ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавр

на тему: **«Розробка чат-боту для підтримки маркетплейсу»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_31[1]
Мітлицький Назар Миколайович
Керівник: Поночовний Юрій
Леонідович
Рецензент: Стрюк Олексій Юрійович

Полтава – 2026 року

ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

Освітня програма Інформаційні управляючі системи
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Юрій УТКІН
«10» червня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Мітлицькому Назару Миколайовичу

1. Тема роботи:
«Розробка чат-боту для підтримки маркетплейсу»,
керівник роботи д.т.н., професор, професор кафедри інформаційних систем та технологій Поночовний Юрій Леонідович.
Затверджено наказом закладу вищої освіти від «10» квітня 2026 року № 384 ст

2. Строк подання здобувачем вищої освіти роботи «08» червня 2026 року

3. Вихідні дані до роботи: стандарти проєктування та розробки програмного забезпечення, дані офіційних сайтів https://www.python.org/, https://pytba.readthedocs.io/ (бібліотека pyTelegramBotAPI), https://core.telegram.org/bots/api (Telegram Bot API), https://dev.mysql.com/ (MySQL), https://apscheduler.readthedocs.io/ (APScheduler), середовища прикладних програм PHPMyAdmin, OpenServer.

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

Розділ 1. Аналіз предметної області, технологій та вимог чат-боту
Розділ 2. Проєктування чат-боту підтримки маркетплейсу на основі UML-моделювання та реляційної бази даних
Розділ 3. Реалізація та оцінка ефективності чат-боту підтримки маркетплейсу

5. Перелік графічного матеріалу: схеми, рисунки, графіки, діаграми за темою та об’єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
Оцінювання економічної ефективності результатів дослідження	Загребельна І. Л., к. е. н., доцент, доцент кафедри економіки та публічного управління	01.04.2026 р.	29.05.2026 р.

7. Дата видачі завдання «10» червня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Вибір і затвердження теми роботи	02.06.2025 р. – 04.06.2025	
2	Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу	05.06.2025 р. – 10.06.2025 р.	
3	Опрацювання джерел інформації	11.06.2025 р. – 15.06.2025 р.	
4	Збір, вивчення і обробка інформації, необхідної для виконання роботи	16.06.2025 р. – 20.06.2025 р.	
5	Виконання теоретико-методологічного розділу роботи	08.09.2025 р. – 01.11.2025 р.	
6	Виконання дослідницько-аналітичного розділу роботи	19.01.2026 р. – 14.02.2026 р.	
7	Виконання проєктно-рекомендаційного розділу роботи	16.02.2026 р. – 31.03.2026 р.	
8	Розрахунок економічної ефективності результатів дослідження	01.04.2026 р. – 29.05.2026 р.	
9	Оформлення тексту роботи	01.06.2026 р. – 06.06.2026р.	
10	Попередній захист роботи на кафедрі	08.06.2026 р.	
11	Доопрацювання роботи з урахуванням зауважень і пропозицій	09.06.2026 р. – 10.06.2026 р.	
12	Нормоконтроль	11.06.2026 р. – 15.06.2026 р.	
13	Захист кваліфікаційної роботи	16.06.2026 р. – 18.06.2026 р.	

Здобувач вищої освіти **Назар МІТЛИЦЬКИЙ**

Керівник роботи **Юрій ПОНОЧОВНИЙ**

ЗМІСТ

ВСТУП

Актуальність. Стрімкий розвиток електронної комерції та цифрових технологій обумовлює зростання попиту на ефективні інструменти автоматизації клієнтської підтримки. Маркетплейси як багатосторонні цифрові платформи, що об'єднують продавців та покупців в єдиному середовищі, щодня обробляють тисячі звернень користувачів: від запитів щодо стану замовлень та умов повернення товарів до технічних питань і вирішення конфліктних ситуацій між учасниками торгового майданчика.

За даними світових аналітичних агентств, обсяг ринку електронної комерції у 2024 році перевищив 6 трильйонів доларів США, а частка маркетплейсів у загальному обсязі онлайн-продажів стабільно зростає і наразі становить понад 60%. В Україні цей ринок також демонструє активну динаміку: такі платформи як Rozetka, Prom.ua, OLX та інші обробляють мільйони транзакцій щорічно, стикаючись з необхідністю масштабування служб підтримки.

Традиційні моделі клієнтської підтримки, засновані на роботі операторів контакт-центрів, мають суттєві обмеження: значні витрати на утримання персоналу, неможливість цілодобового обслуговування без збільшення штату, людський фактор як причина помилок і непослідовності відповідей, а також тривалий час очікування у години пікового навантаження. Все це негативно впливає на рівень задоволеності клієнтів та, як наслідок, на репутацію і фінансові показники платформи.

Чат-боти на основі сучасних технологій штучного інтелекту та обробки природної мови є ефективним вирішенням зазначених проблем. Застосування чат-ботів дозволяє автоматизувати до 70–80% типових звернень користувачів, забезпечити миттєву відповідь незалежно від часу доби та пори року, значно знизити операційні витрати, водночас підвищуючи якість і швидкість обслуговування клієнтів.

Незважаючи на широке використання чат-ботів у провідних міжнародних компаніях (Amazon, Alibaba, eBay), вітчизняні маркетплейси все ще переважно покладаються на традиційні форми підтримки. Розробка спеціалізованого чат-боту для підтримки маркетплейсу, адаптованого до специфіки українського ринку та особливостей функціонування торгових платформ, є актуальним практичним завданням.

Метою кваліфікаційної роботи є розроблення чат-боту для автоматизації клієнтської підтримки маркетплейсу, що забезпечує обробку типових звернень користувачів, інтеграцію з базою даних платформи та підвищення ефективності обслуговування.

Для досягнення поставленої мети необхідно вирішити такі *завдання*:

1) проаналізувати особливості функціонування маркетплейсів та визначити типові сценарії звернень користувачів до служби підтримки;

2) дослідити існуючі рішення у сфері чат-ботів для електронної комерції та визначити їх переваги і недоліки;

3) сформулювати функціональні та нефункціональні вимоги до розроблюваного чат-боту;

4) обрати оптимальні технології та спроектувати архітектуру системи;

5) розробити і реалізувати чат-бот з інтеграцією до інфраструктури маркетплейсу;

6) провести тестування та оцінити ефективність і економічну доцільність впровадження розробленого рішення.

Об’єкт дослідження – процес автоматизованої підтримки користувачів маркетплейсу.

Предмет дослідження – методи, алгоритми та програмні засоби розроблення чат-ботів для систем електронної комерції.

Методи досліджень. У роботі використано методи системного аналізу та порівняльного огляду для дослідження предметної області, методи об'єктно-орієнтованого проєктування для розробки архітектури системи, а також методи

функціонального та навантажувального тестування для оцінки якості розробленого програмного продукту.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі проводиться аналіз предметної області: розглядаються поняття та особливості маркетплейсів, досліджуються існуючі рішення у сфері чат-ботів та формулюються вимоги до системи. Другий розділ присвячено проєктуванню та розробці чат-боту: вибору технологій, проєктуванню архітектури та реалізації інтеграції з платформою. У третьому розділі описано методику тестування, аналіз результатів впровадження та оцінку економічної доцільності розробленого рішення. Загальний обсяг текстової частини кваліфікаційної роботи складає 39 сторінок формату А4. Вона містить 11 рисунків і 8 таблиць.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ТЕХНОЛОГІЙ ТА ВИМОГ ДО ЧАТ-БОТІВ

1.1 Поняття та особливості функціонування маркетплейсів

Електронна комерція є одним із найбільш динамічно зростаючих секторів світової економіки. В її структурі особливе місце займають маркетплейси – цифрові платформи, що забезпечують взаємодію між продавцями товарів чи послуг та покупцями в рамках єдиного онлайн-середовища.

Маркетплейс (від англ. marketplace – торговий майданчик) – це багатостороння цифрова платформа, яка виступає посередником між постачальниками та споживачами, надаючи інфраструктуру для розміщення товарів, проведення платежів, управління замовленнями та вирішення спорів. На відміну від класичного інтернет-магазину, маркетплейс не обов'язково є власником товарів, що реалізуються на його платформі.

Відповідно до типу учасників торгових відносин маркетплейси поділяються на такі категорії:

– B2C (Business-to-Consumer) – платформи, де бізнес реалізує товари кінцевим споживачам (Rozetka, Amazon, AliExpress);

– B2B (Business-to-Business) – майданчики для оптової торгівлі між підприємствами (Alibaba, Prom.ua у B2B-сегменті);

– C2C (Consumer-to-Consumer) – платформи для продажу між фізичними особами (OLX, eBay у частині приватних продавців);

– Mixed – гібридні моделі, що поєднують кілька типів учасників (Amazon Marketplace, Prom.ua).

