МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка чат-боту технічної підтримки відділу комп'ютерних інформаційних технологій Львівського національного університету природокористування для мессенджера Telegram»**

Виконав: студент 4 курсу групи Іт-41

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Некига Максим Ігорович
(прізвище та ініціали)

Керівник: к.е.н. Станько В.Ю.

Рецензент: к.т.н., доцент Сиротюк С.В.

ДУБЛЯНИ – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
126 – «Інформаційні системи та технології»

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Некига Максим Ігорович

1. Тема роботи: «Розробка чат-боту технічної підтримки відділу комп'ютерних інформаційних технологій Львівського національного університету природокористування для мессенджера Telegram»

Керівник роботи Станько Володимир Юрійович, к.е.н.
Затверджені наказом по університету 30.12.2022 року № 453/к-с.

2. Строк подання студентом роботи 10.06.2023 р.

3. Початкові дані до роботи: 1. Вимоги до побудови інформаційних систем. 2. Науково-технічна і довідкова література. 3. Засоби створення та мова програмування. 4. Методика створення інформаційних систем.

4. Зміст розрахунково-пояснювальної записки:
Вступ
1. Аналіз існуючих платформ функціонування чат-ботів технічної підтримки.
2. Постановка задачі створення чат-боту технічної підтримки.
3. Проектування алгоритму роботи чат-боту технічної підтримки.
4. Практичне використання чат-боту технічної підтримки відділу комп'ютерних інформаційних технологій Львівського національного університету природокористування для мессенджера Telegram.
5. Охорона праці та безпека в надзвичайних ситуаціях.
Висновки та пропозиції.
Бібліографічний список.
Додатки.

5. Перелік графічного матеріалу: Презентація із головними результатами кваліфікаційної роботи.

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Станько В.Ю., в.о. доцента кафедри інформаційних систем та технологій*		
5	*Городецький І.М., доцент кафедри управління проектами та безпеки виробництва*		

7. Дата видачі завдання 30.12.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів роботи	Примітка
1.	*Написання першого розділу та означення головних завдань роботи*	*30.12.2022 – 03.02.2023*	
2.	*Виконання другого розділу та формування головних показників для розрахунків*	*06.02.2023 – 03.03.2023*	
3.	*Виконання третього розділу, розрахунків та розробка листів*	*06.03.2023 – 14.04.2023*	
4.	*Написання розділу: «Охорона праці та безпека в надзвичайних ситуаціях»*	*17.04.2023 – 05.05.2023*	
5.	*Вартісне оцінення ефективності пропозицій роботи*	*08.05.2023 – 19.05.2023*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів графічної частини*	*22.05.2023 – 09.06.2023*	
7.	*Завершення роботи в цілому*	*09-16.06.2023*	

Студент _____ Некига М.І.
(підпис)

Керівник роботи _____ Станько В.Ю.
(підпис)

УДК 004.773.6 (477.83)

Розробка чат-боту технічної підтримки відділу комп'ютерних інформаційних технологій Львівського національного університету природокористування для мессенджера Telegram.

Некига М.І. Кафедра ІТ – Дубляни, Львівський НУП, 2023.

Кваліфікаційна робота: 52 с. текстової частини, 19 рисунків, 14 джерел.

Досліджено ринок месенджерів та здійснено порівняння зручності розробки та рівня підтримки чат-ботів для кожного сервісу. Встановлено, що месенджер Telegram має найкращу інтеграцію чат-ботів та найзмістовнішу документацію для розробників.

Розглянуто теоретичні аспекти розробки та роботи чат-ботів, їхні типи та архітектури, методи взаємодії з мессенджером Telegram. Проведено аналіз наявних на ринку інструментів для розробки чат-ботів для Telegram мовою програмування Python та виявлено, що найпопулярнішим інструментом є Aiogram – реалізація Telegram API, що використовує асинхронні інструменти Python.

Здійснено огляд наявних на ринку рішень баз-даних, який дав можливість обрати оптимальний інструмент для зберігання даних чат-ботом з підтримкою перегляду та самостійного редагування.

Визначено вимоги до реалізації чат-бота та спроектовано його архітектуру. Описано процес розробки чат-боту для месенджера Telegram, використовуючи мову програмування Python, фреймворки Aiogram та Google API Client. Розроблено застосунок чат-бота. Детально описано процес розробки, структуру та принцип роботи проєкту. Продемонстровано процес створення користувачем заявки на технічну підтримку методом спілкування з ботом та пояснено основні алгоритми роботи чат-бота. Розроблено заходи щодо охорони праці.

Ключові слова: чат-бот, заявка, Telegram, технічна підтримка, таблиця, chatbot, application, technical support, table.

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій та широке поширення месенджерів стимулюють впровадження інноваційних рішень у сфері комунікаційних систем. У зв’язку з розвитком ІТ виникає потреба надання якісної технічної допомоги відділом КІТ Львівського національного університету природокористування.

Чат-боти, як ефективний інструмент автоматизації та підтримки користувачів, здобувають все більшу популярність в різних галузях, включаючи технічну підтримку, завдяки своїй швидкості та доступності, оскільки користувач може звернутися до них в будь-який час та в будь-якому місці.

Метою даної роботи є розробка ефективного та зручного інструменту для надання швидкої та якісної підтримки користувачам чат-боту комп'ютерної технічної підтримки для мессенджера Telegram. Це дозволить зменшити навантаження на людський персонал та підвищити задоволеність замовників.

Робота передбачає розробку функціональної та нефункціональної специфікації чат-боту, проектування архітектури, розробка боту для Telegram, його тестування, аналіз результатів розробки та формування висновків. Використовуючи мову програмування Python, бібліотеку Aiogram та Google API буде реалізовано функціонал для взаємодії з користувачами через платформу Telegram.

Результати цієї кваліфікаційної роботи будуть корисними для підприємств чи установ, яким потрібне ефективне та зручне рішення для оброблення запитів користувачів і надання технічної підтримки. Розроблений чат-бот для Telegram дозволить користувачам створювати заявки, у яких вони можуть описати свою проблему або питання. Чат-бот приймає ці заявки, зберігає їх та надає можливість працівникам ІТ-відділу обробити або відповісти на них.

1. АНАЛІЗ ІСНУЮЧИХ ПЛАТФОРМ ФУНКЦІОНУВАННЯ ЧАТ-БОТІВ ТЕХНІЧНОЇ ПІДТРИМКИ

1.1. Означення та визначення чат-ботів

Бот (англ. Bot) – спеціальна програма, що виконує автоматично або за заданим розкладом будь-які дії через інтерфейси, призначені для людей. Зазвичай боти призначаються для виконання одноманітної і повторюваної роботи, з максимально можливою швидкістю. Вони застосовуються за необхідності швидшої реакції, ніж у людини. Це ігрові боти, боти для Інтернет аукціонів тощо. Іноді їх застосовують для імітації дій людини в спілкуванні, тоді їх називають чат-ботами [1, с. 77].

Чат-бот – це інтерактивна програмна платформа, які знаходяться в месенджерах, і може вести себе по-людськи. Така програма здатна відповідати і задавати питання. Чат-боти використовуються в різних сферах для розв'язання типових задач. За даними досліджень компанії Business Insider до 2020 року 80% компаній буде користуватися чат-ботами; 42% учасників опитування вважають, що технології чат-ботів покращать якість обслуговування клієнтів; 48% – вже використовують технологію чат-ботів для таких бізнес-функцій і 40% – планують впровадити деяку форму даної технології до 2020 року [2].

Чат-боти можуть виконувати прохання, відповідати на запитання, розважати. Звичайно, замінити людину такий бот не здатний. Але він легко автоматизує рутинну роботу, миттєво відповідаючи на найпопулярніші запити. Його можна порівняти з менеджером з продажів, який підбирає клієнту найбільш відповідні товари або послуги [3, с. 113].

1.2. Історія розвитку технологій чат-ботів

Витоки сучасних чат-ботів можна простежити до 1964 року, коли Джозеф Вайзенбаум з Массачусетського технологічного інституту (MIT) розробив чат-бота під назвою Eliza. Він використовував прості правила спілкування і перефразовував більшість сказаного користувачами, імітуючи роджеріанського терапевта. Хоча це показало, що наївні користувачі можуть бути обмануті, думаючи, що вони розмовляють зі справжнім терапевтом, сама система не розуміла проблеми користувача. Після цього в 1991 році була заснована премія Лебнера, щоб заохотити дослідників штучного інтелекту створювати чат-ботів, здатних пройти тест Тюрінга і просунути стан ШІ. Хоча до 2014 року жоден чат-бот не пройшов цей тест, багато відомих чат-ботів отримали призи за перемогу в інших складних завданнях. Серед них ALICE, JabberWacky, Rose та Mitsuku. Однак у 2014 році на конкурсі тесту Тьюринга, приуроченому до 60-ї річниці смерті Алана Тьюринга, чат-бот на ім'я Юджин Густман, який зображав 13-річного хлопчика, зумів обдурити 33% суддів – і таким чином подолав тест. Мова розмітки штучного інтелекту (AIML) і ChatScript були розроблені як спосіб написання сценаріїв знань і розмовного контенту для більшості цих чат-ботів. Сценарії, розроблені за допомогою цих скриптових мов, можна потім завантажувати в інтерпретатори для створення розмовної поведінки. Чат-боти, розроблені для подолання тесту Тюрінга, були здебільшого балакучими і мали лише одну мету – подолати тест Тюрінга. Багато хто не вважав це досягненням у галузі штучного інтелекту або створенням корисних розмовних помічників [4, с. 9].

З іншого боку, дослідження в галузі штучного інтелекту, зокрема машинного навчання та обробки природної мови, призвели до появи різноманітних розмовних інтерфейсів, таких як системи відповідей на запитання, інтерфейси природної мови до баз даних та системи розмовного діалогу. На відміну від чат-ботів, створених для подолання тесту Тюрінга, ці системи мали дуже чіткі цілі. Системи відповідей на запитання обробляли запитання природною мовою і знаходили відповіді в

неструктурованих текстових масивах даних. Природномовні інтерфейси до систем баз даних (NLIDBS) – це інтерфейси до великих баз даних SQL, які інтерпретували запити до бази даних, задані природною мовою, наприклад, англійською, перетворювали їх на SQL і повертали збіги як відповідь. Системи розмовного діалогу (SDS) – це системи, які могли підтримувати контекстні розмови з користувачами для виконання розмовних завдань, таких як бронювання квитків, управління іншими системами та навчання учнів. Це були попередники сучасних чат-ботів і розмовних інтерфейсів [4, с. 10].

У 2011 році компанія Apple випустила інтелектуального асистента під назвою Siri у складі iPhone. Siri була змодельована як особистий асистент користувача, що виконує такі завдання, як здійснення дзвінків, читання повідомлень, встановлення будильників і нагадувань. Це одна з найбільш значущих подій недавнього минулого, яка перезавантажила історію розмовних інтерфейсів. У перші дні існування Siri користувачі використовували її лише кілька разів на місяць для виконання таких завдань, як пошук в інтернеті, надсилання SMS та здійснення телефонних дзвінків. Незважаючи на новизну, Siri сприймалася як незавершена робота, до якої в наступні роки буде додано ще багато функцій. На початку свого існування Siri мала багато клонів і конкурентів на Android та інших платформах смартфонів. Більшість із них були змодельовані як асистенти і були доступні у вигляді мобільних додатків [4, с. 10].

У травні 2016 року Google анонсував Assistant, свою версію персонального чат-бота, який був доступний на декількох платформах, таких як додаток Allo і Google Home [4, с. 11].

1.3. Огляд технологій розробки чат-ботів

На сьогоднішній день доступно безліч інструментів для створення та розробки чат-ботів. Простого чат-бота, який здатний проконсультувати чи надсилати сповіщення, можна створити самим, не маючи навичок програмування. Для цього можна скористатися одним із безкоштовних конструкторів [3, с. 113]:

- ManyChat, Chatfuel для Facebook Messenger;
- Manybot для Telegram;
- Meya.ai для всіх основних месенджерів.

