МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія (кафедра) комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

TELEGRAM ЧАТБОТ ПРО РОБОТУ ВІДДІЛЕНЬ НОВОЇ ПОШТИ

Виконав: студент групи 2П-21
Спеціальності
121 Інженерія програмного забезпечення
Іван ЛІСУН
Керівник:
Вікторія НЕМЧЕНКО

Черкаси 2025

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

Кафедра комп'ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри КІ та ІТ

_____ Владислав ХОТУНОВ
(підпис)
«_____»_____2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Лісуну Івану Олександровичу

1. Тема кваліфікаційної роботи «Telegram чатбот про роботу відділень "Нової пошти"»

Керівник роботи Немченко Вікторія Юріївна, викладач другої категорії

затверджені наказом закладу вищої освіти від «07» жовтня 2024 року № 68у.
2. Строк подання студентом кваліфікаційної роботи 02.06.2025
3. Вихідні дані до кваліфікаційної роботи створення Telegram чатбота, який дозволить автоматизувати процес отримання інформації про роботу відділень Нової Пошти, їхній графік, доступні послуги та завантаженість. Це сприятиме зручності користувачів та підвищенню ефективності обслуговування клієнтів Нової Пошти.
4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити) огляд існуючих рішень для інформування клієнтів Нової Пошти через чатботи, визначення вимог до Telegram чатбота, розробка архітектури чатбота та його інтеграція з API Нової Пошти, дизайн інтерфейсу взаємодії користувача з ботом, реалізація функціональності для отримання інформації про відділення, графік роботи, послуги та завантаженість, тестування та оптимізація роботи чатбота.
5. Дата видачі завдання 16.09.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання з підписами керівника і студента
1	Вступ	14.10.2024	
2	Розділ 1. Теоретичні відомості предметної області. Актуальність ботів та можливостей їх реалізації	9.12.2024	
3	Розділ 2. Вибір програмних засобів. Проєктування бота	10.03.2025	
4	Розділ 3. Налаштування інструментів та розроблення бота	28.04.2025	
5	Розділ 4. Огляд функціональної частини. Тестування роботи бота	06.05.2025	
6	Висновки	12.05.2025	
7	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2025	
8	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2025	
9	Подання кваліфікаційної роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	10.06.2025	

Студент _____ Іван ЛІСУН
(підпис)

Керівник роботи _____ Вікторія НЕМЧЕНКО
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Розробка Telegram чатбота для інформування користувачів про роботу відділень Нової пошти» включає в себе перелік умовних позначень і скорочень, вступ, основну частину, що складається з чотирьох розділів, висновків та списку використаних джерел. Загальний обсяг роботи становить 90 сторінок, в якій представлено 60 рисунків, 4 таблиці та 3 додатки. Список використаних джерел містить 20 позицій.

Telegram чатботи є сучасними засобами автоматизації взаємодії з користувачами, які широко використовуються в логістичній сфері. У рамках цієї роботи був створений чатбот «NovaPoshta_bot», що забезпечує користувачів актуальною інформацією про відділення Нової пошти, зокрема їх графіки роботи, місцезнаходження та доступні послуги.

Розробка була здійснена за допомогою мови програмування Python, бібліотеки Telebot та Telegram Bot API, що забезпечують ефективну обробку запитів користувачів і зручну інтеграцію з месенджером Telegram. У цьому документі детально викладено етапи проектування бота, підключення до API, реалізацію логіки відповідей на запити, структурування функціоналу, а також вибір інструментів і розгортання на серверній платформі Heroku.

Створений бот ілюструє потенціал автоматизації процесу інформування клієнтів у сфері логістики та може бути адаптований для використання в інших подібних сервісах.

Ключові слова: Python, Telegram, Telebot, Telegram Bot API, інформаційні технології, чатбот, логістика, Нова пошта, https-запити.

ABSTRACT

The qualification work on the topic “Development of a Telegram chatbot to inform users about the work of Nova Poshta branches” includes a list of symbols and abbreviations, an introduction, the main part consisting of four chapters, conclusions and a list of references. The total volume of the work is 90 pages, which includes 60 figures, 4 tables and 3 appendices. The list of references includes 20 items.

Telegram chatbots are modern tools for automating user interaction that are widely used in the logistics sector. As part of this work, NovaPoshta_bot was created to provide users with up-to-date information about Nova Poshta offices, including their working hours, locations, and available services.

The development was carried out using the Python programming language, the Telebot library, and the Telegram Bot API, which ensure efficient processing of user requests and convenient integration with the Telegram messenger. This document describes in detail the stages of bot design, API connection, implementation of the logic for responding to requests, structuring the functionality, as well as the choice of tools and deployment on the Heroku server platform.

The created bot illustrates the potential of automating the process of informing customers in the logistics sector and can be adapted for use in other similar services.

Keywords: Python, Telegram, Telebot, Telegram Bot API, information technology, chatbot, logistics, Nova Poshta, HTTPS requests.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ……3
ВСТУП……..4
РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРЕДМЕТНОЇ ОБЛАСТІ. АКТУАЛЬНІСТЬ БОТІВ ТА МОЖЛИВОСТЕЙ ЇХ РЕАЛІЗАЦІЇ……6
1.1 Формування мети проєкту 6
1.2 Актуальність ботів..... 7
1.3 Порівняльний аналіз платформ для інтеграції бота 9
1.4 Огляд існуючих рішень……12
РОЗДІЛ 2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ. ПРОЄКТУВАННЯ БОТА ……14
2.1 Мова програмування……..14
2.2 Середовище розробки……15
2.3 Створення бота у Telegram.……17
2.4 Вибір серверної платформи для доступу до бота……...18
2.5 Додаткові компоненти для роботи……...19
2.6 Проєктування бота на основі стандартів IDEF0 та UML……...20
РОЗДІЛ 3 НАЛАШТУВАННЯ ІНСТРУМЕНТІВ ТА РОЗРОБЛЕННЯ БОТА…27
РОЗДІЛ 4 ОГЛЯД ФУНКЦІОНАЛЬНОЇ ЧАСТИНИ. ТЕСТУВАННЯ РОБОТИ БОТА……...52
ВИСНОВКИ..……..64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..……66
ДОДАТКИ..……69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

IT – інформаційні технології.

БД – база даних.

ОС – операційна система.

ПЗ – програмне забезпечення.

API (*application programming interface*, програмний інтерфейс додатка*)* – опис методів та можливостей взаємодії однієї програми з іншою.

Telebot – бібліотека Python для роботи з телеграм.

AI (Artificial Intelligence) – штучний інтелект.

НП – Нова пошта.

HTTPS-запит – безпечний запит до вебсерверів через протокол HTTPS.

ВСТУП

Майже неможливо уявити сучасне життя без смартфонів та незліченних онлайн-сервісів, які роблять його простішим. Серед цих корисних інструментів є чат-боти - по суті, комп'ютерні програми, які спілкуються з вами за допомогою тексту. Ви знайдете чат-ботів скрізь, від обслуговування клієнтів і доставки до фінансів і не тільки. Вони створені для того, щоб надавати вам інформацію, виконувати прості завдання та зв'язувати вас з бізнесом у швидкий і простий спосіб.

Однією з найвідоміших платформ для розробки чатботів є месенджер Telegram, що пропонує різноманітні можливості для їх створення. Чатботи в Telegram здатні виконувати функції, які зазвичай виконують мобільні додатки та вебсайти, забезпечуючи користувачам зручний доступ до інформації та послуг без потреби в установці додаткових програм.

У контексті швидкого розвитку електронної комерції та логістичних послуг зростає потреба в інструментах, які забезпечують оперативний доступ до даних про поштові служби. Однією з провідних кур'єрських компаній в Україні є Нова пошта, яка щоденно обробляє тисячі відправлень. Користувачі часто стикаються з необхідністю отримання актуальної інформації про роботу відділень, їх графік, доступні послуги та місцезнаходження найближчих пунктів обслуговування.

Об'єктом дослідження є створення бота на мові Python та його інтеграція з Telegram Bot API.

Предметом дослідження є способи інтегрування чатботів у Telegram для автоматизованого надання користувачам інформації про роботу відділень Нової Пошти.

Метою розробки Telegram чатбота «NovaPoshta_bot» є розробка зручного інструменту, який надає інформацію про роботу відділень Нової пошти. Даний бот дозволить користувачам швидко знаходити найближчі відділення, отримувати інформацію про їхній графік роботи, доступні послуги та інші корисні дані, що сприятимуть покращенню взаємодії з поштовим оператором.

Telegram чатбот «NovaPoshta_bot» матиме такі можливості:

- Пошук найближчих відділень за геолокацією або введеною адресою.
- Перегляд графіку роботи та переліку доступних послуг для кожного відділення.
- Відстеження статусу відправлення за номером ТТН.
- Отримання актуальних новин та оновлень щодо роботи Нової пошти.
- Інтеграція з API Нової пошти для швидкого доступу до офіційних даних.

Апробація роботи. Часткові результати кваліфікаційної роботи були апробовані на XVII Студентській науково-практичній конференції студентів, аспірантів та молодих вчених «ІТенденції розвитку ІТ-технологій в Україні» [1].

Створення даного чатбота значно полегшить користувачам доступ до необхідної інформації, зменшуючи потребу в відвідуванні вебсайтів або зверненнях до служби підтримки. Він стане дієвим інструментом для автоматизації процесу отримання довідкових даних та підвищення якості взаємодії клієнтів з логістичним оператором.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРЕДМЕТНОЇ ОБЛАСТІ. АКТУАЛЬНІСТЬ БОТІВ ТА МОЖЛИВОСТЕЙ ЇХ РЕАЛІЗАЦІЇ

1.1 Формування мети проєкту

У сучасному світі інформаційні технології займають центральне місце в багатьох сферах, зокрема в логістиці та доставці. Все більше людей звертаються до цифрових рішень для швидкого та зручного отримання необхідної інформації. Одним із таких рішень є чатботи, які автоматизують рутинні процеси і суттєво полегшують взаємодію між сервісами та користувачами.

Сьогодні "Нова пошта" є однією з провідних логістичних компаній в Україні, обробляючи тисячі відправлень щоденно. Однак, користувачі часто стикаються з труднощами у пошуку актуальної інформації про відділення, їх графік роботи, доступні послуги або статус відправлень. Традиційні способи отримання цієї інформації, такі як дзвінки до служби підтримки або перегляд веб-сайту, можуть бути незручними і вимагати значного часу [2].

Застосування чатбота в Telegram для автоматизації збору даних про відділення Нової пошти є розумним рішенням, яке сприяє поліпшенню досвіду користувачів. Цей бот надає миттєвий доступ до інформації в зручному форматі, усуваючи потребу в додаткових діях з боку користувача.

Основні причини актуальності проєкту:

- Оперативність та доступність – чатбот працює 24/7, надаючи користувачам актуальну інформацію в будь-який час.
- Автоматизація процесів – бот дозволяє уникнути необхідності дзвінків або пошуку інформації вручну.

- Зручність для користувачів – уся необхідна інформація доступна безпосередньо в Telegram, без необхідності відвідування вебсайтів чи встановлення додаткових додатків.
- Інтеграція з API Нової пошти – дозволяє отримувати достовірні та актуальні дані про роботу відділень, статуси відправлень тощо.

Метою даного дипломного проєкту стала розробка чатбота для Telegram "NovaPoshta_bot". Цей бот призначений для надання користувачам миттєвого доступу до необхідної інформації про відділення «Нової пошти», включаючи контактні дані, графік роботи, доступні послуги та статус відправлень. Кінцева мета - впорядкувати та покращити клієнтський досвід, зробивши взаємодію з логістичною компанією простішою, зручнішою та ефективнішою [3].

1.2 Актуальність ботів

Чатбот можна охарактеризувати як інструмент, що автоматизує взаємодію з користувачами, надаючи необхідну інформацію або допомагаючи виконувати різні дії. Якщо заглибитися в суть, чатбот є комп'ютерною програмою, яка функціонує на основі алгоритмів або технологій машинного навчання, здатною виконувати широкий спектр завдань — від простих бесід до складних інтеграцій з зовнішніми сервісами [4].

Чатботи здобули велику популярність і продовжують активно інтегруватися в різні сфери життя. Їх застосовують у фінансах, медицині, електронній комерції, обслуговуванні та логістиці. Особливо в умовах зростаючої цифровізації, поштові та кур'єрські компанії починають автоматизувати свої послуги, щоб покращити якість обслуговування клієнтів.

У січні 2021 року "Укрзалізниця" представила чатбот у Telegram та Viber, який дозволяє купувати та скасовувати квитки. Це нововведення значно полегшило процес придбання квитків для пасажирів. Такі рішення не лише підвищують зручність для користувачів, але й сприяють оптимізації роботи операторів служби підтримки.

Згідно з дослідженням CNBC, впровадження чатботів може допомогти компаніям зекономити понад 8 мільярдів доларів щорічно в найближчому майбутньому. Все більше підприємств, від малих до великих, переходять до автоматизації своїх бізнес-процесів, створюючи ботів для виконання рутинних завдань.

Актуальність створення Telegram чатбота для Нової пошти є надзвичайно актуальним через зростаючу потребу в швидкому та зручному доступі до інформації про роботу відділень, статус відправлень та інші послуги. Автоматизація цього процесу за допомогою бота надасть користувачам можливість отримувати необхідні дані в реальному часі, уникаючи телефонних дзвінків або пошуку інформації на веб-сайтах.

Рисунок 1.1 – Діаграма прогнозу ринку чатботів з 2018 по 2027 роки

На рисунку 1.1 представлено прогноз щодо еволюції ринку чатботів. Фахівці в галузі логістики та автоматизації очікують, що до 2027 року прибуток цього ринку зросте в 11 разів. Це підкреслює важливість інтеграції ботів у сферу поштових і кур'єрських послуг для підвищення ефективності роботи та поліпшення обслуговування клієнтів [5].

1.3 Порівняльний аналіз платформ для інтеграції бота

Сучасні месенджери та цифрові платформи стали важливими елементами нашого щоденного життя, особливо в галузі логістики та поштових послуг. Використання чатботів для автоматизації обслуговування клієнтів значно полегшує процес взаємодії користувачів із службами доставки.

Для успішного впровадження чатбота, який надаватиме інформацію про діяльність відділень «Нової пошти», важливо вибрати найкращу платформу для його інтеграції. У цьому розділі здійснено аналіз найбільш популярних месенджерів, що підходять для запуску бота, а також проведено порівняння їхніх функцій, щоб ухвалити обґрунтоване рішення.

На рисунку 1.2 представлено рейтинг мобільних додатків, які використовують українці на початок січня 2025 року, виражений у відсотках. У рамках цього дослідження нас цікавлять платформи, що дозволяють створювати чатботи для автоматизації обслуговування клієнтів у галузі логістики та доставки. Зокрема, це Telegram, Viber, Facebook Messenger і WhatsApp, оскільки вони можуть бути інтегровані з ботами для отримання інформації про роботу відділень «Нової пошти» [6].

Рисунок 1.2 – Рейтинг мобільних додатків на січень 2025 року

Для порівняння вказаних месенджерів була створена таблиця, що містить основні характеристики, які мають значення для розробки та впровадження телеграм-бота. Це допоможе виявити найбільш підходящу платформу для інтеграції бота та забезпечити його ефективну роботу.

Вивчення та аналіз платформ для інтеграції навчального телеграм-бота є вирішальним етапом у визначенні найкращого середовища для його впровадження. У наступному тексті представлені основні характеристики, що впливають на вибір платформи.

Хмарне сховище виконує ключову роль, дозволяючи зберігати користувацькі дані, такі як історія взаємодії з ботом, результати тестів і навчальні ресурси, на віддалених серверах. Це означає, що навіть якщо користувач видалить месенджер або змінить пристрій, він зможе відновити свій прогрес без втрати важливої інформації. Така можливість є критично важливою для ефективного навчання, оскільки забезпечує безперервність роботи з ботом і доступ до матеріалів у будь-який час.

Охоплення населення України є вирішальним фактором при виборі платформи для інтеграції навчального телеграм-бота. Чим більше людей користується конкретним месенджером, тим більша ймовірність, що вони виявлять інтерес до бота, що, в свою чергу, сприятиме його успішному розповсюдженню.

Таблиця 1.1 – Порівняння платформ для інтеграції телеграм-бота на 2025 рік

Функціонал	WhatsApp	Viber	Fb Messenger	Telegram
Хмарне сховище	-	-	+	+
Охоплення 50+% населення України	-	+	+	+
Відкритий API для розробників	-	+	+	+
Можливість інтеграції навчальних матеріалів	-	-	+	+
Шифрування, анонімність	-	-	-	+
Підтримка Dekstop версії	-	-	+	+
Можливість створення інтерактивних тестів	-	-	+	+
Гнучкість у налаштуванні бота	-	-	+	+

Відкритий API є ключовим елементом для інтеграції бота, оскільки він дозволяє безкоштовно взаємодіяти з платформою. Серед усіх розглянутих месенджерів лише WhatsApp пропонує закритий API, що вимагає платного доступу до його функцій. Оскільки навчальний бот не має комерційних цілей, найкращим варіантом буде вибір платформи з відкритим API, що забезпечить користувачам можливість безкоштовного використання бота.

Двостороння відмова означає, що жоден користувач не може отримувати повідомлення без свого дозволу. Це важливо для навчального бота, оскільки він не повинен нав’язувати свої послуги, а має мотивувати користувачів до навчання лише за їхньою власною ініціативою.

Шифрування та анонімність є однією з основних переваг Telegram, що робить його найнадійнішим серед інших платформ. Завдяки передовим технологіям захисту даних, користувачі можуть бути впевнені в конфіденційності своїх даних. Ще однією важливою характеристикою Telegram є відсутність реклами, яка може відволікати від навчання та створювати незручності під час роботи з ботом.

Крім того, Telegram надає значні можливості для розробників, зокрема для створення та інтеграції ботів. Це стало основою для висновку, що Telegram є найкращою платформою для впровадження навчального бота, оскільки він забезпечує безпеку, зручність у використанні та гнучкість у розробці [7].

1.4 Огляд існуючих рішень

Після ретельного дослідження та аналізу доступних інструментів, виявилося, що немає жодного телеграм-бота, який би повністю задовольняв вимоги користувачів щодо автоматизації навчального процесу в обраній області.

Проте існує безліч альтернативних ресурсів для самостійного вивчення матеріалу, зокрема:

- довідники та документація;
- онлайн-курси та вебінари;
- інтерактивні платформи для навчання;
- офлайн-лекції та майстер-класи;
- конференції та семінари;
- онлайн-ресурси з можливістю отримання сертифікатів;
- відеоуроки та презентації.

При розгляді доступних варіантів, можна відзначити, що найкращі навчальні ресурси зазвичай є платними, оскільки вони пропонують структуровані курси, індивідуальні консультації та можливість отримання сертифікатів. Однак такий підхід може бути не зовсім зручним для початківців, які лише починають освоювати тему. Висока ціна курсів, складність навчальних матеріалів і відсутність інтерактивних елементів можуть стати серйозними перешкодами для багатьох користувачів.

Таким чином, важливим є розробка телеграм-бота, який забезпечить доступність, інтерактивність та поетапне навчання користувачів, сприяючи їхньому освоєнню матеріалу в зручному форматі [8].

РОЗДІЛ 2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ. ПРОЄКТУВАННЯ БОТА

Після визначення платформи для створення телеграм-чатбота та усвідомлення його ролі в автоматизації навчального процесу, наступним етапом є підбір найбільш підходящих програмних засобів для реалізації цього проєкту. Це передбачає вибір мови програмування, фреймворків, бібліотек, а також інструментів для розгортання та підтримки бота. Коректний вибір технологічного стеку гарантує стабільну роботу чатбота, його масштабованість і зручність у використанні.

2.1 Мова програмування

Першим кроком у процесі вибору програмного забезпечення для створення телеграм-бота є вибір мови програмування.

Програмування – це засіб, що надає розробникам можливість спілкуватися з комп'ютером, формуючи алгоритми та команди для виконання конкретних завдань.

Телеграм-чатбот можна створити за допомогою будь-якої сучасної мови програмування, що має підтримку API. У нашому випадку ми обрали Python.

Python є універсальною, гнучкою та високорівневою мовою програмування, що знайшла широке застосування в розробці вебдодатків, автоматизації завдань, створенні чатботів і в сфері штучного інтелекту. Серед її основних переваг варто відзначити зрозумілий синтаксис, обширну стандартну бібліотеку та активну спільноту розробників [9].

Однією з причин, чому Python обирають для розробки телеграм-ботів, є зручність роботи з API Telegram. Крім того, існує безліч готових бібліотек,

які спрощують взаємодію з месенджером, серед яких варто відзначити Aiogram, pyTelegramBotAPI та python-telegram-bot.

Щоб продемонструвати різницю в синтаксисі, давайте подивимося на те, як виглядає основна команда для виведення тексту на екран у Java та Python (дивитися рисунок 2.1).

Рисунок 2.1 – Порівняння коду програми Hello World між Python та Java

Якщо звернути увагу на рисунок 2.1, навіть людина без програмістського досвіду зможе помітити, що код на Python виглядає значно простіше і зрозуміліше. Це суттєво полегшує навчальний процес і знижує бар'єри для новачків, що є критично важливим при розробці телеграм-бота для освоєння основ програмування [10].

Однією з основних переваг використання Python для створення телеграм-бота є можливість інтеграції різноманітних бібліотек і додаткових модулів. Це значно полегшує та покращує процес кодування, що дозволяє швидко впроваджувати потрібні функції для навчального бота.

2.2 Середовище розробки

Середовище розробки, або IDE (Integrated Development Environment), є програмним забезпеченням, що забезпечує всі необхідні інструменти для створення, тестування та налагодження коду.

