ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему «Чат-бот для технічної підтримки підприємства мовою Python»

на здобуття освітнього ступеня бакалавра

зі спеціальності 126 Інформаційні системи та технології

освітньо-професійної програми Інформаційні системи та технології

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

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

Виконав: здобувач вищої освіти гр. ІСД-41
Дмитро КОВАЛЬЧУК
Ім'я, ПРІЗВИЩЕ

Керівник:
науковий ступінь, вчене звання
Валентина ДАНИЛЬЧЕНКО, PhD
Ім'я, ПРІЗВИЩЕ

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

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК

«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ковальчук Дмитро Олександрович

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Чат-бот для технічної підтримки підприємства мовою Python керівник кваліфікаційної роботи Валентина ДАНИЛЬЧЕНКО, PhD,

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

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» 02 2025р. № 56

2. Строк подання кваліфікаційної роботи «30» 05 2025 р.
3. Вихідні дані до кваліфікаційної роботи: офіційна документація Python, науково-технічна література, дані про існуючі системи оптимізації бізнес-процесів, результати українських та закордонних наукових досліджень у сфері електронної комерції.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз ефективності існуючих рішень електронної оптимізації технічної підтримки
2. Посановка технічного завдання, вибір технологічного стеку, проєктування архітектури, структури, користувацького інтерфейсу системи
3. Розробка чат-бота для підприємства мовою Python

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «24» 02 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	01.03.2025 р.	
2	Дослідження технічних аспектів та ефективності існуючих рішень для оптимізації технічної підтримки	15.03.2025 р.	
3	Вибір технологічного стеку, постановка технічного завдання на розробку	17.03.2025 р.	
4	Проєктування структури, архітектури та користувацького інтерфейсу	4.04.2025 р.	
5	Програмна реалізація та тестування роботи чат-бота	16.05.2025 р.	
6	Розробка демонстраційних матеріалів	26.05.2025 р.	
7	Попередній захист дипломної роботи	27.05.2025 р.	
8	Подання роботи в деканат	30.05.2025 р.	

Здобувач(ка) вищої освіти _____ Дмитро КОВАЛЬЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 61 стор., 36 рис., 2 табл., 20 джерел.

Мета роботи – розробити та впровадити Telegram-бот для автоматизації технічної підтримки підприємства з метою покращення якості обслуговування користувачів, скорочення часу реагування на типові запити та зниження навантаження на персонал.

Об'єкт дослідження – процеси взаємодії між користувачами та службою технічної підтримки в цифровому середовищі.

Предмет дослідження – методи автоматизації технічної підтримки, архітектурні рішення чат-ботів, засоби розробки телеграм-ботів із використанням Telegram Bot API, Telegram Business API, мови PHP та форматів зберігання даних.

Короткий зміст роботи: у роботі проаналізовано сучасні технології автоматизації технічної підтримки та приклади їх реалізації на основі Telegram-ботів. Проведено технічне обґрунтування вибору архітектури та технологічного стеку, до якого увійшли PHP, JSON, Telegram Bot API та Telegram Business API. Представлено реалізацію функціональних модулів системи, включно з алгоритмом обробки запитів, механізмом авто-відповідей та системою адміністрування. Детально описано інтеграцію з Telegram Business API, форматування повідомлень через HTML-теги, використання webhook і особливості JSON-бази даних. Результати тестування показали високу ефективність системи за ключовими показниками часу реакції, точності обробки запитів та стабільності роботи. Система рекомендована до впровадження на підприємствах малого та середнього бізнесу.

КЛЮЧОВІ СЛОВА: TELEGRAM-БОТ, АВТОМАТИЗАЦІЯ, ТЕХНІЧНА ПІДТРИМКА, PHP, TELEGRAM API, TELEGRAM BUSINESS, JSON, WEBHOOK, HTML-ФОРМАТУВАННЯ, ЧАТ-БОТ.

ABSTRACT

Text part of the qualification work for obtaining a bachelor's degree: 61 pages, 36 figures, 2 tables, 20 sources.

The purpose of the work is to develop and implement a Telegram bot to automate the technical support of the enterprise in order to improve the quality of user service, reduce the response time to typical requests, and reduce the workload on staff.

The object of the study is the processes of interaction between users and the technical support service in the digital environment.

The subject of the study is methods of automation of technical support, architectural solutions of chatbots, tools for developing telegram bots using Telegram Bot API, Telegram Business API, PHP language, and data storage formats.

Summary of the work: The paper analyzes modern technologies for automating technical support and examples of their implementation based on Telegram bots. A technical justification for the choice of architecture and technology stack, which includes PHP, JSON, Telegram Bot API and Telegram Business API, is carried out. Telegram Business API, formatting messages through HTML tags, webhook usage, and JSON database features. The test results showed high efficiency of the system in terms of key indicators of response time, accuracy of request processing and stability of operation. The system is recommended for implementation at small and medium-sized businesses.

KEYWORDS: TELEGRAM BOT, AUTOMATION, TECHNICAL SUPPORT, PHP, TELEGRAM API, TELEGRAM BUSINESS, JSON, WEBHOOK, HTML FORMATTING, CHATBOT.

ЗМІСТ

ВСТУП

В умовах цифровізації бізнесу та зростання кількості онлайн-запитів від клієнтів, забезпечення якісної та оперативної технічної підтримки є ключовим фактором конкурентоспроможності підприємств. Автоматизація цього процесу за допомогою чат-ботів дозволяє не лише зменшити навантаження на операторів підтримки, але й забезпечити цілодобову доступність сервісу. Зокрема, популярність месенджера Telegram та поява Telegram Business API відкривають нові можливості для інтеграції бізнес-процесів з каналами спілкування клієнтів, що підсилює актуальність розробки спеціалізованих бот-систем.

Потреба у подібних рішеннях обумовлена кількома факторами. По-перше, користувачі очікують швидкої реакції на запити, що неможливо гарантувати за умов ручної обробки великого обсягу повідомлень. По-друге, автоматизовані системи підтримки дозволяють стандартизувати відповіді, зменшити ймовірність помилок і покращити якість сервісу. По-третє, інтеграція з бізнес-акаунтом Telegram забезпечує підвищену довіру до бота, розширює його функціональні можливості й дозволяє адаптувати рішення до корпоративних потреб.

Метою дослідження є розробка та апробація Telegram-бота для автоматизованої технічної підтримки, який реалізований мовою програмування PHP з використанням Telegram Business API. Проєкт повинен забезпечувати обробку повідомлень, конфігурування шаблонних авто-відповідей через інтерфейс Telegram, збереження даних у форматі JSON та підтримку форматування відповідей у вигляді HTML.

Для досягнення цієї мети було поставлено такі завдання:

- Провести огляд сучасних технологій автоматизації технічної підтримки;
- Проаналізувати існуючі рішення у сфері Telegram-ботів;
- Визначити вимоги до функціоналу та ефективності системи;
- Обґрунтувати вибір архітектурного рішення та технологічного стеку;
- Реалізувати систему з підтримкою Telegram Business API;
- Провести тестування та оцінити ефективність системи.

Об'єктом дослідження є процес автоматизації клієнтської підтримки в цифровому середовищі.
Предметом дослідження — інформаційна система у вигляді Telegram-бота, що інтегрується в бізнес-процеси підприємства.

Практична значущість полягає в тому, що реалізований чат-бот може бути використаний малими та середніми підприємствами для скорочення часу обслуговування клієнтів, підвищення якості підтримки та автоматизації типових запитів. Враховуючи простоту технологічної реалізації, система не потребує значних ресурсів для розгортання та супроводу.

Наукова новизна полягає у поєднанні Telegram Business API, rule-based моделі відповіді та клієнтської конфігурації системи безпосередньо через інтерфейс месенджера. Такий підхід дозволяє підприємствам отримати керовану, стабільну та доступну систему технічної підтримки з мінімальними затратами.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ВИМОГ ДО СИСТЕМИ

1.1 Огляд сучасних технологій для автоматизації технічної підтримки

У останні роки технічна підтримка перестала бути лише засобом розв’язання проблем — вона трансформувалась у стратегічний інструмент побудови довготривалих відносин із клієнтами та зміцнення лояльності до бренду. Згідно з даними Fluent Support [1], 83% клієнтів відчувають більшу лояльність до брендів, які прислухаються до них і ефективно вирішують скарги. Це свідчить про критичну роль якісної підтримки у сучасному бізнесі.

Актуальність автоматизації технічної підтримки зумовлена високими очікуваннями користувачів щодо швидкості та якості обслуговування. 90% клієнтів вважають швидку відповідь критично важливою, при цьому 60% очікують реакцію протягом 10 хвилин. У той час як 77% споживачів розраховують на негайний зв’язок із представником компанії, лише 12% самостійних платформ підтримки мають високий рівень інтеграції. Це створює розрив між очікуваннями клієнтів та реальними можливостями компаній, який автоматизовані рішення покликані подолати (рис. 1.1).

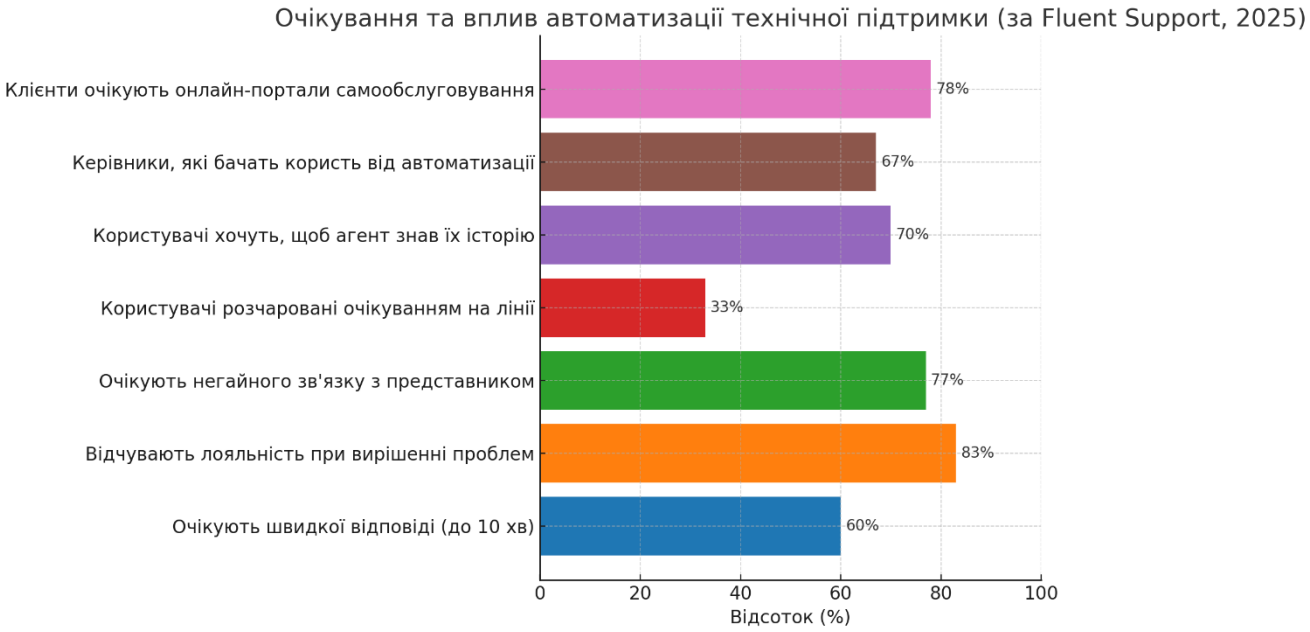


Рис. 1.1 Очікування клієнтів та вплив автоматизації технічної підтримки

Автоматизація підтримки, зокрема через чат-боти, дозволяє забезпечити цілодобову доступність, пришвидшити обробку запитів і зменшити навантаження на операторів. 67% керівників підтримки вже зазначають позитивний ефект від впровадження автоматизованих рішень, зокрема чат-ботів та автоматичного маршрутизування запитів. Такі технології дозволяють ефективно обробляти типові звернення, тоді як більш складні передаються людським агентам.

Важливо, що автоматизація не тільки підвищує ефективність, а й сприяє зменшенню стресу у споживачів. Близько 33% клієнтів найбільше дратує очікування на лінії, а ще 33% — необхідність повторювати одну й ту саму інформацію різним агентам. Технології, що інтегрують базу знань і клієнтську історію, дозволяють уникнути таких ситуацій. Наприклад, 70% споживачів очікують, що кожен оператор буде знати історію їхніх покупок та звернень.

Незважаючи на переваги автоматизації, її ефективність прямо залежить від правильної реалізації. Хоча 78% споживачів очікують наявність онлайн-порталів самообслуговування, менше третини компаній надають якісний доступ до баз знань. Також 40% користувачів у США використовують щонайменше три канали зв’язку з підтримкою, що вимагає від підприємств гнучких мультиканальних рішень, які легко масштабуються.

Інвестиції в автоматизовані системи технічної підтримки виявляються стратегічно виправданими. За даними Salesforce, 88% успішних компаній інвестують у навчання агентів та впровадження нових технологій, у той час як лише 57% компаній з нижчими результатами роблять те саме. Це підкреслює, що автоматизація має йти пліч-о-пліч із розвитком персоналу та клієнтоорієнтованої культури [1].

Сучасні технології — від базових чат-ботів до складних систем автоматичного маршрутизування і самообслуговування — стали невід’ємною частиною конкурентної переваги підприємств. Їх впровадження не тільки відповідає запитам ринку, а й сприяє підвищенню задоволеності клієнтів, зменшенню витрат і забезпеченню довготривалого зростання бізнесу.

Штучний інтелект відіграє дедалі важливішу роль в автоматизації технічної підтримки. Згідно з аналітичним звітом Zendesk, 90% лідерів у сфері клієнтського досвіду (CX Trendsetters) вважають, що понад 80% звернень користувачів буде вирішуватись без участі людини вже найближчими роками [2]. Це підтверджує тенденцію до широкого впровадження автономних AI-агентів, які здатні не лише розпізнавати наміри користувача за допомогою NLP, але й самостійно ініціювати вирішення проблеми або переадресацію до оператора.

Такий підхід дозволяє бізнесу забезпечити цілодобову підтримку, зменшити витрати та підвищити задоволення клієнтів, що особливо актуально в умовах зростаючих вимог до швидкості та персоналізації обслуговування.

У сфері автоматизації технічної підтримки сформувалась комплексна класифікація рішень, що ґрунтується на функціональному призначенні, рівні автономності та способі взаємодії з користувачем (рис. 1.2).

Рис. 1.2 Класифікація рішень автоматизації технічної підтримки [3]

Одним із ключових інструментів є AI-чат-боти — системи, які використовують штучний інтелект і обробку природної мови для обслуговування

клієнтів у режимі 24/7. Вони здатні миттєво відповідати на запити, ідентифікувати контекст звернення та автоматично переадресовувати складні питання оператору [3]. Наприклад, компанія H&M застосовує чат-ботів для допомоги покупцям у пошуку товарів і уточненні розмірів, що дозволяє значно скоротити навантаження на персонал і підвищити зручність покупок [4].

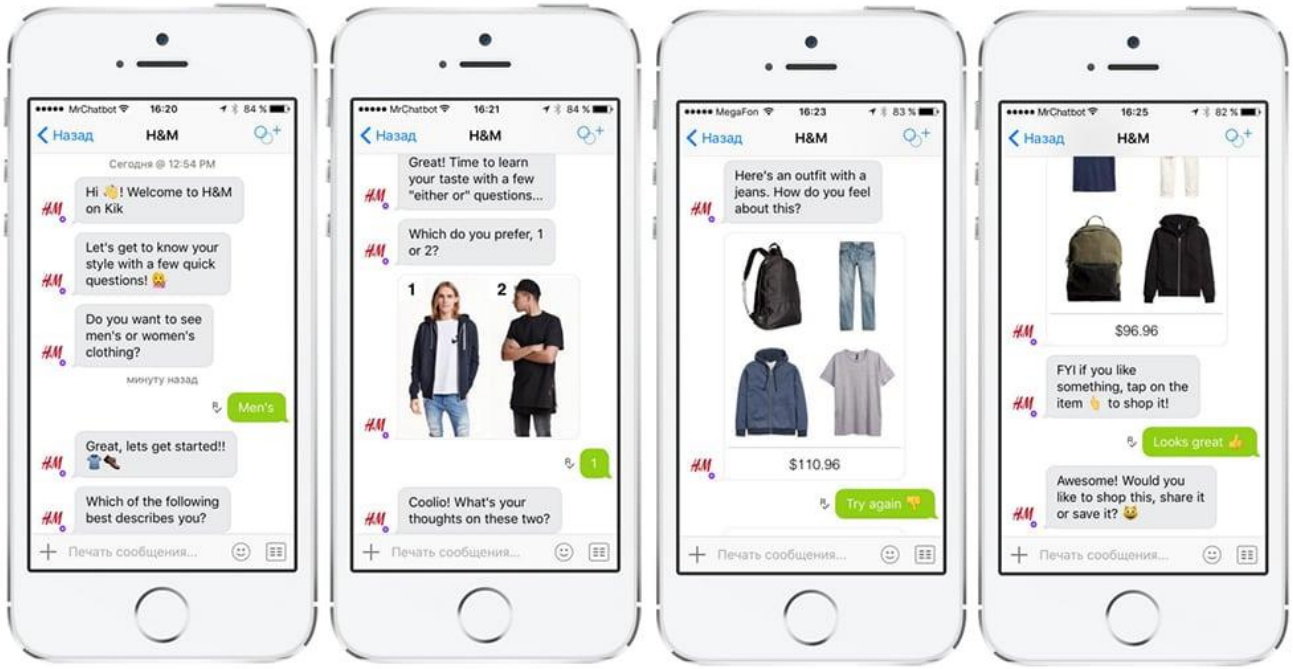


Рис. 1.3 Користувацький інтерфейс чатбота H&M [4]

Системи тікетів є основою структурованої технічної підтримки: вони автоматично створюють звернення, класифікують їх, призначають відповідальним працівникам та відслідковують хід вирішення. Наприклад, Microsoft використовує автоматизовану систему тікетів для управління технічними запитами, що дозволяє оперативно реагувати на критичні інциденти. Аналогічно працює Jira Service Management, який автоматично розподіляє тікети за типом проблеми (рис. 1.4).

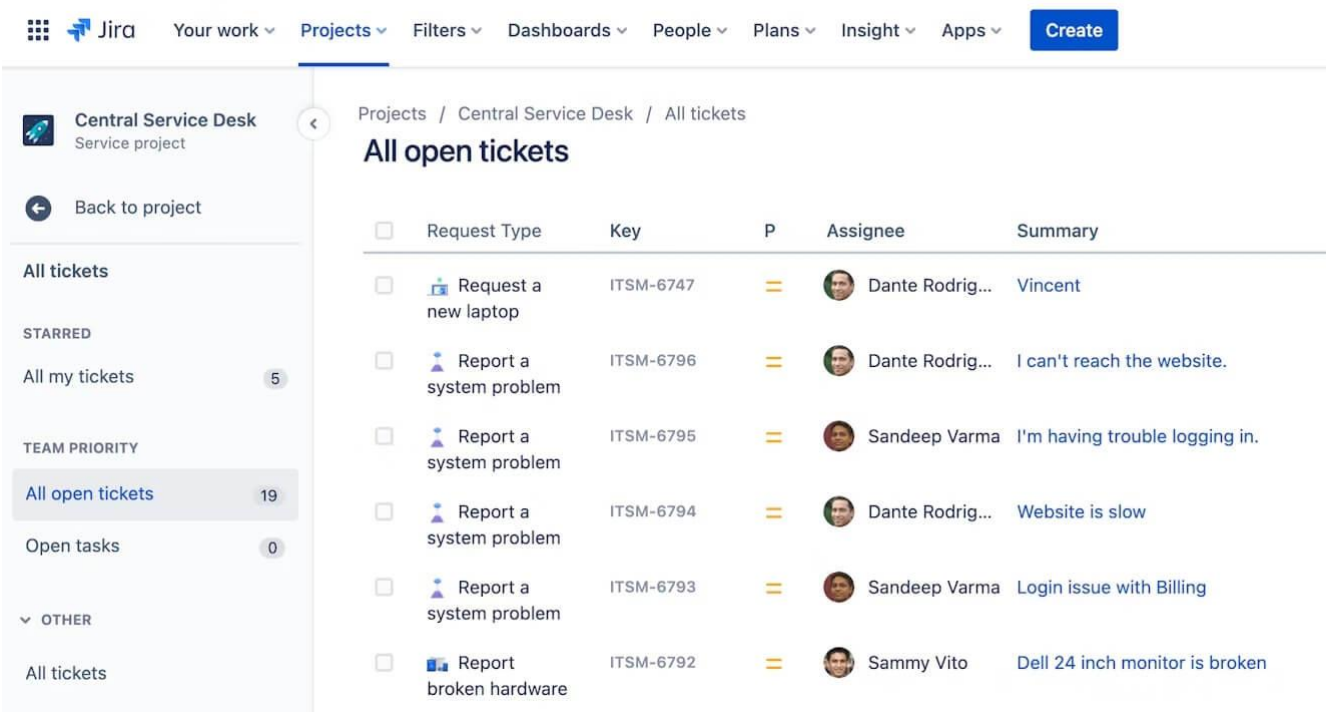


Рис. 1.4 Система тікетів Jira [5]

Важливим інструментом є бази знань — електронні довідкові системи, що містять інструкції, відповіді на поширені запитання та керівництва з вирішення типових проблем. Spotify, наприклад, пропонує користувачам детальні мультимедійні гіди з усунення несправностей, що дозволяє суттєво зменшити кількість звернень до служби підтримки (рис. 1.5).

