ДИПЛОМНА РОБОТА

БАКАЛАВРА

ДРБ. ІП - 60.00.00.000 ПЗ

***Група ІП*-21-2**

Щербій Андрій

2025

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Щербію Андрій Миколайович

(прізвище, ім'я, по батькові)

_____

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка бота ресторану для замовлення страв на доставку та бронювання столиків засобами Python (назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121– Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня ***А.М. Щербій***

(підпис, ініціали та прізвище здобувача)

Науковий керівник Піх М.М. асистент

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

***ЗАТВЕРДЖУЮ*:**
Зав. кафедрою ***ІПЗ***
к.т.н.,доц. В.В. Бандура
“ ” 2025 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Щербію Андрію Миколайовичу

(прізвище, ім’я, по-батькові)

1. ***Тема проекту (роботи)*** ***“ Розробка бота ресторану для замовлення страв на доставку та бронювання столиків засобами Python”***

керівник проекту (роботи) Піх Марія Михайлівна, асистент

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 р. № 264/7

2. ***Строк подання студентом проекту (роботи)*** 10 червня 2025 р.

3. ***Вихідні дані до проекту (роботи)*** Результати і матеріали отримані під час проходження переддипломної практики

4. ***Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)***

1. Аналіз предметної області й існуючих рішень
2. Аналіз вимог до проекту
3. Опис використаних технологій
4. Розробка та тестування ПЗ
5. Представлення результатів

5. ***Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)***

1. Загальний Use-Case для бота (рис. 1.10., ст.20)
2. Use-Case оформлення замовлення. (рис. 1.11., ст.21)
3. Діаграма діяльності. рис. 1.12., ст.22)
4. Використання БД (рис. 2.2., ст.36)
5. Добування API-ключа (рис. 3.3., ст.42)

6. ***Консультанти розділів проекту (роботи)***

Розділ	***Консультант***	***Підпис, дата***	
		Завдання видав	***Завдання прийняв***

7. ***Дата видачі завдання*** **2025** ***р.***

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	***Назва етапів дипломного проекту (роботи)***	***Строк виконання етапів проекту***	***Примітка***
1	Вибір теми та технологій розробки	15.02.2025	Виконано
2	Опис предметної області	10.03.2025	Виконано
3	Аналіз вимог	25.04.2025	Виконано
5	Розробка бота	30.04.2025	Виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	10.06.2025	Виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 65 сторінкок, 35 рисунків, список використаних джерел із 30 найменуваннями.

Об’єкт дослідження - процес розробки програмних рішень на мові програмування Python з використанням Telegram Bot API.

Предмет дослідження - методи, інструменти та бібліотеки Python, що застосовуються для створення Telegram-ботів, зокрема із використанням фреймворку `telebot` у середовищі розробки PyCharm.

Мета дослідження - дослідити технології та інструменти для створення Telegram-бота на мові Python, обґрунтувати вибір відповідного середовища розробки та бібліотек, а також реалізувати функціональний бот на основі сучасних програмних підходів.

Результати дослідження - розроблено бота для оформлення замовлень з ресторану на доставку та бронювання столиків засобами Python **В першому розділі** описано предметну область та аналіз вимог.

В другому, третьому розділі проведений опис використаних технологій та підготовка до розробки.

У четвертому розділі описано саму розробку бота для оформлення замовлень з ресторану та бронювання столиків.

Висновки - у роботі досліджено сучасні засоби створення Telegram-ботів на мові програмування Python. Обрані технології дозволили ефективно реалізувати основні функції взаємодії з користувачем.

КЛЮЧОВІ СЛОВА: **ПАЙТОН, TG-БОТ, МЕСЕДЖЕР, API, БІБЛІОТЕКА TELEBOT, МОДУЛЬ “TYPES”, МОДУЛЬ “RE”, ОБРОБКА ПОДІЙ, ПЗ.**

ANNOTATION

The bachelor's thesis contains 65 pages, 35 figures, a list of used sources with 30 names.

The object of the study is the process of developing software solutions in the Python programming language using the Telegram Bot API.

The subject of the study is methods, tools and Python libraries used to create Telegram bots, in particular using the telebot framework in the PyCharm development environment.

The purpose of the study is to investigate technologies and tools for creating a Telegram bot in the Python language, to justify the choice of an appropriate development environment and libraries, and to implement a functional bot based on modern programming approaches.

Research results - a bot for placing restaurant orders for delivery and booking tables using Python has been developed.

The first section describes the subject area and analysis of requirements.

The second and third sections describe the technologies used and preparation for development.

The fourth section describes the development of a bot for placing restaurant orders and booking tables.

Conclusions - the paper explores modern tools for creating Telegram bots in the Python programming language. The selected technologies allowed for the effective implementation of the main functions of user interaction.

KEYWORDS: **PYTHON, TELEGRAM BOT, TG-API, TELEBOT LIBRARY, MODULE TYPES , RE LIBRARY, EVENT PROCESSING.**

ЗМІСТ

ВСТУП

Бакалаврська робота базується на розробці бота для поширеного меседжеру з метою зручної обробки ресторанних замовлень на доставку та бронювання столиків. Сьогодні стрімко зростає популярність програмних рішень для замовлення їжі та іншого тому , що такі сервіси забезпечують зручність і швидкість у процесі замовлення та доставки страв.

Актуальність теми У сучасному цифровому світі автоматизація процесів спілкування, обслуговування користувачів і взаємодії з клієнтами стає все більш затребуваною. Telegram-боти, як інструменти миттєвої взаємодії, знаходять широке застосування у бізнесі, сервісах, навчанні та особистій продуктивності. Python — одна з найпопулярніших мов програмування завдяки своїй простоті, гнучкості та великій кількості готових бібліотек, зокрема для створення ботів. Дослідження процесу розробки Telegram-бота на Python дозволяє краще зрозуміти особливості цієї технології та відкриває можливості для її широкого застосування у різних сферах.

Метою моєї дипломної роботи є розробка бота засобами мови програмування Python та її обширної кількості бібліотек, яка дозволить користувачам замовляти їжу з ресторану без лишніх дій у декілька кліків. Боти для популярних меседжерів стають все більш актуальними, оскільки велика кількість аудиторії надає перевагу використовувати меседжери як альтернативу соціальним мережам по типу: Facebook, Instagram, Twitter/

У процесі розробки використав Python тому, що вона має багатий інструментарій та широкі можливості для розробки ботів подібного роду. Також використав різні бібліотеки та фреймворки, включаючи Telebot для взаємодії з API Telegram і роботи з повідомленнями.

Основними завданнями розробленого бота будуть:

- Відображення меню ресторану з переліком доступних страв та їх цінами.
- Додавання страв в кошик із зазначенням кількості.

- Оформлення замовлення із зазначенням ПІБ, адреси доставки та контактного телефону.
- Відправлення замовлення адміністратору для обробки та доставки.
- Можливість користувачів залишати відгуки та коментарі про сервіс.
- Бронювання столиків у ресторані

Виконання даної задачі сприятиме автоматизації процесу замовлення та доставки страв із ресторану, що зробить його більш зручним та швидким для користувачів. Використання такого бота дозволяє ресторанам та кафе покращити свою ефективність та конкурентоспроможність, надаючи клієнтам новий рівень зручності та персоналізації та крокуючи в ногу з часом.

Об'єкт дослідження - процес розробки програмних рішень на мові програмування Python з використанням Telegram Bot API.

Предмет дослідження -методи, інструменти та бібліотеки Python, що застосовуються для створення Telegram-ботів, зокрема із використанням фреймворку telebot у середовищі розробки PyCharm.

Окрім того, розробка такого бота, надає змогу ресторанам збирати дані про замовлення, вподобання клієнтів та історію їхніх покупок. Відповідно ресторани отримують дорогоцінну інформацію про популярність страв та вчасну реакцію на зміни попиту та персоналізовані рекомендації, що в подальшому можна використати в цілях підвищення показника задоволеності клієнтів та зміцнення їхньої лояльності.

В роботі були використані такі **методи дослідження**: аналіз літературних джерел, системний аналіз, метод моделювання, практичне програмування, тестування та відлагодження

Результати дослідження - створено функціональний Telegram-бот із графічним інтерфейсом у форматі клавіатур та обробкою повідомлень.

Наукова новизна - поєднує сучасні програмні практики розробки ботів із прикладним застосуванням Python-бібліотек, зокрема telebot, для створення інтерактивного Telegram-бота. Запропоновано підхід до побудови структури

Telegram-бота, що поєднує простоту реалізації та розширюваність, що може бути адаптоване для створення складніших автоматизованих систем у сфері комунікацій.

Використання Python для розробки бота дозволяє ефективно використовувати потужність цієї мови програмування та використовувати різні інтеграції та розширення. Python має велику кількість модулів і пакетів, які дозволяють розробникам розширити функціональність бота.

Бакалаврська робота містить 76 сторінок, 35 рисунків, 4 розділи, список використаних джерел із 30 найменуваннями.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ ВИМОГ

1.1. Аналіз сучасних рішень для автоматизації замовлень у закладах харчування за допомогою ботів

Сьогодні будь-яка сфера послуг потребує сучасних рішень, які автоматизують і прискорять алгоритми роботи . Автоматизація бізнес-процесів у дуже актуальна тому, що методи для автоматизацій , які надає програмна інженерія – можуть зекономити колосальну суму коштів. Рішення такого роду надають перевагу над конкурентами і дозволяють бізнес-структурам йти в ногу з часом.

В бакалаврській роботі - задачею є – реалізація боту для інтеграції у популярні меседжери, основною задачою якого є автоматизація процесів прийому та доставки замовлень з ресторану, а також бронювання столиків. Тобто бот повинен містити фото, назви та ціни позицій. Усі сформовані клієнтом замовлення та бронювання повинні відправлятись працівнику в робочий чат, а відгуки на канал закладу . У випадку моєї розробки , якщо підприємець хоче збільшити можливі варіанти отримання більше замовлень і надати своїм клієнтам зручний спосіб оформлення замовлень та бронювання столиків, але при цьому не витрачати багато грошей , то цей варіант йому ідеально підходить. Саме це є метою моєї дипломної роботи – надати доступний стартовий варіант для автоматизації таких процесів методом діджиталізації.

Для того щоб оформити замовлення клієнт натискає на одну з категорій в меню , обирає товар, кількість , натискає кнопку “назад” та при потребі добавляє до корзини інші позиції із категорій , натискає оформити замовлення в вводить ім’я, адресу доставки та номер телефону.

Сьогодні все більше кафе та ресторанів шукають способи оптимізувати та автоматизувати замовлення й доставку їжі. Використання ботів для таких цілей є одним із поширених методів. Боти спрощують взаємодію з клієнтами та

дозволяють їм робити замовлення через месенджер. У цьому розділі буде проаналізовано кілька сучасних ботів популярного меседжера для замовлення та доставки в ресторанах.

1. Chatfuel: Популярний інструмент для створення ботів без програмування є Chatfuel. Через інтуітивний інтерфейс користувача ресторани можуть створювати власних ботів для прийняття замовлень та бронювань столиків і зворотнього зв'язку з клієнтами. Замовлення, оплата товару, вибір меню та отримання інформації про доставку. Окрім того, він конектиться з іншими сервісами та платіжними системами, що спрощує процес замовлення та доставки.

2. ManyChat: Ще однин популярний інструмент бота, який надає спосіб реалізації функціоналу замовлення та доставляти, є ManyChat. Це дозволяє ресторанам створювати спеціальних ботів із попередньо визначеними темами повідомлень, кнопками вибору та варіантами оплати. Окрім того, пропонує аналітику та статистику, що надає змогу ресторанам відстежувати ефективність своїх візків.

3. Chaufuel: Програма для створення ботів без програмування. Ресторани можуть створювати власних ботів, використовуючи простий інтерфейс користувача, щоб приймати замовлення та взаємодіяти з клієнтами. Chatfuel включає всі функції замовлення, оплати, вибору меню та інформації про доставку. Процес замовлення та доставки спрощені в результаті його взаємодії з іншими службами та методами оплати.

4. ManyChat: Бот, який спрощує замовлення та доставку. Ідальням надається можливість створювати персоналізованих ботів із попередньо визначеними темами повідомлень, кнопками вибору та варіантами оплати. Орім того, надає статистичну інформацію та аналітичні дані, які дозволяють ресторанам стежити за продуктивністю своїх візків.

Вибір залежатиме від потреб ресторану, бюджету, наявних технологій та інших речей. Кожне з цих рішень має як плюси, так і мінуси. Для того щоб

забезпечити найкращі результати та рівень задоволення клієнтів, потрібно провести дослідження та тестування, перш ніж обрати бота для замовлення та доставки в ресторані.

Перед ти як розробляти бота для реалізації замовлення, доставки та бронювання столиків у ресторані, потрібно проаналізувати рішення, які успішно працюють на ринку. Даний аналіз допоможе нам зрозуміти, що можна запозичити в інших ботів, які вже працюють на ринку, а також виявити можливості для абгрейду та визначити необхідні деталі для правильної реалізації проекту.

Ознайомившись з низкою існуючих ботів для замовлення та доставки їжі, було зроблено наступні висновки:

1. Наявність переліку страв з цінами , фотографіями та описом є важливим для наглядності та зручного вибору клієнтами .
2. Можливість авторизації та збереження особистої інформації для швидкого та зручного оформлення замовлення.
3. Інтуїтивний дизайн, щоб користувачі не губилися в складних меню і могли швидко оформляти замовлення.

Мені дуже сподобався бот під назвою “«Roll Club »” . Тому, що в даному боті успішно реалізувані вимоги, що зазначені вище, а саме реалізований зручний вибір страв, можливість авторизації та простий інтерфейс.

1.2 Розробка Telegram-бота для автоматизації замовлень у закладах харчування

Основою проекту є мова програмування Python та бібліотека telebot для реалізації взаємодії з користувачами..

Головна мета проекту є забезпечення зручного та швидкого способу замовлення їжі та напоїв з використанням бота. Користувачі зможуть переглядати доступний асортимент страв, салатів та напоїв, обирати позиції,

вказувати кількість, а також оформлювати замовлення з доставкою на вказану адресу та бронювати столик у закладі.

Функціональні можливості боту включають:

1. Відображення основного меню з товарними категоріями: страви, салати, напоями , яке реалізоване через списки. Кожен вид страви у списку зберігає дані: назву, фото, ціну. (Рисунок 1.1.) Все це висвічується після натискання кнопки на одну із категорій .

```
menu = {
    'Страви': {
        'Вареники': {'price': 50, 'photo': 'photos/varenyk.jpg'},
        'Деруни': {'price': 70, 'photo': 'photos/deruny.jpg'},
        'Борщ': {'price': 60, 'photo': 'photos/borshch.jpg'},
        'Солянка': {'price': 65, 'photo': 'photos/solyanka.jpg'},
        'Холодник': {'price': 55, 'photo': 'photos/holodnik.jpg'},
        'Піца Маргарита': {'price': 120, 'photo': 'photos/pizza_margarita.jpg'},
        'Піца 4 Сири': {'price': 150, 'photo': 'photos/pizza_4cheese.jpg'},
        'Піца Пепероні': {'price': 140, 'photo': 'photos/pizza_pepperoni.jpg'},
        'Картопля фрі': {'price': 45, 'photo': 'photos/french_fries.jpeg'},
        'Картопля по-селянськи': {'price': 50, 'photo': 'photos/country_potatoes.jpg'}
    },
    'Салати': {
        'Цезар': {'price': 80, 'photo': 'photos/salat_cezar.png'},
        'Олів\'є': {'price': 90, 'photo': 'photos/salat_olive.jpg'},
        'Салат Айзберг': {'price': 75, 'photo': 'photos/salat_iceberg.jpg'},
        'Грецький': {'price': 85, 'photo': 'photos/greek_salad.jpg'},
        'Вінігрет': {'price': 70, 'photo': 'photos/vinigret.jpg'}
    },
    'Напої': {
        'Кока-Кола': {'price': 30, 'photo': 'photos/photo_koka_cola.jpg'},
        'Фанта': {'price': 30, 'photo': 'photos/photo_fanta.jpg'},
```

Рисунок 1.1 - Приклад написання списків

2. Зберігання даних реалізоване через списки. Кожна створена категорія містить список страв на замовлення із назвою, ціною та зображення страви . (Рисунок 1.2.)

```
menu = {
    'Страви': {
        'Вареники': {'price': 50, 'photo': 'photos/varenyk.jpg'},
        'Деруни': {'price': 70, 'photo': 'photos/deruny.jpg'},
        'Борщ': {'price': 60, 'photo': 'photos/borshch.jpg'},
        'Солянка': {'price': 65, 'photo': 'photos/solyanka.jpg'},
        'Холодник': {'price': 55, 'photo': 'photos/holodnik.jpg'},
        'Піца Маргарита': {'price': 120, 'photo': 'photos/pizza_margarita.jpg'},
        'Піца 4 Сири': {'price': 150, 'photo': 'photos/pizza_4cheese.jpg'},
        'Піца Пепероні': {'price': 140, 'photo': 'photos/pizza_pepperoni.jpg'},
        'Картопля фрі': {'price': 45, 'photo': 'photos/french_fries.jpeg'},
        'Картопля по-селянськи': {'price': 50, 'photo': 'photos/country_potatoes.jpg'}
    },
```

Рисунок 1.2 - Категорія в списку

3. Користувач має змогу вибрати категорію товару, переглянути список доступних товарів. Ось як це повинно виглядати в самому боті (Рис. 1.3):

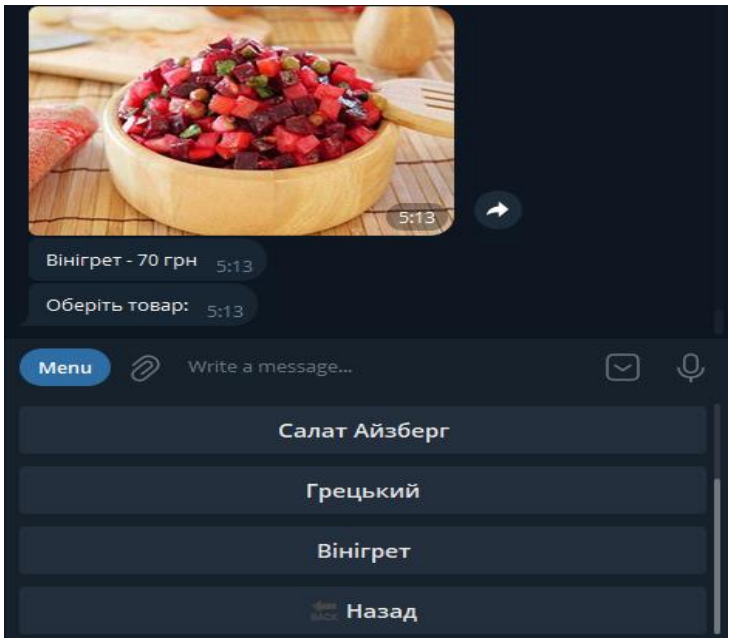


Рисунок 1.3 - Категорії та асортимент

4. Далі користувач повинен обрати конкретну позицію і вказати кількість одиниць або порцій, яку бажає замовити. Виглядатиме це наступним чином (рис. 1.4.)

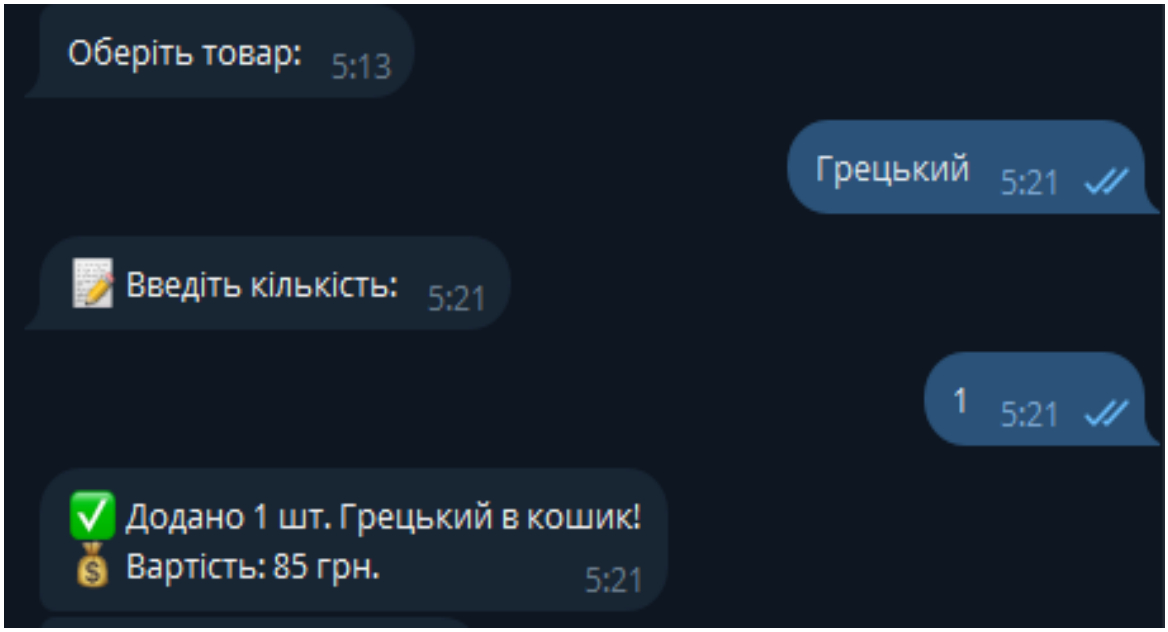


Рисунок 1.4 - Товар і корзина

5. Замовлені товари додаються в кошик разом з інформацією про ціну, кількість та загальну вартість, щоб перейти до кошика, потрібно повернутись в головне меню і нажати кнопку “Кошик”.

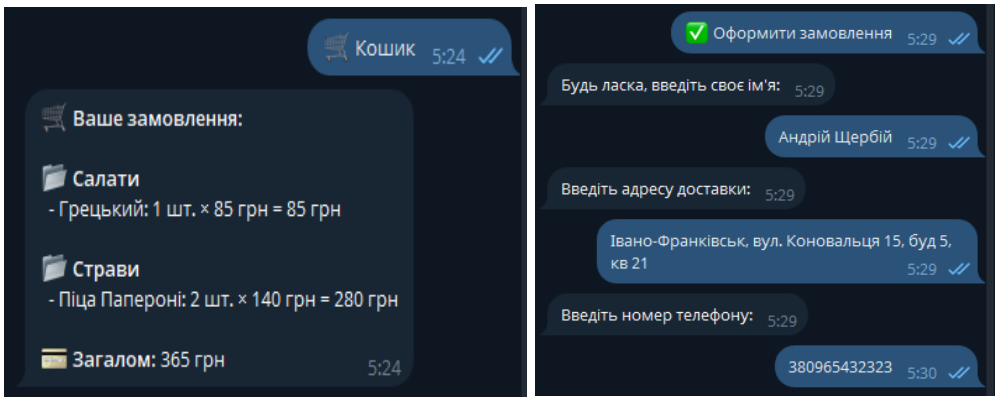


Рисунок 1.5 - Оформлений ордер

6. Користувач після того як перейшов у корзину натискає кнопку “Оформити замовлення” та вводить дані (ПІБ, адресу доставки, номер телефону) і відправити замовлення. Ось так це буде виглядати (Рисунок 1.5. – 1.6.)

