МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Чат-бот для месенджеру Телеграм по замовленню доставки їжі

Здобувача(ки) групи ІТ-91 Деменко Анастасії Миколаївни
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Анастасія ДЕМЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к. т. н., доц. Світлана ВАЩЕНКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«_____»____ 2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Деменко Анастасії Миколаївні

1 Тема роботи *Чат-бот для месенджеру Телеграм по замовленню доставки їжі*

керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент*,

затверджені наказом по університету від « 29 » травня 2023 р.

2 Строк подання студентом роботи « 19 » червня 2023 р.

3 Вхідні дані до роботи технічне завдання, спектр послуг закладу харчування та меню

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, постановка задачі, моделювання та проектування, розробка, використання програмного продукту, висновки

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) порівняння продуктів-аналогів чат-боту, архітектура програмного продукту, контекстна діаграма у нотації IDEF0, діаграма декомпозиції першого рівня у нотації IDEF0, діаграма варіантів використання, діаграма послідовності, діаграма діяльності, приклади роботи програмного продукту.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання__22.02.2023_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підготовка специфікації та визначення властивостей бота	до 21.04.2023	
2	Розробка меню бота	до 05.05.2023	
3	Створення та наповнення бази даних	до 20.05.2023	
4	Тестування	до 30.05.2023	
5	Перевірка працездатності	до 02.06.2023	
6	Написання супровідної документації	до 05.06.2023	
7	Реліз боту	06.06.2023	

Студент _____ Анастасія ДЕМЕНКО
(підпис)

Керівник роботи _____ к.т.н., доц. Світлана ВАЩЕНКО
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Чат-бот для месенджеру Телеграм по замовленню доставки їжі».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 22 найменувань, додатків. Загальний обсяг роботи – 102 сторінок, у тому числі 59 сторінки основного тексту, 3 сторінки списку використаних джерел, 40 сторінка додатків.

Кваліфікаційну роботу бакалавра присвячено розробці чат-боту для месенджеру Телеграм по замовленню доставки їжі.

У першому розділі була визначена актуальність роботи, сформована мета проекту, було досліджено літературні джерела які стосуються теми обраної роботи. А також провівся аналіз аналогів продукту, аналіз існуючих технологій які допоможуть у вирішенні проблем.

У другому розділі виконане структурно-функціональне моделювання, моделювання варіантів використання ІТ-продукту, інші діаграми та проектування моделі бази даних.

У третьому розділі описані основні етапи створення продукту та бази даних, наведені приклади використанні створеного програмного продукту.

Результатом є розроблений чат-бот для месенджеру Телеграм по замовленню доставки їжі.

Ключові слова: чат-бот, Телеграм, доставка, їжа, база даних, ІТ-продукт.

ЗМІСТ

ВСТУП

У порівнянні з минулим століттям, цифровий прогрес значно просунувся у своєму розвитку. У 1966 Вайценбаум розробив комп'ютерну програму під назвою «Еліза», яка і стала першим чат-ботом в історії [1]. У наступні десятиліття розробники чат-ботів спиралися на модель Вейценбаума, щоб прагнути до взаємодії, більш схожої на людську. Проходження тесту Тюрінга переросло в спільну мету, яка перевіряє навички спілкування нових ботів проти комісії суддів-людей [2].

На разі всі сфери діяльності людини - освіта, підприємництво, військова, медицина, харчова промисловість – використовують цифрові технології, до яких відносяться і чат-боти.

Використання в охороні здоров'я – персоналізований медичний асистент посилається на алгоритми штучного інтелекту для підтримання щоденних розмов, та рекомендує заняття для підтримання здоров'я.

Подорожі – ці боти можуть рекомендувати план подорожей, оснований на персональних вподобаннях з історії подорожей яка була отримана з попереднього рейсу, готелю, оренди автомобіля. Потім можу виконуватися фільтрування на основі рейтингів.

Освіта – чат-боти можуть бути використані для навчання студентів концепції базових комп'ютерних наук, адаптуючись під потреби окремого учня. За допомогою влаштованих модулів можна відслідковувати якість та прогрес навчання, для підвищення загальної якості освіти.

Фінанси – оскільки фінансова галузь стає все більш дерегульованою, багато фінансових операцій оцифровується і це дозволяє фінансовому бізнесу надавати різноманітних послуг онлайн. Наприклад, чат-боти можуть використовуватися для допомоги фінансових консультантів із стратегіями прийняття рішень, базованих на попередньо виконаних фінансових операціях чи тенденціях [3].

Чат-боту в крос-платформній системі Telegram для мережі швидкого харчування значно полегшить процес отримання їжі, так як Telegram є одним із найуживаніших додатків для телефону та комп’ютерів на даний момент, тож сам процес не займатиме багато часу.

Основною метою цієї роботи є розробка чат-бот для месенджеру Телеграм по замовленню доставки їжі.

Мета може бути досягнута за рахунок вирішення таких задач:

- аналіз програмних аналогів по замовленню їжі з метою визначення необхідного функціоналу бота, підготовка специфікації;
- моделювання роботи бота;
- розробка меню при спілкуванні з користувачем, створення та наповнення бази даних;
- тестування та створення релізу боту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Актуальність створення чат-ботів обумовлена не тільки тим, що вони доступні 24/7 для користувача, а ще й такими чинниками:

- швидкі внутрішні процеси (питання користувача - відповідь);
- постійна доступність необхідної інформації, відсутня необхідність в поглиблених знаннях користування комп'ютерними технологіями;
- необмежений запас терпіння яке іноді необхідне при спілкуванні.

Telegram набирає великої популярності, бо в порівнянні, наприклад, з Facebook, він має набагато менше проблем з безпекою[5].

Всесвітньо відомі боти : «Perry», «Siri», «Google now / Google Assistant», «Cortana», «Alexa», «ChatGPT». Ці боти наразі є найвідомішими помічниками для користувача мобільних пристроїв [2].

Чат-боти Telegram публічного користування які часто використовуються українцями в 2022-2023 роках : «єВорог» - чат-бот, за допомогою якого українці можуть повідомити про рух та розташування окупантів та ворожої техніки, «SaveEconomyBot» - повідомлення щодо незаконних дій з гуманітарною допомогою, фактів зловживання з бюджетними коштами, перешкоджання бізнесу, незаконний обіг підакцизних товарів, прихованих активів країни-агресора, інших економічних правопорушень [6], «АТБ» - мережа магазинів [7], «PrivatBank» - бот ПриватБанку [8] та багато інших.

Головна функція (або навіть завдання) віртуального помічника – це розпізнати запит клієнта та коректно відреагувати на нього.

До основних переваг чат-бота відноситься такі [9]:

- оптимізація витрат. Потрібно лише один раз вкласти кошти у розробку або придбання чат-бота. Тільки уявіть, як ви зможете автоматизувати рутинні процеси компанії та направити сили на інші, важливіші питання;

- цілодобова підтримка. Як було зазначено вище, чат-бот може відповідати навіть у нічний час та на вихідних, коли компанія відпочиває. Це також дає можливість обробляти лідів без участі команди;
- глибший аналіз спілкування. Залежно від програмного забезпечення, можна проаналізувати контент, який був запропонований користувачу, скільки разів, яка була відповідь тощо;
- збільшення залученості аудиторії. Завдяки чат-боту, люди отримують інформацію швидко, а вчасно надана підтримка майже гарантує “підігрітого” ліда, що потім може перетворитися на клієнта;
- генерація та кваліфікування лідів. Якщо ви правильно побудуєте ланцюжок відповідей у системі чат-бота, це може привести покупця від першого контакту до оформлення замовлення. Потім, можна зробити більш точкову та персоналізовану розсилку на різних етапах продажів [9];

Результати дослідження State of Chatbots 2018 року показали, що споживачі повідомили, що 43% віддають перевагу спілкуванню з людиною, 30% хвилюються, що чат-бот зробить помилку, наприклад під час покупки або під час бронювання, а також 27% обмежуються використанням чат-ботів лише через Facebook. Однак варто також зазначити, що 15% споживачів сказали, що ніщо не завадить їм використовувати чат-ботів[10]. І ця цифра був незалежна від всіх вікових груп. Висновок: не всі споживачі готові відмовитися взаємодії між людьми. Наведені дані відображені на рисунку 1.1

