ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних технологій

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка чат-боту для Telegram мовою Python»

Студента 4 курсу, 8 групи,

спеціальності
122 «Комп'ютерні науки»

Буі Тхі Лі

_____
підпис студента

Науковий керівник
кандидат педагогічних наук, доцент _____ Базурін Віталій Миколайович
підпис керівника

Гарант освітньої програми
кандидат технічних наук, професор _____ Демідов Павло Георгійович
підпис керівника

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу (проект) студентці

Буі Тхі Лі
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка чат-боту для Telegram мовою Python»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка телеграм-боту на мові Python
Об'єкт дослідження: особливості обміну інформацією у телеграм-ботах.
Предмет дослідження: алгоритм та структура телеграм-бота.
4. Перелік графічного матеріалу_____
_____
_____
_____
_____
5. Консультанти по роботі із зазначенням розділів, за якими

здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Базурін В.М.	15.12.2022 р.	15.12.2022 р.
2	Базурін В.М.	15.12.2022 р.	15.12.2022 р.
3	Базурін В.М.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Аналітичне дослідження специфіки застосування телеграм-ботів для здійснення послуг консультування

1.1. Аналіз стану проблеми

1.2. Особливості обміну даними в мережі Інтернет

1.3. Огляд існуючих програмних рішень

РОЗДІЛ 2. Розробка моделі телеграм-бота

2.1. Загальна концепція

2.2. Моделі обміну даними

2.3. Алгоритми обміну даними

РОЗДІЛ 3. Розробка телеграм-боту

3.1. Розробка інформаційно-логічної моделі

3.2. Програмна реалізація чат-боту та його тестування

3.4. Технологія використання чат-боту

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*

2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1. Аналітичне дослідження специфіки застосування телеграм-ботів для здійснення послуг консультування*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2. Розробка моделі телеграм-бота*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3. Розробка телеграм-боту*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 -01.06.2023*	*31.05.2023 -01.06.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

Керівник випускної кваліфікаційної роботи (проекту)
Базурін В.М.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Буі Тхі Лі
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи (проекту)

Керівник випускної кваліфікаційної роботи (проекту)
_____ 30.05.2023 р.
(підпис, дата)

10. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.
(підпис, прізвище, ініціали)

«_____»_____2023 р.

Анотація

У випускній кваліфікаційній роботі представлені теоретичні дослідження методів обміну даних в мережі Інтернет, історичні та теоретичні відомості про розробки та використання чат-ботів, алгоритми та моделі обміну даних у чат-ботах. Представлена схема алгоритму та програмна реалізація чат-боту на мові програмування Python для месенджеру Телеграм для автоматизації адміністрування в стоматологічних клініках. Використання Google Sheets і Google Cloud у розробці електронної таблиці для адміністрування.

Ключові слова: чат-боти Телеграм, обмін даними в Інтернеті, алгоритми та моделі обміну даних в Телеграм, автоматизація процесів адміністрування, Python.

Anotation

The graduation thesis presents theoretical studies of data exchange methods on the Internet, historical and theoretical information about the development and use of chatbots, algorithms and models of data exchange in chatbots. The block diagram and software implementation of a chatbot in the Python programming language in Telegram messenger for the automation of administration in dental clinics are presented. Using of Google Sheets and Google Cloud in development of a spreadsheet for administration.

Keywords: Telegram chatbots, data exchange on the Internet, algorithms and models of data exchange in Telegram, automation of administration processes, Python.

ЗМІСТ

ВСТУП

Актуальність обраної теми. З набуттям популярності використанням смартфонів та інтенсивної цифровізації, чат-боти стали популярним способом взаємодії людини зі сферами послуг без участі другої особи. Чат-бот – програма написана мовою програмуванням з попередньо прописаним сценарієм відповідей на конкретне запитання для комунікації з користувачем. Через свою гнучкість і простоту створення, чат-бот часто використовуються у сферах адміністрування та ведення бізнесу. Наприклад, вони використовуються компаніями для збору інформації про клієнта: особистої інформації, отримання відгуків від клієнта про надані послуги або продукт. Найпоширенішим прикладом у сфері надання послуг є використання чат-боту для запису клієнта у електронну чергу.

З розвитком мов програмування та розширенням документації для їх створення на різних платформах, все більше розробників реалізовують функціонал чат-ботів у своїх додатках.

Соціальні мережі та месенджери стали потужним інструментом для комунікації та збору інформації компаніями від своїх потенційних клієнтів. Для автоматизованої роботи все частіше використовується набір функціоналу чат-боту для спілкування та надання послуг консультанта клієнтам. Наприклад, у стоматологічній клініці чат-бот може записати пацієнта на прийом, представити перелік доступних послуг у клініці та прайс-лист до нього. Також, у банківській сфері часто використовують чат-ботів для роботизації обробки заявок клієнтів, які не потребують участі безпосереднього робітника банку.

Тому для випускної кваліфікаційної роботи була обрана тема «Розробка чат-боту для Telegram мовою Python». Чат-бот імітуватиме функції віртуального консультанта, який інформуватиме та записуватиме клієнта на прийом. У чат-боті, що розробляється користувач зможе використовувати інтерфейс з кнопками вибору команди для бота. Також бот

може приймати та надсилати повідомлення від клієнта для обробки його адміністрацією. Всі записи у електронну чергу будуть записуватися ботом автоматично у Google таблиці, і адміністрація, маючи доступ до файла, зможе як переглядати, так і редагувати його.

Метою роботи є теоретичне обґрунтування і розроблення чат-боту для Telegram.

Мета і завдання дослідження.

Основним завданнями дослідження дипломної роботи є:

1) проаналізувати стан проблеми у науковій літературі і в практиці роботи компаній і приватних підприємств;
2) розкрити особливості обміну даними у мережі Інтернет;
3) проаналізувати існуючі на час написання дипломної роботи видів чат-боту;
4) визначити основні концептуальні ідеї чат-ботів;
5) аналіз загальної моделі обміну даних чат-ботів;
6) дослідити методи та алгоритми обміну даними для чат-ботів.
7) розробити схема алгоритмів ат-бота
8) розробити чат-бота консультанта для автоматизації процесу запису на прийом клієнтів для стоматологічної клініки

Об'єкт дослідження: процес обміну даними в Telegram.

Предмет дослідження: алгоритм та структура телеграм-бота.

Методи дослідження:

- системний аналіз для визначення вимог щодо функціоналу чат-боту
- тестування програмного забезпечення
- моделювання та діаграмне представлення програмних систем

Практичне значення одержаних результатів. Отримана інформація з дослідження, можна використати для розробки чат-ботів для адміністрування та автоматизації комунікації з клієнтом для бізнесів за моделлю business-to-consumer (B2C).

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 24 найменувань, додатків і містить 47 сторінки основного тексту, 25 рисунків і 1 таблиці.

РОЗДІЛ 1. АНАЛІТИЧНЕ ДОСЛІДЖЕННЯ СПЕЦИФІКИ ЗАСТОСУВАННЯ ТЕЛЕГРАМ-БОТІВ ДЛЯ ЗДІЙСНЕННЯ ПОСЛУГ КОНСУЛЬТУВАННЯ

1.1. Аналіз стану проблеми

Середина 20-ого століття стала у сфері телекомунікацій та комп'ютерної дослідницької діяльності. Зі зростанням технологічних можливостей комп'ютерів постало вагоме питання уможливлення автоматизації процесів та імітація людського мислення на базі комп'ютеру, створення, відомого на сьогоднішній день, штучного інтелекту. Ідея імітація людського розуму дала початок різним галузям використання штучного інтелекту. Серед них найбільшу популярність отримала розробка чат-боту або інтелектуальної програмної системи спілкування.

Чат-бот – це інтелектуальна комп'ютерна система спілкування розроблена, щоб імітувати діалог з людиною [1,2]. Чат-боти можуть обробляти вхідні повідомлення та повертати вихідне повідомлення [3]. Іншими словами, чат-бот можна ідентифікувати, як "онлайн-систему діалогу між людиною та комп'ютером у природній формі" [1].

Перша успішна реалізація чат-бота, як самостійної інтелектуальної системи спілкування була досягнута в 1966 році – ELIZA, яка була розроблена Джозефом Вейценбаумом. Ця програма спілкувалася з користувачем через програму зіставлення ключових слів. Через таку обмеженість ELIZA могла спілкуватися лише вбудованими в програму шаблонами, які мали низький рівень розпізнавання контексту вхідного повідомлення.[4] Тим не менш, це дало поштовх для продовження досліджень у сфері розробки інтелектуальних систем спілкування.

Чат-боти можуть поділятися на різні категорії в залежності від цілі або платформи на якій вони використовуються. Загалом, чат-боти прийнято

поділяти на дві категорії: ті що засновані на правилах та з використанням штучного інтелекту [5, 6].

Суть цієї моделі чат-ботів заснованих на правилах полягає у попередньо визначених шляхах розмови, де користувачі отримують заздалегідь сформовані розробником варіанти запитань і відповідей. Чат-боти засновані на правилах не можуть самонавчатися та аналізувати вхідну інформацію і не спроможні виходити за рамки сценарію, за яким вони були запрограмовані.

Чат-боти, які використовують штучний інтелект (англомовне визначення - AI bots) здатні вести інтелектуальні розмови з користувачем. В основі такої моделі лежить їхні можливості штучного інтелекту: машинне навчання (ML) та обробка природної мови (NLP) [5]. Машинне навчання дозволяє штучному інтелекту розпізнавати шаблони сформовані завдяки отриманих попередньо масиву вхідних даних. Іншими словами, машинне навчання дає можливість самонавчатися, що розширює бібліотеку як вхідних даних, так і вихідної. Обробка природної мови допомагає чат-боту зі штучним інтелектом розпізнавати значення слів отриманого повідомлення та відокремлює ключові слова у реченнях.

Хоча моделі чат-ботів заснованих на правилах, зазвичай, легше розробити та реалізувати, але вони мають обмежені можливості і сильно залежать від шаблонів відповідей. Іноді вихідне повідомлення сформоване на основі шаблонів не є точними або не відповідають контексту вхідних даних, що робить таку модель не гнучкою для використання у більш складних цілях як, наприклад, для розпізнавання характеру та тону вхідного повідомлення або контекст речення, яке не було запрограмоване у методи імплементації чат-бота.

Також, основною із труднощів простих за технологією чат-ботів залишається їх неспроможність завершити backend процеси. Натомість вони покладаються на те, що кінцевий користувач самостійно вживе

подальші заходи. Наприклад, чат-боти клієнт сервісів можуть використовувати декілька систем одночасно для обробки вхідних запитів [7]. Тому більш складні для чат-бота запити мають довший час обробки і клієнт може не дочекатися відповіді, звернувшись до реального консультанта або взагалі відмовитися від сервісу.

Утім свою популярність чат-ботів заснованих на правилах отримали завдяки їх реалізації у додатках месенджерах за методологією Send/Receive API, яка дозволяє відправляти, а також отримувати текстові повідомлення або медіафайли. Після успішного запуску можливості створювати чат-ботів у Facebook Messenger у 2016 році, багато компаній, працюючі за моделлю B2C (business-to-consumer), почали розробляти методи автоматизації консалтингових послуг через чат-боти. До вересня 2017 року кількість чат-ботів у додатку зросла з 30 тис. до 100 тис [8]. Також, створення чат-ботів були реалізовані у інших популярних додатках месенджерах як Viber, WhatsApp, Telegram і т.п.

Чат-боти широко використовуються у інституціях сфер послуг та адміністрування:

- глобальні логістичні компанії (для автоматичної обробки вхідних дзвінків);
- навчальні заклади (як частина сайту для надання інформації відвідувачам);
- адміністративні установи (для консультації та запису в електронну чергу);
- комерційні установи за моделлю B2C (консультація клієнтів, продаж продуктів та послуг).

Чат-бот, який розробляється призначений для стоматологічної клініки для адміністрування та консультування клієнтів щодо основної інформації для відвідувачів клініки. У загальному, чат-бот матиме перелічені функції:

- запису пацієнта на прийом у електронну чергу (також можливість скасувати прийом);
- надаватиме інформацію про перелік доступних процедур та прайс-лист;
- можливість залишати клієнтами відгуки;
- надаватиме відповіді на перелік найпоширеніших питань.

Для чат-боту, який розробляється будуть притаманні ознаки чат-боту заснованих на правилах. Чат-боти консультанти для месенджерів не потребують складного штучного інтелекту для виконання простих команд за встановленим сценарієм.

Загалом, вибір типу чат-боту залежатиме від роду діяльності компанії або від поставленої задачі. Як висновок, чат-боти на базі штучного інтелекту і засновані на правилах мають однакову значимість у методах автоматизації процесу. У різних ситуаціях можуть бути взаємозамінними або, навпаки, недоречними і ресурсо затратними у реалізації.

1.2. Особливості обміну даними в мережі Інтернет

На сьогоднішній день Інтернет став основним інструментом для обміну інформації. Завдяки дослідженням та розширенням можливостей використання мережі Інтернет були розроблені основні методи та технології обміну даними в Інтернеті:

- APPANET з packet-switching – технології закладені у фундамент створення електронного листування;
- Всесвітнє Павутиння (World Wide Web або скор. WWW) – об’єднання мережі Інтернет для колективного обміну інформації;
- Служби миттєвих повідомлень (Instant messaging, скор. IM) – служби для миттєвого обміну текстовими повідомленнями.

Розвиток мережі Інтернет бере початок з 1969 році. На етапі старту основною ціллю експериментальної розробки полягала у створенні мережі, яка могла функціонувати навіть, якщо одна з частин його була пошкоджена. Така розроблена модель потребувала мати у своїй схемі зв'язку основний комп'ютер для керування мережею, що означало мережа перестала працювати як тільки виникав збій у роботі керуючого пристрою.

Після завершення низки експериментів зі створенням локальних мережей у 1970-х роках Стенфордський університет опублікував свої дослідження, в якому були отримані результати розробки протоколів Інтернету: TCP (протокол керування передачею) і IP (Інтернет-протокол). TCP являє собою протокол для керування передачі даних, які згодом розбиваються на пакети і з'єднуються з одержувачем. IP – протокол, який визначає спосіб адресації. Спочатку використання протоколів була заборонена для комерції, але після 1991 ці обмеження були зняті, що позитивно вплинуло на популяризацію Інтернету для широкого використання. [9]

Основним досягненням, завдяки якому можна було отримувати і надсилати дані глобально, є створення Всесвітнього Павутиння (WWW – World Wide Web), яке почалося у 1991. Всесвітнє Павутиння представляє собою згрупований інформаційний простір, який зв'язаний між собою гіперпосиланнями.

