Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інфокомунікацій
(повна назва)
Кафедра інформаційно-мережної інженерії
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка телеграм-бота для автоматизованої діагностики інтернет-з'єднання
(тема)

Виконав:
здобувач 4 року навчання,
групи ТРІМІ-21-1
Даніїл Скалівенко
(власне ім'я, прізвище)
Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційно-мережна інженерія
( повна назва освітньої програми)
Керівник ас. Інна Штих
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ Валерій Безрук
(підпис) (власне ім'я, прізвище)

2025 р.

Не містить відомостей заборонених до відкритого публікування

Здобувач _____ /Даніїл Скалівенко/
(підпис) (власне ім’я, прізвище )
Керівник _____ /Інна Штих /
(підпис) (власне ім’я, прізвище)

Харківський національний університет радіоелектроніки

Факультет інфокомунікацій

Кафедра інформаційно-мережної інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 172 Телекомунікації та радіотехніка

Тип програми освітньо-професійна
(код і повна назва)

Освітня програма інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

# ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Скалівенку Даніїлу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка телеграм-бота для автоматизованої діагностики інтернет-з'єднання

затверджена наказом університету від 23 травня 2025 р. № 410 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 19 червня 2025 р.

3. Вихідні дані до роботи

Провести аналіз сучасного стану діагностики інтернет-з'єднання.

Розробити телеграм-бота для автоматизованої діагностики інтернет-з'єднання.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ

1. Аналітичний огляд сучасного стану діагностики інтернет-з'єднання

2. Розробка телеграм-бота для автоматизованої діагностики інтернет-з'єднання

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____

Слайди у форматі Power Point (назва роботи, актуальність, мета і завдання, аналіз аналогів, архітектура проєкту, використані технології, функція перевірки швидкості, функція перевірки доступності сайтів, функція збереження історії, функція діагностики, функція надання рекомендацій, розгортання та тестування, висновки)

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Ознайомлення із завданням. Уточнення ТЗ. | 23.05.2025 | вик. |
| 2 | Підбір літератури за темою роботи | 24.05.-30.05.2025 | вик. |
| 3 | Аналітичний огляд сучасного стану діагностики | 31.05.-06.06.2025 | вик. |
| | інтернет-з'єднання | | |
| 4 | Розробка телеграм-бота для автоматизованої | 07.06.-13.06.2025 | вик. |
| | діагностики інтернет-з'єднання | | |
| 5 | Оформлення презентаційного матеріалу, | 14.06.-17.06.2025 | вик. |
| | підготовка до захисту в ЕК | | |
| | | | |
| | | | |
| | | | |

Дата видачі завдання 23 травня 2025 р.

Здобувач _____ Даніїл Скалівенко
(підпис)

Керівник роботи _____ ас. Інна Штих
(підпис) (посада, власне ім'я, прізвище)

# РЕФЕРАТ

Пояснювальна записка: 59 с., 21 рис., 4 табл., 13 джерел, 3 додатки.

Метою кваліфікаційної роботи є розробка телеграм-бота, який дозволяє здійснювати автоматизовану діагностику інтернет-з’єднання, перевіряти швидкість доступу, доступність окремих сайтів, зберігати результати тестів, виявляти можливі причини нестабільної роботи мережі та надавати рекомендації з налаштування. Об'єкт дослідження – процес діагностики інтернет-з’єднання користувача.

У роботі досліджено створення телеграм-бота, призначеного для автоматизованої діагностики інтернет-з’єднання. Використано підходи структурного аналізу та модульного проєктування, а також мову програмування Python та фреймворк Aiogram. Реалізовано функції вимірювання швидкості з’єднання, перевірки доступності вебресурсів, автоматичної діагностики й генерації порад. Бот розгорнутий на хмарному хостингу Render.com із застосуванням технології Telegram Webhook. Отримані результати можуть бути використані як у побутових цілях, так і в освітніх чи сервісних проєктах, що потребують швидкої діагностики мережі.

ТЕЛЕГРАМ-БОТ, ДІАГНОСТИКА ІНТЕРНЕТ-З’ЄДНАННЯ, PYTHON, ШВИДКІСТЬ ІНТЕРНЕТУ, AIOGRAM, SPEEDTEST, PING, АВТОМАТИЗАЦІЯ, ТЕСТУВАННЯ

# THE ABSTRACT

Explanatory note: 59 p., 21 fig, 13 sources, 3 app.

The aim of this qualification work is to develop a Telegram bot that enables automated diagnostics of an internet connection, tests connection speed and website availability, saves test results, identifies possible causes of network instability, and provides configuration recommendations. The object of research is the process of diagnosing a user's internet connection.

The work explores the creation of a Telegram bot designed for automated network diagnostics. Structural analysis and modular design approaches were applied, along with the Python programming language and the Aiogram framework. The bot implements functions for measuring connection speed, checking website availability, performing automated diagnostics, and generating user-specific advice. It is deployed on the cloud hosting platform Render.com using the Telegram Webhook technology. The results obtained can be applied in both household use and educational or service-oriented projects that require rapid network diagnostics.

TELEGRAM BOT, INTERNET CONNECTION DIAGNOSTICS, PYTHON, INTERNET SPEED, AIOGRAM, SPEEDTEST, PING, AUTOMATION, TESTING

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ..... 8
ВСТУП ..... 9
1 АНАЛІТИЧНИЙ ОГЛЯД СУЧАСНОГО СТАНУ ДІАГНОСТИКИ ІНТЕРНЕТ-З'ЄДНАННЯ ..... 11
1.1 Аналіз предметної області ..... 11
1.2 Огляд та аналіз аналогів – сайтів онлайн-діагностик інтернет-з’єднання ..... 14
1.3 Огляд телеграм-ботів ..... 17
1.4 Постановка завдання ..... 19
2 РОЗРОБКА ТЕЛЕГРАМ-БОТА ДЛЯ АВТОМАТИЗОВАНОЇ ДІАГНОСТИКИ ІНТЕРНЕТ-З'ЄДНАННЯ ..... 21
2.1 Обґрунтування вибору технологій розробки та середовища розробки телеграм-бота ..... 21
2.2 Розробка телеграм-бота ..... 23
2.2.1 Розробка функції перевірки швидкості інтернет-з'єднання ..... 24
2.2.2 Розробка функції перевірки доступності сайтів ..... 26
2.2.3 Розробка функції збереження історії тестів користувача ..... 28
2.2.4 Розробка функції пошуку можливих причин низької швидкості чи нестабільного з'єднання ..... 31
2.2.5 Розробка функції рекомендацій з налаштування мережі ..... 33
2.3 Підключення та налаштування роботи телеграм-бота на хостингу ..... 35
2.4 Тестування телеграм-бота ..... 38
ВИСНОВКИ ..... 40
ПЕРЕЛІК ПОСИЛАНЬ ..... 42
ДОДАТОК А ..... 44
ДОДАТОК Б ..... 50
ДОДАТОК В ..... 51

# ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface – спосіб взаємодії комп'ютерних програм між собою;

DSL – Digital Subscriber Line – сімейство технологій, які забезпечують широкосмуговий доступ в Інтернет через звичайні телефонні лінії;

FTTH – Fiber To The Home – технологія, яка означає, що оптичний кабель прямує до конкретного будинку або квартири, а не до розподільного вузла в багатоповерховому будинку або котеджного поселення;

HTTPS – Hyper Text Transfer Protocol Secure – шифрований протокол передачі гіпертекстових даних.

# ВСТУП

У сучасних умовах стабільне інтернет-з’єднання стало життєво важливим не лише для особистих потреб, а й для ефективної професійної діяльності. Безперервний доступ до мережі значною мірою впливає на роботу підприємств, якість освіти, швидкість комунікацій та інші аспекти повсякденного життя. Водночас, попри розвиток технологій, користувачі й далі зіштовхуються з проблемами – від нестабільного з’єднання до зниження швидкості передачі даних або обмеженої доступності ресурсів [1].

Щоб вчасно виявляти та усувати такі проблеми, важливо проводити діагностику інтернет-з’єднання. Вона допомагає швидко оцінити якість каналу, зрозуміти потенційні джерела несправностей і визначити наступні кроки. Хоча на ринку представлено чимало сервісів для такої діагностики, більшість з них орієнтовані на ручне використання через веб-інтерфейс, не зберігають історії тестів або не надають чітких порад.

В останні роки телеграм-боти здобули популярність як інструменти автоматизації завдяки своїй доступності, зручності та можливості інтеграції з іншими сервісами [2]. Створення телеграм-бота, здатного автоматизувати процес діагностики інтернет-з’єднання, відкриває шлях до ефективного, зручного та мобільного рішення для моніторингу якості мережі.

Метою кваліфікаційної роботи є розробка телеграм-бота, який дозволяє здійснювати автоматизовану діагностику інтернет-з’єднання, перевіряти швидкість доступу, доступність окремих сайтів, зберігати результати тестів, виявляти можливі причини нестабільної роботи мережі та надавати рекомендації з налаштування.

Основними завданнями кваліфікаційної роботи є:

- дослідити предметну область та проаналізувати наявні рішення;
- розробити архітектуру системи й підібрати відповідні технології;
- реалізувати функціональність діагностики;

- забезпечити збереження історії тестів та генерацію рекомендацій;

- провести тестування створеного продукту.

Отже, створення телеграм-бота для автоматизованої діагностики інтернет-з’єднання є на сьогодні актуальним і корисним завданням, що сприятиме підвищенню ефективності роботи користувачів з мережевими сервісами.