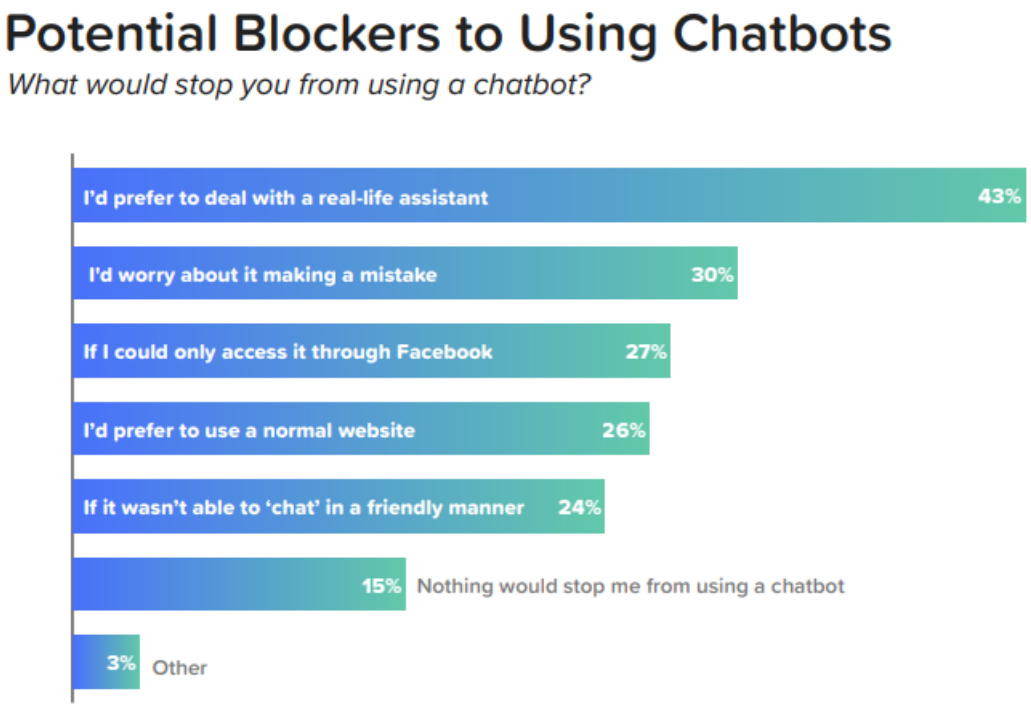

Рисунок 1.1 – Чинники, через які люди не використовують чат-боти

Чат-боти — це нові додатки, але вони ще повністю не замінили потребу в телефоні та електронній пошті. У п’яти з десяти споживачі віддають перевагу чат-ботам, а не програмам у розглянутих категоріях можливостей, які включали не лише отримання швидких відповідей на прості запитання та 24 год обслуговування, але й отримання швидких відповідей на складні питання та отримання детальних/експертних відповідей [10]. Отримана статистика відображена на рисунку 1.2.

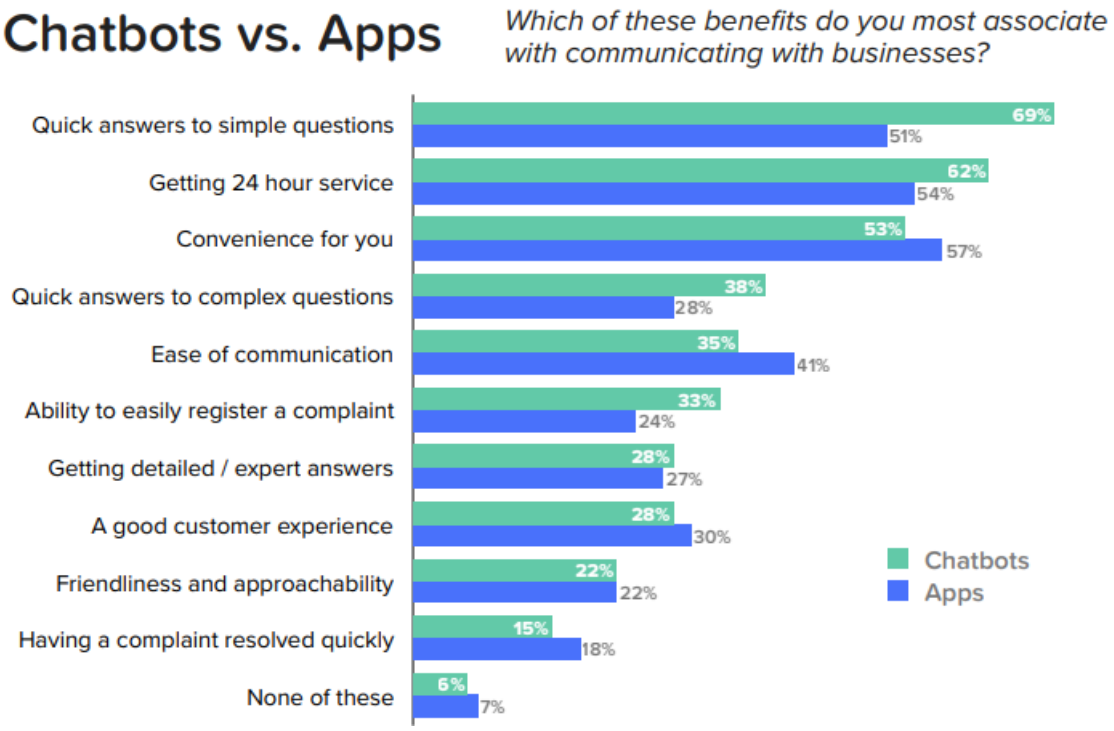

Рисунок 1.2 – Порівняння чат-ботів та додатків у використанні

1.2 Аналіз програмних продуктів – аналогів

На рисунку 1.3 скріншот Telegram-боту М’ясторія з доставки їжі [11]. Після старту відображується меню бота з різними функціями : про нас, меню, ресторани, контакти, самовивіз, доставка і оплата, особиста інформація, зв’язок з менеджером.

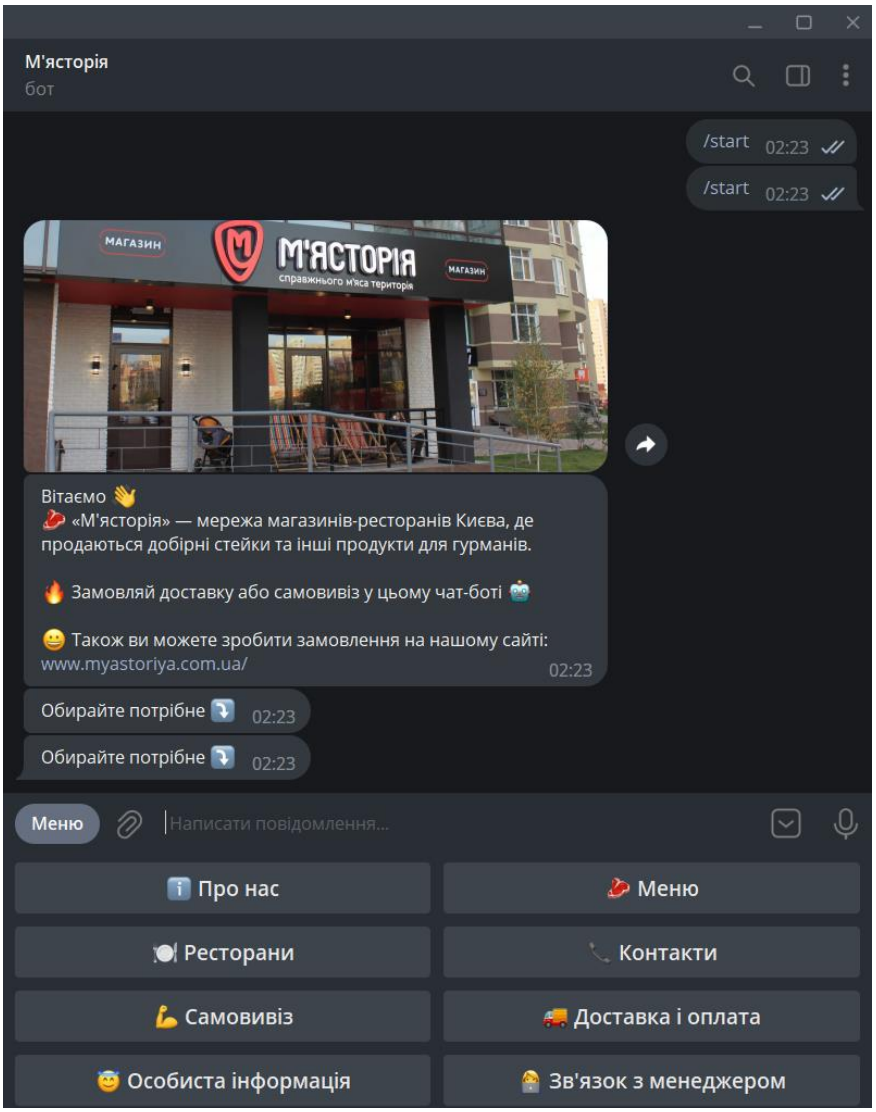

Рисунок 1.3 – Вигляд запущеного боту М’ясторія

Після натискання кнопки «про нас», з’являється повідомлення з інформацію про мережу магазинів М’ясторія, це відображено на рисунку 1.4