При виборі середовища для розробки навчального телеграм-бота важливо враховувати його сумісність з обраною мовою програмування, зручність використання, а також наявність необхідного функціоналу для швидкої розробки та налагодження. Нижче наведено аналіз кількох популярних середовищ розробки:

- VSCode – розробка компанії Microsoft, яка підтримує безліч мов програмування, зокрема Python. Є легким та гнучким редактором коду з можливістю розширення функціоналу за допомогою плагінів. Однак, для повноцінної роботи з Python необхідно встановити додаткові розширення та налаштування [11].
- Sublime Text 3 – мінімалістичний редактор коду, зручний у використанні, що підтримує Python. Однак, через відсутність вбудованого терміналу та необхідність постійного завантаження бібліотек, він менш зручний для роботи над проєктами, які потребують інтеграції зовнішніх модулів [12].
- PyCharm – середовище розробки, спеціально орієнтоване на Python. Підтримує розробку на Django, має широкий функціонал, автоматичне управління бібліотеками та інтегрований термінал. PyCharm пропонує безкоштовну версію, яка містить усі необхідні інструменти для розробки телеграм-бота [13].

Враховуючи особливості створення освітнього телеграм-бота, середовище PyCharm є найкращим вибором, оскільки воно забезпечує всі необхідні інструменти для зручної роботи з Python і підтримує розширені можливості для інтеграції з API та базами даних.

2.3 Створення бота у Telegram

Перед тим як почати розробку освітнього телеграм-бота, слід спочатку зареєструвати його в Telegram. Для цього потрібно скористатися офіційним ботом BotFather, який відповідає за створення та адміністрування ботів.

Процес реєстрації включає такі кроки:

- У додатку Telegram у рядку пошуку необхідно знайти @BotFather.
- Після запуску бота викликати команду /newbot для створення нового бота.
- Далі необхідно задати ім’я бота та унікальний юзернейм (який повинен закінчуватися на bot).
- Після успішної реєстрації BotFather надасть токен – унікальний ключ, який використовується для взаємодії з Telegram Bot API через середовище розробки.
- Отриманий токен необхідний для підключення бота до коду програми, а також для його подальшої інтеграції та налаштування функціоналу.

BotFather також пропонує ряд додаткових команд для управління ботом. Нижче представлений список основних команд разом з їх описом (таблиця 2.1).

Таблиця 2.1 – Основні функції для управління ботом

Функції	Характеристика
/help	Виводить список усіх доступних команд
/setname	Зміна імені бота у додатку
/setuserpic	Встановлення зображення в іконку чатбота
/mybots	Надає можливість змінити бота
/setprivacy	Вмикає приватний режим у чаті

Команди BotFather, згадані раніше, є інтегрованими інструментами Telegram, що дозволяють зручно управляти ботом. Вони надають можливість налаштовувати різні параметри, оновлювати інформацію, відновлювати доступ у випадку його втрати, а також повністю видаляти бота, якщо це потрібно.

Завдяки цій функції Telegram забезпечує зручне управління та гнучкість у налаштуванні бота, що робить його ідеальним вибором для створення навчального телеграм-бота [13].

2.4 Вибір серверної платформи для доступу до бота

Для забезпечення стабільної роботи бота та його швидкої взаємодії з користувачами важливо реалізувати серверне розгортання. Це дозволить боту функціонувати цілодобово, обробляючи запити без затримок.

Найкращим варіантом для вирішення цієї задачі є застосування Heroku – хмарної платформи, яка забезпечує автоматичне розгортання та інтеграцію з системою контролю версій Git [15].

Щоб розгорнути бота на серверій платформі Heroku, необхідно:

- Встановити Heroku CLI – інструмент командного рядка для роботи з платформою.
- Авторизуватися у системі за допомогою команди heroku login.
- Виконати кілька команд для синхронізації файлів між локальним середовищем та сервером.

Після успішної конфігурації в терміналі PyCharm з'явиться повідомлення, яке підтверджує успішне підключення. Це свідчить про те, що бот готовий до роботи, що забезпечує швидку реакцію на запити користувачів і безперебійну роботу в Telegram.

2.5 Додаткові компоненти для роботи

Після того як було оберано мову програмування, середовище для розробки, серверну платформу та зареєструєте бота, наступним кроком буде інтеграція відповідних бібліотек і модулів. Це необхідно для забезпечення належної роботи бота та його коректної взаємодії з Telegram API.

Основні компоненти:

- Бібліотека telebot – Python-обгортка для взаємодії з Telegram API. Вона значно спрощує роботу з ботом, дозволяючи легко створювати запити та обробляти відповіді.
- Telegram Bot API – HTTP-інтерфейс, через який бот отримує та надсилає повідомлення. Під час реєстрації в Telegram боту надається унікальний токен, що має формат: 123456:DEF-WBA7434HeIkv-qnm79J6k3u111kq1

Запити до API мають стандартний вигляд:

- Використовується HTTP-протокол
- Вказується токен для автентифікації
- Задається назва методу (наприклад, надсилання повідомлення)

Формат відповіді API – JSON-об'єкт із полями:

- ok – статус виконання (true/false)
- description – детальна інформація про результат запиту
- У разі помилки JSON-об’єкт містить причину її виникнення

Ці компоненти є основою для розробки ефективного і надійного телеграм-бота, здатного оперативно реагувати на запити та забезпечувати стабільну взаємодію з користувачами.

2.6 Проєктування бота на основі стандартів IDEF0 та UML

Метод IDEF0 надає можливість розробнику візуалізувати концепцію процесу, включаючи його входи (I), елементи управління (C), виходи (O) та механізми (M), які впливають на цей процес. Усі ці компоненти разом відомі як ICOM [16].

Кожна з чотирьох сторін функціонального блоку має своє певне значення (роль), при цьому:

- верхні стрілки являють собою елементи управління;
- ліва сторона є вхідними значеннями, які провокують початок процесу;
- права сторона є вихідними значеннями, тобто результат процесу в кінці;
- нижня сторона означає механізми, які використовуються для виконання завданого процесу.

На рисунку 2.2 представлена контекстна діаграма, що ілюструє процес створення бота «NovaPoshta_bot» у Telegram.

Згідно з контекстною діаграмою, доцільно створити діаграму декомпозиції моделі. Ця діаграма дозволяє більш детально проаналізувати процес, представлений у контекстній діаграмі, шляхом розподілу основного процесу на кілька додаткових етапів. Розробка діаграми декомпозиції забезпечує можливість детально описати процес створення бота. Це сприятиме кращому розумінню та організації виконання завдань, що буде корисним як у подальшій розробці, так і для замовника, щоб визначити терміни виконання проєкту.

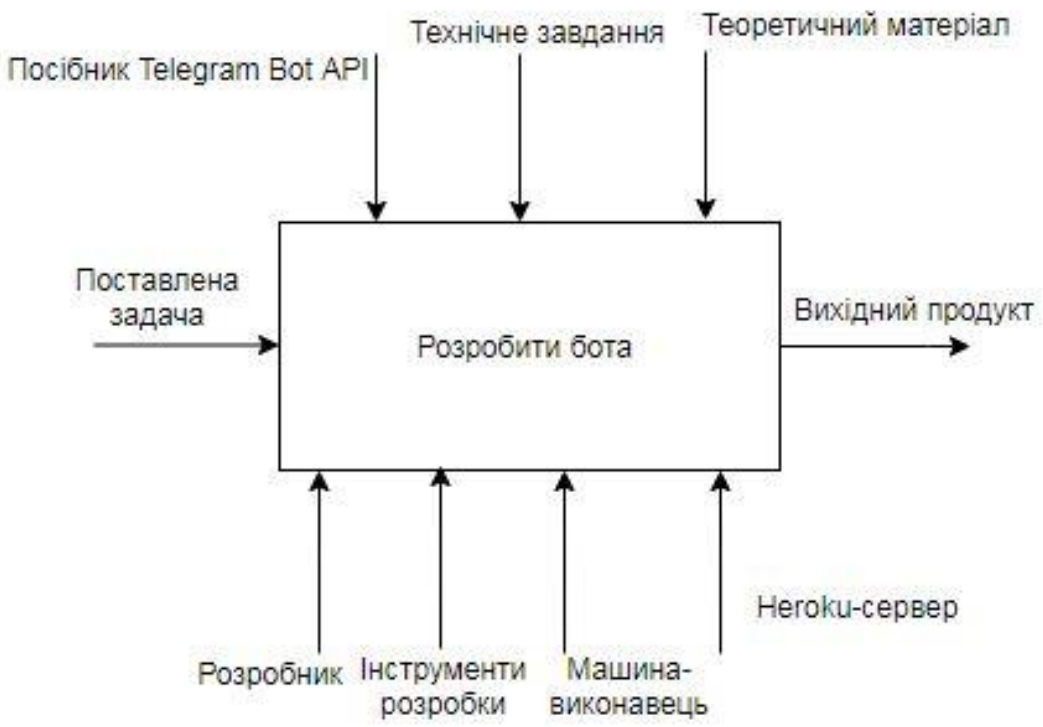


Рисунок 2.2 – Контекстна діаграма процесу розробки бота

Оскільки в контекстній діаграмі описується процес створення бота, діаграма декомпозиції надасть детальний аналіз того, як різні фактори та механізми впливають на окремі етапи виконання завдання. Зображення діаграми декомпозиції можна знайти на рисунку 2.3.

Діаграма поділяє процес розроблення бота на такі основні етапи.

1. Реєстрація бота у Telegram.
2. Процес розробки.
3. Тестування бота.

Після успішного завершення тестування проект вважається закінченим. Оскільки в діаграмі декомпозиції були визначені основні етапи розробки, наступним кроком стало створення діаграми взаємодії.

Діаграма варіантів використання ілюструє, які користувачі взаємодіють із системою та які функції їм доступні. Це допомагає

встановити зв'язок між функціоналом інтерфейсу бота та його користувачами. Завдяки цьому, під час розробки можна краще зрозуміти, як організувати навігацію та визначити певні обмеження. Зображення цієї діаграми представлено на рисунку 2.4.

Існує два типи дозволених користувачів. Адміністратор має повний доступ до всіх функцій, доступних звичайному користувачу, а також отримує дані про активність користувача в системі.

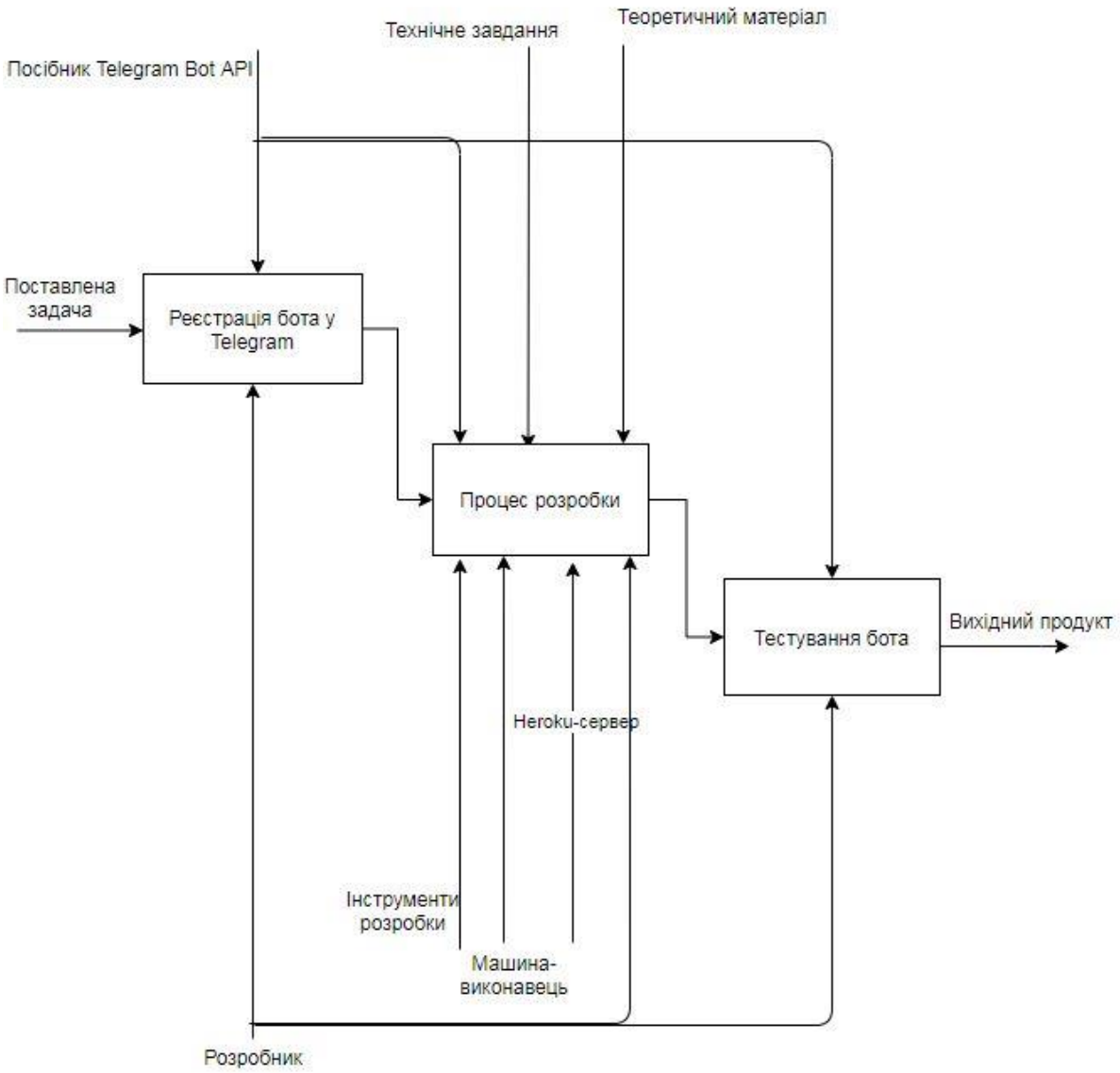


Рисунок 2.3 – Діаграма декомпозиції розроблення бота

Наступна схема ілюструє, як бот здійснює перевірку наявності нових повідомлень у системі. Щоб надати відповідь на запит користувача, боту потрібно періодично звертатися до інтерфейсу для моніторингу нових запитів. Схема, що демонструє процес перевірки оновлень у системі бота, представлена на рисунку 2.5.

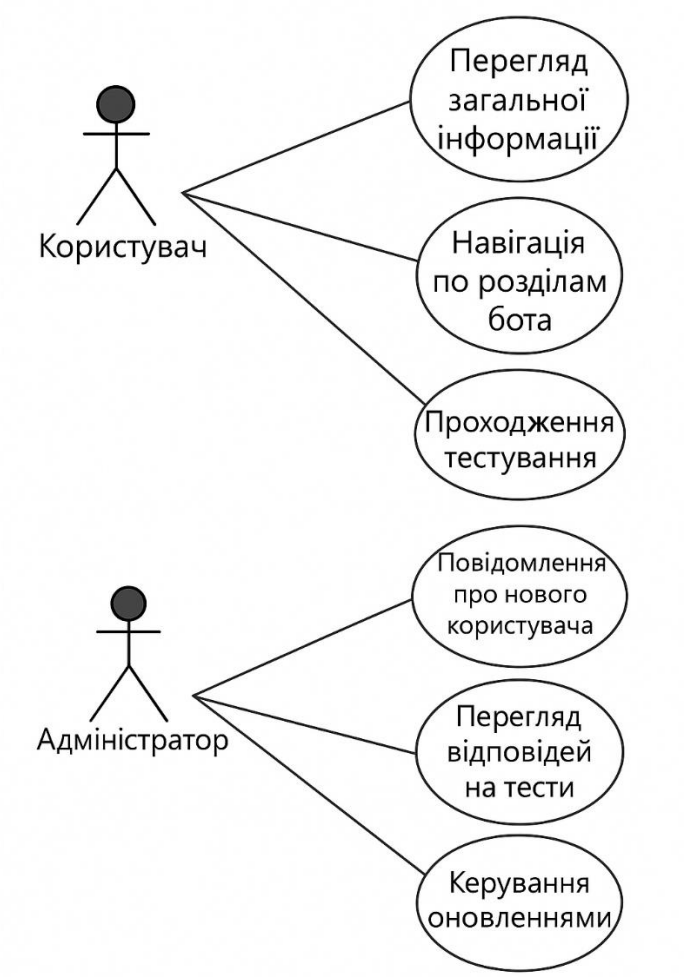


Рисунок 2.4 – Діаграма варіантів використання бота у системі

Перевірка нових повідомлень у телеграм-боті здійснюється через метод getUpdates. Цей метод є частиною Telegram Bot API і

використовується для моніторингу взаємодії між користувачем і ботом. Його структура представлена на рисунку 2.8.

На рисунку 2.6 ілюструється процес взаємодії між користувачем, Telegram API та сервером, на якому функціонує бот. Користувач надсилає запит (команду), який обробляється Telegram API. Після цього відбувається виклик відповідної функції на сервері, що реалізує певну логіку (наприклад, пошук відділення або перегляд послуг). У відповідь бот формує повідомлення, яке надсилається користувачеві через Telegram. Ця послідовність дій забезпечує чітку та логічну структуру обробки запитів у боті.

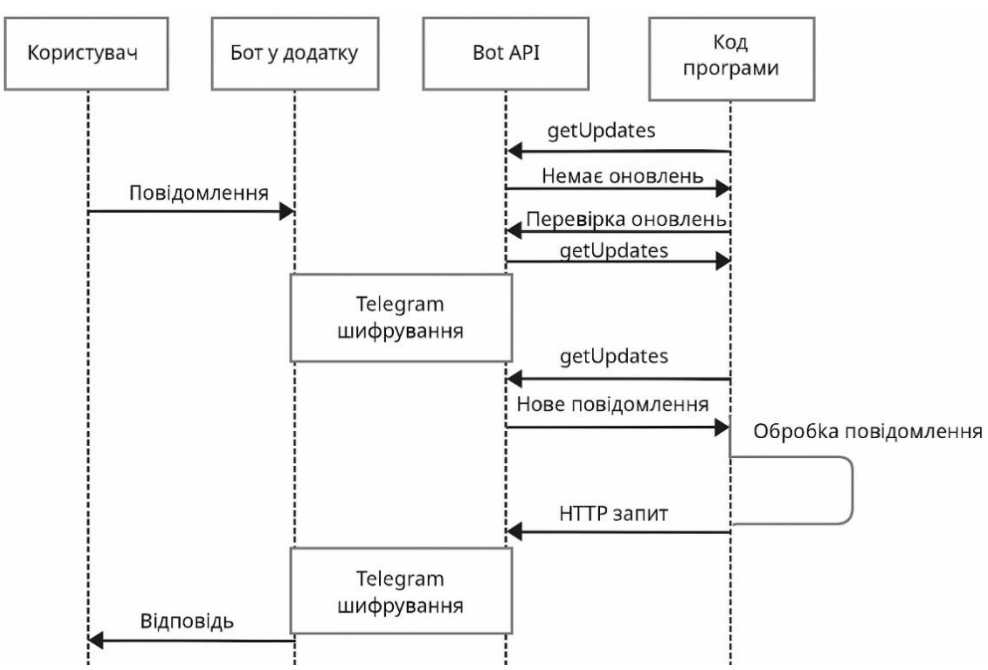


Рисунок 2.5 – Перевірка нових повідомлень у чатботі

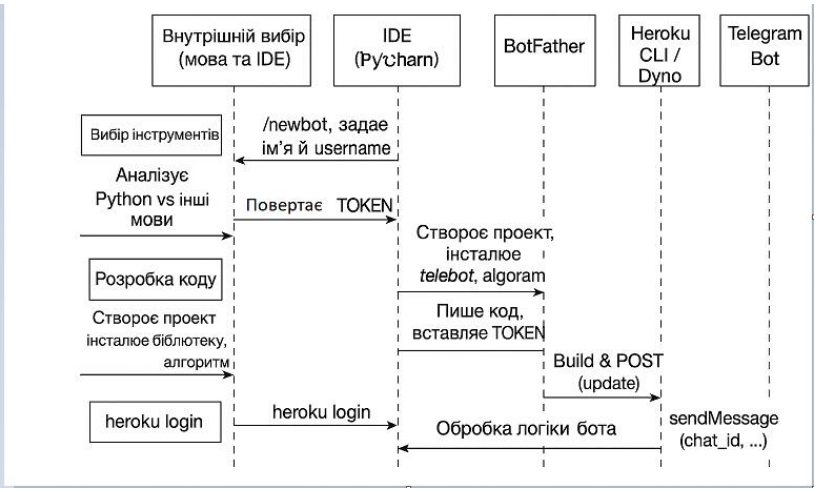


Рисунок 2.6 – Діаграма послідовностей розробки чатбота

На рисунку 2.7 представлена схема інтеграції Telegram чатбота з API компанії «Нова пошта». Коли бот отримує запит від користувача, наприклад, щодо пошуку відділення, він формує HTTP-запит до зовнішнього API, передаючи необхідні дані, такі як місто та індекс. Відповідь, отримана у форматі JSON, обробляється ботом, і за потреби ці дані зберігаються у внутрішній базі для кешування або швидшого доступу. Після цього сформована відповідь надсилається користувачу через Telegram.

