ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**Чат-бот Telegram на мові програмування Python для отримання погоди**»

на здобуття освітнього ступеня вищої освіти бакалавр
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ярослав Крицький
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Ярослав КРИЦЬКИЙ
група КНД–41

Керівник: Віталій КОТЕЛЯНЕЦЬ
науковий ступінь, вчене звання к.т.н.

Рецензент:
науковий ступінь, вчене звання _____
(Ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Комп'ютерних наук

Ступінь вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ

«_____» _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Крицькому Ярославу Юрійовичу

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Чат-бот Telegram на мові програмування Python для отримання погоди

керівник кваліфікаційної роботи Віталій КОТЕЛЯНЕЦЬ к.т.н.,

(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. № 56

2. Строк подання кваліфікаційної роботи «30» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи:

- Огляд існуючих інструментів для створення чат-бот Telegram на мові програмування Python для отримання погоди.
- Науково-технічна література щодо створення чат-бот Telegram на мові програмування Python для отримання погоди.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз існуючих інструментів для отримання погодних даних у чат-боті Telegram.
2. Дослідження застосування Python у розробці чат-ботів Telegram для отримання погодних даних.
3. Технічна реалізацію функціоналу отримання погоди в чат-боті Telegram

5. Перелік графічного матеріалу: *презентація*

1) Тема дипломної роботи

2) Мета роботи. Об’єкт та предмет дослідження. Постановка завдання дослідження.

3) Актуальність роботи.

4) Огляд існуючих інструментів та методів створення чат-бот Telegram на мові програмування Python для отримання погоди.

5) Огляд основних можливостей Python та API OpenWeatherMap.

6) Демонстрація роботи розроблених інструментів.

7) Висновки дослідження

8) Апробація результатів дослідження

Дата видачі завдання 25.02.2025 р

КАЛЕНДАРНИЙ ПЛАН

№ п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02-26.02.25	виконане
2	Дослідження поставленої задачі	26.02-20.03.25	виконане
3	Аналіз існуючих рішень у сфері погодних сервісів та чат-ботів у Telegram	23.02- 25.02.25	виконане
4	Вивчення інструментів Python для створення Telegram-ботів та взаємодії з API	25.02- 20.03.25	виконане
5	Розробка програмного забезпечення Telegram-бота для отримання погодних даних через OpenWeatherMap	21.03-27.03.25	виконане
6	Аналіз можливостей розширення функціоналу бота (прогноз на кілька днів, обробка помилок)	28.03-30.03.25	виконане
7	Реалізація додаткових функцій: прогноз погоди на 3 дні, перевірка коректності введених даних	01.04-07.04.25	виконане
8	Розробка рекомендацій щодо подальшого розвитку Telegram-бота та масштабування проєкту	08.04-17.04.25	виконане
9	Вступ, висновки, реферат	18.04-24.04.25	виконане
10	Розробка обов’язкових демонстраційних матеріалів	25.04-28.04.25	виконане
11	Доповідь та презентація	29.04-20.05.25	виконане
12	Попередній захист роботи	До 30.05	виконане

Здобувач вищої освіти _____ Ярослав КРИЦЬКИЙ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Віталій КОТЕЛЯНЕЦЬ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра 53 с., 18 табл.,11 рис., 40 джерел.

Мета роботи – Підвищення ефективності отримання користувачами актуальної інформації про погодні умови та прогнози на найближчі дні в інтерактивному форматі.

Об'єкт дослідження – Процес забезпечення взаємодії користувача з чат-ботом Telegram для отримання інформації про погоду.

Предмет дослідження – Інструменти програмної реалізації чат-бота Telegram, який забезпечує доступ до погодних даних в інтерактивному форматі, використовуючи мову програмування Python та API OpenWeatherMap.

Короткий зміст роботи:

Наукове завдання – Оцінка ефективності розроблених інструментів для програмної реалізації чат-бота Telegram, який забезпечує доступ до погодних даних в інтерактивному форматі, використовуючи мову програмування Python та API OpenWeatherMap.

Методи дослідження – Теоретичні методи (метод порівняльного аналізу, метод системного аналізу); аналітичні методи (метод метрики ефективності, метод логування) та практичні методи (метод моделювання, метод проєктування програмного забезпечення, експериментальний метод (практична реалізація), метод тестування та налагодження).

Актуальність теми – У сучасному світі, де інформаційні технології розвиваються з неймовірною швидкістю, месенджери стали важливою складовою повсякденного життя мільйонів людей по всьому світу. Одним із найбільш популярних засобів комунікації є Telegram — універсальний месенджер, який, крім традиційного обміну повідомленнями, пропонує розробникам можливість створювати чат-ботів. Завдяки мові програмування Python створення функціонального чат-бота для отримання погоди є актуальним завданням.

Галузь використання – Повсякденне життя користувачів, туризм і готельно-ресторанна сфера, сільське господарство, транспорт і логістика, міські сервіси та розумні міста (smart city).

КЛЮЧОВІ СЛОВА: чат-бот погоди, Telegram Bot API, Python, програмування, користувацький інтерфейс.

ABSTRACT

Text part of the bachelor's thesis 53 p.,18 таb., 11 fig., 40 sources.

The purpose of the work – Increasing the efficiency of users receiving up-to-date information about weather conditions and forecasts for the coming days in an interactive format.

Object of research – The process of ensuring user interaction with a Telegram chatbot to receive weather information.

Subject of research – Tools for the software implementation of the Telegram chatbot, which provides access to weather data in an interactive format using the Python programming language and the OpenWeatherMap API.

Summary of the work:

Scientific task – Evaluating the effectiveness of the developed tools for the software implementation of the Telegram chatbot, which provides access to weather data in an interactive format using the Python programming language and the OpenWeatherMap API.

Research methods – Theoretical methods (comparative analysis method, system analysis method); analytical methods (performance metrics method, logging method) and practical methods (modeling method, software design method, experimental method (practical implementation), testing and debugging method).

Relevance of the topic – In the modern world, digital technologies and mobile services play an important role in people's daily lives. One of the most popular communication tools is messengers, including Telegram, which supports the creation of chatbots - automated assistants that can interact with users in a convenient format. One of the daily needs of users is to receive timely and reliable information about weather conditions. The use of a chatbot for this purpose allows for quick access to weather data at any time and from any device. Thanks to the Python programming language, which is simple, flexible, and has a wide range of libraries for working with APIs and Telegram, creating a functional chatbot for weather is a relevant task.

Field of application – Daily life of users, tourism and hospitality, agriculture, transportation and logistics, urban services and smart cities.

KEYWORDS: weather chatbot, Telegram Bot API, Python, programming, user interface.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	-	Application Programming Interface
HTTP	-	HyperText Transfer Protocol
JSON	-	JavaScript Object Notation
XML	-	Extensible Markup Language
IDE	-	Integrated Development Environment
HTTP	-	HyperText Transfer Protocol
GET	-	HTTP- метод запиту для отримання даних із сервера
Telegram API	-	Інтерфейс для розробки програмного забезпечення для Telegram
PyTelegramBotAPI	-	Python-бібліотека для створення Telegram- ботів
OpenWeatherMap API	-	API для отримання погодних даних
Token	-	Унікальний ідентифікатор доступу до API
Request	-	HTTP-запит до зовнішнього сервісу
Response	-	Відповідь сервера на запит клієнта

ВСТУП

В атестаційній роботі бакалавра розглядається актуальна проблема створення чат-бот Telegram на мові програмування Python для отримання погоди різних міст за допомоги API OpenWeatherMap.

Актуальність теми. У сучасному світі цифрові технології та мобільні сервіси відіграють важливу роль у щоденному житті людей. Одним із популярних інструментів комунікації є месенджери, зокрема Telegram, який підтримує створення чат-ботів — автоматизованих помічників, здатних взаємодіяти з користувачем у зручному форматі. Серед щоденних потреб користувачів — отримання оперативної та достовірної інформації про погодні умови. Використання чат-бота для цієї мети дозволяє забезпечити швидкий доступ до метеоданих у будь-який час і з будь-якого пристрою. Завдяки мові програмування Python, яка є простою, гнучкою та має широкий спектр бібліотек для роботи з API та Telegram, створення функціонального чат-бота для отримання погоди є актуальним завданням. Це поєднує актуальні тенденції в IT-галузі, такі як автоматизація сервісів, інтеграція з зовнішніми API та розвиток штучного інтелекту. Таким чином, розробка чат-бота Telegram на Python для отримання погодної інформації є актуальною як з практичної точки зору, так і з погляду подальшого розвитку цифрових комунікацій.

Ступінь вивченої проблеми. Проблематика розробки чат-ботів є предметом широкого дослідження в технічній літературі та онлайн-ресурсах[1, 8, 9, 25, 31]. Існує безліч прикладів ботів, які варіюються за рівнем складності — від простих відповідей на запити до комплексних віртуальних асистентів. Приклади інтеграції з погодними сервісами, зокрема з OpenWeatherMap API, детально опрацьовані, але їх використання, особливо для прогнозів на кілька днів та локалізації під конкретного користувача, є менш поширеним. Тому ця тема має значний потенціал для подальшого розвитку та вдосконалення.

Постановка завдання дослідження: Створення чат бота Telegram на мові програмування Python для отримання погоди полягає в забезпеченні користувачів доступною, оперативною та точною інформацією про погодні умови в інтерактивному форматі. Такої чат-бот дозволяє користувачам швидко дізнаватися прогноз погоди у будь-якому місті світу, просто ввівши команду в чаті, без необхідності відкривати сайти чи мобільні додатки. Тому виникає задача створення програмного застосунку у вигляді чат бота Telegram на мові програмування Python для отримання погодних даних в інтерактивному форматі.

Специфіка джерельної бази. Дослідження джерельної бази відзначає, що джерела інформації неоднорідні та її треба перебрати за певними критеріями щодо їх різновиду, а саме: до важливості, якості, унікальності та ін., щоб відмітити проблеми та недоліки. У ході дослідження були залучені такі джерела: офіційна документація Telegram Bot API; документація OpenWeatherMap API; література з програмування на Python; аналітичні та оглядові статті, присвячені розробці чат-ботів;

Мета і завдання дослідження: Підвищення ефективності отримання користувачами актуальної інформації про погодні умови та прогнози на найближчі дні в інтерактивному форматі.

Завдання роботи:

1. Виконати аналіз існуючих інструментів для отримання погодних даних у чат-боті Telegram.
2. Провести дослідження застосування Python у розробці чат-ботів Telegram для отримання погодних даних.
3. Зробити технічну реалізацію функціоналу отримання погоди в чат-боті Telegram

Об'єкт дослідження: Процес забезпечення взаємодії користувача з чат-ботом Telegram для отримання інформації про погоду.

Предмет дослідження: Інструменти програмної реалізації чат-бота Telegram, який забезпечує доступ до погодних даних в інтерактивному форматі, використовуючи мову програмування Python та API OpenWeatherMap.

Методи дослідження. У процесі дослідження було застосовано ряд методів: теоретичні методи (метод порівняльного аналізу, метод системного аналізу); аналітичні методи (метод метрики ефективності, метод логування) та практичні методи (метод моделювання, метод проєктування програмного забезпечення, експериментальний метод (практична реалізація), метод тестування та налагодження).

Наукове завдання – Оцінка ефективності розроблених інструментів для програмної реалізації чат-бота Telegram, який забезпечує доступ до погодних даних в інтерактивному форматі, використовуючи мову програмування Python та API OpenWeatherMap.

Результати дослідження. Внаслідок проведеної роботи було розроблено чат-бот Telegram, який: відповідає на запити користувачів про погодні умови в будь-якому місті світу; надає інформацію про температуру, відчутну температуру, швидкість та пориви вітру, а також загальні погодні умови; додатково пропонує прогноз погоди на наступні три дні; створений з використанням мови програмування Python та бібліотек telebot, requests, json, datetime; пройшов тестування в різних сценаріях і демонструє стабільну роботу.

Наукова новизна. Наукова новизна цього дослідження полягає в розробці функціоналу, що дозволяє отримувати прогноз погоди на кілька днів, що перевищує стандартні можливості інформування за рахунок поєднанні кількох технологій у зручний та гнучкий програмний продукт чат-бот Telegram. Цей бот не лише отримує актуальну інформацію з API, але й динамічне обробляє прогнози на кілька днів, адаптуючись до потреб користувача, а також включає елементи обробки помилок. Реалізовано функціонал, який можна легко масштабувати та розширювати.

Практична значущість результатів дослідження. Розроблений бот здатний функціонувати як повноцінний сервіс для надання інформації про погодні умови. Він є універсальним інструментом, який може бути використаний як для особистого користування, так і в якості додаткового сервісу для організацій. Отриманий програмний продукт має потенціал для подальшого вдосконалення

шляхом додавання нових функцій, таких як інтеграція з картографічними сервісами, локалізація тощо.

Апробація результатів та публікації. Апробація результатів надається в 1 тезисах в рецензованих наукових журналах і виданнях. Матеріали були опубліковані:

В тезисах:

1. Крицький Я.Ю. ІНТЕГРАЦІЯ ЧАТ-БОТ TELEGRAM З INTERNET OF THINGS // V Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез 15 квітня 2025. – К.: ДУІКТ, https://duikt.edu.ua/ua/news-1-570-12409-v-mizhnarodna-naukovo-tehnichna-konferenciya-suchasniy-stan-ta-perspektivi-rozvitku-iot_kafedra-inzhenerii-programnogo-zabezpechennya-avtomatizovanih-sistem

2. Крицький Я.Ю. Особливості розробки Telegram-бота для отримання погоди з позиції кібербезпеки// Всеукраїнська науково-практична конференція "Цифрова трансформація кібербезпеки" Збірник тез 24 квітня 2025. – К.: ДУІКТ, https://duikt.edu.ua/ua/news-1-574-14029-duikt-zaproshue-do-uchasti-u-vseukrainskiy-naukoviy-konferencii--cifrova-transformaciya-kiberbezpeki_kafedra-informaciynoi-ta-kibernetichnoi-bezpeki

1 АНАЛІЗ ІСНУЮЧИХ ІНСТРУМЕНТІВ ДЛЯ ОТРИМАННЯ ПОГОДНИХ ДАНИХ У ЧАТ-БОТІ TELEGRAM

1.1 Основи створення чат-боту Telegram у Python

У сучасному світі, де інформаційні технології розвиваються з неймовірною швидкістю, месенджери стали важливою складовою повсякденного життя мільйонів людей по всьому світу. Одним із найбільш популярних засобів комунікації є Telegram — універсальний месенджер, який, крім традиційного обміну повідомленнями, пропонує розробникам можливість створювати чат-ботів. Ці програмні агенти автоматизують різні дії, надають інформацію або навіть можуть замінювати цілі веб сайти та мобільні додатки в рамках діалогу.

У сучасному світі цифрові технології та мобільні сервіси відіграють важливу роль у житті людей. Одним із популярних інструментів комунікації є месенджер, зокрема Telegram, який підтримує створення чат-ботів — автоматизованих помічників, здатних взаємодіяти з користувачем у зручному форматі. Серед щоденних потреб користувачів — отримання оперативної та достовірної інформації про погодні умови. Використання чат-бота для цієї мети дозволяє забезпечити швидкий доступ до метеоданих у будь-який час і з будь-якого пристрою. Завдяки мові програмування Python, яка є простою, гнучкою та має широкий спектр бібліотек для роботи з API та Telegram, створення функціонального чат-бота для отримання погоди є актуальним завданням.

На сучасному етапі розвитку цифрових технологій чат-боти займають важливу роль у системах автоматизації обслуговування клієнтів. Telegram-боти, зокрема, здобули велику популярність завдяки простому API, надійній інфраструктурі та відкритості для сторонніх розробників. Однією з основних переваг цієї платформи є можливість створення функціональних додатків без

необхідності у складній серверній архітектурі, що значно спрощує процес розробки ботів навіть для новачків у програмуванні (Рис.1.1).

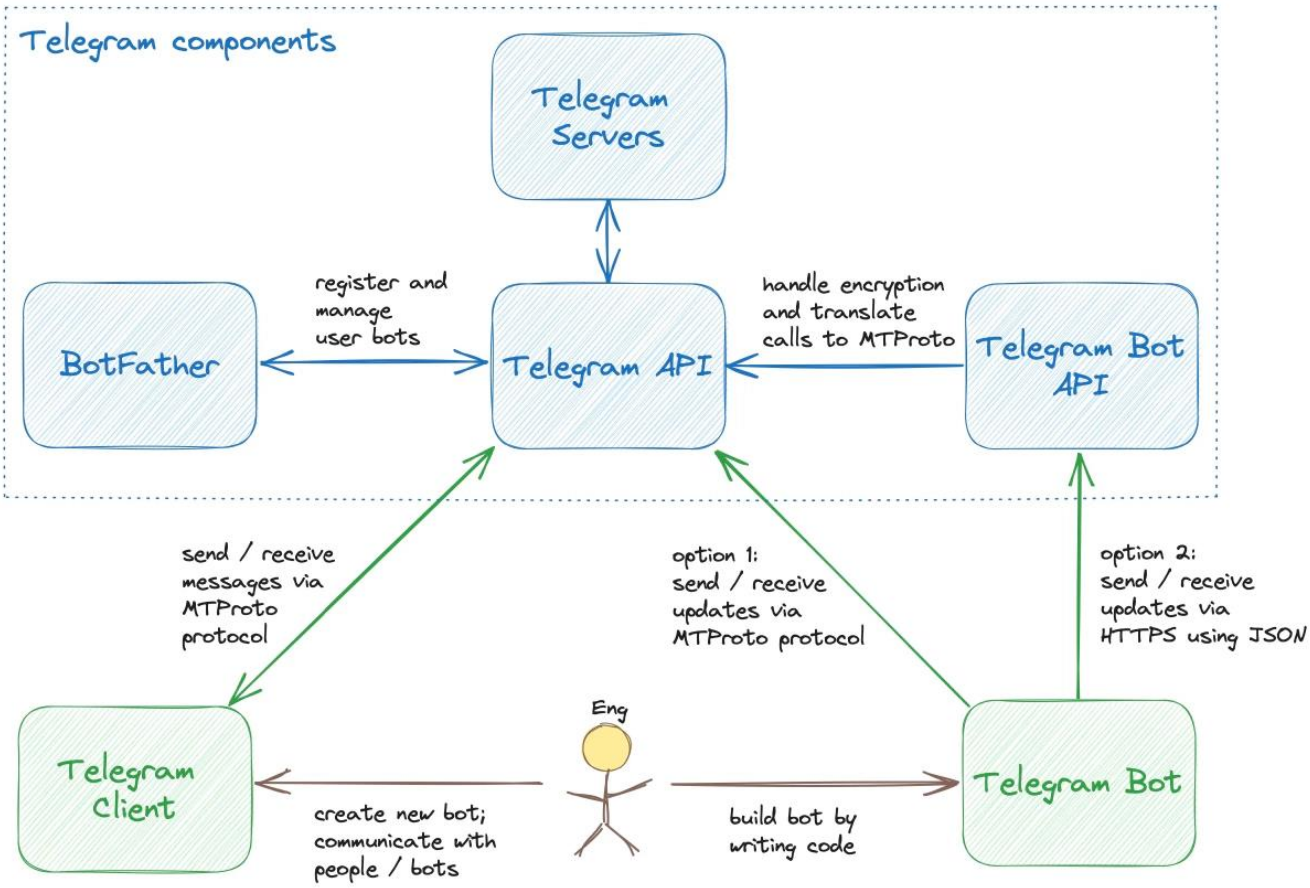


Рисунок 1.1 - Компоненти системи Telegram [31]

Мова програмування Python є відмінним вибором для розробки Telegram-ботів завдяки своєму зрозумілому синтаксису, багатому набору бібліотек та активній спільноті розробників. Зазвичай програмісти обирають такі бібліотеки, як pyTelegramBotAPI, python-telegram-bot або aiogram, кожна з яких має свої особливості, що залежать від складності проєкту та вимог розробника. Наприклад, pyTelegramBotAPI ідеально підходить для швидкого запуску та простих рішень, тоді як aiogram пропонує асинхронність, що є важливим аспектом для масштабованих систем.

