Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Чат-бот пошуку та замовлень товарів з онлайн-магазину»

08-54.БКР.011.00.000 ПЗ

Виконав: студент 4 курсу, групи 1КІ-21б
спеціальності — 123 Комп'ютерна інженерія
освітня програма — «Комп'ютерна інженерія»
Костюк О. Д. _____

Керівник: к.т.н, доц. каф. ОТ
_____ Кадук. О. В.
Рецензент: д.т.н., професор, голова секції УБ кафедри МБІС
_____ Яремчук Ю. Є.

ДОПУЩЕНО
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д
"16" 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо — кваліфікаційний рівень бакалавр
Спеціальність 123 — Комп'ютерна інженерія
Освітньо-професійна програма – Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф._____ Азаров О. Д
"21" березня 2025 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Костюку Олександру Дмитровичу

1 Тема проєкту: «Чат-бот пошуку та замовлень товарів з онлайн-магазину», керівник роботи к.т.н., доц. каф. ОТ Кадук О. В.. затверджені наказом ВНТУ від «20» березня 2025 року №97.

2 Строк подання студентом проєкту: 13.05.2025

3 Вихідні дані до проєкту: Telegram-канал, де будуть публікуватися сповіщення про товари; бот, система якого функціонуватиме цілодобово, щоб забезпечити доступність для юзерів з різних часових зон; база даних для зберігання інформації про продукти, клієнтів та історію замовлень; середовище розробки VS Code, мова програмування Python, Telegram Bot API, а також SQLite для управління базою даних.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити): вступ, огляд існуючих чат-ботів для замовлень товарів, огляд проблематики інтеграції чат-ботів у Telegram, функціональний аналіз системи, аналіз архітектури Telegram-бота, проєктування бази даних для інформаційної системи,

вибір методів та засобів розробки, реалізація та тестування чат-бота, формув висновків за результатами розробки.

5 Перелік графічного матеріалу: ER-схема бази даних чат-бота; U схема компонентів Telegram-бота.

6 Перелік ілюстративного матеріалу: головна сторінка чат – бота; гол сторінка онлайн – магазину.

7 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдан прийн
1-3	к.т.н., доц. каф. ОТ Кадук О. В.	21.03.25	13.05.2
Нормоконтроль	ас.. каф. ОТ Швець С. І.	14.05.25	30.05.2

8 Дата видачі завдання 21.03.2025 р.

9 Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Підп
1	Постановка задачі роботи	21.03.25	
2	Пошук матеріалів та інформаційних джерел	21.03.25— 30.03.25	
3	Аналіз існуючих прототипів та визначення вимог до створюваної системи	21.03.25— 30.03.25	
4	Обгрунтування структури та принципів роботи системи	31.03.25— 13.04.25	
5	Розробка й тестування апаратно-програмної системи	14.04.25— 27.04.25	
7	Оформлення пояснювальної записки та ілюстративного матеріалу	28.04.25— 11.05.25	
8	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	13.05.25	

Студент Олександр КОСТЮК

Керівник роботи к.т.н, доц. каф. ОТ Олександр КАД

АНОТАЦІЯ

УДК 004.9

Костюк О. Д. Чат-бот пошуку та замовлень товарів з онлайн-магазину. Бакалаврська кваліфікаційна робота зі спеціальності 123 – Комп’ютерна інженерія. Вінниця: ВНТУ, 2025. 71 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 36; таб.: 3.

У кваліфікаційній роботі виконано способи впровадження автоматизації процесу пошуку-замовлення товарів в інтернет-магазині з використанням чат-ботів, інтегрованих із середовищем, наприкладі такого популярного месенджера як Telegram. Актуальність обраної теми обумовлена дуже активним розвитком електронної комерції та зростаючою необхідністю встановлення доступного й оперативного зв'язку з клієнтами.

Було розглянуто основи сучасних ботів (зокрема, Telegram-ботів), досліджено відомі рішення для автоматизації системи онлайн-продажів та визначено основні вимоги до функціональності системи. Вибір був очевидний: Python як мова програмування, python-telegram-bot як фреймворк для взаємодії з Telegram API та SQLite. Було створено архітектурне рішення для чат-бота, спроектовано структуру бази даних та реалізовано програмний модуль, що виконує пошук товарів, допомагає у процесі замовлення та зберігає історію замовлень.

ABSTRACT

Kostiuk, O. D. Chat-Bot for Product Search and Ordering from an Online Store. Bachelor's Qualification Thesis in Specialty 123 – Computer Engineering. Vinnytsia: VNTU, 2025. 71 pp.

In Ukrainian. Bibliography: 21 titles; figures: 36; tables: 3.

In this qualification work, methods for automating the product search and ordering process in an online store using chat-bots integrated into a platform—specifically the popular Telegram messenger—are implemented. The relevance of the chosen topic is driven by the rapid growth of e-commerce and the ever-increasing need for accessible and prompt communication with customers.

The thesis examines the fundamentals of modern bots (with particular emphasis on Telegram bots), investigates existing solutions for automating online sales systems, and identifies the main functional requirements for such a system. The choice was clear: Python as the programming language, the python-telegram-bot framework for interacting with the Telegram API, and SQLite for the database. An architectural solution for the chat-bot was developed, the database schema was designed, and a software module was implemented to perform product searches, assist in the ordering process, and record order history.

ЗМІСТ

ВСТУП 3

1 **АНАЛІЗ ІСНУЮЧИХ ЧАТ-БОТІВ ДЛЯ АВТОМАТИЗАЦІЇ ЗАМОВЛЕНЬ** 5

1.1 Основні принципи роботи чат-ботів 5

1.2 Популярні системи автоматизації продажів у месенджерах 9

1.3 Аналіз існуючих ботів для замовлення товарів 11

2 **АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ** 18

2.1 Вибір архітектури чат-бота 18

2.2 Проектування бази даних для збереження товарів та історії покупок 22

2.3 Обґрунтування вибору технологій (Telegram API, Python, SQLite/PostgreSQL) 27

3 **РОЗРОБКА ЧАТ-БОТА ДЛЯ АВТОМАТИЗАЦІЇ ЗАМОВЛЕНЬ** 31

3.1 Архітектура та структура інформаційної системи 31

3.2 Розробка бази даних та реалізація функції пошуку товарів 34

3.3 Розробка функціоналу отримання даних із Telegram-каналу 38

3.4 Програмна реалізація бота (замовлення, команди, обробка запитів) 43

3.5 Тестування роботи бота 46

ВИСНОВКИ 53

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 55

ДОДАТОК А Технічне завдання 57

ДОДАТОК Б Протокол перевірки кваліфікаційної роботи на наявність запозичень 61

ДОДАТОК В Графічна частина 62

ДОДАТОК Г Ілюстративна частина 64

ДОДАТОК Д Лістинг коду обробки дій користувача чат-бота 66

ВСТУП

На фоні такого стрімкого розвитку електронної торгівлі, зросла потреба в новаторських методах автоматизації обслуговування клієнтів. Одним із варіантів є чат-боти, тобто програми, здатні імітувати спілкування в реальному часі. Чат-боти, які вбудовані в популярні месенджери, зокрема, стали корисним інструментом для консультацій, замовлення продуктів, отримання технічної допомоги та навіть оплати. Вони підвищують якість обслуговування, забезпечують безперебійну комунікацію з клієнтами та зменшують навантаження на персонал.

Вибраний месенджер є одним із найпопулярніших, який використовується для бізнесу. Він є чудовою платформою для інтеграції ботів через відкритий API, просту інфраструктуру для розробників і велику активну аудиторію. Чат-боти можуть працювати з запитами, базами даних, платіжними системами та API інших постачальників. Телеграм зараз є чудовим інструментом для тисяч компаній, які спілкуються зі своїми клієнтами, особливо з малими та середніми підприємствами.

Актуальність роботи зумовлена зростаючим попитом на прості, доступні та ефективні рішення для малого бізнесу, що працює без власного сайту, використовуючи для торгівлі лише Telegram-канал. Саме Telegram, завдяки відкритому API та активній аудиторії, є ідеальною платформою для інтеграції чат-ботів, які можуть працювати з базами даних, платіжними системами та зовнішніми сервісами.

Серед ефективних рішень – чат-боти, що автоматизують процес пошуку та замовлення товарів, взаємодіючи безпосередньо з Telegram-каналами. Такі боти надають користувачу можливість перегляду товарів, отримання опису та цін, а також оформлення замовлень без необхідності переходу на сторонні вебресурси.

Метою дослідження є створення чат-бота, що дозволяє користувачам Telegram знаходити потрібні товари та оформлювати замовлення безпосередньо в месенджері. Бот буде інтегровано з Telegram-каналом магазину, забезпечуючи перегляд товарів, фільтрацію за категоріями, збереження історії покупок та зручний інтерфейс.

Для досягнення цієї мети в роботі розв’язано такі **задачі**:

— проаналізувати принципи роботи чат-ботів та існуючі рішення у сфері автоматизації продажів;

— вивчити інструменти та технології (мова програмування, API, СУБД), необхідні для створення бота;

— спроєктувати структуру інформаційної системи;

— реалізувати функції бота: пошук, перегляд, замовлення товарів

— налагодити інтеграцію з Telegram-каналом магазину;

— перевірити зручність та коректність роботи чат-бота.

Об’єкт дослідження — процес взаємодії користувача з інформаційною системою магазину за допомогою чат-бота.

Предмет дослідження — технології створення чат-ботів для Telegram, механізми пошуку товарів, обробки замовлень та зберігання даних.

Апробація результатів роботи здійснена у доповіді на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»[1].

Костюк О. Д. Кадук О. В. Чат-бот пошуку та замовлень товарів з онлайн-магазину // Матеріали всеукраїнської науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25436

1 АНАЛІЗ ІСНУЮЧИХ ЧАТ-БОТІВ ДЛЯ АВТОМАТИЗАЦІЇ ЗАМОВЛЕНЬ

1.1 Основні принципи роботи чат-ботів

Інструменти, що забезпечують ефективну взаємодію між людьми та інформаційними системами, особливо актуальні в даний час швидке зростання цифрових технологій та суспільної інформатизації. Chatbots - агенти програмного забезпечення, що імітують розмову з користувачем у текстовому або голосовому форматі, щоб запропонувати відповіді на запити,

Виконання або навігація в інформаційних умовах - серед цих технологій. Цей предмет аналізує широкі концепції чатів, їх класифікації, архітектури, основних компонентів, а також технологій, що використовуються для розробки таких систем[2].

Чат-бот - це програмне забезпечення, яке взаємодіє з користувачем через інтерфейс, що веде автоматизовану розмову. На відміну від звичайних програмних інтерфейсів, чати пропонують моделювання розмови в прямому ефірі, що інтуїтивно зрозуміло взаємодію для користувача.

Взагалі, чат-бот - це проміжний зв’язок між користувачем та програмним середовищем, завдяки якому виконуються обробка запитів, пошук інформації, транзакції чи послуги.

Основними функціями чат-бота є:

— розпізнавання повідомлень користувача;

— інтерпритація запиту;

— прийняття рішеня згідно заданою логікою;

— формування та передача відповіді користовачу;

— збереження або обробка даних, пов’язаних із сеансом взаємодії.

Чат-бот може бути вбудований у веб-сайт, мобільний додаток у велику інформаційну систему, модуль CRM або ERP, самостійний програмний агент у месенджерах або навіть голосовий асистент. Електронна комерція, технічна підтримка, банківські послуги, телемедицина, державне адміністрування, онлайн-освіта, логістика, наймання, маркетинг, внутрішні бізнес-процеси тощо

де його використання особливо ефективне.

Чат-боти класифікуються за різними критеріями, включаючи тип взаємодії, рівень інтелекту, функціональність і канали зв’язку. Такі групи є основними критеріями класифікації.

За принципом побудови діалогу існують два види — сценарні (на основі правил), які базуються на заздалегідь визначених шаблонах і логіці діалогу, і інтелектуальні (на основі машинного навчання та обробки природної мови), які забезпечують гнучку реакцію на запити, адаптацію до контексту та навчання на нових даних[3],[4].

За функціональним призначенням — інформаційні (погода, новини, відповіді на типові запитання), функціональні або операційні (замовлення товарів, реєстрація, запис на прийом, фільтрація контенту), розважальні (ігрові діалоги, вікторини, гумористичні боти).

За середовищем функціонування — голоси (Google Assistant, Siri), веб-боти (вбудовані веб-чати на сайтах), месенджери (Telegram, Viber, Facebook Messenger, WhatsApp)

Архітектура чат-бота забезпечує обробку запитів, управління дискусією та інтеграцію з зовнішніми системами, щоб він міг працювати.

Чат-бот складається з таких основних компонентів:

— інтерфейс взаємодії, який включає чат-вікно або голосовий вхід;

— модуль обробки повідомлень, який обробляє текст, виділяє ключові слова та нормалізує;

— менеджер діалогу, який контролює сценарії та перемикає стани;

— модуль інтеграції, який надає доступ до баз даних, API та сторонніх сервісів;

— система зберігання даних, яка включає сховище діалогів.

Чат-бот може працювати як однопотоковий сценарій (наприклад, Python-скрипт з обробкою запитів) або як розподілена система з використанням мікросервісної архітектури, брокерів повідомлень (наприклад, Kafka, RabbitMQ), балансування навантаження та резервування даних, у залежності від складності та масштабованості проєкту.

Базовий цикл функціонування чат-бота складається з таких етапів: отримання запиту від користувача, попередня обробка запиту (очищення, нормалізація, виявлення ключових слів), визначення інтенції (бажання) користувача, пошук відповідного шаблону або сценарію дій, формування та відправлення відповіді, а також збереження сесії для подальшої контекстної взаємодії.

Виявлення сутностей (entities), підтримка контексту, класифікація намірів і автоматичне навчання на основі розмов користувачів є прикладами складніших моделей взаємодії, які можна реалізувати за допомогою інструментів штучного мови. Найвідоміші платформи NLP включають Google Dialogflow, Microsoft LUIS, Rasa та Wit.ai. Вони підтримують багатомовність, адаптуються до тематики та є точними для розпізнавання.

Інтеграція чат-бота з каналами комунікації є ключовим етапом реалізації. Данний проєкт використовує месенджер Telegram, який надає офіційний інтерфейс Telegram Bot API. Два основні способи взаємодії бота з користувачами: Long Polling (бот опитує сервера Telegram з затримкою, отримуючи нові події) і Webhook (бот самостійно надсилає HTTP-запити на вказану адресу, щоб надсилати нові оновлення).

Python (aiogram, python-telegram-bot), Node.js (node-telegram-bot-api), Java, PHP та багато інших бібліотек для мов програмування підтримують Telegram API, який є потужним і добре документованим інструментом. Це сприяє масовому впровадженню спеціалізованих програмних чат-ботів у різних сферах, включаючи логістику, обслуговування клієнтів, технічну підтримку та електронну комерцію [4].

Існує безліч інструментів для створення потрібного нам бота, від програмованих інструментів до графічних конструкторів.

Chatfuel є однією з найпопулярніших платформ для створення ботів для Telegram, WhatsApp і Facebook Messenger. У ньому є аналітика, сегментація аудиторії, підключення до API, підтримка сценаріїв і простий зрозумілий інтерфейс. Chatfuel широко використовується для автоматизації онлайн-продажів, підтримки клієнтів і маркетингу.

ManyChat — конструктор чат-ботів, призначений для бізнес-задач. Можна працювати з Facebook Messenger, WhatsApp, Telegram, SMS і email на платформі. Маркетингові інструменти, такі як форуми захоплення лідів, автоворонки та гнучкий візуальний редактор, а також можливість підключення платіжних систем, є її основними перевагами. Автоматизація продажів, залучення потенційних клієнтів і збереження повторної комунікації є всіма перевагами ManyChat.

Tars — платформа для створення чат-ботів, зосереджена на побудові діалогових форм. Її основною метою є збільшення кількості людей, які відвідують веб-сайти, за допомогою інтерактивної взаємодії. Tars використовується у фінансових послугах, охороні здоров'я та державних сферах, надаючи користувачам зрозумілу логіку взаємодії у формі дискусій замість традиційних веб-форм.

Python і бібліотека Python-telegram-bot використовуються в даному проекті для асинхронної обробки запитів, підтримки finite-state-machine (станів діалогу), кастомних клавіатур, зручного керування callback-запитами та інтеграції з базами даних.

Такий вибір має високу швидкість, низький поріг входу, підтримку офіційного API Telegram, велику кількість прикладів і активну спільноту. Використовуючи вбудовану систему управління базами даних SQLite, інформація про товари, користувачів, замовлення та історію сеансів зберігається. Це дозволяє швидко інтегрувати з Python-додатком без необхідності додаткової серверної інфраструктури.

Таким чином, технологічна основа чат-ботів дозволяє створювати ефективні інструменти автоматизації, а гнучкість вибраного нами месенджера як платформи комунікації сприяє швидкому впровадженню цих рішень у практичну діяльність онлайн-магазинів [5].

Боти чату — це складні програмні об'єкти, які використовують алгоритмічну логіку, обробку природної мови та інтеграцію з інформаційними системами. Їхня робота базується на сценарному або інтелектуальному реагуванні, ітеративній обробці запитів користувача та забезпеченні зручного

каналу взаємодії. Для створення ефективного чат-бота, який може шукати та обробляти замовлення товарів із онлайн-магазинів, тому важливо розуміти ці принципи.

1.2 Популярні системи автоматизації продажів у месенджерах

Збільшення попиту на інструменти, які гарантують автоматизовану взаємодію з клієнтами, є результатом швидкого розвитку електронної комерції, особливо в мобільних додатках. Однією з найбільш перспективних платформ у цьому контексті є Telegram, який став привабливим середовищем для впровадження чатів та систем автоматизації бізнес-процесів завдяки його відкритим API, високому рівню захисту даних та великій активній базі користувачів. Найпоширеніші та найефективніші системи автоматизації продажів на основі телеграми, які широко використовуються в сучасній практиці торгової комерції є предметом цього розділу.

