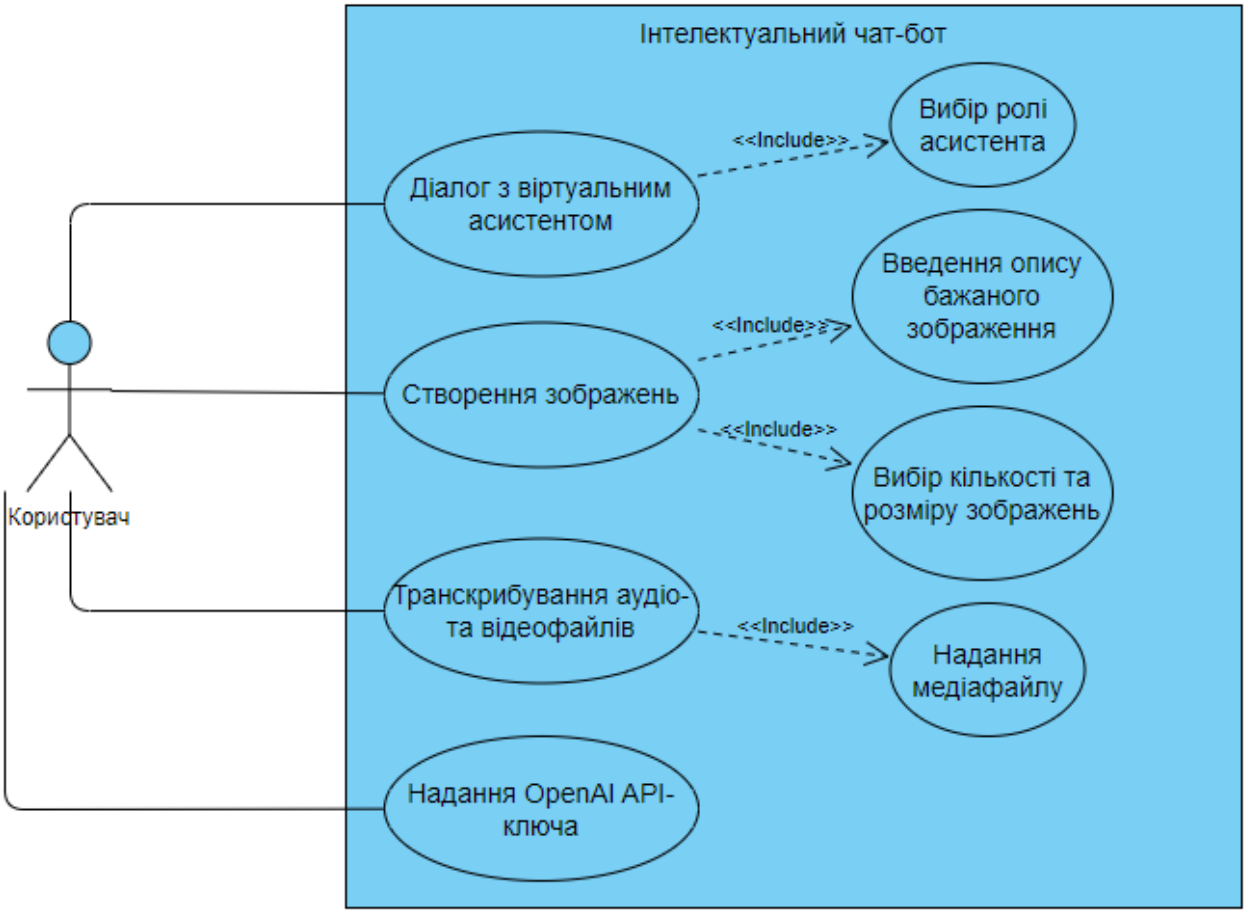

Рисунок 5.15 – Use Case діаграма розроблюваних чат-ботів

З даної діаграми можна побачити, що система може надавати користувачу можливість діалогу з віртуальним асистентом, включаючи вибір його ролі. Також користувач здатний створювати зображення по заданому опису, обрати кількість цих зображень та їх розмір. Також діаграма показує, що чат-бот надає можливість транскрибування аудіо- та відеофайлів. Для цього клієнту достатньо просто надати відповідний медіафайл.

Тепер перейдемо до побудови Class діаграми розроблюваних чат-ботів (див. рис. 5.16), яка відображає всі класи, що присутні в даному коді, та поверхневі взаємозв’язки між ними. Кожен клас представляється у вигляді прямокутника, назва якого відповідає назві класу. В середині нього записуються атрибути та функції даного класу. Класи поєднуються між собою певними

залежностями, як, наприклад: агрегація, композиція, реалізація та наслідування [38].

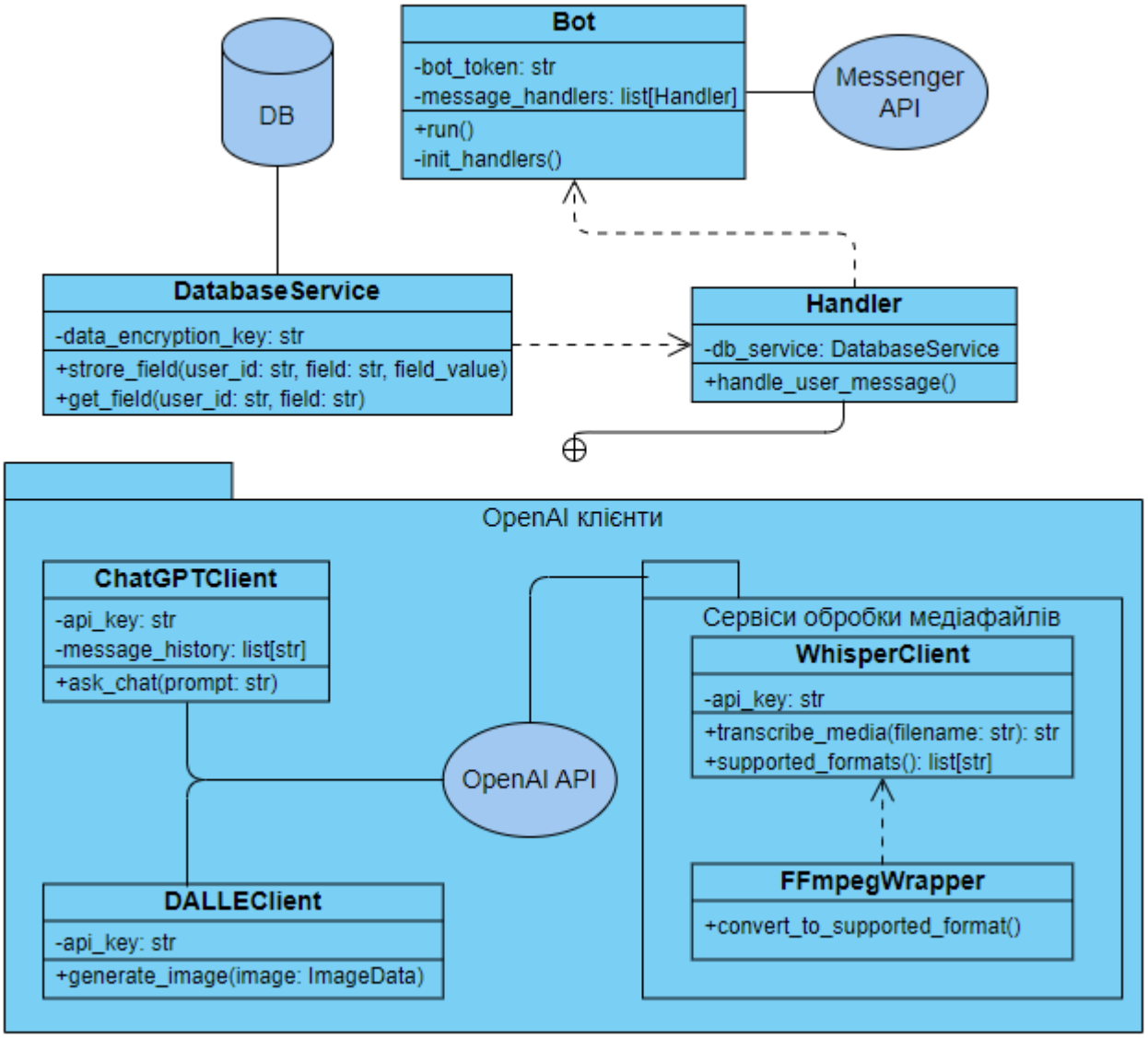

Рисунок 5.16 – Діаграма класів розроблюваних чат-ботів

Bot – це основний компонент (клас) розроблюваної системи, який взаємодіє з користувачами через Telegram чи Viber API. Для цієї мети він він використовує бібліотеки «python-telegram-bot» та «viberbot» відповідно, які дозволяють обробляти повідомлення від користувачів та надсилати їм зворотні відповіді. Коротше кажучи, цей компонент забезпечує взаємодію між

користувачами та різними обробниками, які на даній діаграмі позначені як Handlers.

Handlers, в свою чергу – це окремі функції, що відповідають за обробку різних команд та дій, які надходять від користувачів. Для досягненні цієї мети обробники використовують наступні, створені в рамках кваліфікаційної роботи, сервіси: ChatGPTClient, DALLEClient, WhisperClient та DatabaseService. Також Handlers взаємодіють з базою даних з метою зберігання та отримання певних даних користувача, таких як, наприклад, API-ключ та після обробки вхідного повідомлення повертають результат, який Bot відправляє користувачу.