Крім вибору мови програмування та бібліотеки, важливо належним чином організувати структуру коду бота. Рекомендується дотримуватися принципів

модульності, розміщуючи окремі функції, такі як обробка команд, запити до зовнішніх API або генерація відповідей, у різних файлах. Такий підхід не лише спрощує підтримку проєкту, але й забезпечує можливість його подальшого масштабування.

Важливим аспектом є також безпека бота. Telegram-боти використовують унікальні токени для ідентифікації, які отримуються через BotFather. Ці токени слід зберігати в безпечному середовищі, наприклад, у файлах .env, а не вносити їх безпосередньо в основний код. Крім того, доцільно обмежити доступ до бота лише для певних користувачів або впровадити систему авторизації, якщо бот обробляє конфіденційну інформацію.

Для забезпечення безперебійної роботи чат-бота важливо також розглянути обробку екстраординарних ситуацій, таких як введення користувачем некоректних даних або недоступність стороннього API. У цих випадках слід передбачити чіткі повідомлення про помилки та механізми для повторних запитів.

Розробка Telegram-бота на Python потребує не лише технічних навичок, але й стратегічного підходу. Це включає в себе вибір відповідних бібліотек, організацію коду, а також налаштування безпеки та стійкості до помилок. Такий всебічний підхід сприяє створенню ефективного, надійного та зручного для користувачів цифрового асистента.

Чат-бот у Telegram є віртуальним співрозмовником, що функціонує в рамках інтерфейсу Telegram. Він має можливість реагувати на команди користувачів, обробляти отримані повідомлення та надавати відповіді у текстовому або мультимедійному форматах. Основою його роботи є Telegram Bot API — спеціалізований інтерфейс програмування, який забезпечує взаємодію з Telegram-сервером через HTTP-запити (Рис. 1.2).

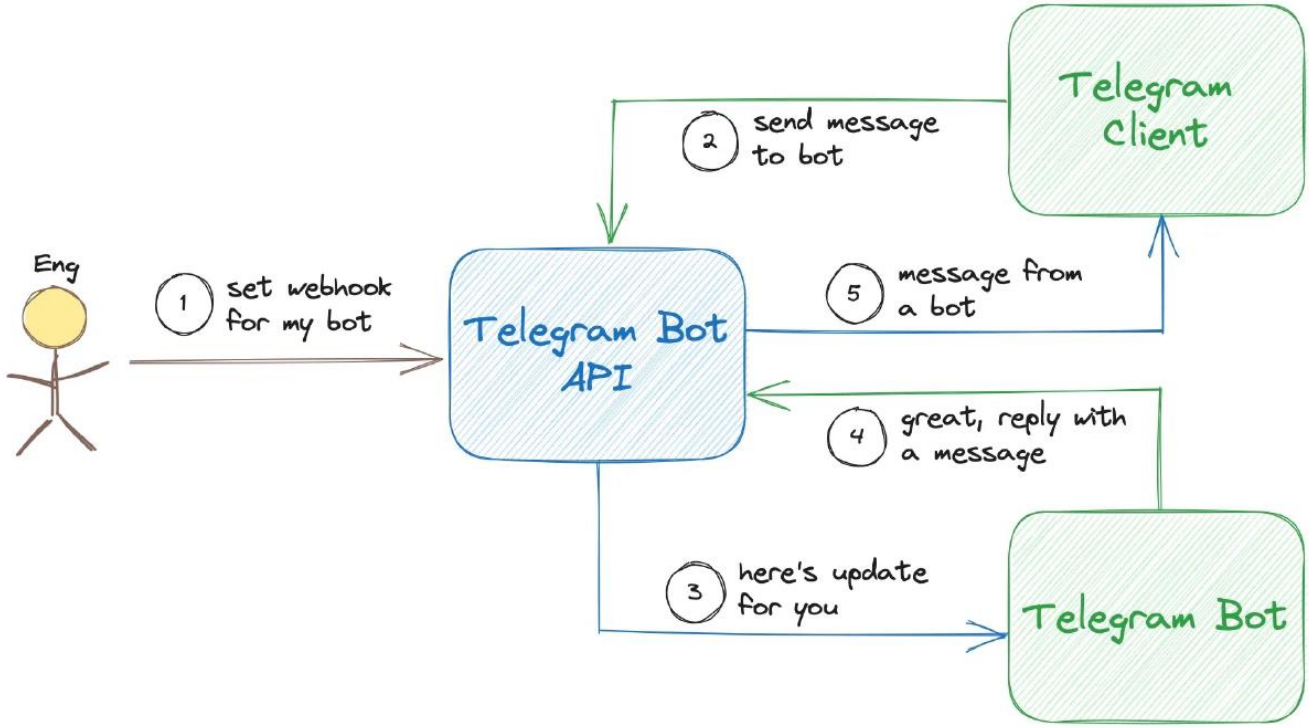


Рисунок 1.2 — Робота Телеграм бота [31]

В таблиці 1.1 надані команди Telegram-бота і їх функції.

Таблиця 1.1— Команди Telegram-бота і їх функції

Команда	**Призначення**
/start	Вітальне повідомлення, запрошення ввести назву міста
Назва міста	Отримання поточної погоди та прогнозу
Некоректне місто	Повідомлення про помилку, запрошення повторити ввід

Для розробки подібного бота існує безліч мов програмування, проте Python став одним з найпопулярніших виборів серед програмістів завдяки своїй простоті, високому рівню абстракції, наявності численних готових бібліотек та активній спільноті. У рамках цієї дипломної роботи Python було обрано як основний

інструмент для реалізації Telegram-бота, що надає погодну інформацію. Це рішення обумовлене зручним синтаксисом мови, який сприяє швидкому створенню функціональних прототипів, а також широким можливостям для роботи з мережевими запитами та обробкою даних у форматі JSON.

Першим етапом у процесі створення Telegram-бота є його реєстрація через спеціального сервісного бота BotFather. Цей бот надає можливість встановити ім'я, опис, профільну фотографію, а також, що є найважливішим, отримати унікальний токен доступу до API. Цей токен забезпечує взаємодію між ботом і Telegram-сервером.

Після отримання токена, розробник розробляє Python-скрипт і імплементує необхідні бібліотеки. У цій реалізації застосовується бібліотека pyTelegramBotAPI, яка слугує обгорткою для Telegram Bot API. Вона забезпечує зручні можливості для обробки повідомлень, надсилання відповідей, управління кнопками, меню, мультимедійними елементами тощо. Крім того, підключаються додаткові бібліотеки (Рис.1.3):

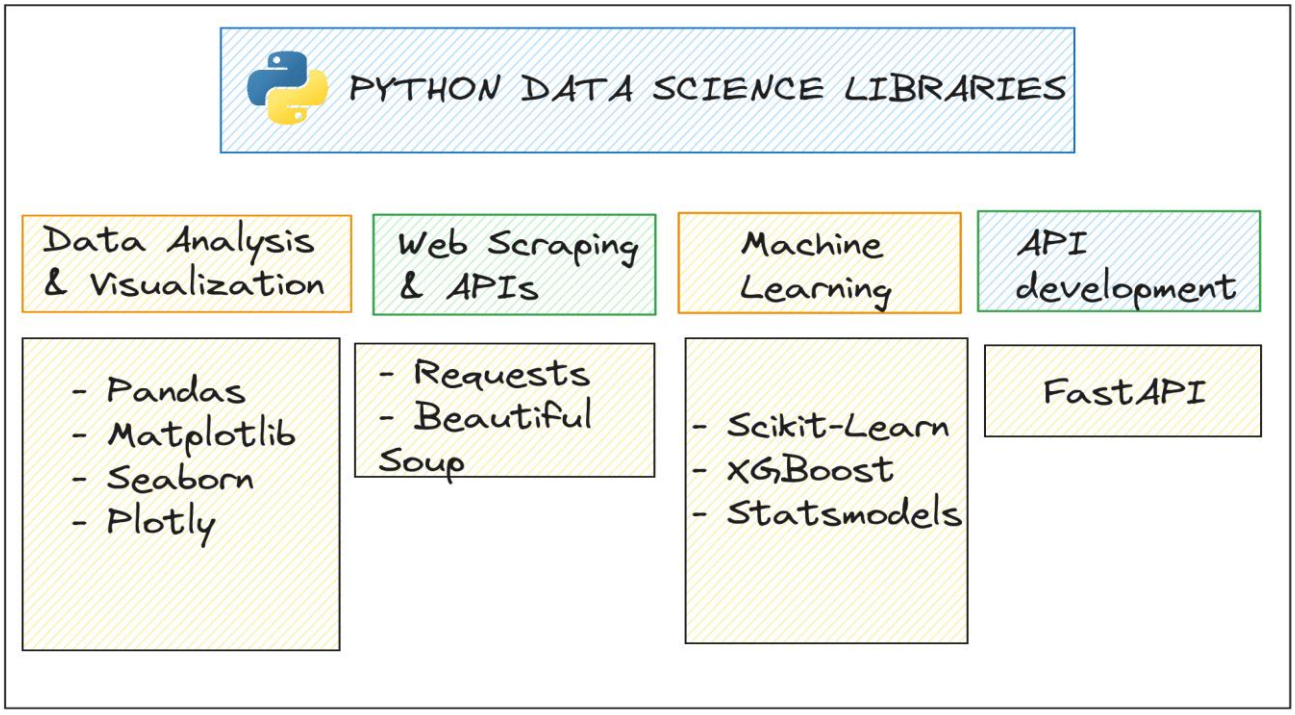


Рисунок 1.3 - Інші бібліотеки Python [32]

В таблиці 1.2 надані основні бібліотеки, використані в Telegram-боті.

Таблиця 1.2 — Основні бібліотеки, використані в Telegram-боті

Назва бібліотеки	**Призначення**
telebot	Робота з Telegram Bot API
requests	Надсилання HTTP-запитів до зовнішніх API
json	Обробка структурованих JSON-відповідей
datetime	Робота з датою і часом, вибір прогнозів на потрібні години

Код бота структурований через обробники повідомлень. Наприклад, використовуючи декоратор @bot.message_handler(commands=['start']), визначається реакція бота на команду /start, що запускає діалог з користувачем. Інший декоратор, @bot.message_handler(content_types=['text']), дає змогу обробляти текстові повідомлення, в яких користувач вводить назву міста для отримання інформації про погоду.

У процесі розробки бота ключовою функцією є отримання назви міста від користувача, надсилання запиту до API сервісу OpenWeatherMap для збору актуальних метеорологічних даних та надання користувачеві стиснутого, але інформативного повідомлення про поточні погодні умови та прогноз на найближчі дні. Уся логіка запиту, обробка відповіді та її форматування реалізовані в функції get_weather.

Отже, розробка Telegram-бота на Python є відносно простою та ефективною методикою для впровадження корисних сервісів. Завдяки легкодоступним інструментам і розширеним можливостям API, навіть простий

бот може надати значний обсяг функцій при незначних зусиллях з боку програміста.

1.2 Використання API-сервісів для отримання погодної інформації

API-сервіси створюють значні можливості для розробки гнучких і масштабованих програмних рішень, які забезпечують доступ до погодних даних з міжнародних метеорологічних баз. У ході дослідження було встановлено, що сучасні API, такі як OpenWeatherMap, WeatherAPI, AccuWeather та інші, пропонують інформацію як про поточні, так і про історичні погодні умови. Це включає параметри, такі як температура повітря, вологість, швидкість і напрямок вітру, атмосферний тиск, рівень хмарності, видимість, а також спеціалізовані показники, такі як індекс ультрафіолетового випромінювання та ймовірність опадів. В таблиці 1.3 надана порівняльна характеристика популярних API для погоди.

Таблиця 1.3 — Порівняльна характеристика популярних API для погоди

API сервіс	**Доступність (Free/Paid)**	**Формат відповіді**	**Прогноз**	**Підтримка української**	**Надійність**
Open Weather Map	Free + Paid	JSON, XML	5 днів	Так	Висока
WeatherAPI	Free + Paid	JSON	7 днів	Ні	Висока
Accu Weather	Paid	JSON	15 днів	Ні	Дуже висока
Weather Stack	Free + Paid	JSON	Поточна	Ні	Середня

API має унікальну характеристику, що дозволяє обирати формат даних — JSON або XML, що спрощує інтеграцію в Python-додатки. Це забезпечує гнучкість у обробці запитів та подальшому форматуванні інформації. Крім того, для підвищення продуктивності можна оптимізувати виклики до API шляхом кешування запитів та використання API-ключів, які мають певні обмеження на кількість запитів за одиницю часу.

Завдяки ретельному аналізу структури відповідей API вдалося реалізувати гнучке формування повідомлень для користувачів, які відображають не лише актуальну інформацію, а й прогнози погоди на кілька днів уперед. Додатково, API забезпечує доступ до координат міста, що відкриває нові можливості для функціоналу — наприклад, можливість включення географічної локації, регіональних прогнозів або автоматичного визначення місцезнаходження користувача за IP-адресою чи GPS-координатами в майбутніх версіях бота.

Для забезпечення функціоналу отримання актуальної інформації про погоду в Telegram-боті застосовується зовнішній API-сервіс, зокрема OpenWeatherMap API. Ці сервіси пропонують структуровані дані про погодні умови в реальному часі, прогнози на кілька днів, а також інформацію про вітер, вологість, атмосферний тиск, погодні явища та інші аспекти.

Основні переваги застосування API-сервісів:

- Отримання актуальних погодних даних у реальному часі.
- Можливість доступу до прогнозів як на короткий, так і на тривалий термін.
- Підтримка різних мов та форматів відповідей (наприклад, JSON або XML).

Гнучка система запитів, що дозволяє передавати назву міста, координати або інші параметри для більш точної інформації.

Взаємодія з API OpenWeatherMap. Після реєстрації на платформі OpenWeatherMap, розробник отримує унікальний ключ доступу (API key), який потрібно додати до HTTP-запитів. Бот формує GET-запит, включаючи параметри

(місто, одиниці вимірювання, API-ключ), і отримує відповідь у форматі JSON (Рис 1.4).

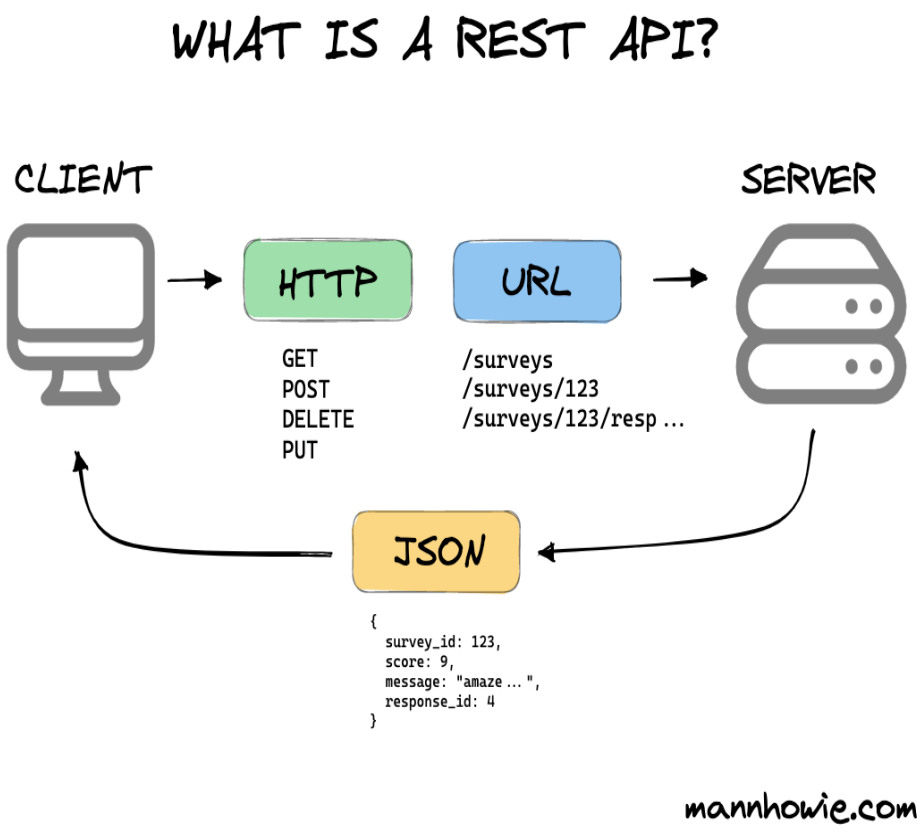


Рисунок 1.4 - Приклад роботи JSON [33]

Обробка відповіді. Відповідь API містить великий масив даних, з якого бот витягує необхідну інформацію:

- поточну температуру (main['temp']);
- температуру, яка відчувається (main['feels_like']);
- опис погоди (weather[0]['description']);
- швидкість вітру (wind['speed']);
- за наявності — пориви вітру (wind['gust']).

В таблиці 1.4 надані основні параметри JSON-відповіді API

Таблиця 1.4 — Основні параметри JSON-відповіді API

Ключ JSON	**Опис**	**Приклад значення**
main['temp']	Температура повітря	15.73
main['feels_like']	Температура, що відчувається	13.96
weather[0]['description']	Опис погодних умов	"хмарно"
wind['speed']	Швидкість вітру	3.1
wind.get('gust')	Порив вітру	5.7

Ці дані форматуються у зручне текстове повідомлення і надсилаються користувачеві в чат Telegram, що описується в табл.1.5.