Паралельно з цим розвивалася і мова гіпертекстуальної розмітки, що дала змогу створювати текстові файли, які можуть вміщувати у собі не тільки текст, а й медіафайли. Вся інформація таких текстових файлів зберігається на сервері, який при запиті користувача-клієнта на скачування відображає точну копію сторінки. Така схема отримала назву «клієнт-серверне з'єднання».

До впровадження Всесвітнього Павутиння в 1969 році була створена перша свого роду система електронного листування між комп'ютерами

мережі APPANET. На момент тестування таке листування могло проводитися лише всередині локальної мережі, але саме APPANET заклав фундамент для створення packet-switching технологій, який давав можливість відправленому повідомленнню самому обирати шлях доставки електронного листа до одержувача [10]. У свою чергу такий метод ніяк не впливав на кінцевий адрес доставки листа.

Серед популярних сервісів електронного листування можна виділити:

- Gmail від компанії Google
- Outlook компанії Microsoft
- Yahoo
- Mail.com

Всі сучасні електронні скриньки працюють за одним принципом. Кожний користувач отримує унікальний адрес електронної скриньки у обраному ним сервісі, на який адресат може надіслати повідомлення. Листування можливе навіть, якщо адресат і одержувач мають поштові скриньки різних сервісів.

З розвитком Інтернету як засобу комунікації ідея електронного листування була імплементована у розробку моделі соціальної мережі. Соціальна мережа представляє собою соціальну структуру, яка утворена групою людей або організацією з ціллю спілкування або обміну інформації у вигляді тексту та медіафайлів.

Перша соціальна мережа, яка отримала велику аудиторію користувачів стала Myspace. Myscpace на початку свого піку популярності в 2008 представляла собою вебсайт, який підбирав у стрічку новин інформацію, спираючись на обрані цікаві теми користувачем. Також, Myspace формував розділ з іншими учасниками соціальної мережі, які будуть цікаві для знайомств користувачу, в залежності від інтересів та персональних даних. Користувачі могли спілкуватися у приватних повідомленнях, залишати

коментарі на сторінках один одного і додавати один одного у «Друзі» (список друзів на сторінці користувача).

Через успіх MySpace в Інтернеті з'явилась велика кількість аналогічних соціальних мереж. Серед них Facebook, Twitter, tumblr, Instagram і т.п. Всі перелічені соціальні мережі мають відмінності у інтерфейсі, але однакові основні механізми взаємодії, такі як комунікація з іншими користувачами, редагування особистого профілю та створення блок-постів з інформацією.

З удосконаленням технологій та з винаходом швидкісного мобільного Інтернету мережі 3G та 4G, постала задача можливість замінити SMS-листування. Соціальні мережі добре виконують роль інструмента для спілкування, але така комунікація мала занадто масовий характер. Тому для відокремлення приватного спілкування від масового спілкування та елементів соціалізації, такі як формат сторінки-анкети, створення інформаційних блоків. Додатки месенджери стали зручним та простим рішенням для приватного спілкування. Термін «месенджер» почав масово використовуватися для всіх додатків для обміну миттєвими повідомленнями після запуску Facebook Messenger. Месенджер представляє собою комунікаційний додаток для обміну миттєвими текстовими повідомленнями або медіафайлами між різними пристроями. Спілкування у додатках месенджерах подібний до листування через електронне листування. Але основна відмінність полягає у тому, що спілкування в месенджері має характер чату, який відбувається у реальному часі. При електронному спілкування відправлене користувачем повідомлення утримується на сервері сервісу електронної пошти і доступне лише після його відкриття в електронній скриньці. Тому обидва методи представляють різні способи комунікації. Задача електронного листування полягає у імітації реального поштового листування. У свою чергу,

месенджер представляє собою інтерпретацію діалогу або монологу в реальному часі через Інтернет.

З популяризацією месенджери стали не лише зручним інструментом для комунікації через Інтернет, як і соціальні мережі, платформою для колективного спілкування та комерції. Для прикладу, месенджер Telegram після успішного запуску свого додатка для користувачів мобільних пристроїв та персональних комп'ютерів додав функцію створювати «канали», в яких власник «каналу» може розміщувати блоки інформації (пости). Маючи посилання на канал, користувач може долучитися і переглядати розміщену власником інформацію каналі. Така функція стала дуже зручним способом поширення інформації у маси. Велика кількість компаній створюють канали власного бренду для комунікації з клієнтами та безпосереднього просування продукту чи послуг, які надаються.

У Telegram для адміністрування та обробки повідомлень, які можуть надходити від учасників каналу, часто використовуються Telegram-боти. Їх задача та функціонал різняться в залежності від ролі, яку виконуватиме чат-бот для автоматизації того чи іншого процесу. Наприклад, чат-боти можуть виконувати роль «колектора» повідомлень. Власник каналу може надати посилання на чат-бот, в якому учасники можуть залишити повідомлення для комунікації з адміністрацією каналу. Такі чат-боти зазвичай використовуються для адміністрування Telegram каналів з великою кількістю учасників.

Як висновк, чат-боти у мсеенджерах можуть виконувати будь-яку роль, яка має машинальний характер. Основна перевага інтеграції чат-ботів у месенджери являється автоматизація процесів обміну інформації між великою кількістю користувачів. Для комерційних цілей, чат-боти можуть виконувати роль консультанта, надаючи інформацію або збору даних клієнтів. Для контролю великих за обсягом учасників об'єднань, груп або

каналів у месенджерах, чат-боти можуть виступати роль посередника між адміністрацією та учасниками.

1.3. Огляд існуючих програмних рішень

У наш час все більше бізнесів використовують чат-ботів у месенджерах для взаємодії зі своїми клієнтами. Найбільш популярними месенджерами для комерційних чат-ботів стають ті, що мають найбільшу за кількістю та за охопленнями вікових категорій аудиторію. Серед таких можна виділити: Telegram, Viber, Facebook Messengers, WhatsApp.

Існуючі на сьогоднішній день програмні рішення для чат-боту помічника/консультанта не різняться між собою, але в залежності від особливостей послуг або продукту, які пропонує компанія, може відрізнятися наповнення та функціонал чат-боту.

Чат-бот від компанії «Київстар» [11], яка надає послуги зв'язку розробила віртуального асистента у вигляді чат-бота Зоряна для Messenger Facebook. Даний чат-бот, як і всі аналогічні чат-боти працюють за попереднім прописаним сценарієм. Після запуску чат-боту через відправлення будь-якого повідомлення, користувач може обрати мову, якою йому зручніше спілкуватися. Зоряна надає інформацію клієнта про стан рахунку абонента, про тарифний план та бонуси на рахунку. Перевага цього чат-боту є те, що користувач може вести діалог самостійно. Чат-бот, розбираючи отримане повідомлення, виділяє ключові слова з повідомлення і запускає відповідну функцію із меню команд. На етапі старту чат-бот мав недолік щодо конфіденційності рахунків абонентів. Кожен, хто знав номер телефон іншого абонента міг дізнатися про суму на рахунку номера телефону і його тарифний план. Для багатьох користувачів це не була суттєвою проблемою, але така інформація могла легко використовуватися шахраями, щоб дізнатися про стан рахунку абонента. Утім, недолік був усунений і тепер для того, щоб дізнатися про стан рахунку абонентського

номера треба отримати SMS з кодом підтвердження, який вписується в чат з віртуальним асистентом Зоряна.

Також серед чат-ботів помічників велику низку займають суто функціональні чат-боти, які використовуються у більшості для полегшення збору даних та для отримання користувачем інформації. Чат-бот «Київводоканал» [12] у застосунку месенджері Viber, який збирає від користувачів дані з лічильників води, передає їх на обробку та формування комунальних рахунків за водопостачання. Інший за призначенням чат-бот компанії поштових перевезень «Нова пошта» для додатку месенджера Telegram. Його функції різняться від попереднього чат-бота через різну сферу діяльності компанії. Основними функціями чат-бота є можливість відслідковувати поштові відправлення за номером телефону та отримати адрес найближчих відділень. Аналогічний чат-бот був розроблений і компанією поштових перевезень Justin [13], який відрізняється додатковою функцією приблизного розрахунку комісії за поштове перевезення. Перевага суто функціональних чат-ботів в їх лаконічності. Для простих машинальних обов'язків як збір даних або для автоматичної проекції інформації, чат-боти не потребують природнього спілкування та деталізації сценарію поведінки.

Одним з прикладів аналогічний до чат-боту, який розробляється, можна виділити чат-бот для месенджеру Telegram косметологічної клініки «CPG» [14]. Його функціонал включає в себе: привітання клієнта, проекція контактних даних та адрес клініки, перелік послуг, інформація про кожного спеціаліста клініки. Даний чат-бот має особливості Єдина різниця від функціоналу чат-боту, який розробляється є відсутність функції запису на прийом. Таке рішення може бути пов'язаним з особливостями адміністрування клініки і клієнт має записуватися персонально через номер телефону. У деяких випадках відсутність можливості онлайн запису на прийом може рахуватися за недолік зручності користування сервісом.

Зручність автоматичного запису полягає у тому, що така система може функціонувати цілодобово, потребуючи лише технічного обслуговування, наприклад, очищення старої бази даних клієнтів для звільнення місця під нову. Також, автоматична система запису на прийом зменшує навантаження на адміністрацію, що допомагає мінімізувати випадки, коли клієнт незадоволений наданим йому консультативні послуги.

Проаналізувавши існуючі програмні рішення, виділено основні позитивні характеристики, на які спиратиметься розробка чат-бота консультанта:

- Чат-бот збираючи персональну інформацію про клієнта має зберігати її у базі даних доступ до якої має лише адміністратор та технічний спеціаліст. Іншими словами, закриту базу даних.
- Чат-бот має функціонувати цілодобово.
- Функція запису на прийом має бути обов'язково реалізована у чат-боті для полегшення задачі бронювання клієнтом.
- Інтеракція користувача з чат-ботом має бути реалізована у спрощеній формі. Наприклад, використання чат-ботів форми спілкування через запропоновані варіанти відповідей.

Усі представлені вище чат-боти відрізняються за роллю, яку виконують і набором доступних для користувача функціоналу. Основна характеристика, яка прослідковується в усіх чат-ботах месенджерів – це наявність процесу, який був автоматизований для полегшення роботи людини. За проаналізованою інформацією можна зробити висновок, що основні функції чат-ботів є реалізація віртуального спілкування, яка імітує діалог з людиною та автоматизація процесів відповідно до мети розробки.

РОЗДІЛ 2 РОЗРОБКА МОДЕЛІ ТЕЛЕГРАМ-БОТУ

2.1. Загальна концепція

Концепція чат-ботів для месенджерів полягає в створенні програм, які можуть спілкуватися з користувачами через інтерфейс месенджера. Ці програми можуть мати різні функції, такі як відповіді на запитання, обробка замовлень, розклад подій, нагадування про події та інші.

Основна ідея полягає в тому, що користувач може звернутися до чат-бота з будь-якого месенджера, використовуючи текстові повідомлення або голосові команди, і отримати відповіді на свої запитання або діяти відповідно до певних інструкцій.

Для реалізації концепції чат-ботів для месенджерів потрібні високоякісні моделі машинного навчання, які можуть розуміти наміри користувача та інтерпретувати їх запити. Також потрібна налагоджена інтеграція з месенджерами, що дозволяє ботам спілкуватися з користувачами в їхньому улюбленому месенджері без необхідності встановлювати додаткові програми або виходити зі своєї звичної програми обміну повідомленнями.

Концепція чат-ботів для месенджерів використовується в різних галузях, в якиї основну роль відіграє , включаючи бізнес, медіа, освіту та інші сфери. Вона забезпечує зручний та ефективний спосіб спілкування з клієнтами та користувачами та дозволяє автоматизувати багато процесів, що забезпечує збільшення продуктивності та зниження витрат.

Концепція чат-ботів для Telegram полягає в створенні програмного забезпечення, яке може спілкуватися з користувачами у Telegram в автоматизований спосіб. Це означає, що чат-бот може відповідати на запитання користувачів, надавати інформацію та виконувати різні завдання за запитом користувачів. [15]

Чат-боти для Telegram можуть мати різні функції і використовуватися для різних цілей. Наприклад, вони можуть виконувати наступні завдання:

- Комунікація з користувачами: чат-боти можуть відповідати на запитання користувачів, надавати інформацію про продукти та послуги компанії, розповідати про новини та події, а також забезпечувати підтримку користувачів.
- Автоматизація бізнес-процесів: чат-боти можуть виконувати різні операції, такі як розрахунок вартості замовлення, оформлення замовлення, обробка платежів тощо.
- Розваги та розважальні послуги: чат-боти можуть надавати розважальні послуги, такі як ігри, вікторини, головоломки.

Для прикладу, чат-бот подібний до чат-боту, який розробляється має структуру звичайного бота консультанта. Основні функції такого виду чат-боту є привітання користувача, надання переліку методів функцій для взаємодії, надання відповіді на запитання користувача і т.п. Зазвичай функціонал таких чат-ботів обмежується набором функцій та методів, які потрібні у розробці тієї чи іншої галузі. Наприклад, чат-боти, які обробляють замовлення клієнтів можуть мати у наборі функціоналу функції: перевірки стану готовності замовлення, редагування замовлення або надання консультації з приводу замовлення.

Концепція чат-боту для Telegram полягає в створенні програмного забезпечення, яке може взаємодіяти з користувачами у Telegram у автоматизованому режимі. Чат-бот може відповідати на запитання користувачів, надавати інформацію про продукти або послуги, виконувати операції, робити розсилки та багато іншого.

Чат-бот, який розробляється володітиме функціоналом бот-консультанта. Бот поєднуватиме в собі основні концептуальні ідеї комунікативного чат-боту та боту для автоматизації бізнес-процесів.

- Привітання користувача: після старту чат-бот відправлятиме користувача повідомлення з привітанням і головне меню, реалізоване у вигляді інтерактивного поля кнопок. Головне меню включатиме “Інформація про клініку та перелік спеціалістів”, “Запис на прийом”, “Контакти”, “Залишити відгук”.
- Запис на прийом: головна функція у списку меню. Її задача є автоматизація запису на прийом без участі живого консультанта. Функція працюватиме за заданим сценарієм. Користувач спочатку обирає день серед доступних у електронній записній книжці, яка реалізована у Google Таблиці. Далі обирає вільний час у обраний серед запропонованих день.
- Отримання інформації: включатиме у себе функції “Інформація про клініку та перелік спеціалістів”, “Запис на прийом”, “Контакти”. Цей блок функціоналу чат-боту буде представлений як набір інтерактивних кнопок, який при виклику відправлятиме повідомлення, яке заготовлене під кожну функцію блоку.