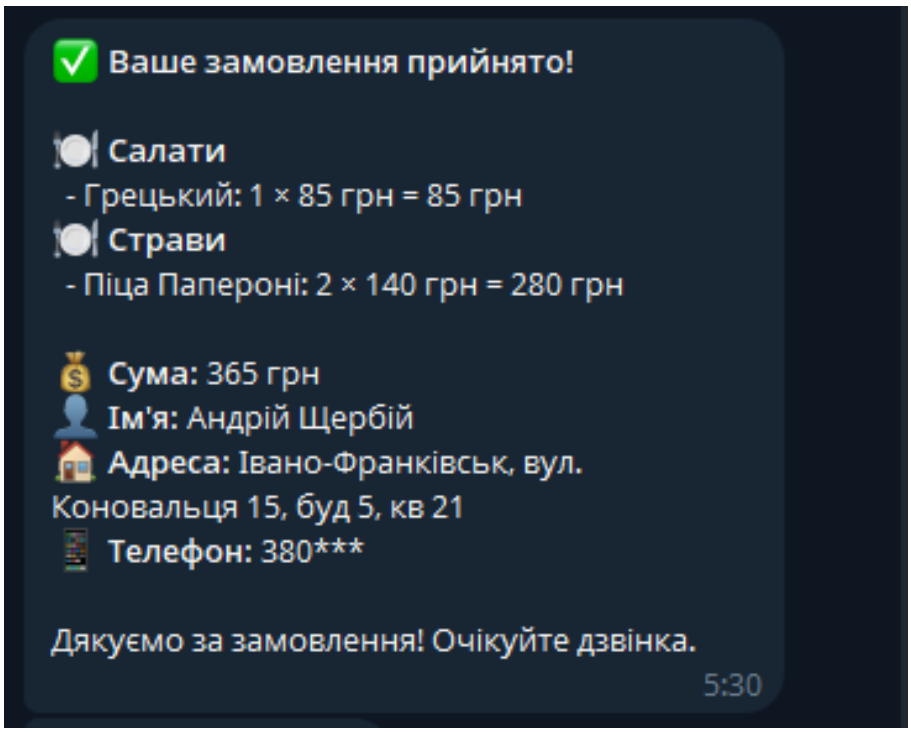


Рисунок 1.6 - Продовження Рисунку 1.5. Оформлений ордер

7. Замовлення надсилається адміністратору або в робочий чат пересоналу у вигляді повідомлення з деталями замовлення.

8. Після оформлення замовлення кошик і дані користувача очищаються. Про це буде свідчити цей текст від бота:

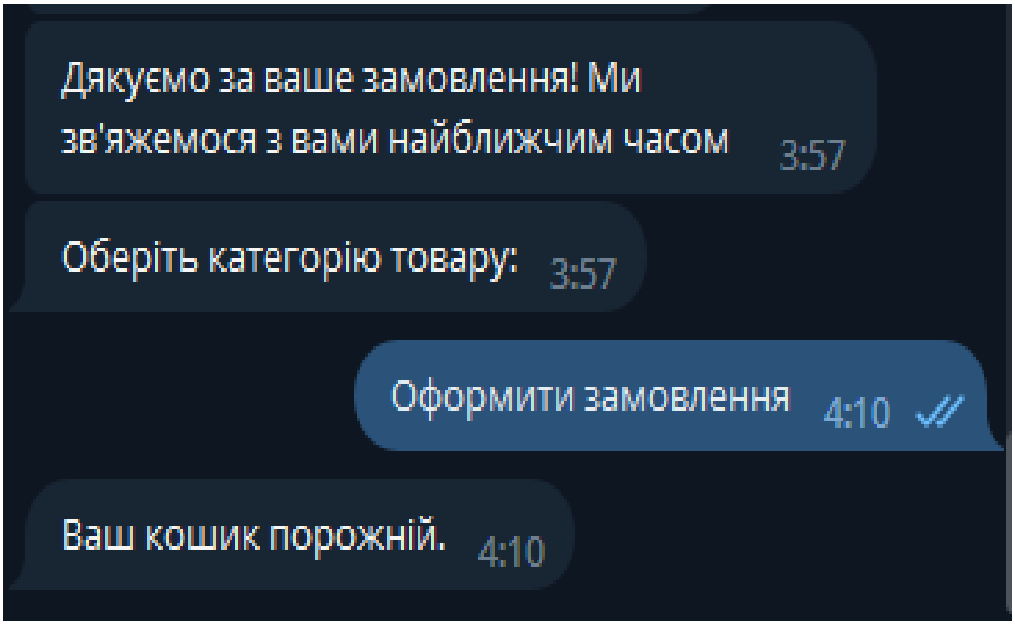


Рисунок 1.7 - Порожня корзина

9. Користувач має можливість залишити відгук про сервіс. Даний процес виглядатиме наступним чином (рис. 1.8.)

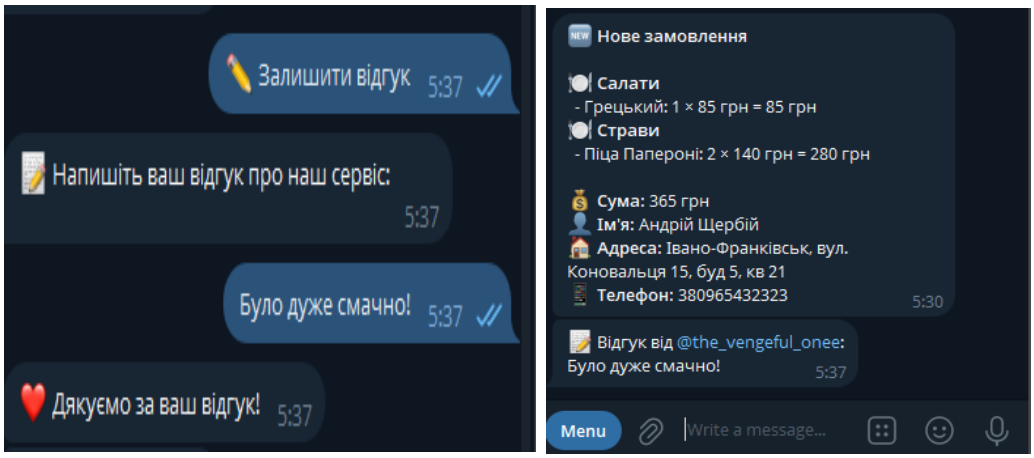


Рисунок 1.8 - Відгук користувача

У сферу застосування даного проекту входять ресторани, кафе та інші заклади громадського харчування, які бажають надати своїм клієнтам можливість замовити їду в зручний та ефективний спосіб. Використання бота дозволяє залучити велике коло користувачів тому, що даний меседжер є популярним та доступним на різних платформах.

Проект автоматизує процес оформлення замовлень та бронювання столиків і спрощує взаємодію клієнтів із закладами харчування. Його можна використовувати як основний інструмент для замовлення їжі та напоїв або як додатковий для покращення досвіду клієнтів.

Також цей програмний проект може сприяти збільшенню продажів та популярності закладу тому, що забезпечує зручний спосіб замовлення.

1.3 Специфікація функціональних вимог

Оскільки проект розроблений з мінімальними затратами, що є основною цілю його швидкої розробки , то по суті наявна всього лиш одна роль в боті і це – клієнт . Оскільки в проекті реалізовано відправлення оформлених клієнтами

замовлень та бронювань в чат персоналу закладу, відповідно й відгуків на канал.

Ролі користувачів:

- Звичайний користувач (User)
- Працівник закладу (User)
- Адміністратор (Admin)

Характеристики ролей:

Користувач має доступ до перегляду перегляду категорій, страв та їхніх параметрів.

Працівник закладу отримує оформлений ордер від клієнта на свій акаунт і обробляє його , а саме зв’язується з клієнтом по даних , що він ввів і уточнює деталі замовлення.

Адміністратор фіксить баги , вносить зміни в програмний код.

Описав функціональні можливості, які повинні бути для користувача, щоб він зміг оформити замовлення за допомогою застосунку оскільки фокус в більшості на ньому. А саме щоб користувач міг зручно здійснювати замовлення.

1.3.1 Use-Case

Use-Case – простий інструмент, що відіграє велике значення у світі програмної інженерії та використовується для проектування програмного забезпечення, а саме для опису взаємодії між акторами (користувачами або системами) і системою, яка виконує певні дії для досягнення поставлених результатів. Він визначає, як система має працювати в реальних ситуаціях, як користувачі взаємодіятимуть із системою та які очікувані результати. Коротко кажучи він візуалізує взаємодію акторів з системою. Приклад Use-Case на Рисунок 1.9.

Рисунок 1.9 - Приклад Use-Case на основі рішень повязаних з освітою

Основні переваги використання use-case в розробці ПЗ:

1) **Комунікація**: дозволяє спілкуватися з клієнтом, користувачами та іншими учасниками системи, використовуючи спільну мову. Це допомагає зрозуміти, як користувачі використовують систему та які їхні вимоги.

2) **Вимоги**: сценарій використовується для визначення, опису та організації системних вимог, що дає змогу визначити функціональність системи, ролі користувачів та їх взаємодію з системою.

3) **Проектування**: допомагає в проектуванні архітектури системи, що дає змогу ідентифікувати ключові компоненти, взаємодію між ними та розділити відповідальність між різними частинами системи.

4) **Тестування:** сценарій використання може бути використаний як основа для тестування системи, що дозволяє перевіряти, чи відповідає система вимогам користувачів та їх очікуванням.

5) **Планування:** варіанти використання дозволяють використовувати інтерактивний підхід до розробки, а саме коли кожен сценарій можна розглядати окремо, що допомагає керувати розробкою та планувати ресурси.

Use-Case є важливим інструментом для аналізу системних вимог і розробки програмного забезпечення. Він дозволяє визначити функціональні та нефункціональні вимоги, встановити послідовність дій і взаємодій, а також перевірити, чи відповідає система потребам юзерів.

Усі варіанти використання мають три основні елементи:

Актор - особи, системи або зовнішні компоненти, котрі взаємодіють із системою. Ними являються користувачі, адміністратори, зовнішні служби або інші системи.

Мета - ціль або завдання, яких потрібно досягти шляхом виконання конкретного варіанту використання. Вказує, чого актор (користувач або система) повинна досягти або отримати після виконання даного сценарію використання.

Система - набір процесів, етапів, функціональних вимог та їх очікуваної поведінки, котрі спрямовані на досягнення кінцевої мети та взаємодії з акторами.

У цій ситуації я створив загальну діаграму прецендентів для можливого функціоналу для кожної групи користувачів. (Рисунок 1.10.)

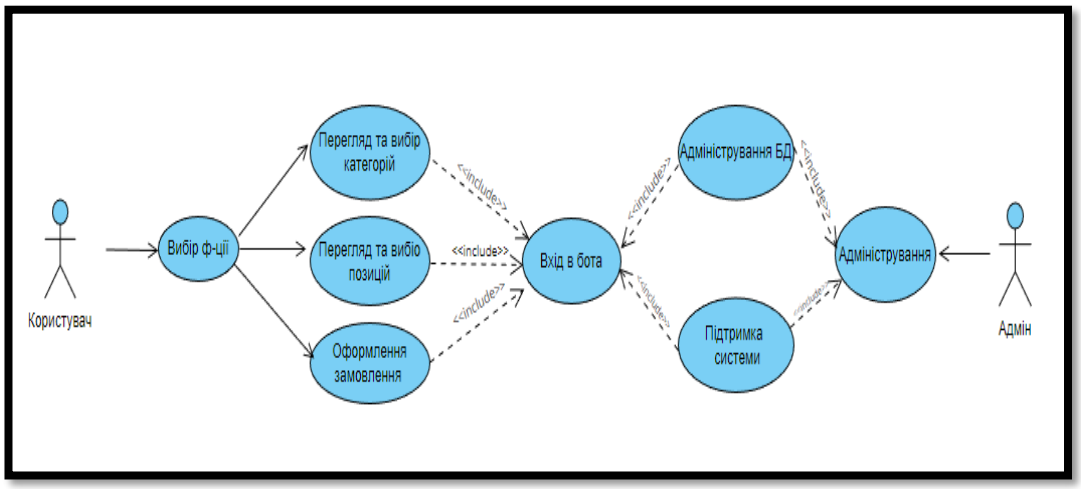


Рисунок 1.10 - Загальний Use-Case для бота

На Рис. 1.11. зображено весь наявний функціонал користувачів. А саме :

- **Користувач :** вибір категорії , вибір товару та його кількості, добавлення в корзину, перехід в корзину,оформлення замовлення.

- **Адміністратор:** підтримка БД, підтримка програмної частини.

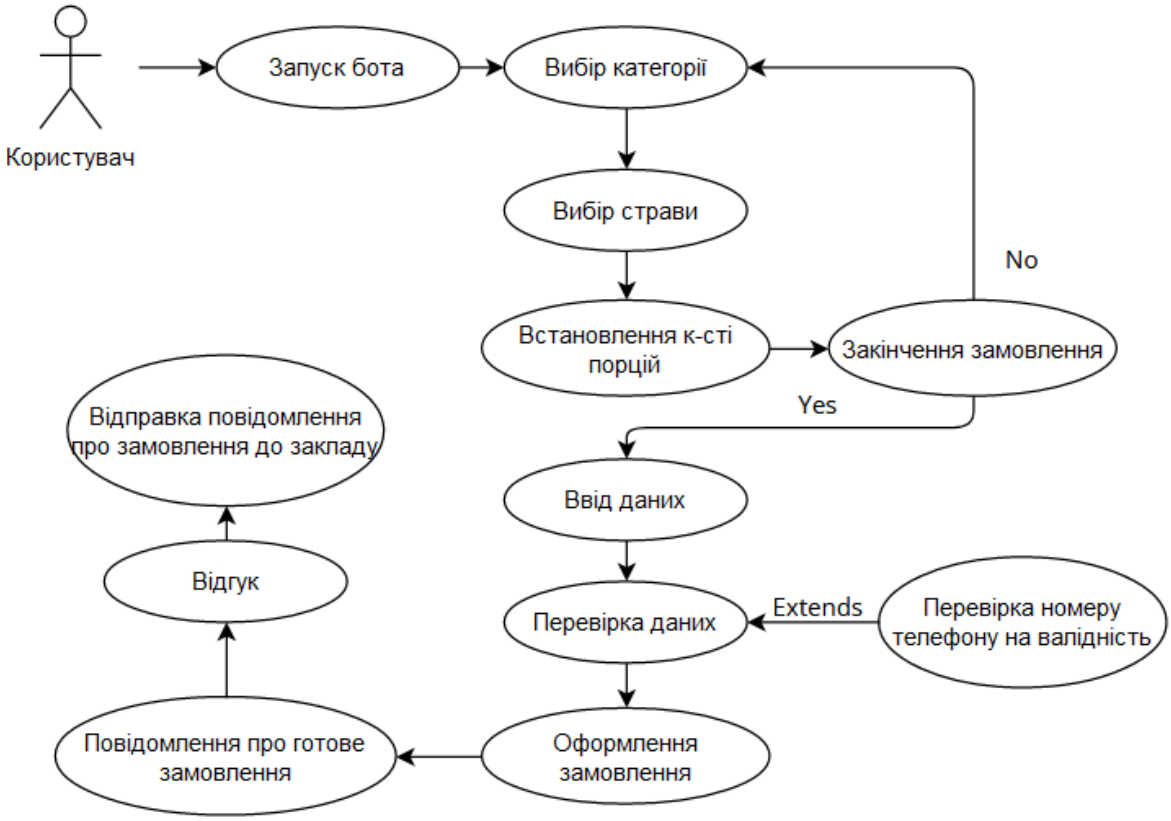


Рисунок 1.11 - Use-Case оформлення замовлення.

Короткий опис: Користувач створює нове замовлення.

Опис дій:

1. Користувач запускає бота.

2. Вибирає потрібну категорію.

3. Вибирає страву.

4. Користувач встановлює к-сть порцій.

5.1. Якщо замовлення не закінчене, то переходить на крок 2.

5.2. Якщо закінчене, то переходимо на наступний крок.

6. Ввід даних.

7. При коректному вводі даних система видає повідомлення про готове замовлення.

8. При некоректному виводиться повідомлення та користувач може ввести дані повторно.

9. Користувач може залишити відгук.

10. Система відправляє повідомлення про нове замовлення на канал закладу.

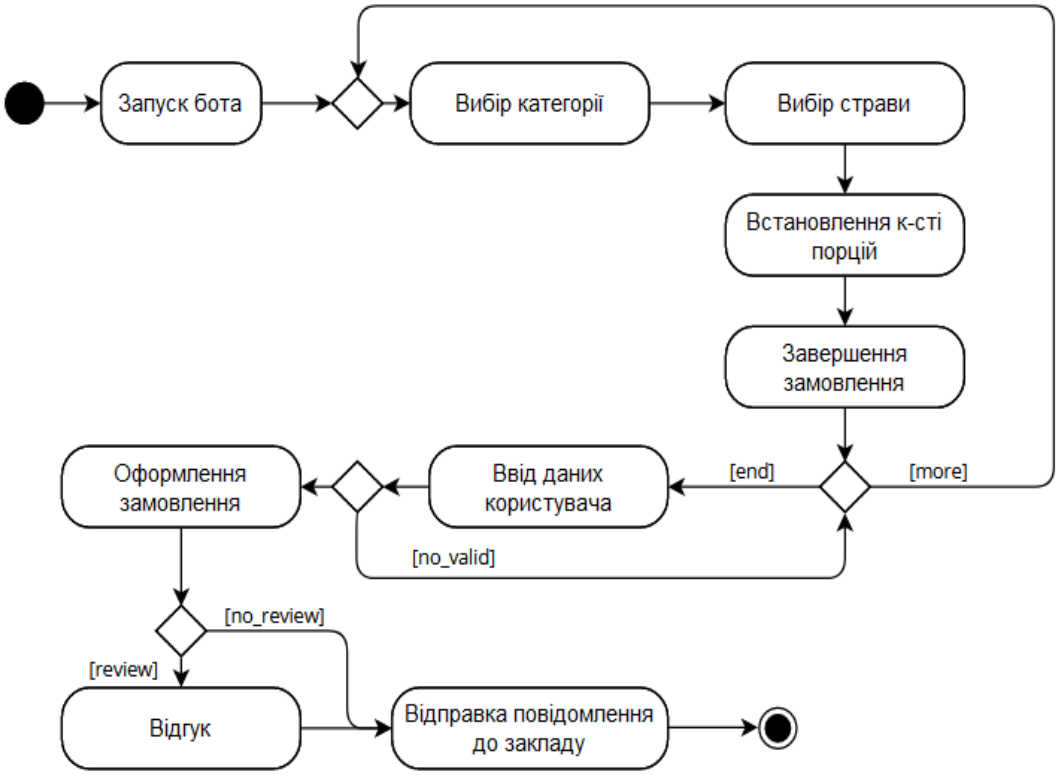


Рисунок 1.12 - Діаграма діяльності.

Опис :

1. Клієнт нажимає / start :

- висвічує меню

2. Далі він заходить в категорію

2.1. Обирає товар

2.2. Кількість порцій

- Висвічує “Товар добавлено в корзину”

3. Нажимає кнопку “назад”

4. Обирає по аналогічному алгоритму товар з інших категорій.

5. Нажимає кнопку “Оформити замовлення”

- Висвічує текст “Будь ласка, введіть своє ім'я:”

6. Клієнт вводить ім’я та відправляє.

- Висвічується текст: ” Будь ласка, введіть адресу доставки: ”

7. Клієнт вводить адресу.

- Висвічується текст “Будь ласка, введіть номер телефону:”

8. Клієнт ввів букви або більше 12-и цифр

- висвічує текст “Невірний формат номера телефону. Введіть тільки цифри, не більше 12 символів.”

9. Клієнт ввів номер у правильному форматі.

- висвічує чек клієнта з його замовленням і даними.
- висвічує текст “Дякуємо за ваше замовлення! Ми зв'яжемося з вами найближчим часом”.

10. Клієнт клікає на кнопку “Залишити відгук”

11. Вписує текст

- висвічує “Відгук від користувача @нікнейм:

ТЕКСТ..”

1.4 Висновки по розділу

У результаті проведеного аналізу сучасних рішень було визначено основні функціональні вимоги до Telegram-бота для автоматизації процесу замовлень у закладах харчування. Запропонований бот дозволяє ефективно оптимізувати обробку замовлень і взаємодію з клієнтами, забезпечуючи зручність, швидкість та сучасний цифровий підхід до обслуговування.

РОЗДІЛ 2. ПІДГОТОВКА ДО РОЗРОБКИ ТА ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

У даному розділі я описую підготовку до розробки та всі використані технології. Теж описую використані паттерни розробки та інше.

Проекти розроблялись в середовищі розробки: PyCharm 2025 1.1.1 , на платформі Windows 10.

2.1 Аналіз вибору Python як основної мови, опис його переваг, а також характеристику ключових бібліотек

У даній роботі бот створений на зручній та багатогранній мові програмування Python з використанням фреймворка для розробки боту для популярного меседжеру - Telebot, що надає зручний інтерфейс для роботи з API обраного нами месенджера. Крім того, бот використовує бібліотеку types - модуль, який містить класи і типи даних, пов'язані з Telegram Bot API. Він містить об'єкти, такі як ReplyKeyboardMarkup, що дають можливість створювати різноманітні клавіатури для бота, та інші класи для роботи з повідомленнями, фотографіями, аудіо, відео. Модуль “re” – вбудований модуль в Python для роботи з регулярними виразами, який забезпечує функціональність для роботи зі стрічками та пошуку певних шаблонів у тексті. Модуль “datetime” – компонент стокової бібліотеки Python для роботи з датою та часом. Містить функції поточного часу, парсингу рядків та форматування. Також модуль “apihelper” являється внутрішнім модулем бібліотеки, що має низькорівневий діапазон функцій для HTTP- запитів до API обраного нами меседжеру.