ChatGPTClient у даній архітектурі – це клас, який відповідає за взаємодію з ChatGPT API. Він обробляє запити на генерацію відповіді від сервісу ChatGPT на повідомлення, які надходять до нього від певного обробника. Варто зазначити, що даний клас також зберігає історію всіх повідомлень між користувачем та ChatGPT в рамках відкритого діалогу, яку потім використовує для того, щоб мати змогу генерувати відповіді для користувача в залежності від контексту розмови.

DALLEClient – це клас, який являється обгорткою для OpenAI DALL-E API, яке дозволяє створювати картинки за поданим описом, і надає зручний інтерфейс для роботи з ним. Цей клас містить всього один метод, який приймає на вхід опис зображення, яке потрібно згенерувати, їх кількість та розмір, а на виході повертає список із посилань, за якими ці зображення можуть бути знайдені. Обробники повідомлень, в свою чергу, надсилають ці посилання екземпляру класу Bot, який створює повідомлення відповідного типу і надсилає його користувачам.

WhisperClient відповідає за взаємодію з Open AI Whisper API, який дозволяє отримати транскрипцію для аудіо- чи відеофайлу певного формату. Він обробляє запити на транскрибування аудіо чи відео від Handlers, результат яких в подальшому буде повернений користувачу або використаний для подальшої взаємодії з ChatGPT у Telegram боті. Whisper API підтримує медіа файли наступних форматів: mp3 , mp4 , mpeg , mpga , m4a , wav та webm. А оскільки

голосові повідомлення від користувача в Telegram мають формат .oga, то приходиться їх конвертувати у підтримуваний формат. Саме цим і займається FFmpegWrapper. На вхід він приймає аудіофайл певного формату і повертає аудіофайл у форматі .wav, який в свою чергу вже підходить для Whisper API. Для конвертації аудіофайлів FFmpegWrapper використовує програму з відкритим доступом – ffmpeg.

Насамкінець розглянемо клас DatabaseService. Він використовується для роботи з SQLite базою даних і відповідає за зберігання та отримання різних даних користувачів, таких як OpenAI API-ключі тощо. Окрім, власне, зберігання та отримання цих ключів, DatabaseService також забезпечує їх шифрування перед зберіганням у базі даних за допомогою модуля Fernet, що є частиною криптографічної бібліотеки. Відповідно після отримання ключа з бази даних, відбувається його дешифрування. Fernet працює за принципом симетричного шифрування. Обробники повідомлень використовують DatabaseService для отримання та зберігання різних даних користувачів, що необхідні для коректної роботи чат-ботів.

Підсумовуючи, можна сказати, що взаємодія між класами відбувається наступним чином:

1) Користувач відправляє повідомлення або команду через певний месенджер;
2) Bot отримує повідомлення та передає його відповідному Handler;
3) Handler аналізує повідомлення та викликає потрібні методи з класів: ChatGPTClient, DALLEClient, WhisperClient чи DatabaseService, щоб виконати відповідні дії;
4) Після отримання результатів від цих класів, Handler формує відповідь для користувача та передає її назад класу Bot;
5) Bot, у свою чергу, надсилає відповідь користувачу через Viber чи Telegram API.

Тепер перейдемо до Sequence діаграм, які є одними з найбільш важливих типів діаграм і дають змогу змоделювати поведінку окремих частин програми.

Вона детально описує послідовність взаємодій між користувачами та об’єктами. Варто відзначити, що ця послідовність має бути описана в строго хронологічному порядку [38]. Створимо декілька Sequence діаграм, які будуть відображати послідовність взаємодій між всіма компонентами чат-ботів при наступних користувача: встановлення OpenAI API-ключа (див. рис. 5.17), початок та ведення діалогу з чат-ботом (див. рис. 5.18), транскрипція аудіо- та відеофайлів (див. рис. 5.19) і створення зображень (див. рис. 5.20).

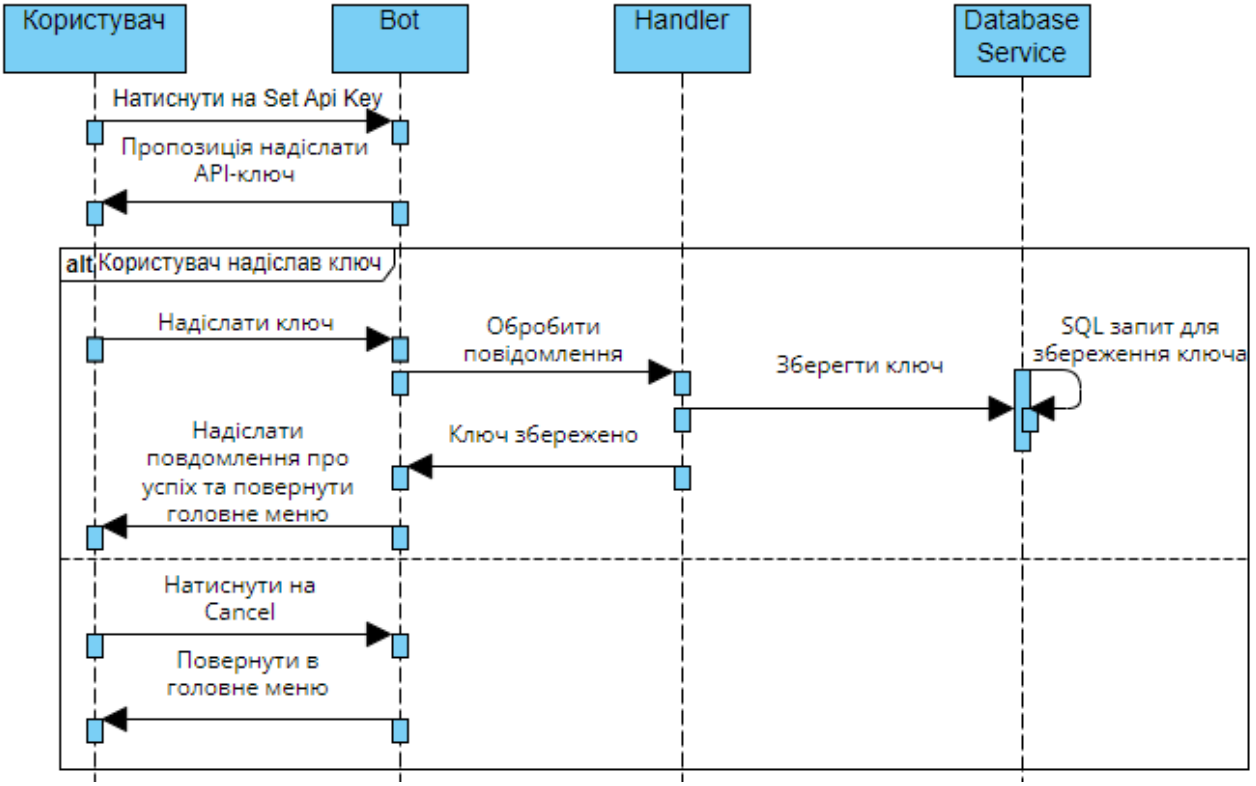

Рисунок 5.17 – Sequence діаграма для надання OpenAI API-ключа користувача

На рис. 5.17 можна побачити взаємодію компонентів чат-ботів під час процесу надання користувачем свого OpenAI API-ключа. Після того як користувач натисне кнопку «Set Api Key», йому буде запропоновано ввести його або скасувати операцію. Якщо ключ було надіслано, Bot передасть його обробнику, який в свою чергу звернеться до DatabaseService з метою зберегти цей ключ. Після того Handler повідомить Bot про успішне збереження ключа – користувач буде повернений в головне меню чат-бота і повідомлений про успішне збереження даних. Якщо ж користувач вирішить скасувати операцію – він

просто буде переміщений назад в меню. Надалі буде розглянуто тільки позитивні сценарії поведінки користувача при використанні функціоналу розроблюваних систем.