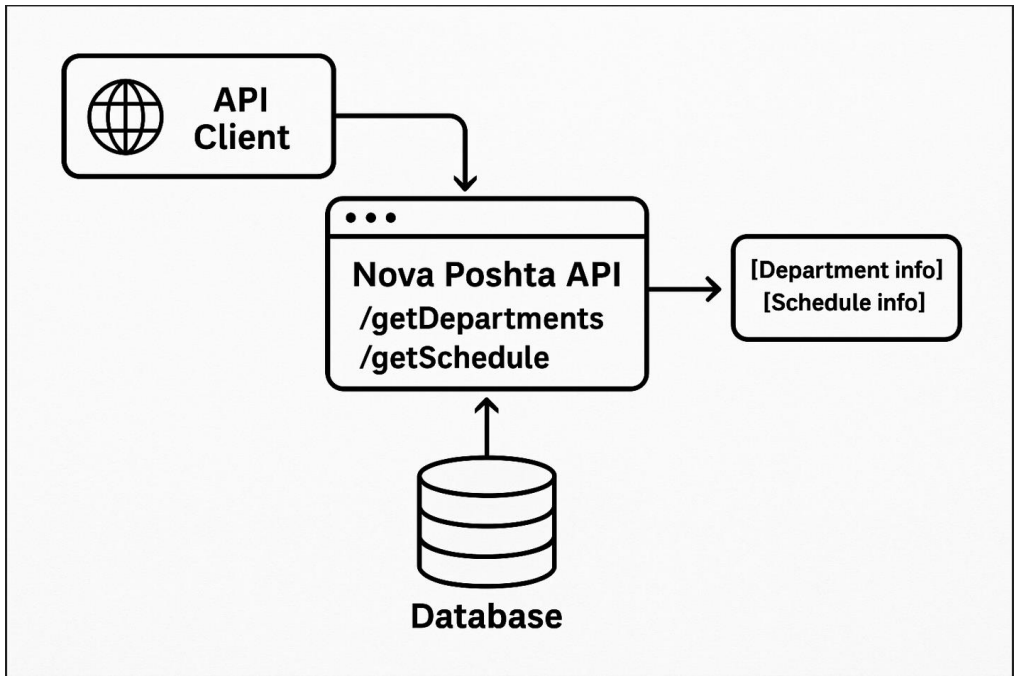


Рисунок 2.7 – Підключення API “Нової Пошти” до бази даних бота

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


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

Рисунок 2.8 – Лістинг коду отримання запитів від getUpdates

РОЗДІЛ 3 НАЛАШТУВАННЯ ІНСТРУМЕНТІВ ТА РОЗРОБЛЕННЯ БОТА

Після того як обрані технології та програмне забезпечення для реалізації проєкту, можна приступати до його виконання. Першим кроком слід детальніше вивчити процес реєстрації бота. Для цього необхідно ввести @botfather у пошукову стрічку додатку та розпочати діалог з ботом (дивитися рисунок 3.1).

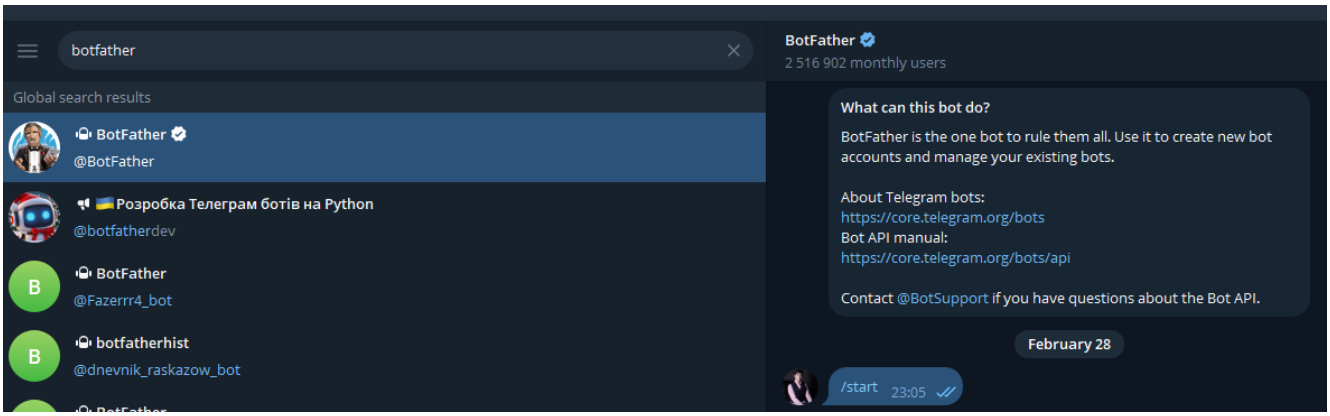


Рисунок 3.1 – Введення у пошукову стрічку @botfather

Використовуючи команду «/start», ви можете надіслати запит, який спонукає штучний інтелект надати відповідь. Після обробки запиту, система відправляє меню доступних функцій (дивитися рисунок 3.2).

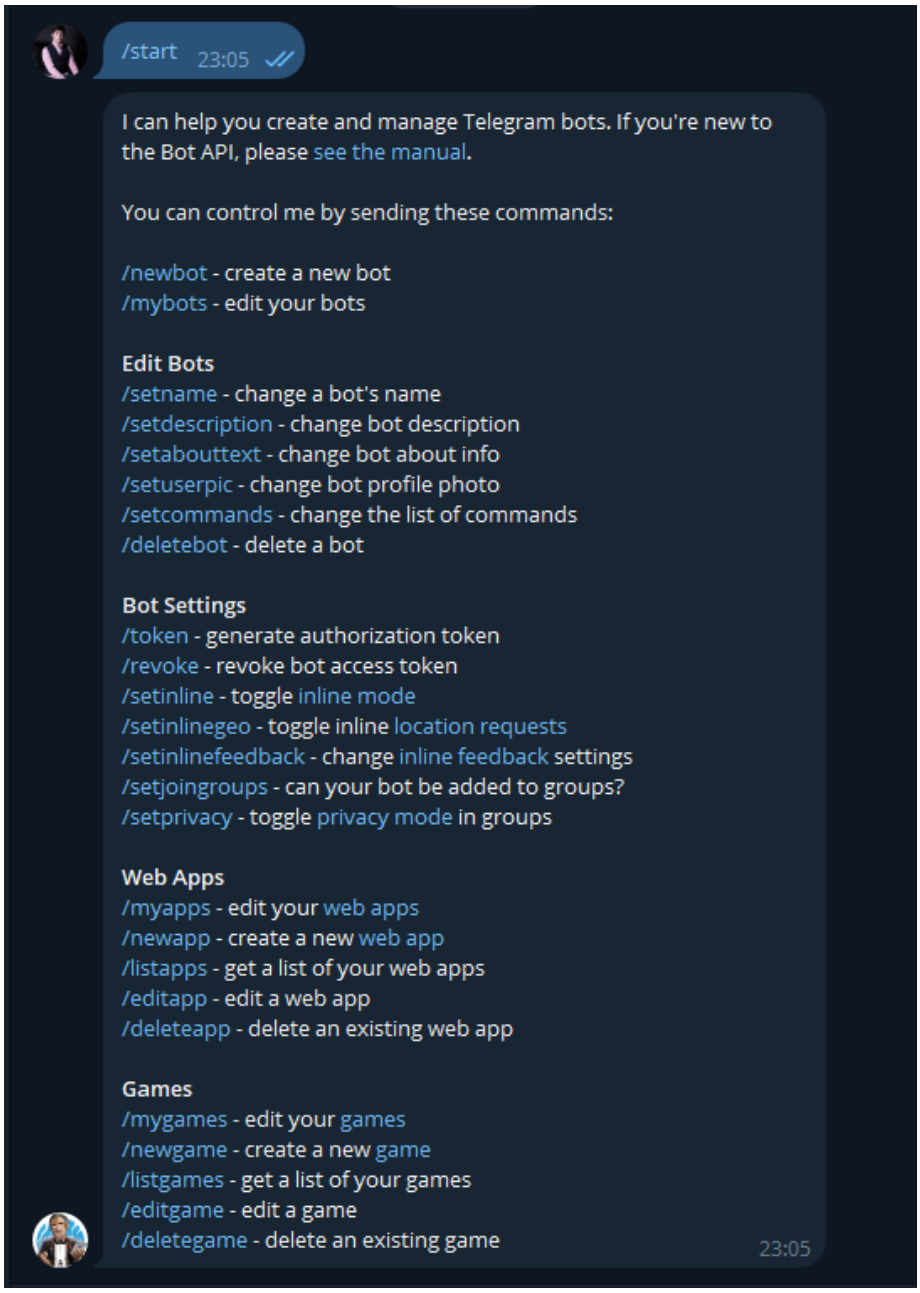


Рисунок 3.2 – Функції, що надає BotFather

Щоб створити бота в додатку, необхідно ввести команду «/newbot». Далі потрібно в чаті вказати назву для вашого бота. Хоча ця назва не вплине на функціонування бота, вона повинна давати користувачам уявлення про його призначення. Після цього BotFather запропонує вибрати ім’я користувача для бота. Це ім’я має бути унікальним і може містити лише

латинські літери, цифри та символ підкреслення. Важливо, щоб воно закінчувалося на "bot".

Після завершення попередніх етапів у чат надходить повідомлення, що реєстрацію бота завершено. У цьому повідомленні міститься токен для доступу до HTTP API, а також посилання на бота в додатку Telegram (дивитися рисунок 3.3).

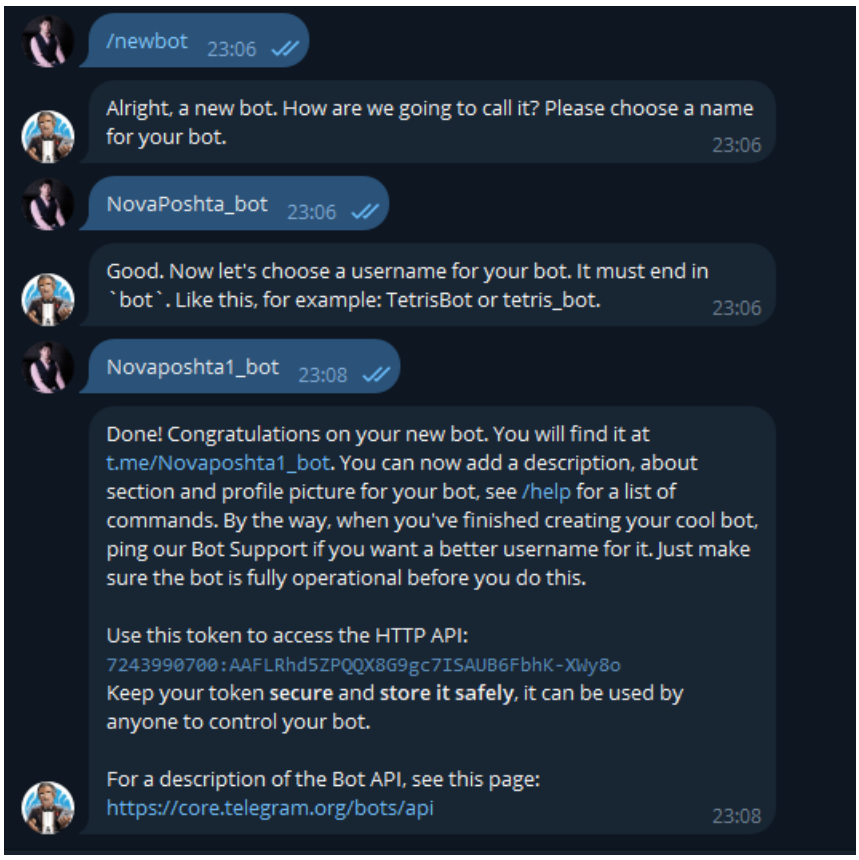


Рисунок 3.3 – Повідомлення з ключем доступу та посиланням на бота

Далі ви можете перейти за посиланням, щоб ознайомитися з ботом, який з часом буде доповнюватися новими командами та набувати завершеного вигляду. Наразі він виглядає як порожня сторінка, що відкриває простір для експериментів (рисунок 3.4). Клацнувши на іконку бота, ви

зможете переглянути інформацію про профіль, подібно до звичайного користувача (рисунок 3.5).

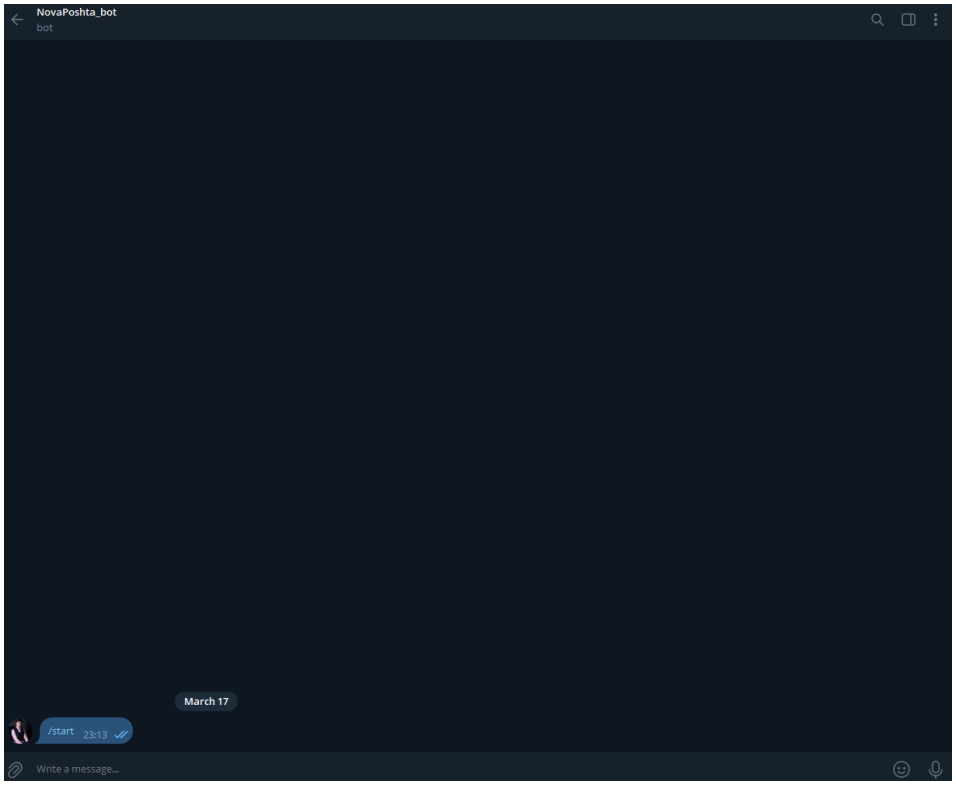


Рисунок 3.4 – Вікно чату з зареєстрованим ботом

Перед тим, як розпочати взаємодію з ботом, необхідно ввести команду «/start». На цей запит не буде жодної відповіді, оскільки бот поки що не здатний обробляти http-запити. З часом, завдяки програмуванню, він навчиться цьому.

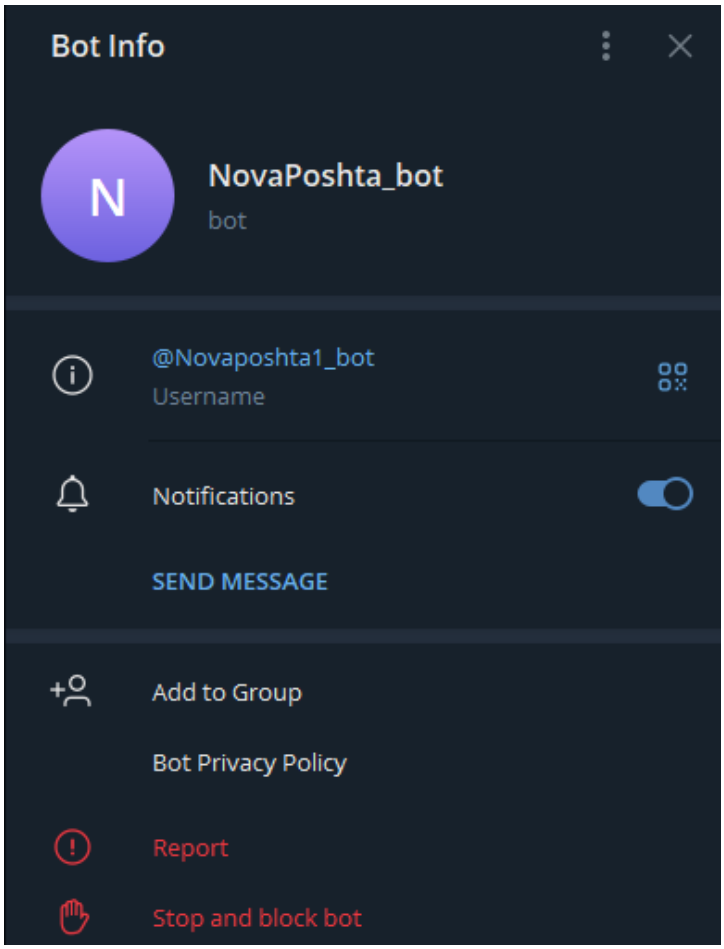


Рисунок 3.5 – Вікно додаткової інформації користувача

Дивлячись на цей інформаційний блок, стає зрозуміло, що він функціонує як звичайний чат з реальними людьми. Уявіть собі це вікно чату як чисте полотно. Код, який ви пишете в середовищі розробки, обробляється і виводиться у вікні чату бота. Цей вивід може мати різні форми: текст, зображення або кнопки. По суті, це інтерфейс для взаємодії з програмою. Важливо, що методи комунікації та шифрування, які тут використовуються, відрізняються від власних протоколів шифрування Telegram. Щоб бот відповідав на запити користувачів, ці запити надсилаються за протоколом HTTPS. Формат цих запитів має певну структуру:

https://api.telegram.org/bot'токен'/назва_методу.

Рисунок 3.6 – Оновлення зображення для бота

Щоб оновити іконку бота, потрібно зайти до BotFather і ввести команду «/setuserpic», після чого надіслати потрібне зображення в чат (дивитися рисунок 3.6). Після зміни іконки, якщо ви повернетеся до чату з ботом, ви зможете побачити, що нова іконка профілю вже активована (дивитися рисунок 3.7). Візуальне оформлення бота та його правильне представлення можуть суттєво вплинути на зацікавленість користувачів, спонукаючи їх натиснути на бота та почати взаємодію.

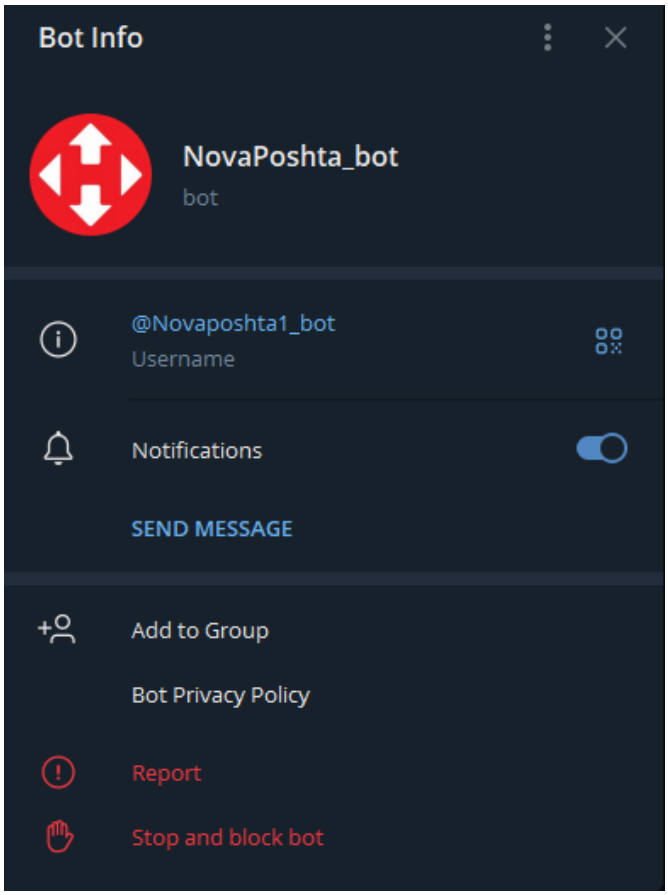
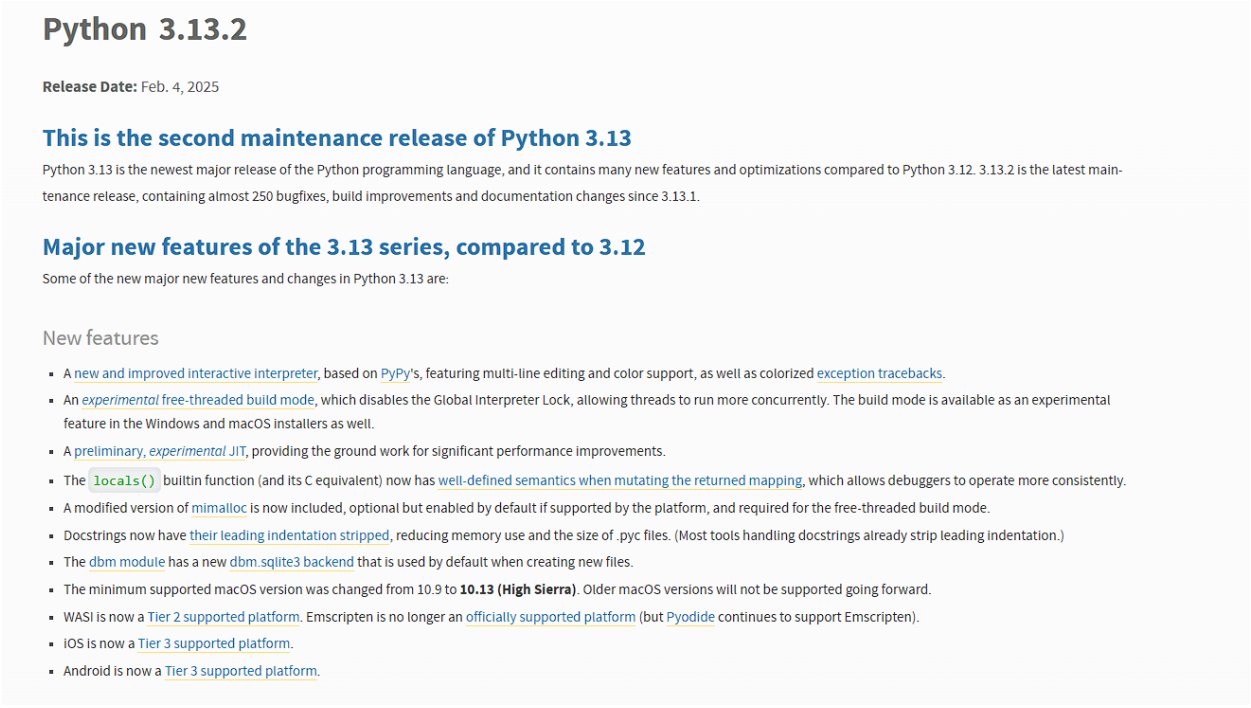


Рисунок 3.7 – Нове зображення профіля бота

Гаразд, давайте налаштуємо середовище розробки! Наступним кроком буде налаштування всього і встановлення всіх необхідних елементів для безперешкодного кодування. Насамперед вам потрібно встановити останню стабільну версію Python на ваш комп'ютер. Завантажуючи інсталятор з офіційного сайту Python, переконайтеся, що він сумісний операційною системою і що отримуєтеться стабільний випуск. Станом на березень 2025 року поточною рекомендованою версією є Python 3.13.2. Ця версія може похвалитися значними покращеннями продуктивності, більш оптимізованим інтерпретатором та оновленнями стандартних бібліотек. Хоча раніше використовувалися старіші версії, такі як Python 3.8 та 3.9, вони більше не отримують регулярних оновлень. Наприклад, підтримка Python 3.8 закінчилася у 2024 році, і тепер вона отримує лише критичні виправлення безпеки. Тому найкраще використовувати Python

3.13 або, якщо вам потрібно зберегти сумісність зі старими бібліотеками, Python 3.12. Перед встановленням будь-якої нової версії Python варто ознайомитися з примітками до випуску та змінами. Ці оновлення можуть суттєво вплинути на те, як ви пишете код, і, зрештою, зробити процес розробки більш ефективним.