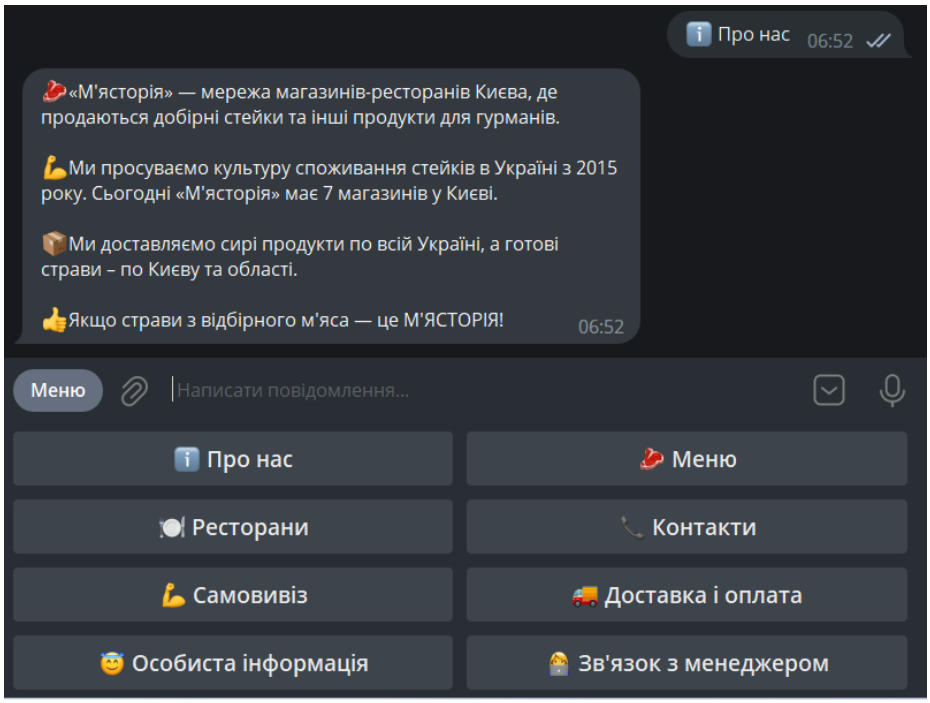

Рисунок 1.4 – Вигляд бота після натискання кнопки «про нас»

Після натискання кнопки для перегляду меню, з’являється нове меню з кнопка ми для перегляду кошика, гриль меню, ресторанного меню та сирих стейків, рисунок 1.5.

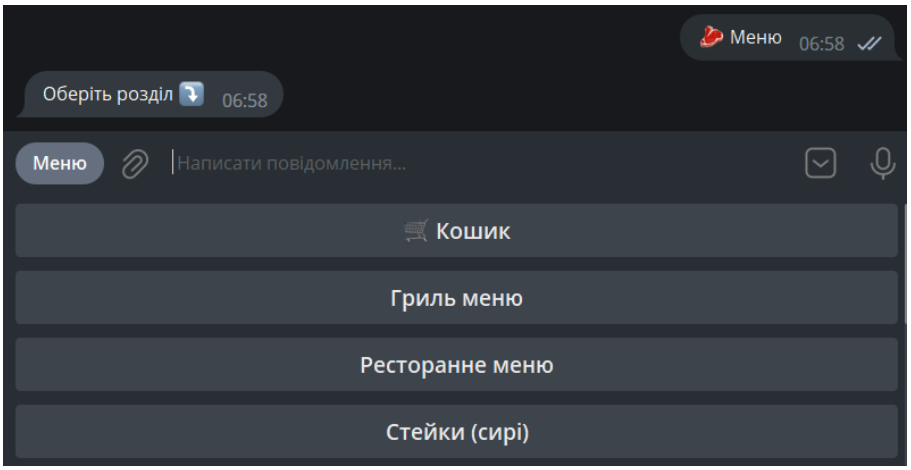

Рисунок 1.4 – Вигляд бота після натискання кнопки «про нас»

Якщо користувач натиснув «гриль меню», відображаються нові інлайн кнопки, з різними пунктами меню для кожного виду стейків, рисунок 1.5.

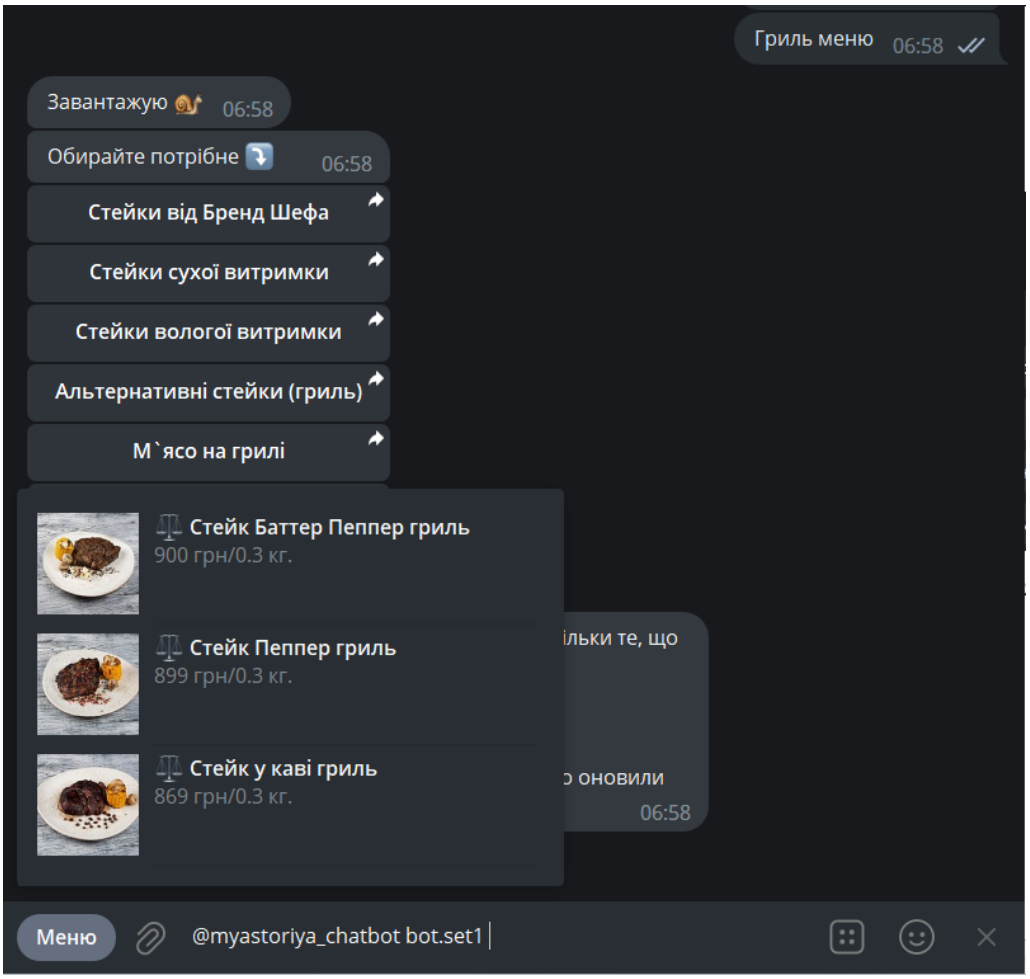

Рисунок 1.5 – Вигляд бота після натискання кнопки «про нас»

Обраний стейк відображається після вибору користувача та надається можливість повернутися у попереднє меню чи додати страву у кошик, це відображено на рисунку 1.6

Рисунок 1.6 – Обрана користувачем страва

Після того, як страва була додана у кошик, відображається інформація про обрану страву – її назва, ціна та загальна сума за всі страви. Також з’явилося нове меню з кнопками «очистити», «оформити», «до розділу», «змінити», рисунок 1.7

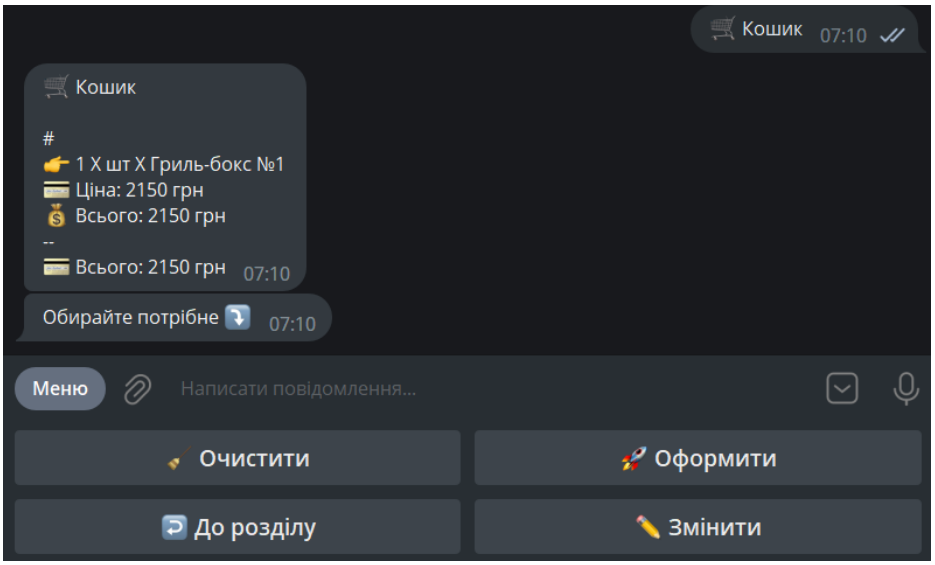

Рисунок 1.7 – Відображення кошику

Оформлення доставки відображено на рисунку 1.8

Рисунок 1.8 –Налаштування доставки

Також є можливість налаштувати профіль для клієнта – ПІБ, телефон, електронну пошту та кнопка «вийти», рисунок 1.9