Функціонування маркетплейсу передбачає виконання кількох ключових функцій: агрегація пропозицій від множини продавців, управління каталогом

товарів, обробка платежів і забезпечення безпеки транзакцій, логістична підтримка, а також клієнтська підтримка всіх учасників платформи – як покупців, так і продавців.

Маркетплейси мають низку структурних особливостей, що відрізняють їх від інших форм електронної торгівлі. По-перше, наявність мережевого ефекту: чим більше продавців і покупців на платформі, тим вищою є її цінність для кожного учасника. По-друге, двостороння природа відносин: платформа одночасно обслуговує інтереси як продавців (надання інструментів продажу та аудиторії), так і покупців (широкий вибір, зручність, гарантії). По-третє, масштабованість: маркетплейс здатний обробляти мільйони транзакцій без пропорційного збільшення операційних витрат.

Ця двостороння природа створює специфічні виклики для організації клієнтської підтримки. Служба підтримки маркетплейсу змушена одночасно вирішувати різні типи звернень: питання щодо замовлень і доставки від покупців, технічні питання щодо лістингу та продажів від продавців, спори між сторонами угод, а також загальні питання щодо функціоналу платформи. Дослідження компанії Zendesk показують, що в середньому покупець маркетплейсу звертається до служби підтримки 2–4 рази на рік, а продавець – до 10–15 разів, що зумовлює значне навантаження на контакт-центри.

Серед провідних маркетплейсів, що функціонують на українському ринку, варто виділити Rozetka (більше 3 млн товарів від тисяч продавців), Prom.ua (понад 500 тис. активних продавців), OLX (понад 30 млн оголошень) та Hotline. Кожна з цих платформ стикається з необхідністю ефективного управління зверненнями своїх користувачів.

Таким чином, маркетплейс є складною багатошаровою цифровою екосистемою, функціонування якої вимагає ефективних механізмів автоматизованої підтримки всіх категорій учасників. Саме це обумовлює актуальність розробки спеціалізованого чат-боту для підтримки маркетплейсу.

1.2 Аналіз існуючих чат-ботів у сфері маркетплейсів

Чат-бот – це програмний застосунок, що імітує розмову з людиною у текстовому або голосовому форматі. У контексті електронної комерції чат-боти виконують роль першої лінії підтримки, здатної автономно відповідати на запити користувачів, не залучаючи живих операторів.

За архітектурним підходом сучасні чат-боти поділяються на два основних типи:

– Чат-боти на основі правил (rule-based) – функціонують за наперед визначеним сценарієм, обробляють запити через дерево рішень і ключові слова. Прості у розробці та передбачувані в поведінці, проте обмежені у роботі з нестандартними запитами.

– Чат-боти на основі штучного інтелекту (AI-powered) – використовують методи обробки природної мови (NLP) та машинного навчання для розуміння контексту і намірів користувача. Здатні обробляти складні запити, але вимагають значних ресурсів для навчання та підтримки.

На ринку представлено ряд спеціалізованих рішень для підтримки електронної комерції. Розглянемо найбільш поширені з них.

Amazon Lex – хмарний сервіс від Amazon Web Services для побудови розмовних інтерфейсів. Використовує ті ж технології, що й голосовий асистент Alexa. Забезпечує глибоку інтеграцію з екосистемою AWS та сервісами Amazon. Основні переваги: висока масштабованість, підтримка голосових і текстових інтерфейсів, вбудовані механізми NLU (Natural Language Understanding). Недоліки: значна вартість для малих і середніх бізнесів, складність початкового налаштування, орієнтованість на міжнародний ринок без адаптації до українського ринку.

Tidio – комплексна платформа для клієнтської підтримки з вбудованими можливостями чат-бота. Інтегрується з популярними CMS та e-commerce платформами (Shopify, WooCommerce, Magento). Переваги: простота

налаштування, готові шаблони для e-commerce, доступна вартість. Недоліки: обмежені можливості кастомізації логіки бота, відсутність глибокої інтеграції з кастомними маркетплейсами, не підтримує роботу з Telegram або Viber (популярних в Україні).

ManyChat – платформа для створення чат-ботів у месенджерах (Facebook Messenger, Instagram, WhatsApp, Telegram). Орієнтована на маркетингові сценарії та автоматизацію комунікацій. Переваги: підтримка популярних в Україні каналів (Telegram, Viber), візуальний конструктор сценаріїв, доступна вартість. Недоліки: орієнтована на маркетинг, а не на технічну підтримку; слабкі можливості інтеграції з базами даних маркетплейсів.

Zendesk Answer Bot – компонент платформи Zendesk для автоматизації відповідей на типові звернення. Переваги: глибока інтеграція з CRM-системою Zendesk, корисна аналітика, навчання на базі реальних тікетів підтримки. Недоліки: висока вартість, залежність від екосистеми Zendesk.

Таблиця 1.1 – Порівняльний аналіз існуючих чат-ботів для маркетплейсів

Критерій	Amazon Lex	Tidio	ManyChat	Zendesk Bot
NLP / AI	Так	Частково	Частково	Так
Telegram/Viber	Ні	Ні	Так	Ні
Інтеграція з БД	Висока	Низька	Низька	Середня
Вартість	Висока	Низька	Низька	Висока
Кастомізація	Висока	Середня	Середня	Середня
Укр. ринок	Частково	Ні	Так	Ні

Проведений аналіз свідчить, що жодне з розглянутих рішень не задовольняє повною мірою потреби вітчизняних маркетплейсів: або через відсутність підтримки популярних в Україні каналів комунікації (Telegram, Viber), або через недостатні можливості інтеграції з кастомними системами, або через неадекватну ціну для малого та середнього бізнесу. Це підтверджує доцільність розробки власного рішення.

1.3 Визначення вимог до чат-боту підтримки маркетплейсу

На основі проведеного аналізу предметної області та огляду існуючих рішень сформуємо систему вимог до розроблюваного чат-боту. Вимоги поділяються на функціональні (що система повинна робити) та нефункціональні (як система повинна це робити).

Функціональні вимоги описують конкретні функції та сценарії використання системи:

1. Обробка типових запитів покупців: надання інформації про статус замовлення та доставку; відповіді на питання щодо умов повернення та обміну товарів; консультація щодо оплати та доступних платіжних методів; допомога в пошуку товарів за параметрами.

2. Обробка запитів продавців: відповіді на питання щодо розміщення лістингів; консультація з умов роботи на платформі та комісійної політики; допомога в управлінні замовленнями та поверненнями.

3. Ескалація складних запитів: виявлення звернень, що виходять за межі компетенції бота, та їх передача живому оператору з усією контекстною інформацією розмови.

4. Інтеграція з базою даних маркетплейсу: отримання актуальних даних про замовлення, товари, профілі користувачів та статуси доставки в режимі реального часу.

5. Ведення історії звернень: збереження журналу всіх взаємодій з можливістю аналізу для поліпшення сервісу.

Нефункціональні вимоги визначають якісні характеристики системи:

6. Продуктивність: час відповіді чат-боту на запит користувача не повинен перевищувати 2 секунди при нормальному навантаженні; система повинна забезпечувати одночасне обслуговування не менше 500 активних сесій.

7. Доступність: система повинна функціонувати цілодобово (24/7) з рівнем доступності не нижче 99,5% на місяць (не більше 3,6 годин простою на місяць).

8. Надійність: усі дані про звернення мають зберігатися в надійній базі даних з резервним копіюванням; у разі відмови модуля NLP бот має коректно деградувати до режиму «rule-based».

9. Безпека: передача персональних даних користувачів здійснюється виключно по захищеному протоколу HTTPS/TLS; система не зберігає платіжних даних та дотримується вимог GDPR та законодавства України про захист персональних даних.

10. Юзабіліті: інтерфейс чат-боту має бути інтуїтивно зрозумілим; бот повинен надавати підказки щодо можливих дій, уникати технічного жаргону у відповідях.

Для систематизації функціональних вимог доцільно виділити основні сценарії використання (use cases) системи. Ключовими акторами є: Покупець, Продавець та Адміністратор платформи. Основні сценарії включають: отримання інформації про замовлення, ініціювання процедури повернення, отримання консультації по товару, реєстрацію скарги та ескалацію запиту до оператора. Сформульовані вимоги стануть основою для прийняття архітектурних рішень та вибору технологічного стеку у наступному розділі роботи.

