МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА

на тему

Розробка інформаційно-пошукової системи для покращення роботи з клієнтами

Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-1-23
Бученко О.Є.
Науковий керівник:
к.т.н., доц. Астістова Т.І.
Рецензент:
к.т.н., доц. Мельник Г.В.

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

_____ Наталія ЧУПРИНКА

«_____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Бученку Олександру Євгенійовичу

1. Тема кваліфікаційної роботи Розробка інформаційно-пошукової системи для покращення роботи з клієнтами, науковий керівник роботи к.т.н., доц. Астістова Тетяна Іванівна, затверджені наказом КНУТД від "3" вересня 2024 року № 118-уч
2. Строк подання студентом роботи 22.11.2024р.
3. Вихідні дані до кваліфікаційної роботи Розробки кафедри комп'ютерних наук , рекомендована література, додатки
4. Зміст дипломної кваліфіційної роботи : Розділ 1.Аналіз предметної області, Розділ 2. Проектування програмної реалізації, РозділЗ. Розробка чат-боту.

5 . Дата видачі завдання:_03. 09. 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	29.09.2024	
2	Розділ 1 Аналіз предметної області	01.10.2024	
3	Розділ 2. Проектування програмної реалізації	15.10.2024	
4	Розділ 3. Розробка чат-боту	20.10.2024	
5	Висновки	29.10.2024	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	11.11.2024	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку	12.11.2024	
8	Подача кваліфікаційної роботи для рецензування	18.11.2024	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату	20.11.2024	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри	22.11.2024	

Студент _____ Олександр БУЧЕНКО

Науковий керівник роботи _____ Тетяна АСТІСТОВА

АНОТАЦІЯ

БученкоО.Є. Розробка інформаційно-пошукової системи для покращення роботи з клієнтами.

Дипломна магістерська робота за спеціальністю 122 – «Комп'ютерні науки». – Київський національний університет технологій та дизайну, Київ, 2024 рік.

Дипломна робота присвячена аналізу використання інформаційно-пошукових систем в бізнес цілях. Під час роботи був розроблений чат-бот, який має інтуїтивно зрозумілий інтерфейс, що дозволяє клієнтам легко знаходити потрібну інформацію.

Для реалізації поставленого завдання було обрано сучасний і зручний інструмент AIogram API, який є бібліотекою для створення асинхронних чат-ботів у месенджері Telegram. Мова програмування Python була використана для написання коду бота завдяки її простоті, універсальності та великій кількості готових рішень у вигляді бібліотек. Для збереження і передачі даних між ботом та сервером було використано формат JSON, який забезпечує ефективну взаємодію між сервісами і дає змогу працювати з великими обсягами даних без втрати продуктивності.

Робота детально аналізує сучасні тенденції впровадження чат-ботів у бізнесі та їх вплив на автоматизацію процесів.

Ключові слова: чат-бот, API, Python, месенджери, JSON, мобільний додаток.

ANNOTATION

Buchenko Oleksandr. Topic Development of an information retrieval system to improve customer service.

Master's thesis in the specialty 122 – “Computer Science”. – Kyiv National University of Technology and Design, Kyiv, 2024.

The thesis is devoted to the analysis of the use of information retrieval systems for business purposes. During the work, a chatbot was developed that has an intuitive interface that allows customers to easily find the information they need.

To realize the task, we chose a modern and convenient tool AIogram API, which is a library for creating asynchronous chatbots in the Telegram messenger. The Python programming language was used to write the bot code due to its simplicity, versatility, and a large number of ready-made solutions in the form of libraries. The JSON format was used to store and transfer data between the bot and the server, which ensures efficient interaction between services and allows working with large amounts of data without losing performance.

The paper analyzes in detail the current trends in the implementation of chatbots in business and their impact on process automation.

Keywords: chatbot, API, Python, messengers, JSON, mobile application.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні бізнес-процеси активно інтегрують автоматизовані рішення для забезпечення ефективної взаємодії з клієнтами, підвищення якості сервісів і оптимізації ресурсів. Інформаційно-пошукові системи, зокрема чат-боти, стали ключовими інструментами, які дозволяють компаніям не лише оперативно реагувати на запити клієнтів, але й персоналізувати послуги відповідно до їхніх потреб. В умовах глобальної цифровізації, популярність таких рішень стрімко зростає, особливо у контексті їх інтеграції в популярні месенджери, такі як Telegram, Viber, WhatsApp та Facebook Messenger. Ці платформи забезпечують зручний доступ до інформації та послуг, що робить їх актуальними для більшості компаній, які прагнуть бути ближчими до своїх клієнтів.

Об'єкт дослідженняю Об'єктом дослідження є автоматизовані засоби взаємодії з клієнтами, зокрема чат-боти, які функціонують як інформаційно-пошукові системи. Вивчення охоплює їх роль у бізнес-середовищі, можливості інтеграції з різними платформами та вплив на ефективність процесів обслуговування.

Завдання. Робота має на меті створення Telegram-бота, що забезпечить автоматизацію комунікацій та підвищення ефективності взаємодії між компаніями та їхніми клієнтами. Для досягнення цієї мети поставлено такі завдання:

- Провести аналіз існуючих інформаційно-пошукових систем та їх застосування в бізнесі.
- Здійснити вибір платформи, що найкраще підходить для розробки бота.

- Спроектувати функціональні можливості бота з урахуванням вимог цільової аудиторії.
- Розробити Telegram-бот із використанням сучасних інструментів, таких як Python та бібліотека Aiogram.
- Забезпечити тестування та налаштування бота для інтеграції з бізнес-процесами.

Наукова новизна. Новизна дослідження полягає в розробці інноваційного рішення, яке спирається на використання асинхронних технологій та інтеграцію з Telegram API. На відміну від традиційних підходів, цей підхід дозволяє створити багатофункціональний бот, здатний адаптуватися до специфічних вимог різних компаній та виконувати завдання, пов’язані з автоматизованим обслуговуванням клієнтів. Важливим аспектом є можливість інтеграції бота з іншими системами, що розширює його функціонал і сприяє впровадженню сучасних цифрових технологій у бізнес.

Апробація результатів в виступу на конференції:

1. Астістова Т. І. Інформаційно-пошукові системи в задачах оптимізації клієнтського обслуговування / Т. І. Астістова, О. Є. Бученко // Тези VIII Міжнародної науково-практичної конференції «Мехатронні системи: інновації та інжиніринг – «MSIE-2024» К. КНУТД , 7 листопада 2021р. - С. 171-172

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальні поняття про інформаційно-пошукові системи

Інформаційно-пошукова система (ІПС) – це сукупність програм або інструментів,які допомагають вам знайти потрібну інформацію серед великого обсягу даних. Наприклад, що у вас є велика бібліотека і ваша задача- знайти певну книгу або сторінку з певною інформацією. Ця пошукова система допоможе вам швидко знайдети те, що ви шукаєте.[1]

Сама пошукова система з'явилася в 1990-1992 роках. Пошук за запитом "хто є користувачем" розпочався в 1982 році, а багаторазовий пошук користувачів інформаційного сервісу Knowbot був вперше проведений в 1989 році. Першою добре задокументованою пошуковою системою для пошуку файлів вмісту, тобто файлів FTP, була Archie, яка дебютувала 10 вересня 1990року. Google з'явився в 1998 році. Це ще один проект студентів Стенфорда, цього разу Ларрі Пейджа та Сергія Бріна. Зараз в наших реаліях все більше людей користуються месенждерами, які поступово починають замінювати старі пошукові системи, наприкладі Telegram, WeChat, WhatsApp ці додатки можуть замінити всі програми які встановлені на смартфонах. Зараз WeChat, WhatsApp та Telegram є одними з найбільших месенджерів, яким постійно користується близько мільярда осіб. Ця цифра співмірна із розмірами аудиторії Facebook.

Основна мета створення пошукової системи - забезпечити доступність і зручність для користувачів при використанні інформації, а також можливість просування бізнесу і спілкування з аудиторією через Інтернет. Для досягнення цих цілей важливо враховувати потреби і очікування вашої цільової аудиторії, створювати зручний і простий користувальницький інтерфейс, забезпечувати швидкий і безперешкодний доступ до інформації.[2]

1.2. Види інформаційно- пошувих систем

Інформаційно-пошукові системи стали невід'ємною частиною нашого повсякденного життя і допомагають нам виконувати різні дії в будь-якій сфері діяльності. Завдяки їм ми можемо швидко знайти потрібний товар, забронювати авіаквиток, визначити найкоротший маршрут до пункту призначення, записатися на прийом до лікаря. Така система має величезний вплив на наше повсякденне життя і стала необхідним інструментом для багатьох.

ІПС надає доступ до різноманітної інформації, від прогнозів погоди і дорожніх умов до рекомендацій про те, коли краще вийти з дому, щоб вчасно дістатися до потрібного місця, що забезпечує швидкість і точність, які роблять їх незамінними в сучасному світі, де інформація відіграє вирішальну роль.[3]

Використання інформаційно-пошукових систем відкриває нові можливості для бізнесу та поліпшує обслуговування та взаємодію з клієнтами. Хоча офіційної класифікації інформаційно-пошукових систем (ІПС) не існує, їх можна розрізняти за різними характеристиками, що визначають особливості їх функціонування та призначення. Загалом можна виділити такі типи ІПС:

Як швидко знайти інформацію:

- Пошук Google,Yahoo,Bing: універсальний інструмент для пошуку будь-якої інформації.
- Elasticsearch: пошукова система баз даних.

Для пошуку всередині компанії:

- Microsoft SharePoint: пошук внутрішньої інформації та обмін нею.
- Confluence: систематизація знань про компанію та документації.

Для автоматизації пошуку і роботи з клієнтами:

- Telegram-боти: для автоматичного надання інформації клієнтам (наприклад, пошуку статусу замовлення).
- CHATGPT API: інтеграція пошукової системи в месенджер.

Як аналізувати поведінку клієнтів:

- Google Analytics: аналіз взаємодії клієнтів з веб-ресурсами.
- CRM-система: збір і аналіз даних про клієнтів.

Як зберігати і систематизувати інформацію:

- Notion: База знань з гнучкою структурою.
- Evernote: для особистих нотаток та файлів.

1.3. Порівняльний огляд платформ для впровадження чат-бота

На сучасному ринку API стають все більш важливими в цифровому світі, оскільки вони дозволяють обмінюватися даними та іншими ресурсами між різними додатками. За допомогою API для обміну повідомленнями користувачі можуть спілкуватися один з одним простіше, ніж будь-коли раніше. Вони пропонують широкий спектр можливостей взаємодії, від телефонних і відеочатів та текстових повідомлень до завантаження та обміну фотографіями.

У даному порівнянні ми розглянемо п'ять найважливіших функцій API для обміну повідомленнями:

- Можливості API: Ми оцінили численні комунікаційні можливості кожного API.
- Доступність платформи: ми проаналізували різні середовища, в яких можуть працювати програми.
- Популярність: ми оцінили популярність додатків для обміну повідомленнями, щоб допомогти вам вирішити, чи варто їх інтегрувати у ваш сценарій використання.
- Вартість: ми проаналізували вартість використання API для створення комунікаційних функцій.
- Простота використання: ми оцінили, наскільки легко інтегрувати кожен API для обміну повідомленнями в конкретний додаток.

WhatsApp

WhatsApp вважається крос-платформним сервісом обміну повідомленнями з найкращим API для обміну миттєвими повідомленнями, який належить Facebook.

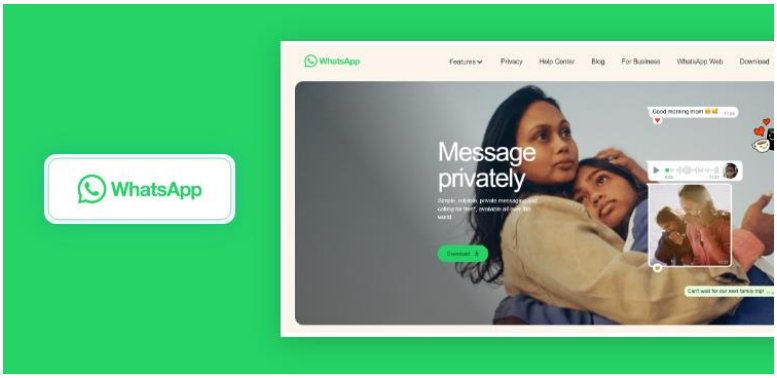


Рисунок 1.1 месенджер WhatsApp

Можливості API: WhatsApp Business API дозволяє компаніям використовувати функціонал сервісу для спілкування з клієнтами. Ви можете використовувати цей API для створення корпоративного профілю, який описує ваші пропозиції, взаємодіяти з клієнтами в режимі реального часу, передавати зображення та інші медіа, вести аудіо- та відео-розмови тощо.[4]

Доступність платформи: мобільні пристрої, стаціонарні комп'ютери та Інтернет.

Популярність: З жовтня 2018 року щомісячна база користувачів складає понад 1,5 мільярда людей з усього світу.

Ціна: Безкоштовно

Простота використання: Facebook відомий тим, що надає велику документацію, яка допомагає розробникам легко інтегрувати послугу у свої програми, а API WhatsApp є доказом цього.

Telegram

Telegram — це додаток для обміну повідомленнями, який обіцяє надати користувачам безпечне, швидке та просте спілкування.

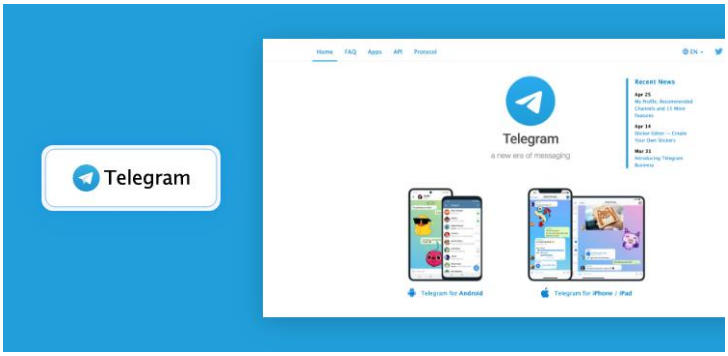


Рисунок 1.2 месенджер Telegram

Можливості API: Telegram пропонує два типи найкращих API для обміну повідомленнями (API для веб-сайтів і додатків) - API для ботів і API для Telegram.

Ви можете створювати ботів, які підключаються до платформи Telegram і виконують такі завдання, як автоматичне підключення до інших сторонніх сервісів, отримання сповіщень та реагування на користувачів за допомогою API ботів. Telegram API (API для чат-додатків) дозволяє створювати додатки Telegram і виконувати деякі специфічні для платформи операції, включаючи надсилання та отримання повідомлень, перевірку повідомлень і створення груп за допомогою API для групових чатів.[5]

Доступність платформи: мобільні пристрої, веб та стаціонарні комп'ютери.

Популярність: З 2018 року Telegram офіційно має понад 900 мільйонів активних користувачів щомісяця.

Вартість: Безкоштовно

Простота використання: Отримайте детальну документацію про те, як використовувати API. Крім того, існує процвітаюча спільнота розробників, яка допомагає один одному впроваджувати API Telegram.

Facebook messenger

Автономний інструмент обміну повідомленнями від Facebook, який називається Messenger, дозволяє легко надсилати та отримувати миттєві повідомлення та інші інтерактивні матеріали.

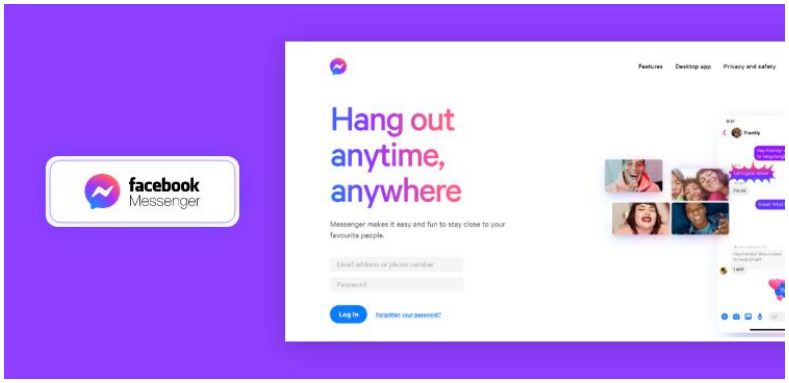


Рисунок 1.3 месенджер Facebook Messenger

Можливості API: За допомогою Facebook Messenger API розробники можуть інтегрувати функції для надсилання та отримання різних типів контенту, прийому платежів, знайомства з новими людьми, отримання ідентифікаторів користувачів та профілів, використання технологій обробки природної мови Facebook та відстеження ефективності за допомогою Facebook Analytics. [6]

Доступність платформи: веб та мобільні операційні системи.

Популярність: З 2018 року Facebook Messenger має понад 1,3 мільярда активних користувачів щомісяця, що робить його другою найпопулярнішою мережею чатів після WhatsApp.

Ціна: Безкоштовно

Простота використання: Проста документація, шаблони та навчальні посібники допоможуть вам швидко та легко інтегрувати додаток Messenger.

Viber

Платформа Viber – це додаток для обміну повідомленнями та чату, який дозволяє користувачам взаємодіяти один з одним за допомогою голосових дзвінків, текстових повідомлень, відеочатів та зображень.