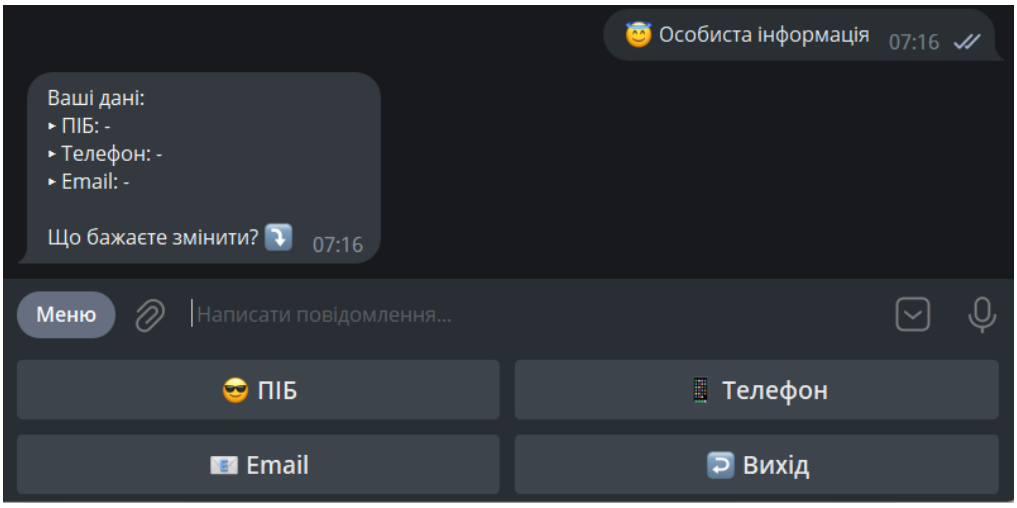

Рисунок 1.9 – Перегляд особистої інформації

Після натискання кнопки «зв’язок з менеджером» користувачеві надалося посилання на чат з оператором, рисунок 1.10

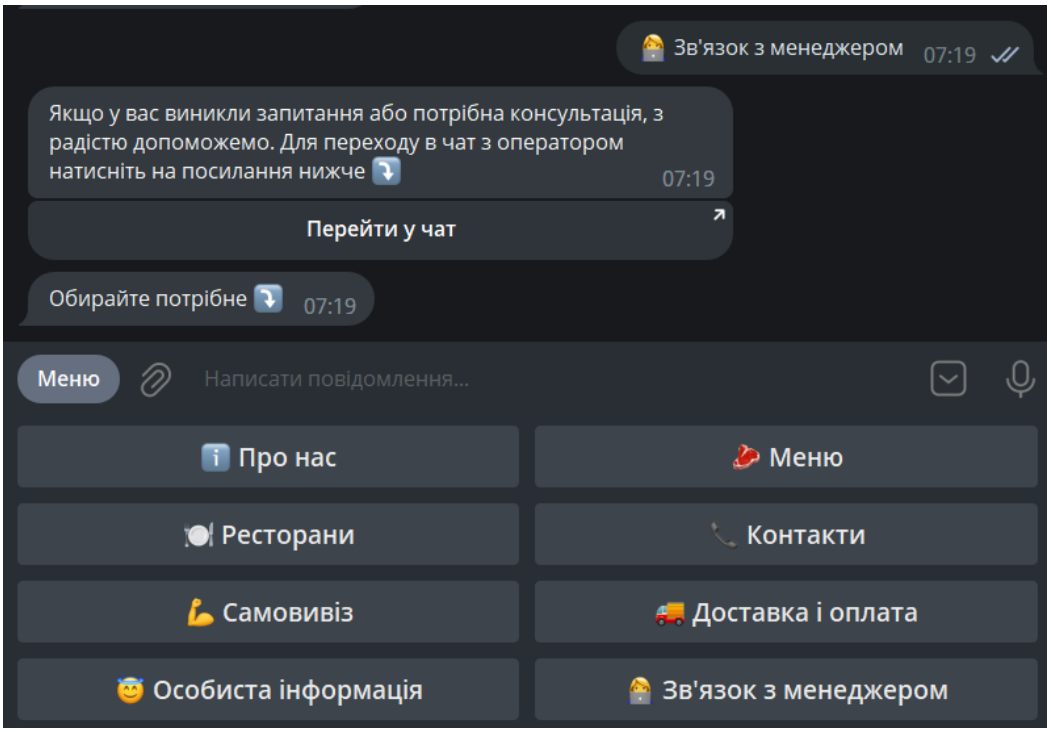

Рисунок 1.10 – Перегляд особистої інформації

На рисунку 1.11 скріншот Telegram-боту TRUE ICE BOT [12]. Бот ще не запущений, але присутній опис бота та його функції та вказівна для подальших дій користувачеві.

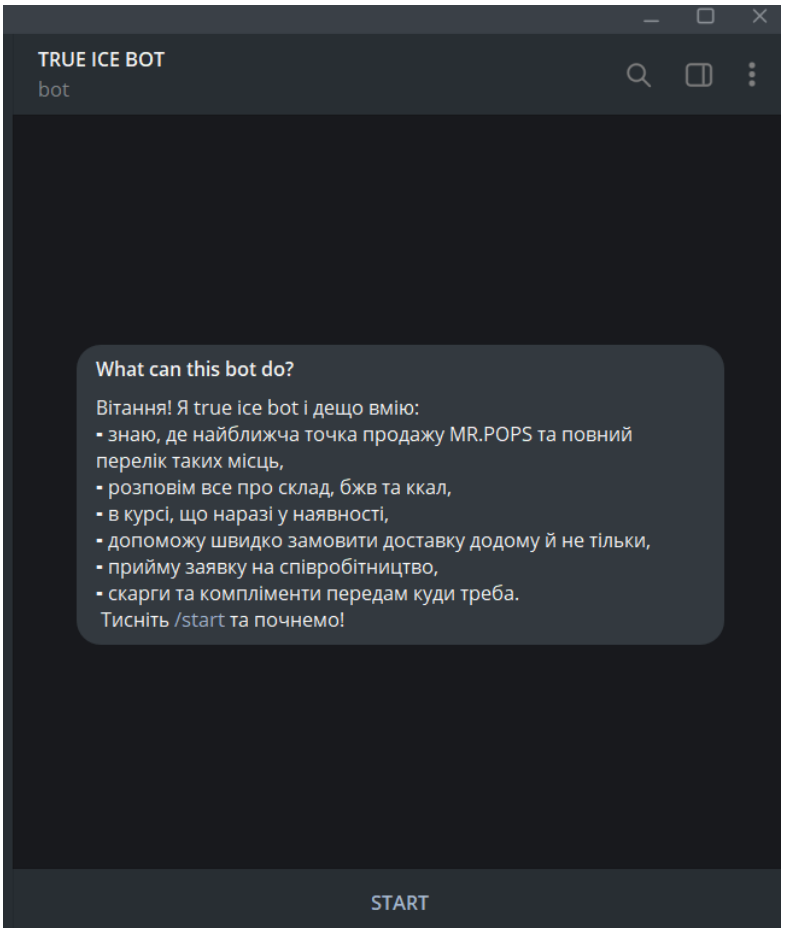

Рисунок 1.11 – Кошик для замовлення

На рисунку 1.12 скріншот, на якому меню бота, яке відкрилося після запуску бота для вибору міста для доставки та можливість змінити мову бота.

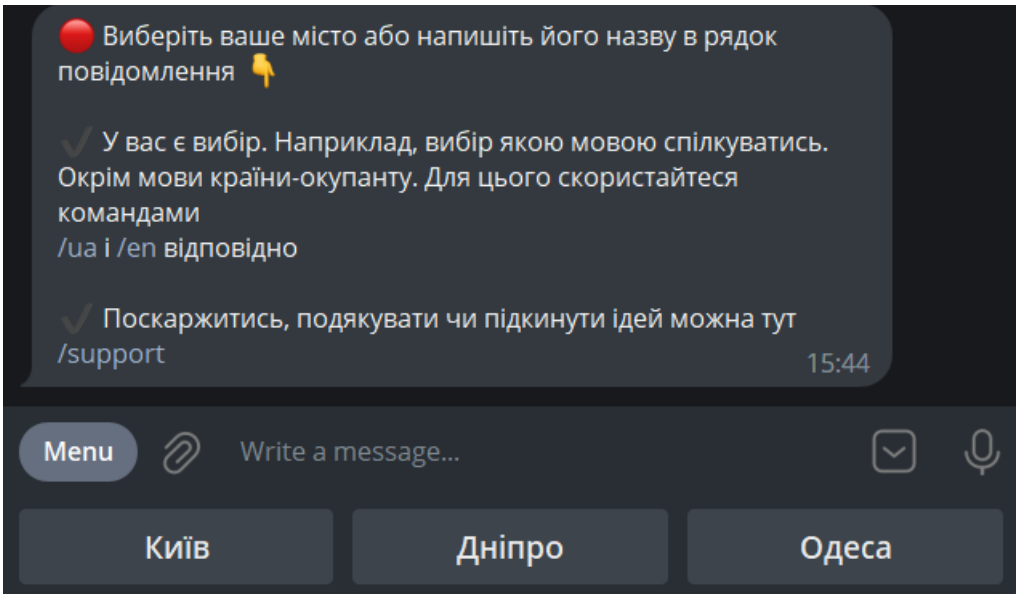

Рисунок 1.12 – Кошик для замовлення

Після обраного міста з’являється нове меню з переліком морозива. Нове меню з продуктами для доставки відображене на рисунку 1.13