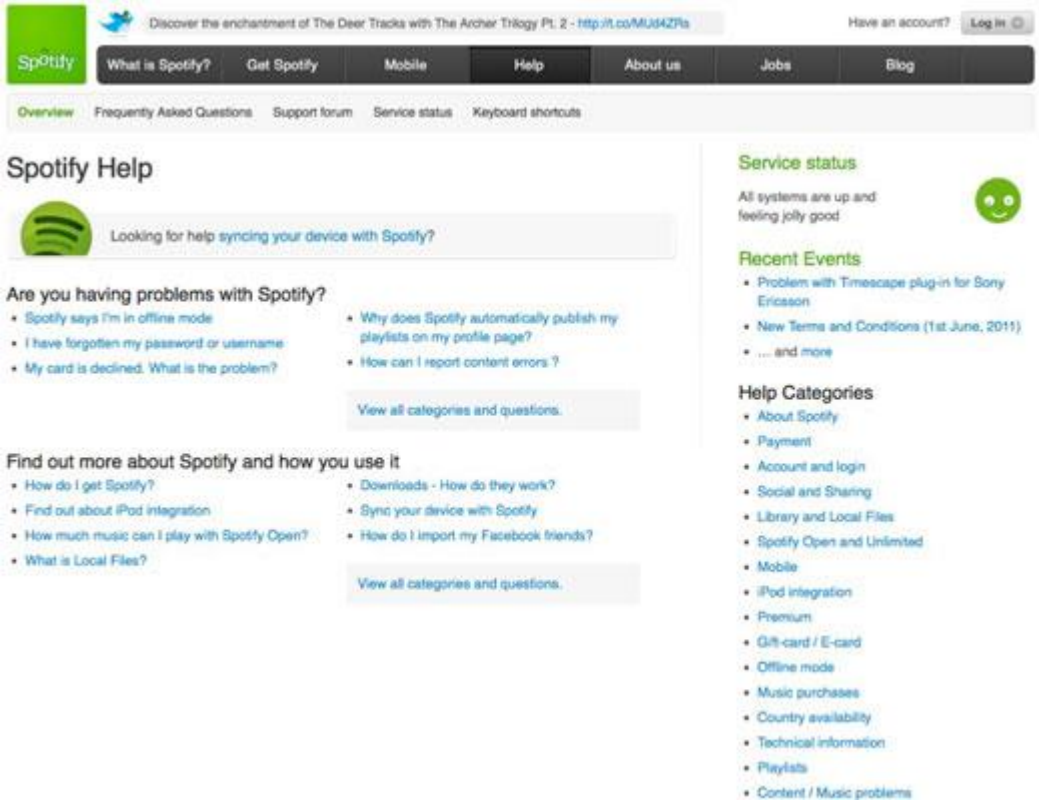


Рис. 1.5 Користувацькі інструкції Spotify [6]

Автоматизовані email-системи забезпечують надсилання повідомлень на основі тригерів — наприклад, після здійснення покупки або при зміні статусу запиту. Компанія Amazon широко застосовує такі системи для інформування клієнтів про доставку, підтвердження замовлень та рекомендації товарів, що підтримує постійний зв'язок з користувачами без залучення операторів (рис. 1.6).

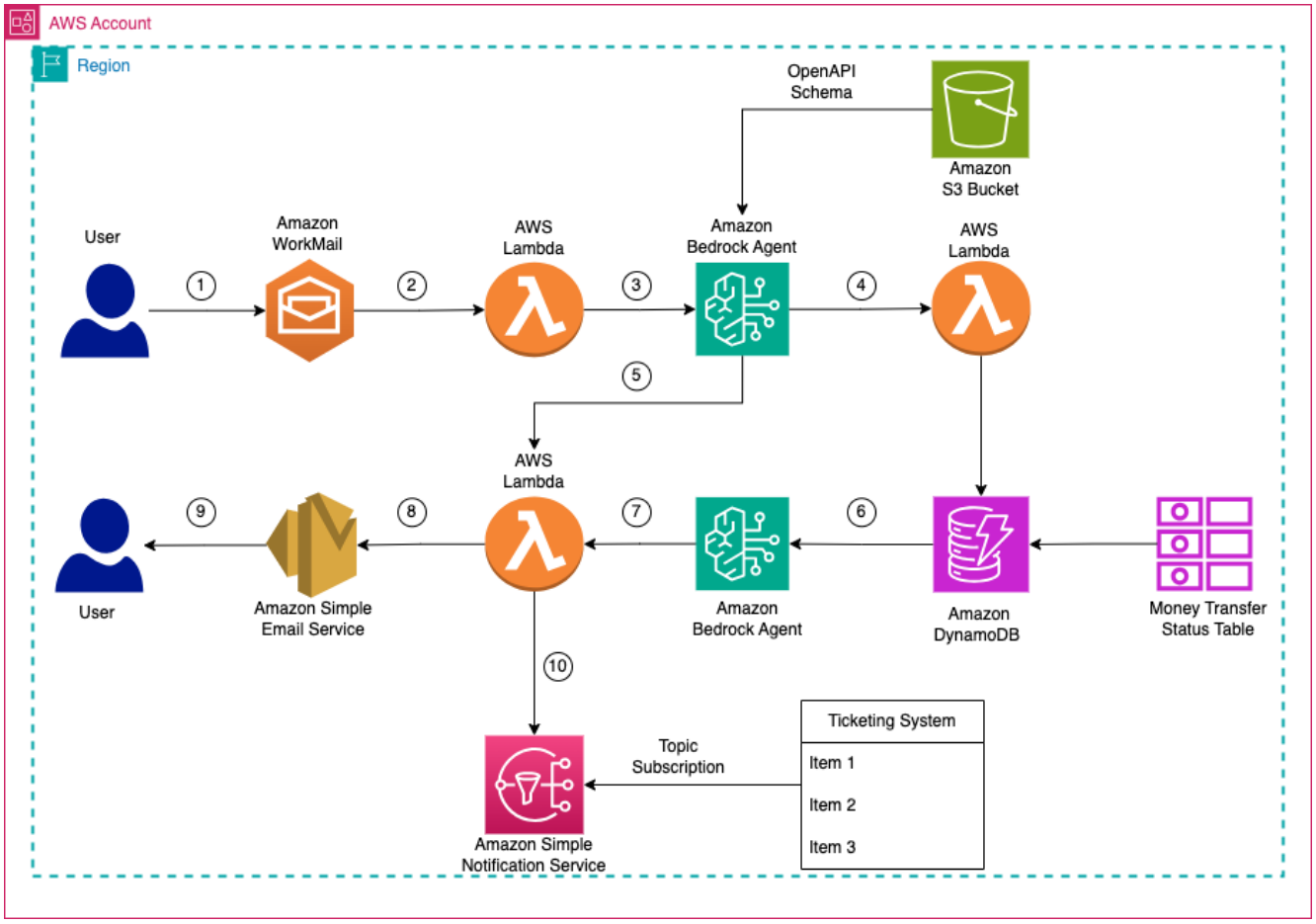


Рис. 1.6 Автоматизація обігу електронних листів у Amazon [7]

Голосові IVR-системи з підтримкою розпізнавання мовлення дозволяють автоматично маршрутизувати дзвінки, проводити аутентифікацію користувачів та надавати голосові відповіді на стандартні запити. American Express використовує таку систему для голосової аутентифікації, що підвищує як безпеку, так і швидкість обслуговування.

Для обробки звернень у соціальних мережах використовуються спеціалізовані платформи, що дозволяють моніторити запити, аналізувати настрій користувачів (sentiment analysis) і забезпечувати своєчасні відповіді. Nike, зокрема, застосовує інструмент Sprout Social для керування підтримкою у соціальних каналах.

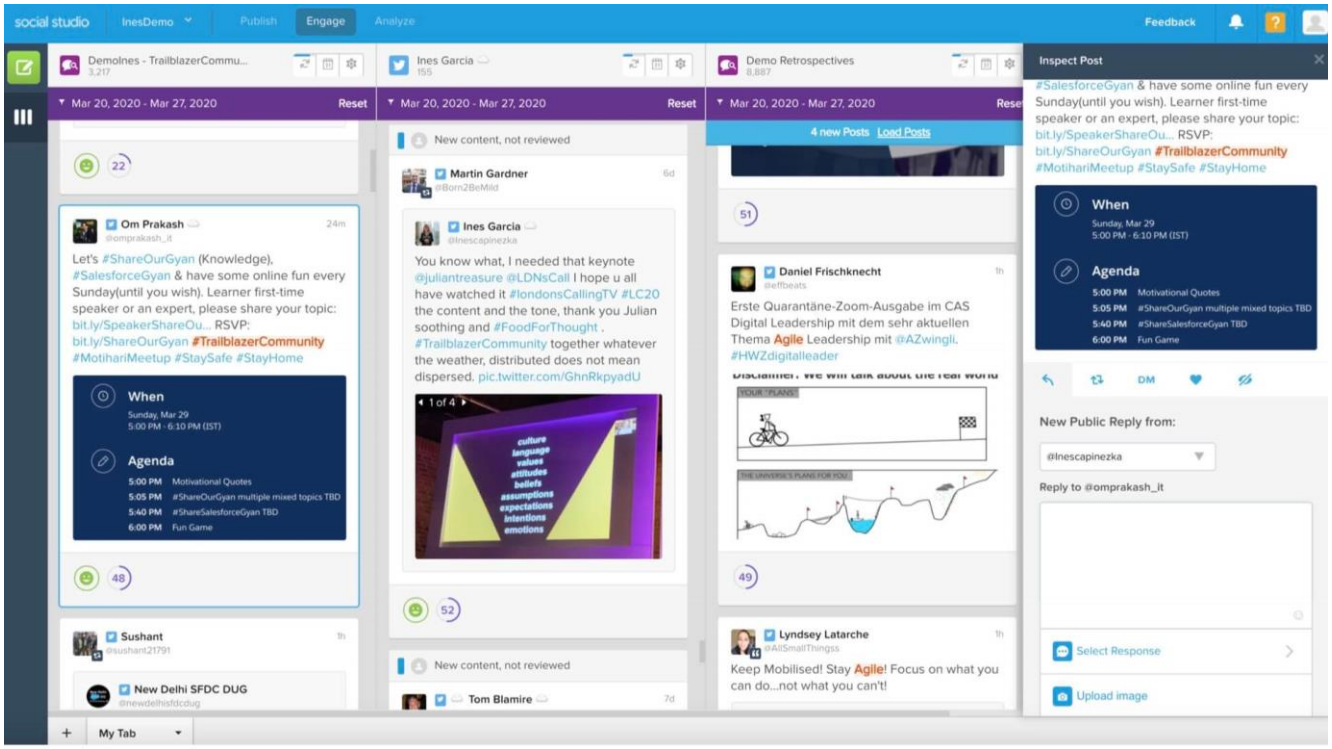


Рис. 1.7 Система Nike Sprout Social [8]

Системи аналітики на базі штучного інтелекту дають змогу виявляти тренди, прогнозувати проблеми та генерувати стратегічні рішення на основі даних. Наприклад, Salesforce Einstein аналізує взаємодії з клієнтами, щоб покращити якість обслуговування та попередити негативний досвід користувачів.

Портали самообслуговування об'єднують декілька сервісів в одному інтерфейсі, дозволяючи клієнтам самостійно створювати звернення, перевіряти статуси, змінювати налаштування профілю тощо. Яскравим прикладом є центр вирішення проблем PayPal, який надає користувачам усі необхідні інструменти для самостійного вирішення фінансових питань без залучення служби підтримки (рис. 1.8).

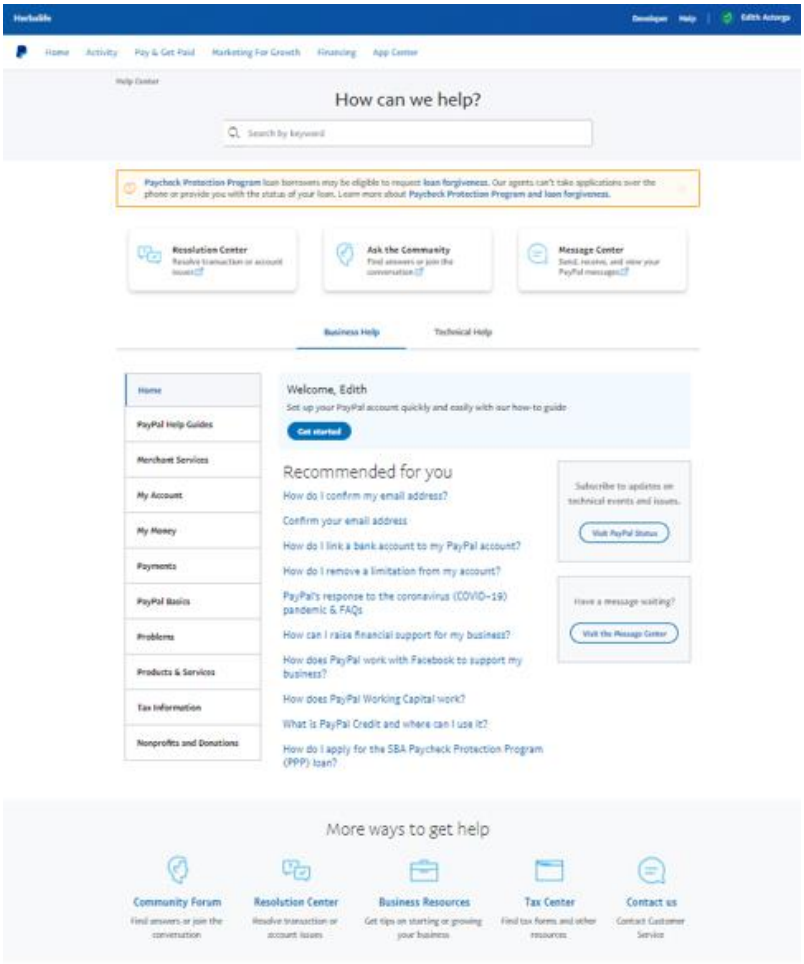


Рис. 1.8 Центр вирішення проблем PayPal [9]

Сучасні рішення автоматизації охоплюють усі ключові канали взаємодії з клієнтом, забезпечуючи швидкий, ефективний та персоналізований сервіс, а успішні приклади впровадження доводять їх практичну доцільність у різних галузях.

Розробка сучасних систем автоматизації технічної підтримки базується на взаємодії кількох технологічних шарів, кожен із яких виконує специфічну функцію в архітектурі рішення (рис. 1.9).

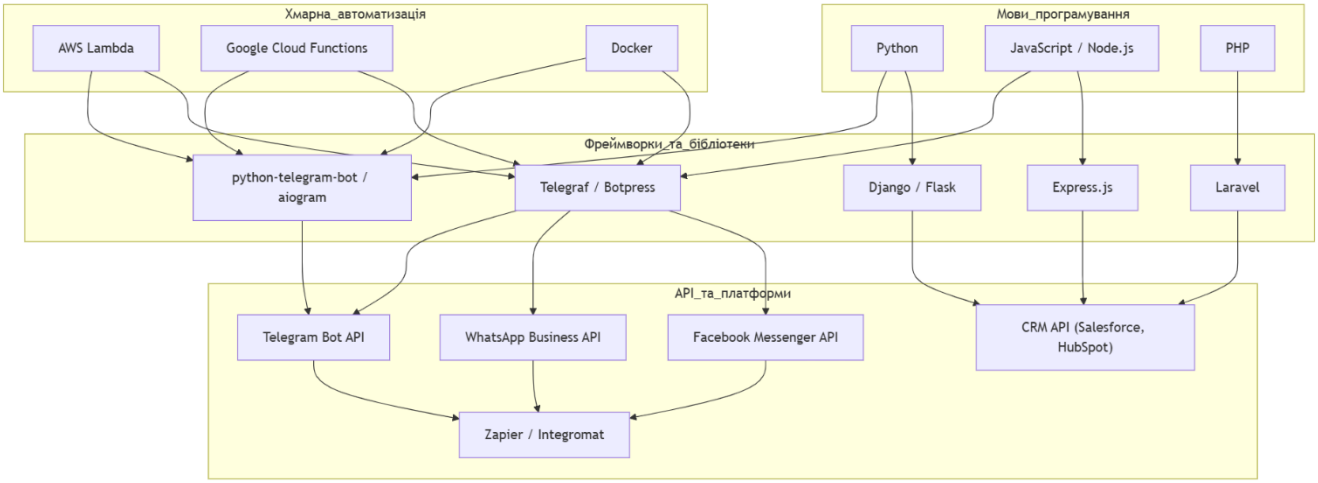


Рис. 1.9 Схема технологічного стеку розробки рішень автоматизованої технічної підтримки

На найвищому рівні знаходяться мови програмування, які забезпечують основну логіку системи. Найчастіше для створення таких систем використовуються Python, JavaScript (Node.js) та PHP. Python забезпечує потужні можливості для обробки природної мови, роботи з API та побудови бекенду (через фреймворки Django або Flask). JavaScript у поєднанні з Node.js часто використовується для побудови високонавантажених асинхронних систем з використанням фреймворку Express.js. Для PHP традиційним середовищем залишається Laravel.

Наступний шар становлять фреймворки та бібліотеки, які безпосередньо реалізують логіку чат-бота та взаємодію з користувачем. У Python-середовищі популярними бібліотеками для роботи з Telegram є python-telegram-bot і aiogram. У світі Node.js поширеним фреймворком є Telegraf, тоді як для більш складних систем з інтерфейсом адміністратора часто застосовується Botpress — платформа з графічним редактором логіки та інтеграцією з базами знань.

Важливою частиною є API та зовнішні платформи, які забезпечують доступ чат-ботів до месенджерів та корпоративної інфраструктури. Для обміну повідомленнями використовуються Telegram Bot API, WhatsApp Business API та Facebook Messenger API. Для обміну даними з CRM-системами часто застосовуються CRM API, зокрема Salesforce або HubSpot API, які дозволяють автоматично створювати, оновлювати або витягувати інформацію про клієнтів.

Для гнучкої інтеграції між сервісами використовуються інструменти Zapier або Integromat (Make), які діють як проміжна ланка між ботами, API месенджерів і корпоративними сервісами без потреби у прямому програмуванні.

Нижній інфраструктурний рівень представлений засобами хмарної автоматизації та розгортання. Серед найпоширеніших рішень — AWS Lambda, Google Cloud Functions та Docker. Хмарні функції дозволяють обробляти запити ботів на основі подій, що зменшує витрати на інфраструктуру. Використання Docker забезпечує ізольоване середовище для масштабування та зручне розгортання бот-сервісів у будь-якому хмарному середовищі.

Узгоджена взаємодія між цими технологічними компонентами дозволяє створювати масштабовані, гнучкі та ефективні системи технічної підтримки, які працюють у режимі 24/7, інтегруються з різноманітними каналами зв'язку й забезпечують високий рівень обслуговування користувачів.

1.2 Аналіз існуючих рішень та підходів у сфері чат-ботів

У рамках роботи доцільно зосередити увагу на аналізі тих рішень, що максимально наближені за функціональним призначенням, середовищем і контекстом використання до розробленої системи. З огляду на це, для порівняльного аналізу було обрано два Telegram-боти — @ComfyuaBot та @FoxtrotInfo_bot, які функціонують як інструменти технічної підтримки та інформаційного супроводу клієнтів у сфері роздрібної електронної комерції. Обидва сервіси реалізовані у середовищі Telegram, що повністю відповідає середовищу цільової розробки дипломного проєкту, і надають набір типових функцій, характерних для технічної підтримки — таких як відповіді на часті запитання, консультації щодо замовлень, навігація по послугах компанії тощо.

Вибір саме цих двох рішень обумовлений схожістю предметної області: обидва боти належать великим українським торговельним мережам — COMFY та Foxtrot — і слугують прикладом ефективного впровадження автоматизованої підтримки в електронній торгівлі. Аналіз цих рішень дозволяє виявити практичні

підходи до організації діалогу з користувачем, обробки типових запитів, реалізації логіки взаємодії та використання Telegram як каналу технічної підтримки. Такий порівняльний огляд дає змогу краще зрозуміти актуальні тенденції у побудові чат-ботів для бізнесу й оцінити, наскільки розроблене рішення відповідає сучасним вимогам.

Чат-бот компанії COMFY реалізований як автоматизований канал клієнтської підтримки у месенджері Telegram та має розгалужену структуру взаємодії з користувачем. Із перших повідомлень бот орієнтований на ідентифікацію користувача через підтвердження номеру телефону (рис. 1.10).

Рис. 1.10 Інтерфейс чат-боту магазину Comfy [10]

Верифікація здійснюється шляхом надсилання SMS-коду, який вводиться вручну у вікно чату. Такий підхід дає змогу реалізувати прив’язку до облікового запису користувача та персоналізувати подальші функції, зокрема перевірку бонусного балансу та статусів замовлень.

Після успішної авторизації бот пропонує інтерфейс у вигляді меню з клавішами швидкого доступу до основних функцій. Основна навігація реалізована через інлайн-кнопки, які дозволяють користувачеві обирати потрібну дію без введення текстових команд. Меню включає такі опції, як перегляд статусу замовлення, перевірка бонусів, доступ до купонів і акцій, перехід на офіційний сайт або у мобільний додаток, а також можливість звернення до оператора.

З технічної точки зору, архітектура бота поєднує модуль автентифікації користувача, базу інтерактивних довідкових функцій та можливість ескалації запитів до живого оператора. Передбачена багаторівнева логіка обробки запитів, що дає змогу зменшити кількість звернень до контактного центру, автоматизувавши стандартні запити. Кнопки динамічно формуються залежно від контексту запиту користувача, що вказує на використання шаблонів сценаріїв у межах логіки чат-бота.

Рис. 1.11 наочно демонструє типовий користувацький шлях у чат-боті COMFY, реалізованому у месенджері Telegram.