Python 3.13.2

Release Date: Feb. 4, 2025

This is the second maintenance release of Python 3.13

Python 3.13 is the newest major release of the Python programming language, and it contains many new features and optimizations compared to Python 3.12. 3.13.2 is the latest maintenance release, containing almost 250 bugfixes, build improvements and documentation changes since 3.13.1.

Major new features of the 3.13 series, compared to 3.12

Some of the new major new features and changes in Python 3.13 are:

New features

- A new and improved interactive interpreter, based on PyPy's, featuring multi-line editing and color support, as well as colorized exception tracebacks.
- An *experimental* free-threaded build mode, which disables the Global Interpreter Lock, allowing threads to run more concurrently. The build mode is available as an experimental feature in the Windows and macOS installers as well.
- A preliminary, *experimental* JIT, providing the ground work for significant performance improvements.
- The `locals()` builtin function (and its C equivalent) now has well-defined semantics when mutating the returned mapping, which allows debuggers to operate more consistently.
- A modified version of mimalloc is now included, optional but enabled by default if supported by the platform, and required for the free-threaded build mode.
- Docstrings now have their leading indentation stripped, reducing memory use and the size of .pyc files. (Most tools handling docstrings already strip leading indentation.)
- The dbm module has a new dbm.sqlite3 backend that is used by default when creating new files.
- The minimum supported macOS version was changed from 10.9 to **10.13 (High Sierra)**. Older macOS versions will not be supported going forward.
- WASI is now a Tier 2 supported platform. Emscripten is no longer an officially supported platform (but Pyodide continues to support Emscripten).
- iOS is now a Tier 3 supported platform.
- Android is now a Tier 3 supported platform.

Рисунок 3.8 – Оновлення набуті у версії Python 3.13.2 від 4 лютого 2025 року

Зосередившись на виборі стабільної версії, ми можемо визначити оптимальний реліз. Після запуску інсталяційної програми натискаємо кнопку "Upgrade Now" і чекаємо на завершення процесу встановлення (дивитися рисунок 3.9).

Рисунок 3.9 – Встановлення оптимальної версії Python

Важливим процесом є налаштування середовища розробки. Відкривши середовище PyCharm створемо новий проєкт. Для цього перейдемо до вкладки File – NewProject (рис. 3.10), обрано папку в якій буде зберігатися проєкт, обрано середовище серед запропонованих:

- virtualenv;
- pipenv;
- conda.

Наразі для розробки на Python доступні три віртуальні середовища. Після створення віртуального середовища за допомогою virtualenv, ми все ще використовуємо pip для інсталяції пакетів і бібліотек. Потім, щоб зберегти список встановлених пакетів, викликаємо команду pip freeze.

PipEnv виник як відповідь на численні недоліки virtualenv. Це середовище дозволяє не лише встановлювати необхідні пакети для конкретного проєкту, але й безпосередньо додавати їх до файлу pipenv. Завдяки цьому, при повторному використанні середовища, можна швидко

отримати доступ до всіх раніше підключених пакетів, оскільки вони зберігаються у pip-файлі.

"Conda" спеціалізується на використанні пакетів NumPy та SciPy, які створені для роботи з масивами, а також для виконання складних математичних і наукових розрахунків. Найбільш оптимальним варіантом є pipenv, оскільки це вдосконалена версія іншого віртуального середовища.

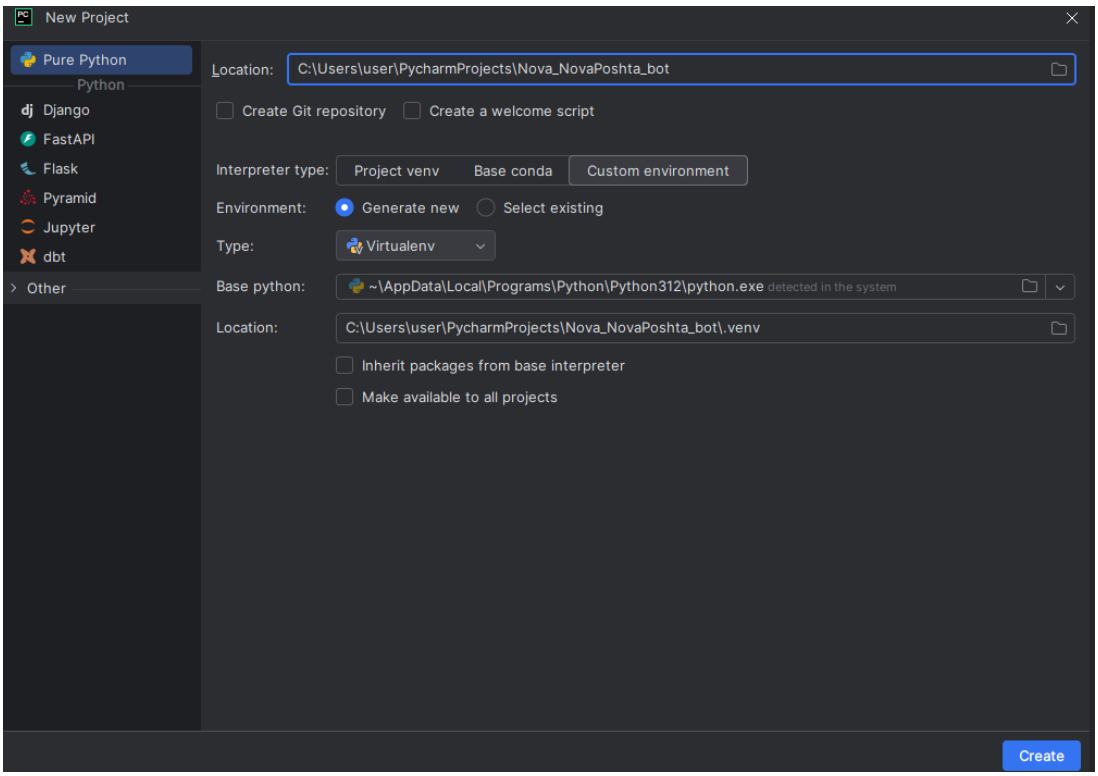


Рисунок 3.10 – Налаштування проєкту в середовищі розробки

Вибрано основний інтерпретатор. Коли ви створюєте новий проєкт, середовище розробки автоматично пропонує останню версію інтерпретатора, який знайдено на вашому комп'ютері. Це і є мета встановлення Python. Інтерпретатор – це програма, яка слугує інструментом для виконання кодів, написаних у середовищі розробки. Він обробляє весь

код, починаючи з першого рядка і до останнього, перетворюючи логіку, закладену в програмі, на машинний код, який і виконує цю програму.

Зазвичай складне програмне забезпечення створюється за допомогою мов програмування високого рівня. Ці мови потім перекладаються в машинний код, який процесор комп'ютера може зрозуміти і виконати. У процесі розробки програмного забезпечення розробники постійно модифікують і вдосконалюють вихідний код. При використанні компіляторів кожна зміна у вихідному коді вимагає від компілятора зібрати всі файли, перш ніж програма може бути запущена. Отже, якщо код програми великий, процес компіляції може зайняти значну кількість часу. З іншого боку, інтерпретаторам не потрібно бачити всю програму одразу; вони зосереджуються лише на коді, який виконується в цей момент. Такий підхід зазвичай призводить до швидшого виконання програми (як показано на рисунку 3.11, діаграма, створена за допомогою ресурсу, призначеного для створення діаграм).

Рисунок 3.11 – Принцип роботи інтерпретатора

Pip є системою управління пакетами, призначеною для встановлення та адміністрування програмних бібліотек, написаних на Python. Після налаштування робочого середовища важливо завантажити необхідні бібліотеки. Починаючи з версії Python 3.4, pip постачається разом з

інтерпретатором. Оскільки ви використовуєте версію 3.12.3, додаткове встановлення pip не є необхідним.

Основні команди pip:

- piphelp – допомога згідно доступних команд;
- piplist – список встановлених додатків;
- pipsearch – пошук пакетів за назвою;
- pipinstallpackage_name – встановлення бібліотек та пакетів;
- pipuninstallpackage_name – видалення бібліотек та пакетів;
- pipshowpackage_name – надає інформацію про встановлену бібліотеку;
- pipinstall –U – оновлення обраних бібліотек;

Давайте почнемо. Спочатку було відкрито термінал середовища розробки. Подумайте про термінал або консоль як про текстовий спосіб взаємодії з операційною системою. Хоча ви можете використовувати командний рядок операційної системи, він не такий потужний. Тому найкраще встановлювати бібліотеки безпосередньо через інтерпретатор Python у середовищі розробки. Щоб встановити пакунки за допомогою системи керування пакунками, вам потрібно відкрити термінал і ввести команду `pip install "ім'я_пакету"` (без лапок). Процес встановлення бібліотеки TelegramBotApi у проект показано нижче (рисунок 3.12).

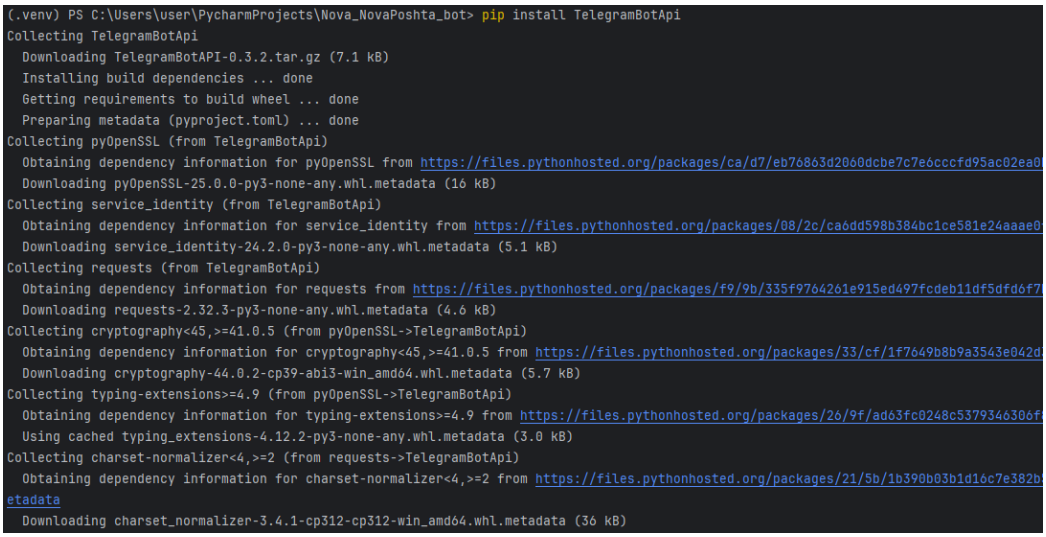


Рисунок 3.12 – Завантаження бібліотеки до проєкту

Telegram Bot API – це спеціалізований модуль, розроблений на основі Telegram, призначений для створення ботів. Його використання є досить інтуїтивно зрозумілим. Вам не потрібно занурюватися в деталі роботи протоколу шифрування Telegram [17].

Зв’язок з додатком буде здійснюватися через HTTPS-запити до проміжного сервера. Використання сторонніх бібліотек, розроблених користувачами, значно спрощує процес у порівнянні з ручним створенням HTTP-запитів. Існує два способи отримання оновлень від бота: метод getUpdates та вебхуки. Всі вхідні оновлення зберігаються на сервері протягом 24 годин, поки їх не буде оброблено.

Об’єкт Update розглядається як вхідне оновлення, яке охоплює будь-яку взаємодію користувача з ботом. У межах одного оновлення може бути присутній лише один з наведених нижче параметрів, і цей процес повторюється щоразу, коли користувачі взаємодіють з ботом у чаті.

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. Цей метод призначений для отримання оновлень за допомогою технології, що дозволяє отримувати дані про нові дії через тривалі запити. Відповідь на ці запити надається у формі масиву об'єктів Update.

Таблиця 3.2 – Команди методу getUpdates

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

Типи, які використовуються в модулі, представлені у вигляді JSON-об’єктів. На наступних малюнках (рисунки 3.13-3.15) показані різні елементи управління 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 3.13 – Об’єкт User та його команди

Цей об'єкт дозволяє отримати інформацію про ідентифікатор користувача або бота, а також їхні імена в чаті. Слід зазначити, що всі об'єкти, чутливі до регістру, повинні 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 3.14 – Об’єкт Chat та його команди

Цей об'єкт надає команди для управління інтерфейсом зв'язку між ботом і користувачем, тобто чатом. Завдяки цим командам можна змінювати тип чату, його опис, а також ідентифікувати ім'я користувача в чаті та налаштовувати інші параметри.

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 3.15 – Об’єкт Messageта його команди

Існуючий список бібліотек для вирішення поставленої задачі:

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

Для ботів, які не потребують великого обсягу трафіку та складних інтеграцій з зовнішніми ресурсами, оптимальним вибором буде пакет telebot. Він є більш простим у використанні і добре підходить для вирішення поставлених завдань. Всі бібліотеки, по суті, є варіаціями API Telegram і мають багато спільного. Проте кожна з них має свої унікальні переваги. Наприклад, aiogram є асинхронною бібліотекою, що робить її ідеальною для

реалізації складних проектів з інтеграцією додаткових ресурсів. Процес встановлення бібліотеки детально описано нижче (рис. 3.16).

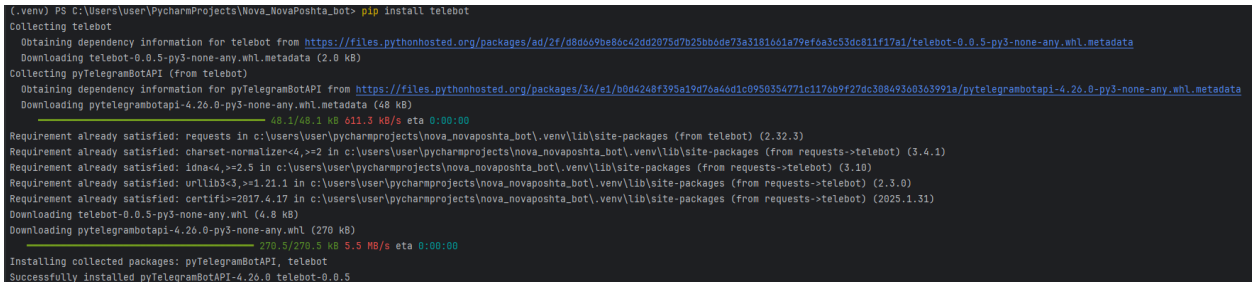


Рисунок 3.16 – Встановлення бібліотеки у папку проєкту

Бібліотека «requests» - це універсальний інструмент для створення HTTP-запитів. Її програмний інтерфейс спрощує часто складне завдання побудови цих запитів, дозволяючи переключити увагу на роботу з даними та взаємодію з сервісами. Як і в попередніх прикладах, бібліотека «requests» була інтегрована у віртуальне середовище [18].

Метод «Get» у цій бібліотеці є найбільш часто використовуваним. Він сигналізує про спробу отримати дані з певного ресурсу. Відповідь на запит «Get» майже завжди містить цінну інформацію, так зване «корисне навантаження», яке знаходиться в тілі повідомлення. Для доступу до відповіді на запит використовується об'єкт «Response». Цей об'єкт слугує інструментом для аналізу результатів ваших запитів, дозволяючи вам вивчати отримані дані. Коди статусу дають уявлення про результат виконання запиту. Код статусу 200 означає успішне виконання запиту. Команда для отримання статусу показана нижче (рисунок 3.17).

```
>>> response.status_code
200
```

Рисунок 3.17 – Команда коду станів та відповідь на неї

Сервер підтвердив успішність запиту користувача, повернувши код статусу 200. Це означає, що сервер обробив запит і готовий надати відповідь. Для доступу до даних в тілі запиту використовується атрибут `.content`. Це дозволяє отримати будь-які дані, що містяться в запиті. Однак для зручності використання ці дані часто потрібно перетворити в стандартний формат кодування. Відповіді сервера зазвичай надходять у форматі JSON. Хоча коди статусу пропонують значну функціональність, для їх ефективного використання необхідне глибоке розуміння фундаментальних концепцій протоколів шифрування. Після налаштування середовища розробки та встановлення необхідних бібліотек створюється основний файл Python (розширення “.py”). Цей файл імпортує основні бібліотеки та модулі. Цей процес імпорту гарантує, що компоненти і команди в цих бібліотеках функціонуватимуть правильно під час кодування, запобігаючи виникненню помилок у середовищі розробки. Модулі імпортуються безпосередньо у вікні програми, використовуючи стандартний синтаксис коду (див. рисунок 3.18).

```
main.py
1  import telebot
2  from telebot import types
3  import requests
```

Рисунок 3.18 – Імпортування бібліотек у код проєкту

Щоб бот почав відповідати на повідомлення в чаті та пропонувати свої послуги, необхідно ввести команду «/start». Після цього бот надасть меню з варіантами інформації. Перед цим слід додати токен, отриманий від BotFather, щоб конкретний бот міг отримувати команди, які будуть генеруватися в коді програми (див. рисунок 3.19).

Після надання токена доступу програмі, можна безпосередньо взаємодіяти з ботом, використовуючи команди, що містяться в бібліотеках, завантажених у віртуальне середовище проєкту. Наприклад, наступна команда дозволяє боту розпізнавати текстовий контент (див. рисунок 3.20).

```
6
7       @bot.message_hadler(content_types=['text'])
```

Рисунок 3.19 – Команда розпізнавання тексту

main.py

```
1   import telebot
2   from telebot import types
3   import requests
4
5   bot = telebot.TeleBot('7243990700:AAFLRhd5ZPQQX8G9gc7ISAUB6FbhK-XWy8o')
```

Рисунок 3.20 – Команда на додавання токену у програму

Після того, як ми з'ясували, як працює команда "/start", наступним важливим кроком стала візуалізація загального потоку роботи бота. Аналіз різних дизайнів інтерфейсу для вікна чату привів до рішення додати кнопки. Таке доповнення дозволило б користувачам легко переходити між різними інформаційними розділами, створивши зручну навігаційну систему. Для початку важливо зрозуміти, які типи кнопок доступні в додатку Telegram. Один з них - "Розмітка клавіатури для відповіді", яка, по суті, представляє собою заздалегідь визначені шаблони повідомлень. Бот ставить користувачеві запитання,

пропонуючи текстові варіанти відповідей, які користувач може вибрати і просуватися по структурі бота. Користувачі можуть або вводити свої відповіді вручну, або просто натискати на відповідні кнопки в інтерфейсі бота. Ці кнопки є суто функціональними, вони не містять жодної додаткової інформації. Після натискання кнопки бот отримує саме той текст, який відображається на ній. Приклад створення такої кнопки показано нижче (рисунок 3.21).

main.py

```
import telebot
from telebot import types

bot = telebot.TeleBot('7243990700:AAFLRhd5ZPQQX869gc7ISAUB6FbhK-XWy8o')

button_hi = types.KeyboardButton('Привіт! 👋')
greet_kb = types.ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)
greet_kb.add(button_hi)

@bot.message_handler(commands=['start'])
def start(message):
    bot.send_message(message.chat.id, text: "Вітаю у єдиному офіційному боті Нової Пошти у телеграм! 👋", reply_markup=greet_kb)

@bot.message_handler(func=lambda message: message.text == "Привіт! 👋")
def reply_hello(message):
    bot.send_message(message.chat.id, text: "Привіт! Як я можу допомогти? 😊", reply_markup=types.ReplyKeyboardRemove())

bot.polling()
```

Рисунок 3.21 – Приклад написання шаблонних кнопок

У цьому випадку представлено приклад створення шаблонної кнопки з використанням альтернативної бібліотеки для роботи з Telegram, проте суть залишається незмінною. Ця кнопка здатна лише відправляти в чат текст, який на ній написано, і більше підходить для простих чат-ботів, основним завданням яких є виконання базових функцій, таких як взаємодія з користувачем (дивитися рисунок 3.22).

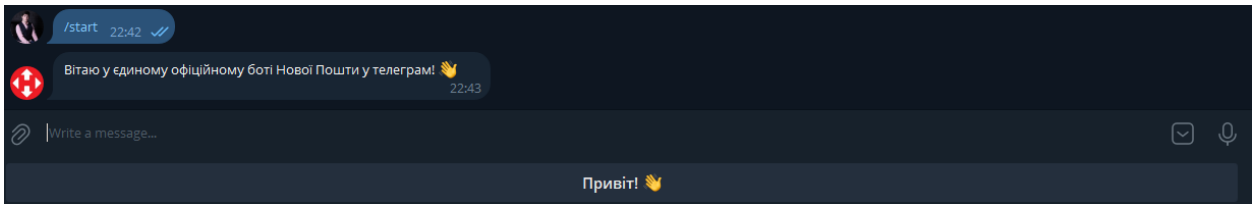


Рисунок 3.22 – Зовнішній вигляд шаблонної кнопки

Вона знаходиться в меню керування і виглядає як звичайна клавіатура - така, якою ми користуємося щодня, щоб писати повідомлення друзям або переглядати веб-сторінки. Ця "Вбудована розкладка клавіатури" - це багатофункціональна клавіатура з розширеними можливостями. Вона призначена для безпосередньої взаємодії з повідомленням або командою, до якої вона прикріплена. Ці кнопки можуть багато чого: відкривати посилання в Telegram, перенаправляти користувачів на зовнішні веб-сайти, створювати форми для опитувань тощо. Приклад створення такої кнопки показано нижче (рисунок 3.23).

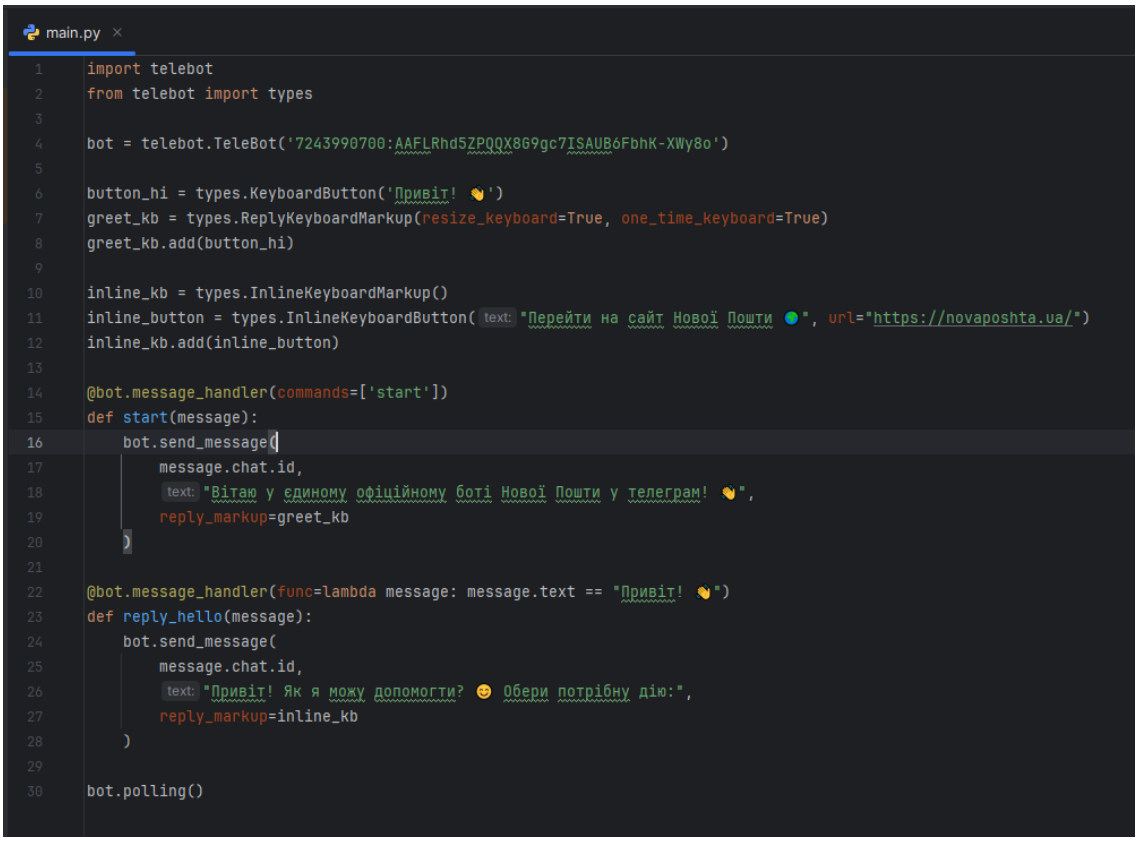


Рисунок 3.23 – Приклад коду з inline-кнопками

Назва методу створення цих кнопок вказує на те, що вони розташовані безпосередньо у вікні чатбота, маючи форму округленого прямокутника. Зовнішній вигляд кнопки приблизно відповідає зображенню, представленому на рисунку 3.24.

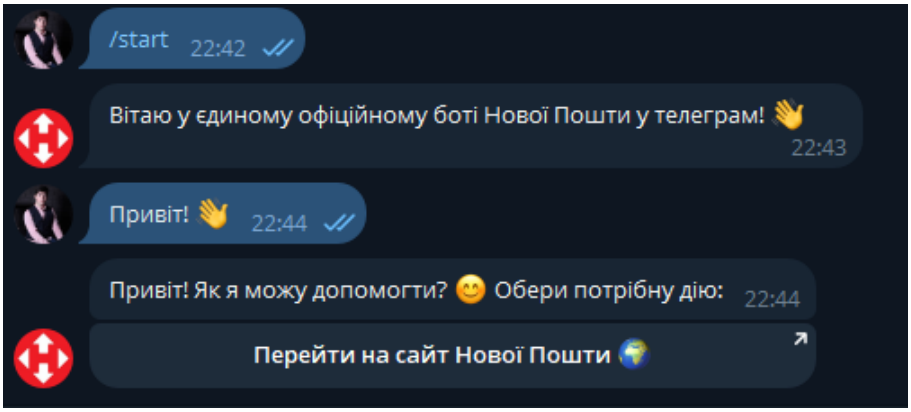


Рисунок 3.24 – Зовнішній вигляд inline-кнопки

− кнопка «/start» для виклику головного меню бота;

− головне навігаційне меню з доступом до основних послуг Нової Пошти;

− кнопка «Знайти відділення», яка дозволяє швидко знайти найближче відділення Нової Пошти за допомогою геолокації або введення назви міста;

− реалізовано можливість відстеження посилки за номером ТТН, що дозволяє користувачам швидко отримувати актуальний статус доставки;

− додано функцію розрахунку вартості доставки, яка дозволяє дізнатися приблизну ціну перевезення, вказавши параметри посилки;

− кнопка «Графік роботи», яка надає інформацію про години роботи відділень у вибраному місті;

− можливість оформлення експрес-накладної без відвідування відділення, що значно економить час клієнтів;

− функція виклику кур’єра для відправлення посилки без необхідності самостійного візиту у відділення;

− кнопка «Допомога», яка містить відповіді на найпоширеніші запитання клієнтів;

− реалізовано інтеграцію з сайтом Нової Пошти через інлайн-кнопки, що дозволяє швидко переходити на відповідні розділи ресурсу;

− передбачено можливість зворотного зв’язку з оператором підтримки безпосередньо у боті;

− інтеграція із системою оцінювання якості обслуговування, що дозволяє користувачам залишати відгуки після отримання послуги.

На рисунку 3.25 ілюструється процес автентифікації користувача на серверній платформі Heroku, здійснюваний безпосередньо з інтегрованого середовища розробки PyCharm. Цей етап надає розробнику можливість з'єднати свій локальний проєкт з хмарною інфраструктурою Heroku, що є необхідним для подальшого розгортання застосунку. Після успішного входу

відкривається доступ до управління додатками, деплою та налаштуванням хостингу, що дозволяє уникнути переходу до вебінтерфейсу Heroku.

Рисунок 3.25 – Вхід на серверну платформу Heroku через PyCharm

На рисунку 3.26 показано, як розміщений код чатбота в репозиторії на платформі GitHub. Це створює зручний доступ до вихідного коду, сприяє ефективній співпраці з іншими розробниками, дозволяє відстежувати зміни, здійснювати контроль версій і забезпечує прозорість проєкту. Використовуючи GitHub, можна без труднощів впроваджувати оновлення, вносити корективи в код, а також застосовувати систему pull-запитів для рецензування та обговорення змін у команді.

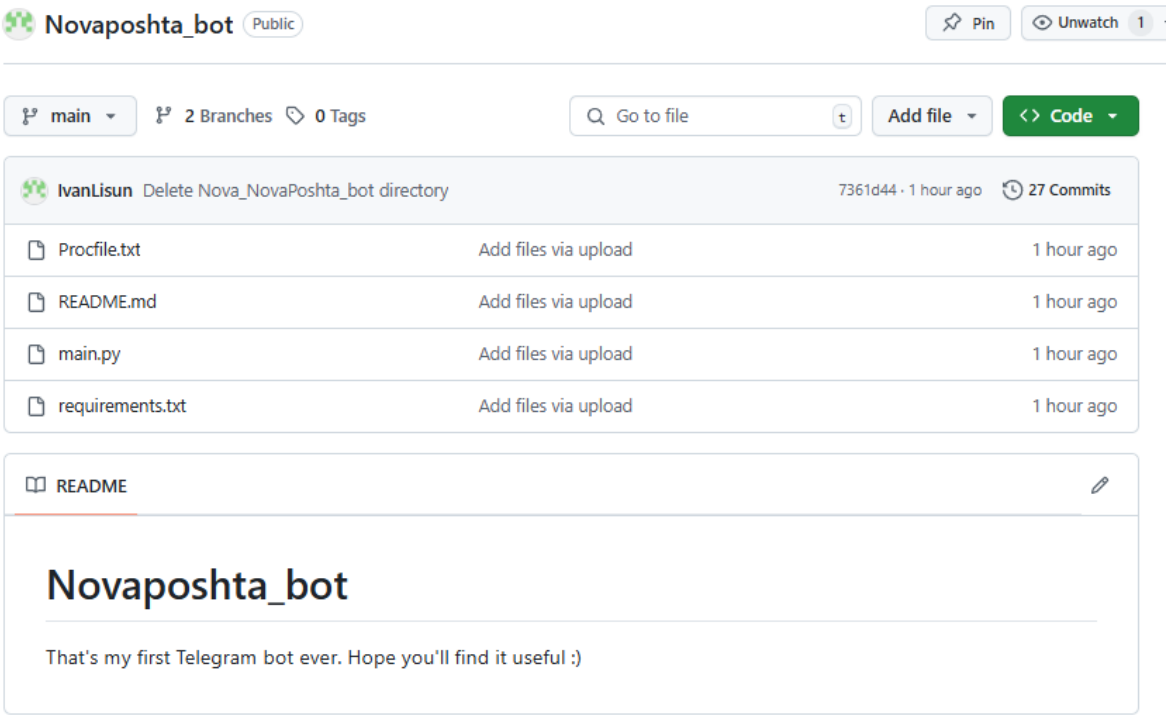


Рисунок 3.26 – Код бота на Github

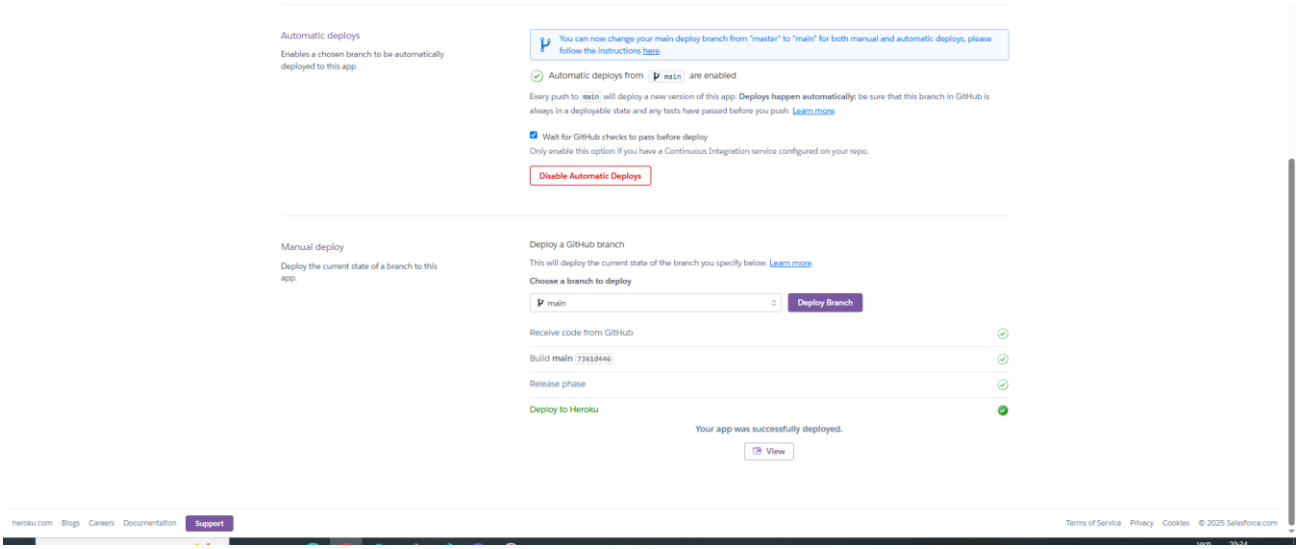


Рисунок 3.27 – Розгортання бота на серверній платформі Heroku через Pycharm

На рисунку 3.27 представлено процес розгортання чатбота на серверній платформі Heroku безпосередньо з середовища розробки PyCharm. Цей метод дозволяє автоматизувати процес розміщення програми, поєднуючи локальний проєкт із хмарною інфраструктурою. Використовуючи відповідні плагіни або термінал у PyCharm, розробник може підключитися до Heroku, налаштувати змінні середовища, внести зміни через push та запустити бота в реальному середовищі. Це значно спрощує процес деплою, роблячи управління та оновлення програми більш зручними.

РОЗДІЛ 4 ОГЛЯД ФУНКЦІОНАЛЬНОЇ ЧАСТИНИ. ТЕСТУВАННЯ РОБОТИ БОТА

Наступним етапом роботи стало формування тест-плану.

Тест-план – це документ або частина документу, що описує повний обсяг тестових робіт, таких як:

- об’єкт тестування;
- стратегії розвитку;
- графік проведення тестів;
- перевірка працездатності;
- оцінка ризиків та шляхи їх усунення.

Нижче представлено опис процесу тестування. Тестування чатбота Нової пошти здійснювалося згідно з розробленим тест-планом, ознайомитися з яким можна в додатку Б [19].

У процесі створення програмного забезпечення тестування проводилося безперервно, що дозволяло оперативно виявляти та виправляти помилки під час написання коду. Основні етапи тестування включали перевірку функціонування інтерфейсу, адекватності реакцій бота на запити користувачів, швидкості відповідей, а також коректності побудови навігаційної структури з кнопками для вибору типових дій, таких як пошук відділення, перевірка ТТН та отримання контактної інформації.

Перший етап тестування полягав у перевірці реакції бота на запити користувачів. Для цього було виконано HTTP-запит через API Telegram до бота Нової пошти. У адресному рядку браузера був введений запит у форматі getMe, що надає загальну інформацію про бота. Після відправлення запиту браузер відобразив сторінку з короткою JSON-відповіддю (дивитися рисунок 4.1).

Цей код був скопійований і вставлений у вебредактор JSON, що

забезпечує зручний перегляд структури відповіді. Виходячи з отриманих даних (дивитися рисунок 4.2), можна стверджувати, що бот успішно зареєстрований у Telegram, адекватно обробляє запити, підтримує зв'язок із сервером і готовий до взаємодії з користувачами.

Онлайн JSON-редактор – це інструмент, доступний у веб-форматі, який дозволяє аналізувати кодову структуру, перевіряти коректність форматування та взаємодіяти з API-відповідями. У цьому випадку редактор використали для обробки даних, отриманих у відповідь на запит getMe, що підтвердило готовність чатбота до наступних етапів інтеграції та тестування [20].

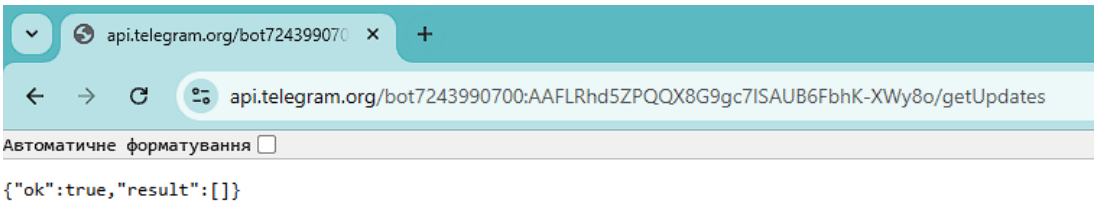


Рисунок 4.1 – Відповідь сервера телеграм на поданий запит

Перенісши цей код у JSON-редактор і запустивши програму, можна отримати більш чітке уявлення, представлене у вигляді ланцюгів (див. рисунок 4.2).

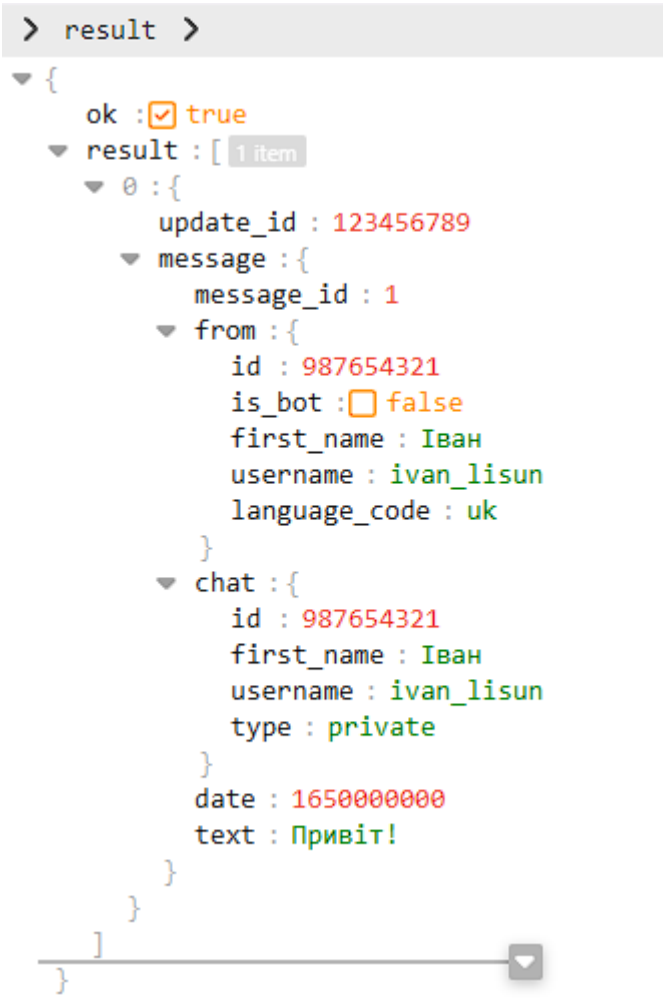


Рисунок 4.2 – Дерево ланцюгів відповіді сервера

Після аналізу коду було встановлено, що сервер Telegram повністю інтегрований з ботом, що свідчить про правильну його налаштування та підключення до додатку. Сервер відповідає статусом «ok»: true, що підтверджує успішне виконання запиту та його отримання. У випадку помилки сервер надав би відповідне повідомлення про помилку (див. рисунок 4.3).

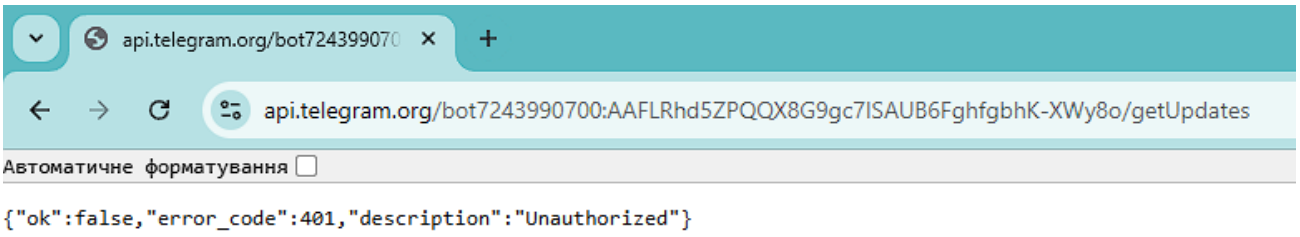


Рисунок 4.3 – Відмова сервера на запит

Змінивши код, зокрема видаливши кілька символів з URL запиту, можна виявити, що сервер не зміг знайти бота і, відповідно, не може надати позитивну відповідь. Це відбувається, коли сервер не вдається ідентифікувати бота. У таких випадках код відповіді на запит повертає помилку HTTP 401 Unauthorized, що свідчить про те, що запит не був прийнятий через відсутність необхідних особистих даних для доступу до ресурсу. Після перевірки взаємодії бота з серверною частиною Telegram і підтвердження його доступності в мережі, проводиться аналіз навігації та логіки команд в інтерфейсі. Для цього потрібно відкрити додаток Telegram на комп'ютері або мобільному пристрої та знайти бота, ввівши його тег у пошуковій стрічці (див. рисунок 4.4).

Рисунок 4.4 – Тег бота у додатку

Перебуваючи в чаті відповідного бота, після введення команди для старту (дивитися рисунок 4.5), розпочалася перевірка роботи кнопок та точності відповідей бота.

Рисунок 4.5 – Інтерфейс чату бота

Відповідь бота на першу команду зайняла кілька секунд, що може статися через об'ємний код. Результат, який було отримано, є коректним і відповідає запланованому меню, яке програма вперше демонструє у вигляді головного меню (див. рисунок 4.6). Меню представлено у формі текстового блоку з інформацією, зображенням та кількома кнопками для подальшої навігації в додатку. Як видно з часу надсилання повідомлень, бот відповів на запит користувача миттєво. Оскільки навантаження на користувачів невелике, програма здатна швидко надавати результати. Проте, якщо кількість запитів зросте, час очікування обробки може збільшитися, і в такому випадку знадобиться шукати альтернативні рішення.