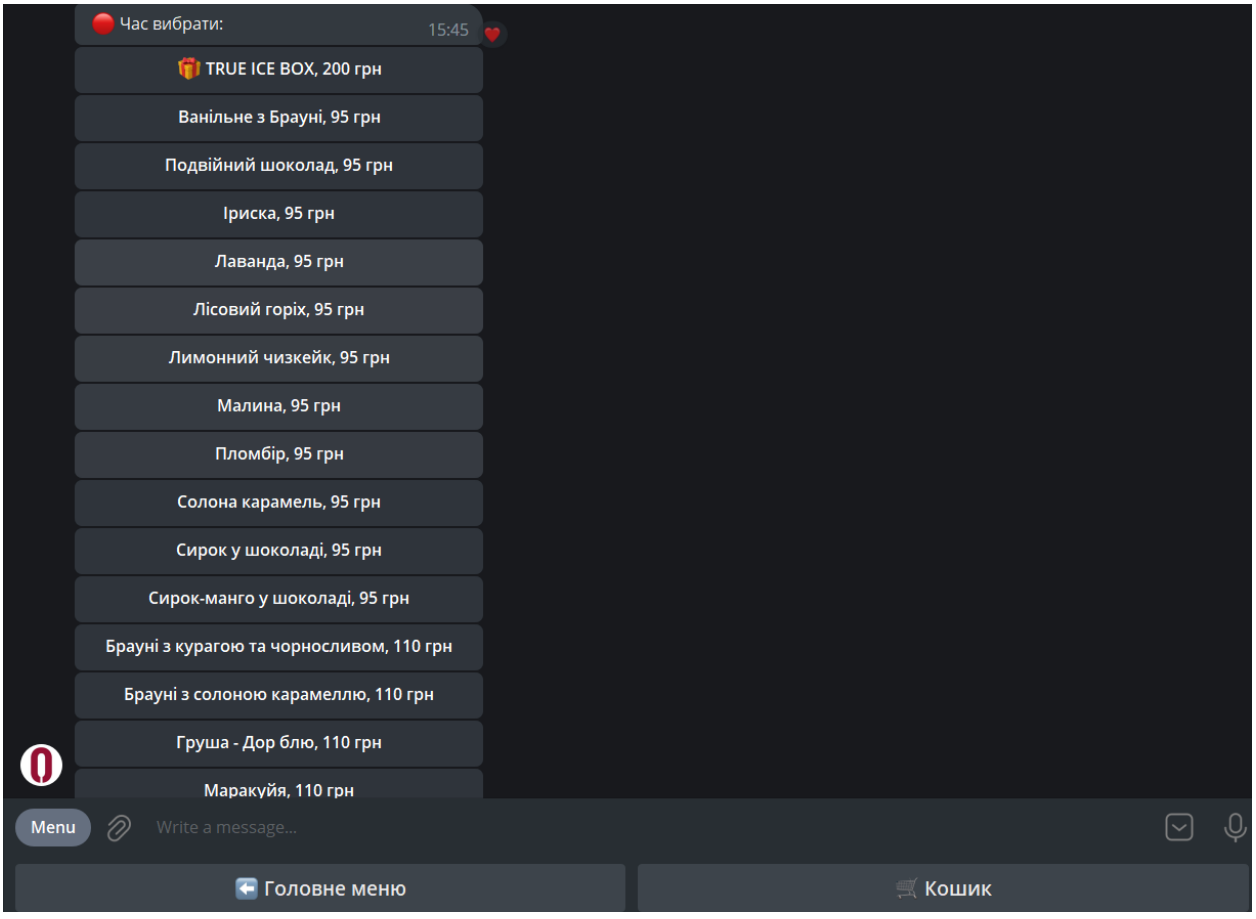

Рисунок 1.13 – Кошик для замовлення

Таблиця 1.1 – Порівняння ботів

Характеристика	TRUE ICE BOT	М’ясторія
Опис бота перед запуском	-	+
Опис пункту меню	+	+
Картинка страви з меню	+	+
Задіяти промокод	-	-
Змінити місцеположення	+	кнопки для вибору з існуючих точок
Замовлення поштучно	-(лише > 30 шт)	+
Зміна особистих даних профілю користувача	-	+

1.3 Постановка задачі

Чат-бот для месенджеру Телеграм по замовленню доставки їжі значно полегшить процес замовлення їжі та перегляд/вибір страв на замовлення, так як Telegram є одним із найуживаніших додатків для телефону та комп’ютерів на даний момент, тож сам процес не займатиме багато часу.

Основні вимоги до створюваного програмного продукту:

- викладене меню страв, варіант замовлення певної страви (наприклад торту для банкету з варіантом самовивозу), бронювання столика, замовлення доставки, робітники закладу можуть перевіряти замовлення та редагувати їх;
- розробити діалог між користувачем і ботом;
- у діалозі має бути сценарій для кожного виходу (описані можливі кроки користувача).

Для досягнення мети проекту необхідно виконувати наступні задачі:

- побудувати контекстну діаграма процесу замовлення їжі через телеграм-бот у нотації IDEF0, діаграму декомпозиції першого рівня ; діаграма варіантів використання, діаграма послідовності, діаграма діяльності;

- розробити базу даних: таблиця користувачів, страви меню, вільні столики;
- обрати технології розробки боту;
- виконати практичну реалізацію та провести тестування.

Цільовою аудиторією даного проекту є робітники закладу та люди які хочуть скористатися послугами, які зацікавлені у полегшеному процесі обслуговування.

Використання бота не потребує додаткових спеціалістів, спрощує процес сприйняття інформації. Проект є життєздатним бо є необхідність у людини харчуватися та просте використання, доступність 24/7, швидкість внутрішніх процесів. Програма має інтерфейс написаний українською мовою, назви деяких страв меню можуть бути на англійською мовою.

Технічне завдання на розробку продукту у повному обсязі наведено у додатку А.

Для реалізації Telegram-боту було обрано такі технології, як мову запитів SQL [13], мову програмування Python, середовище програмування PyCharm та полегшена реляційна система керування базами даних DB Browser.

Мову програмування Python було обрано для даної задачі через те, що є вже розроблені бібліотеки для створення Telegram-ботів, методи управління даними, методи прив’язки бази даних та управління витягнутих даних з бази – додавання, видалення, оновлення, редагування [14]. PyCharm має такі функції:

– графічний налагоджувач;
– інтегрований блок-тестер;
– підтримка інтеграції систем контролю версій (VCS)[15].

DB Browser - це високоякісний візуальний інструмент із відкритим кодом для створення, проектування та редагування файлів бази даних. Головні переваги у використанні :

– створення файлів баз даних;
– створення, визначення, зміна та видалення таблиць; створення, визначення та видалення індексів;

- перегляд, редагування, додавання та видалення записів; імпорт та експорт таблиць із/у файли CSV;
- надсилання SQL-запитів та перевірка результатів; перегляд журналу усіх команд SQL, виданих програмою;
- побудова простих графіків на основі даних таблиці або запиту [16].

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ЧАТ-БОТУ

2.1 Структурно-функціональне моделювання

IDEF0 - це методологія графічного опису систем і процесів діяльності організації як безлічі взаємозалежних функцій.

Інформаційні та матеріальні потоки, які перетворюються в бізнес-процесі позначають стрілками зліва та є входами. Матеріальні та інформаційні потоки, які перетворюються на процеси познають стрілками зверху та є управлінням Інструменти та ресурси, за допомогою яких бізнес-процес реалізується позначаються стрілками знизу та є механізмами. Вихід бізнес-процесу, описаного в стандарті IDEF0, повністю відповідає за змістом виходу процесу [17].

Процес зазначений як «Оформлення замовлення за допомогою чат-боту».

Входами до процесу є «попередній перелік бажаних страв», «дата та час бронювання», «потрібна операція».

Керуванням є «меню страв», «перелік доступних послуг» та «Правила користування».

Механізми : «технічні засоби зі встановленим Telegram», «rapid_eats_bot».

Результати : «сформоване замовлення», «оформлення бронювання».