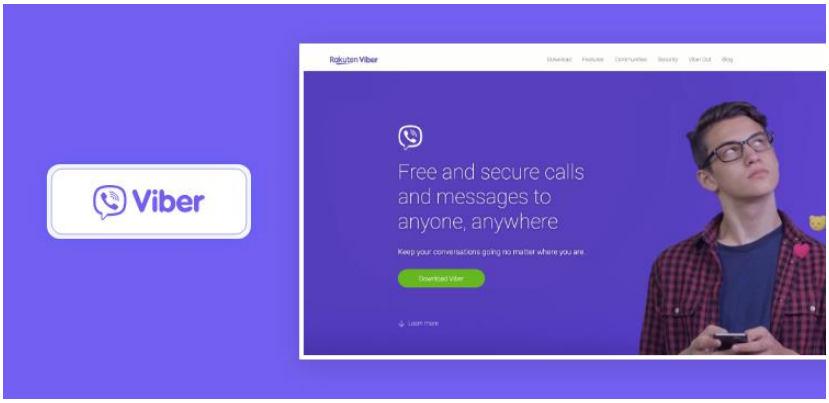


Рисунок 1.5 месенджер Viber

Можливості API: API обміну повідомленнями в реальному часі Viber для Android та API обміну повідомленнями для iOS дозволяють додавати програмні функції та створювати унікальний користувацький досвід. API включає методи для спілкування за допомогою текстових повідомлень, взаємодії з відео-повідомленнями, надсилання повідомлень кільком користувачам Viber, отримання інформації про користувача та багато іншого.[7]

Доступність платформи: мобільні пристрої та стаціонарні комп'ютери

Популярність: З 2018 року має понад 260 мільйонів активних користувачів щомісяця.

Ціна: Безкоштовно

Простота використання: Отримайте детальну документацію, яка дозволить розробникам швидко і легко інтегрувати API.

WeChat

WeChat – це відомий сервіс обміну миттєвими повідомленнями, який дозволяє безкоштовно обмінюватися відеоповідомленнями, текстовими чатами та голосовими дзвінками.

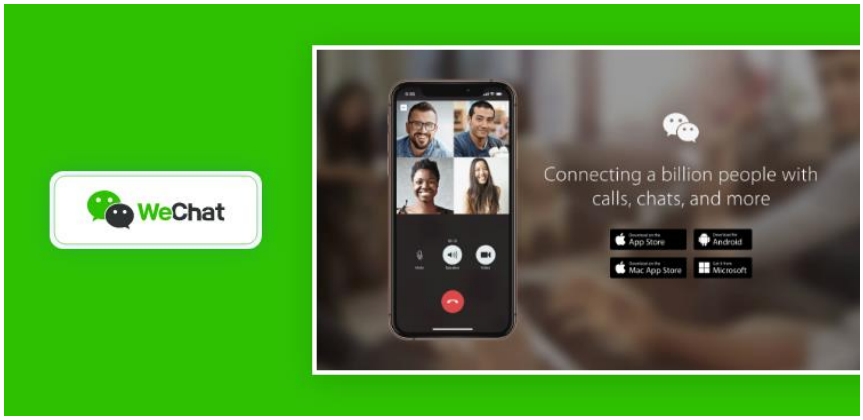


Рисунок 1.5 месенджер WeChat

Можливості API: WeChat API дозволяє розробникам отримувати доступ до функцій програмного забезпечення та інтегрувати їх у свої випадки використання. API включає методи для передачі та отримання повідомлень, отримання інформації про користувача, визначення місцезнаходження користувачів, взаємодії з голосом, відео, зображеннями та музичними матеріалами, а також багато іншого.[8]

Доступність платформи: мобільні пристрої, настільні комп'ютери та Інтернет.

Популярність: З 2018 року має понад 1 мільярд активних користувачів щомісяця, що робить її другою за популярністю платформою для обміну повідомленнями після Facebook Messenger.

Ціна: Безкоштовно.

Простота використання: Отримайте вичерпну документацію для розробників та SDK, щоб швидко і без зусиль розширити можливості вашого додатку за допомогою API.

У таблиці 1.1 можна побачити порівняння існуючих аналогів.

Табліця 1.1 – Порівняння подібних месенджерів

Назва критерію	**Назва ресурсу (сайту)**				
	WhatApp	**Telegram**	**Viber**	**WeChat**	**Fb messenger**
Відкритий API	-	+	+	+	+
Популярність	-	+	+	-	-
Шифрування, анонімність	-	+	-	+	-
Наявність мультиплатформленості	+	+	+	+	+

Проведений аналіз платформ для інтеграції чат-ботів продемонстрував різноманітність можливостей і переваг, які надають ці сервіси для бізнесу. Популярні платформи, такі як WhatsApp, Telegram, Facebook Messenger, Viber та WeChat, забезпечують доступ до інструментів для інтеграції API, які дозволяють автоматизувати комунікацію, надавати персоналізовані послуги та покращувати взаємодію з клієнтами.

Для конкретних бізнес-задач вибір платформи залежить від потреб компанії та її клієнтів. Наприклад, завдяки своїй гнучкості та активній спільноті розробників, Telegram може стати найкращим вибором для створення навчальних чат-ботів, тоді як WhatsApp або Facebook Messenger є хорошим вибором для компаній, орієнтованих на широку аудиторію.[9]

Інтеграція чат-ботів через ці платформи допомагає оптимізувати робочі процеси, автоматизувати відповіді на запити, забезпечити швидкий доступ до інформації, а також підвищити якість сервісу та лояльність клієнтів.

1.4. Огляд існуючих рішень

Провівши велике дослідження та оцінивши численні ресурси, ми не змогли знайти чат-ботів, які були б орієнтовані виключно на підтримку інтернет-магазинів та ремонтних сервісів і охоплювали б усі необхідні для цих сфер функції. Проте існує чимало рішень, які можуть допомогти компаніям оптимізувати свої процеси:

- Платформа для створення чат-ботів з найважливішими функціями;
- Управління замовленнями - система для комунікації з клієнтами;
- CRM-системи, що інтегровані веб-сайт і месенджери ;
- Гайдлайни та шаблони для створення чат-ботів;
- Сервіси для аналізу запитів клієнтів і створення автовідповідачів;
- Платформи, що дозволяють клієнтам бронювати час для ремонту або консультації в режимі реального часу.

Після детального вивчення цих інструментів можна зробити висновок, що багато з них надають якісний функціонал, але для власників інтернет-магазинів або ремонтних сервісів, які тільки починають впроваджувати автоматизацію і фокусуються на певних аспектах бізнесу, таке рішення підійде для інтеграції з базами даних клієнтів, оформлення замовлень, створення розкладів або налаштування автоматичних підказок для вашої ремонтної служби.

Висновки до розділу 1

В даному розділі було проаналізовано принципи функціонування інформаційно-пошукових систем та їх значення для сучасних бізнес-процесів. Проведено порівняльний огляд платформ для впровадження чат-ботів, що дозволило оцінити їхні переваги, недоліки та можливості адаптації під потреби різних організацій. Окремо було розглянуто існуючі аналоги чат-ботів, що дозволило виокремити їхні сильні сторони та виявити недоліки, які можуть бути вдосконалені у новій розробці.

Проведений аналіз показав, що впровадження чат-ботів сприяє автоматизації процесів, покращенню якості обслуговування клієнтів і підвищенню ефективності роботи організацій. Отримані результати створюють основу для подальшої розробки ефективного рішення у наступних розділах.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1 Призначення чат-боту

Чат-бот створений на базі aiogram з метою автоматизувати спілкування з користувачами шляхом забезпечення зручного та ефективного способу взаємодії, який дозволяють отримувати необхідну інформацію та функціональні сервіси для полегшення виконання завдань в їх повсякденному житті та економити час користувачів покращуючи роботу обслуговуючого персоналу. [10]

Функціональні обов'язки бота:

Обробка запитів від користувачів - головне завдання чат-бота: надавати відповіді на питання і надавати необхідну інформацію або консультацію за допомогою заздалегідь підготовлених сценарій або інтеграцію з зовнішними базами даних.

Збір і аналіз інформаційних даних може включати у себе опитування користувачів або прийом заявок для запису на послуги чи реєстрацію на подію.

Автоматизація обов'язків: бот призначений для виконання стандартних завдань, таких як надсилання повідомлень, нагадування про подій чи автоматичний розрахунок даних на підставі введеною інформацію.

Персоналізація вхід для користувачів - бот мають зберігати профілі осіб з їх уподобаннями та минулими запитами для надання індивідуальних відповідей і рекомендацій.

Розширення підтримки мовності важливе для того, щоб чат-бот міг працювати на декількох мовах залежно від потреб користувача.

Розроблений бот буде всебічним інструментом для автоматизації процесів у різних сферах діяльності. Він сприятиме оптимізацій комунікацій та покращенню якості обслуговування в інтернет-магазинах та сервісах підтримки клієнтів чи освітніх галузях навчання.

2.2 Опис вимог до чат-боту

Чат-бот розроблений на підставі бібліотеки python-telegram-bot і повинен відповідати сучасним вимогам якості та зручності у використанні для користувача. Основна задача полягає у створенні інструменту, який допомагатиме людям швидко отримати потрібну інформацію або виконати певні дійства.

Одні з основних вимог – це виконання обробки текстових команд типу /start/, /help/, /service/, а також команди пов'язані з основною функціональністю бота. Чат-бот мають мати можливість відповідати на запити користувачів у письмовому форматі та надсилати різноманітні мультимедійні файли - зображення (картинки), відео чи документація (файли). За допомогою бібліотеки python-telegram-bot потрібно створити зручну систему меню на основі інлайн кнопок для спрощення навігації та полегшення користувачам оріентування у функцііоналі. [11]

Система мають бути здатна обробляти дані від користувачів шляхом заповнення форм або проведення опитувань і зберігання цих даних у базі даних (наприклад у PostgreSQL або SQLite) для подальшого аналізу та відображення результатів. Також важливо мати можливість інтегруватися з різними зовнішніми сервісами через API - на приклад для обробки платежів чи перевірки стану замовлень або взаюмодій з CRM-системами.

Зручність взаэмодіі користувача з чат-ботом – ключовий компонент успішного проекту. Інтерфейс чат-бота повинен бути легким для розуміння, з простими текстовими підказками і інформативними повідомленнями, які сприяють комунікаційному процесу між користувачем та ботом. Використання бібліотеки python-telegram-bot допоможе реалізувати ефективне управління помилками: чат-бот сповіщатиме про помилкові дії користувача та пропонуватиме альтернативні шляхи для їх вирішення.

Безпека є ще одним важливим аспектом у цьому контексті. Оскільки боти можуть працювати з особистою інформацію користувачів, дуже важливо

захищати дані під час їх передачі шляхом використання HTTPS для серверних запитів та контролю доступу до бази даних. [12]

Розробка повинна включати повний цикл тестування з перевіркою роботи кожного модуля, коректність обробки різних сценарій взаимодействия з ботом і перевірку на стійкість до навантаження. Особливу увагу слід приділити тестуванню інтеграція з Telegram API для уникнення помилок у роботі.

2.3 Проектування функціональних можливостей чат-боту

У цьому магістерському дослідженні створюється чат-бот для обслуговування клієнтів з такими можливостями:

- Зручний інтерфейс дозволить клієнтам спілкуватися з чат-ботом легко та без зусиль завдяки інтуітивно зрозумілому інтерфейсу.
- Контактна інформація: клієнти можуть отримати потрібні контактні дані через чат. Додатково буде доступна вся необхідна інформація.
- Поширені труднощі включають у себе можливість для клієнтів отримати доступ до бази найчастіше зустрічаних проблем в галузі надання послуг та пропонувати вирішення цих питань.

Кожен з ключових розділів чатбота виконуватиме певну функцію:

- Головна сторінка чат-бота містить загальну інформацію про компанію.
- Меню сервіс отримати повну інформацію про всі доступні послуги, а також ознайомитися з ціною та умовами надання кожною з них.
- Меню про нас – це розділ, де ви можете знайти місію та принципи роботи компанії.
- Пункт доставка та оплата описуватимемо способи доставки та методи оплати.

Адміністратори чат-бота матимуть можливість виконувати наступні операції:

- Керування контентом включаючи додавання текстів і зображень або видалення медіафайлів.
- Конфігурація розділів: адміністратор має можливість змінювати меню, структуру навігаційного меню та додавати чи редагувати сторінки.
- Керування функціоналом: можливість налаштування форми зворотнього зв'язку, реєстраційних форм та додавання інших інтерактивних елементів для поліпшення спілкування з користувачами.

Змінити налаштування може адміністратор - вони можуть корегувати заголовки, мета описи та ключові слова для оптимізація SEO і кольорову схему для поліпшення візуального вигляду чатбота.

На рис. 2.1 На діаграмі прецедентів UML будуть представлені основні функції чат-бота, які виконуються двома типами користувачів: звичайним користувачем, який взаємодіє з ботом для отримання інформації та виконання певних дій, і адміністратором, який має доступ до налаштувань та змін в контенті бота для забезпечення ефективної роботи системи.[13]

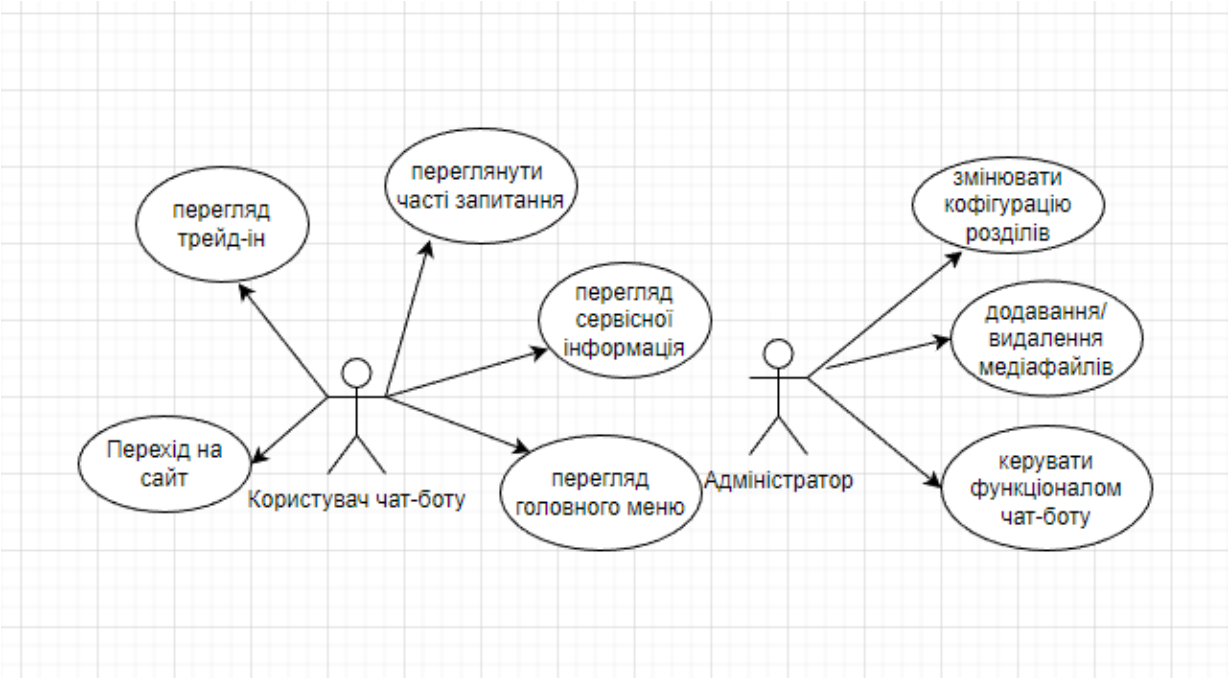


Рис. 2.1. UML-Діаграма прецедентів

Метод IDEF0 дозволяє зобразити процес у вигляді структурної моделі, яка враховує чотири ключові аспекти: входи (I), елементи керування (C), виходи (O) та механізми (M). Ці аспекти разом формують концепцію ICOM, яка описує взаємодію між різними складовими процесу.[14]

Кожна сторона функціонального блоку у діаграмі IDEF0 має своє призначення:

- Верхня сторона відображає елементи керування, які спрямовують процес.
- Ліва сторона представляє вхідні дані, які є тригером для запуску процесу.
- Права сторона показує вихідні дані, тобто результат виконання процесу.
- Нижня сторона ілюструє механізми, що використовуються для виконання процесу.

На рис. 2.2 представлена контекстна діаграма процесу розроблення чат-бота для платформи Telegram.

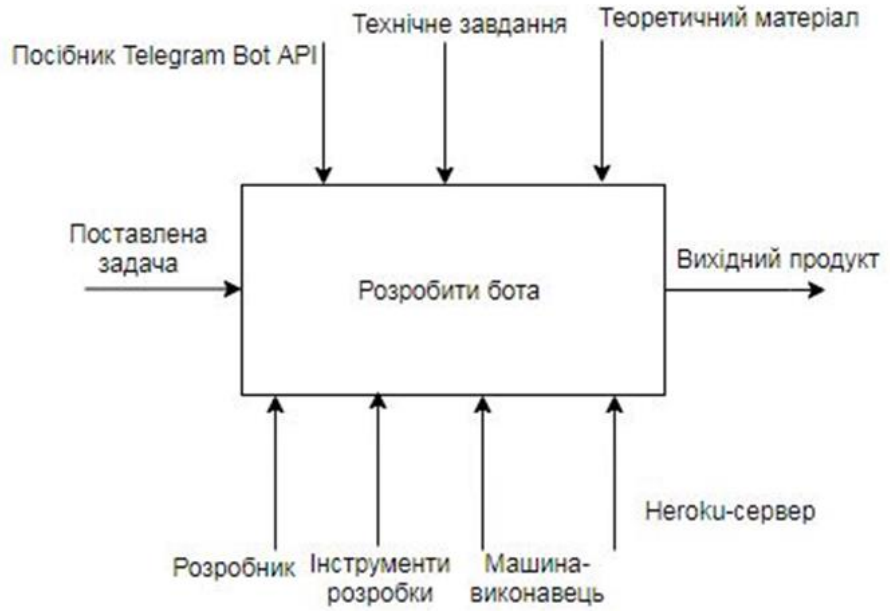


Рисунок 2.2 – Контекстна діаграма бота

Ця діаграма дає загальне уявлення про структуру процесу розробки, його ключові компоненти та зв'язки між ними.

