НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «__Програмне забезпечення чат-бота для автоматизації підтримки клієнтів у сфері електронної комерції __»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми к.т.н., доцент	_____ (підпис)	**Ганна ВАЙГАНГ**
Керівник бакалаврської кваліфікаційної роботи науковий ступінь та вчене звання	_____ (підпис)	**Дмитро Ніколаєнко**
Виконав	_____ (підпис)	**Олександр Максімцев**

КИЇВ – 2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Максімцеву Олександру Сергійовичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення чат-бота для автоматизації підтримки клієнтів у сфері електронної комерції»

затверджена наказом від «19» Грудня 2025 р. № 3196 «С»
Термін подання завершеної роботи на кафедру «22» Травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи: надання інформації про питання клієнта до магазину, статуси замовлень та доставки._
Перелік питань, що підлягають дослідженню:

- Аналіз проблемної області
- Моделювання предметної області
- Проєктування програмної системи
- Впровадження та експлуатація системи

Перелік графічних документів (за необхідності).

Дата видачі завдання «19» Грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Дмитро Ніколаєнко**

Завдання взяв до виконання _____ (підпис) **Олександр Максімцев**

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційних технологій та зростання цифрової економіки суттєво трансформували сферу електронної комерції. Сьогодні інтернет-магазини активно замінюють традиційні канали збуту, надаючи клієнтам можливість шукати, порівнювати та купувати товари дистанційно. Водночас сучасні користувачі очікують швидкого та зручного спілкування з інтернет-магазинами, особливо щодо інформації про товар, статусу замовлення, питань оплати та технічної підтримки. Однак обробка великої кількості запитів клієнтів вручну вимагає значного часу та ресурсів. Тому розробка програмного забезпечення для чат-ботів для автоматизації підтримки клієнтів в електронній комерції є актуальним та своєчасним завданням.

Актуальність дослідження визначається зростаючим попитом на автоматизовані системи підтримки клієнтів, необхідністю оптимізації процесів комунікації між клієнтами та інтернет-магазинами, а також важливістю підвищення швидкості та якості обслуговування за допомогою сучасних цифрових технологій. Впровадження програмного забезпечення для чат-ботів дозволяє бізнесу зменшити навантаження на менеджерів служби підтримки, мінімізувати людські помилки, забезпечити швидке реагування клієнтів та забезпечити безперервне спілкування в режимі реального часу.

Метою цієї бакалаврської роботи є покращення способів комунікації клієнта та продавця у сфері електронної комерції шляхом розробки програмного забезпечення чат-бота для автоматизації комунікації між клієнтами та платформою електронної комерції. Розроблена система повинна дозволяти користувачам отримувати інформацію про товари, перевіряти статус замовлення, отримувати відповіді на поширені запитання та спілкуватися зі службою підтримки клієнтів через зручний інтерфейс чат-бота.

Об'єктом дослідження є процес підтримки клієнтів у системах електронної комерції з використанням сучасних інформаційних технологій.

Предметом дослідження є методи, моделі та програмні засоби, що використовуються для проєктування та впровадження програмного забезпечення чат-бота для автоматизованої підтримки клієнтів на основі технологій PHP, Telegram Bot API та MySQL.

Для досягнення визначеної мети були поставлені такі завдання:

1. проаналізувати поточний стан систем підтримки клієнтів в електронній комерції та визначити їх основні особливості та обмеження;
2. визначити функціональні та нефункціональні вимоги до програмного забезпечення чат-бота;
3. спроєктувати архітектуру системи та структуру бази даних;
4. розробити програмне забезпечення чат-бота з використанням PHP та Telegram Bot API;
5. реалізувати ключові функціональні можливості, такі як взаємодія з користувачем, перегляд каталогу товарів, отримання інформації про замовлення та автоматизована підтримка клієнтів;

Тестування програмного застосунку було проведено шляхом функціонального та інтеграційного тестування для перевірки його відповідності визначеним вимогам.

Структура роботи: бакалаврська робота складається зі вступу, чотирьох розділів, висновків, списку використаної літератури та додатків. Загальний обсяг роботи становить 69 сторінок. Список використаної літератури містить 36 джерел. Робота також містить 2 додатки.

У першому розділі аналізуються існуючі системи підтримки клієнтів та сучасні технології чат-ботів, що використовуються в електронній комерції. У другому розділі описано системні вимоги, функціональне моделювання, проєктування баз даних. У третьому розділі представлено реалізацію програмного забезпечення чат-бота, включаючи опис основних модулів та функцій. Четвертий розділ охоплює тестування програмного забезпечення, оцінку результатів та аналіз ефективності розробленої системи.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Швидкий розвиток електронної комерції суттєво змінив спосіб взаємодії компаній з клієнтами. Інтернет-магазини працюють у висококонкурентному середовищі, де якість та швидкість підтримки клієнтів відіграють вирішальну роль у залученні та утриманні користувачів. Клієнти очікують миттєвих відповідей на свої запитання щодо наявності товарів, умов доставки, способів оплати, статусу замовлення та політики повернення. Однак традиційні методи підтримки клієнтів, засновані на людських операторах, часто не здатні швидко та ефективно обробити велику кількість запитів.

Однією з головних проблем сучасних систем електронної комерції є перевантаження служб підтримки клієнтів. Зі збільшенням кількості користувачів та онлайн-замовлень стрімко зростає також кількість вхідних запитів клієнтів. Менеджери служби підтримки повинні відповідати на повторювані запитання, такі як відстеження замовлень, деталі товару, умови доставки та проблеми з оплатою. Це призводить до збільшення робочого навантаження, уповільнення часу реагування та зниження задоволеності клієнтів. Крім того, ручна обробка запитів збільшує ризик людських помилок та невідповідностей у відповідях [1].

Ще однією важливою проблемою є відсутність безперервної підтримки клієнтів. Більшість інтернет-магазинів надають підтримку лише в робочий час, що обмежує можливості клієнтів отримувати допомогу в будь-який час. У сучасному цифровому середовищі користувачі очікують цілодобового зв'язку та швидкого зворотного зв'язку. Якщо клієнти не отримають своєчасної допомоги, вони можуть залишити веб-сайт та обрати інший інтернет-магазин, який надає кращий сервіс. Тому забезпечення безперервної та ефективної комунікації з клієнтами є одним із ключових завдань для платформ електронної комерції.

Сучасні технології дозволяють вирішувати ці проблеми за допомогою використання автоматизованих систем підтримки клієнтів, зокрема чат-ботів. Чат-боти дозволяють компаніям автоматизувати спілкування з користувачами, надавати швидкі відповіді на поширені запитання та значно зменшувати навантаження на персонал служби підтримки клієнтів. Інтегруючи програмне забезпечення чат-ботів у платформу електронної комерції, стає можливим покращити швидкість комунікації, підвищити задоволеність клієнтів та оптимізувати бізнес-процеси, пов'язані з взаємодією з клієнтами.

Незважаючи на існуючі рішення у сфері розробки чат-ботів, багатьом інтернет-магазинам досі бракує ефективних та гнучких систем чат-ботів, які можна легко інтегрувати в їхні платформи. Деякі існуючі рішення є занадто складними, дорогими або їх важко адаптувати до конкретних потреб малого та середнього бізнесу. Тому розробка програмного забезпечення чат-ботів для автоматизованої підтримки клієнтів в електронній комерції залишається актуальним та важливим завданням.

Аналіз проблемної області показує, що впровадження програмного забезпечення чат-ботів може значно підвищити ефективність систем підтримки клієнтів, зменшити навантаження на менеджерів служби підтримки, забезпечити безперервний зв'язок з користувачами та підвищити загальну ефективність платформ електронної комерції. Це підтверджує актуальність та практичну важливість теми дослідження та обґрунтовує необхідність розробки програмного забезпечення чат-ботів для автоматизованої підтримки клієнтів в електронній комерції [2].

Предметом цієї бакалаврської роботи є автоматизована підтримка клієнтів у системах електронної комерції з використанням технологій чат-ботів. Платформи електронної комерції широко використовуються для продажу товарів і послуг через Інтернет і включають набір програмних інструментів, що забезпечують управління каталогом продукції, обробку замовлень, обробку платежів, організацію доставки та взаємодію з клієнтами. Одним з найважливіших компонентів будь-якої сучасної платформи

електронної комерції є ефективна система підтримки клієнтів, яка дозволяє користувачам швидко отримувати необхідну інформацію та вирішувати проблеми, пов'язані з їхніми замовленнями.

Підтримка клієнтів в електронній комерції передбачає спілкування між користувачем та інтернет-магазином на різних етапах взаємодії. Ці етапи включають пошук товарів, отримання інформації про товар, розміщення замовлення, здійснення платежу, відстеження доставки та вирішення проблем, пов'язаних з поверненням або технічними проблемами. Традиційно підтримка клієнтів надається операторами-людьми, які відповідають на запити клієнтів через електронну пошту, онлайн-чат або соціальні мережі. Однак, зі збільшенням кількості користувачів, ручна обробка запитів стає неефективною та трудомісткою.

Предметною галуззю також є використання програмного забезпечення чат-ботів як інструменту для автоматизації спілкування між клієнтами та інтернет-магазинами. Чат-бот – це програмний додаток, який імітує спілкування з користувачем за допомогою текстових повідомлень та надає автоматичні відповіді на основі попередньо визначених сценаріїв або логіки. У контексті електронної комерції чат-боти можуть виконувати різні функції, такі як надання інформації про продукти, допомога користувачам у навігації по каталогу, відповіді на поширені запитання, перевірка стану замовлення та допомога клієнтам у вирішенні поширених проблем.

Важливою частиною предметної області є взаємодія між чат-ботом та інтерфейсом користувача. Чат-бот спілкується з користувачами через платформи обміну повідомленнями, такі як Telegram, або веб-інтерфейси чату. Це дозволяє користувачам взаємодіяти з системою простим та зручним способом без необхідності встановлення додаткового програмного забезпечення. Чат-бот обробляє введені користувачем дані, аналізує запит та генерує відповідну відповідь на основі попередньо визначених команд та сценаріїв. Крім того, чат-бот може зберігати та обробляти основні дані користувача, такі як номери замовлень, запити та історія взаємодії.

Предметна область також охоплює використання сучасних веб-технологій для розробки та впровадження програмного забезпечення для чат-ботів. Ці технології включають мови програмування, такі як PHP, системи управління базами даних, такі як MySQL, та інтерфейси прикладного програмування, такі як Telegram Bot API. Поєднання цих технологій дозволяє створити ефективну та масштабовану систему автоматизованої підтримки клієнтів в електронній комерції.

Детальний опис класів та атрибутів предметної області наведено у таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	**Атрибут**	**Опис**
Повідомлення	Текст повідомлення	Повідомлення, яке надсилає користувач або чат-бот
	Дата повідомлення	Дата та час відправлення повідомлення
	Тип повідомлення	Повідомлення від користувача або від чат-бота
Запит до підтримки	Тема запиту	Тема звернення користувача (оплата, доставка, замовлення тощо)
	Опис проблеми	Текст звернення користувача
	Дата створення	Дата створення звернення
	Статус запиту	Поточний стан звернення (новий, обробляється, вирішено)
Чат-бот	Назва бота	Назва чат-бота
	Опис	Короткий опис функціоналу чат-бота
	Мова спілкування	Мова, якою бот взаємодіє з користувачем
	Тип підтримки	Тип підтримки (автоматична, частково автоматична)

1.2 Огляд існуючих рішень

З розвитком електронної комерції значно зросла потреба у швидкій та ефективній взаємодії між онлайн-магазинами та клієнтами. Одним із найбільш популярних інструментів автоматизації клієнтської підтримки стали чат-боти. На сучасному ринку представлено велику кількість програмних рішень, які дозволяють автоматизувати відповіді на запити користувачів, обробляти повідомлення та покращувати якість обслуговування клієнтів. Найбільш поширеними є платформи для створення чат-ботів, CRM-системи з вбудованими чат-ботами та спеціалізовані рішення для електронної комерції.

Одним із найбільш відомих інструментів для створення чат-ботів на сучасному ринку є платформа Chatfuel. Це програмне рішення призначене для автоматизації комунікації між бізнесом і клієнтами за допомогою популярних месенджерів, зокрема Facebook Messenger, Instagram та WhatsApp. Платформа дозволяє створювати чат-ботів без необхідності програмування, що робить її доступною для малого та середнього бізнесу [3].

Рис. 1.1 Чат-бот “Chatfuel”

Основною функцією Chatfuel є автоматизація спілкування з клієнтами. За допомогою цієї платформи підприємства можуть налаштовувати автоматичні відповіді на запити користувачів, створювати сценарії взаємодії, збирати контактні дані клієнтів та забезпечувати підтримку в режимі 24/7. Платформа також підтримує інтеграцію з різними сервісами, такими як CRM-системи, платіжні системи та інструменти аналітики, що дозволяє використовувати її як комплексне рішення для автоматизації клієнтської підтримки.

Chatfuel використовує зручний графічний інтерфейс для створення чат-ботів. Користувач може створювати діалоги за допомогою спеціального конструктора сценаріїв (Flow Builder), який дозволяє задавати логіку спілкування без написання програмного коду. Крім того, платформа надає готові шаблони чат-ботів для різних сфер діяльності, включаючи електронну комерцію, маркетинг та клієнтську підтримку.

Важливою перевагою Chatfuel є можливість автоматизації типових бізнес-процесів. Чат-бот, створений за допомогою цієї платформи, може відповідати на часті запитання користувачів, надавати інформацію про товари, приймати замовлення, надсилати повідомлення про статус замовлення та збирати відгуки клієнтів. Це дозволяє значно зменшити навантаження на службу підтримки та підвищити швидкість обробки запитів [3].

Незважаючи на значну кількість переваг, платформа має і певні обмеження. Наприклад, деякі функції доступні лише у платних тарифних планах, а можливості гнучкого налаштування можуть бути обмеженими для складних систем підтримки. Крім того, платформа орієнтована переважно на месенджери соціальних мереж і може бути менш зручною для створення повністю індивідуальних чат-ботів.

Отже, Chatfuel є популярним і ефективним інструментом для створення чат-ботів, який широко використовується у сфері електронної комерції для автоматизації клієнтської підтримки. Проте обмеження гнучкості та залежність від сторонніх платформ підтверджують доцільність розробки

власного програмного рішення, що є однією з основних цілей даної бакалаврської роботи.

Розглянемо інше програмне рішення на ринку.

ManyChat – це популярна платформа для створення чат-ботів, яка дозволяє автоматизувати взаємодію між бізнесом та клієнтами у різних месенджерах та соціальних платформах, таких як Facebook Messenger, Instagram, WhatsApp та SMS. Основною метою ManyChat є надання інструментів для створення чат-ботів без необхідності програмування, що робить її доступною для широкого кола користувачів, включаючи фахівців з маркетингу, власників малого бізнесу та підприємців у сфері електронної комерції [4].

Рис. 1.2 Чат-бот “ManyChat”

Головним компонентом ManyChat є візуальний конструктор діалогів (Flow Builder), за допомогою якого користувачі можуть створювати сценарії взаємодії з клієнтами. Інтерфейс побудований на блоках, які представляють

собою окремі кроки діалогу, відповіді бота, кнопки навігації та події. Такий підхід дозволяє без написання коду створювати досить складні логіки спілкування, автоматичні відповіді на запити, опитування, тригерні повідомлення та інші функції.

Однією з ключових можливостей ManyChat є автоматизація маркетингових і сервісних процесів. Платформа підтримує сегментацію аудиторії, створення персоналізованих повідомлень, планування розсилок та інтеграцію з CRM-системами, сервісами аналітики та зовнішніми базами даних. Це дозволяє не лише автоматизувати відповіді на типові клієнтські запити, але й використовувати чат-ботів як інструмент для підвищення залучення користувачів, лояльності та повернення клієнтів у онлайн-магазинах.

Перевагою платформи ManyChat є її універсальність і гнучкість. Вона підтримує роботу з різними каналами комунікації, що дозволяє підприємствам централізовано управляти повідомленнями в різних месенджерах. Крім того, ManyChat надає можливість створювати автоматичні фрази, відповіді на тригерні слова та складні сценарії обробки запитів, що робить її ефективним інструментом для автоматизації підтримки клієнтів.

Водночас ManyChat має певні обмеження, зокрема щодо складності реалізації глибокої кастомізації без використання програмних API або додаткових інструментів. Деякі функції, такі як інтеграція з зовнішніми системами або використання складних умов обробки даних, можуть вимагати додаткових технічних знань. Також частина розширених можливостей доступна лише у платних тарифах, що може бути обмеженням для малого бізнесу з обмеженим бюджетом [4].

Таким чином, ManyChat є потужною платформою для створення чат-ботів з широким спектром функцій, що дозволяє автоматизувати комунікацію з клієнтами в електронній комерції. Незважаючи на деякі обмеження, її використання сприяє підвищенню ефективності обслуговування клієнтів, скороченню часу відповіді на запити та оптимізації бізнес-процесів.

Огляд ManyChat свідчить про існуючі можливості та підтверджує актуальність розробки власного програмного рішення для автоматизованої підтримки клієнтів, яке може більш гнучко адаптуватись під конкретні вимоги інтернет-магазину.

1.3 Постановка завдання

Організація системи автоматизованої підтримки клієнтів в інтернет-магазинах базується на необхідності забезпечення оперативної, ефективної та персоналізованої взаємодії між продавцями та покупцями. Сучасні користувачі очікують миттєвої відповіді на запити щодо товарів, замовлень, оплат та доставки, а також зручного каналу для отримання допомоги без очікування у черзі. Водночас власникам електронної комерції потрібні інструменти для ефективного управління зверненнями клієнтів та автоматизації рутинних процесів обслуговування.

Основним завданням системи є автоматизація обробки запитів користувачів та забезпечення їх структурування, що включає реєстрацію запитів, визначення теми питання, перенаправлення складних запитів на менеджерів та формування історії комунікації. Ключові параметри, що підлягають моніторингу та обробці, включають:

- тематика запитів користувачів, що дозволяє системі класифікувати повідомлення та надавати релевантні відповіді;
- статус обробки звернень, включаючи автоматичне підтвердження отримання запиту та інформування клієнта про подальші дії;
- історія взаємодії з клієнтом, що дозволяє аналізувати ефективність підтримки та покращувати якість обслуговування;
- час відповіді на запит, який є критичним показником задоволеності клієнтів;

- рівень персоналізації відповідей, що включає врахування попередніх взаємодій та уподобань користувача [6].

Програмний додаток підтримує централізовану базу даних, яка зберігає інформацію про користувачів, їх запити, категорії запитів та відповіді чат-бота. Узгодженість та цілісність даних є критично важливими, оскільки система повинна запобігати дублюванню відповідей, некоректній маршрутизації запитів та втраті інформації.

Процес обробки запиту організований за чітко визначеним життєвим циклом. Коли користувач надсилає повідомлення, система автоматично визначає його тематику та перевіряє наявність готових сценаріїв відповіді. У разі наявності стандартного сценарію чат-бот надає автоматичну відповідь, а якщо запит потребує додаткового втручання менеджера, створюється внутрішнє завдання для персоналу. Всі дії фіксуються у базі даних, що забезпечує прозорість та можливість подальшого аналізу.

Інтегрований механізм комунікації дозволяє обмінюватися повідомленнями безпосередньо через платформу, уточнювати деталі запитів та вирішувати потенційні проблеми у режимі реального часу. Всі повідомлення пов’язані з обліковими записами користувачів і конкретними запитами, що підвищує ефективність обслуговування та дозволяє зберігати історію взаємодій для аналітики.

Інтерфейс користувача розроблений для підтримки інтуїтивної навігації по основних функціональних модулях системи. Окремі розділи передбачені для відправки запитів, перегляду статусу обробки, взаємодії з чат-ботом та звернення до живого менеджера. Такий поділ зменшує складність використання та забезпечує зручність для клієнта.

З архітектурної точки зору, система побудована за багаторівневою моделлю, що розділяє компоненти презентації, бізнес-логіки та управління даними. Бекенд відповідає за обробку запитів, маршрутизацію та виконання бізнес-правил, тоді як фронтенд забезпечує доступний інтерфейс для взаємодії користувача.

Завдяки такій структурованій організації, чат-бот для автоматизованої підтримки клієнтів підвищує ефективність обслуговування, скорочує час реагування на запити, забезпечує прозору комунікацію та дозволяє підприємствам оптимізувати процеси взаємодії з покупцями в електронній комерції.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги визначають основні дії та поведінку системи, які дозволяють реалізувати цільову функціональність чат-бота для автоматизованої підтримки клієнтів у електронній комерції.

Функціональні вимоги:

- реєстрація та автентифікація користувачів для забезпечення персоналізованого доступу;
- інтерактивна навігація по меню та категоріях товарів для спрощення пошуку необхідної продукції;
- прийом та обробка запитів клієнтів у режимі реального часу, включаючи проблеми з замовленням, питання щодо оплати, доставки та повернення товарів;
- автоматична генерація відповідей на типові питання на основі заздалегідь визначених сценаріїв [7];
- ведення журналу взаємодій користувачів з ботом для подальшого аналізу та поліпшення якості підтримки;
- можливість передачі складних або нестандартних запитів на ручну обробку менеджером;
- інтеграція з базою даних товарів та замовлень для надання актуальної інформації клієнтам;
- відправка повідомлень про статус замовлень, оплат та доставку, включаючи сповіщення про зміни;

- забезпечення можливості зворотного зв’язку, оцінки роботи бота та залишення коментарів користувачами.

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на її продуктивність, надійність та зручність використання.

Нефункціональні вимоги:

- швидкість обробки запитів користувачів та відповіді бота, що забезпечує комфортну взаємодію в реальному часі;
- доступність системи 24/7 для підтримки клієнтів у будь-який час;
- надійність та безпека передачі даних, включаючи захист персональної інформації користувачів;
- масштабованість системи для підтримки збільшення кількості користувачів та запитів;
- простота та інтуїтивність інтерфейсу для забезпечення швидкого навчання та використання ботом;
- можливість інтеграції з зовнішніми сервісами електронної комерції та платіжними системами;
- збереження та відновлення даних у випадку технічних збоїв;
- підтримка багатомовності та локалізації для роботи з різними сегментами користувачів.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача чат-бота повинен забезпечувати зручну та інтуїтивно зрозумілу взаємодію, дозволяючи користувачам швидко знаходити потрібну інформацію та виконувати необхідні дії. Меню та кнопки повинні мати чітку структуру, що забезпечує швидкий доступ до основних функцій бота, таких як перегляд каталогу товарів, управління замовленнями та звернення до служби підтримки. Важливо, щоб процес навігації між категоріями та функціональними розділами був логічним та послідовним, а

відображення актуальної інформації про товари, замовлення та статуси запитів – зрозумілим та наочним.

Користувач повинен мати можливість надсилати повідомлення у режимі реального часу, а також швидко обирати типові запити за допомогою інтерактивних кнопок, таких як «Проблема з замовленням» або «Питання про оплату». При цьому система повинна підтримувати введення текстових повідомлень для вирішення нестандартних ситуацій та уточнень. Всі дії користувача повинні супроводжуватися повідомленнями про успішне виконання або виниклі помилки, що забезпечує прозорість та контроль за взаємодією.

Інтерфейс має бути адаптивним і коректно відображатися на різних пристроях, включаючи смартфони та планшети. Особлива увага приділяється простоті, приємному візуальному оформленню, що відповідає бренду електронної комерції, а також доступності елементів інтерфейсу для людей з обмеженими можливостями, зокрема через підтримку кольорів, розміру шрифтів і навігаційних елементів. Всі ці вимоги спрямовані на підвищення ефективності взаємодії користувача з чат-ботом та забезпечення позитивного досвіду використання системи [9].

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Unified Modeling Language (UML) є універсальною мовою моделювання, що дозволяє розробникам, аналітикам та дизайнерам систем формалізувати та візуалізувати складні програмні рішення. UML забезпечує стандартизований спосіб опису програмного забезпечення, який сприяє узгодженню бачення між усіма учасниками процесу розробки та дозволяє документувати систему у формі, зрозумілій як технічним спеціалістам, так і замовникам. Основна перевага UML полягає у його здатності представляти систему одночасно у структурному та поведінковому аспектах, що дозволяє забезпечити комплексне розуміння архітектури та функціональних можливостей програмного продукту.

UML включає різноманітні типи діаграм, які можна умовно розділити на дві групи: структурні та поведінкові. Структурні діаграми відображають компоненти системи, їх властивості та взаємозв’язки, а також формалізують модель даних. До них належать діаграми класів, об’єктів, компонентів, пакетів та розгортання. Поведінкові діаграми призначені для демонстрації логіки роботи системи, послідовності дій та взаємодії між об’єктами або акторами. До таких діаграм належать діаграми прецедентів, діяльності, послідовностей, станів та часові діаграми. Використання цих інструментів дозволяє не лише описати систему, але й перевірити її коректність, передбачити можливі конфлікти та забезпечити узгодженість бізнес-процесів.

У контексті розробки чат-бота для автоматизованої підтримки клієнтів в електронній комерції UML є незамінним інструментом для формалізації та моделювання ключових функціональних та нефункціональних вимог. Діаграми прецедентів дозволяють чітко визначити ролі користувачів, таких як клієнт, менеджер служби підтримки та адміністратор системи, і проілюструвати всі сценарії взаємодії з системою, включно з реєстрацією

запитів, обробкою повідомлень, перевіркою статусу замовлень та отриманням зворотного зв’язку. Це дозволяє створити логічну карту системи, де кожен сценарій взаємодії між користувачем і чат-ботом має своє місце та структуровану послідовність дій [10].

Діаграми класів демонструють структуру даних та взаємозв’язки між основними сутностями системи, такими як замовлення, товари, користувачі, повідомлення та сеанси чат-бота. Використання таких діаграм дозволяє забезпечити цілісність даних, запобігти дублюванню інформації та конфліктам при паралельній обробці запитів. Діаграми послідовностей наочно показують порядок виконання процесів, від моменту надходження повідомлення від користувача до формування відповідної відповіді чат-ботом, включаючи обробку логіки бізнес-правил, перевірку даних у базі та взаємодію з адміністративною панеллю.

Крім того, UML дозволяє моделювати різні сценарії помилок та винятків, що є важливим для забезпечення стабільності роботи чат-бота. Діаграми діяльності допомагають аналізувати бізнес-процеси системи, оптимізувати логіку обробки запитів та визначити критичні точки, де може знадобитися втручання адміністратора. Діаграми станів корисні для відстеження змін статусів замовлень, повідомлень та сесій користувачів, що забезпечує прозорість роботи системи та підвищує рівень контролю над процесами обслуговування клієнтів.

Використання UML у процесі розробки чат-бота забезпечує численні переваги: полегшує комунікацію між членами команди, дозволяє формалізувати вимоги на ранніх етапах проєктування, підвищує зрозумілість системи для тестувальників та спрощує подальший супровід програмного забезпечення. У результаті, застосування UML сприяє створенню надійного, масштабованого та ефективного чат-бота, який здатен автоматизувати підтримку клієнтів в електронній комерції, забезпечуючи одночасно високу якість обслуговування та оптимізацію внутрішніх бізнес-процесів компанії.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма прецедентів (Use-Case Diagram) є одним із ключових інструментів UML, який використовується для моделювання функціональної поведінки системи та взаємодії користувачів із програмним забезпеченням. Основна мета діаграми прецедентів полягає у наданні наочної моделі того, що система повинна виконувати, та які взаємодії відбуваються між системою та її користувачами. Вона дозволяє чітко визначити ролі учасників процесу (акторів) та прецеденти, тобто конкретні дії або сценарії, які реалізовує система [11].

Для чат-бота, призначеного для автоматизованої підтримки клієнтів в електронній комерції, діаграма прецедентів є надзвичайно корисною, оскільки допомагає формалізувати всі можливі сценарії взаємодії користувачів із системою. Серед основних акторів можна виділити клієнта, який отримує послуги та задає запити, менеджера служби підтримки, який обробляє звернення та контролює виконання запитів, а також адміністратора системи, який управляє структурою чат-бота та контролює його налаштування.

Прецеденти для клієнта включають такі дії, як надсилання запиту у чат-бот, перегляд каталогу товарів, оформлення замовлення, перевірка статусу замовлення та отримання відповіді на поставлене питання. Кожен із цих прецедентів демонструє окремий функціональний аспект системи, забезпечуючи повноту уявлення про можливості чат-бота. Для менеджера служби підтримки прецеденти включають обробку вхідних запитів, комунікацію з клієнтами, надання рекомендацій та перевірку виконання замовлень. Адміністратор відповідає за налаштування правил обробки повідомлень, управління базою даних користувачів та контроль над доступом до системи.

Діаграма прецедентів забезпечує візуальну структуру взаємодій, де актори з’єднуються лініями з відповідними прецедентами, що дозволяє наочно бачити, хто виконує яку дію та які дії підтримує система. Такий підхід сприяє

ідентифікації всіх критичних функціональних елементів, виявленню потенційних вузьких місць у бізнес-процесах та плануванню подальшого розвитку системи. Крім того, діаграма прецедентів дозволяє визначити пріоритети для реалізації функціоналу, що важливо для поетапної розробки чат-бота та тестування його модулів.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

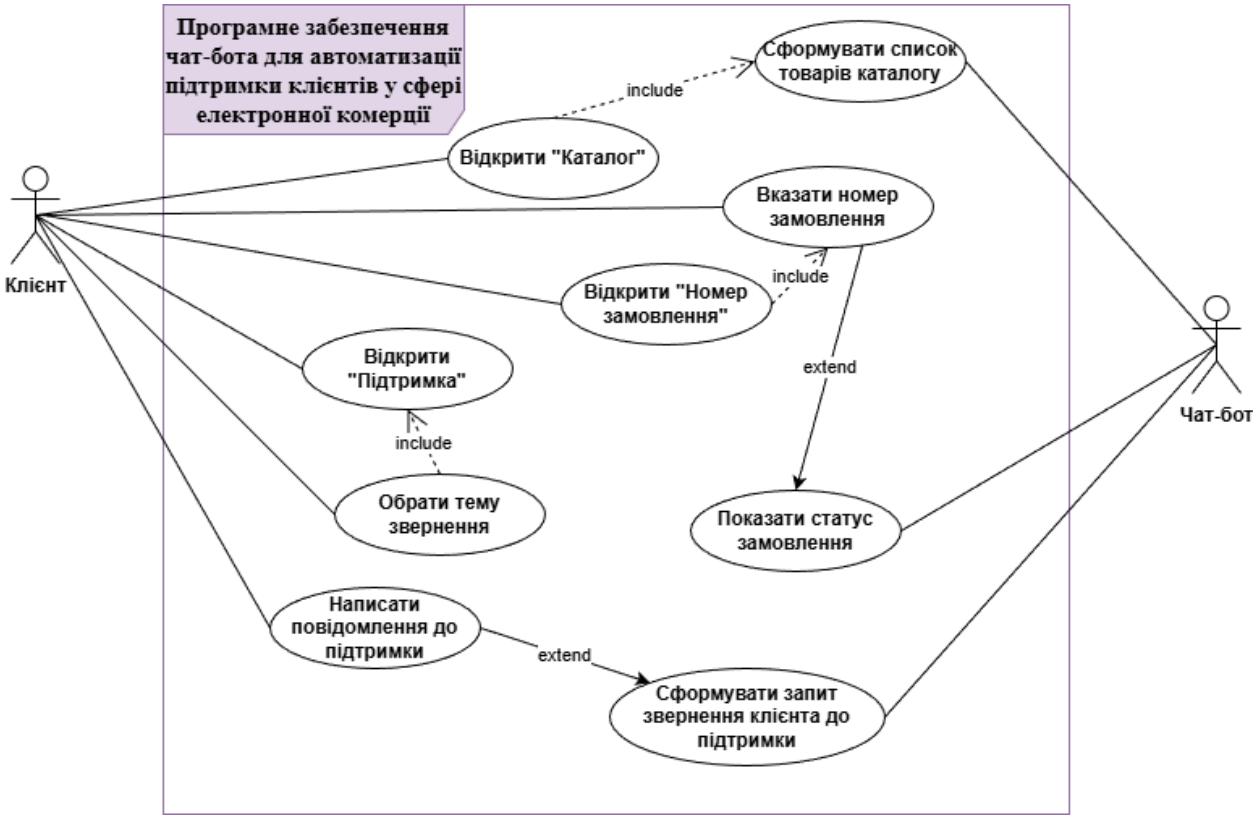

Рис. 2.1 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Клієнт”;
- “Чат-бот”.

Актор «Клієнт» включає такі прецеденти:

- відкрити "Каталог";
- відкрити "Номер замовлення";
- вказати номер замовлення;
- відкрити "Підтримка";

- обрати тему звернення;
- написати повідомлення до підтримки.

Актор «Чат-бот» включає такі прецеденти:

- сформувати список товарів каталогу;
- показати статус замовлення;
- сформувати запит звернення клієнта до підтримки.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Варіант використання: Відкрити номер замовлення

Актор: Клієнт

Опис: Приклад використання «Відкрити номер замовлення» описує процес, за допомогою якого клієнт може отримати деталі конкретного замовлення в системі електронної комерції через чат-бот. Клієнт вводить номер замовлення, а система перевіряє його дійсність та відображає інформацію про замовлення, включаючи дату, суму, статус та список товарів. Цей процес дозволяє клієнту швидко отримати актуальні дані про своє замовлення без необхідності звертатися до менеджера служби підтримки.

Основний потік:

1. клієнт відкриває чат-бот та обирає опцію «Мої замовлення»;
2. система запитує у клієнта номер замовлення;
3. клієнт вводить номер замовлення;
4. система перевіряє правильність введеного номера та наявність замовлення у базі даних;
5. система відображає деталі замовлення: дату створення, суму, статус, список товарів та кількість одиниць;
6. клієнт переглядає отриману інформацію та може задати додаткові питання через чат-бот;
7. чат-бот пропонує повернутися до головного меню або переглянути інше замовлення.

Альтернативні потоки:

1. якщо введений номер замовлення некоректний або не існує в базі даних, система відображає повідомлення типу «Замовлення не знайдено, перевірте номер» та запрошує ввести номер повторно;
2. якщо клієнт не вводить номер замовлення протягом певного часу, система нагадує про необхідність введення або пропонує повернутися до головного меню;
3. якщо база даних тимчасово недоступна, система відображає повідомлення про технічні проблеми та пропонує повторити спробу пізніше;
4. клієнт може скасувати введення номера замовлення в будь-який момент, натиснувши «Назад до меню», після чого система повертає його до головного меню.

Випадок використання «Відкрити номер замовлення» дозволяє клієнтам швидко отримати детальну інформацію про конкретне замовлення та забезпечує прозору комунікацію із системою без необхідності безпосередньої взаємодії з менеджером. Це підвищує ефективність обслуговування та зручність користування платформою електронної комерції.

Розглянемо ще один сценарій використання.

Варіант використання: Надіслати повідомлення у підтримку

Актор: Клієнт

Опис: Приклад використання «Надіслати повідомлення у підтримку» описує процес, за допомогою якого клієнт може зв’язатися зі службою підтримки через чат-бот для отримання допомоги або консультації щодо замовлень, оплат, доставки чи повернення товарів. Клієнт обирає тему звернення, вводить текст повідомлення, а система передає його менеджеру служби підтримки та підтверджує клієнту отримання запиту. Цей процес дозволяє клієнту отримати швидку та структуровану підтримку без необхідності телефонного дзвінка або письмового листування.

Основний потік:

1. клієнт відкриває чат-бот та обирає опцію «Підтримка»;

2. система відображає доступні категорії звернень, наприклад: «Проблема з замовленням», «Питання про оплату», «Доставка та трекінг», «Повернення товару»;
3. клієнт обирає відповідну тему звернення;
4. система запитує у клієнта текст повідомлення з детальним описом проблеми або питання;
5. клієнт вводить повідомлення у чат;
6. система приймає повідомлення, зберігає його у базі даних та передає менеджеру служби підтримки;
7. система відправляє клієнту підтвердження отримання повідомлення та орієнтовний час відповіді;
8. клієнт може надіслати додаткові уточнення або повернутися до головного меню.

Альтернативні потоки:

1. якщо клієнт не вводить повідомлення протягом певного часу, система нагадує про необхідність введення тексту або пропонує повернутися до головного меню;
2. якщо система тимчасово недоступна для надсилання повідомлень, відображається повідомлення про технічні проблеми та пропонується повторити спробу пізніше;
3. клієнт може змінити вибрану тему звернення до відправлення повідомлення, після чого система оновлює категорію звернення;
4. після надсилання повідомлення клієнт може отримати статус обробки запиту через чат-бот, наприклад «Ваше повідомлення отримано, очікуйте відповіді».

Використання сценарію «Надіслати повідомлення у підтримку» дозволяє забезпечити прозору комунікацію між клієнтом та службою підтримки, підвищує ефективність обслуговування та мінімізує час очікування відповіді. Система забезпечує структуровану обробку звернень, що дозволяє менеджерам швидко реагувати на запити та зберігати історію взаємодії для

подальшого аналізу.

2.2.2 Діаграма послідовності. Діаграма послідовності є важливим інструментом моделювання, який відображає порядок обміну повідомленнями між об’єктами або компонентами системи під час виконання конкретного сценарію. Вона показує, як учасники взаємодіють із системою в часовому вимірі, демонструючи послідовність викликів методів або передачі даних [12].

У контексті чат-бота для автоматизації підтримки клієнтів діаграма послідовності дозволяє деталізувати процеси, такі як обробка повідомлень користувача, визначення теми звернення, передача повідомлення менеджеру та отримання відповіді. Вона допомагає розробникам та аналітикам чітко уявити логіку роботи системи, взаємозв’язки компонентів і порядок обробки запитів у реальному часі.

На діаграмі послідовності зазвичай виділяють такі ключові компоненти:

актор – користувач або зовнішня система, яка ініціює запит, наприклад клієнт чат-бота;

об’єкти системи – компоненти або модулі, які взаємодіють для обробки запиту, наприклад: модуль обробки повідомлень, база даних звернень, модуль повідомлень менеджера;

повідомлення – дії або методи, які передаються між об’єктами у певній послідовності;

лінії життя – вертикальні лінії, що показують час існування об’єкта під час сценарію.

Завдяки діаграмам послідовності можна наочно показати, як чат-бот приймає повідомлення користувача, перевіряє його тему, надсилає дані до бази та інформує менеджера, а потім формує відповідь для користувача. Це дозволяє оцінити ефективність процесу обробки звернень та виявити можливі вузькі місця в логіці взаємодії системи.

Використання діаграм послідовності підвищує якість проєктування системи, оскільки вони допомагають формалізувати процеси, передбачити

сценарії обробки даних та забезпечують зрозумілу документацію для подальшої реалізації програмного забезпечення.

Діаграма послідовності, зображена на рис. 2.2, включає наступні об'єкти:

- “Клієнт”;
- “Чат-бот”;
- “Замовлення”;
- “Підтримка”.

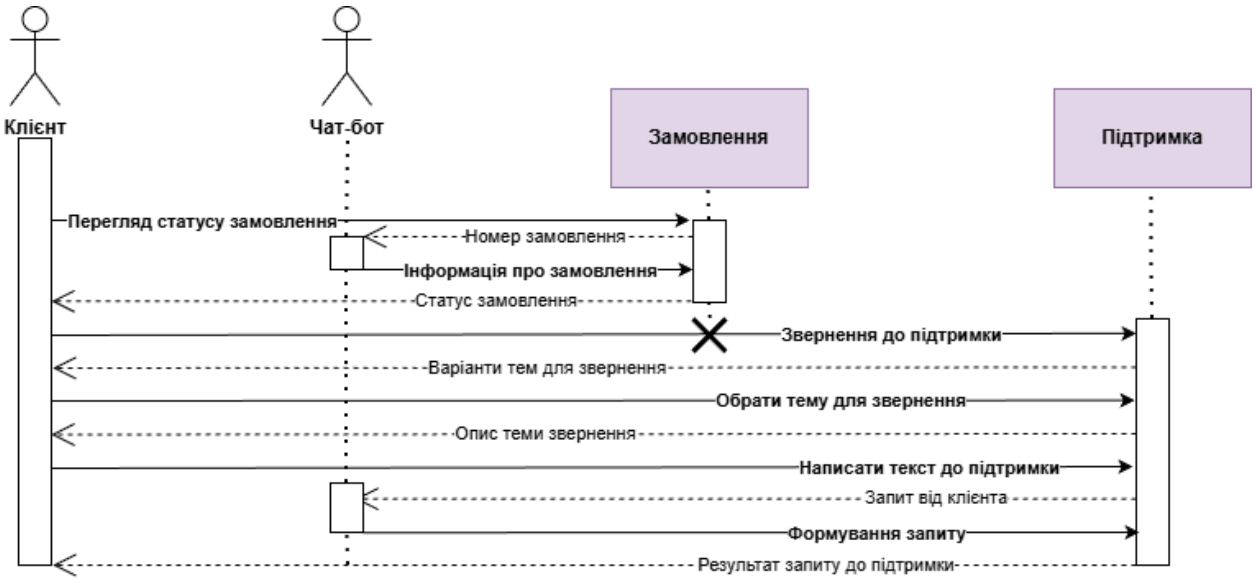

Рис. 2.2 Діаграма послідовності

Дана діаграма послідовності описує процес взаємодії клієнта з чат-ботом у двох основних сценаріях: перевірка статусу замовлення та звернення до служби підтримки. Вона побудована так, щоб показати послідовність повідомлень між учасниками – клієнтом, чат-ботом, системою замовлень і модулем підтримки.

Спочатку клієнт звертається до чат-бота, щоб дізнатися статус свого замовлення. Чат-бот просить номер замовлення, клієнт його вводить, і система замовлень повертає інформацію. Потім чат-бот показує клієнту результат – статус замовлення. Це демонструє автоматизовану частину роботи, де бот виступає посередником між користувачем і базою даних замовлень.

Далі йде сценарій звернення до підтримки. Клієнт ініціює запит, чат-бот пропонує варіанти тем для звернення, клієнт обирає потрібну і додає опис. Потім він пише повідомлення, яке чат-бот формує у структурований запит і передає до служби підтримки. Після цього підтримка повертає результат, який чат-бот показує клієнту. Тут видно, як бот допомагає правильно оформити звернення і передати його далі.

Таким чином, діаграма показує логіку роботи системи: чат-бот бере на себе роль координатора, який спрощує взаємодію клієнта з внутрішніми сервісами та забезпечує швидкий доступ до інформації чи допомоги.

2.2.3 Діаграма активності. Діаграма активності використовується для моделювання потоків управління та логіки виконання процесів у системі. Вона дозволяє відобразити, які дії виконуються користувачем і системою, у якому порядку та як вони взаємопов’язані. На відміну від діаграми послідовності, яка фокусується на взаємодії між об’єктами у часі, діаграма діяльності показує логіку обробки процесів, включаючи альтернативні та паралельні шляхи виконання завдань.

У контексті чат-бота для підтримки клієнтів діаграма діяльності дозволяє наочно показати процес взаємодії користувача із системою, починаючи від моменту надсилання повідомлення і завершуючи отриманням відповіді від менеджера. Вона демонструє перевірку умов і вибір альтернативних потоків, наприклад у випадку порожнього повідомлення або некоректного запиту, а також паралельні процеси, такі як одночасне збереження звернення у базі даних та надсилання повідомлення менеджеру.

Діаграма діяльності включає початкову точку, яка позначає старт процесу, дії користувача і системи, потоки управління, умовні вузли для розгалуження процесу, а також кінцеву точку, що відображає завершення обробки звернення. Використання діаграми діяльності дозволяє спростити аналіз бізнес-процесів, виявити потенційні помилки у логіці роботи системи, оптимізувати порядок обробки запитів та забезпечити ефективну взаємодію між користувачем і чат-ботом.

Діаграма діяльності також відображає альтернативні шляхи виконання процесу. Наприклад, якщо користувач надіслав порожнє повідомлення або текст, який система не може обробити, чат-бот генерує уточнююче повідомлення та пропонує повторити спробу. Паралельно може здійснюватися логування запиту для статистики, відстеження часу відповіді та пріоритизації звернень за категоріями. Таке моделювання дозволяє не лише візуалізувати послідовність дій, але й проаналізувати ефективність системи, забезпечити своєчасну реакцію менеджерів і покращити загальний користувацький досвід при взаємодії з чат-ботом.

Розроблена діаграма активності представлена на рис. 2.3

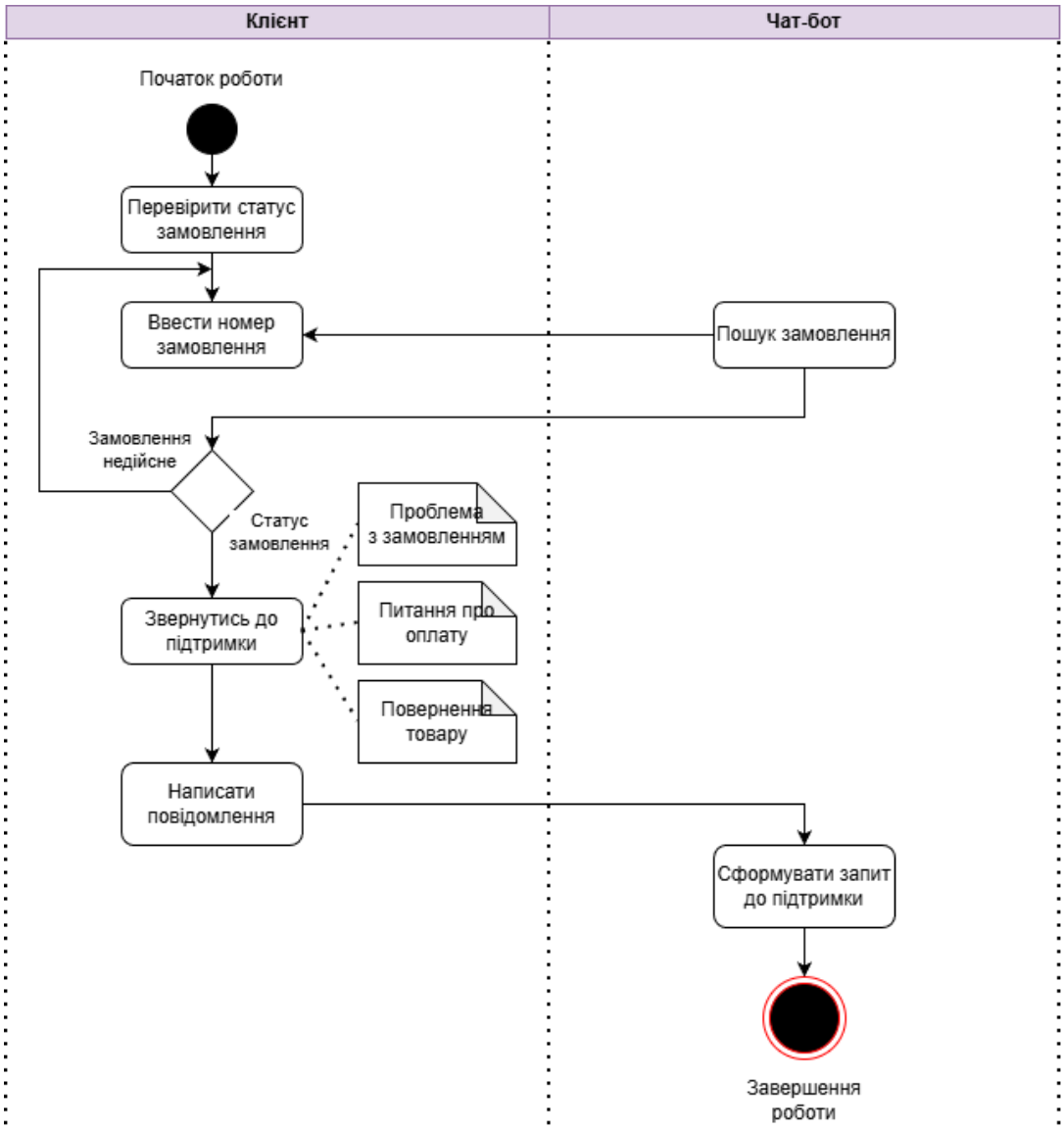

Рис. 2.3 Діаграма активності

Ця діаграма активності показує логіку роботи чат-бота під час взаємодії

з клієнтом. Вона побудована у вигляді двох паралельних потоків – дії клієнта та відповідні реакції чат-бота. Це дозволяє наочно побачити, як система автоматизує процеси підтримки.

Спочатку клієнт починає роботу з чат-ботом і може обрати перевірку статусу замовлення. Він вводить номер замовлення, після чого чат-бот здійснює пошук у системі. Якщо номер недійсний, клієнт отримує повідомлення про помилку. Якщо ж замовлення знайдено, бот показує його статус. Таким чином реалізується базовий сценарій перевірки.

Далі можливий інший варіант – клієнт стикається з проблемою і звертається до підтримки. Тут він може уточнити характер проблеми: питання про оплату, повернення товару чи інша проблема із замовленням. Після цього клієнт формулює повідомлення, а чат-бот перетворює його на структурований запит і надсилає до служби підтримки. Завершення роботи відбувається після того, як клієнт отримує відповідь.

Діаграма демонструє два ключові процеси: автоматизовану перевірку замовлення та ескалацію проблеми до підтримки. Вона добре показує, що чат-бот бере на себе рутинні завдання, а у складних випадках допомагає клієнту правильно оформити звернення.

2.3 Абстракції предметної області

Абстракції предметної області в UML є засобом концептуалізації та формалізації основних елементів системи, що дозволяє розробникам, аналітикам та замовникам мати спільне розуміння структури і функцій системи. Вони відображають ключові сутності, процеси та взаємозв’язки в межах предметної області на високому рівні узагальнення. Завдяки абстракціям стає можливим виділити головні об’єкти та їх властивості, ролі, які вони виконують, а також логіку їх взаємодії, не зосереджуючись на конкретних технічних деталях реалізації.

Важливою характеристикою абстракцій є їх здатність спрощувати

складні системи, зменшуючи кількість деталей до тих, які є критичними для розуміння сутності предметної області. Це дозволяє ефективно визначати межі системи, розділяти її на логічні компоненти та встановлювати взаємозв’язки між цими компонентами. Абстракції забезпечують основу для подальшого проєктування системи, включаючи розробку UML-діаграм класів, діаграм прецедентів, діаграм активностей та послідовності. Вони допомагають ідентифікувати основні функції системи, розробити сценарії її використання та виявити потенційні проблеми на ранньому етапі проєктування [13].

Крім того, абстракції слугують інструментом узгодження вимог між різними учасниками проєктної діяльності. Вони дозволяють аналітикам формулювати вимоги в термінах предметної області, а розробникам – зрозуміти, які компоненти необхідно реалізувати та як вони повинні взаємодіяти. У цьому сенсі абстракції виступають як мова спільного розуміння, що забезпечує ефективну комунікацію між технічними та нетехнічними учасниками проєкту.

Особливу роль абстракції відіграють у процесі моделювання поведінки системи. Вони дозволяють формалізувати сценарії використання, визначати динаміку взаємодії між об’єктами та описувати логіку виконання бізнес-процесів. Це, у свою чергу, полегшує побудову діаграм послідовності та активностей, а також сприяє виявленню вузьких місць та оптимізації потоків інформації. Використання абстракцій робить систему більш зрозумілою, структурованою та зручною для подальшого розвитку і масштабування.

Абстракції предметної області в UML є фундаментальним елементом процесу проєктування систем. Вони забезпечують можливість системного аналізу, моделювання та документування предметної області, створюючи міцну основу для розробки програмного забезпечення. Без їх використання складні системи ризикують залишатися неструктурованими, що ускладнює їх розробку, тестування та підтримку. Абстракції дозволяють зосередитися на суті проблеми, ігноруючи другорядні деталі, що робить процес проєктування більш ефективним та передбачуваним.

Рис. 2.4 Абстракції предметної області

У нас виділено дві ключові абстракції – Замовлення та Підтримка. Вони описують основні сутності, з якими працює чат-бот у сфері електронної комерції.

Абстракція «Замовлення» відповідає за всі дані, пов’язані з покупкою клієнта. Вона має властивості, що описують товар: назву, опис, ціну, кількість, а також дату замовлення. Її обов’язки полягають у тому, щоб дозволяти перевіряти статус замовлення, здійснювати пошук за номером і змінювати статус (наприклад, «оплачено», «доставлено», «скасовано»). Це сутність, яка забезпечує прозорість і контроль над процесом покупки.

Абстракція «Підтримка» описує взаємодію клієнта зі службою допомоги. Вона містить властивості, що характеризують звернення: ПІБ клієнта, тему, дату, текст повідомлення та статус запиту. Її обов’язки включають вибір теми звернення, написання повідомлення, зміну статусу звернення та формування запиту до служби підтримки. Таким чином, ця абстракція моделює процес комунікації між клієнтом і компанією, коли виникають проблеми чи додаткові питання.

Разом ці дві абстракції утворюють основу предметної області: одна відповідає за управління замовленнями, інша – за обробку звернень клієнтів. Вони тісно пов’язані між собою, адже проблеми із замовленням часто стають причиною звернення до підтримки.

2.4 Діаграма класів

Діаграма класів (Class Diagram) у UML є одним із ключових інструментів для моделювання статичної структури програмної системи. Вона відображає основні об’єкти предметної області, їх властивості (атрибути), поведінку (методи) та взаємозв’язки між об’єктами. Діаграма класів дозволяє розробникам отримати чітке уявлення про те, як дані організовані в системі та як компоненти взаємодіють один з одним, що є критично важливим для правильного проєктування програмного забезпечення.

Класи на діаграмі представлені у вигляді прямокутників, розділених на три частини: назва класу, атрибути та методи. Зв’язки між класами можуть мати різний характер, включаючи асоціації, наслідування, реалізацію інтерфейсів, агрегацію та композицію. Ці зв’язки дозволяють відобразити, як об’єкти співпрацюють і як дані передаються між компонентами системи. Завдяки цьому можна спроєктувати логічну структуру програми до початку її реалізації [15].

Діаграма класів відіграє важливу роль у процесі розробки, оскільки вона забезпечує основу для розробки бази даних, формування інтерфейсів, а також для організації коду у вигляді модулів і пакетів. Вона допомагає розробникам уникнути дублювання даних, забезпечує узгодженість структур і сприяє підвищенню повторного використання компонентів. Крім того, діаграма класів слугує документацією, яка полегшує підтримку та масштабування системи.

Створення діаграми класів починається з ідентифікації основних об’єктів та їхніх властивостей у предметній області. Потім визначаються методи класів, які описують їх поведінку, і встановлюються зв’язки між класами. При цьому особлива увага приділяється точності та логічній коректності моделі, оскільки від цього залежить подальша реалізація системи. Діаграма класів може доповнюватися різними примітками та обмеженнями, що уточнюють правила взаємодії об’єктів, типи атрибутів та допустимі

значення.

Використання діаграм класів дозволяє розробникам краще планувати архітектуру системи, забезпечує чітке розуміння структури та взаємозв'язків компонентів, а також підвищує ефективність комунікації між учасниками проєкту. Це робить її незамінним інструментом у процесі розробки програмного забезпечення будь-якого рівня складності, дозволяючи перетворити концептуальні ідеї на конкретні об'єкти та модулі для подальшої реалізації.

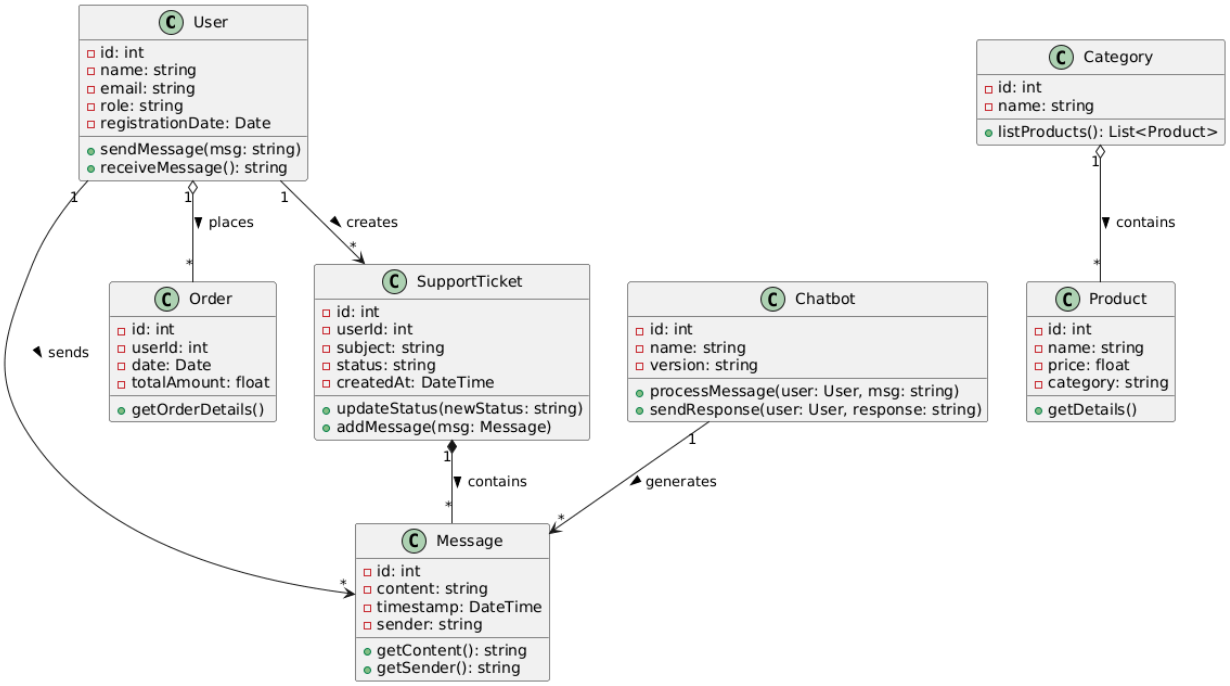

Рис. 2.5 Діаграма класів

Ця діаграма класів моделює систему електронної комерції з інтегрованим чат-ботом для підтримки клієнтів. Вона охоплює ключові сутності, які взаємодіють між собою: користувачі, замовлення, звернення до підтримки, повідомлення, товари, категорії та сам чат-бот.

У центрі взаємодії знаходиться клас User, який має атрибути, що описують особу користувача – ідентифікатор, ім'я, email, роль і дату реєстрації. Користувач може надсилати повідомлення, створювати звернення до підтримки та оформлювати замовлення. Зв'язки показують, що один користувач може мати багато замовлень, багато звернень і надсилати багато повідомлень.

Клас Order містить інформацію про замовлення: ідентифікатор, дату, загальну суму та прив’язку до користувача. Він має метод для отримання деталей замовлення. Це дозволяє системі зберігати історію покупок і пов’язувати її з конкретним користувачем.

Клас SupportTicket моделює звернення до служби підтримки. Він містить тему, статус, дату створення та прив’язку до користувача. Звернення може містити багато повідомлень, що реалізовано через зв’язок із класом Message. Повідомлення мають текст, час створення та інформацію про відправника. Вони можуть бути створені як користувачем, так і чат-ботом.

Клас Chatbot має методи для обробки повідомлень і надсилання відповідей. Він виступає як автоматизований посередник між користувачем і системою, генеруючи повідомлення, які потрапляють у звернення або напряму до користувача.

Окремо представлена товарна частина системи: Category і Product. Категорія містить багато товарів, а кожен товар має назву, ціну та прив’язку до категорії. Це дозволяє структурувати каталог і забезпечити зручний пошук.

Діаграма демонструє добре продуману модель, де кожен клас має чіткі обов’язки, а зв’язки між ними логічно відображають реальні процеси: покупки, звернення, обробку повідомлень і взаємодію з чат-ботом. Система виглядає масштабованою та готовою до розширення – наприклад, додавання ролей, статусів замовлень чи інтеграції з платіжними сервісами.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

ERwin – це професійний інструмент для моделювання даних, який широко використовується для створення, управління та документування структур баз даних. Його головне призначення – допомогти розробникам, аналітикам і архітекторам даних візуалізувати структури баз даних, зв’язки між таблицями, атрибути та обмеження, що забезпечує більш точне і структуроване проєктування баз даних.

Інструмент підтримує різні типи моделювання: концептуальне, логічне та фізичне. Концептуальне моделювання дозволяє відобразити загальні сутності та їх взаємозв’язки без прив’язки до конкретної СУБД. Логічне моделювання деталізує атрибути, типи даних та ключі, але ще без фізичної реалізації. Фізичне моделювання дозволяє генерувати SQL-коди для конкретної системи управління базами даних, включаючи індекси, обмеження, типи даних та таблиці.

ERwin має інтуїтивно зрозумілий графічний інтерфейс із підтримкою drag-and-drop для створення сутностей, зв’язків та атрибутів. Він також включає механізми автоматичної перевірки цілісності даних, генерації документації та порівняння моделей, що дозволяє ефективно контролювати зміни та синхронізацію між різними версіями бази даних.

Інструмент активно використовується в корпоративному середовищі для проєктування великих і складних баз даних, оптимізації структури зберігання даних і поліпшення комунікації між командами розробки та аналітики. Завдяки своїм можливостям ERwin забезпечує високий рівень точності, узгодженості та прозорості в процесі проєктування баз даних.

ER-діаграма (Entity-Relationship Diagram, діаграма «сутність-зв’язок») – це графічне подання структури бази даних, яке описує сутності, їх атрибути та зв’язки між ними. Вона є ключовим інструментом концептуального та

логічного моделювання даних, дозволяючи розробникам, аналітикам і архітекторам даних чітко бачити, як різні елементи системи взаємодіють один з одним [14].

Основними компонентами ER-діаграми є сутності, які представляють об’єкти або концепції предметної області; атрибути, що описують характеристики цих сутностей; та зв’язки, які відображають логічні взаємозв’язки між сутностями. Кожен зв’язок може мати кардинальність і участь, що визначає, скільки екземплярів однієї сутності можуть бути пов’язані з екземплярами іншої сутності.

ER-діаграми дозволяють створювати наочну модель бази даних, яка спрощує розуміння структури інформації, забезпечує цілісність даних і слугує основою для подальшого фізичного проєктування в конкретній СУБД. Використання таких діаграм допомагає виявити дублювання даних, неправильні зв’язки та потенційні проблеми на ранніх етапах розробки, що значно знижує ризик помилок при створенні або модифікації бази даних.

ER-діаграми також широко застосовуються для документування існуючих систем, навчання нових членів команди та забезпечення ефективної комунікації між розробниками, бізнес-аналітиками та керівництвом проєкту. Вони створюють основу для стандартизації даних і підтримки масштабованості та модульності системи.

Логічна модель системи представлена на рисунку 3.1.

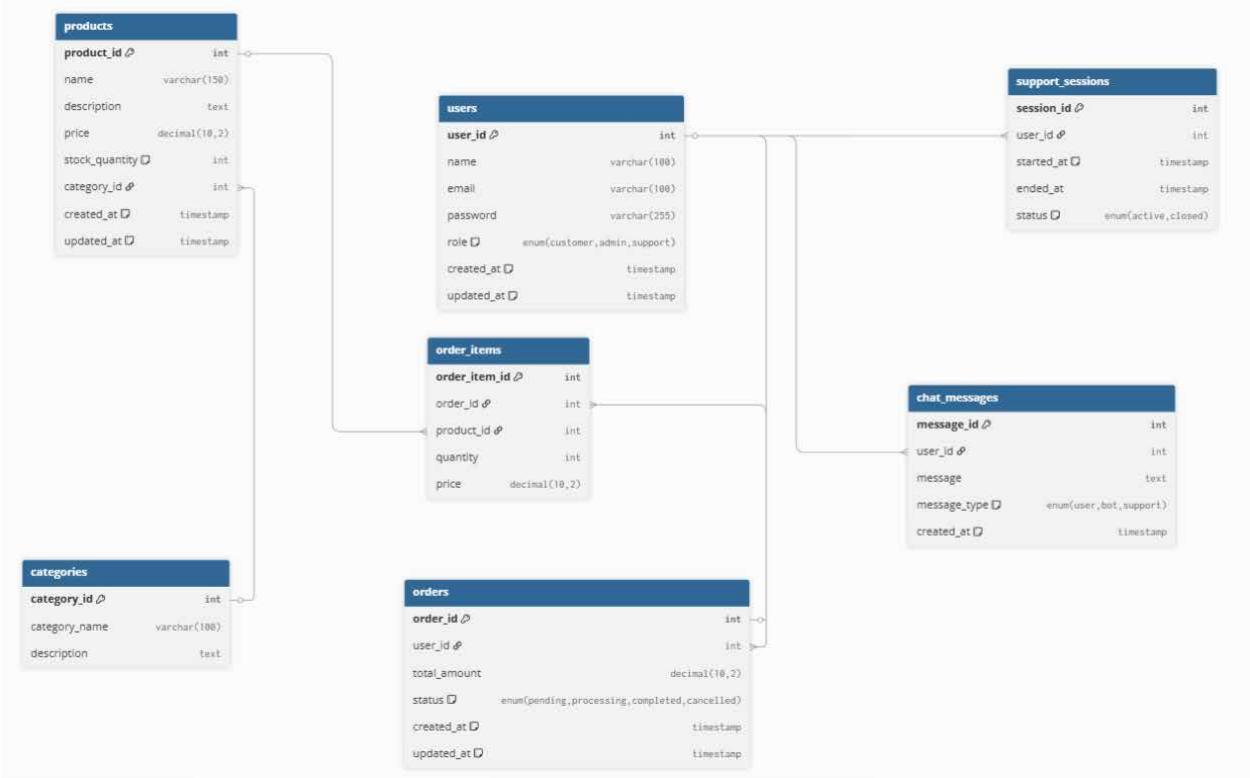

Рис. 3.1 ER-діаграма “chatbot_db”

Ця ER-діаграма моделює структуру бази даних для системи електронної комерції з функціями підтримки клієнтів через чат. Вона охоплює ключові сутності, їхні атрибути та зв’язки між таблицями, що дозволяє зрозуміти, як дані організовані та взаємодіють у межах платформи.

У центрі знаходиться таблиця users, яка зберігає інформацію про користувачів: ім’я, email, пароль, роль (клієнт, адміністратор або оператор підтримки), а також дати створення та оновлення. Кожен користувач може створювати замовлення, брати участь у сесіях підтримки та надсилати повідомлення в чаті.

Таблиця orders пов’язана з користувачами через поле user_id. Вона містить загальну суму замовлення, статус (наприклад, «в обробці» або «скасовано») та часові мітки. Деталі кожного замовлення зберігаються в таблиці order_items, яка включає кількість товару, його ціну та зв’язок із конкретним продуктом і замовленням.

Товари описані в таблиці products, де зберігаються назва, опис, ціна, кількість на складі, категорія та часові мітки. Кожен товар належить до певної

категорії, що реалізовано через зовнішній ключ category_id, який веде до таблиці categories. Категорії мають назву та опис, що дозволяє структурувати каталог товарів.

Система підтримки реалізована через таблиці support_sessions та chat_messages. Сесія підтримки містить інформацію про користувача, час початку та завершення, а також статус (активна чи закрита). Повідомлення в чаті зберігаються окремо в таблиці chat_messages, де фіксується текст, тип повідомлення (від користувача, бота або оператора) та час створення.

Загалом ця ER-діаграма демонструє добре продуману структуру, яка дозволяє ефективно керувати товарами, замовленнями, користувачами та процесом підтримки. Вона забезпечує гнучкість для масштабування системи, додавання нових функцій і підтримку складних бізнес-процесів.

3.2 Вибір системи управління базою даних та її реалізація

При розробці програмного забезпечення для автоматизованої підтримки клієнтів у сфері електронної комерції критичною складовою є правильний вибір системи управління базами даних (СУБД). База даних є центральним компонентом, що забезпечує збереження, обробку та організацію всієї інформації, пов’язаної з користувачами, товарами, замовленнями, повідомленнями чату та сесіями підтримки. Від ефективності роботи СУБД безпосередньо залежить продуктивність системи, швидкість відповіді на запити користувачів та надійність збережених даних.

Існує велика кількість систем управління базами даних (СУБД), які відрізняються за моделлю даних, функціональністю, продуктивністю та способом масштабування. Реляційні СУБД використовують таблиці з фіксованими стовпцями та рядками, підтримують SQL, транзакції та зв’язки між таблицями, що робить їх зручними для систем з чіткою структурою даних. Прикладами реляційних СУБД є MySQL, яка є популярною безкоштовною системою з відкритим кодом та широко використовується у веб-розробці,

PostgreSQL, що забезпечує підтримку складних типів даних і високу надійність, Oracle Database, яка відома своєю масштабованістю та корпоративним рівнем підтримки, та Microsoft SQL Server, що добре інтегрується в екосистему Microsoft і ефективна для бізнес-додатків. Документоорієнтовані СУБД зберігають дані у вигляді документів, наприклад JSON або XML, що робить їх зручними для роботи з неструктурованими та напівструктурованими даними, прикладами таких систем є MongoDB та CouchDB. СУБД типу «ключ-значення» забезпечують високу продуктивність і масштабованість за рахунок зберігання даних у парах «ключ-значення», до них відносяться Redis та Amazon DynamoDB.

Графові СУБД призначені для зберігання та аналізу взаємозв'язків між об'єктами і добре підходять для соціальних мереж, систем рекомендацій та побудови маршрутів, серед них Neo4j та OrientDB. Колонкоорієнтовані СУБД зберігають дані по стовпцях, що оптимально для аналітичних задач та роботи з великими обсягами даних, прикладами є Apache Cassandra та ClickHouse. Багатомодельні СУБД підтримують одночасно кілька моделей даних, такі як таблиці, документи і графи, серед них ArangoDB та OrientDB. Вибір конкретної СУБД залежить від особливостей проєкту, проте для веб-додатків та чат-ботів найчастіше використовують реляційні СУБД, наприклад MySQL або PostgreSQL, завдяки зручній роботі з таблицями, підтримці транзакцій та широкій спільноті розробників.

При виборі СУБД були враховані такі критерії: підтримка реляційної моделі даних, що забезпечує логічну організацію інформації у вигляді таблиць з чіткими зв'язками між ними; можливість використання складних запитів SQL для фільтрації, агрегації та сортування даних; масштабованість та здатність обробляти значні обсяги інформації; стабільність та надійність роботи; простота інтеграції з обраними технологіями бекенду, такими як PHP та Laravel; наявність документації та підтримки спільноти розробників [16].

На підставі аналізу сучасних рішень було обрано MySQL як СУБД для реалізації проєкту. MySQL забезпечує надійне зберігання реляційних даних,

підтримує транзакції, індекси та обмеження цілісності, що важливо для відстеження замовлень та сесій користувачів. Крім того, MySQL добре інтегрується з PHP-екосистемою, забезпечуючи простий доступ до даних через ORM (Object-Relational Mapping) або прямі SQL-запити.

Реалізація бази даних включає проєктування структур таблиць, визначення типів даних, первинних та зовнішніх ключів, а також зв’язків між таблицями для забезпечення цілісності даних. Основними таблицями є users, products, categories, orders, order_items, chat_messages та support_sessions. Для кожної таблиці були визначені атрибути, їхні типи та обмеження, які забезпечують коректність даних та унеможливлюють виникнення логічних помилок.

Після розробки структури таблиць бази даних була проведена її безпосередня реалізація у середовищі MySQL Workbench. Було створено всі таблиці, встановлено зв’язки між ними, додано індекси для оптимізації запитів та забезпечено автоматичне оновлення полів дат створення та зміни записів. Для полегшення подальшої роботи та інтеграції з бекендом було передбачено використання ORM Laravel, що дозволяє маніпулювати записами бази даних через об’єктно-орієнтовані моделі, спрощуючи процес розробки та підтримки програмного коду.

Впровадження обраної СУБД забезпечує надійне управління даними та високу швидкодію системи, дозволяючи користувачам отримувати швидкі та точні відповіді від чат-бота, а адміністраторам контролювати замовлення, сесії підтримки та стан товарів у режимі реального часу. Така організація даних створює міцну основу для подальшого розширення функціональності та масштабування системи.

Таким чином було створено базу даних в середовищі MySQL Workbench (Рис. 3.2).

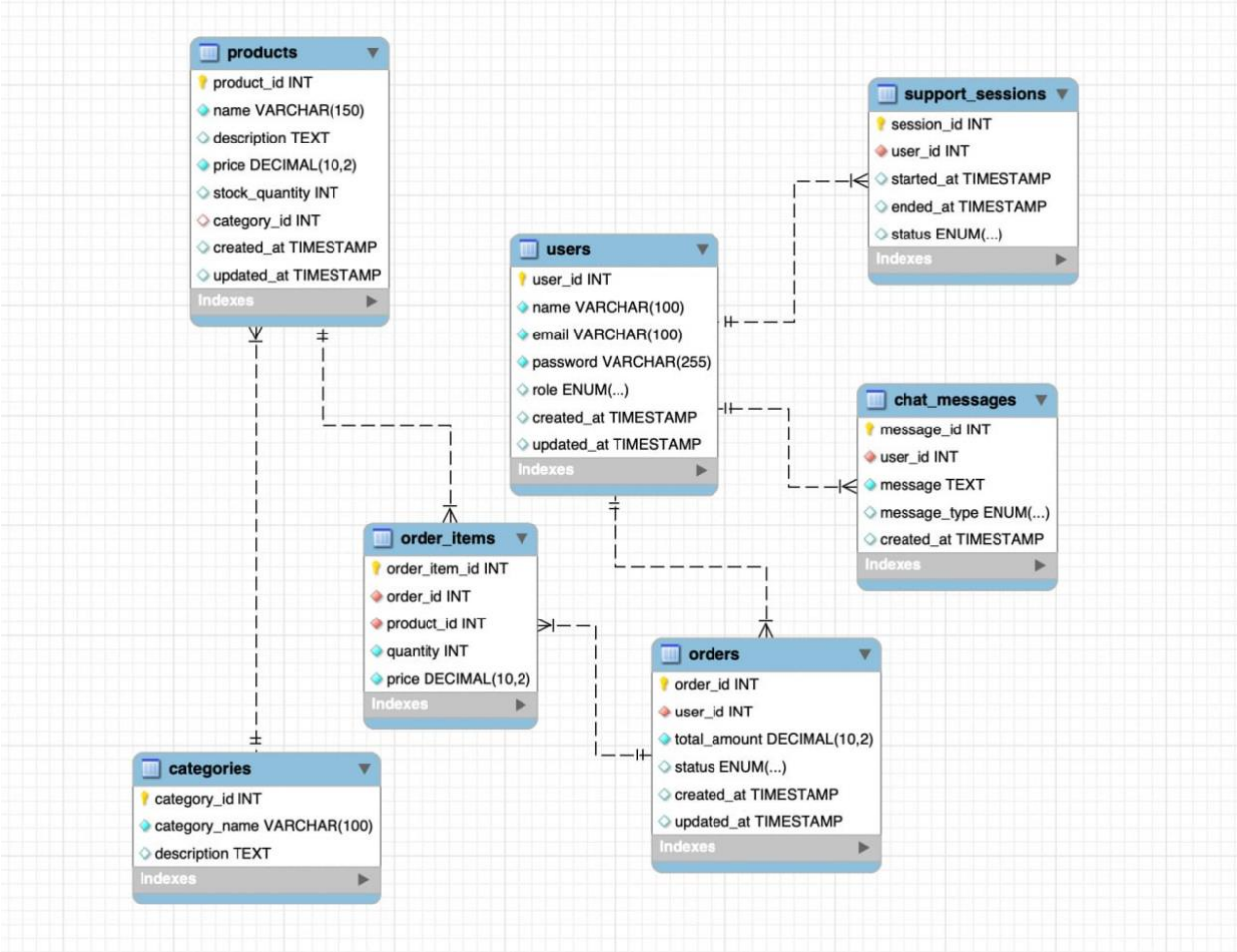

Рис. 3.2 База даних

Таблиця products зберігає інформацію про товари. Тут є ідентифікатор товару, назва, опис, ціна, кількість на складі, категорія, а також дати створення й оновлення. Вона пов’язана з таблицею категорій через поле category_id.

Таблиця categories описує групи товарів. Вона містить ідентифікатор категорії, назву та опис. Кожна категорія може включати багато товарів, що дозволяє структурувати каталог.

Таблиця users відповідає за дані користувачів. Тут є ідентифікатор, ім’я, email, пароль, роль (клієнт, адміністратор або оператор підтримки), а також часові мітки створення й оновлення. Користувачі можуть робити замовлення, створювати звернення до підтримки та надсилати повідомлення.

Таблиця orders зберігає інформацію про замовлення. Вона містить ідентифікатор замовлення, користувача, загальну суму, статус (наприклад, «pending», «processing», «completed», «cancelled») та часові мітки. Кожне замовлення пов’язане з користувачем через user_id.

Таблиця order_items деталізує замовлення. Вона містить ідентифікатор позиції, замовлення, товар, кількість і ціну. Це дозволяє відстежувати, які саме товари входять у кожне замовлення.

Таблиця support_sessions моделює сесії підтримки. Вона включає ідентифікатор сесії, користувача, час початку й завершення, а також статус (активна чи закрита). Це дозволяє відстежувати звернення клієнтів у рамках окремих діалогів.

Таблиця chat_messages зберігає повідомлення, що надсилаються в чаті. Вона містить ідентифікатор повідомлення, користувача, текст, тип повідомлення (від клієнта, бота чи оператора) та час створення. Це забезпечує історію комунікації між клієнтом і системою.

Зв’язки між таблицями побудовані так, щоб відображати реальні бізнес-процеси: користувачі створюють замовлення, замовлення складаються з товарів, товари належать до категорій, а користувачі можуть звертатися до підтримки й вести діалог через повідомлення. Це робить модель цілісною та готовою до використання в реальній системі.

Таблиця **users:**

- user_id – первинний ключ;
- name – ім’я користувача;
- email – електронна пошта;
- password – пароль;
- role – роль (customer, admin, support);
- created_at, updated_at – часові мітки.

Таблиця **orders:**

- order_id – первинний ключ;
- user_id – зовнішній ключ на users;
- total_amount – загальна сума замовлення;
- status – стан замовлення (pending, processing, completed, cancelled);
- created_at, updated_at – часові мітки.

Таблиця **order_items:**

- order_item_id – первинний ключ;
- order_id – зовнішній ключ на orders;
- product_id – зовнішній ключ на products;
- quantity – кількість товару;
- price – ціна за одиницю.

Таблиця **products:**

- product_id – первинний ключ;
- name – назва товару;
- description – опис;
- price – ціна;
- stock_quantity – кількість на складі;
- category_id – зовнішній ключ на categories;
- created_at, updated_at – часові мітки.

Таблиця **categories:**

- category_id – первинний ключ;
- category_name – назва категорії;
- description – опис.

Таблиця **support_sessions:**

- session_id – первинний ключ;
- user_id – зовнішній ключ на users;
- started_at, ended_at – час початку та завершення;
- status – стан (active, closed).

Таблиця **chat_messages:**

- message_id – первинний ключ;
- user_id – зовнішній ключ на users;
- message – текст повідомлення;
- message_type – тип (user, bot, support);
- created_at – час створення.

Таким чином, твоя база даних побудована за принципом «один-до-багатьох»: один користувач може мати багато замовлень, багато повідомлень і кілька сесій підтримки. Замовлення складаються з позицій, кожна з яких пов’язана з конкретним товаром, а товари групуються у категорії.

3.3 Архітектура програмного забезпечення

Існує кілька основних підходів до архітектури програмного забезпечення, кожен з яких визначає спосіб організації компонентів системи та їх взаємодії. Одні з найпоширеніших включають монотонну (монолітну) архітектуру, де всі функції та модулі інтегровані в один великий блок програми; така структура спрощує розробку на початкових етапах, але складно масштабується і важко підтримується у великих проєктах. Багаторівнева (n-tier) архітектура розділяє систему на логічні рівні: презентації, бізнес-логіки та управління даними, що підвищує масштабованість, безпеку та зручність обслуговування.

Існує також сервісно-орієнтована архітектура (SOA), де функції програми реалізовані як незалежні сервіси, що взаємодіють через стандартизовані інтерфейси; це дозволяє легко інтегрувати сторонні модулі та оновлювати систему без повного переписування. Мікросервісна архітектура є сучасним розвитком SOA і передбачає, що кожен сервіс виконує одну конкретну функцію, має власну базу даних і може масштабуватися незалежно від інших сервісів.

Ще існують подієво-орієнтовані архітектури, де взаємодія компонентів відбувається через події, що дозволяє системі асинхронно реагувати на дії користувачів або зовнішніх процесів. Архітектура з використанням клієнт-серверної моделі базується на розділенні ролей між клієнтом, який ініціює запити та відображає дані, і сервером, що обробляє запити та керує базою даних. Також існують peer-to-peer архітектури, де всі вузли системи

рівноправні і можуть обмінюватися інформацією без центрального сервера [17].

Кожна з цих архітектур має свої переваги та обмеження, і вибір залежить від масштабів системи, очікуваного навантаження, вимог до надійності, безпеки та зручності підтримки. У сучасній розробці часто використовують гібридні підходи, комбінуючи елементи багаторівневої, мікросервісної та подієво-орієнтованої архітектури для досягнення максимальної гнучкості та масштабованості.

У процесі розробки чат-бота для автоматизації підтримки клієнтів у сфері e-commerce було обрано чотирьохрівневу архітектуру програмного забезпечення як оптимальне рішення для забезпечення стабільності, масштабованості та зручності подальшого розвитку системи. Така архітектура дозволяє чітко розділити обов’язки між різними компонентами системи, що зменшує взаємозалежності між модулями та підвищує надійність програмного продукту.

Перший рівень, презентаційний, відповідає за взаємодію користувача з системою через інтерфейс чат-бота. Він забезпечує відображення інформації, отриманої від бізнес-логіки, та отримання запитів від користувача. Завдяки цьому рівню користувач отримує зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє швидко формулювати запити та отримувати відповіді, що критично для систем підтримки клієнтів, де час реакції безпосередньо впливає на задоволеність користувачів.

Другий рівень, рівень бізнес-логіки, реалізує основні функції системи. Він обробляє запити користувачів, керує сценаріями взаємодії, забезпечує валідацію даних та контролює правила бізнес-процесів. Відокремлення цієї логіки від презентаційного рівня дозволяє змінювати або додавати нові сценарії взаємодії без необхідності переробляти інтерфейс користувача. Також бізнес-логіка виконує роль посередника між користувачем і базою даних, абстрагуючи роботу з інформаційними ресурсами та забезпечуючи узгодженість даних.

Третій рівень, рівень доступу до даних, слугує для управління взаємодією з базою даних. Він ізолює бізнес-логіку від безпосередніх запитів до СУБД, що забезпечує незалежність системи від конкретної реалізації сховища даних і дозволяє при необхідності легко змінити СУБД або структуру таблиць без порушення роботи верхніх рівнів. Цей рівень також відповідальний за оптимізацію запитів та кешування даних, що підвищує швидкодію системи при великому навантаженні.

Четвертий рівень, рівень збереження даних, забезпечує надійне і безпечне зберігання інформації про користувачів, замовлення, історію взаємодій та параметри роботи чат-бота. Він гарантує цілісність і узгодженість даних, що критично для підтримки правильного функціонування системи та точного відстеження запитів користувачів.

Вибір чотирьохрівневої архітектури також обумовлений її масштабованістю та гнучкістю. Кожен рівень може розвиватися окремо, що дозволяє розширювати функціонал системи без ризику порушення роботи інших компонентів. Наприклад, можна додати аналітичні модулі, інтеграцію з новими платіжними системами або автоматизовану обробку запитів, не змінюючи інтерфейс користувача. Крім того, така архітектура значно спрощує тестування та відлагодження, оскільки помилки легко локалізувати на конкретному рівні, що підвищує ефективність процесу розробки та обслуговування системи.

Загалом, використання чотирьохрівневої архітектури дозволяє створити надійну, масштабовану та гнучку систему автоматизації підтримки клієнтів, яка відповідає сучасним вимогам e-commerce та забезпечує високий рівень задоволеності користувачів за рахунок швидкої та точної обробки запитів, прозорої комунікації та структурованого управління даними.

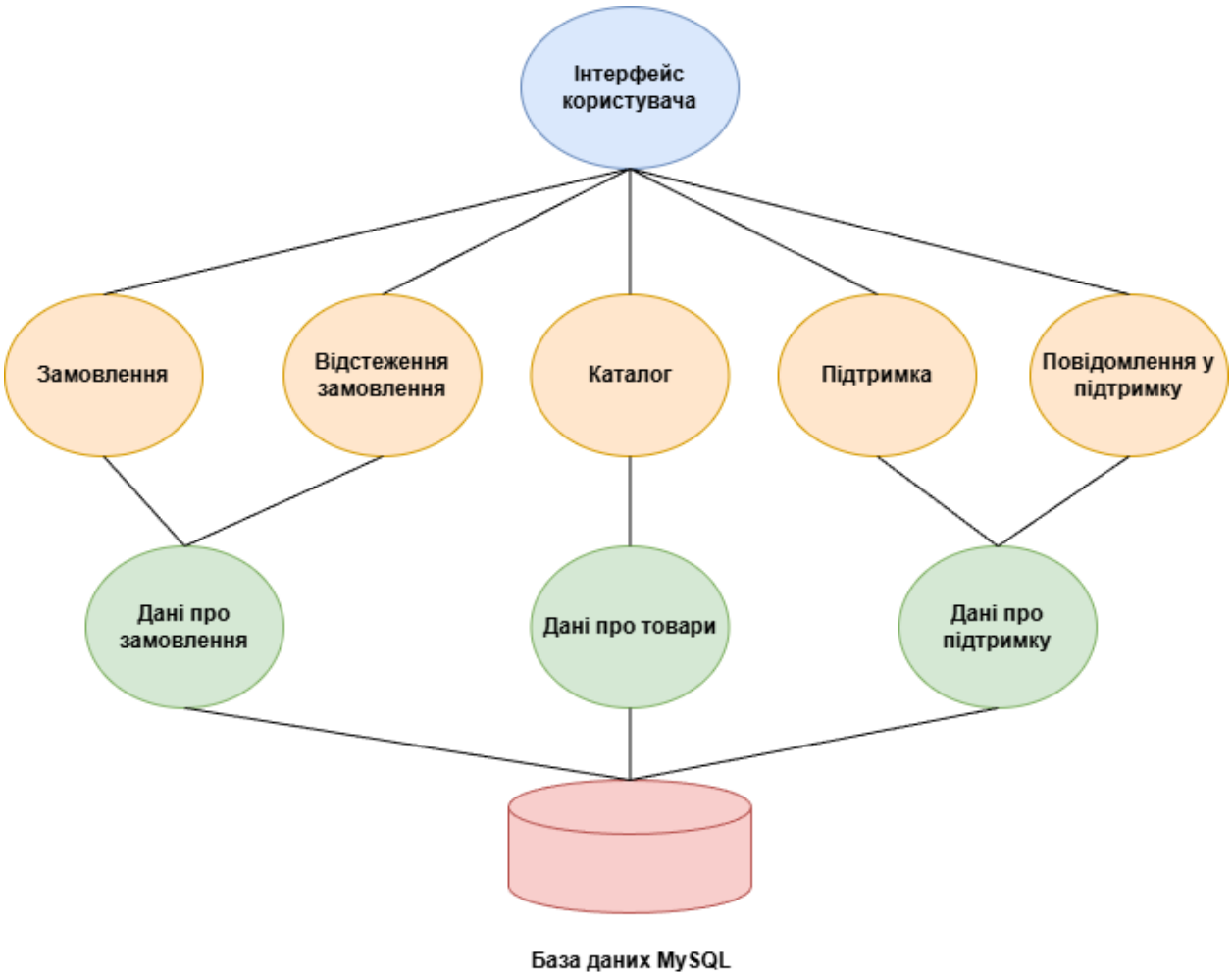

Рис. 3.3 Багатошарова архітектура

Ця чотиришарова архітектура програмного забезпечення побудована так, щоб чітко розділити рівні взаємодії між користувачем, бізнес-логікою та базою даних.

На верхньому рівні знаходиться інтерфейс користувача. Саме він забезпечує доступ до функціональних модулів системи: оформлення замовлення, відстеження його статусу, перегляд каталогу товарів, звернення до підтримки та надсилання повідомлень. Це той шар, з яким безпосередньо працює клієнт, і він відповідає за зручність та зрозумілість взаємодії.

Другий рівень складають функціональні модулі. Кожен із них виконує окрему задачу: модуль замовлень обробляє покупки, модуль відстеження показує статус, каталог відповідає за структуру товарів, а модуль підтримки дозволяє клієнту звернутися із проблемою чи питанням. Тут реалізується бізнес-логіка системи.

Третій рівень – це дані, які відповідають кожному модулю. Є дані про замовлення, дані про товари та дані про підтримку. Вони виступають проміжним шаром між бізнес-логікою та базою даних, забезпечуючи структуроване зберігання й доступ до потрібної інформації.

На четвертому рівні знаходиться база даних MySQL, яка є центральним сховищем усіх даних системи. Саме сюди потрапляє інформація про замовлення, товари та звернення до підтримки, і саме звідси вона витягується для роботи модулів та відображення в інтерфейсі користувача.

Уся архітектура побудована за принципом чіткого розділення відповідальностей: інтерфейс забезпечує взаємодію, модулі реалізують логіку, дані структурують інформацію, а база даних гарантує її збереження. Це робить систему масштабованою, зрозумілою та стійкою до змін.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Діаграма пакетів у UML є інструментом для візуалізації високорівневої організації програмного забезпечення та відображення логічних залежностей між різними частинами системи. Пакет у такій діаграмі – це групування класів, інтерфейсів або інших елементів моделі, яке дозволяє структурувати програму на окремі логічні компоненти та приховати деталі реалізації всередині пакетів.

Основна мета діаграми пакетів – продемонструвати структуру системи на рівні модулів і взаємозв’язки між ними, що особливо корисно для великих проєктів, де багато класів та компонентів. Вона показує, які частини системи залежать одна від одної, і допомагає визначити точки інтеграції, а також спростити підтримку та масштабування програмного забезпечення [18].

У діаграмі пакетів зв’язки між пакетами можуть бути різних типів: залежності, імпорти, реалізації або асоціації. Це дозволяє розробникам швидко оцінити, які модулі впливають на інші, і визначити області, які можна модифікувати без ризику порушення роботи системи. Крім того, діаграма

пакетів допомагає визначити логічні блоки для командної роботи, оскільки кожен пакет можна призначити конкретній групі розробників.

Діаграми пакетів використовуються як для проєктування нових систем, так і для документування існуючих, забезпечуючи узгоджене уявлення про структуру програмного забезпечення на високому рівні. Вони добре доповнюють діаграми класів, оскільки дозволяють бачити не окремі класи, а їх логічні об’єднання, і тим самим спрощують управління складними проєктами.

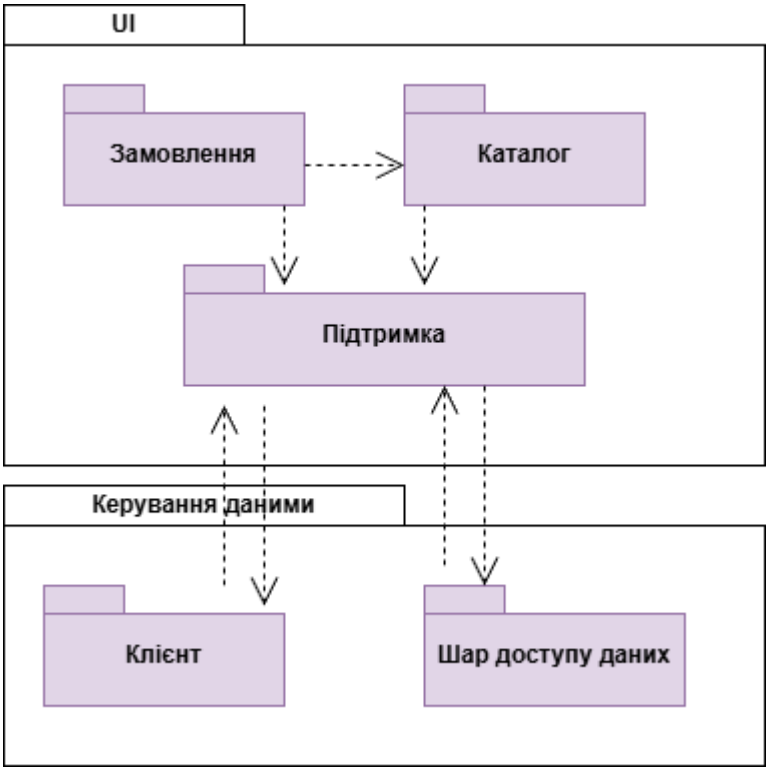

Рис. 3.4 Діаграма пакетів

Ця діаграма пакетів показує, як програмна система розділена на окремі модулі та рівні, кожен з яких відповідає за свою частину функціональності. У верхньому шарі знаходиться UI, тобто інтерфейс користувача. Тут виділені пакети «Замовлення» та «Каталог», які забезпечують роботу з оформленням покупок і переглядом товарів. Між ними є залежність: модуль замовлень може звертатися до каталогу, щоб отримати інформацію про товари.

Обидва ці пакети пов’язані з центральним модулем Підтримка, який виступає посередником між інтерфейсом і нижчими рівнями системи. Він

відповідає за обробку звернень, інтеграцію бізнес-логіки та передачу даних далі.

У нижньому шарі «Керування даними» розташовані два пакети: Клієнт і Шар доступу даних. Вони отримують інформацію від модуля підтримки. «Клієнт» відповідає за взаємодію з користувачем у контексті даних, а «Шар доступу даних» забезпечує роботу з базою даних, тобто фактичне збереження та отримання інформації.

Таким чином, архітектура побудована за принципом багатошаровості: інтерфейс користувача звертається до бізнес-логіки, бізнес-логіка працює з клієнтськими даними та шаром доступу, а той у свою чергу взаємодіє з базою даних. Це дозволяє чітко розділити відповідальність між пакетами й зробити систему більш зрозумілою та масштабованою.

3.5 Вибір інструментарію для створення програмного забезпечення

Для розробки програмного було здійснено ретельний вибір інструментів та технологій, які дозволяють створити сучасну, надійну та масштабовану систему. Основними критеріями при виборі програмного забезпечення були функціональні можливості, сумісність з веб-технологіями, популярність серед розробників, активна підтримка спільноти, наявність документації, а також здатність забезпечити ефективну інтеграцію з різними сервісами та API. Такі вимоги обумовлені необхідністю реалізації чат-бота, який здатний в режимі реального часу взаємодіяти з користувачами, обробляти їх запити та вести облік замовлень у структурованій формі.

Серверна частина додатку реалізується на мові PHP, що дозволяє обробляти запити клієнтів, виконувати бізнес-логіку та працювати з базою даних. Для організації коду, управління бізнес-логікою та забезпечення масштабованості використовується фреймворк Symfony. Він надає багаторівневу архітектуру додатку, яка дозволяє відокремити логіку

презентації, бізнес-процеси та управління даними, підвищує повторне використання компонентів і полегшує підтримку та розвиток проєкту у майбутньому. Symfony також забезпечує безпеку додатку, обробку помилок, систему роутингу запитів та інтеграцію з зовнішніми сервісами.

Для збереження та обробки даних обрана реляційна система управління базами даних MySQL, яка забезпечує надійне зберігання інформації, підтримку складних запитів та узгодженість даних. MySQL дозволяє ефективно працювати з великими обсягами інформації, зберігати історію взаємодій користувачів, облік замовлень та повідомлень, а також здійснювати аналітику дій користувачів. Для проєктування структури бази даних, візуалізації зв’язків між таблицями та подальшої генерації SQL-коду застосовується MySQL Workbench. Це дозволяє розробникам отримати наочну картину структури даних та полегшує процес внесення змін у базу даних [19].

Інтерактивна взаємодія користувача з системою забезпечується через Telegram API, що дозволяє створити чат-бота, який у режимі реального часу отримує повідомлення від користувачів, обробляє їх та надсилає відповіді. API Telegram дозволяє реалізувати функціонал автоматичного оброблення запитів клієнтів, відправку повідомлень про статус замовлення, створення кнопок та меню для зручної навігації, а також інтегрувати бота з іншими сервісами для підвищення ефективності взаємодії.

Розробка та написання коду ведеться у середовищі Visual Studio Code, яке підтримує PHP, HTML, CSS та JavaScript, а також надає інтеграцію з системами контролю версій, що дозволяє організувати ефективну роботу над проєктом, вести облік змін та спільну розробку. Вибір цього інструменту обумовлений його легкістю, високою функціональністю, наявністю численних плагінів та можливістю налаштування середовища відповідно до потреб розробника.

Використання цього комплексу інструментів і технологій забезпечує створення повнофункціональної, стабільної та масштабованої системи для автоматизації підтримки клієнтів у сфері електронної комерції. Комбінація

PHP, Symfony, MySQL, Telegram API, MySQL Workbench та Visual Studio Code дозволяє інтегрувати всі рівні додатку – від інтерфейсу користувача до бази даних – у єдину систему, що забезпечує швидку, безпечну та надійну роботу чат-бота. Усі вибрані засоби дозволяють реалізувати основні функції системи, такі як реєстрація та автентифікація користувачів, облік замовлень, ведення історії взаємодій та обробку запитів у режимі реального часу, а також забезпечують подальший розвиток і масштабування проєкту.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для забезпечення коректної та стабільної роботи системи чат-бота для автоматизації підтримки клієнтів у сфері електронної комерції визначено апаратні та програмні вимоги, що враховують обсяг оброблюваних даних, кількість користувачів, які одночасно взаємодіють із системою, та інтеграцію з зовнішніми сервісами.

Щодо апаратних вимог, серверна частина повинна мати достатню продуктивність для обробки одночасних запитів користувачів та виконання бізнес-логіки додатку. Рекомендується використовувати процесор не нижче чотирьох ядер із тактовою частотою 2,0 ГГц або вище, оперативну пам’ять об’ємом не менше 8 ГБ для забезпечення стабільної роботи PHP та бази даних, а також швидкий SSD-диск для зберігання даних та прискорення доступу до них. Сервер має мати стабільне підключення до Інтернету для інтеграції з Telegram API та обробки повідомлень у реальному часі. Для розробки та тестування на локальному середовищі досить сучасного персонального комп’ютера або ноутбука з мінімальними параметрами, що відповідають вимогам програмного забезпечення.

Програмні вимоги включають операційну систему з підтримкою веб-серверів та PHP. Для серверного середовища рекомендується використовувати Linux або Windows із встановленим веб-сервером Apache або Nginx, PHP версії 8.0 або вище, а також необхідними розширеннями для роботи з базами даних MySQL. Для управління та проєктування бази даних використовується MySQL Workbench, що дозволяє візуалізувати структуру даних, створювати та модифікувати таблиці, а також генерувати SQL-запити для взаємодії з сервером.

Клієнтська частина додатку передбачає наявність сучасного веб-браузера для тестування та відображення інтерфейсу адміністратора або користувача. Для розробки коду застосовується Visual Studio Code, що підтримує синтаксис PHP, HTML, CSS, JavaScript, а також інтеграцію із системами контролю версій, що забезпечує ефективне ведення проєкту та спільну роботу над ним. Для інтеграції з Telegram API необхідно мати обліковий запис у Telegram та доступ до Bot API для обробки повідомлень та реалізації функціоналу чат-бота.

Узгодження апаратних та програмних вимог дозволяє забезпечити стабільну та безперебійну роботу системи, мінімізувати ризики помилок, підвищити швидкість обробки запитів та забезпечити ефективну взаємодію з користувачами. Вибір надійного апаратного забезпечення та сучасних інструментів розробки є ключовим для досягнення високого рівня продуктивності та масштабованості системи, а також для забезпечення її подальшого розвитку та інтеграції з іншими сервісами електронної комерції.

Також було зроблено діаграму розгортання для візуального огляду деплою системи (Рис. 4.1).

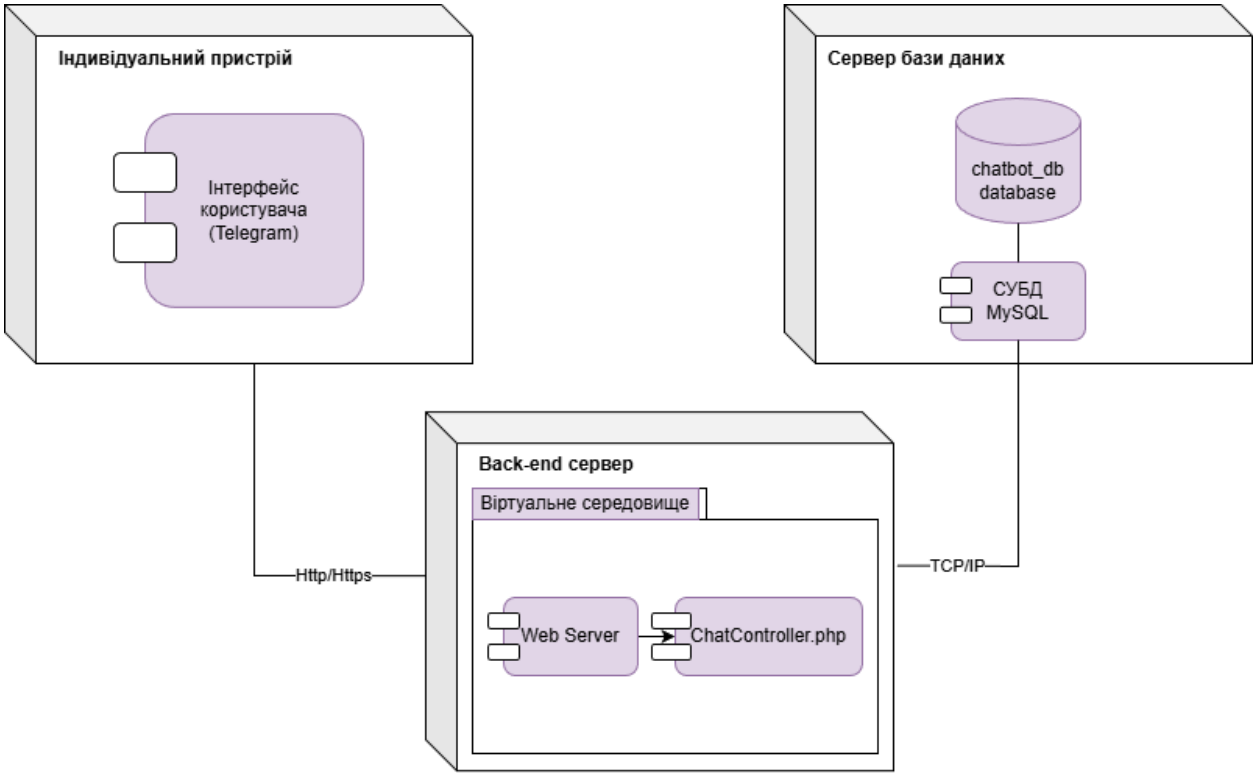

Рис. 4.1 Діаграма розгортання

4.2 Тестування системи

Запустивши систему, потрапляємо у початкове меню вибору дій. Перед нами чекає вибір з 4 основних дій нашого бота - Каталог, замовлення, підтримка та інформація про бота (рис. 4.2).

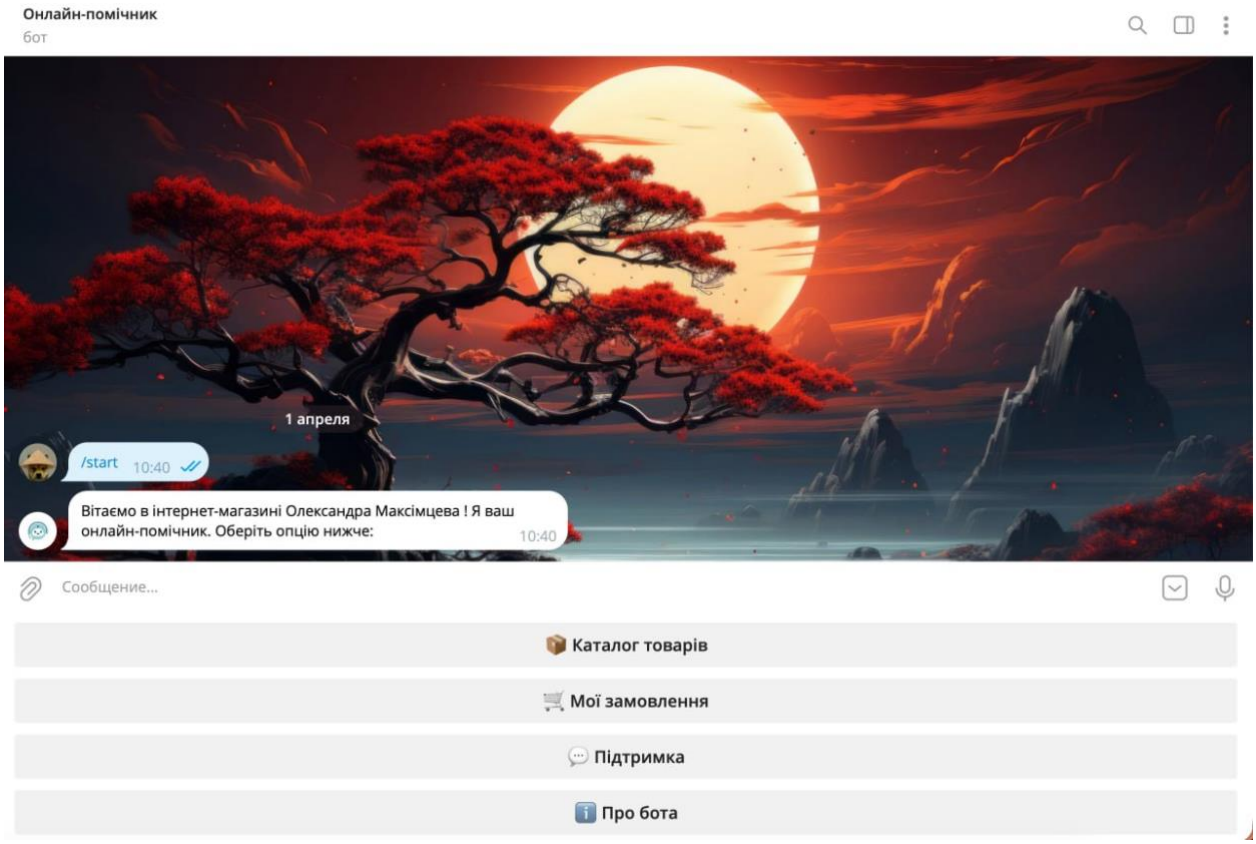

Рис. 4.2 Початкове меню

Почнемо з вибору каталогу товарів. Натискаємо на даний варіант у меню та бот виводить перед нами існуючі товари із сервера магазину (Рис. 4.3).

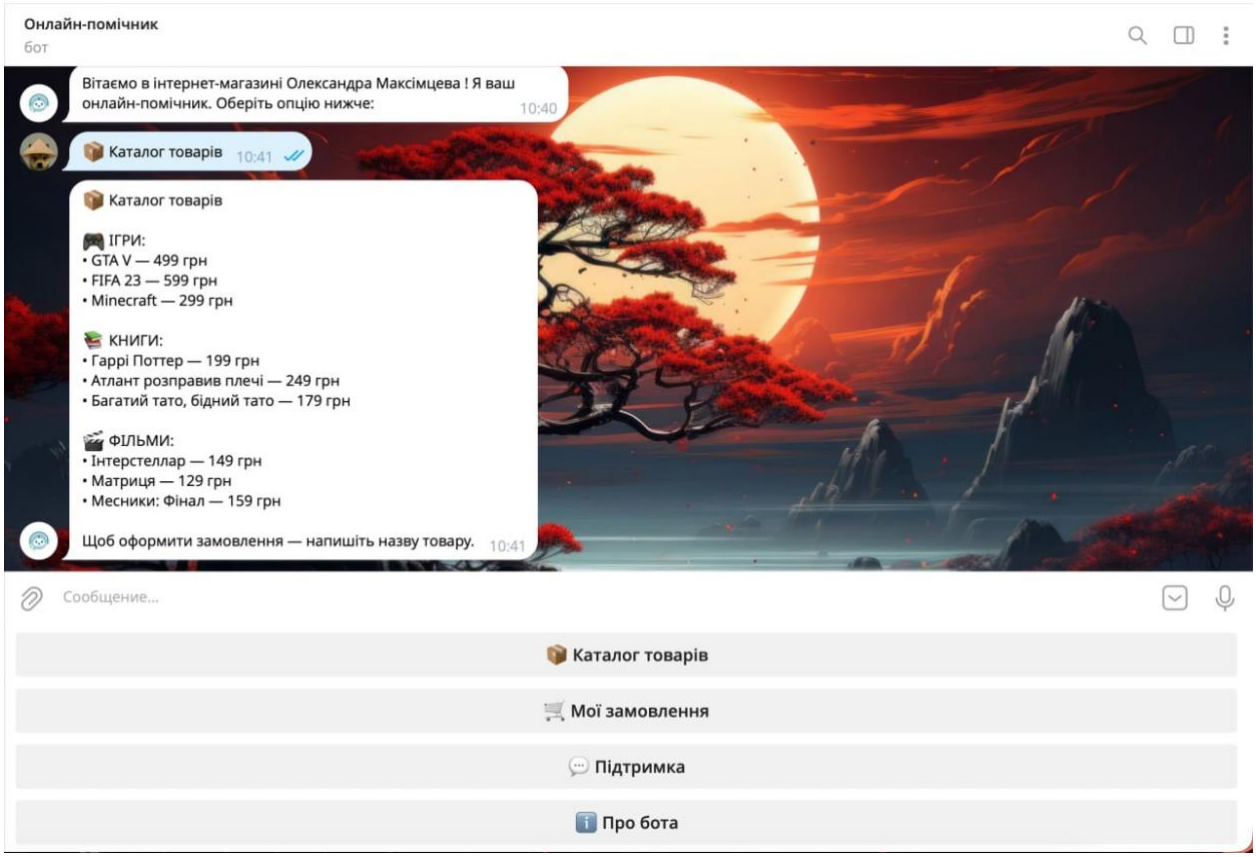

Рис. 4.3 Варіант “Каталог товарів”

Тепер можемо перейти на "Мої замовлення", чат-бот попросить нас ввести наш номер замовлення, ми вводимо його номер та отримуємо поточну інформацію щодо статусу нашого замовлення (Рис. 4.4).

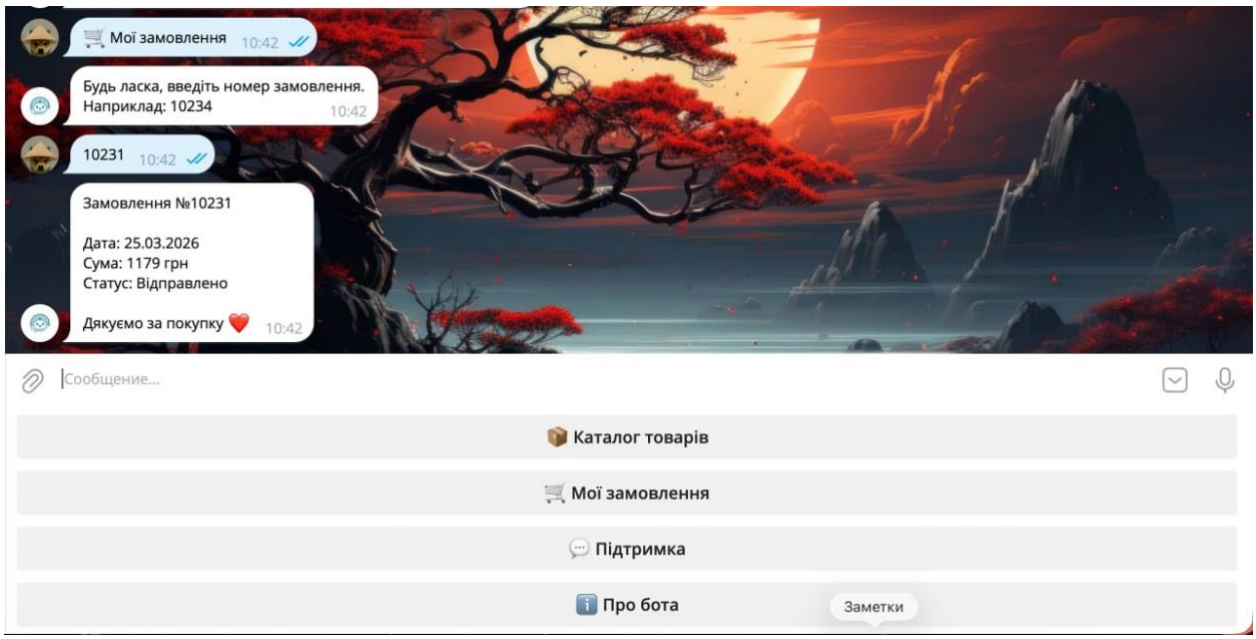

Рис. 4.4 Варіант “Мої замовлення”

Також можемо додатково дізнатися інформацію про робота, тут з'явиться його поточна версія і автор (Рис. 4.5).

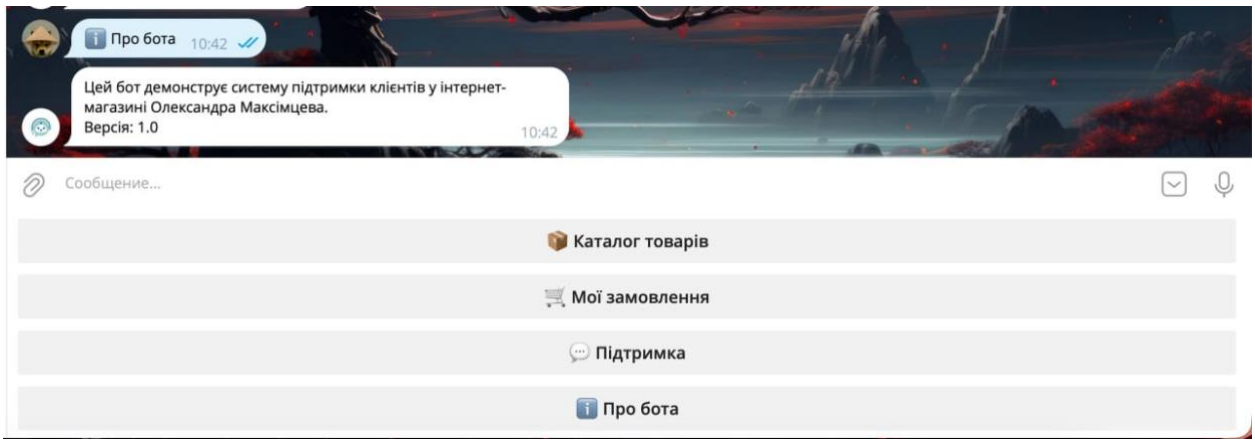

Рис. 4.5 Варіант “Про бота”

Вибираємо варіант "Підтримка", тут ми переходимо в меню вибору тем для звернення на підтримку (Рис. 4.6).

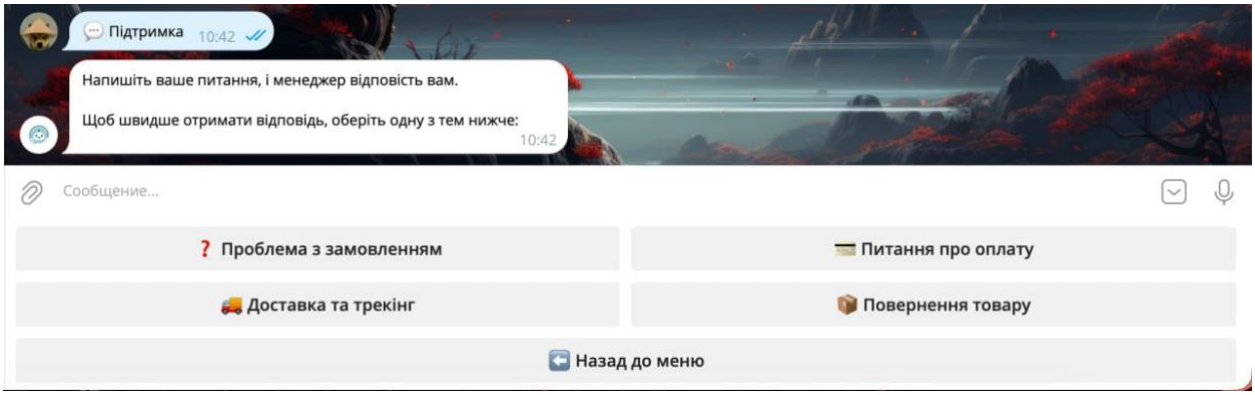

Рис. 4.6 Варіант “Підтримка”

Тут вибираємо цікаву для нас тему для звернення, наприклад "Проблеми із замовленням", далі бот просить нас описати нашу проблему (Рис. 4.7).

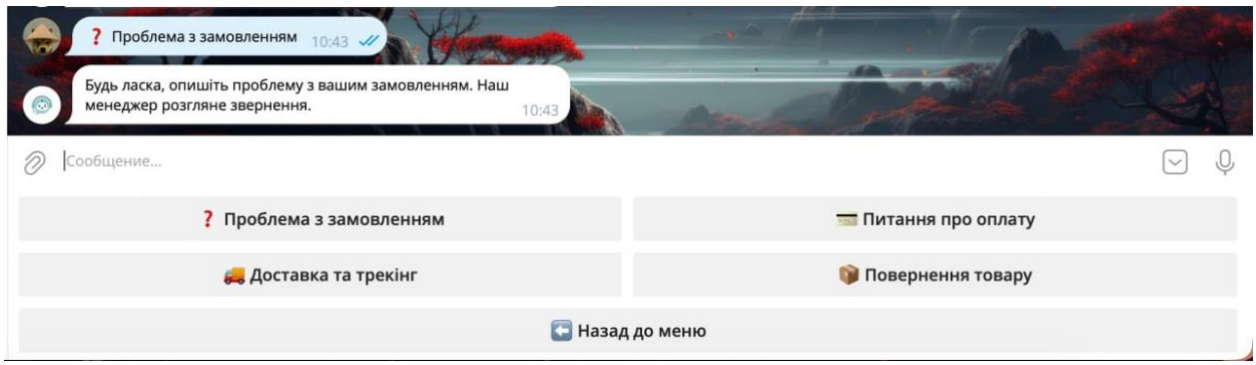

Рис. 4.7 Обрана тема звернення “Проблема з замовленням”

Після цього користувач може вписати текст своєї проблеми у розгорнутому вигляді. Дане сполучення обробиться та потрапить у службу підтримки магазину для подальшої взаємодії клієнта та власника магазину (Рис. 4.8).

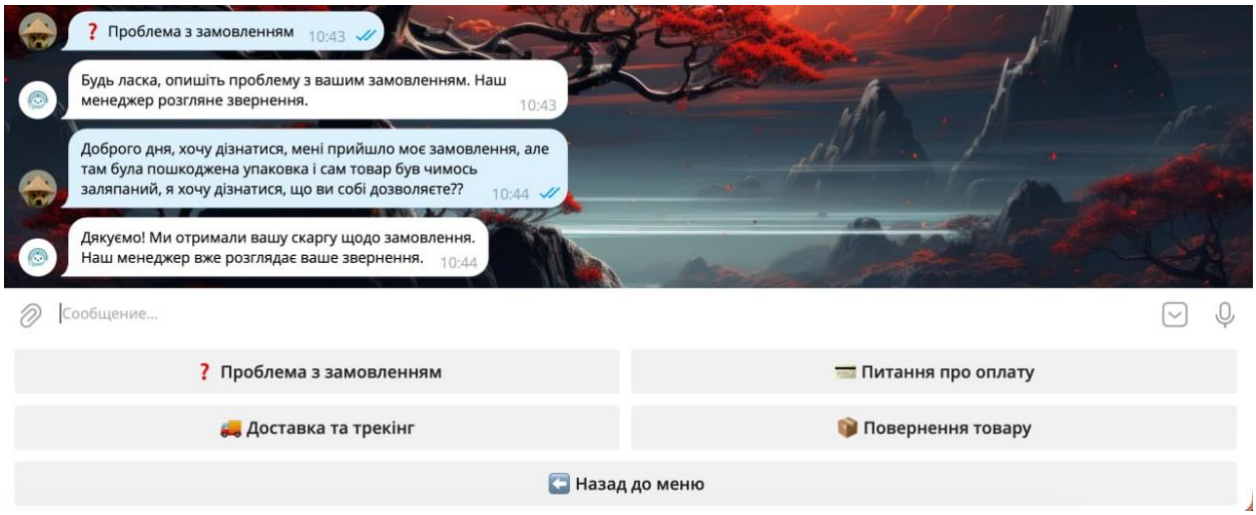

Рис. 4.8 Текст повідомлення до підтримки

ВИСНОВКИ

У цій бакалаврській роботі було вироблено розробку програмного забезпечення чат-бота для автоматизації підтримки клієнтів у сфері електронної комерції. У вступі обґрунтовано актуальність дослідження, визначено об’єкт та предмет дослідження, а також поставлено мету – створення інтерактивного чату, який забезпечує ефективну взаємодію між клієнтами та адміністраторами інтернет-магазину в режимі реального часу.

Було проведено детальний аналіз предметної області, визначено основні терміни, атрибути та взаємозв’язки між ними, а також охарактеризовано існуючі рішення на такому ринку як Chatfuel та ManyChat, що дозволило виділити сильні та слабкі сторони сучасних систем і сформувати вимоги до власного рішення. На основі цього було сформульовано функціональні та нефункціональні вимоги, описані вимоги до інтерфейсу користувача, що забезпечують зручність взаємодії та персоналізацію відповідей чат-бота.

Для моделювання систем використовується UML-методика, зокрема діаграми прецедентів, компонентів, діяльності, класів та пакетів. Було розроблено сценарії використання основних варіантів, таких як «Відкрити номер замовлення» та «Написати повідомлення до підтримки», що дозволило відтворити логіку взаємодії користувачів із системою. Використання класових діаграм та пакетів діаграм забезпечило чітку організацію компонентів системи та їх взаємозв’язків, а аналіз абстракцій предметної області дозволив формалізувати основні сутності та зв’язки між ними.

Було здійснено вибір та обґрунтовано архітектуру програмного забезпечення. Було обрано чотирьохрівневу архітектуру, яка розділяє презентаційний рівень, рівень бізнес-логіки, рівень управління даними та інтеграційний рівень, що дозволило підвищити масштабованість, безпеку та

зручність систем підтримки. Для реалізації програмного забезпечення вибрані інструменти та технології: PHP, Symfony, MySQL, Telegram API, Visual Studio Code та MySQL Workbench. Таке підключення забезпечило стабільну роботу сервера, інтеграцію платформи Telegram та ефективну роботу з базою даних.

У рамках роботи було спроєктовано та реалізовано базу даних, що включає таблиці користувачів, замовлень, повідомлень та категорій, а також забезпечено підтримку структурованих запитів і груп даних. Було розроблено логіку обробки повідомлень користувачів, реалізовано функціональні каталоги товарів, обробку запитів на підтримку, а також генерацію випадкових даних для демонстрації роботи системи.

Результатом роботи є функціональний прототип чат-бота для автоматизації підтримки клієнтів в інтернет-магазині, що дозволяє користувачам швидко отримувати інформацію про замовлення, звертатися до служби підтримки та взаємодіяти з системою в зручному режимі. Реалізація цього рішення демонструє ефективність застосування сучасних веб-технологій, автоматизації бізнес-процесів та інтеграції з комунікаційними платформами.

Таким чином, виконана робота підтверджує доцільність використання чат-ботів для підвищення якості обслуговування клієнтів, зменшення завантажень на підтримку персоналу та оптимізацію операційних процесів інтернет-магазинів, а отримані результати можуть стати основою для подальшого розвитку та масштабування подібних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chatbot Tutorial – Chatbots Magazine – [Електронний ресурс] – Режим доступу: https://chatbotsmagazine.com/
2. Towards Data Science – Introduction to Chatbots – [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/introduction-to-chatbots-a70a759f22d0/
3. Gartner – Market Guide for Virtual Customer Assistants – [Електронний ресурс] – Режим доступу: https://www.gartner.com/en/documents/
4. ResearchGate – Chatbot AI in E-commerce (наукові статті) – [Електронний ресурс] – Режим доступу: https://www.researchgate.net/
5. Google Cloud – Best Practices for API Design – [Електронний ресурс] – Режим доступу: https://cloud.google.com/apis/design
6. Amazon AWS – Designing Scalable Systems – [Електронний ресурс] – Режим доступу: https://aws.amazon.com/architecture/
7. FreeCodeCamp – Build a Chatbot with PHP – [Електронний ресурс] – Режим доступу: https://www.freecodecamp.org/news/how-to-build-a-chatbot-in-php/
8. Гамма, Е., Гельм, Р., Джонсон, Р., Вліссідес, Дж. Шаблони проєктування: елементи багаторазового об'єктно-орієнтованого програмного забезпечення. – Addison-Wesley, 1995.
9. Fowler, M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2002.
10. Larman, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. – Prentice Hall, 2004.
11. Date, C.J. An Introduction to Database Systems. – Pearson, 2003.
12. Ambler, S. The Object Primer: Agile Model-Driven Development with UML 2. – Cambridge University Press, 2004.
13. Buschmann, F., et al. Pattern-Oriented Software Architecture, Volume 1. – Wiley, 1996.

14. Tilkov, S., Vinoski, S. Node.js: Architecting Scalable Server-Side Solutions. – O'Reilly Media, 2012.
15. Hartl, M. Ruby on Rails Tutorial: Learn Web Development with Rails. – Addison-Wesley, 2016.
16. Ullman, L. PHP for the Web: Visual QuickStart Guide. – Peachpit Press, 2021.
17. Beighley, L., Morrison, M. Head First PHP & MySQL. – O'Reilly Media, 2009.
18. PHP Manual – [Електронний ресурс] – Режим доступу: https://www.php.net/manual/
19. Symfony Documentation – [Електронний ресурс] – Режим доступу: https://symfony.com/doc/current/index.html
20. MySQL 8.0 Reference Manual – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/
21. Telegram Bot API Documentation – [Електронний ресурс] – Режим доступу: https://core.telegram.org/bots/api
22. MySQL Workbench Manual – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/workbench/en/
23. MDN Web Docs – JavaScript Guide – [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
24. W3C HTML5 Specification – [Електронний ресурс] – Режим доступу: https://www.w3.org/TR/html52/
25. CSS-Tricks – A Complete Guide to Flexbox – [Електронний ресурс] – Режим доступу: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
26. REST API Tutorial – [Електронний ресурс] – Режим доступу: https://restfulapi.net/
27. OWASP Top Ten Security Risks – [Електронний ресурс] – Режим доступу: https://owasp.org/www-project-top-ten/
28. IBM Developer – Microservices Architecture – [Електронний ресурс] – Режим доступу: https://developer.ibm.com/articles/what-are-microservices/

29. Oracle – Database Design Concepts – [Електронний ресурс] – Режим доступу: https://docs.oracle.com/en/database/
30. StackOverflow – Various Q&A on PHP and MySQL – [Електронний ресурс] – Режим доступу: https://stackoverflow.com/
31. Russell, S., Norvig, P. Artificial Intelligence: A Modern Approach. – Pearson, 2016.
32. Jurafsky, D., Martin, J.H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. – Prentice Hall, 2021.
33. IBM Developer – Building Conversational AI with Watson Assistant – [Електронний ресурс] – Режим доступу: https://developer.ibm.com/articles/building-conversational-ai-with-watson-assistant/
34. Microsoft Docs – Bot Framework Documentation – [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/azure/bot-service/
35. Chatbot News Daily – Chatbots and Customer Support Trends – [Електронний ресурс] – Режим доступу: https://chatbots.org/news/
36. Medium – How to Build AI Chatbots for E-Commerce – [Електронний ресурс] – Режим доступу: https://medium.com/@someauthor/how-to-build-ai-chatbots-for-e-commerce

ДОДАТОК А

Фрагменти програмного коду. Функція перегляду статусу замовлення

```
<?php

$host = "localhost";
$db = "ecommerce_db";
$user = "root";
$pass = "";

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect_error) {
   die("Помилка підключення: " . $conn->connect_error);
}

function displayOrder($orderNumber, $conn) {
   $stmt = $conn->prepare("SELECT id, status, created_at, total FROM orders WHERE id = ?");
   $stmt->bind_param("i", $orderNumber);
   $stmt->execute();
   $result = $stmt->get_result();

   if($result->num_rows > 0) {
      $order = $result->fetch_assoc();
      echo "<h2>Замовлення #{$order['id']}</h2>";
      echo "<p>Статус: <b>{$order['status']}</b></p>";
      echo "<p>Дата оформлення: {$order['created_at']}</p>";
      echo "<p>Сума замовлення: {$order['total']} грн</p>";

      $historyStmt = $conn->prepare("SELECT event_date, event_description FROM order_history WHERE order_id = ? ORDER BY event_date ASC");
      $historyStmt->bind_param("i", $orderNumber);
      $historyStmt->execute();
      $historyResult = $historyStmt->get_result();

      echo "<h3>Історія замовлення:</h3>";
      echo "<ul>";
      while($event = $historyResult->fetch_assoc()) {
         echo "<li>{$event['event_date']} - {$event['event_description']}</li>";
```

```
        }
        echo "</ul>";

        $historyStmt->close();
    } else {
        echo "<p>Замовлення з номером $orderNumber не знайдено.</p>";
    }

    $stmt->close();
}

if(isset($_POST['order_number'])) {
    $orderNumber = intval($_POST['order_number']);
    displayOrder($orderNumber, $conn);
} else {
    echo '<form method="post">
            Введіть номер замовлення: <input type="text" name="order_number" required>
            <input type="submit" value="Перевірити статус">
          </form>';
}

$conn->close();
?>
```

ДОДАТОК Б

Фрагменти програмного коду. Функціонал звернення до підтримки

```
<?php

$host = "localhost";
$db = "ecommerce_db";
$user = "root";
$pass = "";

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect_error) {
   die("Помилка підключення: " . $conn->connect_error);
}

function createSupportRequest($userId, $topic, $message, $conn) {
   $stmt = $conn->prepare("INSERT INTO support_requests (user_id, topic, message, created_at, status) VALUES (?, ?, ?, NOW(), 'Очікує відповіді')");
   $stmt->bind_param("iss", $userId, $topic, $message);
   if($stmt->execute()) {
      echo "<p>Дякуємо! Ваш запит на підтримку успішно створено. Менеджер зв'яжеться з вами найближчим часом.</p>";
   } else {
      echo "<p>Сталася помилка при створенні запиту. Спробуйте пізніше.</p>";
   }
   $stmt->close();
}

if(isset($_POST['topic']) && isset($_POST['message'])) {
   $userId = 1;
   $topic = $_POST['topic'];
   $message = $_POST['message'];
   createSupportRequest($userId, $topic, $message, $conn);
} else {
   echo '<form method="post">
         <label>Оберіть тему запиту:</label>
         <select name="topic" required>
            <option value="Проблема з замовленням">Проблема з замовленням</option>
            <option value="Питання про оплату">Питання про оплату</option>
```

```
            <option value="Доставка та трекінг">Доставка та трекінг</option>
            <option value="Повернення товару">Повернення товару</option>
        </select><br><br>
        <label>Опишіть вашу проблему:</label><br>
        <textarea name="message" rows="5" cols="50" required></textarea><br><br>
        <input type="submit" value="Відправити запит">
      </form>';
}

$conn->close();
?>
```

ДОДАТОК В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, ____Максімцев_Олександр_Сергійович_____, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «**Програмне забезпечення чат-бота для автоматизації підтримки клієнтів у сфері електронної комерції**» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [+] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, ___Gemini__Claude _____інші (вказати назву)) були використані для:
        - [+] перекладу іншомовних джерел;
        - [+] стилістичного редагування та перевірки граматики;
        - [+]допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Максімцев О.С**
(підпис)