Перевагами використання Python для подібних або аналогічних задач є простота та зручність, а саме тому, що порівняно із іншими мовами програмування Python має найпростіший і найзрозуміліший синтаксис , що робить його легшим для вивчення і використання та дозволяє розробникам

швидко створювати ботів і отримати у висновку зрозумілий код. Також однією з переваг даної мови програмування є велика та активна спільнота розробників, яка має велике значення у дослідженні чи повноцінному її вивченні. А саме плюс в тому, що завжди є можливість отримати підтримку , пораду або знайти готові бібліотеки та модулі під потрібні рішення. Масштабна екосистема Python дає нм широкий вибір сторонніх бібліотек та модулів. Ця мова програмування крос-платформенна тому, що підтримується на усіх попуярних OS : Windows, macOS і Linux. Також є можливість інтеграції з сторонніми сервісами . Ще можна виділити перевагу розширюваності, а саме наявність великої кількості функцій і можливостей, що дає змогу розробникам розширювати функціональність своїх ботів для конкретних задач. Узагальнено, Python - це потужний та зручний інструмент для розробки ботів, що надає широкий діапазон можливостей і спрощує процес створення ботів на будь-який смак.

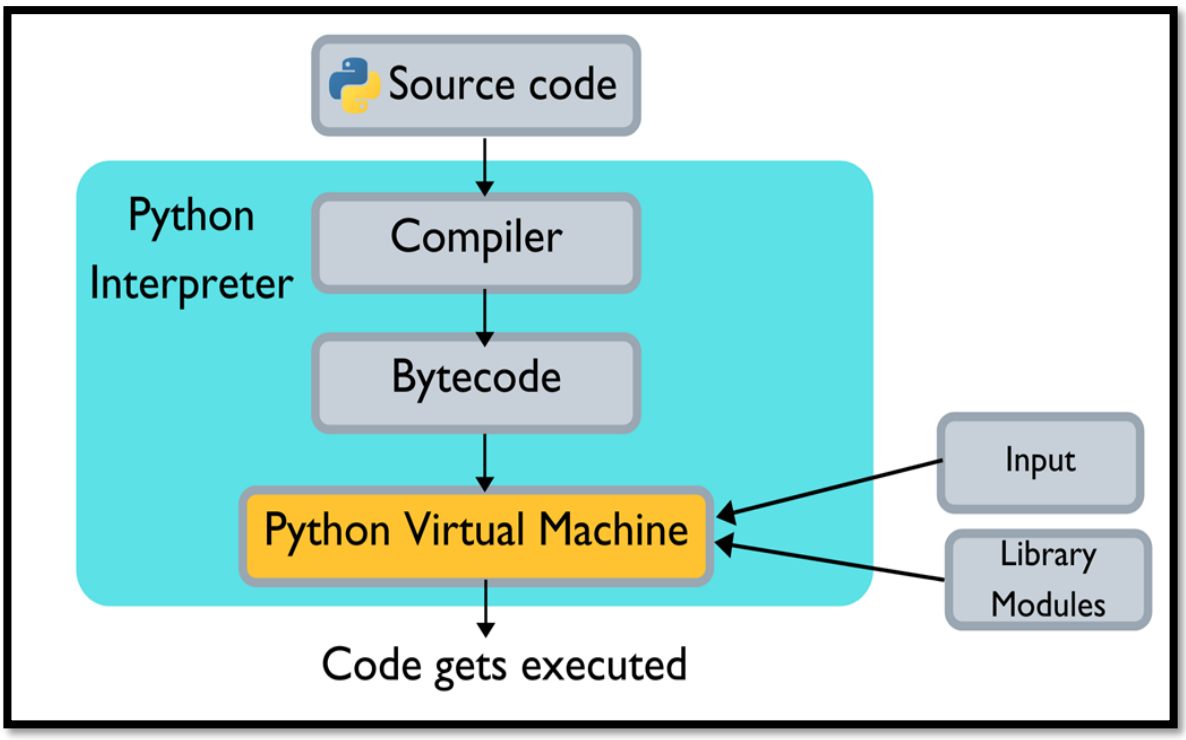


Рисунок 2.1 - Компіляція в Python

Як наслідок, вивчення мови та створення ботів легше та доступніше для

новачків.

2.2 Використання PyCharm IDE

Найкращим і найпоширенішим інтегрованим середовищем розробки (IDE) для програмування на Python є PyCharm. Щоб розробники створювали, налагоджували та керували своїми проектами Python ефективніше та ефективніше, він пропонує широкий діапазон функцій та інструментів. У цьому пункті ми розглянемо деякі з найкорисніших функцій PyCharm і те, як ними користуватися.

1. Запуск проекту – спрощує процес запуску нових проектів Python. Ви маєте змогу вибрати тип проекту (наприклад, консольна програма, веб-програма або бібліотека), версію Python для використання та налаштування середовища розробки, коли починаєте новий проект.

2. Потужний редактор коду, а саме являється однією з ключових переваг PyCharm є його хороший редактор коду, який пропонує зміни, підсвічування синтаксису, автоматичне доповнення коду, відступи та багато інших функцій. Редактор підтримує роботу над збереженими проектами та дозволяє відкривати та редагувати декілька файлів одночасно. Також нещодавно було добавлено штучний інтелект PyCharm Goes AI, що в рази пришвишує кодинг та відладку коду.

3. Налагоджувач, який дозволяє відстежувати та виправляти неполадки у вашому коді. Ви можете покроково проходити код, створювати контрольні точки, переглядати значення змінних і спостерігати за виконанням програми. У результаті налагодження стає набагато легшим, як і процес розробки.

4. Полегшує керування залежностями та віртуальними середовищами. Використовує програму керування пакетами, по типу pip або conda для того щоб керувати проектом. Безпосередньо з IDE можна з легкістю додавати, видаляти та оновлювати пакети. Крім того, PyCharm дозволяє створювати та

використовувати віртуальні середовища, які відокремлюють залежності проектів один від одного.

5. Робота з SVC(системами контролю версій): Git, SVN і Mercurial – це лише одні з багатьох систем контролю версій, з якими PyCharm взаємодіє. Це дає змогу виконувати коміти, отримувати та об’єднувати зміни, створювати гілки, легко працювати з віддаленими сховищами та робити багато іншого. Тобто з IDE ви можете працювати з іншими розробниками та відстежувати зміни у своєму коді.

6. Автоматизація завдань, що дає змогу використовувати інструменти збирання та запуску такі, як Makefile, Bash або сценарій Python, для автоматизації дій. Для будь-якого процесу, який використовується у вашому проекті, включаючи збірку, тестування та розгортання, можна створювати власні завдання.

7. Інтеграція з іншими інструментами - можна працювати з різноманітними інструментами розробки, включаючи лінтери коду, фреймворки тестування, фреймворки CI/CD (безперервна інтеграція/безперервне розгортання) тощо. У результаті ви можете використовувати всі інструменти, необхідні для вашого проекту, прямо з PyCharm.

2.3 Технології для розробки ботів на Python

Для початку робити потрібно мати певний набір знань та навиків. Тому розлянемо що саме нам потрібно для професійної та повноцінної розробки від початку до кінця, щоб не виникло ніяких нюансів:

1) Основи Python:

- Python — це інтерпретована мова програмування, що означає, що код виконується рядок за рядком без необхідності компіляції.
- Підтримує об’єктно-орієнтований підхід, який дозволяє організовувати код у класи та об’єкти для кращої організації та повторного використання коду.

2) Встановлення та налаштування середовища розробки:

- Для розробки ботів даного характеру потрібно встановити інтерпретатор Python та інтегроване середовище розробки, у нашому випадку PyCharm.
- Після того як встановили Python ви можете встановити необхідні залежності та бібліотеки, наприклад telebot, для створення ботів.

3) Робота з TG Bot API обраного нами меседжеру:

- TG Bot API надає набір інструментів для створення та взаємодії з ботами в TG(меседжер).
- Потрібно створити бота та отримати токен API від BotFather у TG.
- Бібліотека telebot дає змогу зручно створювати та взаємодіяти з ботами, надсилаючи та отримуючи повідомлення, фотографії, відео та інші файли.

4) Основні моменти для роботи з ботом:

- Для того щоб відповідати на повідомлення, команди та події потрібно використовувати декоратори з бібліотеки telebot, такі як @bot.message_handler та @bot.command_handler.
- Є можливість створювати різні функції обробки для різних типів повідомлень або команд.
- Завдяки методам, таким як bot.send_message, bot.send_photo, bot.send_video та інші, можливо відправляти повідомлення, фотографії, відео з бота до користувачів.
- Використавши bot.send_message, bot.send_photo, bot.send_video та інші, можна надсилати повідомлення, фотографії, відео від свого бота користувачам.

5) Розмітка та кнопкаи:

- TG Bot API дає можливість створювати розмітку повідомлень за допомогою HTML-подібного синтаксису, при цьому дозволяючи форматувати текст, додавати посилання тощо.

- Кнопки надають можливість створювати інтерактивні меню для користувачів. Бібліотека telebot має усі потрібні функції для створення кнопок, реагування на натискання та обробки відповідей користувачів.

6) Зберігання даних:

- Для зберігання даних бота зазвичай використовують такі бази даних, як SQLite, MySQL, PostgreSQL, MongoDB. У Python є різні бібліотеки для роботи з базами даних, наприклад sqlite3 або MySQLdb.
- Присутня можливість зберігати дані як текстові файли або використовувати інші формати, по типу JSON.

7) Розгортання бота:

- Після завершення стадії розробки щоб запустити бота до використання, можна розгорнути його на сервері або в хмарній службі. Для прикладу є такі хмарні платформи, як Heroku або AWS, що спрощують розгортання бота.

8) Безпека:

- Щоб захистити свого бота від несанкціонованого доступу, ви можете використовувати маркери та інші методи автентифікації
- Також важливо дотримуватися поширених принципів безпеки ПЗ, а саме перевірка введених даних та обмеження прав доступу до функцій бота.

2.4 Теоретичні аспекти розробки ботів з Python

Існує декілька популярних бібліотек, які допомагають в розробці ботів на Python. Декілька з них:

- **Python-TG-bot –** дана бібліотека дозволяє легко та швидко розробляти ботів для платформи TG. Надає зручний інтерфейс для отримання повідомлень, відправлення відповідей та керування ботом.
- **Discord.py –** бібліотека для створення ботів для Discord - платформи для спілкування геймерів. Надає широкий спектр функціоналу, а саме

можливість взаємодіяти зі службами Discord, включаючи отримання повідомлень, керування серверами та інше.

- **Tweepy** – бібліотека для створення ботів на платформі Twitter, яка надає інтерфейс для взаємодії з Twitter API. Дозволяє отримувати стрічку новин, відправляти твіти, отримувати повідомлення тощо.

- **Pyautogui** – дана бібліотека дає змогу автоматизувати дії на екрані ПК. Використовується для розробки ботів, які взаємодіють з додатками та веб-сайтами, натискають кнопки, вводять дані та інше.

Розглянемо детальніше кожну з цих бібліотек:

1) Python-telegram-bot — поширена бібліотека для розробки ботів для платформи TG, що надає зручний інтерфейс для взаємодії з TG Bot API, який дозволяє отримувати повідомлення, надсилати відповіді, керувати групами та багато іншого. Основні особливості бібліотеки включають:

- **Простота використання:** має чистий та зрозумілий синтаксис, який спрощує розробку. Надає зручні класи та методи для реагування на різні типи повідомлень та подій.

- **Висока функціональність:** підтримує багато функціональних можливостей, а саме таких, як робота з клавіатурою, обробка фотографій, відео, аудіо та документів, використання inline-режиму та інше.

- **Асинхронність:** підтримує асинхронний режим, що відповідно дає змогу обробляти багато запитів одночасно та забезпечує ефективну та стабільну роботу бота в умовах навантажень.

- **Підтримка каналів та груп:** дозволяє керувати каналами та групами, відправляти повідомлення, налаштовувати розсилки та контролювати доступ до вмісту.

Python-TG-bot є однією з найпопулярніших бібліотек для розробки ботів для TG, та має велике ком’юніті розробників, яке надає підтримку та навчальні матеріали.

2) Discord.py - це бібліотека для розробки ботів на платформи Discord.

Discord - це популярна платформа, яка є кращим аналогом Skype для геймерів, а Discord.py надає зручний інтерфейс для взаємодії з Discord API. Особливості бібліотеки включають:

- **Простота використання:** пропонує простий та зрозумілий синтаксис, що дозволяє швидко створювати ботів для Discord. Надає зручні класи та методи для реагування на події, відправлення повідомлень, керування серверами тощо.
- **Гнучкість:** дозволяє реалізовувати різноманітні функції, а саме такі як обробка повідомлень, реагування на смайлики, керування голосовими каналами, створення команд тощо.
- **Підтримка багатопоточності:** підтримує асинхронний режим та багатопотокову обробку подій, що дозволяє ботам ефективно взаємодіяти з серверами Discord та обслуговувати багато запитів одночасно.
- **Розширені функції:** надає доступ до різноманітних функцій Discord, а саме ролі, дозволи, категорії каналів та можливість створювати інтерактивні команди тощо.

Discord.py – популярний вибір для розробки ботів для на платформу Discord. Також має активну спільноту розробників, які надають підтримку та документацію.

tweepy — бібліотека для розробки ботів для платформи Twitter. Забезпечує швидкий та простий спосіб взаємодії з Twitter API, дозволяючи отримувати дані, надсилати твіти, взаємодіяти з користувачами та інше. Особливості даної бібліотеки:

- **Простота використання –** має простий у використанні інтерфейс, який полегшує розробку ботів. Надає зручні класи та методи отримання стрічки новин, надсилання твітів, пошуку користувачів тощо.
- **Повна підтримка Twitter API:** бібліотека надає доступ до всіх основних функцій Twitter API, включаючи доступ до твітів, користувачів, хештегів, зображень, геолокації тощо.
- **Потужні можливості фільтрації:** tweepy дозволяє використовувати

різноманітні фільтри для отримання й обробки лише необхідних даних, наприклад фільтрування за користувачем, геолокацією, мовою тощо.

- **Обробка потокових подій:** бібліотека дозволяє отримувати потокові дані з Twitter API і реагувати на різні події в реальному часі.

Tweepy є популярним вибором для розробки ботів для Twitter і має активну спільноту розробників, яка надає підтримку та приклади коду.

3) **PyAutoGUI** — це бібліотека для автоматизації рухів миші та клавіатури на різних платформах, включаючи Windows, macOS і Linux. Він надає можливість контролювати взаємодію з елементами на екрані, наприклад клацання, перетягування, введення тексту тощо. Основні особливості бібліотеки:

- **Рухи миші: PyAutoGUI** дозволяє керувати рухами миші на екрані, включаючи рух курсору, клацання лівою та правою кнопкою миші, подвійне клацання, прокручування тощо. Він дозволяє автоматизувати взаємодію з програмами та веб-сторінками.
- **Клавіатура:** бібліотека надає можливість імітувати натискання та відпускання клавіш на клавіатурі. Ви можете вводити текст, надсилати гарячі клавіші, взаємодіяти з текстовими полями тощо. Це корисно під час автоматизації рутинних завдань або тестування програм.
- **Скріншоти:** PyAutoGUI дозволяє робити скріншоти та зчитувати пікселі зображення. Це дає можливість отримувати інформацію про кольори окремих ділянок екрана або виконувати розпізнавання об’єктів на зображеннях.
- **Кросплатформеність –** дозволяє використовувати його в Windows, macOS і Linux, що дає змогу створювати автоматизовані рішення, які працюють на різних операційних системах.

2.5 Висновки до розділу

У цьому розділі було проаналізовано підготовку до розробки Telegram-бота мовою Python, з використанням IDE PyCharm та відповідних бібліотек. Python

було обрано завдяки його простоті, зручному синтаксису, великій екосистемі та активній спільноті. Розгляд інструментів і теоретичних аспектів показав, що створення ботів вимагає знань у сфері роботи з API, бібліотеками, обробки даних і безпеки, а також дозволяє використовувати різноманітні патерни та технології для побудови гнучких та масштабованих рішень.

РОЗДІЛ 3. ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ TELEGRAM-БОТІВ: ВІД ВИБОРУ БІБЛІОТЕКИ ДО УПРАВЛІННЯ ВЕРСІЯМИ"

3.1 Порівняльна характеристика бібліотеки Telebot та її конкурентів для створення Telegram-ботів

Telebot — є однією з найпопулярніших бібліотек для створення TG-ботів на мові програмування Python. У неї в характеристиці ряд особливостей та переваг, які роблять її хорошим інструментом для реалізації проектів . Являється хорошим конкурентом відповідно до аналогів. Далі розглянемо особливості цієї бібліотеки.

Порівняння за ключовими критеріями:

1. Простота використання – має зрозумілий та простий інтерфейс, який дозволяє швидко розпочати розробку бота TG без особливих зусиль.

2. Багатомовність – підтримує різні мови програмування, включаючи Python 2.7 і Python 3.x.

3. Розширена взаємодія з користувачем – дає можливість створити клавіатури, кнопки, меню, параметри, наклейки та інші елементи, що полегшують взаємодію з користувачем.

4. Обробка різних типів повідомлень – підтримує обробку текстових повідомлень, зображень, аудіо та відео-файлів, голосових повідомлень, контактів, місць розташування та іншого.

5. Робота з базою даних – дозволяє зберігати та отримувати дані з бази даних для зберігання інформації про користувачів, замовлення та інші дані, пов’язані з ботом.

6. Асинхронна обробка повідомлень – підтримує асинхронний підхід до обробки повідомлень, що дозволяє боту працювати швидко й ефективно обробляти багато запитів водночас.

7. Підтримка розширень – має велику кількість розширень та plugins, які

дозволяють легко розширювати функціональність бота, додавати нові функції та інтегруватися з сторонніми сервісами.

Далі порівняємо Telebot з його найближчим конкурентом - python-TG-bot , що фактично являється аналогом та прямим конкурентом , але все ж по деяких пунктах має відмінність. А порівняємо їх згідно нашої основної задачі в даній дипломній роботі.

Telebot VS python-TG-bot:

1. Простота використання – Telebot володіє простим і зрозумілим синтаксисом, який полегшує розробку. Python-TG-bot також має чіткий синтаксис, але може бути складнішим у використанні порівняно з Telebot.

2. Функціональність – перелічені бібліотеки надають широкий спектр функцій для розробки бота нашого характеру, як одна, так і друга. Підтримують різні типи повідомлень, клавіатури, можливість роботи з базою даних та інші компоненти бота.

3. Розширені можливості – Python-TG-bot буде більш гнучкий у розробці ресторанного бота та має деякі додаткові функції, що дозволяє реалізувати більш складну логіку та взаємодію з ботом.

4. Документація та спільнота – обидві бібліотеки мають добре документований API та приклади використання. Також мають активні спільноти розробників, які надають підтримку та допомогу.

5. Налаштування – бібліотеки дозволяють налаштувати бота під конкретні потреби ресторану та розширити його можливості.

3.2 Використання систем зберігання даних у Telegram-боті: реляційні та нереляційні підходи

При розробці ботів на основі бібліотеки Telebot для Python для зберігання даних можна використовувати різні типи баз даних. Вибір бази даних залежить від ваших потреб та обмежень проекту. Нижче згадано про декілька можливих

варіантів роботи з базами даних в Telebot:

1) **SQLite** — легке вбудоване бездискове сховище даних без сервера, що зберігає всю базу даних в одному файлі на диску. Використання SQLite в Telebot доволі просте, а саме можна використовувати модуль sqlite3 для створення, читання, оновлення та видалення даних у базі даних SQLite. Є можливість зберігати інформацію про користувачів, їх налаштування, статус бота та інші дані, необхідні для роботи вашого бота.

2) **MySQL і PostgreSQL** — потужні серверні бази даних, які надають широкі можливості для зберігання й обробки даних. Для роботи з MySQL або PostgreSQL необхідно встановити відповідні драйвери та бібліотеки. Наприклад, ви можете використовувати mysql-connector-python для роботи з MySQL або psycopg2 для роботи з PostgreSQL. У базі даних можна зберігати складні дані, по типу структуровані таблиці з багатьма зв'язками, що особливо корисно, якщо вашому боту потрібно зберігати багато користувачів, повідомлень, налаштувань та інших даних, які можуть знадобитися у зберіганні та маніпуляції ними.

3) **MongoDB** — база даних NoSQL на основі документів. Зберігає дані як документи, що схожі на JSON. Розподілена база даних з можливістю роботи в режимі кластера. Для роботи з нею потрібно встановити офіційний драйвер pymongo. MongoDB, що надасть можливість зберігати динамічні дані, які можуть змінюватися з часом, і дозволяє виконувати швидкі запити для доступу до даних. Дана СУБД (Система Управління Базами Даних) особливо корисна для проектів із інтенсивним об'ємом даних або проектів, де потрібно зберігати незадокументовані дані.

4) **Redis** — відкрита база даних ключ-значення, яка зберігає дані в пам'яті для швидкого доступу. А також Redis надає можливості для зберігання різних типів даних, а саме таких як рядки, списки, хеші, набори та відсортовані набори. Щоб працювати з Redis встановлюється драйвер redis. Redis часто використовується для кешування даних, роботи з чергами повідомлень і зберігання тимчасової інформації, що може бути корисно для проектів, яким

потрібен швидкий доступ до даних.

Після бази даних потрібно буде створити підключення до бази даних у вашому боті. Для цього потрібно використовувати відповідні методи та класи, надані драйверами бази даних.

Вибір бази даних залежить від індивідуальних потреб та вимог проекту. Бази даних, які розглянуто вище, є поширеними варіантами, але є й кращі альтернативи, які можуть відповідати вашим потребам.

Важливо чітко оцінити вимоги вашого проекту, беручи до уваги обсяг даних, характер діяльності, швидкість, масштабованість і зручність використання даних. Інші бази даних, а саме такі як PostgreSQL, SQLite, та багато інших, окрім MySQL і MongoDB, можуть бути більш підходящими згідно поставлених потреб.

Для того щоб отримати найкращі результати, вибір вашої бази даних має базуватися на особливостях та специфікаціях проекту. розташування та порт вашої бази даних.