Хоча цей месенджер не є традиційною платформою електронної комерції, адаптованість її інфраструктури дозволяє інтегрувати велику кількість функцій інтернет-магазину безпосередньо в месенджер. Управління каталогами, обробка замовлень, спілкування з клієнтами, сегментація аудиторії,

Підтримка знижок, промо-кодів та інтеграція з платіжними системами містяться в цій категорії. Більшість цих функцій реалізуються за допомогою Telegram Bot або спеціалізованих сторонніх платформ, які пропонують заздалегідь виготовлені рішення автоматизації.

Адаптивність до будь-якої шкали бізнесу є унікальною характеристикою систем автоматизації, які варіюються від середніх інтернет-магазинів до окремих підприємців. Завдяки модульній архітектурі цих рішень, які можуть охоплювати різноманітні компоненти, включаючи модулі CRM, інтерфейси адміністратора, інтеграцію Google Heets, інструменти обміну повідомленнями, системи аналітики тощо.

В вибраному месенджері помітна здатність підтримувати різноманітні функції, включаючи інтегровані меню, платежі, підтвердження замовлення, медіа -контент та кнопки, крім текстових повідомлень. Це дозволяє розробити

користувальницький інтерфейс, який є інтуїтивним і не потребує користувача для виходу з месенджера.

Вивчення функціональних можливостей існуючих магазинів телеграм.

Аналіз практичних прикладів реалізованих магазинів, що реалізуються, є корисним для того, щоб отримати більш всебічне розуміння потенціалу Telegram як комерційної платформи. Це розкриває технічні рішення, що використовуються, а також методи організації діалогу користувачів, структури каталогу, систем замовлення та методів зв'язку.

У контексті секторів малого бізнесу магазини Telegram зазвичай керуються за допомогою каналів, керованих вручну або спеціальних ботів. Магазини, які спеціалізуються на одязі, косметиці, техніці, гаджетах, або продукти ручної роботи часто керують каналами, які містять зображення, описи та ціни. Асистент супутника дозволяє користувачам видавати запити, розміщувати замовлення або отримувати потрібні консультації.

Типові реалізації можуть охоплювати наступне:

— канал, що містить списки продуктів, кожен з яких супроводжується хештегами для прискореної функції пошуку (наприклад, #shoes, #sale, #new);

— бот, який має можливість проводити запити продукту за назвою чи категорією, переглядати деталі продукту, додавати елементи до кошика, вхідна контактна інформація та розміщувані замовлення;

— історія замовлень, що зберігаються в локальній базі даних;

— необов’язкове забезпечення платіжної підтримки через API Telegram payments або готівку в налаштуваннях доставки.

Деякі з більш складних реалізацій включають інтеграцію CRM, автоматичне ретаргетинг обміну повідомленнями, програми лояльності та профілі користувачів. Системи, побудовані на замовлення, які налаштовуються для певних галузей, є нормою.

Серед найпомітніших прикладів — Telegram Automaton в українському інтернет - магазині Rozetka. Незважаючи на більш обмежену функціональність порівняно з його веб-сайтом, він дозволяє користувачам переглядати рекламні пропозиції, моніторинг замовлень та отримувати сповіщення про натискання.

Деякі інші видатні фірми, які використовують месенджері — Єва - чат, який надає персоналізовані пропозиції та відповідає на запити користувачів.

Моуо - бот, який призначений для надання підтримки клієнтів та перегляду продуктів. Гаряча лінія, агрегатор бота для порівняння цін та пропозицій. Citrus, автоматизований помічник, який надає допомогу на базі месенджера.

Ці приклади є прикладом еволюції чат-ботів з простого обміну повідомленнями до середовища електронної комерції, яке має інтерактивний інтерфейс, високу персоналізацію та постійне залучення клієнтів.

Незважаючи на очевидні переваги використання телеграм-ботів для автоматизації продажів, розробка та впровадження таких систем повинні вирішити численні перешкоди:

Безпека даних: Обов’язково для впровадження шифрування, контролю доступу, перевірки даних та захисту від несанкціонованого доступу, оскільки боти взаємодіють з даними користувачів (наприклад, адресами, контактами та платежами).

Відсутність єдиної парадигми інтерфейсу в телеграмі призводить до широкого спектру якості, зручності та використання бота.

Обмеження API Telegram включають труднощі в управлінні складною логікою бізнесу (наприклад, знижки на основі кількості), обмеження швидкості запиту та обмеження формату медіа.

Юридичні міркування, різні аспекти, включаючи обробку персональних даних, вимоги публічної пропозиції, відповідальність за контент та механізми повернення політики, повинні дотримуватися відповідних законів.

Тому створення автоматичного продажу, який базується на телеграмі, потребує не лише технічного знання, але й розуміння бізнес-логіки, зручність користувача, проблеми безпеки, юридична відповідність та стабільність системи.

1.3 Аналіз існуючих ботів для замовлення товарів

Щоб бути конкурентоспроможним та забезпечити швидку підтримку споживачам, нинішня цифрова економіка закликає компанії використовувати ефективні методи комунікації. Завдяки їх пристосовності, їх прибутковість та

інтерфейс, сприятливий для користувачів, чат-боти, засновані на телеграмі, стали досить популярними.

Використання ботів телеграм показує більш загальні тенденції цифрової трансформації, де надання послуг та автоматизації в режимі реального часу є важливими.

Вибраний месенджер пропонує розробникам надійний API та багато функцій для побудови чат-ботів.

Крім основних відповідей повідомлень. Ці чат-боти можуть запропонувати підтримку в реальному часі, провести платіж, прийняти замовлення, надати доступ до каталогів товарів та керувати складними питаннями. З'єднання з внутрішніми системами, такими як CRM, бази даних продуктів та аналітичні інформаційні панелі дозволяють компаніям контролювати активність споживачів та вдосконалювати послуги.

Доступність месенджера та знайомий досвід користувачів є основними перевагами.

Взаємодія відбувається просто всередині потрібного месенджера;

Користувачам не потрібно завантажувати інші програми або реєструватися на зовнішніх сайтах. Це покращує участь та знижує складність в користуванні для користувачів.

Тип статей компанії визначають експлуатацію роботів управління продуктами.

Деякі включають пошук продуктів, управління кошиками та онлайн-платежі серед повних функціональних можливостей електронної комерції всередині телеграму.

Інші в основному грають інформативні або підтримуючі функції, надсилаючи людей до зовнішніх ресурсів або людських операторів.

У рамках цього проекту було проведено дослідження, що вносять депрест, в даний момент, які використовуються відомими українськими магазинами наведеними в таблиці 2.

Після порівняльного дослідження кожен бот у таблиці повністю описаний, тим самим визначаючи їх структуру, функціональні функції та способи участі

користувачів. Офіційний бот роздрібної мережі EVA, має на меті запропонувати довідкову інформацію та допомогти споживачам у переговорах про категорії товарів.

Таблиця 2 — Порівняльний аналіз українських чат-ботів

Назва бота	Тип товарів	Можливість пошуку	Оплата онлайн	Мова інтерфейсу	Зовнішня база даних	Автоматична обробка замовлень
@EVAofficialBot	Косметика, гігієна	Обмежена	Ні	Українська	Google Sheets	Частково
@Allohelp_bot	Електроніка, аксесуари	Так	Ні	Українська	Інтеграція з API	Частково
@MOYObot_bot	Комп'ютерна техніка, гаджети	Так	Ні	Українська	Власна БД	Частково
@Rozetka_helpBot	Електроніка, гаджети, побутові товари	Так	Так	Українська	CRM – система	Повна

Користувачі бота можуть знаходити найближчий магазин, зареєструватися на програму лояльності, дивитися нові приїзди та дізнатись про поточні угоди. Пошук продукту дещо реалізується - в основному за вибором категорії. Бот веде споживачів до офіційних сайтів або зовнішніх підприємств, але не дозволяє їм замовляти чи платити.

Хоча він має обмежені функції, бот - чудовий інструмент для маркетингу та взаємодії з брендом представлений на рисунку 1.1.

Серед українських магазинів Allohelp_bot - один з найбільш багатих на функції ботів. Від вибору продукту до розміщення замовлення, він забезпечує цілий цикл взаємодії споживачів, включаючи вибір онлайн товару та його оформлення. Бот пропонує надійну пошукову систему, дозволяючи вам досягти персоналу підтримки, інформації про ціну та акції в режимі реального часу та гнучкий перегляд каталогів. Використання власних API компанії гарантує велику швидкість та надійність. Чат-бот представлений на рисунку 1.2.

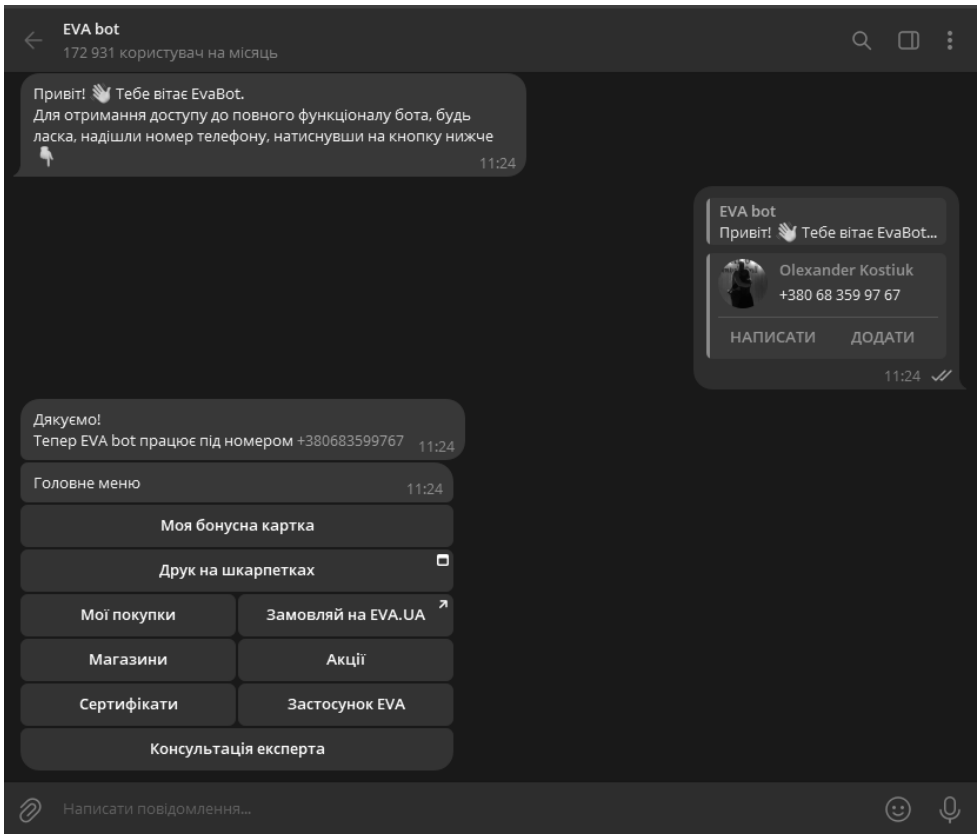


Рисунок 1.1 — Інтерфейс чат-бота EVAofficialBot мережі EVA

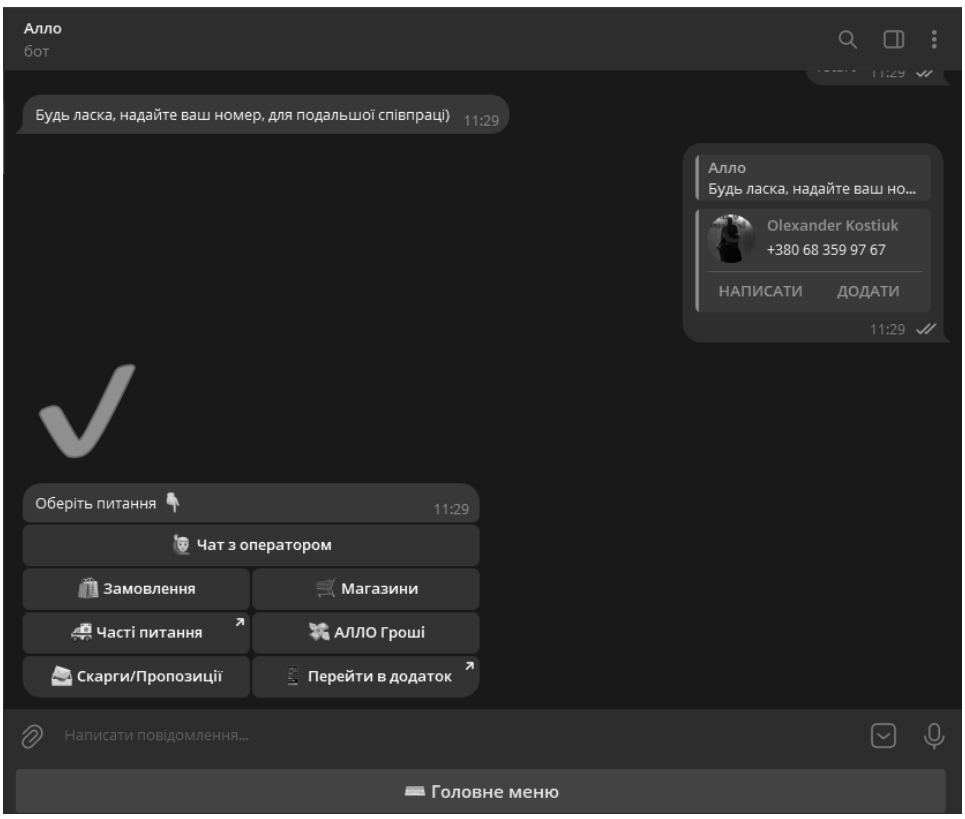


Рисунок 1.2 — Інтерфейс чат-бота Allohelp_bot від мережі АЛЛО

В основному, MOYObot_bot, який означає роздрібну фірму Moyo, надає поради та інструктивні інструменти.

Він пропонує основну інформацію про продукцію, перегляд категорій та контактну інформацію для фактичних роздрібних магазинів. Замовлення не можна розмістити повністю. Часткова автоматизація складається з варіантів взаємодії з людськими операторами або формами запиту зворотного дзвінка. Бот збільшує довіру споживачів і прокладає шлях для екосистеми послуг компанії як навчального інструменту також представлений на рисунку 1.3.

Rozetka_helpBot - чат для обслуговування клієнтів, призначений для забезпечення підтримки клієнтів. Основними його цілями є запропонувати поради, керувати питаннями та відповідати на те, що стосується доставки, гарантійних послуг, повернення коштів, і більше. Бот інтегрується до внутрішньої системи CRM компанії та має прямий, простий у користуванні інтерфейс користувача. Більшість запитів користувачів завершуються автоматизованою маршрутизацією без участі людини. Він показує справедливе застосування ботів у утриманні клієнтів та допомоги після продажу. Чат-бот представлений на рисунку 1.4.

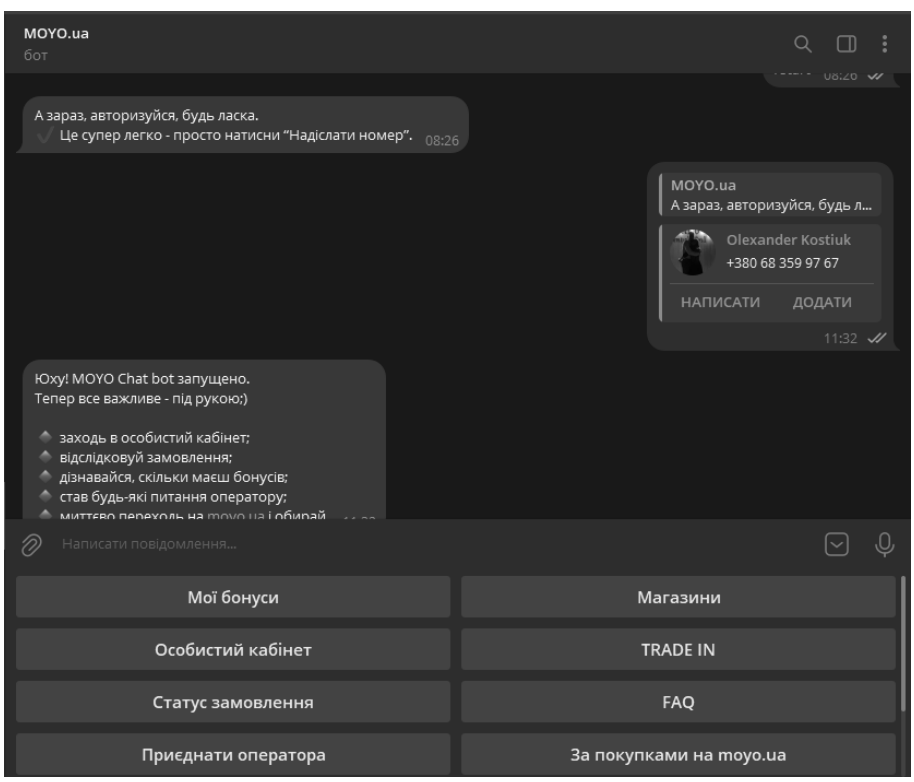


Рисунок 1.3 — Інтерфейс чат-бота MOYObot_bot від мережі MOYO

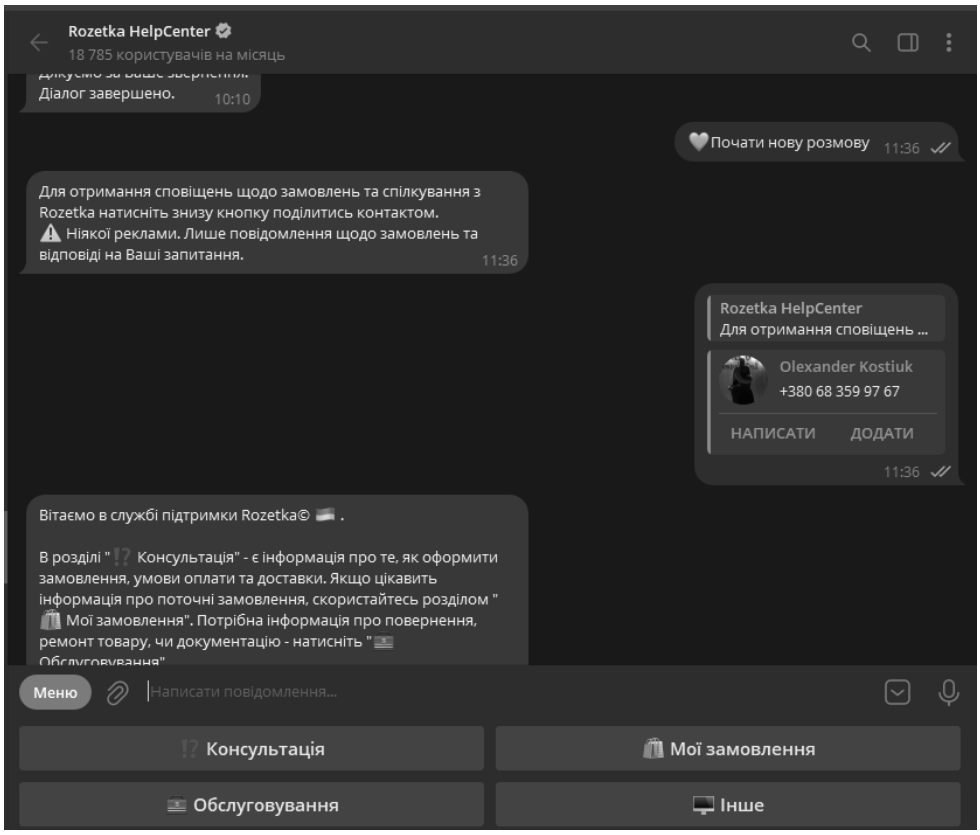


Рисунок 1.4 — Інтерфейс чат-бота Rozetka_helpBot від мережі Rozetka

Дослідження показує помітні функціональні зміни між ботами.