Діаграма IDEF0 для чат-боту для месенджеру Telegram по замовленню доставки їжі зображена на рисунку 2.1

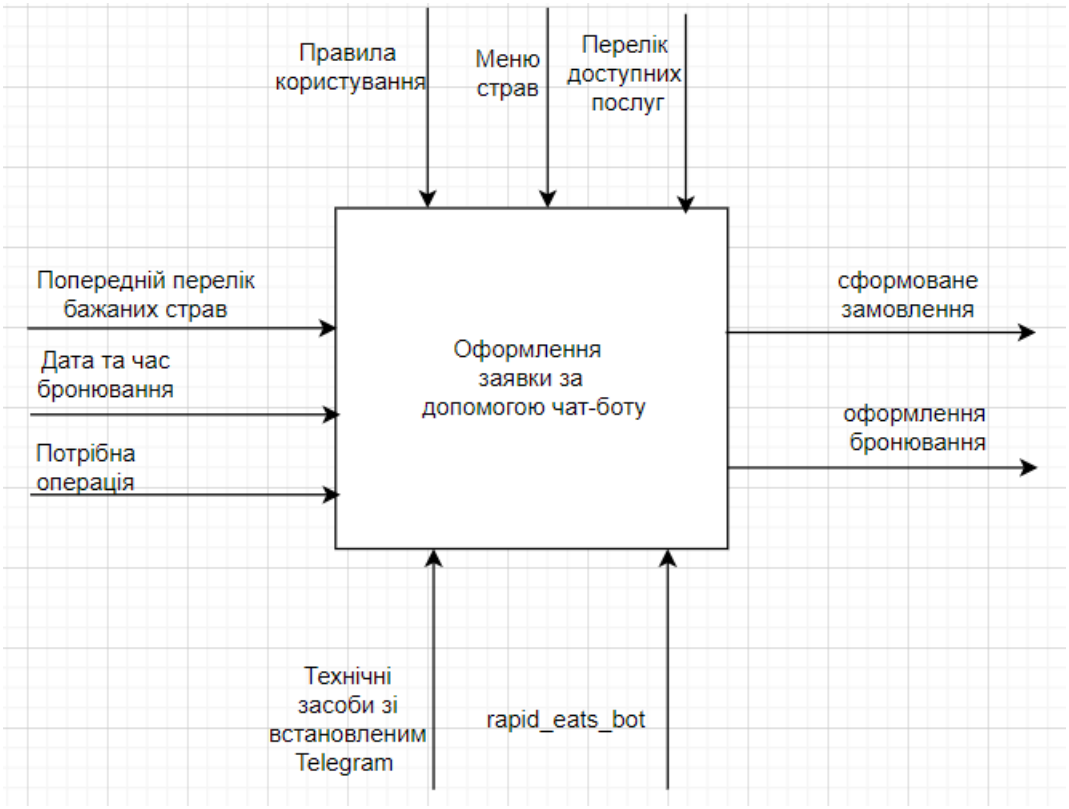

Рисунок 2.1 – Контекстна діаграма бізнес-процесів у нотації IDEF0

Дані для діаграми декомпозиції поміщено в таблицю 2.1

Таблиця 2.1 – Дані діаграми декомпозиції IDEF0

Підпроцес	Вхід	Управління	Механізм	Вихід
Вибір пункту меню	Потрібна операція	Правила користування, перелік доступних послуг	Технічні засоби зі встановленим Telegram, rapid_eats_bot	Оформлення замовлення

Продовження таблиці 2.1 – Дані діаграми декомпозиції IDEF0

Вибір категорії меню страв	Потрібна операція	Правила користування, перелік доступних послуг, меню страв	Технічні засоби зі встановленим Telegram, rapid_eats_bot	Оформлення замовлення
Перегляд меню	Потрібна операція	Правила користування, меню страв	Технічні засоби зі встановленим Telegram, rapid_eats_bot	Оформлення замовлення
Подача заявки на замовлення	Потрібна операція	Правила користування, меню страв	Технічні засоби зі встановленим Telegram, rapid_eats_bot	Оформлення замовлення
Подача заявки на бронювання	Дата та час бронювання	Правила користування, перелік доступних послуг	Технічні засоби зі встановленим Telegram, rapid_eats_bot	Оформлення бронювання

Для деталізації процесів було виконано декомпозицію першого рівня – рисунок 2.2

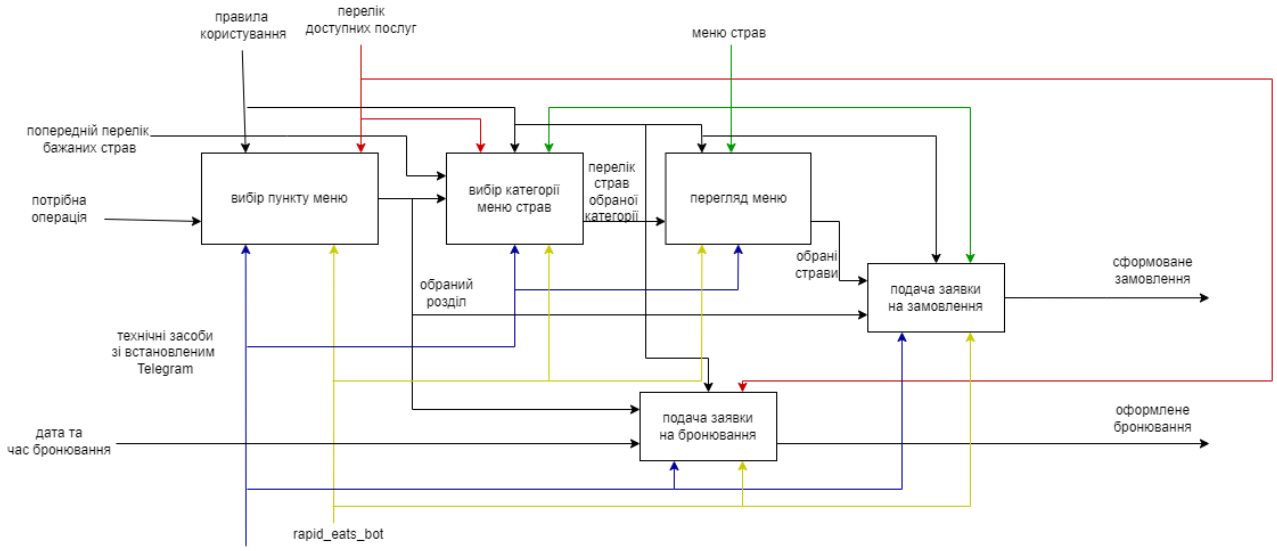

Рисунок 2.2 – Діаграма першого рівня декомпозиції IDEF0

2.2 Моделювання варіантів використання

Діаграма варіантів використання — дозволяє уявити типи ролей та їх взаємодію із системою. Але порядок виконання кроків не показується. З точки зору користувача описуються функціональні вимоги. Може описуватись текстом або у вигляді діаграми.

Діаграми варіантів використання складаються з 4 об’єктів: система, актор, варіант використання, зв’язок.

Актор – хтось чи щось, що взаємодіє з системою, та знаходиться за межами системи .Позначається у вигляді чоловічка та підписується (наприклад, клієнт чи адмін).

Всі актори є розділеними на два типи : на первинних (ті, що ініціюють взаємодію з системою) та вторинних (ті що реагують на взаємодію). Первинні зазвичай зображуються зліва, а вторинні — справа від системи.

Випадок використання визначають очікувану поведінку та відповідають на питання, що робить система. Представляють набір можливих функцій, дій або завдань. Зображується у вигляді еліпса з назвою дії (дієсловом) у ньому. Прецедент вказує, що має трапитись, проте не відповідає на запитання, як це має статись.

Система-це що моделюється. (сайт, мобільний додаток або навіть модуль програмного забезпечення). Зображується у вигляді прямокутника з назвою системи у верхній частині.

Зв’язки (відношення). Усі актори мають бути пов’язані з прецедентом (випадками використання). Частіше всього для зображення зв’язку використовується суцільна лінія.

Всього існує чотири типи зв’язків : асоціація, розширення, включення, генералізація [18].

На рисунках 2.3 – 2.4 відображені діаграми варіантів використання для адміністратора та користувача.