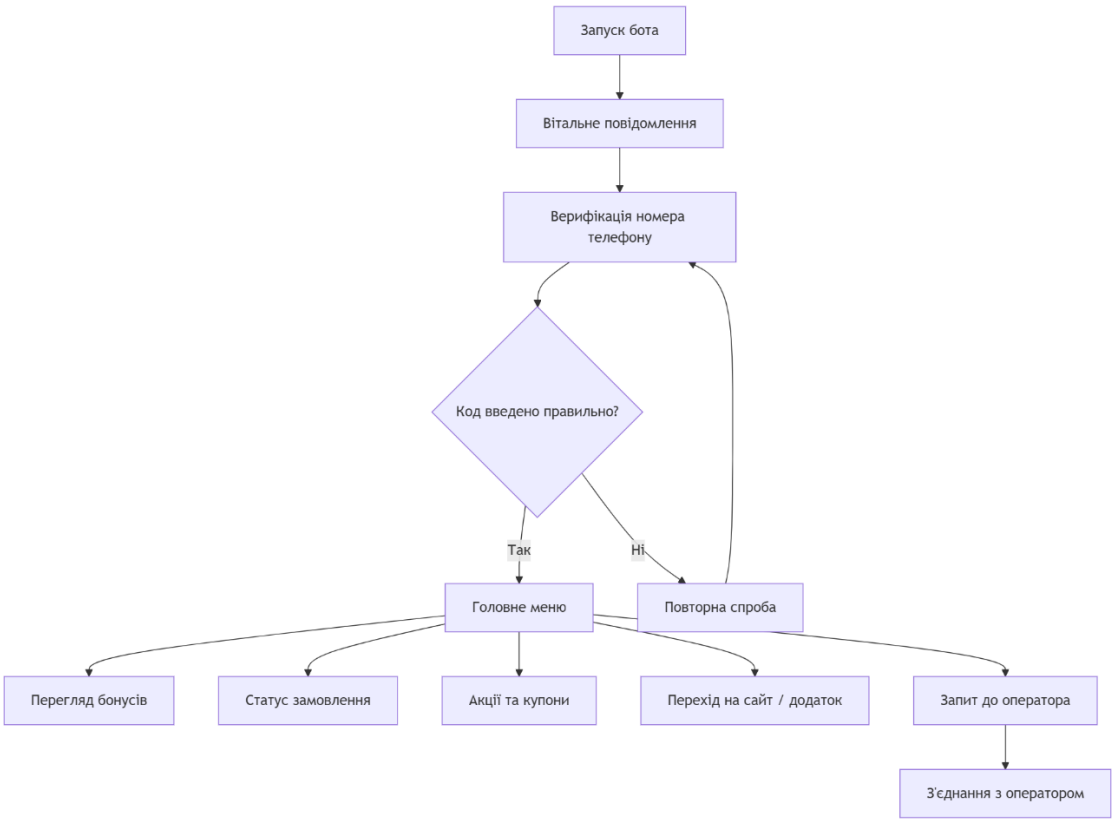


Рис. 1.11 Користувацький шлях чат-боту магазину Comfy

Взаємодія розпочинається із запуску бота, після чого користувач отримує вітальне повідомлення з базовою інформацією. Наступним кроком є верифікація номера телефону за допомогою SMS-коду. У разі правильного введення коду користувач отримує доступ до головного меню, яке містить декілька основних напрямів навігації: перегляд бонусного рахунку, статус замовлення, акційні пропозиції, доступ до мобільного додатку або офіційного сайту, а також можливість звернення до оператора.

Якщо ж верифікаційний код введено некоректно, передбачено гілку повторної спроби, що забезпечує додаткову зручність та безперервність взаємодії. У межах основного меню реалізоване як інформування, так і прямий канал підтримки через форму запиту до оператора, який веде до реального діалогу. Така побудова користувацького маршруту свідчить про багаторівневу логіку та гнучкість у роботі з різними типами запитів.

Загалом реалізація чат-бота COMFY відповідає сучасним підходам до побудови інтерактивних систем підтримки у сфері e-commerce. Поєднання авторизації, інформування, навігації та можливості зворотного зв’язку з оператором свідчить про комплексне використання можливостей Telegram API та орієнтацію на покращення клієнтського досвіду.

Чат-бот мережі Foxtrot реалізований як інтерактивний інструмент клієнтської підтримки та інформування у месенджері Telegram. Основна функціональність бота полягає у швидкому доступі до довідкової інформації, персоналізованих розділів кабінету клієнта, сервісів самостійного обслуговування та каналу зв’язку з оператором. Стартова взаємодія з користувачем відбувається через привітальне повідомлення з коротким переліком можливостей, серед яких — перегляд історії покупок, перевірка бонусів, а також інформація про оплату, доставку та гарантії (рис. 1.12).

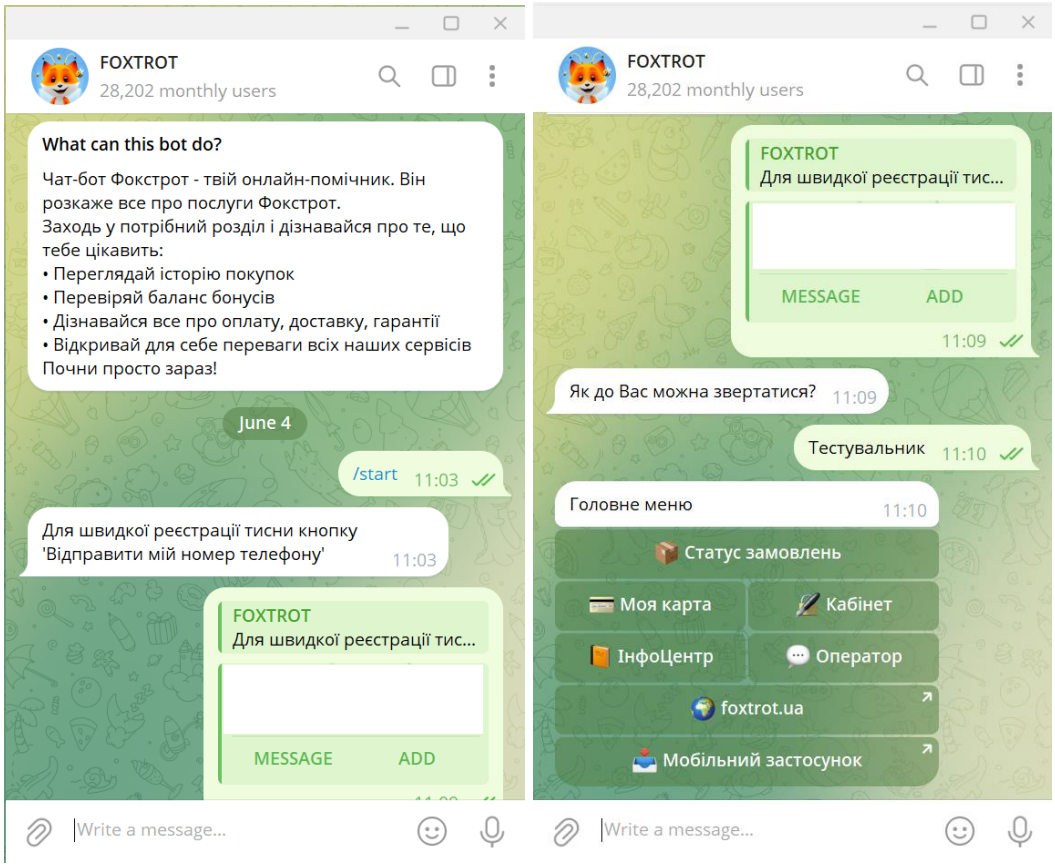


Рис. 1.12 Користувацький інтерфейс чат-боту магазину Фокстрот [11]

Ініціація сесії передбачає спрощену форму реєстрації, яка здійснюється шляхом натискання кнопки "Надіслати мій номер телефону". Така інтеграція з Telegram API забезпечує зручний і швидкий процес автентифікації без потреби у ручному введенні номера. Наступним кроком бот запитує ім’я користувача, що може використовуватись для персоналізації звернень. Після реєстрації відкривається головне меню, представлене інлайн-кнопками з доступом до основних сервісів: перевірки статусу замовлень, перегляду особистої картки, кабінету, інфоцентру, оператора, переходу на офіційний сайт або до мобільного застосунку.

Архітектурно бот орієнтований на швидке надання структурованої інформації за запитом користувача з використанням кнопкового інтерфейсу. Кожен пункт меню передбачає подальшу навігацію або відкриття окремого діалогу (наприклад, з оператором). Очевидно, що логіка побудована за сценарієм "меню-

відповідь", без ознак складної контекстної обробки вільного тексту, що свідчить про rule-based підхід до реалізації.

Бот демонструє ефективну реалізацію базових сервісних функцій: швидку реєстрацію, зручну навігацію, адаптивність під типові запити клієнтів. Такий формат є оптимальним для торговельних мереж, де підтримка має бути доступною, стандартизованою та водночас персоналізованою за рахунок авторизації користувача.

На рис. 1.13 показано користувацький шлях чат-боту магазину Фокстрот.

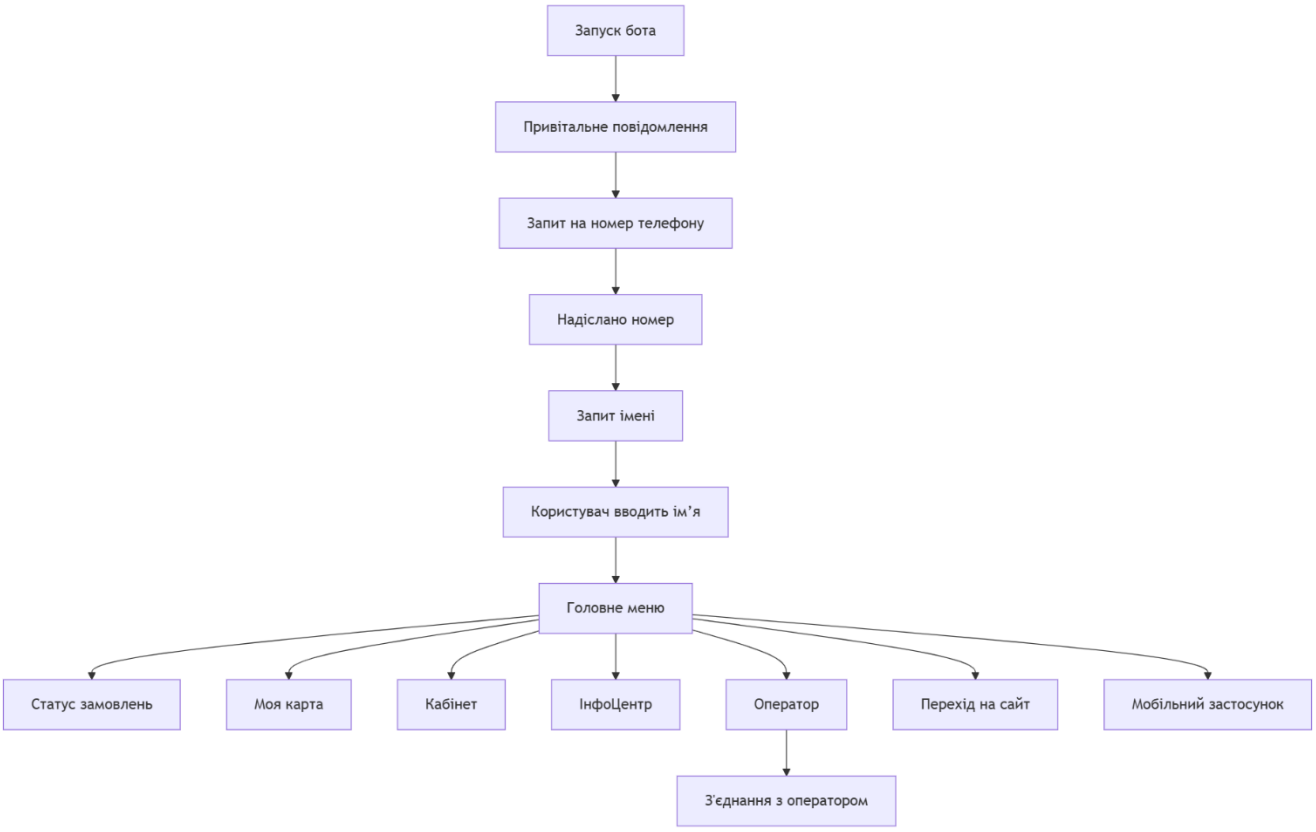


Рис. 1.13 Користувацький шлях чат-боту магазину Фокстрот

Схема відображає логіку користувацького інтерфейсу чат-бота Foxtrot у Telegram. Взаємодія починається зі стартового повідомлення, яке супроводжується запитом на надання номера телефону. Після автоматичної передачі контактної інформації, бот уточнює ім'я користувача, що дозволяє персоналізувати подальшу взаємодію. Усі етапи попередньої реєстрації є обов'язковими для переходу до функціонального меню.

Головне меню складається з кнопок швидкого доступу до основних сервісів: перегляду статусу замовлень, інформації по карті клієнта, переходу до особистого кабінету, інфоцентру, можливості зв’язку з оператором, а також посилань на сайт компанії та мобільний застосунок. Окрема гілка веде до ініціації чату з оператором, що реалізує сценарій ескалації запитів. Бот охоплює як інформаційно-довідкову, так і сервісну функціональність, забезпечуючи швидкий доступ до ключових послуг компанії Foxtrot.

Результати проведеного тестування двох чат-ботів узагальнено у табл. 1.1.

Таблиця 1.1

Результати порівняльного тестування чат-ботів Comfy та Фокстрот

Критерій / Тест-кейс	**COMFY Bot**	**Foxtrot Bot**
Платформа	Telegram	Telegram
Авторизація користувача	SMS-код (4 цифри) після введення номера телефону	Передача номера через Telegram-кнопку
Персоналізація	Не використовується ім’я, персоналізація через акаунт	Запитує ім’я користувача для звернення
Інтерфейс	Інлайн-кнопки у головному меню	Інлайн-кнопки у головному меню
Функції головного меню	Бонуси, замовлення, акції, сайт, оператор, додаток	Замовлення, карта, кабінет, сайт, застосунок, оператор
Запит до оператора	Пряма кнопка, автоматичне з’єднання	Пряма кнопка, перехід до чату з оператором
Тип реалізації	Rule-based із динамічною побудовою кнопок	Rule-based сценарії з фіксованим меню
Швидкість входу до функціоналу	Середня (вимагає ручного введення коду)	Висока (достатньо натиснути кнопку)
Відповідність потребам e-commerce	Висока	Висока
Гнучкість взаємодії	Висока: кілька гілок сценаріїв, авторизація, переадресація	Помірна: сценарій меню-відповідь, немає вільного вводу

Чат-боти COMFY і Foxtrot мають багато спільного в архітектурі та сценаріях взаємодії. Основна відмінність полягає в механізмі авторизації: COMFY передбачає введення SMS-коду, тоді як Foxtrot використовує швидку Telegram-автентифікацію. Крім того, Foxtrot реалізує початкову персоналізацію через запит імені користувача. Обидва рішення є прикладом ефективного застосування rule-based логіки в електронній комерції для підтримки та інформування клієнтів.

1.3 Визначення вимог та критеріїв ефективності системи

На основі аналізу сучасних технологій автоматизації технічної підтримки, а також огляду функціональних особливостей популярних Telegram-ботів (@ComfyuaBot та @FoxtrotInfo_bot), можна сформулювати перелік базових вимог до систем, що покликані забезпечити ефективну взаємодію з користувачами у сфері підтримки.

Насамперед, одним з ключових висновків є доцільність використання месенджера Telegram як основної платформи для побудови подібних рішень. Telegram надає інтуїтивно зрозумілий інтерфейс, підтримку кнопок, авторизацію через API, а також широкі можливості інтеграції з зовнішніми сервісами. Обидва проаналізовані чат-боти демонструють, що мінімізація дій користувача на старті (наприклад, авторизація в один клік) значно покращує зручність і прискорює доступ до функціоналу.

Водночас важливо забезпечити персоналізований досвід взаємодії, зокрема шляхом збереження даних користувача, звернень на ім’я або підключення до індивідуального кабінету. Також відзначено, що користувач очікує чіткого меню, побудованого за логікою сценаріїв із фіксованими або динамічними кнопками. Це дозволяє уникнути неоднозначностей у запитах і спрощує навігацію.

З огляду на вищезазначене, до функціональних вимог до системи чат-бота підтримки в Telegram доцільно віднести:

- підтримку стартової автентифікації користувача (автоматизованої або з верифікацією);

- реалізацію інтуїтивного меню з кнопковим управлінням;
- доступ до ключових сервісів (довідка, замовлення, бонуси, контакт із оператором);
- можливість переадресації запиту до оператора у разі потреби;
- інтеграцію з базою даних або внутрішньою CRM.

Окрім цього, важливо визначити критерії ефективності такої системи, які дозволяють оцінити її придатність у практичному застосуванні. До базових критеріїв можна віднести:

- Швидкість доступу до цільової функції — кількість дій до отримання потрібної відповіді;
- Зрозумілість навігації — логічна структура меню без надлишкових кроків;
- Відсоток успішно оброблених запитів без залучення оператора;
- Час відповіді бота на стандартні запити (норматив — не більше 1–2 секунд);
- Рівень задоволеності користувачів (наприклад, за результатами опитування або відгуків).

Визначені вимоги та критерії забезпечують основу для створення ефективної системи підтримки, орієнтованої на користувача, та є орієнтиром при реалізації власної розробки чат-бота в Telegram. Врахування зазначених факторів дозволяє не лише підвищити якість обслуговування, але й зменшити навантаження на операторів, автоматизувавши обробку типових звернень.

РОЗДІЛ 2. РОЗРОБКА ТЕХНІЧНОГО ЗАВДАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Постановка мети та завдань розробки чат-бота, визначення функціональних та нефункціональних вимог

У сучасних умовах цифрової трансформації бізнесу важливою складовою ефективної взаємодії з клієнтами є своєчасна та доступна технічна підтримка. Враховуючи швидкість комунікацій, зростаючі очікування користувачів і навантаження на контактні центри, впровадження автоматизованих чат-ботів стало раціональним рішенням для підприємств. Особливо актуальним є створення ботів на основі Telegram API, що поєднує простоту використання, високу швидкість реагування та популярність серед аудиторії.

Метою даної кваліфікаційної роботи є розробка Telegram-бота для технічної підтримки підприємства, здатного ефективно реагувати на типові запити користувачів у режимі реального часу, підтримуючи мультимедійні формати відповідей та зберігаючи налаштування у зручному форматі для адміністратора.

Для досягнення поставленої мети були визначені такі основні завдання:

- дослідити сучасні технології створення чат-ботів з акцентом на технічну підтримку користувачів;
- провести аналіз існуючих рішень та визначити переваги і недоліки найпоширеніших реалізацій;
- розробити Telegram-бота з підтримкою авто-відповідей, текстового і мультимедійного контенту;
- реалізувати інтеграцію з Telegram Business API та підтримку бізнес-режиму взаємодії;
- забезпечити механізм налаштування авто-відповідей через адмін-команди без потреби редагування коду;
- здійснити збереження бази відповідей у форматі JSON для подальшої обробки;

- протестувати бот у робочому середовищі, перевіривши точність відповідей, швидкодію та стабільність функціонування.

На основі аналізу системи та її структури сформульовано функціональні вимоги, які бот повинен задовольняти:

- приймати повідомлення від користувача через Telegram API;
- знаходити відповідну відповідь на основі ключових слів або фраз (тригерів);
- надсилати відповіді у вигляді тексту, зображень, відео, стікерів, документів або голосових повідомлень;
- дозволяти адміністратору додавати або видаляти авто-відповіді безпосередньо через чат;
- забезпечити підтримку HTML-розмітки у відповідях;
- працювати з офіційним бізнес-акаунтом Telegram.

Крім того, проект враховує нефункціональні вимоги, які гарантують стабільність та якість сервісу:

- бот повинен відповідати у межах 1–2 секунд на стандартний запит;
- структура коду має бути достатньо модульною для майбутніх змін (наприклад, додавання NLP-модуля);
- дані зберігаються у форматі db.json, що спрощує переносимість і резервне копіювання;
- система повинна бути безпечною — лише адміністратор має змогу змінювати базу відповідей;
- бот має бути сумісним з Telegram Business API та оновленнями його протоколів.

Рис. 2.1 відображає логічну структуру формування цілей, завдань і вимог до системи Telegram-бота для технічної підтримки, а також демонструє послідовність їх взаємозв’язку. Центральним елементом виступає сформульована мета проєкту, з якої симетрично розгалужуються два ключові напрями: завдання та вимоги до системи.

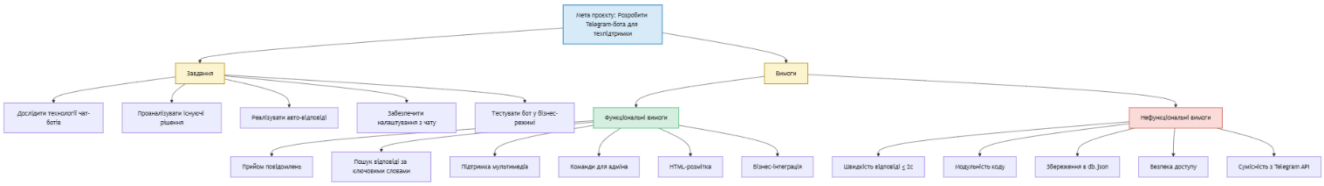


Рис. 2.1 Взаємозв’язок мети, завдань та вимог у проєкті Telegram-бота технічної підтримки

Група завдань розкриває, які практичні дії необхідні для реалізації мети: дослідження технологій, аналіз аналогів, програмна реалізація, конфігурація взаємодії з бізнес-акаунтом, а також тестування. Кожне з цих завдань є обґрунтованим кроком до задоволення визначених вимог — саме тому блок «Вимоги» логічно пов’язаний не тільки з метою, а й опосередковано відображає результат кожного з завдань.

Далі вимоги розділено на функціональні та нефункціональні. Такий поділ не лише дозволяє структурувати очікувану поведінку системи, а й підкреслює, що задоволення функціональних вимог (наприклад, підтримка HTML, мультимедіа, обробка команд) базується на виконанні відповідних завдань (реалізація, налаштування, збереження даних). Водночас нефункціональні вимоги — такі як швидкодія, безпечний доступ чи сумісність із Telegram API — тісно пов’язані з якісними критеріями, що мають бути враховані ще на етапі проєктування та тестування.

Схема ілюструє не лише структурну організацію компонентів проєкту, а й взаємозалежність між концептуальними елементами, що забезпечують узгодженість та цілісність розробки системи.

Побудова системи базується на принципах простоти використання, ефективного обслуговування типових запитів та гнучкої конфігурації без втрати якості обробки звернень. Це забезпечує підприємству потужний інструмент технічної підтримки, який мінімізує людське навантаження та водночас зберігає високий рівень сервісу.

2.2 Обґрунтування вибору технологічного стеку

У процесі реалізації системи Telegram-бота для автоматизованої технічної підтримки було обрано набір технологій, які забезпечують необхідний функціонал, простоту розгортання, адаптацію до бізнес-режиму Telegram та підтримку гнучкої модифікації системи. Нижче наведено детальне обґрунтування використаних технологій.