Недоліком такого підходу до розробки є погана гнучкість та неможливість інтеграції сторонніх сервісів: неможливо повністю налаштувати чат-бота під свої потреби. Вирішити цю проблему дозволяє розробка чат-бота, використовуючи програмування та офіційні API-документації чат-ботів від розробників месенджерів. API – абревіатура від Application Programming Interface. Дослівно перекладається як інтерфейс прикладного програмування. На практиці API – це набір набір визначень та протоколів, які дозволяють двом програмним компонентам взаємодіяти один з одним [5]. Для усіх популярних месенджерів можна знайти офіційну документацію розробки чат-ботів у відкритому доступі. Часто офіційні API реалізовані на різних мовах програмування, що робить інструменти розробки чат-ботів доступними для більшого кола розробників. До популярних інструментів розробки чат-ботів для месенджерів відносяться:

- **Aiogram** – це популярна бібліотека для розробки Telegram-ботів на мові Python. Вона надає зручний інтерфейс для взаємодії з Telegram Bot API.
- **Botpress** – відкрита платформа для розробки чат-ботів, яка підтримує різні канали зв'язку, включаючи Telegram, Facebook Messenger, Slack та багато інших;
- **Microsoft Bot Framework** – це набір інструментів та ресурсів для створення чат-ботів. Він підтримує різні месенджери, включаючи Facebook Messenger, Skype, Microsoft Teams, Slack та інші;

- **Dialogflow**, раніше відомий як API.AI, є сервісом розробки чат-ботів, який підтримує інтеграцію з різними месенджерами, включаючи Telegram, Facebook Messenger, Viber, Slack та інші;
- **Wit.ai** – платформа штучного інтелекту, яка надає API для розробки чат-ботів. Вона підтримує інтеграцію з різними месенджерами, включаючи Facebook Messenger, Slack, Telegram та інші;
- **Chatfuel** – це інструмент для створення чат-ботів у Facebook Messenger. Він має вбудований візуальний редактор та надає можливості для автоматизації та налаштування чат-ботів;
- **Twilio Programmable Chat** – це платформа зв'язку, яка надає API для розробки рішень у сфері зв'язку. Twilio Programmable Chat дозволяє створювати чат-ботів та інтегрувати їх з різними месенджерами, включаючи WhatsApp, Facebook Messenger та інші.

1.4. Архітектура чат-ботів

Розробка програмного пакету чат-бота вимагає визначення складових частин. Чат-бота можна можна розділити на три частини: “респондент”, “класифікатор” та “графмайстер” (рис. 1.1) [6], які описуються наступним чином:

- **Респондер**: це частина, яка відіграє роль сполучної ланки між основними процедурами бота та користувачем. Завданнями є: передача даних від користувача до класифікатора і контроль введення та виведення даних;
- **Класифікатор**: це частина між респондером і графмастером. Функції цього рівня: фільтрація та нормалізація вхідних даних, сегментація вхідних даних, введених користувачем, на логічні складові, передача користувачем на логічні складові, передача нормалізованого речення до графмайстра, обробка вихідних даних від графмайстра, а також обробка інструкцій графмайстра;

- **Графмайстер**: це частина для зіставлення шаблонів, що впорядковує вміст логіки, зберігання та утримання алгоритмів зіставлення зразків.

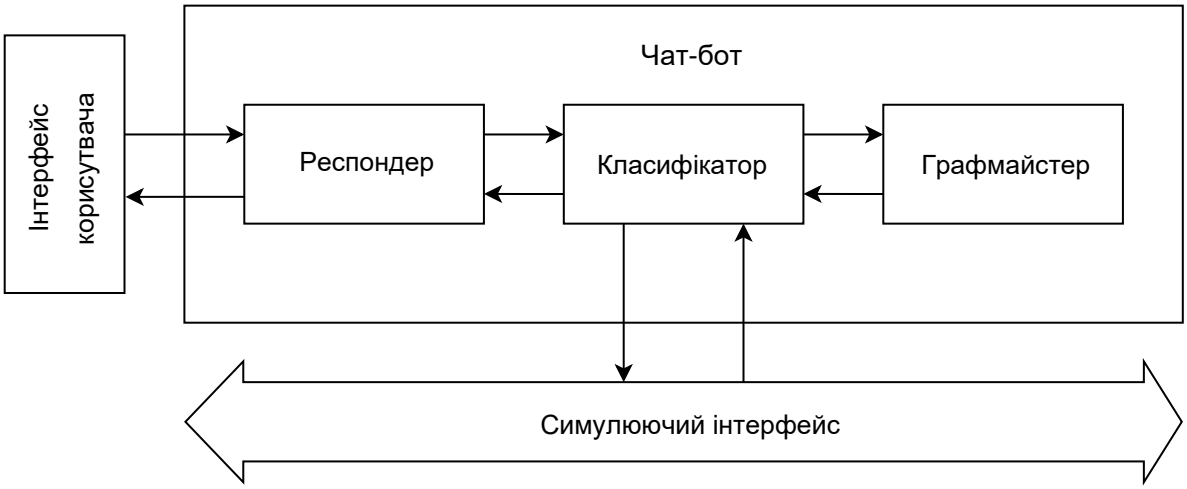


Рисунок 1.1 – Компоненти чат-бота

1.5. Загальний огляд чат-ботів для технічної підтримки

Чат-боти для технічної підтримки є одним з найпоширеніших застосувань чат-бот технологій. Вони дозволяють компаніям автоматизувати та полегшити процес надання технічної допомоги своїм клієнтам. Основною метою таких чат-ботів є надання швидких та ефективних відповідей на запитання та проблеми, пов'язані з комп'ютерною технікою.

Характеристики типових чат-ботів для технічної підтримки включають наступне:

- **Функціональність**: Чат-боти для технічної підтримки можуть мати різні функції, що допомагають вирішувати проблеми користувачів. Вони можуть надавати інформацію про продукти та послуги, допомагати в усуненні неполадок, проводити діагностику проблем, надавати посилання на документацію та інструкції, а також створювати та відстежувати заявки на технічну підтримку;
- **Інтерактивність**: Чат-боти для технічної підтримки можуть бути розроблені таким чином, щоб взаємодіяти з користувачами у форматі

діалогу. Вони можуть задавати додаткові питання, уточнювати інформацію та надавати інструкції в режимі реального часу, щоб забезпечити більш точну та персоналізовану підтримку;

- **Інтеграція з базою знань**: Багато чат-ботів для технічної підтримки підтримують інтеграцію з базами знань або системами управління знаннями. Це дозволяє ботам отримувати доступ до великої кількості інформації та надавати користувачам точні та релевантні відповіді на їх запитання;
- **Мультиплатформеність**: Багато чат-ботів для технічної підтримки підтримують різні месенджери та платформи, такі як Telegram, Facebook Messenger, WhatsApp тощо. Це дозволяє користувачам звертатися за підтримкою за допомогою зручної для них платформи;
- **Аналітика та вдосконалення**: Багато чат-ботів для технічної підтримки надають аналітичні звіти та метрики використання. Це дозволяє підприємствам аналізувати ефективність своїх чат-ботів, виявляти слабкі місця та вдосконалювати їх функціональність для забезпечення найкращої технічної підтримки.

Крім цього, чат-боти для технічної підтримки дозволяють підприємствам заощадити кошти. Інвестиції потрібні тільки на етапі розробки та впровадження. Надалі технічна підтримка коштує набагато дешевше, ніж утримання штату операторів онлайн-чату або call-центру. Та й технічних засобів потрібно набагато менше (відсутня необхідність в комп’ютерному обладнанні, робочих місцях тощо) [7, с. 27].

1.6. Аналіз існуючих рішень на основі Telegram API

Telegram API є одним з популярних інструментів для розробки чат-ботів. Цей API надає потужний набір функцій та можливостей, що дозволяють

створювати різноманітні чат-боти для різних цілей, включаючи технічну підтримку.

Серед існуючих рішень на основі Telegram API можна виділити:

- Чат-бот технічної підтримки сервісу «Дія» – надає підказки та відповіді на поширені питання щодо сервісу “Дія” в автоматизованому режимі (рис. 1.2). За потреби виступає посередником між клієнтом та оператором. Чат-бот дозволяє отримати якісну допомогу та автоматизовані відповіді на часті питання. Це значно зменшує навантаження на людський персонал технічної підтримки.

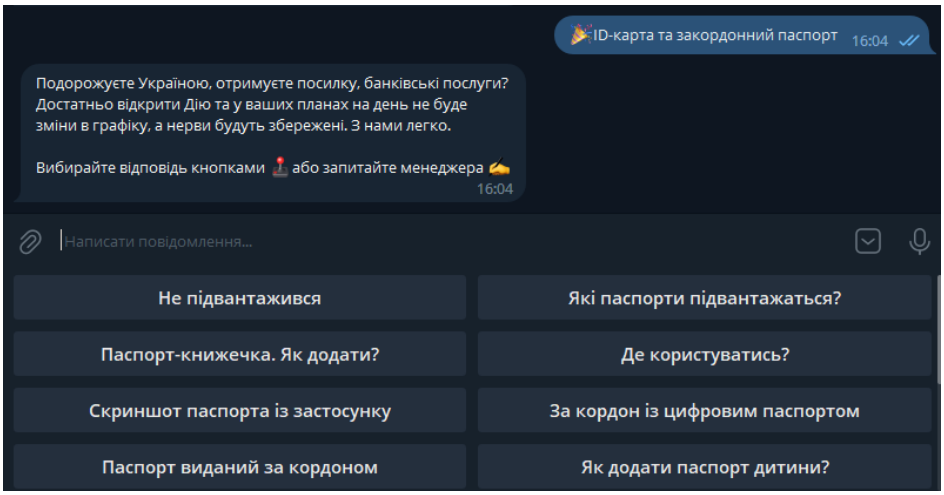


Рисунок 1.2 – Чат-бот технічної підтримки сервісу “Дія”

- Чат-боти ПриватБанку – дозволяють надіслати запитання менеджеру та отримувати на них відповіді, здійснювати платежі, подавати заявки по еквайрингу і переглядати історію транзакцій. ПриватБанк надає свої послуги одразу за допомогою кількох чат-ботів, кожен із них має своє призначення. Перевагою такого підходу підвищена загальна стабільність, оскільки неполадки в роботі одного чат-боту не впливають на роботу інших. Недоліком такого рішення є можлива плутанина при користуванні одразу кількома чат-ботами. На рисунку 1.3 зображено чат-бота ПриватБанк для зв’язку з менеджером та отримання технічної підтримки.

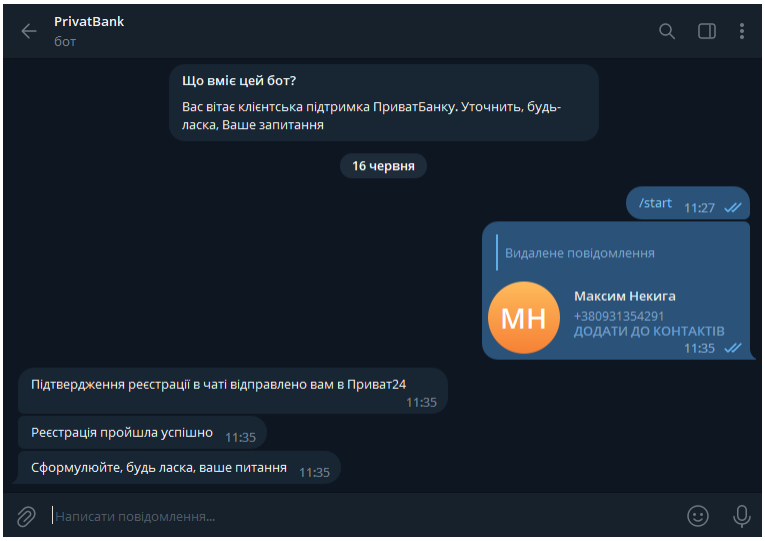


Рисунок 1.3 – Чат-бот технічної підтримки ПриватБанк

- Чат-бот Зоряна від Київстар – віртуальний асистент, який підтримує різні месенджери та використовує технології машинного навчання (рис. 1.4). Головною перевагою цього програмного рішення є використання машинного навчання, що дозволяє чат-боту розпізнавати значення тексту клієнтів і давати точні та релевантні відповіді.

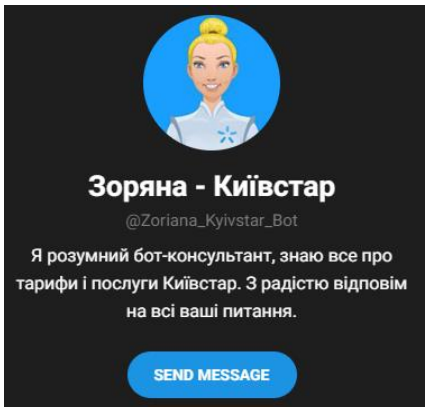


Рисунок 1.4 – Віртуальний асистент “Зоряна” від Київстар

Загалом, Telegram API надає широкі можливості для створення потужних та функціональних чат-ботів для технічної підтримки. Завдяки цьому він є найпопулярнішою платформою для розробки чат-ботів. Розробники можуть ефективно використовувати ці можливості для створення інтерактивних, зручних та ефективних рішень для забезпечення технічної підтримки користувачів.

2. ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ЧАТ-БОТУ ТЕХНІЧНОЇ ПІДТРИМКИ

2.1. Функціональні вимоги

Функціональні вимоги до чат-боту технічної підтримки визначають необхідні функції та можливості, які має мати розроблене рішення. Ці вимоги допоможуть забезпечити ефективну та якісну взаємодію між користувачами та чат-ботом, а також забезпечити виконання основних цілей розробки.