Хоча деякі, як Rozetka_helpBot, пропонують повний досвід покупки всередині месендежра, інші, як EVAofficialBot, підкреслюють інформацію та допомогу. Ці відмінності показують менталітет кожного бізнесу та характер їх вибору продукту. Інтеграція бази даних заслуговує на особливу увагу.

Хоча великим компаніям потрібні повні CRM або власні бази даних, щоб запропонувати швидкий доступ до інформації, масштабованості та якості даних за допомогою аркушів Google, є розумним рішенням для невеликих підприємств. Інтеграція внутрішньої системи також дозволяє проводити аналітику - відстеження ефективності маркетингової кампанії, поведінки клієнтів та популярності продукту.

Ще одним важливим елементом є дизайн взаємодії користувачів. Деякі використовують традиційний командний вхід, інші надають інтерактивні кнопки на розгалужувальні меню з цікавим функціоналом. Більш складні боти включають персоналізацію: вдосконалений потік розмови, пропозиції продукту на основі активності користувачів та збереження історії покупки. У таких

ситуаціях, чат-бот виступає як програма лояльності клієнтів та інструмент продажу. Зручність користувача в основному визначається часом реакції, простою навігацією та мінімальним необхідним введенням користувача. Технічно надійний бот повинен включати одночасну обробку запиту, керувати сеансами, обробляти збої та оптимізувати пошук баз даних.

В деяких ситуаціях, виконання функції платежів надає захист даних та конфіденційності особливе значення. Ключові елементи успіху - надійність та масштабованість. Боти з високим навантаженням повинні включати можливості відновлення при аварійних ситуаціях, інструменти адміністрування, резервні копії та журнал. Відсутність цих компонентів може спричинити велику шкоду репутації бренду та втрати споживачів[11].

Враховуючи все, вивчення чат-ботів, які займаються українським бізнесом, виявляють кілька стратегій впровадження. Успішні боти мають автоматизовану обробку замовлень, гнучку архітектуру, інтеграцію системи, масштабований дизайн та настроювані інтерфейси. Особливо щодо навігації, безпеки даних, простоти введення і ефективність обробки, ці якості слід враховувати в еволюції чат-бота, запропонованої в цьому кваліфікаційному проєкті. У нашому месенджері, ретельна стратегія забезпечить сильний, швидкий механізм замовлення товарів, орієнтований на користувачів.

2. АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

2.1 Вибір архітектури чат-бота

Ретельне планування вибору архітектури необхідно для створення ефективного чат-бота основаного на месенджера Telegram, який може автоматизувати пошук і замовлення товарів. Архітектура є основою, яка визначає логіку взаємодії між компонентами системи. Це також визначає відповідальність, рівень надійності та масштабованості, а також метод обміну даними. Здатність бота запускати та стабільно працювати на персональному комп’ютері користувача при мінімальних витратах ресурсів є основною вимогою, оскільки цей проект не використовує хмарну інфраструктуру. Підрозділ вибрав архітектуру, яка відповідає цим вимогам, зосереджуючись на простоті використання, автономності та інтеграції з Telegram Bot API.

Функціональний чат-бот забезпечує зв’язок між логікою користувача та сервером, обробляючи запити, вибираючи продукти, зберігаючи дані про покупки та надаючи зворотний зв’язок. З використанням Python і бібліотеки python-telegram-bot вибрано локальну архітектуру, щоб реалізація була ефективною та простою. Ця архітектура є найкращою для локального тестування, оскільки вона дозволяє створювати MVP (Minimum Viable Product) і демонструвати функціональність без додаткових витрат на хостинг або серверну потужність.

Спеціалізований чат-бот складається з декількох функціональних частин:

— Інтерфейс користувача — вікно, у якому користувач має можливість взаємодіяти з ботом, надсилати запити, і обирати можливі дії;

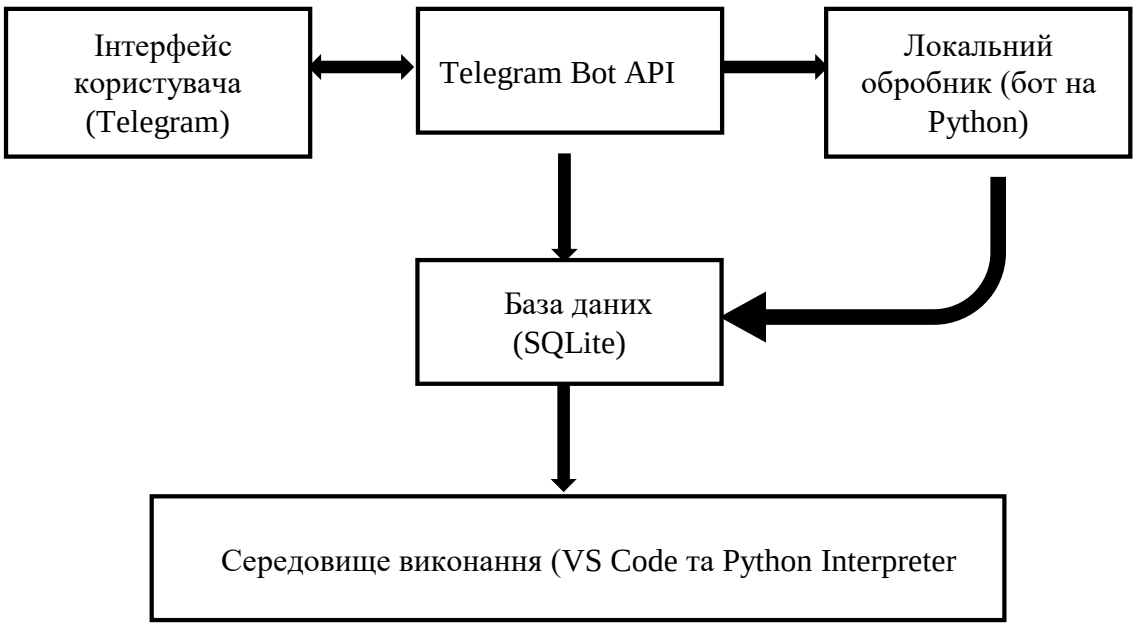
— Telegram Bot API — офіційний інтерфейс, який передає запити до бота, а потім відповіді назад до користувача. Він функціонує як посередник між Telegram-сервером і локальним скриптом бота;

— Локальний обробник (бот на Python) — ядро застосунку, яке приймає вхідні запити, визначає тип запиту, формує логіку відповіді, звертається до бази даних, фільтрує товари, оформлює замовлення, та генерує вихідне повідомлення;

— Бази даних (SQLite) — локальне сховище, що зберігає інформацію про товари, категорії, користувачів, історію запитів і замовлень [5],[12];

— Середовище виконання — Visual Studio Code та мова програмування Python, середовище розробки, у якому створюється, редагується, тестується, та запускається програма наведено на схемі 2.1.

Схема 2.1 — Архітектурна схема чат-бота з локальним виконанням



На схемі показано, як користувач надсилає запит через Telegram, який надходить до локального скрипта через API Telegram Bot. Далі бот обробляє запит, звертається до бази даних і створює відповідь, яку потім повертає користувачу[13].

Однопоточність — бот працює в одному потоці в межах даної реалізації, що дозволяє зменшити складність реалізації; однак при значному навантаженні оптимізація може бути необхідною.

Локальність — бот запускається в середовищі розробника (на комп’ютері) і працює, поки термінал або вікно Visual Studio Code відкриті. Це хороший варіант для демонстрацій, тестування, невеликих локальних магазинів або Telegram-каналів[14].

Автономність означає, що рішення не вимагає зовнішнього сервера, що робить його економічним і підходить для студентських або навчальних проектів.

Безпечність — ризик витоку особистих даних зменшується, оскільки дані

не передаються на сторонні сервери.

Було порівняно дві основні бібліотеки Python, які використовуються для створення чат-бота який оснований на месенджері Telegram — python-telegram-bot і aiogram, в рамках обраної архітектури.

У таблиці 2.1 наведено основні зміни, які вплинули на вибір бібліотеки, яка була використана в цьому проекті.

Таблиця 2.1 — Порівняння бібліотек для реалізації чат-бота

Параметр	Python-telegram-bot	Aiogram
Тип	Синхронна	Асинхронна
Простота використання	Висока	Середня
Поріг входу	Низький	Вищий
Документація	Дуже хороша	Хороша
Підтримка callback	Так	Так
Потреби в async	Немає	Обов’язкова
Підходить для локального запуску	Підходить ідеально	Можливо, але складніше реалізувати

Для забезпечення підтримки та розширення архітектури, структура коду буде мати модульний вигляд:

— main.py — запуск бота, основний цикл отримання повідомлень;
— handlers.py — обробники команд /start, /search, /order тощо;
— db.py — функції для з’єднання з SQLite, CRUD-операції;
— config.py — збереження токену, назви БД, налаштувань;
— keyboards.py — функції для створення inline та reply клавіатур.

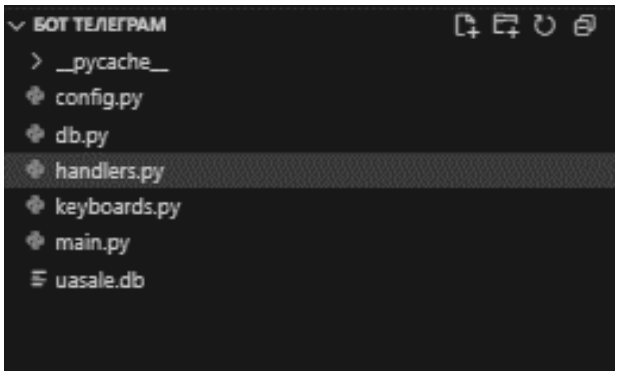


Рисунок 2.2 — Структура файлів чат-бота у середовищі VS Code

Окрім лише логістичного розуміння того, як було здійснено чат-бот, існує багато практичних та архітектурних переваг для реалізованої структури бота.

Наприклад, для такого невеликого масштабного проекту-включаючи науковців, прототипування, або навіть невеликі комерційні починання - багато міркувань з розробки/тестування та безпеки виконуються за допомогою місцевого середовища виконання.

Наприклад, одним із найвигідніших аспектів цієї архітектури є цикл зворотного зв'язку з розробкою дуже тісно. Оскільки все працює і розміщується на локальному комп’ютері розробника, будь -яке коригування до логіки, UX, або бази даних можна застосовувати та перевірити майже негайно. Це зменшення очікування до розвитку дозволяє швидко реагувати на щойно виявлені потреби чи помилки.

Крім того, архітектура дозволяє отримати повний контроль доступу користувачів та дані про продукт. Це особливо ефективно на етапах розвитку та тестування.

Через відсутність віддалених баз даних або хостингу не існує можливості ризику для існуючих даних розробників або конфіденційних даних клієнтів. Таким чином, це регульоване розповсюдження інформації є конфіденційною.

Крім того, архітектура дозволяє досягти масштабованого поступово.

Незважаючи на те, що він працює як локальний сценарій після розгортання, висококомпоненталізований характер проекту з різними обробниками, операціям для файлів, бази даних та конфігурації та окремими функціями забезпечує відмінну основу для хмарного переміщення або контейнсеризація. Майже не потрібно рефакторингу.

Крім того, за допомогою Python-Telegram-Bot, тривалі механізми опитування є рівномірними, як це призначено для нерозгорнутих додатків. Наприклад, розгорнуті програми, які використовують Webhooks, потребують 24/7 в інтернеті,

Публічний сервер, який може відповісти на запити HTTPS, щоб підтримувати постійне з'єднання з серверами Telegram; загальнодоступний в режимі офлайн.

Ця програма спрощує процес для того, щоб працювати на початковому рівні та неускладнювати розробку для розробників-початківців.

Для того, щоб проект продовжував працювати в майбутньому, важливо мати правильний запис архітектурного дизайну. У кодуванні ремонтопридатності та співпраці в команді можуть допомогти чіткі макети папок, вбудовані коментарі та блок-схеми, що показують, як обробляються повідомлення по модулях. Цей метод забезпечує прозорість і створює основу для переходу до більш складної архітектури обслуговування проекту.

Враховуючи це, вибраний дизайн є найкращим способом поєднати перспективне зростання та практичну простоту. Він задовольняє основні вимоги телеграм-ботів, включаючи економічну ефективність, конфіденційність користувачів, простий розвиток і логічне розділення проблем, необхідні для роботи в обмежених або навчальних середовищах [6]. Вимоги кваліфікаційного проєкту не тільки відповідають цій рамці, але й створюють основу для розробки повного комерційного рішення.

2.2 Проектування бази даних для збереження товарів та історії покупок

Одним із найважливіших компонентів інформаційної системи є база даних. Якщо чат-бот розроблений у месенджері Telegram для автоматизації пошуку та замовлення товарів, він виконує функцію централізованого сховища, де зберігається важлива інформація про користувачів, асортимент і замовлення. Таким чином, важливим є правильне проектування структури бази даних, її логічна побудова та технічна реалізація.

Під цей проєкт було рішення вибрати SQLite, реляційну систему управління базами даних, яка працює як один файл і не вимагає встановлення додаткового сервера. Оскільки SQLite є простим, швидким і інтегрованим у Python за допомогою модуля sqlite3, вона ідеально підходить для Telegram-ботів, які працюють на локальному рівні[17].

Основні переваги SQLite:

— не потребує встановлення або підключення до мереж;

— має високу продуктивність при роботі з малими і середніми об’ємами

даних;

— повністю зберігається у файлі формату .db;

— підтримує транзакції, зовнішні ключі та базові SQL-команди.

Структура бази даних:

— категорії товарів;

— користувачі;

— замовлення;

— товари;

— склад замовлень.

Структура бази даних включає п'ять основних таблиць, кожна з яких виконує специфічні функції в системі:

— categories (категорії товарів) — таблиця зберігає інформацію про категорії товарів, що дозволяє організувати каталог у зручній ієрархічній структурі. Категоризація товарів є важливою для зручності навігації користувачів та ефективного пошуку.

— products (товари) — центральна таблиця системи, що містить повну інформацію про товари. Окрім базових атрибутів (назва, опис, ціна), таблиця включає посилання на зображення товару та зв'язок з категорією через зовнішній ключ.

— users (користувачі) — зберігає профільну інформацію користувачів Telegram. Особливістю є збереження telegram_id, що є унікальним ідентифікатором користувача в месенджері та дозволяє встановити зв'язок між обліковим записом у боті та профілем Telegram.

— orders (замовлення) — містить основну інформацію про замовлення: дату створення, статус виконання та зв'язок з користувачем. Статус замовлення може приймати значення: "створено", "підтверджено", "в обробці", "відправлено", "виконано".

— order_items (склад замовлень) — Реалізує зв'язок багато-до-багатьох між замовленнями та товарами, зберігаючи кількість кожного товару в замовленні. Ця таблиця є ключовою для формування детального складу замовлення.

Отже ці таблиці пов’язані між собою через зовнішні ключі. Наприклад, є товар і він належить до певної категорії, а кожне зроблене замовлення належить певному користувачу.

Приклад коду SQL-структури таблиць:

```
CREATE TABLE categories (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL
);
CREATE TABLE products (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    price REAL NOT NULL,
    category_id INTEGER,
    image_path TEXT,
    FOREIGN KEY (category_id) REFERENCES categories(id)
);
CREATE TABLE users (
    id INTEGER PRIMARY KEY,
    telegram_id TEXT NOT NULL,
    username TEXT,
    first_name TEXT,
    last_name TEXT,
    phone_number TEXT
);
CREATE TABLE orders (
    id INTEGER PRIMARY KEY,
    user_id INTEGER,
    created_at TEXT,
    status TEXT,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
CREATE TABLE order_items (
    id INTEGER PRIMARY KEY,
    order_id INTEGER,
    product_id INTEGER,
    quantity INTEGER,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```

Цей зразок дає змогу реалізувати повноцінну логіку замовлень, включно з вибором товару, збереженням потрібних даних користувача і переглядом історій покупок.

Для взаємодії з базою у чат-боті буде використовуватися стандартна бібліотека sqlite3 [10].

Приклад коду зчитування товарів за категорією:

```
import sqlite3
conn = sqlite3.connect("shop.db")
cursor = conn.cursor()
cursor.execute("SELECT * FROM products WHERE category_id = ?", (1,))
products = cursor.fetchall()
conn.close()
```

Цей код зчитує всі товари з певної категорії та повертає їх у вигляді списку, який можна передати через бот користувачеві для показу.

Розробка бази даних є важливою частиною чат-бота, оскільки вона зберігає та зв’язує всі необхідні дані, включаючи історії замовлень користувачів і списки товарів. У цьому підрозділі було визначено логічну структуру бази, яка охоплює основні елементи інформаційної системи: товари, категорії, користувачі, замовлення та склад замовлення. Кожна з них виконує певну функцію, а відповідно до реляційної моделі зовнішні ключі організовують зв’язки між таблицями.

SQLite є хорошою системою управління базами даних, оскільки вона проста, не потребує багато ресурсів і сумісна з Python, що дозволяє інтегрувати базу безпосередньо в локальне середовище розробки. Конструкція таблиць полегшує виконання запитів, підтримує структурованість і запобігає надмірній кількості даних [7],[8].

Перший етап розвитку орієнтований на обслуговування до тисячі активних користувачів. На цьому етапі SQLite залишається основною СУБД, але впроваджуються механізми оптимізації продуктивності. Основний акцент робиться на створенні ефективних індексів, впровадженні кешування найчастіших запитів у оперативній пам'яті та налаштуванні системи автоматичного резервного копіювання. Також на цьому етапі важливо

впровадити моніторинг продуктивності для своєчасного виявлення вузьких місць.

Другий етап призначений для роботи з аудиторією від тисячі до десяти тисяч користувачів. Ключовою зміною стає міграція на PostgreSQL, що дозволить забезпечити ефективну роботу з конкурентними запитами та більшими обсягами даних. На цьому етапі впроваджується пул з'єднань для оптимізації роботи з базою даних, а також система реплікації master-slave для розділення операцій читання та запису. Це значно підвищує загальну продуктивність системи та забезпечує можливість горизонтального масштабування.

Третій етап розрахований на роботу з понад десятьма тисячами активних користувачів. На цьому рівні стає необхідним впровадження кращої бази даних, можливо за географічним принципом або за типами даних. Також доцільним є використання NoSQL рішень для зберігання аналітичних даних та впровадження системи розподіленого кешування з використанням CDN для статичного контенту.

Подальший розвиток системи передбачає інтеграцію з широким спектром зовнішніх сервісів. Інтеграція з платіжними системами дозволить підтримувати різні способи оплати, включаючи банківські карти, електронні гаманці та криптовалюти. Система буде відстежувати статус платежів та автоматично оновлювати стан замовлень.

Інтеграція з сервісами доставки забезпечить автоматичний розрахунок вартості доставки, відстеження посилок та надання користувачам актуальної інформації про статус їх замовлень. Система підтримуватиме роботу з декількома службами доставки одночасно, вибираючи оптимальну за ціною та швидкістю.

Структура бази даних розроблена відповідно до принципів нормалізації та відповідає стандартам побудови інформаційних систем. Це забезпечує міцну основу для подальшої роботи бота, включаючи зберігання, обробку та відображення даних у його логіці.

Розвиток системи передбачається у три основні етапи відповідно до

зростання кількості користувачів та навантаження на базу даних.

2.3 Обґрунтування вибору технологій (Telegram API, Python, SQLite/PostgreSQL)

Розробка програмних систем, викликає ретельний вибір технологій, що гарантують не тільки функціональність, але й надійність системи, масштабованість, простоту та сумісність з сучасними технологічними ресурсами.

Виходячи з логічної експертизи та захисту основних інструментів проекту: мова програмування Python, API Telegram Bot та системи управління базами даних SQLite.

Вибір мови програмування Python базується на його великій ступені абстракції, чіткий синтаксис, універсальність, і велика бібліотечна екосистема призвела до того, що Python був обраний як основна мова для проєкту. Багато в чому через наявність конкретних бібліотек для інтеграції Telegram API, Python став фактичним стандартом для розробки великої кількості чат-ботів.

Telegram пропонує API Telegram Bot як формальний інтерфейс для створення бота. Серед багатьох його функцій цей API дозволяє надсилати та приймати повідомлення, обробку зворотного виклику, обробку медіа, режим вбудованого режиму та інтерактивні клавіатури. API дозволяє повний цикл взаємодій між користувачем та ботом при використанні разом з бібліотекою Python - Telegram -Bot.

Добре задокументований, дуже надійний і не потребуючи складних процесів аутентифікації лише токен бота все що необхідно для Telegram Bot API. Крім того, він пропонує два способи спілкування: тривале опитування (підходить для локального тестування) та Webhook (використовується в розгортаннях на основі сервера).

Цей проект оцінює дві системи управління базами даних SQLite, для локальної версії бота та Postgresql як можливу відповідь на масштабований сервер. На основі реалізації залежно від сценарію розгортання. Хоча вони є обидва реляційними системами баз даних, їх архітектурний дизайн та

призначення використовується по різному[15],[16].

На основі файлів та простоти, SQLite підходить для локальних налаштувань розробки або вбудованих програм для однозабезпечення. PostgreSQL- це потужний об'єкт добре підходить для систем високого навантаження, оскільки він підтримує одночасний доступ, гнучкість та відповідність стандартів SQL. Нижче наведена порівняльна таблиця вибраних технології в таблиці 2.2.

Обгрунтування вибору технологій пройшло успішно, використання для розробки чат-бота буде базуватися на аналізі функціональних, технічних та експлуатаційних характеристик кожного з обраних інструментів. Мова програмування Python, з поєднанням бібліотеки python-telegram-bot, дозволяє створити ефективний механізм для взаємодії з Telegram API без потреб в складній конфігурації або зовнішніх залежностях.

SQLite обрана як оптимальне рішення для зберігання даних у локальному середовищі завдяки своїй легковажності, відсутності потреби у сторонньому сервері та простоті інтеграції з додатком. Ця СУБД забезпечує високу продуктивність для невеликих та середніх обсягів даних, що робить її ідеальним вибором для прототипування та локальної розробки.

PostgreSQL була визначена природною альтернативою для подальшого масштабування даного проєкту у серверному або хмарному середовищі. Завдяки своїй потужності, розширеній функціональності та надійності при роботі з великими обсягами даних, PostgreSQL дозволить забезпечити стабільну роботу системи при збільшенні навантаження та кількості користувачів. Міграція між цими СУБД спрощується завдяки стандартизованому SQL-синтаксису та наявності відповідних інструментів для переносу структури бази даних.

Поєднання таких технологій дозволяє підтримувати простоту реалізації з потенціалом на подальше масштабування [8].

Крім технічного аналізу, логічну структуру системи відображено у UML-схемі компонентів, наведеної у додатку В. Ця схема демонструє, як взаємодіють ключові модулі програми: main.py, який відповідає за запуск бота, handlers.py — обробник дій користувача, db.py — модуль бази даних, keyboards.py — генерація

кнопок, config.py — конфігураційні змінні. Взаємозв'язки між цими компонентами забезпечують узгоджену логіку обробки запитів і відповіді ботом.

Для кращого розуміння внутрішньої архітектури Telegram-бота створено UML-схему компонентів, яка відображає основні модулі та взаємозв’язки між ними.

Схема містить такі ключові компоненти:

— main.py — головний модуль, з якого починається робота бота;

— підключаються всі обробники подій;

— запускається цикл обробки повідомлень за допомогою методу updater.start_polling() (режим long polling).

handlers.py — відповідає за логіку обробки команд і дій користувача:

— команда /start, вибір категорій, показ товарів;

— взаємодія з кнопками (Inline/Reply Keyboards);

— додавання замовлень до бази.

db.py — модуль для роботи з базою даних (SQLite):

— зберігає інформацію про користувачів, товари, категорії, замовлення;

— реалізує функції для створення таблиць, додавання, пошуку та оновлення даних;

— дозволяє абстрагувати SQL-запити від основної логіки.

keyboards.py — окрема частина відповідальна за генерацію кнопок:

— меню, кнопки для вибору категорій, «Замовити» тощо;

— реалізовані через InlineKeyboardMarkup, ReplyKeyboardMarkup.

config.py — файл із конфігураційними параметрами:

— токен Telegram-бота;

— ID адміністратора;

— при потребі — інші налаштування, що легко змінюються без редагування коду.

Ця модульна структура дозволяє:

— легко масштабувати проєкт;

— спростити супровід і відлагодження коду;

— забезпечити розділення обов’язків, що відповідає принципам чистої архітектури (clean code);

— чітко організувати потоки даних і взаємодій між компонентами.

Таблиця 2.2 — Порівняльна таблиця вибраних технологій

Технологія	Характеристика	Переваги	Обгрунтування використання
Python	Мова програмування високого рівня	Простота, велика кількість бібліотек	Оптимально підходить для чат-ботів, швидка розробка
Telegram API	API для створення Чат-ботів	Пряма взаємодія з Telegram, стабільність , велика документація	Необхідна основа для реалізації логіки взаємодії в проєкті
SQLite	Вбудована файлова СУБД	Не потребує сервера, проста інтеграція, швидкість	Ідеальна для локального запуску бота
PostgreSQL	Потужна серверна СУБД із відкритим кодом	Масштабованість, транзакції, розширення	Перспективне рішення для майбутнього
Python-telegram-bot	Бібліотека Python для роботи з Telegram API	Легка реалізація, велика підтримка команд, клавіатур	Дозволяє якісно та швидко створити чат-бота

3. РОЗРОБКА ЧАТ-БОТА ДЛЯ АВТОМАТИЗАЦІЇ ЗАМОВЛЕНЬ

3.1 Архітектура та структура інформаційної системи

У сучасних умовах цифровізованої торгівлі особливу роль відіграють автоматизовані засоби обслуговування клієнтів, враховуючи чат-ботів, які забезпечують зручний доступ до інформації та функціоналу онлайн-магазинів. Проєктована інформаційна система реалізує чат-бота для автоматизації пошуку та замовлення товарів.

Структура цієї системи базується на принципах модульності, ізоляції відповідальностей, масштабованості та повторного використання коду. Такий підхід дозволяє не лише пришвидшити розробку, і спростити подальше обслуговування та розвиток функціоналу.

Telegram-бот є асинхронним програмним забезпеченням, яке взаємодіє як з користувачами, так і з Telegram-каналом, який публікує публікації товарів. Окрім інтерфейсу взаємодії, система включає локальну базу даних, яка містить структуровану інформацію про користувачів, товари, категорії та замовлення. Логіка складається з окремих модулів, які виконують конкретні завдання.

Проєкт було реалізовано у вигляді окремих python — модулів:

— main.py — є головною точкою входу в чат-бот. Він відповідає за ініціалізацію бази даних, запуск чат-бота через ApplicationBuilder та реєстрацію обробників всіх подій. В цьому модулі зосереджено мінімум логіки, що відповідає принципу розділення обов’язків.

— handlers.py — містить усю бізнес логіку системи. У цьому файлі реалізовано обробку текстових повідомлень, callback — кнопки, такі команди як /start, /orders, /categories, підтримує також логіку діалогу для оформлення замовлень. Саме в цьому модулі реалізуються основні сценарії взаємодії з ботом, пошук по категоріям, перегляд товарів та оформлення замовлень

— db.py — відповідає за роботу з базою даних, побудованою на SQLite. У цьому файлі реалізовано функції створення таблиць, додавання записів, вибірки товарів за категоріями, збереження замовлень і так далі. Дуже важливо що всі SQL запити централізовано зібрані саме в цьому модулі, які підвищують керованість даними.

— keyboards.py — утримує функції формування користувацьких клавіатур таких як reply – клавіатур, що з’являються в полі вводу повідомлень, до цього модуля також відносяться inline – кнопок, що відображаються під повідомленнями бота.

— config.py — конфігураційний файл, у якому зберігається токен чат-бота, а також назва локальної бази даних. Наявність окремого конфігураційного модуля дозволяє змінювати критичні параметри чат – бота без зміни основного кода.

Цей метод структурування коду відповідає основним принципам програмної інжинерії, зокрема принципу Separation of Concerns, який передбачає розділення логіки на модулі з чітко визначеною відповідальністю[18].

Запуск і активація Telegram-бота починаються з виконання файлу main.py, який викликає функцію init_db(), яка виконує створення структури бази даних, і збирає застосунок за допомогою конструкці ApplicationBuilder().token(BOT_TOKEN).build(). У ньому також реєструються всі обробники, які надалі керуватимуть роботою системи (рис 3.1).

```python
from telegram.ext import ApplicationBuilder
from config import BOT_TOKEN
from db import init_db
from handlers import register_handlers

def main():
    # Ініціалізація бази даних
    init_db()

    # Створення застосунку бота
    application = ApplicationBuilder().token(BOT_TOKEN).build()

    # Реєстрація обробників
    register_handlers(application)

    # Запуск бота
    print("Бот запущено...")
    application.run_polling()

if __name__ == "__main__":
    main()
```

Рисунок 3.1 — Підключення обробників та запуск бота

Таким чином, запуск системи максимально спрощено та централізовано, зо дозволяє розгортати її на будь якому комп’ютера або сервері із python

середовищем.

Після старту бота, одразу буде запропоновано головне меню із трьома основними кнопками:

— переглянути товари;

— категорії;

— про бота.

Головне меню формується за допомого функції main_menu(), показано на рисунку 3.2.

```
def main_menu():
    keyboard = [
        [KeyboardButton("🛒 Переглянути товари")],
        [KeyboardButton("📂 Категорії"), KeyboardButton("ℹ️ Про бота")]
    ]
    return ReplyKeyboardMarkup(keyboard, resize_keyboard=True)
```

Рисунок 3.2 — Код реалізації інтерфейсу навігації користувача

Дозволяє реалізувати інтуїтивно зрозумілу модель взаємодії, яка не вимагає від користувача ботом, сильних знань команд чат-бота.

Кожна подія у чат–боті викликається відповідним обробником. Наприклад, коли ви натискаєте кнопку «Категорії», викликається функція show_categories(), яка отримує список категорій з бази даних і створює inline-кнопки. У відповідь на це повідомлення користувач отримує можливість вибрати категорію.

Усі callback – запити мають певний ідентифікатор, який називається callback_data. Це дозволяє системі точно визначити, яку дію потрібно виконати. Це забезпечує кероване та гнучке реагування на чат-події.

Архітектура чат – бота побудована відповідно до принципів модульності, повторного використання да відкритості до розширення функціоналу. Кожен модуль відповідає за чітко визначену частину функціональності чат – бота, що дозволяє ефективно реалізувати нові функції без внесення глобальних змін у код. Враховуючи сучасні вимоги до взаємодії з користувачем забезпечує зручність та логічність інтерфейсу, а також сприяє широкому залучення цільової аудиторії.

3.2 Створення бази даних та функції пошуку товарів

Добре спроектована база даних є важливою частиною будь-якої інформаційної системи. У випадку розробки Telegram – бота, орієнтованого на роботу з онлайн-магазинами, база даних відповідає за зберігання, обробку та видачу даних щодо користувачів, товарів, категорій і замовлень. Це гарантує безпеку системи, швидкий доступ до інформації та можливість аналізу та подальшої масштабованості рішень.

У цьому проекті створена база даних за допомогою SQLite, системи управління базами даних, яка є найкращим варіантом для локального використання.

Легкість, автономність, відсутність необхідності розгортання серверного середовища та висока швидкість доступу до інформації є її перевагами. Це значно спрощує розгортання та налаштування Telegram – бота, особливо на етапі прототипу.

Процес ініціалізації бази даних здійснюється під час першого запуску програми. Реалізація відбувається за допомогою функції init_db(), як міститься у файлі db.py.

Саме в цій функції відбувається створення основних таблиць необхідних для роботи системи[19].

```
import sqlite3
from config import DB_NAME
from datetime import datetime
# Ініціалізація бази даних та таблиць
def init_db():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    # Категорії
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS categories (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL UNIQUE
        )
    """)
    # Продукти
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS products (
```

```
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT NOT NULL,
        description TEXT,
        price REAL NOT NULL,
        category_id INTEGER,
        image TEXT,
        FOREIGN KEY (category_id) REFERENCES categories(id)
    )
"""")
# Користувачі
cursor.execute("""
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        telegram_id TEXT NOT NULL,
        username TEXT,
        first_name TEXT,
        last_name TEXT
    )
"""")
# Замовлення (оновлено з ім’ям і телефоном)
cursor.execute("""
    CREATE TABLE IF NOT EXISTS orders (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id TEXT,
        created_at TEXT,
        status TEXT,
        customer_name TEXT,
        customer_phone TEXT
    )
"""")
# Товари у замовленні
cursor.execute("""
    CREATE TABLE IF NOT EXISTS order_items (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        order_id INTEGER,
        product_id INTEGER,
        quantity INTEGER,
        FOREIGN KEY (order_id) REFERENCES orders(id),
        FOREIGN KEY (product_id) REFERENCES products(id)
    )
"""")
```

Усього створюється п’ять таблиць, кожна з яких виконує свою функцію в системі:

— categories — зберігає список категорій товарів, які будуть відображені

користувачеві;

— products — містить інформацію про товар, що включає в себе назву, опис, ціну, категорію товару та фото;

— users — дані про користувачів Telegram, які мали взаємодію з ботом;

— orders — фіксує загальну інформацію про оформлені замовлення;

— order_items — забезпечує деталізацію товарного складу кожного замовлення.

Всі ці таблиці пов’язані між собою через зовнішні ключі, які забезпечують логічну цілісність даних.

Після створення таблиць, система викликає функцію init_categories(), яка додає у таблицю categories заздалегідь визначений список типових категорій товарів. Це дозволяє одразу після запуску мати готовий до використання каталог (рис. 3.3).

```
def init_categories():
    categories = [
        "Худі", "Шорти", "Джинси", "Куртки",
        "Взуття", "Футболки", "Спортивні костюми"
    ]
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    for name in categories:
        cursor.execute("INSERT OR IGNORE INTO categories (name) VALUES (?)", (name,))
    conn.commit()
    conn.close()
```

Рисунок 3.3 — Ініціалізація категорій у базі даних SQLite

Таким чином, без втручання адміністратора база даних вже містить базові речі для навігації, і забезпечує готовність чат – бота до використання.

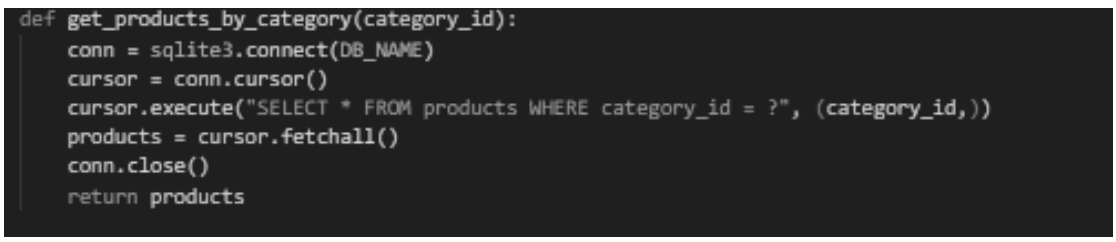
Одна з ключових функцій Telegram – бота — це можливість автоматичного додавання нових товарів на основі повідомлень з Telegram каналу. Реалізація такої функції відбувалася за допомогою add_product() (рис. 3.4).

```
def add_product(name, description, price, category_id=None, image=None):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO products (name, description, price, category_id, image)
        VALUES (?, ?, ?, ?, ?)
    """, (name, description, price, category_id, image))
    conn.commit()
    conn.close()
```

Рисунок 3.4 — Функція додавання товару до бази даних

Ця функція викликається під час обробки повідомлення з каналу, якщо структура тексту відповідає заданому шаблону (наявність назви, опис, ціна та категорія).

Щоб реалізувати зручний механізм навігації для Telegram користувача, було впроваджено функцію get_products_by_category(), яка повертає список товарів, що належать до конкретної категорії (рис. 3.5).

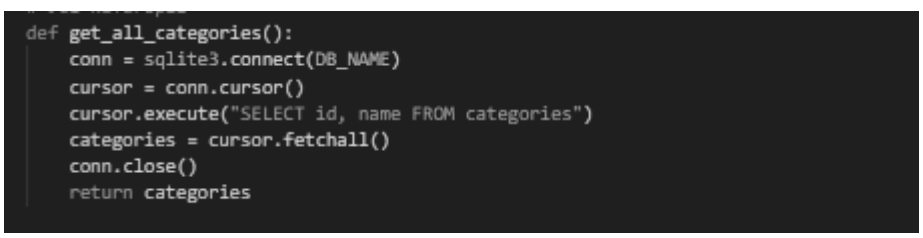


```
def get_products_by_category(category_id):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM products WHERE category_id = ?", (category_id,))
    products = cursor.fetchall()
    conn.close()
    return products
```

Рисунок 3.5 — Отримання товарів за категорією

Ця функція дозволяє боту відображати тільки ті товари, які відповідають заданим боту категоріям, тим самим підвищуючи зручність користування.

Для формування кнопок вибору категорій використовується функція get_all_categories(), яка повертає всі наявні категорії у вигляді id і назви (рис. 3.6).



```
def get_all_categories():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("SELECT id, name FROM categories")
    categories = cursor.fetchall()
    conn.close()
    return categories
```

Рисунок 3.6 — Отримання списку категорій з бази даних

У результаті завершення цього етапу було розроблено та успішно впроваджено базу даних. Ця база даних має вирішальне значення для зберігання, організації та обробки інформації. Це необхідно для того, щоб Telegram-бот працював як цифровий інструмент. Структура, яка була створена, відзначається тим, що вона є логічно цілісною, її можна реалізувати без великих затрат та вона відповідає практичним вимогам систем, які працюють у режимі реального часу.

Застосування SQLite як локальної системи управління базами даних є технічно можливим для завдань даного рівня. Крім того, вона має багато переваг,

таких як автономність, відсутність потреби в додатковому серверному програмному забезпеченні, низькі витрати ресурсів і висока швидкість обробки запитів. Це дозволило створити мобільний, швидкий і надійний Telegram-бот без додаткових витрат на інфраструктуру.

Особлива увага була приділена гнучкості реалізації: база даних підтримує як автоматизоване додавання даних із Telegram-каналу. Такий метод поєднує в собі переваги централізованого управління з можливістю адаптації та масштабування відповідно до вимог конкретного онлайн-магазину.

Розширення можливостей бота легко досягається за допомогою запропонованої структури зберігання даних. Підтримка складів, управління залишками, ведення історії оплат і статистична аналітика поведінки користувачів можуть бути додані в майбутньому. Масштабування не вимагатиме суттєвої зміни архітектури системи, оскільки структура базується на нормалізованій моделі з чіткими зв’язками між сутностями.

Таким чином, проєктована база даних є не лише технічною основою системи, а й стратегічним елементом, який гарантує, що система працює добре, також є ефективною та може адаптуватися до можливостей у майбутньому [9].

3.3 Розробка функціоналу отримання даних із Telegram-каналу

Одним з самих ефективних напрямків розвитку автоматизованих сервісів є впровадження інформаційної системи разом із каналами зовнішнього введення даних. Розробка Telegram-бота для онлайн-магазину особливо важлива через можливість автоматичного імпорту нових товарів із Telegram-каналу адміністратора або менеджера каналу, на якому вони публікують товари у вигляді звичайних постів. Це не тільки покращує ефективність бізнес–процесів, але й усуває залежність від введення даних у систему ручним способом.

Бот Telegram автоматично зчитує повідомлення з публічних каналів Telegram, фільтрує їх, ідентифікує важливу інформацію та додає нові продукти до локальної бази даних. Цей аспект є основою для повноцінної системи синхронізації даних між інтерфейсом Telegram-бота та контентом Telegram-каналу.

Головною метою розробленого функціоналу є створення максимально зручного процесу додавання товарів до системи, який не потребує складного втручання або технічних знань з боку адміністратора. У рамках проєкту реалізовано підхід, за якого адміністратор просто публікує товари у Telegram-каналі у звичному форматі постів, а бот автоматично зчитує ці дані, обробляє їх і додає до внутрішньої бази. Після цього товари миттєво стають доступними для користувачів бота, що дозволяє значно скоротити час між публікацією і моментом, коли товар можна замовити.

Такий підхід забезпечує низький поріг входу в систему для менеджерів та власників бізнесу: їм не потрібно працювати з формами, панелями адміністрування чи спеціалізованим ПЗ — весь процес зводиться до простої дії в Telegram. Це підвищує швидкість оновлення асортименту, зменшує кількість помилок, пов’язаних із людським фактором, та усуває потребу в дублюванні інформації в різних середовищах.

Більше того, реалізація автоматичного імпорту з каналу дозволяє розглядати Telegram не лише як платформу для взаємодії з клієнтом, а й як повноцінний інтерфейс керування товарним наповненням, що відкриває нові можливості для розвитку подібних систем у сфері мікробізнесу та електронної комерції.

З технічної точки зору API Telegram Bot надає розробнику доступ до події channel_post, яка викликається кожного разу, коли з’являється новий допис у каналі, де присутній бот. Цей механізм реалізовано шляхом додавання обробника до файлу handlers.py (рис 3.7).

```
application.add_handler(MessageHandler(filters.UpdateType.CHANNEL_POST, handle_channel_post))
```

Рисунок 3.7 — Реєстрація обробника публікацій з Telegram-каналу

Це дозволяє чат-боту реагувати а нові пости каналу в реальному часі.

Повідомлення, що надходить із Telegram – каналу, повинне відповідати заздалегідь визначеному щаблону. У межах нашого проєкту було запропоновано наступну структуру повідомлення:

— худі Staff khaki logo;

— опис — Стильне худі для щоденного носіння;

— ціна — 999 грн;

— категорія — Худі.

Кожен такий пост містить чітко визначені маркери (Назва, Опис, Ціна, Категорія), що дозволяє точно розпізнати необхідну інформацію для внесення до бази.

Обробку повідомлень реалізовано у функції handle_channel_post(), яка викликається автоматично при надходженні публікації каналу (рис.3.8).

Ця частина коду здійснює:

— розбиття підпису публікації на рядки;

— витяг основної інформації з відповідних рядків;

— перетворення ціни у числовий формат;

— перевірка коректності даних.

```python
async def handle_channel_post(update: Update, context: ContextTypes.DEFAULT_TYPE):
    message = update.channel_post
    if not message.photo or not message.caption:
        return

    caption = message.caption
    photo = message.photo[-1].file_id

    try:
        lines = caption.strip().split("\n")
        name = lines[0].strip()
        desc = next((l.replace("Опис:", "").strip() for l in lines if l.startswith("Опис:")), "")
        price_line = next((l for l in lines if "Ціна:" in l), None)
        price = float(price_line.replace("Ціна:", "").replace("грн", "").strip())
        category_line = next((l for l in lines if l.startswith("Категорія:")), None)
        category = category_line.replace("Категорія:", "").strip()
    except Exception as e:
        print(f"❌ Помилка при обробці поста: {e}")
        return

    categories = get_all_categories()
    cat_id = next((cid for cid, cname in categories if cname.lower() == category.lower()), None)
    if cat_id is None:
        print(f"⚠️ Категорія не знайдена: {category}")
        return

    add_product(name, desc, price, cat_id, photo)
    print(f"✅ Товар додано: {name} | {price} грн")
```

Рисунок 3.8 — Логіка парсингу повідомлення з Telegram – каналу

Після того, як дані з публікації розібрано, чат – бот перевіряє наявність відповідної категорії в базі даних (рис. 3.9).

Якщо категорія була в базі, то викликається функція додавання товару. Це дозволяє уникнути некоректних ситуацій.

```
categories = get_all_categories()
cat_id = next((cid for cid, cname in categories if cname.lower() == category.lower()), None)
if cat_id is None:
    print(f"⚠ Категорія не знайдена: {category}")
    return
```

Рисунок 3.9 — Перевірка існування категорії в базі

Функція add_product() розташована у модулі db.py. Вона відповідає за запис нового товару у таблицю products бази даних (рис 3.10).

Наявна функція є функціональною і універсальною — її також можна викликати вручну, якщо буде реалізовано адмінастративну панель.

У разі якщо адмін Telegram – каналу випадково опублікує повідомлення з некоректним форматом або пропустить одне з ключових полів, код у try-except блоку гарантує, що програма не припинить виконання, а лише виведе діагностичне повідомлення в консоль розробника[20].

```
def add_product(name, description, price, category_id=None, image=None):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO products (name, description, price, category_id, image)
        VALUES (?, ?, ?, ?, ?)
    """, (name, description, price, category_id, image))
    conn.commit()
    conn.close()
```

Рисунок 3.10 — SQL запит для збереження товару до таблиці products

Це підвищує стійкість і надійність системи, знижує ймовірність «падіння» програми та дозволяє безпечно вводити нові дані навіть під час ручного введення даних. Такий підхід забезпечує гнучкість в управлінні контентом і дозволяє підтримувати актуальність каталогу товарів без залучення технічного персоналу.

Реалізація такого функціоналу є критично важливою для загальної ефективності. Це дозволяє компаніям швидко змінювати свій асортимент, не витрачаючи часу на повторне введення. Наприклад, оператор може створити від двадцяти до тридцяти постів із товарами за десять хвилин, і всі ці пости автоматично зберігаються в базі даних бота.

Автоматизоване зчитування інформації з публічного Telegram-каналу

значно знижує вплив людського фактору на точність даних. Відсутність потреби у дублюванні інформації в різних системах веде до зменшення ймовірності помилок, таких як неправильна ціна, некоректне фото або відсутність опису.

У результаті така система підтримує не лише швидкість і простоту оновлення товарного наповнення, але й зберігає цілісність і відповідність даних між Telegram-каналом і базою, що критично важливо в умовах динамічного електронного ринку.

У перспективі, така структура дозволяє реалізувати інтелектуальні механізми фільтрації контенту, наприклад автоматичне визначення категорій товарів за ключовими словами або розмітками в описі. Це відкриває шлях до впровадження елементів машинного навчання, що можуть аналізувати історію замовлень та формувати персоналізовані рекомендації для користувача вже на етапі перегляду товарів у боті.

Також, автоматизація через Telegram-бота може стати основою для побудови повноцінної мікросервісної інфраструктури, де кожен компонент (бот, база даних, API товарів, логістика) функціонуватиме окремо, забезпечуючи масштабованість і відмовостійкість. Подібний підхід дозволить легко інтегрувати нові функції, як от чат із оператором, онлайн-оплата, відстеження статусу замовлень або навіть підключення зовнішніх аналітичних модулів для вивчення поведінки користувачів.

Навіть на локальному рівні без використання серверної інфраструктури, система демонструє високий рівень адаптивності та є готовою до розвитку у більш комплексне рішення, орієнтоване на потреби малого та середнього бізнесу в умовах цифрової трансформації.

Крім того, функціональність можна розширити. У майбутньому можна включити такі можливості, як:

— автоматичне сортування за тегами;
— прив’язка до складів або залишків;
— фотоаналітика та пошук по зображенню;
— автоматичне підтвердження через панель адміністратора.

Таким чином, реалізована інтеграція чат-бота з Telegram – каналом є не

лише додатковим сервісом, і чудовим рішенням, яке визначає гнучкість і адаптивність усієї системи до реальних умов цифрового ринку.

3.4 Програмна реалізація бота (замовлення, команди, обробка запитів)

Чат – бот, створений у межах даної інформаційної системи реалізує повний цикл взаємодії користувача з онлайн – магазином від перегляду товарів до оформлення замовлення.

Основні компоненти логіки бота реалізовано у файлі handlers.py, він містить функції обробки команд, повідомлень, кнопок і так далі.

Для локального зберігання інформації про товари, замовлення та користувачів використовується вбудована СУБД SQLite, яка дозволяє ефективно виконувати запити без додаткового програмного забезпечення. Дані зберігаються у вигляді окремого .db-файлу, що підвищує автономність системи та полегшує перенесення або резервне копіювання.

Обробка команди /start (рис.3.11).

```
    await update.message.reply_text(
        "Привіт! Вітаємо в Telegram-магазині \U0001F448\nОберіть дію нижче:",
        reply_markup=main_menu()
    )
```

Рисунок 3.11 — Обробка команди /start

Якщо користувач натискає кнопку «Переглянути товари», викликається функція show_products(), яка отримує список усіх товарів з бази (рис. 3.12).

```
async def show_products(update: Update, context: ContextTypes.DEFAULT_TYPE):
    products = get_all_products()
    if not products:
        await update.message.reply_text("Список товарів наразі порожній.")
        return
    for prod_id, name, desc, price, _, image in products:
        text = f"\U0001F4E6 *{name}*\n\U0001F4B8 Ціна: {price} грн\n\U0001F4DD {desc or 'Без опису'}"
        if image:
            await update.message.reply_photo(
                photo=image,
                caption=text,
                parse_mode=ParseMode.MARKDOWN,
                reply_markup=product_actions(prod_id)
            )
        else:
            await update.message.reply_text(
                text,
                parse_mode=ParseMode.MARKDOWN,
                reply_markup=product_actions(prod_id)
            )
```

Рисунок 3.12 — Функція відображення всі товарів у чат – боті

Товари завантажуються через функцію get_all_products() (рис. 3.13).

```
def get_all_products():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM products")
    products = cursor.fetchall()
    conn.close()
    return products
```

Рисунок 3.13 — Отримання усіх товарів із бази даних

Якщо користувач обирає кнопку «Категорії», викликається функція show_categories() (рис. 3.14).

```
async def show_categories(update: Update, context: ContextTypes.DEFAULT_TYPE):
    categories = get_all_categories()
    if not categories:
        await update.message.reply_text("Категорій поки немає.")
        return
    await update.message.reply_text("Оберіть категорію:", reply_markup=category_menu(categories))
```

Рисунок 3.14 — Вивід категорії товарі

Формування inline – кнопок відбуваються таким чином (рис. 3.15).

```
def category_menu(categories):
    buttons = []
    for cat_id, name in categories:
        buttons.append([InlineKeyboardButton(name, callback_data=f"cat_{cat_id}")])
    return InlineKeyboardMarkup(buttons)
```

Рисунок 3.15 — Створення кнопок для вибору категорій

При виборі певної категорії спрацьовує callback – запит, що обробляється через category_selected() (рис. 3.16).

```
async def category_selected(update: Update, context: ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()
    cat_id = int(query.data.replace("cat_", ""))
    products = get_products_by_category(cat_id)
    if not products:
        await query.edit_message_text("У цій категорії поки немає товарів.")
        return
    for prod_id, name, desc, price, _, image in products:
        text = f"\U0001F4E6 *{name}*\n\U0001F4B8 Ціна: {price} грн\n\U0001F4DD {desc or 'Без опису'}"
        if image:
            await query.message.reply_photo(
                photo=image,
                caption=text,
                parse_mode=ParseMode.MARKDOWN,
                reply_markup=product_actions(prod_id)
            )
        else:
            await query.message.reply_text(
                text,
                parse_mode=ParseMode.MARKDOWN,
                reply_markup=product_actions(prod_id)
            )
```

Рисунок 3.16 — Обробка вибору категорії

Кнопка «Замовити» під’днана для кожного товару (рис. 3.17).

```
def product_actions(product_id):
    buttons = [
        [InlineKeyboardButton("🛒 Замовити", callback_data=f"order_{product_id}")]
    ]
    return InlineKeyboardMarkup(buttons)
```

Рисунок 3.17 — Створення кнопки «Замовити»

Після натискання кнопки «Замовити» запускається діалог, у якому бот послідовно запитує ім’я користувача та його номер телефону. Реалізація через ConversationHandler, який дозволяє керувати логікою крок за кроком (рис. 3.18).

Перший крок діалого — запит імені, викликається order_product() (рис. 3.19).

Другий крок — запит телефону, після введення імені викликається функція receive_name() (рис. 3.20).

```
conv_order = ConversationHandler(
    entry_points=[CallbackQueryHandler(order_product, pattern="^order_")],
    states={
        WAITING_NAME: [MessageHandler(filters.TEXT & ~filters.COMMAND, receive_name)],
        WAITING_PHONE: [MessageHandler(filters.TEXT & ~filters.COMMAND, receive_phone)],
    },
    fallbacks=[CommandHandler("cancel", cancel)],
)
application.add_handler(conv_order)
```

Рисунок 3.18 — Реєстрація діалогу для оформлення замовлення

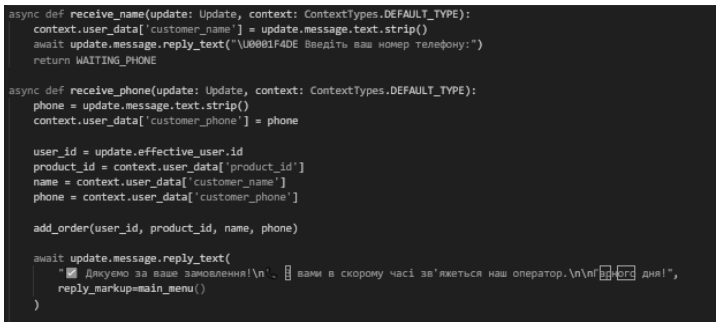
```
async def order_product(update: Update, context: ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()
    context.user_data['product_id'] = int(query.data.replace("order_", ""))
    await query.message.reply_text("\U0001F9FE Введіть ваше ім'я:")
    return WAITING_NAME
```

Рисунок 3.19 — Запит імені користувача

```
async def receive_name(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['customer_name'] = update.message.text.strip()
    await update.message.reply_text("\U0001F4DE Введіть ваш номер телефону:")
    return WAITING_PHONE
```

Рисунок 3.20 — Перехід до запиту телефону

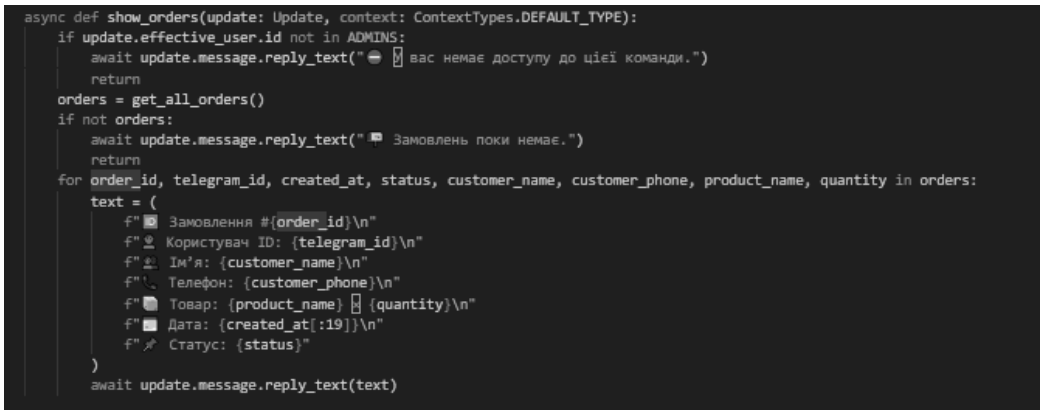
Завершення замовлення — збереження в базу даних (рис. 3.21).



```
async def receive_name(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['customer_name'] = update.message.text.strip()
    await update.message.reply_text("\U0001F4DE Введіть ваш номер телефону:")
    return WAITING_PHONE

async def receive_phone(update: Update, context: ContextTypes.DEFAULT_TYPE):
    phone = update.message.text.strip()
    context.user_data['customer_phone'] = phone

    user_id = update.effective_user.id
    product_id = context.user_data['product_id']
    name = context.user_data['customer_name']
    phone = context.user_data['customer_phone']

    add_order(user_id, product_id, name, phone)

    await update.message.reply_text(
        "✅ Дякуємо за ваше замовлення!\n З вами в скорому часі зв'яжеться наш оператор.\n\nГарного дня!",
        reply_markup=main_menu()
    )
```

Рисунок 3.21 — Збереження замовлення після введення телефону

Щоб адміністратор міг переглядати заявки, реалізовано функцію /orders, яка доступна лише для певних ID (рис. 3.22).



```
async def show_orders(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if update.effective_user.id not in ADMINS:
        await update.message.reply_text("⛔ У вас немає доступу до цієї команди.")
        return
    orders = get_all_orders()
    if not orders:
        await update.message.reply_text("📭 Замовлень поки немає.")
        return
    for order_id, telegram_id, created_at, status, customer_name, customer_phone, product_name, quantity in orders:
        text = (
            f"🆔 Замовлення #{order_id}\n"
            f"👤 Користувач ID: {telegram_id}\n"
            f"👥 Ім'я: {customer_name}\n"
            f"📞 Телефон: {customer_phone}\n"
            f"📦 Товар: {product_name} × {quantity}\n"
            f"📅 Дата: {created_at[:19]}\n"
            f"📌 Статус: {status}"
        )
        await update.message.reply_text(text)
```

Рисунок 3.22 — Команда /orders: вивід усіх заявок для адміністратора

3.5 Тестування роботи бота

У межах тестування буде перевірена повна функціональна взаємодія між Telegram – каналом, базою даних і Telegram – ботом. Нижче буде представлено покрокову ілюстрацію механізму роботи бота.

Для тестування також було створено Telegram – канал, у який адміністратор буде розміщувати товари, а чат – бот буде автоматично обробляти повідомлення, додавши їх до бази даних та показувати користувачам у категоріях.

Тестування включало декілька ключових етапів:

— інтеграційне тестування взаємодії між ботом і Telegram-каналом (обробка нових повідомлень, імпорт товарів);

— функціональне тестування основних команд /start, вибір категорії,

перегляд товару, оформлення замовлення, команда /orders для адміністратора;

— тестування збереження даних у базі SQLite — додавання користувача, замовлення, зв’язок з товарами;

— перевірка стійкості при багаторазовому взаємодіянні: повторні замовлення, швидкі переходи між категоріями;

— UI/UX тестування у середовищі Telegram для мобільних пристроїв (Android, iOS) для оцінки зручності взаємодії.

Для проведення інтеграційного тесту було створено Telegram-канал, до якого додавався бот. Адміністратор публікував у канал пости у відповідному форматі, після чого бот автоматично зчитував інформацію, зберігав її у базу даних і формував відповідні категорії товарів. Такий підхід суттєво спрощує оновлення товарного каталогу та мінімізує потребу у технічній підготовці персоналу магазину.

Було перевірено також сценарій оформлення замовлення. Користувач обирав товар, натискав кнопку «Замовити», після чого бот запускав діалог з уточненням імені та номера телефону. Ці дані зберігалися у відповідних таблицях бази, а адміністратор міг переглядати заявки через спеціальну команду /orders.

Під час тестування особлива увага приділялась:

— відповідності обробки запитів до логіки, закладеної у програмному коді;

— швидкості реакції бота на дії користувача (відправка відповіді до 1 секунди);

— коректності збереження даних у таблицях users, orders, order_items;

— відсутності збоїв або помилок при обробці неочікуваних дій користувача (наприклад, введення тексту замість номера телефону).

Виявлено, що система демонструє стабільну роботу:

— повна автономність без використання сторонніх хостингів чи серверів;

— робота з мінімальними ресурсами комп’ютера розробника;

— збереження всіх замовлень навіть при випадковому завершенні

роботи бота (завдяки локальній базі).

Така архітектура створює зручний робочий процес для управління товарами: адміністратор просто публікує пост у каналі з дотриманням встановленого формату, а система самостійно забезпечує актуалізацію каталогу. Це рішення також дозволяє легко масштабувати процес додавання товарів та забезпечує резервування даних через збереження оригінальних повідомлень у месенджері.

Така реалізація дозволяє уникнути потреби у складних веб-інтерфейсах для менеджменту товарів, що є суттєвою перевагою для невеликих бізнесів або індивідуальних підприємців. Telegram-канал виступає у ролі єдиного джерела інформації про товари, а бот автоматично синхронізує ці дані із внутрішньою базою. Це забезпечує відповідність вмісту каналу і фактичного товарного каталогу, мінімізуючи ризик неузгодженості інформації.

Додатковою перевагою такого підходу є те, що адміністратор зберігає повний контроль над вмістом товарів без необхідності взаємодії з окремими службами чи сервісами. Оскільки Telegram вже має вбудовану підтримку форматування тексту, медіа, тегів та кнопок, він фактично виконує роль універсального редактора товарного контенту. Це дозволяє адаптувати опис, змінювати ціну або додавати фото в режимі реального часу — без залучення сторонніх систем або спеціалістів.

Крім цього, Telegram забезпечує збереження історії змін у постах, що робить можливим відновлення попередніх версій товарів або відстеження помилок. У поєднанні з локальним збереженням копій у базі даних, така система забезпечує двофакторне резервування даних, що критично важливо у разі технічних збоїв чи втрати з’єднання.

Завдяки використанню такого підходу, система стає стійкою до помилок, спричинених людським фактором, та забезпечує історичну трасованість усіх змін, оскільки Telegram дозволяє переглядати попередні версії повідомлень. У поєднанні з локальним збереженням копій записів у базі даних, це створює надійний фундамент для підтримки актуальності та цілісності інформації. (рис. 3.23).

Telegram – бот, підключений до каналу, автоматично фіксує новий пост, зчитує всі потрібні поля та додає товар у базу (рис. 3.24).

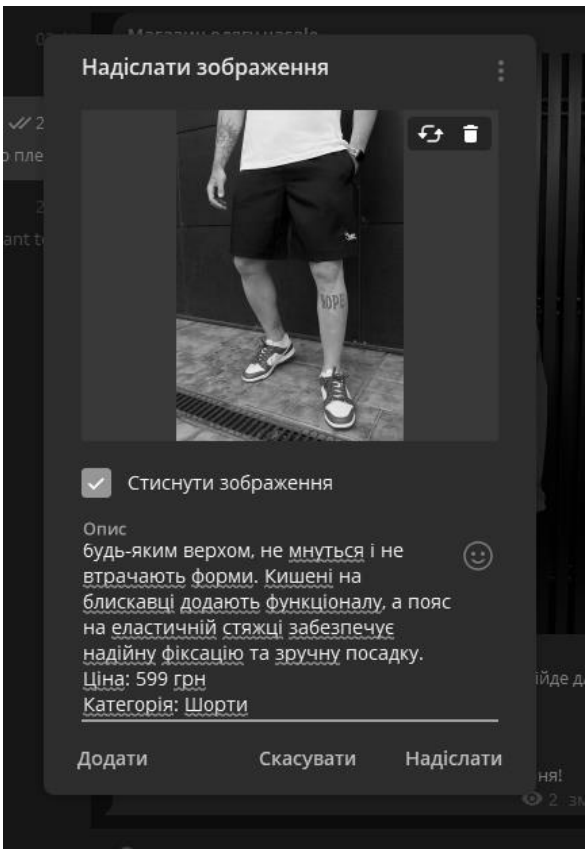


Рисунок 3.23 — Публікація нового товару

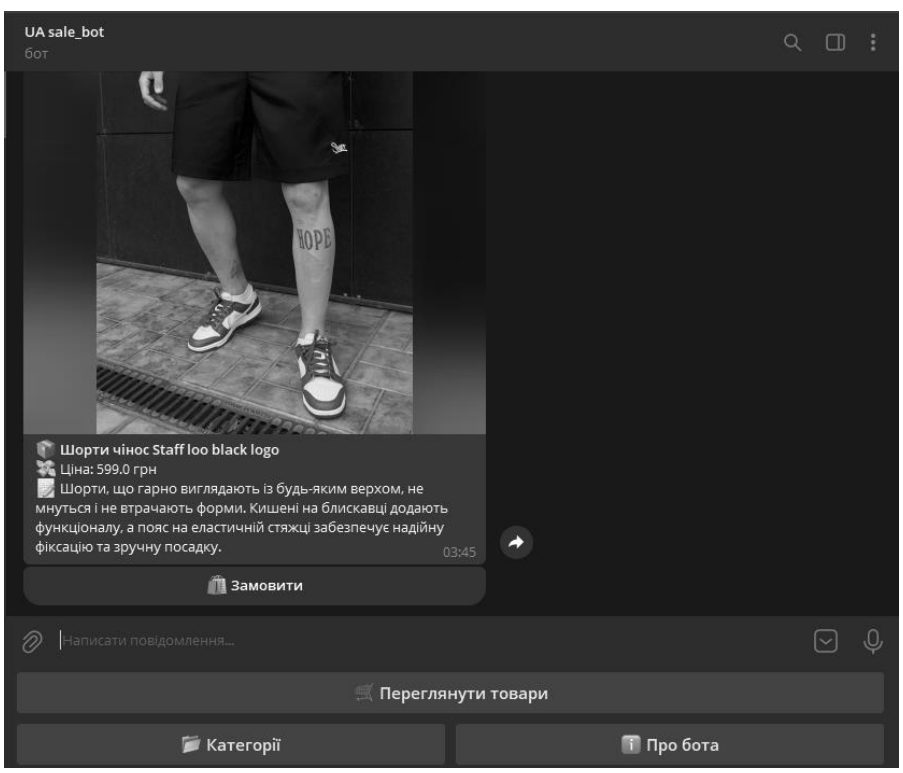


Рисунок 3.24 — Підтвердження додавання товару до бази

Запуск бота та відкриття головного меню.

Користувач запускає чат – бота, перед ним появляється меню, в якому можна обрати перегляд товарів або категорій (рис. 3.25).

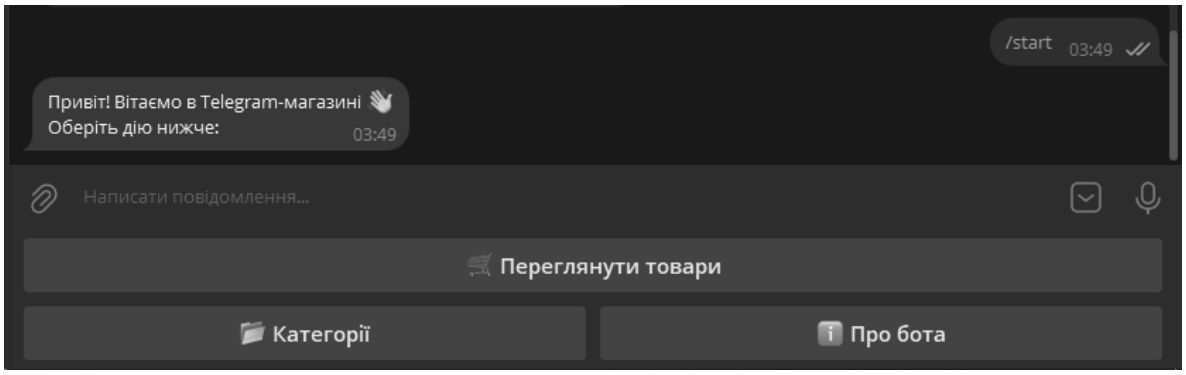


Рисунок 3.25 — Інтерфейс чат – бота: головне меню після /start

Перегляд списку категорій після вибору «Категорії» бот показує всі доступні категорії товарів. Товар який добавили в Telegram – каналі уже доступний у відповідній категорії (рис. 3.26).

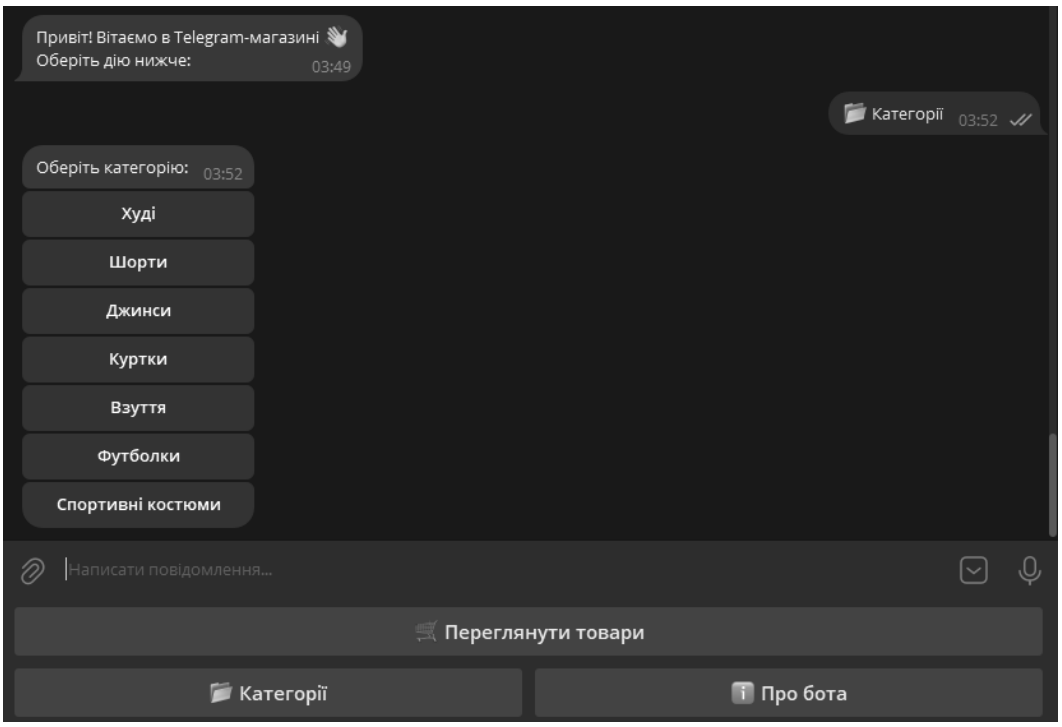


Рисунок 3.26 — Відображення категорій чат – бота

Перегляд товарів у вибраній категорії.

Користувач обирає потрібну йому категорію. Бот автоматично показує всі товари які належать цій категорії (рис. 3.27).

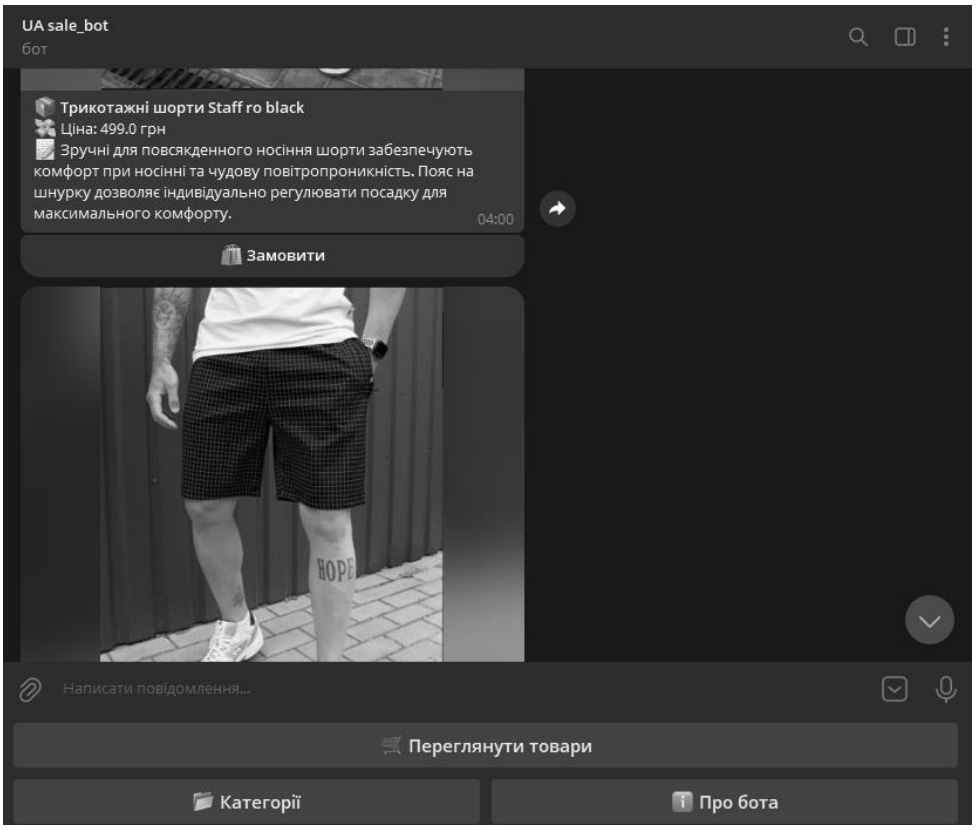


Рисунок 3.27 — Перелік товарів у категорій «Шорти» з кнопкою «Замовити»

Процес оформлення замовлення.

Користувач натискає кнопку «Замовити». Бот запускає діалог, у якому запитує ім’я і номер телефону (рис. 3.28, 3.29).

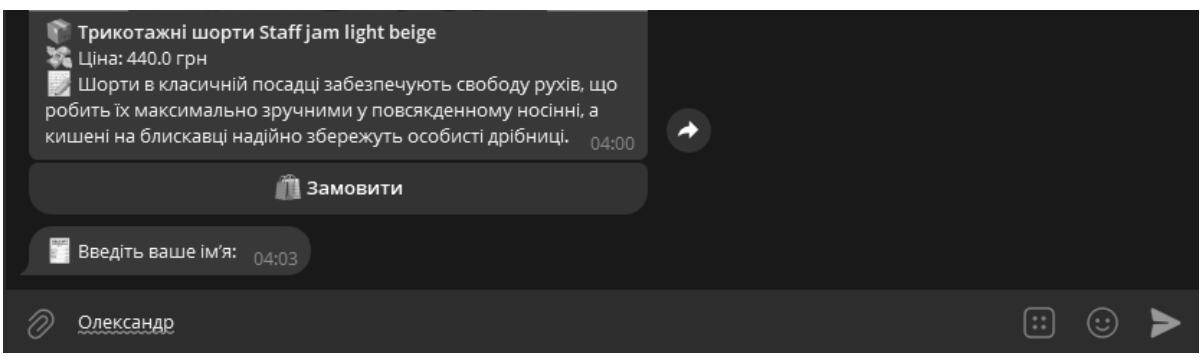


Рисунок 3.28 — Запит імені під час оформлення замовлення

Рисунок 3.29 — Запит номера телефону під час оформлення замовлення

Підтвердження замовлення відбувається після введення телефону, бот підтверджує прийняття заявки. Адміністратор може переглянути всі замовлення через команду /orders (рис. 3.30).

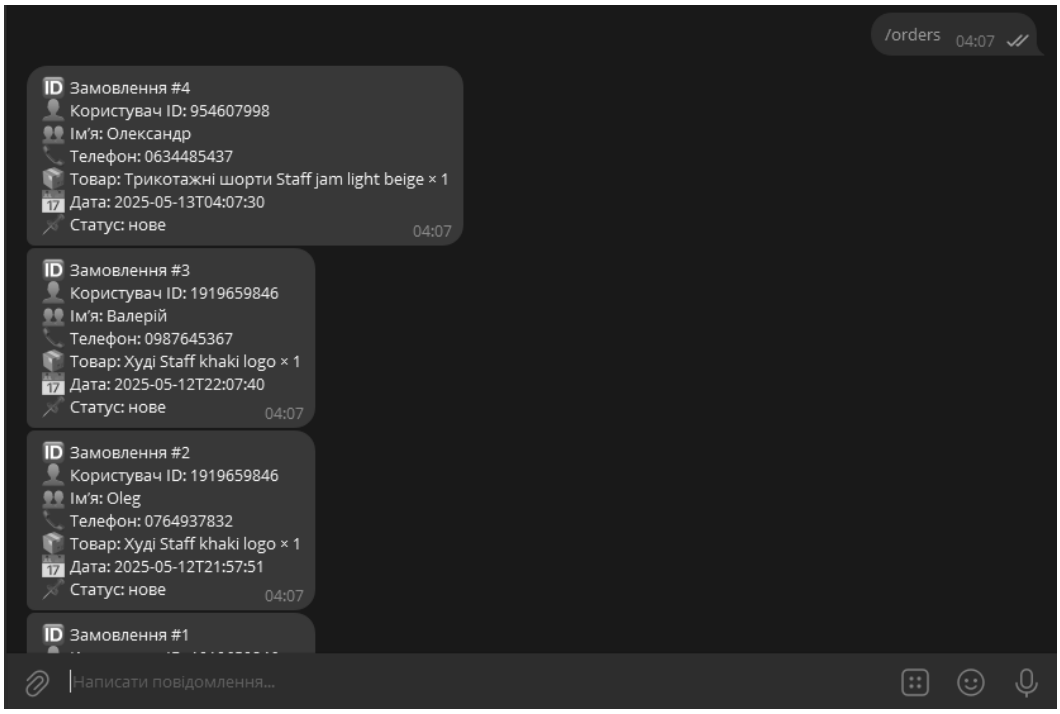


Рисунок 3.30 — Повідомлення про успішне замовлення

За результатами тестування встановлено, що чат – бот працює стабільно, коректно реагує на публікації з каналу, додає товари до бази, підтримує фільтрацію за категоріями та дозволяє оформляти замовлення. Автоматизація додавання товарів значно спростила процес ведення магазину через месенджер Telegram [21].

ВИСНОВКИ

У ході виконання дипломної роботи було повністю реалізовано Telegram – бота, призначеного для автоматизації процесу пошуку та замовлення товарів з онлайн – магазину.

Основною метою проєкту було спрощення взаємодії між покупцем і продавцем, усунення ручного обліку товарів і замовлень, також підвищення зручності користування сучасними цифровими платформами.

У першому розділі було проведено аналіз існючих рішень у сфері торгових чат – ботів, вивчено принципи роботи чат – ботів, і також розглянуто приклади популярних систем, що забезпечують аналогічний функціонал. Самец це дозволило сформулювати вимоги до власного продукту та визначити ключові напрями розробки.

Другий розділ присвячено обгрунтуванню вибору технологій, включаючи Telegram Bot API, мову програмування Python та СУБД SQLite. Було обгрунтовано вибір клієн – серверної архутектури, визначено структуру бази даних та розроблено її схему. Вибрана реалізація передбачає автономну роботу без підключення зовнішніх серверів, що спрощує обслуговування бота.

У третьому розділі розглянуто програмну реалізацію системи. Створено головне меню, інтерфейс перегляду товарів та категорій, механізм оформлення замовлень, обробку callback – запитів, інтеграцію з Telegram – каналом для автоматичного додавання товарів до бази.

Також реалізовано адміністративний інтерфейс у вигляді команди /orders, яка дозволяє переглядати вся заявки через месенджер Telegram.

Основні переваги реалізованої системи:

— простота використання як для користувача, так і для адміністратора;
— автоматичне оновлення товарів;
— зберігання історії замовлень у базі з прив’язкою до користувача;
— обробка помилок без порушення роботи бота;
— модульна архутектура, що дозволяє легко масштабувати систему.

Таким чином, розроблений чат – бот повністю відповідає поставленим

вимогам та може бути ефективно використаний у малих і середніх бізнес – процесах, пов’язаних з продажем товарів через онлайн – магазини.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Костюк О. Д. Кадук О. В. Чат-бот пошуку та замовлень товарів з онлайн-магазину // Матеріали всеукраїнської науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» (15 листопада 2024 р. – 14 червня 2025 р., Вінниця) URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25436

2. Платформи для продажу товарів через інтернет. weblium. [Електронний ресурс]. Доступ за адресою: https://ua.weblium.com/blog/platformi-dlya-prodazhu-tovariv.

3. Рейтинг популярних месенджерів в Україні та світі у 2024 році, який обрати [Електронний ресурс]. Доступ за адресою: https://sitecat.net/review/top-10-popular-messengers/.

4. Найкращі IDE для Python [Електронний ресурс]. Доступ за адресою: https://dan-it.com.ua/uk/blog/luchshie-ide-dlja-python/.

5. База даних Python: огляд сумісних БД та як їх підключити [Електронний ресурс]. Доступ за адресою: https://foxminded.ua/baza-danykh-python/.

6. Як створити телеграм бота через BotFather? - Ukr-Bot [Електронний ресурс]. Доступ за адресою: https://ukr-bot.com/yak-stvoryty-telehram-bota-cherez-botfather/.

7. BotFather. Можливості, команди та функціонал [Електронний ресурс]. Доступ за адресою: https://gerabot.com/article/botfather_mozhlivosti_ta_funkcional.

8. The Python Tutorial — Python 3.13.3 documentation [Електронний ресурс]. Доступ за адресою: https://docs.python.org/3/tutorial/index.html.

9. Яковенко А. В. Основи програмування. Python. Частина 1 [Електронний ресурс]. Київ, 2018. 150 с. Доступ за адресою: https://ela.kpi.ua/server/api/core/bitstreams/dbbe8ff5-11d7-4a92-918c-d1445c3d20a7/content.

10. sqlite3 DB-API 2.0 interface for SQLite databases [Електронний ресурс]. Доступ за адресою: https://docs.python.org/3/library/sqlite3.html.

11. Як працює інтернет-магазин: основні процеси. URL: https://wezom.com.ua/ua/blog/biznes-processy-internet-magazina-i-ih-optimizaciya.

12. Python Software Foundation. Python 3.12.0 final now available. URL: https://web.archive.org/web/20231010192149/https://blog.python.org/2023/10/python-3120-final-now-available.html.

13. Python Documentation. What's New In Python 3.0. URL: https://web.archive.org/web/20081004052720/http://docs.python.org/dev/3.0/whatsnew/3.0.html.

14. PHP [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/PHP.

15. Все про чат-боти: типи і приклади, якому бізнесу підійде, список конструкторів для створення. Webpromo. [Електронний ресурс]. Доступ за адресою: https://web-promo.ua/ua/blog/vse-o-chat-botah-tipy-i-primery-kakomu-biznesupodojdet-spisok-konstruktorov-dlya-sozdaniya/#types.

16. Розробка та створення чат бота telegram, viber, fb, instagram – компанія Gerabot [Електронний ресурс]. Доступ за адресою: https://gerabot.com/.

17. SQLite home page. SQLite Home Page. [Електронний ресурс]. Доступ за адресою: https://www.sqlite.org.

18. Python Documentation [Електронний ресурс]. Доступ за адресою: https://docs.python.org/3/.

19. Telegram APIs [Електронний ресурс]. Доступ за адресою: https://core.telegram.org/.

20. Створюємо Telegram бота на Python. Частина 1 [Електронний ресурс]. Доступ за адресою: https://devzone.org.ua/post/stvoriuyemo-telegram-bota-na-python-chastyna-1.

21. Півень, А. В. "Автоматизована система обліку клієнтів та замовлень онлайн магазину на базі telegram бота." (2022). Доступ за адресою: https://essuir.sumdu.edu.ua/handle/123456789/89704

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н.. Азаров О.Д.
«___» 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи
«Чат-бот пошуку та замовлень товарів з онлайн-магазину»
08-54.БКР.011.00.000 ТЗ

Науковий керівник: к.т.н., доц. каф. ОТ
Кадук О. В. _____
Студент групи 1КІ-21б
Костюк О. Д._____

ВНТУ 2025

1 Підстави для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 Актуальність дослідження обумовлена в умовах стрімкого розвитку електронної комерції. Особливої популярності набувають платформи та месенджери як засоби взаємодії між продавцем і покупцем. Чат – бот надає широкі можливості, та автоматизує процеси пошуку та замовлення товарів. Що сутєво спрощує роботу онлайн – магазинів, зменшуючи навантаження на персонал.

1.2 Тема бакалаврської кваліфікаційної роботи затверджена наказом ректора університету №123 від 01.09.2023 року.

2 Мета БКР і призначення розробки

2.1 Метою проєкту є розробка Telegram – бота, який забезпечує автоматизований пошук, перегляд та оформлення замовлень товарів на основі публікації у Telegram – каналі. Система має забезпечити зручний інтерфейс для користувача, та спростити процес управління онлайн – продажами.

2.2 Призначення розробки полягає у створенні програмного інструменту, який дозволяє автоматично імпортувати товари з Telegram – каналу, відображати їх користувачеві в зручному меню, фіксувати заявки на замовлення в локальній базі даних для подальшої обробки.

3 Вихідні дані для виконання БКР

3.1 Дослідження популярних Telegram – ботів для продажу товарів, а також вивчення переваг і недоліків вже існуючих автоматизованих чат – ботів у месенджерах.

3.2 Розробка структурної та функціональної схеми, створення логічної структури Telegram – бота, визначення основних модулів, побудова діаграм зв’язків між об’єктами системи.

3.3 Написання коду бота на мову програмування Python з використанням Telegram Bot API та бібліотеки python-telegram-bot, підключення бази данних SQLite, модульне тестування ключових компонентів.

3.4 Економічна доцільність використання визначається як безкоштовна для мікробізнесів, автономної, зручної та гнучкої платформи без потреб у дорогому хостингу чи VPS.

3.5 Відповідність стандартам, забезпечення відповідності вимогам до захисту персональних даних, дотримання загальних стандартів розробки вебсервісів та принципів безпечного зберігання інформації.

4 Вимоги до виконання БКР

Головна вимога – реалізація Telegram – бота, здатного виконувати всі поставлені йому задачі в цьому проєкті.

5 Етапи БКР та очікувані результати

5.1 Підготовчий етап, збір та аналіз літератури, технічної документації та прикладів Telegram – ботів для електронної комерції.

5.2 Етап проєктування, побудова архітектури бота, створення структури бази даних, опис інтерфейсів взаємодії, створення свого Telegram – каналу як джерело товарів.

5.3 Етап розробки, реалізація коду бота, налаштування обробки команд, кнопок, повідомлень, імпорту товарів з каналу, збереження замовлень. Результат — робочий Telegram – бот.

5.4 Етап тестування, моделювання поведінки користувача, тестування обробки замовлень, коректності додавання товарів з каналу, перевірка бази. Стабільна робота всіх функцій.

5.5 Етап впровадження, розгортання бота в реальному середовищі, підключення до публічного Telegram – каналу, отримання перших замовлень.

6 Матеріали, що подаються до захисту — пояснювальна записка БКР, графічні та ілюстративні матеріали, протокол попереднього захисту на кафедрі, відгуки наукового керівника та опонента, протоколи складання державних екзаменів, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Виконання етапів документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 – «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2022.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ—03.02.02—П.001.01:21».

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Чат-бот пошуку та замовлень товарів з онлайн-магазину

Тип роботи: бакалаврська кваліфікаційна робота

(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1.5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

Крупельницький Л.В., доц. каф ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Кадук О. В. к.т.н., доц. каф. ОТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Костюк О. Д.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Графічна частина

ЧАТ-БОТ ПОШУКУ ТА ЗАМОВЛЕНЬ ТОВАРІВ З ОНЛАЙН-МАГАЗИНУ

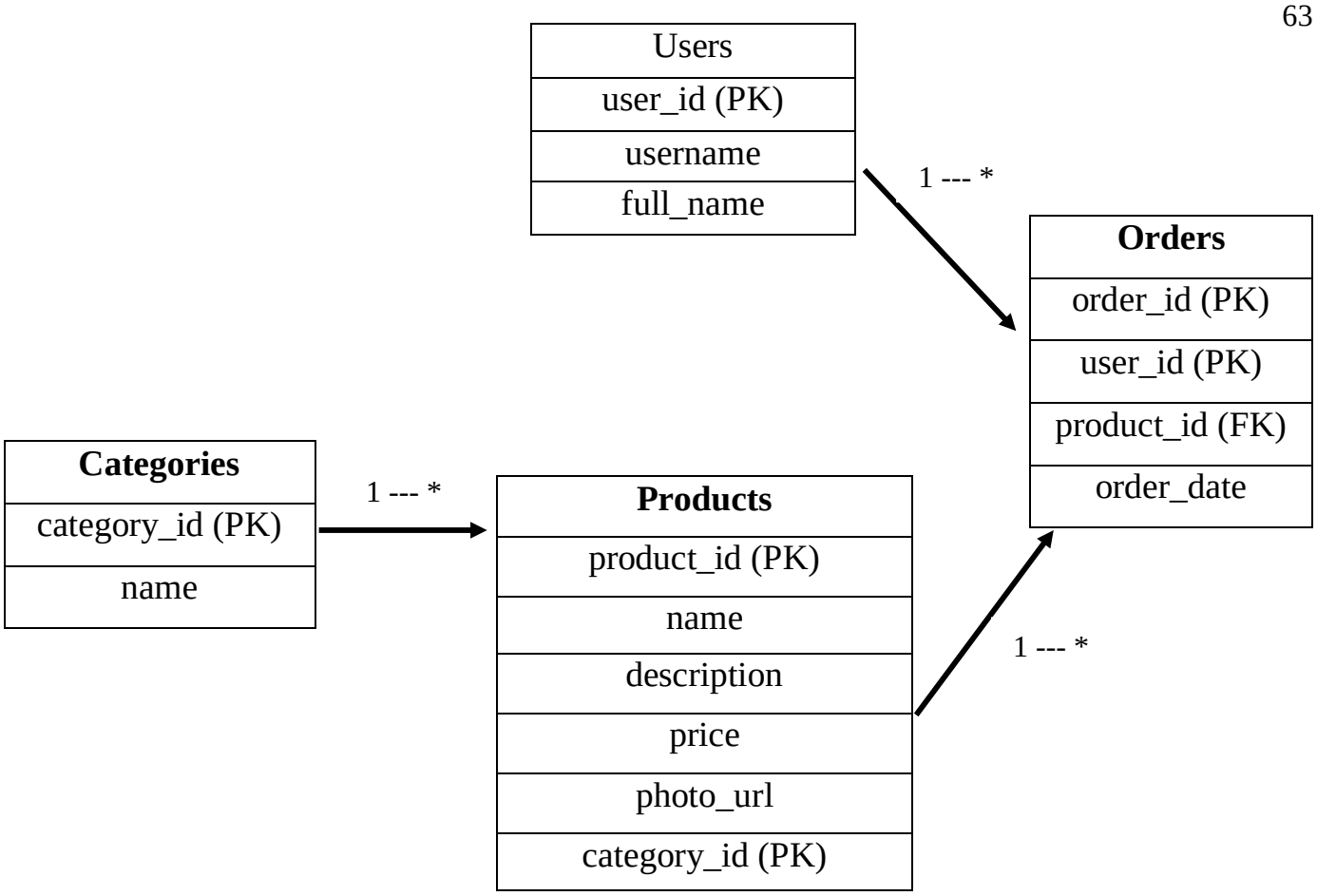


Рисунок В.1 — ER-схема бази даних чат-бота

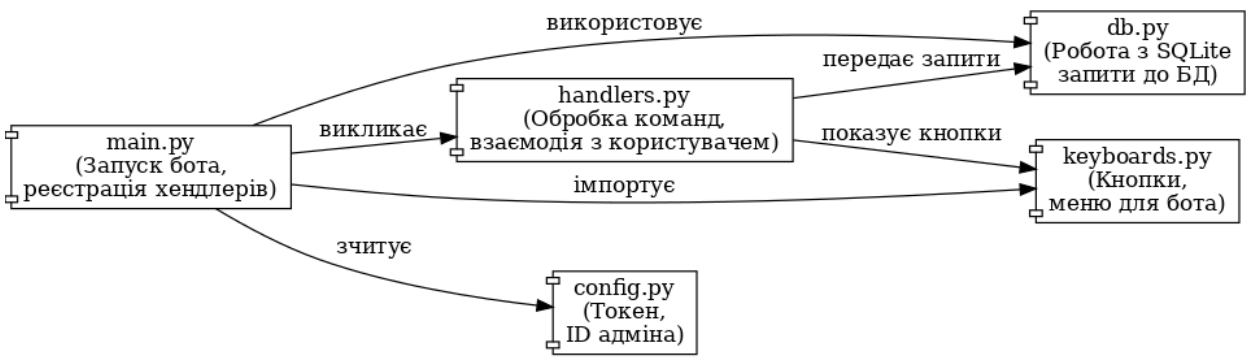


Рисунок В.2 — UML-схема компонентів Telegram-бота

ДОДАТОК Г

Ілюстративна частина

ЧАТ-БОТ ПОШУКУ ТА ЗАМОВЛЕНЬ ТОВАРІВ З ОНЛАЙН-МАГАЗИНУ

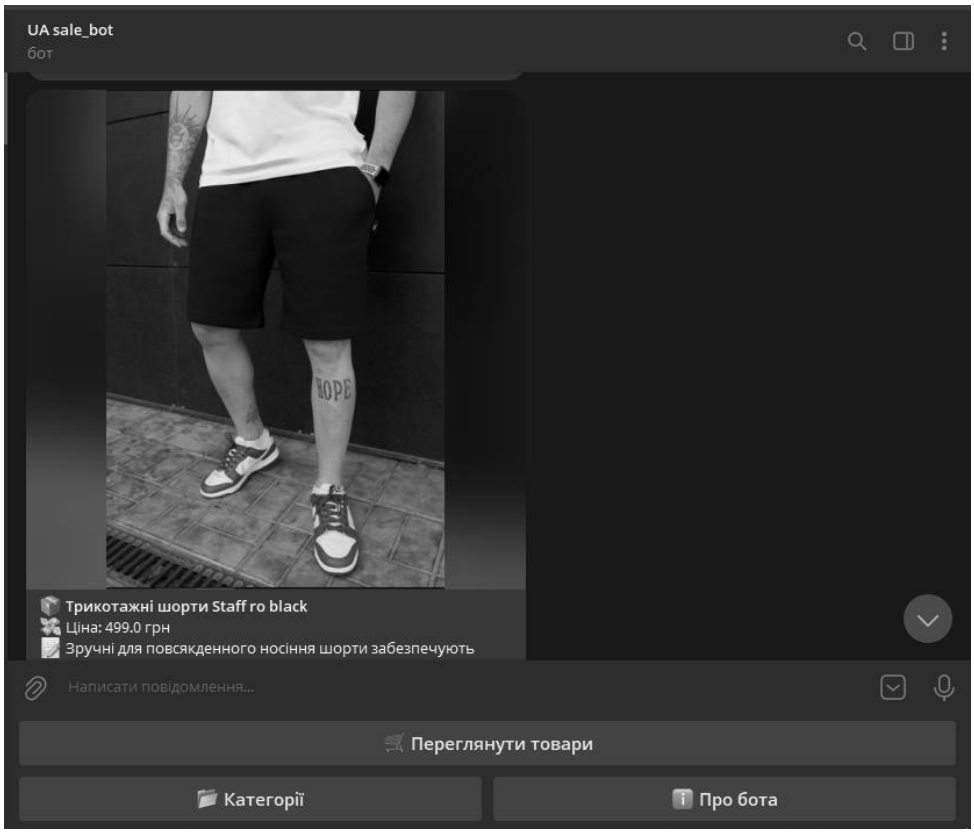


Рисунок Г.1 — Головна сторінка чат – бота

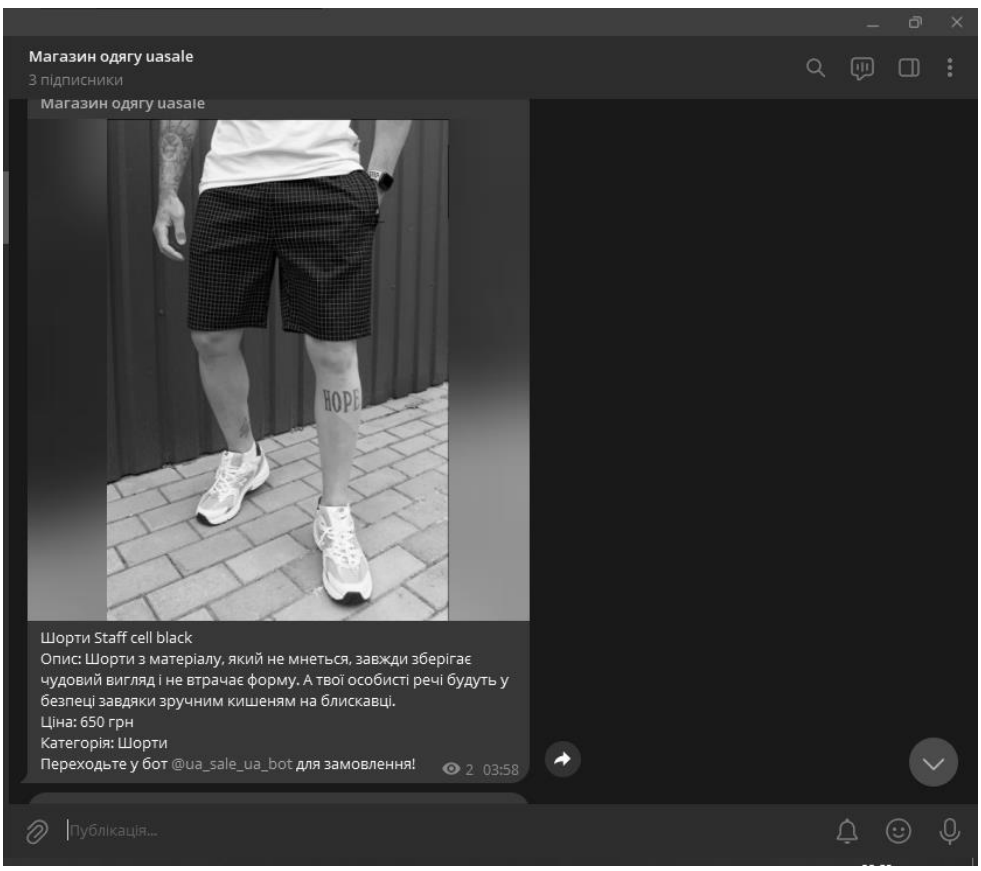


Рисунок Г.2 — Головна сторінка онлайн – магазину

ДОДАТОК Д

Лістинг коду обробки дій користувача чат-бота

```
from telegram import Update, InlineKeyboardMarkup, InlineKeyboardButton,
ReplyKeyboardMarkup, KeyboardButton
from telegram.constants import ParseMode
from telegram.ext import (
    CommandHandler,
    MessageHandler,
    filters,
    CallbackQueryHandler,
    ConversationHandler,
    ContextTypes,
)
from db import (
    get_all_products,
    add_product,
    get_all_products,
    get_all_categories,
    get_products_by_category,
    add_order,
    get_all_orders
)
from keyboards import main_menu
def get_product_name(product_id):
    try:
        products = get_all_products()
        for prod in products:
            if prod[0] == product_id:
                return prod[1]
    except Exception as e:
        print(f"❌ Не вдалося отримати назву товару: {e}")
    return "Невідомий товар"
ADMINS = [954607998]
WAITING_NAME, WAITING_PHONE = range(2)
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    args = context.args
    if args and args[0].startswith("product_"):
        try:
            product_id = int(args[0].split("_")[1])
            products = get_all_products()
            for prod in products:
                if prod[0] == product_id:
                    _, name, desc, price, _, image = prod
```

```
                text = f"\U0001F4E6 *{name}*\n\U0001F4B8 Ціна: {price}
грн\n\U0001F4DD {desc or 'Без опису'}"
                if image:
                    await update.message.reply_photo(
                        photo=image,
                        caption=text,
                        parse_mode=ParseMode.MARKDOWN,
                        reply_markup=product_actions(product_id)
                    )
                else:
                    await update.message.reply_text(
                        text,
                        parse_mode=ParseMode.MARKDOWN,
                        reply_markup=product_actions(product_id)
                    )
                return
        except:
            pass
    await update.message.reply_text(
        "Привіт! Вітаємо в Telegram-магазині \U0001F44B\nОберіть дію нижче:",
        reply_markup=main_menu()
    )
async def show_products(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    products = get_all_products()
    if not products:
        await update.message.reply_text("Список товарів наразі порожній.")
        return
    for prod_id, name, desc, price, _, image in products:
        text = f"\U0001F4E6 *{name}*\n\U0001F4B8 Ціна: {price}
грн\n\U0001F4DD {desc or 'Без опису'}"
        if image:
            await update.message.reply_photo(
                photo=image,
                caption=text,
                parse_mode=ParseMode.MARKDOWN,
                reply_markup=product_actions(prod_id)
            )
        else:
            await update.message.reply_text(
                text,
                parse_mode=ParseMode.MARKDOWN,
                reply_markup=product_actions(prod_id)
            )
async def show_categories(update: Update, context:
ContextTypes.DEFAULT_TYPE):
```

```
    categories = get_all_categories()
    if not categories:
        await update.message.reply_text("Категорій поки немає.")
        return
    await update.message.reply_text("Оберіть категорію:",
reply_markup=category_menu(categories))
async def category_selected(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()
    cat_id = int(query.data.replace("cat_", ""))
    products = get_products_by_category(cat_id)
    if not products:
        await query.edit_message_text("У цій категорії поки немає товарів.")
        return
    for prod_id, name, desc, price, _, image in products:
        text = f"\U0001F4E6 *{name}*\n\U0001F4B8 Ціна: {price}
грн\n\U0001F4DD {desc or 'Без опису'}"
        if image:
            await query.message.reply_photo(
                photo=image,
                caption=text,
                parse_mode=ParseMode.MARKDOWN,
                reply_markup=product_actions(prod_id)
            )
        else:
            await query.message.reply_text(
                text,
                parse_mode=ParseMode.MARKDOWN,
                reply_markup=product_actions(prod_id)
            )
async def about_bot(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Цей бот створено для перегляду та
замовлення товарів \U0001F6CD")
async def order_product(update: Update, context: ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()
    context.user_data['product_id'] = int(query.data.replace("order_", ""))
    await query.message.reply_text("\U0001F9FE Введіть ваше ім’я:")
    return WAITING_NAME
async def receive_name(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['customer_name'] = update.message.text.strip()
    await update.message.reply_text("\U0001F4DE Введіть ваш номер телефону:")
    return WAITING_PHONE
async def receive_phone(update: Update, context: ContextTypes.DEFAULT_TYPE):
    phone = update.message.text.strip()
```

```
    context.user_data['customer_phone'] = phone
    user_id = update.effective_user.id
    product_id = context.user_data['product_id']
    name = context.user_data['customer_name']
    phone = context.user_data['customer_phone']
    add_order(user_id, product_id, name, phone)
    await update.message.reply_text(
        "✅ Дякуємо за ваше замовлення!\n📞 З вами в скорому часі зв'яжеться наш оператор.\n\nГарного дня!",
        reply_markup=main_menu()
    )
    products = get_all_products()
    product_name = next((p[1] for p in products if p[0] == product_id), "Невідомий товар")
    for admin_id in ADMINS:
        try:
            await context.bot.send_message(
                chat_id=admin_id,
                text=(
                    f"🆕 НОВЕ ЗАМОВЛЕННЯ!\n"
                    f"👨‍👩‍👧 Ім’я: {name}\n"
                    f"📞 Телефон: {phone}\n"
                    f"🎁 Товар: {product_name}"
                )
            )
        except Exception as e:
            print(f"⚠️ Не вдалося надіслати повідомлення адміну: {e}")
    return ConversationHandler.END
async def cancel(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("\u274C Дію скасовано.", reply_markup=main_menu())
    return ConversationHandler.END
async def show_orders(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if update.effective_user.id not in ADMINS:
        await update.message.reply_text("⛔ У вас немає доступу до цієї команди.")
        return
    orders = get_all_orders()
    if not orders:
        await update.message.reply_text("📭 Замовлень поки немає.")
        return
    for order_id, telegram_id, created_at, status, customer_name, customer_phone, product_name, quantity in orders:
        text = (
            f"🆔 Замовлення #{order_id}\n"
            f"👤 Користувач ID: {telegram_id}\n"
```

```
            f"👥 Ім’я: {customer_name}\n"
            f"📞 Телефон: {customer_phone}\n"
            f"🎁 Товар: {product_name} × {quantity}\n"
            f"📅 Дата: {created_at[:19]}\n"
            f"📌 Статус: {status}"
        )
        await update.message.reply_text(text)
async def handle_channel_post(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    message = update.channel_post
    if not message.photo or not message.caption:
        return
    caption = message.caption
    photo = message.photo[-1].file_id
    try:
        lines = caption.strip().split("\n")
        name = lines[0].strip()
        desc = next((l.replace("Опис:", "").strip() for l in lines if l.startswith("Опис:")),
"")
        price_line = next((l for l in lines if "Ціна:" in l), None)
        price = float(price_line.replace("Ціна:", "").replace("грн", "").strip())
        category_line = next((l for l in lines if l.startswith("Категорія:")), None)
        category = category_line.replace("Категорія:", "").strip()
    except Exception as e:
        print(f"❌ Помилка при обробці поста: {e}")
        return
    categories = get_all_categories()
    cat_id = next((cid for cid, cname in categories if cname.lower() ==
category.lower()), None)
    if cat_id is None:
        print(f"⚠ Категорія не знайдена: {category}")
        return
    add_product(name, desc, price, cat_id, photo)
    print(f"✅ Товар додано: {name} | {price} грн")
def category_menu(categories):
    return InlineKeyboardMarkup(
        [[InlineKeyboardButton(name, callback_data=f"cat_{cat_id}")] for cat_id, name
in categories]
    )
def product_actions(product_id):
    return InlineKeyboardMarkup([
        [InlineKeyboardButton("\U0001F6CD Замовити",
callback_data=f"order_{product_id}")]
    ])
def register_handlers(application):
```

```
    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("categories", show_categories))
    application.add_handler(CommandHandler("orders", show_orders))
    application.add_handler(MessageHandler(filters.TEXT & filters.Regex("🛒 Переглянути товари"), show_products))
    application.add_handler(MessageHandler(filters.TEXT & filters.Regex("📂 Категорії"), show_categories))
    application.add_handler(MessageHandler(filters.TEXT & filters.Regex("ℹ️ Про бота"), about_bot))
    application.add_handler(CallbackQueryHandler(category_selected, pattern="^cat_"))
    conv_order = ConversationHandler(
        entry_points=[CallbackQueryHandler(order_product, pattern="^order_")],
        states={
            WAITING_NAME: [MessageHandler(filters.TEXT & ~filters.COMMAND, receive_name)],
            WAITING_PHONE: [MessageHandler(filters.TEXT & ~filters.COMMAND, receive_phone)],
        },
        fallbacks=[CommandHandler("cancel", cancel)],
    )
    application.add_handler(conv_order)
    application.add_handler(MessageHandler(filters.UpdateType.CHANNEL_POST, handle_channel_post))
```