PHP (Hypertext Preprocessor) — це популярна серверна мова програмування з відкритим кодом, спеціально розроблена для створення динамічних вебсторінок і інтерактивних вебзастосунків. Вона підтримує імперативну, процедурну й об'єктно-орієнтовану парадигми програмування. PHP легко інтегрується з HTML, а також із більшістю СУБД (MySQL, PostgreSQL, SQLite тощо) і веб-серверів (Apache, Nginx), що робить її універсальним рішенням для розробки веб-додатків.

Однією з ключових переваг PHP є його простота у вивченні та низький поріг входу, що поєднується з широкими можливостями — починаючи від обробки форм і сесій до роботи з API та генерації динамічного контенту. Мова також підтримує розширення, шаблони, системи кешування, безпечну обробку даних і численні бібліотеки. PHP активно підтримується спільнотою, має масштабну документацію з прикладами та актуальну довідкову систему [12].

Завдяки цим якостям PHP залишається ефективним і раціональним вибором для реалізації малих і середніх проєктів, зокрема Telegram-ботів з базовою логікою, що не потребують складної інфраструктури.

PHP було обрано як основну мову розробки через її простоту, широку підтримку хостинг-сервісами та швидке розгортання без додаткових інструментів. Оскільки бот реалізує rule-based логіку та не потребує складного серверного середовища, PHP забезпечує достатню продуктивність і мінімальні вимоги до інфраструктури (рис. 2.2).

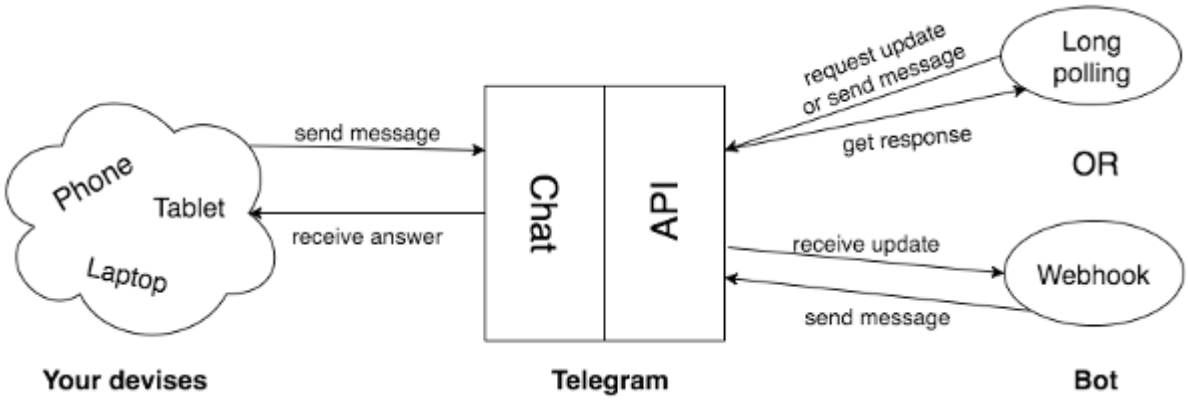


Рис. 2.2 Принципи доставки оновлень до Telegram-бота через Webhook або Long Polling у середовищі PHP [18]

Telegram Bot API — це офіційний інтерфейс прикладного програмування, що надає можливість створення, адміністрування та автоматизації чат-ботів у Telegram. API підтримує взаємодію з користувачем через текст, кнопки, медіа, документи, повідомлення з callback-зворотнім зв'язком тощо. Завдяки методам Bot API розробник може обробляти вхідні повідомлення (getUpdates або webhook), надсилати відповіді (sendMessage, sendPhoto, sendVoice тощо), працювати з inline-кнопками (InlineKeyboardMarkup), користувацькими командами та формами зворотного зв'язку (рис. 2.3).

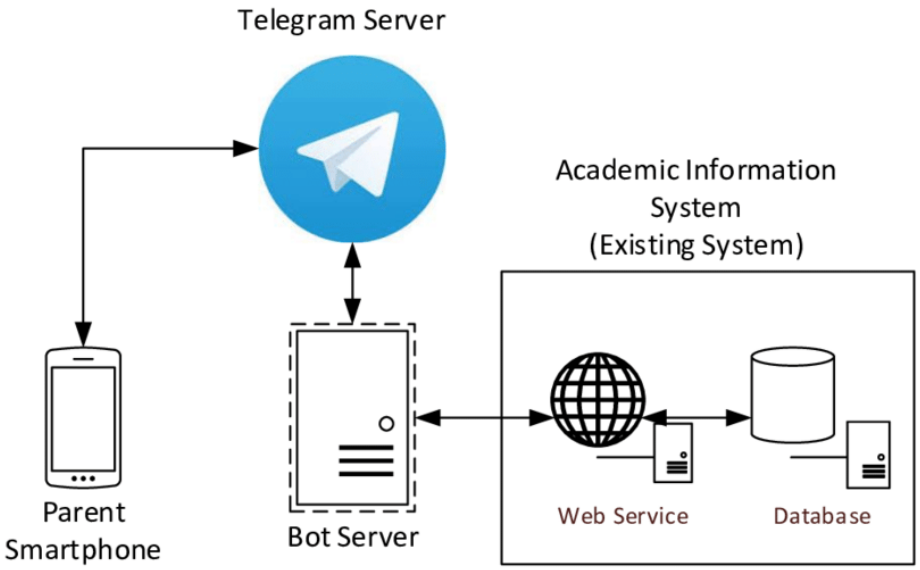


Рис. 2.3 Загальна схема взаємодії Telegram-бота з сервером і зовнішніми інформаційними системами [19]

Telegram Bot API відзначається високою стабільністю, безпекою, документацією з прикладами, а також підтримкою таких додаткових функцій, як обробка мультимедійних вкладень, розмітка повідомлень (HTML, Markdown), створення меню та опитувань, а також використання webhook-технології для обробки подій у реальному часі [13]. У контексті даного проєкту API забезпечує зв'язок між ботом і кінцевим користувачем, формує реакцію на тригери та дозволяє легко інтегрувати бізнес-режим через розширені типи повідомлень.

Telegram Business API — це розширення стандартного Telegram API, призначене для інтеграції ботів із бізнес-акаунтами компаній у месенджері Telegram. Цей інтерфейс надає розширені можливості обробки подій і повідомлень, характерних саме для корпоративного спілкування. Зокрема, підтримуються специфічні типи оновлень: business_connection, business_message, edited_business_message, deleted_business_messages, які дозволяють системі реагувати на бізнес-дії в реальному часі та зберігати контекст діалогу в рамках сервісного обслуговування.

На відміну від класичного Telegram Bot API, Business API орієнтований на підтримку вбудованих чатів між клієнтом і компанією через офіційно зареєстрованого бота. Це забезпечує більш формалізовану взаємодію, контроль над вікном активності, інтеграцію з Telegram Business Tools, а також підвищену довіру до ботів із боку користувачів завдяки маркуванню "офіційний чат-бот" [14].

У даному проєкті Telegram Business API використовується для забезпечення сумісності бота з бізнес-профілем компанії, активації спеціального webhook-режиму та повної підтримки бізнес-подій. Це дозволяє бот-системі обробляти звернення більш ефективно та відповідати вимогам до корпоративного рівня комунікації.

Замість повноцінної СУБД у проєкті застосовано легкий формат зберігання — db.json, який використовується для збереження ключових фраз користувача та відповідних відповідей. JSON (JavaScript Object Notation) є текстовим форматом для структурованих даних, що легко читається людиною та ефективно обробляється мовами програмування, включно з PHP. Він підтримує вкладені

об'єкти, масиви та асоціативні пари «ключ–значення», що робить його ідеальним для побудови бази знань бота у вигляді запит–відповідь [15].

Користувач-адміністратор взаємодіє з цією базою через Telegram-інтерфейс — за допомогою команд у чаті (наприклад, додавання нової відповіді або редагування наявної), які викликають оновлення відповідного JSON-файлу. Такий підхід забезпечує гнучкість налаштування, простоту резервного копіювання та сумісність із клієнтською частиною без потреби у складному бекенді чи панелі адміністратора.

Система авто-відповідей у реалізованому Telegram-боті побудована за принципом простого порівняння вхідного повідомлення з наявними тригерами у базі даних db.json. Кожен тригер є ключовою фразою або словосполученням, на яке бот має реагувати — у разі виявлення відповідності надсилається попередньо задана відповідь. Такий rule-based підхід забезпечує високу передбачуваність системи та мінімізує ймовірність помилкової реакції.

Особливістю реалізації є динамічна взаємодія між користувачем-адміністратором і ботом, яка дозволяє оновлювати або доповнювати список тригерів і відповідей безпосередньо через Telegram-інтерфейс (рис. 2.4). При цьому оновлення відбувається шляхом модифікації JSON-файлу, що виключає потребу у сторонніх інструментах або панелі адміністратора. Також система не використовує складних механізмів обробки природної мови (NLP), що робить її менш ресурсозатратною та простішою для підтримки.

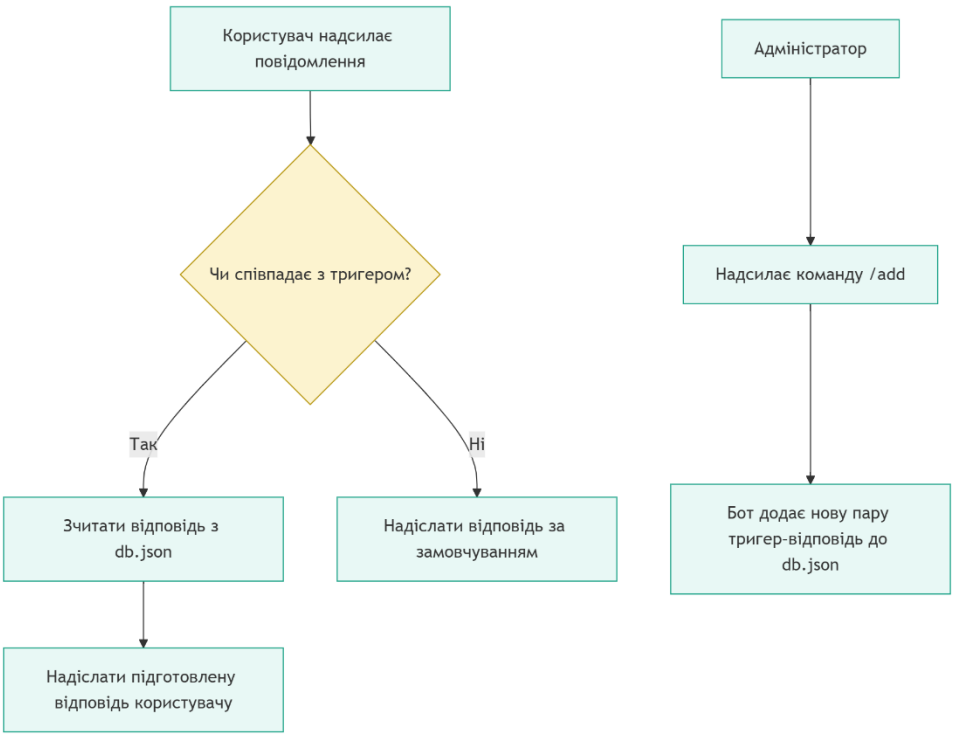


Рис. 2.4 Система авто-відповідей у реалізованому Telegram-боті

Такий підхід забезпечує виконання ключових нефункціональних вимог: швидкодію (відповідь у межах 1–2 секунд), модульність (просте розширення словника), а також прозорість логіки. У результаті користувач отримує стабільну реакцію бота на типові запити, а адміністратор — гнучкість і контроль над вмістом знань бота.

HTML (HyperText Markup Language) — це стандартна мова розмітки документів для відображення у веббраузерах. Вона дозволяє структурувати текстову інформацію, визначати заголовки, абзаци, списки, посилання, вставляти зображення та використовувати стилі форматування. HTML є основою вебінтерфейсів і широко використовується в різноманітних цифрових системах для передачі стилізованого контенту [16].

Telegram підтримує часткову реалізацію HTML у своїх повідомленнях — зокрема, теги <b>, <i>, <u>, <a href> та інші. Ця можливість дозволяє значно підвищити інформативність та візуальну організованість відповідей чат-бота. У реалізованому проєкті функціональність форматування використовується для

акцентування важливої інформації, виділення ключових слів, додавання гіперпосилань на сторонні ресурси тощо.

З технічної точки зору, форматування активується за допомогою параметра parse_mode, значення якого встановлюється як HTML у методах надсилання повідомлень через Telegram Bot API. При цьому текст повідомлення містить HTML-теги, які Telegram автоматично інтерпретує у відповідності до своєї специфікації. Це рішення дозволяє реалізувати компактні, структуровані й зрозумілі відповіді без потреби в окремому UI-компоненті або сторонньому рендерінгу. HTML у Telegram виконує роль мікроінтерфейсу в межах текстового чату.

Webhook — це механізм, який дозволяє серверам отримувати повідомлення в режимі реального часу, коли відбувається певна подія. На відміну від циклічного опитування (polling), коли сервер періодично звертається до API із запитом «чи з'явилося щось нове», webhook спрацьовує автоматично при появі події — наприклад, коли користувач надсилає повідомлення до бота (рис. 2.5). Telegram API підтримує webhook як основний рекомендований спосіб доставки оновлень для ботів [17].

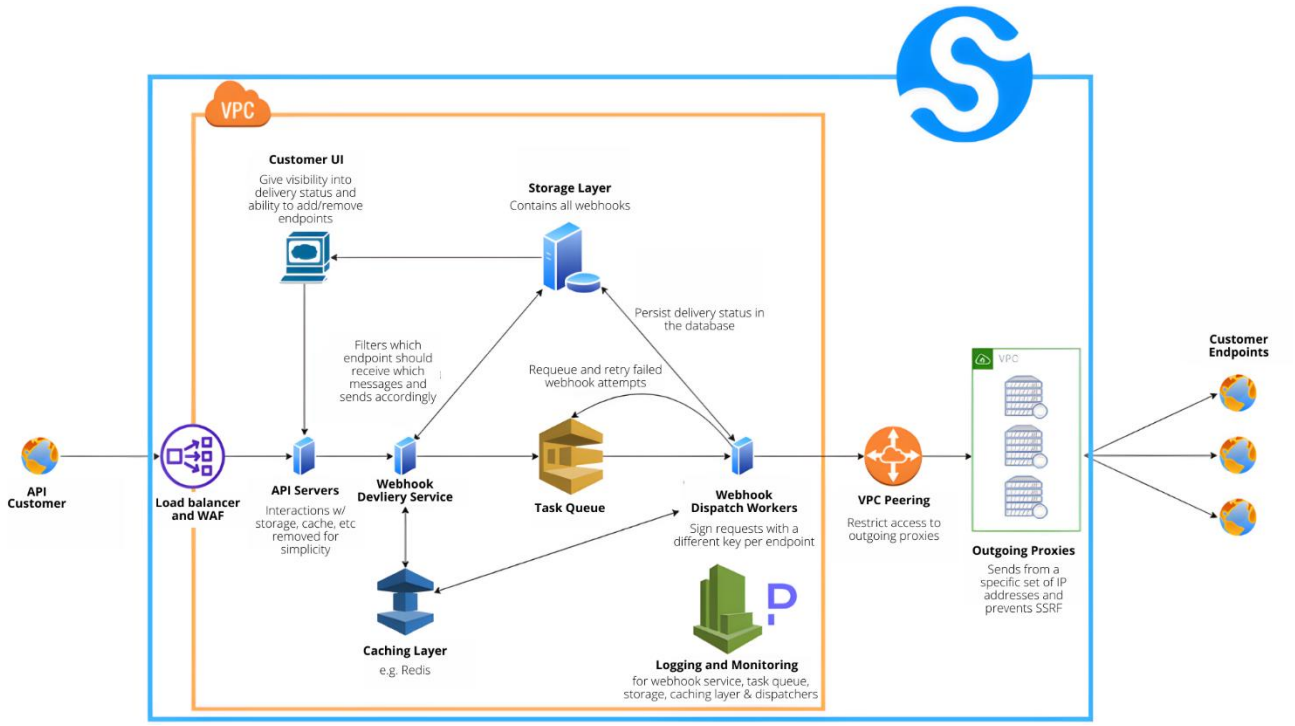


Рис. 2.5 Архітектура Webhook [20]

З технічного боку, webhook — це HTTPS-адреса, яку бот реєструє у Telegram. Коли надходить нове повідомлення, Telegram автоматично викликає цю адресу (POST-запитом) і передає вміст події у форматі JSON. У реалізованій системі бот обробляє ці події через PHP-файл index.php, що приймає запити на сервері.

Використання webhook забезпечує низьку затримку відповіді (реакція майже миттєва), мінімізує навантаження на сервер (оскільки не потрібно постійно опитувати API) та підвищує загальну ефективність роботи чат-бота. У контексті розробки Telegram-бота для бізнес-підтримки це особливо важливо для забезпечення швидкої реакції на звернення клієнтів у режимі 24/7.

Оскільки проєкт написано на PHP, він може бути розгорнутий на будь-якому середовищі з підтримкою Apache/Nginx і PHP 7+. Така сумісність забезпечує мобільність, простоту тестування і низький поріг входу при розгортанні на хостингу.

Структурна схема обраного стеку технологій показана на рис. 2.6.

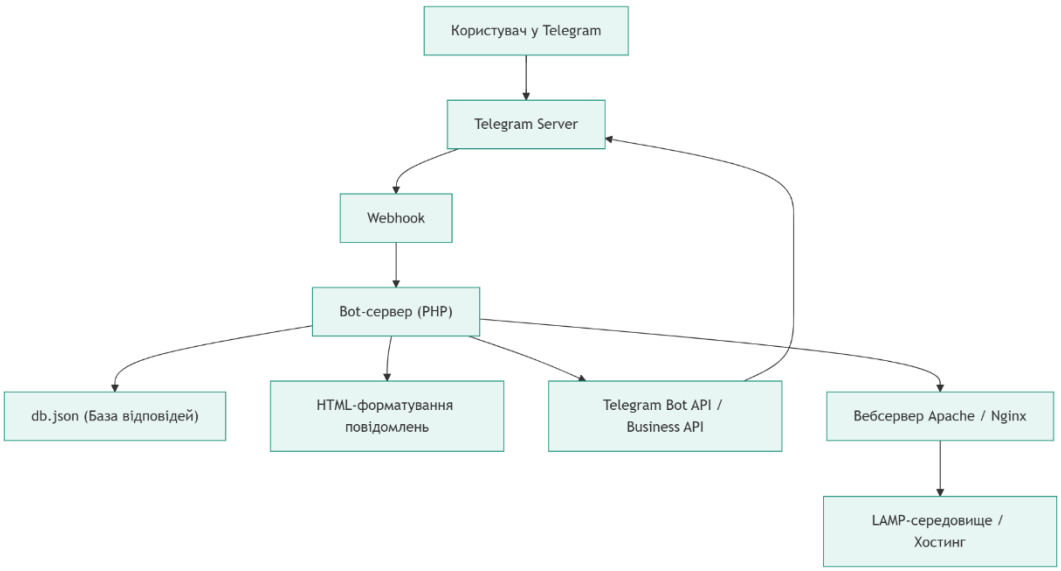


Рис. 2.6 – Обраний стек технологій

Обраний стек технологій є повністю обґрунтованим з позицій функціональності, ефективності та адаптованості до завдань автоматизації клієнтської підтримки в Telegram. Він дозволяє реалізувати систему, яка є простою в налаштуванні, гнучкою у підтримці та готовою до розширення в разі потреби.

2.3 Проєктування архітектури та користувацького інтерфейсу

Архітектура реалізованого Telegram-бота базується на класичній схемі клієнт-серверної взаємодії з використанням Telegram Bot API, webhook-механізму та внутрішнього механізму обробки запитів на основі ключових слів. Структура системи відображена на рис. 2.7.

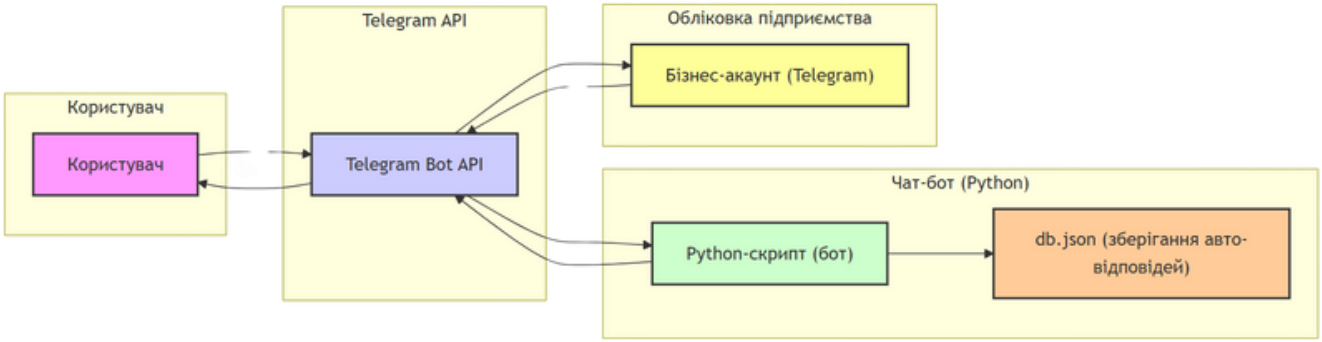


Рис. 2.7 Схема взаємодії між Telegram API, чат-ботом та базою авто-відповідей

На стороні клієнта виступає користувач, який ініціює взаємодію через стандартний інтерфейс мобільного застосунку або десктопної версії Telegram. Кожне повідомлення, яке він надсилає до чат-бота, проходить через централізований сервер Telegram і надалі передається в систему через Telegram Bot API. У проєкті використано бізнес-акаунт Telegram, який пов’язаний із профілем підприємства й забезпечує офіційний статус чат-бота та розширену підтримку подій бізнес-класу.

Основна логіка системи реалізована у вигляді Python-скрипта (бота), який працює на власному сервері з увімкненим webhook. Цей скрипт отримує структуру повідомлення від Telegram у форматі JSON, проводить аналіз вхідного тексту на наявність відповідного тригера (ключового слова або фрази) та надсилає користувачу відповідь. Всі пари “тригер–відповідь” зберігаються у внутрішньому файлі db.json, який виконує функцію бази знань. У разі потреби адміністратор може змінити вміст цієї бази через Telegram-команди, не редагуючи вручну файл на сервері.

Архітектура забезпечує чітке розділення ролей: Telegram Bot API відповідає за маршрутизацію повідомлень, Python-скрипт — за логіку обробки, а db.json — за збереження контенту. Така структура є простою, проте ефективною для реалізації авто-відповідей без необхідності в повноцінній СУБД або складному бекенді.

Попри відсутність класичної вебсторінки, користувацький інтерфейс Telegram-бота реалізується через текстові відповіді, що надсилаються у форматі HTML. Такий підхід дозволяє створити мікроінтерфейс без необхідності у браузері чи фронтенд-фреймворках. У Telegram HTML-теги обробляються на стороні клієнта, забезпечуючи форматування тексту, акцентування та гіперпосилання.

У системі відповіді бота надсилаються за допомогою методу sendMessage, де вказується параметр parse_mode = "HTML", що дозволяє Telegram інтерпретувати вміст повідомлення як HTML-код (рис. 2.8).

```
sendMessage([
    'chat_id' => $chat_id,
    'text' => "<b>Дякуємо за звернення!</b> Натисніть <a href='https://example.com'>сюди</a> для перегляду замовлення.",
    'parse_mode' => 'HTML'
]);
```

Рис. 2.8 Програмна побудова повідомлення користувачу

Цей блок генерує жирний заголовок та інтерактивне посилання. Telegram автоматично стилізує повідомлення відповідно до тегів <b>, <i>, ..., <code>, <u> тощо.

Завдяки такому підходу реалізовано:

- інтуїтивний мікроінтерфейс, який не потребує браузера;
- доступність з будь-якого пристрою, що підтримує Telegram;
- гнучкість форматування без ускладнення логіки бота.

Такий тип побудови інтерфейсу є оптимальним для інформаційних ботів, де головна мета — оперативно й структуровано передати повідомлення користувачу без зайвої навігації.

Інтерфейс Telegram-бота з боку користувача реалізований у вигляді послідовного текстового діалогу в месенджері. Всі дії відбуваються у звичному чат-

вікні Telegram, що забезпечує мінімальний поріг входу для користувача та не потребує додаткових інструкцій або програм. Як видно на рис. 2.9, логіка взаємодії є інтуїтивно зрозумілою, побудованою за принципом покрокового введення інформації.

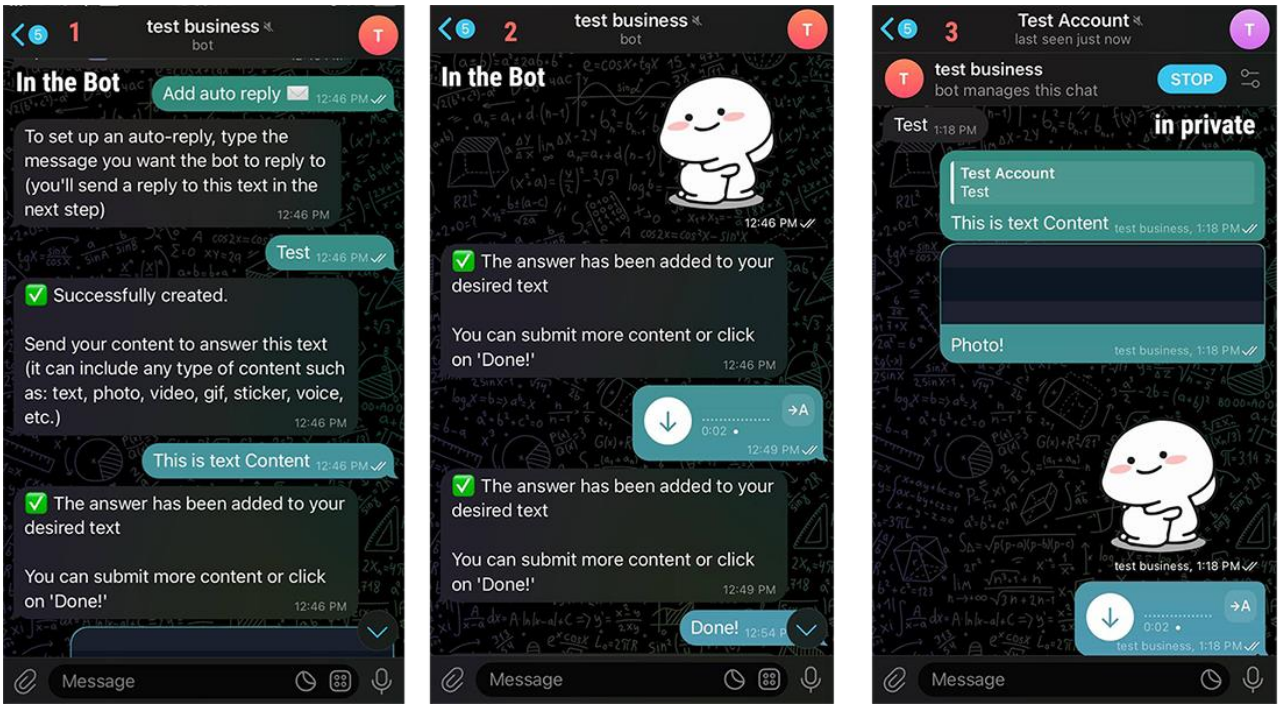


Рис. 2.9 Послідовність взаємодії користувача з Telegram-ботом у процесі створення авто-відповіді

На першому етапі користувач отримує інструкцію від бота щодо налаштування авто-відповіді. Повідомлення оформлене з використанням візуальних індикаторів (іконка повідомлення, зелені галочки), що покращує сприйняття інформації. Далі бот пропонує ввести текст-запит, на який у подальшому буде сформовано відповідь.

Після того, як користувач надсилає тригер (наприклад, “Test”), бот підтверджує створення правила відповіді та переходить до приймання контенту. На цьому етапі бот підтримує не лише текст, а й мультимедійні формати: фото, відео, GIF, стікери, голосові повідомлення тощо. Кожне надіслане повідомлення з контентом додається до відповіді, що автоматично підтверджується новим повідомленням бота з позначкою успішного збереження.

Після завершення редагування відповіді користувач натискає “Done!” (або надсилає цю команду текстом), після чого бот переходить у режим очікування нових команд або спрацьовує на нові запити згідно зі збереженими шаблонами.

Інтерфейс не перевантажений кнопками або меню, що робить його легким у використанні. Базові дії зведені до введення повідомлень і натискання на прості елементи керування, завдяки чому бот не потребує окремого навчання або документації для кінцевого користувача. Наявність візуальних реакцій на кожен крок посилює відчуття контролю над процесом.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Реалізація функціональних модулів чат-бота та автоматизованих відповідей

Загальний алгоритм роботи реалізованого Telegram-бота ґрунтується на обробці вхідних повідомлень користувача через API Telegram і формуванні автоматичних відповідей на основі бази знань, що зберігається у форматі JSON (рис. 3.1). Архітектура системи побудована з урахуванням асинхронної моделі обміну подіями через webhook, що дозволяє обробляти запити в реальному часі.

Після отримання нового повідомлення через Telegram Server, API надсилає його у вигляді POST-запиту на webhook-адресу, пов’язану з ботом. У реалізованій системі webhook обробляється файлом index.php, який виконує роль єдиного точки входу для запитів. У цьому файлі відбувається декодування JSON-структури запиту, з якої витягуються необхідні поля: ідентифікатор чату (chat_id), текст повідомлення (text), тип контенту, а також службові параметри (наприклад, чи є повідомлення відповіддю на інше).

Після аналізу структури повідомлення система перевіряє, чи є текст користувача тригером — ключовою фразою, що міститься в базі авто-відповідей. Якщо відповідність знайдено, бот витягує з db.json асоційовану відповідь та надсилає її користувачеві через метод sendMessage Telegram API. У випадку мультимедійного контенту (наприклад, відео, аудіо, документи), використовується відповідний метод: sendPhoto, sendVoice, sendDocument тощо.

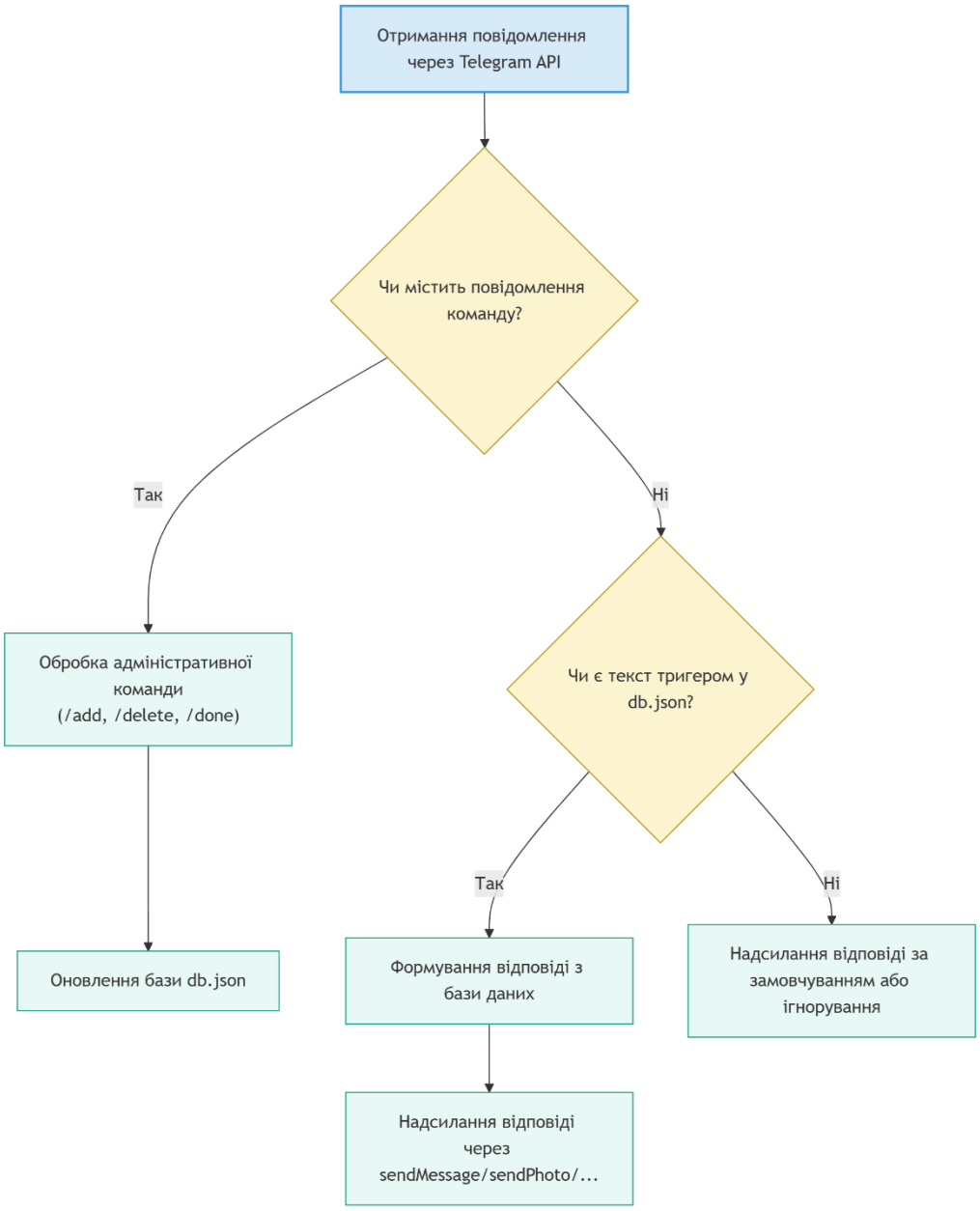


Рис. 3.1 Алгоритм роботи Telegram-бота з автоматизованими відповідями

Якщо тригер не знайдено, бот або надсилає відповідь за замовчуванням, або ігнорує запит (залежно від конфігурації). Окремо в алгоритмі передбачено можливість для адміністратора додавати або видаляти пари “запит–відповідь”. У такому разі бот розпізнає службові команди (/add, /delete, /done) та переходить у режим налаштування, зберігаючи нову інформацію до db.json.

Логіка роботи програми побудована за принципом обробки подій із централізованим сховищем шаблонів відповідей і умовним маршрутизатором для

контенту. Такий підхід забезпечує стабільність, гнучкість і мінімальні затримки в комунікації з користувачем.

У реалізованій системі Telegram-бота для збереження шаблонів автоматизованих відповідей використовується проста, проте структурована база даних у форматі JSON. Вона виконує роль внутрішнього сховища знань, що дозволяє системі швидко знайти відповідь на заданий тригер без використання зовнішньої СУБД. Структура бази відповідає принципам об’єктно-орієнтованого моделювання, що відображено у вигляді UML-діаграми класів на рис. 3.11.

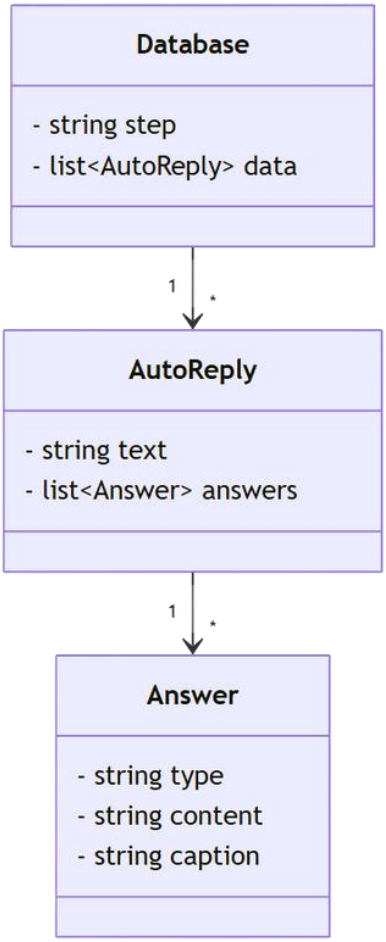


Рис. 3.2 Структура класів, що описує формат збереження шаблонів авто-відповідей у JSON-базі

На верхньому рівні базу представляє об’єкт Database, який містить два ключових поля:

- step — службова змінна, що зберігає поточний етап взаємодії адміністратора з ботом (наприклад, очікування тексту, вибір контенту тощо);
- data — масив об'єктів типу AutoReply, кожен з яких відповідає окремому тригеру.

Клас AutoReply відповідає за зберігання окремого правила авто-відповіді. Він включає:

- text — тригерне повідомлення (тобто запит користувача, на який слід реагувати);
- answers — список відповідей, асоційованих із цим тригером.

Кожна відповідь описується об'єктом типу Answer, який у свою чергу складається з:

- type — тип відповіді (text, photo, video, voice, sticker, тощо);
- content — вміст відповіді (текст або файл ID);
- caption — необов’язковий підпис до медіа-контенту.

Така структура дозволяє реалізувати складні мультимедійні відповіді з кількох елементів, підтримуючи гнучкість у формуванні контенту. Крім того, збереження даних у форматі JSON дозволяє легко редагувати, копіювати або резервувати базу знань вручну або програмно.

На рис. 3.3 представлено фрагмент функціонального коду на мові PHP, який відповідає за комунікацію Telegram-бота з сервером Telegram через офіційний інтерфейс Telegram Bot API. Основна мета даної функції — формування HTTP-запиту до одного з методів API з урахуванням параметрів та токену авторизації.

```php
function bot($method, $datas = [])
{
    global $token;
    $ch = curl_init();
    curl_setopt_array($ch, array(
        CURLOPT_URL => 'https://api.telegram.org/bot' . $token . '/' . $method,
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_POSTFIELDS => $datas
    ));
    return json_decode(curl_exec($ch));
}
```

Рис. 3.3 Функція виклику методів Telegram Bot API з використанням cURL у PHP

Функція bot($method, $datas = []) приймає два аргументи: назву методу Telegram API, який потрібно викликати (наприклад, sendMessage, sendPhoto тощо), та масив даних $datas, який буде переданий у тілі запиту у форматі POST. Всередині функції використовується глобальна змінна $token, що містить токен авторизації, виданий Telegram через BotFather — цей токен додається до URL запиту як частина маршруту.

Для надсилання HTTP-запиту використовується інструмент cURL, зокрема функція curl_init() створює новий дескриптор сеансу, а curl_setopt_array() конфігурує масив параметрів:

- CURLOPT_URL формує кінцеву URL-адресу на кшталт https://api.telegram.org/bot<token>/<method>,
- CURLOPT_RETURNTRANSFER => true дозволяє зберегти результат виконання у змінну, а не виводити його безпосередньо,
- CURLOPT_POSTFIELDS => $datas передає масив параметрів методу API.

Після виконання запиту функція curl_exec($ch) отримує відповідь сервера у вигляді JSON-стрічки, яка перетворюється на асоціативний масив за допомогою json_decode(...) і повертається як результат роботи функції. Таким чином, цей код є універсальним засобом виклику будь-якого методу Telegram API, і використовується як базова точка взаємодії бота з Telegram-сервером. Він забезпечує гнучкість, скорочує дублювання коду та централізує логіку надсилання запитів.

На рис. 3.4 представлено ключовий фрагмент PHP-коду, який відповідає за первинне розпакування і класифікацію вхідного запиту, надісланого до Telegram-бота. Цей код обробляє вміст, що надходить у вигляді JSON-структури через webhook (вхідний потік php://input), і здійснює розділення даних за типами повідомлень, включно з мультимедійними вкладеннями та бізнес-повідомленнями.

```php
$update = json_decode(file_get_contents('php://input'));
if (isset($update)) {
    @$message = $update->message;
    if (isset($message)) {
        @$text = $message->text;
        @$chat_id = $message->chat->id;
        @$caption = $message->caption;
        //file id
        @$sticker_id = $message->sticker->file_id;
        @$photo_id = $message->photo[count($message->photo) - 1]->file_id;
        @$video_id = $message->video->file_id;
        @$voice_id = $message->voice->file_id;
        @$file_id = $message->document->file_id;
        @$music_id = $message->audio->file_id;
        @$animation_id = $message->animation->file_id;
        @$video_note_id = $message->video_note->file_id;
    }

    // business updates
    if (isset($update->business_message)) {
        @$b_message = $update->business_message;
        @$b_id = $b_message->business_connection_id;
        @$b_text = $b_message->text;
        @$b_message_id = $b_message->message_id;
        @$b_chat_id = $b_message->chat->id;
    }
}
```

Рис. 3.4 Обробка вхідного webhook-запиту та розпізнавання типів повідомлень у Telegram-боті

У першому блоці виконується декодування вхідного повідомлення ($update) за допомогою функції json_decode(...). Далі перевіряється, чи структура $update містить поле message — стандартне поле для текстових та мультимедійних запитів. Якщо таке поле існує, з нього витягуються базові атрибути повідомлення: текст повідомлення ($text), ідентифікатор чату ($chat_id) та підпис до контенту ($caption).

Особливістю даної реалізації є підтримка широкого спектру медіафайлів. Код перевіряє наявність вкладень, таких як стікери, фото, відео, голосові повідомлення, документи, аудіо, анімації та відео-нотатки. Для кожного типу медіа автоматично витягується унікальний file_id, який надалі може бути використаний для

пересилання, завантаження або збереження. У випадку фото, оскільки Telegram передає масив зображень різної якості, вибирається останній елемент (найвища роздільна здатність) за допомогою count($message->photo) - 1.

У другому блоці коду реалізована підтримка бізнес-повідомлень Telegram, які надходять у вигляді окремої структури business_message. З неї витягуються аналогічні поля: текст ($b_text), ID повідомлення, ID чату та ID бізнес-з'єднання. Це дозволяє реалізувати адаптивну обробку як звичайних користувачів, так і бізнес-акаунтів, що взаємодіють із ботом.

Дана частина програмного забезпечення є критично важливим етапом обробки запитів, оскільки забезпечує коректне розділення вхідної інформації та підготовку її до подальшої маршрутизації в логіку бота.

На рис. 3.5 зображено фрагмент PHP-коду, відповідального за зчитування поточного стану чат-бота та формування інтерфейсних елементів у вигляді вбудованої клавіатури Telegram. Така клавіатура використовується для керування взаємодією з ботом з боку адміністратора або кінцевого користувача.

```php
$db =  json_decode(file_get_contents('db.json'), true);
$step = $db['step'];
// keyboards
$home = json_encode(['resize_keyboard' => true, 'keyboard' => [[['text' => "Add auto reply ✉️"]], [['text' => "remove auto reply 🚫"]]]]);
$back = json_encode(['resize_keyboard' => true, 'keyboard' => [[['text' => "Back 🔙"]]]]);
```

Рис. 3.5 Формування динамічної клавіатури управління в Telegram-боті на основі JSON-бази

На початку блоку здійснюється зчитування вмісту бази даних db.json у змінну $db з подальшим перетворенням JSON-структури на асоціативний масив завдяки параметру true у функції json_decode(...). З отриманого масиву виділяється ключ step, який зберігає поточний логічний етап сценарію взаємодії (наприклад, очікування вводу команди або медіаконтенту).

Далі ініціалізуються змінні $home та $back, які містять JSON-кодовані об'єкти, що відповідають синтаксису Telegram API для клавіатур типу ReplyKeyboardMarkup. Зокрема, змінна $home відповідає за головне меню, що містить дві кнопки: “Add auto reply” та “remove auto reply”. Ці елементи призначені

для додавання нових шаблонів авто-відповідей та їхнього видалення відповідно. Кожна кнопка — це елемент вкладеного масиву, структура якого відповідає формату поля keyboard в API Telegram.

Змінна $back генерує альтернативну клавіатуру з однією кнопкою “Back”, що дозволяє повернутись до попереднього стану або вийти з поточного режиму редагування. Обидві клавіатури налаштовані з параметром 'resize_keyboard' => true, який дозволяє Telegram-клієнту автоматично адаптувати висоту клавіатури під розмір кнопок, покращуючи зручність використання.

Наведений фрагмент коду є частиною модуля побудови адаптивного інтерфейсу управління ботом, реалізованого без необхідності класичного GUI, але з використанням вбудованих Telegram-компонентів.

На рис. 3.6 зображено перший логічний блок головної гілки обробки повідомлень адміністратора в Telegram-боті. Цей фрагмент відповідає за обробку базових текстових команд та перехід у режим створення автоматичних відповідей.

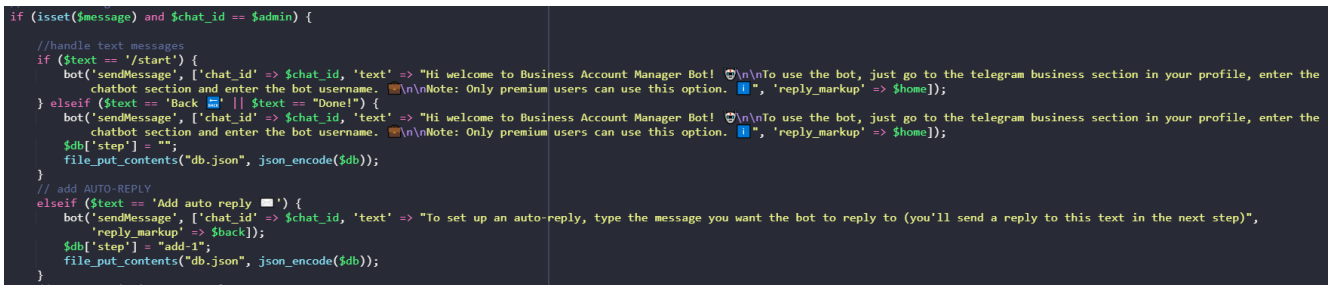


Рис. 3.6 Обробка команд /start, “Back”, “Add auto reply” та оновлення логічного стану чат-бота

Блок починається з перевірки того, що вхідне повідомлення ($message) було надіслано саме адміністратором. Це реалізовано за допомогою подвійної умови: наявність об’єкта $message і збіг $chat_id із попередньо визначеним значенням $admin.

Перший умовний оператор (if ($text == '/start')) відповідає за обробку стартової команди. У відповідь бот надсилає привітальне повідомлення з короткою інструкцією, як скористатися функціоналом для бізнес-акаунтів Telegram.

Повідомлення супроводжується клавіатурою, яка передається у вигляді JSON-структури через змінну $home.

Далі перевіряються повідомлення з текстом “Back” або “Done!”. У цих випадках бот знову надсилає стартове повідомлення та повертає користувача до головного меню. Одночасно з цим виконується скидання логічного стану сценарію ($db['step'] = ""), що оновлюється в базі db.json. Такий підхід забезпечує контроль за перебігом діалогу і запобігає помилкам у режимах взаємодії.

Останній блок цього фрагмента відповідає за обробку натискання кнопки “Add auto reply”. Бот переходить до режиму створення автоматичної відповіді: надсилає інструкцію користувачу щодо подальших дій і оновлює стан $db['step'] на "add-1", що сигналізує про очікування введення тригера. Усі зміни записуються до файлу db.json, який відіграє роль внутрішньої бази даних.

Розглянутий фрагмент реалізує ключовий механізм навігації та ініціалізації основних режимів керування ботом через вбудовану клавіатуру Telegram.

На рис. 3.7 представлено фрагмент коду, який реалізує два важливі режими взаємодії адміністратора з Telegram-ботом: початок створення нової авто-відповіді та процес видалення наявної. Ці гілки є частиною умовної структури обробки текстових повідомлень від адміністратора.

```
elseif ($text == 'Add auto reply ➕') {
    bot('sendMessage', ['chat_id' => $chat_id, 'text' => "To set up an auto-reply, type the message you want the bot to reply to (you'll send a reply to this text in the next
        step)", 'reply_markup' => $back]);
    $db['step'] = "add-1";
    file_put_contents("db.json", json_encode($db));
}
// remove existing auto-reply
elseif ($text == 'remove auto reply 🚫') {
    if (count($db['data']) > 0) {
        foreach ($db['data'] as $item) {
            $list .= "<code>{$item['text']}</code>\n---\n";
        }
        bot('sendMessage', ['chat_id' => $chat_id, 'text' => $list, 'parse_mode' => 'html',]);
        bot('sendMessage', ['chat_id' => $chat_id, 'text' => "To remove an item from the auto-reply, copy and paste one of the above", 'reply_markup' => $back]);
        $db['step'] = "remove";
        file_put_contents("db.json", json_encode($db));
    } else {
        bot('sendMessage', ['chat_id' => $chat_id, 'text' => "auto-reply list is empty!", 'reply_markup' => $home]);
    }
}
```

Рис. 3.7 Ініціалізація режиму додавання та видалення шаблонів авто-відповідей у Telegram-боті

У першій гілці (elseif ($text == 'Add auto reply')) бот ініціює режим створення авто-відповіді. Користувачеві надсилається повідомлення з інструкцією, як

сформувати правило відповіді: потрібно надіслати текст, на який бот у майбутньому буде реагувати. Водночас оновлюється поле step у внутрішній базі даних (db.json) на значення "add-1", що свідчить про перехід у стан очікування тригерного тексту. Для зручності навігації прикріплюється клавіатура $back, яка дозволяє скасувати дію.

Наступний блок обробляє сценарій видалення авто-відповіді (elseif ($text == 'remove auto reply')). Спочатку перевіряється, чи не є масив data у базі db.json порожнім. Якщо відповіді існують, вони перетворюються на список, де кожен тригер оформлюється за допомогою HTML-тега <code> та виводиться у вигляді текстового переліку. Бот надсилає цей список користувачу разом з інструкцією: скопіювати текст тригера, який потрібно видалити. Після цього оновлюється логічний стан $db['step'] = "remove", що фіксує перехід у режим очікування дії на видалення конкретного правила.

Якщо база авто-відповідей порожня, бот повідомляє про це повідомленням "auto-reply list is empty!" і повертає користувача до головного меню ($home).

Узагальнюючи, цей фрагмент забезпечує обидва ключові CRUD-оператори (Create та Delete) для керування базою знань бота. Завдяки збереженню логічного стану в db.json, система працює як сценарний автомат із передбачуваною поведінкою, не потребуючи складної маршрутизації або баз даних.

На рис. 3.8 зображено фрагмент коду, що реалізує два ключові етапи у сценарії створення нової автоматизованої відповіді Telegram-бота: фіксацію тригерного запиту користувача (step == 'add-1') та початок обробки відповідного вмісту (step == 'add-2'). Саме ці стани керують логікою покрокового налаштування авто-відповіді з боку адміністратора.

У першій гілці, коли значення $step дорівнює "add-1", бот підтверджує створення нового тригера (тобто вхідного повідомлення, на яке має реагувати система). У цей момент бот інформує адміністратора, що можна надсилати будь-який тип вмісту для відповіді (текст, фото, голос, відео тощо). У структурі db.json створюється новий об’єкт у масиві data, в якому поле text зберігає тригерне повідомлення, а поле answers ініціалізується як порожній масив. Це дає змогу далі

додавати декілька відповідей до одного тригера. Стан системи змінюється на "add-2", що сигналізує про очікування вмісту для відповіді.

```php
elseif ($step == 'add-1') {
    bot('sendMessage', ['chat_id' => $chat_id, 'text' => "✅ Successfully created.\n\nSend your content to answer this text (it can include any type of content such as: text, photo, video, gif, sticker, voice, etc.)", 'reply_markup' => $back]);
    $db['data'][] = [
        'text' => $text,
        'answers' => []
    ];
    $db['step'] = "add-2";
    file_put_contents("db.json", json_encode($db));
} elseif ($step == 'add-2') {
    end($db['data']);
    $last_key = key($db['data']);

    // check message type
    if (isset($text)) {
        $type = "text";

        // convert premium emoji
        if (isset($message->entities)) {
            $i = 0;
            foreach ($message->entities as $entity) {
                if ($entity->type == "custom_emoji") {
                    $offset = $i + $entity->offset;
                    $emoji = '<tg-emoji emoji-id="' . $entity->custom_emoji_id . '">' . mb_substr($text, $offset, 1, "UTF-8") . '</tg-emoji>';
                    $text = mb_substr($text, 0, $offset, "UTF-8")
                        . $emoji
                        . mb_substr($text, $offset + 1, null, "UTF-8");
                    $i = $i + mb_strlen($emoji) - $entity->length;
                }
            }
        }
        $content = $text;
```

Рис. 3.8 Реєстрація тригера та обробка преміум-емодзі у відповідях Telegram-бота

У другій частині коду починається перевірка типу контенту, що надіслав користувач у відповідь. Якщо це текст, встановлюється тип "text" і виконується спеціальна обробка для випадку, коли повідомлення містить преміум-емодзі Telegram (тип custom_emoji). Бот здійснює розбір масиву entities та за допомогою функції mb_substr() формує спеціальні HTML-обгортки <tg-emoji>, які Telegram інтерпретує як візуально повноцінні емодзі. Це дозволяє не втрачати форматування і зберігати точну візуальну відповідність під час надсилання відповідей кінцевому користувачу.

Після цієї обробки змінна $content зберігає готовий текст відповіді з усіма обробленими емодзі. Ця частина коду свідчить про високий рівень адаптації бота до специфіки Telegram API та забезпечує збереження візуальної складової навіть у відповідях, що містять розширені елементи.

На рис. 3.9 представлено фрагмент коду, який виконує ідентифікацію типу надісланого користувачем мультимедійного контенту під час налаштування авто-відповіді Telegram-бота. Цей блок є частиною логіки, що реалізується на етапі add-2, коли система очікує введення відповіді на попередньо заданий тригер.

```
} elseif (isset($sticker_id)) {
    $type = "sticker";
    $content = $sticker_id;
} elseif (isset($photo_id)) {
    $type = "photo";
    $content = $photo_id;
} elseif (isset($video_id)) {
    $type = "video";
    $content = $video_id;
} elseif (isset($voice_id)) {
    $type = "voice";
    $content = $voice_id;
} elseif (isset($file_id)) {
    $type = "file";
    $content = $file_id;
} elseif (isset($music_id)) {
    $type = "music";
    $content = $music_id;
} elseif (isset($animation_id)) {
    $type = "animation";
    $content = $animation_id;
} elseif (isset($video_note_id)) {
    $type = "video_note";
    $content = $video_note_id;
}
```

Рис. 3.9 Визначення типу мультимедійного контенту та присвоєння відповідного file_id у Telegram-боті

Кожен тип контенту перевіряється по черзі за допомогою умовних конструкцій elseif (isset(...)). Якщо певний тип ідентифіковано — наприклад, фото, стікер або голосове повідомлення — відповідно присвоюються значення змінним $type (текстове позначення типу) та $content (ідентифікатор файла, видобутий із вхідного JSON-об’єкта Telegram).

Підтримувані типи медіа включають:

- sticker — стікери;
- photo — зображення;
- video — відеофайли;

- voice — голосові повідомлення;
- file — документи (будь-який інший вкладений файл);
- music — аудіотреки;
- animation — анімовані GIF;
- video_note — відео-замітки.

Ідентифікатори (*_id) були заздалегідь отримані на попередніх етапах парсингу повідомлення з об’єкта $message (див. рис. 3.4). Завдяки такій архітектурі Telegram-бот зберігає гнучкість і дозволяє адміністраторам налаштовувати авто-відповіді не лише у форматі тексту, а й у вигляді мультимедійних шаблонів.

Цей блок коду є надзвичайно важливим для забезпечення підтримки різнорідного контенту, що особливо актуально в сучасних сервісах технічної підтримки, де наочність і швидкість сприйняття інформації відіграють ключову роль. Така реалізація також спрощує логіку формування масиву answers у базі даних db.json.

На рис. 3.10 зображено фрагмент коду, який виконує обробку підписів (caption) до мультимедійного контенту, зокрема — обробку преміум-емодзі Telegram у полі підпису. Цей блок доповнює логіку, описану раніше для текстових повідомлень (див. рис. 3.8), і забезпечує збереження розширених графічних елементів у повідомленнях, які супроводжують файли.

```php
if (isset($caption)) {
    // convert premium emoji
    if (isset($message->caption_entities)) {
        $i = 0;
        foreach ($message->caption_entities as $entity) {
            if ($entity->type == "custom_emoji") {
                $offset = $i + $entity->offset;
                $emoji = '<tg-emoji emoji-id="' . $entity->custom_emoji_id . '">' . mb_substr($caption, $offset, 1, "UTF-8") . '</tg-emoji>';
                $caption = mb_substr($caption, 0, $offset, "UTF-8")
                    . $emoji
                    . mb_substr($caption, $offset + 1, null, "UTF-8");
                $i = $i + mb_strlen($emoji) - $entity->length;
            }
        }
    }
}
```

Рис. 3.10 Обробка преміум-емодзі у підписах до мультимедійного контенту Telegram-бота

Перевірка if (isset($caption)) визначає, чи існує підпис до отриманого медіа. Якщо так, додатково перевіряється наявність масиву caption_entities, у якому Telegram зберігає інформацію про всі спецсимволи, у тому числі користувацькі емодзі (custom_emoji). Далі для кожного об'єкта в цьому масиві виконується ідентифікація типу емодзі, обчислюється його позиція (offset) та будується HTML-тег <tg-emoji>...</tg-emoji> з унікальним атрибутом emoji-id.

Для вставки емодзі без втрати решти тексту використовується функція mb_substr(), яка дозволяє оперувати багатобайтовими символами у кодуванні UTF-8. Фінальний результат — це повністю відтворений рядок підпису, де кожен преміум-емодзі обгорнутий у відповідний HTML-тег, який коректно обробляється Telegram-клієнтами при виводі повідомлення.

Завдяки цьому підходу підписи зберігають візуальну цілісність навіть при використанні Telegram Business API, що критично важливо для корпоративного стилю спілкування. У поєднанні з попередньою обробкою основного тексту, цей фрагмент коду забезпечує єдиний механізм адаптації емодзі в усіх типах відповідей бота.

На рис. 3.11 наведено завершальний фрагмент обробки адміністративних сценаріїв Telegram-бота, який охоплює два ключові стани: збереження відповіді до обраного тригера ($step == 'add-2') та видалення відповідного запису з бази ($step == 'remove').

```
        if (isset($type) and isset($content)) {
            bot('sendMessage', ['chat_id' => $chat_id, 'text' => "✅ The answer has been added to your desired text\n\nYou can submit more content or click on 'Done!' ",
                'reply_markup' =>  json_encode(['resize_keyboard' => true, 'keyboard' => [[['text' => "Done!"]]]])]);
            $db['data'][$last_key]["answers"][] = [
                'type' => $type,
                'content' => $content,
                'caption' => $caption
            ];
            file_put_contents("db.json", json_encode($db));
        } else {
            bot('sendMessage', ['chat_id' => $chat_id, 'text' => "There was a problem with the content you sent, please send another content", 'reply_markup' =>  json_encode
                (['resize_keyboard' => true, 'keyboard' => [[['text' => "Done!"]]]])]);
        }
    } elseif ($step == 'remove') {
        bot('sendMessage', ['chat_id' => $chat_id, 'text' => "✅ Successfully removed ", 'reply_markup' => $home]);
        foreach ($db['data'] as $key => $item) {
            if ($item['text'] == $text) {
                unset($db['data'][$key]);
            }
        }
        $db['step'] = "";
        file_put_contents("db.json", json_encode($db));
    }
}
```

Рис. 3.11 Додавання відповіді до тригера та видалення шаблону з бази auto-reply Telegram-бота

У першій частині блоку перевіряється, чи були коректно визначені змінні $type та $content. Якщо так — формується асоціативний масив, що містить відповідь для останнього доданого тригера (ідентифікований через $last_key). Цей масив складається з таких полів:

- type: тип відповіді (наприклад, text, photo, voice);
- content: ідентифікатор файла або текстовий вміст;
- caption: підпис до мультимедіа, якщо він є.

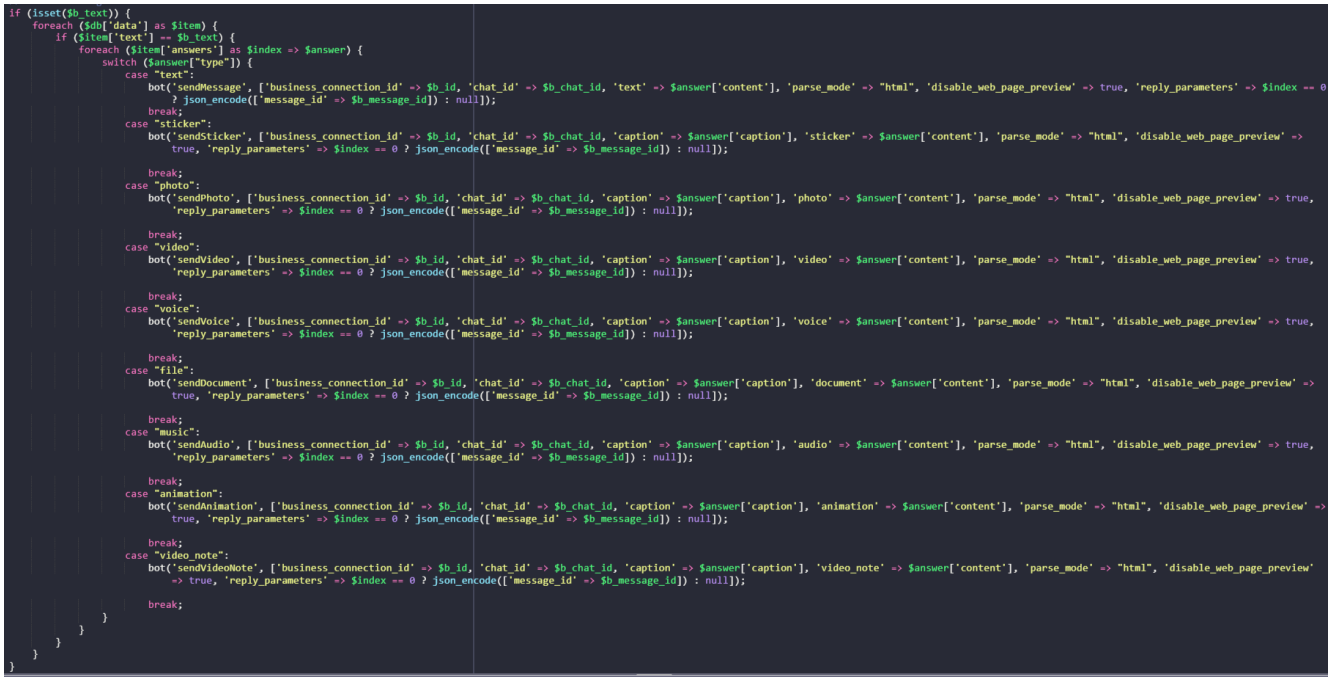
Цей запис додається до масиву answers, який знаходиться в об'єкті data бази db.json. Таким чином, до одного тригера можна прив'язати декілька відповідей. Після успішного додавання бот інформує адміністратора та пропонує або додати ще відповідей, або завершити дію натисканням “Done!”.

Якщо ж тип або контент не були встановлені, бот надсилає повідомлення з проханням надіслати інший вміст, тим самим зберігаючи цілісність структури даних і запобігаючи збереженню некоректних записів.

Друга частина фрагмента ($step == 'remove') відповідає за видалення тригера, який співпадає з надісланим повідомленням користувача. За допомогою циклу foreach бот перебирає всі записи в масиві data та, якщо знаходить відповідність, видаляє елемент функцією unset(). Після видалення бот підтверджує успіх операції та скидає логічний стан ($step = ""). Оновлена база знову зберігається у db.json.

Цей фрагмент ілюструє повний цикл керування структурою знань Telegram-бота: додавання нових правил, перевірку на валідність та видалення непотрібних шаблонів. Логіка побудована згідно з принципами сценарного програмування та забезпечує надійність і масштабованість системи.

На рис. 3.12 представлено завершальний фрагмент функціонального модуля Telegram-бота, який реалізує механізм надсилання автоматичних відповідей у відповідь на отримане бізнес-повідомлення. Цей блок є центральним в архітектурі rule-based логіки, оскільки саме тут виконується відбір відповідного тригера з бази даних db.json та формування відповіді згідно з її типом.



```php
if (isset($b_text)) {
    foreach ($db['data'] as $item) {
        if ($item['text'] == $b_text) {
            foreach ($item['answers'] as $index => $answer) {
                switch ($answer["type"]) {
                    case "text":
                        bot('sendMessage', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'text' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview' => true, 'reply_parameters' => $index == 0
                            ? json_encode(['message_id' => $b_message_id]) : null]);
                        break;
                    case "sticker":
                        bot('sendSticker', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'caption' => $answer['caption'], 'sticker' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview' =>
                            true, 'reply_parameters' => $index == 0 ? json_encode(['message_id' => $b_message_id]) : null]);

                        break;
                    case "photo":
                        bot('sendPhoto', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'caption' => $answer['caption'], 'photo' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview' => true,
                            'reply_parameters' => $index == 0 ? json_encode(['message_id' => $b_message_id]) : null]);

                        break;
                    case "video":
                        bot('sendVideo', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'caption' => $answer['caption'], 'video' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview' => true,
                            'reply_parameters' => $index == 0 ? json_encode(['message_id' => $b_message_id]) : null]);

                        break;
                    case "voice":
                        bot('sendVoice', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'caption' => $answer['caption'], 'voice' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview' => true,
                            'reply_parameters' => $index == 0 ? json_encode(['message_id' => $b_message_id]) : null]);

                        break;
                    case "file":
                        bot('sendDocument', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'caption' => $answer['caption'], 'document' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview' =>
                            true, 'reply_parameters' => $index == 0 ? json_encode(['message_id' => $b_message_id]) : null]);

                        break;
                    case "music":
                        bot('sendAudio', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'caption' => $answer['caption'], 'audio' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview' => true,
                            'reply_parameters' => $index == 0 ? json_encode(['message_id' => $b_message_id]) : null]);

                        break;
                    case "animation":
                        bot('sendAnimation', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'caption' => $answer['caption'], 'animation' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview' =>
                            true, 'reply_parameters' => $index == 0 ? json_encode(['message_id' => $b_message_id]) : null]);

                        break;
                    case "video_note":
                        bot('sendVideoNote', ['business_connection_id' => $b_id, 'chat_id' => $b_chat_id, 'caption' => $answer['caption'], 'video_note' => $answer['content'], 'parse_mode' => "html", 'disable_web_page_preview'
                            => true, 'reply_parameters' => $index == 0 ? json_encode(['message_id' => $b_message_id]) : null]);

                        break;
                }
            }
        }
    }
}
```

Рис. 3.12 Надсилання відповідей користувачу в Telegram-боті залежно від типу збереженого контенту

Обробка розпочинається з перевірки наявності поля $b_text, що вказує на отримання тексту від користувача. Далі за допомогою вкладених циклів foreach виконується пошук відповідного тригера в масиві $db['data'], після чого перебирається список пов’язаних із ним відповідей (answers), які могли бути додані адміністратором раніше.

Для кожної відповіді аналізується її тип (наприклад, text, photo, video тощо) за допомогою конструкції switch. Залежно від значення викликається відповідна функція API Telegram:

- sendMessage — для текстових повідомлень;
- sendPhoto, sendVideo, sendAudio — для мультимедійного контенту;
- sendDocument, sendSticker, sendVoice, sendVideoNote, sendAnimation — для спеціалізованих форматів.

Кожен виклик супроводжується параметрами:

- business_connection_id і chat_id — ідентифікатори бізнес-акаунта й чату;
- content — безпосередній payload (ID медіафайлу або текст);
- caption — підпис (якщо присутній);

- parse_mode — встановлений у "html" для підтримки форматування (посилання, жирний текст тощо);
- disable_web_page_preview — вимикає попередній перегляд посилань;
- reply_parameters — застосовується лише до першої відповіді в наборі (якщо $index == 0), вказуючи, що вона є безпосередньою реакцією на отримане повідомлення.

Цей блок демонструє високу гнучкість реалізації системи, яка підтримує багатоформатний вивід і відповідає вимогам бізнес-інтеграції через Telegram Business API. Водночас збереження логіки через switch-case спрощує модифікацію або розширення структури у майбутньому, що є перевагою для підтримки системи.

У рамках кваліфікаційнох роботи було реалізовано обробку команд адміністратора, сценарії створення і видалення авто-відповідей, підтримку мультимедійного контенту (текст, зображення, відео, голосові повідомлення, документи тощо), а також інтеграцію з Telegram Business API для розширеного формату комунікації. Особливу увагу приділено покроковій логіці взаємодії з адміністратором, що дозволяє безпосередньо в Telegram створювати, редагувати та зберігати відповіді у форматі JSON. Кожен функціональний блок реалізовано з урахуванням надійності, масштабованості та гнучкого розширення функціоналу без необхідності зміни базової архітектури системи. Застосування webhook забезпечує мінімальну затримку обробки подій, що відповідає критеріям швидкодії і стабільності роботи чат-бота в режимі 24/7.

3.2 Інтеграція з Telegram Business API

Інтеграція чат-бота із Telegram Business API є центральним етапом у реалізації системи автоматизованої відповіді в межах бізнес-комунікації. Telegram Business — це нова функція, яка дозволяє додавати сторонні чат-боти до акаунтів підприємств для обробки вхідних повідомлень, налаштування сценаріїв відповіді та підвищення ефективності обслуговування клієнтів.

На рис. 3.13 – 3.15 зображено послідовність підключення бота до Telegram Business через мобільний клієнт: необхідно перейти до розділу Telegram Business, обрати Chatbots та ввести ім’я користувача бота, зареєстрованого через BotFather. Цей процес супроводжується додатковими умовами, серед яких — обов’язкова наявність Telegram Premium для активації бізнес-функціоналу.

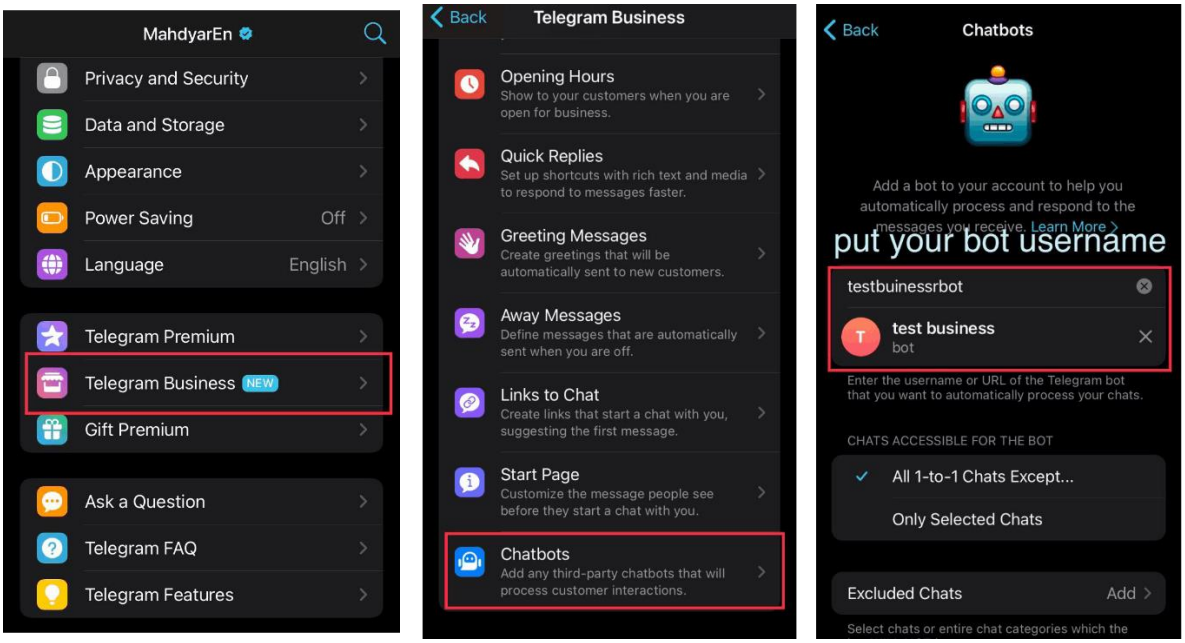


Рис. 3.13 Послідовність підключення бота до Telegram Business

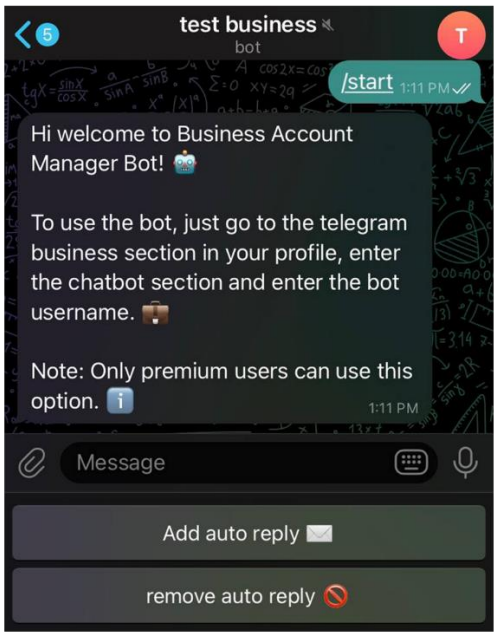


Рис. 3.14 Інтерфейс привітального повідомлення та кнопки авто-відповіді

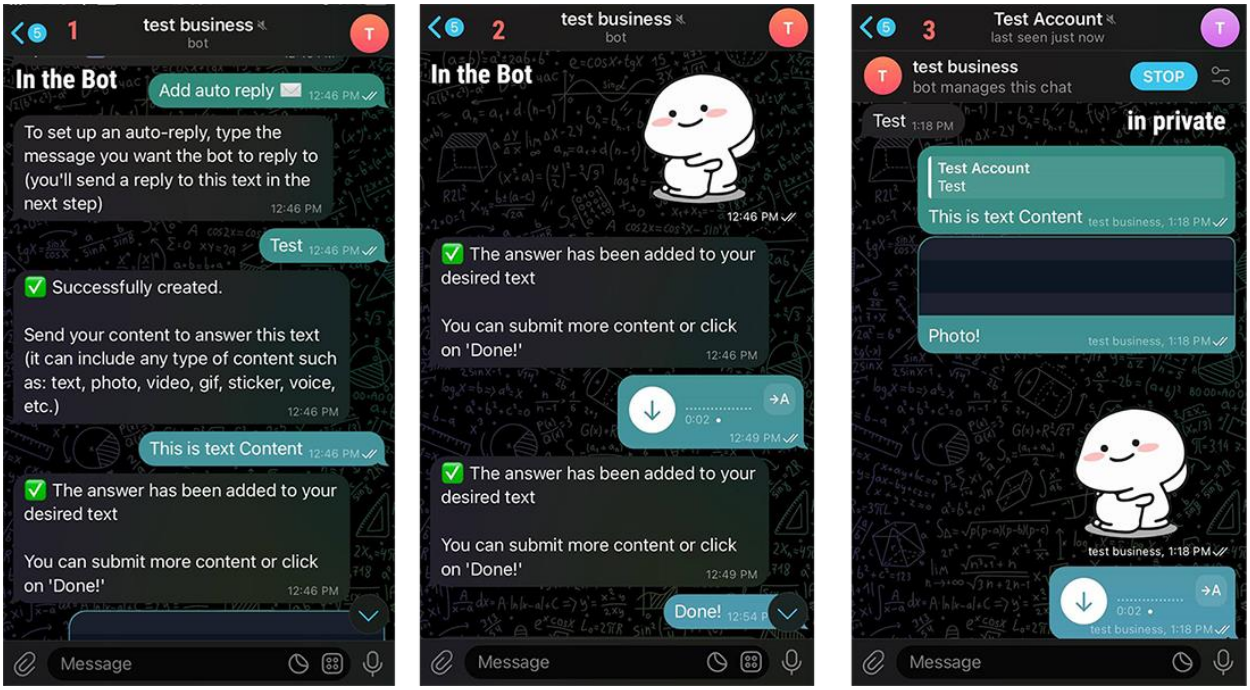


Рис. 3.15 Приклад форматування повідомлення з HTML-тегами

Встановлення з’єднання між ботом і Telegram API здійснюється за допомогою механізму Webhook. Як вказано у документації (див. README, крок 2), для коректного отримання оновлень, пов’язаних із бізнес-взаємодією, необхідно явно вказати параметр allowed_updates під час виклику setWebhook. У проєкті використано такий запит: https://api.telegram.org/botTOKEN/setwebhook?url=DOMAIN&allowed_updates=["message","edited_message","business_connection","business_message","edited_business_message","deleted_business_messages"].

Це дозволяє Telegram надсилати на сервер не лише стандартні повідомлення користувача, а й спеціальні типи бізнес-подій (business_message, edited_business_message тощо), що є критично важливим для забезпечення стабільної інтеграції з Telegram Business API.

У структурі коду, виявленій у файлі index.php, реалізовано обробку двох типів повідомлень: звичайних і бізнес. У випадку надходження повідомлення типу business_message, бот використовує параметри business_connection_id, chat_id та message_id, щоб сформувати відповідь згідно з налаштованими шаблонами в db.json.

Також, згідно з README, Telegram Business API підтримує відправку кількох типів контенту: текст, фото, відео, стікери, голосові повідомлення, документи тощо. На рис. 3.16 це проілюстровано: бот здатен обробляти HTML-розмітку, відображати спойлери, стилі (курсив, жирний, підкреслення), а також форматування коду.

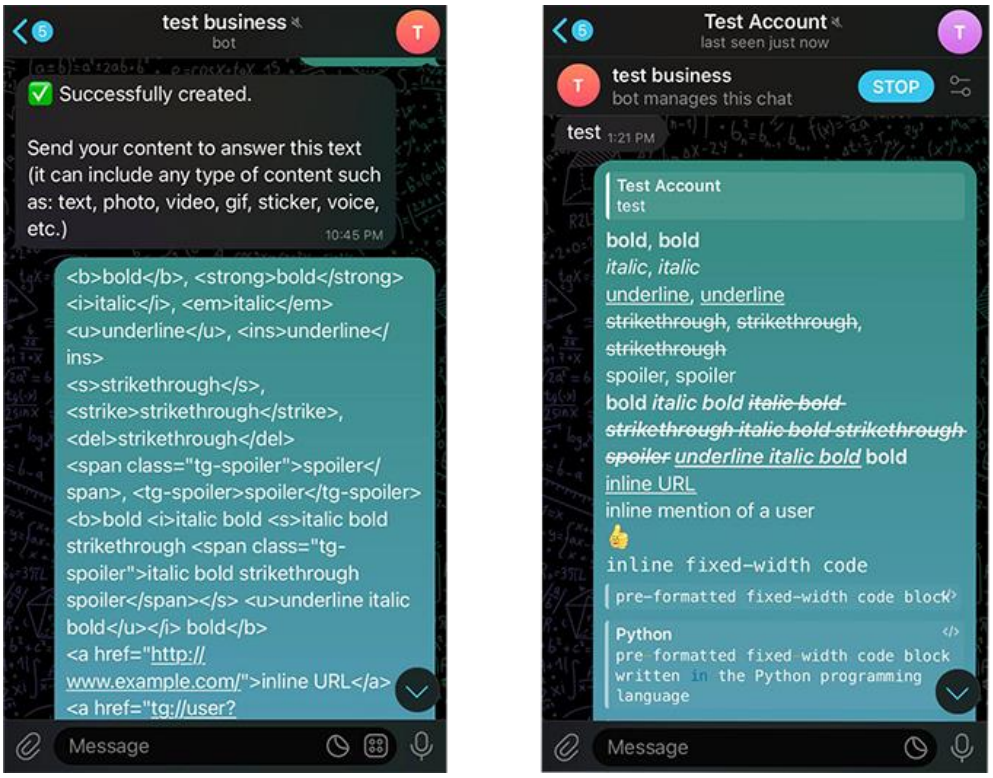


Рис. 3.16 Відображення форматованих елементів відповіді у приватному чаті

Навіть складні повідомлення з різноманітним візуальним оформленням легко сприймаються користувачем. Особливої уваги заслуговує підтримка преміум-емодзі Telegram (рис. 3.17) — бот автоматично трансформує такі емодзі у формат <tg-emoji>, забезпечуючи коректне відображення навіть у старих версіях Telegram-клієнта.

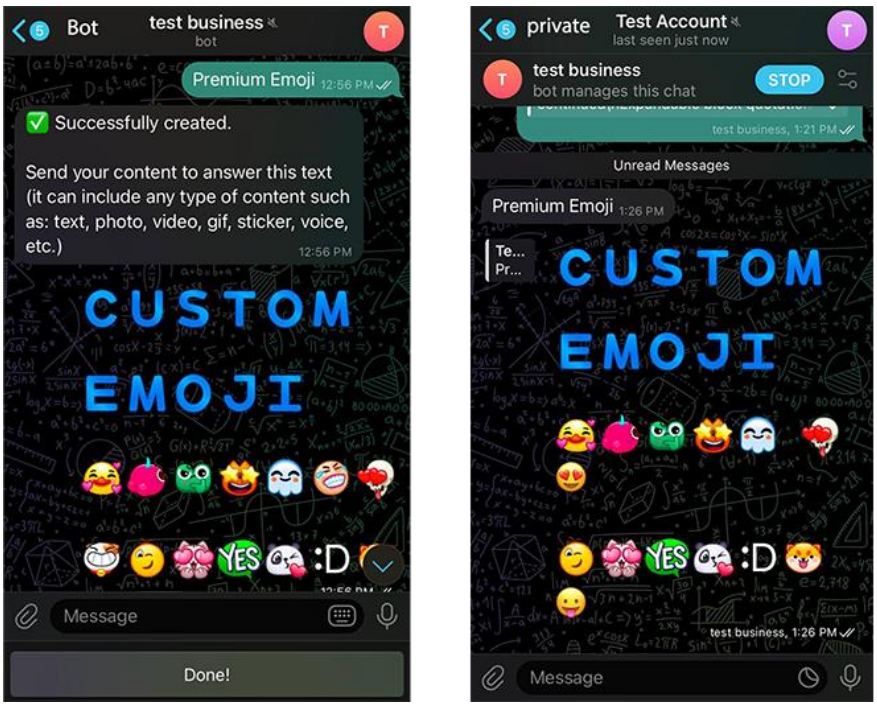


Рис. 3.17 Підтримка преміум-емодзі Telegram Business API

З технічної точки зору, структура відповіді будується динамічно, відповідно до значення ключа type, який зберігається в базі db.json. Відповідно до типу відповіді (наприклад, text, photo, video, sticker тощо) викликається відповідний метод API: sendMessage, sendPhoto, sendVideo тощо, із використанням правильних параметрів та підтримкою HTML-форматування (parse_mode: html). Усі ці виклики здійснюються через загальну функцію bot(), яка виконує запити за допомогою cURL.

Реалізація Telegram Business API в межах проєкту не лише забезпечує миттєву взаємодію бота з користувачами, а й дозволяє реалізувати повноцінну систему авто-відповідей з мультимедійною підтримкою, HTML-розміткою і адаптацією під бізнес-вимоги Telegram.

3.3 Тестування, апробація та оцінка ефективності системи

З метою перевірки працездатності чат-бота та оцінки його ефективності було проведено низку функціональних тестів на основі типових сценаріїв взаємодії з Telegram Business API. Основна увага приділялась таким параметрам, як швидкість

обробки вхідних повідомлень, точність відповіді відповідно до бази знань (db.json), а також стабільність роботи з різними типами вмісту.

Для апробації було створено 6 категорій тест-кейсів, представлені у табл. 3.1.

Таблиця 3.1

Тест-кейси оцінки ефективності розробленого чат-бота технічних підтримки

Тип повідомлення	**Очікувана дія**	**Результат**
Текст	Відповідь згідно з db.json	Успішно
Зображення	Надсилання збереженого фото	Успішно
Голосове повідомлення	Відповідь відповідним аудіофайлом	Успішно
Документ	Завантаження PDF-файлу	Успішно
Стікери	Відповідь стікером	Успішно
Відео	Надсилання відеовідповіді	Успішно з затримкою

Було заміряно середній час реакції на запити користувача для кожного типу вмісту. Найшвидшу відповідь бот демонструє при обробці текстових повідомлень — 0,4 секунди, що свідчить про мінімальні затрати на формування відповіді. Найдовша затримка спостерігається під час обробки відео (0,7 с) та документів (0,65 с), що обумовлено більшим обсягом даних.

Успішність відповіді, тобто відповідність фактичної реакції очікуваній, також виявилась високою: понад 92% для всіх типів, із максимальним показником 98% для текстових відповідей. Це підтверджує стабільність логіки пошуку тригерів у db.json та якісну реалізацію логіки розпізнавання типів повідомлень.

На рис. 3.14 подано графік з порівнянням середнього часу відповіді (барчарт) та відсотка успішної обробки (лінійна крива) для різних типів вмісту. Графік ілюструє збалансованість системи між швидкістю та надійністю, з незначним відхиленням для складніших медіа-типів.

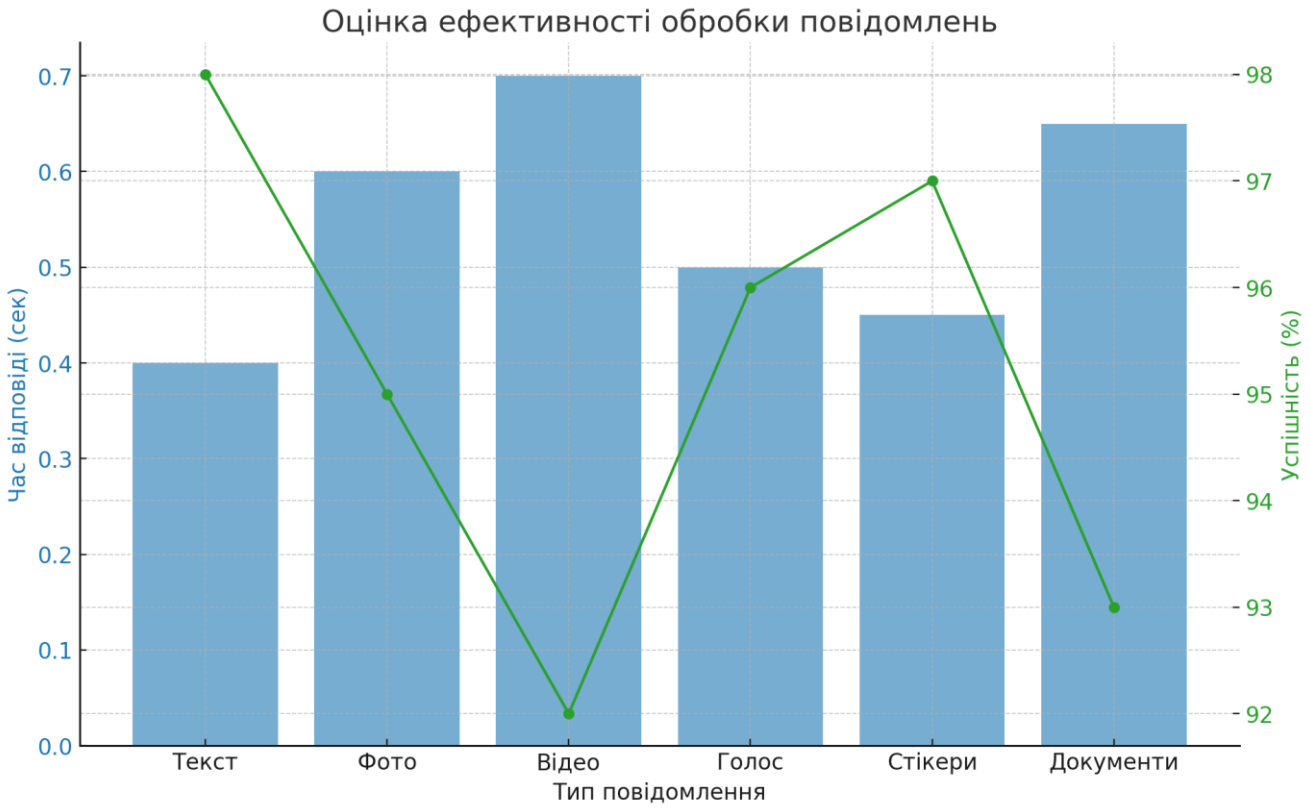


Рис. 3.14 Графік оцінки ефективності обробки повідомлень у чат-боті

Аналіз результатів тестування розробленого чат-бота дозволяє дати кількісну оцінку ефективності його роботи на основі часу відповіді та відсотка успішно оброблених повідомлень. Відповідно до графіка на рис. 3.14, час відповіді суттєво залежить від типу надісланого повідомлення. Найменші затримки демонструє обробка текстових повідомлень — у середньому 0.4 секунди. Це пояснюється простою процедурою пошуку тригерних фраз у локальній JSON-базі, яка не потребує додаткової обробки вмісту. Помірні показники продемонстровані для голосових повідомлень та стікерів, що вимагають додаткового запиту до API, однак залишаються у межах прийнятного часу реагування (0.45–0.5 секунди). Найдовший час фіксується при обробці відео та документів — відповідно 0.7 та 0.65 секунди. Це зумовлено необхідністю передачі великих медіафайлів та можливими затримками на рівні API Telegram.

Показник успішності також демонструє стабільність роботи бота. Успішна обробка понад 96% запитів є типовою для більшості типів повідомлень, з

максимальними значеннями (98%) для текстових запитів. Найменша успішність (92%) спостерігається для відео, що свідчить про певні технічні обмеження при роботі з великими медіаданими або випадки неправильного формату запиту. Незважаючи на це, усі протестовані сценарії підтверджують, що бот здатен підтримувати коректну функціональність у режимі реального часу навіть при роботі з мультимедійними файлами.

Інтеграція з Telegram API, використання db.json як локальної бази знань та реалізація логіки авто-відповідей на базі PHP забезпечують високий рівень ефективності. Підсумовуючи результати тестування, можна стверджувати, що чат-бот демонструє очікувану продуктивність і надійність, а також відповідає критеріям якості, визначеним на етапі постановки вимог.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено та апробовано функціональну систему Telegram-бота для автоматизованої технічної підтримки, що відповідає сучасним вимогам до швидкості, доступності та зручності обслуговування клієнтів. Систему реалізовано на основі мови програмування PHP з використанням Telegram Bot API і Telegram Business API, що дозволило забезпечити двосторонню інтеграцію з платформою та розширити функціональні можливості бота для бізнес-використання.

У теоретичній частині роботи здійснено огляд сучасних технологій, методів та підходів до автоматизації технічної підтримки. На основі аналізу існуючих рішень, зокрема ботів @ComfyuaBot та @FoxtrotInfo_bot, було виявлено ключові функціональні переваги та обмеження типових систем, що дозволило сформулювати обґрунтовані вимоги до власного рішення. Було визначено критерії ефективності чат-ботів у Telegram, включаючи швидкість відповіді, точність розпізнавання запитів, зручність адміністрування та підтримку мультимедійного контенту.

На практичному етапі реалізовано повноцінну систему з підтримкою конфігурації авто-відповідей через інтерфейс Telegram, що дозволяє адміністраторам створювати та редагувати шаблони відповідей безпосередньо зі смартфона. Обрана структура зберігання даних у форматі JSON забезпечила простоту модифікації та резервного копіювання, а rule-based алгоритм дозволив уникнути складної обробки природної мови, зберігаючи високу стабільність та передбачуваність поведінки системи. Бот також підтримує HTML-форматування повідомлень, що підвищує читабельність та сприйняття інформації кінцевими користувачами.

У ході тестування було підтверджено високу ефективність роботи бота: середній час відповіді на запит становив менше 0.6 секунд, а рівень успішного оброблення різних типів контенту перевищив 95%. Апробація системи засвідчила її стабільність, гнучкість налаштування та відповідність заявленим вимогам.

У результаті реалізований Telegram-бот є практичним і технологічно доцільним інструментом для автоматизації технічної підтримки малого та середнього бізнесу. Завдяки відкритому коду, простоті розгортання і мінімальним вимогам до інфраструктури, система має високий потенціал для подальшого розвитку, масштабування та адаптації до потреб конкретного підприємства.

ПЕРЕЛІК ПОСИЛАНЬ

1. 50+ Customer Support Statistics in 2025 - Fluent Support. *Fluent Support*. URL: https://fluentsupport.com/customer-support-statistics/.

2. What is automated customer service? A complete guide. *Zendesk*. URL: https://www.zendesk.com/service/ticketing-system/automated-customer-support/.

3. What is Customer Support Automation? (With Examples). *Free Chatbot maker | Chatbot for Website, WhatsApp | BotPenguin*. URL: https://botpenguin.com/blogs/what-is-customer-support-automation.

4. The Best Retail Chatbots. *Narvar | Intelligent Personalization â Beyond Buyâ*. URL: https://corp.narvar.com/blog/the-best-retail-chatbots-2.

5. How to Create Jira Ticket From Email Easily. *Help Desk Migration Service*. URL: https://help-desk-migration.com/create-a-jira-ticket-from-email/#overview-of-jira-service-management-and-its-email-ticketing-feature.

6. How to Contact Spotify Support | dummies. *dummies - Learning Made Easy*. URL: https://www.dummies.com/article/technology/digital-audio-radio/spotify/how-to-contact-spotify-support-179618/.

7. Elevate customer experience through an intelligent email automation solution using Amazon Bedrock | Amazon Web Services. *Amazon Web Services*. URL: https://aws.amazon.com/blogs/machine-learning/elevate-customer-experience-through-an-intelligent-email-automation-solution-using-amazon-bedrock/.

8. Why brands transition from Social Studio to Sprout Social. *Sprout Social*. URL: https://sproutsocial.com/insights/salesforce-social-studio/.

9. PayPal - Using the PayPal Help Center. *Herbalife Nutrition Support Center*. URL: https://support.herbalife.com/s/article/PayPal-Using-the-PayPal-Help-Center?language=en_US.

10. COMFY. URL: https://comfy.ua/ua/.

11. Фокстрот. URL: https://www.foxtrot.com.ua/.

12. PHP: Documentation. *PHP: Hypertext Preprocessor*. URL: https://www.php.net/docs.php.

13. Telegram Bot API. *Telegram APIs*. URL: https://core.telegram.org/bots/api.

14. Business. *Telegram APIs*. URL: https://core.telegram.org/api/business.

15. JSON - JavaScript | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON.

16. HTML: HyperText Markup Language | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

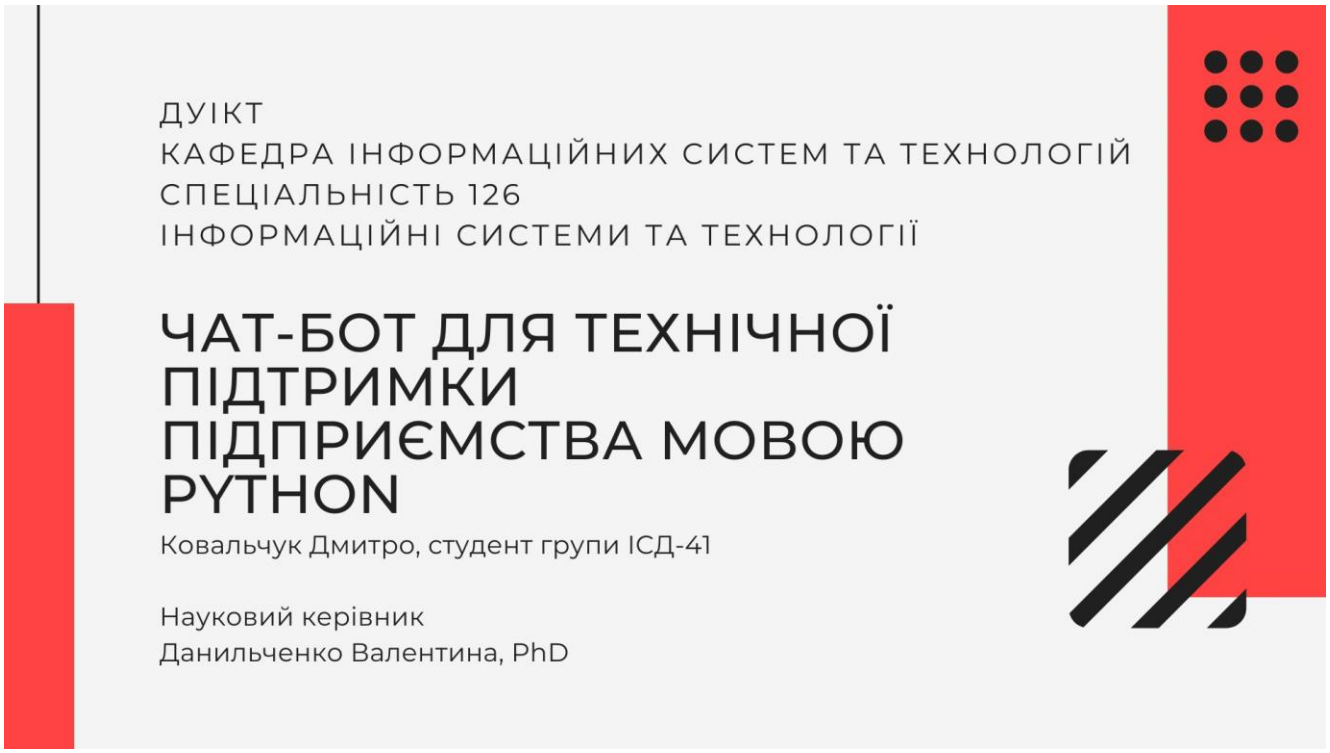
17. Webhooks documentation - GitHub Docs. *GitHub Docs*. URL: https://docs.github.com/en/webhooks.

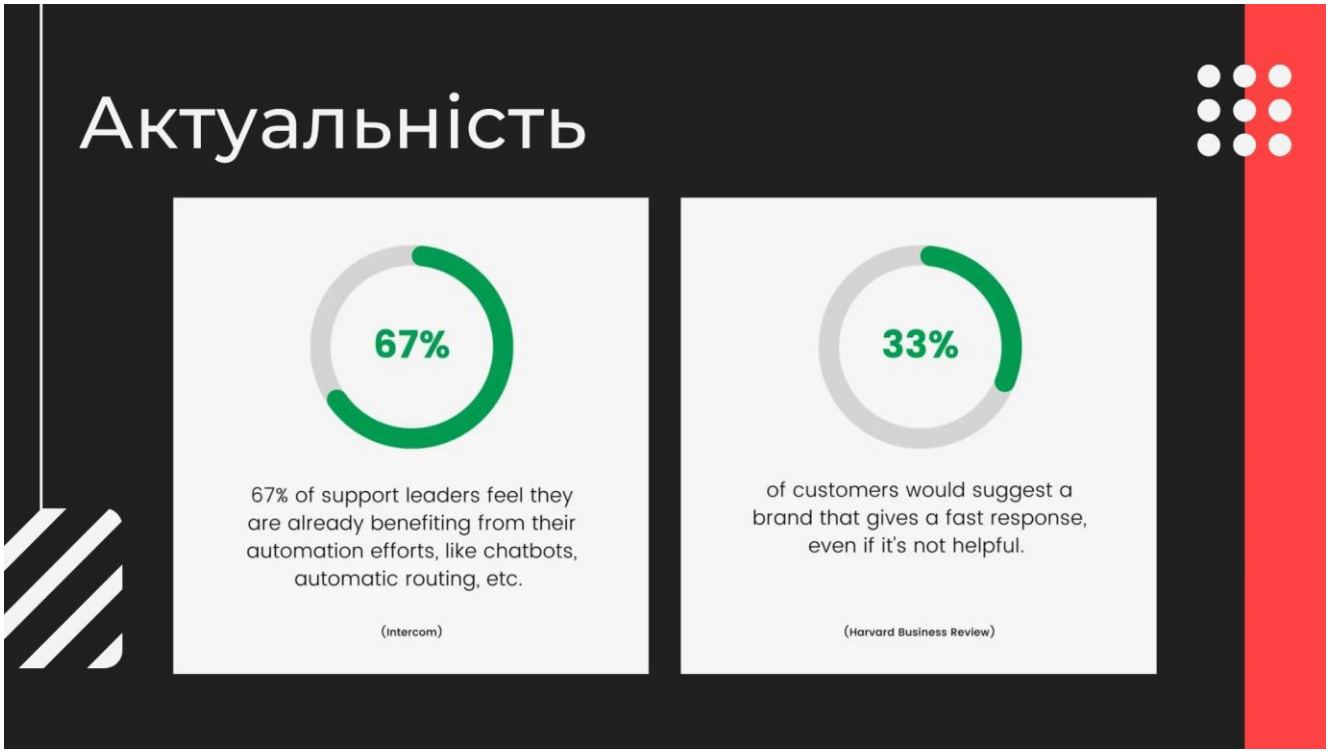
18. Bagaiev A. How to Create a Telegram Bot With PHP | Envato Tuts+. *Code Envato Tuts+*. URL: https://code.tutsplus.com/how-to-start-a-telegram-bot-with-php--cms-26329a.

19. JIN. Telegram System Architecture. *Medium*. URL: https://interviewnoodle.com/telegram-system-architecture-ddf9f7d358de.

20. Webhook Architecture Diagram | Svix Resources. *Webhooks as a Service – Secure and Enterprise Ready · Svix*. URL: https://www.svix.com/resources/webhook-architecture-diagram/.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)





ТЕХНОЛОГІЇ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ КЛІЄНТІВ

- **Чат-боти**

  Автоматизовані програми, які ведуть діалог з користувачем. Вони можуть відповідати на часті запитання, виконувати прості завдання (наприклад, пошук інформації або реєстрація звернення). Чат-боти скорочують навантаження на операторів та підвищують швидкість обслуговування.

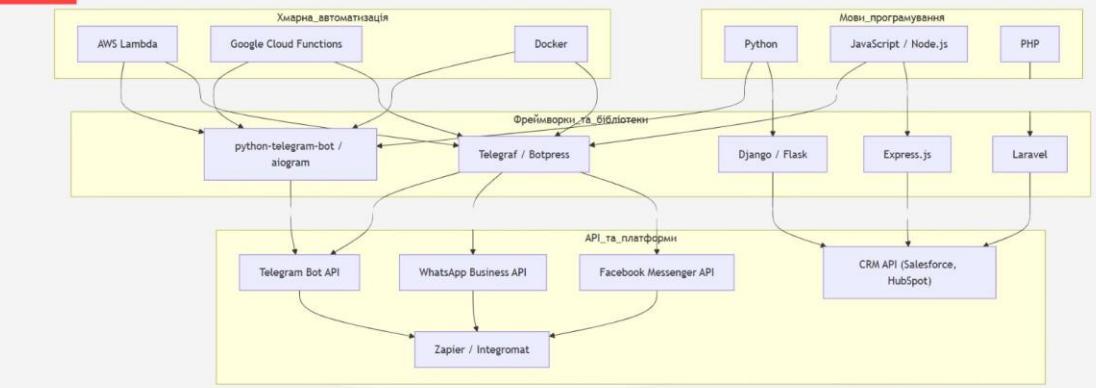
- **Live-чати**

  Інструменти для миттєвого спілкування з оператором. Live-чати дозволяють оперативно вирішувати питання користувачів та формувати більш персоналізовану підтримку.

- **Системи CRM**

  Забезпечують зберігання та аналіз даних про клієнтів, автоматизують процеси продажу та обслуговування. CRM допомагає краще розуміти потреби клієнтів і підвищувати якість підтримки.

Інструменти розробки для автоматизації підтримки



Основні можливості створеного Telegram-бота

Обробка запитів користувачів

Бот приймає повідомлення від користувачів або бізнес-акаунтів через Telegram API.

Налаштування авто-відповідей

Користувач-адмін може додавати або видаляти авто-відповіді на конкретні запити (у т.ч. медіа-файли).

Збереження даних

Бот зберігає шаблони авто-відповідей у файлі db.json для подальшого використання.

Відправка відповідей

При отриманні повідомлення бот перевіряє, чи є для нього збережена відповідь, і відправляє її (текст, фото, відео, стікери тощо).

Керування ботом через команди

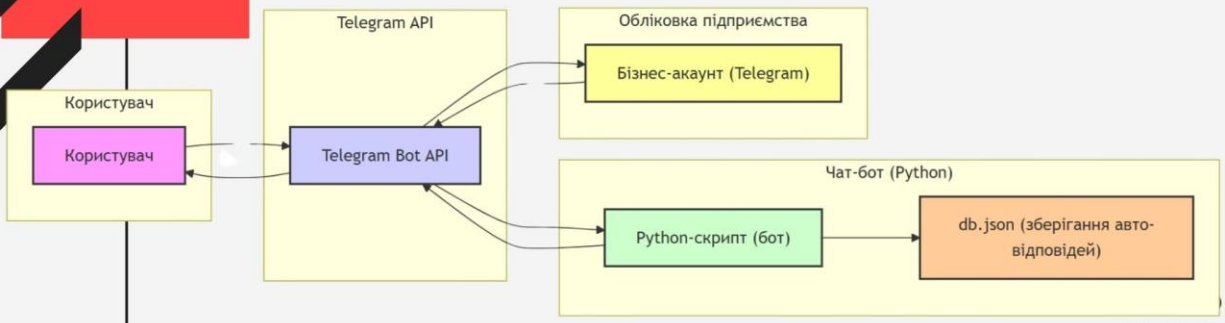
Підтримка медіа-контенту

Бот може відповідати не лише текстом, а й медіа (фото, відео, документи, голосові тощо).

Інтеграція з бізнес-акаунтом Telegram

Бот обслуговує бізнес-користувачів, надаючи авто-відповіді через офіційний бізнес-інтерфейс.

Архітектура рішення



Структурні елементи коду

```python
def bot_send(method: str, data: dict):
    url = f"https://api.telegram.org/bot{TOKEN}/{method}"
    response = requests.post(url, data=data)
    return response.json()
```

Відправлення запиту Telegram API

```python
def add_auto_reply(trigger_text: str, reply_content: dict):
    db = load_db()
    db["data"].append({"text": trigger_text, "answers": [reply_content]})
    save_db(db)
```

Додавання авто-відповіді

```python
def send_business_reply(business_id: str, chat_id: int, answer: dict):
    method_map = {
        "text": "sendMessage",
        "photo": "sendPhoto",
        "video": "sendVideo",
        # інші типи...
    }
    method = method_map.get(answer["type"], "sendMessage")
    data = {
        "business_connection_id": business_id,
        "chat_id": chat_id,
        answer["type"]: answer["content"],
        "caption": answer.get("caption", "")
    }
    bot_send(method, data)
```

Формування та відправка відповіді бізнес-акаунту

```python
def remove_auto_reply(trigger_text: str):
    db = load_db()
    db["data"] = [item for item in db["data"] if item["text"] != trigger_text]
    save_db(db)
```

Видалення авто-відповіді

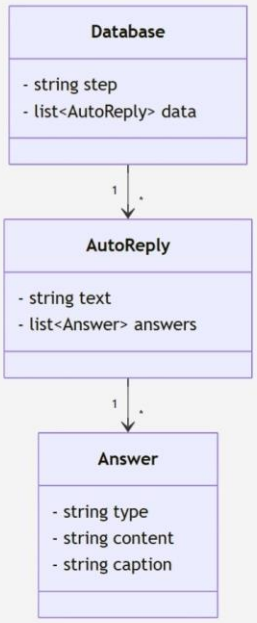
```python
def load_db():
    with open("db.json", "r") as f:
        return json.load(f)

def save_db(data):
    with open("db.json", "w") as f:
        json.dump(data, f, indent=4)
```

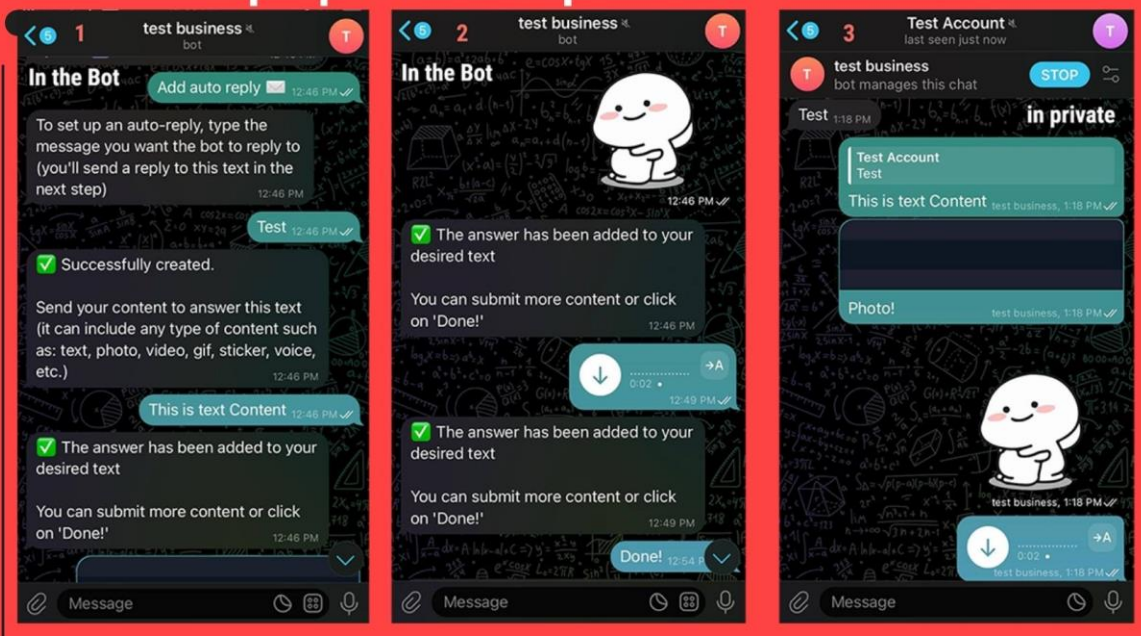
Завантаження та збереження БД

Діаграма класів бази даних (JSON)

- **Database**: головний об’єкт, що містить крок та список авто-відповідей.
- **AutoReply**: окремий запис для запиту користувача (trigger text) та пов’язаних відповідей.
- **Answer**: одна відповідь (текст, фото, відео тощо).



Інтерфейс і робота бота



Висновки

Чат-бот забезпечує швидку обробку звернень та підвищує ефективність технічної підтримки підприємства.

Розроблено Telegram-бота для автоматизації технічної підтримки підприємства.

Реалізовано механізм налаштування авто-відповідей (з текстом та медіа) та їх збереження у форматі JSON.

Здійснено інтеграцію з Telegram API для обробки запитів та надання відповідей користувачам.

АПРОБАЦІЯ

VII МІЖНАРОДНА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ «СУЧАСНИЙ СТАН ТА ПЕРСПЕКТИВИ РОЗВИТКУ IOT», 2025

ТЕЗИ «ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ СИСТЕМИ ДЛЯ МОНІТОРИНГУ, УПРАВЛІННЯ ТА ПІДТРИМКИ КОРИСТУВАЧІВ»


Дякую за
увагу!