На основі аналізу потреб користувачів та орієнтації на технічну підтримку, були визначені наступні функціональні вимоги:

- **Приймання та обробка запитів**: Чат-бот повинен приймати запити користувачів у форматі тексту та передавати їх відділу КІТ для обробки. Для забезпечення точної відповіді бот повинен виявляти ключові слова та поняття. Крім цього він повинен розпізнавати та обробляти медіафайли за наявності.
- **Надання вказівок та рішень**: Чат-бот повинен мати можливість надавати користувачам зрозумілі та конкретні відповіді на їхні запитання. Крім того, чат-бот може надавати посилання на документацію, рекомендації або проводити діагностику проблеми та запропоновувати відповідні кроки.
- **Доступ до контексту користувача**: Чат-бот повинен мати здатність зберігати та використовувати контекст розмови з користувачем. Це дозволить забезпечити послідовну та зручну взаємодію, де чат-бот може розуміти дії користувача та надавати належні відповіді.
- **Інтеграція з існуючими системами**: Чат-бот повинен мати можливість інтегруватися з існуючими системами моніторингу чи керування інфраструктурою, базами даних тощо. Це дозволить чат-

боту отримувати необхідну інформацію для точних та персоналізованих відповідей.

Ці функціональні вимоги є основою для розробки та реалізації чат-боту технічної підтримки. У наступних розділах ми детальніше розглянемо архітектуру та технології, які використовуються для реалізації цих вимог.

2.2. Нефункціональні вимоги

Нефункціональні вимоги визначають якості та властивості системи, які впливають на ефективність та досвід користувачів. З метою забезпечення ефективної та надійної роботи чат-боту технічної підтримки, були визначені наступні нефункціональні вимоги:

- **Продуктивність**: Відгук чат-боту повинен бути миттєвим або відбуватися у межах прийнятного часу відповідно до типу запиту. Чат-бот повинен мати здатність обробляти одночасно багато запитів і підтримувати високу швидкодію відповідей.
- **Надійність**: Чат-бот повинен бути стійким до помилок та збоїв. У разі виникнення помилок, система повинна забезпечувати коректну обробку та надання відповідей про помилку. Чат-бот повинен мати можливість відновлення роботи після збоїв та забезпечувати безперебійну доступність для користувачів.
- **Безпека**: Чат-бот повинен дотримуватися високих стандартів безпеки, не зберігати особисту інформацію користувачів та захищати передачу даних по мережі. Система повинна бути стійкою до несанкціонованого доступу та належно забезпечувати контроль доступу до функцій і даних.
- **Масштабованість**: Чат-бот повинен мати здатність масштабуватися, щоб забезпечити ефективну роботу при зростанні кількості користувачів та обсягу запитів. Система повинна бути гнучкою для

впровадження нових функцій та можливостей без впливу на її продуктивність та надійність.

- **Користувальницький інтерфейс**: Інтерфейс чат-боту повинен бути простим та інтуїтивно зрозумілим для користувачів різного рівня технічної підготовки. Система повинна мати здатність використовувати різні медіаформати для відповідей, такі як текст, графіка, аудіо або відео, залежно від потреб користувача.

Нефункціональні вимоги мають велике значення для успішної реалізації та ефективної роботи чат-боту. Для досягнення цих вимог рекомендується використовувати розповсюджені технології та інструменти, такі як Microsoft Azure, Google Cloud, Amazon Web Services та інші, які надають надійні та безпечні рішення для розробки чат-ботів технічної підтримки.

2.3. Проектування архітектури чат-боту

Архітектура чат-боту визначає компоненти системи, їх взаємозв'язок та організацію для досягнення функціональних та нефункціональних вимог. Спроектована архітектура чат-боту повинна використовувати модульну та розширювану модель, яка дозволяє додавати нові функції та компоненти без впливу на існуючу систему. Також важливо враховувати принципи безпеки, надійності та продуктивності при виборі технологій та підходів до реалізації архітектури.

З метою ефективного розподілення ресурсів, підвищення стійкості роботи програми, обмеження доступу та оптимізації написання програмного коду було вирішено організувати проєкт, розділивши його структуру за функціональним призначенням на складові частини (рис. 2.1):

- **Початковий алгоритм підготовки до запуску** – виконує роль початкової точки запуску чат-бота. Саме цей алгоритм виконуватиметься першочергово під час запуску програми.

- **Базові модулі функціонування чат-бота** – включають алгоритм запуску чат-бота, специфікацію можливих сценаріїв розмов з користувачами, кнопки з готовими відповідями, планувальник завдань тощо.
- **Обробники дій користувачів** – обробляють вхідні оновлення: повідомлення, відредаговані повідомлення, публікації в каналах, відредаговані публікації в каналах тощо [8]. До дій користувача відносяться повідомлення, команди та натиски на кнопки.
- **Модулі зв'язку з базами даних**. Для кожної інтегрованої бази даних. повинні бути реалізованими алгоритми ініціалізації зв'язку та моделі користувача, працівника і заявки, які обмінюватимуться інформацією з цією базою даних.
- **Сценарії майстра налаштувань** – описують алгоритми, що надають інструкції для допомоги адміністратору системи в самостійному налаштуванні чат-бота.

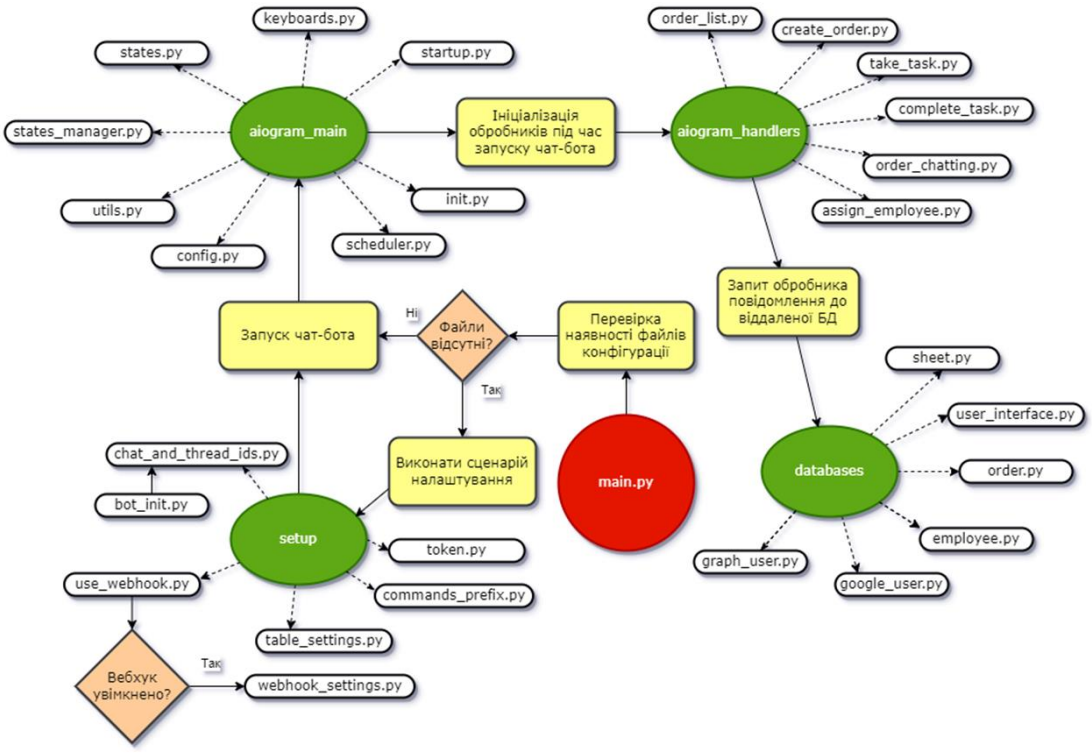


Рисунок 2.1 – Архітектура чат-бота відділу КІТ

3. ПРОЕКТУВАННЯ АЛГОРИТМУ РОБОТИ ЧАТ-БОТУ ТЕХНІЧНОЇ ПІДТРИМКИ

3.1. Розробка боту на основі Telegram API

Перед початком розробки потрібно створити обліковий запис для чат-бота. В Telegram для цього передбачений власний чат-бот “BotFather”. Цей чат-бот дозволяє створювати та керувати наявними чат-ботами. Створення нового бота виконується командою “/newbot” або за допомогою вбудованого меню. Виконавши інструкції чат-бота, ми отримали посилання на чат-бота та токен – секретний ключ доступу, за допомогою якого розроблений застосунок отримає доступ до функцій чат-бота та буде обмінюватись даними з його обліковим записом Telegram. У разі, якщо отриманий токен стане відомим стороннім особам, можна його відкликати та створити новий за допомогою кнопки “Revoke current token” в меню керування створеним чат-ботом (рис. 3.1). Попередній токен API стане недійсним і за допомогою нього більше не можна буде отримати доступ до чат-бота.

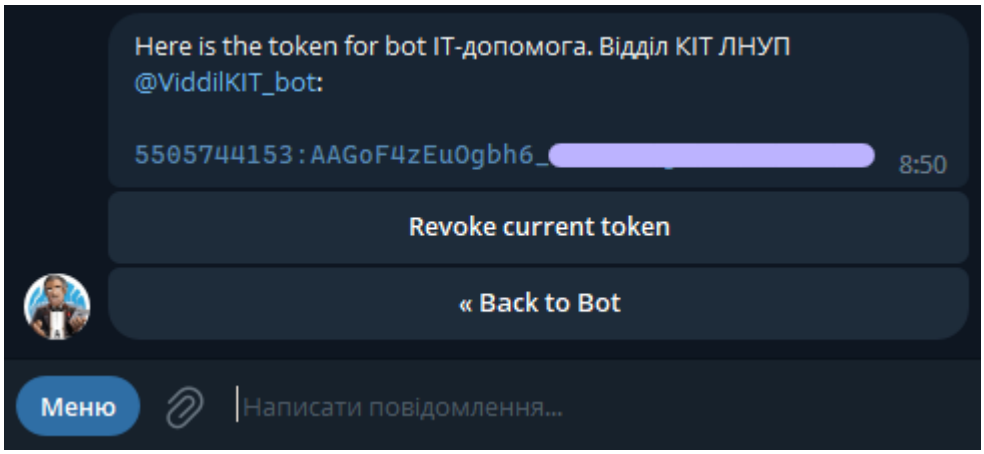


Рисунок 3.1. – Меню токену чат-бота в BotFather

Наступним етапом є вибір протоколу зв’язку чат-бота з сервісами Telegram. Telegram-боти підтримують два протоколи. Обидва мають свої переваги та недоліки, тому можуть використовуватися в різних ситуаціях:

- Протокол **“Long polling”** – найпростіший спосіб підтримки з’єднання з веб-сервером, що надає оновлення в режимі реального часу клієнтам без потреби в постійному надсиланні запитів. Клієнт надсилає запит на сервер і тримає його відкритим доти, доки сервер не отримає нові дані для відправки назад або доки не закінчиться час очікування. Це дозволяє серверу надсилати дані клієнту, як тільки вони стають доступними, а не чекати, поки клієнт ініціює новий запит. Протокол не потребує додаткового налаштування, оскільки сервісам Telegram потрібно лише відправити відповідь на адресу, від якої надійшов запит. Недоліком цього методу є підвищене використання системних ресурсів та додаткове навантаження на мережу [9].
- Протокол **“Webhook”** – невибагливий до ресурсів спосіб сповіщення клієнта сервером про нові події. Протокол особливий тим, що саме серверна частина програми викликає клієнтську частину, а не навпаки. Цей метод є складнішим у реалізації, оскільки вимагає статичної IP-адреси, SSL-сертифікатів та налаштування окремого сервера на клієнтському обладнанні для забезпечення захищеного з’єднання. [10]

Для нашого чат-бота ми вирішили використовувати webhook, оскільки він використовує менше ресурсів обладнання. Telegram дозволяє перевірити статус webhook для чат-бота за посиланням https://api.telegram.org/bot<токен чат-бота>/getwebhookinfo [11]. Якщо webhook не налаштовано, відповідь сервера Telegram матиме вигляд, зображений на рисунку 3.2.

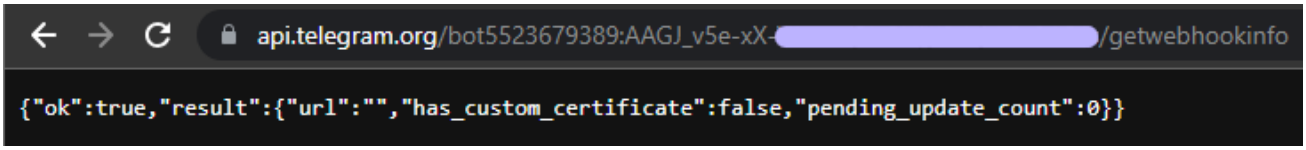


Рисунок 3.2. – Порожня відповідь на запит про дані webhook

Якщо webhook налаштовано коректно, сервіси Telegram повернуть відповідь, зображену на рисунку 3.3. Текст відповіді міститиме інформацію про сертифікати

захищеного з’єднання, адресу та порт веб-сервера webhook чат-бота та кількість оновлень, які ще не отримані чат-ботом.