# 1 АНАЛІТИЧНИЙ ОГЛЯД СУЧАСНОГО СТАНУ ДІАГНОСТИКИ ІНТЕРНЕТ-З'ЄДНАННЯ

## 1.1 Аналіз предметної області

Інтернет-з’єднання – це комплексне поєднання апаратних, програмних і організаційних рішень, що забезпечують взаємодію користувача з глобальною мережею Інтернет через передачу та прийом даних. На практиці це охоплює роботу кабелів, маршрутизаторів, програмного забезпечення, мережевих протоколів, а також інфраструктури провайдерів. У сучасних реаліях постійний та якісний доступ до Інтернету перетворився з простої зручності на базову потребу – як для повсякденного життя окремої людини, так і для ефективного функціонування організацій, установ, бізнесів та навіть державних інституцій.

Стабільне інтернет-з’єднання необхідне не лише для перегляду вебсторінок чи використання соцмереж, а й для виконання професійних обов’язків, взаємодії у віддалених командах, участі в онлайн-навчанні, проведення телемедичних консультацій, реалізації електронної комерції, доступу до хмарних платформ, сервісів відеоконференцій та інших критичних цифрових сервісів. Саме тому якість з’єднання стає одним із визначальних факторів цифрової ефективності.

Розглянемо основні параметри, що визначають якість інтернет-з’єднання. По-перше, це швидкість завантаження (Download speed) – показник, який демонструє, скільки даних користувач може отримати з Інтернету за одиницю часу. Зазвичай вимірюється у мегабітах за секунду (Мбіт/с). Висока швидкість завантаження є ключовою для комфортного перегляду відео у високій якості, оперативного відкриття сторінок, швидкого завантаження файлів та стабільної роботи у хмарних середовищах.

Швидкість вивантаження (Upload speed) – параметр, що вказує, який обсяг даних пристрій здатен передати до мережі за секунду. Вона критично важлива при відеозв'язку, стримінгу, резервному копіюванні, надсиланні великих документів або використанні систем для обміну даними в реальному часі.

Затримка або пінг (Ping) – це час, який потрібен пакету даних, щоб досягти сервера і повернутися назад. Вимірюється у мілісекундах. Низький пінг є особливо важливим у застосунках, де час реакції має вирішальне значення – наприклад, в онлайн-іграх, голосових викликах (VoIP) або під час потокового передавання відео з малою затримкою.

Втрати пакетів (Packet loss) – відображають частку інформації, яка не доходить до адресата через помилки в мережі. Навіть незначний відсоток втрат може призвести до помітних збоїв: спотворення звуку, зависання зображення, розриви зв'язку.

Стабільність з'єднання (Jitter) – це показник, який описує змінність затримки у часі. Чим більш нерівномірною є затримка між пакетами, тим більший jitter. Його високі значення викликають "пригальмовування" голосу під час дзвінків, сіпання зображення у відеоконференціях, складності в управлінні у відеоіграх тощо [1].

Залежно від потреб і доступної інфраструктури, інтернет-з'єднання може бути реалізовано через різні технології, кожна з яких має свої особливості використання, переваги та обмеження.

Фіксовані з'єднання: до них належать оптоволоконні лінії (FTTH), DSL, Ethernet та кабельний інтернет. Такі типи підключень забезпечують високу швидкість, мінімальні затримки та стабільність сигналу. Вони найкраще підходять для стаціонарного використання – вдома або в офісі, але вимагають наявності фізичної інфраструктури й витрат на її прокладку.

Мобільні мережі: включають стандарти 3G, 4G, LTE та новітні 5G. Їх головною перевагою є мобільність – можливість залишатися на зв'язку в дорозі, на дачі чи в сільській місцевості. Проте якість зв'язку може суттєво варіюватися

залежно від покриття, кількості користувачів у мережі та особливостей місцевості.

Бездротові технології: насамперед Wi-Fi у локальній мережі та супутниковий інтернет. Вони зручні для організації підключення у приміщеннях або в умовах, де неможливо провести кабель. Проте супутниковий інтернет часто має вищу затримку, а Wi-Fi – залежить від якості маршрутизатора та наявності перешкод у приміщенні.

Попри стрімкий розвиток технологій, користувачі регулярно стикаються з проблемами: низька швидкість, високий пінг, періодичні перебої або недоступність окремих сайтів. У таких випадках виникає потреба у діагностиці, яка допомагає оцінити технічний стан з'єднання, виявити причини збоїв та обрати шляхи їх усунення.

Комплекс діагностичних заходів зазвичай включає:

- вимірювання швидкості з'єднання (завантаження/вивантаження);
- перевірку затримок і втрат пакетів;
- трасування маршруту даних до конкретного сайту або сервера;
- аналіз доступності вебресурсів;
- порівняння отриманих показників із заявленими параметрами.

Для цього використовуються різноманітні інструменти, які можна умовно поділити на кілька категорій.

Командні утиліти ОС: ping, tracert (у Windows), traceroute, netstat, speedtest-cli (у Linux та macOS). Вони ефективні, але потребують знання командної строки.

Графічні програми: такі як NetLimiter, GlassWire, Wireshark, які пропонують глибший аналіз і візуалізацію трафіку.

Онлайн-сервіси: Speedtest.net, Fast.com, Pingdom, Cloudflare Speed Test – прості у використанні, але найчастіше обмежені в деталізації та можливості аналізу.

Незважаючи на доступність цих засобів, більшість із них розраховані на досвідчених користувачів, які мають технічні навички. Окрім цього, діагностика

часто передбачає перехід між кількома сайтами або консольними командами, а результати можуть подаватись у складному, неінтуїтивному вигляді.

У зв’язку з цим актуалізується потреба у простих, автоматизованих та персоналізованих засобах діагностики, які здатні:

- запускатися у два кліки, без зайвих налаштувань;
- виконувати необхідний набір перевірок;
- надавати результати у зрозумілому вигляді;
- зберігати історію для подальшого аналізу;
- пропонувати індивідуальні рекомендації залежно від ситуації.

Одним із найперспективніших підходів у цьому напрямі є використання телеграм-ботів. Їх популярність серед користувачів, простота запуску, гнучкість Telegram Bot API і підтримка інтеграцій роблять їх ідеальними помічниками у повсякденній діагностиці мережі. Такий бот може об'єднати в собі функціональність декількох інструментів, бути завжди під рукою і надавати допомогу навіть тим, хто не має технічної підготовки.

## 1.2 Огляд та аналіз аналогів – сайтів онлайн-діагностик інтернет-з’єднання

У межах дослідження було обрано три найпопулярніші сервіси для онлайн-діагностики якості інтернет-з’єднання: Speedtest by Ookla, Fast.com та PingPlotter. Вони суттєво різняться за своїми можливостями, зручністю використання й глибиною аналізу.

Speedtest by Ookla (рис. 1.1) – один із найвідоміших інструментів для перевірки швидкості інтернету. Сервіс надає дані про швидкість завантаження, вивантаження та затримку (пінг). Також користувач може обрати сервер для тестування та переглянути історію попередніх перевірок за умови реєстрації. Разом з тим сервіс зосереджується переважно на базових параметрах, без глибшого аналізу причин можливої нестабільності.

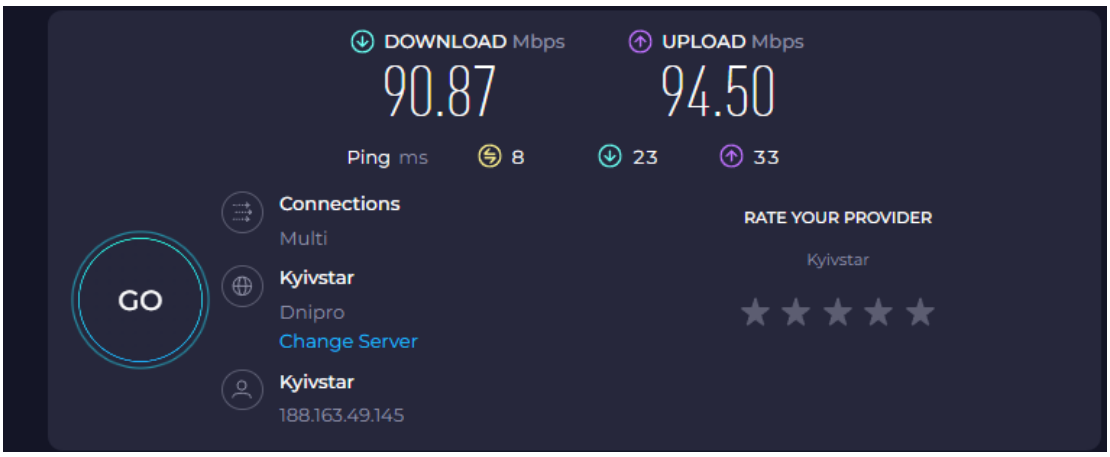


Рисунок 1.1 – Інтерфейс сервісу Speedtest by Ookla

Fast.com, створений компанією Netflix, є мінімалістичним сервісом, який автоматично запускає перевірку швидкості завантаження без потреби у додаткових налаштуваннях (рис. 1.2). Після запуску можна переглянути також інші параметри: пінг, швидкість вивантаження та затримку при навантаженні. Недоліками сервісу є відсутність збереження історії перевірок, неможливість вибору сервера і відсутність рекомендацій користувачеві.