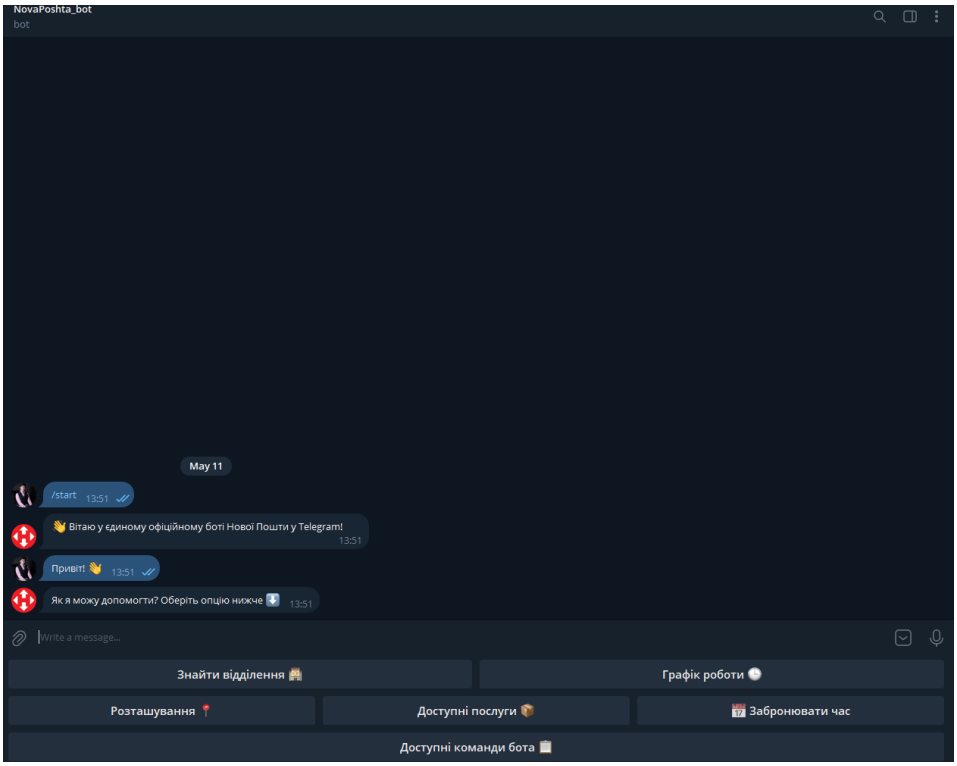


Рисунок 4.6 – Відповідь бота на стартову команду

Тепер ми можемо обирати команди, якими хочемо скористатись. Перевіримо кнопку «Знайти відділення»:

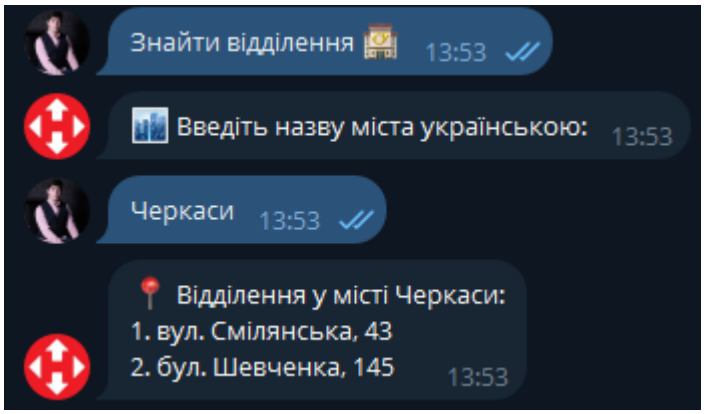


Рисунок 4.7 – Відповідь бота на команду «Знайти відділення»

Тепер перевіряємо команду «Графік роботи»:

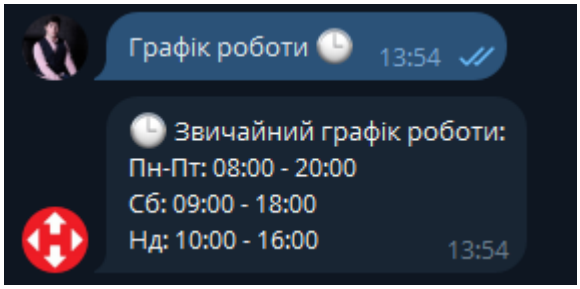


Рисунок 4.8 – Відповідь бота на команду «Графік роботи»

Перевіряємо команду «Розташування», яка повинна показати нам відділення у Києві на мапі та через яку можна перейти на вебсайт Нової Пошти:

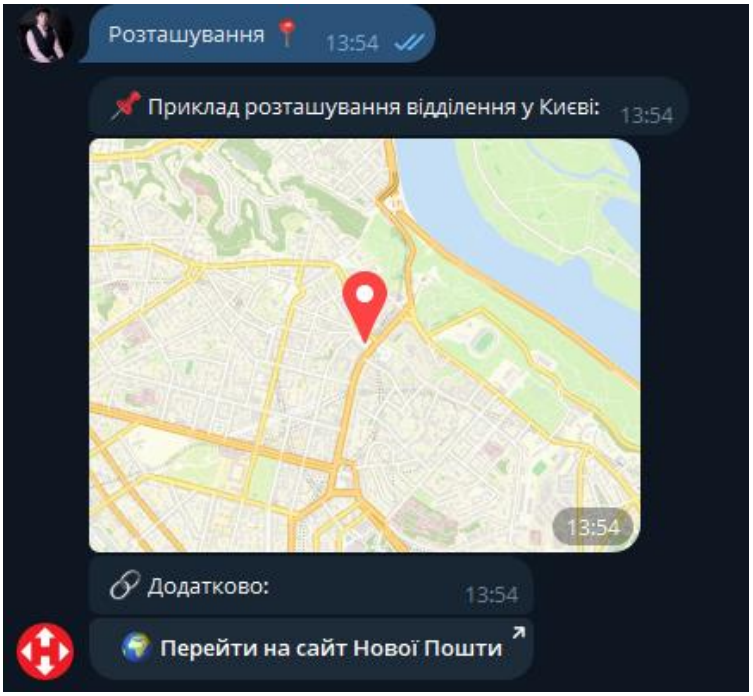


Рисунок 4.9 – Відповідь бота на команду «Розташування»

Перевіряємо команду «Забронювати час»:

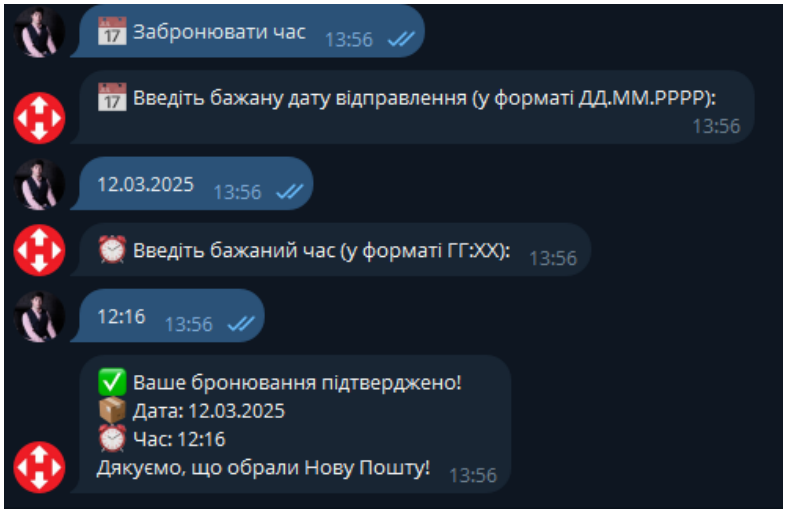


Рисунок 4.10 – Відповідь бота на команду «Забронювати час»

Перевіряємо команду «Відправлення та отримання»:

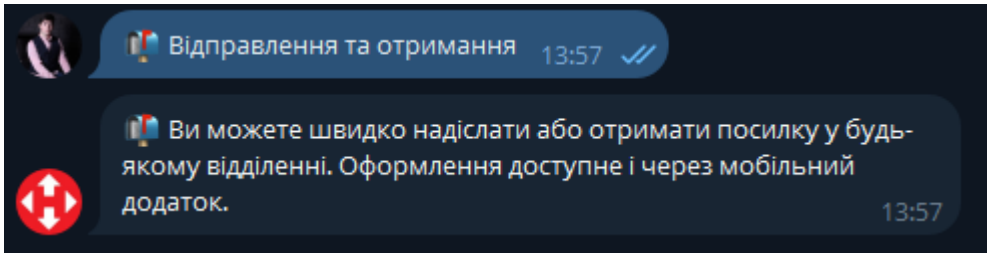


Рисунок 4.11 – Відповідь бота на команду «Відправлення та отримання»

Перевіряємо команду «Післяплата»:

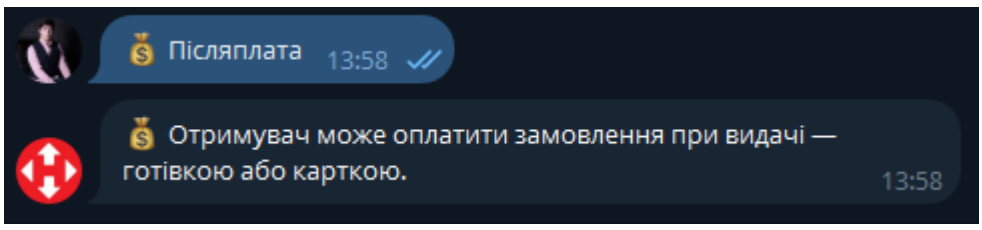


Рисунок 4.12 – Відповідь бота на команду «Післяплата»

Перевіряємо команду «Кур’єрська доставка»:

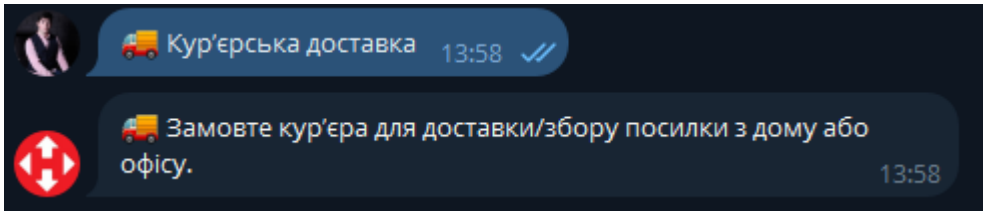


Рисунок 4.13 – Відповідь бота на команду «Кур’єрська доставка»

Перевіряємо команду «Зберігання посилок»:

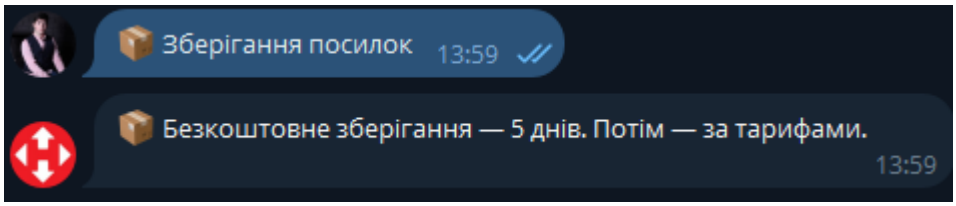


Рисунок 4.14 – Відповідь бота на команду «Зберігання посилок»

Перевіряємо команду «Переадресація»:

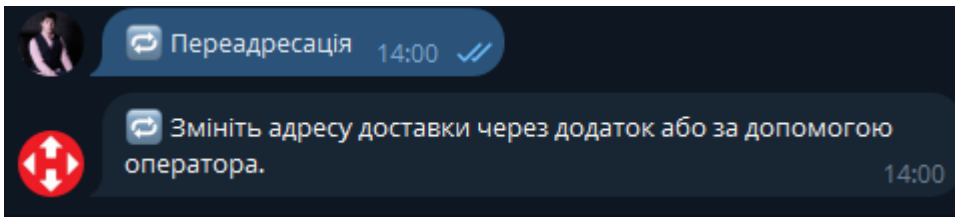


Рисунок 4.15 – Відповідь бота на команду «Переадресація»

Тепер перевіряємо команди, список яких відображається при введенні команди /commands:

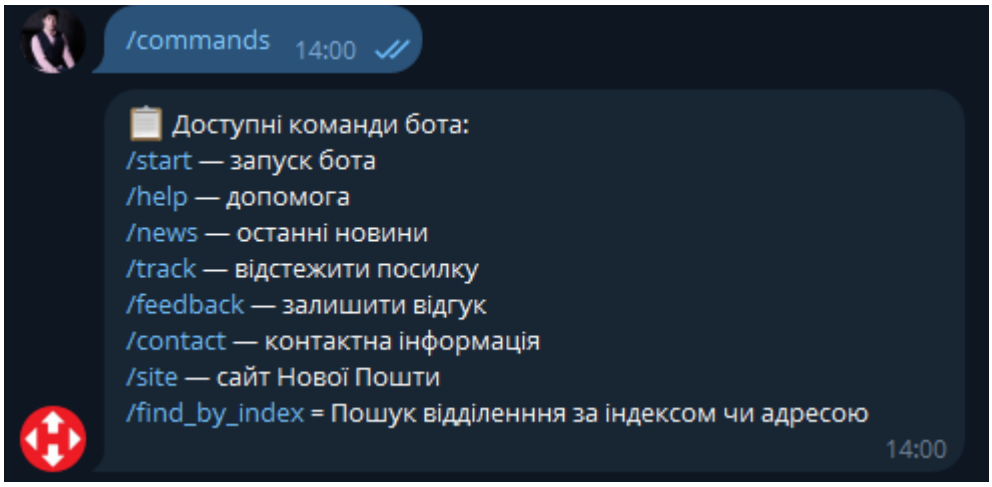


Рисунок 4.16 – Відповідь бота на команду /commands

Команда /help, яка показує доступні команди:

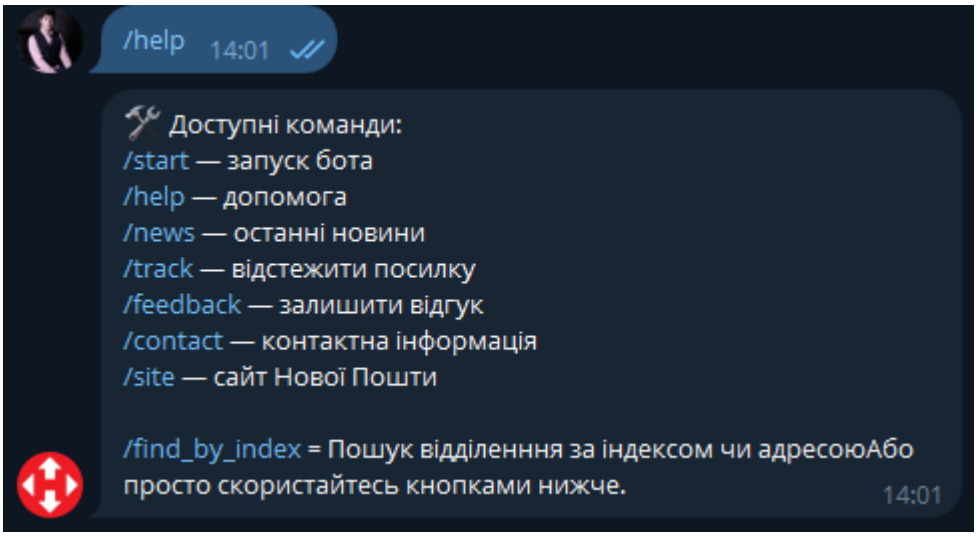


Рисунок 4.17 – Відповідь бота на команду /help

Команда /news:

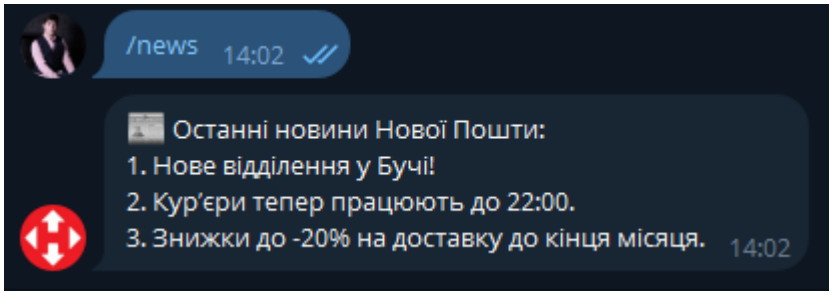


Рисунок 4.18 – Відповідь бота на команду /news

Команда /track:

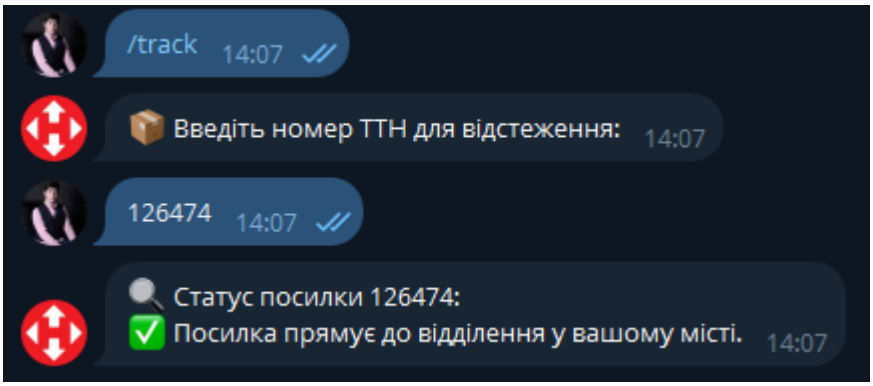


Рисунок 4.19 – Відповідь бота на команду /track

Команда /feedback:

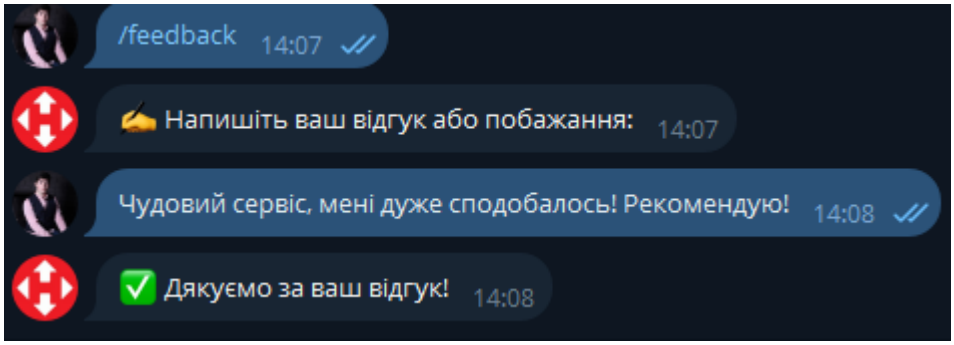


Рисунок 4.20 – Відповідь бота на команду /feedback

Команда /contact:

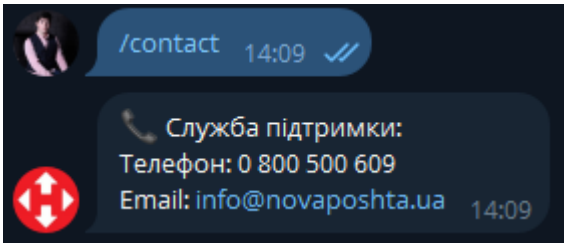


Рисунок 4.21 – Відповідь бота на команду /contact

Команда /site:

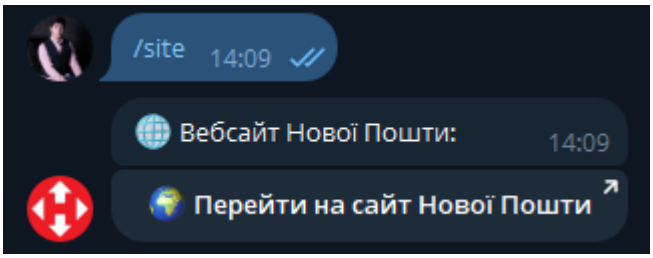


Рисунок 4.22 – Відповідь бота на команду /site

Команда /find_by_index:

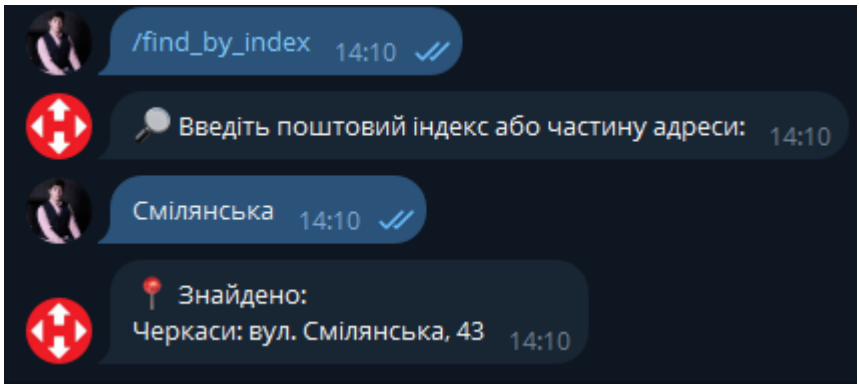


Рисунок 4.23 – Відповідь бота на команду /find_by_index

ВИСНОВКИ

Метою даного кваліфікаційного проєкту стало розроблення Telegram чатбота для компанії «Нова Пошта» з використанням мови програмування Python. Під час реалізації проєкту було проведено аналіз сучасного ринку логістичних послуг та цифрових рішень у сфері обслуговування клієнтів. Окрім того, детально вивчено ринок чатботів, їх класифікацію, функціональні можливості та актуальні тенденції впровадження в бізнес-процеси. Виявлено, що чатботи активно використовуються в електронній комерції, банківських послугах, доставці та технічній підтримці як ефективний засіб автоматизації взаємодії з клієнтами.

Дослідження показали, що використання автоматизованих помічників суттєво знижує навантаження на операторів служби підтримки, покращує якість обслуговування та прискорює обробку запитів. Telegram був обраний як платформа через зручний API, велике охоплення користувачів і популярність цього месенджера серед клієнтів Нової Пошти.