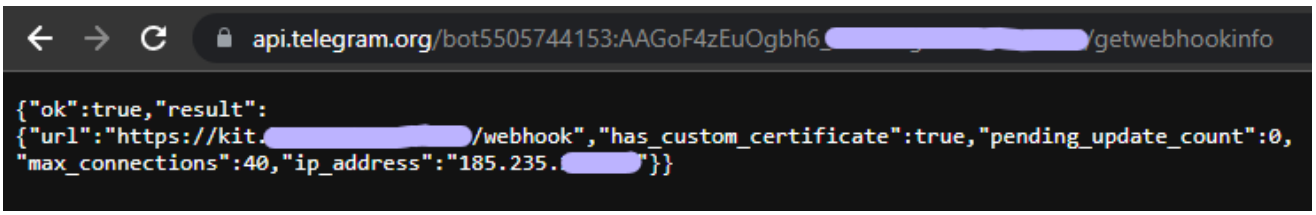


Рисунок 3.3. – Дані та статус активного webhook-сервера

Параметри налаштованого webhook використовуються при розробці алгоритму чат-бота. Алгоритм роботи чат-боту визначатиме послідовність кроків для обробки запитів користувачів та рішення, які потрібно приймати в залежності від їхніх дій. Крім цього в алгоритмі повинен бути передбачений додатковий функціонал для забезпечення коректної роботи чат-бота, звітування про помилки, збору статистики, автоматизованого налаштування системи та інші процеси, які не вимагають дій користувача.

В алгоритмі створення заявок на технічну допомогу в деяких випадках крім збору інформації також буде задіяне надання підказок для самостійного вирішення проблеми замовником. Це дозволить частково автоматизувати процес надання допомоги та значно зменшить кількість звернень і навантаження на працівників відділу ІТ.

Однією з головних вимог до алгоритму є багатопотоковість – властивість застосунку, що полягає в тому, що процес може складатися з кількох потоків, що виконуються паралельно [12]. Багатопотоковість дозволить чат-боту працювати одночасно з багатьма користувачами та не продовжувати спілкування з поточною особою, якщо інший користувач також розпочав роботу з ботом. Крім цього в алгоритм впроваджуються інструменти асинхронного виконання. Асинхронне програмування – це вид паралельного програмування, в якому якась одиниця роботи може виконуватися окремо від основного потоку виконання програми. Коли робота завершується, основний потік отримує повідомлення про завершення

робочого потоку або про помилку. У такого підходу є безліч переваг, таких як підвищення продуктивності застосунків та підвищення швидкості відгуку [13]. Асинхронне виконання та багатопотоковість дозволять чат-боту працювати окремо з кожним користувачем в ізольованому середовищі виконання, що збільшить загальну стабільність. На рис. 3.4 зображено блок-схему загального алгоритму створення заявки користувачем, який може виконуватися паралельно в багатьох потоках, не перериваючи роботу програми та дозволяючи новим користувачам розпочинати роботу з чат-ботом.

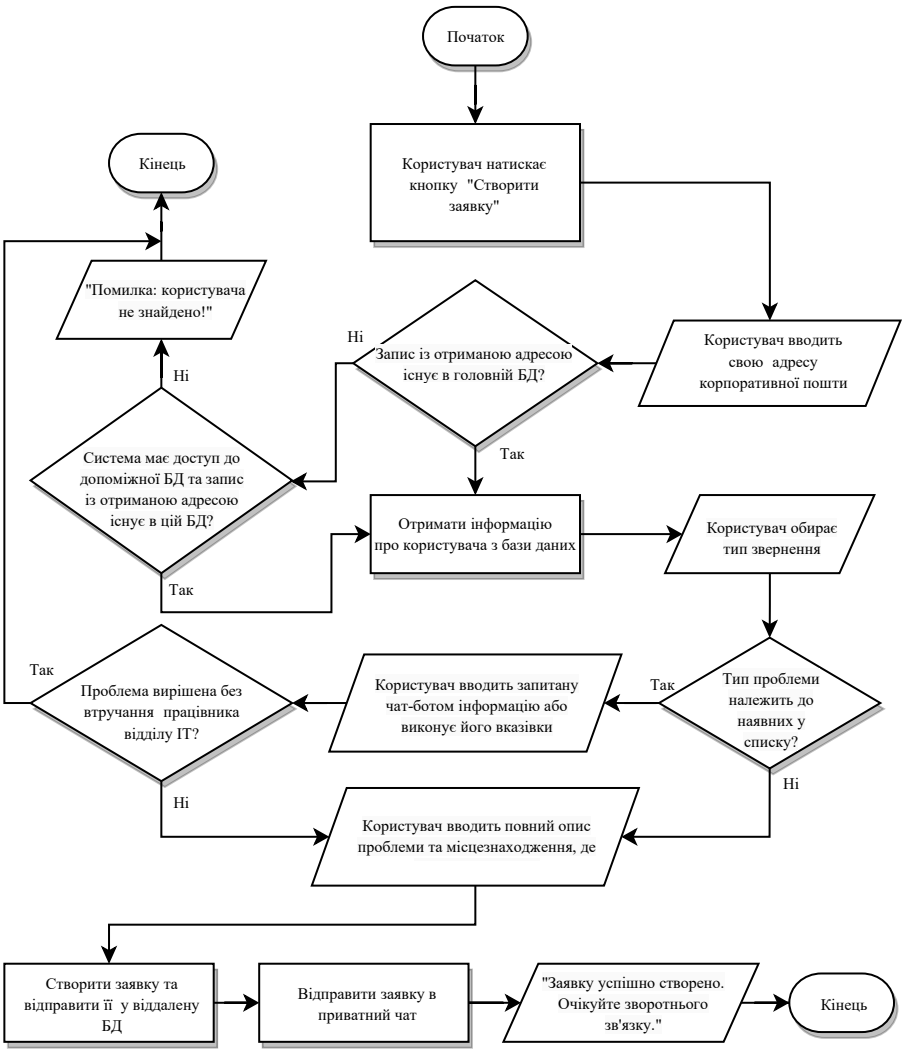


Рисунок 3.4 – Блок-схема алгоритму створення заявки

З метою ефективного розподілення ресурсів, підвищення стійкості роботи програми та оптимізації написання програмного коду було вирішено організувати

проєкт, розділивши файли по функціональному призначенню на такі складові частини:

- Директорія **aiogram_handlers** – набір усіх обробників дій користувачів, включаючи повідомлення, команди та натиски на кнопки. Усі модулі у цій директорії містять функцію setup(), що ініціалізує обробник. Щоб задіяти усі обробники під час запуску чат-бота, виконується імпорт файлу init.py з цієї директорії. Імпортований init.py автоматично виконує ітерацію вказаного списку модулів: для кожного елементу списку за допомогою вбудованої в Python функції exec() імпортує модуль та виконує його функцію setup() (рис. 3.5);

Рисунок 3.5 – Програмний код алгоритму ініціалізації усіх обробників

- Директорія **aiogram_main** – містить модулі запуску та забезпечення базового функціонування чат-бота, серед яких планувальник завдань для подій, які повинні виконуватись регулярно; набори кнопок для готових відповідей; менеджер станів, який автоматично скасовує активні дії з користувачами, якщо вони бездіяльні протягом тривалого часу тощо;
- Директорія **databases** – містить інструменти для обробки інформації та забезпечення зв’язку з віддаленими базами даних, такими як таблиця Google Sheets та база даних організації в доменному імені Microsoft;

- Директорія **setup** – збірка сценаріїв майстра налаштувань, що виконуються у випадку часткової або повної відсутності параметрів конфігурації.

Також кореневий каталог проєкту містить файли інтерпретатора та сторонніх бібліотек:

- Файл **.gitignore**, з якого система керування версіями Git отримує список файлів та папок, які не слід включати до проекту, що дозволяє запобігти випадковому вкладенню до проєкту тимчасових файлів чи об’єктів, які містять чутливу інформацію (рис. 3.6);

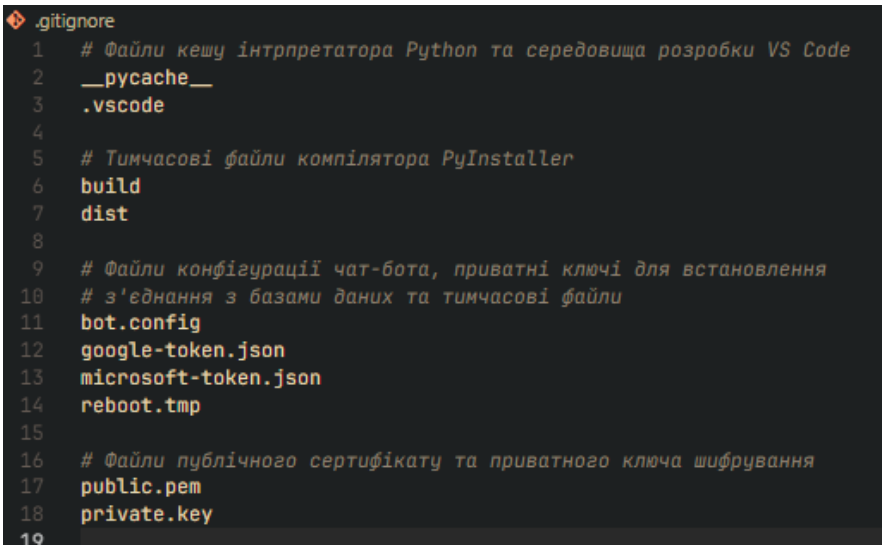


Рисунок 3.6 – Список ігнорованих об’єктів у файлі .gitignore

- Файл **main.spec**, що описує назву, параметри готового застосунку (рис. 3.7, а) та список бібліотек, що повинні бути включені в цей застосунок для коректної роботи (рис. 3.7, б); цей файл використовується бібліотекою PyInstaller для компіляції вихідного коду в готову програму;
- Файл **main.py**, який являє собою головний виконавчий файл, що здійснює початкову перевірку відповідності вимогам для запуску, при потребі запускає майстра налаштування та, в кінцевому результаті, ініціює запуск чат-бота шляхом імпортування модуля aiogram_main.startup. Блок-схему алгоритму роботи файлу main.py зображено на рис. 3.8;

```
exe = EXE(
    pyz,
    a.scripts,
    a.binaries,
    a.zipfiles,
    a.datas,
    [],
    name='botExecutable', # Назва файлу застосунку
    debug=False, # Режим налагодження неполадок
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
    upx_exclude=[],
    runtime_tmpdir=None,
    console=True,
    disable_windowed_traceback=False,
    argv_emulation=False,
    target_arch=None,
    codesign_identity=None,
    entitlements_file=None,
)
```

```
# Список файлів та директорій,
# які потрібно імпортувати в пакунок проекту:
datas += [
    ('./aiogram_main', './aiogram_main'),
    ('./aiogram_handlers', './aiogram_handlers'),
    ('./databases', './databases'),
    ('./setup', './setup'),
]

# Список бібліотек, які потрібні для роботи чат-бота
# та які не були автоматично включені у пакунок:
hiddenimports = [
    "aiogram",
    "aiogram.contrib.fsm_storage",
    "aiogram.contrib.fsm_storage.memory",
    "aiohttp",
    "asyncio",
    "apiclient",
    "apscheduler",
    "bs4"
]
```

а) **б)**

Рисунок 3.7 – Назва, параметри (а) і вкладені бібліотеки (б) застосунку в файлі .gitignore