Для більш детального аналізу контекстна діаграма може бути розширена у вигляді діаграми декомпозиції, яка розбиває основний процес на кілька підпроцесів. Це дозволяє отримати глибше розуміння кожного етапу розробки бота. Створення діаграми декомпозиції допомагає:

- Описати всі етапи розробки, забезпечуючи чіткий порядок виконання дій.
- Підготувати план розробки, який сприятиме ефективній реалізації проєкту.
- Надати замовнику наочне уявлення про обсяг роботи та визначити реалістичні терміни виконання завдань.

Таке моделювання є корисним як для планування, так і для контролю процесу розробки бота.

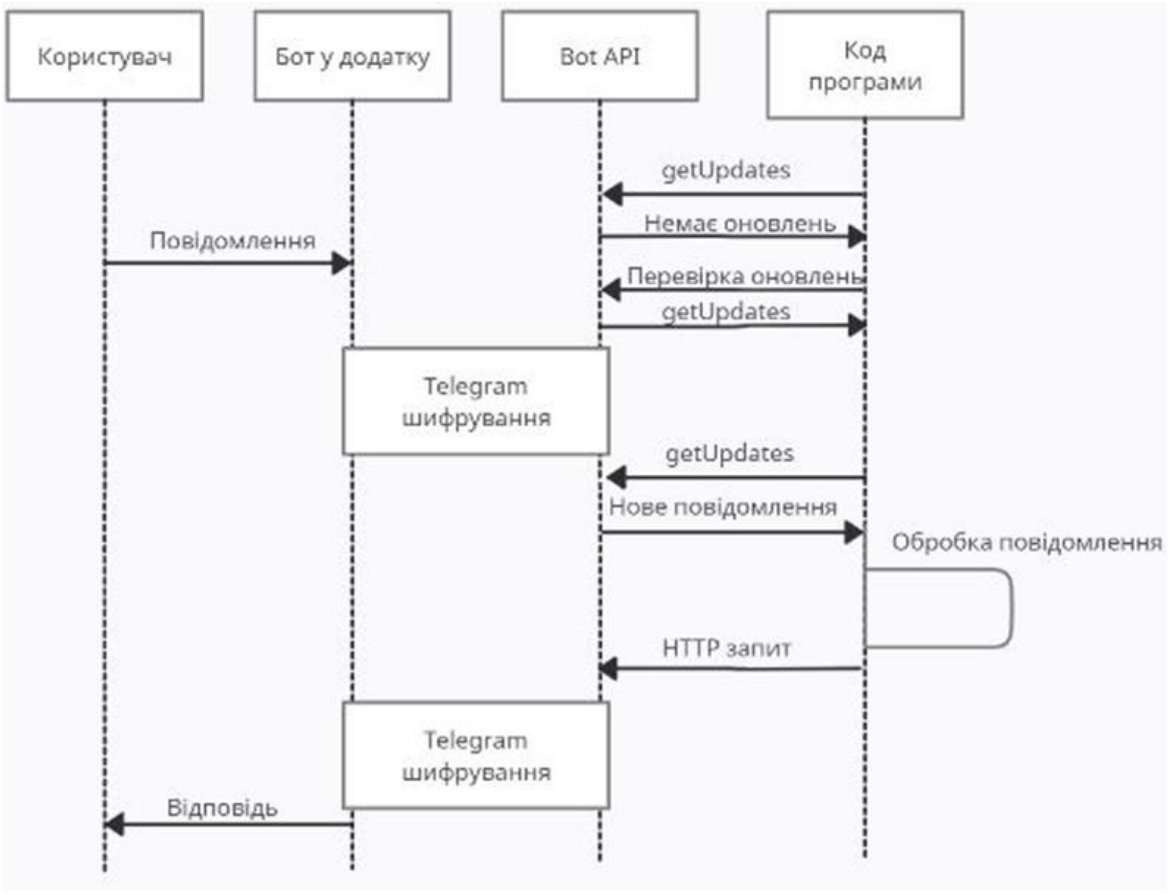


Рисунок 2.5 – Перевірка нових повідомлень

Процес перевірки нових повідомлень реалізується за допомогою спеціального методу getUpdates, який входить до складу Telegram Bot API. Цей метод є ключовим для забезпечення взаємодії між користувачем і чат-ботом, оскільки він дозволяє отримувати інформацію про нові повідомлення, що надходять у діалоговий інтерфейс. Завдяки цьому методу бот може реагувати на запити користувача та підтримувати інтерактивність у спілкуванні. Структура та функціональні можливості цього методу зображені на рисунку 2.6.[15]

```
MethodGetUpdates = 'https://api.telegram.org/bot{token}/getUpdates'.format(token=TOKEN)


while True:
    response = requests.post(MethodGetUpdates)
    result = response.json()
    print result
```

Рисунок 2.6 – Лістинг коду отримання запиту від getUpdate

Цей чат-бот дозволить автоматизувати процес взаємодії з клієнтами компанії, надаючи їм швидкий доступ до необхідної інформації, а також значно покращить обслуговування завдяки інтерактивним та персоналізованим функціям.

2.4 Вибір засобів розробки

Для реалізації даної роботи будуть використовуватися наступні мови та технології для створення чат-боту :

2.4.1 Python

Python — це інтерпретована мова програмування, яку створив у 1991 році голландський програміст Гвідо ван Россум. Це означає, що Python використовує інтерпретатор для безпосереднього виконання коду, на відміну від компільованих мов, які залежать від машинного коду. Ван Россум прагне

зробити Python настільки зрозумілою і чіткою, як англійська мова. Він також зробив Python мовою з відкритим вихідним кодом, дозволяючи будь-кому внести свій вклад у її розвиток.[16]

«Читабельність» є ключовою у філософії Python. Мова спрямована на зменшення кількості блоків коду та використання відступів для створення більш чіткого та менш перевантаженого коду. Python — це універсальна мова, сумісна з багатьма системами, що робить її використання широко поширеним. Мова Python має такі переваги:

Розробники можуть легко читати і розуміти програми на Python, оскільки мова має базовий синтаксис, схожий на синтаксис англійської.

Python допомагає підвищити продуктивність розробників, оскільки дозволяє писати програми на Python з меншою кількістю рядків коду, ніж іншими мовами.

Python має велику стандартну бібліотеку, що містить багаторазово використовувані коди практично для будь-якого завдання. У результаті розробникам не потрібно писати код з нуля.

Розробники можуть легко поєднувати Python з іншими популярними мовами програмування: Java, C і C++.

Активна спільнота Python складається з мільйонів розробників, які підтримують її, з усього світу. У разі виникнення проблем спільнота допоможе в їх вирішенні.

Крім того, в Інтернеті є безліч корисних ресурсів для вивчення Python. Наприклад, ви можете легко знайти відеоролики, навчальні посібники, документацію та керівництва для розробників.

Python можна переносити на різні операційні системи: Windows, macOS, Linux і Unix.[17]

2.4.2 Python-telegran-bot

Python-telegram-bot – одна з найпопулярніших бібліотек для розробки ботів у Telegram за допомогою мови програмування Python. Вона надає простий і потужний API для інтеграції з Telegram Bot API, що дозволяє розробникам створювати функціональних ботів з багатьма можливостями. [18]

До особливостей Python-telegram-bot можна віднести:

Простота використання: бібліотека має простий синтаксис, що дозволяє будь-кому, навіть новачкам, створювати та налаштовувати ботів ;

Асинхронна підтримка: забезпечує асинхронну підтримку через asyncio для обробки декількох запитів одночасно ;

Повноцінні функції: API має всі функції, які є в Telegram Bot API, з підтримкою обміну повідомленнями з ботами, вірусами та кнопками. Підтримка веб-хуків та довгих опитувань;

Детальна документація та приклади: офіційна документація бібліотеки містить багато прикладів та ілюстрацій, що спрощують використання бібліотеки.

```python
from telegram import Update
from telegram.ext import Application, CommandHandler, MessageHandler, filt

# Функція обробки команди /start
async def start(update: Update, context):
    await update.message.reply_text("Привіт! Я ваш Telegram бот.")

# Функція обробки текстових повідомлень
async def echo(update: Update, context):
    user_message = update.message.text
    await update.message.reply_text(f"Ви сказали: {user_message}")

# Основний блок програми
if __name__ == "__main__":
    # Створення додатка
    application = Application.builder().token("YOUR_BOT_TOKEN").build()

    # Додавання обробників команд і повідомлень
    application.add_handler(CommandHandler("start", start))
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND

    # Запуск бота
    application.run_polling()
```

Русинок 2.2. Приклад створення бота за допомогою Python-telegram-bot

Можливості Python-telegram-bot:

Обробка повідомлень: можливість приймати текстові повідомлення відправку відповідей у вигляді тексту, фото, відео тощо.

Створення клавіатур, важелів і можливість дінамічної зміни їх для інтерактивної взаємодії обробка команд можливість зареєструвати команди які користувач зможе вводити, наприклад, /start, /help, прив'язка певних exemplary дій до команди обробка.

Медіа функціонал же, що й при роботі з повідомленнями пересилання та отримання фото, відео, аудіо, документів зберігання та обробка нових файлів.

вебхуки бібліотека дає можливість працювати з вебхуком для оперативного оброблення запитів, що дозволяє використовувати її для масштабних проєктів.

Масова розсилка реплікації великих кількостей повідомлень списки розсилки адаптуються за допомогою бази даних.

Обробка станів дозволяє легко реалізувати бота зі складними сценаріями, наприклад, форми для заповнення чи багаторівневі меню, Локалізація можливість скласти багатомовного бота.

Переваги Python-telegram-bot:

Простота у використанні.

Регулярні оновлення для підтримки поточного Telegram API.

Велика спільнота розробників та доступність навчальних ресурсів.

Обмеження:

Щоб використовувати web Intercept, у вас повинен бути доступний сервер або хмарна платформа.

Для обробки великих навантажень може знадобитися додаткова оптимізація.

Зокрема, Python-telegram-bot - відмінний вибір для створення Telegram-ботів, від простих інформаційних до складних інтерактивних систем.

2.4.3 Aiogram

Aiogram – це асинхронна бібліотека для створення Telegram-ботів на Python з високою швидкістю та гнучкістю для легко повного розвитку ботів з новими функціями Telegram bot API та зручним синтаксисом. [19]

Основні особливості Aiogram:

Асинхронна архітектура

Асинхронне програмування (asyncio) забезпечуватиме можливість одночасно обробляти багато запитів без зупинки виконання програми.

Еластичність

Підтримування різноманітних сценарій обробки повідомлень.

Оновлений API бота Telegram, який підтримуватиметься.

Aiogram постійно оновлюють та підтримують усі можливості Telegram API і додають нові функціональності такі як взаимодействия кнопки опитування та веб хуки.

Зручний синтаксис

Система обробників з організованою структурою дозволять зручно виділити основну ідею роботи бота за допомогою функцій, команд та інших подій.

Велика група людей та письмові матеріали

Aiogram підтримуваний живою спільнотою людей, яка готова допомогти розробникам у вирішенні їх проблем та питань щодо бібліотечною документацію.

```python
from aiogram import Bot, Dispatcher, types
from aiogram.utils import executor

# Токен бота
BOT_TOKEN = "YOUR_BOT_TOKEN"

# Ініціалізація бота та диспетчера
bot = Bot(token=BOT_TOKEN)
dp = Dispatcher(bot)

# Обробник команди /start
@dp.message_handler(commands=['start'])
async def start_command(message: types.Message):
    await message.reply("Привіт! Я Telegram-бот, створений за допомогою A

# Обробник текстових повідомлень
@dp.message_handler()
async def echo_message(message: types.Message):
    await message.reply(f"Ви сказали: {message.text}")

# Запуск бота
if __name__ == "__main__":
    executor.start_polling(dp, skip_updates=True)
```

Рисунок 2.3 Приклад створення бота за допомогою Aiogram

Можливості Aiogram:

- Обробка текстових повідомлень та інструкцій.
- Реєстрація обробників для специфічних команд (наприклад «/start»).
- Робота з будь якими текстовими повідомленнями або медіа.
- Панелі керування та клавіатури
- Створення кнопок, вбудованих у текст та клавіатури.
- Розробка інтерактивних меню з задуманою послідовністю дій.
- Взаємодія з медіа
- Обмін фотографіями та відеофайлами або аудіозаписами та документами.
- Перегрузка і зберігання даних.
- Фільтри для обробки
- Використання заздалегідь визначених або індивідуальних фільтрів для обробки конкретних подій.
- Робота з вебхуками
- Налаштування вебхуків для ефективного оброблення запитів.
- Підтримка кінцевого автомата (FSM)

- Виконання сценарій із використанням станів (наприклад: складні форми або опитування).
- Обробка запитів зворотнього виклику
- Взаимодействувати з інтерактивними кнопками за допомогою callback даних.
- Розсилка великої кількості повідомлень
- Надсилання повідомлень одразу кільком користувачам.

Переваги Aiogram:

- Простий синтаксис: зручно для новачків і професіоналів.
- Швидка інтеграція: легко впровадити в існуючі проєкти.
- Велика спільнота: багато прикладів, навчальних матеріалів і підтримки.
- Гнучкість: підходить як для невеликих ботів, так і для великих систем.
- Недоліки:
- Потребує базового розуміння асинхронного програмування.

Для використання вебхуків необхідний доступний сервер або хмарна платформа.

Aiogram — чудовий вибір для створення сучасних Telegram-ботів із потужним функціоналом, що відповідає вимогам як малого, так і великого бізнесу.

2.5 Вибір середовища розробки

Visual Studio Code (VS Code) – це редактор коду з великою кількістю функцій, який користуються попитом серед розробників програмного забезпечення і що розробляють у компанія Microsoft; він доступний для використання на різних операційних системах - Windows, macOS та Linux (дивись рисунок 2.4). [20]

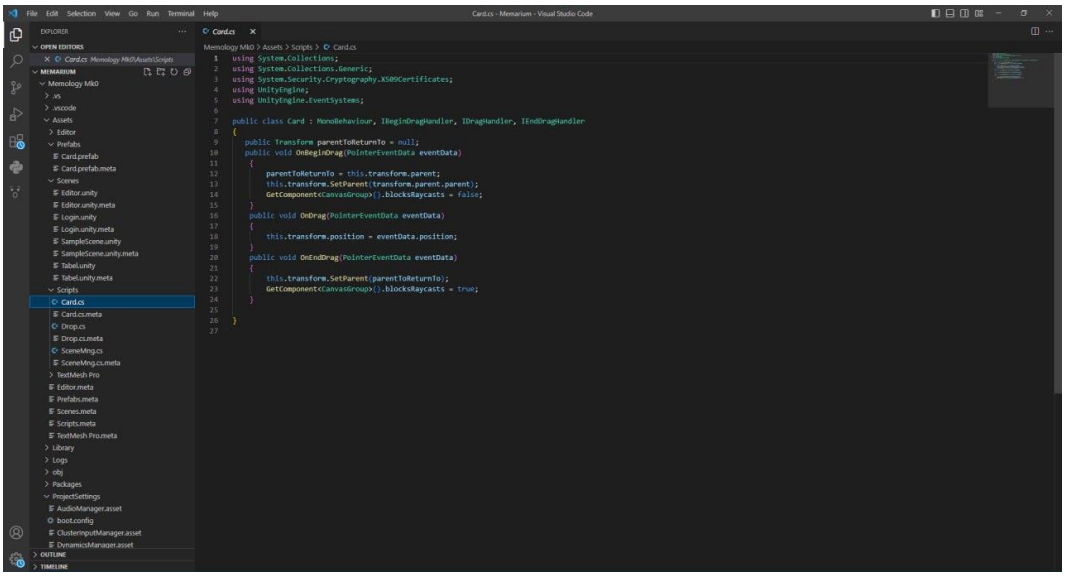


Рис. 2.4. Інтерфейс Visual Studio Code

Visual Studio Code був створений корпорацію Microsoft і вийшов у світ у травні 2015 року с великим успіхом серед програмістів завдяки його функціональності та простоти використання на будь-якій платформі. За допомогою Visual Studio Code ви можете розробляти різноманітні види програм - веб-, мобільні та хмарні додатки.

Visual Studio Code підтримувати розширення для різних мов програмування і дають можливість отримати додатковий функціонал для програмування на певній мові програмування. Крім цього, у нього є вбудована підтримка для систем керування версіями таких як Git для зручності роботи з змінами в коді та командами Git.

Visual Studio Code мають різноманітні корисні функціональності. Наприклад, воно підтримує розширення для покращення продуктивності, надаються автоматичні підказки для завершення коду збудована консоль для запуску команд та відлагодження коду і можливостей налаштування параметрів для зручно прцботи з кодом. Крім цього Visual Studio Code є безкоштовним та має вихідний код доступний до зміни та вдосконалення за допомогою внесеням своіх корекцій у його вихідного розпоряджения.