Чатбот реалізовано з використанням бібліотек telebot, types та модулів datetime. Він має інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам здійснювати такі дії:

- знайти найближче відділення у своєму місті;
- дізнатися графік роботи;
- переглянути доступні послуги компанії;
- залишити відгук або отримати контактну інформацію;
- забронювати час для відправки;
- відстежити відправлення за номером ТТН;
- перейти на офіційний сайт компанії для отримання додаткової інформації.

Також реалізовано розширений пошук відділень за індексом або частиною адреси. Бот було офіційно зареєстровано в Telegram під ідентифікатором @Novaposhta1_bot.

Під час розробки проводилися тестування всіх функціональних компонентів. Це включало перевірку адекватності реакцій бота на команди, запити, а також на некоректні формати дати та часу, а також обробку введення користувача. Усі модулі пройшли ручну перевірку на правильність функціонування. Для цього використовувалися перевірка HTTP-відповідей і внутрішня обробка повідомлень.

Проєкт має можливості для подальшого вдосконалення, зокрема через інтеграцію з API системи, що забезпечує відстеження в реальному часі, підключення картографічних сервісів для точного визначення місцезнаходження відділень, а також впровадження персоналізованих повідомлень і аналітики звернень.

В цілому, проєктні цілі були успішно реалізовані: створено надійного бота, який підвищує якість обслуговування клієнтів Нової Пошти та слугує яскравим прикладом ефективного використання Python в автоматизації сервісних рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Лісун І.О., Немченко В.Ю. Розробка Telegram чатбота для роботи відділень Нової пошти. Матеріали XVII Студентської науково-практичної конференції студентів, аспірантів та молодих вчених за тематикою «Тенденції розвитку ІТ-технологій в Україні». 26-27 березня 2025 р., Черкаси, Україна. Черкаси : Черкаський державний фаховий бізнес-коледж, 2025. С. 120-121. URL: http://csbc.edu.ua/documents/student/260325.pdf (дата звернення: 27.03.2025).

2.Нова Пошта. Офіційний сайт.URL: https://novaposhta.ua/ (дата звернення: 31.03.2025).

3.Нова пошта. API для доступу до інформації про відділення.URL: https://developers.novaposhta.ua/ (дата звернення: 01.04.2025).

4.SendPulse.Що таке чатбот: визначення, види, як створити та застосовувати.

URL: https://sendpulse.ua/support/glossary/chatbot (дата звернення: 02.04.2025)

5.AllTheResearch. Chatbot Market 2018-2027.URL: https://rss.globenewswire.com/en/news-release/2021/09/22/2301477/0/en/Chatbot-Market-is-expected-to-reach-USD-14-9-Bn-by-2027-with-a-growing-CAGR-of-24-60-AllTheResearch.html (date of appeal: 02.04.2025).

6.Рейтинг мобільних додатків у січні 2025 року.

URL:https://www.village.com.ua/village/business/news/307845-naypopulyarnishi-mobilni-zastosunki-sered-ukrayintsiv-u-sichni-2021-roku (дата звернення: 03.04.2025).

7.Telegram System Architecture.URL:https://readmedium.com/telegram-system-architecture-ddf9f7d358de (дата звернення: 03.04.2025)

8. Suárez-Díaz, J. L., Rodríguez-Barroso, N., & Gómez-Trenado, G. (2024). *Interactive Telegram Bot as a Support Tool for Teaching in Higher Education*. Lecture

Notes in Networks and Systems, 957, 352–361. URL: https://link.springer.com/chapter/10.1007/978-3-031-75016-8_33 (date of appeal: 04.04.2025).

9.Python 3.13.2 documentation.URL: https://docs.python.org/release/3.13.2/ (date of appeal: 11.04.2025).

10.Java documentation.URL: https://docs.oracle.com/javase/8/docs/api/ (date of appeal: 11.04.2025).

11.VS Code documentation.URL: https://code.visualstudio.com/docs (date of appeal: 13.04.2025).

12.Sublime Text 3 documentation.URL: https://www.sublimetext.com/docs/ (date of appeal: 13.04.2025).

13.Pycharm documentation.URL: https://www.jetbrains.com/help/pycharm/getting-started.html (date of appeal: 13.04.2025).

14.Команди для налаштування телеграм-бота.

URL: https://lessondelivery.org/chatbot/chat-boti-telegram-nalashtuvannya.html (date of appeal: 15.04.2025).

15.Heroku documentation.URL:https://devcenter.heroku.com/ (date of appeal: 17.04.2025).

16.IDEFO method.URL: National Institute of Standards and Technology. Integration Definition for Function Modeling (IDEF0). Federal Information Processing Standards Publication 183.URL: https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub183.pdf

(date of appeal: 22.04.2025)

17.Telegram Bot API. Офіційна документація.

URL: https://core.telegram.org/bots/api (date of appeal: 24.04.2025).

18.Requests python documentation.URL: https://pypi.org/project/requests/ (date of appeal: 05.05.2025).

19.Test plan.URL: https://qalight.ua/baza-znaniy/test-plan-2/ (date of appeal: 10.05.2025).

20.Онлайн редактор JSON-коду.URL: https://jsoneditoronline.org/ (date of appeal: 15.05.2025).

ДОДАТКИ

ДОДАТОК А ДІАГРАМА ДЕРЕВА ВУЗЛІВ НАВІГАЦІЇ БОТУ

На рис. А.1 зображено діаграму дерева вузлів, що слугує навігацією сторінками боту.

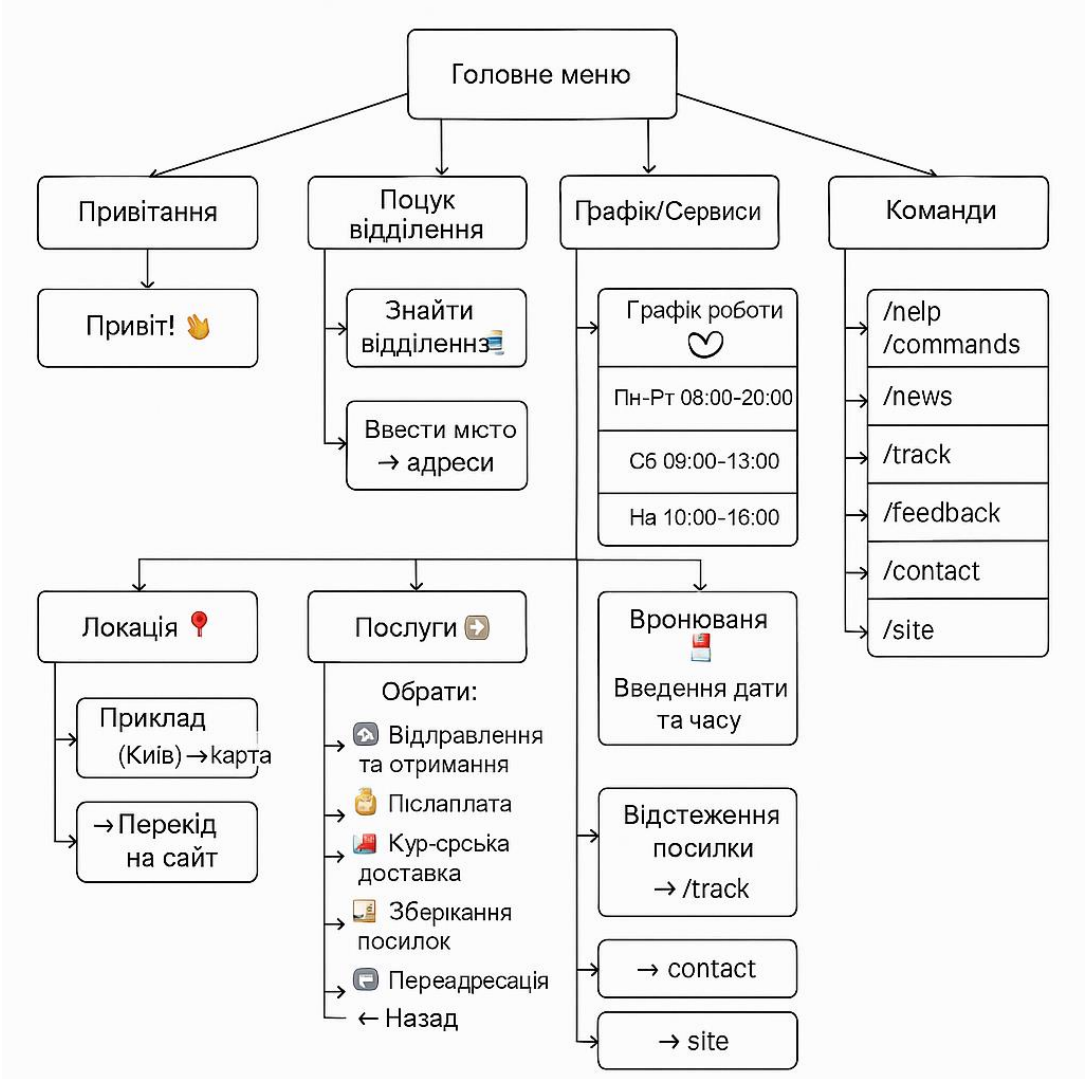
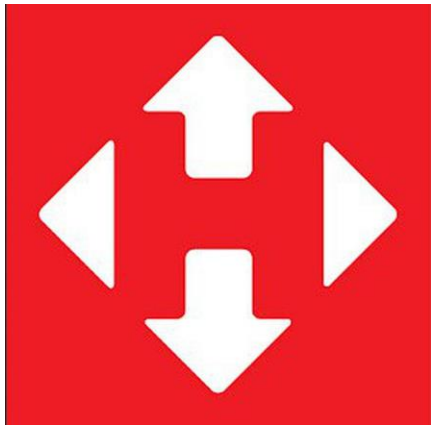


Рис А.1 – Діаграма дерева вузлів

ДОДАТОК Б ТЕСТ-ПЛАН ПЕРЕВІРКИ РОБОТИ ЧАТБОТА



Telegram Chatbot «NovaPoshta_bot»

Тест план

Версія 1.0

Історія оновлень

Дата	Версія	Вид оновлення	Автор
11.04.2025	0.10	Створення базового функціоналу: запуск бота, просте меню	Лісун І.О
14.04.2025	0.20	Додавання пошуку відділень за містом	Лісун І.О
16.04.2025	0.30	Реалізація команди /help і базових інформаційних команд	Лісун І.О
20.04.2025	0.40	Додано кнопки для швидких відповідей, покращено UI	Лісун І.О
22.04.2025	0.50	Додано бронювання часу з валідацією дати і часу	Лісун І.О
25.04.2025	0.60	Реалізовано інлайн кнопку для переходу на сайт Нової Пошти	Лісун І.О
29.04.2025	0.70	Додано послугу відображення розташування відділення	Лісун І.О
06.05.2025	0.80	Додано команду /track для відстеження посилки	Лісун І.О
10.05.2025	0.90	Впроваджено збір відгуків через команду /feedback	Лісун І.О
12.05.2025	0.99	Покращена навігація по меню, додано кнопку "Доступні команди"	Лісун І.О
15.05.2025	1.0	Повна рефакторинг коду, оптимізація обробників повідомлень, завершення проєкту	Лісун І.О

Зміст

1. Вступ 73
1.1 Мета 73
1.2 Вихідні дані 73
1.3 Мета тестування 73
2. Умови для тестування 74
3. Стратегія процесу тестування 74
3.1. Типи тестування 75
4. План проведення робіт 76
5. Підведення підсумків 77
5.1 Висновок 77

1. Вступ

1.1 Мета

Метою створення даного Тест Плану є опис процесу тестування Telegram чатбота «Нова Пошта» (повний доступ: @Novaposhta1_bot). Цей документ забезпечує структурований підхід до перевірки функціональності бота, дозволяючи сформувати чітке уявлення про стан проєкту на поточному етапі.

1.2 Вихідні дані

«Нова Пошта» – Telegram чатбот, створений для надання користувачам швидкого доступу до основних сервісів компанії: пошуку відділень, перегляду графіка роботи, бронювання часу, перегляду послуг, новин, контактної інформації тощо. Мета бота – полегшити взаємодію з сервісами Нової Пошти прямо у додатку Telegram.

1.3 Мета тестування

Метою тестування є перевірка коректної роботи всіх функцій бота, таких як:

- Обробка стандартних команд: /start, /help, /news, /track, /feedback, /contact, /site, /find_by_index;
- Робота з кнопками та меню;
- Коректність пошуку відділень по містах;
- Можливість бронювання дати та часу;
- Надання інформації про послуги, розташування, графік роботи;
- Швидкість обробки запитів та стабільність взаємодії.
- Результатом тестування буде:
- Висновок щодо готовності бота до розгортання;

- Тестовий звіт з описом знайдених помилок і рівня відповідності очікуваному функціоналу.

Тестування проводиться вручну, у форматі ad-hoc з точки зору звичайного користувача Telegram. Такий підхід дозволяє виявити реальні сценарії взаємодії і помилки, які могли бути пропущені при автоматизованому або формальному тестуванні.

2. Умови для тестування

Бот повинен ефективно задовольняти потреби користувача, забезпечуючи швидку відповідь на запити, зручність у користуванні та інтуїтивно зрозумілий інтерфейс. Особливу увагу приділяється простоті взаємодії, коректності відповідей і адаптивності під різні пристрої.

3. Стратегія процесу тестування

Оскільки відсутня детальна специфікація та існують обмеження в ресурсах для повноцінного тестування, буде застосовано ad-hoc тестування. Це дозволить виявити ключові помилки під час природної взаємодії з ботом. Тестування проводиться на комп’ютері (через вебінтерфейс або емулятор), а також додатково на мобільному пристрої, щоб перевірити адаптивність і зручність використання на різних екранах.Характеристики пристроїв для перевірки:

Домашній комп’ютер:

ОС – Windows 10;

Процесор – Ryzen 5 1600 3.60GHz;

Оперативна пам’ять (ОЗУ) – 16,00 ГБ;

Тип системи – 64-розрядна операційна система;

Мобільний пристрій:

Марка – Iphone;

Модель – Iphone 11;

Версія IOS – 17.6.1;

Оперативна пам’ять – 4 ГБ;

Стандарти зв’язку –4G (LTE);

Процесор – Apple A13 Bionic;

3.1. Типи тестування

Функціональне, UI та інтеграційне тестування.

Мета: виявити функціональні помилки у роботі бота, зокрема неправильну логіку навігації, некоректну видачу інформації, помилки під час надсилання зображень, а також перевірити правильність роботи тестових завдань, реалізованих у боті.

Опис процесу:

Перше звернення до бота (/start):

– Перевірка швидкості реакції бота на перший запит.

– Перевірка коректності вітального повідомлення та доступності подальших дій.

Перевірка роботи кнопок:

– Натискання на кожну кнопку та перевірка, чи відповідає результат очікуваному запиту.

– Перевірка логічного порядку надання інформації.

– Перевірка того, що кнопки мають послідовне, невипадкове розташування.

Перевірка тестових питань:

– Перевірка, чи надсилаються тестові запитання в правильному місці діалогу.

– Поведінка бота при правильній відповіді (правильна реакція, оновлення стану).

– Поведінка бота при неправильній відповіді (пояснення, підказка або перехід до наступного етапу).

– Перевірка коректної роботи кнопки підтвердження відповіді.

– Навігація після вибору відповіді (перехід до наступного питання, повідомлення про завершення тощо).

Перевірка відповіді сервера Telegram:

– Надсилання запиту до Telegram API.

– Перегляд відповіді у форматі JSON для аналізу структури, помилок або затримок у відповідях.

4. План проведення робіт

Завдання	**Об’єм роботи**	**Дата початку**	**Дата завершення**
Написання тест плану	6 годин	09.05.2025	09.05.2025
Проведення тестування	2 години	12.05.2025	12.05.2025
Аналіз тестування	1.5 години	15.05.2025	15.05.2025
Створення висновків	1 година	22.05.2025	22.05.2025

5. Підведення підсумків

5.1 Висновок

У результаті проведеного функціонального тестування Telegram чатбота було перевірено коректність його роботи, зокрема: реакцію на команду /start, стабільність функціонування кнопок, логіку проходження тестових завдань, а також відправку зображень. Тестування проводилося як на комп’ютері, так і на мобільному пристрої, що дозволило оцінити зручність користування ботом на різних платформах. Детальний опис процесу тестування наведений у розділі 3 цієї роботи.

ДОДАТОК В ПРОГРАМНА РЕАЛІЗАЦІЯ ТА РОЗГОРТАННЯ БОТА

```
import telebot
from telebot import types
from datetime import datetime
import requests # Import the requests library

bot = telebot.TeleBot('7243990700:AAFLRhd5ZPQQX8G9gc7ISAUB6FbhK-XWy8o')

# === Nova Poshta API Key ===
NOVA_POSHTA_API_KEY = '8d0b69fd759e773e2a3fd05a7e186c61'

# === Кнопки ===
button_hi = types.KeyboardButton('Привіт! 👋')
button_commands = types.KeyboardButton('Доступні команди бота 🛍')  # нова кнопка
greet_kb = types.ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)
greet_kb.add(button_hi)

main_kb = types.ReplyKeyboardMarkup(resize_keyboard=True)
main_kb.add(
    types.KeyboardButton("Знайти відділення 🏤"),
    types.KeyboardButton("Графік роботи 🕒"),
)
main_kb.add(
    types.KeyboardButton("Розташування 📍"),
    types.KeyboardButton("Доступні послуги 🛍️"),
    types.KeyboardButton("📆 Забронювати час"),
    button_commands  # додана кнопка
)

inline_kb = types.InlineKeyboardMarkup()
inline_button = types.InlineKeyboardButton("🌍 Перейти на сайт Нової Пошти", url="https://novaposhta.ua/")
inline_kb.add(inline_button)

# === Словник відділень ===
branches = {
    "київ": [
        "вул. Хрещатик, 10",
        "вул. Саксаганського, 45",
        "вул. Пирогівський шлях, 135",
        "вул. Васильківська, 34",
        "вул. Васильківська, 55",
        "вул. Межигірська, 56",
        "вул. Бережанська, 9",
        "просп. Оболонський, 21Б",
        "вул. Володимирська, 45",
        "вул. Льва Толстого, 7",
        "вул. Січових Стрільців, 55",
        "вул. Дорогожицька, 5",
        "вул. Червоноармійська, 20",
        "вул. Глибочицька, 40",
        "вул. Каунаська, 10",
        "вул. Дружби Народів, 18",
        "вул. Мельникова, 42",
        "просп. Перемоги, 66",
        "вул. Борщагівська, 101",
        "вул. Набережно-Рибальська, 12"
    ],
    "харків": ["вул. Сумська, 35",
    "просп. Науки, 21",
    "вул. Клочківська, 190",
    "вул. Пушкінська, 89",
    "вул. Мироносицька, 54",
    "вул. Академіка Павлова, 44",
    "просп. Героїв Сталінграда, 153",
    "вул. Біблика, 12",
    "вул. Римарська, 20",
    "вул. Чкалова, 16",
    "вул. Роганська, 50",
```

"вул. Валентинівська, 24",
"вул. Маршала Конєва, 12",
"вул. Культури, 10",
"вул. Плеханівська, 128",
"вул. Січеславська, 5",
"вул. Чернишевська, 18",
"вул. Академіка Проскури, 7",
"вул. Холодногірська, 119"],
"одеса": ["вул. Дерибасівська, 12",
"просп. Шевченка, 4",
"вул. Канатна, 15",
"вул. Гагаріна, 5",
"вул. Академіка Корольова, 34",
"вул. Космонавтів, 12",
"вул. Рішельєвська, 45",
"вул. Базарна, 23",
"вул. Новощепний ряд, 7",
"вул. Люстдорфська дорога, 110",
"вул. Балківська, 77",
"вул. Пушкінська, 33",
"вул. Мала Арнаутська, 29",
"вул. Варненська, 2",
"вул. Іцхака Рабіна, 16",
"вул. Артилерійська, 20",
"просп. Небесної Сотні, 9",
"вул. Генерала Петрова, 10",
"вул. Академіка Воробйова, 9",
"вул. Івана Франка, 21"],
"дніпро": ["вул. Центральна, 15",
"просп. Гагаріна, 20",
"вул. Короленка, 5",
"вул. Дмитра Яворницького, 45",
"просп. Слобожанський, 33",
"вул. Робоча, 14",
"вул. Володимира Моссаковського, 72",
"вул. Калинова, 50",
"вул. Набережна Перемоги, 19",
"вул. Пастера, 28",
"просп. Пушкіна, 85",
"вул. Академіка Янгеля, 30",
"вул. Юності, 10",
"вул. Криворізька, 37",
"вул. Чкалова, 48",
"вул. Мечникова, 16",
"вул. Каруни, 11",
"вул. Воскресенська, 3",
"вул. Леніна, 95",
"вул. Парусна, 7"],
"львів": ["вул. Городоцька, 85",
"просп. Чорновола, 67",
"вул. Степана Бандери, 20",
"вул. Личаківська, 55",
"вул. Наукова, 7",
"вул. Сахарова, 10",
"вул. Івана Франка, 40",
"вул. Вітовського, 48",
"вул. Зелених Ентузіастів, 12",
"вул. Шевченка, 30",
"вул. Княгині Ольги, 105",
"вул. Зелена, 120",
"вул. Замарстинівська, 32",
"вул. Кульпарківська, 95",
"вул. Академіка Сахарова, 14",
"вул. Мазепи, 1",
"вул. Підвальна, 18",
"вул. Джерельна, 35",
"вул. Пластова, 45",
"вул. Коновальця, 20"],
"запоріжжя": ["вул. Перемоги, 36",
"просп. Соборний, 77",
"вул. Дніпровське шосе, 15",