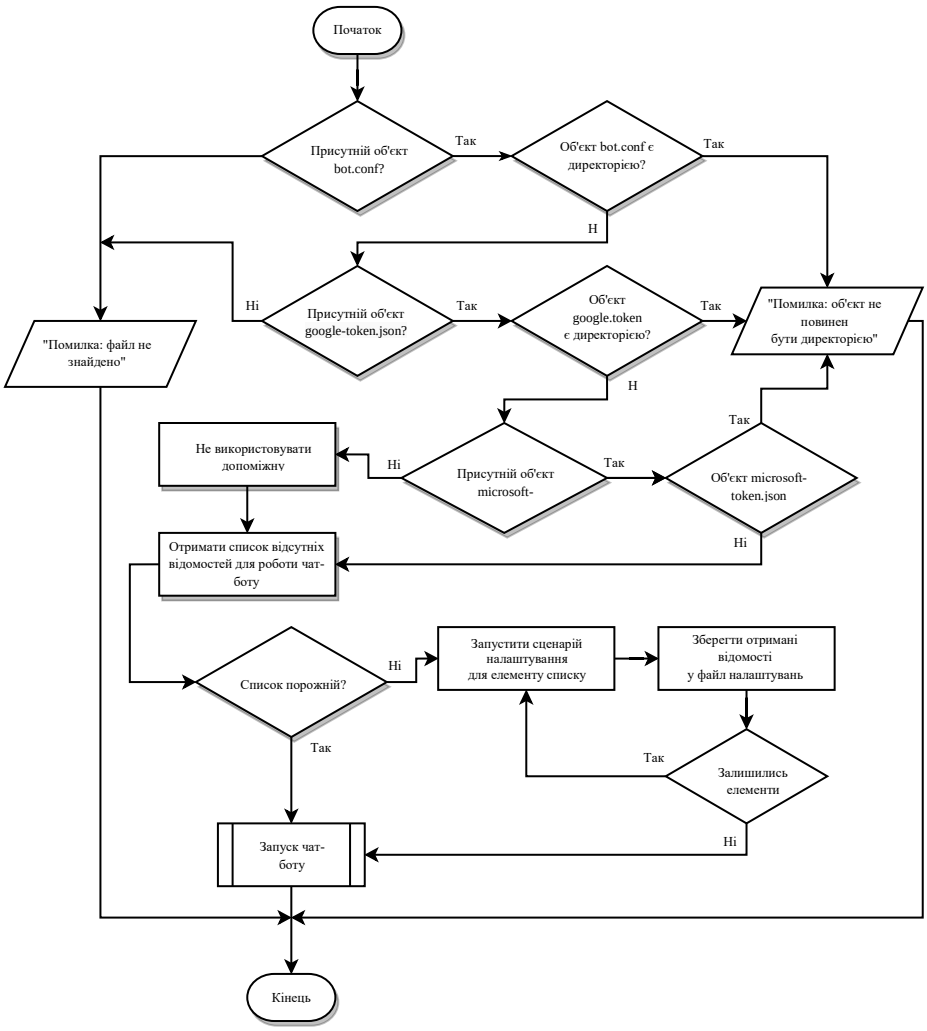


Рисунок 3.8 – Блок-схема алгоритму роботи файлу main.py

- Порожній файл **__main__.py**, що дозволяє запустити виконання програми з командного рядка, вказавши в якості цільового об’єкта кореневий каталог проєкту;
- Порожній файл **__init__.py**, який позначає поточну директорію як каталог з модулями, що дозволяє інтерпретатору Python бачити та імпортувати файли з цієї директорії.

Директорія **aiogram_main** містить такі елементи:

- **keyboards.py** – імпортується в init.py, містить список “клавіатур” – об’єктів, що описують готові відповіді на повідомлення у вигляді кнопок під повідомленнями та під рядком введення повідомлення;
- **states.py** – імпортується в init.py, містить список “станів” – попередньо визначених можливих етапів, яких може досягати користувач при спілкуванні з чат-ботом;
- **init.py** – створює екземпляри класів Bot та Dispatcher з бібліотеки aiogram для подальшого їх запуску, імпортує модулі keyboards.py та states.py (рис. 3.9);

aiogram_main > init.py > ...

```python
import asyncio
import logging

from aiogram import Bot, Dispatcher
from aiogram.contrib.fsm_storage.memory import MemoryStorage

from aiogram_main.config import Config

CONFIG = Config()


logging.basicConfig(level=logging.INFO)
log = logging.getLogger("broadcast")

# Ініціалізація бота
storage = MemoryStorage()
loop = asyncio.get_event_loop()
bot = Bot(token=CONFIG.token, loop=loop)
dp = Dispatcher(bot, storage=storage)
logging.getLogger("schedule").propagate = False

from .states import *
from .keyboards import *
```

Рис. 3.9 – Програмний код алгоритму роботи файлу init.py

- **scheduler.py** – імпортується в startup.py, являє собою планувальник завдань та список регулярних та нерегулярних завдань, наприклад надсилання списку незавершених заявок у чат;
- **states_manager.py** – здійснює моніторинг активності користувачів та автоматично завершує роботу з неактивними користувачами; блок-схему алгоритму роботи файлу states_manager.py зображено на рисунку 3.10;

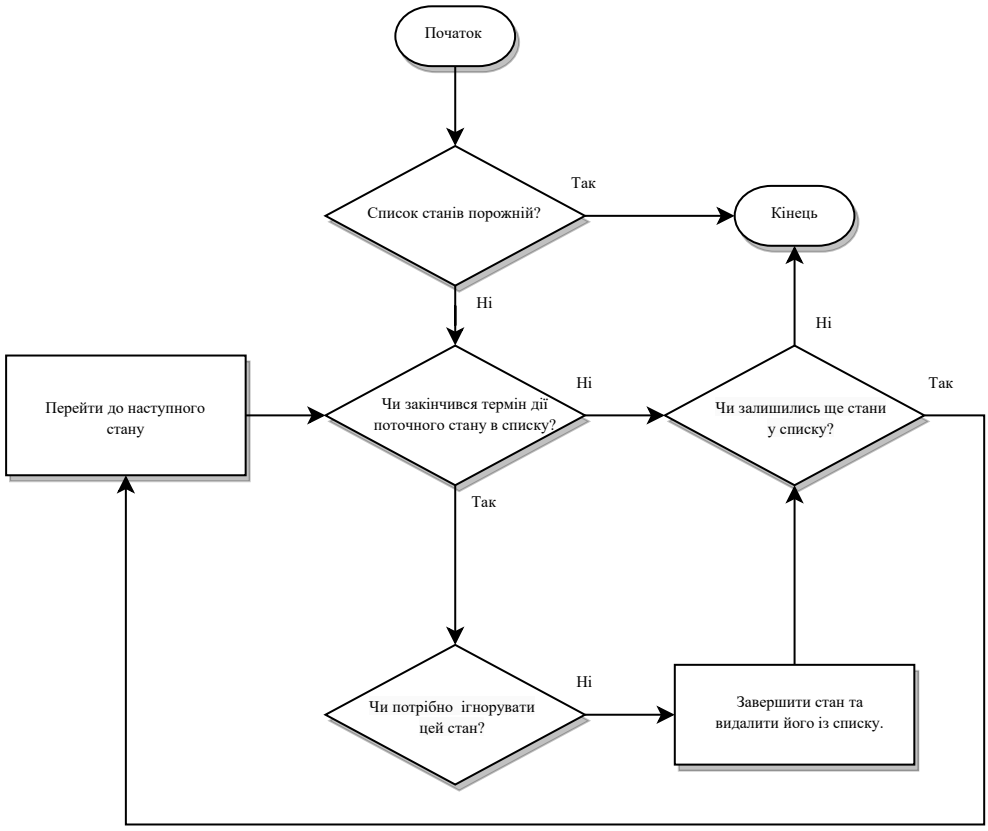


Рисунок 3.10 – Блок-схема алгоритму роботи файлу states_manager.py

- **startup.py** – використовує створені в модулі init.py екземпляри та безпосередньо запускає чат-бота;
- **utils.py** – містить додаткові інструменти, що можуть знадобитися під час роботи чат-бота, такі як відправлення звітів про помилки та аналітики, форматування тексту заявок тощо.

Директорія **aiogram_handlers** містить наступні файли сценаріїв обробки дій користувачів:

1. **basic_actions.py** – сценарії обробки базових дій, які є простими в реалізації та які не є доцільно поміщати в окремі файли через малий об’єм програмного коду. Сюди входять:

   1.1. Обробник кнопок “Скасувати”, “Скасувати заявку” та команди /cancel. Усі три елементи виконують один і той же метод cancel_handler();

   1.2. Обробник кнопки “Видалити це повідомлення”;

   1.3. Кнопка “Будь ласка, зачекайте...”, яка виконує лише функцію підказки;

   1.4. Обробники, пов’язані з меню перезавантаження чат-бота або обладнання, на якому він працює. Доступ до меню перезавантаження можуть отримати лише довірені адміністратори. При введенні працівником IT-відділу команди /reboot в чаті працівників відбувається перевірка привілеїв особи та відкривається меню перезавантаження, якщо перевірку пройдено успішно. Меню залишається доступним впродовж 30 секунд та автоматично закривається, якщо не обрано жодної дії.
2. **add_employee.py** – додавання нового працівника IT-відділу до бази даних для надання доступу до керування заявками. Новий працівник повинен ввести команду /addemployee в чаті відділу, після чого відправити чат-боту своє ім’я та прізвище. Ця процедура вирішує одразу кілька проблем:

   2.1. Telegram дозволяє користувачам встановлювати довільне ім’я свого профілю. Працівники можуть встановити ім’я, по якому його буде важко ідентифікувати та яке не відповідатиме вимогам ведення обліку робіт. Реєстрація працівника дозволяє використовувати його реальне ім’я для ведення обліку;

   2.2. Зловмисники можуть отримати доступ до чату працівників та керувати заявками без обмежень. Реєстрація дозволяє цього уникнути;

   2.3. З’являється можливість виконувати дії із працівниками, наприклад, призначати конкретну особу як виконавця заявки.

3. **assign_employee.py** – алгоритм присвоєння виконавця заявці. Працівник може натиснути на кнопку “Призначити виконавця” під заявкою та вибрати іншого працівника із списку, що відкрився. Присвоєний виконавець отримує сповіщення про доручену йому заявку, в якому міститься текст заявки та посилання на оригінальне повідомлення. Це дозволяє ефективно розподіляти обов’язки між працівниками та сповіщати їх про завдання;
4. **take_task.py** – алгоритм позначення заявки як розпочатої з внесенням імені виконавця у базу даних;
5. **complete_task.py** – позначення заявки як виконаної. Виконавець підтверджує виконання завдання натиснувши на відповідну кнопку та вводить опис виконаних робіт. Отримана інформація зберігається в таблиці обліку робіт;
6. **create_order.py** – ініціювання процесу створення заявки. Тут обробляється отримана від користувача електронна пошта: введені дані звіряються з усіма наявними базами даних користувачів. Якщо є збіг, отримана від бази даних інформація про замовника використовується для подальшого процесу створення заявки. Ідентифікувавши користувача, алгоритм запитує тип звернення, запропонувавши користувачеві вибрати відповідь, натиснувши на потрібну кнопку. Щойно замовник обере тип звернення, запуститься виконання одного із сценаріїв problem_*.py відповідно до обраного типу;
7. **multiple_file_messages.py** – алгоритм обробки та пересилання медіафайлів, надісланих користувачами. Отримані від замовників файли зберігаються в гілці “Вкладення” чату працівників;
8. **order_chatting.py** – спілкування між замовниками та працівниками IT-відділу стосовно заявок. Чат-бот виступає посередником та пересилає повідомлення між замовником та чатом працівників. Будь-який працівник може переглядати історію спілкування за допомогою кнопки “Історія

спілкування” під заявкою та надсилати повідомлення кнопкою “Написати замовнику”. Замовник отримає повідомлення у приватному чаті із чат-ботом та зможе відправити відповідь за допомогою кнопки “Відповісти на це повідомлення”;

9. **print_order.py** – алгоритм виведення повідомлення із списком невиконаних заявок. Регулярно викликається планувальником завдань за встановленим графіком. Алгоритм отримує від бази даних список усіх заявок, відбирає із нього лише незавершені, будує повідомлення-список та відправляє його в гілку “Список” чату;
10. **problem_computers.py** – сценарій дій для створення заявки, яка стосується неполадок з комп’ютерним обладнанням. В процесі виконання сценарію в користувача запитуються відомості про комп’ютерне обладнання, такі як назва, модель, характеристики обладнання, серійний чи інвентарний номер, MAC-адреса та інша інформація. У випадку отримання всіх даних запускається виконання алгоритму problem_other.py;
11. **problem_no_internet.py** – створення заявки стосовно проблем з інтернетом. Користувачу пропонуються різноманітні підказки, які можуть допомогти самостійно вирішити неполадки з підключенням, наприклад, перевірити підключення Ethernet-кабеля чи мережі Wi-Fi тощо. Якщо цього виявилось не достатньо, запускається виконання алгоритму problem_other.py;
12. **problem_printers.py** – створення заявки щодо обслуговування друкарок. Це включає ремонт друкарського обладнання та заправку картриджів. У користувача запитуються модель, серійний номер друкарки чи картриджа та місцезнаходження обладнання. У випадку отримання всіх даних запускається виконання алгоритму problem_other.py;

13. **problem_other.py** – кінцевий алгоритм процесу створення заявки, що викликається обробниками усіх інших типів заявок для завершення створення заявки. Запитує в користувача опис суті проблеми, локацію звернення та відправляє заявку в чат працівників. Якщо замовник обрав тип звернення “Інше”, розпочинається безпосереднє виконання цього обробника, оскільки тип звернення “Інше” не передбачає збір інформації крім причини та локації звернення;
14. **start_command.py** – обробник команди /start, який надсилає користувачеві вітальне повідомлення та кнопку “Створити заявку” при початку спілкування.