З ростом популярності автоматизації бізнес-процесів, чат-бот розширюють можливості та сфери діяльності, в яких вони можуть використатися. За зібраною у пункті інформацією можна зробити висновок, що всі представлені концепції чат-ботів засновані на ідеях на автоматизації процесів та імітація природньої мови. У залежності для якої сфери людської діяльності був розроблений чат-бот, він матиме ознаки притаманні лише для цієї галузі.

2.2. Моделі обміну даними

Модель обміну даних – це стандарт або сукупність правил, які визначають, як дані передаються та обробляються між різними системами. Це може включати формат передачі даних, структуру та типи даних, протоколи передачі даних, методи аутентифікації та авторизації, які

використовуються для забезпечення безпеки та конфіденційності даних під час передачі.

Модель обміну даних може бути розроблена для взаємодії між різними комп'ютерними системами, програмами, апаратними засобами, пристроями IoT, веб-сайтами та іншими сутностями, які можуть передавати дані один одному. Це може бути частиною архітектури програмного забезпечення, яка визначає, як різні компоненти системи будуть взаємодіяти та обмінюватися даними між собою.

Моделі обміну даних використовуються для забезпечення стандартизації та ефективності передачі даних між різними системами. Вони можуть знижувати витрати на розробку та технічну підтримку програмного забезпечення, забезпечуючи однаковий формат даних для взаємодії між різними системами.

Існує кілька моделей обміну даними, але я надам загальний огляд найпопулярніших:

REST (Representational State Transfer) – це стандарт, який використовує HTTP протокол для передачі даних між сервером та клієнтом. REST використовує HTTP методи, такі як GET, POST, PUT, DELETE, щоб дозволити клієнтам отримувати, створювати, оновлювати та видаляти ресурси на сервері. REST є одним з найпопулярніших і широко використовується для створення API.

SOAP (Simple Object Access Protocol) - це протокол, який використовує XML для обміну даними між різними компонентами програмного забезпечення. SOAP використовує певні стандарти для передачі повідомлень, таких як WSDL (Web Services Description Language) та UDDI (Universal Description, Discovery and Integration).

GraphQL - це запитова мова та середовище виконання запитів, яке дозволяє клієнтам отримувати тільки ті дані, які вони потребують. GraphQL

використовує один єдиний end point для взаємодії з сервером та дозволяє клієнтам специфікувати, які дані вони хочуть отримати.

MQTT (Message Queuing Telemetry Transport) – це протокол передачі повідомлень, який використовується для збору даних з IoT-пристроїв та інших джерел даних. MQTT використовує легкий протокол транспорту та підтримує взаємодію клієнта та сервера за допомогою публікації/підписки на теми.

Ці моделі обміну даними мають свої переваги та недоліки, тому важливо вибрати той, який найкраще відповідає потребам вашого проєкту для обміну даними чат-ботів в Telegram зазвичай базуються на протоколі обміну повідомленнями (Messaging Protocol). Протокол дозволяє взаємодіяти між користувачами та ботами в режимі реального часу.

Основні моделі обміну даними, які використовуються в чат-ботах Telegram:

- **Request-Response Model**

Представляє собою архітектурну модель, яка використовується в Телеграм чат-ботах, де користувач робить запит до бота, а бот надсилає відповідь на запит користувача.

В рамках цієї моделі, коли користувач починає спілкування з ботом, він надсилає запит боту, наприклад, запит про погоду в певному місці або запит про розклад автобусів. Бот отримує цей запит і обробляє його за допомогою свого алгоритму. Після обробки запиту, бот надсилає відповідь користувачеві. Відповідь може бути у вигляді текстового повідомлення, зображення або голосового повідомлення. Користувач може продовжувати спілкування з ботом, надсилаючи нові запити, і процес буде повторюватися доти, доки користувач не закінчить спілкування з ботом. [16]

- **Push Model**

Push Model – це архітектурна модель, яка використовується в телеграм чат-ботах для автоматичної відправки повідомлень користувачам без запиту від них.

У Push Model, бот може надсилати повідомлення користувачам на основі певних подій, які відбуваються в системі. Наприклад, бот може автоматично надсилати повідомлення про новини, погоду, оновлення у соціальних мережах або інші події, які можуть зацікавити користувача.

Для використання Push Model, бот повинен мати доступ до різних API, щоб відслідковувати певні події і надсилати відповідні повідомлення користувачам. Ця модель може бути особливо корисною для бізнесу, який бажає повідомляти своїх клієнтів про нові товари, послуги або знижки.

Проте важливо зазначити, що Push Model повинна використовуватися з обережністю, оскільки надлишкове «спамнення» може призвести до негативного враження користувачів та недовіри до бота. Тому, у більшості випадках при розробці рекомендовано використовувати Push Model з обмеженнями, наприклад, дозволяти користувачам підписуватися на повідомлення або надсилати повідомлення не частіше, ніж один раз на день.

- **Webhooks Model**

Webhooks Model – це архітектурна модель, яка використовується в чат-ботах Telegram для отримання автоматичних повідомлень про події, що відбуваються в чат-боті.

У Webhooks Model відбувається двостороннє спілкування між Telegram API та сервером чат-бота. Коли користувач починає спілкування з чат-ботом, Telegram API надсилає запит на сервер чат-бота, щоб повідомити про новий вхідний запит від користувача. Сервер чат-бота отримує цей запит, обробляє його та надсилає відповідь Telegram API, яка буде відображена у чаті з користувачем. Після цього, Telegram API надсилає

повідомлення на сервер чат-бота про будь-яку іншу подію, яка відбувається у чат-боті (наприклад, коли користувач натискає кнопку в чаті). [17]

Webhooks Model є корисним інструментом для чат-ботів, які потребують постійного зв'язку з сервером, і дозволяє забезпечити швидкий та надійний обмін даними між Telegram API та сервером чат-бота. Проте, важливо забезпечити надійність та безпеку серверу чат-бота, оскільки невідомі запити можуть стати загрозою для безпеки сервера та даних користувачів.

Кожна модель має свої переваги та недоліки та може бути використана в залежності від потреб користувачів та типу бота. В будь-якому випадку, чат-боти Telegram дозволяють забезпечити ефективну взаємодію з користувачами та автоматизувати бізнес-процеси.

Чат-бот для Телеграм, що розробляється є типовим прикладом моделі Request-Response. Користувач обиратиме із запропонованих чат-боту йому варіантів для взаємодії у інтерактивній панелі з кнопками. Наприклад, користувач обере функцію «Записатися на прийом». Отримавши запит, чат-бот надасть користувачу список доступних днів для запису. Обравши дату, чат-бот надасть список доступних годин для запису. У разі, якщо із доступних годин для запису не задовольняють користувача, то він може вийти із обраної дати і запустити команду знов. Чат-бот надасть знову перелік доступних дат. Кількість запитів за проміжок часу буде залежати від можливостей швидкого завантаження наявних даних для обробки запита.

2.3. Алгоритми обміну даними

Алгоритм обміну даних представляє собою чіткий набір інструкцій та правил для передачі даних між двома або більше пристроями чи програмами. Він визначає, які дані передаються, у якому форматі та за якою логікою обмін відбувається.

У контексті чат-ботів Telegram алгоритми обміну даними – це способи взаємодії між чат-ботом та серверами Telegram, які дозволяють передавати дані з користувачами чату, виконувати операції з базою даних та інші функції. Для обміну даними з чат-ботами Telegram можуть використовуватися різні алгоритми, зокрема:

HTTP API. Це стандартний протокол обміну даними, який використовується для передачі запитів до Telegram-серверів. Для цього використовуються HTTP-запити, які можуть бути відправлені з будь-якої мови програмування. API Telegram дозволяє виконувати різні операції з чат-ботами, зокрема, надсилати та отримувати повідомлення, відправляти файли та створювати клавіатури. [18]

Bot API. Цей алгоритм дозволяє розробникам створювати власні чат-боти для Telegram з використанням спеціальної бібліотеки Bot API. Ця бібліотека надає доступ до різних функцій Telegram, таких як отримання повідомлень, відправка повідомлень, створення клавіатур тощо. Bot API можна використовувати з різними мовами програмування, включаючи Python, Java, PHP та інші. [19]

Webhooks. Цей алгоритм дозволяє розробникам отримувати повідомлення в режимі реального часу, необхідні для роботи з чат-ботом, через HTTP-запити. При використанні цього алгоритму чат-бот спочатку реєструється на серверах Telegram, після чого сервери Telegram відправлять HTTP-запит до URL, вказаного розробником, коли будь-який користувач надішле повідомлення до чат-бота. [20-21]

Наприклад, одним з найпоширеніших алгоритмів є HTTP API, який дозволяє взаємодіяти з Telegram-серверами через HTTP-запити. Алгоритми, такі як Bot API та Webhooks, дозволяють розробникам взаємодіяти з серверами Telegram, використовуючи спеціальні бібліотеки та підключення до URL-адрес. Bot API визначає, які дані можуть передаватися між ботом

та Telegram-серверами, які команди може виконувати бот та які дані можуть бути доступні для користувача.

Для розробки Telegram чат-боту використовуватиметься можливості алгоритму Bot API. Найбільша перевага у використанні Bot API для розробки чат-ботів Telegram є простота використання бібліотек з широким спектром регулюючих функцій. Зокрема для написання чат-боту використовуватиметься мова Python та відповідна йому бібліотека набору функцій для розробника python-telegram-bot [21]. Для запису використовуватиметься веб-застосунок Google Sheets. Google Sheets - веб-додаток, який розроблений компанією Google для роботи з електронними таблицями з вбудованим у систему інтелектуальних технологій для аналізу даних. [22] Для роботи безпосередньо з гугл таблицями для запису даних в них буде використовуватися платформа Google Cloud, яка представляє собою веб додаток для управління хмарним сховищем у проектах розроблених у продукціях Google. [23] Отримані дані з повідомлення після успішного виконання функції будуть записані у відповідний рядок для даних.

Кожен алгоритм має свої переваги та недоліки, тому при розробці чат-бота треба визначити, який з перелічених алгоритмів підійде для реалізації функціоналу того чи іншого проєкта. В будь-якому випадку, алгоритми обміну даними в контексті чат-бот Telegram є ключовим елементом взаємодії між чат-ботом та його користувачами.

РОЗДІЛ 3. РОЗРОБКА ТЕЛЕГРАМ-БОТУ

3.1. Розробка інформаційно-логічної моделі

Основним етапом у розробці чат-боту є створення набору функціоналу та сценаріїв їх поведінки при взаємодії. Для графічного відображення сценарію поведінки чат-боту при виконанні обраної користувачем функції, використовується схема алгоритму. схема алгоритму представляє собою схематичну реалізацію процесу або алгоритму. Для створення логічного ланцюга взаємодії користувача з ботом виконуватимуться наступні кроки:

1) Визначення основних функцій, з якими користувач може взаємодіяти
2) Визначення змісту функції
3) Створення умов виконання функцій та ланцюгу зв'язків функцій

Після визначення основних функцій взаємодії та логічних ланцюгів зв'язку можна розпочати створення

Також, слід визначити вихідну точку із блок схеми. Для чат-боту, який розробляється для виходу із схеми алгоритму взаємодії чат-боту з користувачем буде встановлена exit-point після успішного запису на прийом користувачем.

1) Визначення основних функцій, з якими користувач може взаємодіяти

Відповідно до галузі для якої розробляється чат-бот (сфера надання послуг стоматологічної клініки), основними категоріями функцій будуть:

- відображення інформації (info)

Дана категорія функцій після виконання не потребуватиме додаткових вхідних даних

- запис на прийом (action)

Дана категорія представлятиме функції після виконання яких потрібна буде додаткова вхідна інформація від користувача

Більш детально перелік функцій взаємодії користувачем з ботом можна представити у вигляді наступної таблиці (таблиця 3.1).

Табл. 3.1 Перелік функцій взаємодії користувача з чат-ботом

Назва функції взаємодії	**Назва категорії функцій**
“Інформація про клініку” (info_клініка)	відображення інформації (info)
“Інформація про спеціалістів” (display info_спеціалісти)	відображення інформації (info)
“Послуги” (info_послуги)	відображення інформації (info)
“Контакти” (info_контакти)	відображення інформації (info)
“Запис на прийом”	запис на прийом(info)
“Назад”	відображення інформації (info)
Input_Date wait	запис на прийом(action)
Input_Time wait	запис на прийом(action)
display_date	запис на прийом (action)
display_time	запис на прийом (action)

Input_Name wait	запис на прийом (action)

2) Визначення змісту функцій

Кожна функція взаємодії чат-боту представляє собою команду, яка при виклику користувачем відображає інформацію або виконує функцію, яка була викликана.

Всі функції категорії “відображення інформації” (info) відображають інформацію обрану користувачем:

- функція “Інформація про клініку” (info_клініка) - відображає інформацію з файла “Про клініку”
- функція “Інформація про спеціалістів” (info_спеціалісти) - відображає інформацію з файла “Спеціалісти”
- функція “Послуги” (info_послуги) - відображає інформацію з файла “Перелік послуг”
- функція “Контакти” (info_контакти) - відображає інформацію із файла “Контакти”
- функція “Назад” - слугує функцією повернення у головне меню

Категорія функцій “Запису на прийом” представляє собою набір взаємозалежного ланцюгу виконання запису на прийом до клініки:

- функція Input_Date wait та display_date - взаємозалежні функці: перша функція очікує, коли користувач обере бажану дату з переліку, друга відповідає за відображення користувачу доступних днів для запису
- функція Input_Time wait та display_time - взаємопов’язані та виконуються тільки після виконання двох попередніх функцій. Аналогічно до попередніх функцій: перша очікує вхідних даних бажаного користувачем часу прийому, друга відповідає за відображення доступних годин прийому.

3) Створення умов виконання функцій та ланцюгу зв’язків функцій

Усі функції чат-боту стають доступні лише після запуску користувачем команди Start. Вона буде стартовою точкою у схему алгоритмів взаємодії користувача з чат-ботом. Після успішного старту чат-боту, будуть доступне головне меню функціоналу чат-боту, яка складається із перелічених функцій:

- функція “Інформація про клініку”
- функція “Інформація про спеціалістів”
- функція “Послуги”
- функція “Контакти”
- функція “Запис на прийом”

При виконанні перелічених функцій окрім “Запису на прийом” слідкуватиме контент, який описаний у пункті 2 (“Визначення змісту функцій”) та можливість повернутися у Головне меню або у інші розділи меню через функцію “Назад”.