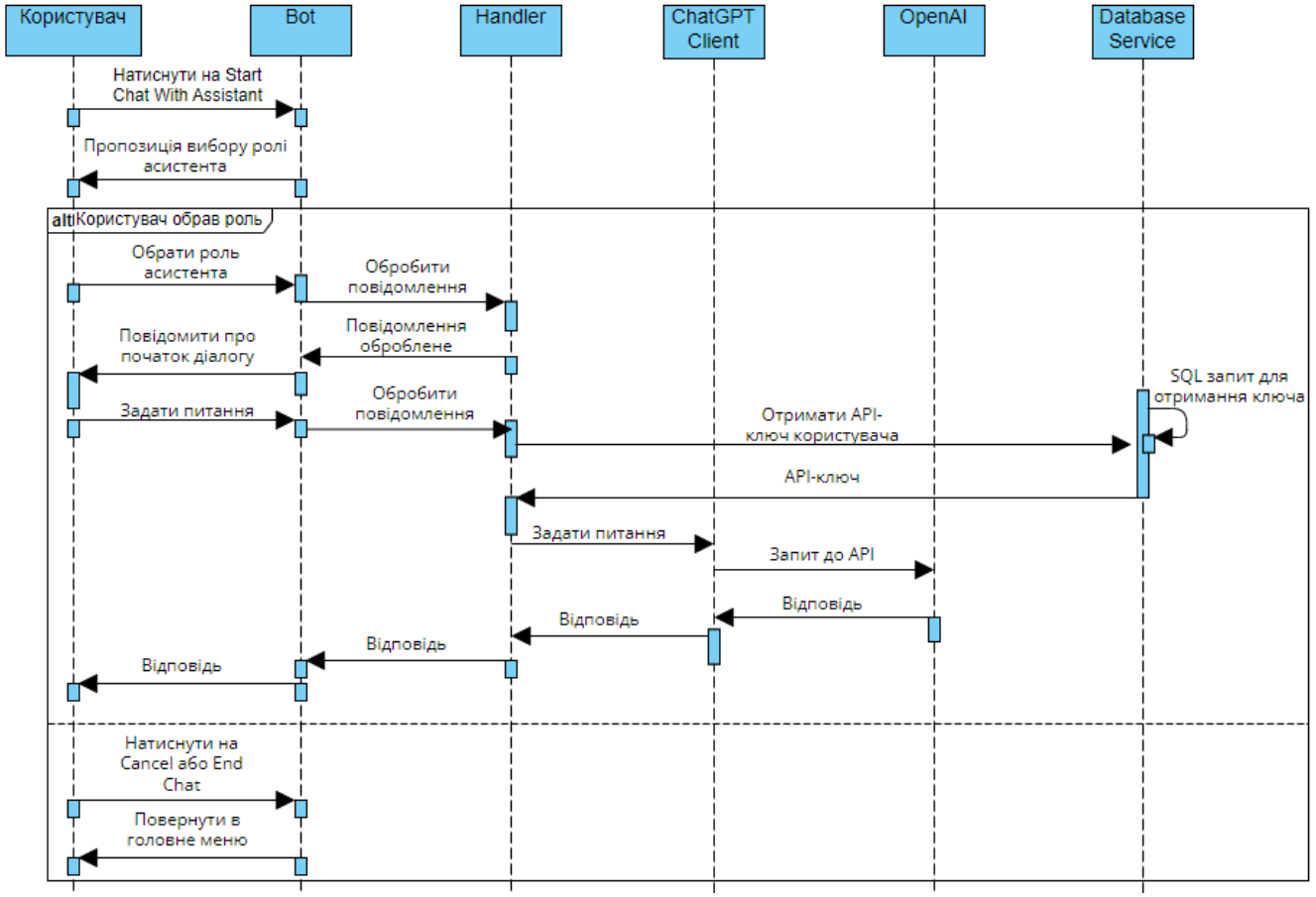

Рисунок 5.18 – Sequence діаграма для початку діалогу з віртуальним інтелектуальним помічником

На рис. 5.18 відображено комунікацію класів чат-ботів під час початку діалогу з розумним асистентом. Після того як користувач натисне кнопку «Start Chat With Assistant», йому буде запропоновано обрати роль свого співрозмовника. Як тільки роль буде обрана, то буде повідомлено про те, що чат-бот готовий приймати повідомлення від користувача і відповідати на них. Цей процес працює наступним чином: користувач надсилає чат-боту якийсь текст, той у свою чергу передає це повідомлення відповідному обробнику. Handler робить запит на отримання OpenAI API-ключа користувача до DatabaseService, оскільки він необхідний для роботи з ChatGPT API. Далі Handler створює об’єкт

класу ChatGPTClient і викликає його метод, який відповідає за отримання відповіді на текстове повідомлення. Як тільки відповідь від API була повернута, клієнт повертає її обробнику, той віддає її класу Bot, який в свою чергу, надсилає це повідомлення користувачу. Якщо користувач вирішить продовжити діалог, то цей процес буде відбуватись циклічно до того моменту, коли розмова не буде завершеною. Після завершення розмови, користувач буде повернений в головне меню.

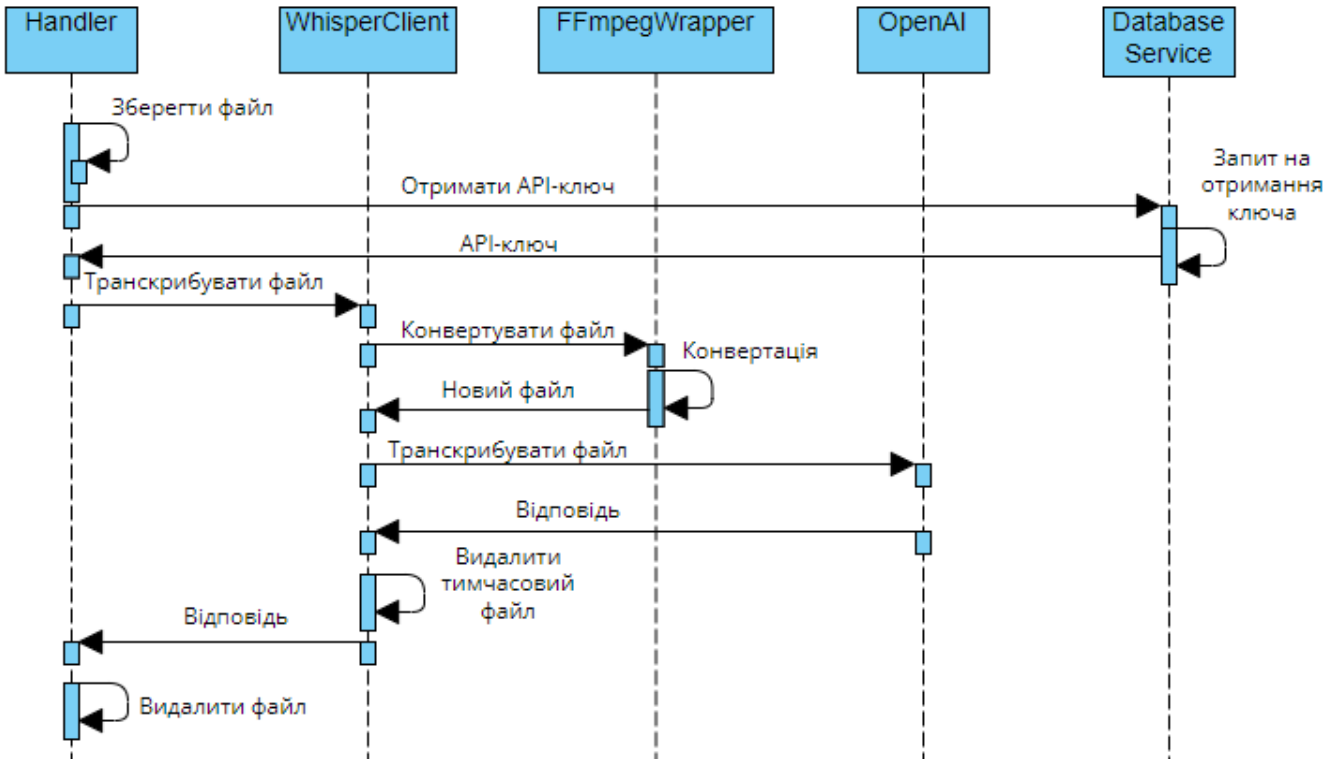

Рисунок 5.19 – Sequence діаграма для процесу транскрибування файлів

На рис. 5.19 описано детальну послідовність процесу транскрибування файлів. Медіафайл, який було отримано від користувача, спочатку зберігається в локальному сховищі на пристрої, де запущений чат-бот. Після цього знову ж таки, як і в процесі діалогу з ChatGPT, відбувається запит OpenAI API-ключа, бо сервіс Whisper також його потребує для виконання своїх функцій. Після отримання ключа, Handler викликає метод транскрибування файлів у класу WhisperClient. Якщо формат файлу підтримується даним клієнтом, тоді відбувається звернення до API з метою транскрибування аудіо- чи відеофайлу, а

інакше Whisper клієнт викликає функцію класу FFmpegWrapper, яка конвертує файл користувача у підтримуваний формат, і вже після цього WhisperClient робить запит до відповідного OpenAI API. Отримана відповідь буде миттєво надіслана користувачу, а всі локальні файли, які були створені для виконання операції транскрибування, будуть видалені в цілях безпеки та економії пам’яті на пристрої. Варто відзначити, що на даній діаграмі опущено частину взаємодії з користувачем, оскільки вона вже була описана у попередніх двох прикладах і в даному випадку нічим не відрізняється.

Спілкування з віртуальним асистентом за допомогою голосових повідомлень у месенджері Telegram не вимагає додаткових діаграм, оскільки цей процес вже описаний в попередніх двох діаграмах і являє собою комбінацію транскрибування голосового повідомлення та звернення до ChatGPT API з отриманим результатом з подальшою передачею користувачу відповіді від API.

Останній функціонал, який залишилось розглянути – це створення зображень (див. рис. 5.20).

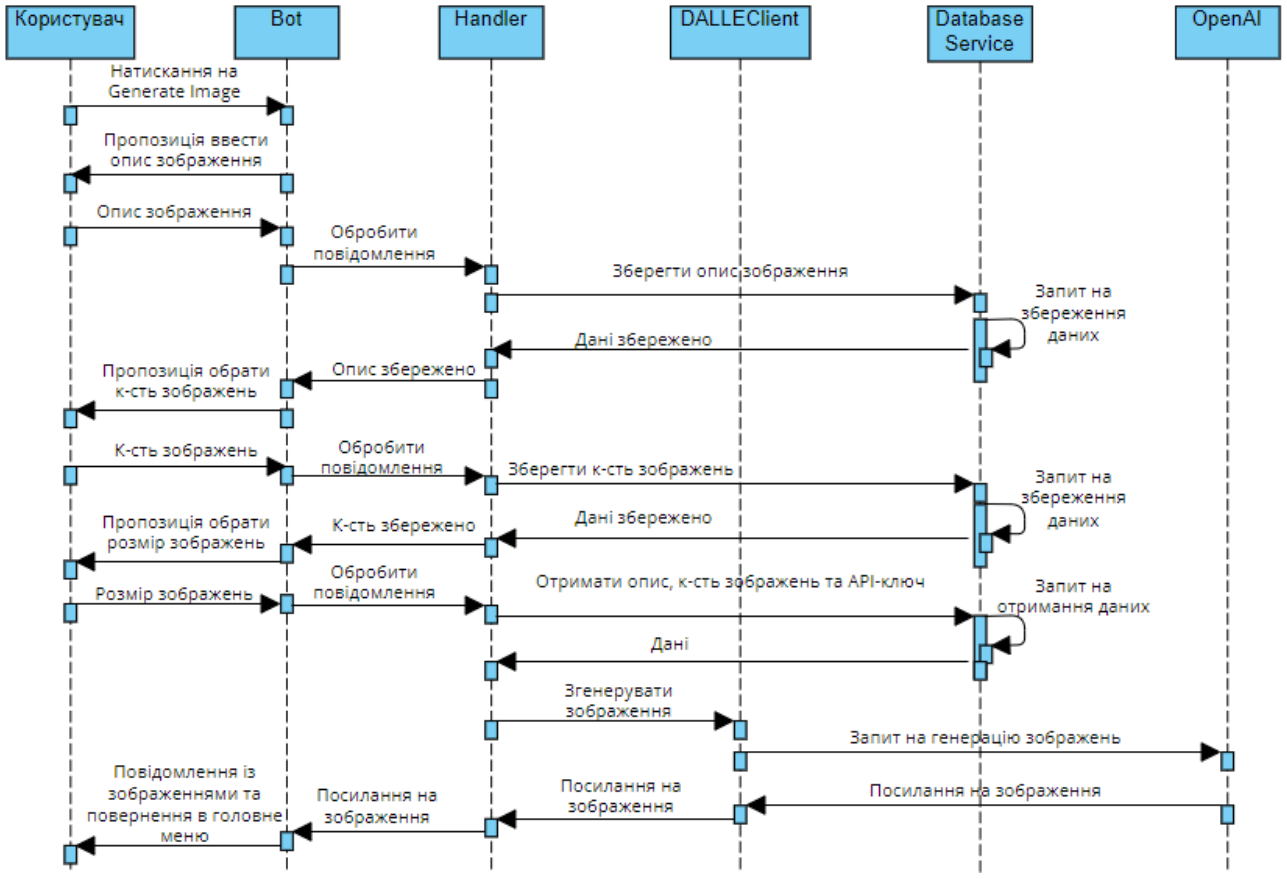

Рисунок 5.20 – Sequence діаграма для процесу створення зображень за поданим описом

Поглянувши на дану діаграму можна помітити, що вона дещо об’ємніша за попередні. Це обумовлено тим, що процес створення картинок складається з більшої кількості кроків, які зараз розглянемо детальніше. Насамперед, після натискання на кнопку «Generate Images» в головному меню чат-ботів, користувачу буде запропоновано ввести опис зображення, яке той хоче згенерувати. Як тільки він його введе, Bot передасть це повідомлення обробнику повідомлень, а той, в свою чергу, використовуючи DatabaseService, збереже цей опис в базі даних. Потрібно запам’ятати цей опис, оскільки до того моменту, як почнеться, власне, створення малюнків, потрібно отримати від користувача ще деяку уточнюючу інформацію. Далі користувач аналогічно до опису надає всі інші необхідні дані: розмір та кількість. Кількість зображень також зберігається в базі даних, а от розмір – ні, оскільки він є останнім вводом від користувача і може бути збережений тільки в пам’яті програми. На етапі надання розміру Handler робить запит до DatabaseService з метою отримання опису та кількості зображень, які були збережені на попередніх етапах, а також OpenAI API-ключа користувача, який знадобиться пізніше для надсилання запитів до API. Як тільки вся необхідна інформація є отриманою, обробник повідомлень викликає функцію генерації зображень DALLEClient класу, а той у свою чергу посилає відповідний запит до OpenAI DALL-E API. Як результат буде повернено посилання на створені картинки, які будуть надіслані по зворотному ланцюгу до класу Bot, який перетворить отримані посилання на повідомлення відповідного типу, надішле їх користувачу та поверне його до головного меню чат-ботів.

Розроблена архітектура забезпечує гнучкість та масштабованість, дозволяючи легко додавати нові функції та інтегрувати інші сервіси. Кожен клас має свою чітку зону відповідальності, що сприяє зручному управлінню кодом та його подальшому легкому розширенню.

5.5 Особливості реалізації інтелектуальних чат-ботів для Telegram та Viber

Оскільки в попередньому розділі була повністю розроблена та описана архітектура створюваних чат-ботів, настав час перейти до певних особливостей в їхній реалізації.

Перш за все варто відмітити, що в проєктах було використане логування всіх подій та процесів, які відбуваються під час взаємодії користувача та продукту. Логи зберігаються у спеціальних файлах, і містять інформацію про дії користувачів, операції системи, зовнішні запити, можливі помилки та інші події, які допомагають розробникам та аналізувати стан програми та її роботу. Отримані записи можна використовувати для перегляду будь-яких подій у системі, включаючи збої та зміни стану. Впровадження логування обумовлене багатьма причинами. Воно допомагає виявити та вирішити проблеми та помилки, які можуть виникнути під час роботи чат-ботів, оскільки записи логів повинні містити детальну інформацію про контекст та причину проблеми, що полегшує її діагностику та виправлення. Також логування важливе для контролю за забезпеченням якості програмного забезпечення. Воно дозволяє розробникам перевіряти, чи працює програма так, як від неї очікують, та визначати області, які можуть потребувати оптимізації або поліпшення [39, 40].

Як вже було зазначено раніше, кожен чат-бот, незалежно він месенджеру, повинен мати свій унікальний токен, який являє собою рядок із символів, за допомогою якого можна буде керувати ботом із розроблюваної програми. Він має бути доступним тільки обмеженому колу людей, а ще краще одній людині, яка володіє цим чат-ботом, оскільки токен є ключем доступу до бота. Будь-хто, хто його отримає, зможе керувати відповідним ботом. Тому критично необхідно, щоб дана інформація була схована від сторонніх осіб і категорично заборонено зберігати її десь в коді програми. Тому було прийнято рішення, помістити цей токен в змінну середовища операційної системи під назвою «BOT_TOKEN» і забороняти запуск бота, якщо такої змінної не існує. Якщо ж вона існує, тоді

програма зчитує її значення та використовує в подальшій роботі. Аналогічна ситуація склалась із ключем для шифрування OpenAI API-ключів користувачів, який використовується для шифрування API-ключа при його збереженні в базу даних та розшифруванні при його отриманні звідти. Його також було вирішено тримати в окремій змінній середовища під назвою «API_KEYS_DB_ENCRYPTION_KEY», оскільки це значно збільшує безпеку даних користувачів.

Розробку даних інтелектуальних чат-ботів було вирішено проводити як три різні проєкти, які зберігаються в трьох окремих Github репозиторіях. В одному з них знаходиться бізнес-логіка чат-бота для месенджеру Telegram, в другому – бізнес-логіка чат-бота для месенджеру Viber, а в третьому розміщений код OpenAI клієнтів, який був винесений в окремий проєкт, тому що він використовується в обох чат-ботах. Для того, щоб уникнути копіювання коду цих клієнтів, було вирішено використовувати репозиторій з їхнім кодом як окремий git підмодуль в проєктах із чат-ботами. Git підмодуль – це функціональність у системі контролю версій Git, яка дозволяє включати один репозиторій Git в якості піддиректорії в інший репозиторій. Це корисно в тому випадку, коли потрібно використовувати зовнішню бібліотеку або компонент в своєму проєкті та забезпечити легке оновлення цього компонента з оригінального репозиторію [41]. Дане рішення значно спростило внесення змін в код OpenAI клієнтів та кардинально пришвидшило інтеграцію цих змін в обидва чат-боти.

Тепер трохи детальніше розглянемо імплементацію сервісів, які дозволяють обробляти текстові повідомлення від користувачів та надавати відповіді на них. З функціональних вимог ми знаємо, що чат-боти повинні відповідати на запити користувачів в головному меню без зберігання історії спілкування, та зберігати її в рамках відкритого діалогу з асистентом, з метою надання відповідей, які враховують контекст розмови. Для цих цілей було вирішено використовувати дві різні моделі OpenAI: **text-davinci-003** та **gpt-3.5-turbo**. Перша з них використовується для надання відповіді на окремі

повідомлення, оскільки є більш до цього пристосованою. Вона може генерувати відповідь на будь-який текстовий запит з кращою якістю, довшим результатом і послідовним виконанням інструкцій, ніж моделі curie, babbage або ada. text-davinci-003 натренована даними, датованими до червня 2021 року та здатна видавати до 4097 токенів за одну відповідь. Один токен – це, приблизно, 4 літери в слові. Для діалогу зі збереженням історії було обрано іншу OpenAI модель – gpt-3.5-turbo, оскільки вона є потужнішою та економічною моделлю в сімействі GPT-3.5, яка була спеціально оптимізована для чату за 1/10 від вартості text-davinci-003. Це означає, що використання моделі gpt-3.5-turbo коштує в десять разів менше, ніж використання моделі text-davinci-003. Таким чином, gpt-3.5-turbo пропонує оптимальне співвідношення вартості та якості для реалізації чат-ботів та інших сценаріїв розмови. gpt-3.5-turbo натренована даними, датованими до вересня 2021 року та здатна видавати до 4096 токенів за одну відповідь. Варто відзначити, що існує більш потужна та нова модель чату – gpt-4. GPT-4 – це найновіша модель сімейства GPT, яка здатна вирішувати складні проблеми з більшою точністю, ніж будь-яка з попередньо описаних моделей, завдяки своїм ширшим загальним знанням і розширеним міркуванням. Проте, вона досі знаходиться в обмеженому бета-доступі і доступна тільки для невеликої кількості користувачів [42].