"вул. Незалежної України, 50",
"вул. Товариська, 5",
"вул. Перша Ливарна, 18",
"вул. Чумаченка, 30",
"просп. Металургів, 25",
"вул. Космічна, 10",
"вул. Лермонтова, 55",
"вул. Тюленіна, 7",
"вул. Запорізька, 40",
"вул. Патріотична, 12",
"вул. Хортицьке шосе, 22",
"вул. Українська, 9",
"вул. Бородінська, 17",
"вул. Героїв України, 33",
"вул. Воскресенська, 4",
"вул. Радянська, 21",
"вул. Каховська, 6"],
"вінниця": ["вул. Соборна, 15",
"вул. Келецька, 78",
"вул. Хмельницьке шосе, 12",
"вул. Замостянська, 45",
"вул. Театральна, 10",
"вул. Пирогова, 88",
"вул. Князів Коріатовичів, 32",
"вул. Ватутіна, 18",
"вул. Магістратська, 6",
"вул. Космонавтів, 50",
"вул. Пушкіна, 22",
"вул. Привокзальна, 7",
"вул. Київська, 14",
"вул. Литвиненка, 5",
"вул. Воїнів-Інтернаціоналістів, 24",
"вул. Максимовича, 30",
"вул. Порика, 3",
"вул. Брацлавська, 40",
"вул. Чорновола, 27",
"вул. Коцюбинського, 15"],
"миколаїв": ["вул. Адміральська, 10",
"просп. Центральний, 56",
"вул. Лазурна, 22",
"вул. Генерала Карпенка, 30",
"вул. Потьомкінська, 44",
"вул. Космонавтів, 11",
"вул. Нікольська, 15",
"вул. Чкалова, 8",
"вул. Героїв України, 50",
"вул. Погранична, 12",
"вул. Велика Морська, 33",
"вул. Центральна, 75",
"вул. Озерна, 27",
"просп. Богоявленський, 19",
"вул. 3-я Слобідська, 17",
"вул. В. Лягіна, 23",
"вул. Океанівська, 4",
"вул. Фалеєвська, 20",
"вул. Будівельна, 38",
"вул. Троїцька, 44"],
"чернігів": ["вул. Шевченка, 23",
"вул. Рокоссовського, 12",
"вул. Мазепи, 45",
"вул. Попудренка, 34",
"вул. Козацька, 56",
"вул. Магістратська, 11",
"вул. Горького, 18",
"вул. Пушкіна, 9",
"вул. Мстиславська, 21",
"вул. Захисників України, 10",
"вул. Ремісника, 7",
"вул. Любецька, 53",
"вул. Гонча, 26",
"вул. Приміська, 12",

"вул. Льотна, 30",
"вул. Шевченка, 90",
"вул. Першотравнева, 40",
"вул. Івана Мазепи, 15",
"вул. Лермонтова, 8",
"вул. Київська, 60"],
"черкаси": ["вул. Смілянська, 43",
"бул. Шевченка, 145",
"вул. Хіміків, 20",
"вул. Надпільна, 67",
"вул. В. Чорновола, 9",
"вул. Гетьмана Сагайдачного, 12",
"вул. Благовісна, 88",
"вул. Пацаєва, 30",
"вул. Бидгощська, 15",
"вул. 30-річчя Перемоги, 45",
"вул. Гоголя, 11",
"вул. Святотроїцька, 33",
"вул. Волкова, 22",
"вул. Припортова, 5",
"вул. 1 Травня, 18",
"вул. Сержанта Жужоми, 10",
"вул. Гагаріна, 25",
"вул. Можайського, 6",
"вул. Героїв Дніпра, 40",
"вул. Кривалівська, 7"],
"суми": ["вул. Харківська, 22",
"вул. Козацький Вал, 3",
"вул. Петропавлівська, 18",
"вул. Іллінська, 50",
"вул. Прокоф'єва, 25",
"вул. Кондратьєва, 11",
"вул. Садова, 40",
"вул. Металургів, 30",
"вул. Леваневського, 15",
"вул. Олександра Сергієнка, 22",
"вул. Роменська, 5",
"вул. Першотравнева, 42",
"вул. Горького, 9",
"вул. Курська, 17",
"вул. Квітки-Основ’яненка, 10",
"вул. Соборна, 55",
"вул. Заливна, 8",
"вул. Фрунзе, 27",
"вul. Ярослава Мудрого, 60",
"вул. Пролетарська, 20"],
"полтава": ["вул. Європейська, 12",
"вул. Шевченка, 77",
"вул. Великотирнівська, 35",
"вул. Соборності, 45",
"вул. Монастирська, 18",
"вул. Пушкіна, 22",
"вул. Зіньківська, 50",
"вул. Ковпака, 10",
"вул. Ватутіна, 27",
"вул. Маршала Бірюзова, 13",
"вул. Сінна, 9",
"вул. Квітки-Основ’яненка, 5",
"вул. Олеся Гончара, 40",
"вул. Олександра Кошиця, 17",
"вул. Вагнера, 12",
"вул. 23 Вересня, 6",
"вул. Європейська, 55",
"вул. Новий Базар, 7",
"вул. Грушевського, 20",
"вул. Гоголя, 14"],
"івано-франківськ": ["вул. Незалежності, 24",
"вул. Галицька, 125",
"вул. Василя Стуса, 18",
"вул. Коновальця, 45",
"вул. Мазепи, 32",

"вул. Сахарова, 10",
"вул. Мельника, 27",
"вул. Грушевського, 50",
"вул. Тичини, 12",
"вул. Симоненка, 7",
"вул. Пасічна, 5",
"вул. Вовчинецька, 20",
"вул. Хоткевича, 15",
"вул. Незалежності, 99",
"вул. Карпатська, 11",
"вул. Івасюка, 22",
"вул. Лепкого, 30",
"вул. Огієнка, 14",
"вул. Пулюя, 38",
"вул. Вагилевича, 6"],
"ужгород": ["вул. Корзо, 13",
"вул. Минайська, 23",
"вул. Капушанська, 45",
"вул. Залізнична, 10",
"вул. Легоцького, 7",
"вул. Гагаріна, 32",
"вул. Волошина, 20",
"вул. Довженка, 18",
"вул. Собранецька, 55",
"вул. Володимирська, 40",
"вул. Льва Толстого, 9",
"вул. Яна Комендаря, 15",
"вул. Підгірна, 6",
"вул. Кошута, 3",
"вул. Ракоці, 22",
"вул. Швабська, 27",
"вул. Острівна, 11",
"вул. Баб'яка, 14",
"вул. Східна, 5",
"вул. Гагаріна, 50"],
"тернопіль": ["вул. Руська, 18",
"вул. Живова, 7",
"вул. Чорновола, 33",
"вул. Київська, 10",
"вул. Степана Бандери, 44",
"вул. Львівська, 55",
"вул. Бродівська, 22",
"вул. Микулинецька, 15",
"вул. 15 Квітня, 7",
"вул. Золотогірська, 20",
"вул. Крушельницької, 3",
"вул. Сахарова, 12",
"вул. Миру, 6",
"вул. Винниченка, 25",
"вул. Острозького, 18",
"вул. Валовий, 9",
"вул. Пушкіна, 40",
"вул. Мазепи, 29",
"вул. Богдана Хмельницького, 13",
"вул. Гоголя, 21"],
"рівне": ["вул. Соборна, 110",
"вул. Київська, 36",
"вул. Гагаріна, 12",
"вул. Вербова, 24",
"вул. Струтинська, 33",
"вул. Чорновола, 44",
"вул. Князя Острозького, 50",
"вул. Мазепи, 7",
"вул. Соборна, 80",
"вул. Симона Петлюри, 15",
"вул. Поштова, 9",
"вул. Князя Володимира, 20",
"вул. 16 Липня, 13",
"вул. Огієнка, 37",
"вул. Відінська, 6",
"вул. Київська, 55",

"вул. Басівкутська, 18",
"вул. Волинська, 29",
"вул. Черняка, 11",
"вул. Шухевича, 22"],
"луцьк": ["просп. Волі, 15",
"вул. Конякіна, 18",
"вул. Винниченка, 20",
"вул. Рівненська, 34",
"вул. Данила Галицького, 5",
"вул. Богдана Хмельницького, 22",
"вул. Кравчука, 10",
"вул. Карбишева, 7",
"вул. Шопена, 12",
"вул. Глушець, 25",
"вул. Гнідавська, 40",
"вул. Сагайдачного, 8",
"вул. Лесі Українки, 3",
"вул. Огієнка, 50",
"просп. Президента Грушевського, 9",
"вул. Липинського, 17",
"вул. Ковельська, 45",
"вул. Теремнівська, 14",
"вул. Залізнична, 6",
"вул. Злуки, 28"],
"хмельницький": ["вул. Кам’янецька, 40",
"вул. Зарічанська, 3",
"вул. Проскурівська, 55",
"вул. Героїв Майдану, 22",
"вул. Панаса Мирного, 11",
"вул. Свободи, 8",
"вул. Толстого, 30",
"вул. Подільська, 14",
"вул. Курчатова, 19",
"вул. Інститутська, 25",
"вул. Гагаріна, 40",
"вул. Огієнка, 33",
"вул. Старокостянтинівське шосе, 17",
"вул. Володимирська, 10",
"вул. Пушкіна, 12",
"вул. Вишнева, 45",
"вул. Тернопільська, 5",
"вул. Водопровідна, 29",
"вул. Миру, 21",
"вул. Приміська, 7"],
"чернівці": ["вул. Головна, 123",
"вул. Руська, 56",
"вул. Героїв Майдану, 45",
"вул. Південно-Кільцева, 30",
"вул. Небесної Сотні, 22",
"вул. Воробкевича, 12",
"вул. Ольги Кобилянської, 99",
"вул. Кобилянської, 101",
"вул. Ентузіастів, 7",
"вул. Шевченка, 40",
"вул. Руська, 75",
"вул. Лесі Українки, 20",
"вул. Університетська, 15",
"вул. Нахімова, 27",
"вул. Хотинська, 33",
"вул. Заньковецької, 18",
"вул. Чкалова, 8",
"вул. Червоноармійська, 3",
"вул. Грушевського, 21",
"вул. Головна, 150"],
"кропивницький": ["вул. Велика Перспективна, 89",
"вул. Полтавська, 33",
"вул. Героїв України, 10",
"вул. Короленка, 55",
"вул. Волкова, 27",
"вул. Тімірязєва, 18",
"вул. Київська, 40",

```
    "вул. Преображенська, 12",
    "вул. Вокзальна, 22",
    "вул. Маланюка, 5",
    "вул. Кропивницького, 15",
    "вул. Декабристів, 30",
    "вул. Карла Маркса, 8",
    "вул. Олександрійська, 44",
    "вул. Лейтенанта Покладова, 3",
    "вул. Садова, 20",
    "вул. Грушевського, 16",
    "вул. Ватутіна, 7",
    "вул. Шевченка, 25",
    "вул. Пашутінська, 9"],
    "житомир": ["вул. Київська, 25",
    "вул. Перемоги, 10",
    "вул. Лесі Українки, 45",
    "вул. Велика Бердичівська, 33",
    "вул. Покровська, 18",
    "вул. Коцюбинського, 7",
    "вул. Вокзальна, 12",
    "вул. Черняховського, 20",
    "вул. Вітрука, 15",
    "вул. Кибальчича, 9",
    "вул. Ватутіна, 28",
    "вул. Домбровського, 6",
    "вул. Гагаріна, 37",
    "вул. Мала Бердичівська, 10",
    "вул. Святослава Ріхтера, 44",
    "вул. Котовського, 13",
    "вул. Вітрука, 50",
    "вул. Новопречистенська, 8",
    "вул. Мазепи, 22",
    "вул. Бородія, 11"]
}

# === Команда /start ===
@bot.message_handler(commands=['start'])
def start(message):
    bot.send_message(
        message.chat.id,
        "👋 Вітаю у єдиному офіційному боті Нової Пошти у Telegram!",
        reply_markup=greet_kb
    )

@bot.message_handler(commands=['book_time'])
def book_time(message):
    msg = bot.send_message(message.chat.id, "📅 Введіть бажану дату відправлення (у форматі ДД.ММ.РРРР):")
    bot.register_next_step_handler(msg, ask_time)

def ask_time(message):
    date = message.text.strip()
    if not validate_date(date):
        msg = bot.send_message(message.chat.id, "❗ Невірний формат дати. Спробуйте ще раз (ДД.ММ.РРРР):")
        bot.register_next_step_handler(msg, ask_time)
        return
    message.chat.booking_date = date
    msg = bot.send_message(message.chat.id, "🕒 Введіть бажаний час (у форматі ГГ:ХХ):")
    bot.register_next_step_handler(msg, confirm_booking, date)

def confirm_booking(message, date):
    time = message.text.strip()
    if not validate_time(time):
        msg = bot.send_message(message.chat.id, "❗ Невірний формат часу. Спробуйте ще раз (ГГ:ХХ):")
        bot.register_next_step_handler(msg, confirm_booking, date)
        return
    bot.send_message(
        message.chat.id,
        f"✅ Ваше бронювання підтверджено!\n Дата: {date}\n🕒 Час: {time}\nДякуємо, що обрали Нову Пошту!"
    )
```

```
@bot.message_handler(func=lambda message: message.text == "⌨️ Забронювати час")
def book_time_button(message):
    book_time(message)

# === Команда /help ===
@bot.message_handler(commands=['help'])
def help_command(message):
    bot.send_message(
        message.chat.id,
        "🛠 Доступні команди:\n"
        "/start — запуск бота\n"
        "/help — допомога\n"
        "/news — останні новини\n"
        "/track — відстежити посилку\n"
        "/feedback — залишити відгук\n"
        "/contact — контактна інформація\n"
        "/site — сайт Нової Пошти\n\n"
        "/find_by_index — Пошук відділення за індексом чи адресою\n" # Corrected command name
        "Або просто скористайтесь кнопками нижче.",
        reply_markup=main_kb
    )

# === Команда /commands ===
@bot.message_handler(commands=['commands'])
def commands(message):
    bot.send_message(
        message.chat.id,
        "📋 Доступні команди бота:\n"
        "/start — запуск бота\n"
        "/help — допомога\n"
        "/news — останні новини\n"
        "/track — відстежити посилку\n"
        "/feedback — залишити відгук\n"
        "/contact — контактна інформація\n"
        "/site — сайт Нової Пошти\n"
        "/find_by_index — Пошук відділення за індексом чи адресою\n" # Corrected command name
    )

# === Вітання ===
@bot.message_handler(func=lambda message: message.text == "Привіт! 👋")
def reply_hello(message):
    bot.send_message(
        message.chat.id,
        "Як я можу допомогти? Оберіть опцію нижче ⬇️",
        reply_markup=main_kb
    )

# === Знайти відділення ===
@bot.message_handler(func=lambda message: message.text == "Знайти відділення 📮")
def ask_city(message):
    msg = bot.send_message(message.chat.id, "🏙 Введіть назву міста українською:")
    bot.register_next_step_handler(msg, find_branch)

def find_branch(message):
    city = message.text.strip().lower()
    if city in branches:
        reply = f"📍 Відділення у місті {city.capitalize()}:\n" + "\n".join(
            [f"{i+1}. {addr}" for i, addr in enumerate(branches[city])])
    else:
        reply = "🚫 На жаль, немає інформації про відділення в цьому місті. Спробуйте інше місто."
    bot.send_message(message.chat.id, reply)

# === Графік роботи ===
@bot.message_handler(func=lambda message: message.text == "Графік роботи 🕒")
def schedule(message):
    bot.send_message(
        message.chat.id,
        "🕒 Звичайний графік роботи:\nПн-Пт: 08:00 - 20:00\nСб: 09:00 - 18:00\nНд: 10:00 - 16:00"
    )
```

```
# === Розташування ===
@bot.message_handler(func=lambda message: message.text == "Розташування 📍")
def location(message):
    bot.send_message(
        message.chat.id,
        "📌 Приклад розташування відділення у Києві:"
    )
    bot.send_location(message.chat.id, latitude=50.4501, longitude=30.5234)
    bot.send_message(message.chat.id, "🔗 Додатково:", reply_markup=inline_kb)

# === Послуги ===
@bot.message_handler(func=lambda message: message.text == "Доступні послуги 🛍️")
def services(message):
    services_kb = types.ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)
    services_kb.add(
        "📦 Відправлення та отримання", "💰 Післяплата",
        "🚚 Кур'єрська доставка", "🛍️ Зберігання посилок",
        "🔁 Переадресація", "⬅️ Назад"
    )
    bot.send_message(
        message.chat.id,
        "🛍️ Оберіть послугу для детальнішої інформації:",
        reply_markup=services_kb
    )

@bot.message_handler(func=lambda message: message.text in [
    "📦 Відправлення та отримання",
    "💰 Післяплата",
    "🚚 Кур'єрська доставка",
    "🛍️ Зберігання посилок",
    "🔁 Переадресація",
    "⬅️ Назад"
])
def service_detail(message):
    details = {
        "📦 Відправлення та отримання": "📦 Ви можете швидко надіслати або отримати посилку у будь-якому відділенні. Оформлення доступне і через мобільний додаток.",
        "💰 Післяплата": "💰 Отримувач може оплатити замовлення при видачі — готівкою або карткою.",
        "🚚 Кур'єрська доставка": "🚚 Замовте кур'єра для доставки/збору посилки з дому або офісу.",
        "🛍️ Зберігання посилок": "🛍️ Безкоштовне зберігання — 5 днів. Потім — за тарифами.",
        "🔁 Переадресація": "🔁 Змініть адресу доставки через додаток або за допомогою оператора.",
    }

    if message.text == "⬅️ Назад":
        bot.send_message(
            message.chat.id,
            "🔙 Повертаємось до головного меню:",
            reply_markup=main_kb
        )
    else:
        bot.send_message(message.chat.id, details[message.text])

# === Команда /news ===
@bot.message_handler(commands=['news'])
def news(message):
    bot.send_message(
        message.chat.id,
        "📰 Останні новини Нової Пошти:\n"
        "1. Нове відділення у Бучі!\n"
        "2. Кур'єри тепер працюють до 22:00.\n"
        "3. Знижки до -20% на доставку до кінця місяця."
    )

@bot.message_handler(commands=['find_by_index'])
def find_by_index(message):
    msg = bot.send_message(message.chat.id, "🔎 Введіть поштовий індекс або частину адреси:")
```

```
    bot.register_next_step_handler(msg, search_by_index)

def search_by_index(message):
    query = message.text.strip().lower()
    results = []
    for city, addresses in branches.items():
        for addr in addresses:
            if query in addr.lower():
                results.append(f"{city.capitalize()}: {addr}")
    if results:
        bot.send_message(message.chat.id, "🔎 Знайдено:\n" + "\n".join(results))
    else:
        bot.send_message(message.chat.id, "⛔ Нічого не знайдено за запитом.")

# === Команда /track ===
@bot.message_handler(commands=['track'])
def track(message):
    msg = bot.send_message(message.chat.id, "📦 Введіть номер ТТН для відстеження:")
    bot.register_next_step_handler(msg, process_tracking)

def process_tracking(message):
    ttn = message.text.strip()
    # Prepare the payload for the Nova Poshta API
    payload = {
        "apiKey": NOVA_POSHTA_API_KEY,
        "modelName": "TrackingDocument",
        "calledMethod": "getStatusDocuments",
        "methodProperties": {
            "Documents": [
                {
                    "DocumentNumber": ttn,
                    "Phone": "" # You might allow users to input their phone number for more specific tracking if needed
                }
            ]
        }
    }
    headers = {'Content-Type': 'application/json'}
    nova_poshta_api_url = "https://api.novaposhta.ua/v2.0/json/"

    try:
        response = requests.post(nova_poshta_api_url, json=payload, headers=headers)
        response.raise_for_status()  # Raise an exception for HTTP errors (4xx or 5xx)
        data = response.json()

        if data and data['success'] and data['data']:
            status_data = data['data'][0]
            status = status_data.get('Status', 'Статус невідомий')
            warehouse_recipient = status_data.get('WarehouseRecipient', 'Невідомий склад')
            scheduled_delivery_date = status_data.get('ScheduledDeliveryDate', 'Невідома дата')

            bot.send_message(
                message.chat.id,
                f"🔍 Статус посилки **{ttn}**:\n"
                f"**Статус**: {status}\n"
                f"**Очікується прибуття до**: {warehouse_recipient}\n"
                f"**Планована дата доставки**: {scheduled_delivery_date}"
            )
        else:
            bot.send_message(message.chat.id, f"⛔ Не вдалося отримати інформацію для ТТН: {ttn}. Перевірте правильність номера.")
    except requests.exceptions.RequestException as e:
        bot.send_message(message.chat.id, f"❌ Виникла помилка при зверненні до сервісу відстеження: {e}")
    except KeyError:
        bot.send_message(message.chat.id, "⛔ Не вдалося обробити відповідь від Нової Пошти. Спробуйте ще раз.")

# === Команда /feedback ===
@bot.message_handler(commands=['feedback'])
def feedback(message):
    msg = bot.send_message(message.chat.id, "✍ Напишіть ваш відгук або побажання:")
    bot.register_next_step_handler(msg, save_feedback)
```

```
def save_feedback(message):
    feedback_text = message.text.strip()
    bot.send_message(message.chat.id, "✅ Дякуємо за ваш відгук!")

def validate_date(date_str):
    try:
        datetime.strptime(date_str, "%d.%m.%Y")
        return True
    except ValueError:
        return False

def validate_time(time_str):
    try:
        datetime.strptime(time_str, "%H:%M")
        return True
    except ValueError:
        return False

# === Команда /contact ===
@bot.message_handler(commands=['contact'])
def contact(message):
    bot.send_message(
        message.chat.id,
        "📞 Служба підтримки:\n"
        "Телефон: 0 800 500 609\n"
        "Email: info@novaposhta.ua"
    )

# === Команда /site ===
@bot.message_handler(commands=['site'])
def site(message):
    bot.send_message(
        message.chat.id,
        "🌐 Вебсайт Нової Пошти:",
        reply_markup=inline_kb
    )

# === Запуск бота ===
bot.polling()
```

лістинг В.1 – код main.py

Розгортання коду бота на Github:

https://github.com/IvanLisun/Novaposhta_bot

Розгортання бота на Heroku-сервер:

https://novaposhtabot1-87542b05dd95.herokuapp.com/