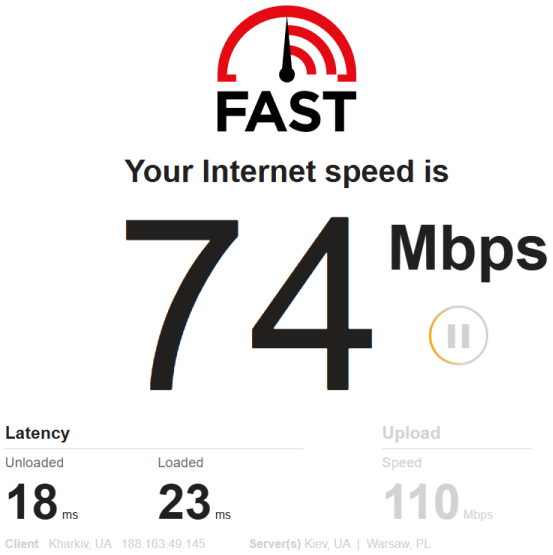


Рисунок 1.2 – Інтерфейс сервісу Fast.com

PingPlotter (рис. 1.3) орієнтований на глибоку діагностику стабільності мережі та трасування маршрутів передачі даних. Його функціонал дозволяє виявляти точки втрат пакетів, серйозні затримки на шляху до сервера та аналізувати стабільність з’єднання в динаміці. Проте сервіс більше підходить для фахівців, оскільки має складний інтерфейс і вимагає знань у сфері мережевих протоколів.

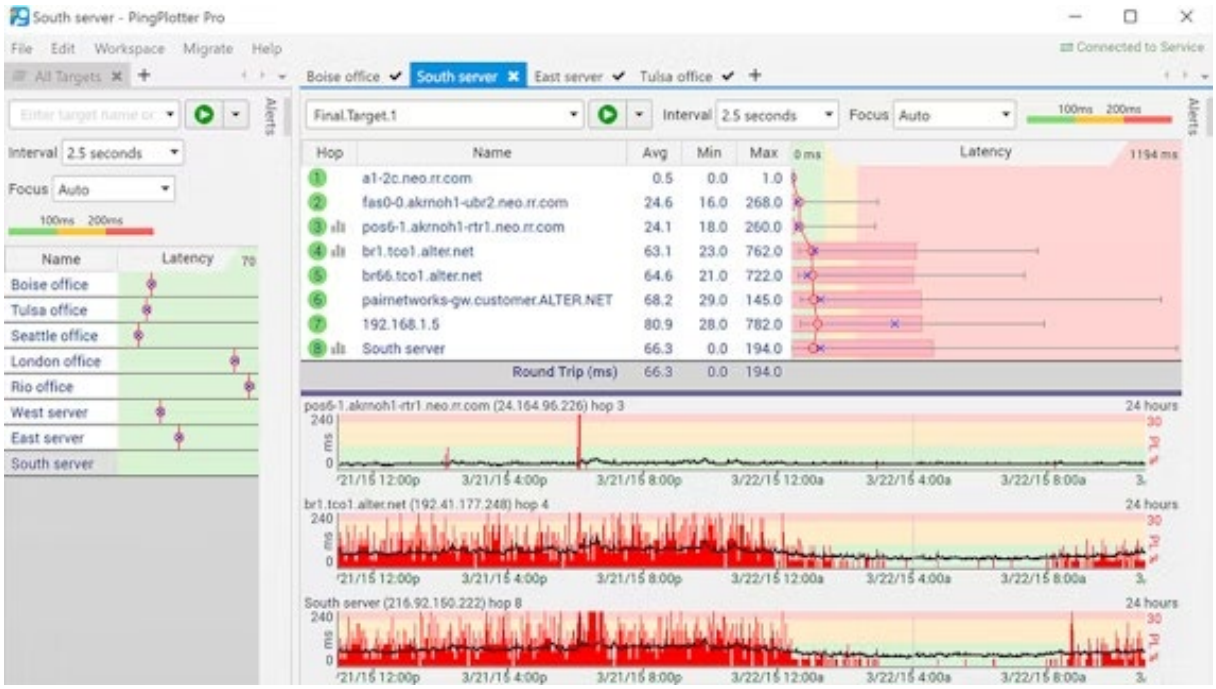


Рисунок 1.3 – Інтерфейс сервісу Fast.com

На основі проведеного аналізу було складено порівняльну табл. 1.1. Вона демонструє, що жоден з розглянутих сервісів не поєднує в собі повний спектр діагностичних можливостей, автоматичне формування рекомендацій і простоту використання, орієнтовану на пересічного користувача. Це підкреслює доцільність створення телеграм-бота, який об’єднає ключові переваги згаданих сервісів: простоту запуску, збереження історії перевірок і можливість отримання корисних порад.

Таблиця 1.1 – Порівняльна таблиця онлайн-сервісів діагностики інтернет-з’єднання

| Критерій | Speedtest by Ookla | Fast.com | PingPlotter |
|---|---|---|---|
| Вимірювання швидкості завантаження/вивантаження | + | + | - |
| Вимірювання затримки (пінгу) | + | + | + |
| Вимірювання втрат пакетів | - | - | + |
| Можливість обрати сервер | + | - | + |
| Збереження історії тестів | частково (після реєстрації) | - | + |
| Надання рекомендацій щодо проблем | - | - | частково |
| Орієнтація на технічних фахівців | ні | ні | так |
| Вартість | безкоштовно | безкоштовно | 29$/місяць |

## 1.3 Огляд телеграм-ботів

Сьогодні телеграм-боти активно застосовуються як ефективний інструмент для автоматизованої обробки запитів користувачів. Їхні переваги – це легкий доступ, підтримка різних платформ, простота впровадження та інтеграція з уже існуючими сервісами. Завдяки цьому боти знайшли широке застосування в інформаційних, сервісних і комунікаційних системах [2].

Під час дослідження проводився пошук телеграм-ботів, які можуть виконувати функції перевірки швидкості інтернету, доступності сайтів або повної діагностики з’єднання. В результаті було виявлено, що частина з них, наприклад, @SpeedTestBot, @PingToolsBot, на момент дослідження вже не функціонують (рис. 1.4).

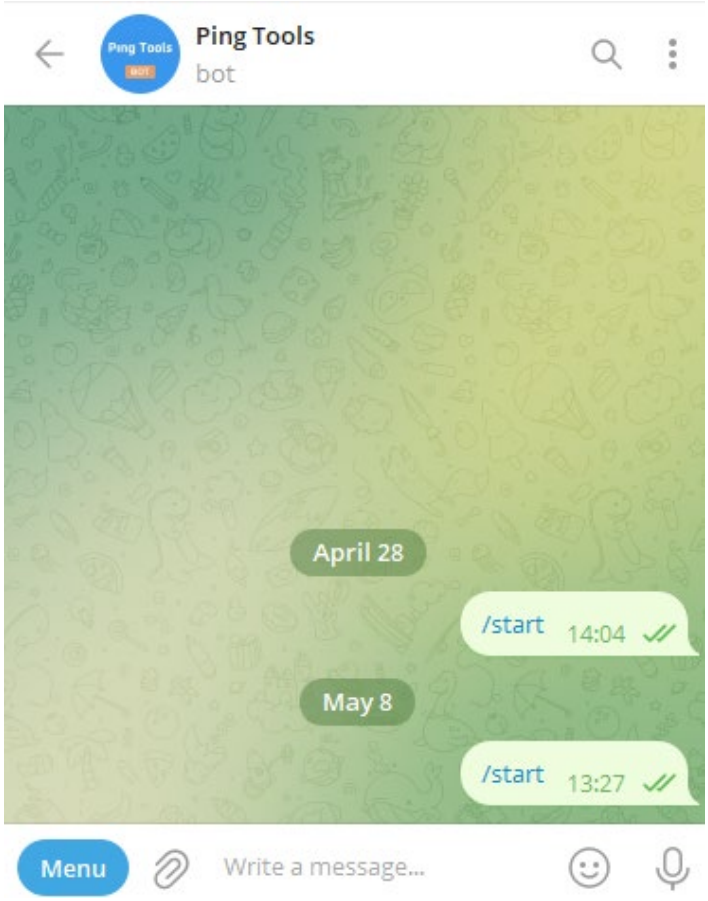


Рисунок 1.4 – Інтерфейс бота @PingToolsBot (бот не працює)

Інші ж наявні боти, такі як @SitePingBot, в основному орієнтовані на завдання моніторингу сайтів або управління серверними ресурсами, і не надають повного функціоналу діагностики для звичайного користувача (рис. 1.5). Їхня основна функціональність полягає в перевірці доступності вебресурсів шляхом надсилання ping-запитів до зазначених адрес, що може бути корисним для адміністраторів сайтів або системних інженерів. Проте такі рішення зазвичай не включають можливості вимірювання швидкості з’єднання, аналізу втрат пакетів або зберігання результатів перевірок для подальшого аналізу. Крім того, інтерфейс таких ботів часто не адаптований під потреби пересічного користувача, що обмежує їхнє застосування в побутових умовах або при виникненні типових проблем з підключенням.