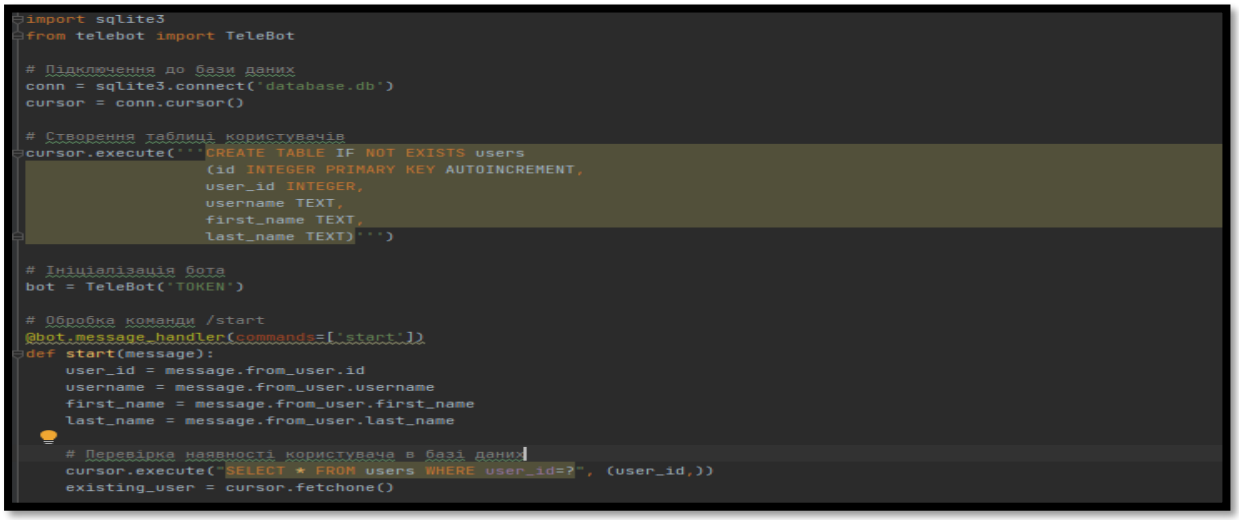
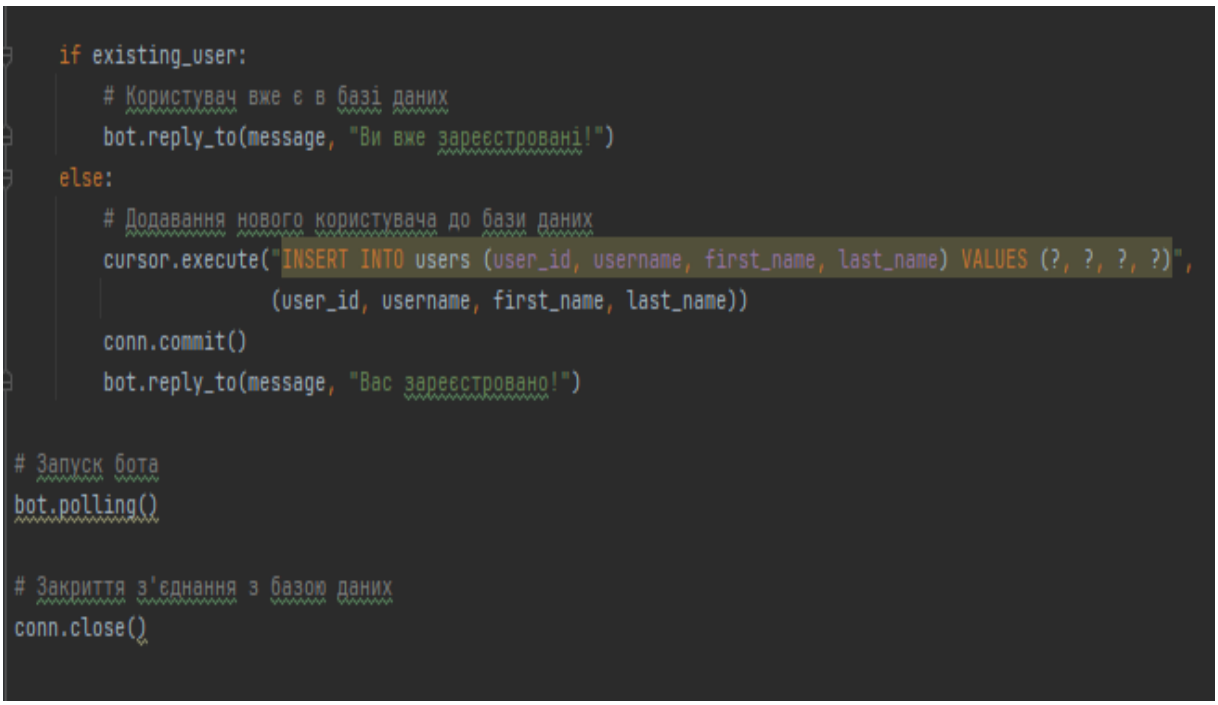


Рисунок 3.1 - Використання БД

Наведемо приклад використання бази-даних MySQL (Рисунки 3.1, 3.2.):



```
    if existing_user:
        # Користувач вже є в базі даних
        bot.reply_to(message, "Ви вже зареєстровані!")
    else:
        # Додавання нового користувача до бази даних
        cursor.execute("INSERT INTO users (user_id, username, first_name, last_name) VALUES (?, ?, ?, ?)",
                       (user_id, username, first_name, last_name))
        conn.commit()
        bot.reply_to(message, "Вас зареєстровано!")

# Запуск бота
bot.polling()

# Закриття з'єднання з базою даних
conn.close()
```

Рисунок 3.2 - Продовження Рисунку 2.2. Використання БД

Починаємо з імпорту потрібних модулів, а саме включаючи TeleBot і sqlite3, які використовуються для створення бота та взаємодії з базою даних SQLite.

За допомогою sqlite3.connect('database.db'), де 'database.db' — ім'я файлу бази даних, потім встановлюється підключення до бази даних SQLite. Після того як підключили створюємо курсор, який використовується для виконання інструкцій SQL щодо бази даних.(Рисунки 3.1-3.2.)

Потім створюємо таблицю юзерів оператором SQL CREATE TABLE IF NOT EXISTS, який генерує структуру таблиці зі стовпцями first_name, last_name, user_id та username. Запит не використовуватиметься, якщо таблиця вже є.

Потім використовуйте api-токен, який ви отримали від BotFather у TG, щоб ініціалізувати бота, ввівши bot = TeleBot("TOKEN").

Функція запуску обробляє команду /start та визначається під час стартового налаштування бота. На нього надсилаються дані про користувача, візповідно з user_id, username, first_name та last_name. Через запиту SQL SELECT визначається чи існує користувач в базі даних, і якщо так, то з’являється повідомлення “Ви вже зареєстровані!” повертається. Якщо ні, використовується

запит INSERT INTO SQL для додавання нового користувача до бази даних і повідомлення "Ви зареєстровані!" видається. Conn.commit() використовується для фіксації змін після додавання користувача до бази даних.

Завдяки bot.polling() бот працює та готовий отримувати повідомлення. Функція старту запускається, коли бот отримує команду /start для початку обробки.

Розглянемо приклад використання нереляційної бази даних – MongoDB . Її відмінність від минулої, що у реляційній - базі даних інформація зберігається у форматі таблиці та ретельно структурована та пов’язана з іншою інформацією. Кожен рядок і стовпець у таблиці представляє запис, а кожен стовпець представляє поле з певним типом даних. Інформація вноситься в кожну комірку відповідно до шаблону. Нереляційна містить дані без чіткої структури чи зв’язків між вмістом. База даних не має ієрархічних таблиць і наповнена різноманітними непов’язаними даними, а саме таким як зображення, відео та навіть публікації в соціальних мережах. Бази даних NoSQL не приймають запити SQL, на відміну від реляційних баз даних.

Але MongoDB має достатню кількість переваг у порівнянні з традиційними реляційними базами даних:

1. Гнучкість схеми даних – не вимагає визначеної структури даних. Можливість зберігати різні типи даних, не змінюючи дизайн бази даних тому, що кожен документ може мати окрему структуру.

2. Масштабування – розподіл даних у кластері серверів стає можливим завдяки функції, яку вмикає дана СУБД.

3. Швидкість – внутрішні механізми для оптимізації запитів та кешування, які забезпечують швидкість роботи з базою даних.

4. Реплікація та надійність – підтримка реплікації, що дозволяє робити копії даних на різних серверах. Наведено приклад використання нереляційної бази-даних MongoDB в комбінації з нашою бібліотекою для створення боту (Рисунок 3.3.):

```python
import telebot
from pymongo import MongoClient

# Підключення до бази даних MongoDB
client = MongoClient('mongodb://localhost:27017/')
db = client['telegram_bot']
collection = db['messages']

# Ініціалізація бота
bot = telebot.TeleBot('YOUR_TELEGRAM_BOT_TOKEN')


@bot.message_handler(func=lambda message: True)
def handle_message(message):
    # Зберігання повідомлення в базі даних
    message_data = {
        'chat_id': message.chat.id,
        'text': message.text,
        'from_user': {
            'id': message.from_user.id,
            'username': message.from_user.username,
            'first_name': message.from_user.first_name,
            'last_name': message.from_user.last_name
        }
    }
    collection.insert_one(message_data)

    # Відповідь користувачу
    bot.reply_to(message, 'Ваше повідомлення збережено.')


# Запуск бота
bot.polling()
```

Рисунок 3.3 - Використання MongoDB.

У даній програмі прописано підключення до локальної бази даних MongoDB, додається колекція повідомлень і зберігається кожне отримане повідомлення разом із даними користувача. Бот відповідає користувачеві після збереження повідомлення.

Перед тим як запускати цей код, повинен бути виконаний запуск локальної бази даних. Якщо потрібно, то можна змінити рядок підключення для MongoClient('mongodb://localhost:27017/'), щоб указати host і port бази даних.

3.3 Система контролю версій - Git.

Git — розподілена система керування версіями файлів та спільної роботи. Проект створив Лінус Торвальдс для керування розробкою ядра Linux, а сьогодні

підтримується Джуніо Хамано (англ. Junio C. Hamano). Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Програма є вільною і випущена під ліцензією GNU GPL версії 2.

Система спроектована як набір програм, спеціально розроблених з врахуванням їхнього використання у скриптах. Це дозволяє зручно створювати спеціалізовані системи керування версіями на базі Git або користувацькі інтерфейси. Система має ряд користувацьких інтерфейсів: наприклад, **gitk** та **git-gui** розповсюджуються з самим Git. Віддалений доступ до репозиторіїв Git забезпечується git-демоном (службою), SSH або HTTP сервером.

Налаштуваня Git – системи контролю версій (SCV) безпосередньо в середовищі програмування PyCharm – це процес інтеграції Git в PyCharm.

Система контролю версій - це система, що записує зміни у файл або набір файлів протягом деякого часу, так що ви зможете повернутися до певної версії пізніше.

Не потребуючи зовнішнього ПЗ чи терміналу, що дає можливість програмістам правильно керувати своїми проектами, використовуючи саме такі функції Git, як відстеження змін, коміти, розгалуження, злиття.

Під час використання командного терміналу середовища PyCharm або інтерфейс користувача є можливість виконувати операції Git. Наведено основні кроки в інтеграції Git з PyCharm:

1. Треба переконатись, що Git встановлений на ПК та налаштований на роботу з репозиторієм – перший крок до налаштування. Перехід безпосередньо до наступного кроку, якщо Git уже налаштовано.

2. Розгорнення проекту – у головному меню виберіть «Відкрити» або «Відкрити проєкт», щоб запустити проєкт у середовищі PyCharm.

3. Ініціалізація Git – за допомогою опції “VCS” у головному меню можна підключити Git, якщо проект ще не налаштовано як сховище. Потрібно обрати “Включити інтеграцію керування версіями” та “Git” як метод керування

версіями.

4. Додавання файлів до Git – є можливість додати файли до системи керування версіями після налаштування Git. Обираємо “Додати до Git” у меню після вибору файлів або папок, які ми хочемо додати. Окрім того, ми можемо додавати файли за допомогою вікна “Git” на панелі інструментів.

5. Коміт змін – після додавання файлів виникає потреба у збереженні змін закріпивши їх у сховищі. Вибираємо потрібні файли або каталоги, також “Закріпити” в контекстному меню або полі “Git”, а потім вводимо текст, де пишемо пояснення змін, які ми добавили. Потім зміни можна зберегти в сховищі.

6. Робота з розгалуженнями та злиттями можлива завдяки інструментам PyCharm. У PyCharm можна створювати нові гілки, перемикатися між гілками, об’єднувати зміни та вирішувати конфлікти злиття.

3.4 Висновки по розділу

У процесі розробки Telegram-ботів важливо обирати оптимальні інструменти, зокрема бібліотеку для роботи з Telegram API, систему зберігання даних відповідно до задач (реляційну чи нереляційну), а також систему контролю версій для зручного відстеження змін у коді. Використання бібліотеки Telebot, підтримка сучасних баз даних та інтеграція з Git у середовищі PyCharm забезпечують гнучкість, масштабованість та ефективність розробки бота.

РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ ФУНКЦІОНАЛЬНОГО МОДУЛЯ TELEGRAM-БОТА

4.1 Ініціалізація проекту та підключення до Telegram A

Як вже зазначалось вище, я використовував IDE – PyCharm 2023. 1.2

Для початку роботи я створив проект (Рисунок 4.1.) в PyCharm 2023.

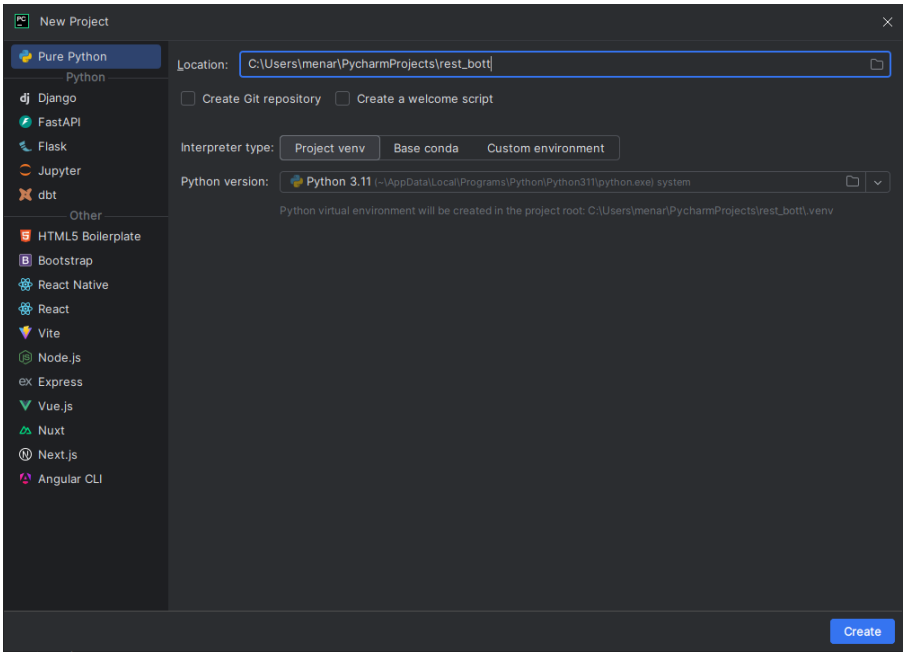


Рисунок 4.1 - Створення проекту.

Далі встанови бібліотеку telebot через термінал IDE за допомогою команди:

```
(.venv) PS C:\Users\menar\PycharmProjects\rest_lis> pip install pyTelegramBotAPI
```

Рисунок 4.2 - Встановлення бібліотеки telebot.

Завдяки TG-боту @BotFather було створено бота та встановлено його назву “Ресторан Лісова Пісня” та добавлено нікнейм-посилання, після чого отримано API-ключ:

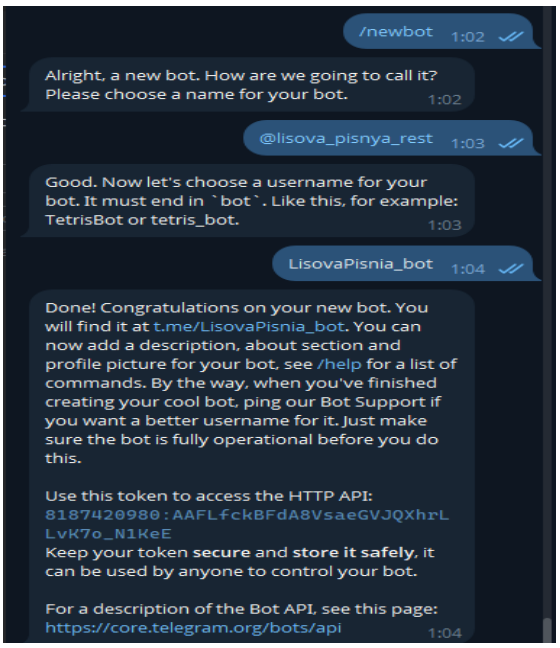


Рисунок 4.3 - Добування API-ключа.

Дані рядки коду встановлюють залежності, а також налаштовують з'єднання з TG-API та надають доступ до функціональності Telebot для подальшої обробки повідомлень та взаємодії користувачів з ботом. (Рис. 4.4.)

```
import telebot
from telebot import types
import re
from datetime import datetime, timedelta
import telebot.apihelper

# Ініціалізація бота з токеном
bot = telebot.TeleBot('8187420980:AAFLfckBFdA8VsaeGVJQXhrLLvK7o_N1KeE')
```

Рисунок 4.4 -Імпортував потрібні модулі та опридилив ключ

1. import telebot – імпортує головний модуль бібліотеки Telebot, який є модулем telebot, що дає змогу використовувати можливості цієї бібліотеки у коді.

2. from telebot import types – модуль типів імпортується з бібліотеки Telebot. Класи, які потрібні для опису типів повідомлень та інших об’єктів, які використовуються для спілкування з користувачами та ботами, знаходяться в модулі types. Для прикладу візьмемо клас ReplyKeyboardMarkup у цьому модулі

використовується для створення клавіатур відповідей.

3. import re – імпортує вбудований модуль re, який використовується для роботи з регулярними виразами. Ми можемо використовувати постійні вирази для того, щоб виконувати складні операції пошуку та заміни тексту на основі шаблонів. Модуль re можна використовувати в цій ситуації для перегляду та обробки текстових повідомлень, надісланих користувачем.

4. bot=telebot.TeleBot('6056246270:AAG_Wyp9uNDSq0O0gz65x8b2SLP1Jzcn7xg')) – створює об'єкт бота класу TeleBot, який є зовнішнім джерелом взаємодії Telebot з API TG. API, що ми отримали від BotFather після реєстрації бота на платформі TG, надсилається до конструктора класу. Маркер розпізнає нашого бота та дозволяє йому надсилати й отримувати повідомлення через TG API.

Проект реалізовано без бази даних, тому я створив словник “menu”, який містить категорії , страви , ціни та фото страв з шляхами у директорії , які вказують звідки буде видобуватись фото:

```
menu = {
    'Страви': {
        'Вареники': {'price': 50, 'photo': 'photos/varenyk.jpg'},
        'Деруни': {'price': 70, 'photo': 'photos/deruny.jpg'},
        'Борщ': {'price': 60, 'photo': 'photos/borshch.jpg'},
        'Солянка': {'price': 65, 'photo': 'photos/solyanka.jpg'},
        'Холодник': {'price': 55, 'photo': 'photos/holodnik.jpg'},
        'Піца Маргарита': {'price': 120, 'photo': 'photos/pizza_margarita.jpg'},
        'Піца 4 Сири': {'price': 150, 'photo': 'photos/pizza_4cheese.jpg'},
        'Піца Пепероні': {'price': 140, 'photo': 'photos/pizza_pepperoni.jpg'},
        'Картопля фрі': {'price': 45, 'photo': 'photos/french_fries.jpeg'},
        'Картопля по-селянськи': {'price': 50, 'photo': 'photos/country_potatoes.jpg'}
    },
    'Салати': {
        'Цезар': {'price': 80, 'photo': 'photos/salat_cezar.png'},
        'Олів\'є': {'price': 90, 'photo': 'photos/salat_olive.jpg'},
        'Салат Айсберг': {'price': 75, 'photo': 'photos/salat_iceberg.jpg'},
        'Грецький': {'price': 85, 'photo': 'photos/greek_salad.jpg'},
        'Вінігрет': {'price': 70, 'photo': 'photos/vinigret.jpg'}
    },
    'Напої': {
        'Кока-Кола': {'price': 30, 'photo': 'photos/photo_koka_cola.jpg'},
        'Фанта': {'price': 30, 'photo': 'photos/photo_fanta.jpg'},
        'Соки': {'price': 40, 'photo': 'photos/photo_soki.jpg'},
        'Узвар': {'price': 35, 'photo': 'photos/uzvar.jpg'},
        'Квас': {'price': 30, 'photo': 'photos/kvas.jpg'}
    }
}
```

Рисунок 4.5 - Реалізація menu

Створив словники ‘cart’(корзина) і ‘user_data’ (дані користувачів), reservation(дані заброньованих столиків) (Рис. 4.6.):

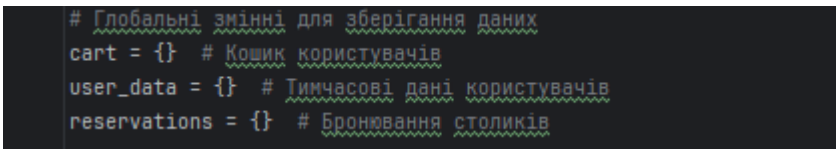


Рисунок 4.6 - Реалізація menu.

Кошик користувача, дані користувача та дані бронювань зберігаємо та отримуємо за допомогою цих змін, які мають глобальну видимість і можуть використовуватися в багатьох частинах програм. (Рисунок 4.6.)

4.2 Реалізація меню та обробка команд

Функція main_menu(message) створює та відправляє головне меню у вигляді клавіатури та приймає параметр message як параметр, який дає змогу визначити у який саме чат потрібно відправити меню. (Рис. 4.7.)

Функція ReplyKeyboardMarkup , що представляє спеціальну клавіатуру меню. Row-width=2 вказує на те що кнопка повинна бути розташованою по 2 в рядку для компактного розміщення.

Функція визначає чотири основні кнопки головного меню, кожна з них містить смайлики для, того щоб повідомлення від боту не виглядали сиро. Кнопки: “Меню”(для перегляду категорій страв), “Кошик”(для перегляду та оформлення замовлень), “Залишити відгук”(написання відгуку), “Забронювати столик”(бронювання столика).

Метод add() додає кнопки до клавіатури, використовуючи “*”(оператор розпакування), якиц передає всі елементи button як окремі аргументи.

Потім функція відправляє сформоване меню в чат користувача завдяки bot.send_message(message.chat.id) з ідентифікатором айді чату і з текстом “Оберіть опцію” та створена клавіатура markup як параметр reply_markup.

```python
def main_menu(message):  8 usages
    """Показує головне меню з основними опціями"""
    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    buttons = ['🍽 Меню', '🛒 Кошик', '📝 Залишити відгук', '🍴 Забронювати столик']
    markup.add(*buttons)
    bot.send_message(message.chat.id, text: 'Оберіть опцію:', reply_markup=markup)
```

Рисунок 4.7 - Реалізація меню з категоріями .

Обробник команди /start (Рис. 3.8.), який викликає функцію main_menu(message):

```python
@bot.message_handler(commands=['start'])
def start(message):
    """Обробник команди /start"""
    main_menu(message)
```

Рисунок 4.8 - Обробник команди /start .