Таблиця 1.5 — Схема взаємодії Telegram-бота з API OpenWeatherMap

	Дія	**Опис**
1	Користувач вводить назву міста у чаті Telegram	Наприклад: "Kyiv"
2	Бот формує HTTP-запит до API OpenWeatherMap	GET-запит з параметрами: місто, ключ API, одиниці вимірювання
3	API повертає JSON-відповідь	Включає температуру, опис погоди, вітер тощо
4	Python-бот обробляє отримані дані	Витягує ключову інформацію з JSON
5	Бот формує текстове повідомлення	Наприклад: "Хмарно, Температура: 15°C"
6	Telegram-бот надсилає відповідь користувачеві	У вигляді повідомлення в чаті Telegram

1.3 Огляд існуючих бібліотек та рішень для реалізації погодних ботів

У процесі дослідження було вивчено велику кількість публічних репозиторіїв з відкритим кодом [21, 22, 23], що містять проєкти погодних ботів. Було встановлено, що найпоширенішими є бібліотеки telebot (PyTelegramBotAPI), python-telegram-bot, aiogram, а також модулі для виконання HTTP-запитів, такі як requests та httpx, які сприяють легкій інтеграції з API-сервісами.

Варто окремо виділити бібліотеку python-telegram-bot, яка використовує асинхронний підхід і пропонує розширені функції для обробки команд, клавіатур, inline-запитів, callback-кнопок та інших елементів. Це відкриває можливості для створення більш інтерактивних і ефективних ботів, хоча вимагає додаткового вивчення та налаштування асинхронного середовища в Python

Аналізуючи рішення на GitHub [21, 22, 23], виявлено, що багато ботів мають обмежений функціонал, зазвичай надаючи лише інформацію про поточну температуру та опис погоди. У деяких випадках спостерігається відсутність обробки винятків або неправильне форматування відповідей, що може викликати незручності для користувачів.

Розробка погодних чат-ботів вимагає не лише основних навичок програмування, а й використання готових інструментів, які суттєво полегшують реалізацію функціоналу. Існує безліч бібліотек та рішень, що дозволяють швидко створити бота, який взаємодіє з користувачем, надсилає HTTP-запити до погодних API та обробляє отримані дані.

Python-бібліотеки для розробки Telegram-ботів. Серед найбільш вживаних бібліотек для взаємодії з Telegram виділяється pyTelegramBotAPI (Telebot). Ця бібліотека є обгорткою для Telegram Bot API, що пропонує зручний та інтуїтивно зрозумілий інтерфейс для:

- обробки повідомлень;
- створення команд;
- надсилання відповідей;
- роботи з клавіатурами (інлайн, звичайні).

Застосування pyTelegramBotAPI забезпечує швидке створення прототипу чат-бота, не вимагаючи глибокого розуміння низькорівневої реалізації HTTP-запитів до серверів Telegram. Бібліотека містить усі необхідні методи для інтеграції з Telegram, включаючи підтримку розширених функцій, таких як медіа, реакції, кнопки тощо.

API-сервіси для отримання погодної інформації. Серед найпоширеніших сервісів, що забезпечують доступ до погодних даних, можна виділити:

1. OpenWeatherMap API — безкоштовний (з певними обмеженнями) сервіс, який пропонує:

- актуальну інформацію про погоду;
- прогнози на кілька днів уперед;
- погодні умови в реальному часі;
- дані про вітер, вологість, тиск та інші параметри.

2. WeatherAPI — альтернатива з розширеними функціями, що включає підтримку історичних даних про погоду.

3. AccuWeather API та WeatherStack API — популярні платні рішення, які забезпечують високу точність прогнозів і широкі можливості для налаштування запитів.

Бібліотеки JSON для обробки відповідей. Після отримання відповіді від погодного API, яка зазвичай представлена у форматі JSON, для її обробки використовуються стандартні бібліотеки Python, такі як:

json — ця бібліотека дозволяє конвертувати відповідь сервера у словник Python, що спрощує роботу з даними;

requests — основна бібліотека, що використовується для виконання HTTP-запитів до зовнішніх API.

Поєднання цих бібліотек з Telegram API створює повний набір інструментів для розробки простого погодного бота.

Приклади існуючих рішень. Існує велика кількість відкритих репозиторіїв на GitHub, де розробники публікують свої погодні боти, наприклад:

- Simple Weather Bot — бот, який показує погоду по місту;

- Telegram Weather Forecast Bot — з підтримкою прогнозу на кілька днів, використанням кнопок та повідомлень з форматуванням;
- WeatherBot with geolocation — більш складний бот із підтримкою геолокації

1.4 Визначення недоліків існуючих рішень та мотивація для розробки власного інструменту

Вивчивши численні рішення, можна зазначити, що основними недоліками більшості з них є обмежений функціонал, недостатня адаптація до потреб реальних користувачів, а також відсутність валідації введених даних. У багатьох випадках інтерфейс бота не є інтуїтивно зрозумілим: повідомлення не мають чіткої структури, відсутня локалізація, а відповіді виглядають як необроблені JSON-дані або англомовні фрази без належних пояснень.

Більшість реалізацій також не враховує потреби україномовної аудиторії. Деякі боти не здатні обробляти випадки, коли введене місто не вдається знайти або коли API повертає помилку, що призводить до завершення діалогу або надсилання неінформативного повідомлення, що негативно впливає на користувацький досвід.

Мотивація для розробки власного рішення з'явилася внаслідок прагнення створити зручний, адаптивний та зрозумілий для користувачів Telegram-бот. Цей бот не лише надає інформацію про погоду, а й формує детальний звіт, враховуючи локальні мовні налаштування, перевіряє введені дані, структурує повідомлення у зрозумілій формі та забезпечує можливість подальшого розширення функціоналу.

Основні недоліки існуючих рішень (Табл.1.6):

Таблиця 1.6 — Недоліки наявних рішень

Недолік	**Приклад або наслідки**
Відсутність локалізації	Вивід англійською мовою
Недостатня валідація вводу	"Kyivvvv" не розпізнається – бот зависає або мовчить
Погана структура повідомлення	Відповідь – просто JSON без оформлення
Складний інтерфейс	Надмірно довгі повідомлення без акцентів
Низька масштабованість	Неможливо додати прогноз на кілька днів без змін у всьому коді

1) Складний або перевантажений інтерфейс. Деякі боти намагаються надати надмірну кількість інформації в одному запиті, що ускладнює її сприйняття користувачами. Це особливо стосується ботів, які представляють прогнози погоди на кілька днів у вигляді великого безструктурного тексту без належного форматування.

2) Недостатня обробка помилок і виключень. Часто боти не здатні адекватно реагувати на некоректні запити користувачів. Наприклад, якщо користувач помилково введе назву міста, бот може або "зависнути", або зовсім не реагувати.

3) Обмежені можливості. Деякі боти надають лише інформацію про поточну температуру, не враховуючи інші важливі параметри, такі як швидкість вітру, порив вітру, як відчувається та інші.

4) Відсутність можливості масштабування. Численні проєкти не забезпечують можливості для розширення. Наприклад, додати прогноз на кілька днів або змінити джерело погодних даних без істотних змін у коді є неможливим.

5) Не оптимізований код. У деяких рішеннях виявляється повторення коду, недостатня модульність та відсутність чіткої структури, що ускладнює підтримку та подальший розвиток проєкту.

Приклади не оптимізованого коду надані в додатку А.

Мотивація для створення власного бота. У відповідь на виявлені проблеми було прийнято рішення розробити власного Telegram-бота, який включатиме:

- зручний та інтуїтивно зрозумілий інтерфейс;
- ясну структуру подання погодної інформації;
- базову перевірку введених даних користувача;
- використання API OpenWeatherMap, одного з найпопулярніших і зручних погодних сервісів;
- можливість простого розширення функціоналу в майбутньому.

В таблиці 1.7 надані основні вимоги до створення власного Telegram-бота

Таблиця 1.7 — Основні вимоги до створення власного Telegram-бота

Вимога	**Обґрунтування**
Простота у використанні	Інтуїтивний інтерфейс і короткі відповіді
Українська мова	Актуально для локального користувача
Підтримка помилок	Повідомлення при некоректному вводу
Гнучкість структури коду	Можливість масштабування в майбутньому
Ефективність	Швидке формування і надсилання відповіді

Отже, створення власного інструменту дозволяє не лише уникнути типових помилок попередніх версій, але й розробити продукт, максимально орієнтований на зручність та ефективність взаємодії з користувачем.

Таким чином, в цьому розділу було розглянуто теоретичні та практичні аспекти, які є необхідними для розробки Telegram-бота, що забезпечує отримання інформації про погоду. Проведено аналіз принципів створення чат-ботів у платформі Telegram за допомогою мови програмування Python, а також визначено роль і можливості Telegram Bot API, які слугують основою для взаємодії бота з користувачами.

Особлива увага була зосереджена на застосуванні зовнішніх API-сервісів для збору погодних даних, зокрема API OpenWeatherMap, що пропонує різноманітні погодні параметри у зручному для користувача форматі. Було розглянуто принципи взаємодії з такими API, основи обробки HTTP-запитів та структуру даних, які надходять у відповідь.

Було здійснено огляд відомих бібліотек і реалізацій погодних ботів, щоб вивчити досвід, накопичений розробниками. У результаті цього аналізу були виявлені основні недоліки таких рішень, зокрема недостатня обробка помилок, обмежена гнучкість та брак можливостей для масштабування.

В результаті проведеного аналізу було сформульовано чітке обґрунтування необхідності розробки власного програмного рішення, яке відзначатиметься простотою у використанні, стабільністю, орієнтацією на користувача та можливістю подальшого функціонального розширення. Отримані теоретичні знання стали основою для реалізації власного Telegram-бота, призначеного для отримання інформації про погоду, що буде детально розглянуто в наступному розділі.

2 ДОСЛІДЖЕННЯ ЗАСТОСУВАННЯ PYTHON У РОЗРОБЦІ ЧАТ-БОТІВ TELEGRAM ДЛЯ ОТРИМАННЯ ПОГОДНИХ ДАНИХ

2.1 Загальний огляд Telegram Bot API та його можливостей

Telegram Bot API — це спеціалізований інтерфейс, створений компанією Telegram для розробки ботів, які є автоматизованими програмами, що взаємодіють з користувачами на платформі Telegram. Цей API пропонує набір HTTP-методів, які дозволяють розробникам створювати ботів, здатних отримувати та відповідати на повідомлення, реагувати на команди, надсилати медіа та виконувати інші дії в чатах.

Telegram Bot API є безкоштовним і відкритим ресурсом, що сприяло його популярності серед розробників по всьому світу. Його легкість у використанні, чітка документація та надійність стали основою для швидкого зростання екосистеми ботів у Telegram.

У рамках дипломної роботи було застосовано Telegram Bot API для обробки команд користувачів, зокрема команди /start, а також для перехоплення текстових повідомлень, які надсилаються до бота. Головною перевагою цього API є його зручність: усі взаємодії відбуваються через HTTPS-запити, що легко реалізується в Python за допомогою бібліотек, таких як telebot. API підтримує як звичайні запити, так і Webhook, проте в даній реалізації було обрано метод опитування (polling) через його простоту налаштування для локального використання.

Окрім основних функцій, Telegram Bot API також надає можливість реалізації клавіатур, inline-кнопок, меню для вибору міста, відображення зображень та інтеграції з іншими платформами. Це дозволяє створювати складні системи, які перевищують простий текстовий чат. У майбутньому ці можливості

можуть бути використані для покращення бота, наприклад, шляхом додавання геолокаційного пошуку або push-повідомлень про зміни погодних умов.

Основні можливості Telegram Bot API (Рис.2.1):

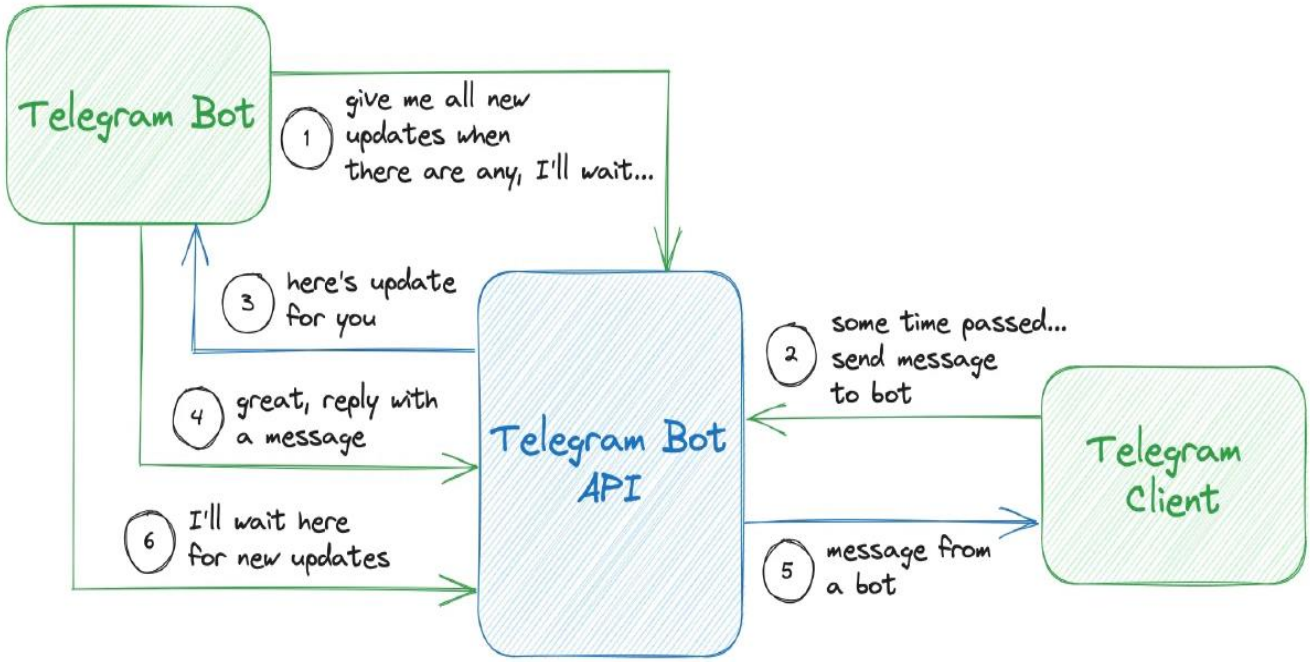


Рисунок 2.1 - Бот отримує оновлення через long polling [31]

На рис.2.1 надані такі основні можливості Telegram Bot API о

1. Прийом і обробка повідомлень. Бот має можливість отримувати текстові повідомлення, команди, документи, зображення, аудіофайли та відео, а також відповідати на них відповідно до визначеної логіки.
2. Використання команд. Боти здатні реагувати на команди, що починаються зі знака /, такі як /start, /help, /weather. Це сприяє організації взаємодії з користувачем і спрощує процес навігації.
3. Робота з кнопками та клавіатурами. API надає можливість створювати інлайн-кнопки та індивідуальні клавіатури, що робить навігацію в боті більш зручною та інтуїтивно зрозумілою.
4. Підтримка мультимедійного контенту. Чат-боти здатні обмінюватися зображеннями, відео, голосовими повідомленнями, а також файлами різних форматів.

5. Геолокаційні дані та контакти. Користувачі мають можливість надсилати свою геолокацію або контактні дані, що створює численні можливості для інтерактивних сервісів.
6. Вебхуки (Webhooks). Telegram пропонує два методи взаємодії з ботами: опитування (long polling) та вебхуки. Використання вебхуків дає змогу налаштувати сервер, що автоматично отримуватиме повідомлення при кожній новій події.

В таблиці 2.1 надано характеристика можливості Telegram Bot API

Таблиця 2.1 — Характеристики можливості Telegram Bot API

№	**Можливість**	**Короткий опис**
1	Обробка повідомлень	Отримання та відповідь на текст, документи, зображення
2	Використання команд	Наприклад: /start, /help, /weather
3	Кнопки та клавіатури	Inline-кнопки, reply-клавіатури
4	Підтримка медіа	Надсилання фото, відео, голосових повідомлень
5	Геолокація та контакти	Отримання координат або контактів користувача
6	Inline-запити	Використання бота в сторонніх чатах без переходу до нього
7	Аутентифікація	Авторизація користувача через Telegram Login Widget
8	Методи розгортання	Polling (локально) або Webhook (на сервері)

Обробка inline-запитів. API Telegram підтримує режим "inline", що дозволяє використовувати бота безпосередньо в будь-якому чаті, без потреби переходити до нього.

Аутентифікація користувачів. Використовуючи API віджета авторизації Telegram, можна забезпечити надійну авторизацію користувачів у зовнішніх веб-додатках.

Розгортання бота. Для створення бота потрібно виконати такі кроки (Рис 2.2):

- використати спеціального бота @BotFather у Telegram для реєстрації нового бота;
- отримати токен доступу — унікальний ключ, що слугує для автентифікації;
- застосувати цей токен у програмі на Python за допомогою бібліотек, таких як python-telegram-bot або telebot.