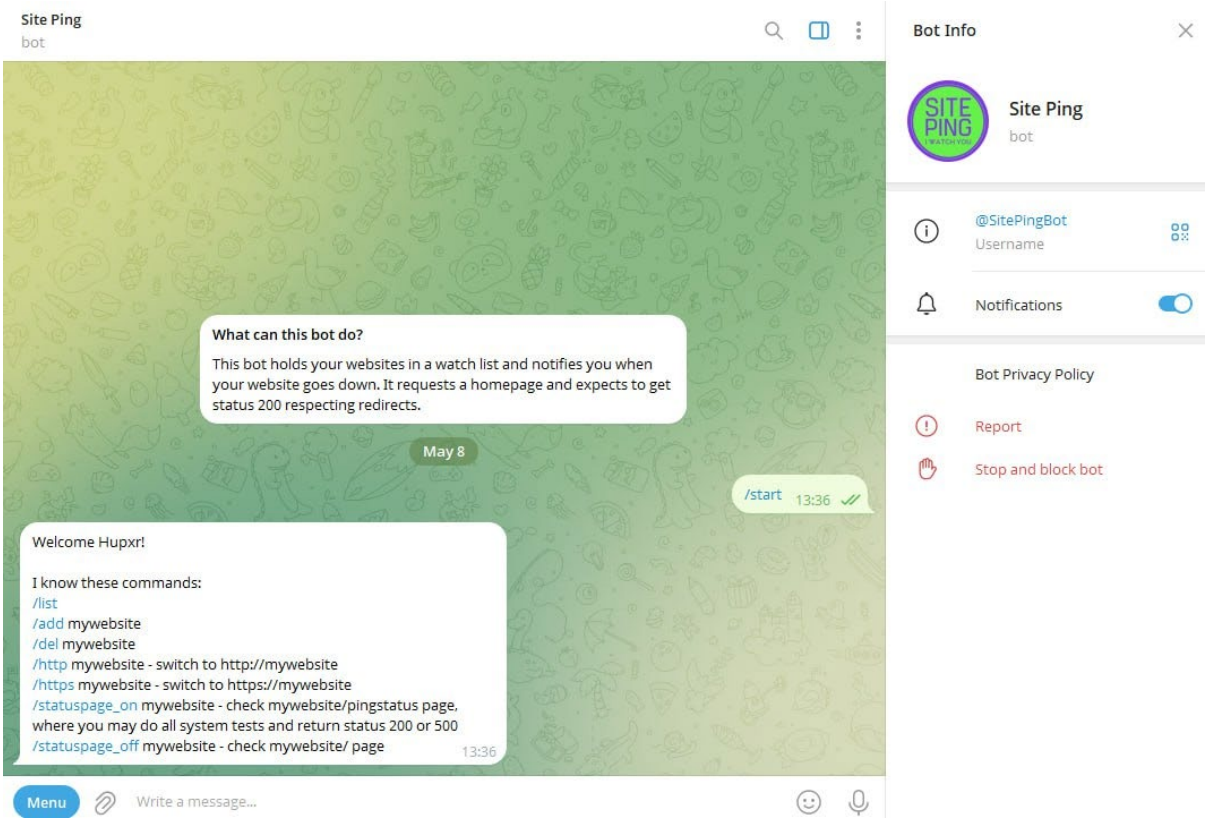


Рисунок 1.5 – Інтерфейс бота @SitePingBot для моніторингу сайтів

Таким чином, в результаті аналізу не були виявлені телеграм-боти, здатні проводити повноцінну або навіть часткову автоматизовану діагностику інтернет-з’єднання, включаючи перевірку швидкості, тестування доступності сайтів, збереження результатів і надання порад.

Це ще раз підтверджує актуальність та інноваційність теми кваліфікаційної роботи й потребу у створенні такого програмного рішення, яке б поєднувало простоту використання, широкий функціонал і було б корисним для широкої аудиторії.

## 1.4 Постановка завдання

У результаті проведеного аналітичного огляду предметної області та існуючих аналогів встановлено, що на сьогодні немає телеграм-ботів, які б

дозволяли користувачам проводити повноцінну автоматизовану діагностику інтернет-з’єднання.

Метою кваліфікаційної роботи є розробка телеграм-бота для автоматизованої діагностики інтернет-з’єднання, який забезпечуватиме перевірку основних характеристик мережі, аналіз доступності ресурсів, збереження історії тестів та надання рекомендацій щодо усунення виявлених проблем.

Об'єкт дослідження – процес діагностики інтернет-з’єднання користувача.

Предмет дослідження – методи автоматизації діагностики інтернет-з’єднання із застосуванням телеграм-бота.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати предметну область діагностики інтернет-з’єднання;
- провести огляд та аналіз існуючих онлайн-сервісів діагностики мережі;
- визначити вимоги до функціональності телеграм-бота;
- обґрунтувати вибір технологій і середовища розробки;
- реалізувати телеграм-бот із такими основними функціями:

1. перевірка швидкості завантаження та вивантаження даних;
2. перевірка доступності обраних сайтів;
3. збереження історії тестувань для кожного користувача;
4. аналіз результатів і пошук можливих причин проблем;
5. надання рекомендацій щодо оптимізації роботи мережі;

- налаштувати роботу телеграм-бота на хостингу;
- провести тестування розробленого програмного продукту та оцінити його відповідність заданим вимогам.

Очікується, що створений телеграм-бот буде зручним у використанні, доступним для різних категорій користувачів і забезпечить ефективну первинну діагностику з’єднання без потреби у зверненні до фахівців.

# 2 РОЗРОБКА ТЕЛЕГРАМ-БОТА ДЛЯ АВТОМАТИЗОВАНОЇ ДІАГНОСТИКИ ІНТЕРНЕТ-З'ЄДНАННЯ

## 2.1 Обґрунтування вибору технологій розробки та середовища розробки телеграм-бота

Підбір відповідних технологій на етапі проєктування програмного забезпечення має ключове значення, оскільки саме від цього залежить успішність усього життєвого циклу продукту – від першого рядка коду до масштабування готової системи. Від правильного рішення залежить не лише зручність реалізації логіки програми чи її продуктивність, а й стабільність роботи, можливість розширення функціоналу в майбутньому, простота підтримки та взаємодія з іншими системами [3]. В умовах динамічного розвитку технологій і зростання вимог до якості програмних рішень, вибір перевірених і гнучких інструментів стає особливо критичним.

У рамках цього проєкту для створення телеграм-бота, який здійснює автоматизовану діагностику інтернет-з’єднання, було свідомо обрано набір технологій, які поєднують у собі надійність, гнучкість, активну підтримку спільноти розробників та зручність у впровадженні:

Мова програмування Python – через її читабельний синтаксис, високу продуктивність при прототипуванні, універсальність та активну екосистему, що охоплює тисячі бібліотек для різних сфер: від обробки даних до машинного навчання.

Фреймворк Aiogram для роботи з Telegram Bot API – вибір саме цього фреймворку обумовлений його асинхронною архітектурою, високою швидкістю обробки запитів та зручним поділом логіки на модулі, що дозволяє чітко структурувати код бота.

Бібліотеки speedtest-cli та ping3 для тестування з’єднання – вони дозволяють реалізувати основну функцію проєкту без потреби у зовнішніх

вебсервісах, при цьому забезпечують точність, стабільність і простоту використання.

База даних SQLite для зберігання історії перевірок – ідеальне рішення для ботів, які не потребують масштабованого сховища, але мають обробляти й зберігати персоналізовані дані з високою швидкістю.

Хмарний хостинг Render.com для розгортання бота – сучасна хостинг-платформа з підтримкою CI/CD-процесів, інтеграцією з GitHub, автоматичним HTTPS і можливістю гнучкого налаштування середовища виконання.

Python на сьогодні залишається однією з найбільш використовуваних мов програмування у світі, особливо у сфері веброзробки, автоматизації, аналізу даних і створення ботів [4]. Простота її синтаксису полегшує підтримку проєкту, а велика кількість бібліотек дозволяє швидко реалізовувати навіть складну функціональність. Для задач, пов’язаних із мережею, зокрема діагностики інтернет-з’єднання, Python має все необхідне – бібліотеки speedtest-cli, ping3, requests тощо. Крім того, мова підтримує асинхронне програмування, що критично важливо для бота, який повинен паралельно обробляти десятки запитів без затримок у роботі.

Фреймворк Aiogram дозволяє легко створювати телеграм-ботів будь-якої складності. Його структура побудована на асинхронному програмуванні із використанням asyncio, що забезпечує високий рівень продуктивності. Aiogram підтримує всі актуальні можливості Telegram Bot API: обробку команд, inline-запитів, повідомлень, callback-кнопок тощо. До того ж, спільнота цього фреймворку активно розвивається, регулярно оновлюючи документацію та адаптуючи код під зміни API [5].

Для реалізації перевірки швидкості з’єднання обрано бібліотеку speedtest-cli – це Python-обгортка до сервісу Speedtest by Ookla, яка дозволяє зчитувати всі ключові параметри з’єднання (швидкість завантаження, вивантаження, затримку) прямо з коду програми, без необхідності переходити на сторонні вебсайти [6]. Такий підхід робить систему повністю автономною та незалежною від браузерів.

Доступність сайтів перевіряється за допомогою бібліотеки requests, яка забезпечує просте й водночас гнучке виконання HTTP-запитів [7]. Вона дозволяє обробляти як стандартні відповіді, так і коди помилок, тим самим забезпечуючи точне виявлення недоступних ресурсів.

Уся історія тестувань зберігається у вбудованій базі SQLite – це реляційна база даних, яка не потребує окремого сервера, легко інтегрується в Python-проєкти, працює з локальними файлами та дозволяє ефективно обробляти невеликі обсяги персоналізованої інформації [8]. Її застосування оптимальне в контексті телеграм-ботів, де важлива швидкість доступу й відсутність зайвих зовнішніх залежностей.

У якості середовища розгортання використано Render.com – хмарну платформу нового покоління, яка поєднує простоту використання із професійними можливостями: автоматичним деплоєм із репозиторію GitHub, захищеним HTTPS-доступом, масштабуванням та зручним управлінням сервісами [9]. Це дозволяє безперервно оновлювати бота, не зупиняючи його роботу.

Підключення до Telegram Bot API реалізується через механізм Webhook, який дає змогу моментально отримувати запити від користувачів. На відміну від підходу polling, webhook дозволяє уникнути затримок, зменшує навантаження на сервер і забезпечує блискавичну реакцію бота на повідомлення.

## 2.2 Розробка телеграм-бота

Процес створення телеграм-бота для автоматичної діагностики з’єднання охоплює декілька важливих етапів: проєктування структури бота, розробку окремих функціональних модулів, інтеграцію засобів тестування мережі, організацію збереження результатів та створення системи надання рекомендацій користувачу [10].