Насамкінець, розглянемо структуру баз даних обох чат-ботів.

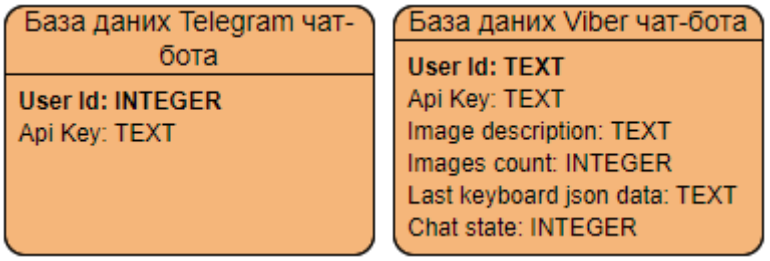

Рисунок 5.21 – Структура таблиць баз даних для Telegram та Viber чат-ботів

Як можна побачити, у них є дві спільні колонки: унікальний ідентифікатор користувача (UUID) та OpenAI API-ключ. Єдина відмінність між цими двома колонками у тому, що ідентифікатор користувача Telegram є цілочисельним

числом, в той час як UUID користувача Viber являється текстовим рядком. Зберігання API-ключа в базі даних вимагається специфікацією проєкту і обумовлюється тим, що після перезапуску чат-ботів з різних причин, користувачу не доведеться вводити його заново. Також, дивлячись на рис. 5.21, можна помітити, що таблиця бази даних Viber чат-бота зберігає значно більше інформації: опис зображень, кількість зображень, стан чату та дані про клавіатуру, яка останньою була надіслана користувачу. Почнемо розглядати ці відмінності з кінця. Месенджер Viber має одну цікаву особливість, яка відсутня у Telegram: після того, як користувач надсилає чат-боту якесь повідомлення чи просто натискає на будь-яку кнопку на існуючій клавіатурі – Viber примусово видаляє меню користувача, оскільки вважає його одноразовим. В результаті цього потрібно після кожної взаємодії користувача з ботом повертати йому це меню з різними кнопками, які були до видалення, і саме для цього воно повинне зберігатись в базі даних. Тепер розглянемо такі колонки як: опис зображень, кількість зображень та стан чату. Вони використовуються як в одному чат-боті, так і в іншому для різних цілей. Але існує відмінність в місці, де ці дані зберігаються. У випадку Telegram бота, вся ця інформація може бути збережена в спеціальному місці, яке надається бібліотекою «python-telegram-bot» і називається – «chat_data». Механізм зберігання інформації в даній локації є дуже зручним та ефективним. Там можна зберігати примітивні типи даних, такі як: цілочисельні типи, текстові рядки тощо, так і користувацькі типи, як, наприклад, екземпляри класу ChatGPTClient. На жаль, у бібліотеці для створення Viber чат-ботів таке сховище відсутнє, тому для цієї мети прийшлось використовувати саме базу даних, оскільки вона вже була інтегрованою в проєкт.

5.6 Порівняння розроблених чат-ботів з аналогами

Почнемо з порівняння створеного Viber бота з аналогами. Було знайдено два чат-боти зі схожим функціоналом: «IvanGPT» та «AI Chat & Create». Перший з них має 161 підписників та володіє тільки функціоналом ведення діалогу з користувачем. При чому, цей діалог є постійним, і немає можливості його почати чи завершити. Варто відмітити, що «IvanGPT» також не зважає на історію повідомлень між користувачем та асистентом і не надає відповідей в залежності від контексту розмови. Також даний чат-бот зовсім не має користувацького меню та не пропонує жодної допомоги в експлуатації. При початку користування ним не зовсім зрозуміло, що потрібно робити, оскільки будь-яка допоміжна інформація відсутня. Насамкінець, варто зазначити, що у даного чат-бота відсутня можливість специфікації ролі віртуального асистента, що робить його менш гнучким.

Тепер перейдемо до огляду «AI Chat & Create», що нараховує 100 тисяч 423 підписники. Він є дуже популярним, володіє значно більшою кількістю функціоналу та дозволяє: вести діалог з віртуальним асистентом, створювати зображення по заданому опису, генерувати випадковий малюнок та поділитися чат-ботом з друзями. Для виконання попередньо описаних функцій існують чотири відповідні кнопки в головному меню: «Ask a Question», «Create Art», «Inspire Me» та «Share with Friends», які значно покращують користувацький досвід під час використання чат-бота. «AI Chat & Create» також не надає можливості обирати роль асистента, з яким відбувається діалог, проте він зберігає контекст розмови і чудово справляється з відповідями на питання, що стосуються тих чи інших попередніх повідомлень з діалогу. Даний чат-бот дозволяє створити зображення по заданому опису, але тільки одне та тільки фіксованого розміру. Можливість вибору цих параметрів відсутня.

Порівнювати розроблений Telegram чат-бот будемо з наступними аналогами: «GPT4Telegrambot» та «gpt3_unlim_chatbot». При першому ж повідомленні «GPT4Telegrambot» надає користувачу детальну інструкцію щодо

його експлуатації. За допомогою нього можна вести діалог з віртуальним асистентом (без вибору ролі), генерувати зображення за заданим описом та створювати коротке резюме відеоролика на Youtube. Як стверджує автор, для імітації спілкування з віртуальним асистентом використовується модель gpt-3.5-turbo, а створювати зображення можна за допомогою двох сервісів: DALL-E та Midjourney. Проте, щоб користуватись Midjourney, необхідно купити преміум підписку у автора. Чудовою функцією даного чат-бота є створення резюме відеоролика на Youtube. Це можна зробити надіславши асистенту команду «/summary» та відповідне посилання. «GPT4Telegrambot» також дозволяє спілкуватись з асистентом за допомогою голосових повідомлень, але для цього, знову ж таки, потрібно купити преміум підписку. Також, варто зазначити, що у даного чат-бота повністю відсутнє меню і користуватись ним можна тільки виконуючи відповідні команди, що є не дуже комфортно для користувача.

Насамкінець, розглянемо «gpt3_unlim_chatbot». У нього є головне меню, за допомогою якого можна робити наступні речі: спілкуватись з віртуальним асистентом, обирати його роль та модель ChatGPT, яка буде використовуватись при спілкуванні. Також даний чат-бот єдиний надає можливість зміни мови користувацького інтерфейсу. Не зовсім зрозумілою є функція вибору GPT моделі, оскільки на даний момент наявні тільки дві опції: gpt-3.5 та gpt-3.5-turbo, яка є найновішою моделлю, що доступна в OpenAI ChatGPT API. Було б доцільно завжди її використовувати і не надавати інших варіантів, оскільки в цьому немає ніякого сенсу. «gpt3_unlim_chatbot» також надає можливість створювати одне, три чи п’ять зображень по заданому опису, проте ця функція є доволі неочевидною, оскільки вона захована глибоко в меню, а не розташована десь на поверхні. Даний асистент зберігає історію повідомлень, а також надає можливість очистити її в будь-який момент. Щодо ролей співрозмовника, то в меню їх доступно тільки дві: «Programmer» та «Default», проте є можливість надіслати будь-яку іншу роль, яка цікавить користувача.

«gpt3_unlim_chatbot» сприймає тільки текстові повідомлення від користувача і повністю ігнорує голосові.

Підсумовуючи, можна сказати, що розроблені чат-боти переважають своїх конкурентів в багатьох речах. Вони надають зручний та інтуїтивний користувацький інтерфейс, а також функціонал транскрибування аудіо- чи відеофайлів, який відсутній абсолютно у всіх розглянутих аналогах. Тільки один з чат-ботів, які використовувались для порівняння, надає можливість обрати роль віртуального асистента, з яким буде проводитись спілкування. При чому, цей вибір реалізований не надто зручним методом. Також Telegram чат-бот чудово справляється з розпізнаванням голосових повідомлень користувача та надає коректні відповіді на них, що може зробити тільки «GPT4Telegrambot» з платною підпискою. У розроблених чат-ботів відсутня явна функція щодо написання резюме для Youtube відеоролика, проте це з легкістю може зробити віртуальний асистент з будь-якою роллю, якщо його про це попросити в явному вигляді. Унікальною особливістю створених систем є вибір розміру зображень, які користувач бажає згенерувати. Варто зазначити, що ніякий з розглянутих чат-ботів не вимагає від користувача API-ключа для продовження використання функціоналу, що є їхньою перевагою. Також певні аналоги використовують для створення зображень ще й сервіс Midjourney, тоді як розроблені системи пропонують функцію генерації картинок за тільки допомогою DALL-E.

РОЗДІЛ 6. РОЗГОРТАННЯ РОЗРОБЛЕНИХ ЧАТ-БОТІВ НА СЕРВЕРІ ТА ЇХНЄ ТЕСТУВАННЯ

6.1 Розміщення та запуск створеного Telegram чат-бота на сервері

Оскільки однією з нефункціональних вимог системи є її цілодобова доступність, то було вирішено розгорнути її на веб-сервері, оскільки саме це може гарантувати цілодобову та безперебійну роботу чат-ботів. Для даної цілі був обраний сервіс PythonAnywhere. Це платформа для хостингу і запуску програм, написаних на мові програмування Python, яка забезпечує доступ до веб-сервера, віртуальної машини Python і баз даних, які можуть бути використані для розгортання веб-додатків, сервісів та інших програм. PythonAnywhere має багато переваг:

- простий і зручний інтерфейс, який дозволяє легко налаштовувати та запускати програми;
- запускається в хмарі, що означає, що програми можуть бути запущені і доступні для користувачів з будь-якого місця, де є доступ до мережі інтернет;
- підтримує багато популярних бібліотек Python, включаючи NumPy, SciPy, Matplotlib, Pandas, Django, Flask і багато інших;
- має інтеграцію з GitHub, що дозволяє користувачам легко і швидко розгортати свої проєкти;
- забезпечує безпеку даних, використовуючи шифрування та інші заходи безпеки.

Для масштабованості PythonAnywhere пропонує різні тарифні плани, які можна змінювати в міру розвитку розроблюваного продукту. А найголовніше те, що він ще й пропонує безкоштовний тарифний план, який чудово згодиться для початкових розробників і їхніх невеликих проєктів. Кожен тарифний план

включає в себе певну кількість процесорних секунд щодня. CPU-секунда – це одна секунда використання повної потужності процесора на сервері. Код програми використовує секунди CPU у той момент, поки він фактично зайнятий. Якщо ж код не використовує жодної потужності процесора (можливо, він не запущений або очікує на введення чи повернення веб-запиту), тоді вважається, що він не використовує жодних секунд процесора. Але навіть якщо користувач вичерпав всі CPU-секунди, які надаються йому в межах тарифного плану, то не варто хвилюватись, запущені процеси все одно працюватимуть, просто вони будуть виконуватись з нижчим пріоритетом, ніж процеси користувачів, які ще не досягли свого ліміту. Але вони отримують будь-які вільні ресурси, які є в кластері серверів PythonAnywhere, якщо ці процеси не використовують гігабайти оперативної пам’яті [43, 44].

Для хостингу розробленого Telegram бота для початку потрібно створити безкоштовний обліковий запис користувача, ресурсів якого буде більш ніж достатньо для початкового етапу життєдіяльності бота. Далі на офіційному сайті PythonAnywhere необхідно перейти у вкладку Files та завантажити на сервер директорію з кодом Telegram бота, що знаходиться в Github репозиторії. Після цього необхідно перейти у вкладку Consoles та створити нову консоль, яка відкриє нам доступ до командного рядка сервера і дозволить налаштувати та запустити наш чат-бот. Його налаштування полягає у встановленні зовнішніх залежностей для проєкту, які знаходяться у файлі «requirements.txt». Їх можна інсталювати за допомогою команди *pip install – r requirements.txt*. Також, як вже було згадано раніше, для того, щоб можна було запустити чат-бот, необхідно встановити 2 змінні оточення: токен для Telegram бота та ключ для шифрування і дешифрування інформації, що зберігається в базі даних. Це можна зробити за допомогою наступних вказівок командного рядка:

```
export API_KEYS_DB_ENCRYPTION_KEY="xxxxx"
export TELEGRAM_BOT_TOKEN="xxxxx"
```

, де замість *xxxxx* потрібно підставити відповідні значення. Після завершення налаштувань можна перейти до фінального етапу – запуску чат бота, для якого достатньо виконати одну єдину команду – *python ./bot.py*. В результаті описаних дій, Telegram чат-бот почне працювати, приймати повідомлення від користувачів та виконувати їхні вказівки.

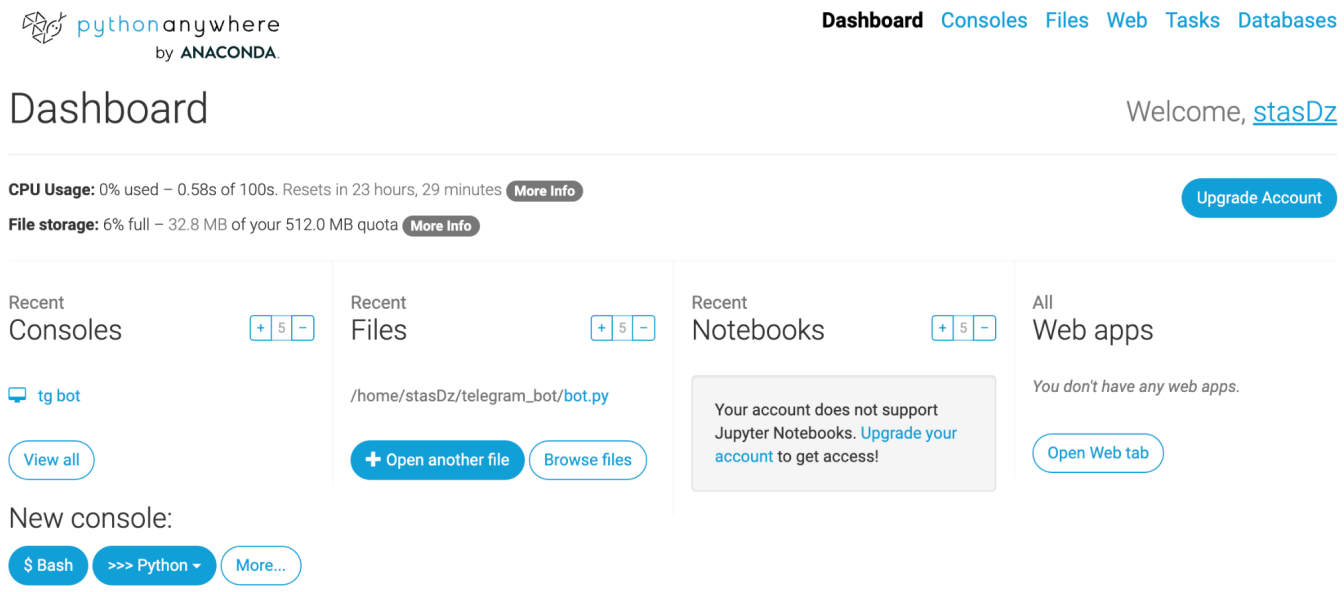

Рисунок 6.1 – Головний інтерфейс сервісу PythonAnywhere

6.2 Розміщення та запуск створеного Viber чат-бота на сервері

Для хостингу Viber чат-бота був також обраний сервіс PythonAnywhere, який був розглянутий у попередньому підрозділі. Як вже було сказано раніше, Viber боти повинні бути створені у вигляді веб-додатків, мати свою URL-адресу та бути доступними ззовні. На щастя, PythonAnywhere дозволяє розміщувати створені веб-додатки на сервері і виділяти під них унікальну адресу. Варто відзначити, що в рамках безкоштовного тарифного плану дозволяється створити тільки один веб-сервер з фіксованою URL-адресою, яка містить в собі інформацію про ім’я користувача та назву сервісу.

Тож, приступимо до розміщення чат бота на сервері. Перш за все нам потрібно перейти до вкладки Files та завантажити код нашого Viber чат-бота таким самим чином, як це було зроблено для Telegram. Після цього аналогічно

встановлюємо всі зовнішні залежності проєкту та 2 змінні оточення: токен для Viber бота та ключ для шифрування і дешифрування інформації, що зберігається в базі даних. Детальна інструкція як це зробити була наведена у попередньому підрозділі. Як тільки всі налаштування були виконанні, необхідно перейти до вкладки Web, натиснути на кнопку «Add new web app» та слідувати всім вказівкам, які приведуть нас до сторінки налаштувань нашого веб сервера (див. рис. 6.2).

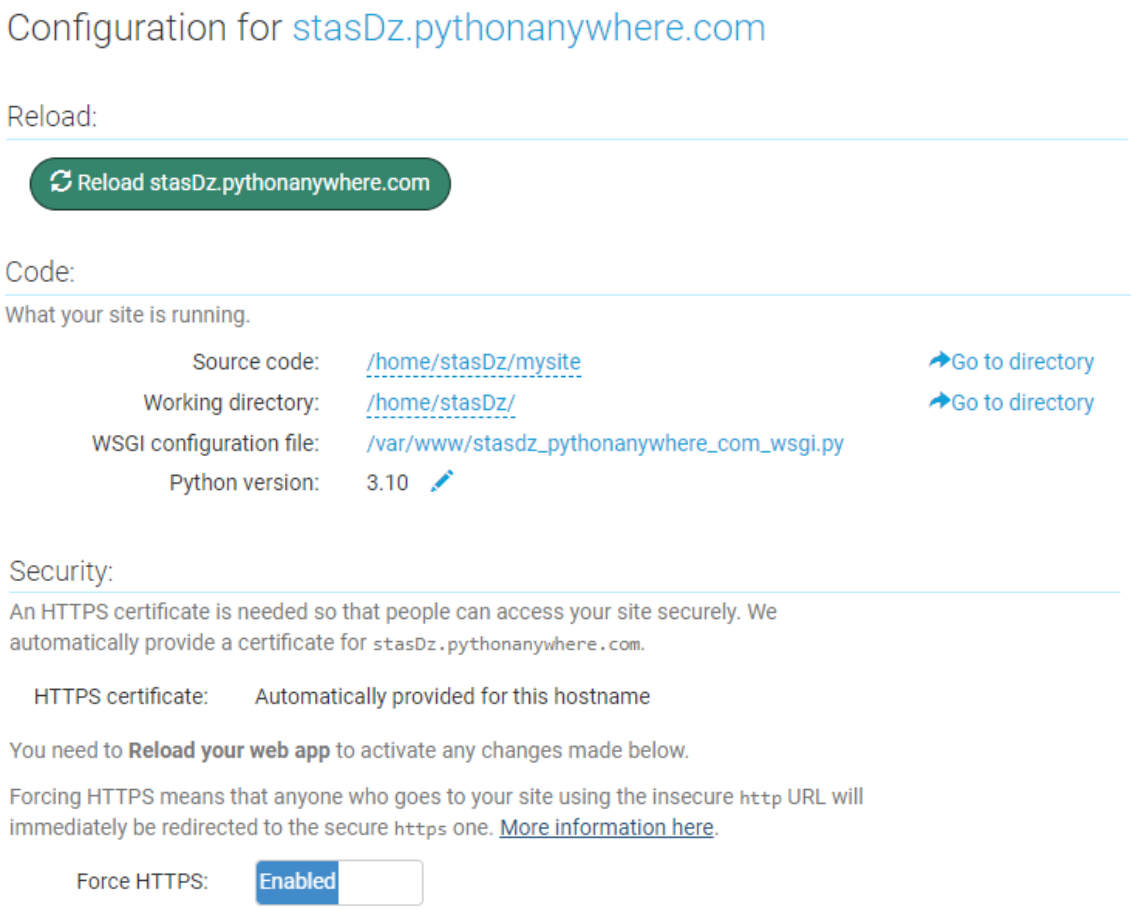

Рисунок 6.2 – Сторінка налаштувань створеного веб-сервера

На даній сторінці необхідно внести деякі зміни. Насамперед, потрібно встановити шлях до коду чат-бота у поле «Source code» та шлях до директорії, де будуть зберігатись логи проєкту, база даних та інші тимчасові файли у поле «Working directory». Наступне, що слід зробити – це додати підтримку HTTPS протоколу, оскільки це є необхідною умовою від Viber. Чат-бот, розміщений на HTTP сервері, працювати не буде. На щастя, PythonAnywhere дозволяє

створювати HTTPS сервери дуже просто – достатньо перевести повзунок «Force HTTPS» в позицію «Enabled». Після цього наш сервер може бути запущений натисканням на кнопку «Reload URL» і стане доступний за певною URL-адресою. Останнім кроком для того, щоб зробити Viber чат-бот активним, стане встановлення Webhook. Саме на нього Viber сервер передаватиме повідомлення, які надходять від користувачів, а той, у свою чергу, перенаправлятиме їх у розроблений веб-додаток. Зробити це можна однією простою вказівкою командного рядка:

curl -# -i -g -H "X-Viber-Auth-Token:%VIBER_BOT_TOKEN%" -d '{"url":"https://bot_url"}' -X POST https://chatapi.viber.com/pa/set_webhook -v

, де *VIBER_BOT_TOKEN* – це токен створеного Viber бота, а *https://bot_url* – це унікальна URL-адреса, яку ми отримали в результаті розміщення нашого чат-бота на сервері PythonAnywhere. Після проведених маніпуляцій, інтелектуальний Viber чат-бот буде здатен приймати та обробляти вхідні повідомлення від користувача.

Варто зазначити, що під час імплементації чат-бота було б незручно постійно завантажувати його оновлений код на сервер та перевіряти розроблений функціонал, це б значно збільшило час розробки. Тому дана система може бути розгорнута локально та висвітлена в мережу інтернет за допомогою утиліти **ngrok**. Ngrok – це інструмент, який допомагає розробникам створювати тунелі для забезпечення доступу до локальних серверів через інтернет. Він потрібен для тестування, розробки, демонстрації веб-додатків, API або інших сервісів, які працюють на локальному комп'ютері розробника [45]. Локальний сервер може стати доступним для мережі інтернет за допомогою виконання наступної команди:

ngrok http --host-header=rewrite localhost:port

, де *port* – це порт, на якому запущений розроблюваний локальний сервер. Після цього ngrok надасть унікальну URL-адресу, яку можна буде використати для встановлення webhook.

6.3 Тестування розроблених систем

Процес розробки програмного забезпечення неможливий без контроля якості продукту, що розробляється. Тестування ПЗ являє собою таку ж невід'ємну частину процесу розробки, як і написання коду і дозволяє оцінити якість продукту, що розробляється [46].

З метою забезпечити якість, знайти та виправити можливі дефекти та перевірити відповідність чат-ботів функціональним та нефункціональним вимогам було проведене ручне тестування. Команди чат-ботів та їх функціонал перевірялися вручну через обидва месенджери з облікових записів декількох користувачів. Варто відзначити, що перевірка відбувалась на різних операційних системах, таких як: Android, iOS, Windows та macOS. Було перевірено можливість асистентів відповідати на повідомлення різного обсягу та різними мовами. Також, окремо для Telegram чат-бота, було перевірено можливість системи відповідати на голосові повідомлення різної тривалості. Поведінка асистентів з різними ролями працює відмінно, специфікація ролі дійсно допомагає отримувати якісніші відповіді на запитання.

Функціонал створення зображень за описом користувача був також ретельно протестований з великим набором вхідних даних та їх різноманітними комбінаціями.

Під час тестування інструменту транскрибування текстів використовувались файли різних розмірів та форматів. З кожним з них чат-боти справляються чудово і надають відповідь в рамках розумного проміжку часу. Також було перевірено поведінку асистентів при отриманні файлів, формат яких не підтримується сервісом Whisper.

Насамкінець було протестовано поведінку розроблених програм при некоректно встановленому API-ключі. В такому випадку система поводить себе без збоїв, повідомляє користувачів про помилку і пропонує встановити цей ключ ще раз.

На підставі проведеного тестування можна зробити висновок, що розроблені інтелектуальні чат-боти для месенджерів Viber та Telegram відповідають встановленим для них функціональним та нефункціональним вимогам, а також забезпечують зручний та ефективний інтерфейс для користувачів. Всі виявлені недоліки та помилки були виправлені, і було вирішено, що чат-боти готові до впровадження та використання в реальних умовах.

ВИСНОВКИ

В результаті виконання даної роботи були отримані наступні результати:

– Розглянуто такі поняття, як месенджери та чат-боти, і проаналізовано їхню роль у повсякденному житті людей. Було з’ясовано, що використання даних технологій є невід’ємним атрибутом у сучасному суспільстві та особливо набуло своєї популярності під час пандемії в 2020 році. Дані програми значно спрощують життя людей;

– Розглянуто список найпопулярніших месенджерів на момент написання кваліфікаційної роботи, проведено огляд месенджерів Viber та Telegram і проаналізовано принципи і методи створення чат-ботів для них;

– Проведено ознайомлення з компанією OpenAI та сферою її діяльності. Проаналізовано сервіси, які надає дана компанія, їхні API, принципи взаємодії з ним та як вони можуть бути використані для створення інтелектуальних чат-ботів для месенджерів Viber та Telegram;

– Сформовано специфікацію до розроблюваних систем, яка включає в себе функціональні та нефункціональні вимоги до проєктів. На основі визначених вимог було проведене об’єктно-орієнтоване проєктування чат-ботів та створена їхня архітектура;

– Спроєктовано зручний та сучасний користувацький інтерфейс для обох чат-ботів з урахуванням особливостей їх технічної реалізації, який відповідає всім теперішнім стандартам якості;

– Проведений аналіз підходів до розробки чат-ботів та обрані мова програмування Python та бібліотеки «python-telegram-bot» і «viberbot» як інструментальні засоби реалізації чат-ботів для Viber та Telegram;

– На основі специфікації для проєктів, їх архітектури та користувацького інтерфейсу, були розроблені чат-боти для месенджерів Viber та Telegram;

– Розміщено розроблені продукти на веб-серверах сервісу PythonAnywhere з метою організації їх цілодобової доступності та проведене ручне тестування чат-ботів.

Дана програмна розробка має практичну значущість, оскільки надає користувачам Viber та Telegram дуже зручний доступ до одних з найпопулярніших сервісів у світі на даний момент.

Створені програмні продукти можуть використовуватись будь-якими користувачами месенджерів Telegram або Viber з метою отримання відповідей на різноманітні запитання чи просто ведення діалогу з інтелектуальним віртуальним помічником на бажані теми. За допомогою розроблених чат-ботів можна створювати зображення за поданим описом, навіть якщо таких досі не існує в мережі інтернет, та робити транскрибування аудіо- чи відеофайлів.

У подальшому продукти, створені в рамках кваліфікаційної роботи, можуть бути значно розширені. Для початку необхідно портувати розроблені чат-боти для месенджеру WhatsApp, який також є дуже популярним у світі. Дана задача може бути з легкістю виконана, зважаючи гнучку архітектуру створених систем. Також потрібно додати підтримку різних мов для користувацького інтерфейсу, оскільки не всі люди можуть знати англійську. Наступним чудовим нововведенням був би розширений список ролей для віртуального асистента, оскільки на даний момент їх існує всього шість. Ну і, звичайно, доцільно буде розширити функціонал чат-ботів, інтегрувавши ще деякі сервіси OpenAI чи інших компаній.

Чат-бот для месенджера Telegram доступний за наступним посиланням: https://t.me/smart_chat_gpt_assistant_bot.

Чат-бот для месенджера Viber може бути знайдений за його іменем: **chatgptassistant**.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ashby E. Messenger App [Електронний ресурс] / Erik Ashby // Helpshift. – 2022. – Режим доступу до ресурсу: https://www.helpshift.com/glossary/messenger-app/
2. Why Messengers are Becoming More Popular Than Social Media [Електронний ресурс] // Medium. – 2019. – Режим доступу до ресурсу: https://medium.com/@amocrmglobalmarketing/why-messengers-are-becoming-more-popular-than-social-media-3bf2ec80e146
3. Shweta B. What Is A Chatbot? Everything You Need To Know [Електронний ресурс] / B. Shweta, K. Main // Forbes. – 2022. – Режим доступу до ресурсу: https://www.forbes.com/advisor/business/software/what-is-a-chatbot/
4. Зоряна – перший в Україні інтелектуальний чат-бот у Facebook Messenger [Електронний ресурс] // IT News. – 2016. – Режим доступу до ресурсу: http://itnews.com.ua/news/81357-zoryana-pervyj-v-ukraine-intellektualnyj-chat-bot-v-facebook-messenger
5. Why are AI Chatbots So Popular Globally Read more: Why are AI Chatbots So Popular Globally [Електронний ресурс] // Umni – Режим доступу до ресурсу: https://umni.bg/en/blog/why-are-ai-chatbots-so-popular-globally/
6. ELIZA: a very basic Rogerian psychotherapist chatbot [Електронний ресурс] // Information Services and Technology – Режим доступу до ресурсу: https://web.njit.edu/~ronkowit/eliza.html
7. Dean B. How Many People Use Telegram in 2023? 55 Telegram Stats [Електронний ресурс] / Brian Dean // Backlinko. – 2023. – Режим доступу до ресурсу: https://backlinko.com/telegram-users
8. MTProto Mobile Protocol [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/mtproto
9. Introducing Inline Bots [Електронний ресурс] // Telegram. – 2016. – Режим доступу до ресурсу: https://telegram.org/blog/inline-bots?setln=it

10. How to set up your Telegram Bot using BotFather [Електронний ресурс] // Dev Genius. – 2022. – Режим доступу до ресурсу: https://blog.devgenius.io/how-to-set-up-your-telegram-bot-using-botfather-fd1896d68c02
11. What Is An API? [Електронний ресурс] // Amazon – Режим доступу до ресурсу: https://aws.amazon.com/what-is/api/
12. HTTP Request Methods [Електронний ресурс] // W3Schools – Режим доступу до ресуру: https://www.w3schools.com/tags/ref_httpmethods.asp
13. A Query on Using Query Strings/Parameters [Електронний ресурс] // Claravine – Режим доступу до ресурсу: https://www.claravine.com/a-query-on-using-query-strings-parameters/
14. Hasan R. What is JSON? And why do you need it? [Електронний ресурс] / Rizwan Hasan // Dev. – 2021. – Режим доступу до ресурсу: https://dev.to/techlearners/what-is-json-and-why-do-you-need-it-21nd
15. Telegram Bot API [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots/api
16. What is Viber? [Електронний ресурс] // WebWise – Режим доступу до ресурсу: https://www.webwise.ie/parents/what-is-viber/
17. What is end-to-end encryption? [Електронний ресурс] // IBM – Режим доступу до ресурсу: https://www.ibm.com/topics/end-to-end-encryption
18. Vuleta B. Viber Statistics [Електронний ресурс] / Branka Vuleta // 99 Firms. – 2022. – Режим доступу до ресурсу: https://99firms.com/blog/viber-statistics/#gref
19. Viber REST API [Електронний ресурс] – Режим доступу до ресурсу: https://developers.viber.com/docs/api/rest-bot-api/
20. Verma P. What to know about OpenAI, the company behind ChatGPT [Електронний ресурс] / Pranshu Verma // The Washington Post. – 2023. – Режим доступу до ресурсу: https://www.washingtonpost.com/technology/2023/02/06/what-is-openai-chatgpt/

21. Тартачний О. Історія OpenAI: як страх перед штучним інтелектом створив ChatGPT [Електронний ресурс] / Олександр Тартачний // SPEKA. – 2023. – Режим доступу до ресурсу: https://speka.media/istoriya-openai-yak-strax-pered-stucnim-intelektom-stvoriv-chatgpt-v5dzz9
22. Truly A. GPT-4: how to use, new features, availability, and more [Електронний ресурс] / Alan Truly // Digitaltrends. – 2023. – Режим доступу до ресурсу: https://www.digitaltrends.com/computing/chatgpt-4-everything-we-know-so-far/
23. OpenAI API Reference [Електронний ресурс] – Режим доступу до ресурсу: https://platform.openai.com/docs/api-reference
24. Wu G. What Is DALL-E and How Does It Create Images From Text? [Електронний ресурс] / Garling Wu // MUO. – 2023. – Режим доступу до ресурсу: https://www.makeuseof.com/what-is-dall-e-ai-image-generator/
25. Briggs J. Fixing YouTube Search with OpenAI's Whisper [Електронний ресурс] / James Briggs // Pinecone. – 2022. – Режим доступу до ресурсу: https://www.pinecone.io/learn/openai-whisper/
26. Lorenc K. What is app specification and why should you write one? [Електронний ресурс] / Katarzyna Lorenc // Ready4S. – 2016. – Режим доступу до ресурсу: https://www.ready4s.com/blog/what-is-app-specification
27. Functional and Nonfunctional Requirements: Specification and Types [Електронний ресурс] // Altexsoft – Режим доступу до ресурсу: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/
28. What Is User Interface (UI)? [Електронний ресурс] // Indeed. – 2022. – Режим доступу до ресурсу: https://www.indeed.com/career-advice/career-development/user-interface

29. White L. 5 Reasons Why A Good User Interface Is Important [Електронний ресурс] / Lucas White // Codersera. – 2022. – Режим доступу до ресурсу: https://codersera.com/blog/why-a-good-user-interface-is-important/
30. Why do we need to choose the right technology stack for Your Project? [Електронний ресурс] // Great Innovus. – 2022. – Режим доступу до ресурсу: https://www.greatinnovus.com/blogs/why-do-we-need-to-choose-the-right-technology-stack-for-your-project/
31. Araujo G. JavaScript vs Python for Chatbot Development [Електронний ресурс] / Gabe Araujo // Dev Genius. – 2023. – Режим доступу до ресурсу: https://blog.devgenius.io/javascript-vs-python-for-chatbot-development-which-one-is-right-for-you-dfe84f77c85
32. As R. What is SQLite? Everything You Need to Know [Електронний ресурс] / Ravikiran As // Simplilearn. – 2023. – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/sql-tutorial/what-is-sqlite
33. Chakraborty S. Fernet (symmetric encryption) using Cryptography module in Python [Електронний ресурс] / Shreyasi Chakraborty // Geeksforgeeks. – 2020. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/fernet-symmetric-encryption-using-cryptography-module-in-python/
34. Ffmpeg [Електронний ресурс] // api.video – Режим доступу до ресурсу: https://api.video/what-is/ffmpeg/
35. python-telegram-bot [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/python-telegram-bot/python-telegram-bot
36. What is Flask Python [Електронний ресурс] // Pythonbasics – Режим доступу до ресурсу: https://pythonbasics.org/what-is-flask-python/
37. Ferguson K. Application architecture [Електронний ресурс] / Kevin Ferguson // Techtarget – Режим доступу до ресурсу: https://www.techtarget.com/searchapparchitecture/definition/application-architecture

38. UML Diagram Types Guide: Learn About All Types of UML Diagrams with Examples [Електронний ресурс] // Creately. – 2022. – Режим доступу до ресурсу: https://creately.com/blog/diagrams/uml-diagram-types-examples/
39. Logging vs Monitoring: Best Practices for Integration [Електронний ресурс] // AppDynamics – Режим доступу до ресурсу: https://www.appdynamics.com/product/how-it-works/application-analytics/log-analytics/monitoring-vs-logging-best-practices#~cloudops-vs-devops
40. Why Should You Care About Logging? [Електронний ресурс] // Towards Data Science. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/why-should-you-care-about-logging-442a195b80a1
41. Git Submodule [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/git/tutorials/git-submodule
42. OpenAI Models [Електронний ресурс] – Режим доступу до ресурсу: https://platform.openai.com/docs/models/overview
43. PythonAnywhere [Електронний ресурс] – Режим доступу до ресурсу: https://www.pythonanywhere.com/
44. What are “CPU seconds”? [Електронний ресурс] – Режим доступу до ресурсу: https://help.pythonanywhere.com/pages/WhatAreCPUSeconds
45. What is ngrok? [Електронний ресурс] // PubNub – Режим доступу до ресурсу: https://www.pubnub.com/guides/what-is-ngrok/
46. Sharma L. Why is Testing Necessary? [Електронний ресурс] / Lakshay Sharma // ToolsQA. – 2022. – Режим доступу до ресурсу: https://www.toolsqa.com/software-testing/istqb/why-is-testing-necessary/