При обранні користувачем функції “Запис на прийом” слідкуватимуть наступні функції display_date та Input_Date wait. Після того як користувач обере дату: display_time та Input_Time wait. Користувач повинен обрати із запропонованого списку годину прийому. Далі бот очікує введення користувачем свого ім’я.

4) Створення схеми алгоритмів

Після того як основний набір функцій був визначений і були встановлені взаємозв’язки між ними, будується схема адгоритму. Як було зазначено вище у пункті 3 (“Створення умов виконання функцій та ланцюгу зв’язків функцій”) кожна схема алгоритму починається з стартової точки, який задає початок процесу. Ним слугуватиме блок “/start”. Робота чат-боту буде безперервною, але для забезпечення логічного завершення у схему алгоритму вихідна точка “End” успішним виходом з неї. Вихід з схеми алгоритму відбувається одразу після блоку як користувач введе свій номер

телефону і система автоматично збереже дані. Схема алгоритму представлена у додатку (Додаток А).

3.2. Програмна реалізація чат-боту та його тестування

Розробку бота-консультанта для стоматологічної клініки в Телеграм можна розбити на такі етапи:

1) Створення чат-боту за допомогою офіційного менеджера ботів BotFather в Телеграм та отримання токену доступу.
2) Обрання бібліотеку для розробки чат-боту Телеграм та імпортувати у файл розробки
3) Встановлення конфігурації для чат-боту
4) Створення інтерактивного меню кнопок
5) Підключення електронної таблиці Google Sheets. Виведення та запис даних.
6) Підключення чат-боу до серверу хостинга
7) Тестування функціоналу чат-боту

1) Створення чат-боту за допомогою офіційного менеджера ботів BotFather в Телеграм та отримання токену доступу.

Першим кроком у розробці чат-боту для Telegram - це отримання токену. Токен представляє собою ключ з яким розробник матиме доступ до бота: доступ до керування та редагування властивостей та функціоналу бота. Для цього у пошуковому полі месенджеру Телеграм вписуємо назву офіційного бота BotFather. У меню чат-бота або у поле прописати команду /newbot (рис. 3.1). Першим, потрібно вказати ім’я для чат-боту. Після цього користувацьке ім’я (username). Для чат-боту, який розробляється використано ім’я Dentalist та username - dentalist_bot.

Рис. 3.1. Меню чат-боту BotFather з командами

Як тільки всі дані будуть введені, бот надасть посилання на створеного бота та токен доступу (рис. 3.2). Цей токен потрібно буде використати як ключ підключення до функціоналу чат-боту.

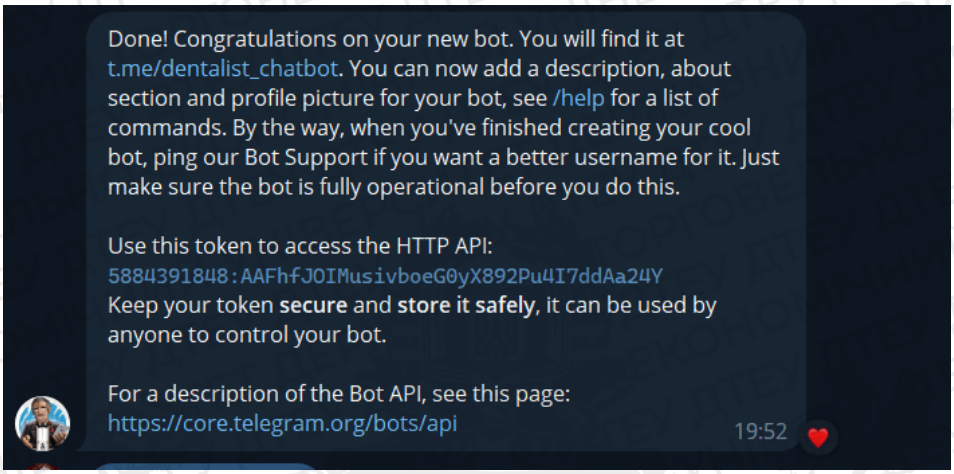


Рис. 3.2. Токен доступу до чат-бота

2) Вибір бібліотеки для розробки чат-боту Телеграм

Для розробки чат-ботів для Телеграм на мові програмування Python існує численна кількість бібліотек. Бібліотеки для написання чат-ботів включають у собі набір методів та функцій, які потрібні для розробки чат-ботів. Серед найбільш використовуваних бібліотек для розробки чат-ботів виділяються:

- TeleBot
- aiogram
- python-telegram-bot

Для розробки чат-боту Telegram використовуватиметься бібліотека TeleBot. Вона підходить для розробки невеликих проектів, має широкий спектр функцій і методів для написання функціоналу чат-боту Telegram. Також, обрана бібліотека може розробити бота, який працюватиме як асинхронно, так і синхронно. Синхронні боти працюють у порядку черги, тобто, якщо перший запит не був оброблений наступний запит буде в режимі очікуванні. Асинхронний бот працює незалежно від місця запиту у

черзі, а в порядку готовності відповіді на запит. Чат-бот, який розробляється буде асинхронним.

3) Встановлення конфігурації для чат-боту

Після отримання токену та вибору бібліотеки для розробки чат-боту, потрібно налаштувати конфігурації та імпортувати бібліотеки для розробки. Для початку треба встановити необхідно встановити основну бібліотеку для безпосередньої розробки чат-бота. Попередньо, в етапі “Вибір бібліотеки для розробки чат-боту Телеграм” була обрана бібліотека TeleBot. Через термінал встановлюємо бібліотеку TeleBot. Для цього в терміналі прописуємо команду pip install telebot. Після того як telebot був встановлений імпортуємо його у файл розробки main.py, за допомогою import. Також додатково для наступних етапів розробки потрібно встановити та підключити бібліотеки gspread для читання даних з електронних таблиць, oauth2client, а саме модуль ServiceAccountCredentials клієнтська для встановлення доступу до електронної таблиці даних через посилання, також імпортувати модуль types для створення інтерактивних кнопок.

Після того як всі бібліотеки були встановлені, потрібно встановити конфігурацію чат-бота. Для цього нам треба використати токен доступу до редагування функціоналу чат-боту. Записуємо у змінну токен чат-боту та передаємо його у параметри методу Telebot. (рис. 3.3)

main.py × {} autoregister_cred.json

```
import telebot
import gspread
from oauth2client.service_account import ServiceAccountCredentials
from telebot import types

api_token = '5884391848:AAEvsI9sWSMerjK9EED-D16wKZSQdhLKVtA'
bot = telebot.TeleBot(api_token)
bot.set_webhook()
```

Рис. 3.3. Підключення бібліотек та конфігурація чат-боту

4) Створення інтерактивного меню кнопок

Для початку треба встановити message handler який буде розпочинати роботу чат-боту після отримання від користувача команди /start або при першому запуску бота користувач може натиснути на “Розпочати”.

Після цього створимо функцію bot_message, яка в параметр буде приймати message (повідомлення від користувача). Для створення меню з кнопками потрібно створити клавіатуру. Спочатку створимо змінну markup, в яку пізніше додаватимуться усі створені кнопки. За допомогою методу types.KeyboardButton створюємо кнопки. Набір кнопок були описані у питанні 3.1. Назви кнопок: “Про нас”, “Наші спеціалісти”, “Перелік послуг”, “Контактні дані”, “Запис на прийом” У параметри передаємо назви кнопок. Метод дублюється для кожної кнопки. Також, додатково додаємо повідомлення з привітанням при запуску чат-бота. У кінці запускаємо передаємо в метод bot.send_message параметри змінну з привітанням, markup з доданими в нього кнопками і розпарсинг рядка привітання з урахуванням гіпертекстуальної розмітки (HTML).

```
@bot.message_handler(commands=['start'])
def start(message):
    markup = types.ReplyKeyboardMarkup(True, True)
    gen_info = types.KeyboardButton('Про нас')
    staff_info = types.KeyboardButton('Наші спеціалісти')
    service_info = types.KeyboardButton('Перелік послуг')
    contact_info = types.KeyboardButton('Контактні дані')
    register = types.KeyboardButton('Запис на прийом')
    markup.add(gen_info, staff_info, service_info, contact_info, register)
    greetings = 'Вас вітає стоматолічна клініка <b>Dentalist</b>! Наш чат-бот допоможе Вам записатися на прийом.'
    bot.send_message(message.chat.id, greetings, parse_mode='html', reply_markup=markup)
```

Рис. 3.5. Блок коду головного меню кнопок після старту

Далі треба прописати функціонал кожної кнопки чат-боту. У меню є набір інтерактивних кнопок, які після натискання виводять лише інформацію. Такими кнопками є: “Про нас”, “Наші спеціалісти”, “Перелік послуг”, “Контактні дані” (детальніше описано у питанні 3.1). Так як для кожної такої кнопки потрібно виводити повідомлення, можна створити окрему функцію (openfilename), яка буде читати текстовий файл і

передавати дані з них у reply повідомлення чат-бота. (рис. 3.6) Функція прийматиме в якості параметра повідомлення на яке треба відповісти та шлях до текстового файлу.

```
4 usages
def openfilename(message, path):
    with open(path, "rb") as f:
        contents = f.read().decode("UTF-8")
        bot.reply_to(message, contents)
        return
```

Рис. 3.6. Функція відкриття файлу з текстом

Для того, щоб користувач міг зручно переходити в інші пункти головного меню, потрібно створити для кожної кнопки клавіатуру з усіма кнопками, крім кнопки, на яку натиснув користувач. Також, для зручності, можна додати інтерактивну кнопку “Назад”, натискання на яку повертає користувача у головне меню.

Кнопки створюються аналогічно до головного меню, тільки за винятком для цього буде створена окрема функція, в якій кожна кнопка буде виступати умовою при натисканні на неї. Її код дублюється з кнопки головного меню. Також, слід додати умову: перевірка, де був викликаний чат-бот. Якщо це приватні повідомлення, то функція активується. Якщо бота додали до каналу Телеграм, то бот не запускатиметься. (рис. 3.7) За винятком, замість кнопки, на яку натиснув користувач буде кнопка “Назад”. (рис. 3.8)

```python
@bot.message_handler(content_types=['text'])
def bot_message(message):
    if message.chat.type == 'private':
        if message.text == 'Про нас':
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
            staff_info = types.KeyboardButton('Наші спеціалісти')
            service_info = types.KeyboardButton('Перелік послуг')
            contact_info = types.KeyboardButton('Контактні дані')
            register = types.KeyboardButton('Запис на прийом')
            back = types.KeyboardButton('Назад')
            markup.add(staff_info, service_info, contact_info, register, back)
            openfilename(message, "gen_info.txt")
            photo_about = 'about.jpg'
            with open(photo_about, 'rb') as photo_file:
                bot.send_photo(message.chat.id, photo_file)
            bot.send_message(message.chat.id, 'Про нас', reply_markup=markup)
```

Рис. 3.8. Блок функціональної кнопки “Про нас”

```python
elif message.text == 'Назад':
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    gen_info = types.KeyboardButton('Про нас')
    staff_info = types.KeyboardButton('Наші спеціалісти')
    service_info = types.KeyboardButton('Перелік послуг')
    contact_info = types.KeyboardButton('Контактні дані')
    register = types.KeyboardButton('Запис на прийом')
    markup.add(gen_info, staff_info, service_info, contact_info, register)
    bot.send_message(message.chat.id, 'Назад', reply_markup=markup)
```

Рис. 3.9. Блок функціональної кнопки “Назад”

Після цього можна буде підключити створену функцію читання з текстового файла openfilename для виведення даних в повідомлення. Для цього викликаємо її у блоку функціоналу кожної кнопки і передаємо в параметри повідомлення та шлях до текстового файлу.

Тепер коли функціонал інформаційних кнопок був завершений, переходимо до розробки функціоналу для кнопки “Запис на прийом”. Її основна функція - це виводити інформацію користувачу про вільні дати для запису, запис користувача на обрану ним вільний час запису та збір персональних даних клієнта для запису в електронну таблицю.

5) Підключення електронної таблиці Google Sheets. Виведення та запис даних.

Для початку треба створити електронну таблицю з якої бот буде отримувати інформацію про вільні дні запису. Для цього Google Sheets створюємо новий файл “Електронний запис Dentalist 2023”. У діапазоні B3:O3 будуть розміщені дати, в діапазоні A4:A14 - години роботи клініки. Відповідно кожної дати будуть розміщені контактні дані кожного клієнта. Неділя буде вихідним днем у розкладі, тому в кожній клітинці колонки з неділями потрібно проставити “вихідний”, щоб коли функція буде проходити по клітинках, не визначить ці дні як вільні для запису. Загальний вигляд електронної таблиці наведено у скріншоті нижче. (рис. 3.9)

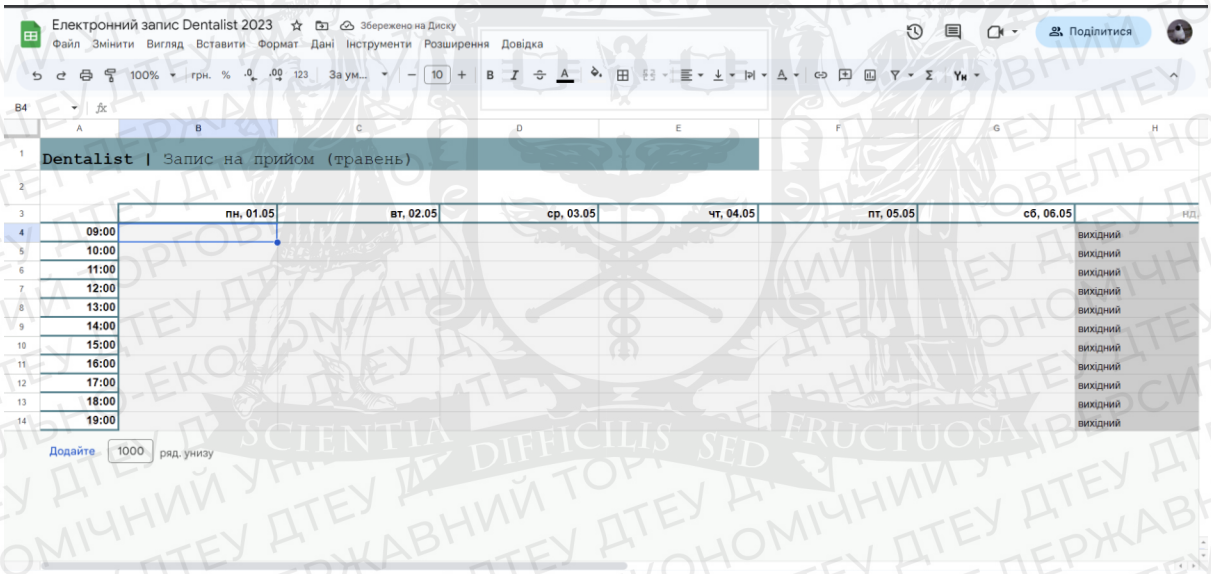


Рис. 3.9. Таблиця в Google Sheets для запису на прийом

Після того як була створена гугл таблиця потрібно створити новий проект в Google Cloud. Це дозволить отримати безпосередній доступ до читання та редагування файлу, який зберігається в хмарному сховищі Google Drive. Для цього перейти в акаунт Google Cloud та у “Select a project” > “New project”. Далі треба обрати ім’я і натиснути “Create”. Також, для працювання саме з електронними таблицями треба встановити додатковий програмний інтерфейсі (API) для проєкту. А саме у вкладці “API & services” > “Library” у пошукове поле вписуємо дві назви API Google Sheets API і

Google Drive API. (рис. 3.10) Почергово підключаємо кожен з них. Перший API потрібен для редагування та читання файлів Google Sheets, другий - для доступу до хмарного сховища.