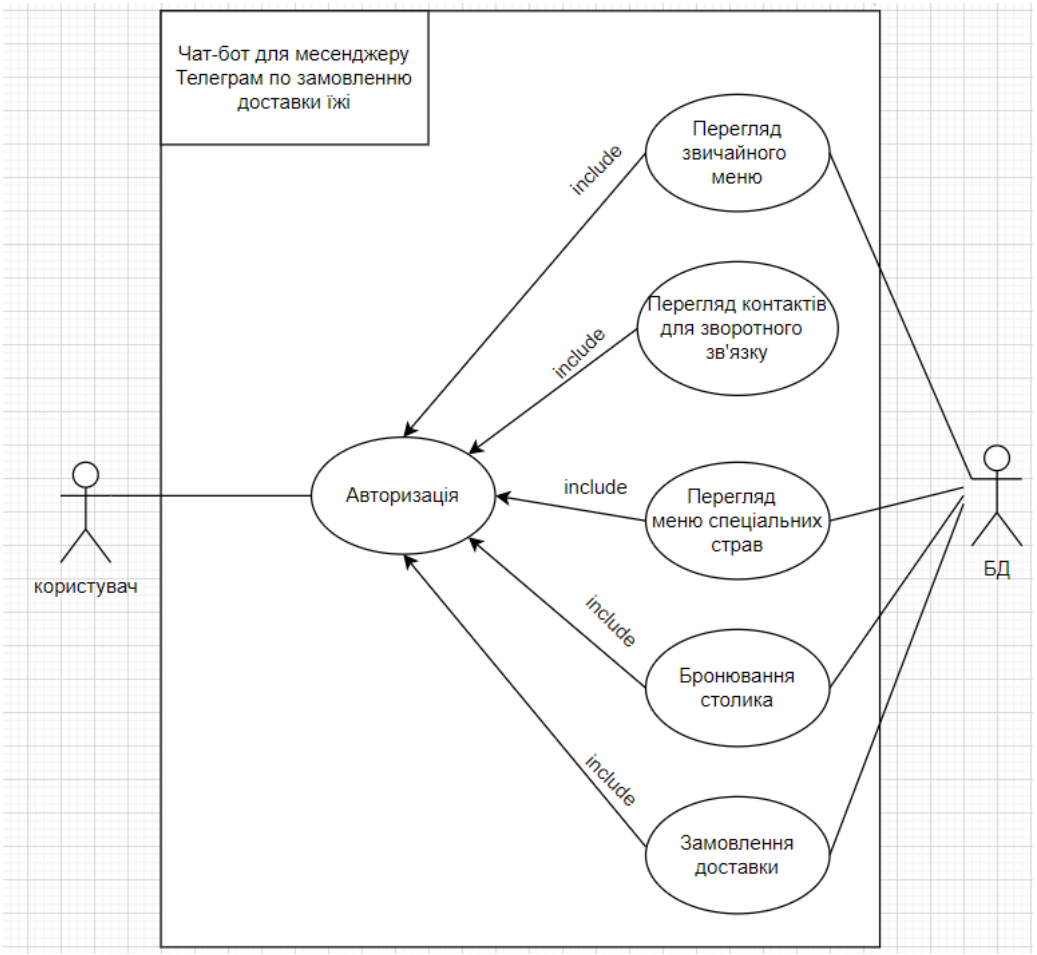

Рисунок 2.3 – Діаграма варіантів використання для користувача

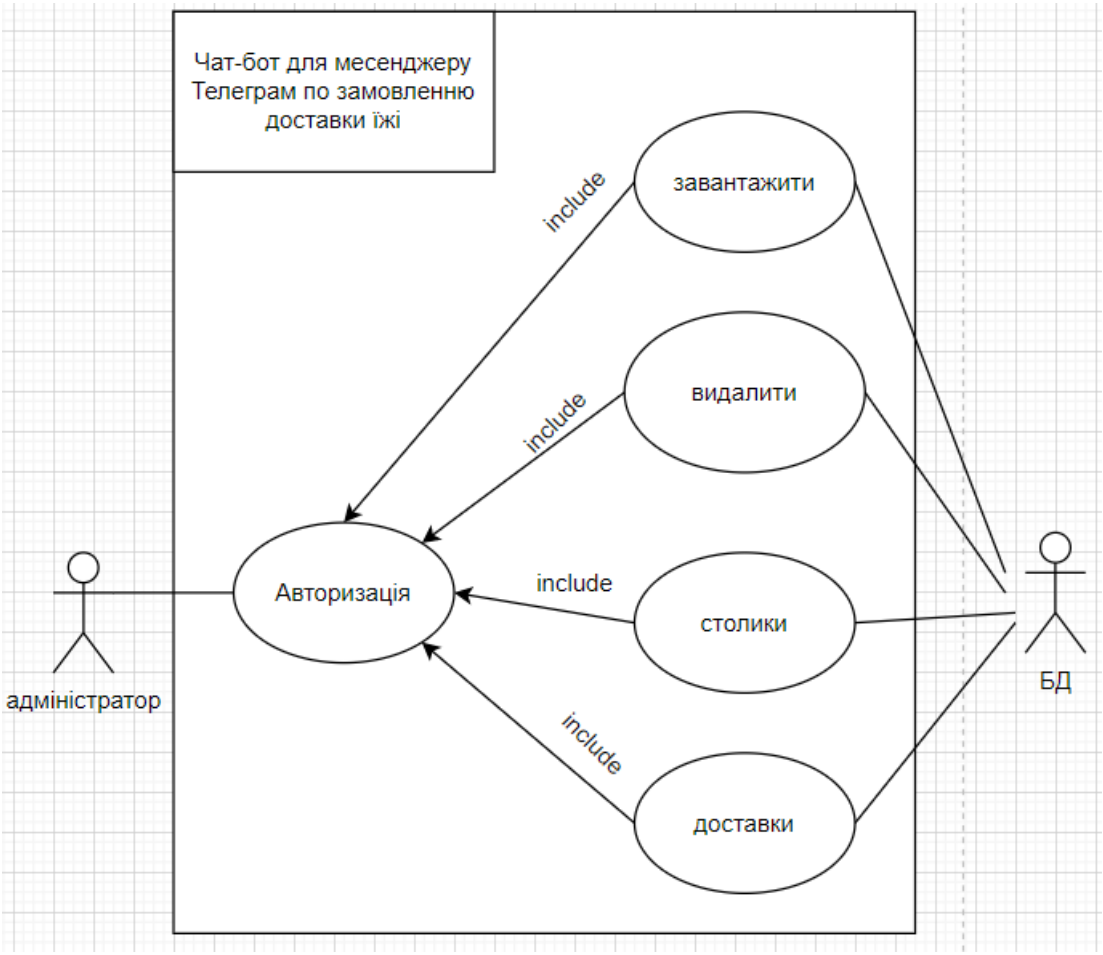

Рисунок 2.4 – Діаграма варіантів використання для адміністратора

Клієнт після авторизації має можливість подивитися меню звичайних страв, меню спеціальних страв, замовити доставку, замовити столик, переглянути контакти. Адміністратор після авторизації може додати нову страву до меню, видалити страву з меню, переглянути та опрацювати замовлені столики та доставки.

2.3 Діаграми послідовності

Діаграма послідовності (Sequence Diagram) — показує часові особливості передачі і прийому повідомлень об’єктами. Впорядкованість за часом слід розуміти як послідовність дій і не плутати з часовими діаграмами.

Елементи діаграми послідовності :

– Об’єкт (учасник) — позначення лінії життя та екземпляр класу у горизонтального прямокутника.

– Актор — використовується, коли конкретна діаграма послідовності належить варіанту використання.

– Сутність (entity) — представляє системні дані. Наприклад, у програмі обслуговування клієнтів суб’єкт — клієнт керує даними (сутність), пов’язаними з клієнтом.

– Межа/кордон (boundary) — вказує на межу системи/ граничний елемент у системі; наприклад, екрани інтерфейсу користувача, шлюзи баз даних або меню, з якими взаємодіють користувачі

– Управління (control) вказує на керівну сутність або менеджера. Він організовує та планує взаємодії між кордонами та сутностями та служить посередником між ними [18].

На рисунках 2.5-2.6 зображено діаграми послідовностей для дій адміністратора та користувача.