Архітектура бота є модульною – кожен компонент відповідає за певну частину функціоналу:

- модуль обробки команд користувача;

- модуль тестування швидкості інтернет-з'єднання;

- модуль перевірки доступності сайтів;

- модуль збереження історії результатів;

- модуль аналізу результатів і формування рекомендацій.

Користувач взаємодіє з ботом за допомогою команд Telegram: /start, /check_speed, /check_sites, /history, /diagnose, /advice. Уся логіка реалізована в асинхронному режимі, що дозволяє системі ефективно обслуговувати багатьох користувачів одночасно – без затримок і конфліктів.

Кожна з основних функцій бота докладно описана в наступних підрозділах.

### 2.2.1 Розробка функції перевірки швидкості інтернет-з'єднання

Одна з базових функцій телеграм-бота – перевірка фактичної швидкості інтернет-з’єднання користувача. Для її реалізації використовується бібліотека speedtest-cli, яка є оболонкою до сервісу Speedtest by Ookla і дозволяє запускати перевірку без браузера.

Ця бібліотека має декілька важливих переваг: автоматичний вибір оптимального сервера для тестування, вимірювання швидкості завантаження (Download) і вивантаження (Upload), вимірювання затримки (Ping), можливість інтеграції результатів у власні програми [6].

Логіка роботи цієї функції така:

- користувач надсилає команду /check_speed боту;

- бот надсилає відповідь про початок тестування ("Перевіряю швидкість з’єднання...");

- використовуючи бібліотеку speedtest-cli, бот проводить вибір найближчого сервера для тестування, вимірювання пінгу, тестування швидкості завантаження даних, тестування швидкості вивантаження даних;

- бот формує звіт у текстовому форматі та надсилає його користувачу [11].

Фрагмент коду функції перевірки швидкості інтернет-з’єднання показаний на рис. 2.1 та 2.2.

```python
def run_speedtest():
    try:
        st = speedtest.Speedtest()
        st.get_best_server()
        download = st.download() / 1_000_000
        upload = st.upload() / 1_000_000
        ping = st.results.ping
        return (f"▼ Завантаження: {download:.2f} Мбіт/с\n"
                f"▲ Вивантаження: {upload:.2f} Мбіт/с\n"
                f"📡 Пінг: {ping:.0f} мс")
    except Exception:
        return "Не вдалося виконати перевірку швидкості. Спробуйте ще раз"
```

Рисунок 2.1 – Фрагмент коду перевірки швидкості з’єднання (опис функції)

```python
@router.message(Command("check_speed"))
async def check_speed(message: types.Message):
    await message.answer("⏳ Перевіряю швидкість з’єднання...")
    result = run_speedtest()
    save_test_result(message.from_user.id, "speedtest", result)
    await message.answer(result)
```

Рисунок 2.2 – Фрагмент коду перевірки швидкості з’єднання (виклик функції)

Результат взаємодії з ботом відображений на рис. 2.3.

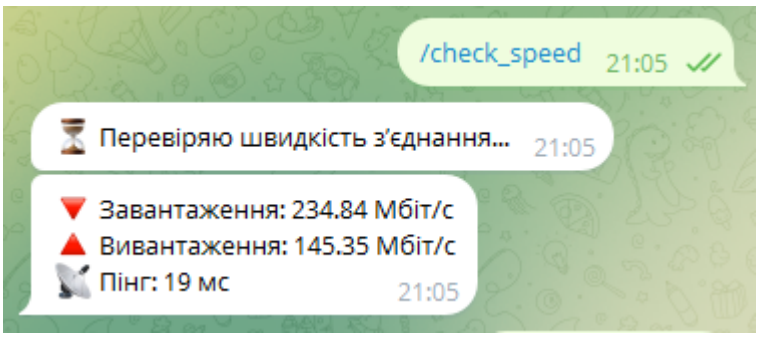


Рисунок 2.3 – Результат перевірки швидкості з’єднання ботом

Застосування асинхронного підходу дає змогу уникнути блокування інших запитів під час виконання тесту швидкості, що значно покращує взаємодію з

ботом і робить його використання більш комфортним для користувачів. Реалізована функція є ключовим елементом системи діагностики, адже саме вона дозволяє швидко отримати основні характеристики інтернет-з’єднання, які в подальшому використовуються для формування відповідних висновків та надання рекомендацій.

### 2.2.2 Розробка функції перевірки доступності сайтів

Ще однією важливою частиною діагностики інтернет-з’єднання є визначення доступності окремих вебресурсів. Це дозволяє зрозуміти, чи проблема полягає у самому з’єднанні користувача, чи вона викликана зовнішніми чинниками, як-от збій на стороні провайдера або сервера.

Для реалізації цієї функціональності застосовано комбінований підхід. Щоб оцінити досяжність сайту на рівні IP-з’єднання, використовується бібліотека ping3. Для перевірки доступності сайтів через HTTP-запити використовується бібліотека requests, яка дозволяє аналізувати відгук вебсерверів [12].

Алгоритм роботи функції перевірки доступності сайтів:

- користувач надсилає команду /check_sites;

- бот запитує перелік сайтів для перевірки у форматі текстового повідомлення (наприклад, "google.com, wikipedia.org, example.com");

- для кожного сайту бот виконує: перевірку пінгом (із зазначенням часу відповіді або факту недоступності), HTTP-запит до головної сторінки сайту, а також формує коротке пояснення результату;

- на завершення бот надсилає користувачу детальний звіт про кожен перевірений сайт.

Фрагмент коду перевірки доступності сайтів наведений на рис. 2.4 та 2.5.

```python
def run_ping_check(site_list):
    result = ""
    for site in site_list.split(","):
        site = site.strip()
        if not site.startswith("https"):
            url = f"https://{site}"
        else:
            url = site
        try:
            r = requests.get(url, timeout=3)
            status = r.status_code
            if status == 200:
                status_text = "сайт доступний"
            elif status == 403:
                status_text = "доступ заборонено (403)"
            elif status == 404:
                status_text = "сторінку не знайдено (404)"
            elif status >= 500:
                status_text = "помилка на сервері"
            else:
                status_text = f"відповідь: {status}"
        except Exception:
            status_text = "сайт недоступний або помилка з'єднання"

        result += f"🌐 {site}:\n• HTTP статус: {status_text}\n\n"
    return result.strip()
```

Рисунок 2.4 – Фрагмент коду перевірки доступності сайтів (опис функції)

```python
@router.message(Command("check_sites"))
async def check_sites(message: types.Message, state: FSMContext):
    await message.answer("Введіть сайти через кому (наприклад: google.com, wiki
    await state.set_state(Form.waiting_for_sites)
```

Рисунок 2.5 – Фрагмент коду перевірки доступності сайтів (виклик функції)

В результаті перевірки користувач отримує зрозуміле пояснення навіть у разі виникнення технічної помилки. Це дозволяє проводити повноцінну й при цьому інтуїтивно зрозумілу діагностику доступності ресурсів без залучення фахівців. Результат перевірки ботом доступності сайтів показаний на рис. 2.6.

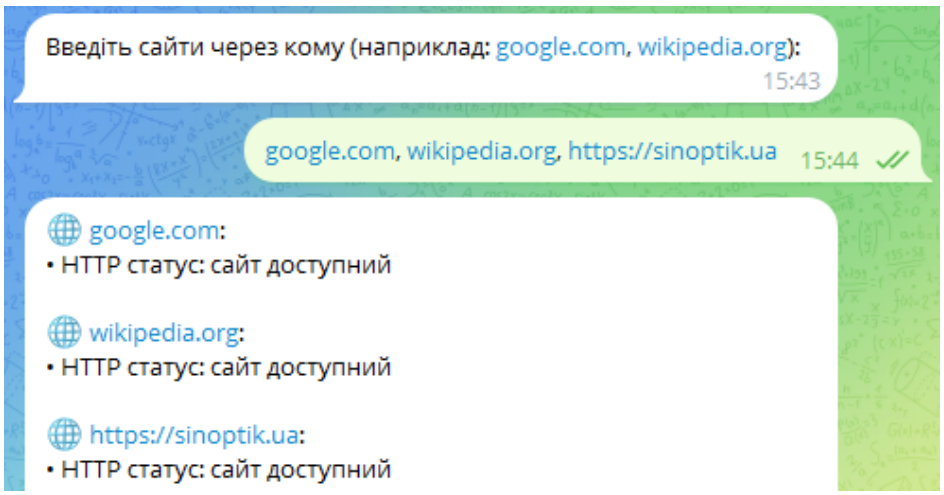


Рисунок 2.6 – Результат перевірки доступності сайтів

### 2.2.3 Розробка функції збереження історії тестів користувача

Щоб надати користувачу можливість аналізувати зміну стану з'єднання у динаміці, було реалізовано функцію збереження історії тестувань. Ця функція дозволяє повертатися до попередніх результатів та відстежувати, як змінювалась якість інтернету з часом.

У якості сховища даних використовується SQLite – легка вбудована база даних, яка не вимагає окремого сервера й повністю інтегрується в середовище Python через стандартну бібліотеку sqlite3 [8]. Вона забезпечує достатню швидкість для роботи з невеликими обсягами даних, зібраними для кожного окремого користувача. Структуру збереження історії тестів наведено в табл. 2.1

Таблиця 2.1 – Структура таблиці бази даних

| Поле | Тип даних | Опис |
|---|---|---|
| id | INTEGER | Унікальний ідентифікатор запису |
| user_id | INTEGER | ID користувача Telegram |
| test_type | TEXT | Тип тесту ("speedtest", "ping_check") |
| result | TEXT | Результати тестування у текстовому вигляді |
| timestamp | DATETIME | Дата та час проведення тесту |

Алгоритм роботи функції збереження історії:

- після проведення перевірки швидкості або доступності, бот формує результати у вигляді тексту;

- ці дані разом з ID користувача та типом тесту записуються до бази даних;

- користувач надсилає команду /history;

- бот виводить останні тести в зручному вигляді.

Завдяки збереженню тестів окремо для кожного Telegram ID забезпечується персоналізований підхід. Також це дозволяє уникнути затримок у роботі бота, оскільки SQLite працює швидко та не вимагає додаткових ресурсів. Фрагменти коду та приклад роботи бота наведені на рис. 2.7-2.9.

```python
import sqlite3
from pathlib import Path

DB_PATH = Path(__file__).parent.parent / "data" / "history.db"

def init_db():
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS history (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER NOT NULL,
            test_type TEXT NOT NULL,
            result TEXT NOT NULL,
            timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
        )
    ''')
    conn.commit()
    conn.close()
```

Рисунок 2.7 – Фрагмент коду для створення таблиці

```
21  def save_test_result(user_id, test_type, result):
22      conn = sqlite3.connect(DB_PATH)
23      cursor = conn.cursor()
24      cursor.execute('''
25          INSERT INTO history (user_id, test_type, result)
26          VALUES (?, ?, ?)
27      ''', (user_id, test_type, result))
28      conn.commit()
29      conn.close()
```

Рисунок 2.8 – Фрагмент коду для збереження результату

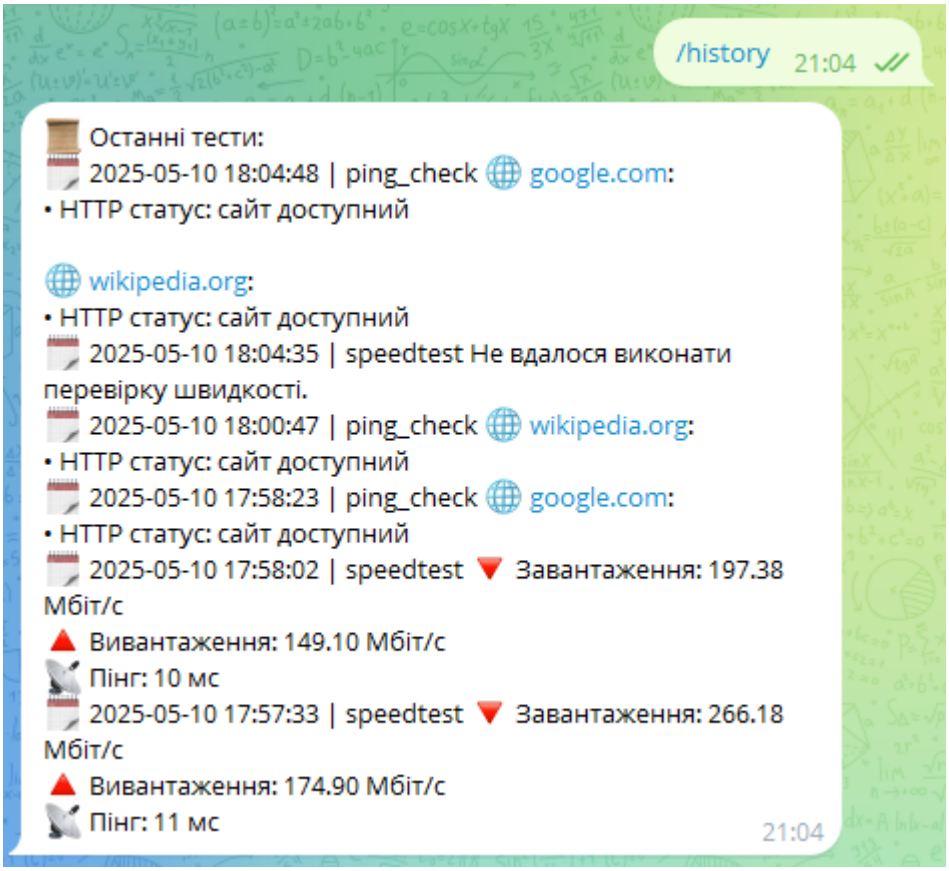


Рисунок 2.9 – Результат виведення історії тестів

Розроблена функція забезпечує зручний доступ до перегляду попередніх результатів та їхнього аналізу. Завдяки їй користувач отримує можливість відстежувати зміни в якості свого інтернет-з’єднання з часом і тримати ситуацію під контролем.

2.2.4 Розробка функції пошуку можливих причин низької швидкості чи нестабільного з'єднання

Щоб користувач не просто бачив результати тестів, а міг зрозуміти, що саме може бути причиною проблем із підключенням, у боті реалізовано окрему функцію діагностики. Вона аналізує результати тестів і пропонує гіпотези щодо можливих причин поганої роботи мережі.

Функція працює на базі на простої системи правил, що аналізують типові параметри:

- низька швидкість завантаження чи вивантаження;
- високий пінг;
- наявність недоступних сайтів;
- нестабільні або надто різні результати в історії тестувань.

На основі цього бот формує попередні висновки про ймовірні проблеми. Ключові правила аналізу наведені в табл. 2.2.

Таблиця 2.2 – Умови і можливі причини проблем зі з’єднанням

| Умова | Можлива причина |
|---|---|
| Швидкість завантаження менше 10 Мбіт/с | Перевантаження мережі, проблеми з маршрутизатором, обмеження тарифу |
| Швидкість вивантаження менше 5 Мбіт/с | Слабкий зворотний канал, провайдерські обмеження |
| Пінг більший за 150 мс | Поганий сигнал, перевантаження, складна маршрутизація |
| Недоступні кілька сайтів | Проблеми у провайдера або блокування ресурсів |
| Велика різниця у результатах тестів протягом дня | Нестабільність лінії або пікові навантаження |

Алгоритм роботи функції:

- користувач надсилає команду /diagnose;

- бот отримує останні результати тестування із бази даних та аналізує їх відповідно до умов;

- бот надсилає користувачу звіт із можливими проблемами або повідомлення, що все в нормі.

Формулювання побудовані так, щоб бути зрозумілими для нефахових користувачів, і не замінюють повноцінну діагностику, а лише допомагають зорієнтуватись у ситуації. Фрагменти коду та приклад роботи бота наведені на рис. 2.10-2.11.

```
def diagnose_issues(history, return_codes=False):
    issues = set()
    for ttype, result, _ in history[:1]:
        if "Завантаження" in result:
            try:
                down = float(result.split("Завантаження:")[1].split("Мбіт")[0].strip())
                if down < 10: issues.add("low_download")
                up = float(result.split("Вивантаження:")[1].split("Мбіт")[0].strip())
                if up < 5: issues.add("low_upload")
                ping = int(result.split("Пінг:")[1].split("мс")[0].strip())
                if ping > 150: issues.add("high_ping")
            except:
                continue
        elif "HTTP статус" in result:
            if "недоступний" in result.lower():
                issues.add("site_unreachable")
    if return_codes:
        return list(issues)
    if not issues:
        return "Стан інтернет-з'єднання виглядає стабільним за основними параметрами."
    mapping = {
        "low_download": "⚠ Низька швидкість завантаження.",
        "low_upload": "⚠ Низька швидкість вивантаження.",
        "high_ping": "⚠ Високий пінг.",
        "site_unreachable": "⚠ Деякі сайти недоступні."
```

Рисунок 2.10 – Фрагмент коду для проведення діагностики

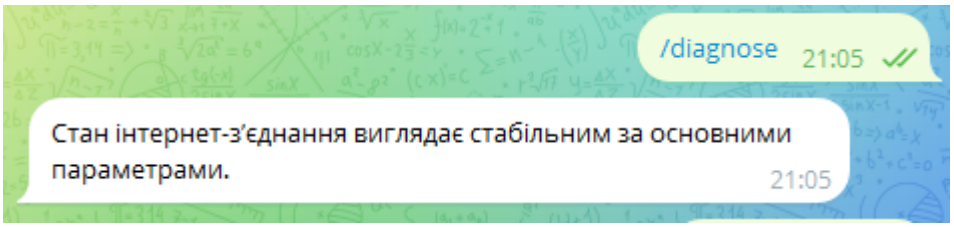


Рисунок 2.11 – Приклад виконання функції діагностики

Таким чином, розроблена функція дозволяє користувачу оперативно отримувати пояснення виявлених проблем із якістю інтернет-з’єднання без необхідності звернення до спеціалістів.

### 2.2.5 Розробка функції рекомендацій з налаштування мережі

Після отримання результатів діагностики користувачу важливо не лише дізнатися про проблему, а й отримати реальні поради щодо її усунення. Для цього в боті реалізовано функцію автоматичного формування рекомендацій.

Функція працює на основі виявлених відхилень у параметрах, аналізує ситуацію і надає відповідні поради. У табл. 2.3 наведено кілька типових ситуацій та відповідні дії рекомендації бота.

Таблиця 2.3 – Перелік проблем і рекомендацій

| Виявлена проблема | Рекомендації |
|---|---|
| Низька швидкість завантаження/ вивантаження | Зменшіть фонове використання мережі. Підключіть пристрій через кабель. Перевірте тариф. |
| Високий пінг | Перемістіть пристрій ближче до роутера. Змініть канал Wi-Fi. |
| Часті втрати з’єднання з сайтами | Спробуйте змінити DNS на 8.8.8.8 або перевірте брандмауер. |

Алгоритм роботи функції рекомендацій:

- користувач надсилає команду /advice;
- бот перевіряє останні результати діагностики;
- залежно від виявленої проблеми, бот надає перелік порад;

- якщо проблем не виявлено, бот пропонує загальні рекомендації для покращення стабільності.

Фрагменти коду та приклад роботи бота наведені на рис. 2.12-2.13.

```
76  def generate_recommendations(codes):
77      if not codes:
78          return ("Проблем не виявлено. Ось кілька загальних порад:\n"
79                  "✅ Перезавантажте роутер.\n"
80                  "✅ Підключайтеся через кабель.\n"
81                  "✅ Уникайте перевантаження мережі у вечірній час.")
82      advices = {
83          "low_download": "✅ Зменшіть фонове використання мережі. Підключіть
            пристрій через кабель.",
84          "low_upload": "✅ Перевірте тариф. Під'єднайтесь до мережі без перешкод.",
85          "high_ping": "✅ Перемістіть пристрій ближче до роутера. Змініть канал
            Wi-Fi.",
86          "site_unreachable": "✅ Спробуйте змінити DNS на 8.8.8.8 або перевірте
            брандмауер."
87      }
88      return "Рекомендації:\n" + "\n".join(advices[c] for c in codes if c in advices)
```

Рисунок 2.12 – Фрагмент коду формування порад

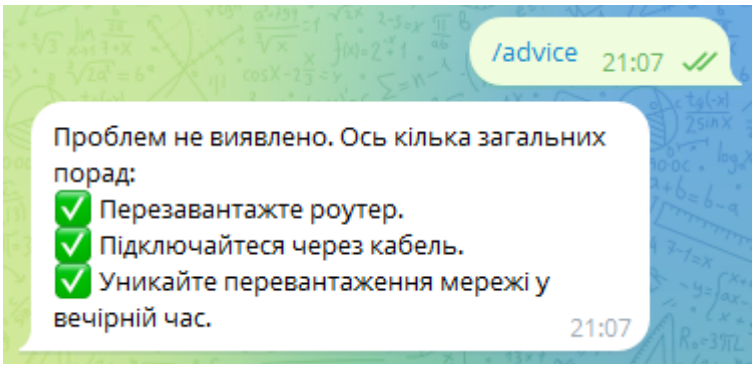


Рисунок 2.13 – Приклад відповіді бота

Рекомендації сформульовані простою, дружньою до користувача мовою. Їхня кількість і варіанти можуть розширюватись у наступних версіях бота. Передбачено як персоналізовані, так і універсальні поради. Такий підхід дає користувачеві відчуття завершеного сервісу: від тестування до пояснення та рекомендацій.

## 2.3 Підключення та налаштування роботи телеграм-бота на хостингу

Для забезпечення постійного доступу до телеграм-бота та його стабільної роботи, програмний продукт було розгорнуто на хмарному хостингу Render.com (рис. 2.14). Ця платформа надає зручні інструменти для автоматичного деплою Python-додатків, підтримує протокол HTTPS за замовчуванням, інтегрується з GitHub [13] і дає змогу швидко розгорнути вебсервіс без потреби налаштування окремого сервера.

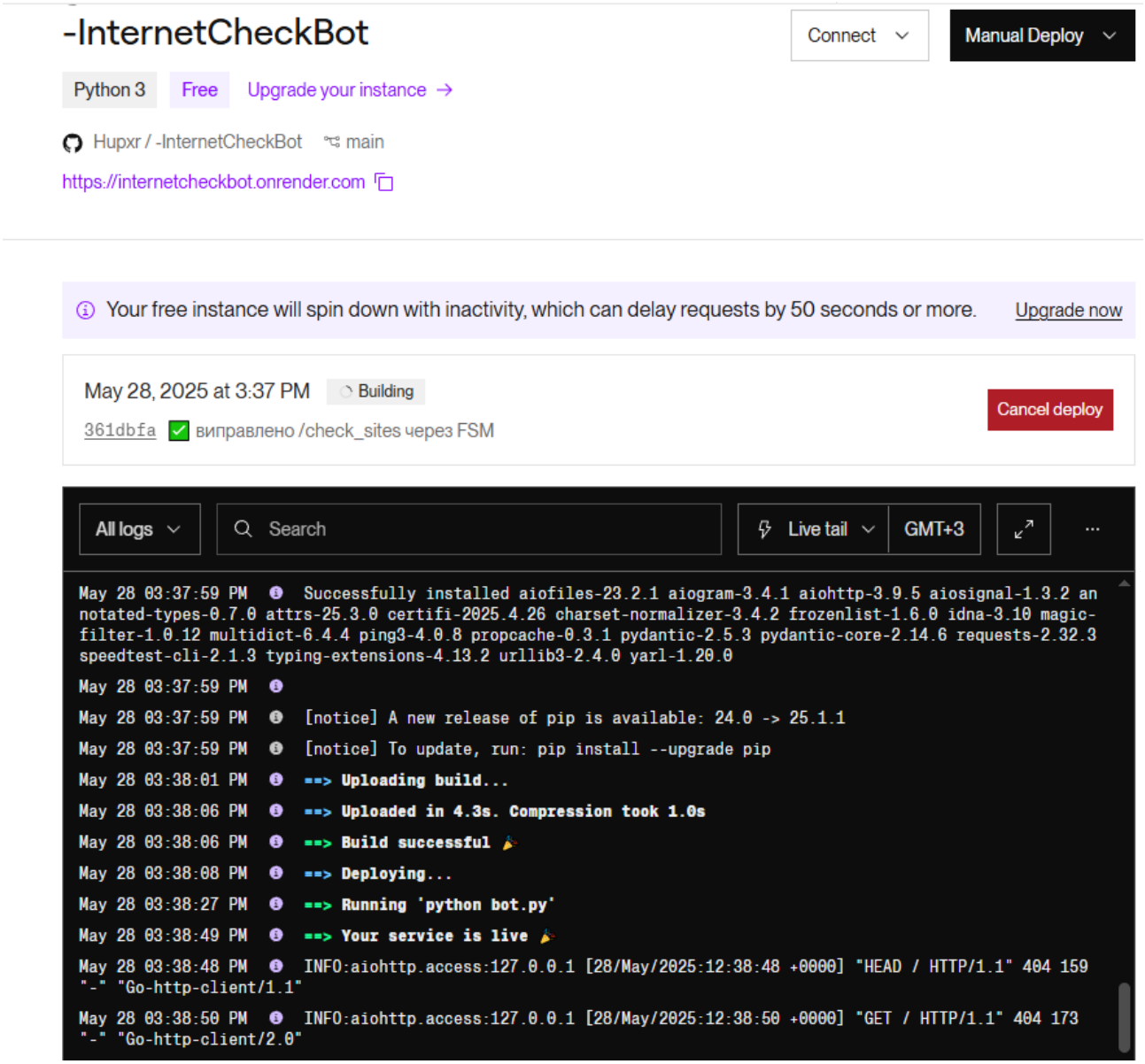


Рисунок 2.14 – Інтерфейс хостингу Render.com

Весь код проєкту зберігається у відкритому репозиторії GitHub (рис. 2.15). Це одна з найпопулярніших платформ для розміщення й керування проєктами, що використовує систему контролю версій Git. Вона дозволяє розробникам фіксувати зміни, впроваджувати нові функції, працювати в команді та, за потреби, повертатися до попередніх версій. Крім того, GitHub дає змогу легко організувати автоматичний деплой, інтегрувати код з хостинг-платформами, такими як Render,  що значно спрощує життєвий цикл розробки [13].

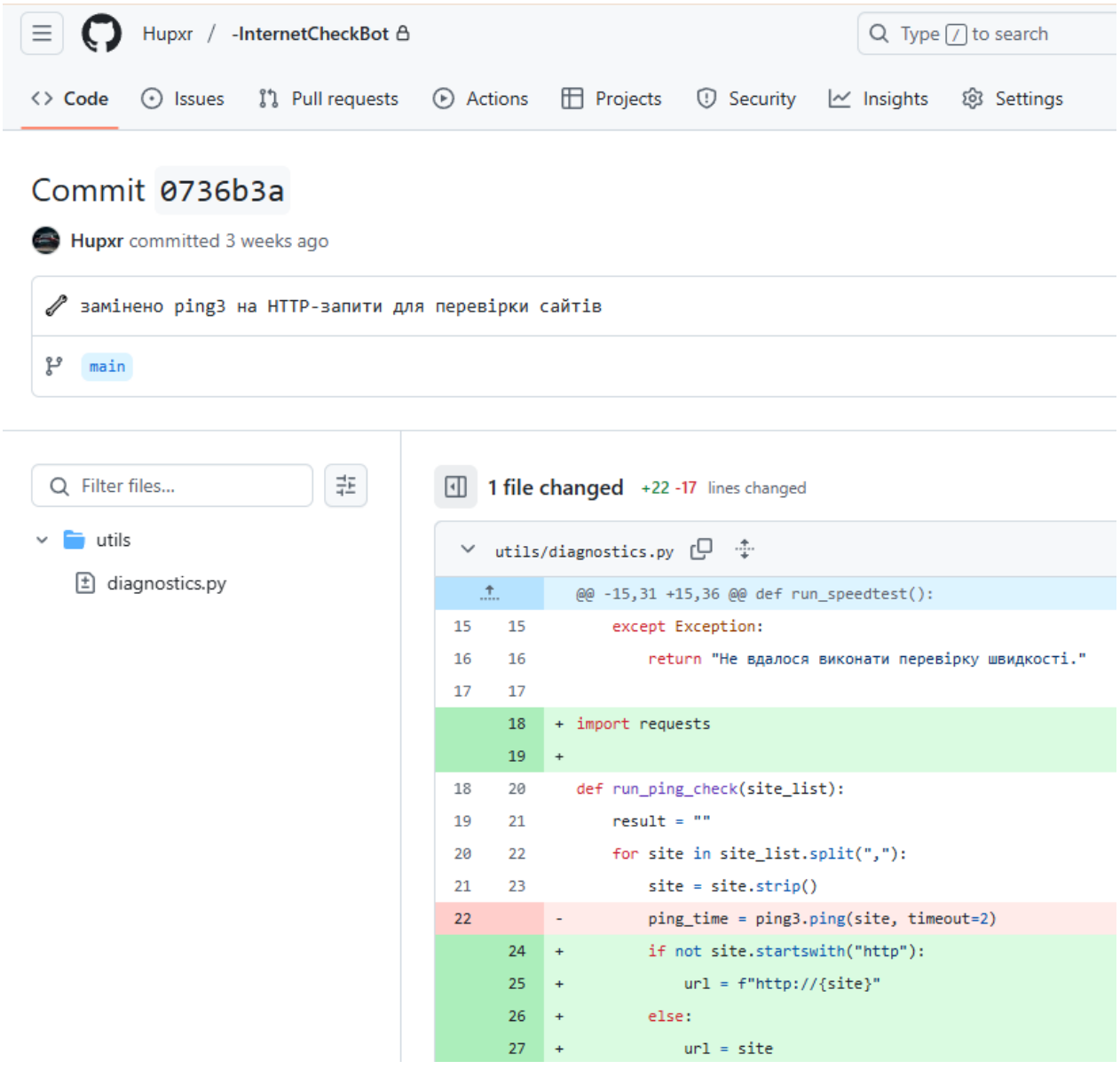


Рисунок 2.15 – Інтерфейс GitHub

Структура репозиторію включає такі основні елементи:

- основний файл запуску bot.py;
- папки handlers (обробники команд) та utils (допоміжні модулі);
- файл requirements.txt із переліком необхідних бібліотек;
- файл бази даних history.db, яка автоматично створюється під час першого запуску.

Вся структура файлів показана на рис. 2.16.

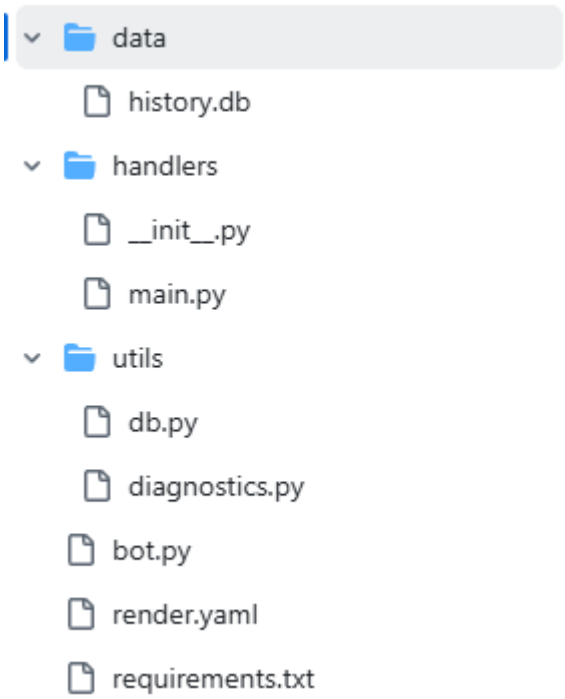


Рисунок 2.16 – Структура файлів проєкту

Render підтримує автоматичну синхронізацію з GitHub. Це означає, що кожного разу після оновлення репозиторію запускається новий цикл збірки й запуску додатку – без потреби ручного втручання.

На хостингу Render.com був підключений власний GitHub-акаунт та створений новий Web Service із середовищем виконання Python. У налаштуваннях було вказано репозиторій із кодом та команду для запуску

програми. Також у налаштуваннях було створено змінну середовища TOKEN, яка містить токен авторизації для Telegram Bot API [11]. Такий підхід дозволяє зберігати чутливу інформацію поза межами вихідного коду.

Render автоматично створює HTTPS-домен, який використовується для підключення Telegram Webhook. Після запуску бот самостійно реєструє свій webhook через Telegram API, завдяки чому отримує повідомлення миттєво після їх надсилання користувачами.

Механізм webhook передбачає, що Telegram надсилає запити до бота на конкретну зареєстровану адресу. У межах проєкту було реалізовано функцію, яка автоматично реєструє webhook, що відповідає домену, виданому Render. Це дозволяє миттєво реагувати на запити користувачів.

## 2.4 Тестування телеграм-бота

Після завершення етапу розробки було проведено повне тестування бота з метою перевірки коректності реалізованих функцій, стабільності його роботи та відповідності фактичної поведінки очікуваним результатам. Для цього було використано метод "чорного ящика", що передбачає перевірку роботи системи без аналізу її внутрішнього коду.

Основною метою тестування було:

- перевірити, як бот реагує на введення команд;
- оцінити, чи правильно працює кожна функція окремо;
- виявити можливі помилки при роботі з Telegram API, мережею або базою даних;
- перевірити, як система поводиться з різними типами вхідних даних, включно з помилковими.

Тестування охоплювало всі реалізовані функції:

- /start – перевірка правильності виводу стартового повідомлення з переліком доступних команд;

- /check_speed – перевірка вимірювання швидкості завантаження, вивантаження, пінгу та правильності збереження результатів;

- /check_sites – тестування перевірки кількох сайтів, аналізу HTTP-відповідей і формування текстового звіту. Особлива увага приділялася обробці неправильних або недоступних доменів;

- /history – тестування коректного виведення історії перевірок на основі Telegram ID користувача;

- /diagnose – перевірка логіки аналізу результатів та формування висновків щодо проблем;

- /advice – тестування формування порад залежно від результатів тестів або надання загальних рекомендацій.

У процесі тестування було виявлено та усунено кілька помилок, зокрема:

- некоректна обробка станів при введенні сайтів;

- відсутність тайм-ауту при HTTP-запитах, що спричиняло зависання;

- виклик асинхронної функції без await, що порушувало порядок виконання коду.

Після усунення цих проблем бот працює стабільно, адекватно реагує на всі передбачені сценарії взаємодії, а реалізовані функції повністю відповідають сформульованим у технічному завданні.

# ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено телеграм-бот, що дозволяє автоматично діагностувати інтернет-з’єднання та значно спрощує цей процес для звичайного користувача. Завдяки цьому користувач може швидко оцінити якість свого підключення без спеціального технічного обладнання або глибоких знань.

На етапі аналітичного дослідження було досліджено основні параметри, що визначають якість з’єднання (швидкість завантаження і вивантаження, пінг, доступність сайтів тощо), а також проаналізовано існуючі онлайн-сервіси. Встановлено, що жоден з них не поєднує повний набір функцій: автоматичний аналіз, збереження історії та формування персоналізованих рекомендацій.

Також було досліджено наявні телеграм-боти, які частково могли б виконувати схожі функції. Однак під час дослідження виявилося, що такі рішення або застарілі, або нерелевантні, що ще раз підтвердило актуальність розробки власного інструменту.

У практичній частині було розроблено телеграм-бота, використовуючи мову програмування Python і фреймворк Aiogram. Зокрема, були реалізовані такі функції:

- перевірка швидкості з’єднання через бібліотеку speedtest-cli;
- перевірка доступності сайтів із використанням ping3 та requests;
- збереження результатів у базі даних SQLite;
- аналіз тестів та виявлення причин проблем;
- формування рекомендацій (персоналізованих і загальних).

Особливу увагу було приділено зручності і зрозумілості інтерфейсу – із поясненнями та структурованими відповідями, що дозволяє комфортно користуватися ботом навіть людям без технічної підготовки.

Бот було задеплоєно на хмарному хостингу Render.com, а весь код розміщено на GitHub. Це дало змогу організувати автоматичне оновлення,

прозору структуру та безпечне зберігання даних. Для підключення до Telegram API використано webhook через HTTPS-домен Render, що гарантує швидку реакцію на запити.

Завершальний етап включав повне тестування функціоналу, під час якого були перевірені всі команди, взаємодія з Telegram API та обробка даних. Усі знайдені помилки були успішно усунені, бот продемонстрував стабільну роботу всіх функцій.

У результаті розроблено повністю готовий програмний продукт, придатний до практичного використання – як у побутових цілях, так і в освітніх чи сервісних проєктах, що потребують швидкої діагностики мережі.

У майбутньому можливе розширення функціоналу: автоматичні періодичні перевірки, додаткові аналітичні можливості та підтримка кількох мов інтерфейсу.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Як забезпечити безперебійний доступ до інтернету: діагностика та вирішення проблем з підключенням [Електронний ресурс]. – Режим доступу: https://support.ua/iak-zabezpechyty-bezperebijnyj-dostup-do-internetu-diahnostyka-ta-vyrishennia-problem-z-pidkliuchenniam/

2. Benefits Of Using Telegram Bot [Електронний ресурс]. – Режим доступу: https://botsify.com/blog/hospitality-industry-using-telegram-bot/

3. Технології створення програмних продуктів та інформаційних систем : навч. посібник / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2017. – 93 с.

4. Python documentation [Електронний ресурс] / Python Software Foundation. – Режим доступу: https://docs.python.org/3/

5. Aiogram Documentation (v3) [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev/en/dev-3.x/

6. Speedtest CLI by Ookla [Електронний ресурс]. – Режим доступу: https://www.speedtest.net/apps/cli

7. Requests: HTTP for Humans [Електронний ресурс]. – Режим доступу: https://docs.python-requests.org/

8. SQLite Documentation [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html

9. Render Docs – Web Services Deployment [Електронний ресурс]. – Режим доступу: https://render.com/docs/web-services

10. How do Chatbots Work? A Guide to Chatbot Architecture [Електронний ресурс]. – Режим доступу: https://marutitech.com/chatbots-work-guide-chatbotarchitecture/

11. Telegram Bot API [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api

12. Aiohttp: Asynchronous HTTP Client/Server for asyncio [Електронний ресурс]. – Режим доступу: https://docs.aiohttp.org/

13. GitHub Docs – Working with repositories [Електронний ресурс]. – Режим доступу: https://docs.github.com/en/repositories