Show_menu(message) є обробником події, що активується, коли користувач натискає "Меню" у головному меню бота. Вона відповідає за показ категорій меню ресторану у зручному форматі.

Коли користувач клікає на пункт "Меню", функція створює нову клавіатуру з чотирма кнопками. Перші три визначають основні категорії страв, а саме: "Страви" для гарячих страв та основних страв, "Салати" для холодних закусок та салатів, і "Напої" для різноманітних напоїв. Четверта кнопка "Назад".

Клавіатура налаштована таким чином, щоб кнопки розташовувались по дві в рядку (row_width=2), що робить інтерфейс більш зручним для використання. Параметр resize_keyboard=True надає адаптивне масштабування клавіатури під різні розміри пристроїв. (Рис. 4.9.)

Після того як пройшло створення клавіатури функція відправляє повідомлення у чат юзера з текстом "Оберіть категорію:", відповідно супроводжуючи його створеною клавіатурою як варіантами відповіді. Це дає змогу користувачу зручно переміщатися між категоріями меню та повертатися у

головне меню при потребі.

Дана функція грає важливу роль у навігації боту, надаючи зручний доступ до асортименту ресторану, структурно розміщеного по категоріях.

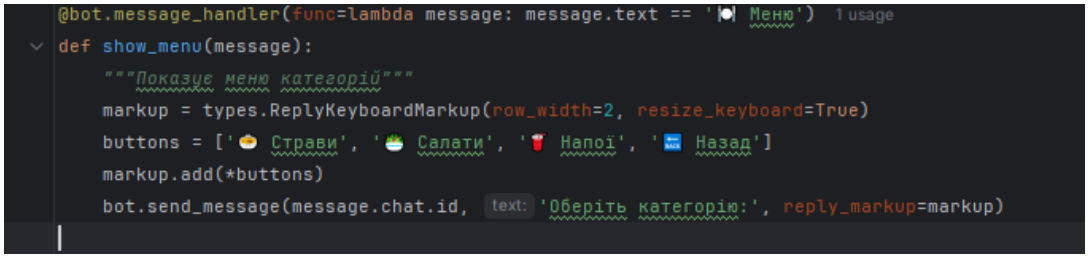


Рисунок 4.9 - Обробники показати_меню (повідомлення).

Обробник go_back(message) (Рис. 4.10.)відповідає за кнопку "Назад" і повертає назад або до головного меню викликавши main_menu(message):

Рисунок 4.10 - Обробники команди /back .

Наступні три функції слугують обробниками вибору конкретних позицій у меню і реагують на вибір певної категорії товару (Рис. 4.10):

1) dish_menu(message) - коли користувач хоче обрати будь-яку страву з розділу "Страви"

2) salad_menu(message) - активується при виборі будь-якого салату з категорії "Салати"

3) drink_menu(message) - обробляє вибір напою з розділу "Напої"

Дані функції працюють за одинаковим принципом, а саме перевіряють чи обраний товар належить до відповідної категорії завдяки перевірці ключів словника menu і якщо все добре, то передають управління до спільної функції handle_product_selection(message) – обробника вибору товару. По суті це функції-

фільтри, які направляють вибір юзера до єдиної функції обробки, що дає змогу уникнути дублювання та спрощує головні задачі роботи з кодом.

Візьмемо handle_product_selection(message) обробляє та фіксує назву обраного продукту з повідомлення користувача та зберігає цю інформацію в пам'яті бота, прив'язуючи до конкретного чату. Після цього бот запитує користувача ввести кількість товару та готує систему до наступного кроку - обробки введеної кількості через функцію process_quantity.

```python
# Обробники вибору страв/салатів/напоїв
@bot.message_handler(func=lambda message: message.text in menu['Страви'].keys())
def dish_menu(message):
    """Обробляє вибір страви"""
    handle_product_selection(message)


@bot.message_handler(func=lambda message: message.text in menu['Салати'].keys())
def salad_menu(message):
    """Обробляє вибір салату"""
    handle_product_selection(message)


@bot.message_handler(func=lambda message: message.text in menu['Напої'].keys())
def drink_menu(message):
    """Обробляє вибір напою"""
    handle_product_selection(message)


def handle_product_selection(message):  3 usages
    """Спільна функція для обробки вибору товару"""
    product = message.text
    user_id = message.chat.id
    user_data[user_id] = {'item': product}
    bot.send_message(user_id,  text: '📝 Введіть кількість:')
    bot.register_next_step_handler(message, process_quantity)
```

Рисунок 4.11 - Обробники генерації меню .

Цей механізм дозволяє уніфікувати обробку вибору для всіх категорій товарів, усуваючи необхідність окремих функцій для кожної позиції меню. Завдяки збереженню проміжних даних бот пам'ятає, що вибрав користувач, поки юзер вводить кількість. Фактично виступає перехрестям куди потрапляють всі вибрані користувачем дані.

4.3 Опис функціонального модуля Telegram-бота для замовлення, бронювання та зворотного зв'язку

Функція process_quantity обробляє кількість товару, введену користувачем під час розміщення замовлення та є ключовою ланкою в у даному процесі. (Рис. 3.12)

Виконує такі операції:

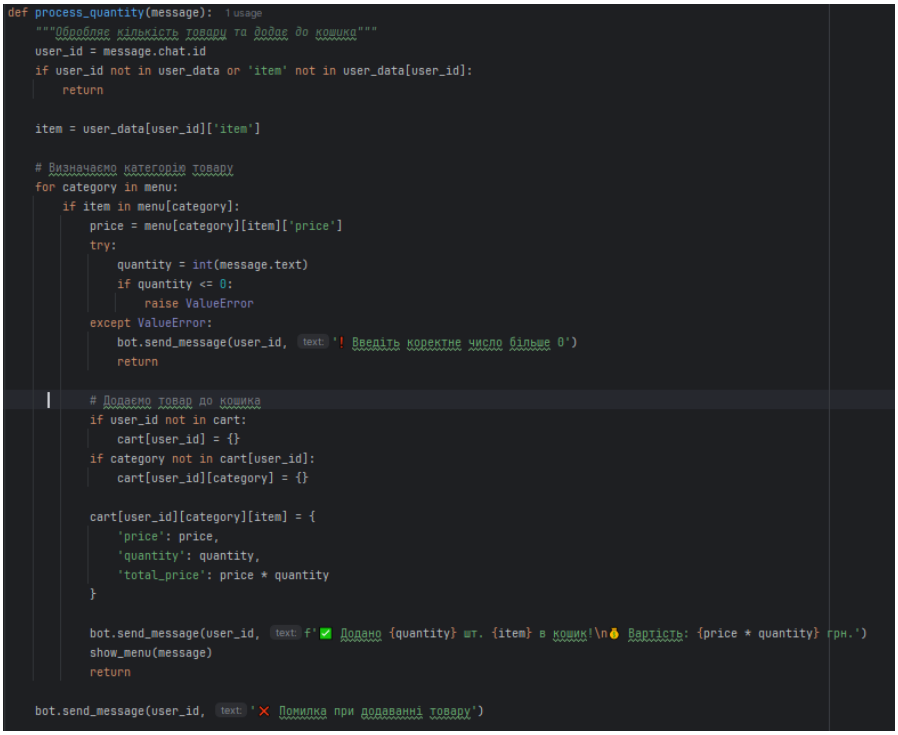
1. Перевірка та ініціалізація даних – отримує user_id з чату для ідентифікації юзера, перевіряє наявність нещодавно збережених даних про вибраний товар, якщо дані відсутні, то припиняє дію.
2. Валідація введених даних – аналізує текст повідомлення на коректну кількість, використовує try-except для перехоплення помилок (перевіряє чи число більше 0) вразі помилки інформує і перериває процес.
3. Робота з кошиком – знаходить категорію товару, створює потрібні структури для зберігання даних (кошик та підкатегорії товарів), заповнює інформацію про товар (ціну, кількість, рахує загальну вартість)
4. Зворотний зв’язок – надає юзеру підтвердження(назву товару, кількість та загальну вартість), в разі успіху повертає до меню з кнопкою оформлення замовлення, при помилці інформує.

Додаткові особливості:

1. Використовує словникову структуру для зберігання даних.
2. Автоматично розраховує вартість (price * quantity).
3. Захист від некоректних даних на кожному етапі.
4. Інтегрована з іншими компанентами, а саме такими як: user_data, cart, menu.

Якщо узагальнено, то дана функція – зручний механізм додавання товарів у кошик, що є основою для подальшого оформлення замовлення та виступає не від’ємною частиною нашого проекту. Обробляє як коректні, так і помилкові сценарії введення, відповідно забезпечуючи стабільну роботу бота у всіх

випадках.



```
def process_quantity(message):  1 usage
    """Обробляє кількість товару та додає до кошика"""
    user_id = message.chat.id
    if user_id not in user_data or 'item' not in user_data[user_id]:
        return

    item = user_data[user_id]['item']

    # Визначаємо категорію товару
    for category in menu:
        if item in menu[category]:
            price = menu[category][item]['price']
            try:
                quantity = int(message.text)
                if quantity <= 0:
                    raise ValueError
            except ValueError:
                bot.send_message(user_id, text: '! Введіть коректне число більше 0')
                return

            # Додаємо товар до кошика
            if user_id not in cart:
                cart[user_id] = {}
            if category not in cart[user_id]:
                cart[user_id][category] = {}

            cart[user_id][category][item] = {
                'price': price,
                'quantity': quantity,
                'total_price': price * quantity
            }

            bot.send_message(user_id, text: f'✅ Додано {quantity} шт. {item} в кошик!\n🔥 Вартість: {price * quantity} грн.')
            show_menu(message)
            return

    bot.send_message(user_id, text: '❌ Помилка при додаванні товару')
```

Рисунок 4.12 - Функція вибору кількості .

Функція view_cart відповідає за перегляд кошику. Коли юзер натискає кнопку з однойменною назвою кошика, то починається перевірка вмісту кошика. Вона отримує унікальний ідентифікатор чату юзера, щоб знайти його персональний кошик , якщо він порожній або відсутній, то завершує свою діяльність.

Для заповнення кошику, створюється детальний звіт у HTML-форматі. Функція обережно організовує товари по категоріях, а саме страви салати та напої виділяються окремими розділами, та кожна позиція супроводжується повною інформацією(назів, кількість одиниць ціна за одиницю та загальна вартість кожної позиції.(Рис. 4.13.)

Функція clear_cart відповідає за очищення кошика та активізується, коли юзер обирає кнопку “Очистити кошик”. Функція повністю обнуляє всі дані кошика для конкретного користувача, присвоюючи йому порожній словник.

Після того як корзину було очищено бот відправляє підтвердження виконаної дії у вигляді повідомлення зі смайликами, що підвищує зручність

взаємодії. На кінцевому етапі функція автоматично перенаправляє користувача назад до головного меню, де він може почати формування нового замовлення або скористатися іншими опціями бота.

```python
@bot.message_handler(func=lambda message: message.text == '🛒 Кошик')
def view_cart(message):
    """Показує вміст кошика"""
    user_id = message.chat.id
    if not cart.get(user_id):
        bot.send_message(user_id, text: '🛒 Ваш кошик порожній.')
        return

    text = '<b>🛒 Ваше замовлення:</b>\n'
    total = 0

    for category, items in cart[user_id].items():
        text += f'\n<b>📂 {category}</b>\n'
        for item, info in items.items():
            text += f'  - {item}: {info["quantity"]} шт. × {info["price"]} грн = {info["total_price"]} грн\n'
            total += info['total_price']

    text += f'\n<b>💰 Загалом:</b> {total} грн'

    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    markup.add( *args: '✅ Оформити замовлення', '❌ Очистити кошик', '🔙 Назад')

    bot.send_message(user_id, text, parse_mode='HTML', reply_markup=markup)


@bot.message_handler(func=lambda message: message.text == '❌ Очистити кошик')
def clear_cart(message):
    """Очищує кошик користувача"""
    user_id = message.chat.id
    cart[user_id] = {}
    bot.send_message(user_id, text: '🧹 Кошик очищено!')
    main_menu(message)
```

Рисунок 4.13 - Додавання інформації у кошик

Даний код, а саме функція place_order стартує процес оформлення замовлення, після того як клієнт натискає кнопку "Оформити замовлення". Працює за чітким алгоритмом, щоб забезпечити коректне проведення клієнта через всі необхідні етапи.

На початку система перевіряє стан кошика. Потім бот ідентифікує користувача за допомогою user_id та перевіряє наявність товарів у його кошику. Якщо система виявла кошик порожній, вона зразу повідомляє про це користувача за допомогою повідомлення з відповідним смайликом, яке відповідно підкреслює стан порожнього кошика.

Якщо перевірка пройшла вдало і товари наявні, функція каталізує процес

оформлення замовлення питаючи дані клієнта, а саме: ім’я, адресу, номер телефону.

Щоб реалізувати процес послідовного оформлення було використано механізм register_next_step_handler, що дозволяє боту запам’ятовувати етап, на якому знаходиться клієнт та переводити його до наступного кроку, а саме до виконання функції process_name, що буде обробляти ім’я.

Дана функція виконує роль контрольного пункту, що своєрідно превіряє готовність даних замовлення до оформлення та відповідно запускає наступні алгоритмічні процеси. Функція переходу від вибору товару до їх оформлення.

```
@bot.message_handler(func=lambda message: message.text == '✅ Оформити замовлення')
def place_order(message):
    """Починає процес оформлення замовлення"""
    user_id = message.chat.id
    if not cart.get(user_id):
        bot.send_message(user_id, text: '🛒 Ваш кошик порожній.')
        return

    bot.send_message(user_id, text: 'Будь ласка, введіть своє ім\'я:')
    bot.register_next_step_handler(message, process_name)
```

Рисунок 4.14 - Обробник повідомлення “Оформлення замовлення” .

Наступна функція process_name() викликається, коли користувач вводить ім’я протягом оформлення замовлення(Рис. 3.16.). Вона виконує наступне:

1. Записує ім'я в спеціальну user_data.
2. Потім переходить до наступного кроку - запиту адреси.
3. Використовує register_next_step_handler для плавного переходу.

Основне:

- Ідентифікація користувача за chat.id
- Збереження імені у форматі {'name': введений_текст}
- Відправка запиту на адресу доставки
- Налаштування наступного кроку - process_address

Наступна функція process_adress() викликається, коли користувач вводить адресу протягом оформлення замовлення (Рис. 3.15.). Вона Отже, ця функція

відповідає за отримання та збереження адреси користувача, а також перехід до наступного кроку - отримання номеру телефону, а саме до process_phone (обробник номеру телефону).

Функція process_phone найбільш складна з трьох, оскільки у ній реалізована перевірка на коректність введеного номеру, а саме кількість цифр повинно бути або 10, або 12 відповідно з форматом “38”, при неправильному введенні повідомляє та запитує знову номер, якщо введено правильно, то завершує оформлення та відправляє користувачу його чек і подяку за замовлення і обіцянку швидко з ним звязатися щодо замовлення. Також ордер відправляється в чат робочого персоналу, що відповідає за виконання замовлень.

```python
def process_name(message):  1 usage
    """Обробляє ім'я для замовлення"""
    user_id = message.chat.id
    user_data[user_id] = {'name': message.text}
    bot.send_message(user_id,  text: 'Введіть адресу доставки:')
    bot.register_next_step_handler(message, process_address)


def process_address(message):  1 usage
    """Обробляє адресу доставки"""
    user_id = message.chat.id
    user_data[user_id]['address'] = message.text
    bot.send_message(user_id,  text: 'Введіть номер телефону:')
    bot.register_next_step_handler(message, process_phone)


def process_phone(message):  2 usages
    """Обробляє номер телефону та завершує замовлення"""
    user_id = message.chat.id
    phone = message.text

    if not re.match( pattern: r'^\d{10,12}$', phone):
        bot.send_message(user_id,  text: '! Невірний формат телефону. Введіть 10-12 цифр')
        bot.register_next_step_handler(message, process_phone)
        return

    user_data[user_id]['phone'] = phone

    # Повідомлення для клієнта
    order_text = "<b>✅ Ваше замовлення прийнято!</b>\n\n"
    total = 0
```

Рисунок 4.15 - Обробник введених користувачем даних(ім’я та адреса)

.

Далі описуємо функції що відповідають за бронювання столику у закладі, а саме такі функції, як: book_table, process_guests, process_date, process_time,

process_booking_name та process_booking_phone.

Клієнт натискає кнопку "Забронювати столик", бот запускає певний алгоритм дій для збору потрібної інформації, а саме запитує кількість гостей, очікуючи число від 1 до 20, якщо клієнт вводить неправині дані, система виводить повідомлення " Введіть число від 1 до 20" і пропонує спробувати знову.

Коли клієнт успішно ввів кількість осіб, то далі бот генерує список вільних дат у форматі на наступні 7 днів, починаючи з поточної дати. Юзер бачить клавіатуру з варіантами, де дати розраховані за допомогою функції datetime.now() + timedelta(days=i).

Наступний етап – це вибір часу, де система пропонує певні часові інтервали, а саме: 10:00, 12:00, 14:00, 16:00, 18:00, 20:00, 22:00. У випадку якщо клієнт намагається ввести власний час, бот відповідає, що можна обрати тільки час з наявного списку текстом: " Оберіть час зі списку" та знову показує варіанти.

Потім бот запитує ім'я та номер телефону для бронювання, де працюють аналогічні умови обробника для номеру телефону, а саме перевірка на 10-12 цифр за допомогою регулярного виразу r'^\d{10,12}$', де у випадку помилки бот надає повідомлення з підказкою про правильний формат.

Коли успішно завершено усі минулі етапи, клієнт отримує детальне підтвердження з HTML-розміткою, де вказано ім'я (у форматі " Ім'я: {ім'я}") та замаскований номер телефону (перші 3 цифри + ***), кількість гостей, обрану дату і час. Наприклад: "Час: 18:00". Система додає всі ці дані до словника user_data під ідентифікатором чату користувача та відправляє у робочий чат персоналу за допомогою id чату, тільки у даному повідомленні номер не буде прихованим, що є абсолютно логічним.

Також на кожному етапі клієнт має змогу повернутися назад або скасувати бронювання. Усі повідомлення містять смайлики для кращої візуалізації та сприйняття даних. Коли бронювання пройшло успішно, то клієнт отримує фінальне повідомлення фразою "Дякуємо за бронювання! Чекаємо на вас!", що створює позитивне враження від взаємодії з ботом. (Рис. 3.16.)

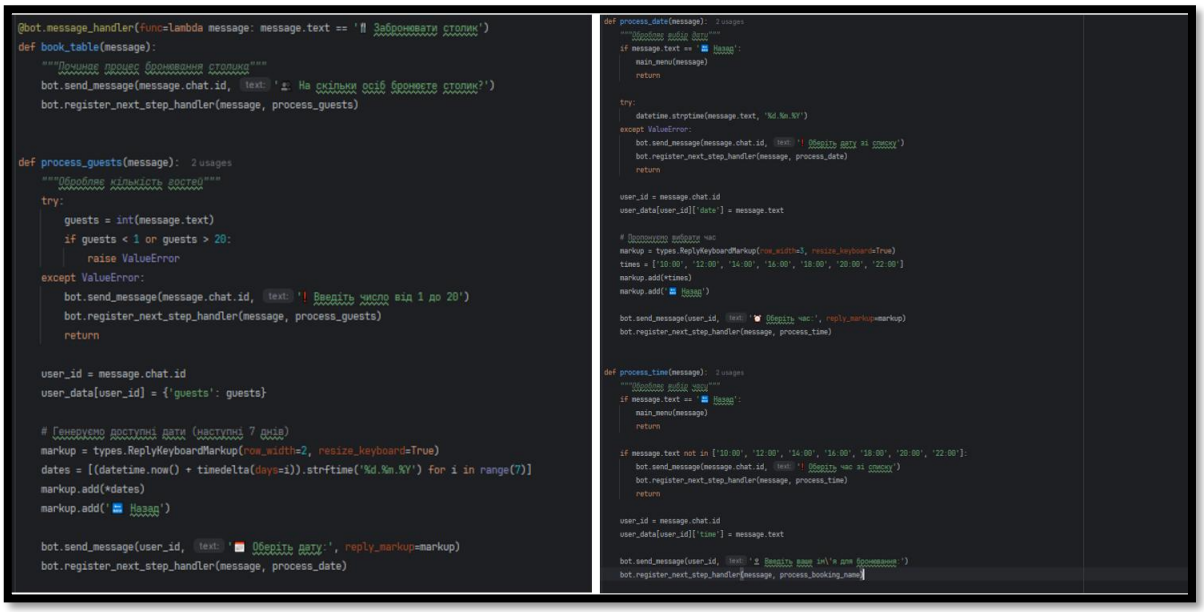


Рисунок 4.16 - Обробники бронювання столику.

Відправка оформленого нового замовлення в робочий чат персоналу для обробки замовлень за допомогою ID чату (Рис. 3.17.):

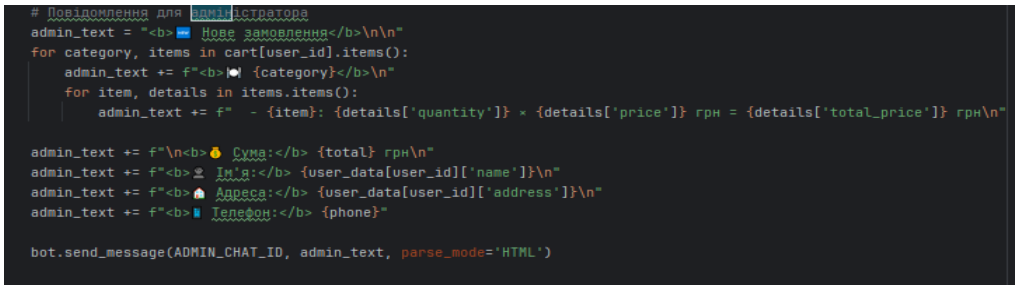


Рисунок 4.17 - Відправка замовлення на обробку.

Відправка даних з новим бронювання в робочий чат аналогічно за допомогою ID чату (Рис. 4.18.):

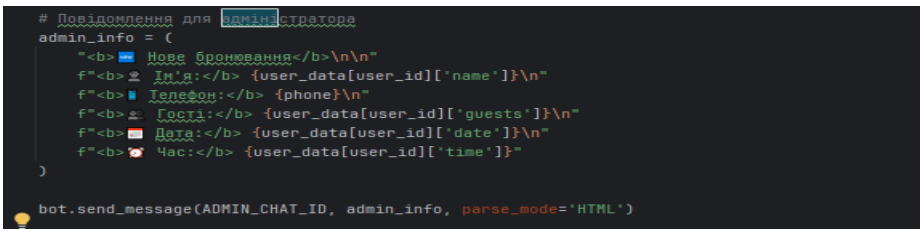


Рисунок 3.18 - Відправка бронювання на обробку.

Наступні дві функції відповідають за отримання відгуку від клієнта, який відповідно також відправляється у певний чат із відгуками закладу, що підв'язаний до каналу закладу. Відправляє такі данні " Відгук від @нікнейм_меседжеру: текст", щоб інші зацікавлені могли перевірити актуальність та правдивість даної інформації від клієнта (Рис. 3.19.).

```python
@bot.message_handler(func=lambda message: message.text == '✍ Залишити відгук')
def leave_feedback(message):
    """Запитує відгук від користувача"""
    bot.send_message(message.chat.id, text: '📝 Напишіть ваш відгук про наш сервіс:')
    bot.register_next_step_handler(message, process_feedback)


def process_feedback(message): 1 usage
    """Обробляє відгук та надсилає адміністратору"""
    feedback = message.text
    bot.send_message(message.chat.id, text: '❤ Дякуємо за ваш відгук!')
    bot.send_message(ADMIN_CHAT_ID,
                     text: f"📝 Відгук від @{message.from_user.username or message.from_user.first_name}:\n{feedback}")
    main_menu(message)
```

Рисунок 4.19 - Відправка відгуку.

Процес відбувається у два етапи: спочатку бот запитує текст відгуку, а потім обробляє його. Після отримання відгуку система надсилає клієнту підтвердження з сердечком, що привносить хороший тон та візуадл. Одночасно відгук передається в спеціальний чат у зручному форматі, де вказується автор (користувач TG) та його текст без змін.

Даний фрагмент коду (рис. 4.20.) формує остаточне підтвердження бронювання, що отримує клієнт після завершення усіх етапів резервування столиків, понаючи повідомлення із зеленої галочки та заголовку, що повідомляє про успішне прийняття бронювання на подальшу обробку персоналом.

Кожен пункт супроводжується смайликом, що покращує сприйняття та дає інформації більш структурований та організований вигляд. Також було реалізовано приховування номеру, що створює хоч якусь безпеку, а саме номер телефону відображається тільки трьома першими цифрами.

У повідомленні повинно збути зазначеним ім'я, під яким було виконано бронювання, що довзволяє клієнту перевірити валідність введених ним даних, а

також кількість гостей, дата та час візиту висвічуються у зручному візуальному форматі, де кожен параметр має відповідний маркер.

Повідомлення завершується фразою подяки та зазначенням того, що з нетепрінням чекають гостя в закладі. Дане оформлення створює імідж закладу, демонструючи увагу до деталей та комфорт відвідувачів (Рисунок 3.20.).

```
# Повідомлення для клієнта
reservation_info = (
    "<b>✅ Ваше бронювання прийнято!</b>\n\n"
    f"<b>👤 Ім'я:</b> {user_data[user_id]['name']}\n"
    f"<b>📱 Телефон:</b> {phone[:3]}***\n"
    f"<b>👥 Гості:</b> {user_data[user_id]['guests']}\n"
    f"<b>📅 Дата:</b> {user_data[user_id]['date']}\n"
    f"<b>⏰ Час:</b> {user_data[user_id]['time']}\n\n"
    "Дякуємо за бронювання! Чекаємо на вас!"
)
```

Рисунок 4.20 - Повідомлення для клієнта.

Також аналогічний підхід повідомлень і після пройдення алгоритму оформлення замовлення та залишення відгуку, але відрізняються за змістом. А також завжди починають з підтвердження дій, щоб користувач зразу було видно результат дій, потім інформація – для бронювання це дата, час та кількість гостей, а для замовлення перелік з страв із їх прайсами, для відгуку просто текст написаний клієнтом.

Важливо те що вони всі моють функцію відправлення на канал/чат/ адміністратору для подальшої обробки чи показу. А також у них наявна спільна риса форматування, а саме HTML-теги для жирного тексту та розділення на рядки, що надає повідомленням рису читабельності. Також смійлики для візуалізації, що зразу візуально їх виділяє.

4.4 Висновки по розділу

У даному підрозділі було детально розглянуто основні функціональні компоненти Telegram-бота, що реалізує автоматизовану систему замовлення,

бронювання та зворотного зв'язку у сфері громадського харчування. Було описано логіку роботи ключових функцій, таких як обробка кількості товарів, перегляд та очищення кошика, оформлення замовлень, бронювання столиків та отримання відгуків від клієнтів.

Шкірна функція чітко інтегрована в загальний алгоритм роботи робота, забезпечуючи послідовність дій користувача, перевірку правильності введених даних та зручний інтерфейс. Особливу увагу приділено валідації, візуальній структуризації повідомлень та захисту персональних даних. Застосування HTML-форматування, смайликів та підтверджувальних повідомлень створює позитивне враження та сприяє підвищенню задоволеності клієнтів.

Результатом є надійна, функціонально повноцінна система, яка дозволяє автоматизувати ключові процеси взаємодії клієнта із закладом, значно підвищуючи ефективність роботи та рівень сервісу.

ВИСНОВОК

У наш час ресторанний бізнес є доволі важкою справою, але все-ще актуальною, не дивлячись на велику конкуренцію на ринку даного ремесла. Багато однотонних закладів , які не мають своєї своєї фічі. Але це не означає, що усі настільки некреативні та примітивні, що використовують старі патерни ведення даного бізнесу. Звісно є достатня кількість, які дорожать своєю репутацією та своїм бізнес-творінням та намаються йти в ногу з часом, придумуючи інтеграцію різноманітних іновацій для покращення взаємодії, а саме зворотнього зв'язку з клієнтом та створення максимально комфортних та простих умов для досягнення лояльності постійних клієнтів, та отримання великої кількості нових, оптимізацією робочих процесів. Отож, моє програмне рішення впевнено можна віднести до низки правильних нововведень, що тільки підвищують імідж, престиж, ефективність роботи закладу, оптимізовуючи процеси замовлення, зворотнього зв'язку та бронювання столиків.

У функціонал розробленого бота я створив меню категорій, перехід між категоріями товарів, відображення їх фото, введення користувачем кількості товару, додавання вибраних товарів у кошик, оформлення замовлення шляхом введення інформації користувача, а саме ім'я, адреса доставки, номер телефону та відправлення замовлення в робочий чат персоналу для подальшої його обробки. Також було добавлено функціонал бронювання столику в боті, таким чином, що користувач повинен натиснути на відповідну кнопну та ввести такі дані, як: кількість гостей, ім'я, обрати потрібну дату та час. Та ще один важливий момент – зворотній зв'язок, що являється базовою функцією, а саме після оформлення замовлення користувач має змогу залишити відгук, натиснувши на відповідну кнопку, та цей відгук буде відправлено на TG-канал закладу, із вмістимим тегом клієнта, що забезпечить прозорість та правдивість інформації, оскільки її можна буде перевірити.

Протягом виконання даної роботи я закріпив знання з програмування

мовою Python та опанував навички робити з бібліотекою Telebot, вивчивши відповідно ринок та проаналізував аналогічні рішення, що дало змогу сформувати правильний набір функцій та засобів їх реалізації.

Посилання на репозиторій з проектом:

https://github.com/san-anderson/rest_bot.git

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. https://pypi.org/project/pyTelegramBotAPI/
2. https://core.telegram.org/bots#botfather
3. https://core.telegram.org/bots/api
4. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/
5. https://github.com/eternnoir/pyTelegramBotAPI
6. https://telegrambots.github.io/book/
7. https://github.com/eternnoir/pyTelegramBotAPI/tree/master/examples
8. https://www.freecodecamp.org/ukrainian/news/yak-stvoryty-telehram-bota-za-dopomohoyu-python/
9. https://tproger.ru/articles/kak-napisat-telegram-bota-na-python-delaem-remajnder/
10. https://maxsite.org/page/python-telegram-bot
11. https://www.tutorialsteacher.com/python
12. https://www.youtube.com/watch?v=BjmW8QQZ8Bg&list=PLOlyZEVllXBHso6SasmoJdNWVO63573RN
13. https://habr.com/ru/articles/697052/
14. https://www.youtube.com/watch?v=AlOBD8C9_yw
15. https://www.youtube.com/watch?v=Gwvv7Z6kzRA&t=654s
16. https://www.w3schools.com/python/
17. https://www.tutorialspoint.com/python/index.htm
18. https://www.dataquest.io/blog/learn-python-the-right-way/
19. https://www.youtube.com/watch?v=kqtD5dpn9C8
20. https://www.kaggle.com/code/colinmorris/working-with-external-libraries
21. https://pymbook.readthedocs.io/en/latest/

22. https://www.techbeamers.com/python-tutorial-step-by-step/

23. https://inventwithpython.com/hacking/

24. Програмування мовою Python / О.М. Васильєв. – Тернопіль: Навчальна книга – Богдан, 2019. – 504 с.; іл.

25. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. 180 с.

26. Навчальний посібник “ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ PYTHON” - А.О. Костюченко 2020

27. Вивчаємо Python, том 1, 5-е вид.: Пер. с англ. — СПб.: ООО “Діалектика”, 2019. — 832 с. : іл. — Парад, тит. англ

28. Марк Лутц “Learining Python ” – книга

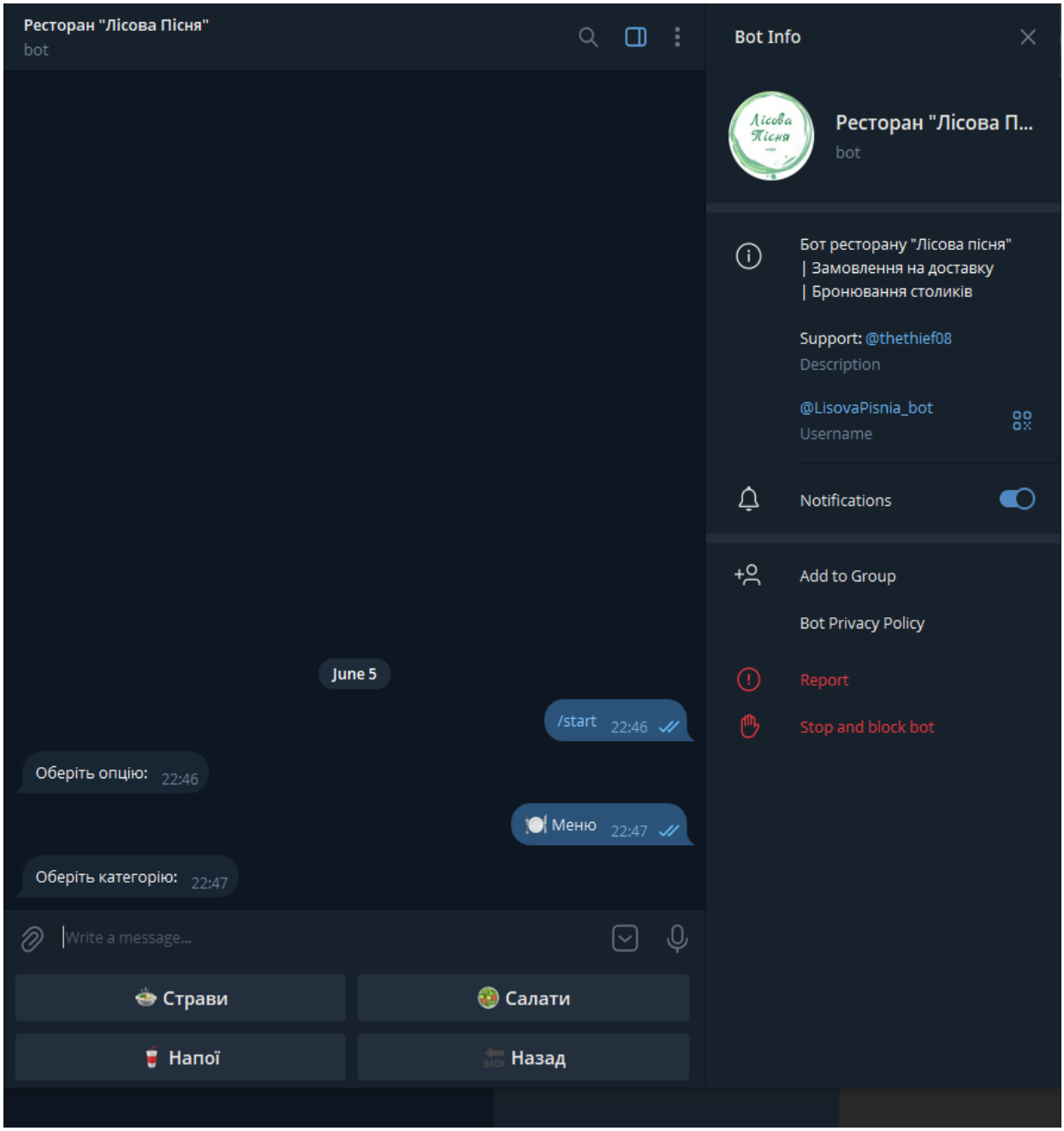
29. Основи програмування на мові Python. – М.: ДМК Пресс, 2017. – 284 с.: іл.

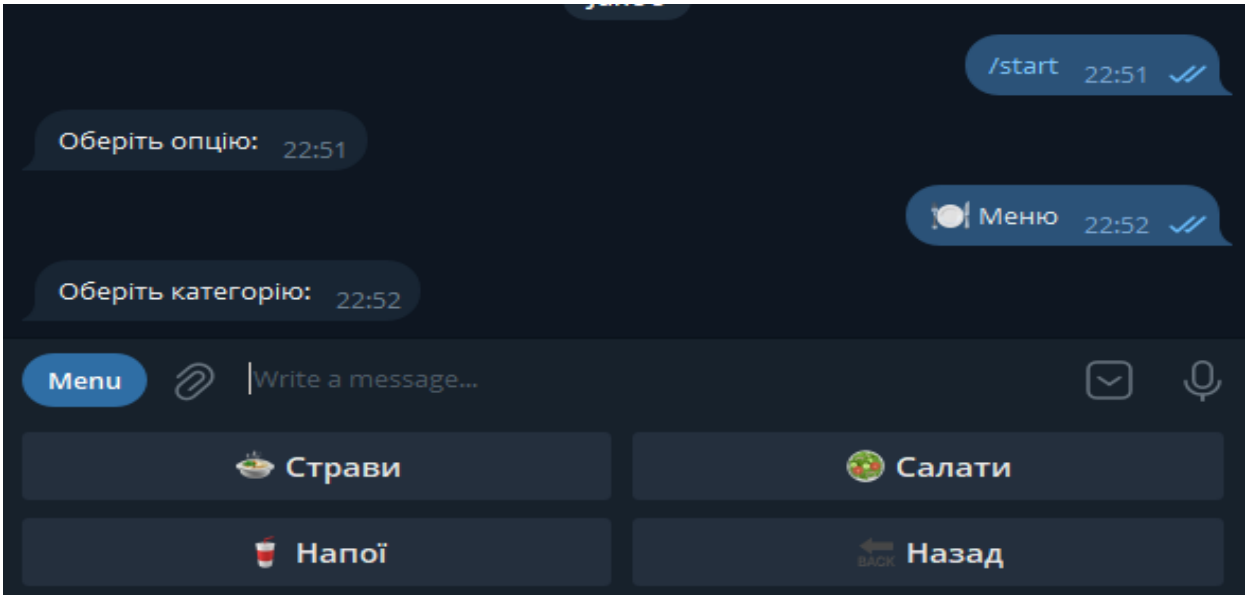
30. TG-бот : https://t.me/LisovaPisnia_bot