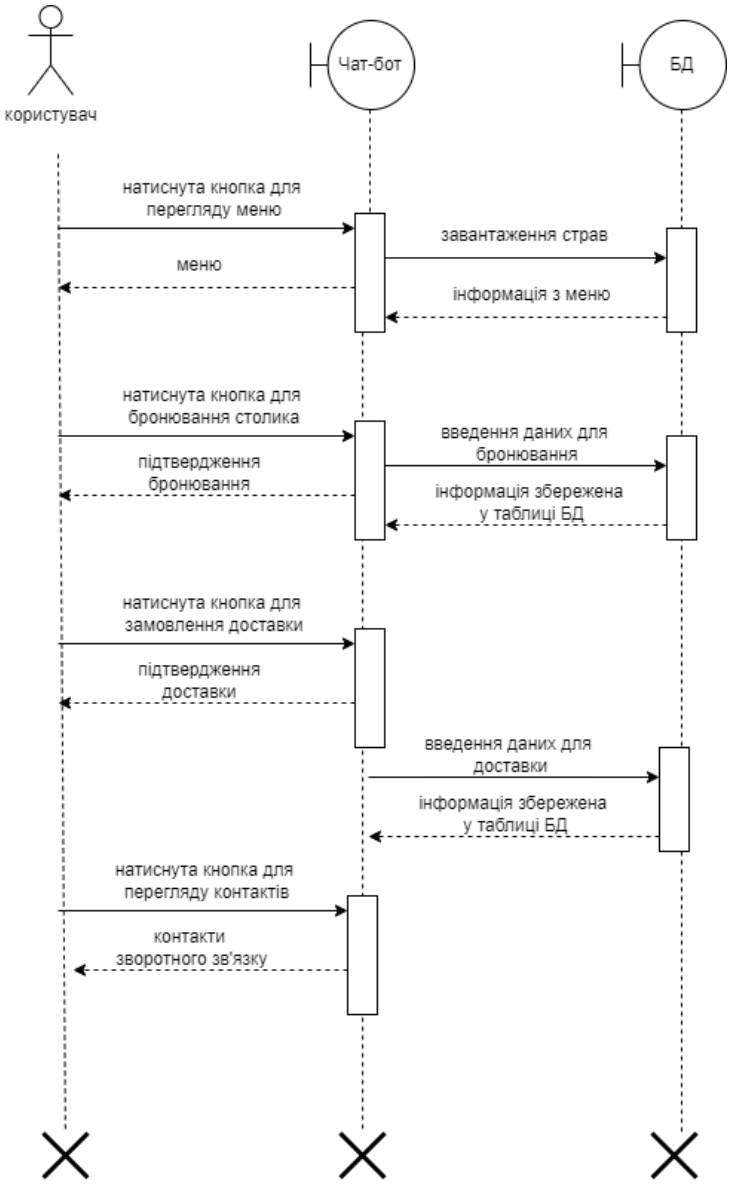

Рисунок 2.5 – Діаграма послідовностей для користувача

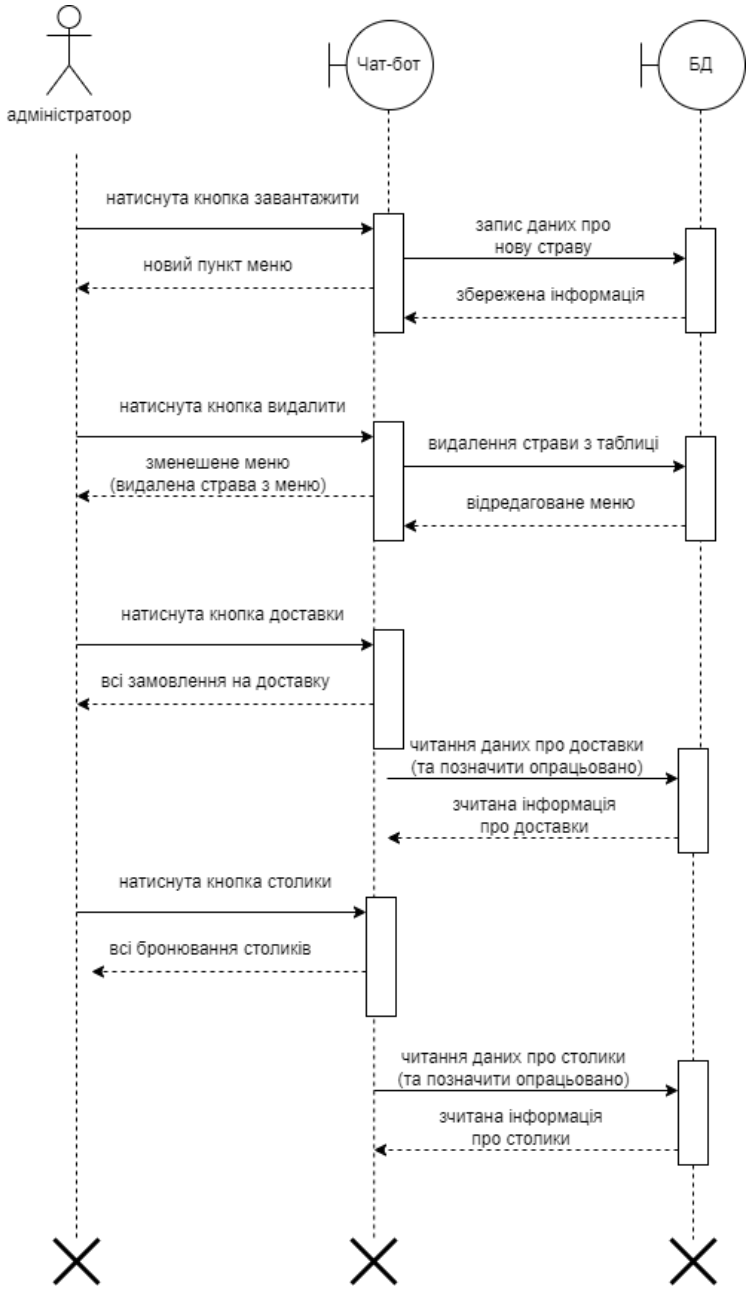

Рисунок 2.6 – Діаграма послідовностей для адміністратора

2.4 Діаграми діяльності

Діаграма діяльності (Activity Diagram) візуалізує процес використання та ілюструє потік повідомлень від однієї дії до іншої. Показує цілісну роботу системи.

Головна мета діаграми послідовності — показати порядок виконання, або послідовність дій. Водночас діаграма діяльності потрібна для опису роботи всієї системи, вона показує перехід від однієї дії до іншої.

Ці дії можуть виконуватися людьми, програмними компонентами або комп’ютерами. Потік керування (порядок виконання) на діаграмі діяльності переходить від однієї операції до іншої. Цей потік може бути послідовним, розгалуженим або одночасним [18].

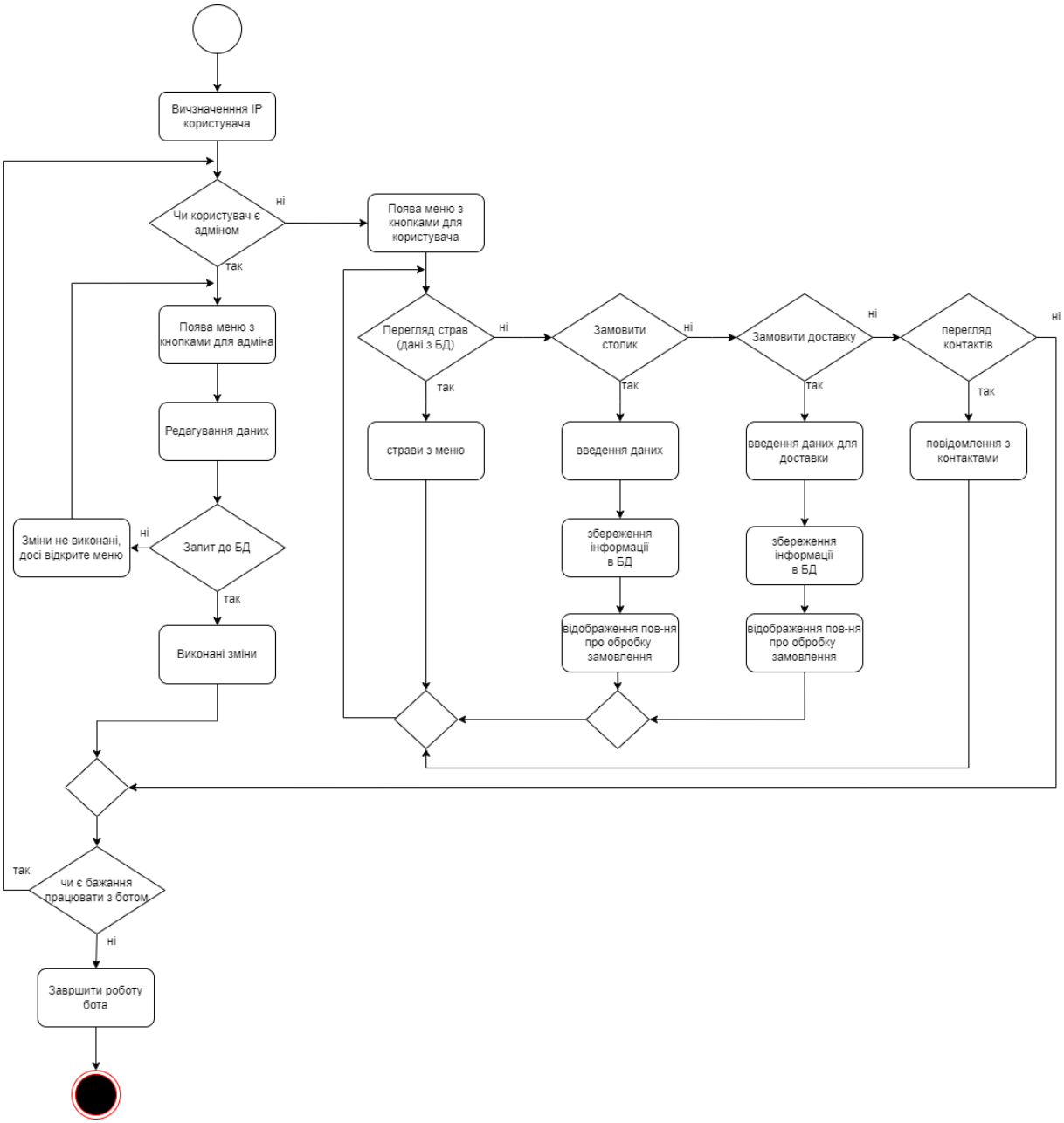

Рисунок 2.7 – Діаграма діяльності

2.5 Моделювання бази даних

****

Таблиця menu відповідає за всі наявні страви, і звичайні, і спеціальні. Має один первинний ключ name. Наявні поля : name, img, description, price, dish_type. Саме у полі dish_type зберігаються значення «спеціальна», та «звичайна». І при виведенні страви з конкретного меню користувачу саме ці значення використовуватимуться. Таблиця delivery містить інформацію про доставку. Один користувач може замовити декілька страв з меню, використаний зв’язок «one mandatory to many optional». Таблиця tables містить інформацію про столики.

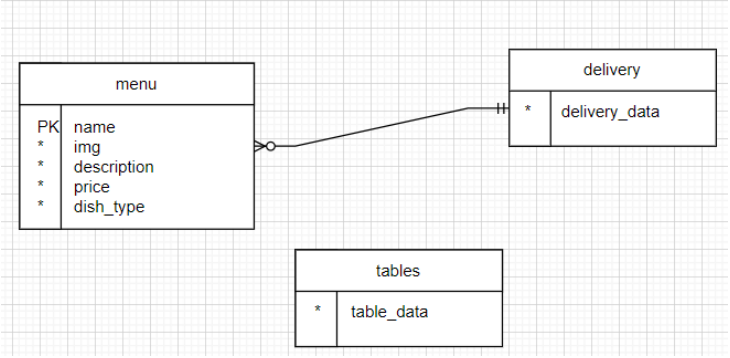

Рисунок 2.8 – База даних

3 РОЗРОБКА ЧАТ-БОТУ ПО ