ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка чат-боту магазину доставки суші»

Студентки 4 курсу, 8 групи,
спеціальності
122 «Комп'ютерні науки» Попик Анастасія Олегівна

_____
підпис студента

Науковий керівник
кандидат педагогічних наук, доцент Дивак Володимир Валерійович

_____
підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент Демідов Павло Георгійович

_____
підпис керівника

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022 р.

Завдання
на випускну кваліфікаційну роботу студенту
Попик Анастасія Олегівна
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи
«Розробка чат-боту магазину доставки суші»
Затверджена наказом ректора від *9 грудня 2022 р. №3332*
2. Строк здачі студентом закінченої роботи *30 травня 2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробити чат-бот магазину доставки суші
Об'єкт дослідження: розробки чат-боту магазину доставки суші
Предмет дослідження: методи та технології розробки чат-ботів магазину доставки суші
4. Перелік графічного матеріалу _____
_____
_____
_____
_____

5.Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Дивак В.В	15.12.2022	15.12.2022
2	Дивак В.В	15.12.2022р.	15.12.2022
3	Дивак В.В	15.12.2022р.	15.12.2022р.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальна характеристика чат-ботів

1.2. Огляд технологій створення чат -ботів

РОЗДІЛ 2. МОДЕЛЬ ЧАТ-БОТУ МАГАЗИНУ ДОСТАВКИ СУШІ

2.1. Модель чат-боту магазину доставки суші

2.2. Методи і технологія розробки чат-боту

2.3. Збір бази даних для чат-боту

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ЧАТ-БОТУ МАГАЗИНУ ДОСТАВКИ СУШІ

3.1. Перевірка чат-боту магазину доставки суші

3.2. Тестування бота

3.3 Рекомендації з упровадження чат-боту магазину доставки суші

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	04.10.2022	04.10.2022
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	15.12.2022	15.12.2022
3	*Вступ*	03.02.2023	03.02.2023
4	*РОЗДІЛ 1. Дослідження та аналіз предметної області*	28.02.2023	28.02.2023
5	*РОЗДІЛ 2. Модель чат-боту магазину доставки суші*	06.04.2023	06.04.2023
6	*РОЗДІЛ 3. Програмна реалізація чат-боту магазину доставки суші*	12.05.2023	12.05.2023
7	*Висновки*	15.05.2023	15.05.2023
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	30.05.2023	30.05.2023
9	*Попередній захист випускної кваліфікаційної роботи*	31.05.2023-01.06.2023	31.05.2023-01.06.2023
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	02.06.2023	02.06.2023
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	05.06.2023	05.06.2023
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022р.

9. Керівник випускної кваліфікаційної роботи
Дивак В.В
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Попик А. О.
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

Керівник випускної кваліфікаційної роботи
_____ 30.05.2023р.
(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента Попик Анастасія Олегівна
(підпис, прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри Пурський О.І.
(підпис, прізвище, ініціали)

«_____» _____2023 р.

Анотація

У випускному кваліфікаційному проєкті було здійснено комплексну розробку чат-боту магазину доставки суші. Досліджено теоретично предметну область, особливості виконання завдання. Обґрунтовано методи та засоби розробки, а також реалізації чат-боту магазину доставки суші.

Запропоновано концепцію розробки чат-боту доставки суші для зручного оформлення замовлення клієнтами даного магазину. У роботі було використано мову програмування Python та бібліотеку python-telegram-bot для взаємодії з Telegram API.

Ключові слова: чат-бот, функціональна модель, методи розробки, база даних.

Anotation

In the graduation qualification project, a comprehensive development of a chatbot for a sushi delivery store was carried out. The theoretical domain and the peculiarities of task execution were investigated. The methods and tools for development, as well as the implementation of the chatbot for the sushi delivery store, were substantiated.

A concept for developing a chatbot for sushi delivery was proposed to facilitate convenient order placement by customers of the store. The project utilized the Python programming language and the python-telegram-bot library for interaction with the Telegram API.

Keywords: chatbot, functional model, development methods, database.

ЗМІСТ

ВСТУП

У сучасному світі все більше людей користуються послугами онлайн-шопінгу та доставки. Це спричинено зростанням швидкості та доступності Інтернету, а також зручністю та ефективністю цього способу покупок. Однак, разом зі зростанням популярності онлайн-шопінгу, зростає і кількість клієнтів, які звертаються до служб доставки.

У зв'язку з цим, компанії, що надають послуги доставки, намагаються покращити якість своїх послуг. Одним із засобів покращення цих послуг є використання чат-ботів. Чат-боти дозволяють підтримувати зв'язок з клієнтами, швидко та ефективно відповідати на запитання та допомагати з вирішенням проблем.

Актуальність полягає в тому що чат-боти є ефективним інструментом взаємодії з клієнтами, який дозволяє швидко та ефективно відповідати на запитання та допомагати з вирішенням проблем. Використання чат-ботів дозволяє покращити якість послуг та привести до збільшення лояльності та задоволеності клієнтів, а отже, збільшення прибутку компанії.

Крім того, розробка чат-боту для магазину доставки може бути вигідною з погляду оптимізації бізнес-процесів. Чат-бот може автоматизувати багато процесів, зменшити кількість запитів до підтримки клієнтів та забезпечити швидку та ефективну обробку замовлень.

Мета та завдання дослідження. Метою є розробка чат-боту для магазину доставки, який буде забезпечувати клієнтам максимальний комфорт та зручність у процесі замовлення та отримання товарів. Для досягнення цієї мети, будуть проведені дослідження та аналіз ринку чат-ботів, розроблений та реалізований чат-бот для магазину доставки, проведені експерименти та оцінки результатів роботи чат-боту.

Зважаючи на поставлену мету, об’єкт та предмет ми визначили наступні завдання для нашого дослідження:

1. Провести загальну характеристику чат-ботів.
2. Вивчити сучасні методи та технології розробки чат-ботів.
3. Розробити модель для чат-боту.
4. Зібрати всю необхідну інформацію на основі якої буде працювати чат-бот.
5. Створити чат-боту за допомогою обраних методів та технологій.
6. Провести тестування чат-бота
7. Описати рекомендації з розробки чат-ботів

Об'єктом дослідження процеси розробки чат-боту магазину доставки суші.

Предмет дослідження методи і технологія розробки чат-боту магазину доставки суші.

Методи дослідження:

- Загальнонауковий аналітичний метод: В розділі 1-2 дослідження проводилось аналізування теоретичних основ та підходів до розробки чат-ботів. Застосування цього методу дозволило узагальнити і систематизувати наявну наукову літературу та практичний досвід у галузі чат-ботів.
- Методи теорії баз даних: Для формування інформаційно-логічної моделі предметної області та бази даних використовувалися методи теорії баз даних. Ці методи допомагали структурувати та організовувати дані про користувачів, замовлення та інші деталі, необхідні для роботи чат-бота.
- Методи алгоритмічного програмування: Для реалізації автоматизованої системи моніторингу соціально-економічного розвитку були використані методи алгоритмічного програмування. Ці методи дозволяли розробити функціональні та надійні програми, що забезпечували взаємодію з Telegram API та опрацювання замовлень.

Практична цінність: Результати цієї роботи можуть бути використані для подальшого розвитку та вдосконалення чат-бота, а також знайти застосування у сфері ресторанного бізнесу та доставки їжі. Використання чат-ботів у сфері бізнесу дозволяє покращити обслуговування клієнтів, автоматизувати процеси та підвищити ефективність роботи компаній.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 18 найменувань, додатків і містить 46 сторінки основного тексту, 47 рисунків і 2 таблиці.

РОЗДІЛ 1.

ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальна характеристика чат-ботів

1.1.1 Визначення чат-бота

Чат-бот - це програма, яка являє собою імітування поведінки людини під час спілкування з іншим користувачем через текстовий інтерфейс Термін «чатер-бот» (англ. ChatterBot) використав Майкл Маулдін у 1994 році щоб описати розмовні програми, які дозволяють спілкування між людиною і чат-ботом з використанням штучного інтелекту. Ця технологія набула великої популярності в останні роки, оскільки вона надає можливість автоматизувати комунікацію з користувачами та забезпечити їм зручний доступ до інформації та послуг. Однією з основних переваг чат-ботів є те, що вони можуть працювати цілодобово та надавати швидкі та точні відповіді на запитання користувачів. [1]. Вони також дозволяють знизити витрати на зв'язок з клієнтами та покращити якість обслуговування.

Чат-боти можуть бути використані в різних сферах життя, таких як бізнес, медицина, освіта, розваги та багато інших. Вони можуть слугувати для виконання різних завдань, включаючи обробку замовлень, відповіді на запитання користувачів, надання рекомендацій та порад, розкриття новин та подій, тестування знань та багато іншого[2]. Основні застосування чат-ботів:

Рис.1.1 Застосування чат ботів

1.1.2 Класифікація чат-ботів

У найбільш актуальній літературі чат-боти поділяють за різними характеристиками, але є основні пункти за якими можна їх класифікувати. Дослідження та аналіз ознак сучасних чат-ботів розділили програми чат-ботів на сім класів.[3]

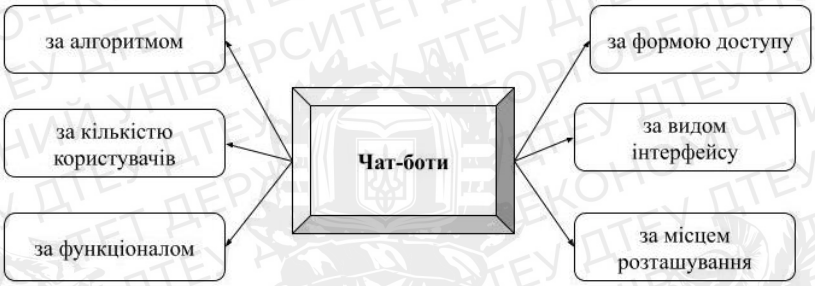


Рис 1.2 Класифікація чат-ботів

За алгоритмом

- Прості чат-боти характеризуються тим, що діють за певним планом, який включає процеси зі списку, залежно від запланованої поведінки користувача. Діалог у таких ботах представляє заздалегідь створений шаблон, а скрипт – це дерево рішень, у якому відповідь на питання відкриває запрограмований сценарій.
- Складні або інтелектуальні чат-боти базуються на штучному інтелекті. Їх програми здатні самостійно навчатися у процесі спілкування. Тому складні боти можуть виконувати більше завдань, пов'язаних не лише з розпізнаванням сенсу тексту, але і фото, відео та аудіо файлів.

За функціоналом

- Інформаційно-комунікаційні, тобто з самої назви можна зрозуміти що це чат-боти які можуть застосовуватись, наприклад, як розсилка певної інформації або ж за допомогою них можна дізнатись загальну інформацію про компанію, послугу або товар. Такі чат-боти є дуже вигідними для компаній і дуже часто використовуються як частина маркетингу, бо так можна повідомити користувача про пропозиції які можуть його зацікавити.

Та в основному за даними досліджень [4-6] приблизно від 85-90% користувачів, відкриває додаток месенджеру впродовж трьох хвилин після отримання повідомлення, що є дуже високим показником.

– Функціональні – це вже більш розвинений вид чат-ботів які можуть виконувати певні дії, також для таких чат-ботів може використовуватись штучний інтелект, який може проводити певні аналізи для покращення роботи системи. Технологія, керована ШІ, перетворює складні проблеми в спрощені рішення. Сучасні платформи чат-ботів мають можливості повністю відтворювати функціонал додатків: створювати, шукати, замовляти, бронювати і т.п. [7]

За кількістю користувачів

– Персональні чат-боти - це боти, що взаємодіють з конкретним користувачем і відповідають на його запити в персоналізованому форматі. Загалом такі чат-боти використовують як особистий помічник користувача, це можуть бути персональні сховища даних, наприклад, чат-боти для зберігання своїх власних зроблених фото або для формування і зберігання особистих записів та файлів користувача.

– Комерційні. Цей вид чат-бота використовується для автоматизації в спілкуванні з клієнтами та виконанні додаткових функцій. Даний тип чат-ботів є найпоширенішим оскільки це швидкий доступ до багатьох можливостей таких як оформлення замовлень, пошук інформації, бронювання квитків, тощо.

За місцем розташування

– На сайтах. Багато компаній вбудовують чат-боти на свої сайти, щоб користувач зміг користуватись його функціями без переходу на інші платформи.

– У месенджерах. Цей вид чат-бота може виконувати набагато більше функцій ніж чат-боти на сайтах. Для прикладу це може бути Slack, Viber, Facebook Messenger або Telegram.

За видом інтерфейсу

– Кнопковий. Кнопковий вид інтерфейсу чат-бота - це спосіб взаємодії з користувачем, коли йому пропонуються кнопки з певними варіантами відповідей.[8-9] Користувач може вибрати одну з цих кнопок, яка відповідає його запиту або питанню. Цей вид інтерфейсу часто використовується для надання швидкої та простої взаємодії з користувачем. Кнопки можуть мати текстову інформацію або іконки, що їх символізують. Іноді кнопки також можуть бути організовані у вигляді меню або списку. Перевагою кнопкового інтерфейсу є те, що він зменшує можливість помилкових розумінь та дозволяє користувачам швидко взаємодіяти з чат-ботом без необхідності вводити текстові повідомлення. Однак, кнопковий інтерфейс може бути обмежувальним у випадках, коли користувачі хочуть задати складніші або неочікувані запити, які не передбачені у списку кнопок.
– Текстовий вид інтерфейсу чат-бота - це спосіб взаємодії з користувачем, коли він може вводити свої запити або питання за допомогою текстового поля. Він дає користувачам більшу свободу у вираженні своїх індивідуальних потреб і питань, які можуть не бути передбаченими у вигляді кнопок.[8-9] Цей вид інтерфейсу може бути корисним у випадках, коли користувачі мають запити, які не передбачені у списку кнопок, або коли вони шукають більш детальну відповідь на своє питання. Він також може допомогти користувачам з різними мовними знаннями, оскільки вони можуть виразити свої запити у своїй власній мові. Однак, текстовий інтерфейс може бути менш простим та більш часовим для взаємодії з користувачем порівняно з кнопковим інтерфейсом. Також він може бути менш ефективним, якщо користувачі не знають, як сформулювати свої запити або питання правильно.

За формою доступу

– Публічні чат-боти - це боти, які доступні для використання всіма користувачами через певну платформу або канал зв'язку. Такі боти можуть взаємодіяти з користувачами, надаючи різноманітні послуги, такі як підтримка клієнтів, продаж товарів, надання інформації тощо.
– Приватні чат-боти - це програми, які створюються для внутрішнього використання в компанії або організації. Основна перевага приватних чат-ботів полягає у тому, що вони дозволяють полегшити та автоматизувати багато рутинних задач, що раніше вимагали багато часу та зусиль від співробітників.[10] Крім того, використання приватних чат-ботів може знизити кількість помилок, пов'язаних з людським фактором, і покращити ефективність внутрішньої комунікації та співпраці між співробітниками.

1.2 Огляд технологій створення чат-ботів

Загалом існує дві основних технології для створення чат-ботів:

- Створення чата в конструкторі
- Створення чата за допомогою мов програмування.

Створення чат боту в конструктор являє собою онлайн-інструмент, який дозволяє користувачам створювати та налаштовувати чат-ботів без необхідності в програмуванні або використовуючи мінімальні знання програмування. Вони часто використовуються в комерційних цілях, таких як клієнтський сервіс або маркетингові кампанії.

1.2.1 Конструктори для створення чат-ботів

Конструктори для створення чат-ботів зазвичай пропонують широкий спектр інструментів для налаштування бота, включаючи настройки повідомлень, розкладу, скриптів, інтеграцій з зовнішніми сервісами, аналітики та багато іншого. Деякі з популярних конструкторів для створення чат-ботів включають[11] :

1. Chatfuel - це безкоштовний конструктор для створення чат-ботів на основі Facebook Messenger. Він надає можливість створювати ботів за допомогою перетягування та опускання блоків, що дозволяє створювати складні

сценарії без програмування. Chatfuel також має функцію AI, яка дозволяє боту розуміти та відповідати на запити користувачів з використанням природної мови.

2. Botkit є не просто конструктором чат-ботів, а цілою спільнота більше ніж з 7000 розробників з усього світу. Однак, це також вражаючий конструктор чат-ботів, який включає візуальний конструктор розмов, бібліотеки з відкритим вихідним кодом, вбудовані статистики та метрики, а також написаний з високою якістю код від кращих розробників чат-ботів, який ви можете використовувати для створення власних ботів. Особливостями платформи є можливості встановлення плагінів та власних модулів, що дозволяє розробникам легко розширювати функціональність своїх чат-ботів та автоматичне зберігання стану розмови, маршрутизація повідомлень, створення коротких відповідей (quick replies), та інші.
3. Tars - це конструктор для створення чат-ботів, який дозволяє користувачам створювати чат-ботів без програмування. Tars пропонує інтуїтивний візуальний інтерфейс, який дозволяє легко налаштовувати чат-бота та інтегрувати його з різними зовнішніми сервісами. Крім того, Tars надає користувачам можливість налаштувати різні варіанти відповідей, щоб забезпечити персоналізований досвід для користувачів.

   Також бля більш складних чат-ботів, Tars дозволяє користувачам налаштовувати кодову базу, щоб забезпечити більш гнучкий та налаштований досвід.
4. Напевно найпопулярніший конструктором є Dialogflow.

   DialogFlow - це конструктор для створення чат-ботів, розроблений компанією Google. Конструктор дозволяє розробникам створювати чат-ботів з використанням штучного інтелекту, щоб забезпечити більш гнучкий та налаштований досвід для користувачів. Розробники можуть налаштовувати агента, який буде взаємодіяти з користувачем, за допомогою мовних моделей та машинного навчання без необхідності використання спеціальних команд.

1.2.2 Мови програмування та їх порівняння

Створення чат-бота за допомогою мов програмування полягає у написанні програмного коду, який буде обробляти запити користувачів та відповідати на них. Даний підхід є більш технічно складним, але дозволяє розробникам вносити будь які корективи в роботу бота, тобто бот створюється з нуля самостійно та є повністю під лаштований під вимоги до функціональності замовника.

Розглянемо більш детально найпоширеніші мови програмування для створення чат-ботів.

Мова програмування Python є найбільш популярною для машинного навчання. Елегантний синтаксис Python, динамічна обробка типів, сумісність з багатьма іншими мовами, кількість бібліотек для аналізу даних та побудови моделей а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.[12] Заразом можна зазначити що написаний код не переводиться автоматично в машино-зчитуваний формат, в той же час більшість інших мов програмування роблять це до того як програма була запущена.

JavaScript ще одна мова дуже популярна програмування. Але спочатку було досить скептичне відношення до мови, проте з швидким розвитком технологій ситуація змінилась, що дозволило мові зайняти серйозне місце серед інших. В результаті, було розроблено та покращено різноманіття шляхів використання JavaScript, створені фреймворки та бібліотеки, використання мови поширилося поза браузером.

JavaScript має властивості об'єктно-орієнтованої мови, але підтримка об'єктів різниться від інших мов через концепції прототипів. Окрім цього, мова має певні властивостей, характерних функціональним мовам.[13-14]

Написання чат-бота на мові програмування C++ може бути складнішим, ніж на деяких інших мовах програмування, таких як Python або JavaScript, оскільки C++ є мовою програмування високого рівня і має складну синтаксичну структуру. Вона підтримує багато різних бібліотек і фреймворків, що можуть

бути використані для створення чат-боту. Одним з популярних фреймворків для розробки чат-ботів на C++ є BotBuilder, який розробляється компанією Microsoft. BotBuilder надає API для побудови інтерактивних ботів на різних платформах. Бібліотека містить ряд готових елементів, таких як введення тексту, кнопки, картинки і т. д., які можна використовувати для створення відповідей бота на повідомлення від користувача.[15] Також для написання чат-бота на C++ можна використовувати бібліотеки та фреймворки, такі як Qt або Boost. Ці бібліотеки містять набір інструментів, які допомагають спростити процес розробки, зокрема роботу з мережевими з'єднаннями та обробкою даних.

Таблиця 1.1 Порівняння мов програмування

	Python	C++	JavaScript
Наявність механізмів підтримки ООП	Повна підтримка ООП	Повна підтримка ООП	Повна підтримка ООП
Синтаксис	високорівневий, з динамічною типізацією	низькорівневий, з статичною типізацією	високорівневий, з динамічною типізацією
Продуктивність	Нижче, порівняно з C++	Висока, близька до мов асемблера	Нижче, порівняно з Python та C++
Стандартні бібліотеки	Широкий вибір, включаючи бібліотеку для створення чат-ботів - PyTorch	Обмежений вибір, можна використовувати сторонні бібліотеки, наприклад, Botkit	Широкий вибір, включаючи бібліотеку BotUI для створення чат-ботів
Рівень абстракції	Дуже високий рівень абстракції, дозволяє розробникам не	Нижчий рівень абстракції, вимагає більше коду	Також має високий рівень

	звертати увагу на деталі		
Управління пам'яттю	Автоматичне управління пам'яттю, що зменшує кількість помилок, але може знизити продуктивність	Ручне управління пам'яттю, що дає більший контроль над ресурсами, але може призвести до багатьох помилок	Автоматичне управління так само як і на Phyton

1.2.3 Найпоширеніші платформи для чат-ботів

Важливим моментом для створення чат-боту також є вибір платформи для реалізації. Існують різні платформи для створення чат-ботів, серед яких популярні WhatsApp, Instagram, Telegram та Facebook Messenger. Кожна з цих платформ має свої особливості та можливості.

- WhatsApp - месенджер, який дозволяє користувачам обмінюватися повідомленнями, файлами та відео. Однак, на відміну від інших платформ, WhatsApp ще не має офіційної підтримки для створення чат-ботів через API. Проте, WhatsApp Business API надає можливість бізнесам створювати чат-ботів для спілкування з клієнтами, але створення чат-бота на WhatsApp може бути більш складним процесом порівняно з іншими платформами через такі обмеження.[16]
- Instagram - соціальна мережа для обміну фотографіями та відео. Для створення чат-бота на Instagram необхідно використовувати Instagram Graph API, який також є платним. На даній платформі можна створювати тільки прості чат-боти, є певні обмеженя, наприклад, обмеження на кількість запитів, які можна виконувати до їх API, що може значно обмежити можливості розробки чат-бота.
- Telegram - месенджер зі широким функціоналом та відкритим API для створення чат-ботів. Розробники можуть використовувати Telegram Bot

API для створення чат-ботів на цій платформі. Це API надає широкі можливості для створення різних типів чат-ботів зі зручним інтерфейсом.

- Facebook Messenger - одна з найпопулярніших платформ для створення чат-ботів. Розробники можуть використовувати Facebook Messenger Platform API для створення чат-ботів на цій платформі. Це API надає розширені можливості для створення різних типів чат-ботів зі зручним інтерфейсом, включаючи можливість роботи зі зображеннями, відео та аудіофайлами. Facebook Messenger Platform підтримує мови програмування, такі як Node.js, Python, PHP, Java, Ruby та інші, що дозволяє розробникам використовувати ту мову програмування, яка їм більше подобається. Зроблена статистика в 2018 році, згідно якої в Facebook Messenger працює понад 300 000 чат-ботів. [17]
- Ще однією цікавою платформою є Slack.

  Slack - це платформа для комунікації в робочому середовищі, яка підтримує створення чат-ботів. Для розробки чат-бота використовуються різні мови програмування.Slack має вбудовані інструменти для створення і налаштування чат-бота, такі як Botkit та Slack API. Botkit - це фреймворк для створення чат-ботів на різних платформах, включаючи Slack. Особливістю є можливість інтеграції з різними інструментами, такими як GitHub, Jira, Trello та інші. Це дозволяє чат-ботам автоматизувати багато робочих процесів та забезпечити більш ефективну комунікацію в команді. Однак, на відміну від деяких інших платформ, Slack не є так популярним серед звичайних користувачів, що може обмежити масштаби розповсюдження чат-бота на цій платформі.

У кожної з цих платформ є свої особливості та обмеження, які необхідно враховувати при виборі платформи для створення чат-бота. Наприклад, для роботи з WhatsApp та Instagram потрібно платити за використання API, тоді як Telegram та Facebook Messenger надають безкоштовне API. Однак, Facebook Messenger може мати обмеження на кількість повідомлень, які можна відправляти протягом певного часу.

РОЗДІЛ 2.
МОДЕЛЬ ЧАТ-БОТУ МАГАЗИНУ ДОСТАВКИ СУШІ

2.1 Модель чат-боту магазину доставки суші

В межах розробленого програмного продукту будуть взаємодіяти такі актори:

1. Користувач –клієнт, який бажає зробити замовлення.
2. Адміністратор – представник магазину

Опис прецедентів їх завдань наведений у таблиці нижче

Таблиця 2.1 Опис можливостей акторів

Користувач	• Огляд меню та акцій • Оформлення нового замовлення • Зв’язок з менеджером • Отримання інформації про замовлення та його вартість • Перегляд особистої інформаціїї
Адміністратор	• Перегляд списку клієнтів • Додавання категорій та страв • Редагування категорій та страв • Огляд меню та акцій • Оформлення нового замовлення • Отримання інформації про замовлення та його вартість

Опис діаграми прецедентів:

Спочатку розглянемо можливості Користувача в даному чат-боті. Користувач має можливість перегляду меню для вибору страви, оформлення замовлення та огляду інформації щодо замовлення. Окрім цього, додатковою функцією є можливість зв'язку з менеджером магазину для уточнення інформації або ж зробити замовлення поза ботом.

Адміністратор має можливість додавати нові категорії та страви до бази даних, а також редагувати або видаляти вже наявні. Він також спроможний переглядати інформацію щодо замовлень, дані про користувачів, оформляти розділ акційних товарів.

Так само як і Користувач, Адміністратор, також може переглядати меню та акції. Далі, за наявності доступних страв, обидва типи акторів можуть оформити на ім'я клієнта замовлення страв які є в наявності, та в подальшому отримати інформацію про обраховану вартість. Можливості Адміністратора схожі з користувацькими коли, він не авторизувався як Адміністратор, в цей момент має ту саму панель команд що Користувач, тобто обмежену кількість доступних функцій. Після того як Адміністратор перейде в режим адміністрування чат-боту він зможе використувати також додаткові команди, які будуть доступні для цього виду актора.

На рисунку 2.1 зображено модель можливостей обох акторів, разом з урахуванням того, що адміністратор, також може виступати в ролі Користувача.

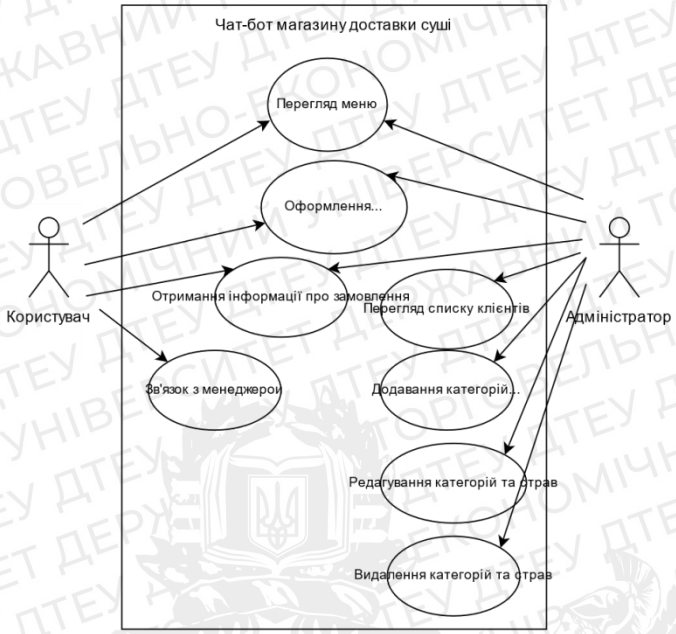


Рис. 2.1 Діаграма прецедентів

Опис деталізованої діаграми прецедентів

При запуску чат-боту Користувачу будуть доступні чотири функції – «Корзина», «Меню», «Акції», «Менеджер».

У стандартному сценарії Користувача він запускає діалог із чат-ботом та з доступних опцій. Він не потребуватиме реєстрації, чат-боту достатньо знати внутрішній ідентифікатор клієнта в месенджері для ведення діалогу. Потім і клієнт і адміністратор має можливість переглянути меню та сформувати замовлення, при цьому процесі потрібно тільки підтвердження системи для виконання цих функцій. Також користувач може зв’язатись з менеджером, для цього мережа дає перенаправлення на контакт менеджера.

Адміністратор має можливість додати нові страви до бази даних, при цьому процесі проводиться перевірка введених даних, та, якщо коректність підтвердилась, створюється новий запис в таблиці страв бази даних. Також наявна можливість редагувати додані записи про страви, при чому теж, перед внесенням змін до таблиці, проводиться перевірка вводу. Процес видалення супроводжується підтвердженням дії, щоб запобігти небажаних змін важливих

даних. Додатково адміністратор має змогу переглядати списки клієнтів, в цьому процесі потрібне тільки підтвердження дії системою

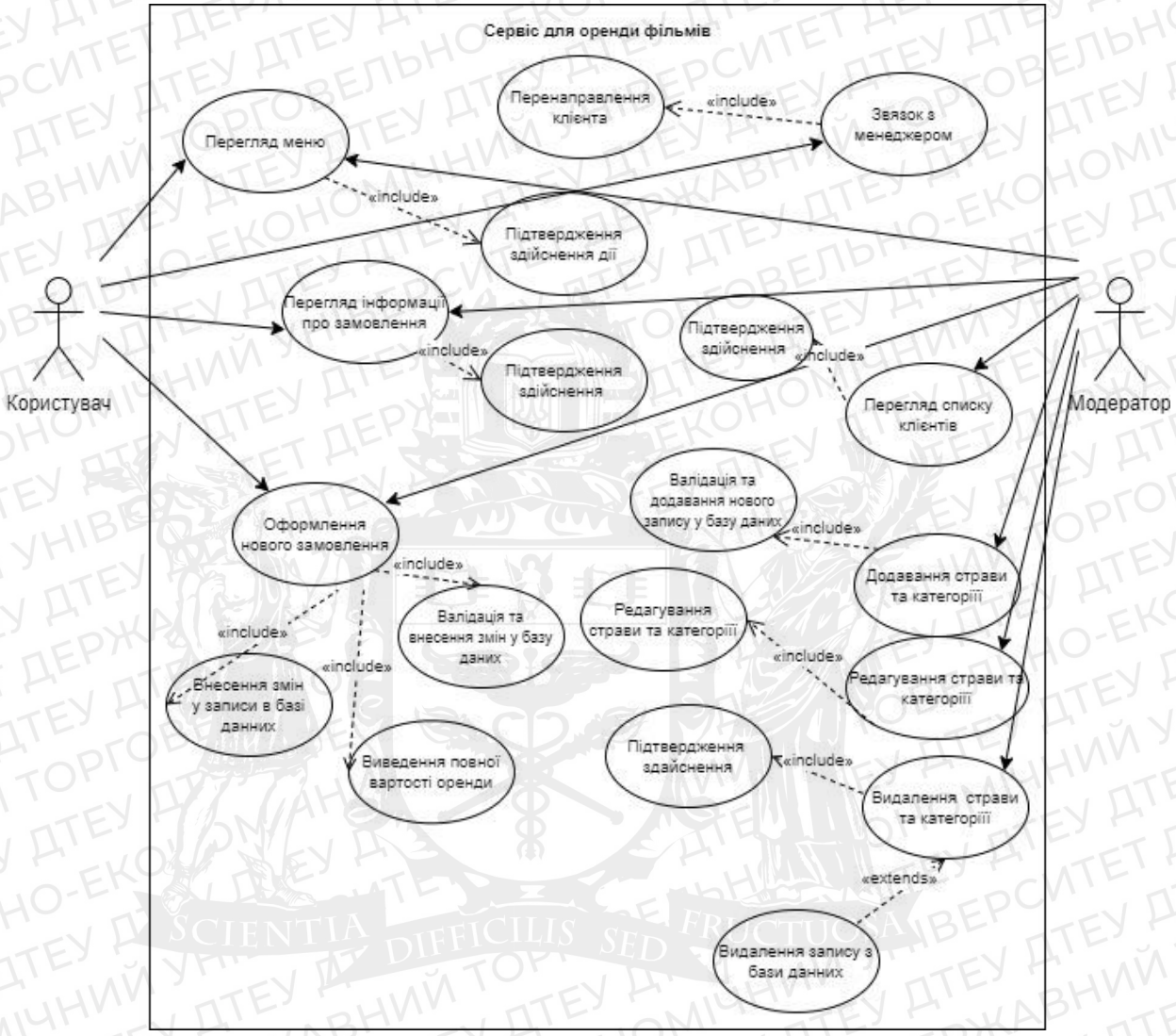


Рис 2.2 Детальна діаграма прецендентів

2.2 Методи та технології розробки

Для розробки чат боту було обрано так засоби реалізації як:

В якості платформи реалізації вибрано месенджер Telegram, тому що він є найбільш популярним месенджером серед українців на сьогоднішній день. Платформа має інтерфейс (Telegram Bot API) на основі HTTP, створений для розробників, які зацікавлені в створенні ботів для Telegram. Telegram Bot API надає багато різних методів для взаємодії з ботами. Розробники можуть використовувати ці методи для відправки повідомлень, отримання оновлень, редагування повідомлень, керування клавіатурами та багато іншого. Деякі з

популярних методів API включають sendMessage, editMessageText, getUpdates. Зручним методом також можна створювати кнопкові чат боти з використанням інтерактивної клавіатури з кнопками для навігації та взаємодії з ботом. Основною бібліотекою для розробки було використано telegram-bot-api яка також підтримує можливість використання локального API серверу. Перевагами використання локального API серверу є можливості скачувати файли будь яких розмірів, використовуючи їх локальний шлях і схему URI файлу також використання будь якої локальної IP-адреси та порту для webhook.[18] Платформа надає повну документацію та розширені можливості. Це дозволяє легко розібратися в функціональності Telegram Bot API та ефективно використовувати його для своїх потреб.

Мова програмування для розробки боту було обрано Python з використанням програмного середовища Pycharm. Python є дуже зручною для розробки боту так як має легкий для розуміння синтаксис, підтримує велику кількість функцій та бібліотек, як наприклад python-telegram-bot, яка є дуже потужною бібліотекою. Також потрібно зауважити що Python це інтерпретована, що дозволяє значно полегшити процес розробки та відлагодження програм.

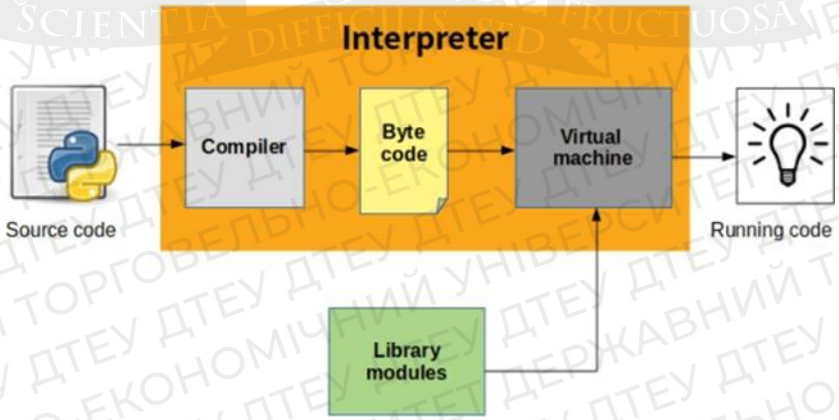


Рис 2.3 Схема роботи інтерпертатора

SQlite було обрано для створення бази даних так як Python має вбудовану підтримку для роботи з базою даних SQLite за допомогою модуля sqlite3. Це дозволяє легко взаємодіяти з базою даних, виконувати запити SQL, створювати таблиці, вставляти, оновлювати та видаляти дані. База даних SQLite має невеликий розмір, що робить її ідеальним вибором для невеликих проектів або

проектів з обмеженими ресурсами. Вона також працює швидко та ефективно, що дозволяє ефективно обробляти запити до бази даних у реальному часі.

2.3 Збір бази даних для чат-боту

Чат-бот для магазину доставки суші матиме базу даних, яка буде записана в файлі з розширенням «.db» та носитиме назву «sushi_bot_db.db». Створити базу даних можна було в застосунку SQlite Studio та через код програми здійснити під'єднання даного файлу до чат-боту. Також одним із шляхів створення бази даних є можливість виконання методом здійснення запитів в коді програми.

Для початку потрібно було розробити структуру для бази даних, тобто таблиці з даними які будуть використовуватись при різних сценаріях, наприклад здійснення замовлення.

Таким чином структура має ось такий вигляд:

Рис 2.4 Архітектура бази даних

Структура включає сім таблиць.

1. Таблиця "cart": використовується для збереження інформації про товари, які знаходяться в кошику користувача, включаючи зв'язки з конкретним користувачем та суші, а також кількість одиниць суші в кошику.

2. Таблиця "order_items": використовується для збереження інформації про елементи замовлення, включаючи зв'язки з конкретними замовленнями та суші, а також кількість замовлених одиниць суші.

3. Таблиця "orders": використовується для збереження інформації про замовлення, включаючи користувача, який зробив замовлення, та час його оформлення.

4. Таблиця "users": використовується для збереження інформації про користувачів, які можуть реєструватися, розміщувати замовлення та зберігати товари у кошику.

5. Таблиця "promotions": використовується для збереження інформації про акції або спеціальні пропозиції, які можуть бути пов'язані з суші.

6. Таблиця "sushi": Ця таблиця використовується для збереження інформації про конкретні види суші та їх атрибути, такі як категорія, назва, склад, вага/кількість та ціна.

7. Таблиця "sushi_categories": використовується для збереження інформації про категорії суші.

Однією з найважливіших таблиць є Таблиця "sushi" та як в ній зібрана загальна інформація щодо позицій, які потім будуть відображатись для користувача якщо він натисне розділ «Меню». Вона служить для створення карток товару в базі даних та вміщає в собі таку інформацію як: «Назва» , «Склад позиції», «Вага» та «Ціна».

id	category_id	name	ingredients	weight_or_quantity	price
Філ...	Фільтр	Фільтр	Фільтр	Фільтр	Філ...
1	1	сет Макі	макі з огірком ½, макі з ...	550 г (40шт)	265
2	1	сет Сімейний	макі з лососем½ , макі філа ...	1210 г (48шт)	585
3	1	сет Філадельфія	філадельфія з лососем, ...	1560 г (48шт)	775
4	1	сет Едо	макі філа з лососем½ , макі ...	870 г (32шт)	479
5	1	сет Філа Хіт	Філадельфія з лососем, ...	1035 г (32шт)	499

Рис 2.4 Приклад наповнення таблиці «sushi»

РОЗДІЛ 3.
ПРОГРАМНА РЕАЛІЗАЦІЯ ЧАТ-БОТУ МАГАЗИНУ ДОСТАВКИ СУШІ

3.1. Перевірка чат-боту магазину доставки суші

3.1.1. Опис функціональних вимог до чат-боту

Функціональні вимоги визначають основні можливості та функції, які повинен виконувати чат-бот магазину доставки суші. Нижче наведено опис основних функціональних вимог:

Збереження персональних даних користувача:

- Система повинна надавати можливість користувачам зареєструватися в системі шляхом введення необхідних персональних даних.
- Після успішної реєстрації користувач повинен мати можливість оформлювати замовлення з вже збереженими даними замовника.

Замовлення суші:

- Користувач повинен мати можливість складати замовлення на суші через чат-бот.
- Система повинна забезпечувати можливість вибору страв і вказання їх кількості.
- Користувач повинен мати можливість вказати адресу доставки та контакті дані.

Відстеження замовлення:

- Система повинна надавати актуальну інформацію про підтвердження замовлення, підготовку страв, відправлення та доставку.

Каталог страв:

- Система повинна надавати користувачам можливість переглядати каталог суші з описом та цінами на страви.
- Користувач повинен мати можливість шукати страви за категоріями.

Підтримка клієнтів:

- Система повинна забезпечувати можливість користувачам звернутися до служби підтримки чат-боту для отримання допомоги або вирішення проблем.

Ці функціональні вимоги є основними і визначають найголовніший функціонал чат-боту магазину доставки суші. Реалізація цих функцій дозволить забезпечити користувачам зручний та ефективний спосіб здійснення замовлень та отримання суші.

3.1.2. Опис нефункціональних вимог до чат-боту

Нефункціональні вимоги визначають якостеві характеристики та вимоги, що не стосуються безпосередньо функціональності чат-боту, але визначають його ефективність, безпеку, масштабованість та інші аспекти. Нижче наведено опис основних нефункціональних вимог:

Швидкодія:

- Чат-бот повинен забезпечувати швидку відповідь на запити користувачів.
- Затримка між запитом користувача та отриманням відповіді в системі повинна бути мінімальною.

Масштабованість:

- Система чат-боту повинна бути масштабованою та здатною обробляти одночасні запити великої кількості користувачів.
- При збільшенні навантаження на систему, чат-бот повинен продовжувати працювати без втрати продуктивності.

Надійність:

- Чат-бот повинен бути стійким до помилок та збоїв.
- В разі виникнення помилок, система повинна надавати зрозумілі повідомлення про помилку та відновлювати свою працездатність.

Безпека:

- Система повинна забезпечувати безпеку персональних даних користувачів.
- Комунікація між користувачем та чат-ботом повинна бути захищеною та шифрованою.

Сумісність:

- Чат-бот повинен бути сумісним з різними платформами та пристроями, такими як комп'ютери, смартфони тощо.

- Він повинен працювати на різних операційних системах та веб-браузерах.

Користувацький інтерфейс:

- Інтерфейс чат-боту повинен бути інтуїтивно зрозумілим для користувачів.
- Він повинен мати зручну навігацію та забезпечувати зручну взаємодію з користувачами.

Розширюваність:

- Система повинна бути розширюваною та здатною до додавання нових функціональностей та модулів без впливу на роботу існуючих компонентів.

Ці нефункціональні вимоги визначають додаткові аспекти, які мають значення для ефективної та надійної роботи чат-боту магазину доставки суші. Врахування цих вимог у розробці допоможе створити потужний та задовільний з точки зору функціональності продукт.

3.1.3. Архітектура програмного забезпечення чат-боту

Архітектура програмного забезпечення чат-боту магазину доставки суші може бути організована з використанням чотирьох основних файлів: db_module.py, kb_module.py, btn_cb_module.py та main.py. Кожен файл відповідає за виконання певних функцій і взаємодію з іншими компонентами системи. Описані нижче компоненти та їхні взаємозв'язки формують архітектуру програмного забезпечення чат-боту:

db_module.py:

- Цей модуль відповідає за збереження та керування базою даних.
- Він містить функції для збереження і отримання інформації про користувачів, замовлення та інші необхідні дані.
- Модуль використовує підключення до бази даних та запити SQL для взаємодії з базою даних.

kb_module.py:

- Цей модуль відповідає за обробку запитів користувачів та взаємодію з базою знань (knowledge base).
- Він містить функції для аналізу та інтерпретації запитів користувачів, виявлення намірів та виконання відповідних дій.

-Модуль також використовує функції з db_module.py для доступу до необхідних даних з бази даних.

btn_cb_module.py:

-Цей модуль відповідає за обробку подій, пов'язаних з кнопками та інтерфейсом користувача.

-Він містить функції для визначення дій, які повинні бути виконані при натисканні на кнопки в чат-боті.

-Модуль також взаємодіє з іншими компонентами системи для отримання необхідних даних та виконання відповідних операцій.

main.py:

-Цей головний файл відповідає за керування загальним потоком роботи чат-боту.

-Він ініціалізує та запускає інші модулі, обробляє вхідні запити користувачів, взаємодіє з інтерфейсом та забезпечує координацію роботи всіх компонентів системи.

Ця архітектура дозволяє розділити функціональність чат-боту на окремі модулі, що полегшує розробку, розширення та підтримку системи. Кожен модуль виконує певну роль і взаємодіє з іншими модулями для забезпечення повного функціоналу чат-боту магазину доставки суші.

3.1.4. Опис розроблених алгоритмів для обробки запитів в чат-боті

У чат-боті магазину доставки суші були розроблені алгоритми для ефективної обробки запитів користувачів. Нижче наведено опис декількох таких алгоритмів:

1. Розпізнавання наміру (Intent Recognition):

- Цей алгоритм використовується для визначення наміру або цілі запиту користувача на основі вхідного тексту.

- Намір визначається шляхом порівняння вхідного тексту з попередньо заведеними командами

У нашому проекті це реалізовано через розгалуження блоків if у файлі main.py. Згідно з отриманим тектом команди викликається логіка завдання у взаємодії, за необхідності, з іншими модулями. Прикладом є метод button_callback(update: Update, context: CallbackContext).

2. Виділення сутностей (Named Entity Recognition):

• Цей алгоритм використовується для виявлення та виділення важливих сутностей з вхідного тексту, таких як імена, адреси, час, суми тощо.

• Він може базуватися на правилах або регулярних виразах.

• Виділені сутності можуть використовуватись для подальшої обробки запиту, наприклад, для здійснення замовлення або пошуку інформації.

У нашому проекті це реалізовано через набір правил у вводі персональних даних користувача. Прикладом є введення номеру телефону у секцію особистих даних у методі text_handler(update: Update, context: CallbackContext)

```
def text_handler(update: Update, context: CallbackContext):
    if context.user_data["action"] == "awaiting_phone":
        phone = update.message.text
        if not phone.startswith("+380"):
            update.message.reply_text("Невірний формат телефону. Спробуйте ще раз.")
            return
        user_id = update.message.from_user.id
        db_module.change_user_phone(user_id, phone)
        update.message.reply_text("Телефон змінено!")
        context.user_data.pop("action")
```

Рис 3.1 Приклад введення номеру телефону

3. Логіка розгалуження (Branching Logic):

• Цей алгоритм використовується для визначення логіки розгалуження в залежності від наміру та інших факторів.

•Він визначає, які дії повинні бути виконані відповідно до запиту користувача.

•Можуть бути використані умовні оператори або правила для прийняття рішення про наступні дії чат-боту.

Це реалізоване через набір умовних операторів, які викликають відповідні методи, задля здійснення завдань, що боту поставив користувач, у методі text_handler(update: Update, context: CallbackContext) та інших.

3.1.5. Опис реалізованих функцій чат-боту

У чат-боті магазину доставки суші було реалізовано ряд функцій, що забезпечують зручну взаємодію з користувачами. Нижче наведено опис деяких з цих функцій:

Замовлення суші:

•Користувач може зробити замовлення суші, обравши потрібні страви з меню.

•Функція передає деталі замовлення до системи та підтверджує успішне оформлення замовлення.

У коді за відповідний функціонал відповідають наступні блоки коду:

- Створення клавіатури з відповідними командами для додання до корзини:

```
def build_add_to_cart_keyboard(sushi_id: int) -> InlineKeyboardMarkup:
    button = InlineKeyboardButton("Додати до корзини", callback_data=f"add_to_cart_{sushi_id}")
    keyboard = [[button]]
    return InlineKeyboardMarkup(keyboard)
```

Рис 3.2 Створення клавіатури

- Потім відповідний запит потрапляє через дерево рішень у головному модулі в наступний блок, де відбувається логіка додавання замовлення у поточну відповідну таблицю:

```
if query.data.startswith("add_to_cart_"):
    sushi_id = int(query.data[len("add_to_cart_"):])
    user_id = query.from_user.id

    db_module.add_sushi_to_cart(user_id, sushi_id, 1)
    sushi_object = db_module.get_sushi_by_id(sushi_id)
    sushi_category_id = sushi_object[1]
    sushi_list = db_module.get_sushi_by_category_id(sushi_category_id)
    # send message, not reply
    sushi_keyboard = []
    for sushi in sushi_list:
        sushi_id, _, sushi_name, _, _, sushi_price = sushi
        button_text = f"{sushi_name} - {sushi_price} грн"
        sushi_button = InlineKeyboardButton(button_text, callback_data=f"sushi_{sushi_id}")
        sushi_keyboard.append([sushi_button])
    sushi_keyboard.append([InlineKeyboardButton("Назад", callback_data="back_to_main")])
    keyboard = InlineKeyboardMarkup(sushi_keyboard)
    query.from_user.send_message("Суші додано в корзину", reply_markup=keyboard)
    return
```

Рис 3.3 Додавання замовлення

- Далі відбувається перехід до оформлення замовлення, який керується цим блоком коду. Код проводить необхідні валідації, викликає методи для внесення змін у бд та формує повідомлення про результат спроби оформлення.

```
def callback_checkout(update: Update, context: CallbackContext):
    query = update.callback_query
    user_id: int = query.from_user.id
    user = db_module.get_user_by_id(user_id)
    if user[3] is None or user[4] is None:
        query.message.reply_text("Ви не вказали номер телефону або адресу")
        return
    cart_items: list = db_module.get_user_cart_items(user_id)
    db_module.create_order(user_id, cart_items)
    db_module.clear_user_cart(user_id)
    query.message.reply_text("Замовлення оформлено!")
```

Рис 3.4 Оформлення замовлення

Відстеження замовлення:

Після оформлення замовлення користувач може переглядати історію всіх своїх замовлень з датою створення та найменуванням позиції меню.

У коді за відповідний функціонал відповідають наступні блоки коду:

- Виклик відповідних методів для отримання інформації конкретного користувача з бази даних та вивід результату в чат:

```
def callback_show_user_orders(update: Update, context: CallbackContext, user_id: int):
    query = update.callback_query

    orders = db_module.get_user_orders(user_id)

    keyboard = [
        [InlineKeyboardButton(f"Замовлення #{order[0]}", callback_data=f"show_order_details_{order[0]}")]
        for order in orders
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    query.message.reply_text("Оберіть замовлення:", reply_markup=reply_markup)
```

Рис 3.5 Відстеження замовлення користувача

Каталог товарів:

Користувач може переглядати каталог доступних суші та інших продуктів.

Функція відображає список товарів, їхні ціни та опис, допомагаючи користувачеві зробити вибір.

Адміністратори мають змогу вносити зміни в цей список.

У коді за відповідний функціонал відповідають наступні блоки коду:

- Побудова відповідного меню категорій:

```
def build_categories_keyboard() -> InlineKeyboardMarkup:
    categories = db_module.get_all_sushi_categories()

    buttons = [InlineKeyboardButton(category[1], callback_data=f"menu_category_{category[0]}") for category in
               categories]
    buttons.append(InlineKeyboardButton("◂ Назад", callback_data="back_to_main"))

    keyboard = [buttons[i:i + 2] for i in range(0, len(buttons), 2)]

    return InlineKeyboardMarkup(keyboard)
```

Рис 3.6 Меню Категорій

- Блок обробки запиту та вивід наявних провозицій для відповідної обраної категорії:

```
if query.data.startswith("menu_category_"):
    category_id = int(query.data[len("menu_category_"):])

    sushi_list = db_module.get_sushi_by_category_id(category_id)

    sushi_keyboard = []
    for sushi in sushi_list:
        sushi_id, _, sushi_name, _, _, sushi_price = sushi
        button_text = f"{sushi_name} - {sushi_price} грн"
        sushi_button = InlineKeyboardButton(button_text, callback_data=f"sushi_{sushi_id}")
        sushi_keyboard.append([sushi_button])

    back_button = InlineKeyboardButton("◂ Назад", callback_data="menu")
    sushi_keyboard.append([back_button])

    query.message.reply_text("Обирайте:", reply_markup=InlineKeyboardMarkup(sushi_keyboard))
    return
```

Рис 3.7 Обробка запиту щодо Категорій

- Побудова відповідного меню категорій для адміністрування:

```python
def build_admin_keyboard() -> InlineKeyboardMarkup:
    buttons = [
        InlineKeyboardButton("Користувачі 🧍", callback_data="show_users"),
        InlineKeyboardButton("Категорії суші 📋", callback_data="edit_categories"),
        InlineKeyboardButton("Додати суші🍣", callback_data="add_sushi_item"),
        InlineKeyboardButton("Акції 📣", callback_data="show_admin_promotions"),
        InlineKeyboardButton("Замовлення 🛒", callback_data="admin_show_user_orders"),
        InlineKeyboardButton("Назад ◀", callback_data="back_to_main"),
    ]

    keyboard = [buttons[i:i + 2] for i in range(0, len(buttons), 2)]

    return InlineKeyboardMarkup(keyboard)
```

Рис 3.8 Створення меню категорій

- Побудова відповідного меню категорій для внесення змін у меню категорій:

```python
def build_edit_categories_keyboard() -> InlineKeyboardMarkup:
    categories = db_module.get_all_sushi_categories()

    buttons = [InlineKeyboardButton(category[1], callback_data=f"edit_category_{category[0]}") for category in
               categories]
    buttons.append(InlineKeyboardButton("Створити нову", callback_data="add_sushi_category"))
    buttons.append(InlineKeyboardButton("◀ Назад", callback_data="back_to_main"))

    keyboard = [buttons[i:i + 2] for i in range(0, len(buttons), 2)]

    return InlineKeyboardMarkup(keyboard)
```

Рис 3.9 Внесення змін в меню категорій

- Метод, який дає змогу видалити обрані елементи в категорії:

```python
def callback_edit_category(update: Update, context: CallbackContext, category_id: int):
    keyboard = kb_module.build_edit_sushi_in_category_keyboard(category_id)
    update.callback_query.message.reply_text("Оберіть дію:", reply_markup=keyboard)
```

Рис 3.10 Видалення Категорій

Користувацький супровід:

- Функція надає змогу зв’язатися з менеджером боту.
- У коді за відповідний функціонал відповідають наступні блоки коду:

- Після вибору у меню користувача, згаданому раніше, відповідного пункту, виклик відповідного методу відбудеться в цьому блоці дерева розгалужень

```python
    if query.data == "manager":
        btn_cb_module.callback_send_manager_contact(update, context)
        return
```

Рис 3.11 Виклик менеджера

- Викликаний метод, у свою чергу, сформує повідомлення з контактними даними менеджера:

```
def callback_send_manager_contact(update: Update, context: CallbackContext) -> None:
    query = update.callback_query
    phone_number = MANAGER_PHONE_NUMBER
    first_name = "Менеджер Анастасія"

    query.answer("Контакт менеджера")
    manager_contact = Contact(phone_number=phone_number, first_name=first_name)
    query.message.reply_contact(MANAGER_PHONE_NUMBER, first_name, contact=manager_contact)
```

Рис 3.12 Вивід повідомлення

Ці функції забезпечують зручну та ефективну взаємодію з користувачами та допомагають забезпечити позитивний досвід використання чат-боту магазину доставки суші.

3.1.6 Оцінка користувацького досвіду

У даному підрозділі проводиться оцінка впливу чат-боту на задоволення та задоволення користувачів. Дослідження користувацького досвіду включає збір даних, аналіз відгуків та оцінок користувачів, а також оцінку ефективності, зручності використання та якості обслуговування.

Для оцінки користувацького досвіду можна використовувати наступні методи:

1. Опитування користувачів: Створення анкет або опитувальних форм, щоб зібрати думки, враження та оцінки користувачів про використання чат-боту. Опитування можуть включати питання про задоволення, легкість використання, якість відповідей чат-боту та загальне враження від взаємодії.

2. Аналіз зворотного зв'язку: Вивчення коментарів, відгуків та пропозицій, які користувачі залишають щодо чат-боту. Це можуть бути повідомлення, які користувачі надсилають через інтерфейс чат-боту, електронні листи або коментарі в соціальних мережах. Аналіз цього зворотного зв'язку може дати уявлення про проблеми, які користувачі зустрічають та можливі шляхи покращення досвіду взаємодії.

3. Метрики продуктивності: Аналіз продуктивності чат-боту, таких як швидкість відповідей, час реакції та час обробки запитів. Вимірювання цих метрик може дати уявлення про ефективність та швидкість роботи чат-боту, що безпосередньо впливає на користувацький досвід.

4. Тестування в реальному часі: Проведення тестів в реальних умовах, де користувачі мають можливість взаємодіяти з чат-ботом і залишати свої враження. Це може включати спостереження за взаємодією користувачів з чат-ботом, реєстрацію їхніх вражень та оцінок, а також збір аналітичних даних для подальшого аналізу.

5. Застосування експертних оцінок: Залучення експертів з області дизайну і взаємодії з користувачем для оцінки дизайну і функціональності чат-боту. Експерти можуть внести цінний внесок, пропонуючи рекомендації щодо поліпшення користувацького досвіду та інтерфейсу.

Після збору і аналізу даних з оцінки користувацького досвіду, варто зробити висновки щодо сильних сторін чат-боту, виявлених проблем або слабких місць, а також розробити рекомендації щодо поліпшення функціональності, дизайну та загального враження від використання чат-боту.

3.2. Тестування бота

3.2.1. Визначення тестових сценаріїв для бота

Для початку тестування боту потрібно переглянути описати тестові можливі сценарії використання чат-бота магазина доставки суші.1. Сценарій замовлення суші:

- Користувач надсилає повідомлення з запитом на замовлення суші.
- Бот перевіряє наявність товарів у каталозі та пропонує користувачу вибрати певний вид суші та їх кількість.
- Користувач обирає потрібний вид суші.
- Бот запитує адресу доставки та інші необхідні дані.
- Користувач надає адресу доставки та інші необхідні дані.

- Бот підтверджує замовлення та надає інформацію про це.

2. Сценарій відстеження замовлення:

- Користувач запитує про свої замовлення.
- Бот надає користувачу його перелік замовлень з необхідною інформацією.

3. Сценарій зміни особистих даних користувача:

- Користувач обирає відповідну категорію меню.
- Користувач вводить нові дані
- Бот підтверджує зміну.

4. Сценарій додавання акції:

- Адміністратор обирає відповідну категорію меню.
- Адміністратор вводить деталі акції
- Бот підтверджує успішне додавання.

5. Сценарій видалення акції:

- Адміністратор обирає відповідну категорію меню.
- Адміністратор обирає акцію
- Бот підтверджує успішне видалення..

6. Сценарій перегляду замовлень користувачів:

- Адміністратор обирає відповідну категорію меню.
- Бот наводить список замовлень для перегляду.
- Адміністратор переглядає обране замовлення

7. Сценарій перегляду замовлень конкретного користувача:

- Адміністратор обирає відповідну категорію меню.
- Бот наводить список користувачів для вибору конкретного.
- Адміністратор обирає користувача
- Бот наводить список замовлень для перегляду.
- Адміністратор переглядає обране замовлення

8. Сценарій отримання інформації про акції та знижки:

- Користувач запитує про поточні акції та знижки.
- Бот надає інформацію про акційні пропозиції та знижки, які діють в магазині доставки суші.

9. Сценарій отримання зворотнього зв’язку:

• Користувач запитує про контакти підтримки.

•Бот надає користувачу необхідні дані.

10. Сценарій зміни списку суші адміністратором:

•Адміністратор заходить у меню з відповідними засобами та обирає опцію редагування списку категорій.

•Бот надає користувачу список для вибору з нього.

•Користувач вносить відповідні зміни

Кожен з цих сценаріїв був повторно виконаний з різними варіантами вхідних даних та різними послідовностями дій користувача, щоб покрити різноманітні можливості та умови використання бота.

3.2.2 Опис методології тестування чат-боту

Для тестування чат-боту магазину доставки суші була використана комбінація різних методологій тестування, яка дозволила забезпечити високу якість та надійність роботи системи. Основні методи тестування, що були використані, включають:

1. Функціональне тестування: Проводилося для перевірки відповідності функціональних вимог бота. Були створені тестові сценарії, які описували послідовність дій користувача та очікувані результати. Під час виконання тестових сценаріїв перевірялася правильність роботи бота.

2. Тестування на безпеку: Використовувалося для перевірки наявності потенційних уразливостей та захисту від атак. Були проведені тести на введення некоректних даних, SQL-ін'єкції та інших типів атак.

3. Тестування на різних платформах та пристроях: Бот був протестований на різних платформах, таких як веб-додатки та мобільні пристрої, щоб переконатися, що він працює належним чином на різних середовищах.

Під час тестування було ретельно вивчено результати, знайдені помилки та виконані необхідні корекції. Усі виявлені проблеми були задокументовані та виправлені, а потім були проведені повторні тести для перевірки коректності внесених змін. Тестування було повторено декілька разів до досягнення

задовільних результатів та впевненості у функціональності та якості роботи чат-бота.

3.2.3. Виконання тестових сценаріїв та реєстрація результатів

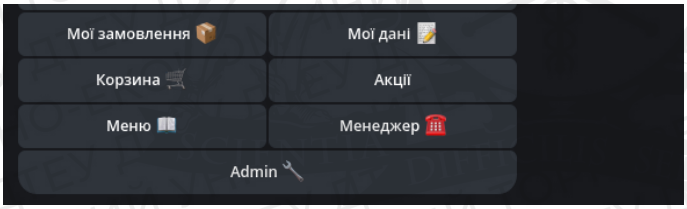
Під час тестування чат-боту були виконані попередньо складені тестові сценарії, а результати були ретельно зареєстровані для подальшого аналізу. Кожен тестовий сценарій складався з послідовності кроків, які включали взаємодію з чат-ботом та очікувані результати.

Під час виконання кожного кроку тестового сценарію було зазначено, чи було досягнуто очікуваний результат. Якщо були виявлені проблеми, помилки або невідповідності, вони були зафіксовані та відзначені як неуспішні кроки. У разі успішного виконання кроку він був позначений як успішний.

Проведення тестування:

Сценарій замовлення суші:

- Користувач надсилає повідомлення з запитом на замовлення суші.



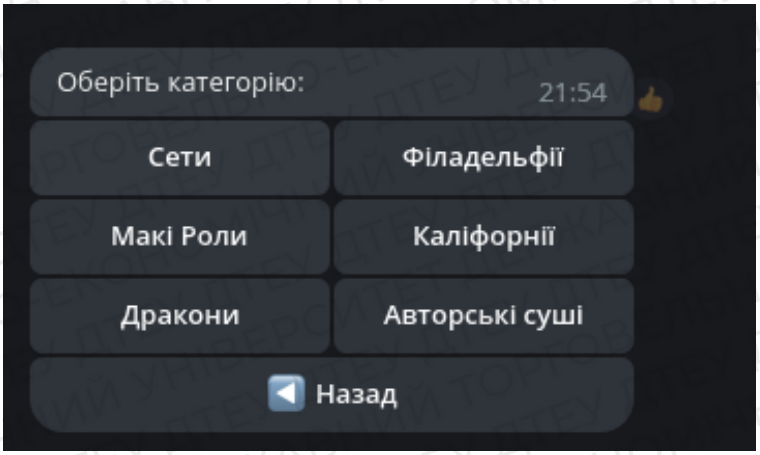


Рис 3.13 Запит від користувача

- Користувач обирає потрібний вид суші.

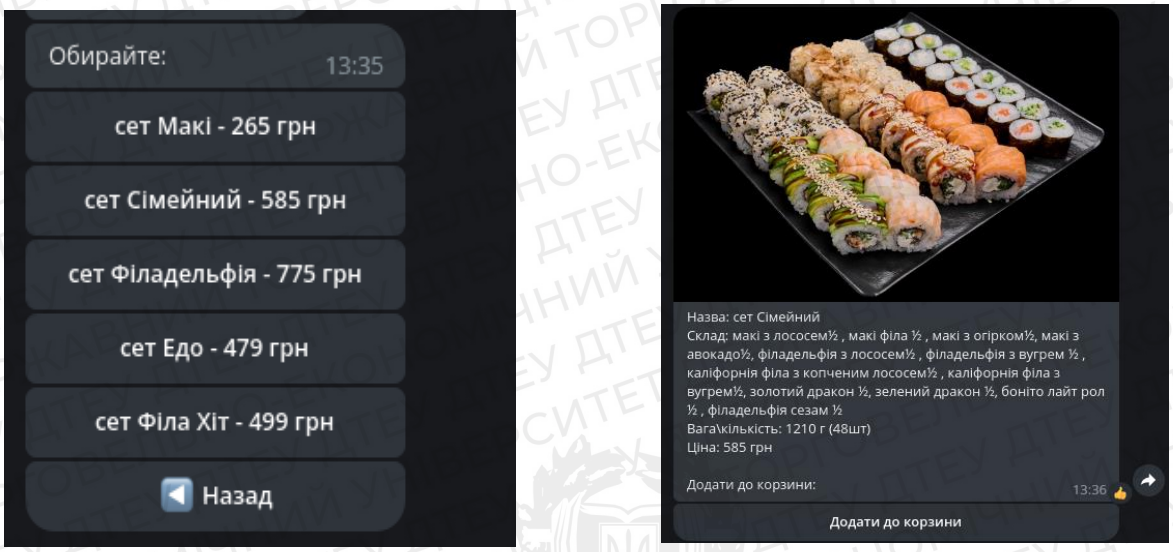


Рис 3.14 Перегляд меню

Користувач заходить в корзину та оформляє замовлення

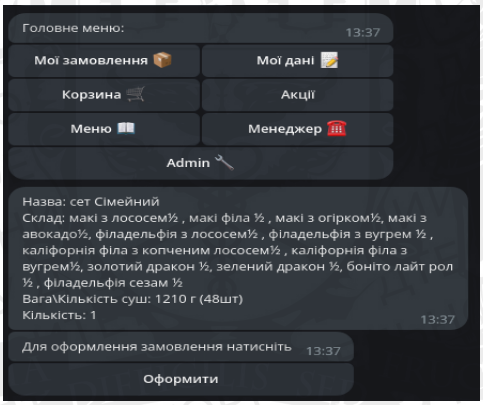


Рис 3.15 Перегляд корзини

- Бот запитує адресу доставки та інші необхідні дані.

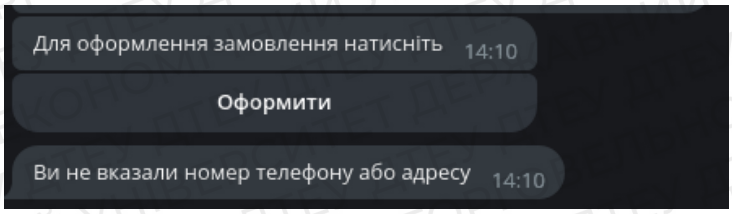


Рис 3.16 Запит інформації користувача

- Користувач надає адресу доставки та інші необхідні дані.

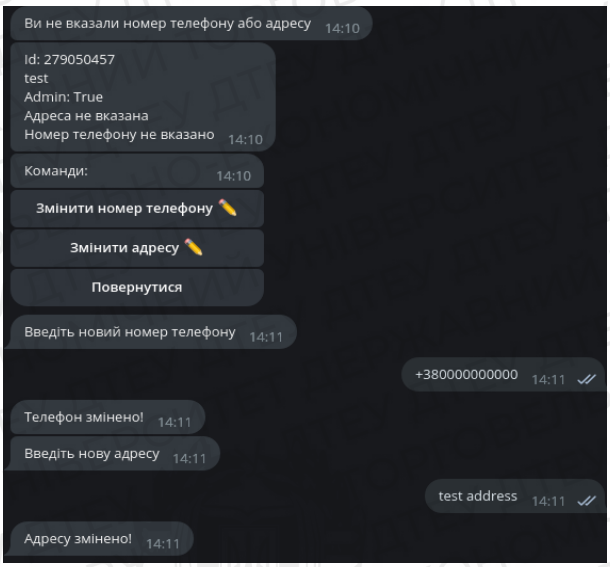


Рис 3.17 Підтвердження зміни особистої інформації користувача

- Бот підтверджує замовлення та надає інформацію про це.

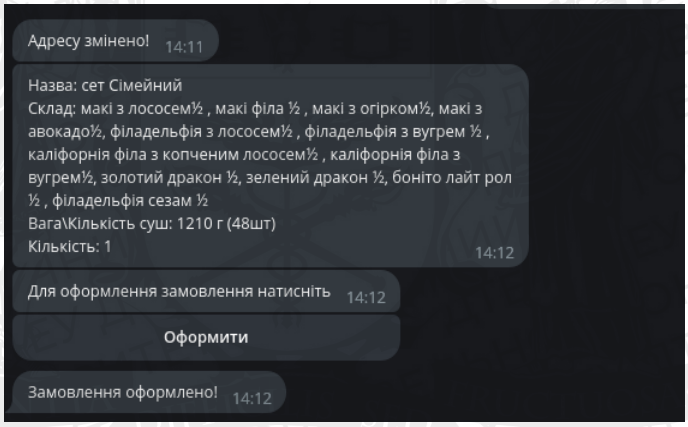


Рис 3.18 Підтвердження замовлення

Сценарій відстеження замовлення:

- Користувач запитує про свої замовлення

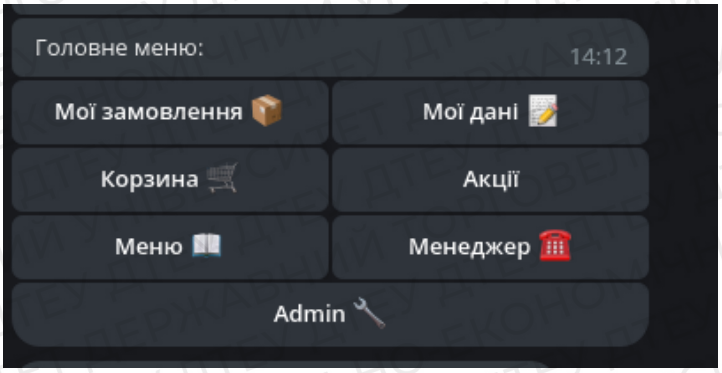


Рис 3.19 Запит користувача про замовлення

- Бот надає користувачу його перелік замовлень з необхідною інформацією.

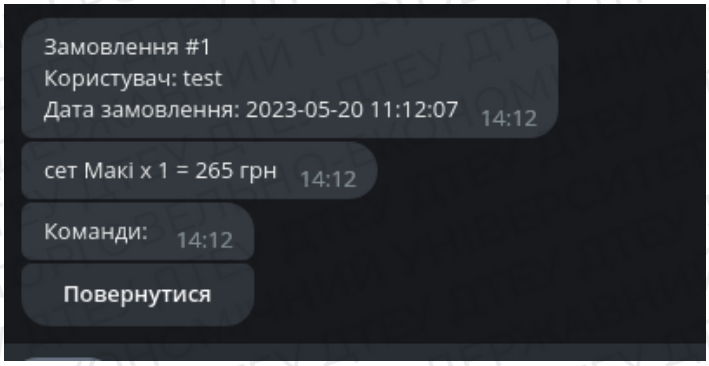


Рис 3.20 Отримання інформації щодо замовлення

Сценарій зміни особистих даних користувача:

- Користувач обирає відповідну категорію меню

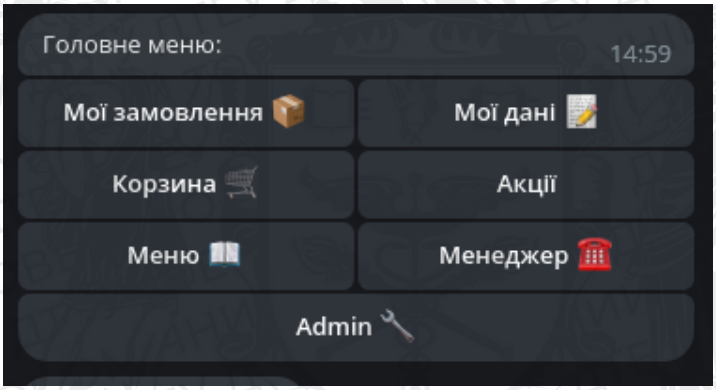


Рис 3.21 Меню «Мої дані»

- Користувач вводить нові дані

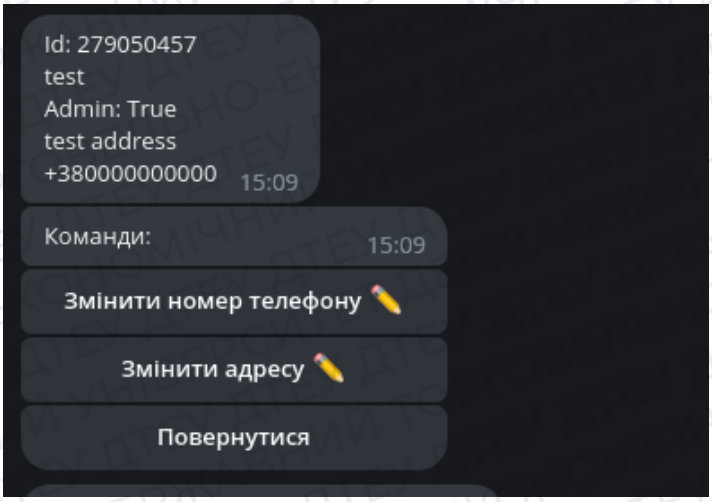


Рис 3.22 Введення нових данних

- Бот підтверджує зміну

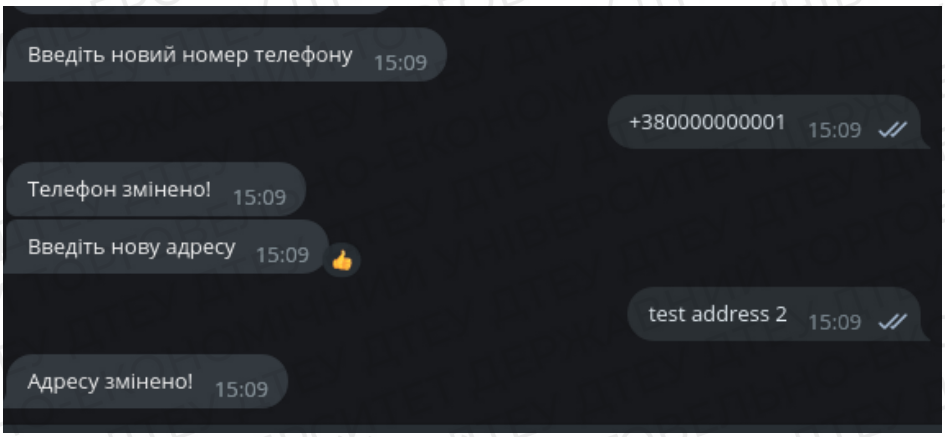


Рис 3.23 Підтвердження зміни данних

Сценарій додавання акції:

•Адміністратор обирає відповідну категорію меню·

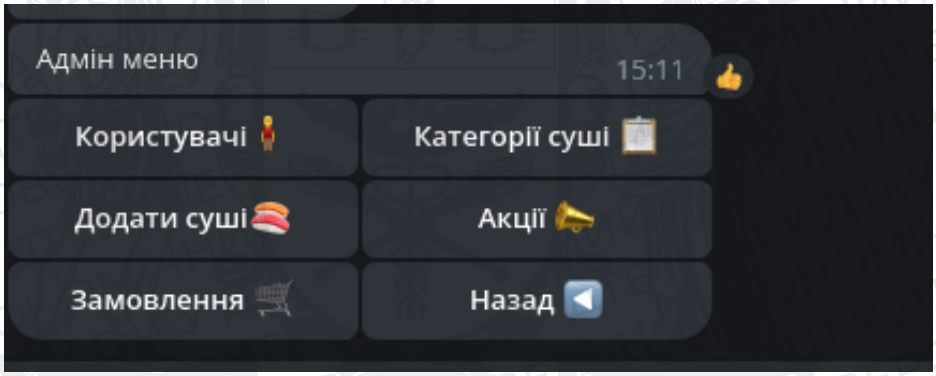


Рис 3.24 Меню « Акції»

• Адміністратор вводить деталі акції

Рис 3.25 Опис акції яку хоче додати Адміністратор

• Бот підтверджує успішне додавання.

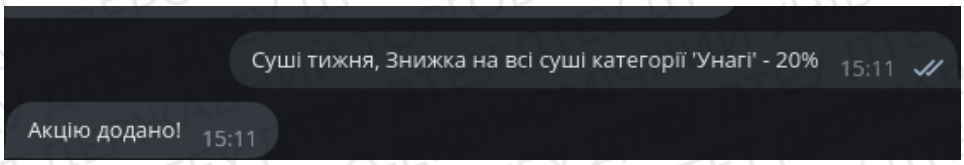


Рис 3.26 Підтвердження від бота про додавання акції

Сценарій перегляду акцій:

- Адміністратор обирає відповідну категорію меню

Рис 3.27 Меню «Акції»

- Адміністратор переглядає акції

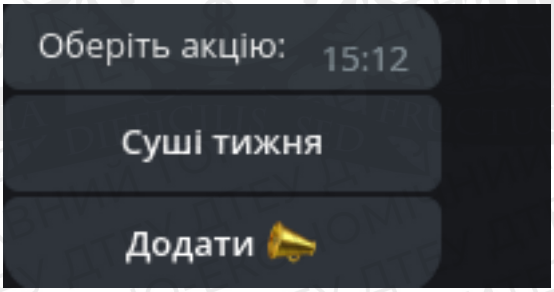


Рис 3.28 Перегляд акцій

Сценарій перегляду замовлень користувачів:

- Адміністратор обирає відповідну категорію меню.

Рис 3.29 Меню «Замовлення»

• Бот наводить список замовлень для перегляду.

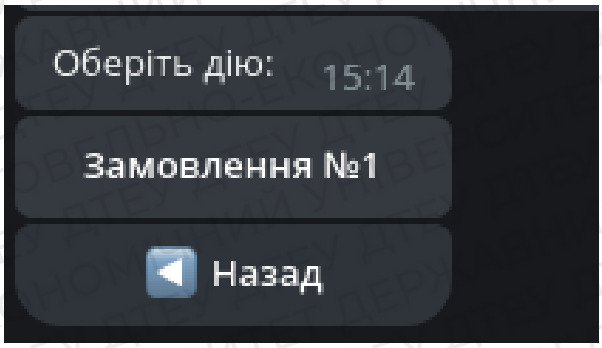


Рис 3.30 Список замовлень всіх користувачів

• Адміністратор переглядає обране замовлення

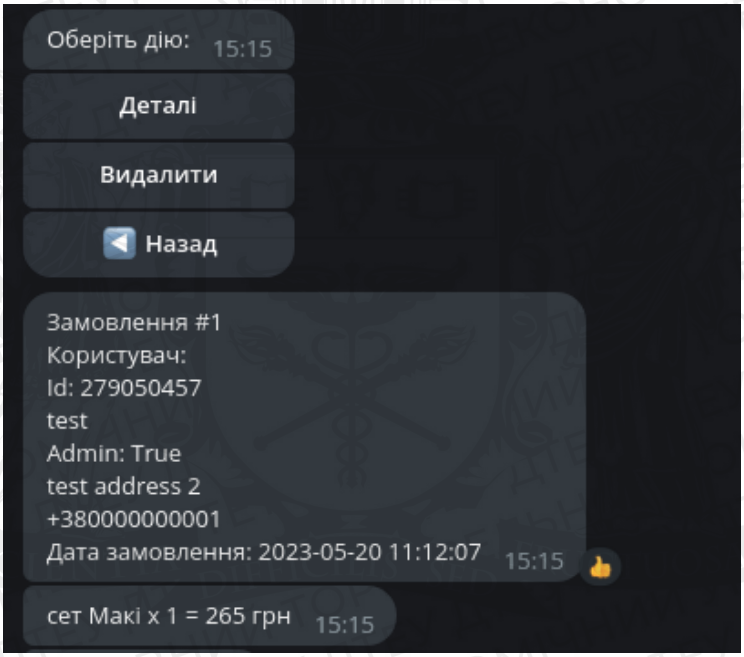


Рис 3.31 Деталі щодо обраного замовлення

Сценарій перегляду замовлень конкретного користувача:

• Адміністратор обирає відповідну категорію меню.

Рис 3.32 Меню «Користувачі»

• Бот наводить список користувачів для вибору конкретного

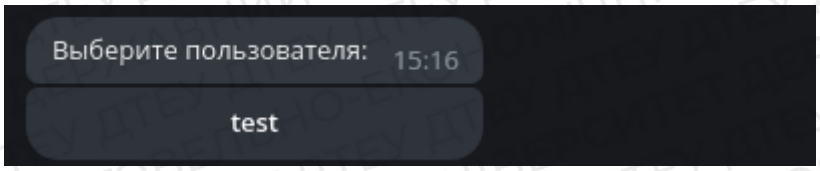


Рис 3.33 Список користувачів

• Адміністратор обирає користувача

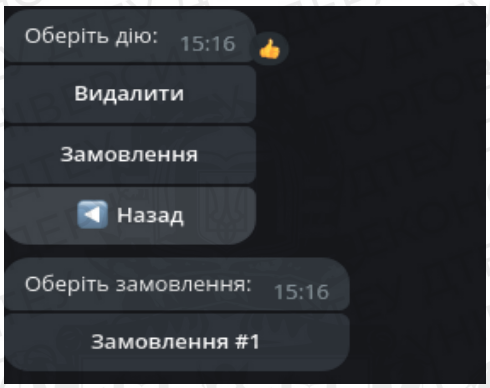


Рис 3.34 Вибір користувача

• Адміністратор переглядає обране замовлення

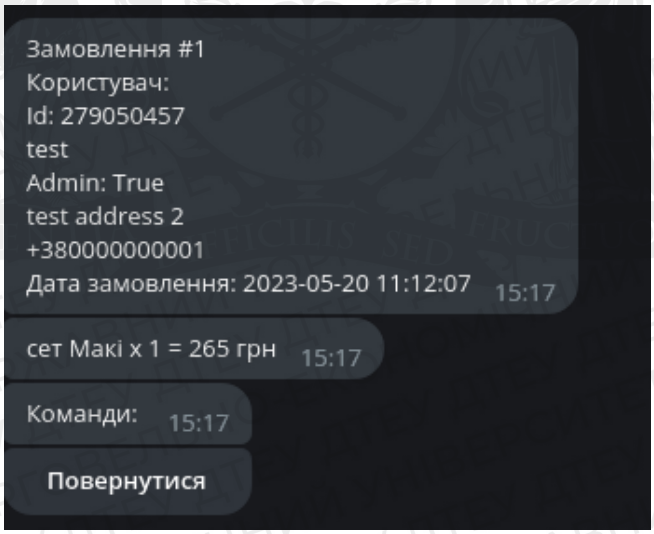


Рис 3.35 Вибір замовлення

Сценарій отримання зворотнього зв’язку:

• Користувач запитує про контакти підтримки.

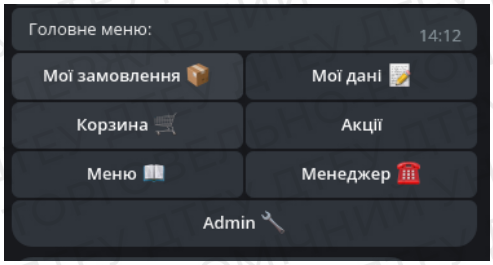


Рис 3.36 Меню «Менеджер»

•Бот надає користувачу необхідні дані.

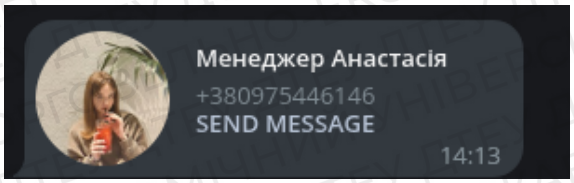


Рис 3.37 Інформація про менеджерів

Сценарій зміни списку суші адміністратором:

•Адміністратор заходить у меню з відповідними засобами та обирає опцію редагування списку категорій.

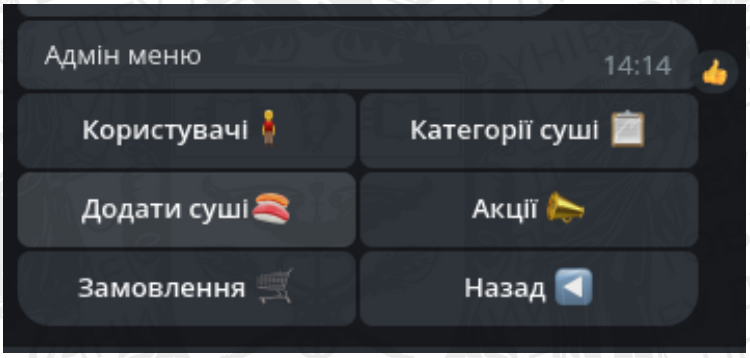


Рис 3.38 Меню «Додати суші»

•Бот надає список для вибору з нього.

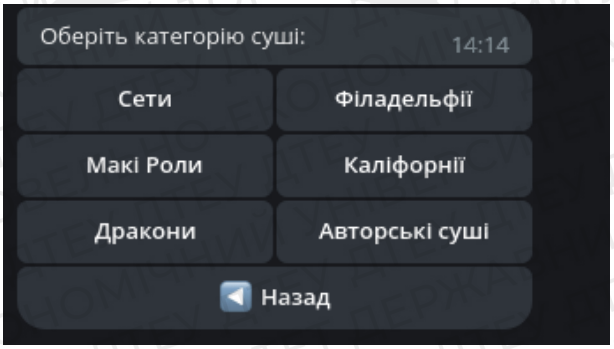


Рис 3.39 Вибір категорії

•Користувач вносить відповідні зміни

Сценарій додавання:

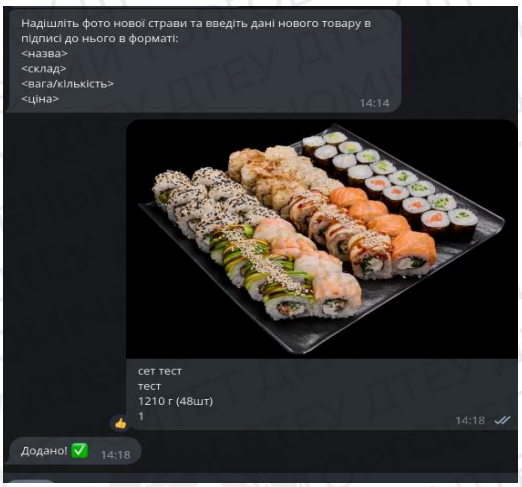


Рис 3.40 Вказуємо дані про новий товар

Сценарій видалення:

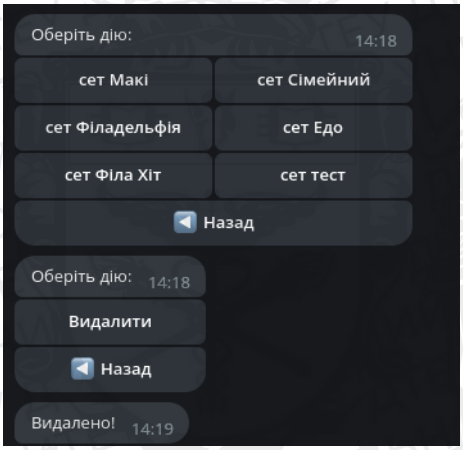


Рис 3.41 Обираємо категорію та видаляємо

Результати виконання тестових сценаріїв включали інформацію про кожен крок, його статус (успішний або неуспішний) та можливі коментарі чи примітки. Також, деякі тестові сценарії можуть мати вхідні дані, які були використані під час виконання тесту.

Усі результати тестування були документовані та збережені у вигляді звіту, що містив детальну інформацію про виконані кроки, виявлені проблеми та їх опис, а також рекомендації щодо подальших кроків, таких як виправлення помилок чи поліпшення функціональності бота.

Цей процес дозволяє забезпечити контроль якості та ідентифікувати можливі проблеми під час розробки та впровадження чат-боту. Результати

тестування надалі використовуються для вдосконалення та поліпшення роботи чат-боту перед його впровадженням у реальне середовище.

3.2.4 Аналіз результатів тестування та виявлення проблем

Після виконання тестових сценаріїв і реєстрації результатів, наступним кроком було проведення аналізу результатів тестування для виявлення потенційних проблем та недоліків у роботі чат-боту. Основні етапи аналізу включали:

1. Перевірка успішності: Було перевірено, який відсоток тестових сценаріїв був успішно виконаний. Це дозволило оцінити загальну якість роботи бота та визначити, наскільки він відповідає очікуванням.

У результаті перевірок, бот продемонстрував повну відповідність запланованому функціоналу.

2. Виявлення помилок: Були проаналізовані неуспішні кроки тестових сценаріїв, а також зареєстровані помилки та проблеми, виявлені під час тестування. Це дозволило ідентифікувати конкретні проблеми та визначити їх причини.

Помилки, виявлені в ході розробки, були виправлені, з метою поліпшення ефективності та надійності користування.

3. Класифікація проблем: Знайдені проблеми були класифіковані за їхнім типом та важливістю. Це допомогло визначити, які проблеми потребують негайного виправлення та які можуть бути вирішені наступними ітераціями розробки.

У ці класифікації могли входити несправності при взаємодії з базою даних або корегування виводу безпосередньо в чат-бот.

4. Виявлення трендів: Аналізуючи результати тестування, було виявлено можливі тренди або шаблони помилок, які повторювалися. Це дало змогу виявити системні проблеми або недоліки, які потребують уваги та ретельного вирішення.

5. Рекомендації та виправлення: На основі виявлених проблем та їх аналізу була проведена робота щодо виправлення та поліпшення роботи чат-боту. Це включало виправлення програмного коду, оптимізацію алгоритмів, покращення

інтерфейсу користувача та інші заходи для підвищення ефективності та надійності чат-боту.

Основні висновки з тестування включають наступне:

1. Бот успішно пройшов тестові сценарії: Тестові сценарії були успішно виконані, що свідчить про правильну реалізацію функціональності чат-боту.
2. Виявлено деякі недоліки та проблеми та здійснено роботи з їхнього усунення: Під час тестування були виявлені деякі недоліки та проблеми, такі як неправильні відповіді бота, помилки обробки даних тощо. Ці проблеми були ретельно досліджені та мною проведена робота з їхнього виправлення.
3. Тестування виконано згідно з методологією: Тестування було проведено з використанням визначених тестових сценаріїв та методології, що дозволило систематично перевірити функціональні та нефункціональні вимоги до чат-боту.

3.3 Рекомендації з упровадження чат-боту магазину доставки суші

Упровадження чат-боту магазину доставки суші є важливим кроком у поліпшенні взаємодії з клієнтами та оптимізації процесу прийому та обробки замовлень. Метою впровадження є забезпечення швидкого та зручного способу замовлення суші, а також підвищення задоволеності клієнтів та ефективності бізнесу. Представлені рекомендації та кроки щодо підготовки до впровадження чат-боту, вибору відповідних платформ та інфраструктури, налаштування інтеграції з існуючими системами та сервісами, а також розглянуто питання навчання бота та впровадження його в робоче середовище.

Ось кілька рекомендацій щодо розгортання та інтеграції чат-боту:

1. Вибір платформи:

Визначте платформу, на якій буде розгорнуто чат-бота. Розгляньте такі аспекти, як підтримка мови програмування, можливості інтеграції з іншими системами, швидкість та масштабованість.

2. Інфраструктура

Забезпечте необхідну інфраструктуру для роботи чат-боту. Це може включати серверне середовище, базу даних, сертифікати безпеки та інші компоненти, необхідні для безперебійної роботи бота.

3. Інтеграція з існуючими системами

Передбачте можливість інтеграції чат-боту з існуючими системами, такими як система управління замовленнями, база даних клієнтів тощо. Це дозволить боту отримувати та обробляти необхідну інформацію.

4. Забезпечення безпеки

Зверніть увагу на забезпечення безпеки чат-боту, зокрема захист від несанкціонованого доступу та збереження конфіденційної інформації, такої як персональні дані клієнтів.

5. Підтримка та оновлення

Передбачте механізми підтримки та оновлення чат-боту. Розгляньте можливість встановлення моніторингу, щоб слідкувати за роботою бота та виявляти можливі проблеми

Визначення оптимальної архітектури інфраструктури для бота

Оптимальна архітектура інфраструктури для чат-боту магазину доставки суші грає важливу роль у забезпеченні швидкої та надійної роботи бота. Аспекти, які слід враховувати при визначенні оптимальної архітектури, включають:

1. Серверне середовище
2. Інтеграція з зовнішніми сервісами
3. Безпека.
4. Моніторинг та логування.

Важливим пунктом для реалізаціїї чат-боту також є просування та стратегія маркетингу, яка допомагає залучати нових клієнтів.

Основні кроки стратегії маркетингу та просування чат-боту:

1. Визначення цільової аудиторії

Ретельно вивчіть свою цільову аудиторію, їхні потреби та очікування. Розуміння хто є вашими потенційними користувачами допоможе вам налаштувати стратегію маркетингу та просування.

2. Розробка унікального пропозиційного промовистого пакету.

Сформулюйте унікальну пропозицію вашого чат-боту, що виділятиме його серед конкурентів. Покажіть, як ваш чат-бот може принести користь та спростити процес замовлення та доставки суші.

3. Використання маркетингових каналів

Оберіть найефективніші маркетингові канали для просування чат-боту. Це можуть бути соціальні медіа, рекламні кампанії в Інтернеті, електронна пошта, блоги та інші канали, що залежать від вашої цільової аудиторії.

4. Використання впливових осіб та співпраця зі спільнотами

Співпрацюйте з впливовими особами у галузі гастрономії, блогерами або іншими фахівцями, які можуть рекомендувати ваш чат-бот своїм прихильникам. Також активно взаємодійте зі спільнотами та форумами, де обговорюються суші та гастрономічні теми.

5. Залучення користувачів через акції та промо-коди

Розробіть акції та спеціальні пропозиції для користувачів, які використовують чат-бот для замовлення суші. Введення промо-кодів та знижок сприятиме залученню нових користувачів та стимулюванню повторних замовлень.

6. Співпраця зі сторонніми партнерами

Розгляньте можливості співпраці з різними гастрономічними закладами, ресторанами, кухнями доставки та іншими партнерами, що можуть посилити вашу присутність на ринку та забезпечити додаткові можливості для користувачів.

ВИСНОВКИ І РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У рамках даної дипломної роботи було успішно розроблено чат-бот для магазину доставки суші, що забезпечує зручне замовлення та обробку замовлень за допомогою месенджеру. Результати роботи підтверджують ефективність використання чат-ботів для автоматизації процесу замовлення та покращення взаємодії з клієнтами.

В результаті виконання роботи та проведених досліджень були отримані такі **висновки і результати:**

1. Розробка чат-боту є ефективним засобом оскільки вона відповідає потребам сучасного бізнесу в ефективній комунікації з клієнтами, автоматизації процесів та підвищенні якості обслуговування. Ця технологія має великий потенціал у різних сферах бізнесу і є важливим інструментом для досягнення успіху.

2. Розроблено модель для визначення функціональних можливостей чат-боту магазину доставки суші за допомогою якої можна переглянути всі можливості здійснення тих чи інших операцій за допомогою чат-бота.

3.Проведено аналіз всіх існуючих методів розробки чат-ботів які включають: конструктори, платформи, мови програмування. Також визначено найефективніший метод розробки боту магазину досавки суші, обрано найбільш зручні та сучасні засоби для розробки та реалізації чат-боту магазину доставки суші.

4. Розроблено базу даних яка відповідає потребам проекту та забезпечує ефективне зберігання та обробку даних. Процес збору даних користувачів та замовлень був успішно реалізований, а забезпечення безпеки та цілісності даних було враховано.

5. У результаті роботи над дипломною роботою було досягнуто поставлені цілі, а саме розроблено функціональний чат-бот для магазину доставки суші, який спрощує процес замовлення та забезпечує зручну взаємодію з клієнтами. Результати цієї роботи можуть бути використані для подальшого розвитку та

вдосконалення чат-бота, а також для застосування у сфері ресторанного бізнесу та доставки їжі.

6. Проведення кінцевого тестування чат-боту виявилося успішним. Під час тестування було перевірено функціональність чат-боту, його взаємодію з користувачами, а також обробку та збереження даних.

Результати тестування підтвердили ефективність використання чат-боту для зручного замовлення та обробки замовлень за допомогою месенджеру. Чат-бот здатний правильно інтерпретувати команди користувачів, відповідати на запитання, а також зберігати інформацію про замовлення та користувачів у базі даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abu Shawar. Chatbots: are they really useful? / Abu Shawar // J. Lang. Technol. Comput. Linguist.- 2022.- 29–49 p.

2. Grace Lau. 7 real-life chatbot use cases across different industries / Grace Lau [Електронний ресурс] -Режим доступу до статті: https://www.dialpad.com/blog/chatbot-use-cases/

3.Розробка інтелектуального чат-бота відділу міжнародного співробітництва /Дягилева О. С., Лещенко А. М., Пазяк А. І., Юрженко А. Ю. : Збірник наукових праць "Information Technologies in Education" (ITE). – 2020. – № 2 (43). – 17-27 с.

4.Тищенко А. Разработка чат-бота в качестве виртуального помощника для мотивации студентов к получению профессии / А. Тищенко // Эргодизайн. – 2021. – № 2(12). – 140-144 с.

5.Виноградова О. В. Можливості месенджер-маркетингу для просування товарів та послуг в інтернеті / О. В. Виноградова, Н. І. Дрокіна, В. Г. Дарчук: Економіка. Менеджмент. Бізнес. – 2020. – № 1(31). – 11-20 с.

6.Pesonen J. ‘Are You OK?’ Students’ Trust in a Chatbot Providing Support Opportunities / J. Pesonen // 8th International Conference on Human-Computer Interaction HCII-2021 “Learning and Collaboration Technologies: Games and Virtual Environments for Learning”. – July 24-29, 2021. – 199-215p.

7. Коцофане О. Все про чат-боти: типи і приклади, якому бізнесу підійде, список конструкторів для створення. [Електронний ресурс]

-Режим доступу до статті: https://web-promo.ua/ua/blog/vse-o-chat-botah-tipy-i-primery-kakomu-biznesu-podojdetspisok-konstruktorov-dlya-sozdaniya/#platformy.

8.. Ушакова І. Підходи до створення інтелектуальних чат-ботів / І. Ушакова В.: Системи обробки інформації. –2019.– № 2(157). – С 86-93.

9. Nimavat, K . Chatbots: an overview types, architecture, tools and future possibilities. / Nimavat, K., Champaneria, T.: International Journal for Scientific Research and Development - 2017 - 1019–1024 p.

10.Vishwakarma A. A Review & Comparative Analysis on Various Chatbots Design /A. Vishwakarma, A. Pandey : International Journal of Computer Science and Mobile Computing. – 2021. – Vol. 10(2). – 72-78p.
11. ChatBot builders in 2023 / A. Brooks [Електронний ресурс]
-Режим доступу до статті: https://www.ventureharbour.com/best-chatbot-builders/
12. Эрик. М. Python. Програмування, візуалізація / Эрик. М// Програмування ігор, візуалізація данних, веб-додатки, - 2017. – 496 с.
13.Rauschmayer D. JavaScript for impatient programmers/ Dr. Axel Rauschmayer.// - 2021. – 319 с.
14.JavaScript / Вікіпедія[Електронний ресурс]
– Режим доступу до ресурсу:https://uk.wikipedia.org/wiki/JavaScript
15. What is the Bot Framework SDK? [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/bot-service/bot-service-overview?view=azure-bot-service-4.0
16.How to create a WhatsApp Chatbot for business? Step-by-step guide & tutorial [Електронний ресурс] – Режим доступу до
ресурсу: https://sleekflow.io/blog/whatsapp-chatbot
17. Facebook Messenger passes 300,000 bots [Електронний ресурс] – Режим доступу до ресурсу:https://venturebeat.com/ai/facebook-messenger-passes-300000-bots/
18. Telegram Bot API [Електронний ресурс] – Режим доступу до ресурсу : https://core.telegram.org/bots/api#using-a-local-bot-api-server

Додаток

Прорамний код реалізації чат-боту

```
import logging
import os

from dotenv import load_dotenv
from telegram import Update, InlineKeyboardButton,
InlineKeyboardMarkup
from telegram.ext import Updater, CommandHandler, MessageHandler,
Filters, CallbackContext, CallbackQueryHandler

import btn_cb_module
import db_module
import kb_module

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s
- %(message)s', level=logging.INFO)
logger = logging.getLogger(__name__)
load_dotenv()
ADMINS_ID = [int(i) for i in os.getenv("ADMINS_ID").split(",")]
MANAGER_PHONE_NUMBER = os.getenv("MANAGER_PHONE_NUMBER")
WELCOME_TEXT = os.getenv("WELCOME_MESSAGE")


def start(update: Update, context: CallbackContext):
    user_id = update.message.from_user.id
    is_admin = user_id in ADMINS_ID

    first_name = update.message.from_user.first_name
    last_name = update.message.from_user.last_name or ""
    if not db_module.user_exists(user_id):
        db_module.add_user(user_id, first_name, last_name,
int(is_admin))

    keyboard = kb_module.build_user_keyboard(is_admin)
    update.message.reply_text(WELCOME_TEXT, reply_markup=keyboard)


def is_user_admin(user_id):
    return user_id in ADMINS_ID


def notify_admin_about_order(update: Update, context:
CallbackContext, order_info: str) -> None:
    for admin_id in ADMINS_ID:
        context.bot.send_message(chat_id=admin_id,
text=order_info)


def button_callback(update: Update, context: CallbackContext):
```

```
    query = update.callback_query
    query.answer()
    if query.data == "edit_person_profile_address":
        btn_cb_module.callback_edit_person_profile_address(update,
context)
        return
    if query.data == "edit_person_profile_phone":
        btn_cb_module.callback_edit_person_profile_phone(update,
context)
        return
    if query.data == "admin_show_user_orders":
        btn_cb_module.callback_admin_show_user_orders(update,
context)
        return
    if query.data.startswith("admin_show_order_actions_"):
        order_id =
int(query.data[len("admin_show_order_actions_"):])
        btn_cb_module.callback_admin_show_order_actions(update,
context, order_id)
        return
    if query.data.startswith("delete_order_"):
        order_id = int(query.data[len("delete_order_"):])
        btn_cb_module.callback_delete_order(update, context,
order_id)
        return
    if query.data.startswith("delete_category_"):
        category_id = int(query.data[len("delete_category_"):])
        btn_cb_module.callback_delete_category(update, context,
category_id)
        return
    if query.data.startswith("delete_sushi_"):
        sushi_id = int(query.data[len("delete_sushi_"):])
        btn_cb_module.callback_delete_sushi(update, context,
sushi_id)
        return
    if query.data.startswith("edit_sushi_"):
        sushi_id = int(query.data[len("edit_sushi_"):])
        btn_cb_module.callback_edit_sushi(update, context,
sushi_id)
        return
    if query.data == "edit_categories":
        btn_cb_module.callback_edit_categories(update, context)
        return
    if query.data.startswith("edit_category_"):
        category_id = int(query.data[len("edit_category_"):])
        btn_cb_module.callback_edit_category(update, context,
category_id)
        return
    if query.data == "my_orders":
        btn_cb_module.callback_my_orders(update, context,
query.from_user.id)
        return
    if query.data == "promotions":
```

```
        btn_cb_module.callback_show_promotions_for_user(update,
context)
        return
    if query.data == "add_promotion":
        btn_cb_module.callback_add_promotion(update, context)
        return
    if query.data == "show_admin_promotions":
        btn_cb_module.callback_show_promotions(update, context)
        return
    if query.data.startswith("delete_"):
        user_id = int(query.data[len("delete_"):])
        if is_user_admin(user_id):
            query.message.reply_text("Неможливо видалити
адміністратора")
            return
        db_module.delete_user(user_id)
        query.message.reply_text("Користувача видалено")
        return
    if query.data == "back_to_main":
        btn_cb_module.callback_return_to_main(update, context)
        return
    if query.data == "my_profile":
        btn_cb_module.callback_my_profile(update, context)
        return
    if query.data == "admin_panel":
        btn_cb_module.callback_admin_panel(update, context)
        return
    if query.data == "cart":
        user_cart =
db_module.get_user_cart_items(query.from_user.id)
        if len(user_cart) == 0:
            query.message.reply_text("Пуста корзина")
            return

        for cart_item in user_cart:
            sushi_id, sushi_object, sushi_name, sushi_ingredients,
sushi_weight_or_quantity, cart_quantity = cart_item
            query.message.reply_text(
                f"Назва: {sushi_name}\n"
                f"Склад: {sushi_ingredients}\n"
                f"Вага\Кількість суш:
{sushi_weight_or_quantity}\n"
                f"Кількість: {cart_quantity}\n"
            )
        # create order button
        keyboard = InlineKeyboardMarkup(
            [
                [
                    InlineKeyboardButton("Оформити",
callback_data="checkout")
                ]
            ]
        )
```

```
        query.message.reply_text("Для оформлення замовлення
натисніть", reply_markup=keyboard)
        return
    if query.data.startswith("add_to_cart_"):
        sushi_id = int(query.data[len("add_to_cart_"):])
        user_id = query.from_user.id

        db_module.add_sushi_to_cart(user_id, sushi_id, 1)
        sushi_object = db_module.get_sushi_by_id(sushi_id)
        sushi_category_id = sushi_object[1]
        sushi_list =
db_module.get_sushi_by_category_id(sushi_category_id)
        # send message, not reply
        sushi_keyboard = []
        for sushi in sushi_list:
            sushi_id, _, sushi_name, _, _, sushi_price = sushi
            button_text = f"{sushi_name} - {sushi_price} грн"
            sushi_button = InlineKeyboardButton(button_text,
callback_data=f"sushi_{sushi_id}")
            sushi_keyboard.append([sushi_button])
        sushi_keyboard.append([InlineKeyboardButton("Назад",
callback_data="back_to_main")])
        keyboard = InlineKeyboardMarkup(sushi_keyboard)
        query.from_user.send_message("Суші додано в корзину",
reply_markup=keyboard)
        return
    if query.data == "checkout":
        btn_cb_module.callback_checkout(update, context)
        return
    if query.data.startswith("user_"):
        user_id = int(query.data[len("user_"):])
        keyboard = kb_module.build_user_actions_keyboard(user_id)
        query.edit_message_text("Оберіть дію:",
reply_markup=keyboard)
        return
    if query.data.startswith("show_order_details_"):
        order_id = int(query.data[len("show_order_details_"):])
        btn_cb_module.callback_show_order_details(update, context,
order_id)
        return
    if query.data == "menu":
        keyboard = kb_module.build_user_categories_keyboard()
        query.edit_message_text("Оберіть категорію:",
reply_markup=keyboard)
        return
    if query.data.startswith("show_user_orders_"):
        user_id = int(query.data[len("show_user_orders_"):])
        btn_cb_module.callback_show_user_orders(update, context,
user_id=user_id)
        return
    if query.data.startswith("sushi_"):
        sushi_id = int(query.data[len("sushi_"):])
        sushi = db_module.get_sushi_by_id(sushi_id)
```

```
        sushi_name = sushi[2]
        sushi_ingredients = sushi[3]
        sushi_weight_or_quantity = sushi[4]
        sushi_price = sushi[5]

        keyboard = kb_module.build_add_to_cart_keyboard(sushi_id)
        with open(f"./images/{sushi_id}.jpg", "rb") as photo:
            query.message.reply_photo(photo,caption=
                f"Назва: {sushi_name}\n"
                f"Склад: {sushi_ingredients}\n"
                f"Вага\кількість: {sushi_weight_or_quantity}\n"
                f"Ціна: {sushi_price} грн\n\n"
                f"Додати до корзини:",
                reply_markup=keyboard,
            )
        return
    if query.data == "manager":
        btn_cb_module.callback_send_manager_contact(update,
context)
        return
    if query.data == "show_users":
        btn_cb_module.callback_show_users(update, context)
        return
    if query.data.startswith("menu_category_"):
        category_id = int(query.data[len("menu_category_"):])

        sushi_list =
db_module.get_sushi_by_category_id(category_id)

        sushi_keyboard = []
        for sushi in sushi_list:
            sushi_id, _, sushi_name, _, _, sushi_price = sushi
            button_text = f"{sushi_name} - {sushi_price} грн"
            sushi_button = InlineKeyboardButton(button_text,
callback_data=f"sushi_{sushi_id}")
            sushi_keyboard.append([sushi_button])

        back_button = InlineKeyboardButton("◀ Назад",
callback_data="menu")
        sushi_keyboard.append([back_button])

        query.message.reply_text("Обирайте:",
reply_markup=InlineKeyboardMarkup(sushi_keyboard))
        return
    if query.data == "add_sushi_category":
        btn_cb_module.callback_add_sushi_category(update, context)
        return
    if query.data == "add_sushi_item":
        context.user_data['action'] = 'add_sushi_item'
        keyboard = kb_module.build_sushi_categories_keyboard()
        query.edit_message_text("Оберіть категорію суші:",
```

```
reply_markup=keyboard)
        return
    if query.data.startswith("category_"):
        category_id = int(query.data[len("category_"):])
        context.user_data['selected_category'] = category_id
        context.user_data['action'] = 'add_sushi_item_details'
        query.edit_message_text("Надішліть фото нової страви та
введіть дані нового товару в підписі до нього в форматі:\n"
                                "<назва>\n"
                                "<склад>\n"
                                "<вага/кількість>\n"
                                "<ціна>")

    else:
        query.message.reply_text(f"Ви обрали пункт: {query.data}")


def text_handler(update: Update, context: CallbackContext):
    if context.user_data["action"] == "awaiting_phone":
        phone = update.message.text
        if not phone.startswith("+380"):
            update.message.reply_text("Невірний формат телефону.
Спробуйте ще раз.")
            return
        user_id = update.message.from_user.id
        db_module.change_user_phone(user_id, phone)
        update.message.reply_text("Телефон змінено!")
        context.user_data.pop("action")
    if context.user_data["action"] == "awaiting_address":
        address = update.message.text
        user_id = update.message.from_user.id
        db_module.change_user_address(user_id, address)
        update.message.reply_text("Адресу змінено!")
        context.user_data.pop("action")
    if context.user_data["action"] == "awaiting_promotion_info":
        text = update.message.text
        name, description = text.split(",")
        db_module.add_promotion(name, description)
        update.message.reply_text("Акцію додано!")
        context.user_data.pop("action")
    if context.user_data['action'] ==
"awaiting_sushi_category_name":
        category_name = update.message.text
        if db_module.category_exists(category_name):
            update.message.reply_text(f"Категорія
'{category_name}' вже існує.")
            return
        db_module.add_sushi_category(category_name)
        update.message.reply_text(f"Категорія '{category_name}'
додана!")
        context.user_data.pop("action")
    elif context.user_data["action"] == "add_sushi_item_details":
        try:
```

```
            if not update.message.photo:
                raise ValueError("No photo provided")

            photo = update.message.photo[-1]
            photo_file = context.bot.get_file(photo.file_id)
            max_id = db_module.get_latest_sushi()

            photo_path = f'./images/{max_id+1}.jpg'
            photo_file.download(photo_path)


            text = update.message.caption
            category_id = context.user_data['selected_category']
            name, ingredients, weight, price = text.split("\n")
            context.user_data.pop('action')
            context.user_data.pop('selected_category')
            db_module.add_sushi(category_id, name, ingredients,
weight, price)
            update.message.reply_text("Додано! ✅")
        except ValueError:
            update.message.reply_text("Помилка вводу даних.
Спробуйте ще раз. ❌")


def error(update: Update, context: CallbackContext):
    logger.warning(f'Update "{update}" caused error
"{context.error}"')


def register_handlers(dp):
    dp.add_handler(CommandHandler('start', start))
    dp.add_handler(MessageHandler(Filters.text | Filters.photo,
text_handler))
    dp.add_handler(CallbackQueryHandler(button_callback))
    dp.add_error_handler(error)


def main():
    db_module.init()
    bot_token = os.getenv("TELEGRAM_BOT_TOKEN")
    updater = Updater(bot_token)

    dp = updater.dispatcher
    register_handlers(dp)

    updater.start_polling()
    updater.idle()


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