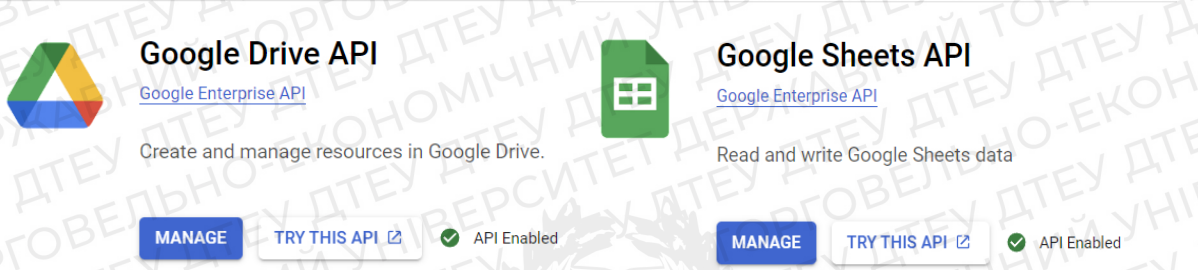


Рис. 3.10. Google Drive API та Google Sheets API

Далі “API & services” > “Credentials” > “Create credentials” > “Service account” . Тут вводиться ім’я для акаунту, ідентифікатор акаунту і опис (наприклад: для чого створений акаунт). Після цього натискаємо “Done”. (рис. 3.11) Цей акаунт потрібен, щоб мати доступ до редагування та читання файлу через python файл. Іншими словами акаунт служить ботівським сервісним акаунтом, в які будуть передаватися команди через python файл.

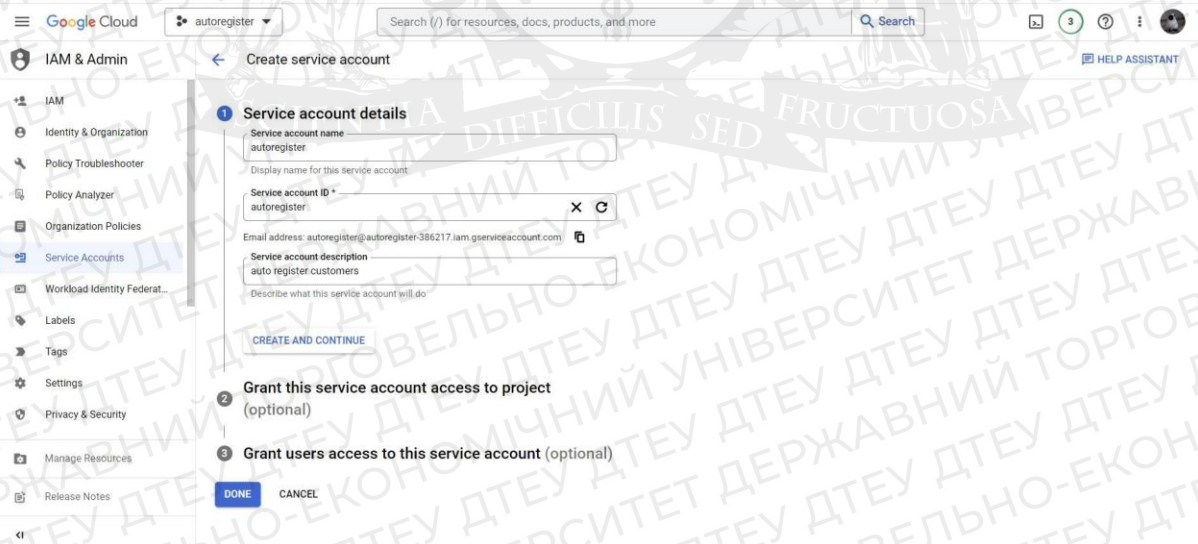


Рис. 3.11. Створення сервісного акаунту в Google Cloud

Після успішного створення акаунту буде автоматично скачений json файл з ключами доступу та електронною поштою новоствореного сервісного облікового запису. Якщо файл з ключами доступу автоматично не завантажився, то його можна скачати: “IAM & Admin” > “Service

Accounts” > “Keys”. Отриманий файл треба додати у папку разом з файлом, в якому розробляється чат-бот (main.py).

Далі потрібно надати доступ до редагування створеному сервісному акаунту у електронній таблиці запису на прийом. Для цього у файлі електронної таблиці натиснути “Поділитися” вписати електронну пошту сервісного акаунту та надати право редагувати файл. Після цього можна встановити всі конфігурації для читання з таблиці у файл розробки чат-боту.

Спочатку у блоці коду функціоналу кнопки “Запис на прийом” надаємо доступ до читання та редагування файлу. Для цього отриманий ключ до сервісного акаунта облікового запису (credentials) запишемо у окрему змінну credentials методом ServiceAccountCredentials.from_json_keyfile_name і передамо туди шлях до json файлу ключа, та змінну scope в якій зберігаються посилання на api які були підключені до проекту в Google Cloud. Далі - авторизація в акаунт через метод бібліотеки gspread, передача посилання на файл електронної таблиці та створення змінної, яка буде зберігати назву листа, з якого будуть отримуватися дані. (Рис. 3.12)

```
elif message.text == 'Запис на прийом':
    scope = ['https://www.googleapis.com/auth/spreadsheets', "https://www.googleapis.com/auth/drive.file",
             "https://www.googleapis.com/auth/drive"]
    credentials = ServiceAccountCredentials.from_json_keyfile_name('autoregister_cred.json', scope)
    client = gspread.authorize(credentials)
    spreadsheet = client.open('Електронний запис Dentalist 2023')
    worksheet = spreadsheet.sheet1
```

Рис. 3.12. Блок конфігурації файлу електронної таблиці Google Sheets

Для того, щоб бот зміг вивести інформацію про наявні вільні дні для запису, спочатку потрібно створити функцію, яка буде зчитувати інформацію з електронної таблиці. Передаємо у новостворену функцію ключ-посилання на електронну таблицю та назву аркушу, в якій знаходяться дані, діапазон, в якому знаходяться дати та години запису. Це

будуть заголовками для кожної колонки і рядка відповідно. Далі створюємо цикл який буде перевіряти кожну колонку, а в кожній колонці рядок на наявність пустої клітинки. Пуста клітинка в даному контексті означатиме, що день та година в яку можна записатися. (рис. 3.13) Також паралельно будуть записуватися у змінну, яка буде зберігати: колонки - вільні дні, рядкі - вільна година і змінна яка зберігатиме адрес клітинки для того, щоб після цього при виборі клієнтом саме цього дня та години запису, записати туди його дані. Діапазон, в якому розташовані заголовки таблиці (робочі дні та години): робочі дні - B3:O3, робочі години - A4:A14.

```python
# Функція для отримання вільних днів
def get_free_days():
    days_range = worksheet.range('B3:O3')
    free_days = []
    for cell in days_range:
        if cell.value == '':
            free_days.append(cell.col)
    return free_days

# Функція для отримання вільних годин
def get_free_times(day):
    times_range = worksheet.range('A4:A14')
    free_times = []
    for i, cell in enumerate(times_range):
        if worksheet.cell(i + 4, day).value == '':
            free_times.append(cell.value)
    return free_times
```

Рис. 3.13. Функції отримання даних з Google Sheets про вільні дні та час запису

Після того як дані отримані, треба вивести їх у назви кнопок. Для цього використовуємо цикл, в якому будуть присвоюватися імена які знаходяться у списку вільних дат для запису. Тобто кнопок буде створено стільки скільки є назв (дат) у списку. Додатково потрібно провести перевірку на те чи є у змінній вільних днів дані. Це можна виконати за допомогою умови: якщо довжина list днів більша нуля, то спрацьовує блок умови. Відповідно

користувачу виводяться всі вільні дати для запису і можливість обрати день. (рис. 3.14) Як тільки користувач обере кнопку з бажаною датою. Спрацює блок з кодом обробки вибору користувачем години запису. Алгоритм роботи блоку обробки години запису аналогічний до обробника дати. За винятком, користувач може повернутися у головне меню з кнопками, якщо серед запропонованих вільних годин запису немає бажаного.

```python
# Обробка вибору користувачем дня
def select_day(update: Update, context: CallbackContext):
    query = update.callback_query
    if query.data == 'book':
        free_days = get_free_days()
        if len(free_days) > 0:
            keyboard = []
            for day in free_days:
                keyboard.append([InlineKeyboardButton(str(day), callback_data=str(day))])
            reply_markup = InlineKeyboardMarkup(keyboard)
            query.message.reply_text('Оберіть день для запису:', reply_markup=reply_markup)
            return SELECTING_DAY
        else:
            query.message.reply_text('На жаль запис на даний момент не можливий :(')
            return ConversationHandler.END
    elif query.data == 'exit':
        return ConversationHandler.END
```

Рис. 3.14. Функція обробки вибору дня користувачем

Далі порібно прийняти від користувача отримати ім’я а номер телефону. Кожне повідомлення користувача буде записане і передане після цього у змінні name і phone, які записуються в адрес клітинки дня та години, на який записався клієнт. (рис. 3.15)

```python
# Отримання контактних даних користувача
def confirm(update, context):
    context.user_data['name'] = update.message.text
    query = update.callback_query
    query.message.reply_text('Введіть своє ім\'я:')
    context.user_data['phone'] = update.message.text
    query = update.callback_query
    query.message.reply_text('Введіть свій номер телефону:')

# Функція для збереження даних та підтвердження запису
def save_data(update, context):
    context.user_data['phone'] = update.message.text
    selected_day = context.user_data['selected_day']
    selected_time = context.user_data['selected_time']
    name = context.user_data['name']
    phone = context.user_data['phone']
    cell_address = chr(selected_day + 64) + str(3 + int(selected_time.split(':')[0]))
    worksheet.update(cell_address, f'{name}, {phone}')
    update.message.reply_text('Ваш запис було успішно збережено!')
    return ConversationHandler.END
```

Рис. 3.15. Функції: отримання контактних даних користувачем та збереження даних у Google Sheets

У електронну таблицю зберігається ім’я користувача та його номер телефону. Всю інформацію у таблиці може редагувати власник електронної таблиці (наприклад: адміністрація) або чат-бот. Навіть, якщо посилання на таблицю отримає третя сторона, вона не зможе вносити зміни так як права на редагування у файлі обмежені за замовчуванням. Повний текст коду представлений у останньому додатку. (Додаток Б)

6) Підключення чат-боу до серверу хостинга

Вибір хостингу для чат-боту перш за все залежить від навантаженості та використання трафіку. Для малих бізнесів, у яких клієнтська база є невеликою, достатньо до 100000 треків в день. Для розробленого чат-боту був обраний хостинговий сервер PythonAnywhere.

Для того, щоб завантажити файли з чат-ботом на сервер хостингу потрібно спочатку створити акаунт. Для випускної кваліфікаційної роботи обрано безкоштовний тариф для демонстрації можливостей чат-боту.

У секції “Dashboard” потрібно обрати “Browse files” та створити нову директорію “DentalistChatBot”. Далі треба завантажити всі файли разом з файлом програмного коду у цю директорію. (рис. 3.16)

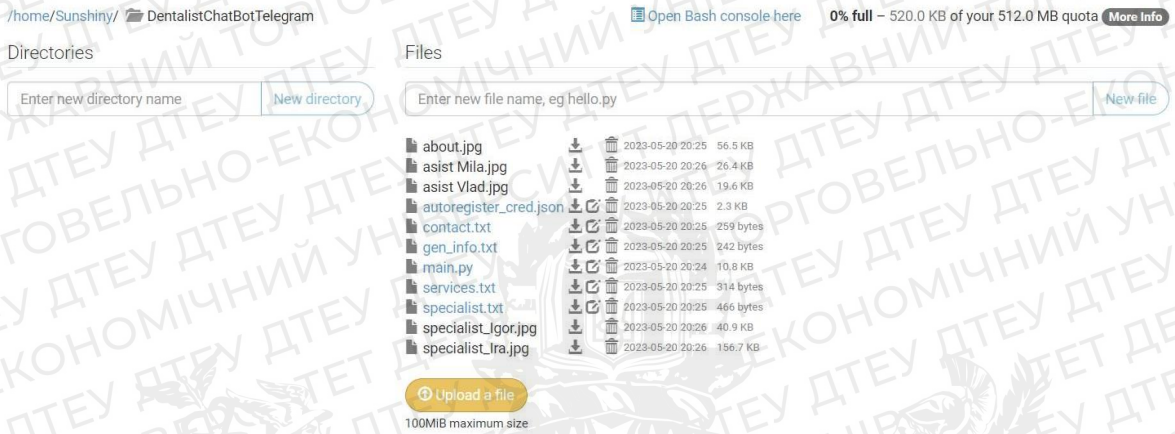


Рис. 3.16. Завантаження файлів в директорію хостинг PythonAnywhere

Далі переходимо Dashboard > Bash. У консолі потрібно встановити всі бібліотеки, які використовуються, щоб бот міг функціонувати на сервері. Для цього використовувати команду pip install та почергово встановити: telebot, gspread, auth2client, telegram. Після цього потрібно через команду ls отримати доступ директорії на сервері, де був завантажений чат-бот. Як і в попередній раз, потрібно запустити файл з програмним кодом чат-боту через команду python main.py. (рис. 3.17) Тепер чат-бот працюватиме автоматично від серверу.