Директорія **databases** містить інструменти для ефективної роботи та обміну інформацією з віддаленими сховищами даних, ідентифікації користувачів, збереження та отримання створених заявок, керування працівниками IT-відділу тощо. Вміст директорії включає такі файли:

1. **sheet.py** – встановлює з’єднання з таблицею Google Sheets та описує клас SheetLogger, що відправляє у таблицю звіти про помилки та іншу корисну для налагодження інформацію.
2. **employee.py** – абстрактна модель, яка описує будову та поведінку об’єктів типу Employee, що служать для ідентифікації працівників відділу IT. Даний модуль керує таблицею працівників та виконує перевірку об’єкту спілкування на наявність привілеїв працівника. Це дозволяє обмежити доступ стороннім особам, навіть, якщо вони отримали доступ до приватного чату із заявками.
3. **user_interface.py** – містить модель, що описує базові вимоги до об’єкту користувача, яким повинна відповідати реалізація зв’язку з базою даних. До цих вимог належать:

   3.1. параметри **_tg_user** та **corp_email**, що пов’язують унікальний ідентифікатор користувача Telegram з ідентифікатором у базі даних;

3.2. необов’язкова функція або змінна **employee_name**, що повертає ім’я зареєстрованого працівника IT-відділу, якщо поточний користувач одночасно є і працівником та якщо поточна база даних використовується для зберігання інформації про працівників;

3.3. словник **data_dict**, у якому зберігатимуться усі дані користувача у вигляді пари “ключ: значення”;

3.4. функція **pull()**, яка вивантажуватиме з віддаленої бази даних усю наявну інформацію про працівника та зберігатиме її у словник data_dict.

3.5. необов’язкова функція **build_tg_url()**, що повертає значення змінної employee_name, якщо користувач є працівником. В іншому випадку повертає його ім’я Telegram.

Цей інтерфейс не виконує явної функції, крім полегшення та допомоги розробникам в інтеграції баз даних користувачів у систему. Створені на цій моделі реалізації можуть використовувати додатковий функціонал, не описаний в списку вище;

4. **google_user.py** – готова модель, реалізована на основі інтерфейсу user_interface, яка використовує зв’язок з попередньо налаштованою онлайн-таблицею Google Sheets за допомогою Google Sheets API для збереження та обміну інформацією про користувачів, ведення обліку заявок, керування працівниками, збереження інформації для налагодження тощо;
5. **graph_user.py** – ще одна модель, що унаслідувана від інтерфейсу user_interface. Використовує інструменти Microsoft Graph API для зв’язку з віртуальним простором організації або компанії для отримання інформації про користувачів, якщо її не було знайдено за допомогою модуля google_user;
6. **order.py** – описує абстрактну модель заявки. Містить вбудовані методи для роботи з віддаленою базою даних, може зберігати та зчитувати себе

із таблиці за допомогою модуля sheet. Клас заявки описує такі властивості та методи:

6.1. **id** – номер заявки;

6.2. **position** – порядковий номер заявки в таблиці;

6.3. **creator_id** – Telegram-ідентифікатор замовника;

6.4. **status_index** – індекс статусу виконання заявки (значення від 0 до 3);

6.5. Функція **get_id()** – визначає ідентифікатор заявки з відомого порядкового номера;

6.6. Функція **get_position()** – діє протилежно get_id(): визначає порядковий номер заявки з відомого ідентифікатора;

6.7. Функція **get_creator_id()** – повертає Telegram-ідентифікатор замовника з таблиці заявок;

6.8. Функція **pull()** – виконує запит до бази даних та отримує від неї усі актуальні відомості про заявку, повертає True або False в залежності від успішності;

6.9. Функція **start()** – змінює стан заявки на розпочату та вносить до її відомостей ім’я працівника час початку виконання;

6.10. Функція **complete()** – змінює стан заявки на завершену, вносить до її відомостей опис виконаних робіт, ім’я виконавця і час завершення виконання;

6.11. Функція **upload()** – відправляє усі відомості про заявку в таблицю. Якщо заявка нова – додає запис у таблицю, якщо ні – усі клітинки заявки перезаписуються;

6.12. Службова функція **get_new_order_pos()** – повертає незайнятий порядковий номер для додавання нової заявки в таблицю;

6.13. Службова функція **get_new_order_id()** – аналогічно get_new_order_pos() повертає ідентифікатор для нової заявки;

6.14. Службова функція **get_current_timestamp()** – повертає поточні дату та час для позначення часу виконання подій, таких як створення заявки чи зміна її статусу;

6.15. Функція класу **get_orders_columns()** – повертає вказані стовпці із таблиці заявок, використовується у випадках, коли потрібно обробити інформацію великої кількості заявок, наприклад, надіслати в чат список заявок або здійснити пошук заявки із списку по ідентифікатору;

6.16. Функція класу **get_unfinished_list()** – повертає список невиконаних заявок, використовуючи та відбираючи заявки по критеріям із переліку, отриманого від функції get_orders_columns();

6.17. Службові методи **exists_id()** та **exists_position()** – здійснюють перевірку на те, чи існує заявка з вказаним ідентифікатором чи порядковим номером. Повертають True або False в залежності від результату.

Директорія **setup** містить сценарії майстра налаштування чат-бота, що виконуються автоматично при запуску програмного застосунку, якщо не виявлено тих чи інших налаштувань у файлах конфігурації. Сценарії з цього каталогу виконуються по черзі, якщо потрібно налаштувати одразу кілька параметрів. Налаштування виконується шляхом виконання адміністратором дій згідно виданої сценарієм інструкції та введення отриманих даних у вікно налаштування. Деякі сценарії запускають та використовують функціонал чат-бота для налаштування чату працівників. Каталог містить такі сценарії налаштування:

1. **token.py** – надає інструкцію із створення та налаштування облікового запису чат-бота в месенджері Telegram. Виконавши вказівки, адміністратор отримає приватний токен (ключ), що дозволяє керувати щойно створеним обліковим записом чат-бота. Цей токен потрібно вставити у вікно додатку. Додаток спробує під’єднатись до чат-бота за допомогою отриманого токену і збереже токен у файл налаштувань, якщо з’єднання успішне;

2. **commands_prefix.py** – запитує в адміністратора бажаний префікс для команд у чаті. За замовчуванням встановлюється префікс “/”, але може бути обраний інший, якщо у чаті присутній інший бот, що використовує цей префікс.
3. **use_webhook.py** – ініціює процедуру вибору методу роботи чат-бота. Адміністратор повинен вибрати один із двох режимів зв’язку чат-бота з сервісами Telegram: “Long polling” або “Webhook”. Метод “Long polling” використовується за замовчуванням, якщо адміністратор пропустив вибір.
4. **webhook_settings.py** – якщо було обрано режим роботи “Webhook”, надає адміністратору системи коротку інструкцію з налаштування локального сервера та отримання SSL-сертифікатів. Виконавши налаштування, адміністратор повинен ввести дані сервера в вікно застосунку.
5. **chat_and_thread_ids.py** – використовує налаштований раніше обліковий запис чат-бота в Telegram для налаштування чату працівників. Адміністратор системи повинен додати чат-бота до приватної групи, увімкнути у цій групі підтримку гілок та надати чат-боту права адміністратора. Після цього чат-бот автоматично закінчить налаштування, перейменувавши головну гілку на “Заявки” та створивши гілки “Вкладення”, “Звіти”, “Сповіщення” та “Список”. Ідентифікатори усіх гілок зберігаються у файл налаштування чат-бота для правильного надсилання повідомлень у відповідні гілки в майбутньому.
6. **bot_init.py** – містить мінімальний набір логіки чат-бота, який використовується сценарієм chat_and_thread_ids.py для налаштування приватного чату працівників у Telegram.
7. **table_settings.py** – надає інструкцію для самостійного налаштування зв’язку з онлайн-таблицею Google Sheets. Адміністратор системи повинен:
   7.1. Створити проєкт в системі Google Cloud;

7.2. Додати до проєкту підтримку інструменту Google Sheets API;

7.3. Створити в проєкті службовий обліковий запис для чат-бота;

7.4. Створити приватний ключ доступу до службового облікового запису у вигляді файлу в форматі JSON та завантажити його;

7.5. Помістити завантажений файл в одну директорію з застосунком чат-бота;

7.6. Самостійно створити порожню онлайн-таблицю Google Sheets та надати до неї доступ чат-боту;

7.7. Надати чат-боту посилання на створену таблицю та дочекатись автоматичного налаштування;

7.8. Виконавши всі сценарії та застосувавши усі отримані від адміністратора параметри, майстер налаштування автоматично запустить чат-бота.

Поділ структури чат-бота на функціональні складові значно спростив навігацію в коді та подальшу розробку. Повна ієрархічна структура файлів та папок проєкту зображена на рисунку 3.11.

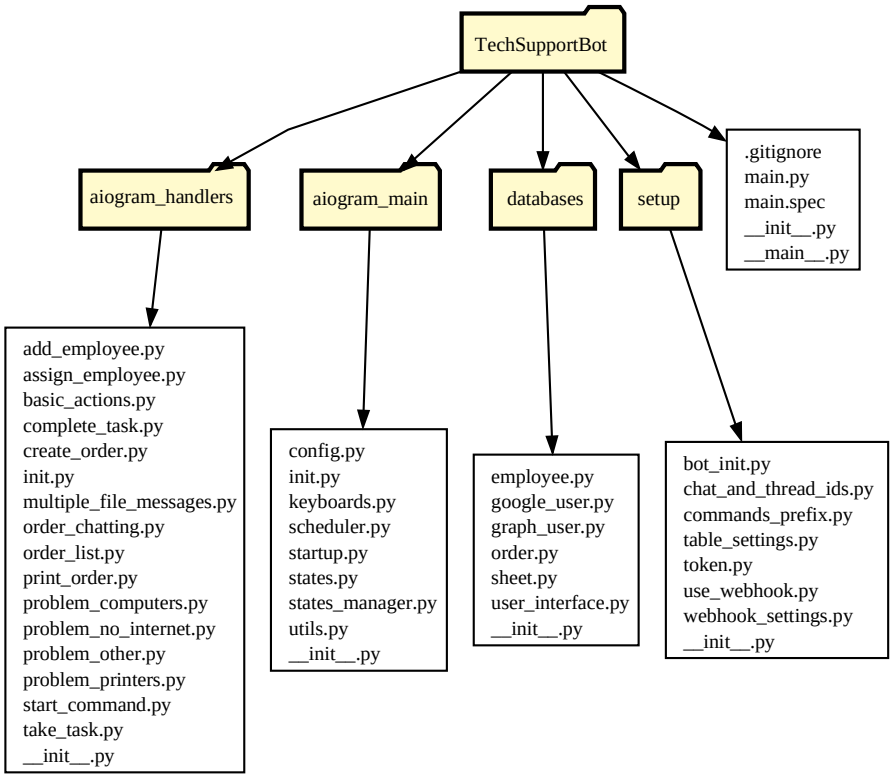


Рисунок 3.11 – Ієрархічна структура проєкту чат-боту

3.2. Розробка бази даних для зберігання інформації

Для обробки та зберігання інформації чат-боту потрібна база даних. База даних – це засіб збирання та впорядкування інформації. Бази даних можуть зберігати відомості про людей, продукти, замовлення або будь-що інше [14]. База даних повинна бути захищеною, доступною цілодобово, зручною у використанні та мати можливість перегляду користувачами. Опираючись на ці вимоги, було вирішено використовувати сервіси онлайн-таблиць, оскільки вони зберігають дані віддалено, надають можливість переглядати та редагувати інформацію в зручному та зрозумілому поданні. Крім цього онлайн таблиці підтримують формули та виконують їх обчислення на стороні сервера, що дозволить зняти навантаження з системи при обробці великих масивів даних. На відміну від більш традиційних реалізацій баз даних на архітектурі «клієнт-сервер», таких як MySQL, SQLite чи Firebird, онлайн-таблиця не вимагає специфічного програмного забезпечення для перегляду та редагування даних, що є перевагою при веденні обліку робіт в установах чи організаціях.

Найбільш популярними сервісами онлайн-таблиць є Google Sheets та Microsoft Excel в середовищі Microsoft 365. Впродовж тривалого терміну використання Google Sheets показав кращу стабільність роботи, швидкість виконання запитів та загальну продуктивність сервісу, ніж аналогічний продукт від Microsoft. Тому в якості основної бази даних було використано сервіс Google Sheets та офіційну Python-бібліотеку від Google google-api-python-client в якості програмного забезпечення для обміну інформацією з таблицею Google Sheets.

Онлайн-таблиця складатиметься з чотирьох листів:

1. Orders – Список усіх отриманих заявок. Кожна заявка отримує порядковий номер та вставляється в перший вільний рядок таблиці. Порядковий номер складається з поточного року та чотирьох цифр порядкового номеру заявки у цьому році. Наприклад, якщо заявка є 147-ю в списку заявок за 2023 рік, вона буде мати номер 230147. З настанням нового 2024 року, префікс порядкового

номеру збільшиться на один та лічильник заявок обнулиться: перша заявка матиме номер 240001.

2. Users – Список усіх користувачів системи. Чат-бот здійснюватиме пошук введеної замовником корпоративної пошти у цьому списку та використовуватиме знайдену інформацію про користувача при створенні заявки.
3. Employees – Список працівників відділу IT та додаткова службова інформація для коректної роботи чат-бота, така як порядковий номер та позиція майбутньої заявки, поточний рік тощо. Список працівників містить імена працівників та ідентифікатори їх облікових записів Telegram.
4. Logs – Список діагностичних повідомлень для налагодження та статистики використання. Якщо при роботі чат-бота виникне помилка, звіт з причиною неполадки відправиться у цей лист. Це значно полегшує процес виправлення неполадок та збирання корисної інформації про роботу чат-бота.

Крім основної БД було використано допоміжну БД у вигляді сховища облікових записів Львівського національного університету природокористування у віртуальному середовищі Microsoft. Оскільки у ЛНУП кожному працівнику та студенту було надано особисті корпоративні облікові записи Microsoft Office 365 для роботи та навчання, віртуальне середовище Microsoft є хорошим рішенням для ідентифікації користувачів. Допоміжна БД дозволить ідентифікувати особу при роботі з системою, якщо запис користувача не був знайдений у головній базі даних.

Для отримання даних з допоміжної БД використовуватиметься Microsoft Office 365 API та вбудована Python-бібліотека requests, що дозволяє надсилати HTTP-запити на вказані інтернет-адреси. Функції requests.get() та requests.post() дозволять надсилати запити на віддалену адресу Microsoft та отримувати відповідь з потрібними даними користувача.

4. ПРАКТИЧНЕ ВИКОРИСТАННЯ ЧАТ-БОТУ ТЕХНІЧНОЇ ПІДТРИМКИ ВІДДІЛУ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ЛЬВІВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ПРИРОДОКОРИСТУВАННЯ ДЛЯ МЕСЕНДЖЕРА TELEGRAM

4.1. Оцінка роботи чат-боту

Для практичного впровадження чат-боту було враховано особливості відділу комп'ютерних інформаційних технологій Львівського національного університету природокористування та потреби його користувачів. Працівники та керівник відділу отримали ефективний інструмент розподілення роботи та керування завданнями. Чат-бот допоміг вирішити наступні проблеми:

- **Звітність роботи:** відділ КІТ щорічно повинен надавати звітність щодо своєї роботи. Працівники повинні були самостійно записувати та заповнювати усі звернення замовників. Чат-бот вирішує цю проблему, значно зменшуючи зусилля для ведення обліку робіт;
- **Нагадування про невиконані завдання:** при великій кількості звернень працівники часто забувають про завдання, які потрібно виконати. Система щоденно нагадує про невиконані заявки, надсилаючи список у чат;
- **Занадто складні для освоєння канали комунікації замовників з відділом:** у минулому відділ КІТ для створення заявок на технічну допомогу використовував Microsoft Forms. Сервіс вимагав авторизації за допомогою особистого корпоративного облікового запису, що дозволило створювати заявки лише довіреним особам. Проте багато користувачів не користувалися своїми корпоративними обліковими записами або не мали до них доступу. Чат-бот використовує простий метод ідентифікації та зрозумілий спосіб спілкування у вигляді листування.

Індивідуальна розробка проєкту для потреб відділу дозволила обробляти різні сценарії та типи звернень замовників. Серед них:

- **Заправка картриджів чи ремонт друкарок.** Передбачається введення користувачем моделі, серійного номеру обладнання та додаткових приміток. оскільки Львівський національний університет природокористування співпрацює з приватною компанією, що надає послуги з обслуговування друкарського обладнання, працівник цієї установи має доступ до чату відділу та обробляє звернення цього типу;
- **Заявки щодо відсутності інтернету.** Алгоритм передбачає надання автоматизованої допомоги шляхом вказівок: користувачу пропонується перевірити з'єднання, перейти за вказаною адресою та спробувати авторизуватися. Якщо вказівки не допомогли, замовник вказує опис проблеми та своє місцезнаходження. Якщо вказівки допомогли вирішити проблему з підключенням – заявка не створюється;
- **Заявки на ремонт комп'ютерного обладнання.** Користувач може звернутися з потребою усунення неполадок чи ремонту комп'ютерної техніки, включно з комп'ютерною периферією.

Якщо причини звернення немає в переліченому списку, користувач повинен обрати тип "Інше", ввести власну причину звернення та місцезнаходження за потреби. Зовнішній вигляд меню вибору звернень зображено на рис. 4.1.

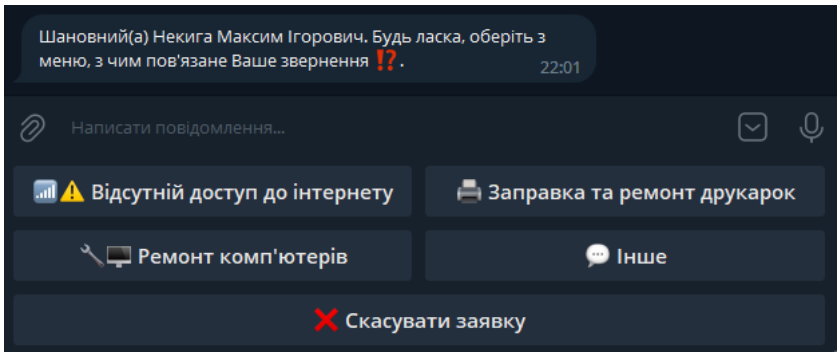


Рисунок 4.1 – Меню вибору причини звернення

Створена замовником заявка автоматично надсилається у чат відділу КІТ (рис. 4.2), а також зберігається в онлайн-таблицю заявок (рис. 4.3). Усі оновлення та події в

автоматичному режимі вносяться чат-ботом у потрібні клітинки заявки за допомогою Google API Client.

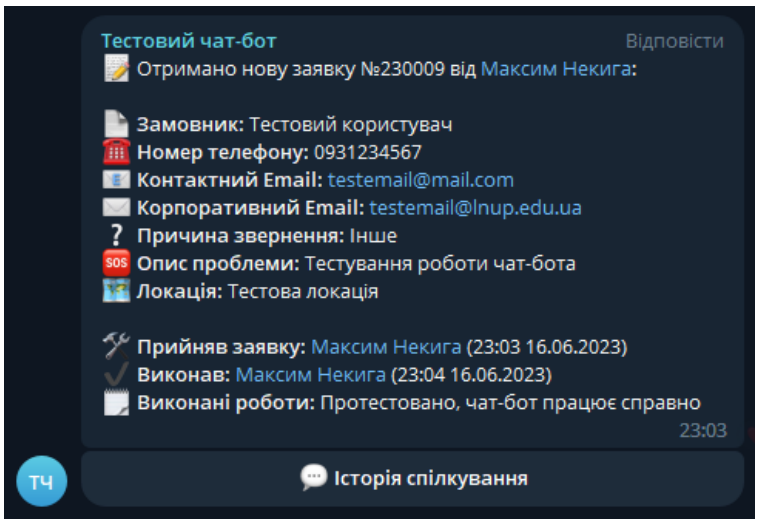
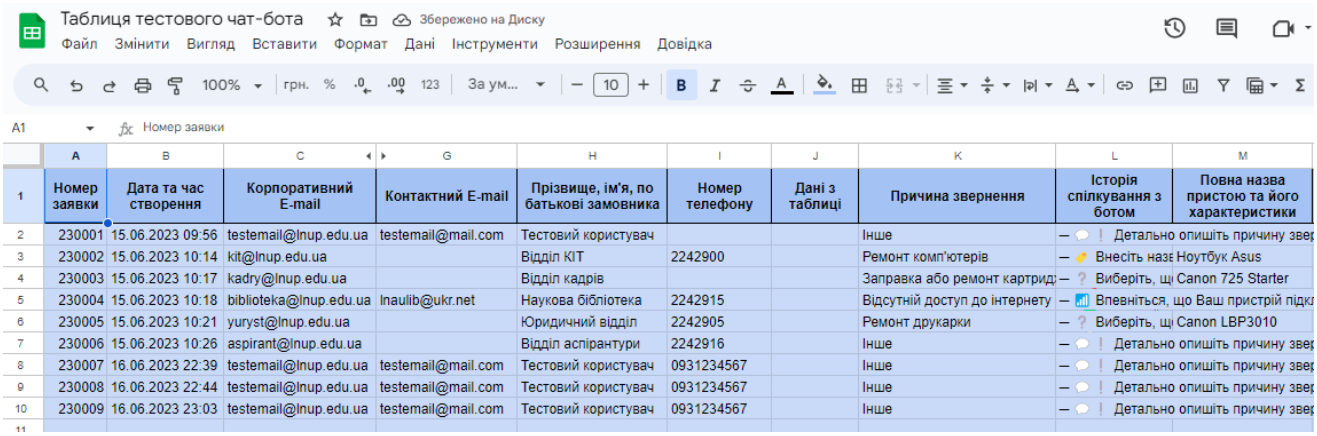


Рисунок 4.2 – Повідомлення заявки у чаті



	A	B	C	G	H	I	J	K	L	M
1	Номер заявки	Дата та час створення	Корпоративний E-mail	Контактний E-mail	Прізвище, ім'я, по батькові замовника	Номер телефону	Дані з таблиці	Причина звернення	Історія спілкування з ботом	Повна назва пристою та його характеристики
2	230001	15.06.2023 09:56	testemail@lnup.edu.ua	testemail@mail.com	Тестовий користувач			Інше	Детально опишіть причину зве	
3	230002	15.06.2023 10:14	kit@lnup.edu.ua		Відділ КІТ	2242900		Ремонт комп'ютерів	Внесіть назв	Ноутбук Asus
4	230003	15.06.2023 10:17	kadry@lnup.edu.ua		Відділ кадрів			Заправка або ремонт картрид	Виберіть, що	Canon 725 Starter
5	230004	15.06.2023 10:18	biblioteka@lnup.edu.ua	lnaulib@ukr.net	Наукова бібліотека	2242915		Відсутній доступ до інтернету	Впевніться, що Ваш пристрій підк	
6	230005	15.06.2023 10:21	yuryst@lnup.edu.ua		Юридичний відділ	2242905		Ремонт друкарки	Виберіть, що	Canon LBP3010
7	230006	15.06.2023 10:26	aspirant@lnup.edu.ua		Відділ аспірантури	2242916		Інше	Детально опишіть причину зве	
8	230007	16.06.2023 22:39	testemail@lnup.edu.ua	testemail@mail.com	Тестовий користувач	0931234567		Інше	Детально опишіть причину зве	
9	230008	16.06.2023 22:44	testemail@lnup.edu.ua	testemail@mail.com	Тестовий користувач	0931234567		Інше	Детально опишіть причину зве	
10	230009	16.06.2023 23:03	testemail@lnup.edu.ua	testemail@mail.com	Тестовий користувач	0931234567		Інше	Детально опишіть причину зве	

Рисунок 4.3 – Фрагмент таблиці заявок

Загалом розроблений чат-бот продемонстрував високу швидкість реакції та масштабованість. Python дозволив швидко та ефективно вдосконалювати функціонал чат-бота за допомогою вбудованих інструментів. Розроблений алгоритм надсилання звітів дозволив у швидкому режимі впроваджувати виправлення помилок та збоїв у роботі системи.

4.2. Порівняння з існуючими рішеннями

Порівняння чат-боту відділу КІТ Львівського національного університету природокористування для мессенджера Telegram з існуючими рішеннями дозволяє визначити його конкурентоспроможність та ефективність.

Порівняння було виконано за такими критеріями:

- **Функціональність**: розроблене рішення показало високу гнучкість при створенні заявки в порівнянні з існуючими рішеннями, оскільки містить велику кількість запрограмованих сценаріїв розмови з користувачами. Крім цього при розробці було враховано потреби відділу КІТ та впроваджено додаткові інструменти для відслідковування статусів завдань, спілкування з замовником заявки для уточнення деталей тощо. Чат-бот також містить ряд додаткових інструментів, що полегшують розробку та виправлення неполадок. В порівнянні з сервісом Microsoft Forms, що використовувався відділом для отримання запитів до впровадження чат-бота, розроблене рішення надає можливість зворотнього зв’язку через посередництво та загалом пропонує більше інтерактивності при використанні;
- **Швидкість та продуктивність**: Чат-бот показав високу швидкість реакції при виконанні користувачами автономних або простих дій. Проте можуть виникнути затримки у відповіді, коли запитується інформація з віддалених баз даних або відбуваються ресурсоємні процеси, наприклад, створення чи змінення статусу заявки, обробка повідомлень з медіафайлами тощо;
- **Взаємодія з користувачами**: При розробці було використано можливості інструментів Telegram Bot API, такі як кнопки з готовими відповідями, вбудовані у повідомлення кнопки та команди. За рахунок цього значно покращився досвід використання у порівнянні з багатьма готовими реалізаціями;

- **Доступність та підтримка**: Завдяки розробленого продукту на віддаленому хостингу, вдалося досягти максимальної доступності: до чат-бота можна звернутися у будь-який час доби. У разі збоїв у роботі чат-бота, для можливості зворотнього зв’язку працівникам відділу КІТ надсилаються контакти особи, яка намагалася створити заявку.

4.3. Аналіз отриманих результатів

У порівнянні з готовими рішеннями на ринку, розроблений чат-бот має як переваги, так і недоліки. Гнучкість інструментів розробки дозволила створити індивідуальне рішення для потреб відділу КІТ та постійно його вдосконалювати. В порівнянні з сервісом Microsoft Forms, чат-бот надав безліч можливостей для впровадження власних індивідуальних інструментів для потреб відділу КІТ.

Загалом відгуки користувачів та працівників відділу після використання чат-бота є позитивними:

- чат-бот не вимагає від користувача складної авторизації, достатньо знати свою адресу корпоративної пошти;
- надає зворотній зв’язок для потреб відділу та є зручним інструментом для ведення обліку робіт;
- є доступним, оскільки більшість працівників та студентів використовують месенджер Telegram;

Розроблений чат-бот технічної підтримки виходить на високий рівень у багатьох аспектах, забезпечуючи гнучкість, швидкість, зручну взаємодію та доступність для користувачів. Його використання може покращити процеси технічної підтримки та задовольнити потреби відділу КІТ та користувачів.

5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Аналіз небезпеки під час роботи за комп'ютером

Виявлено, що під час використання комп'ютера найбільше небезпеки загрожують зоровій, опорно-руховій та нервово-психічній системам. Досі точні причини цих проблем не встановлені, будь-то випромінювання або статична поза.

Головним джерелом небезпеки є дисплей, який випромінює різні види випромінювання, такі як рентгенівське, ультрафіолетове, інфрачервоне та електромагнітне. Існують норми для кожного з цих видів випромінювання, але вони варіюються в залежності від країни. Однак ці норми враховують опромінення всього організму, тоді як фактично вплив спостерігається лише на верхню частину тулуба. Комплексний вплив всіх цих полів на здоров'я людини все ще потребує дослідження.

Відеодисплеї також порушують рівновагу між позитивно й негативно зарядженими іонами у повітрі, що також негативно впливає на здоров'я. Щоб уникнути цього, важливо забезпечити належну вентиляцію робочого приміщення та проникнення свіжого повітря до робочого місця. Встановлено чіткі розміри столу та стільця для роботи з комп'ютером, оскільки неправильна постава може негативно позначитися на скелетно-м'язовій системі. Робочий стіл повинен бути просторим, з підставкою для ніг, а робочий стілець – регульованою висотою, нахилом сидіння та спинки.

Є два джерела випромінювання – системний блок і монітор:

1. Системний блок створює електромагнітне поле, а також шум від вентиляторів. Шкода від електромагнітного поля виникає лише при високому рівні. Однак комп'ютер створює значно менше поля, ніж мобільний телефон.
2. Монітор має два основних шкідливих фактори. Перший – бета-випромінювання, яке створює зображення на екрані. Другий – висока напруга, яка викликає іонізацію повітря. Бета-випромінювання поширюється

з монітора в двох напрямках – вперед і назад. На сьогоднішній день монітори мають дуже низький рівень бета-випромінювання, а електрони виходять за межі екрану лише на кілька сантиметрів. Основне випромінювання монітора спрямоване назад, тому "зона ураження" розповсюджується на відстань до метра-півтора. Висока напруга також перетворює молекули повітря на шкідливі позитивні іони. Виробники моніторів і телевізорів ставлять жорсткі вимоги до використання високих напруг, що є позитивним фактором.

5.2. Освітлення та вентиляція в робочому приміщенні

Згідно з правилами, освітлення при роботі з комп'ютером має падати зліва, а відстань від очей до екрана повинна бути близько 50 сантиметрів. Крім того, крісло слід налаштувати таким чином, щоб очі були на одному рівні з центром монітора. Експерти підкреслюють, що саме очі зазнають найбільшого навантаження під час роботи з комп'ютером. Довгий період спостереження за екраном призводить до зменшення частоти моргання. Це викликає почервоніння, подразнення та сльозотечу, що в свою чергу може призвести до погіршення зору. Наближена відстань до екрану, малий розмір шрифту, мерехтіння та неправильне освітлення в кінцевому підсумку можуть сприяти розвитку короткозорості. Якщо ви помічаєте почервоніння, сльозотечу, печіння та головний біль, це ознаки втоми очей, і вам слід взяти перерву для відпочинку. Однак, краще не доводити свої очі до такого стану, а забезпечувати їм відповідний відпочинок.

5.3. Інструкція з охорони праці під час роботи за комп'ютером

Персонал, що працює на комп'ютері, повинен дотримуватися вимог інструкції, що розроблена на основі Санітарних норм і правил, нести особисту відповідальність за дотримання вимог безпеки своєї праці та уникати створення небезпечних або шкідливих виробничих факторів для інших працівників чи

комп'ютерної техніки. Під час роботи з комп'ютером шкідливими і небезпечними факторами є:

- електромагнітне випромінювання;
- електростатичні поля;
- потужні іонізуючі випромінювання;
- загальна втома;
- втомлюваність очей;
- ризик ураження електричним струмом;
- пожежна небезпека.

Режими праці та відпочинку при використанні комп'ютера повинні бути організовані залежно від типу та категорії трудової діяльності. Трудову діяльність можна розділити на 3 групи:

- Група А – робота з читанням інформації з екрану комп'ютера з переднім запитом;
- Група Б – робота з введенням інформації;
- Група В – творча робота в режимі діалогу.

Основною роботою з комп'ютером слід вважати таку, що займає від 50% часу від загального часу за комп'ютером. Для видів трудової діяльності встановлюються 3 категорії важкості і напруженості роботи з комп'ютером, які визначаються:

- для групи А – за загальною кількістю прочитаних знаків протягом робочого часу з комп'ютером, але не більше 60 000 знаків;
- для групи Б – за загальною кількістю прочитаних або введених знаків протягом робочого часу з комп'ютером, але не більше 40 000 знаків;
- для групи В – за загальним часом безпосередньої роботи з комп'ютером, але не більше 6 годин протягом робочого часу за комп'ютером.

Для забезпечення оптимальної працездатності і збереження здоров'я під час робочого часу з комп'ютером необхідно встановлювати регламентовані перерви.

Перед початком роботи необхідно переконатися, що монітори комп'ютера мають антиблікове покриття (крім групи А) з коефіцієнтом відображення не більше 0,5. Покриття також повинно забезпечувати зняття електростатичного заряду з поверхні екрана, захищати від іскріння і накопичення пилу. Корпус монітора повинен забезпечувати захист від іонізуючих та неіонізуючих випромінювань. Необхідно перевірити правильне розташування комп'ютера, забезпечивши відстань не менше 0,8 метра між стіною з віконними прорізами і столом. Відстань між робочими столами повинна бути не менше 1,2 метра. Заборонено розміщення другого робочого місця позаду комп'ютера.

6. ВИСНОВКИ ТА ПРОПОЗИЦІЇ

6.1. Основні результати дослідження та розробки

Використання чат-боту технічної підтримки, створеного за допомогою гнучких інструментів розробки, виявилося ефективним засобом надання швидкої та зручної допомоги користувачам. Він дозволяє знизити навантаження на технічну підтримку, забезпечує швидкість відповідей та покращує загальне задоволення користувачів.

Чат-бот технічної підтримки заснований на Telegram API дозволяє забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів, а також має потужні можливості для автоматизації відповідей та обробки запитів. Користувачі відзначили зручність та легкість використання чат-боту для отримання необхідної технічної підтримки. Вони оцінили швидкість відповідей та якість наданої інформації.

Функціональні та нефункціональні вимоги до чат-боту технічної підтримки були визначені і виконані успішно, забезпечуючи потрібний рівень функціональності та якості обслуговування. Результати показали, що чат-бот технічної підтримки був ефективним інструментом для вирішення типових технічних питань та проблем, що часто виникають у користувачів. Застосування Aiogram у розробці чат-боту дозволило забезпечити широкі можливості для взаємодії з користувачами та автоматизації процесу надання технічної підтримки.

На основі отриманих результатів можна зробити висновок, що використання чат-боту технічної підтримки є ефективним рішенням для забезпечення швидкого та якісного обслуговування користувачів. Продовження розвитку та вдосконалення чат-бота може сприяти подальшому поліпшенню процесу технічної підтримки та задоволенню потреб користувачів.

6.2. Рекомендації щодо подальшого розвитку та вдосконалення проекту

На основі проведеного дослідження та отриманих результатів, було розроблено список рекомендацій щодо подальшого розвитку проекту чат-боту технічної підтримки:

- Збільшити кількість можливих типів звернень при створенні заявки. Наприклад, додати типи “Налаштування мережевого обладнання”, “Проблеми з Office 365”, “Проблеми з Moodle” тощо;
- Покращити навчання чат-бота: Застосувати технології машинного навчання для постійного вдосконалення чат-бота. Збирати дані про запити користувачів та використовувати їх для покращення відповідей та точності розпізнавання запитів;
- Розширити інтеграцію з іншими системами: Розглянути можливість інтеграції чат-бота з іншими системами відділу комп’ютерних інформаційних технологій. Наприклад, інтеграція з базою знань або системою відстеження заявок;
- Вдосконалити аналітику та звітність: Розглянути можливість включення аналітичних інструментів для відстеження ефективності та задоволення користувачів. Використовувати дані про використання чат-бота для вдосконалення процесу технічної підтримки;
- Проводити користувацьке тестування: Проводити регулярні тести та отримувати зворотний зв'язок від користувачів. Враховувати їхні коментарі та пропозиції для подальшого вдосконалення чат-боту;
- Розглянути можливості мультиплатформеності: Вивчити можливість розширення чат-боту на інші месенджери або платформи, що дозволить більш широкому колу користувачів отримати доступ до технічної підтримки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ушакова І.О. Підходи до створення інтелектуальних чат-ботів. Системи обробки інформації. 2019. № 2(157). С. 76-83.
2. Beaver L. The Chatbots Explainer / L. Beaver. – BI Intelligence Copyright, 2016–2023 р. [Електронний ресурс]. URL: https://www.businessinsider.com/intelligence/chatbots-explainer.
3. Сокол-Торська Ольга, Енциклопедія з маркетингу, просування та таргетингу: 2023, 168 с.
4. Venigalla, V., Hall, P. D., Janarthanam, S. (2017). Hands-On Chatbots and Conversational UI Development: Build Chatbots and Voice User Interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills. Велика Британія: Packt Publishing, 392 с.
5. Що таке API і де їх шукати [Електронний ресурс]. URL: https://info.nic.ua/uk/blog-uk/api-2/
6. D. J. Stoner, L. Ford, and M. Ricci, “Simulating Military Radio Communications Using Speech Recognition and Chat-Bot Technology”, 2003.
7. Бісікало, О. В.; Юрчук, М. С. Класифікація чат-ботів для електронної комерції. «Сучасна молодь в світі інформаційних технологій»: матеріали, 2022, 164 с.
8. Офіційна документація Aiogram для класу Handler [Електронний ресурс]. URL: https://aiogram-birdi7.readthedocs.io/en/latest/dispatcher/#handler-class
9. HTTP Long Polling [Електронний ресурс]. URL: https://medium.com/tech_internals/long-polling-and-sse-7e8ddaffcaa7
10. What is a Webhook? [Електронний ресурс]. URL: https://medium.com/codeburst/what-are-webhooks-b04ec2bf9ca2
11. How to Create and Deploy Telegram Bot with Python [Електронний ресурс]. URL: https://djangostars.com/blog/how-to-create-and-deploy-a-telegram-bot/

12. Багатонитковість. Українська Вікіпедія; версія від 29 січня 2022, 09:55 UTC [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Багатонитковість
13. Вступ до асинхронного програмування на Python [Електронний ресурс]. URL: https://devzone.org.ua/post/vstup-do-asinhronnogo-programuvannya-na-python
14. Основні відомості про бази даних [Електронний ресурс]. URL: https://support.microsoft.com/uk-ua/office/основні-відомості-про-бази-даних-a849ac16-07c7-4a31-9948-3c8c94a7c204