Було проведено комплексний аналіз предметної області. Визначено поняття маркетплейсу як багатосторонньої цифрової платформи та виявлено специфіку організації підтримки в умовах двосторонньої природи відносин між учасниками. На підставі виявлених прогалин сформульовано функціональні та нефункціональні вимоги до розроблюваного чат-боту, які слугуватимуть основою для проєктування системи у наступному розділі.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ЧАТ-БОТУ ПІДТРИМКИ МАРКЕТПЛЕЙСУ НА ОСНОВІ UML-МОДЕЛЮВАННЯ ТА РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ

2.1 Розробка функціональних та нефункціональних вимог

Розробка вимог є фундаментальним етапом життєвого циклу програмного забезпечення, що передує безпосередньому проєктуванню архітектури та структури даних. Відповідно до стандарту IEEE 29148:2018, вимоги до системи поділяються на функціональні, що описують поведінку системи, та нефункціональні, що визначають якісні характеристики її функціонування [28]. Коректно задокументовані вимоги забезпечують однозначне розуміння цілей розробки, слугують основою для верифікації готового продукту та знижують ризик переробок на пізніх етапах [29].

Функціональні вимоги до чат-боту підтримки маркетплейсу сформовано на основі аналізу потреб користувачів, проведеного в першому розділі. Кожна вимога має унікальний ідентифікатор, що спрощує її трасування від специфікації до реалізації та тестування. Повний перелік функціональних вимог із пріоритетами за методом MoSCoW наведено в таблиці 2.1.

Нефункціональні вимоги визначають стандарти якості, яким повинна відповідати система. З точки зору продуктивності час відгуку на будь-який запит користувача не повинен перевищувати 1000 мс за умов стандартного інтернет-з’єднання, а планувальник повинен перевіряти чергу сповіщень з інтервалом не більше 60 секунд. З точки зору надійності система повинна коректно опрацьовувати некоректні введення користувача (текст замість числа, неіснуючу опцію) без аварійного завершення, а всі взаємодії з базою даних виконувати в межах транзакцій.

З точки зору зручності використання інтерфейс повністю реалізується через кнопкове меню Telegram, що виключає необхідність запам’ятовування команд, а всі повідомлення формуються українською мовою. З точки зору

безпеки система зберігає мінімально необхідний обсяг персональних даних – лише Telegram ID користувача та дані його замовлень, а всі запити до бази даних виконуються через параметризовані вирази для унеможливлення SQL-ін’єкцій [31]. З точки зору супроводжуваності код структурується за модульним принципом, а база даних допускає додавання нових товарів і категорій без зміни схеми.

Таблиця 2.1 – Функціональні вимоги до інформаційної системи

ID	Вимога	Пріоритет
FR-01	Система повинна ідентифікувати користувача за його Telegram ID без окремої реєстрації	Must have
FR-02	Система повинна надавати перегляд каталогу товарів за категоріями	Must have
FR-03	Система повинна відображати картку товару з ціною та зображенням	Must have
FR-04	Система повинна дозволяти оформлення замовлення із зазначенням кількості товару	Must have
FR-05	Система повинна автоматично розраховувати вартість замовлення	Must have
FR-06	Система повинна відображати перелік замовлень користувача	Must have
FR-07	Система повинна визначати та відображати поточний статус замовлення	Must have
FR-08	Система повинна надсилати автоматичні сповіщення про зміну статусу замовлення	Must have
FR-09	Система повинна відображати графік запланованих сповіщень замовлення	Should have
FR-10	Система повинна дозволяти скасування замовлення користувачем	Should have
FR-11	Система повинна надавати базу поширених запитань (FAQ)	Should have
FR-12	Система повинна дозволяти створення звернення до служби підтримки	Should have
FR-13	Система повинна формувати перелік рекомендованих популярних товарів	Could have

Для оцінки повноти специфікації використано коефіцієнт покриття варіантів використання, що визначається як відношення кількості варіантів використання, забезпечених щонайменше однією вимогою, до загальної їх кількості:

$$C_{uc} = (UC_r / UC_{total}) \cdot 100\%. \quad (2.1)$$

Для розроблюваної системи визначено вісім варіантів використання, кожен з яких покривається щонайменше однією вимогою з таблиці 2.1, тому

C_{uc} = 100%. Отриманий показник підтверджує повноту специфікації: кожен сценарій взаємодії користувача із системою забезпечений відповідною функціональною вимогою.

2.2 Побудова діаграм UML для моделювання поведінки системи

Уніфікована мова моделювання UML (Unified Modeling Language) є стандартним інструментом візуального опису програмних систем, що дозволяє формалізувати вимоги, архітектуру та поведінку системи у вигляді графічних діаграм [33]. Для розроблюваного чат-боту побудовано три типи діаграм – варіантів використання, послідовності та діяльності, – що разом утворюють повну поведінкову модель, достатню для переходу до проєктування бази даних [34]. Діаграму варіантів використання наведено на рисунку 2.1.

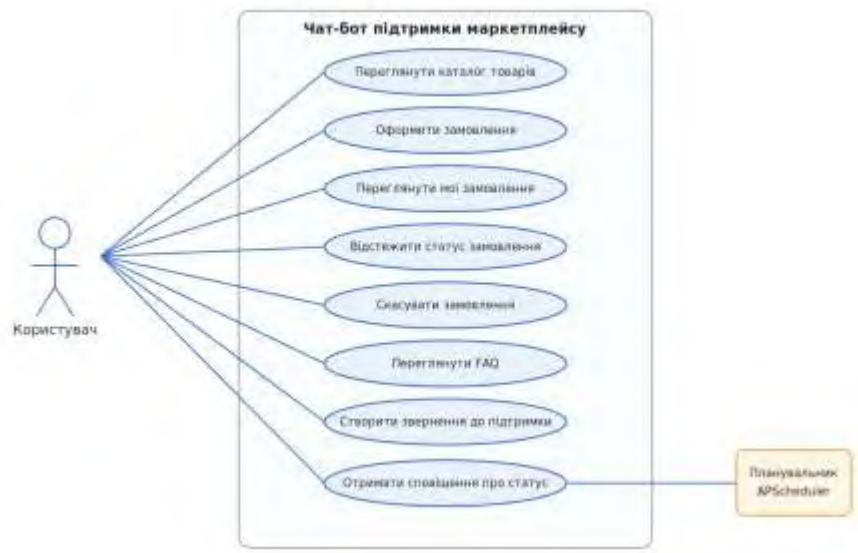

Рисунок 2.1 – Діаграма варіантів використання інформаційної системи

Основним актором системи є Користувач – будь-яка фізична особа, зареєстрована в Telegram, що взаємодіє з ботом через його інтерфейс. Другим актором є Планувальник (APScheduler) – внутрішній компонент, що ініціює

надсилання сповіщень без участі користувача. Для детальнішого опису ключового сценарію складено специфікацію варіанту використання «Оформити замовлення», наведену в таблиці 2.2.

Таблиця 2.2 – Специфікація варіанту використання «Оформити замовлення»

Поле	Зміст
Ідентифікатор	UC-04
Назва	Оформити замовлення
Актор	Користувач
Передумова	Користувач переглядає картку обраного товару
Тригер	Користувач натискає «Замовити» та вводить кількість товару
Основний сценарій	1. Система перевіряє коректність введеної кількості. 2. Обчислює вартість замовлення. 3. Створює запис замовлення та його позицію. 4. Розраховує та планує сповіщення для всіх етапів обробки. 5. Надсилає користувачу підтвердження
Альтернативний сценарій	Якщо кількість введено некоректно – система повторно запитує значення
Постумова	Замовлення створено, сповіщення заплановано
Пов’язані вимоги	FR-04, FR-05, FR-08

Діаграма послідовності відображає хронологічний порядок обміну повідомленнями між компонентами системи під час виконання сценарію оформлення замовлення [34]. Її наведено на рисунку 2.2.

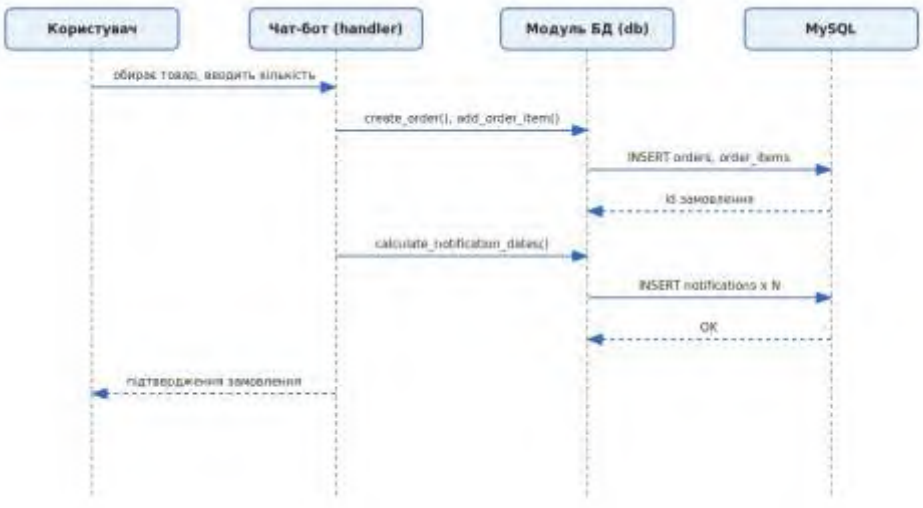

Рисунок 2.2 – Діаграма послідовності для варіанту використання «Оформити замовлення»

Діаграма діяльності описує алгоритм визначення поточного статусу замовлення – ключового алгоритму системи – у вигляді послідовності дій з умовними переходами [33]. Відповідну діаграму наведено на рисунку 2.3.

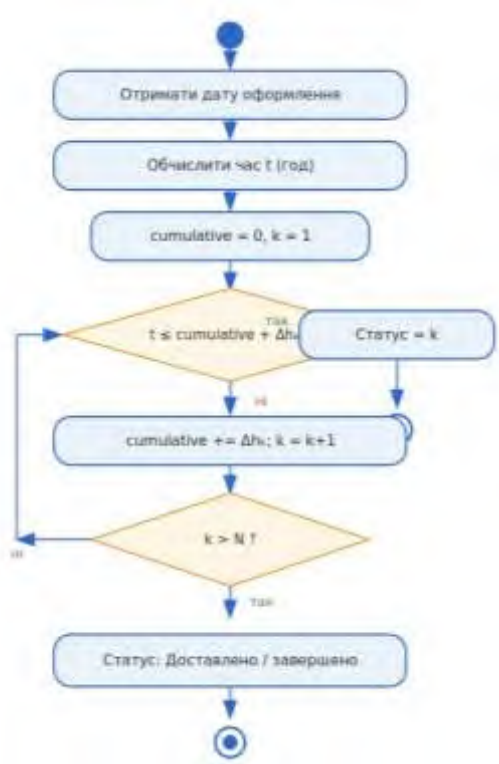

Рисунок 2.3 – Діаграма діяльності процесу визначення статусу замовлення

Алгоритм реалізується через послідовне порівняння часу, що минув від оформлення замовлення, з накопиченою тривалістю етапів обробки. Якщо позначити час від оформлення як t, а накопичену тривалість перших k етапів як S_k, то поточний статус ψ визначається як:

$$\psi = min\{\, k \in N : t \le S_k \,\}, \quad де\ S_k = \Sigma\, \Delta h_i, \tag{2.2}$$

де Δh_i – тривалість і-го етапу обробки в годинах, отримана з таблиці order_statuses.

Якщо t перевищує накопичену тривалість усіх етапів, система повертає статус завершення (замовлення доставлено). Побудовані UML-діаграми слугують основою для проєктування реляційної бази даних.

2.3 Проєктування структури бази даних та опис схеми відносин

Проєктування бази даних є центральним етапом розробки, оскільки якість схеми даних безпосередньо визначає продуктивність системи, зручність супроводу та можливість масштабування [18]. Процес проєктування виконано у три етапи: концептуальне моделювання (визначення сутностей і зв’язків), логічне (нормалізація та визначення атрибутів) та фізичне (визначення типів даних та індексів) [29]. Опис основних таблиць та атрибутів бази даних наведено в таблиці 2.3.

Таблиця 2.3 – Опис таблиць та атрибутів бази даних

Таблиця	Атрибут	Тип	Обмеження	Призначення
users	id	INT	PK, AI	внутрішній ідентифікатор
users	telegram_id	BIGINT	UNIQUE, NN	ідентифікатор у Telegram
categories	id	INT	PK, AI	ідентифікатор категорії
products	category_id	INT	FK → categories.id	посилання на категорію
products	price	DECIMAL	NN	ціна товару
orders	user_id	INT	FK → users.id	посилання на користувача
orders	total_price	DECIMAL	NN	загальна вартість
order_items	order_id	INT	FK → orders.id	посилання на замовлення
order_items	product_id	INT	FK → products.id	посилання на товар
order_statuses	duration_hours	INT	NN	тривалість етапу, год
notifications	order_id	INT	FK → orders.id	посилання на замовлення
notifications	notify_at	DATETIME	NN	час надсилання
support_tickets	user_id	INT	FK → users.id	автор звернення

На етапі концептуального моделювання виявлено вісім основних сутностей: users, categories, products, orders, order_statuses, notifications, faq та support_tickets. Між ними встановлено зв’язки типу «один до багатьох»: одна категорія містить багато товарів, одне замовлення – багато позицій і сповіщень

тощо. Зв’язок «багато до багатьох» між замовленнями та товарами реалізовано через асоціативну таблицю order_items, що містить кількість і ціну кожної позиції [33].

Логічна схема пройшла нормалізацію до третьої нормальної форми (3НФ): відношення перебуває у 3НФ, якщо воно у другій нормальній формі та не містить транзитивних залежностей неключових атрибутів від первинного ключа [32]. Дотримання 3НФ усуває надлишковість і унеможливлює аномалії вставки, оновлення та видалення. Фізичну ER-діаграму бази даних наведено на рис. 2.4.

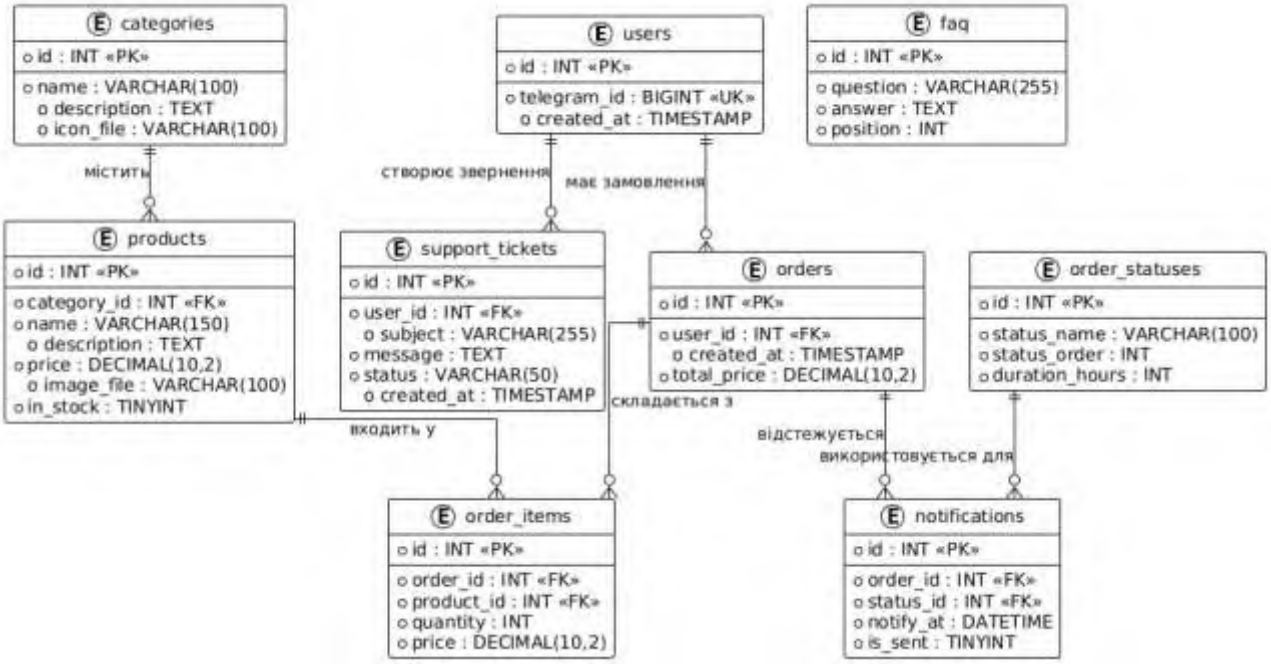

Рисунок 2.4 – ER-діаграма бази даних інформаційної системи

Важливим аспектом фізичного проєктування є визначення індексів. Окрім первинних ключів, що індексуються автоматично, додатковий складений індекс встановлюється на таблицю notifications за полями (notify_at, is_sent), оскільки планувальник кожні 60 секунд виконує запит SELECT … FROM notifications WHERE notify_at ≤ NOW() AND is_sent = 0. Без індексу цей запит виконував би повне сканування таблиці, тоді як індекс дозволяє отримувати результат за логарифмічний час відносно кількості рядків [18].

Загальна кількість таблиць бази даних складає дев’ять, кількість атрибутів – близько сорока, кількість зовнішніх ключів – сім. Коефіцієнт нормалізації схеми, що визначається як відношення кількості таблиць після нормалізації до кількості початкових сутностей, становить $Kn = 9 / 8 \approx 1{,}13$, що є типовим для нормалізованих реляційних схем і свідчить про коректне усунення надлишковості внаслідок додавання асоціативної таблиці order_items. Спроєктована структура є розширюваною: додавання нових товарів або категорій виконується вставкою рядків без зміни схеми.

2.4 Архітектура взаємодії чат-боту з інформаційною системою

Архітектура програмної системи визначає спосіб організації її компонентів та розподіл відповідальності між ними [15]. Для розроблюваного чат-боту обрано трирівневу архітектуру, що забезпечує чітке розмежування між рівнем представлення, рівнем бізнес-логіки та рівнем даних [22]. Така організація спрощує супровід коду, дозволяє незалежно модифікувати окремі рівні та полегшує тестування. Загальну архітектурну схему наведено на рисунку 2.5.

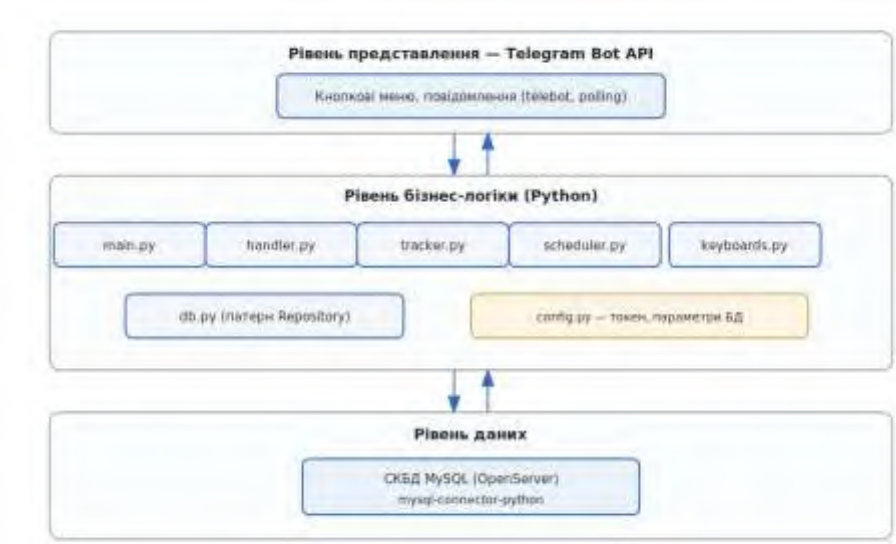

Рисунок 2.5 – Трирівнева архітектура інформаційної системи чат-боту

Рівень представлення реалізований засобами Telegram Bot API і являє собою інтерфейс взаємодії з користувачем у вигляді кнопкових меню та повідомлень. Бібліотека pyTelegramBotAPI забезпечує отримання подій від серверів Telegram через механізм polling – циклічне опитування на наявність нових повідомлень або натискань кнопок [17]. Для локального розгортання в межах роботи обрано polling як простіше рішення, що не потребує зовнішньої IP-адреси та SSL-сертифіката.

Рівень даних представлений реляційною базою даних MySQL, схему якої описано в підрозділі 2.3. Взаємодія між рівнем бізнес-логіки та рівнем даних відбувається виключно через модуль db.py з використанням бібліотеки mysql-connector-python та параметризованих запитів [19]. Розподіл функцій між модулями системи наведено в таблиці 2.4.

Таблиця 2.4 – Модульна структура програмного забезпечення системи

Модуль	Файл	Основні функції	Залежності
Головний модуль	main.py	ініціалізація, запуск polling та планувальника	telebot, scheduler
Обробник подій	handler.py	маршрутизація команд і кнопок	telebot, tracker, db
Трекер статусу	tracker.py	розрахунок статусу, графік сповіщень	db
Модуль БД	db.py	CRUD-операції, керування з’єднанням	mysql-connector
Планувальник	scheduler.py	перевірка та надсилання сповіщень	APScheduler, db
Клавіатури	keyboards.py	формування inline-меню	telebot, db
Налаштування	config.py	токен бота, параметри БД, константи	–

Як видно з таблиці 2.4, модуль db.py є єдиною точкою доступу до бази даних для всіх інших компонентів. Такий підхід, відомий як патерн Repository [22], спрощує заміну або міграцію СКБД у майбутньому: достатньо змінити реалізацію лише одного модуля, не торкаючись логіки обробників чи планувальника. Описана архітектура є збалансованою з точки зору складності реалізації та функціональних можливостей.

Рівень бізнес-логіки є ядром системи й реалізований у вигляді набору Python-модулів. Обробник подій (handler.py) приймає події від telebot, визначає тип взаємодії та делегує виконання відповідному модулю. Модуль трекера (tracker.py) розраховує поточний статус замовлення за алгоритмом із підрозділу 2.2 та формує текст відповіді. Модуль роботи з базою даних (db.py) інкапсулює всі операції зі збереження та читання даних. Планувальник (scheduler.py) на базі APScheduler виконує фонове завдання, перевіряє таблицю notifications та надсилає сповіщення користувачам [20].

У другому розділі виконано повний цикл проєктування чат-боту підтримки маркетплейсу. Сформовано та пріоритизовано функціональні й нефункціональні вимоги із застосуванням методу MoSCoW, побудовано комплект UML-діаграм (варіантів використання, послідовності та діяльності), що формалізують поведінку системи. Спроєктовано нормалізовану до третьої нормальної форми реляційну базу даних із дев’яти таблиць засобами MySQL та визначено трирівневу архітектуру програмного забезпечення з чітким розподілом відповідальності між модулями й застосуванням патерну Repository. Отримані проєктні рішення є достатньою основою для переходу до реалізації системи, що розглядається в третьому розділі.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЧАТ-БОТУ ПІДТРИМКИ МАРКЕТПЛЕЙСУ

3.1 Реалізація бази даних та наповнення тестовими даними

Реалізація бази даних є першим практичним кроком третього розділу і безпосередньо базується на схемі відносин, спроєктованій у підрозділі 2.3. Базу даних розгорнуто засобами MySQL 8.4 у складі пакету OpenServer 6.5.1, що відповідає обраному технологічному стеку [18]. Усі таблиці створено з кодуванням utf8mb4 та зіставленням utf8mb4_unicode_ci, що забезпечує коректне відображення україномовного контенту – назв категорій, товарів та текстів сповіщень. Результат успішного розгортання схеми в PHPMyAdmin наведено на рисунку 3.1.

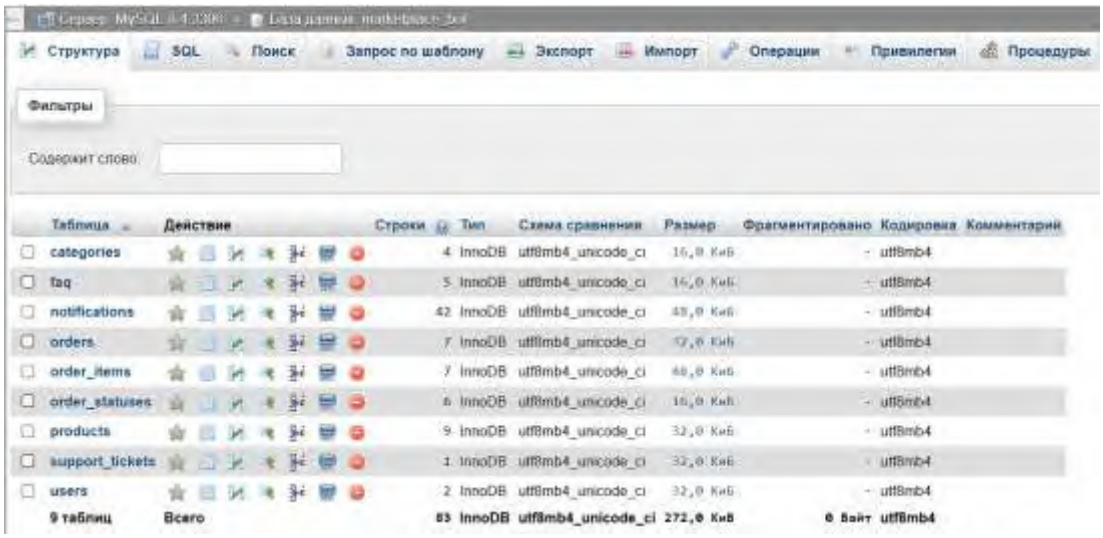

Рисунок 3.1 – Структура бази даних marketplace_bot у PHPMyAdmin

Фізичне створення структури виконувалося шляхом виконання SQL-дампу через інтерфейс PHPMyAdmin, вбудований до OpenServer. Дамп містить послідовне створення дев’яти таблиць із визначенням первинних ключів, зовнішніх ключів та обмежень цілісності. Порядок створення таблиць суворо

дотримується ієрархії залежностей: спочатку довідникові таблиці (users, categories, order_statuses, faq), потім залежні від них (products, orders), і в останню чергу – таблиці з кількома зовнішніми ключами (order_items, notifications, support_tickets). Така послідовність унеможливлює порушення обмежень зовнішніх ключів під час імпорту [35].

Наповнення бази тестовими даними виконувалося у два етапи. Перший етап – наповнення довідкової частини бази, що не залежить від дій користувача. До таблиці categories внесено чотири категорії товарів (електроніка, дім та сад, одяг та взуття, книги), до таблиці products – дев’ять товарів із зазначенням ціни та імені графічного файлу, що зберігається в папці images/products проєкту. До таблиці order_statuses внесено шість етапів обробки замовлення з визначенням їх тривалості в годинах, а до таблиці faq – п’ять поширених запитань із відповідями. Розподіл товарів за категоріями наведено в таблиці 3.1.

Таблиця 3.1 – Розподіл товарів за категоріями каталогу

Категорія	Кількість товарів	Приклади товарів
Електроніка	3	смартфон, навушники, power bank
Дім та сад	2	електрочайник, настільна лампа
Одяг та взуття	2	футболка, кросівки
Книги	2	видання з Python та з алгоритмів
Разом	9	-

Другий етап – внесення тестових користувацьких даних для перевірки функціональності системи. До таблиці users внесено тестового користувача, якому відповідає реальний Telegram ID розробника. До таблиці orders внесено кілька тестових замовлень із датами оформлення, що дозволяє системі розраховувати поточний статус станом на момент перегляду, а до таблиці order_items – відповідні позиції замовлень. До таблиці notifications внесено заплановані сповіщення, частина з яких позначена як відправлені (is_sent = 1), а частина – як заплановані (is_sent = 0). Коректність роботи складеного індексу планувальника перевірялася запитом безпосередньо в PHPMyAdmin. Вміст таблиці notifications наведено на рисунку 3.2.

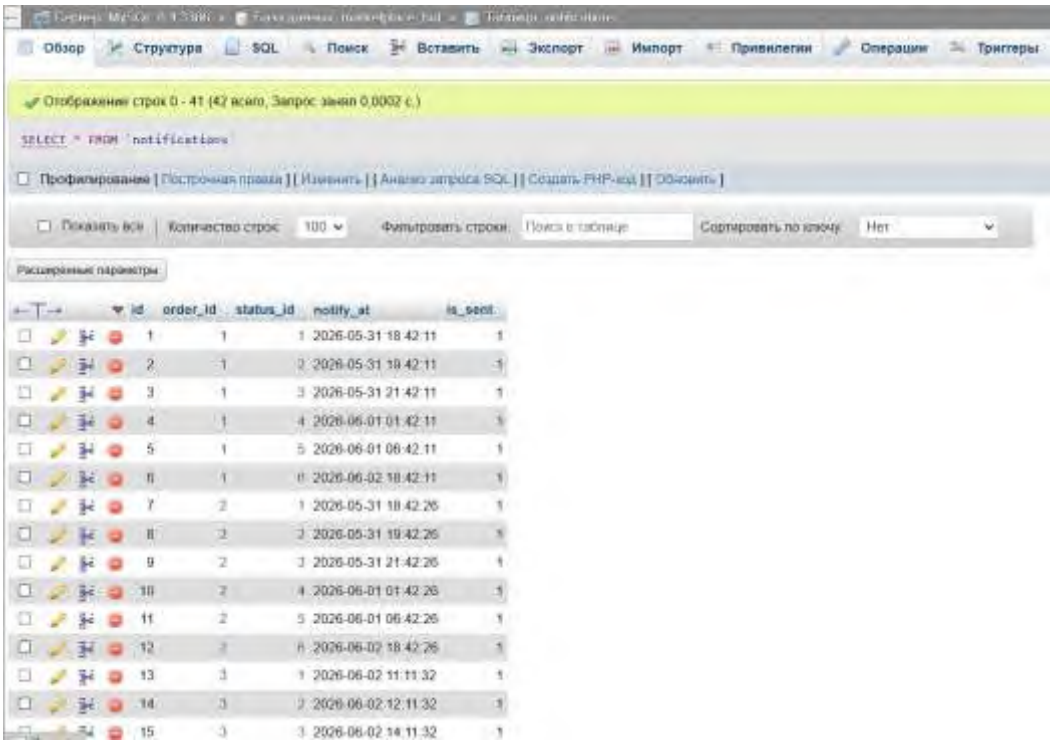

Рисунок 3.2 – Вміст таблиці notifications у PHPMyAdmin

Розгорнута та наповнена база даних є повністю готовою до взаємодії з програмним модулем бота. Усі зовнішні ключі перевірено на цілісність, індекси – на коректність побудови. Наступним кроком є реалізація програмного коду чат-боту, що звертається до цієї бази через модуль db.py.

3.2 Розробка програмного коду чат-боту та опис ключових модулів

Програмний код інформаційної системи реалізовано мовою Python 3.12 і організовано за модульним принципом відповідно до трирівневої архітектури, описаної в підрозділі 2.4. Проєкт складається із семи модулів: main.py, config.py, db.py, handler.py, keyboards.py, tracker.py та scheduler.py. Кожен модуль відповідає за окрему функціональну зону системи і взаємодіє з іншими через чітко визначені інтерфейси [22]. Така організація коду відповідає принципу єдиної відповідальності й суттєво спрощує супровід та налагодження системи [15].

Модуль db.py реалізує патерн Repository і є єдиною точкою доступу всіх компонентів системи до бази даних MySQL [22]. Центральною функцією модуля є execute_query, яка приймає SQL-запит і кортеж параметрів, встановлює з'єднання, виконує запит у межах транзакції та повертає результат. Використання параметризованих запитів у всіх методах унеможливлює SQL-ін'єкції [31]. З'єднання з базою відкривається і закривається при кожному виклику через блок finally, що запобігає витоку з'єднань при тривалій роботі. Основні публічні функції модуля наведено в таблиці 3.2.

Таблиця 3.2 – Основні публічні функції модуля db.py

Функція	Призначення	Операція
get_or_create_user	ідентифікація або реєстрація користувача	SELECT/INSERT
get_all_categories	отримання переліку категорій	SELECT
get_products_by_category	отримання товарів категорії	SELECT
get_product_by_id	отримання даних товару	SELECT
create_order	створення замовлення	INSERT
add_order_item	додавання позиції замовлення	INSERT
get_user_orders	отримання замовлень користувача	SELECT
get_order_items	отримання позицій замовлення	SELECT
delete_order	скасування замовлення	DELETE
add_notification	планування сповіщення	INSERT
get_pending_notifications	отримання сповіщень для надсилання	SELECT
mark_notification_sent	позначення сповіщення відправленим	UPDATE
add_support_ticket	створення звернення до підтримки	INSERT

Модуль tracker.py реалізує ключовий алгоритм системи – визначення поточного статусу замовлення та формування тексту картки замовлення. Функція get_current_status обчислює час, що минув від оформлення замовлення, і послідовно порівнює його з накопиченою тривалістю етапів обробки до знаходження активного статусу. Функція build_order_text на основі знайденого статусу формує структурований текст із позиціями замовлення, сумою та таймлайном етапів. Функція calculate_notification_dates розраховує час

сповіщень для всіх етапів обробки і викликається автоматично при оформленні замовлення.

Точкою входу в систему є модуль main.py, що виконує ініціалізацію компонентів у визначеній послідовності: спочатку запускається планувальник сповіщень у фоновому потоці, потім активується механізм polling для прийому подій від Telegram API. Такий порядок гарантує, що жодне сповіщення не буде пропущено навіть у момент запуску системи. Усі параметри підключення – токен бота, реквізити бази даних та інтервал планувальника – винесено до окремого файлу config.py, що є стандартною практикою для забезпечення гнучкості налаштування та безпеки [31].

Модуль keyboards.py централізовано зберігає всі inline-клавіатури бота, що дозволяє змінювати структуру меню в одному місці. Клавіатури генеруються динамічно: наприклад, функція my_orders_kb щоразу звертається до бази даних і формує кнопки відповідно до актуального переліку замовлень користувача [17]. Головне меню бота наведено на рисунку 3.3, а картку товару з зображенням – на рисунку 3.4.

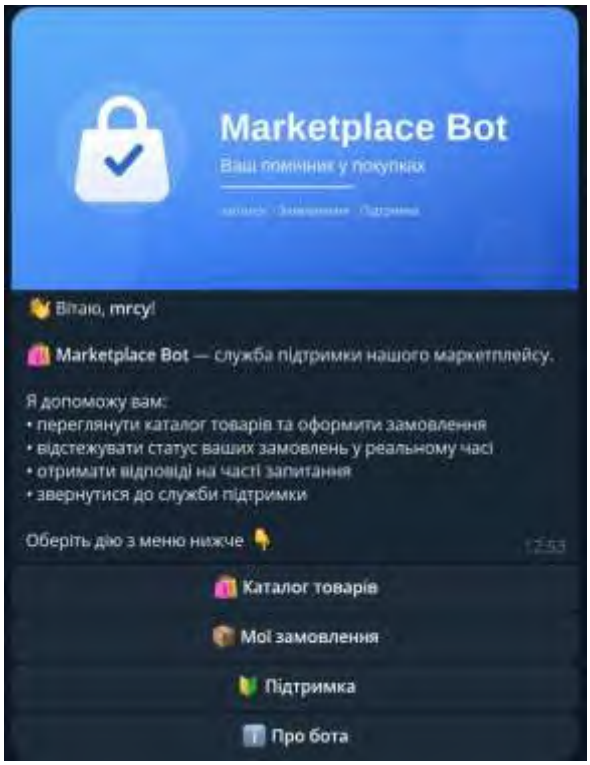

Рисунок 3.3 – Головне меню Telegram-боту

Рисунок 3.4 – Картка товару з зображенням та кнопкою замовлення

Модуль handler.py є найбільшим за обсягом і реалізує маршрутизацію всіх подій – команд (/start, /help) та натискань inline-кнопок. Обробник callback-подій побудовано як єдину функцію handle_callback з розгалуженням за значенням поля data кнопки. Для сценаріїв введення кількості товару та тексту звернення використано просту машину скінченних станів (FSM) на основі словника user_states, що зберігає поточний стан кожного користувача між повідомленнями [31]. При отриманні текстового повідомлення система перевіряє наявність активного стану і або обробляє введені дані, або повертає підказку щодо використання кнопок меню. Картку замовлення з поточним статусом і таймлайном етапів наведено на рисунку 3.5.

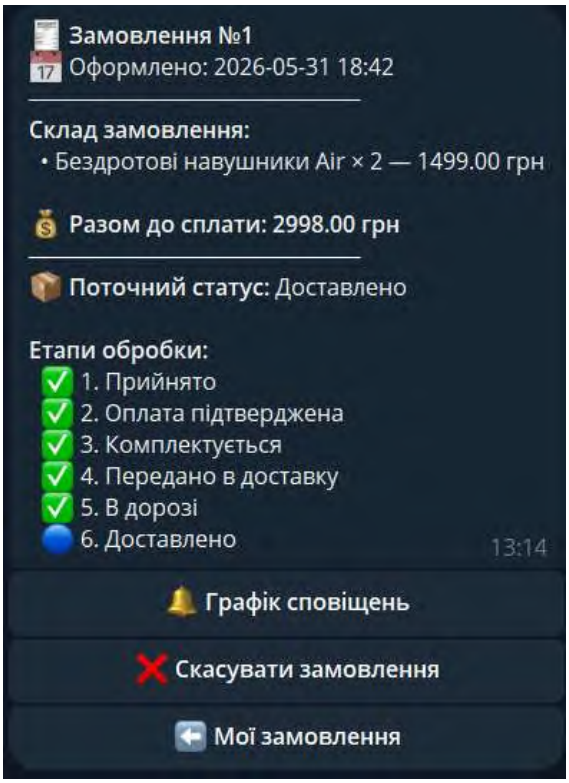

Рисунок 3.5 – Картка замовлення з поточним статусом та таймлайном етапів

Модуль scheduler.py реалізує фонове завдання перевірки сповіщень засобами бібліотеки APScheduler [20]. Планувальник запускається в режимі BackgroundScheduler з часовою зоною Europe/Kyiv, що забезпечує коректне відображення часу для українських користувачів. Завдання виконується із заданим інтервалом, отримує з бази всі невідправлені сповіщення, час яких настав, надсилає повідомлення через Telegram Bot API і позначає їх як відправлені. При помилці надсилання з кодом chat not found сповіщення також позначається як відправлене, щоб уникнути нескінченних повторних спроб для недоступних чатів. Приклад автоматичного сповіщення наведено на рисунку 3.6.

Рисунок 3.6 – Автоматичне сповіщення про зміну статусу замовлення

Загальний обсяг програмного коду системи складає близько 700 рядків, розподілених між сімома модулями. Код супроводжується коментарями українською мовою, що відповідає вимогам до документування програмного забезпечення в навчальних проєктах [29]. Реалізована система повністю відповідає функціональним вимогам категорій Must have та Should have, визначеним у підрозділі 2.1, і готова до етапу тестування.

3.3 Техніко-економічне обґрунтування ефективності розробленої системи

Техніко-економічне обґрунтування ефективності виконується відповідно до методичних рекомендацій щодо оцінки IT-проєктів і передбачає визначення витрат на розробку, оцінку ефектів від впровадження та розрахунок показників економічної ефективності [24]. Оскільки система розроблялася виключно з використанням безкоштовного програмного забезпечення з відкритим кодом – Python, бібліотек pyTelegramBotAPI, APScheduler, mysql-connector-python та пакету OpenServer, – витрати на придбання програмного забезпечення відсутні. Технічне забезпечення також не потребувало окремих витрат, оскільки розробка виконувалася на наявному обладнанні. Таким чином, основну витратну статтю становлять витрати на оплату праці розробника.

Вартість розробки визначається за формулою собівартості інформаційно-обчислювальних послуг відповідно до методики [24]:

$$C = \Sigma\,(Tj \cdot Gj) \cdot (1 + R), \tag{3.1}$$

де Tj – трудомісткість відповідного етапу в годинах;

Gj – тариф оплати праці за одну годину;

R – норматив прибутковості.

Прийнявши погодинну ставку розробника рівня junior в Україні у 2026 році на рівні 150 грн/год та норматив прибутковості R = 0,15, отримаємо:

$$C = 96 \cdot 150 \cdot (1 + 0{,}15) = 14400 \cdot 1{,}15 = 16560 \text{ грн.}$$

Щорічні витрати на утримання системи є мінімальними, оскільки вона функціонує на локальному обладнанні без хмарного хостингу, і в базовій моделі приймаються як несуттєві.

Трудомісткість розробки оцінюється на основі фактичного обсягу виконаних робіт, розподілених за етапами. Розподіл трудовитрат наведено в таблиці 3.3.

Таблиця 3.3 – Трудомісткість розробки інформаційної системи за етапами

Етап розробки	Зміст робіт	Трудомісткість, год
Аналіз предметної області	огляд аналогів, визначення вимог	12
Проєктування	UML-діаграми, ER-діаграма, архітектура	16
Розробка БД	створення схеми, наповнення даними	10
Розробка програмного коду	модулі db, handler, tracker, scheduler та ін.	30
Налагодження та тестування	виявлення та усунення помилок	10
Документування	оформлення пояснювальної записки	18
Разом		96

Оцінка ефектів від впровадження виконується за трьома категоріями. Прямий ефект полягає в розвантаженні служби підтримки маркетплейсу: бот автоматично опрацьовує типові звернення (статус замовлення, доставка, оплата, повернення), вивільняючи робочий час операторів. Якісний ефект полягає в цілодобовій доступності та миттєвості відповіді, що підвищує задоволеність користувачів. Стратегічний ефект – масштабованість: база даних дозволяє додавати нові товари та категорії без зміни архітектури.

Для розрахунку показника рентабельності інвестицій ROI використовується формула [24]:

$$ROI = (AP - INV) / INV \cdot 100\%, \quad (3.2)$$

де *AP* – середньорічний умовний ефект (економія робочого часу операторів служби підтримки) від використання системи у грошовому вираженні;

INV – початкові інвестиції у розробку системи.

Якщо прийняти, що бот автоматично опрацьовує близько 480 типових звернень на місяць із середнім часом опрацювання оператором 3 хвилини, то щомісячна економія робочого часу складає 1440 хвилин (24 години), а за рік – 288 годин. За вартості години роботи оператора підтримки 130 грн річний умовний ефект становить AP = 288 · 130 = 37440 грн. Тоді:

$$ROI = (37440 - 16560) / 16560 \cdot 100\% \approx 126\%.$$

Отримане значення ROI суттєво перевищує мінімальний поріг ефективності, що підтверджує економічну доцільність розробки. Таким чином, техніко-економічне обґрунтування підтвердило, що розроблена система є економічно ефективною: загальні витрати на розробку склали 16560 грн за умови нульових витрат на програмне забезпечення та інфраструктуру, а розрахунковий показник рентабельності інвестицій становить близько 126%, що свідчить про високу доцільність впровадження системи для підтримки маркетплейсу.

Було виконано повний практичний цикл реалізації чат-боту підтримки маркетплейсу. Розгорнуто реляційну базу даних MySQL із дев’яти таблиць у середовищі OpenServer та наповнено її структурованими тестовими даними – чотирма категоріями, дев’ятьма товарами, шістьма етапами обробки замовлення та базою поширених запитань. Реалізовано програмний код системи у складі семи модулів загальним обсягом близько 700 рядків із дотриманням принципів модульності, єдиної відповідальності та безпеки. Систему перевірено в роботі: планувальник коректно надсилає сповіщення за розкладом, модуль трекера визначає поточний статус замовлення та формує таймлайн етапів, навігація через кнопкове меню функціонує без помилок. Виконано техніко-економічне обґрунтування, яке підтвердило економічну доцільність розробки з показником рентабельності інвестицій близько 126% за повної відсутності витрат на програмне забезпечення та інфраструктуру.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне практичне завдання – розроблено чат-бот для підтримки маркетплейсу, що автоматизує обслуговування користувачів безпосередньо в месенджері Telegram. Поставлену мету досягнуто, а всі визначені у вступі завдання виконано в повному обсязі.

У першому розділі проаналізовано предметну область електронної комерції та маркетплейсів, визначено роль і проблеми служби підтримки, що обґрунтувало доцільність автоматизації обслуговування. Проведено огляд існуючих рішень, за результатами якого встановлено, що жоден з аналогів не поєднує в єдиній системі перегляд каталогу, оформлення та відстеження замовлень, автоматичні сповіщення й базу типових запитань. Обґрунтовано вибір технологічного стеку: мова Python, бібліотека pyTelegramBotAPI, система керування базами даних MySQL у середовищі OpenServer та планувальник APScheduler.

У другому розділі виконано проєктування системи: сформовано й пріоритизовано функціональні та нефункціональні вимоги, побудовано комплект UML-діаграм, що формалізують поведінку системи, спроєктовано нормалізовану до третьої нормальної форми реляційну базу даних із дев’яти таблиць та визначено трирівневу архітектуру програмного забезпечення із застосуванням патерну Repository як єдиної точки доступу до даних.

У третьому розділі реалізовано спроєктовану систему: розгорнуто базу даних MySQL та наповнено її тестовими даними, розроблено програмний код у складі семи модулів загальним обсягом близько 700 рядків із дотриманням принципів модульності та безпеки. Перевірено коректність роботи системи: каталог, оформлення та відстеження замовлень, автоматичні сповіщення й база типових запитань функціонують відповідно до вимог. Ключовим алгоритмом системи є автоматичне визначення поточного статусу замовлення на основі накопиченої тривалості етапів обробки.

Виконане техніко-економічне обґрунтування підтвердило економічну доцільність розробки: загальні витрати на створення системи склали 16560 грн за умови нульових витрат на програмне забезпечення та інфраструктуру, а розрахунковий показник рентабельності інвестицій перевищує 126 %.

Практична значущість роботи полягає в тому, що розроблений чат-бот забезпечує цілодобове автоматизоване обслуговування користувачів маркетплейсу та розвантажує службу підтримки від типових звернень. Перспективами подальшого розвитку системи є інтеграція з реальними платіжними сервісами, додавання модуля розпізнавання природної мови для опрацювання довільних запитів та створення вебпанелі адміністрування для керування каталогом і замовленнями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laudon K. C., Traver C. G. E-commerce 2023: Business, Technology, Society. 17th ed. Harlow: Pearson, 2023. 928 p.

2. Retail e-commerce sales worldwide from 2014 to 2027. Statista. URL: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/ (дата звернення: 20.05.2026).

3. Аналітика месенджера Telegram. TGStat. URL: https://tgstat.com (дата звернення: 20.05.2026).

4. Закон України «Про електронну комерцію» від 03.09.2015 № 675-VIII. URL: https://zakon.rada.gov.ua/laws/show/675-19 (дата звернення: 20.05.2026).

5. Telegram Messenger. Telegram FZ-LLC. URL: https://telegram.org (дата звернення: 20.05.2026).

6. Bots: An introduction for developers. Telegram. URL: https://core.telegram.org/bots (дата звернення: 20.05.2026).

7. Telegram Bot API. Official documentation. URL: https://core.telegram.org/bots/api (дата звернення: 20.05.2026).

8. Build a Telegram Bot with Python. Real Python. URL: https://realpython.com/python-telegram-bot/ (дата звернення: 20.05.2026).

9. Python 3 documentation. Python Software Foundation. URL: https://docs.python.org/3/ (дата звернення: 20.05.2026).

10. Lutz M. Learning Python. 5th ed. Sebastopol: O'Reilly Media, 2013. 1648 p.

11. Slatkin B. Effective Python: 90 Specific Ways to Write Better Python. 2nd ed. Boston: Addison-Wesley, 2019. 480 p.

12. pyTelegramBotAPI documentation. URL: https://pytba.readthedocs.io/en/latest/ (дата звернення: 20.05.2026).

13. Advanced Python Scheduler (APScheduler) documentation. URL: https://apscheduler.readthedocs.io/en/3.x/ (дата звернення: 20.05.2026).

14. MySQL Connector/Python Developer Guide. Oracle. URL: https://dev.mysql.com/doc/connector-python/en/ (дата звернення: 20.05.2026).

15. MySQL 8.0 Reference Manual. Oracle. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 20.05.2026).

16. Beaulieu A. Learning SQL. 3rd ed. Sebastopol: O'Reilly Media, 2020. 380 p.

17. Date C. J. An Introduction to Database Systems. 8th ed. Boston: Addison-Wesley, 2003. 1024 p.

18. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Harlow: Pearson, 2015. 1440 p.

19. Codd E. F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 1970. Vol. 13, No. 6. P. 377–387.

20. phpMyAdmin documentation. URL: https://www.phpmyadmin.net/docs/ (дата звернення: 20.05.2026).

21. Open Server Panel: документація. URL: https://ospanel.io (дата звернення: 20.05.2026).

22. OWASP SQL Injection Prevention Cheat Sheet. OWASP Foundation. URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 20.05.2026).

23. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. 3rd ed. Upper Saddle River: Prentice Hall, 2004. 736 p.

24. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston: Addison-Wesley, 2003. 208 p.

25. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Boston: Addison-Wesley, 2005. 496 p.

26. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002. 560 p.

27. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1994. 416 p.

28. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston: Addison-Wesley, 2003. 560 p.

29. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Upper Saddle River: Prentice Hall, 2017. 432 p.

30. Sommerville I. Software Engineering. 10th ed. Harlow: Pearson, 2015. 816 p.

31. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. Geneva: ISO, 2018.

32. Wiegers K., Beatty J. Software Requirements. 3rd ed. Redmond: Microsoft Press, 2013. 672 p.

33. Clegg D., Barker R. Case Method Fast-Track: A RAD Approach. Boston: Addison-Wesley, 1994. 256 p.

34. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 8th ed. New York: McGraw-Hill, 2014. 976 p.

35. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ: ДП «УкрНДНЦ», 2016. 16 с.