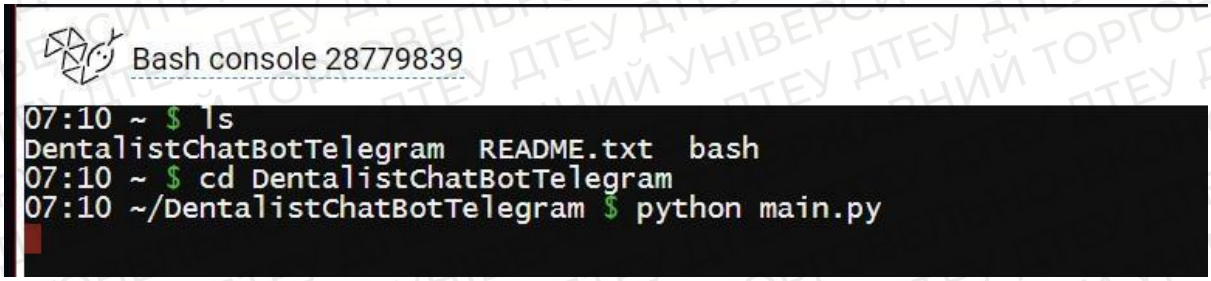


Рис. 3.17 Запуск програмного коду чат-бота на сервері

7) Тестування функціоналу чат-бота

Для тестування чат-боту потрібно перейти у безпосередній чат з ботом за посиланням. Перш за все потрібно протестувати response чат-боту на команду /start. (рис. 3.18)

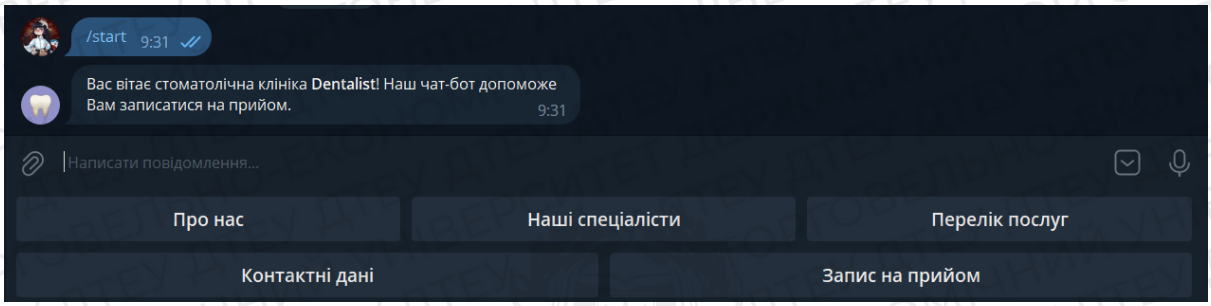


Рис. 3.18. Тестування команди /start

/start - команда яка розпочинає роботу чат-боту і виводить головне меню кнопок для користувача після виклику.

Далі потрібно протестувати роботу кожної кнопки інтерактивного меню кнопок. Всі кнопки крім “Запису на прийом” є інформаційними блоками, в яких лише виводиться текст з інформацією. Кнопки “Про нас” та “Наші спеціалісти” містять в собі крім тексту - фотографії. Треба перевірити, також відображення кнопок “Назад”, яка має повертати користувача в головне меню з кнопками. Для початку протестуємо роботу кнопки “Про нас” (рис. 3.19) та “Наші спеціалісти” (рис. 3.20), які мають виводити загальну інформацію про клініку, спеціалістів та прикріплені фотографії.

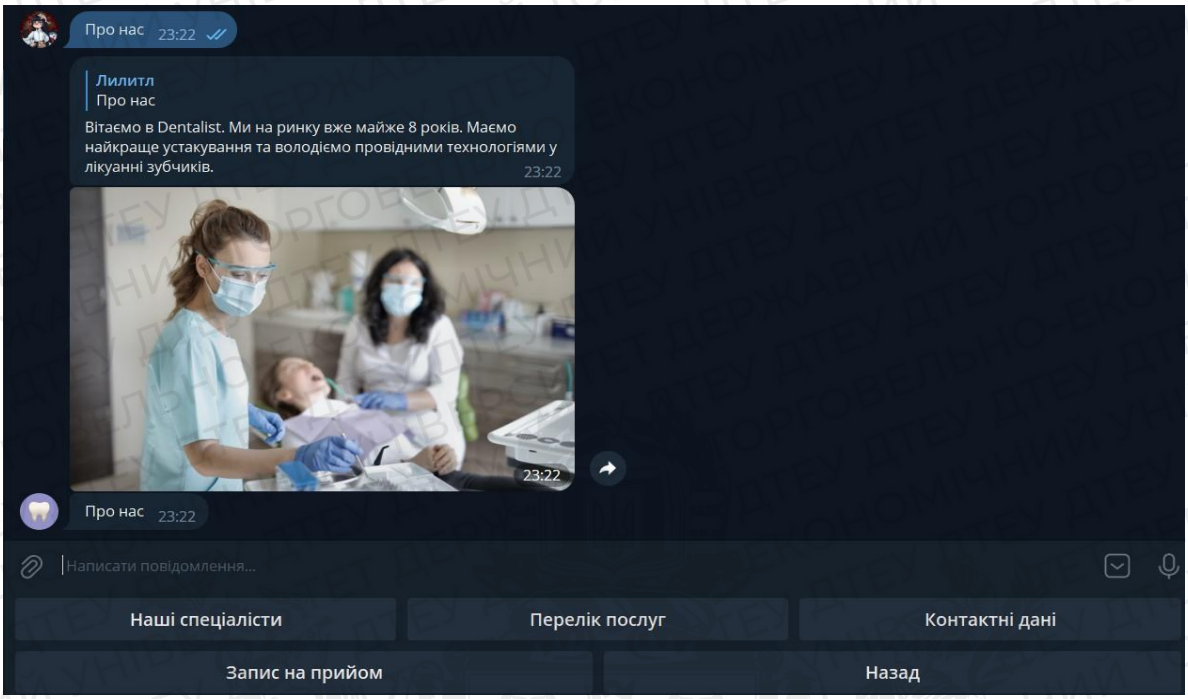


Рис. 3.19. Тестування функціональної кнопки “Про нас”

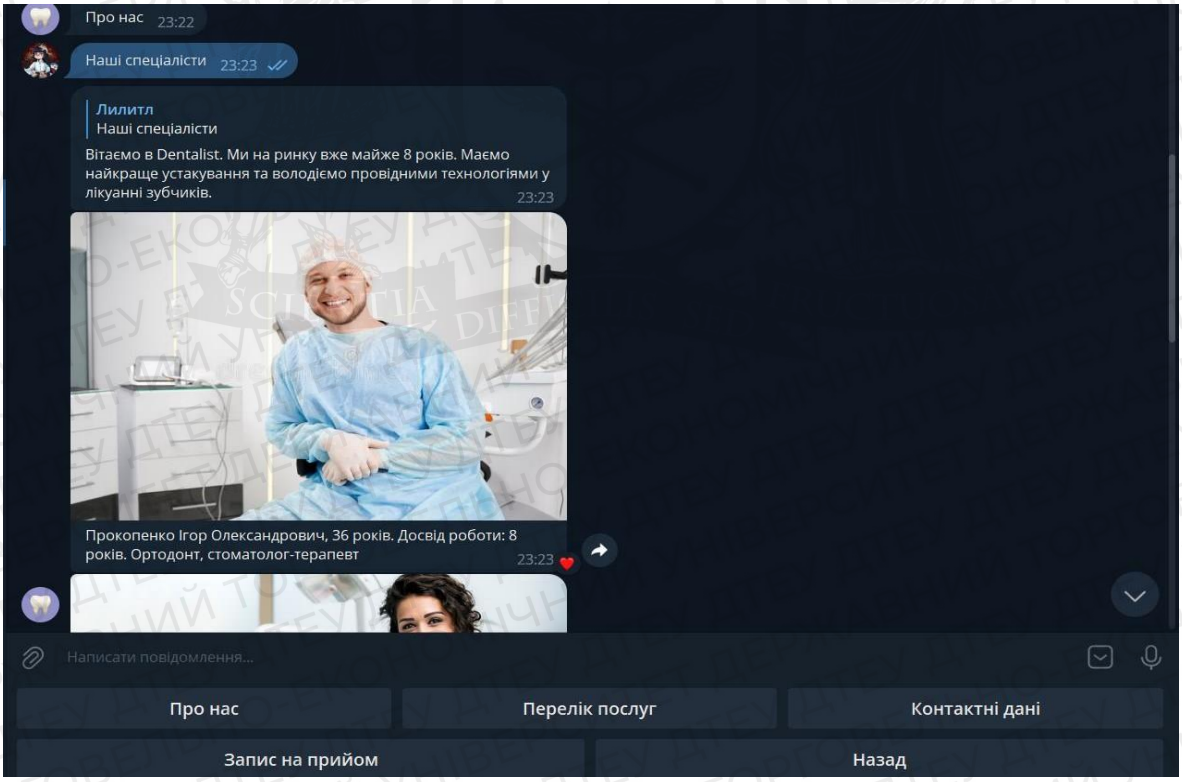


Рис. 3.20. Тестування функціональної кнопки “Наші спеціалісти”

У кнопках “Контактні дані” та “Перелік послуг” відображається лише текст. (рис. 3.21) Також, при натисканні на інформаційні кнопки меню містить всі пункти окрім, в який перейшов користувач. Замість неї відображається функціональна кнопка “Назад”, яка переносить користувача в головне меню.

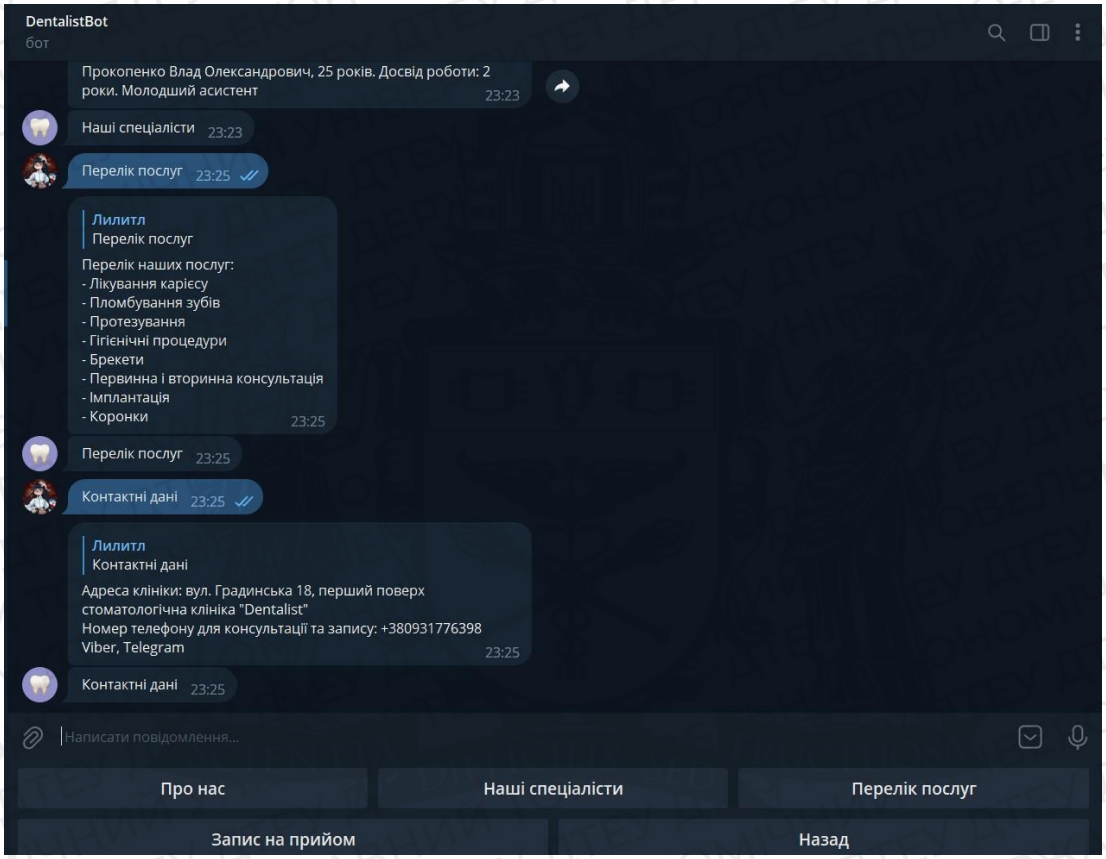


Рис. 3.21. Функціонування кнопок “Перелік послуг” та “Конактні дані”

Далі потрібно перевірити функціонування кнопки “Запис на прийом”. Після натискання на неї, користувачу відображається спочатку вільні дати отримані з електронної таблиці, () а потім як користувач обере серед запропонованих дату для запису - відображаються години запису.

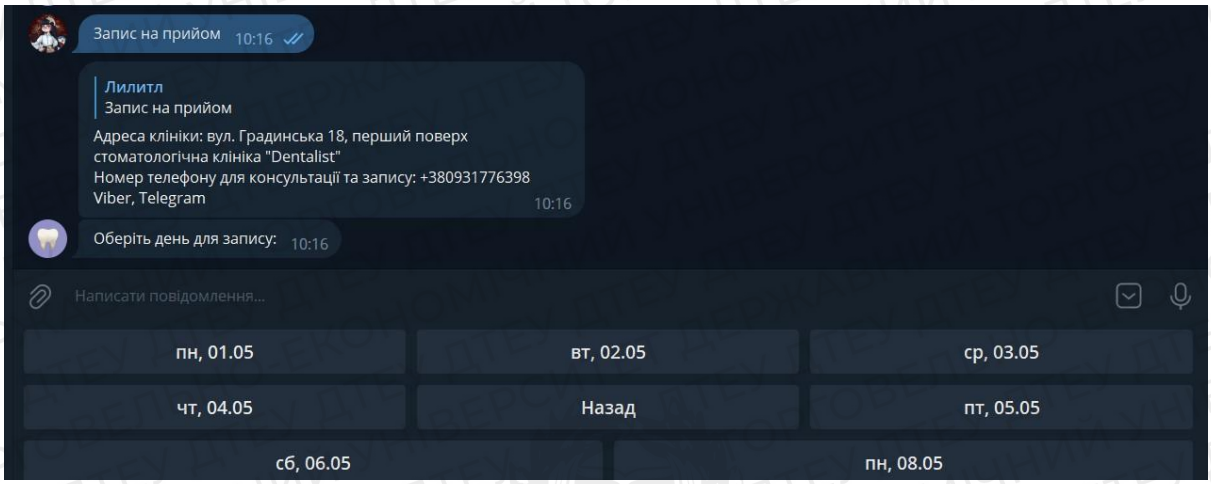


Рис. 3.22. Відображення перелік да для запису

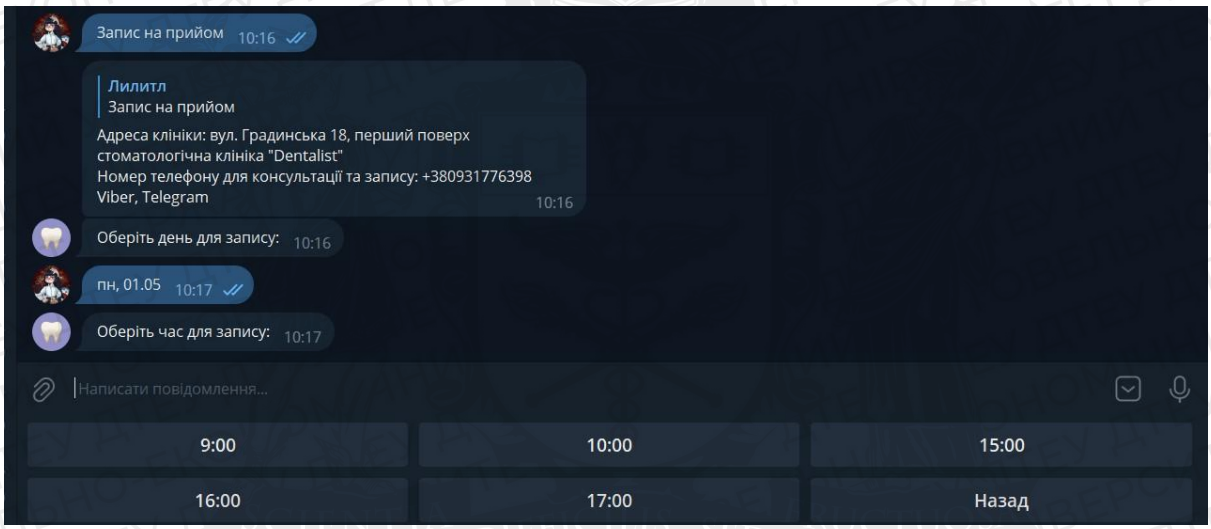


Рис. 3.23. Відображення переліку годин для запису

Далі користувач має ввести свої контактні дані (рис. 3.24) і ця інформація зберігається автоматично у електронну таблицю (рис. 3.25) у відповідну ячейку з обраним днем і часом запису.

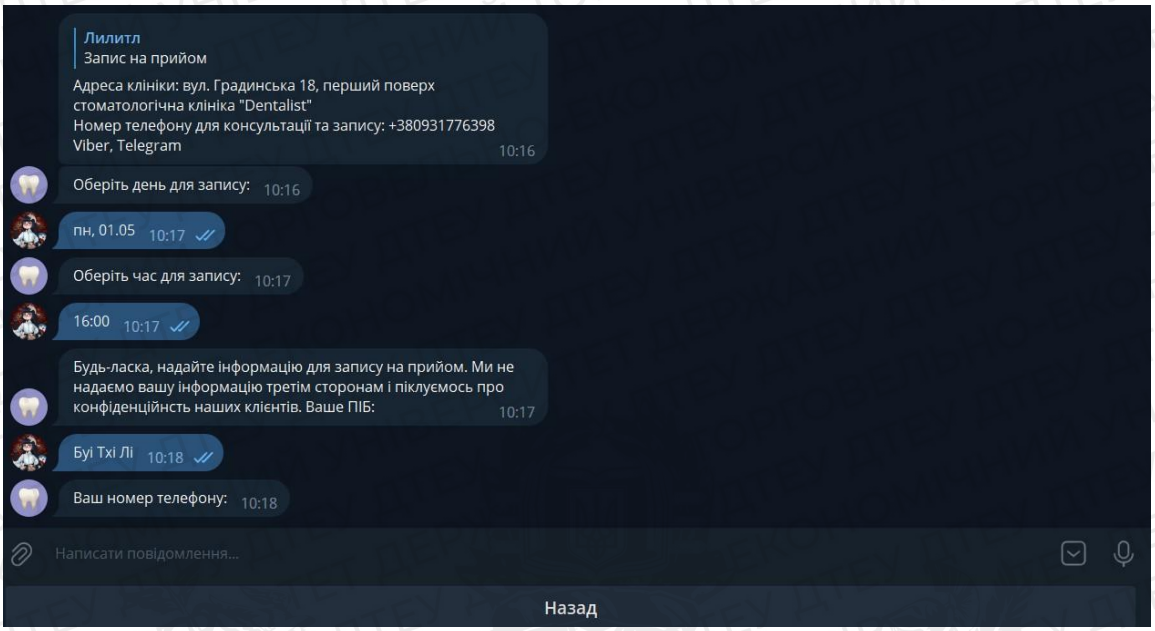


Рис. 3.24. Отримання від користувача контактних даних

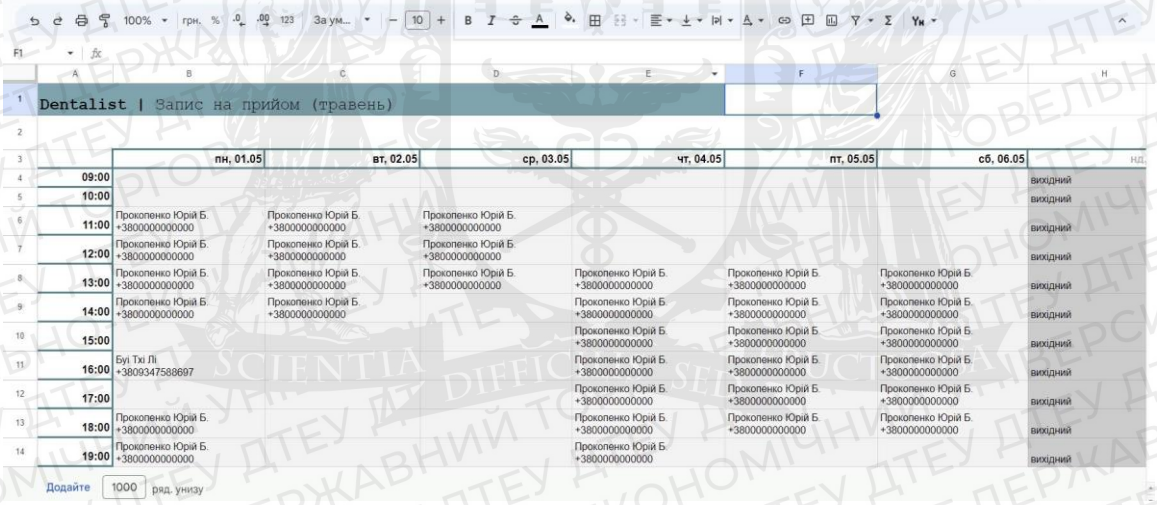


Рис. 3.25. Запис контактних даних у електронну таблицю Google Sheets

У цьому пункті були протестовані всі функціональні кнопки чат-боту. Перевірено: коректність відображення кнопок, відображення тексту у повідомленнях чат-бота, функція “Запису на прийом”, запис контакних даних користувача у Гугл Таблиці.

3.3. Технологія використання чат-боту

На сьогоднішній день найбільшу частку чат-ботів Телеграм використовуються саме у сферах надання послуг. Впровадження Телеграм

ботів-консультантів у в стоматологічних клініках і, в цілому, у сферу надання послуг, зменшить навантаження на адміністрацію закладів чи компаній та вірогідність непередбачених конфліктів спричинені людським фактором.

Розроблений чат-бот у даній випускній кваліфікаційній роботі призначений для автоматизації процесу надання інформації клієнту про стоматологічну клініку та запису на прийом. Функціонал чат-боту включає в собі основні пункти, які зазвичай мають бути представлені клієнтові для отримання ним інформацію про клініку. Це функції:

- “Про нас”, в якій надається загальна інформація про клініку
- “Наші спеціалісти”, в якому список представлений список спеціалістів клініки
- “Перелік послуг”, список послуг, які надаються клінікою
- “Контактні дані”, контакти клініки (адрес, номер телефону)
- “Запис на прийом” пункт, в якому користувач може переглядати вільні дати для запису на прийом і записуватися на обраний час

Інформація, яку виводить чат-бот в повідомлення може бути відредагована, маючи доступ до файлів з текстом. Наприклад, у стоматологічній клініці змінився список послуг, які надаються або список спеціалістів, то текст можна відредагувати.

Також, цей чат-бот може бути переписаний в залежності від напрямку чи галузі, в якому компанія надає послуги. Для цього лише потрібно замінити дані, які виводить чат-бот в повідомлення та назви кнопок. Наприклад, чат-бот може бути переписаний під косметологічну клініку або під сервісний бот-консультанта салон краси.

Розроблений чат-бот, може використовуватися у **мал**их бізнесах які надають стоматологічні послуги або може, в залежності, від фінансових можливостей компаній буи модифікований і перероблений під обробник інформації для мережі клінік. Наприклад, якщо клініка є частиною мережі

бренду, то можна у функціонал запису на прийом додати пункт з обранням адреси клініки, щоб клієнт міг сам обрати, яка клініка є зручною для відвідування. Тоді для кожної клініки треба розробити окрему електронну таблицю і збільшити квоти на читання файлів у Google Cloud.

Підсумовуючи, розроблений чат-бот Телеграм є зручним інструментом у автоматизації та контролю процесу адміністрування у стоматологічних клініках. Було також визначено, що розроблений чат бот може використовуватися для інших сфер послуг крім стоматологічних клінік. Це сфери надання косметологічних, юридичних, адміністративних та інші послуги, які щільно пов’язані з організацією, адмініструванням та плануванням часу клієнтів. Додатково, чат-бот може бути модифікований під більші за розміром проєкти. На даний момент розроблений бот для мессенджеру Телеграм може задовольнити потреби малого бізнесу. Так як не володіє невеликим обсягом виділеного на сервері трафіку, він може обробляти до 100000 вхідних запитів в день.. Зазвичай хостинги, які надають безкоштовні тарифи не можуть забезпечити великими обсягами трафіку та пам’яті хмарного сховища. Тому проєкти з використанням трафіку у більше 100000 запитів потребують додаткової тарифікації.

ВИСНОВКИ

У випускній кваліфікаційній роботі представлені результати виконання теоретичного дослідження структури та алгоритму роботи чат-ботів. Досліджено методи та алгоритми розробки чат-ботів з використанням мови програмування Python, зокрема, бібліотеки для розробки чат-ботів, набір функціоналу чат-ботів для електронного запису, метод запису даних користувачів у Google Sheets. Усі теоретичні дослідження лягли в основу розробки чат-боту Телеграм для стоматологічної клініки. На основі отриманих результатів можна зробити висновки:

1. Месенджери та соціальні мережі стали одним з основних ресурсів отримання інформації та маркетингових платформ для просування послуг та продукції компаній. Більшість малих і великих бізнесів інтегруються у соціальні мережі для комунікації з таргетинговою аудиторію свого бренду.
2. Чат-бот для мессенджера Телеграм - ефективний інструмент у автоматизації процесу адміністрування у сфері послуг для малих і великих бізнесів. Найбільш поширені напрямки використання чат-ботів: консультування клієнтів, надання інформації, реалізація електронного запису.
3. Усі чат-боти сфери послуг мають набір функціоналу, який залежатиме від галузі діяльності компанії чи організації, яка використовує дану розробку. Наприклад, якщо це сфера надання стоматологічних послуг, то важливим для клієнтів-користувачів є функції чат-боту по наданню загальної інформації про клініку (контакти, спеціалісти, перелік послуг) та можливість електронного запису на прийом до лікаря.
4. Google Sheets може використовуватися для обліку даних клієнтів та для автоматизації процесу запису в електронну чергу. За допомогою

Гугл Таблиць можна зберігати персональну інформацію клієнтів та час запису без ризиків порушення конфіденційності даних, що зберігатимуться. Інформація захищена oauth2, доступ до якої можна отримати лише після отримання посилання та прав на редагування.

5. Розроблений чат-бот мессенджера Телеграм може використовуватися для автоматизації процесу адміністрування та планування часу відвідування клієнтів стоматологічної клініки. Безперервна робота чат-боту надає змогу клієнтам записуватися на послуги клініки у будь-який час без відвідування закладу. Клієнтська база розміщена на веб-застосунку Google Sheets, що забезпечує зручний доступ адміністрації до роботи з нею.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chat-Bot For College Management System Using A.I. / Bala K., Kumar M., Hulawale S., Pandita S. / International Research Journal of Engineering and Technology (IRJET) 2017 - p. 8-29.
2. The Study of the Application of a Keywords-based Chatbot System / Jia J. on the Teaching of Foreign Languages/ arXiv 2003 arXiv:cs 0310018 - p. 20-21.
3. A Review on Chatbot Design and Implementation Techniques / Kumar R., Ali M.M. /International Research Journal of Engineering and Technology (IRJET) 2020 - p. 20-21.
4. ELIZA–A Computer Program for the Study of Natural Language Communication between Man and Machine / Weizenbaum J. Commun. ACM 1966 - p. 9.
5. Potor M What are the different types of chatbots? [Електронний ресурс] Режим доступу: https://www.messengerpeople.com/what-are-the-different-types-of-chatbots/
6. Alburger J. Rule-Based Chatbots vs. AI Chatbots: Key Difference. hubtype [Електронний ресурс] Режим доступу: https://www.hubtype.com/blog/rule-based-chatbots-vs-ai-chatbots
7. Why Chatbots Don’t Solve all Customer Service Problems DigitalGenius : [Електронний ресурс] Режим доступу: https://digitalgenius.com/why-chatbots-dont-solve-all-customer-service-problems/
8. Facebook Messenger Hits 100,000 bots VentureBeat: [Електронний ресурс] Режим доступу: https://venturebeat.com/ai/facebook-messenger-hits-100000-bots/
9. A Brief History of the Internet University System of Georgia [Електронний ресурс] Режим доступу: https://www.usg.edu/galileo/skills/unit07/internet07_02.phtml

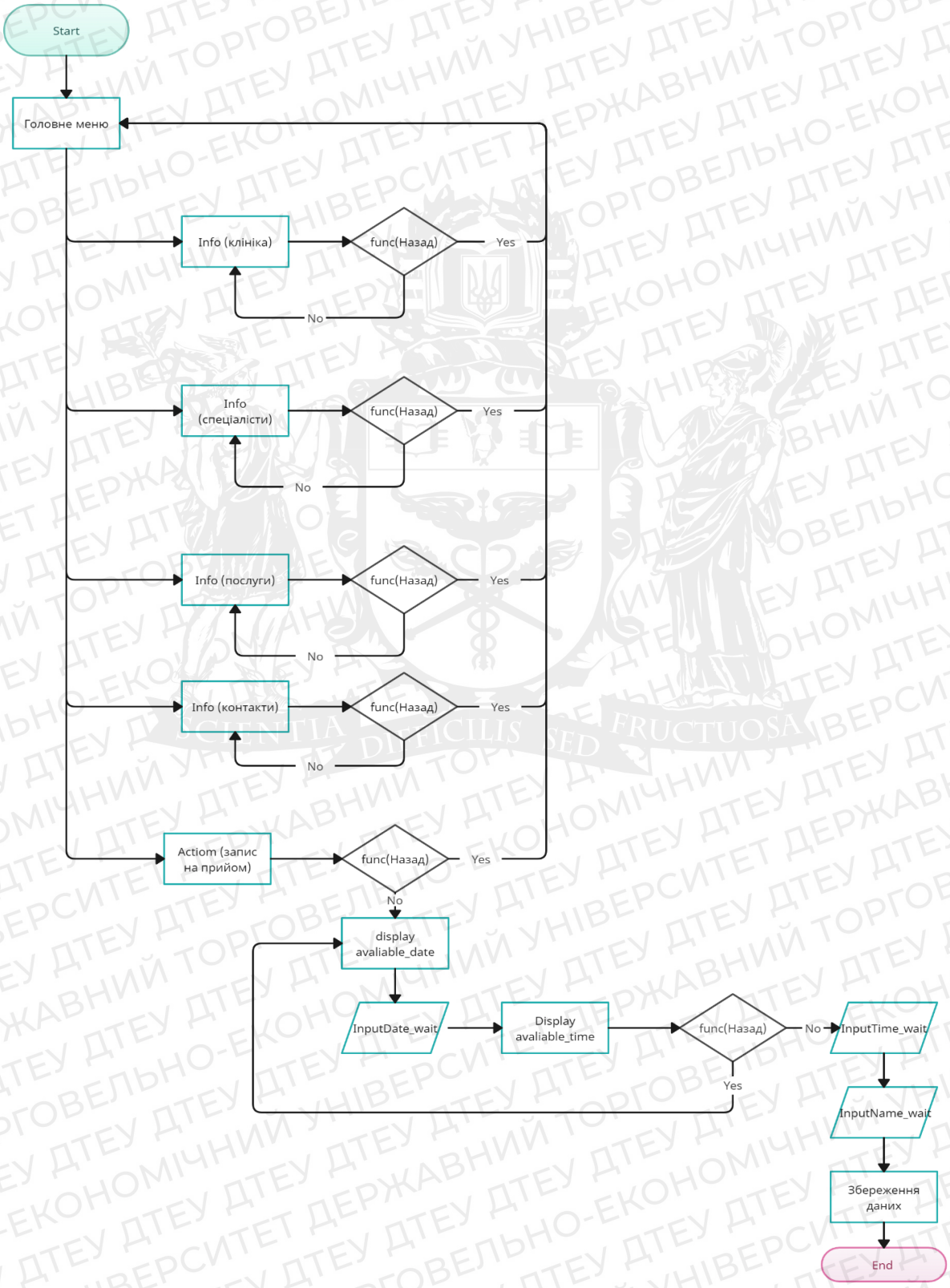
10. ARPANET: Definition & History Study [Електронний ресурс] https://study.com/academy/lesson/arpanet-definition-history-quiz.html
11. “Зоряна”: чат-бот Фейсбук [Електронний ресурс] Режим доступу: http://m.me/Zoriana.Kyivstar
12. “Київводоканал”: чат-бот Viber [Електронний ресурс] Режим доступу: https://chats.viber.com/vodokanalkiev
13. “Justin” : чат-бот Telegram [Електронний ресурс] Режим доступу: https://t.me/justinpostservice_bot
14. CPG Center : чат-бот Telegram [Електронний ресурс] Режим доступу: https://t.me/cpgcenter_bot \
15. OCI: What is chatbot? [Електронний ресурс] Режим доступу: :https://www.oracle.com/chatbots/what-is-a-chatbot/
16. Medium: Difference Between Publish-Subscribe and Request-Response Model [Електронний ресурс] Режим доступу: https://levelup.gitconnected.com/difference-between-publish-subscribe-and-request-response-model-d0aa9f51cf3d
17. Red Hat: What is webhook? [Електронний ресурс] Режим доступу: https://www.redhat.com/en/topics/automation/what-is-a-webhook
18. What are REST APIs? HTTP API vs REST API [Електронний ресурс] Режим доступу: https://www.educative.io/blog/what-are-rest-apis
19. Telegram.org: Telegram Bot API [Електронний ресурс] Режим доступу: https://core.telegram.org/bots/api
20. Web hook to revolutionize the web 2007, archived from the original on 2018-06-30.
21. Jenkins GitHub Commit Hooks HOWTO, archived from the original on 2015-09-25.
22. Telegram.org: Bot API Library Examples [Електронний ресурс] Режим доступу: https://core.telegram.org/bots/samples

23. Google Sheets About [Електронний ресурс] Режим доступу: https://www.google.com/intl/ru/sheets/about/
24. Google Cloud About [Електронний ресурс] Режим доступу: https://cloud.google.com/

ДОДАТКИ

Додаток А

Схема алгоритму роботи чат-боту



Програмний код чат-боту

```
import telebot
import gspread
from oauth2client.service_account import ServiceAccountCredentials
from telebot import types
from telegram import InlineKeyboardButton, InlineKeyboardMarkup, Update
from telegram.ext import Updater, CallbackQueryHandler, CommandHandler, ConversationHandler, CallbackContext

api_token = '5884391848:AAEvsI9sWSMerjK9EED-D16wKZSQdhLKVtA'
bot = telebot.TeleBot(api_token)
bot.set_webhook()

@bot.message_handler(commands=['start'])
def start(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    gen_info = types.KeyboardButton('Про нас')
    staff_info = types.KeyboardButton('Наші спеціалісти')
    service_info = types.KeyboardButton('Перелік послуг')
    contact_info = types.KeyboardButton('Контактні дані')
    register = types.KeyboardButton('Запис на прийом')
    markup.add(gen_info, staff_info, service_info, contact_info, register)
    greetings = 'Вас вітає стоматолічна клініка <b>Dentalist</b>! Наш чат-бот допоможе Вам записатися на прийом.'
    bot.send_message(message.chat.id, greetings, parse_mode='html', reply_markup=markup)

def openfilename(message, path):
    with open(path, "rb") as f:
        contents = f.read().decode("UTF-8")
        bot.reply_to(message, contents)
        return
```

```
@bot.message_handler(content_types=['text'])
def bot_message(message):
    if message.chat.type == 'private':
        if message.text == 'Про нас':
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
            staff_info = types.KeyboardButton('Наші спеціалісти')
            service_info = types.KeyboardButton('Перелік послуг')
            contact_info = types.KeyboardButton('Контактні дані')
            register = types.KeyboardButton('Запис на прийом')
            back = types.KeyboardButton('Назад')
            markup.add(staff_info, service_info, contact_info, register, back)
            openfilename(message, "gen_info.txt")
            with open(photo_about, 'rb') as photo_file:
                bot.send_photo(message.chat.id, photo_file)
            bot.send_message(message.chat.id, 'Контактні дані', reply_markup=markup)

        elif message.text == 'Наші спеціалісти':
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
            gen_info = types.KeyboardButton('Про нас')
            service_info = types.KeyboardButton('Перелік послуг')
            contact_info = types.KeyboardButton('Контактні дані')
            register = types.KeyboardButton('Запис на прийом')
            back = types.KeyboardButton('Назад')
            markup.add(gen_info, service_info, contact_info, register, back)
            openfilename(message, "gen_info.txt")
            photo_spec1 = open('specialist_Igor.jpg')
            with open('specialist_Igor.jpg', 'rb') as photo_file:
                bot.send_photo(message.chat.id, photo_file, caption='Прокопенко Ігор Олександрович, 36 років. Досвід роботи: 8 років. Ортодонт, стоматолог-терапевт ')
            with open('specialist_Ira.jpg', 'rb') as photo_file:
```

```
            bot.send_photo(message.chat.id, photo_file,
caption='Іваненко Ірина Володимирівна, 34 років. Досвід роботи: 8
років. Стоматолог-хірург ')
        with open('asist Mila.jpg', 'rb') as photo_file:
            bot.send_photo(message.chat.id, photo_file,
caption='Прокопенко Міла Михайлівна, 27 років. Досвід роботи: 3
роки. Старша асистентка')
        with open('asist Vlad.jpg', 'rb') as photo_file:
            bot.send_photo(message.chat.id, photo_file,
caption='Прокопенко Влад Олександрович, 25 років. Досвід роботи: 2
роки. Молодший асистент ')
        bot.send_message(message.chat.id, 'Наші спеціалісти',
reply_markup=markup)


    elif message.text == 'Перелік послуг':
        markup =
types.ReplyKeyboardMarkup(resize_keyboard=True)
        gen_info = types.KeyboardButton('Про нас')
        staff_info = types.KeyboardButton('Наші спеціалісти')
        contact_info = types.KeyboardButton('Контактні дані')
        register = types.KeyboardButton('Запис на прийом')
        back = types.KeyboardButton('Назад')
        markup.add(gen_info, staff_info, contact_info, register, back)
        openfilename(message, "services.txt.txt")
        bot.send_message(message.chat.id, 'Перелік послуг',
reply_markup=markup)

    elif message.text == 'Контактні дані':
        markup =
types.ReplyKeyboardMarkup(resize_keyboard=True)
        gen_info = types.KeyboardButton('Про нас')
        staff_info = types.KeyboardButton('Наші спеціалісти')
        service_info = types.KeyboardButton('Перелік послуг')
        register = types.KeyboardButton('Запис на прийом')
        back = types.KeyboardButton('Назад')
        markup.add(gen_info, staff_info, service_info, register, back)
```

```
            openfilename(message, "contact.txt")
            bot.send_message(message.chat.id,      'Контактні      дані',
reply_markup=markup)

        elif message.text == 'Запис на прийом':
            scope   =   ['https://www.googleapis.com/auth/spreadsheets',
"https://www.googleapis.com/auth/drive.file",
"https://www.googleapis.com/auth/drive"]
            credentials                                                  =
ServiceAccountCredentials.from_json_keyfile_name('autoregister_cred.j
son', scope)
            client = gspread.authorize(credentials)
            spreadsheet   =   client.open('Електронний   запис   Dentalist
2023')
            worksheet = spreadsheet.sheet1
            # Функція для отримання вільних днів
            def get_free_days():
                days_range = worksheet.range('B3:O3')
                free_days = []
                for cell in days_range:
                    if cell.value == '':
                        free_days.append(cell.col)
                return free_days
            # Функція для отримання вільних годин
            def get_free_times(day):
                times_range = worksheet.range('A4:A14')
                free_times = []
                for i, cell in enumerate(times_range):
                    if worksheet.cell(i + 4, day).value == '':
                        free_times.append(cell.value)
                return free_times
            # Обробка вибору користувачем дня
            def select_day(update: Update, context: CallbackContext):
                query = update.callback_query
                if query.data == 'book':
                    free_days = get_free_days()
                    if len(free_days) > 0:
```

```
                keyboard = []
                for day in free_days:
                    keyboard.append([InlineKeyboardButton(str(day),
callback_data=str(day))])
                reply_markup = InlineKeyboardMarkup(keyboard)
                query.message.reply_text('Оберіть день для
запису:', reply_markup=reply_markup)
                return SELECTING_DAY
            else:
                query.message.reply_text('На жаль запис на даний
момент не можливий :(')
                return ConversationHandler.END
        elif query.data == 'exit':
            return ConversationHandler.END
    # Обробник вибору часу
    def select_time(update: Update, context: CallbackContext):
        query = update.callback_query
        selected_day = int(query.data)
        free_times = get_free_times(selected_day)
        if len(free_times) > 0:
            keyboard = []
            for time in free_times:
                keyboard.append([InlineKeyboardButton(time,
callback_data=time)])
            keyboard.append([InlineKeyboardButton('Назад',
callback_data='back')])
            reply_markup = InlineKeyboardMarkup(keyboard)
            query.message.reply_text('Оберіть час для запису:',
reply_markup=reply_markup)
            return SELECTING_TIME
        else:
            query.message.reply_text('На жаль запис на даний
момент не можливий :(')
            return ConversationHandler.END
    # Отримання контактних даних користувача
    def confirm(update, context):
        context.user_data['name'] = update.message.text
```

```
            query = update.callback_query
            query.message.reply_text('  Будь-ласка,  надайте інформацію для запису на прийом. Ми не надаємо вашу інформацію третім сторонам і піклуємось про конфіденційність наших клієнтів. Ваше ПІБ:')
            context.user_data['phone'] = update.message.text
            query = update.callback_query
            query.message.reply_text('Ваш номер телефону:')
        # Функція для збереження даних та підтвердження запису
        def save_data(update, context):
            context.user_data['phone'] = update.message.text
            selected_day = context.user_data['selected_day']
            selected_time = context.user_data['selected_time']
            name = context.user_data['name']
            phone = context.user_data['phone']
            cell_address = chr(selected_day + 64) + str(3 + int(selected_time.split(':')[0]))
            worksheet.update(cell_address, f'{name}, {phone}')
            update.message.reply_text('Ваш запис було успішно збережено!')
            return ConversationHandler.END

        elif message.text == 'Назад':
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
            gen_info = types.KeyboardButton('Про нас')
            staff_info = types.KeyboardButton('Наші спеціалісти')
            service_info = types.KeyboardButton('Перелік послуг')
            contact_info = types.KeyboardButton('Контактні дані')
            register = types.KeyboardButton('Запис на прийом')
            markup.add(gen_info, staff_info, service_info, contact_info, register)
            bot.send_message(message.chat.id, 'Назад', reply_markup=markup)

    bot.polling(none_stop=True)
```