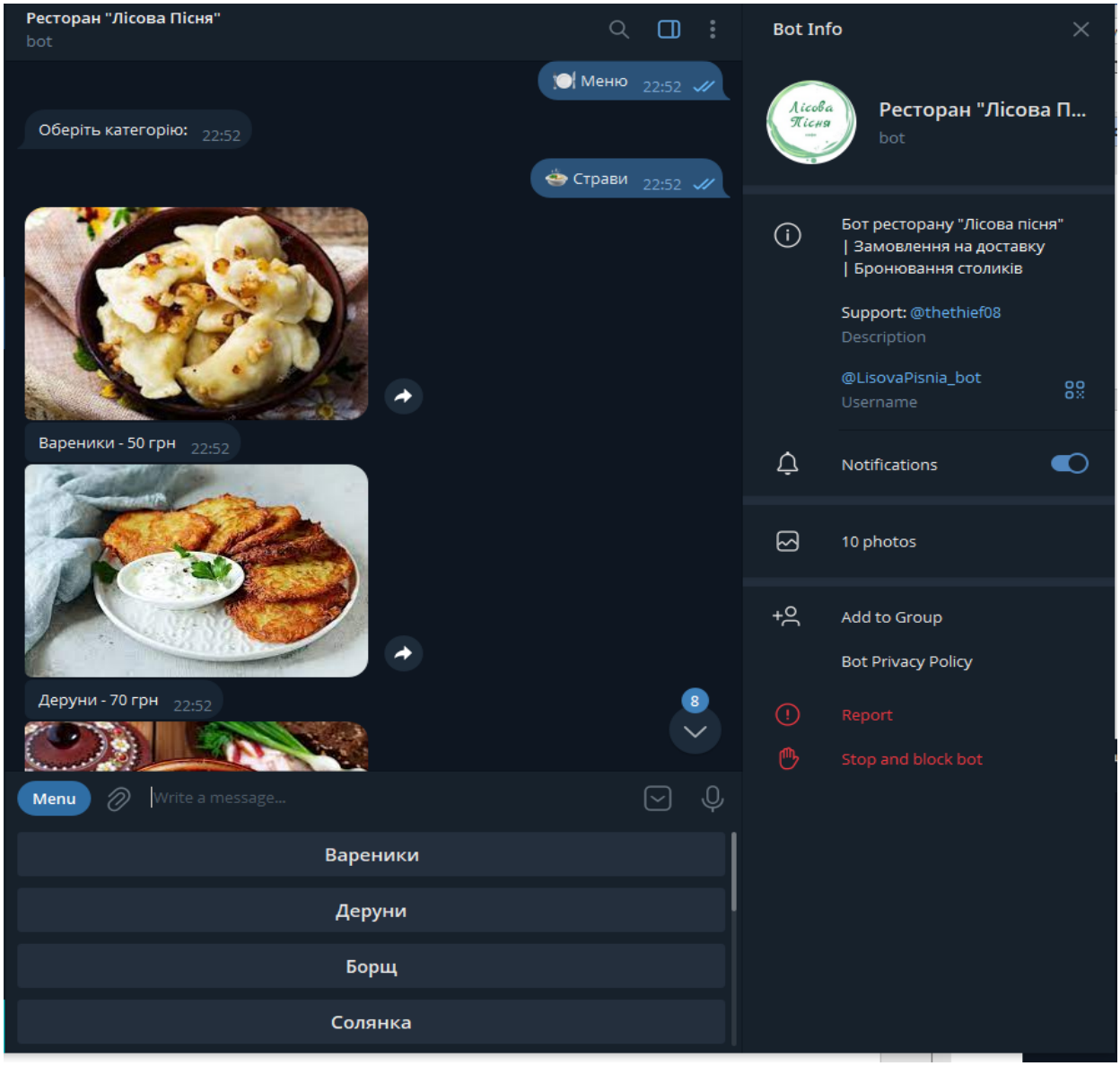
ДОДАТКИ

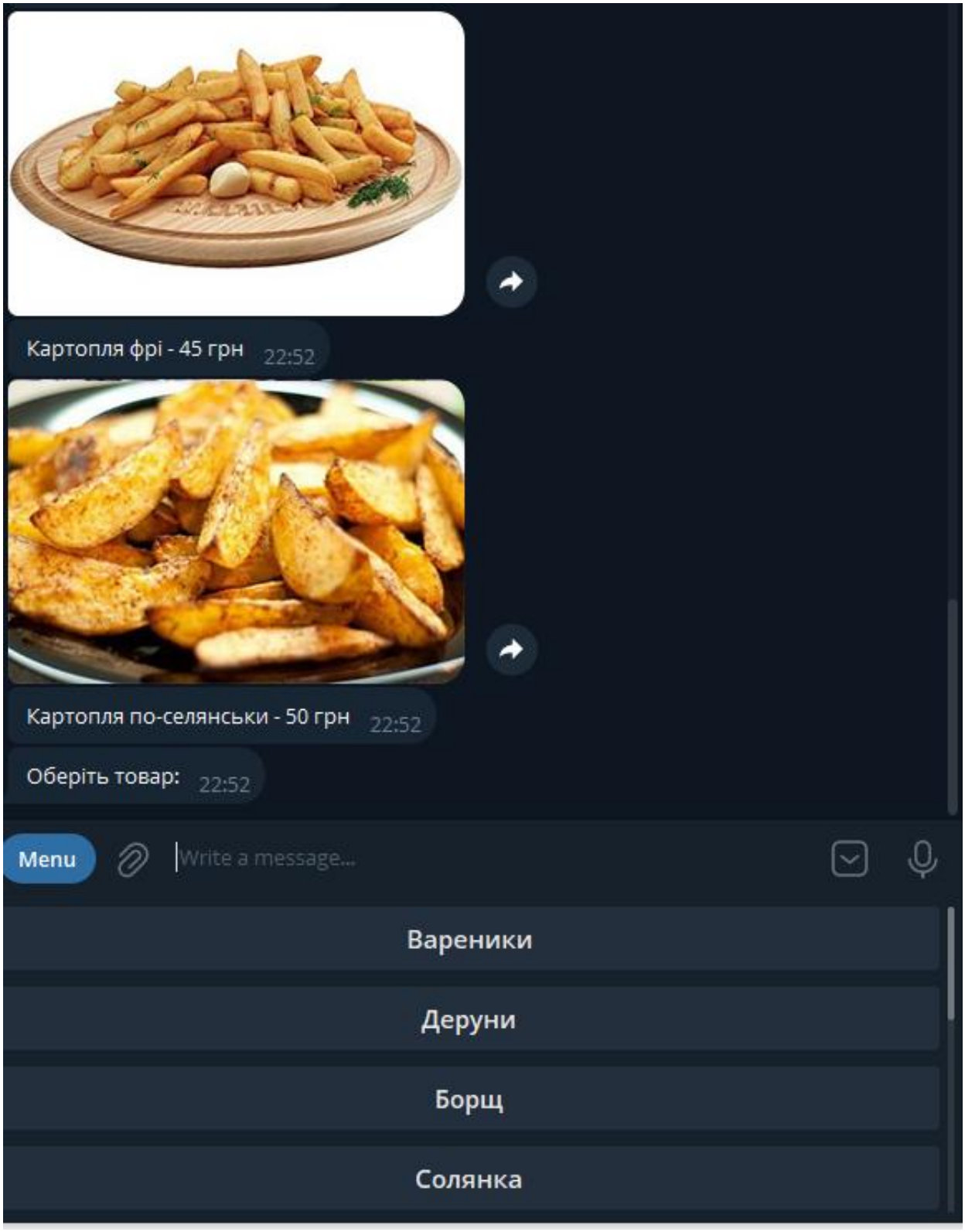
Додаток А

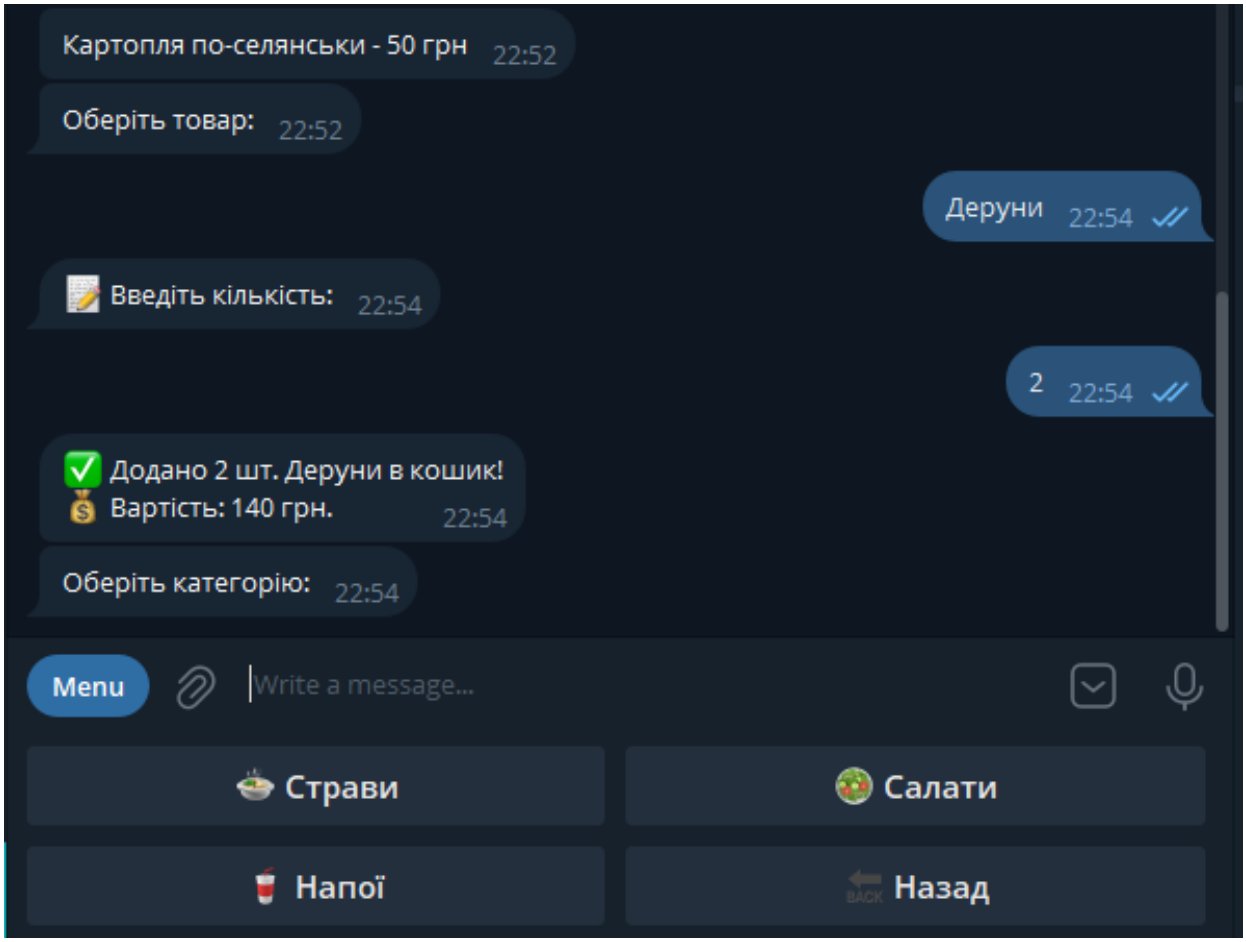
Результати роботи телеграм бота

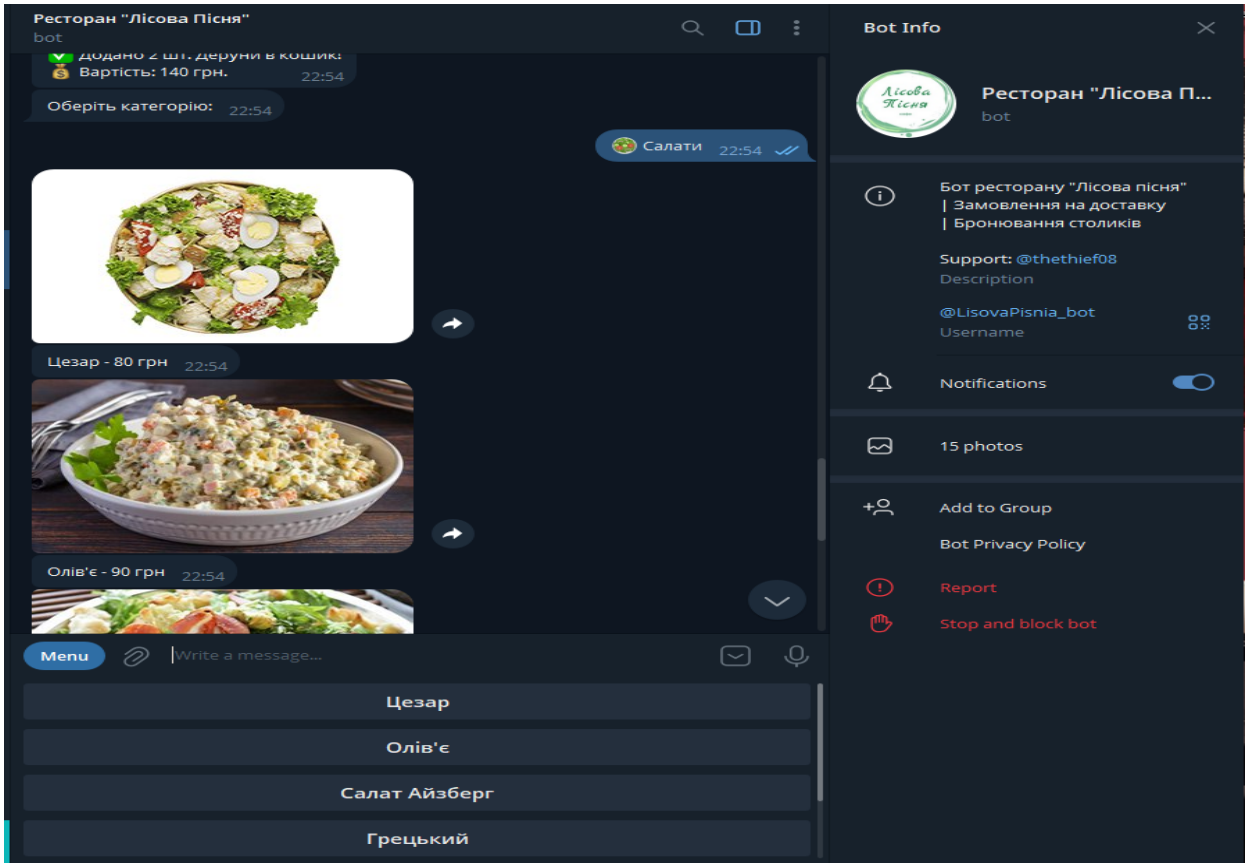


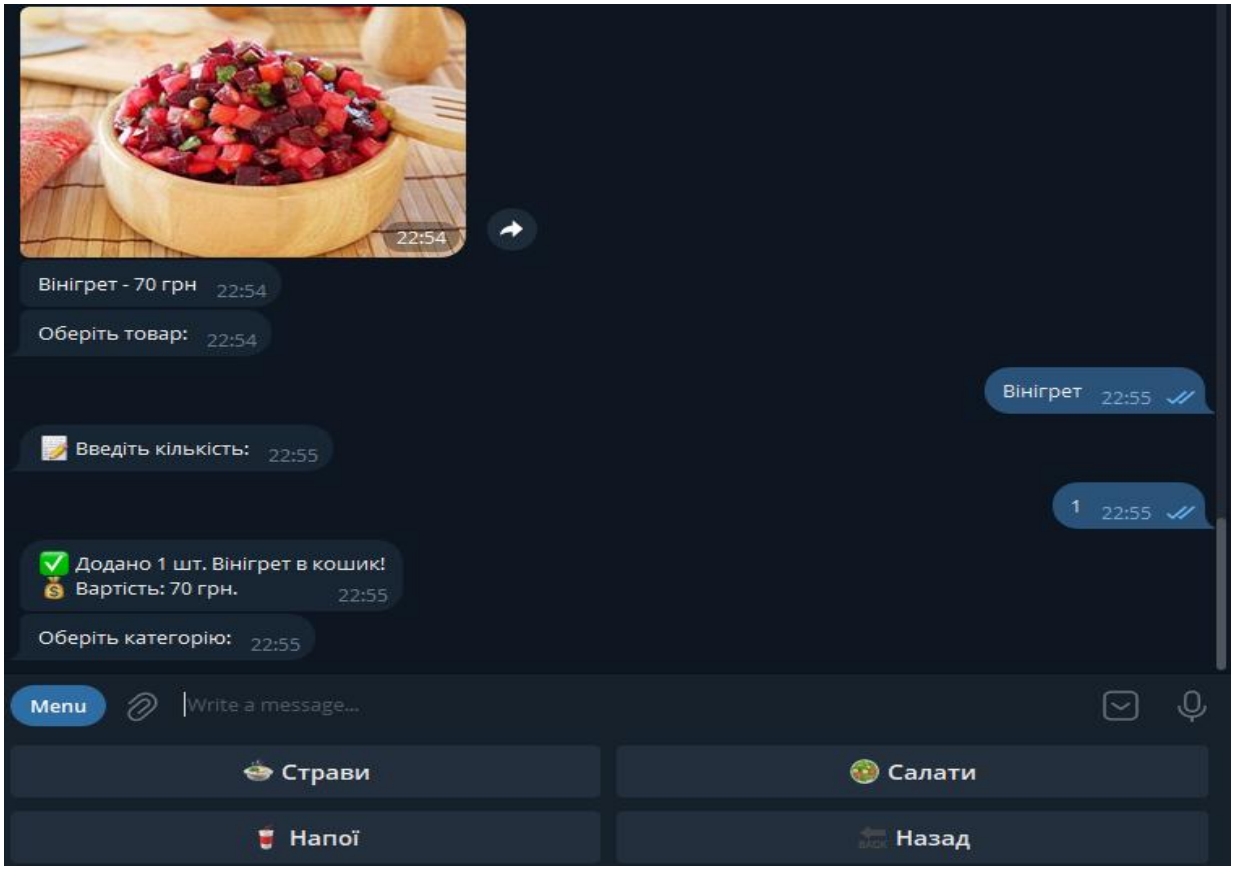

/start 22:51
Оберіть опцію: 22:51
Меню 22:52
Оберіть категорію: 22:52
Menu
Write a message...
Страви
Салати
Напої
Назад

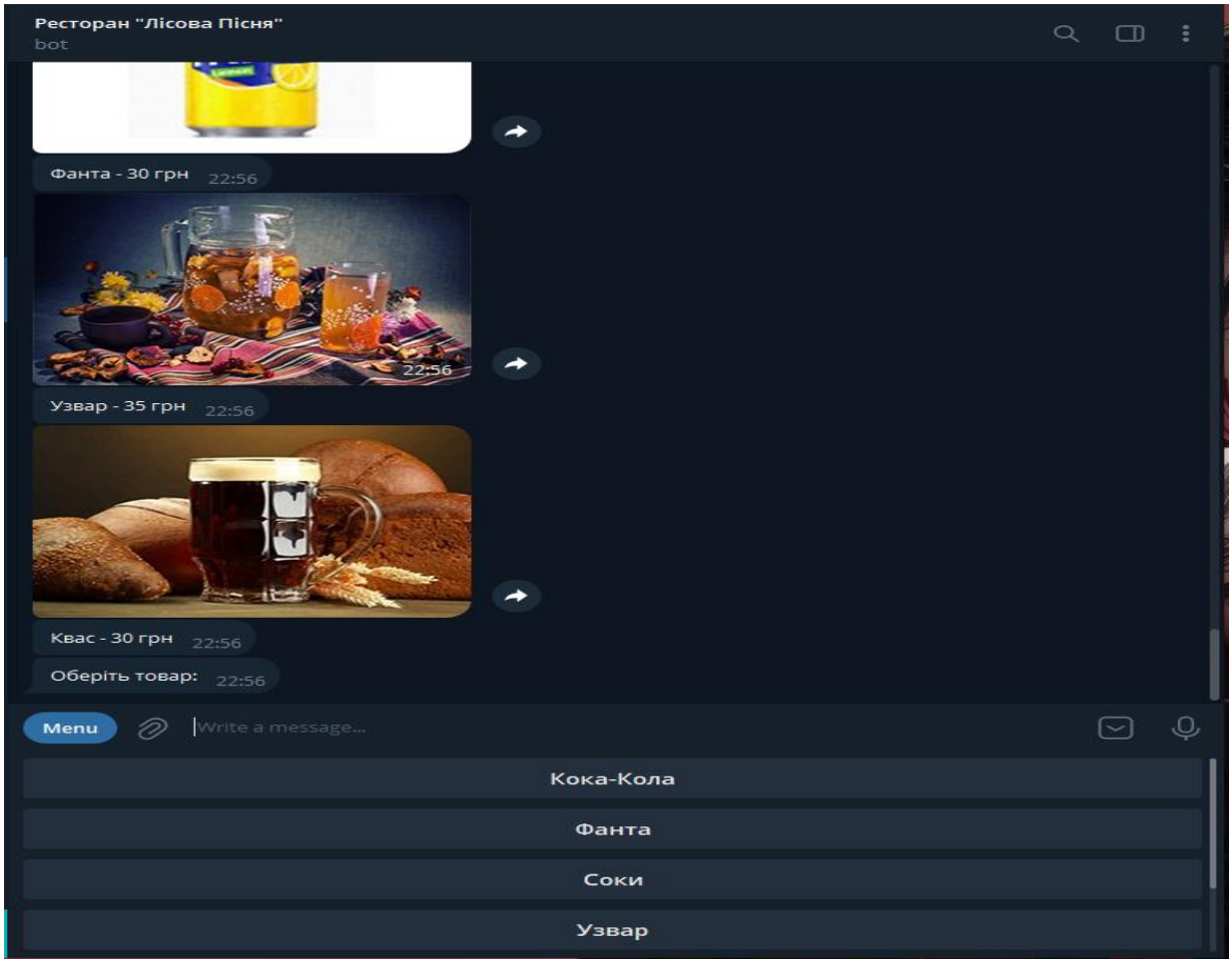

Ресторан "Лісова Пісня"
bot
Меню 22:52
Оберіть категорію: 22:52
Страви 22:52
Вареники - 50 грн 22:52
Деруни - 70 грн 22:52
Menu
Write a message...
Вареники
Деруни
Борщ
Солянка
Bot Info
Ресторан "Лісова П...
bot
Бот ресторану "Лісова пісня"
| Замовлення на доставку
| Бронювання столиків
Support: @thethief08
Description
@LisovaPisnia_bot
Username
Notifications
10 photos
Add to Group
Bot Privacy Policy
Report
Stop and block bot

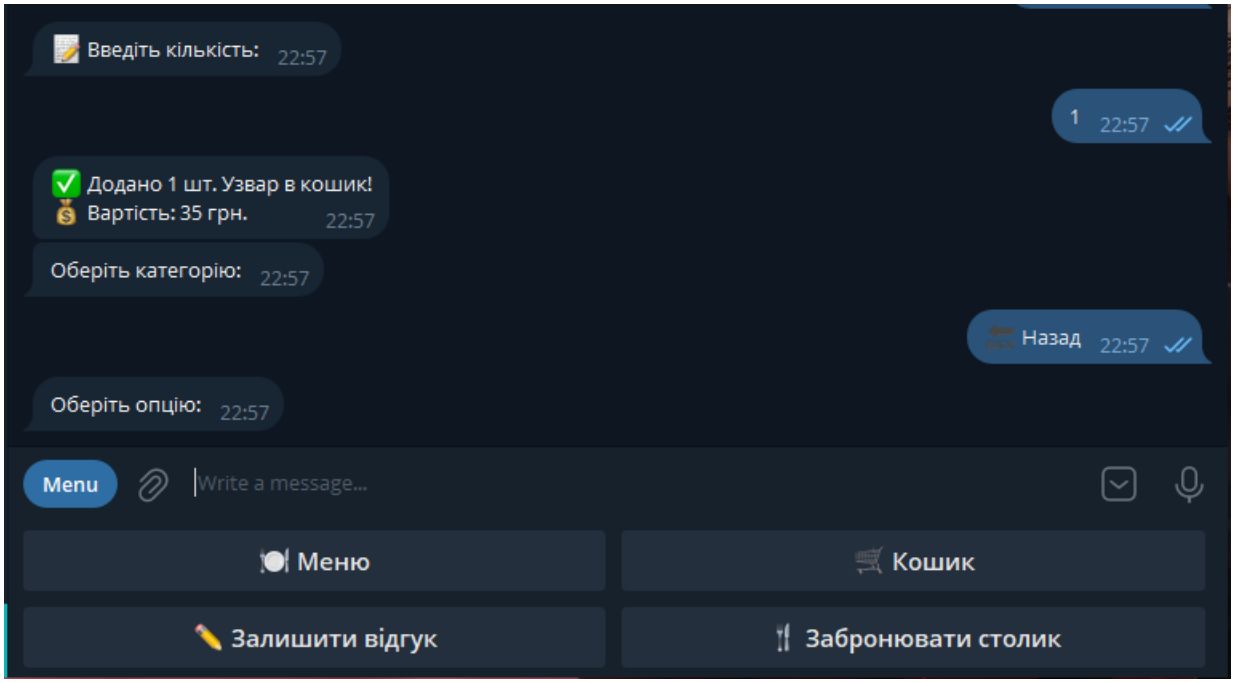

Картопля фрі - 45 грн 22:52
Картопля по-селянськи - 50 грн 22:52
Оберіть товар: 22:52
Menu
Write a message...
Вареники
Деруни
Борщ
Солянка

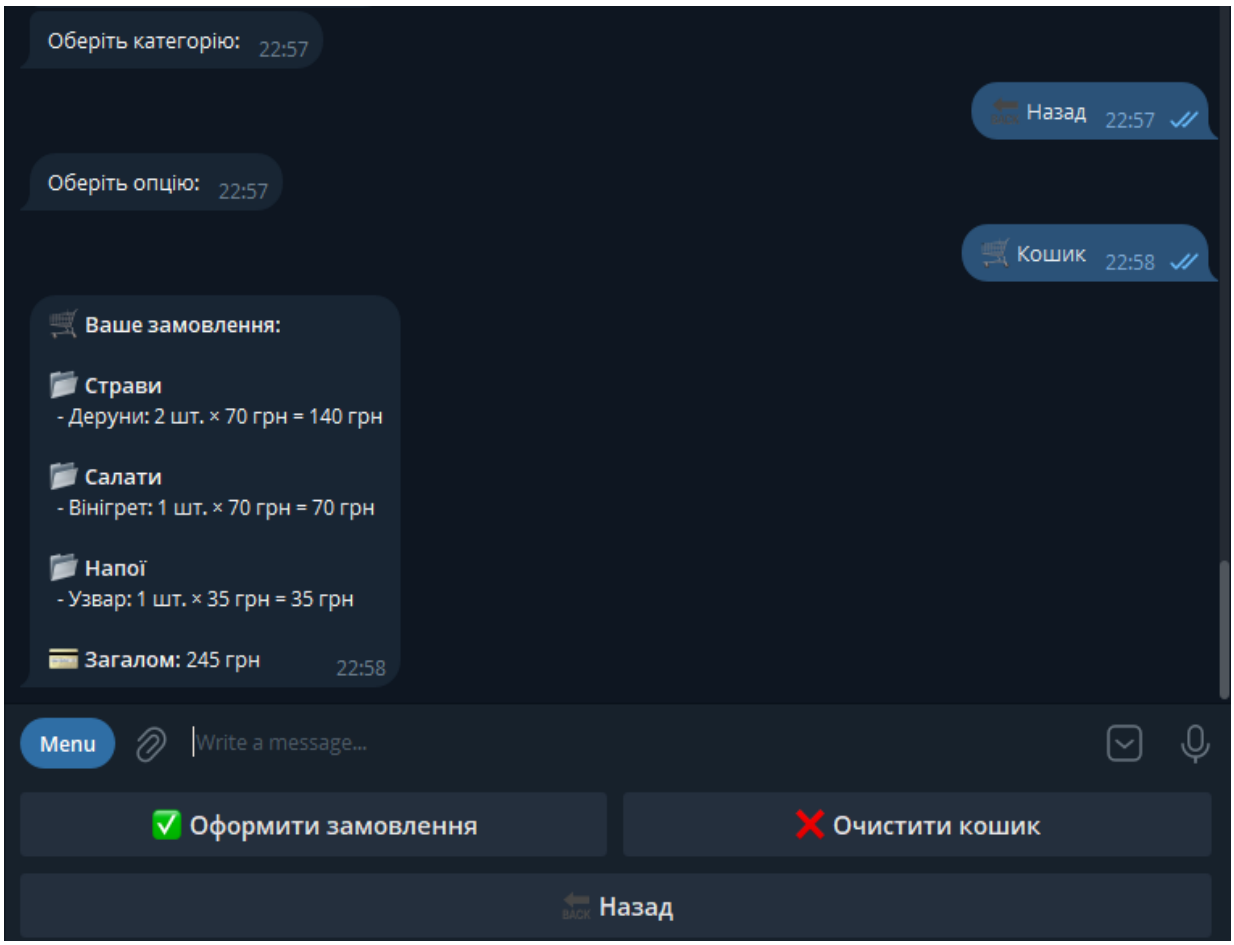

Картопля по-селянськи - 50 грн 22:52
Оберіть товар: 22:52
Деруни 22:54
Введіть кількість: 22:54
2 22:54
Додано 2 шт. Деруни в кошик!
Вартість: 140 грн. 22:54
Оберіть категорію: 22:54
Menu
Write a message...
Страви
Салати
Напої
Назад

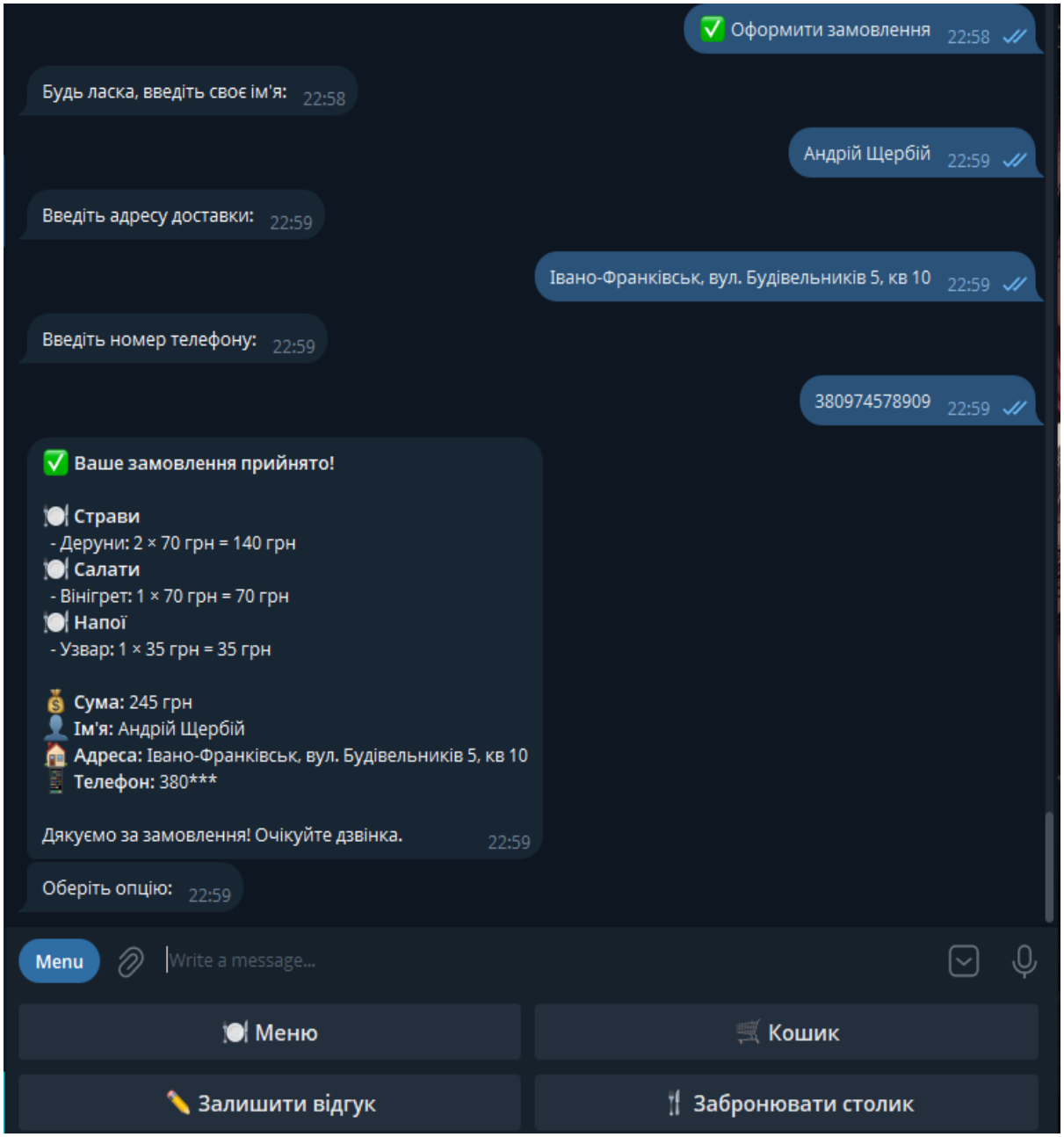

Ресторан "Лісова Пісня"
bot
Вартість: 140 грн. 22:54
Оберіть категорію: 22:54
Салати 22:54
Цезар - 80 грн 22:54
Олів'є - 90 грн 22:54
Menu
Write a message...
Цезар
Олів'є
Салат Айзберг
Грецький
Bot Info
Ресторан "Лісова П...
bot
Бот ресторану "Лісова пісня"
| Замовлення на доставку
| Бронювання столиків
Support: @thethief08
Description
@LisovaPisnia_bot
Username
Notifications
15 photos
Add to Group
Bot Privacy Policy
Report
Stop and block bot

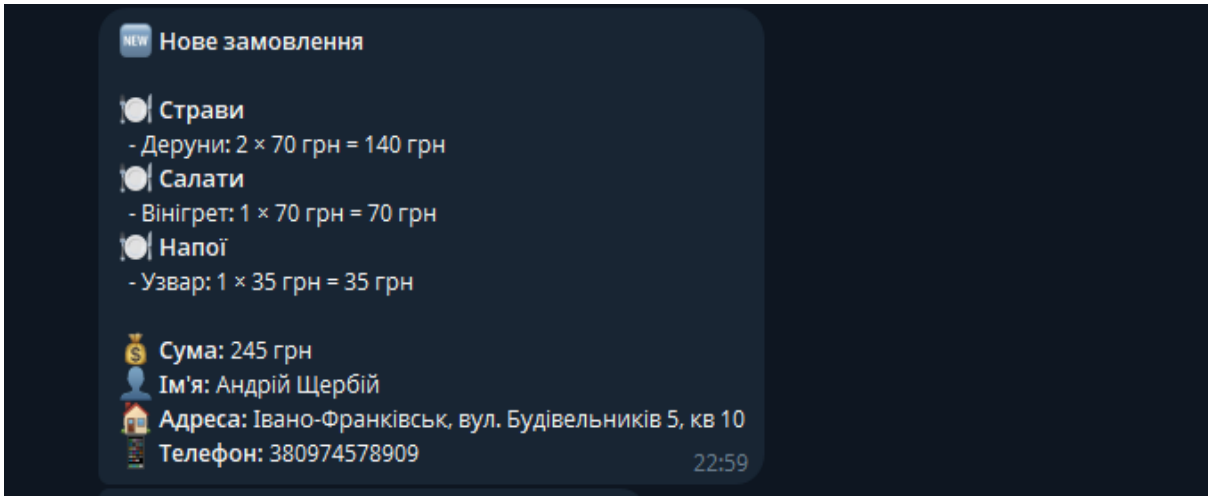

22:54
Вінігрет - 70 грн 22:54
Оберіть товар: 22:54
Вінігрет 22:55
Введіть кількість: 22:55
1 22:55
Додано 1 шт. Вінігрет в кошик!
Вартість: 70 грн. 22:55
Оберіть категорію: 22:55
Menu
Write a message...
Страви
Салати
Напої
Назад

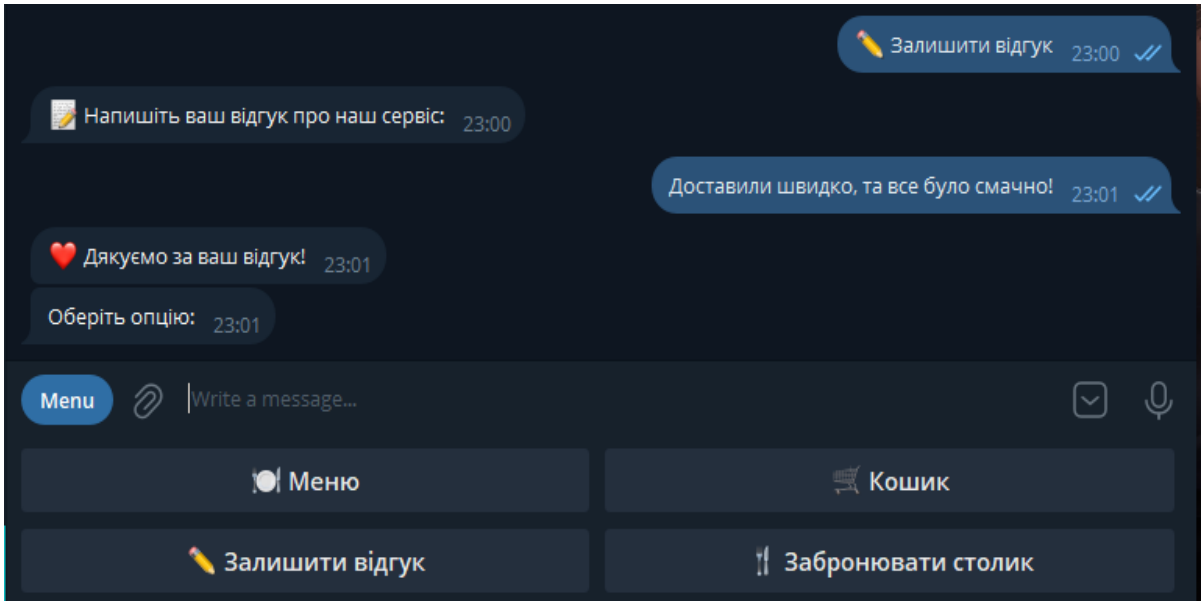

Ресторан "Лісова Пісня"
bot
Фанта - 30 грн 22:56
22:56
Узвар - 35 грн 22:56
Квас - 30 грн 22:56
Оберіть товар: 22:56
Menu
Write a message...
Кока-Кола
Фанта
Соки
Узвар

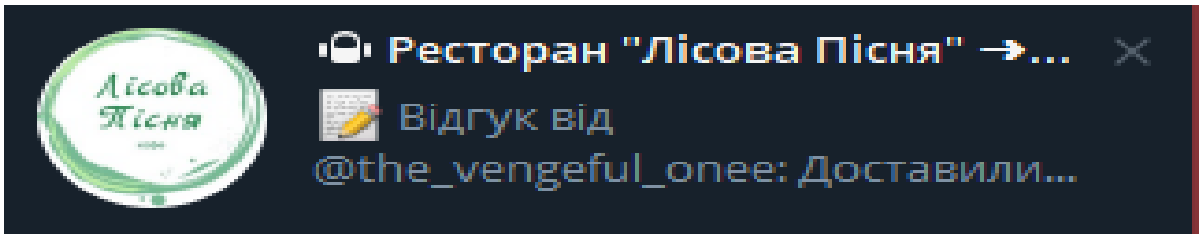

Введіть кількість: 22:57
1 22:57
Додано 1 шт. Узвар в кошик!
Вартість: 35 грн. 22:57
Оберіть категорію: 22:57
Назад 22:57
Оберіть опцію: 22:57
Menu
Write a message...
Меню
Кошик
Залишити відгук
Забронювати столик

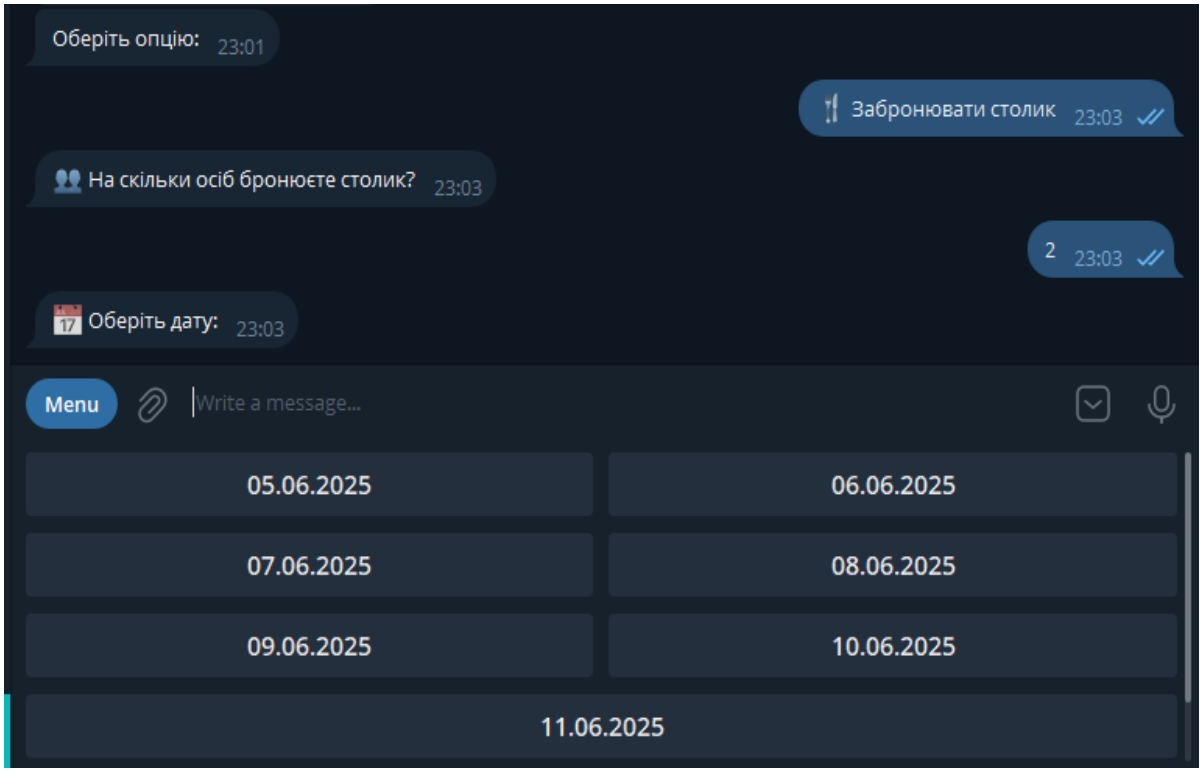

Оберіть категорію: 22:57
Назад 22:57
Оберіть опцію: 22:57
Кошик 22:58
Ваше замовлення:
Страви
- Деруни: 2 шт. × 70 грн = 140 грн
Салати
- Вінігрет: 1 шт. × 70 грн = 70 грн
Напої
- Узвар: 1 шт. × 35 грн = 35 грн
Загалом: 245 грн 22:58
Menu
Write a message...
Оформити замовлення
Очистити кошик
Назад

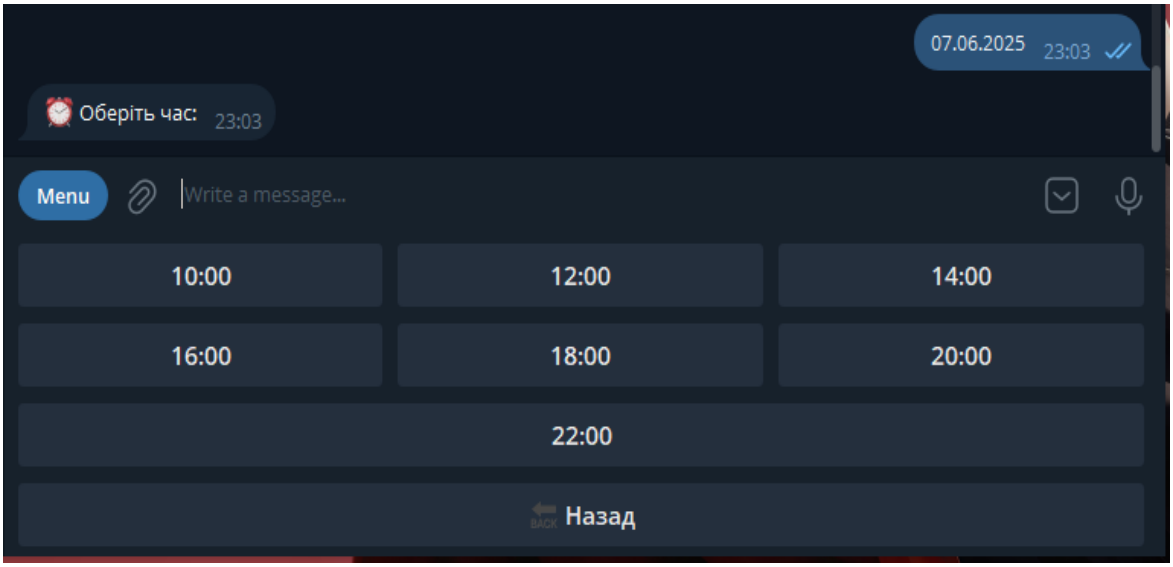

✅ Оформити замовлення 22:58
Будь ласка, введіть своє ім'я: 22:58
Андрій Щербій 22:59
Введіть адресу доставки: 22:59
Івано-Франківськ, вул. Будівельників 5, кв 10 22:59
Введіть номер телефону: 22:59
380974578909 22:59
✅ Ваше замовлення прийнято!
🍽 Страви
- Деруни: 2 × 70 грн = 140 грн
🍽 Салати
- Вінігрет: 1 × 70 грн = 70 грн
🍽 Напої
- Узвар: 1 × 35 грн = 35 грн
💰 Сума: 245 грн
👤 Ім'я: Андрій Щербій
🏠 Адреса: Івано-Франківськ, вул. Будівельників 5, кв 10
📱 Телефон: 380***
Дякуємо за замовлення! Очікуйте дзвінка. 22:59
Оберіть опцію: 22:59
Menu
Write a message...
🍽 Меню
🛒 Кошик
✏️ Залишити відгук
🍴 Забронювати столик

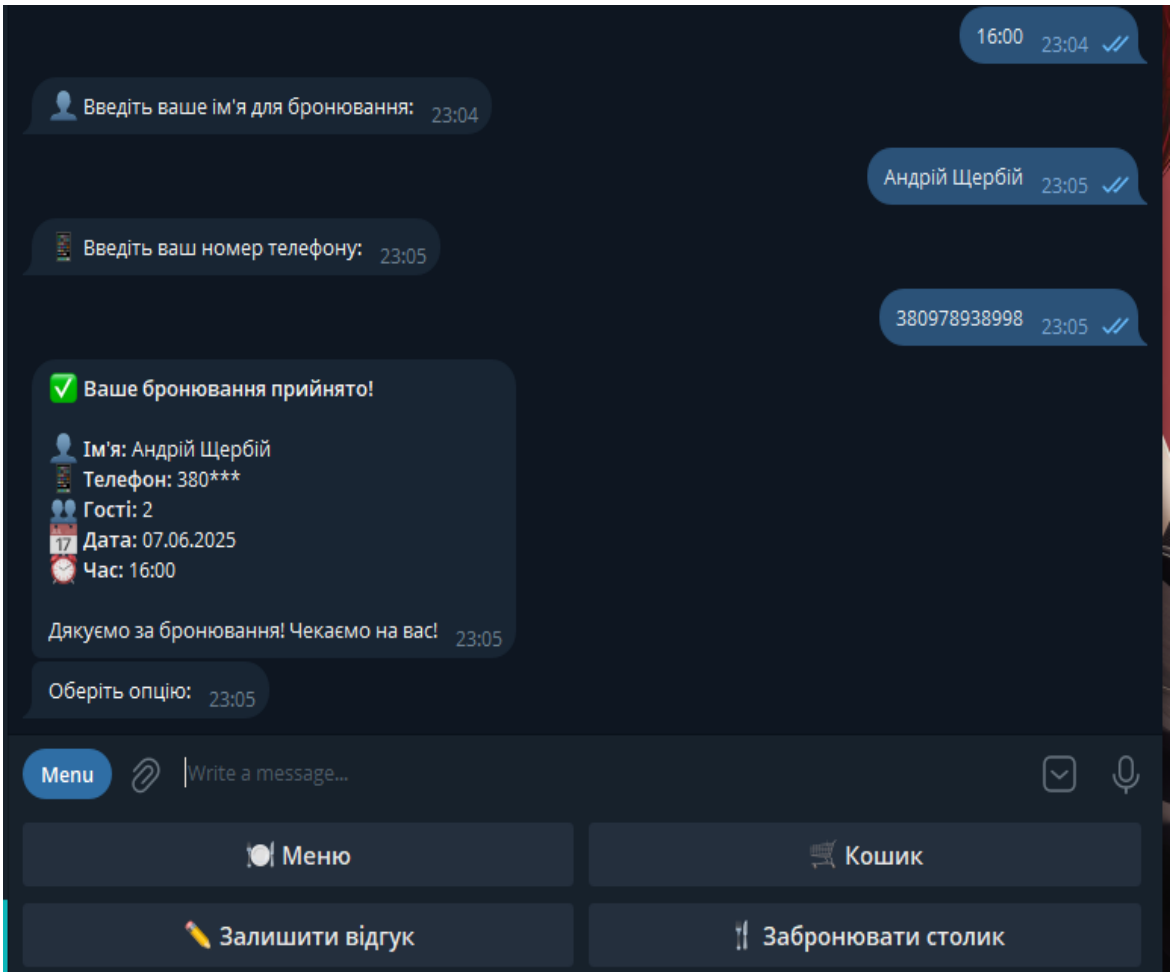

🆕 Нове замовлення
🍽 Страви
- Деруни: 2 × 70 грн = 140 грн
🍽 Салати
- Вінігрет: 1 × 70 грн = 70 грн
🍽 Напої
- Узвар: 1 × 35 грн = 35 грн
💰 Сума: 245 грн
👤 Ім'я: Андрій Щербій
🏠 Адреса: Івано-Франківськ, вул. Будівельників 5, кв 10
📱 Телефон: 380974578909 22:59


✏️ Залишити відгук 23:00
📝 Напишіть ваш відгук про наш сервіс: 23:00
Доставили швидко, та все було смачно! 23:01
❤️ Дякуємо за ваш відгук! 23:01
Оберіть опцію: 23:01
Menu
Write a message...
🍽 Меню
🛒 Кошик
✏️ Залишити відгук
🍴 Забронювати столик


Лісова Пісня
Ресторан "Лісова Пісня" →...
Відгук від
@the_vengeful_onee: Доставили...


Оберіть опцію: 23:01
🍴 Забронювати столик 23:03
👥 На скільки осіб бронюєте столик? 23:03
2 23:03
📅 Оберіть дату: 23:03
Menu
Write a message...
05.06.2025
06.06.2025
07.06.2025
08.06.2025
09.06.2025
10.06.2025
11.06.2025


07.06.2025 23:03
Оберіть час: 23:03
Menu
Write a message...
10:00
12:00
14:00
16:00
18:00
20:00
22:00
Назад


16:00 23:04
Введіть ваше ім'я для бронювання: 23:04
Андрій Щербій 23:05
Введіть ваш номер телефону: 23:05
380978938998 23:05
✅ Ваше бронювання прийнято!
Ім'я: Андрій Щербій
Телефон: 380***
Гості: 2
Дата: 07.06.2025
Час: 16:00
Дякуємо за бронювання! Чекаємо на вас! 23:05
Оберіть опцію: 23:05
Menu
Write a message...
Меню
Кошик
Залишити відгук
Забронювати столик


Нове бронювання
Ім'я: Андрій Щербій
Телефон: 380978938998
Гості: 2
Дата: 07.06.2025
Час: 16:00
23:05

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: “ Розробка бота ресторану для замовлення страв на доставку та бронювання столиків засобами Python”

Обсяг пояснювальної записки: 65 аркушів

Дата закінчення роботи 10 червня 2025 р.

Підпис _____