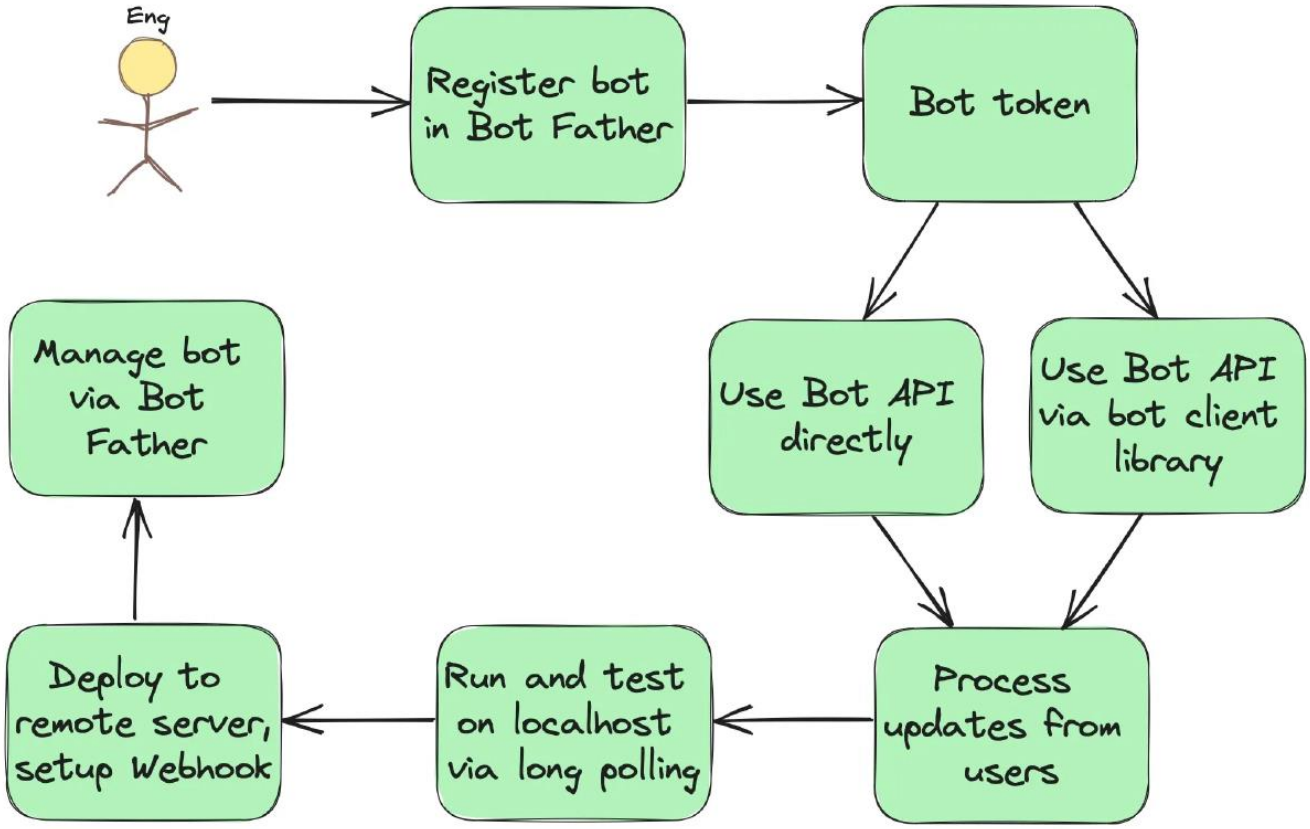


Рисунок 2.2 - Кроки для створення бота [31]

Переваги Telegram Bot API:

- Безкоштовність та доступність
- Гнучкість у налаштуванні взаємодії
- Можливість розширення функціоналу
- Високий рівень захисту
- Розвинена спільнота розробників та підтримка

Telegram Bot API забезпечує всі необхідні умови для розробки функціонального, надійного та зручного чат-бота, що дозволяє розширити можливості взаємодії користувачів з цифровими сервісами без необхідності створення повноцінного мобільного або веб-додатку.

2.2 Використання Python для обробки запитів та форматування відповідей

Python є відмінним вибором для створення Telegram-ботів завдяки своїм численним бібліотекам та простому синтаксису. У рамках цього проєкту Python використовувався для розробки логіки обробки запитів від користувачів, формування HTTP-запитів до зовнішнього API (OpenWeatherMap), обробки отриманих JSON-відповідей та створення відповідних текстових повідомлень для користувачів.

Функція `@bot.message_handler(content_types=['text'])` виконує роль перехоплення текстових повідомлень, які надсилає користувач. Після цього програма обробляє отриманий текст, очищає його та нормалізує, формуючи запит до API OpenWeatherMap. Вона також перевіряє статус відповіді та витягує необхідні елементи з JSON-структури.

Отримані дані потім форматуються у зручний для читання текст, доповнений емодзі для покращення сприйняття.

В таблиці 2.2 надано порівняння методів обробки повідомлень у Python

Таблиця 2.2 — Порівняння методів обробки повідомлень у Python

Метод	**Опис**	**У боті використано**
@bot.message_handler()	Декоратор, що обробляє вхідні повідомлення	Так
bot.send_message()	Відправка повідомлення користувачу	Так
bot.reply_to()	Відповідь безпосередньо на повідомлення користувача	Так
bot.polling()	Запуск нескінченного циклу прослуховування	Так
Webhook	Прослуховування через сервер	Ні

Особливу увагу було приділено форматуванню відповіді за допомогою Markdown, що дозволяє акцентувати увагу на заголовках та робити текст більш структурованим і привабливим. У коді також реалізовано перевірку наявності додаткових параметрів, таких як пориви вітру (wind['gust']), які відображаються лише за необхідності. Використання бібліотек requests, json, datetime у поєднанні з telebot забезпечило реалізацію всього функціоналу без зайвих ускладнень, але з достатньою гнучкістю для подальшого розвитку.

Python забезпечує ефективну реалізацію логіки функціонування бота, зокрема:

- прийом запитів від користувачів;
- взаємодію з погодним API;
- обробку отриманих даних;
- форматування відповіді у зручному для користувача форматі;
- гарантування обробки виключень та помилок.

Основні бібліотеки, що застосовуються:

● telebot (pyTelegramBotAPI). Ця популярна бібліотека на Python призначена для роботи з Telegram Bot API. Вона спрощує обробку команд користувачів, надсилання відповідей, а також взаємодію з кнопками.

● Requests. Ця бібліотека використовується для виконання HTTP-запитів і дозволяє взаємодіяти з OpenWeatherMap API для отримання погодних даних у форматі JSON.

● Json. Цей стандартний модуль Python призначений для обробки даних у форматі JSON, що дозволяє перетворювати відповіді API в об'єкти Python для подальшої обробки.

Обробка запитів від користувачів. Для обробки запитів від користувачів у Telegram-боті існує кілька основних підходів залежно від використовуваної бібліотеки та мови програмування. Для зберігання даних користувачів та історії запитів використовуйте бази даних. Основні типи обробки запитів: текстові команди (/start, /help тощо), текстові повідомлення, кнопки та інтерактивні елементи, Inline-запити, голосові повідомлення та файли

Користувач надсилає боту повідомлення, в якому вказує назву міста, наприклад: Київ. Програма на Python реєструє цей запит і викликає функцію, що взаємодіє з API OpenWeatherMap, використовуючи вказаний параметр. Приклад наданий у додатку А.

Отриманий JSON включає дані про температуру, атмосферний тиск, швидкість вітру, опис погодних умов та інші показники.

Форматування відповідей. Після збору та обробки даних відповідь для користувача подається у зручному текстовому вигляді. Приклад наданий у додатку А.

Таке подання надає користувачеві можливість миттєво отримати всебічну, організовану та зрозумілу інформацію.

В таблиці 2.3 надано формат відповіді користувачу

Таблиця 2.3 — Формат відповіді користувачу

Частина повідомлення	**Опис**
Вступ	Тип погоди (хмарно, ясно тощо)
Основні показники	Температура, відчувається, вітер, пориви
Прогноз на 3 дні	По 1 запису на день, обраного на 12:00
Завершення	Заклик повторити запит, ввести нове місто

Обробка виключень. Python також забезпечує ефективну обробку випадків, коли:

- користувач вводить неправильну назву міста;
- сервер OpenWeatherMap не відповідає;
- API-ключ недійсний або перевищено ліміт запитів.

Приклад наданий у додатку А.

В таблиці 2.4 надано валідація та обробка помилок

Таблиця 2.4 — Валідація та обробка помилок

Ситуація	**Реакція бота**	**Як реалізовано**
Введено неіснуюче місто	Повідомлення: «Немає такого міста, напишіть ше раз»	status_code != 200
API повертає помилку	Бот не падає, а відповідає	Блок else
Відсутній параметр gust	Відповідь без поривів вітру	if 'gust' in data['wind']
Порожнє повідомлення	Бот просить ввести місто	Через strip().lower()

Переваги застосування Python

- Простота інтеграції з Telegram Bot API.
- Широкий асортимент бібліотек для роботи з HTTP-запитами.
- Зручність у обробці даних формату JSON.
- Швидке розгортання та підтримка програмного коду.
- Зрозумілий синтаксис, що підходить як для новачків, так і для досвідчених програмістів.

Отже, Python є відмінним вибором для створення погодного чат-бота завдяки своїм потужним можливостям обробки даних та високій інтеграції з API-сервісами. Використання цієї мови програмування не лише сприяє оптимізації процесу розробки, але й забезпечує надійну та зручну взаємодію між користувачем і ботом.

2.3 Приклади існуючих погодних ботів, написаних на Python

У рамках проведеного аналізу було вивчено кілька прикладів Telegram-ботів, створених на Python, що обробляють метеорологічну інформацію. Усі ці боти реалізують схожий алгоритм: отримання текстового запиту, звернення до погодного API (зазвичай OpenWeatherMap або WeatherAPI) та надсилання відповіді користувачу.

Більшість наведених прикладів ґрунтуються на бібліотеках python-telegram-bot або telebot. Спільною характеристикою є застосування форматування Markdown або HTML, обробка помилок, а іноді — реалізація inline-клавіатур. У деяких ботах реалізовано розширені функції, такі як історія запитів, підтримка GPS-координат, багатомовність та push-сповіщення. Ці приклади стали джерелом натхнення та основою для порівняльного аналізу.

На фоні наведених прикладів, розроблений бот має перевагу завдяки простоті впровадження, зберігаючи при цьому високу інформативність відповідей. У нашій реалізації також використано унікальний підхід — вибір прогнозу саме на полудень, що не завжди враховується в інших аналогічних

рішеннях. Більшість ботів або надають занадто багато інформації одночасно, або обмежуються лише поточними погодними умовами, втрачаючи при цьому прогностичний аспект.

Приклади взяті з GitHub — це веб-платформа для зберігання, керування та спільної роботи над проєктами, які використовують систему контролю версій Git. GitHub — це як соціальна мережа для програмістів, де можна: зберігати свій код (у "репозиторіях"), працювати разом над проєктами, бачити зміни в коді, відстежувати помилки (issues), пропонувати покращення (pull requests). Має такі основні можливості: контроль версій — відстеження змін у коді; спільна робота — кілька людей можуть працювати над одним проєктом, хостинг проєктів — розміщення як публічних, так і приватних репозиторіїв; автоматизація — перевірка коду, тестування, деплой .

Далі представлені кілька прикладів:

1. lesskop/shtosh-weather-bot [21]

Мова реалізації: Python

Технології: aiogram, aiohttp, .env, асинхронна архітектура

API: OpenWeatherMap

Особливості:

- Використовує асинхронну обробку запитів через aiogram, що дозволяє ботам працювати ефективніше.
- Автоматично визначає місцезнаходження користувача (через IP або координати).
- Вивід інформації супроводжується емодзі, що робить відповіді зручними й візуально зрозумілими.
- Безпечне зберігання ключів через .env файл.

2. karvozavr/weather-bot [22]

Мова реалізації: Python

Технології: aiogram, aiohttp, dotenv

API: OpenWeatherMap

Особливості:

- Побудований з використанням сучасного асинхронного фреймворку aiogram, що забезпечує масштабованість.
- Рекомендації з одягу – унікальна фішка: бот оцінює температуру та погодні умови і надає пораду, як одягтися.
- Простий і лаконічний — підходить як шаблон для реалізації навчального бота.
- Відповідає на текстові запити, обробляє міста українською та англійською.
- Підтримує використання Markdown форматування для покращення структури відповідей.

3. mustafababil/Telegram-Weather-Bot [23]

Мова реалізації: Python

Технології: python-telegram-bot, requests

API: OpenWeatherMap

Особливості:

- Приймає текстову назву міста або геолокацію користувача.
- Показує:
  1)Температуру (поточну, мінімальну, максимальну),
  2)Погодний опис,
  3)Іконки погоди (емодзі).
- Реалізований із підтримкою Webhook (для деплою на Google App Engine), що робить його готовим до хостингу.
- Має досить просту структуру коду, що дозволяє легко зрозуміти та модифікувати логіку.
- Ключі API та токен зберігаються в .env.

В таблиці 2.5 надана порівняльна характеристика

Таблиця 2.5 — Порівняльна характеристика

Бот	Асинхронний	Прогноз	Геолокація	Рекомендації	Webhook Ready
№1	Так	Поточна	Так	Ні	Ні
№2	Так	Поточна	Ні	Так	Ні
№3	Ні	Поточна	Так	Ні	Так

2.4 Визначення вимог до функціоналу інструментів

Після проведення аналізу наявних погодних Telegram-ботів та дослідження можливостей Telegram Bot API, стало зрозуміло, що для розробки ефективного, зручного та стабільного інструменту необхідно чітко визначити вимоги до його функціональності. Це не лише допоможе структурувати процес розробки, але й забезпечить відповідність між очікуваннями користувачів і можливостями системи.

Перед створенням Telegram-бота важливо визначити чіткий список вимог до його функціональності, що ґрунтується на потребах цільової аудиторії та поставлених цілях. Для погодного бота ці вимоги можуть включати: можливість введення назви міста у зручному форматі, підтримку української мови, отримання актуальних даних про погоду на сьогодні та на кілька наступних днів, логічне форматування відповідей та швидкий час реакції.

Бот має бути стійким до некоректних введень з боку користувача та мати адаптивну систему відповідей, що включає підказки або приклади. Рекомендується також впровадити команду /start, яка надаватиме коротку інструкцію, забезпечити підтримку клавіатури Telegram для полегшення навігації.

Окрім цього, слід враховувати технічні вимоги, такі як надійність API-сервісу, з якого отримуються погодні дані, стабільність коду, а також легкість

розгортання на сервері чи хостингу, наприклад, через платформи Heroku або Railway. Важливим етапом є також визначення типів повідомлень, які бот повинен надсилати у разі виникнення помилок, відсутності відповіді з API або некоректного введення.

У процесі формулювання вимог також беруться до уваги можливі розширення функціональності в майбутньому, такі як одночасне прогнозування для кількох міст, геолокаційне визначення користувача, надсилання повідомлень у визначений час доби, а також інтеграція з іншими сервісами, наприклад, месенджерами або календарями. Складання повного переліку вимог сприяє зменшенню ризиків виникнення помилок під час розробки та робить проєкт більш адаптивним і масштабованим.

Основні функціональні вимоги:

- Прийом текстових повідомлень від користувача: Бот повинен реагувати на введене повідомлення (наприклад, назву міста), яке буде використано для формування запиту до погодного API.
- Отримання даних з OpenWeatherMap API: Для забезпечення функціональності бота необхідно налагодити правильну взаємодію з сервісом OpenWeatherMap за допомогою HTTP-запитів.
- Відображення актуальної погоди: Головною метою бота є надання користувачеві даних про температуру, погодні умови та інші важливі параметри.
- Відображення прогнозу погоди на три дні: Бот має забезпечувати детальний прогноз погоди на найближчі кілька днів, включаючи основні показники.
- Обробка некоректних запитів: Якщо користувач введе неправильну назву міста або виникне помилка під час звернення до API, бот зобов'язаний надати відповідне повідомлення про помилку.

Автоматичне форматування відповіді: Дані, отримані з API, повинні бути перетворені у зручний для сприйняття формат (наприклад, інформація, структурована за днями, з використанням емодзі тощо).

В таблиці 2.6 надані основні функціональні вимоги до Telegram-бота

Таблиця 2.6 — Основні функціональні вимоги до Telegram-бота

№	Вимога	Призначення
1	Прийом текстових повідомлень	Отримання запиту від користувача (місто)
2	Запит до OpenWeatherMap	Отримання погодних даних
3	Відображення поточної погоди	Температура, опис, вітер
4	Прогноз на 3 дні	Прогноз у певний час (12:00) на кожен день
5	Обробка помилок	Відповідь у разі помилки або невірного вводу
6	Форматування відповіді	Використання емодзі, структурування інформації

Додаткові вимоги (необов'язкові):

- Підтримка різних мов.
- Можливість подальшого розширення функціоналу (наприклад, впровадження тижневого прогнозу, інтеграція з геолокаційними сервісами).
- Запобігання повторним ідентичним запитам (оптимізація шляхом кешування результатів).

Нефункціональні вимоги:

- Продуктивність: Бот зобов'язаний відповідати на запити протягом 1–2 секунд.
- Надійність: Система повинна функціонувати стабільно, навіть у випадку тимчасових збоїв з боку API погоди.
- Масштабованість: Архітектурне рішення повинно забезпечувати можливість розширення функціональних можливостей без значних модифікацій основного коду.
- Легкість розгортання: Проєкт має містити зрозумілу структуру та

інструкції для його запуску.

В таблиці 2.7 надані додаткові та нефункціональні вимоги до бота

Таблиця 2.7 — Додаткові та нефункціональні вимоги до бота

Тип вимоги	**Приклади**
Додаткові	Підтримка кількох мов, можливість подальшого розширення, кешування запитів
Продуктивність	Час відповіді — до 2 секунд
Надійність	Витримка навантаження, стабільна робота навіть при помилках з боку API
Масштабованість	Можливість додавання нових функцій без зміни базової структури
Легкість розгортання	Зрозуміла структура, простий запуск на Heroku або Railway

Чітке визначення вимог сприяє впровадженню структурованого підходу до розробки Telegram-бота. Виходячи з наведених критеріїв, буде створено бота, який надає користувачам точну, актуальну та зручну для сприйняття інформацію про погоду, з можливістю подальшого масштабування та вдосконалення.

Таким чином, в цьому розділу розглянуто основні технічні та архітектурні аспекти розробки Telegram-бота, призначеного для отримання інформації про погоду. Проведено глибокий аналіз можливостей Telegram Bot API, який є ключовим інструментом для створення ботів у месенджері Telegram. Особливу увагу приділено мові програмування Python, яка, завдяки своїй простоті, гнучкості та широкому вибору бібліотек, є надзвичайно зручною для реалізації таких застосунків. Було представлено приклади вже існуючих погодних Telegram-ботів, розроблених на Python, що дало змогу не лише виявити найпоширеніші методи реалізації, а й проаналізувати як переваги, так і недоліки таких рішень. На

ції основі було здійснено формалізацію функціональних та нефункціональних вимог, які мають бути реалізовані в рамках запропонованого проєкту. Розділ заклав надійний фундамент для розробки власного Telegram-бота, що відповідатиме сучасним стандартам зручності, швидкості, точності та надійності. У наступному розділі буде розглянуто етапи впровадження системи, вибір інструментів, структуру програми та детальний опис її ключових компонентів.

3. ТЕХНІЧНА РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ ОТРИМАННЯ ПОГОДИ В ЧАТ-БОТІ TELEGRAM

3.1 Реалізація інструментів для отримання поточної погоди та прогнозу на кілька днів

У цьому проєкті Telegram-бот був розроблений за допомогою бібліотеки telebot, що забезпечує зручну інтеграцію з API Telegram. Основною метою бота є надання користувачеві актуальних даних про погоду, включаючи поточну температуру, метеорологічні умови, вологість, атмосферний тиск, швидкість і напрямок вітру, а також прогноз погоди на кілька днів уперед.

Інформація отримується за допомогою сервісу OpenWeatherMap, що забезпечує можливість виконання HTTP-запитів з такими параметрами, як назва населеного пункту, одиниці вимірювання, мова відповіді та API-ключ. Після відправлення запиту, відповідь надходить у форматі JSON, який потім обробляється за допомогою Python-коду. Бот формує відповідь, що має логічну структуру і є зрозумілою для звичайного користувача.

Прогноз погоди був створений на основі 3-годинних даних, отриманих від OpenWeatherMap. Додатково була реалізована фільтрація, яка виділяє значення, що відповідають 12:00 — часу, що найкраще відображає денні погодні умови. Це полегшує аналіз даних і допомагає уникнути зайвої деталізації.

Локалізація на українську мову реалізована за допомогою параметра lang=ua, що дозволяє відображати описи погодних умов рідною мовою. Це сприяє покращенню взаємодії користувачів з ботом для тих, хто спілкується українською.

Формування запитів до OpenWeatherMap. Для отримання поточної погоди(рис. 3.1) використовується базовий запит на адресу. Приклад наданий у додатку А.

У цьому запиті параметр q визначає місто, для якого потрібна інформація, appid є унікальним API-ключем, що ідентифікує користувача, units=metric забезпечує відображення температури в градусах Цельсія, а lang=ua дозволяє отримати опис погоди українською мовою.

Для прогнозу погоди на кілька днів (рис. 3.2) використовується інший endpoint. Приклад наданий у додатку А.

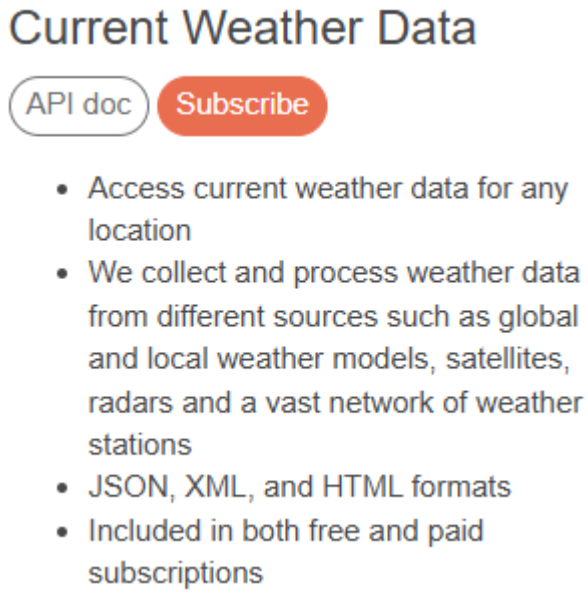


Рисунок 3.1 - Де взяти запит на адресу для поточної погоди [2]

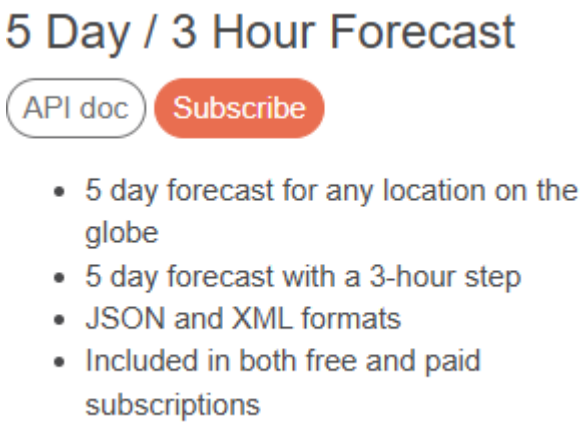


Рисунок 3.2 - Де взяти запит на адресу для прогнозу погоди [2]

У відповідь на цей запит надається перелік прогнозів з трьохгодинним інтервалом, з якого необхідно обрати найбільш значущі дані (наприклад, значення на полудень або середні показники за день) (рис. 3.3).

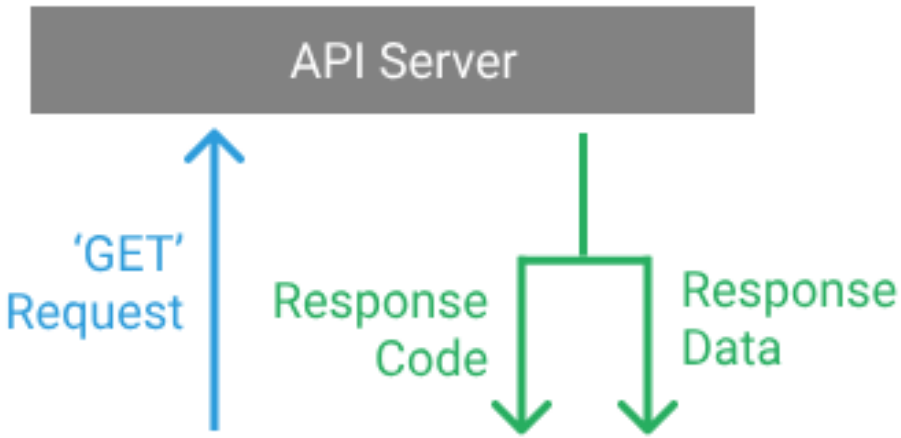


Рисунок 3.3 - Типовий запит до API, який передбачає надсилання запиту до сервера та отримання відповіді. [40]

Обробка даних та їх форматування. Отримані дані представлені у форматі JSON, що дозволяє зручно витягувати потрібну інформацію: температуру, відчутну температуру, опис погодних умов швидкість вітру і так далі. У коді бота реалізовано функцію, яка виконує запит, перевіряє коректність відповіді та формує відформатоване повідомлення для Telegram. Приклад наданий у додатку А.

Для здійснення прогнозування застосовується інша функція, яка перебирає елементи прогнозу та вибирає необхідну інформацію. Приклад наданий у додатку А.

Взаємодія бота . Бот реагує на введення назви міста. Користувач отримує у відповідь актуальну погоду і прогноз.

Обробка помилок . У випадку, якщо користувач вводить неіснуюче місто або виникає проблема з підключенням до API, бот ввічливо інформує про помилку, що підвищує зручність взаємодії.

У цьому підпункті представлені ключові інструменти для збору та обробки метеорологічних даних за допомогою OpenWeatherMap API. Отримані відомості

перетворюються у зручний для користувача формат, що забезпечує комфортне використання Telegram-бота.

3.2. Автоматизація форматування повідомлень для користувача

Ефективність взаємодії користувача з Telegram-ботом значною мірою визначається якістю подачі інформації. Навіть найточніші дані можуть стати незрозумілими або нецікавими, якщо вони представлені в незручному або важкочитабельному форматі. Тому одним із ключових аспектів розробки Telegram-бота є автоматизація форматування повідомлень, які бот надсилає у відповідь на запити користувачів.

Автоматичне форматування відповідей бота здійснюється за допомогою функцій Python, зокрема через форматування рядків і застосування емодзі. Емодзі надають візуальну привабливість і ефективно передають суть повідомлення.

Інформація організована так, що в кожному рядку представлена лише одна характеристика погоди, така як температура, опис, швидкість вітру, тиск, вологість тощо. Прогноз також містить дні тижня з відповідними датами, що підвищує зручність його сприйняття.

Формати часу та дати, які використовуються надані в додатку А.

Було впроваджено повторне запрошення до введення нового міста після отримання відповіді. Це здійснено на основі циклічного спілкування: після кожного запиту користувача бот реагує, надає результат і пропонує продовжити взаємодію. Таким чином, забезпечується безперервний діалог без необхідності перезапуску бота.

Цілі автоматизованого форматування. Автоматизоване форматування повідомлень дозволяє досягнути кількох важливих завдань:

- підвищення читабельності текстів,
- забезпечення структурованості даних
- використання емодзі для швидшого візуального сприйняття інформації,
- формування відповідей, які є лаконічними, але водночас інформативними.

Використання Markdown у Telegram. Telegram підтримує базовий синтаксис Markdown, що дозволяє виділяти текст жирним шрифтом, курсивом, створювати списки, вставляти гіперпосилання та форматувати текст з нових рядків. У Python це реалізується через параметр parse_mode='Markdown' або parse_mode='HTML' у функціях для надсилання повідомлень. Приклад наданий у додатку А.

Форматування повідомлень про поточну погоду. Приклад автоматично сформованого повідомлення для поточної погоди. Приклад наданий у додатку А.

Цей метод забезпечує динамічне формування тексту з інтеграцією актуальних даних та оформленням для зручного відображення в месенджері.

Форматування прогнозу на декілька днів. Оскільки прогноз містить повторювані блоки даних для кожного дня, у форматуванні використовується шаблон з повторюваними елементами. Приклад наданий у додатку А.

Цей формат може бути легко адаптований для більшої кількості днів або параметрів, зберігаючи при цьому високу читабельність.

Локалізація та динаміка мовлення повідомлень. Важливим аспектом є забезпечення підтримки української мови у всіх відповідях. Завдяки параметру lang=ua у запиті до OpenWeatherMap API, бот отримує опис погодних умов безпосередньо українською, що усуває необхідність у додатковому перекладі.

Автоматизація форматування повідомлень є важливим етапом у створенні чат-бота, націленого на кінцевого споживача. Правильно оформлене повідомлення значно спрощує сприйняття інформації, підвищує зручність використання та покращує загальне враження від взаємодії з ботом.

3.3. Розробка функції пошуку погоди за назвою міста з валідацією введення

Функція пошуку погоди реалізована в рамках обробки повідомлень від користувача. Коли бот отримує текстове повідомлення, він інтерпретує його як назву міста. Щоб уникнути помилок, дані очищуються від зайвих пробілів і перетворюються в нижній регістр, що забезпечує однорідність запитів.

На стадії обробки відповіді від API здійснюється перевірка статус-коду. У випадку, якщо він становить 200, бот переходить до етапу аналізу даних. Якщо ж статус-код інший, користувач отримує сповіщення про некоректність введеної назви міста. У цьому сповіщенні також міститься рекомендація, яка допомагає уникнути повторних помилок.

Було впроваджено можливість введення української мови, що дозволяє користувачам вводити назви населених пунктів без необхідності транслітерації. Це стало можливим завдяки підтримці кириличного алфавіту в параметрах запитів до OpenWeatherMap.

Логіка реалізації в боті. У боті, створеному на Python з використанням бібліотеки telebot, реалізація цієї функції здійснюється в обробнику повідомлень типу text. Користувач надсилає боту назву міста, після чого бот намагається отримати відповідь з API OpenWeatherMap. У випадку успішного запиту дані форматуються та повертаються користувачу, в іншому випадку бот сповіщає про виникнення помилки. Приклад наданий у додатку А.

Цей фрагмент коду здійснює перевірку успішності запиту до API (res.status_code == 200). У разі виникнення помилки, користувач отримує детальне повідомлення про помилку.

Аспекти валідації . У простій, але дієвій формі, перевірка полягає в наступних діях:

- Очищення введення: за допомогою функції strip() видаляються зайві пробіли, а функція lower() перетворює текст у нижній регістр для уніфікації.
- HTTP-перевірка відповіді API: якщо API повертає дані — місто існує.
- Повідомлення у разі помилки: бот чітко вказує на те, що місто не знайдено.

Цей метод дозволяє обійти складну попередню верифікацію та базується на перевірці відповіді від API, що суттєво спрощує процес реалізації.

Переваги вибраного підходу:

- Швидкість реалізації. Простий механізм перевірки через API дозволяє зосередитися на обробці результатів.

- Гнучкість. Пошук працює з назвами будь-яких міст, що містяться в базі OpenWeatherMap.
- Зручність для користувача. У разі помилки користувач не стикається з системним кодом, а отримує зрозуміле повідомлення.

Отже, у цьому розділі проаналізовано впровадження механізму пошуку погодних даних за назвою населеного пункту в Telegram-боті з акцентом на валідацію введених даних. Завдяки елементарній логіці перевірки статусу відповіді від API, досягається надійність функціонування та зручність для користувача. Реалізація цього елемента є основоположною, але надзвичайно важливою складовою у розробці будь-якого погодного бота.

3.4. Оптимізація продуктивності чат-бота та обробка помилок у запитах

У процесі створення Telegram-бота ключовим є забезпечення стабільності функціонування та швидкості обробки запитів. Оскільки бот взаємодіє з зовнішнім API (OpenWeatherMap), можуть виникати різноманітні помилки: невірна відповідь API, недоступність сервера, некоректне введення даних користувачем або внутрішні виключення в коді. Оптимізація продуктивності та адекватна обробка таких ситуацій сприяє створенню надійного та зручного для користувача бота.

Для досягнення високої швидкості роботи бота структура запитів розроблена з максимальною лаконічністю. Використовуються лише найнеобхідніші параметри, а обробка JSON-відповідей виконується за допомогою стандартних методів Python, які не вимагають додаткових бібліотек.

Бот не використовує асинхронне програмування, проте завдяки простій логіці та відсутності блокуючих операцій, відповіді надходять швидко. Кількість запитів обмежена лише двома викликами до API, що сприяє зменшенню часу затримки.

Усі запити до API супроводжуються перевіркою статусу відповіді. У випадку виникнення помилки бот не лише інформує про її наявність, але й надає

рекомендації для вирішення проблеми. Наприклад, якщо місто не вдається знайти, користувачеві пропонується перевірити правильність написання або ввести іншу назву.

Основні аспекти оптимізації. У рамках цієї реалізації оптимізація продуктивності включає в себе:

- зменшення затримок у відповідях (швидкий HTTP-запит);
- обмеження кількості запитів до API;
- запобігання аварійному завершенню роботи бота у випадку виникнення винятків.

Застосування мови Python разом із бібліотекою requests забезпечує ефективну взаємодію з API, проте розробник повинен самостійно управляти винятками та незвичайними ситуаціями.

Обробка помилок. У коді реалізовано базову перевірку статусу HTTP-відповіді. Приклад наданий у додатку А.

Це гарантує початкову валідацію — перевіряється, чи API надало успішну відповідь. У разі іншого статусу, користувач отримує чітке повідомлення.

Також аналогічно обробляється відповідь на прогноз погоди. Приклад наданий у додатку А.

Чому це має значення. Без належної обробки помилок користувач може не усвідомлювати, що сталося. Наприклад:

- Неправильно введене місто → бот не реагує → користувач вважає, що він "зламаний".
- Проблеми з інтернет-з'єднанням → бот аварійно завершує роботу → необхідно перезапустити вручну.

Чим більше ситуацій оброблено, тим краще враження користувача та вища надійність системи.

Отже, у цьому розділі було проаналізовано основні та розширені методи обробки помилок під час взаємодії бота з OpenWeatherMap API, а також основи оптимізації функціонування чат-бота для підвищення його надійності та ефективності.

3.5 Впровадження інструментів у робочій процес

Після завершення етапу розробки окремих модулів Telegram-бота, важливим кроком є інтеграція інструментів у загальний робочий процес. Це передбачає об'єднання всіх компонентів — обробки повідомлень, отримання погодних даних, генерації відповідей — в єдину систему, що забезпечує зручну, безперебійну та стабільну взаємодію з користувачем.

Розробка бота здійснювалася поетапно: на початку була створена основна структура та реалізована команда /start. Після цього поступово впроваджувалися функції для отримання інформації про погоду, форматування повідомлень, перевірки введених даних, локалізації та інших вдосконалень.

Кожна зміна підлягала тестуванню відразу після її впровадження, що сприяло запобіганню накопичення помилок. Цей підхід відповідає принципам гнучкої розробки (agile), де кожна нова функція реалізується на окремому етапі та перевіряється незалежно.

Бот має модульну архітектуру, що полегшує його обслуговування та вдосконалення. Кожна функція виконує конкретну задачу, що дає змогу без труднощів модифікувати окремі компоненти коду, не загрожуючи стабільності інших частин програми.

Структура робочого процесу. Telegram-бот реалізовано у вигляді однопоточної події, що постійно чекає на повідомлення від користувача та виконує обробку відповідно до отриманого запиту. Процес роботи включає в себе кілька основних етапів (Рис 3.4):

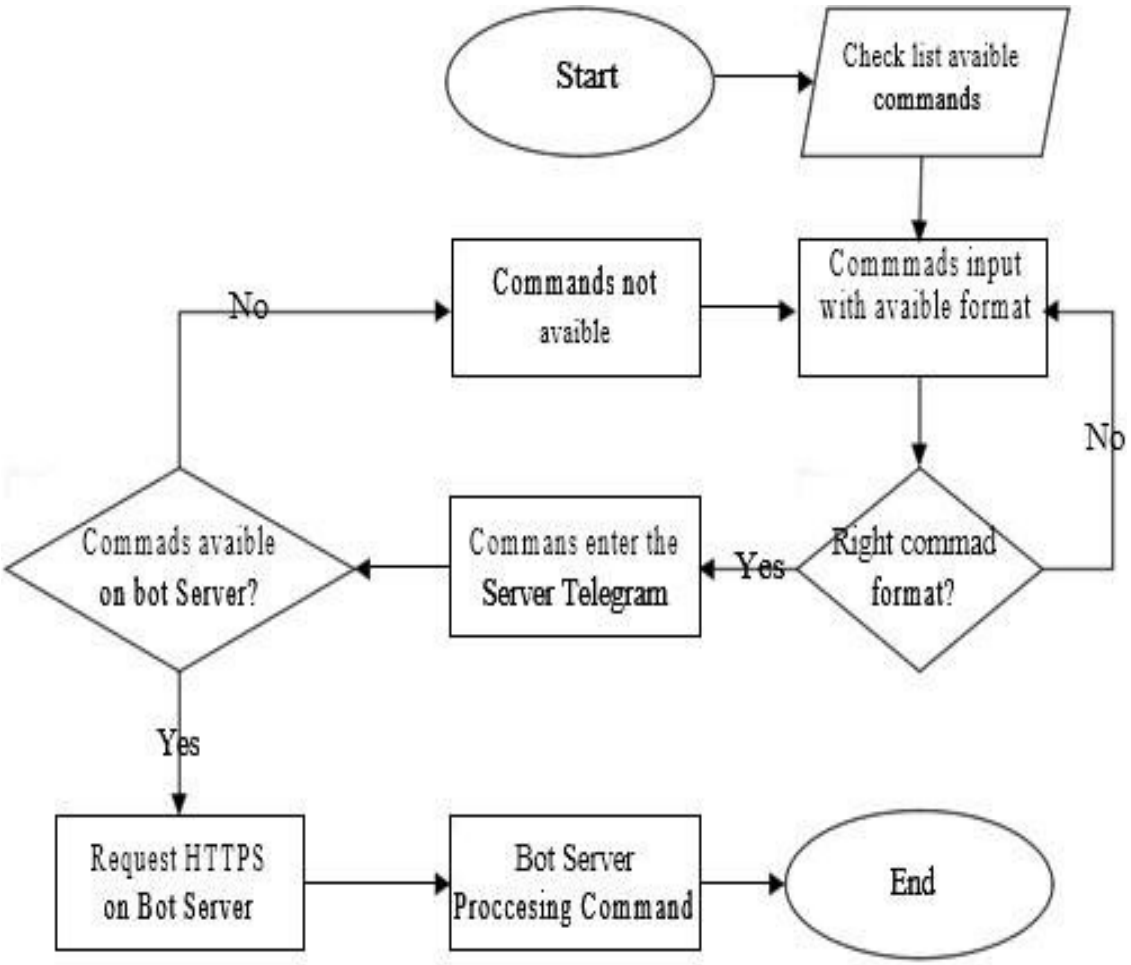


Рисунок 3.4 - Приклад роботи бота [34]

Пояснення:

1) Ініціалізація бота: Після активації скрипта відбувається виклик telebot.TeleBot(...) для з'єднання з API Telegram.

2) Очікування повідомлень: Завдяки @bot.message_handler бот відповідає на команду /start, а також на звичайні текстові повідомлення.

3) Збір та обробка метеорологічних даних: Коли користувач зазначає назву населеного пункту, бот виконує HTTP-запит до API OpenWeatherMap.

4) Формування відповіді: Отримані дані організовуються у структуроване повідомлення з використанням emoji та стилістики Markdown.

5) Надсилання результатів користувачу: Бот передає повідомлення з інформацією про погоду та прогнозом на наступні три дні.

6) Обробка запитів: Завдяки функції bot.polling(none_stop=True) бот продовжує свою діяльність у фоновому режимі, обробляючи нові запити.

Особливості реалізації. Застосування бібліотеки pyTelegramBotAPI (telebot) істотно полегшує робочий процес. Завдяки декораторам можливо чітко розділити функціональність. Приклад наданий у додатку А.

Ця структура забезпечує простоту в розширенні логіки — наприклад, у майбутньому можна буде додати нові команди, такі як /help, /about, або ж розширити обробку форматів даних.

Сценарій взаємодії користувача. З перспективи користувача, процес роботи з ботом виглядає наступним чином:

1) користувач надсилає команду /start і отримує вітальне повідомлення.
2) Далі він вводить назву міста (наприклад, Львів).
3) Бот надає інформацію про поточну погоду та короткий прогноз на три дні.
4) Користувач має можливість повторити запит для іншого міста, що продовжує цикл взаємодії.

Практичні переваги впровадження. Інтеграція всіх функцій в єдиний Telegram-бот має кілька переваг:

- Миттєвий доступ до інформації — користувачі можуть отримати дані про погоду за один клік.
- Уніфікований інтерфейс — взаємодія відбувається через знайомий інтерфейс Telegram.
- Мобільність — бот функціонує на будь-якому пристрої з Telegram (смартфон, ПК).
- Можливість масштабування — додавання нових функцій.

В таблиці 3.1 надані варіанти масштабування функціоналу

Таблиця 3.1— Варіанти масштабування функціоналу

Напрямок розширення	**Що можна додати**
Кнопки	Reply-клавіатура або inline
Геолокація	Запит на координати користувача
Зберігання історії запитів	База даних (SQLite, Firebase)
Інші погодні сервіси	WeatherAPI, AccuWeather

Отже, реалізація функціоналу в Telegram-боті сприяє створенню цілісного та зручного інструменту для отримання погодних даних. Процес роботи базується на telebot, що забезпечує просту архітектуру та гнучкість, а всі компоненти програми взаємодіють між собою синхронно та логічне. Це закладає основу для подальшого розвитку та розширення проєкту.

3.6 Інтеграція Telegram-бота в середовище розробника

Розробка Telegram-бота потребує створення зручного та ефективного середовища для написання, тестування та налагодження коду. У цьому проєкті було обрано PyCharm як основне середовище розробки — потужне інтегроване середовище (IDE) для Python, яке пропонує широкий спектр функцій для комфортної роботи над проєктом, включаючи підсвітку синтаксису, автозаповнення, інтеграцію з Git, віртуальні середовища, інструменти для налагодження тощо.

Вибір середовища для розробки. PyCharm — це потужне інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. PyCharm особливо корисний для великих проектів завдяки своїй інтелектуальній підтримці коду та інструментам рефакторингу. Ключові можливості: інтелектуальний редактор коду; вбудований дебагер; підтримка

віртуальних середовищ (virtualenv, pipenv); інтеграція з системами контролю версій (Git, SVN); підтримка популярних фреймворків (Django, Flask, Pyramid тощо); інструменти для роботи з базами даних; підтримка Jupyter Notebook

PyCharm було вибрано завдяки його інтуїтивно зрозумілому інтерфейсу, широкій підтримці функцій "з коробки" та потужним можливостям для програмістів. Крім того, важливим аспектом є те, що PyCharm автоматично розпізнає структуру проєкту, спрощує роботу з залежностями, дозволяє зручно управляти середовищами розробки (venv) та забезпечує комфортне середовище для виконання та тестування скриптів.

В таблиці 3.2 надані переваги використання PyCharm для розробки.

Таблиця 3.2— Переваги використання PyCharm для розробки

Можливість	**Опис**
Автоматичне форматування	Підсвітка синтаксису, автозавершення
Налагодження (debugging)	Точки зупинки, перегляд змінних
Інтеграція з Git	Контроль версій у візуальному режимі
Управління залежностями	Встановлення бібліотек через Python Packages
Віртуальне середовище	Робота з venv для ізоляції середовища

Налаштування проєкту в середовищі PyCharm.

Для початку роботи слід виконати наступні дії:

1) створити новий проєкт у PyCharm,
2) вибрати або створити нове віртуальне середовище (venv),
3) встановити необхідні бібліотеки через Terminal або інтерфейс "Python Packages".

Розробка бота в Telegram.

Для підключення Telegram-бота необхідно:

- зайти в Telegram і знайти @BotFather,
- створити нового бота за допомогою команди /newbot,
- отримати токен доступу

```
BOT_TOKEN = 'TOKEN'
API_KEY = 'API'
```

Ініціація та тестування. Запуск Telegram-бота в середовищі PyCharm виконується шляхом натискання кнопки Run або комбінації клавіш Shift + F10. PyCharm автоматично відкриває вбудований термінал, де відображаються повідомлення про помилки або логіка виконання. Бот функціонує у фоновому режимі до тих пір, поки його не зупинять вручну. Повідомлення від користувачів Telegram обробляються автоматично відповідно до логіки, закладеної в коді.

Функціональні можливості налагодження в PyCharm. PyCharm надає можливість поетапного налагодження коду за допомогою точок зупинки, візуалізації значень змінних та використання інтерактивної консолі. Це суттєво прискорює процес виявлення помилок та розробки.

Таким чином, інтеграція Telegram-бота в середовище PyCharm сприяла зручному та ефективному процесу розробки, що включає написання коду, тестування, управління залежностями та підключення зовнішніх API. Завдяки розширеним можливостям PyCharm, реалізація Telegram-бота стала більш структурованою, гнучкою та зрозумілою.

3.7 Тестування бота в реальних умовах користування

Тестування є ключовим етапом у процесі розробки будь-якого програмного забезпечення, включаючи Telegram-ботів. Цей процес дозволяє виявити логічні помилки, перевірити стабільність функціонування, оцінити зручність користувацького інтерфейсу та забезпечити відповідність функціоналу очікуванням. У рамках розробки чат-бота для отримання інформації про погоду тестування проводилося в умовах, максимально наближених до реальних, тобто

безпосередньо в середовищі Telegram, з використанням реальних API-запитів та реакцій користувачів.

Було проведено аналіз взаємодії з ботом за умов правильного та неправильного введення інформації. Тестувалися випадки з помилками в написанні назв міст, використанням неіснуючих назв, а також введенням спеціальних символів і цифр.

Усі зазначені ситуації обробляються ботом без помилок. Якщо виникає помилка, користувач отримує роз'яснення та рекомендацію повторити запит. Це допомагає уникнути розчарувань і робить користувацький досвід більш приємним.

Було проведено окреме тестування триденного прогнозу погоди. Перевірено коректність фільтрації значень на 12:00 та логіку вибору лише майбутніх днів. Результати підтвердили, що бот стабільно обробляє дані, надаючи лише актуальну інформацію.

Тестування включало міста з різних країн та різними мовами введення. Завдяки локалізації, бот надає відповіді українською мовою для всіх населених пунктів, які підтримуються сервісом OpenWeatherMap. Це робить його універсальним інструментом для отримання інформації про погоду.

Мета тестування. Метою тестування Telegram-бота було:

- перевірка точності отримання актуальної інформації про погоду та триденний прогноз;
- оцінка обробки некоректних даних (наприклад, помилок у назвах міст);
- аналіз швидкості реакції та стабільності функціонування бота;
- перевірка коректності відображення повідомлень і їх форматування в Telegram;

Сценарії тестування. Було створено декілька типових сценаріїв взаємодії з ботом:

- Успішний запит погоди: Користувач вводить правильну назву міста: Київ Бот повертає актуальну погоду та прогноз.

- Запит з помилкою: Введення неіснуючого міста: Київ Бот повертає повідомлення: "Немає такого міста□♂□, напиши ще раз□"
- Повторні запити: Користувач запитує погоду кілька разів поспіль. Бот не зависає, а стабільно повертає відповіді.
- Тестування форматування повідомлень: Перевірено правильність виводу емодзі, стилю Markdown, розділення прогнозу по днях.
- Мережеві збої: Перевірка, чи не падає бот, а коректне обробляє винятки (Рис.3.5)

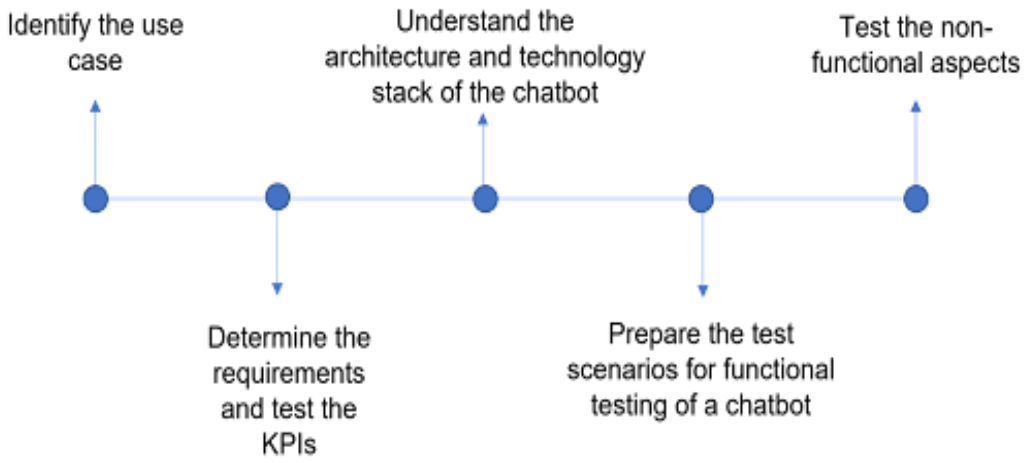


Рисунок 3.5 — Поетапний підхід до тестування чат-ботів [8]

Залучення тестових користувачів. Бот проходив тестування як автором, так і іншими користувачами з Telegram, які мали можливість взаємодіяти з ботом, перевіряти його функціональність та надавати зворотний зв'язок. Цей підхід дав змогу оцінити зручність інтерфейсу та ясність відповідей.

Виявлені недоліки та їх оптимізація. Під час тестування було виявлено кілька проблем:

- потреба в додатковій перевірці наявності поля "gust" (пориви вітру) в окремих відповідях API,
- виправлення граматичних помилок у відповідях бота.

Результати тестування. Тестування засвідчило, що бот:

- адекватно реагує на команди та введення користувачів;

- стабільно взаємодіє з OpenWeather API;
- пропонує зрозумілий інтерфейс для отримання інформації про погодні умови;
- демонструє достатню продуктивність для роботи з обмеженою кількістю одночасних користувачів.

Отже, проведення тестування Telegram-бота в умовах реального використання підтвердило ефективність розробленого рішення, виявило можливі проблеми, які були своєчасно усунуті, що дозволило впевнено перейти до впровадження бота в робоче середовище. Результати свідчать про готовність продукту до використання кінцевими споживачами.

Таким чином, в цьому розділі розглянуто практичні аспекти створення Telegram-чат-бота для отримання інформації про погоду з використанням мови програмування Python. Докладно описано технічні рішення, які забезпечують функціонал отримання актуальної погоди та прогнозу на кілька днів через API-сервіс OpenWeatherMap. Особлива увага приділяється форматуванню відповідей для користувача, що сприяє зручному сприйняттю даних і підвищує загальну ефективність взаємодії з ботом.

У рамках розділу було здійснено обробку введення користувача з валідацією назви міста, що сприяє зменшенню помилок під час запитів до API. Також були розглянуті методи оптимізації продуктивності та обробки помилок, які можуть виникнути в процесі роботи з мережею або через некоректні запити. Окремо було проаналізовано процес інтеграції Telegram-бота в середовище розробки PyCharm, а також тестування створеного рішення в умовах реального використання, що дало змогу виявити та усунути можливі недоліки перед запуском проєкту.

Внаслідок виконаної роботи було розроблено повноцінного Telegram-бота, який у реальному часі надає актуальну інформацію про погоду в зручному форматі, що свідчить про ефективність обраного технічного підходу та доцільність впровадження подібних рішень для спрощення доступу до інформаційних сервісів.

ВИСНОВКИ

У сучасному цифровому середовищі, де спостерігається швидкий розвиток комунікаційних платформ та зростання попиту на зручні сервіси, надзвичайно важливим є розробка ефективних інструментів для доступу до актуальної інформації. Одним із таких інструментів є Telegram-боти, які, завдяки інтеграції з зовнішніми API-сервісами, можуть забезпечити швидку взаємодію користувача з різними типами даних, зокрема з погодними сервісами.

У процесі виконання кваліфікаційної роботи було створено програмний продукт — Telegram-бот для отримання інформації про погоду, який надає користувачу актуальні дані про температуру, опис погодних умов, швидкість вітру та прогноз на кілька днів. Основою його функціонування є API-сервіс OpenWeatherMap, що забезпечує погодні дані в реальному часі, а також Telegram Bot API, який реалізує обробку запитів користувачів у месенджері.

Система, що була розроблена, має можливість автоматично обробляти текстові запити, формувати структуровані відповіді у зручному для сприйняття форматі, перевіряти правильність введених назв міст та адекватно реагувати на помилки. Важливу роль у реалізації відіграла локалізація інтерфейсу українською мовою, структурованість повідомлень та стабільність взаємодії з API, навіть за умов частих запитів.

Розробка та тестування проводились у середовищі PyCharm, що сприяло зручності налагодження та підтримки коду. Результати тестування продемонстрували, що бот ефективно виконує свої функції, швидко реагує на запити, коректно обробляє помилки та є зручним для кінцевого користувача.

Отже, розроблений Telegram-бот слугує зразком сучасного, компактного та ефективного засобу для отримання погодних даних, який може бути просто адаптований або розширений для інших аналогічних сервісів — від новин до нагадувань або інтелектуальних рекомендацій.

В результаті проведених досліджень :

1) Виконано аналіз ключових принципів функціонування Telegram-ботів, зокрема на основі Telegram Bot API, що дозволило сформувати уявлення про архітектуру та можливості таких рішень.

2) Досліджено специфіку роботи з API сервісів для отримання погодних даних, зокрема OpenWeatherMap API, що забезпечує інтеграцію актуальних погодних даних в інтерфейс бота.

3) Здійснено аналіз наявних бібліотек та рішень, що застосовуються для створення погодних ботів, а також виявлено їхні недоліки, що підкреслює необхідність розробки власного функціонального інструменту.

4) Розроблено Telegram-бота на мові Python, який надає інформацію про поточну погоду та прогноз на кілька днів, має зручне форматування відповідей, валідацію введення, обробку помилок та адаптацію до українськомовної аудиторії.

5) Впроваджено створений бот у середовище розробки PyCharm, проведено тестування функціональності в умовах реального використання, що підтвердило стабільність і доцільність даного рішення.

Результати дослідження. Забезпечено покращений доступ до погодних даних для користувачів Telegram шляхом розробки функціонального, адаптивного та локалізованого чат-бота, що використовує сучасні технології обміну інформацією через API.

Наукова новизна. Наукова новизна даного дослідження полягає в інтеграції технологій Telegram Bot API та OpenWeatherMap API для створення чат-бота, який може швидко надавати інформацію про погоду, враховуючи запити користувачів, їх локалізацію, а також застосовуючи алгоритми обробки помилок і оптимізації відповідей.

Практична значущість результатів дослідження - полягає у розробці прикладного програмного рішення, яке здатне функціонувати в реальних умовах для оперативного отримання погодних даних. Результати, що були отримані,

можуть слугувати основою для подальших розробок у галузі чат-ботів, інтеграції API-сервісів та автоматизації процесу інформування користувачів.

Апробація результатів. Апробація результатів надається в 1 тезисах в рецензованих наукових журналах і виданнях. Матеріали були опубліковані:

В тезисах:

1. Крицький Я.Ю. ІНТЕГРАЦІЯ ЧАТ-БОТ TELEGRAM З INTERNET OF THINGS // V Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез 15 квітня 2025. – К.: ДУІКТ, https://duikt.edu.ua/ua/news-1-570-12409-v-mizhnarodna-naukovo-tehnichna-konferenciya-suchasniy-stan-ta-perspektivi-rozvitku-iot_kafedra-inzhenerii-programnogo-zabezpechennya-avtomatizovanih-sistem

2. Крицький Я.Ю. Особливості розробки Telegram-бота для отримання погоди з позиції кібербезпеки// Всеукраїнська науково-практична конференція "Цифрова трансформація кібербезпеки" Збірник тез 24 квітня 2025. – К.: ДУІКТ, https://duikt.edu.ua/ua/news-1-574-14029-duikt-zaproshue-do-uchasti-u-vseukrainskiy-naukoviy-konferencii--cifrova-transformaciya-kiberbezpeki_kafedra-informaciynoi-ta-kibernetichnoi-bezpeki

ПЕРЕЛІК ПОСИЛАНЬ

1. Telegram Bot API Documentation [Електронний ресурс] / Режим доступу: https://core.telegram.org/bots/api (дата звернення: 02.04.2025).
2. OpenWeatherMap API Documentation [Електронний ресурс] / Режим доступу: https://openweathermap.org/api (дата звернення: 02.04.2025).
3. Python Requests Library Documentation [Електронний ресурс] / Режим доступу: https://requests.readthedocs.io (дата звернення: 02.04.2025).
4. pyTelegramBotAPI Documentation [Електронний ресурс] / Режим доступу: https://pytba.readthedocs.io (дата звернення: 02.04.2025).
5. OpenWeatherMap Python Tutorial [Електронний ресурс] / Режим доступу: https://openweathermap.org/current#python (дата звернення: 04.04.2025).
6. OpenWeatherMap API Response Parameters [Електронний ресурс] / Режим доступу: https://openweathermap.org/weather-data (дата звернення: 02.04.2025).
7. OpenWeatherMap API Usage Guide [Електронний ресурс] / Режим доступу: https://docs.openweather.co.uk/appid (дата звернення: 02.04.2025).
8. Chatbot Testing Best Practices [Електронний ресурс] / Режим доступу: https://www.browserstack.com/guide/what-is-chatbot-testing (дата звернення: 02.04.2025).
9. Deploying Telegram Bots to Heroku [Електронний ресурс] / Режим доступу: https://dev.to/dominickoech/deploying-python-telegram-bot-on-heroku-without-using-flask-kmm (дата звернення: 02.04.2025).
10. Using Markdown in Telegram Bots [Електронний ресурс] / Режим доступу: https://core.telegram.org/bots/api#formatting-options (дата звернення: 05.04.2025).
11. Telegram APIs Overview [Електронний ресурс] / Режим доступу: https://core.telegram.org/ (дата звернення: 02.04.2025).

12. json — JSON encoder and decoder [Електронний ресурс] / Режим доступу: https://docs.python.org/3/library/json.html (дата звернення: 04.04.2025).

13. datetime — Basic date and time types [Електронний ресурс] / Режим доступу: https://docs.python.org/3/library/datetime.html (дата звернення: 04.04.2025).

14. telebot (pyTelegramBotAPI) documentation [Електронний ресурс] / Режим доступу: https://pypi.org/project/pyTelegramBotAPI/ (дата звернення: 04.04.2025).

15. pyTelegramBotAPI GitHub Repository [Електронний ресурс] / Режим доступу: https://github.com/eternnoir/pyTelegramBotAPI (дата звернення: 02.04.2025).

16. Python and API Integration [Електронний ресурс] / Режим доступу: https://realpython.com/api-integration-in-python/ (дата звернення: 07.04.2025).

17. Input Validation and Error Handling in Python [Електронний ресурс] / Режим доступу: https://p-kane.medium.com/input-validation-with-python-570953d5d297 (дата звернення: 02.04.2025).

18. PyCharm Documentation [Електронний ресурс] / Режим доступу: https://www.jetbrains.com/pycharm/documentation/ (дата звернення: 08.04.2025).

19. Python Clean Code Practices [Електронний ресурс] / Режим доступу: https://dev.to/devasservice/python-best-practices-writing-clean-efficient-and-maintainable-code-34bj (дата звернення: 02.04.2025).

20. Telegram Bot API GitHub Repository [Електронний ресурс] / Режим доступу: https://github.com/tdlib/telegram-bot-api (дата звернення: 02.04.2025).

21. Example Weather Bot №1 [Електронний ресурс] / Режим доступу: https://github.com/lesskop/shtosh-weather-bot (дата звернення: 02.04.2025).

22. Example Weather Bot №2 [Електронний ресурс] / Режим доступу: https://github.com/karvozavr/weather-bot (дата звернення: 02.04.2025).

23. Example Weather Bot №3 [Електронний ресурс] / Режим доступу: https://github.com/mustafababil/Telegram-Weather-Bot (дата звернення: 02.04.2025).

24. Модульність і структура коду в Python [Електронний ресурс] / Режим доступу: https://docs.python.org/3/tutorial/modules.html (дата звернення: 02.04.2025).

25. Python Telegram Bot Tutorial: Як створити Telegram-бота [Електронний ресурс] / Режим доступу: https://www.geeksforgeeks.org/create-a-telegram-bot-using-python/ (дата звернення: 02.04.2025).

26. How to Parse JSON in Python [Електронний ресурс] / Режим доступу: https://realpython.com/python-json/ (дата звернення: 02.04.2025).

27. Приклади використання requests у Python [Електронний ресурс] / Режим доступу: https://realpython.com/python-requests/ (дата звернення: 02.04.2025).

28. Hosting Telegram Bot on Heroku Guide [Електронний ресурс] / Режим доступу: https://grammy.dev/hosting/heroku (дата звернення: 02.04.2025).

29. Python Requests Module on PyPI [Електронний ресурс] / Режим доступу: https://pypi.org/project/requests/ (дата звернення: 02.04.2025).

30. datetime on PyPI [Електронний ресурс] / Режим доступу: https://pypi.org/project/DateTime/ (дата звернення: 02.04.2025).

31. How Telegram Bots work [Електронний ресурс] / Режим доступу: https://rdiachenko.com/posts/bots/telegram/how-do-telegram-bots-work/ (дата звернення: 02.04.2025).

32. 10 Python Libraries Every Data Scientist Should Know [Електронний ресурс] / Режим доступу: https://www.kdnuggets.com/10-python-libraries-every-data-scientist-should-know (дата звернення: 02.04.2025).

33. REST API Basics - 4 Things you Need to Know [Електронний ресурс] / Режим доступу: https://mannhowie.com/rest-api (дата звернення: 02.04.2025).

34. Proses Request Telegram Bot API [Електронний ресурс] / Режим доступу: https://www.researchgate.net/figure/Proses-Request-Telegram-Bot-API_fig2_330960697 (дата звернення: 02.04.2025).

35. Читання JSON у Python [Електронний ресурс] / Режим доступу: https://www.w3schools.com/python/python_json.asp (дата звернення: 02.04.2025).

36. Як використовувати datetime у Python [Електронний ресурс] / Режим доступу: https://www.w3schools.com/python/python_datetime.asp (дата звернення: 02.04.2025).

37. Python: використання бібліотеки requests для HTTP-запитів [Електронний ресурс] / Режим доступу: https://docs.python-requests.org/en/latest/user/quickstart/ (дата звернення: 02.04.2025).

38. Як форматувати дату у Python [Електронний ресурс] / Режим доступу: https://strftime.org/ (дата звернення: 02.04.2025).

39. Рекомендації по структуруванню коду Python [Електронний ресурс] / Режим доступу: https://www.python.org/dev/peps/pep-0008/ (дата звернення: 02.04.2025).

40. Обробка запитів API у Python [Електронний ресурс] / Режим доступу: https://www.dataquest.io/blog/python-api-tutorial/ (дата звернення: 02.04.2025).

ДОДАТОК А

Приклади кодів

Приклад обробки запитів від користувачів:

```
res=requests.get(f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API}&units=metric&lang=ua')
    data = json.loads(res.text)
```

Приклад форматування відповідей:

```
    weather_description = data['weather'][0]['description']
    temp = data['main']['temp']
    feels_like = data['main']['feels_like']
    wind_speed = data['wind']['speed']
    weather_info = (
      f'Погода: {weather_description}\n'
      f'Температура: {temp} °C\n'
      f'Відчувається як: {feels_like} °C\n'
      f'Швидкість вітру: {wind_speed} м/с\n'
bot.send_message(message.chat.id, weather_info, parse_mode='Markdown')
```

Приклад обробки виключень:

```
if res.status_code == 200:
…
else:
  bot.reply to(message, 'Немає такого міста, напиши ще раз')
```

Приклад формування запитів до OpenWeatherMap:

Для поточної погоди:

https://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={API_key}&units=metric&lang=ua

Для прогнозу погоди:

https://api.openweathermap.org/data/2.5/forecast?q={city_name}&appid={API_key}&units=metric&lang=ua

Приклад обробки даних та їх форматування:

Для поточної погоди:

```
def get_weather(message):
    city = message.text.strip().lower()
res=requests.get(f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API}&units=metric&lang=ua')
    if res.status_code == 200:
        data = json.loads(res.text)
        weather_description = data['weather'][0]['description']
        temp = data['main']['temp']
        feels_like = data['main']['feels_like']
        wind_speed = data['wind']['speed']
        weather_info = (
            f'Погода: {weather_description}\n'
            f'Температура: {temp} °C\n'
            f'Відчувається як: {feels_like} °C\n'
            f'Швидкість вітру: {wind_speed} м/с\n'
        )
 bot.send_message(message.chat.id, weather_info, parse_mode='Markdown')
    else:
        bot.reply_to(message, 'Немає такого міста, напиши ще раз')
```

Для прогнозу погоди:

```
 if forecast_res.status_code == 200:
            forecast_data = json.loads(forecast_res.text)
```

```
        forecast_info = '\n□ *Прогноз на 3 дні:*\n'
        days_translate = {
            'Monday': 'Понеділок',
            'Tuesday': 'Вівторок',
            'Wednesday': 'Середа',
            'Thursday': 'Четвер',
            'Friday': 'П’ятниця',
            'Saturday': 'Субота',
            'Sunday': 'Неділя'
        }
        shown_days = set()
        for item in forecast_data['list']:
            dt = datetime.strptime(item['dt_txt'], '%Y-%m-%d %H:%M:%S')
            today = datetime.now().date()
            if dt.hour == 12 and dt.date() > today:
                day_name = dt.strftime('%A')
                day_name_ua = days_translate.get(day_name, day_name)
                day = f'{day_name_ua} {dt.strftime("%d.%m")}'
                if day not in shown_days and len(shown_days) < 3:
                    temp = item['main']['temp']
                    description = item['weather'][0]['description']
                    forecast_info += f'□ *{day}* — {description}, □ {temp} °C\n'
                    shown_days.add(day)
        weather_info += forecast_info
    weather_info += '\n\n□ Напиши назву міста, щоб дізнатись нову погоду □'
  bot.send_message(message.chat.id, weather_info, parse_mode='Markdown')
else:
  bot.reply_to(message, 'Немає такого міста□♂□, напиши ще раз□')
```

Приклад використання Markdown у Telegram:

bot.send_message(message.chat.id,weather_info,parse_mode='Markdown')

Приклад форматування повідомлень про поточну погоду:

```
weather_info = (
    f'Погода: {weather_description}\n'
    f'Температура: {temp} °C\n'
    f'Відчувається як: {feels_like} °C\n'
    f'Швидкість вітру: {wind_speed} м/с\n'  )
```

Приклад форматування прогнозу на декілька днів:

```
 shown_days = set()
    for item in forecast_data['list']:
       dt = datetime.strptime(item['dt_txt'], '%Y-%m-%d %H:%M:%S')
       today = datetime.now().date()
       if dt.hour == 12 and dt.date() > today:
          day_name = dt.strftime('%A')
          day_name_ua = days_translate.get(day_name, day_name)
          day = f'{day_name_ua} {dt.strftime("%d.%m")}'
          if day not in shown_days and len(shown_days) < 3:
             temp = item['main']['temp']
             description = item['weather'][0]['description']
             forecast_info += f' *{day}* — {description},  {temp} °C\n'
             shown_days.add(day)
```

Приклад логіки реалізації в боті:

```
 def get_weather(message):
    city = message.text.strip().lower()
    res=requests.get(f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API}&units=metric&lang=ua')
     if res.status_code == 200:
```

...

else:

bot.reply_to(message, 'Немає такого міста□♂□, напиши ще раз□')

Приклад обробки помилок:

Для поточної погоди:

res=requests.get(f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API}&units=metric&lang=ua')

if res.status_code == 200:

...

else:

bot.reply_to(message, 'Немає такого міста□♂□, напиши ще раз□')

Для прогнозу погоди:

forecast_res=requests.get(f'https://api.openweathermap.org/data/2.5/forecast?q={city}&appid={API}&units=metric&lang=ua')

if forecast_res.status_code == 200:

…

else:

bot.reply_to(message, 'Немає такого міста□♂□, напиши ще раз□')

Приклад особливості реалізації:

def start(message):

bot.send_message(message.chat.id, 'Привіт□, напиши назву міста□')

Приклади не оптимізованого коду надані на рис. А.1-А.4:

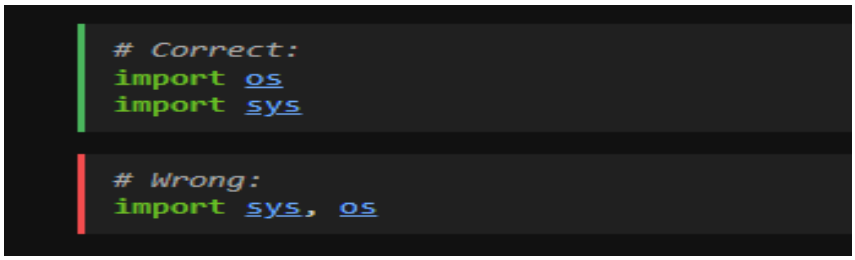


Рисунок А.1- Приклад не оптимізованого коду №1 [39]

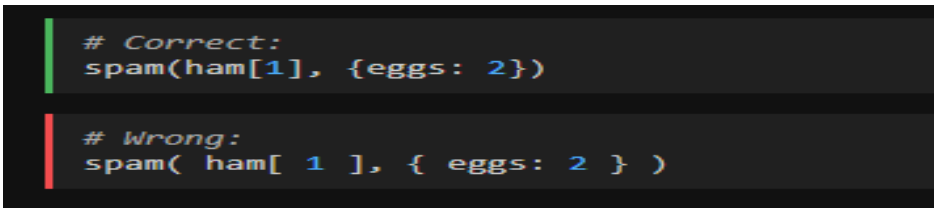


Рисунок А.2-Приклад не оптимізованого коду №2 [39]

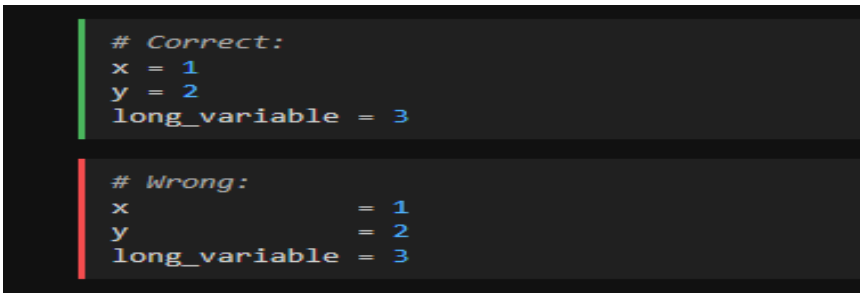


Рисунок А.3-Приклад не оптимізованого коду №3 [39]

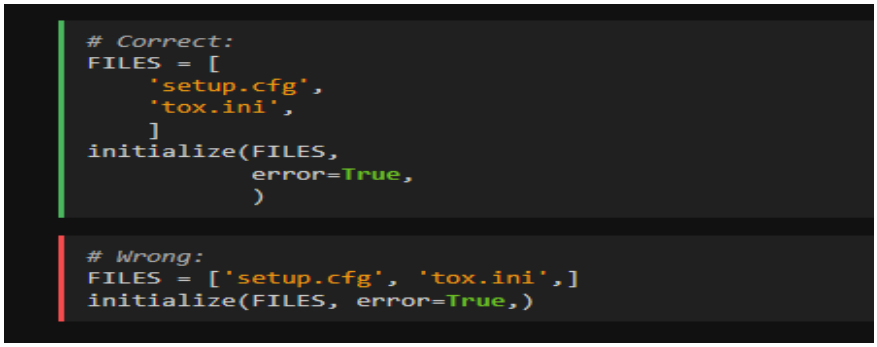


Рисунок А.4-Приклад не оптимізованого коду №4 [39]

Формати часу та дати, які використовуються (Табл.А1):

Таблиця А1 - Формати часу та дати

Формат	**Приклад**	**Використання**
%Y-%m-%d %H:%M:%S	2024-04-18 12:00:00	Формат у відповіді forecast API
%A	Monday	Для визначення назви дня
dt.strftime('%d.%m')	18.04	Для компактного відображення дати
Переклад днів	Monday → Понеділок	Локалізація

ДОДАТОК Б

Програмний код

```
import telebot

import requests

import json

from datetime import datetime


bot = telebot.TeleBot('7729951162:AAHPiY8lnoK52LyzVbRl1NVcLzKoauX_CzA')

API = 'dbaf44e0b897778c80d33f94bea8c36e'


@bot.message_handler(commands=['start'])

def start(message):

    bot.send_message(message.chat.id, 'Привіт, напиши назву міста')


@bot.message_handler(content_types=['text'])

def get_weather(message):

    city = message.text.strip().lower()

    res =
requests.get(f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid
={API}&units=metric&lang=ua')

    if res.status_code == 200:

       data = json.loads(res.text)

       weather_description = data['weather'][0]['description']

       temp = data['main']['temp']

       feels_like = data['main']['feels_like']

       wind_speed = data['wind']['speed']

       weather_info = (
```

```
            f'Погода: {weather_description}\n'
            f'Температура: {temp} °C\n'
            f'Відчувається як: {feels_like} °C\n'
            f'Щвидкість вітру: {wind_speed} м/с\n'
        )
        if 'gust' in data['wind']:
            wind_gust = data['wind']['gust']
            weather_info += f"Порив вітру: {wind_gust} м/с"


        forecast_res =
requests.get(f'https://api.openweathermap.org/data/2.5/forecast?q={city}&appi
d={API}&units=metric&lang=ua')

        if forecast_res.status_code == 200:
            forecast_data = json.loads(forecast_res.text)
            forecast_info = '\n *Прогноз на 3 дні:*\n'
            days_translate = {
                'Monday': 'Понеділок',
                'Tuesday': 'Вівторок',
                'Wednesday': 'Середа',
                'Thursday': 'Четвер',
                'Friday': 'П’ятниця',
                'Saturday': 'Субота',
                'Sunday': 'Неділя'
            }
```

```
            shown_days = set()
            for item in forecast_data['list']:
                dt = datetime.strptime(item['dt_txt'], '%Y-%m-%d %H:%M:%S')
                today = datetime.now().date()

                if dt.hour == 12 and dt.date() > today:
                    day_name = dt.strftime('%A')
                    day_name_ua = days_translate.get(day_name, day_name)
                    day = f'{day_name_ua} {dt.strftime("%d.%m")}'
                    if day not in shown_days and len(shown_days) < 3:
                        temp = item['main']['temp']
                        description = item['weather'][0]['description']
                        forecast_info += f'□ *{day}* — {description}, □ {temp}
°C\n'
                        shown_days.add(day)

            weather_info += forecast_info
            weather_info += '\n\n□ Напиши назву міста, щоб дізнатись нову
погоду □'

        bot.send_message(message.chat.id, weather_info, parse_mode='Markdown')
    else:
        bot.reply_to(message, 'Немає такого міста□♂□, напиши ще раз□')

bot.polling(none_stop=True)
```

ДОДАТОК В

Приклади взаємодії з ботом (Рис.В.1)

Рисунок В.1- Приклади взаємодії з ботом

ДОДАТОК Г

Приклади відповіді API з погодними даними

Для поточної погоди:

{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":803,"main":"Clouds","description":"рвані хмари","icon":"04d"}],"base":"stations","main":{"temp":11.74,"feels_like":10.61,"temp_min":10.27,"temp_max":12.77,"pressure":1003,"humidity":63,"sea_level":1003,"grnd_level":999},"visibility":10000,"wind":{"speed":7.2,"deg":220,"gust":13.38},"clouds":{"all":75},"dt":1744795280,"sys":{"type":2,"id":2091269,"country":"GB","sunrise":1744779734,"sunset":1744829873},"timezone":3600,"id":2643743,"name":"London","cod":200}

Для прогнозу погоди:

{"cod":"200","message":0,"cnt":40,"list":[{"dt":1744804800,"main":{"temp":12.12,"feels_like":10.85,"temp_min":12.12,"temp_max":13.13,"pressure":1004,"sea_level":1004,"grnd_level":1001,"humidity":56,"temp_kf":-1.01},"weather":[{"id":803,"main":"Clouds","description":"рвані хмари","icon":"04d"}],"clouds":{"all":59},"wind":{"speed":7.44,"deg":217,"gust":11.08},"visibility":10000,"pop":0,"sys":{"pod":"d"},"dt_txt":"2025-04-16 12:00:00"},{"dt":1744815600,"main":{"temp":12.74,"feels_like":11.3,"temp_min":12.74,"temp_max":13.31,"pressure":1006,"sea_level":1006,"grnd_level":1002,"humidity":47,"temp_kf":-0.57},"weather":[{"id":802,"main":"Clouds","description":"уривчасті хмари","icon":"03d"}],"clouds":{"all":30},"wind":{"speed":7,"deg":216,"gust":10.08},"visibility":10000,"pop":0,"sys":{"pod":"d"},"dt_txt":"2025-04-16 15:00:00"},{"dt":1744826400,"main":{"temp":11.31,"feels_like":9.91,"temp_min":11.31,"temp_max":11.31,"pressure":1008,"sea_level":1008,"grnd_level":1004,"humidity":54,"temp_kf":0},"weather":[{"id":800,"main":"Clear","description":"чисте небо","icon":"01d"}],"clouds":{"all":9},"wind":{"speed":4.81,"deg":211,"gust":9.13},"visibility":10000,"pop":0,"sys":{"pod":"d"},"dt_txt":"2025-04-16 18:00:00"},{"dt":1744837200,"main":{"temp":8.13,"feels_like":6.52,"temp_min":8.13,"temp_max":8.13,"pressure":1011,"sea_level":1011,"grnd_level":1006,"humidity":65,"temp_kf":0},"weather":[{"id":800,"main":"Clear","description":"чисте небо","icon":"01n"}],"clouds":{"all":1},"wind":{"speed":2.62,"deg":207,"gust":7.49},"visibility":10000,"pop":0,"sys":{"pod":"n"},"dt_txt":"2025-04-16 21:00:00"},{"dt":1744848000,"main":{"temp":6.58,"feels_like":5.71,"temp_min":6.58,"temp_max":6.58,"pressure":1012,"sea_level":1012,"grnd_level":1008,"humidity":74,"temp_kf":0},"weather":[{"id":802,"main":"Clouds","description":"уривчасті хмари","icon":"03n"}],"clouds":{"all":35},"wind":{"speed":1.53,"deg":231,"gust":4.73

},"visibility":10000,"pop":0,"sys":{"pod":"n"},"dt_txt":"2025-04-17
00:00:00"},{"dt":1744858800,"main":{"temp":6.87,"feels_like":6.87,"temp_min":6.87,
"temp_max":6.87,"pressure":1013,"sea_level":1013,"grnd_level":1008,"humidity":78,"t
emp_kf":0},"weather":[{"id":800,"main":"Clear","description":"чисте
небо","icon":"01n"}],"clouds":{"all":1},"wind":{"speed":1.24,"deg":238,"gust":1.93},"
visibility":10000,"pop":0,"sys":{"pod":"n"},"dt_txt":"2025-04-17
03:00:00"},{"dt":1744869600,"main":{"temp":7.03,"feels_like":7.03,"temp_min":7.03,
"temp_max":7.03,"pressure":1014,"sea_level":1014,"grnd_level":1009,"humidity":77,"t
emp_kf":0},"weather":[{"id":800,"main":"Clear","description":"чисте
небо","icon":"01d"}],"clouds":{"all":0},"wind":{"speed":0.52,"deg":270,"gust":0.78},"
visibility":10000,"pop":0,"sys":{"pod":"d"},"dt_txt":"2025-04-17
06:00:00"},{"dt":1744880400,"main":{"temp":12.08,"feels_like":10.75,"temp_min":12.
08,"temp_max":12.08,"pressure":1015,"sea_level":1015,"grnd_level":1010,"humidity":
54,"temp_kf":0},"weather":[{"id":800,"main":"Clear","description":"чисте
небо","icon":"01d"}],"clouds":{"all":4},"wind":{"speed":0.44,"deg":133,"gust":1.02},"
visibility":10000,"pop":0,"sys":{"pod":"d"},"dt_txt":"2025-04-17
09:00:00"},{"dt":1744891200,"main":{"temp":16.02,"feels_like":14.67,"temp_min":16.
02,"temp_max":16.02,"pressure":1014,"sea_level":1014,"grnd_level":1009,"humidity":
38,"temp_kf":0},"weather":[{"id":800,"main":"Clear","description":"чисте
небо","icon":"01d"}],"clouds":{"all":4},"wind":{"speed":1.52,"deg":161,"gust":2.88},"
visibility":10000,"pop":0,"sys":{"pod":"d"},"dt_txt":"2025-04-17
12:00:00"},{"dt":1744902000,"main":{"temp":16.47,"feels_like":15.09,"temp_min":16.
47,"temp_max":16.47,"pressure":1013,"sea_level":1013,"grnd_level":1008,"humidity":
35,"temp_kf":0},"weather":[{"id":803,"main":"Clouds","description":"рвані
хмари","icon":"04d"}],"clouds":{"all":67},"wind":{"speed":3.66,"deg":143,"gust":3.4}
,"visibility":10000,"pop":0,"sys":{"pod":"d"},"dt_txt":"2025-04-17
15:00:00"},{"dt":1744912800,"main":{"temp":14.53,"feels_like":13.27,"temp_min":14.
53,"temp_max":14.53,"pressure":1013,"sea_level":1013,"grnd_level":1009,"humidity":
47,"temp_kf":0},"weather":[{"id":804,"main":"Clouds","description":"хмарно","icon":
"04d"}],"clouds":{"all":85},"wind":{"speed":3.11,"deg":166,"gust":4.18},"visibility":1
0000,"pop":0,"sys":{"pod":"d"},"dt_txt":"2025-04-17
18:00:00"},{"dt":1744923600,"main":{"temp":10.58,"feels_like":9.39,"temp_min":10.5
8,"temp_max":10.58,"pressure":1015,"sea_level":1015,"grnd_level":1010,"humidity":6
5,"temp_kf":0},"weather":[{"id":802,"main":"Clouds","description":"уривчасті
хмари","icon":"03n"}],"clouds":{"all":37},"wind":{"speed":2.65,"deg":140,"gust":7.16
},"visibility":10000,"pop":0,"sys":{"pod":"n"},"dt_txt":"2025-04-17 21:00:00"}

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (презентація)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Державний університет інформаційно-комунікаційних технологій

ДИПЛОМНА РОБОТА
на ступінь вищої роботи бакалавр
із спеціальності 122 Комп'ютерні науки

«Чат-бот Telegram на мові програмування Python для отримання погоди»

Виконав: студент 4 курсу, групи КНД-41
Крицький Ярослав Юрійович
Керівник: д.т.н., професор, Катков Ю.І.

ЗАГАЛЬНА ХАРАКТЕРИСТИКА ДИПЛОМНОЇ РОБОТИ

- **Тема:**

Чат-бот Telegram на мові програмування Python для отримання погоди

- **Мета роботи:**

Підвищення ефективності отримання користувачами актуальної інформації про погодні умови та прогнози на найближчі дні в інтерактивному форматі.

- **Наукове завдання:**

Оцінка ефективності розроблених інструментів для програмної реалізації чат-бота Telegram, який забезпечує доступ до погодних даних в інтерактивному форматі, використовуючи мову програмування Python та API OpenWeatherMap.

- **Об'єкт дослідження:**

Процес забезпечення взаємодії користувача з чат-ботом Telegram для отримання інформації про погоду.

- **Предмет дослідження:**

Інструменти програмної реалізації чат-бота Telegram, який забезпечує доступ до погодних даних в інтерактивному форматі, використовуючи мову програмування Python та API OpenWeatherMap.

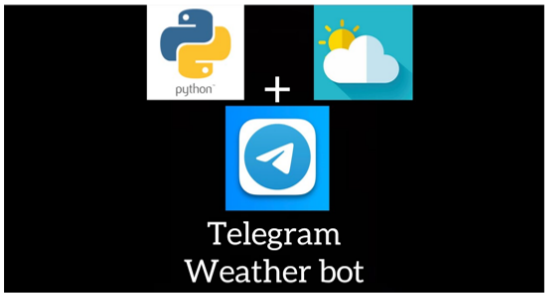
АКТУАЛЬНІСТЬ ТЕМИ

У сучасному світі, де інформаційні технології розвиваються з неймовірною швидкістю, месенджери стали важливою складовою повсякденного життя мільйонів людей по всьому світу. Одним із найбільш популярних засобів комунікації є Telegram — універсальний месенджер, який, крім традиційного обміну повідомленнями, пропонує розробникам можливість створювати чат-ботів. Завдяки мові програмування Python створення функціонального чат-бота для отримання погоди є актуальним завданням.

ПОСТАНОВКА ЗАВДАННЯ

У межах роботи було поставлено наступні завдання:

•Реалізувати Telegram-бота з використанням Python.
•Підключити OpenWeatherMap API та отримувати дані.
•Виводити поточну погоду та прогноз на 3 дні.
•Форматувати відповіді з використанням емодзі та Markdown.
•Забезпечити перевірку введення та обробку помилок.
•Реалізувати безперервну роботу через long polling.
•Провести тестування в реальних умовах.



ІНСТРУМЕНТИ І СЕРЕДОВИЩЕ РОЗРОБКИ

У рамках розробки Telegram-бота для отримання погодної інформації були використані сучасні інструменти та технології:

- **Python 3.10+** – високорівнева мова програмування, що забезпечує швидку розробку, зручний синтаксис і широкий набір бібліотек. Python ідеально підходить для створення Telegram-ботів завдяки підтримці асинхронної роботи та легкості інтеграції з API.
- **PyCharm** – інтегроване середовище розробки (IDE), яке дозволило зручно працювати з кодом, налагоджувати логіку, автоматично форматувати код та керувати залежностями через віртуальне середовище.
- **Бібліотека telebot (PyTelegramBotAPI)** – надає всі необхідні інструменти для обробки повідомлень Telegram-користувачів, створення команд, надсилання відповідей і запуску бота в режимі long polling.
- **requests** – бібліотека для виконання HTTP-запитів, що використовується для взаємодії з OpenWeatherMap API.
- **json** – для обробки структурованої відповіді від API.
- **datetime** – для вибору прогнозів на певну годину (12:00) і перетворення дати у зручний для користувача формат.

Усі ці інструменти створили гнучку, масштабовану та зрозумілу архітектуру бота.

СХЕМА ВЗАЄМОДІЇ КОРИСТУВАЧА З БОТОМ

Алгоритм взаємодії Telegram-бота з користувачем побудований таким чином, щоб забезпечити простоту та швидкість відповіді. Основні етапи виглядають наступним чином:

1) Користувач запускає бота за допомогою команди /start, після чого вводить назву міста, для якого хоче дізнатися погоду.
2) Бот зчитує повідомлення, очищує його від зайвих пробілів, приводить до нижнього регістру, щоб уникнути помилок при формуванні запиту.
3) Формується GET-запит до OpenWeatherMap API, де передаються параметри: назва міста, API-ключ, формат відповіді (JSON), мова (lang=ua), одиниці виміру (units=metric).
4) Відповідь API містить повну інформацію про поточну погоду та прогноз на 5 днів.
5) Бот розбирає відповідь, витягує з неї температуру, опис, вітер, а також відбирає прогноз на наступні три дні саме на 12:00, щоб показати найбільш репрезентативну температуру.
6) Формується повідомлення з емодзі та Markdown-форматуванням, яке відправляється користувачу.
7) Після відповіді бот пропонує ввести інше місто, підтримуючи безперервну взаємодію.

ОСНОВНИЙ БЛОК КОДУ ЗАПИТУ ДО API

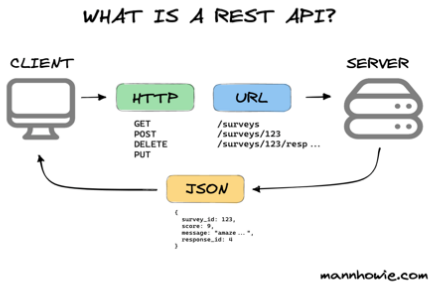
У коді реалізована чітка послідовність дій для надсилання запиту до OpenWeatherMap API:

- Змінна city отримує назву міста з повідомлення користувача.
- Використовуючи requests.get(), формується повний URL з параметрами запиту.
- Якщо відповідь від API успішна (код 200), з допомогою json.loads() вона перетворюється у Python-словник.

Далі з цього словника отримуються потрібні значення:

- data['weather'][0]['description'] – опис погоди;
- data['main']['temp'] – температура;
- data['main']['feels_like'] – температура, що відчувається;
- data['wind']['speed'] – швидкість вітру;
- Опціонально: data['wind']['gust'], якщо є вітрові пориви.

Цей фрагмент коду є основою для формування всіх погодних відповідей бота.



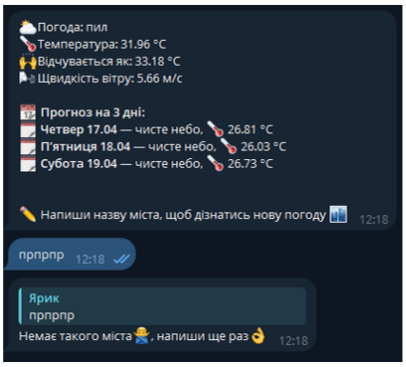
ФОРМАТ ВІДПОВІДІ ТА ПРИКЛАД ВИВОДУ

Telegram-бот надсилає користувачеві детальну відповідь у зручному, компактному і візуально привабливому форматі.

ОБРОБКА ПОМИЛОК ТА ВАЛІДАЦІЯ

У бота реалізовано кілька рівнів валідації та захисту від помилок, зокрема:

- Якщо користувач вводить некоректну назву міста, бот перевіряє статус відповіді API (status_code) і надсилає відповідь: «Немає такого міста🤦‍♂️, напиши ще раз👌»
- Якщо у відповіді відсутнє поле gust, бот не викликає помилку, а просто не виводить цей рядок.
- У випадку тимчасових проблем із сервером OpenWeatherMap, користувач не бачить коду помилки — йому приходить стандартне повідомлення з проханням спробувати знову.
- Для зручності реалізовано перевірку та обробку пробілів, зміни регістру, уникнення порожніх або числових запитів.



Це дозволяє зробити взаємодію максимально стабільною.

ОТРИМАНІ РЕЗУЛЬТАТИ

В результаті реалізації дипломного проєкту отримано такі практичні результати:

- Розроблено робочого Telegram-бота, який надає інформацію про поточну погоду та прогноз.
- Інтерфейс бота зрозумілий, інтуїтивний, з використанням емодзі, форматування та простих повідомлень.
- Реалізовано підтримку трьох днів прогнозу, що підвищує користувацьку цінність.
- Валідація введення, перевірка статусів API, обробка винятків — усе працює без збоїв.

СКРІНШОТ ТА ПРИКЛАД ВИКОРИСТАННЯ

Цей слайд ілюструє реальні приклади використання Telegram-бота:

- Стартове повідомлення після /start
- Запит погоди для міста: Київ, Дубай
- Форматування відповіді
- Повідомлення про помилку, якщо місто не знайдено
- Повторний цикл взаємодії — без необхідності натискати кнопки

Цей скріншот демонструє, що бот готовий до використання користувачами без технічної підготовки.

ВИСНОВКИ ДО РОБОТИ

В рамках дипломної роботи:

- Розроблено функціональний Telegram-бот на Python для отримання погоди.
- Проведено аналіз API-сервісів, обґрунтовано вибір OpenWeatherMap.
- Реалізовано логіку взаємодії з користувачем, прогнозування, обробку помилок.
- Створено універсальну структуру, яка дозволяє легко масштабувати проєкт.
- Telegram-бот продемонстрував стабільну, швидку і коректну роботу.

Цей проєкт підтверджує, що прості рішення можуть значно покращити повсякденне користування цифровими сервісами.

ПЕРСПЕКТИВИ РОЗВИТКУ

У майбутньому проєкт має потенціал розвитку у таких напрямах:

- Додавання геолокації (користувач надсилає свою позицію – бот сам визначає місто).
- Використання Inline-кнопок або Reply-клавіатури – з найчастіше обираними містами.
- Підключення другого API (наприклад, для забруднення повітря чи новин).
- Перехід на Webhook і хостинг на сервері – для оптимізації продуктивності.
- Додавання графіків температури, інтерактивних карт.

Це дозволить підвищити функціональність, привабливість і масштабованість Telegram-бота.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЛЕННЯ

Матеріали були опубліковані:

В тезисах:

1. Крицький Я.Ю. ІНТЕГРАЦІЯ ЧАТ-БОТ TELEGRAM З INTERNET OF THINGS // V Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез 15 квітня 2025. – К.: ДУІКТ, https://duikt.edu.ua/ua/news-1-570-12409-v-mizhnarodna-naukovo-tehnichna-konferenciya-suchasniy-stan-ta-perspektivi-rozvitku-iot_kafedra-inzhenerii-programnogo-zabezpechennya-avtomatizovanih-sistem
2. Крицький Я.Ю. Особливості розробки Telegram-бота для отримання погоди з позиції кібербезпеки// Всеукраїнська науково-практична конференція "Цифрова трансформація кібербезпеки" Збірник тез 24 квітня 2025. – К.: ДУІКТ, https://duikt.edu.ua/ua/news-1-574-14029-duikt-zaproshue-do-uchasti-u-vseukrainskiy-naukoviy-konferencii--cifrova-transformaciya-kiberbezpeki_kafedra-informaciynoi-ta-kibernetichnoi-bezpeki