Переваги можуть бути в такому вигляді:

- Легкість використання юзабіліті коду Visual Studio демонструють інтуітивний та простий у засвоюванні інтерфейси редактора для швидкого старту роботи з ним.
- VS Code мають різноманітні можливості - воно підтриму юе декілька мов програмування та фреймворків і мають усілякі корисні вбудовані функці ї, так і як можливост і для налагодження, автоматичне доповнення коду, керування верс і ями та багато іншого .
- VS Code мае велику кількість додатків, які дозволяють налаштувати редактор під свою власну потребу.
- Підтримка Git: У Visual Studio Code вбудована підтримка Git для управління версіями та спільною роботою над проектами.
- VS Code доступний для використання безкоштовно і дозволяють заощаджувати кошти на ліцензіях.

Недоліки:

- Вимоги до системи для VS Code ставлять високі вимоги до потужності комп’ютера, зокрема у разі розробки великих проектів.
- Недостатньою швидкістю роботи може володіти Visual Studio Code у порівнянні з іншими інтегрованими середовищами розробки (IDE), особливо якщо відкрито велику кількість файлів.
- Деякі функціональності можуть бути важкозрозумілими для користувачів - інтерфейс VS Code складний для початківців і деякі можливості можуть бути неочевидними.
- Складно налаштувати програмувальне середовище VS Code через те, що іноді процес налаштування може бути складним і забирати багато часу, особливо якщо користувач маха кількома параметрами для зміни.
- Обмежені можливості підтримки мов програмування в VS Code означають те, що деякі мови нині не підтримуються або їх підтримка може бути обмеженою.

Висновок до розділу 2

У даному розділі визначено призначення чат-боту, основні вимоги до нього та спроектовано його функціональні можливості. Чат-бот розробляється для автоматизації відповіді на запити користувачів, що підвищує ефективність бізнес-процесів. Вимоги до проекту включають зручність, стабільність роботи та інтеграцію з Telegram.[21]

Для реалізації обрано бібліотеку Aiogram, яка забезпечує асинхронність і функціональність, а також середовище розробки Visual Studio Code, що надає зручні інструменти для роботи з Python. Проведені дослідження та вибір засобів розробки створюють основу для практичної реалізації проекту.

РОЗДІЛ 3. РОЗРОБКА ЧАТ-БОТУ

3.1. Налаштування проекту

3.1.1.Створення робочого середовища

Після вибору технологій та програмних засобів для втілення проекту можна розпочати практичну частину - створення та налаштування бота. Першим кроком є реєстрація бота. Для цього потрібно у пошуковому рядку Telegram ввести @BotFather і перейти до діалогу з цим сервісним ботом (рис. 3.1

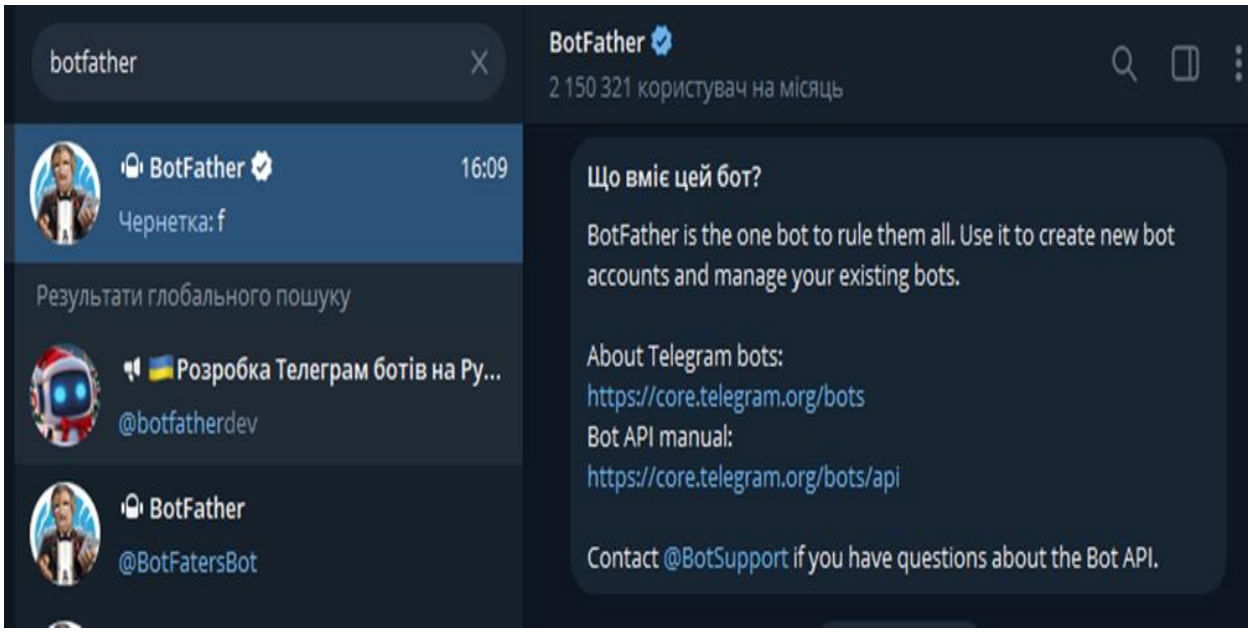


Рисунок 3.1 – Початок реєстрації бота в Telegram

Потрібно натиснути на команду “/start", щоб активувати можливості BotFather. Після цього сервіс висиласть меню з розгорнутим переліком допомог автоматизоване виконання операцій. (рис. 3.2)

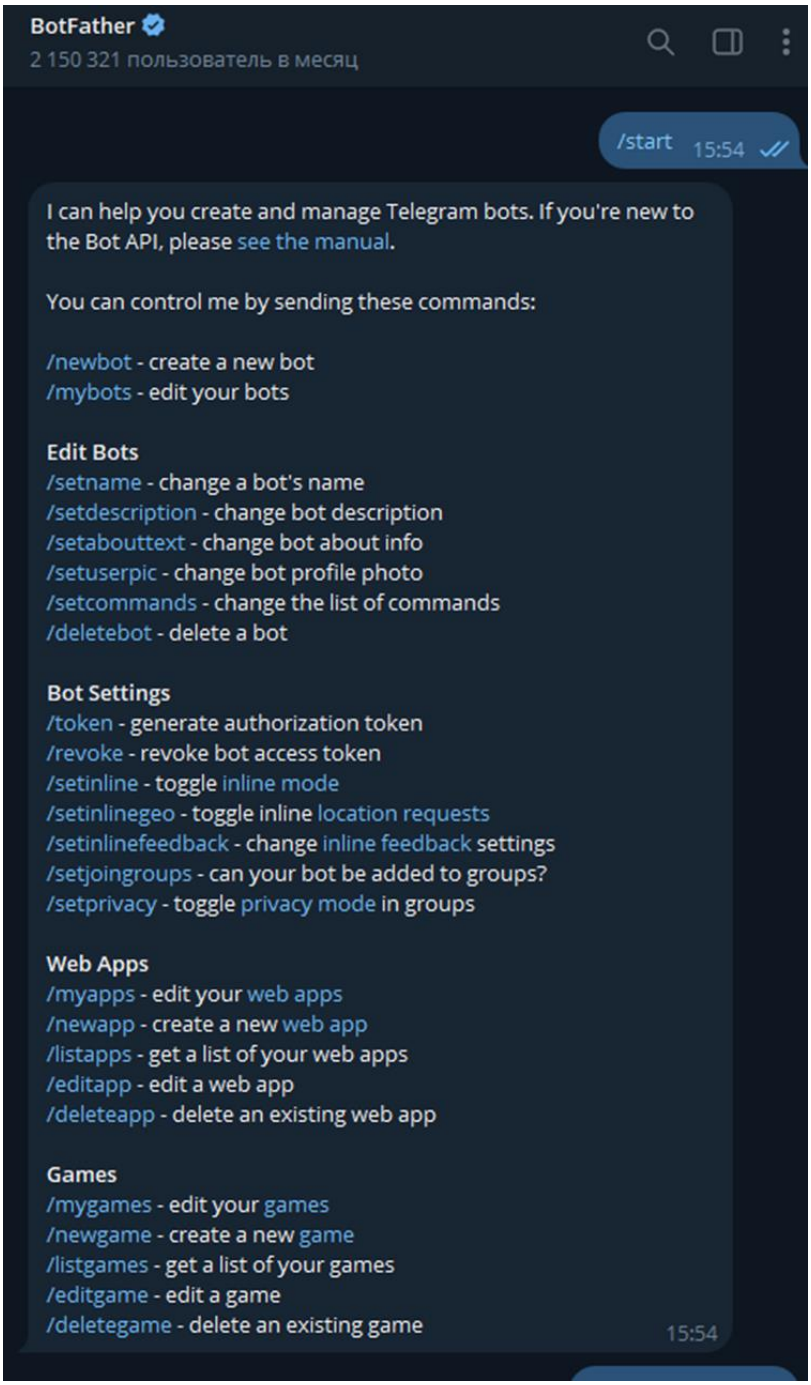


Рисунок 3.2 – Список функцій, доступних у BotFather

Для створення нового бота слід ввести команду «*/newbot*». Далі BotFather запитає назву бота. Назва повинна бути зрозумілою для користувачів таким чином, щоб вони могли розуміти призначення чат-бота. Важливо зауважити, що це лише назва з інформаційною метою і не впливає на функціональність.

Наступний крок полягасть у виборі особливого імені для бота, яке буде унікальним для користувача. Це ім’я не повинно повторюватися серед інших

ботів у Telegram. Воно має містити лише латинські літери, цифри або символи підкреслення (_). Обов’язковою умовою є закінчення імені на слово bot (наприклад, myhelperbot або assistant_bot).

Після завершення цих етапів BotFather надсилають повідомлення про успішну реєстрацію бота, в якому надають токен доступу до API та посилання на бота в Telegram (рисунок 3.3). Токен є унікальним ключем, який дозволяють взаємодіяти з Telegram Bot API та виконувати всі наступні дії.

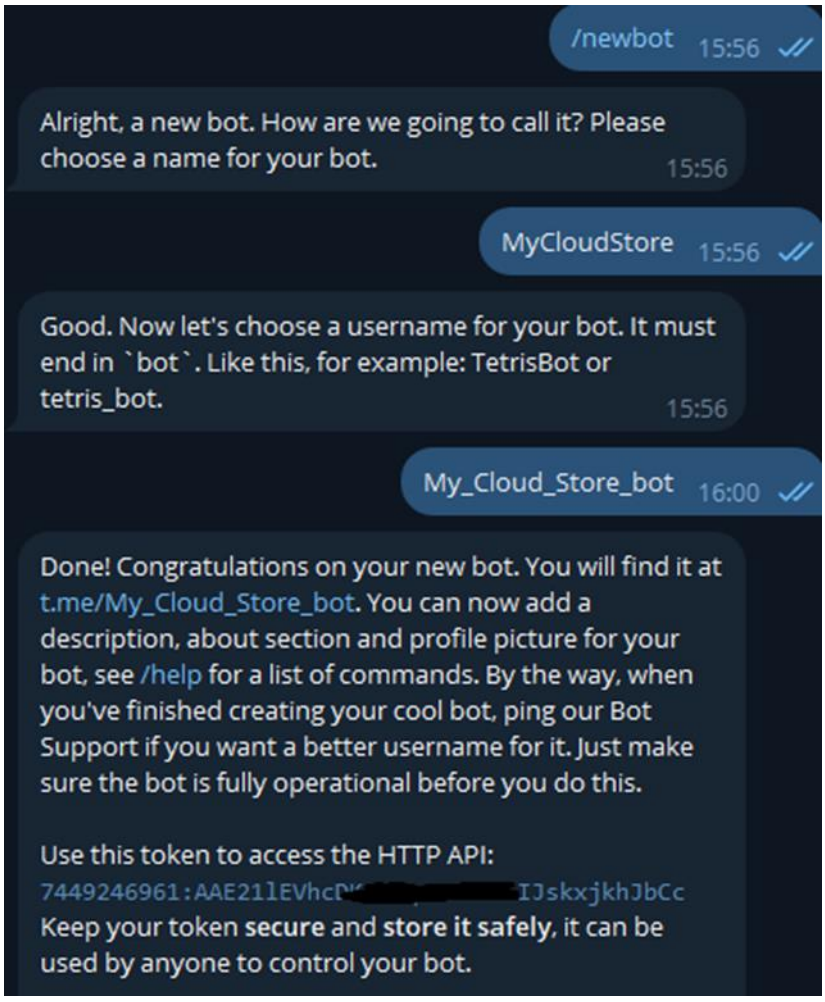


Рисунок 3.3 - Повідомлення щодо успішно проведеною реєстрацію бота з доступним токеном та посиланням

Після того, як закінчилась реєстрації можна перейти за наданим посиланням і побачити свого нового бота. На даному етапі він ще не має

функціоналу і виглядає як «чистий аркуш», що дає безліч можливостей для подальших експериментів та доповнень. У вікні чату з ботом поки що немає жодних команд або відповідей (рис. 3.4).

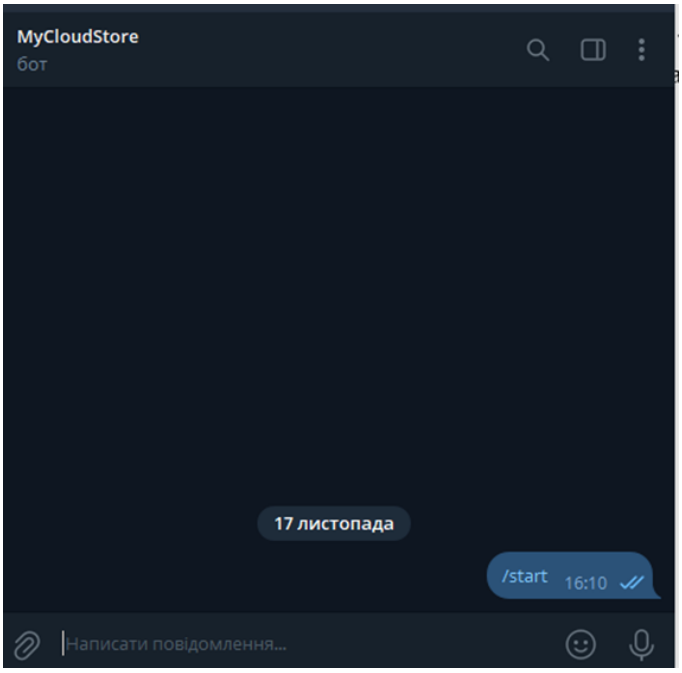


Рисунок 3.4 – Вікно чату з новоствореним ботом

При натисканні на зображення профілю бота ви можете переглянути інформацію про нього. Вона виглядачиме майже так само як профіль звичайного користувача Telegram (рис. 3.5).

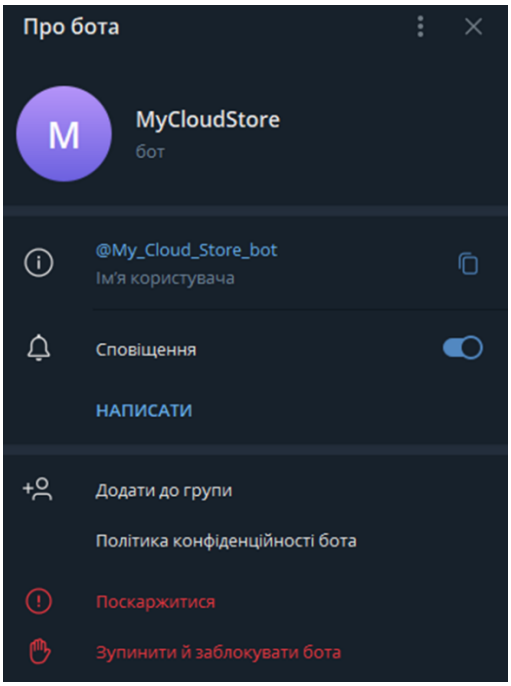


Рисунок 3.5 – Вікно інформації про профіль бота

Для початку взаємодії з ботом треба надіслати команду /start у чаті. Однак наразі бот не відреагуватиме через відсутність програмування для обробки HTTP-запитів на цьому етапі. Це можливо буде лише після створення відповідного коду, який дозволить ботові виконувати запити користувача.

Зазначене вікно чату є інтерфейсом для роботи з ботом. Усі функції та відповіді, які буде реалізовано в коді, відображатимуться в цьому чаті. Усі можливості та відповіді програми будуть відображені у цьому чаті. Це можуть бути текстові повідомлення, медіафайли – зображення, інтерактивні кнопки чи інші елементи для взаємодії з користувачем. Telegram надасть лише платформу для передач даних, а обробка запитань та генераця відповідей буде виконуватися через API Telegram Bot Запити будуть пересилатися через протокол HTTPS у такому форматі:

https://api.telegram.org/bot<токен>/назва_методу

Для оновлення зовнішнього вигляду бота, наприклад, зміни його аватарки, необхідно знову звернутися до BotFather. За допомогою команди /setuserpic можна надіслати в чат зображення, яке стане новою іконкою бота (рис. 3.6).[24]

Рисунок 3.6 – Процес оновлення аватарки бота

Після цього зміни вступають у силу, і в профілі бота відображається нове зображення (рис. 3.7). Правильний дизайн і привабливий вигляд бота можуть стати ключовими факторами у залученні користувачів. Вони зацікавляться, принаймні для того, щоб відкрити бота і розпочати діалог.

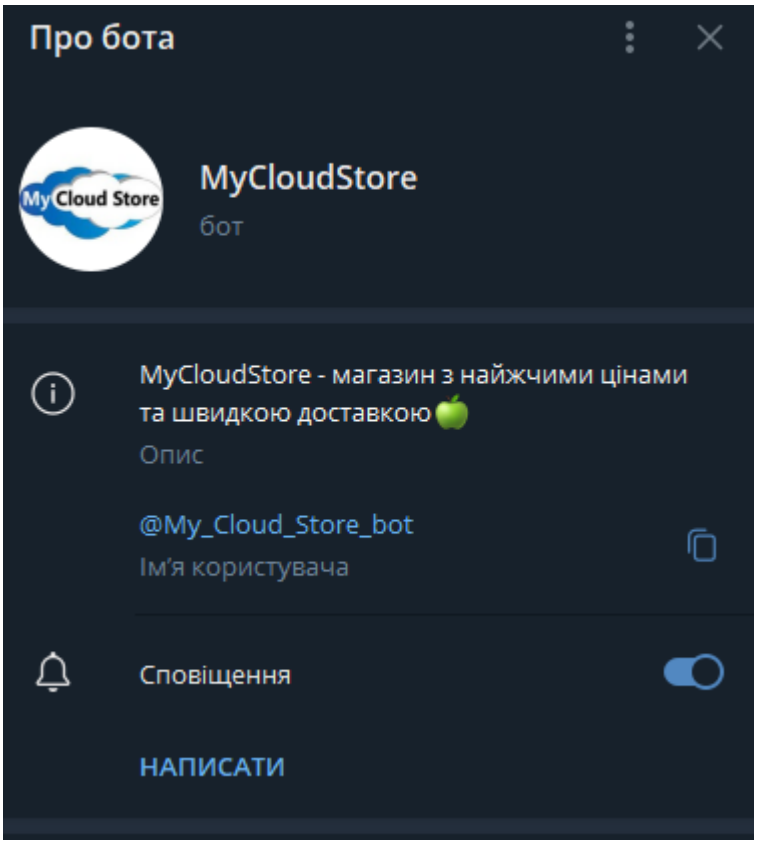


Рисунок 3.7 – Оновлений профіль бота з новою іконкою та описом

Наступним кроком буде налаштування середовища розробки та підключення усіх обраних ресурсів для комфортного розроблення. На початку встановимо останню версію Python на комп'ютер. Завантажуючи інсталятор з офіційного сайту потрібно звернути увагу на сумісність з операційною системою, а також на стабільність версії. Найбільш новою і самою стабільною є версія 3.9.5, вона не підтримується на Windows 7 та більш старі версії. Тому вона не підходить для виконання завдання. Одночасно з нею була викладена версія 3.8.10, яка є останнім регулярним випуском. Встановлено її (рис. 3.8). Починаючи з цього моменту гілка 3.8 буде лише приймати виправлення безпеки, та буде проводитися до кінця 2024 року. Перед встановленням нової версії продукту обов'язково потрібно ознайомитися з новими можливостями та виправленнями, іноді це може спростити написання коду.

Звернувши увагу та обравши стабільний реліз встановлюється оптимальна версія. Після запуску інсталятора, обираємо кнопку Upgrade Now та очікуємо повного встановлення (рис. 3.9).

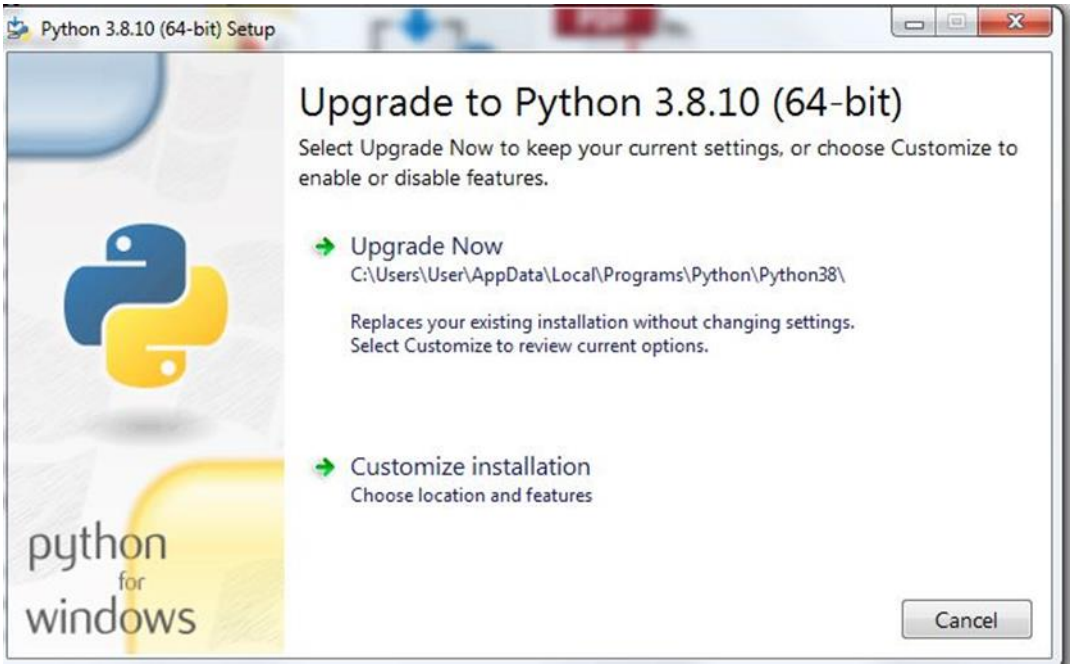


Рисунок 3.9 – Встановлення оптимальної версії Python

Важливим етапом розробки є налаштування середовища розробки. У цьому проєкті використовується Visual Studio Code (VS Code), що є популярним і функціональним редактором коду для роботи з Python. Після встановлення VS Code необхідно створити новий проєкт.

Для початку відкрийте Visual Studio Code і створіть нову папку для вашого проєкту (рис. 3.10). Перейдіть до меню File → Open Folder, виберіть створену папку, яка слугуватиме основним каталогом вашого проєкту. Потім рекомендується налаштувати віртуальне середовище для роботи, оскільки воно дозволяє ізолювати проєкт та його залежності від глобального середовища Python.

У Python доступні такі віртуальні середовища:

- virtualenv – традиційний інструмент для створення ізольованих середовищ. Після створення середовища необхідно використовувати команду pip для встановлення необхідних бібліотек, а для збереження їх списку – pip freeze.
- pipenv – удосконалене віртуальне середовище, яке дозволяє автоматично створювати та зберігати список залежностей у файлах Pipfile та Pipfile.lock. Це спрощує повторне використання середовища на інших пристроях.
- conda – середовище, яке орієнтоване на наукові обчислення, зокрема на роботу з бібліотеками NumPy та SciPy, які використовуються для складних математичних розрахунків.

Для налаштування середовища у Visual Studio Code потрібно виконати наступні кроки:

Відкрийте вбудований термінал VS Code за допомогою комбінації клавіш Ctrl + ~.

У терміналі виконайте команду:

- *pip install pipenv*

Це встановить pipenv, який є найзручнішим вибором для управління залежностями та створення віртуального середовища.

У тому самому терміналі перейдіть до папки вашого проєкту:

- *cd шлях_до_папки*

та створіть віртуальне середовище командою:

- *pipenv install*

Після виконання цієї команди pipenv автоматично створить необхідні файли конфігурації (Pipfile) та встановить Python-інтерпретатор.

Щоб активувати створене середовище, виконайте:

- pipenv shell

Це дозволить працювати з бібліотеками та залежностями безпосередньо в ізольованому середовищі.

У VS Code необхідно вибрати Python-інтерпретатор, що використовується у вашому віртуальному середовищі. Для цього натисніть Ctrl + Shift + P, введіть Python: Select Interpreter і виберіть відповідне середовище.

Перевагою використання pipenv є автоматичне збереження списку залежностей, що спрощує їх подальшу інсталяцію. Це також забезпечує стабільність роботи проєкту та уникнення конфліктів між бібліотеками. Таким чином, середовище pipenv є оптимальним вибором для розробки проєктів у Python.

Рисунок 3.10 – Налаштування проекту в середовищі розробки

Обрано базовий інтерпретатор. За замовчуванням при створенні проєкту середовище розробки пропонує останню доступну версію інтерпретатора Python, встановлену на комп'ютері. Саме для цього попередньо було інстальовано Python.

Інтерпретатор — це програма, яка виконує код, написаний мовою Python. Вона читає та обробляє початковий код, починаючи з першого рядка, і поступово перетворює його в машинний код для виконання комп'ютером.

Основне завдання інтерпретатора — забезпечити виконання логіки програми шляхом постійної обробки коду під час його виконання.

Програми високого рівня зазвичай пишуться на мовах, які потім компілюються або інтерпретуються для виконання процесором. Якщо використовується компілятор, перед виконанням програми весь початковий код компілюється в один файл, що може зайняти багато часу у випадку великих проєктів. Інтерпретатор, навпаки, обробляє лише ту частину коду, яка виконується в конкретний момент. Завдяки цьому час виконання інтерпретованих програм значно скорочується, що є важливою перевагою. Принцип роботи інтерпретатора відображено в спеціальних інструментах для створення діаграм.

Рисунок 3.11 – Принцип роботи інтерпретатора

Pip — це інструмент для управління пакетами та бібліотеками Python. Він дозволяє встановлювати, оновлювати або видаляти пакети, необхідні для роботи з Python-проєктами. Починаючи з версії Python 3.4, pip автоматично встановлюється разом з інтерпретатором. Для обраного інтерпретатора Python 3.8.10 додаткове встановлення pip не потрібне.

Основні команди pip:

- pip help — виводить довідкову інформацію про доступні команди;
- pip list — показує список усіх встановлених пакетів;
- pip search <назва_пакета> — пошук бібліотеки за назвою;

- pip install <назва_пакета> — встановлення нового пакета;
- pip uninstall <назва_пакета> — видалення бібліотеки;
- pip show <назва_пакета> — виводить детальну інформацію про обраний пакет;
- pip install -U <назва_пакета> — оновлення пакета до останньої версії.

На початку роботи з проєктом було відкрито термінал у середовищі розробки Visual Studio Code. Термінал — це текстовий інтерфейс для взаємодії з операційною системою або програмами. Хоча для завантаження бібліотек можна використовувати стандартний командний рядок операційної системи, термінал у VS Code пропонує ширший функціонал, зокрема інтеграцію з Python.

Для встановлення бібліотеки Aiogram, яка призначена для створення асинхронних Telegram-ботів, необхідно виконати наступну команду в терміналі "*pip install aiogram*"

Бібліотека Aiogram забезпечує зручний асинхронний інтерфейс для роботи з Telegram Bot API та є потужним інструментом для створення ботів. Інсталяція Aiogram у проєкт дозволяє використовувати сучасні підходи до обробки запитів завдяки асинхронній архітектурі. Встановлення бібліотеки через термінал у середовищі розробки дозволяє легко інтегрувати її в проєкт, а також керувати оновленнями й залежностями, що забезпечує ефективну розробку Telegram-бота.

Підтримання зв'язку з Telegram-додатком реалізується через HTTPS-запити до проміжного сервера. Використання готових бібліотек, створених спільнотою, значно спрощує роботу, дозволяючи уникнути ручного написання HTTP-запитів.

3.1.2.Імпорт модулів

Telegram Bot API підтримує два основні методи отримання оновлень від бота:

Метод getUpdates – дозволяє боту отримувати оновлення шляхом періодичного надсилання запитів до сервера Telegram.

Використання вебхуків (webhooks) – налаштовується для автоматичної передачі оновлень безпосередньо на сервер розробника.

Вхідні оновлення зберігаються на сервері Telegram протягом 24 годин, після чого їх необхідно обробити.

Об’єкт Update представляє будь-яку взаємодію користувача з ботом, яка вважається вхідним оновленням. Така взаємодія може включати надсилання повідомлень, натискання кнопок або виконання команд. У кожному об’єкті Update може бути присутнім лише один із таких параметрів, який залежить від дій користувача.

Ці оновлення надходять щоразу, коли користувач виконує певну дію в інтерфейсі чату з ботом. Завдяки цьому підходу забезпечується безперервний зв’язок між користувачем і ботом, що дозволяє обробляти запити й взаємодії в реальному часі. (таб.3.1).

Таблиця 3.1 – Отримання оновлень в чаті

Поле	Тип	Опис
update_id	Integer	Унікальний ідентифікатор оновлень.
message	Message	Нове вхідне повідомлення будь-якого типу: текст, фото, стікер, і т.д.
inline_query	InlineQuery	Новий інлайн-запит
chosen_inline_result	ChosenInlineResult	Результат будь-якого запиту, що був відправлений користувачем в чат

Метод getUpdates використовується для отримання оновлень через технологію, яка дозволяє зчитувати нові дії в чаті за допомогою довгих запитів

(long polling). Результат виконання таких запитів повертається у вигляді масиву об’єктів типу Update.

Дані об’єкти передають інформацію про взаємодії з ботом, зокрема ідентифікатор користувача чи бота, а також їхні імена у чаті. Це дозволяє здійснювати керування взаємодією між користувачем і ботом через чат.

Таблиця 3.2 – Команди getUpdates методу

Поле	Тип	Опис
offset	Integer	Ідентифікатор першого повернутого оновлення. Повинен бути більшим ніж самий великий ідентифікатор в попередніх оновленнях.
limit	Integer	Ліміт кількості оновлень. Дозволені числа від 1 до 100.
timeout	Integer	Перерва до запиту в секундах. За замовчуванням 0

Усі типи даних, що використовуються у модулі, представлені у вигляді JSON-об’єктів. На (рис.3.12–3.14) наведені приклади ключових об’єктів, які використовуються для керування ботом.

User

Цей об'єкт зображає бота чи користувача Telegram.

Поле	Тип	Опис
id	Integer	Ідентифікатор користувача чи бота
first_name	String	Ім'я бота
username	String	. Username користувача чи бота

Рисунок 3.12 – Об’єкт User і доступні команди

Цей об'єкт дозволяє отримати інформацію про ідентифікатор користувача або бота, а також їхні імена в чаті. Варто зазначити, що всі об'єкти, які залежать від регістра символів, мають бути представлені у форматі кодування UTF-8.

Chat

Об'єкт чат - інтерфейс зв'язку між ботом та користувачем.

Поле	Тип	Опис
id	Integer	Ідентифікатор чату. Абсолютне значення не перевищує 1e13
type	Enum	Тип чату: “private”, “group”, “supergroup” чи “channel”
title	String	Назва для канала чи групи
first_name	String	. Ім’я користувача в чаті
last_name	String	Прізвище користувача в чаті
all_members_are_administrators	Boolean	.True, якщо усі користувачі являються адміністраторами

Рисунок 3.13 – Об’єкт Chat і доступні команди

Даний об'єкт надає інструменти для керування інтерфейсом взаємодії між ботом і користувачем, тобто чатом. За допомогою цих команд можна виконувати такі дії, як змінення типу чату, його опису, визначення імені користувача в чаті тощо.

Message

Об'єкт повідомлень у чаті.

Поле	Тип	Опис
message_id	Integer	Ідентифікатор повідомлення
from	User	Відправник повідомлення
date	Integer	Дата відправки повідомлення
chat	Chat	Діалог в якому було відправлено повідомлення
forward_from	User	Хто є відправником оригінального повідомлення, у разі пересланих повідомлень
forward_date	Integer	Дата відправки оригінального повідомлення, у разі пересланих повідомлень
text	String	Для текстових повідомлень: текст повідомлення, 0-4096 символів
contact	Contact	Інформація про відправлений контакт
location	Location	Інформація про місцезнаходження
group_chat_created	True	Повідомлення про створення групового чату
pinned_message	Message	Вказане повідомлення було прикріплено до шапки чату

Рисунок 3.14 – Об'єкт Message і досутпні команди

3.1.3 Підключення бібліотеки Aiogram

Існує кілька популярних бібліотек для реалізації поставленої задачі:

- aiogram;
- telebot;
- python-telegram-bot.

Для простих ботів, які не потребують обробки великого трафіку, складних інтеграцій або взаємодії зі сторонніми ресурсами, оптимальним вибором є пакет telebot. Ця бібліотека вирізняється простотою використання і повністю відповідає вимогам для виконання основних завдань. Усі зазначені бібліотеки базуються на Telegram API і мають схожу структуру. Однак кожна з них має свої переваги. Наприклад, aiogram є асинхронною бібліотекою, що робить її ідеальною для створення складних проектів із додатковими інтеграціями. Процес встановлення цієї бібліотеки показано на (рис. 3.15).

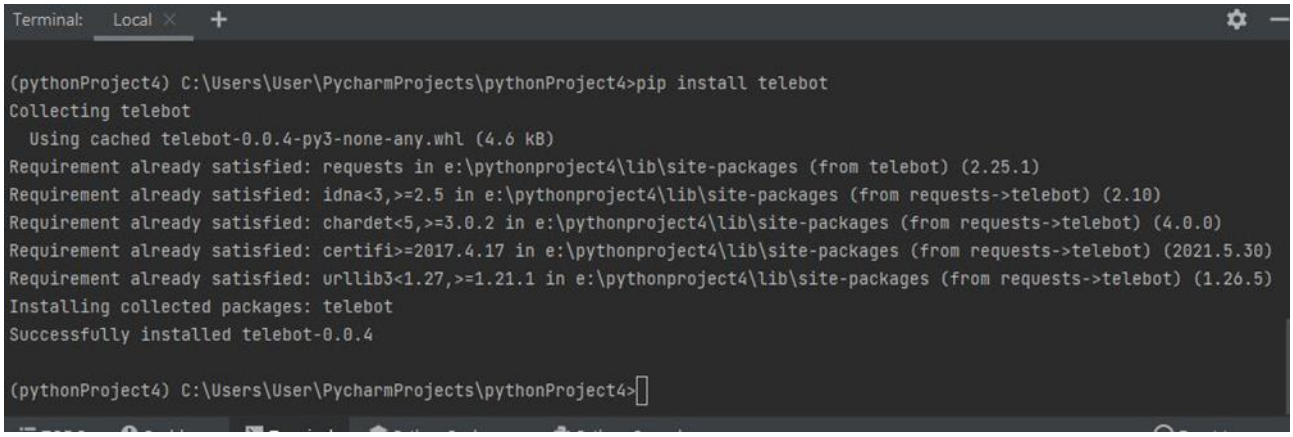


Рисунок 3.15 – Встановлення бібліотеки у папку проекту

Після завершення налаштування робочого середовища та додавання необхідних бібліотек, створено основний файл із розширенням main.py**.** У цьому файлі виконано імпорт основних бібліотек і модулів. Це дозволяє забезпечити коректну роботу їх компонентів і команд у процесі написання коду, а також уникнути появи помилок у середовищі розробки. Процес імпорту модулів здійснюється безпосередньо у вікні програми шляхом написання відповідного коду (рис. 3.16).

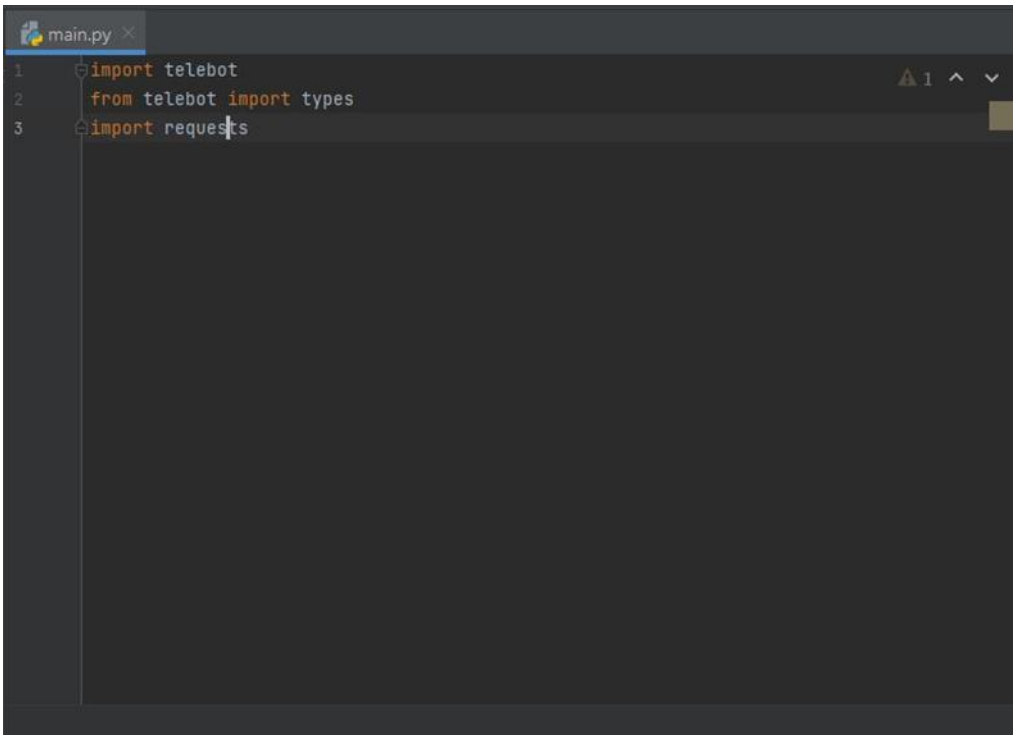


Рисунок 3.16 – Імпортування бібліотек у код проекту

3.2. Реалізаця проекту

Щоб бот почав реагувати на повідомлення у чаті та пропонувати свої послуги, необхідно створити команду «/start», після виконання якої бот запропонує меню з варіантами вибору інформації.

Перед цим у код програми додається токен доступу, який було отримано раніше від Bot Father. Це дозволяє впевнитися, що саме цей бот отримує і виконує команди, згенеровані в програмі.

Після додавання токена можна звертатися до бота безпосередньо, використовуючи команди, доступні в завантажених до проєкту бібліотеках (рис. 3.17).

```
bot = Bot(token="7449246961:AAE21lEVhcDKH3fqw9tWoX6IJskxjkhJbCc")
```

Рисунок 3.17 – Команда на додавання токену у програму

Визначивши, як працює команда «/start», наступним кроком стало формування загальної картини роботи бота. Після аналізу можливих варіантів оформлення інтерфейсу чат- вікна було прийнято рішення додати кнопки для зручного переходу між різними блоками інформації. Це дозволить створити ефективну навігацію по боту, покращуючи взаємодію з користувачем. Для початку необхідно розібратися з відмінностями між типами кнопок в додатку Telegram.

Reply Keyboard Markup – це шаблони повідомлень, через які бот задає питання користувачу у вигляді тексту та пропонує варіанти відповідей для подальшого переходу між блоками бота. Користувач може як самостійно вводити текст на клавіатурі, так і натискати на готові кнопки в інтерфейсі бота. Цей тип клавіатури не містить жодної додаткової інформації. При натисканні на кнопку, буде відправлено тільки той текст, який написаний на самій кнопці. Приклад створення такої кнопки наведено нижче (рис. 3.18).

```
async def cmd_start(message: types.Message):
    kb = [
        [
            types.KeyboardButton(text="Запросить контакт", request_contact=True)
        ]
    ]
    keyboard = types.ReplyKeyboardMarkup(
        keyboard=kb
    )
```

Рисунок 3.18 – Приклад написання шаблонних кнопок

У цьому випадку наведено приклад створення шаблонної кнопки за допомогою іншої бібліотеки для роботи з Telegram, але сутність процесу залишається незмінною. Така кнопка лише виводить у чат текст, який на ній вказано, і в основному використовується в простих чат-ботах, основною метою яких є виконання базових функцій, таких як взаємодія з користувачем (рис. 3.19).

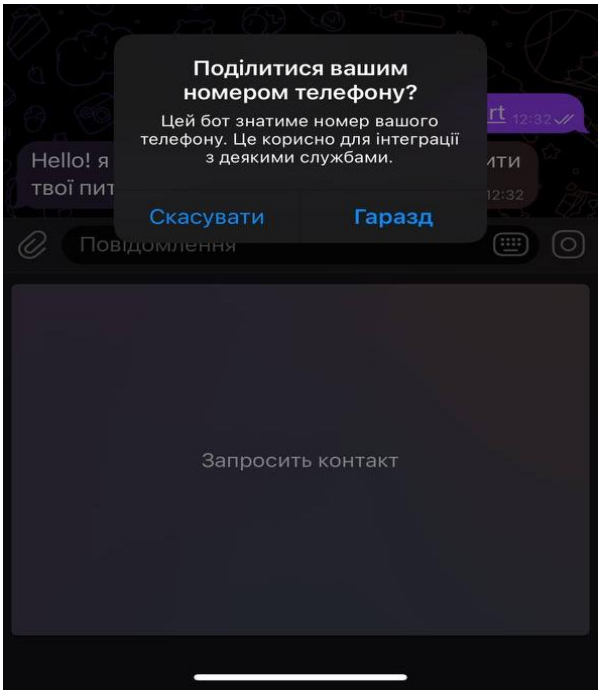


Рисунок 3.19 – Зовнішній вигляд шаблонної кнопки

Шаблонна кнопка розташована в меню керування і виглядає як звичайна клавіатура, якою користувачі зазвичай пишуть повідомлення друзям або шукають статті в Інтернеті.

Inline Keyboard Markup – це клавіатура з широкими можливостями для взаємодії. Вона дозволяє виконувати значно складніші функції, ніж звичайна клавіатура. Кнопки в цьому форматі прив'язані до конкретного повідомлення або команди, з якими вони були надіслані. Функціонал таких кнопок дає змогу переходити за посиланнями в Telegram, відвідувати зовнішні ресурси, заповнювати форми для опитувань, а також реалізовувати багато інших можливостей. Приклад створення такої кнопки представлений нижче (рис. 3.20).

```
@dp.callback_query(F.data == "manager")
async def cmd_manager(callback: types.CallbackQuery):
    kb = [
        [
            types.InlineKeyboardButton(text="Головне меню", callback_data="menu"),
        ],
    ]
    keyboard = types.InlineKeyboardMarkup(
        inline_keyboard=kb,
    )
    await callback.message.answer("Напишіть ваше питання", reply_markup=keyboard)
```

Рисунок 3.20 – Приклад коду inline-кнопок

Назва методу створення таких кнопок вже вказує на те, що кнопка розташована безпосередньо в чаті бота. Вона має форму заокругленого прямокутника.

Зовнішній вигляд такої кнопки відповідає тому, що зображено на рисунку 3.21.

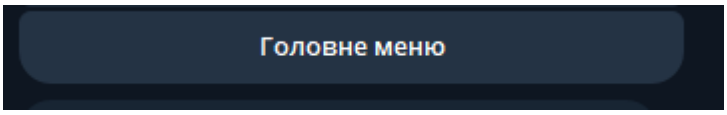


Рисунок 3.21 – Зовнішній вигляд inline-кнопки

Процес вибору програмного забезпечення та методів реалізації був успішно завершений. Наступним кроком було розроблення бота, функціонал якого показано на рисунку 3.22

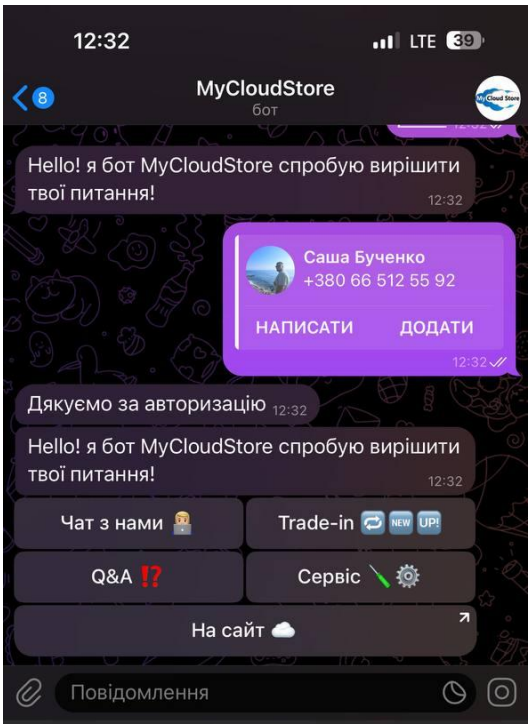


Рисунок 3.22 – Зовнішній вигляд головного меню чат-боту

Чат з нами: Кнопка відкриває можливість для користувача зв'язатися з підтримкою або оператором компанії.

Trade-in: Ця кнопка веде до розділу, де користувач може отримати інформацію про послугу обміну старого товару (техніки) на новий із доплатою. Бот приймає запити на оцінку товару і передає цю інформацію на рисунку 3.23.

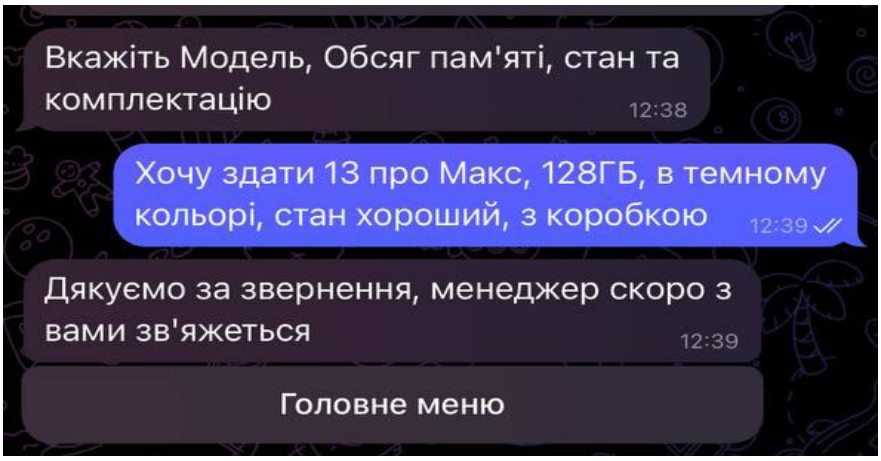


Рисунок 3.23 – Повідомлення, коли клієнт залишає повідомлення на Trade-in

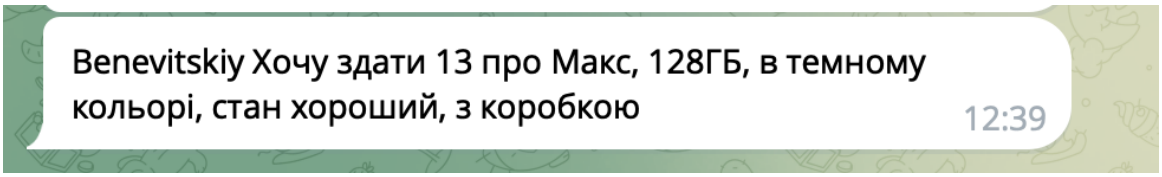


Рисунок 3.24 – Повідомлення, яке отримує менеджер в боті, коли клієнт залишає повідомлення на Trade-in

Q&A: Розділ із часто задаваними питаннями (FAQ). Кнопка дозволяє користувачеві швидко отримати відповіді на популярні питання, щодо доставки, послуг або способів оплати (рис. 3.24 та 3.25)

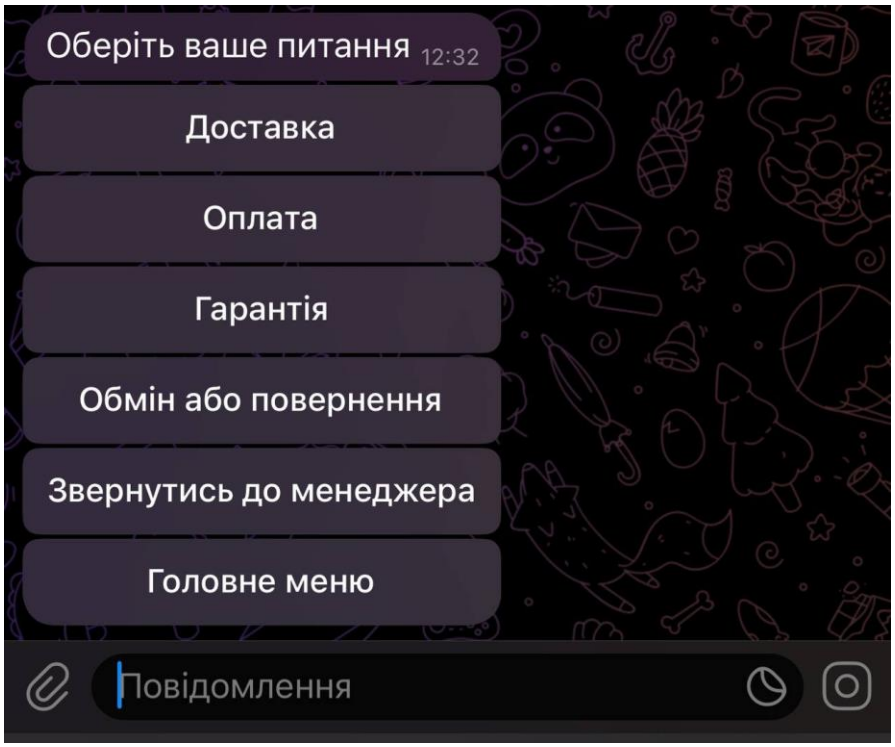


Рисунок 3.24 – Підпункти меню часто задаваними питаннями (FAQ)

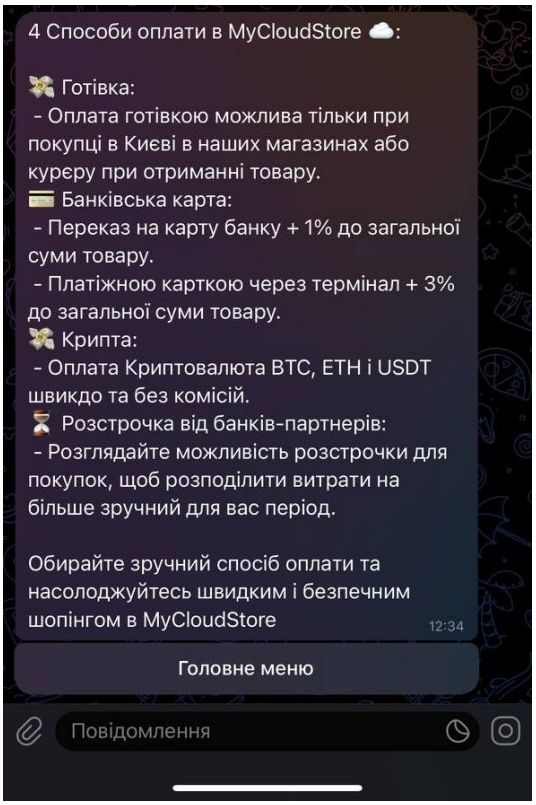


Рисунок 3.25 – Варіант відповіді клієнта

Сервіс: Відповідає за доступ до інформації про послуги сервісного обслуговування. Бот приймає запити стовно питань і передає цю інформацію менеджеру (рис. 3.26).

Рисунок 3.26 – Підпункти меню сервіс

На сайт: Ця кнопка є посиланням, що перенаправляє користувача на головну сторінку офіційного вебсайту компанії.

Висновок до розділу 3

У третьому розділі було детально описано процес розробки чат-бота, зокрема створення робочого середовища, підключення необхідних бібліотек, імпорту модулів і налаштування основного функціоналу бота. Реалізовано команду "/start", яка ініціює взаємодію з користувачами, та створено меню з можливістю отримання відповідей на часті запитання, що забезпечує зручність і простоту використання.

Однією з ключових частин роботи стало налаштування взаємодії з Telegram Bot API за допомогою бібліотеки Aiogram. Ця асинхронна бібліотека забезпечила швидкість і ефективність обробки запитів, що особливо важливо для чат-ботів, які повинні відповідати в реальному часі.

Також у розділі описано роботу з ключем доступу (token), який дозволяє ідентифікувати бота в системі Telegram. Детально розглянуто налаштування середовища розробки та встановлення бібліотек через пакетний менеджер pip, що спрощує управління залежностями.

Результатом розробки є базовий прототип чат-бота, який може бути розширений за рахунок додаткових функцій, таких як кнопки, відповіді на складні запити чи інтеграція зі сторонніми сервісами. Базова структура дозволяє додавати нові методи і забезпечувати масштабування функціоналу бота відповідно до вимог замовника або користувача.

Таким чином, проведена робота дала змогу реалізувати основні функціональні можливості чат-бота, використовуючи сучасні інструменти та технології, а також підготувати платформу для подальшого вдосконалення.

ВИСНОВКИ

В процесі виконання дипломної роботи було здійснено детальний аналіз та розробку чат-бота для використання в інформаційно-пошукових системах, орієнтованих на бізнес-процеси. Було розглянуто широке коло теоретичних, технічних і практичних питань, які забезпечили створення функціонального та ефективного продукту, здатного спростити взаємодію користувачів із бізнесом і зменшити навантаження на персонал.

У першому розділі дипломної роботи було детально вивчено принципи функціонування інформаційно-пошукових систем та їх роль у сучасному бізнесі. Проведено порівняльний аналіз платформ для впровадження чат-ботів, що дало змогу обрати найкращу основу для реалізації поставлених завдань. Розглянуто існуючі аналоги подібних рішень, що дозволило визначити їхні переваги та недоліки, а також сформулювати вимоги до функціональних можливостей чат-бота. Це заклало основу для створення ефективного продукту, що відповідає сучасним бізнес-реаліям.

Другий розділ був присвячений проектуванню і визначенню ключових вимог до чат-бота. Були визначені функціональні можливості, що охоплюють основні сценарії взаємодії користувача з ботом, включно з інформаційним пошуком і обробкою запитів. На основі аналізу вимог було проведено обґрунтований вибір засобів розробки, включно з мовою програмування Python, бібліотекою Aiogram для роботи з Telegram API та середовищем розробки Visual Studio Code. Використання цих інструментів забезпечило високу продуктивність розробки та масштабованість рішення.

Третій розділ дипломної роботи був спрямований на практичну реалізацію проекту. У рамках розробки було створено програмний код, що базується на використанні JSON-формату для обміну даними між ботом і користувачами через протокол HTTPS. Розроблений бот може відповідати на часті запитання, генерувати повідомлення та виконувати запити користувачів через налаштовану систему команд. Використання бібліотеки Aiogram

дозволило реалізувати асинхронну обробку запитів, що забезпечило високу швидкість роботи навіть при великій кількості одночасних користувачів. Особливу увагу було приділено безпеці взаємодії через шифрування даних і безпечне зберігання токена доступу.

Таким чином, дипломна робота досягла поставлених цілей. Розроблений чат-бот відповідає сучасним вимогам у сфері автоматизації бізнес-процесів і забезпечує якісну взаємодію між компаніями та їхніми клієнтами. Використання відкритого API Telegram і популярних бібліотек дозволило створити масштабоване рішення з можливістю інтеграції додаткових функцій у майбутньому.

Результати роботи є корисними для автоматизації рутинних процесів у бізнесі, покращення користувацького досвіду та економії часу і ресурсів компанії. Впровадження подібних рішень у бізнес дозволить підвищити конкурентоспроможність і забезпечити сучасний рівень сервісу для клієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основи створення інформаційних систем : навч. посіб. / А. М. Береза. – 2-ге вид., перероб. і доп. − Київ : Знання, 2018. – 241 с.
2. Інформаційні технології в сучасному бізнесі / Барановський, Л. Б., Київ: Ліра-К, 2020. 23с.
3. Тарасюк О.М., "Основи проєктування інформаційних систем" Видавництво: КНУ імені Тараса Шевченка, 2020. 158с.
4. API to send messages to WhatsApp [Електроний ресурс] URL: https://developers.facebook.com/docs/whatsapp/cloud-api/guides/send-messages/
5. Telegram APIs [Електроний ресурс] URL: https://core.telegram.org/
6. The Messenger Platform Overview details how the platform works and what you need to successfully implement the platform [Електроний ресурс] URL: https://developers.facebook.com/docs/messenger-platform/overview
7. Viber API Documentation [Електроний ресурс] URL: https://developers.viber.com/docs/api/rest-bot-api/#get-started
8. APIs to help developers develop native applications in WeChat [Електроний ресурс] URL: https://developers.weixin.qq.com/miniprogram/en/dev/framework/
9. Lees, D. and LePage, P. (1996). Robots in education: the current state of the art. Journal of Educational Technology Systems, 24 (4), с.299–320.
10. Проскурніна Н.В., Доброскок Ю.Б. Використання цифрових технологій роздрібної торгівлі на прикладі месенджерів та чат-ботів. Економічний розвиток і спадщина Семена Кузнеця : матеріали V Міжнар. наук.- практ. конф., м. Харків, 2020. — 130–131с
11. Проскурніна Н.В. Особливості використання технології чат-ботів у роздрібному бізнесі: тези доповідей міжнар. наук.-практ. конф. : у 2-х ч., м. Харків, 2019. — 129–130с.
12. Довідник Дія.Освіта [Електроний ресурс] URL: https://osvita.diia.gov.ua/vocabulary

13. Mauldin, M.L. (1994). Chatterbots, Tinymuds, and the Turing Test: Entering the Loebner Prize Competition. AAAI, с.16–21.

14. A Pragmatic Guide to Business Process Modelling, Holt, Jon

15. Mark Masse ,"REST API Design Rulebook", 2011 – 69с.

16. В. Ю. Щербань, С. М. Краснитський, Т. І. Астістова, В. М. Яхно. Методи представлення, збереження та аналізу даних інформаційних систем: колективна монографія / – Київ : ТОВ "Фастбінд Україна", 2023, с.276.

17. Книга Програмування мовою Python/ Олексій Васильєв – 1-ше вид, Навчальна книга – Богдан 2019. – 44 с.

18. Python-telegram-bot документація [Електроний ресурс] URL: https://docs.python-telegram-bot.org/en/v21.7/

19. Aiogram is a modern and fully asynchronous framework for Telegram Bot API [Електроний ресурс] URL:https://docs.aiogram.dev/uk-ua/dev-3.x/

20. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers by Bruce Johnson, 2019.— 25с.

21. Антонюк О.В. Розробка telegram-бота за допомогою мови програмування Python і середовища розробки Pycharm. 2019. URL: http://dspace2.regi.rovno.ua:8088/jspui/bitstream/123456789/1808/1/Zbirnyk-12-2019-3-157-162.pdf

22. Programming Telegram Bots: The Ultimate Guide by Nicolas Modrzyk, 2019. — 256с.

23. Mastering API Architecture by James J. Park, 2022. — 234с.

24. Білик, І. Ю. "Телеграм-бот для роботи з курсами валют та відслідковування витрат". Thesis, Чернігів, 2020. [Електроний ресурс] URL: http://ir.stu.cn.ua/123456789/23435

25. Natural Language Processing with Python by Steven Bird, Ewan Klein, and Edward Loper, 2009 — 344с.

26. JSON Book: Easy Learning of JavaScript Standard Object Notation by Steven Keller, 2016. — 23с.

27. Чат-боти як нове покоління каналів комунікації / [Електронний ресурс] URL: https://core.ac.uk/download/pdf/197267488.pdf

28. Що таке чат-боти, для чого їх використовують та яка їхня роль у сучасному світі [Електроний ресурс] URL: https://realpython.com/build-a-chatbot-python-chatterbot/

29. Build a Chatbot With Python [Електроний ресурс] URL: https://mc.today/uk/shho-take-chat-boti/

30. Alammar J., The Illustrated Transformer. Self-published, 2018 – 45с.

ДОДАТОК А. ТЕКСТ ПРОГРАМИ

```
import asyncio

import logging

from aiogram import Bot, Dispatcher, types, methods

from aiogram.filters.command import Command

from aiogram import F

from aiogram import Router

import json

# Включаем логирование, чтобы не пропустить важные сообщения

logging.basicConfig(level=logging.INFO)

# Объект бота

bot = Bot(token="7449246961:AAE21lEVhcDw9tWoX6IJskxjkhJbCc")

# Диспетчер

dp = Dispatcher()

router = Router(name=__name__)

# Id менеджера которому переадресовывают сообщения

id_manager = 478358557

# Хэндлер на команду /start

@dp.message(Command("start"))

async def cmd_start(message: types.Message):

    kb = [ [ types.KeyboardButton(text="Запросить контакт", request_contact=True) ]]

    keyboard = types.ReplyKeyboardMarkup(

        keyboard=kb )

    await message.answer("Hello! я бот MyCloudStore спробую вирішити твої питання!",
reply_markup=keyboard)
```

```
# Хендлери чат
@dp.callback_query(F.data == "chat")
async def cmd_trade(callback: types.CallbackQuery): kb = [
[types.InlineKeyboardButton(text="Зробити замовлення", callback_data="ordermanager")
        ], [
            types.InlineKeyboardButton(text="Інфо по замовленню", callback_data="orderinfo")
        ], [
            types.InlineKeyboardButton(text="Консультація", callback_data="manager")
        ], [
            types.InlineKeyboardButton(text="Головне меню", callback_data="menu")
        ]]
    keyboard = types.InlineKeyboardMarkup(
        inline_keyboard=kb,)
    await callback.message.answer("Оберіть ваше питання", reply_markup=keyboard)
@dp.callback_query(F.data == "ordermanager")
async def cmd_ordermanager(callback: types.CallbackQuery) :
    Mess = await bot.send_message(id_manager, callback.from_user.username + " Хоче зробити замовлення")
    kb = [ [
            types.InlineKeyboardButton(text="Головне меню", callback_data="menu"),
        ],  ]
    keyboard = types.InlineKeyboardMarkup(
        inline_keyboard=kb,)
    await callback.message.answer("Дякуємо за звернення, менеджер скоро з вами зв'яжеться", reply_markup=keyboard)
@dp.callback_query(F.data == "orderinfo")
```

```
async def cmd_orderinfo(callback: types.CallbackQuery): kb = [
 [ types.InlineKeyboardButton(text="Головне меню", callback_data="menu"), ],]
    keyboard = types.InlineKeyboardMarkup(
        inline_keyboard=kb, )
    await callback.message.answer("Введіть номер замовлення", reply_markup=keyboard)
# Хендлери qa
@dp.callback_query(F.data == "qa")
async def cmd_qa(callback: types.CallbackQuery):
    kb = [ [ types.InlineKeyboardButton(text="Доставка", callback_data="qa1") ],
[ types.InlineKeyboardButton(text="Оплата", callback_data="qa2")],
[ types.InlineKeyboardButton(text="Гарантія", callback_data="qa3")],
[types.InlineKeyboardButton(text="Обмін або повернення", callback_data="qa4") ],
[types.InlineKeyboardButton(text="Звернутись до менеджера", callback_data="manager")],
[types.InlineKeyboardButton(text="Головне меню", callback_data="menu")]]
    keyboard = types.InlineKeyboardMarkup( inline_keyboard=kb   )
    await callback.message.answer("Оберіть ваше питання", reply_markup=keyboard)
@dp.callback_query(F.data == "qa1")
async def cmd_qa1(callback: types.CallbackQuery):
    kb = [ [ types.InlineKeyboardButton(text="Головне меню", callback_data="menu"),
        ],  ]
    keyboard = types.InlineKeyboardMarkup(
        inline_keyboard=kb, )
    await callback.message.answer("""
 Нова Пошта:
 - Доставка за 1-2 дні, залежно від місця отримання.
```

- Доставка можлива тільки при 100% передоплаті на карту або рухунок iBAN (+1% комісія). Після передоплати ми вам гарантуємо доставку товару.

- Відправка здійснюється з Києва. Після відправки ви отримуєте смс з кодом декларації Нової Пошти.

- Вартість доставки розраховується за тарифами Нової Пошти і залежить від відстані та обсягів.

□□ Кур'єром по Києву:

Доставка в межах Києва платна - вартість 300 грн.

При покупці пристрою з гарантією 1 рік - доставка безкоштовно або поклейка захисного скла на iPhone !

По Києву доставка більшості товарів здійснюється з 10:00 до 20:00 в рабочі дні.

Зазвичай якщо ви розмістили замовлення до 17:00 — ми доставимо його в той же день.

У будь-якому випадку під час замовлення наші менеджери відразу узгодять з вами зручний час доставки.□""", reply_markup=keyboard)

@dp.callback_query(F.data == "qa2")

async def cmd_qa2(callback: types.CallbackQuery):

    kb = [[types.InlineKeyboardButton(text="Головне меню", callback_data="menu"),   ],]

    keyboard = types.InlineKeyboardMarkup(

        inline_keyboard=kb,)

    await callback.message.answer("""4 Способи оплати в MyCloudStore □□:

□ Готівка:

- Оплата готівкою можлива тільки при покупці в Києві в наших магазинах або курєру при отриманні товару.

□ Банківська карта:

- Переказ на карту банку + 1% до загальної суми товару.

- Платіжною карткою через термінал + 3% до загальної суми товару.

 Крипта:

- Оплата Криптовалюта BTC, ETH і USDT швикдо та без комісій.

 Розстрочка від банків-партнерів:

- Розглядайте можливість розстрочки для покупок, щоб розподілити витрати на більше зручний для вас період.

Обирайте зручний спосіб оплати та насолоджуйтесь швидким і безпечним шопінгом в MyCloudStore """, reply_markup=keyboard)

```
@dp.callback_query(F.data == "qa3")
async def cmd_qa3(callback: types.CallbackQuery):
    kb = [    [
            types.InlineKeyboardButton(text="Головне меню", callback_data="menu"),
        ], ]
    keyboard = types.InlineKeyboardMarkup(
        inline_keyboard=kb,  )
    await callback.message.answer("""Гарантійний сервіс від MyCloudStore:
```

 Технічна Підтримка:

- Наші фахівці готові вам допомогти 24/7. Задавайте питання та отримуйте професійну консультацію.

 Гарантія на Продукцію:

- MyCloudStore надає гарантію на всі ваші покупки, адже ми впевнені в якості нашої продукції. Умови гарантії залежать від обраного пакета послуг та типу товару.

 Обмін та Повернення:

- Ми дбаємо про ваш комфорт, тому пропонуємо зручні опції для обміну чи повернення товарів згідно з правилами гарантії.

 Сервіс та Ремонт:

- Наш сервісний центр завжди готовий прийти вам на допомогу. Ми забезпечимо професійне вирішення будь-яких технічних питань і виконаємо ремонт максимально швидко, щоб ви могли знову насолоджуватися користуванням своїм пристроєм без зайвих затримок.

□□ Персональний Підхід:

- Ми завжди ставимо інтереси клієнтів на перше місце, приділяючи особливу увагу кожному випадку та забезпечуючи індивідуальний підхід до вирішення ваших питань.

MyCloudStore — це ваш надійний партнер, який гарантує високу якість обслуговування та впевненість у кожній покупці! """, reply_markup=keyboard)

@dp.callback_query(F.data == "qa4")

async def cmd_qa4(callback: types.CallbackQuery):

```
kb = [ [
        types.InlineKeyboardButton(text="Головне меню  □", callback_data="menu"),
    ],  ]
keyboard = types.InlineKeyboardMarkup(
    inline_keyboard=kb, )
await callback.message.answer("""□□□ Обмін та Повернення
```

□ 14 днів для обміну чи повернення:

Ви можете скористатися своїм правом протягом 14 днів відповідно до «Закону про захист прав споживача».

□ Умови обміну чи повернення:

Товар повинен бути у належному стані: без слідів використання, збережена повна комплектація, оригінальна упаковка, ярлики та заводське маркування.

□ Право на обмін:

Якщо товар виявився непридатним, ви можете обміняти його на аналогічний.

□ Якщо товар не працює:

Обмін або повернення можливі лише після підтвердження сервісного центру, що умови експлуатації не порушені.

 Де здійснити обмін чи повернення:

-Обмін та повернення проводяться у будь-якому магазині мережі MyCloudStore.com.ua.

 Контактна гаряча лінія:

-Отримайте консультацію за телефоном: 067-210-22-33 або 067-217-22-33.""", reply_markup=keyboard)

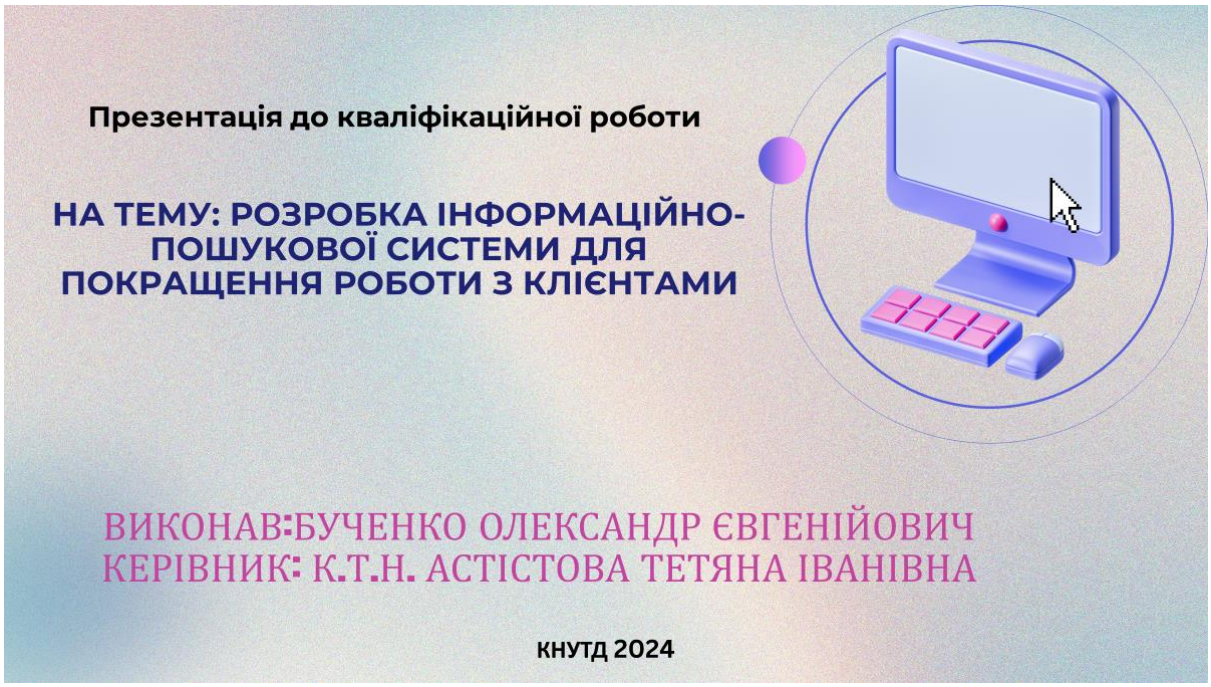
```
# Хэндлер на команду trade
@dp.callback_query(F.data == "trade")
async def cmd_trade(callback: types.CallbackQuery):
    await callback.message.answer("Вкажіть Модель, Обсяг пам'яті, стан та комплектацію")
# Хэндлер на команду service
@dp.callback_query(F.data == "service")
async def cmd_trade(callback: types.CallbackQuery): kb = [
[  types.InlineKeyboardButton(text="Статус звернення", callback_data="statusservice") ],
[ types.InlineKeyboardButton(text="Заявка на сервіс", callback_data="managerservice") ],
[types.InlineKeyboardButton(text="Консультація", callback_data="manager") ],
 [ types.InlineKeyboardButton(text="Головне меню", callback_data="menu")]]
    keyboard = types.InlineKeyboardMarkup( inline_keyboard=kb, )
    await callback.message.answer("Оберіть, що ви хочете дізнатись про сервіс", reply_markup=keyboard)
@dp.callback_query(F.data == "statusservice")
async def cmd_statusservice(callback: types.CallbackQuery) :
    Mess = await bot.send_message(id_manager, callback.from_user.username + " Хоче дізнатись статус сервісного звернення") kb = [
[types.InlineKeyboardButton(text="Головне меню", callback_data="menu"), ], ]
```

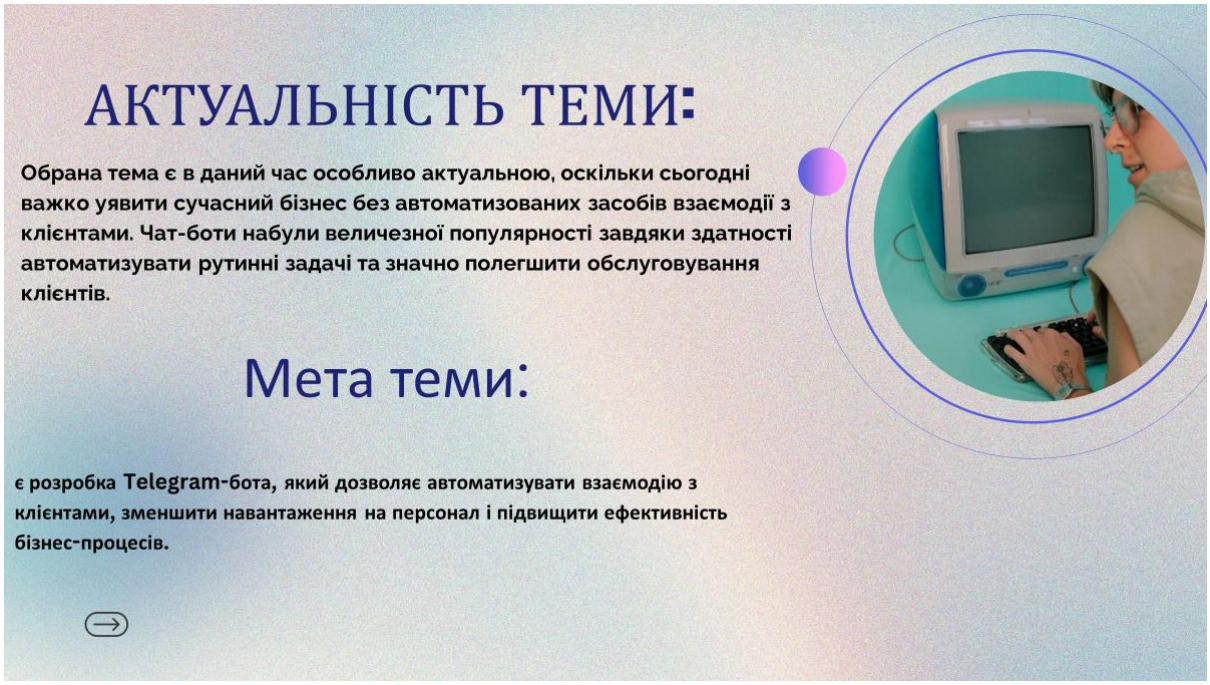
```
    keyboard = types.InlineKeyboardMarkup( inline_keyboard=kb, )
    await callback.message.answer("Дякуємо за звернення, менеджер скоро з вами зв'яжеться", reply_markup=keyboard)
@dp.callback_query(F.data == "managerservice")
async def cmd_managerservice(callback: types.CallbackQuery) :
    Mess = await bot.send_message(id_manager, callback.from_user.username + " Хоче дізнатись статус сервісного звернення")
    kb = [ [  types.InlineKeyboardButton(text="Головне меню", callback_data="menu"), ], ]
    keyboard = types.InlineKeyboardMarkup(
        inline_keyboard=kb, )
    await callback.message.answer("Опишіть ваше сервісне звернення", reply_markup=keyboard)
# Хэндлер на команду manager
@dp.callback_query(F.data == "manager")
async def cmd_manager(callback: types.CallbackQuery):
    kb = [ [ types.InlineKeyboardButton(text="Головне меню", callback_data="menu"),
        ],  ]
    keyboard = types.InlineKeyboardMarkup( inline_keyboard=kb,  )
    await callback.message.answer("Напишіть ваше питання", reply_markup=keyboard)
# Хендлер повернення до меню
@dp.callback_query(F.data == "menu")
async def cmd_menu(callback: types.CallbackQuery):
    kb = [ [ types.InlineKeyboardButton(text="Чат з нами □□□ ", callback_data="chat"),
            types.InlineKeyboardButton(text="Trade-in □□□", callback_data="trade"), ],  [
             types.InlineKeyboardButton(text="Q&A □ ", callback_data="qa"),
```

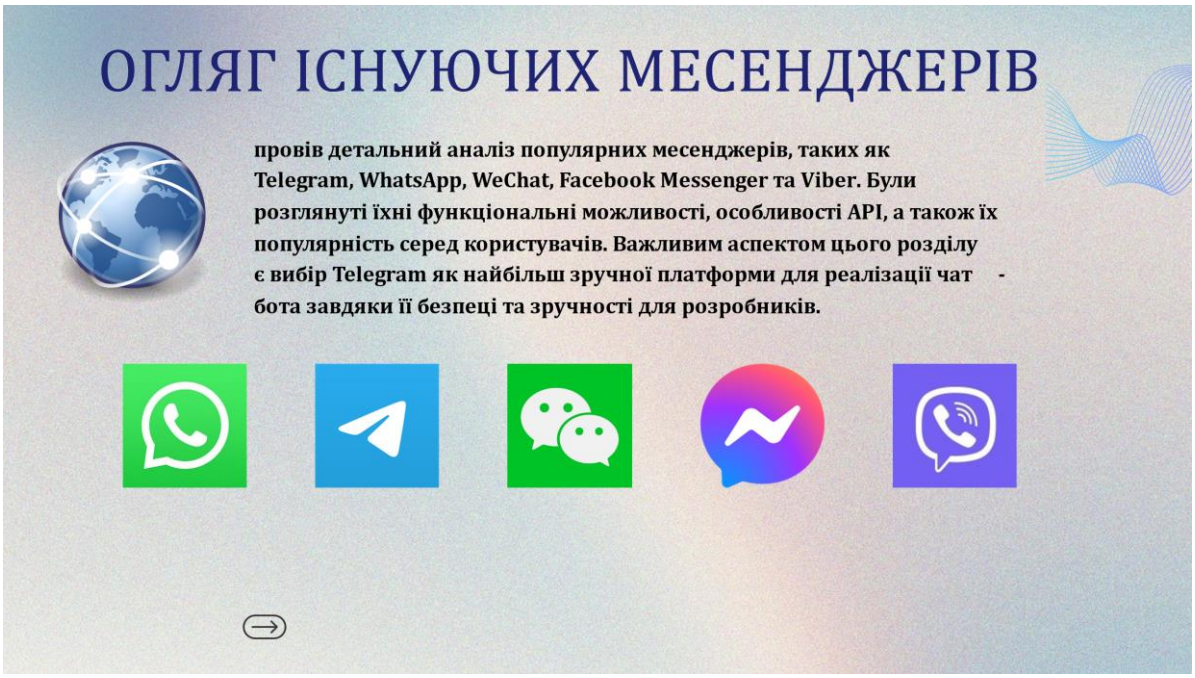
```
            types.InlineKeyboardButton(text="Сервіс □□□", callback_data="service"),
        ], [ types.InlineKeyboardButton(text="На сайт □□", url="https://mcstore.com.ua/")
        ] ]   keyboard = types.InlineKeyboardMarkup( inline_keyboard=kb,    )
    await callback.message.answer("Hello! я бот MyCloudStore спробую вирішити твої питання!", reply_markup=keyboard)
# Хэндлер на произвольные обращения
@dp.message()
async def message_handler(message: types.Message) -> any:
    if(message.contact != None):
        found = False
        phone = message.contact.phone_number
        with open('phone_numbers.json') as f:
            phones = json.load(f)
            phones = phones['numbers']
        for section in phones:
            if section['phone'] == phone :
                found = True
        if found == False :
            newuser = {"phone":phone, "id":message.from_user.id}
            phones.append(newuser)
            to_json = {"numbers":phones}
            with open('phone_numbers.json', 'w') as f:
             json.dump(to_json, f)
        await message.answer("Дякуємо за авторизацію", reply_markup=types.ReplyKeyboardRemove)
```

```
        kb = [    [ types.InlineKeyboardButton(text="Чат з нами □□□ ", callback_data="chat"),
            types.InlineKeyboardButton(text="Trade-in □□□", callback_data="trade"),
        ], [ types.InlineKeyboardButton(text="Q&A □ ", callback_data="qa"),
            types.InlineKeyboardButton(text="Сервіс □□□", callback_data="service"),
        ], [ types.InlineKeyboardButton(text="На сайт □□", url="https://mcstore.com.ua/")
        ]  ]
        keyboard = types.InlineKeyboardMarkup ( inline_keyboard=kb, )
        await message.answer("Hello! я бот MyCloudStore спробую вирішити твої питання!", reply_markup=keyboard)
    else:
        Mess = await bot.send_message(id_manager, message.chat.username + " " + message.text)
        kb = [  [ types.InlineKeyboardButton(text="Головне меню", callback_data="menu"),
            ], ]
        keyboard = types.InlineKeyboardMarkup( inline_keyboard=kb, )
        await message.answer("Дякуємо за звернення, менеджер скоро з вами зв'яжеться", reply_markup=keyboard)
# Запуск процесса поллинга новых апдейтов
async def main():
    await dp.start_polling(bot)
if __name__ == "__main__":
    asyncio.run(main())
```

ДОДАТОК Б. ПРЕЗЕНТАЦІЯ ДОПОВІДІ

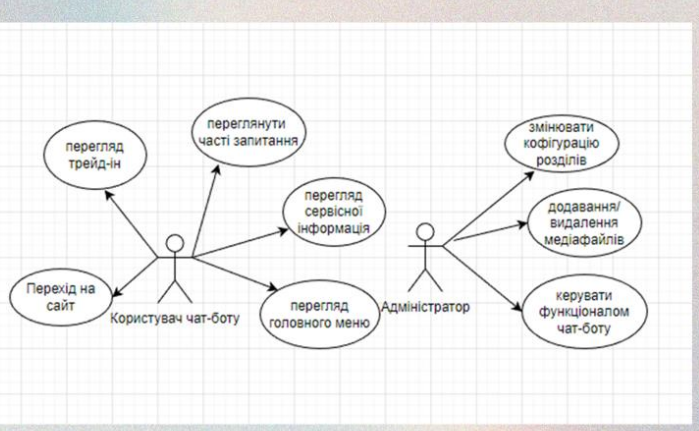




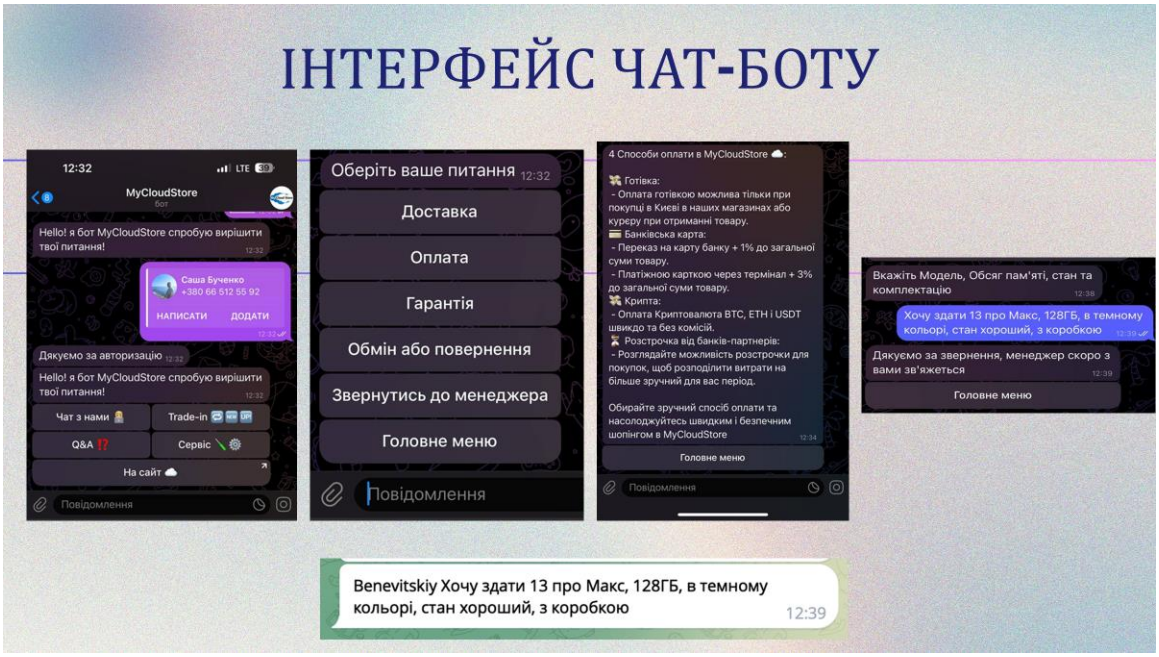

ОГЛЯГ ІСНУЮЧИХ МЕСЕНДЖЕРІВ
провів детальний аналіз популярних месенджерів, таких як Telegram, WhatsApp, WeChat, Facebook Messenger та Viber. Були розглянуті їхні функціональні можливості, особливості API, а також їх популярність серед користувачів. Важливим аспектом цього розділу є вибір Telegram як найбільш зручної платформи для реалізації чат - бота завдяки її безпеці та зручності для розробників.


МОВИ, БІБІОТЕКИ ТА СЕРЕДОВИЩЕ РОЗРОБКИ
Aiogram
це асинхронна бібліотека для створення Telegram-ботів на Python з високою швидкістю та гнучкістю для легко повного розвитку ботів з новими функціями Telegram bot API та зручним синтаксисом.
Python
це універсальна інтерпретована мова, сумісна з багатьма системами, що робить її використання широко поширеним.
Microsoft Visual Code
це багатофункціональний редактор коду, який став популярним серед розробників програмного забезпечення. VS Code розробляється компанією Microsoft і доступний для використання на різних операційних системах, включаючи Windows, macOS та Linux.

ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ



ІНТЕРФЕЙС ЧАТ-БОТУ



ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

У результаті виконаної роботи було створено ефективний Telegram-бот, який автоматизує процеси обслуговування клієнтів і допомагає знизити навантаження на персонал. Головним досягненням є реалізація бота, що дозволяє обробляти запити користувачів, автоматично відповідати на запитання та інтегруватися з іншими системами.
У майбутньому цей проект може бути вдосконалений, додавши нові функціональні можливості, такі як голосові або відео-консультації, а також інтеграцію з іншими месенджерами та платіжними системами.

ДЯКУЮ ЗА УВАГУ!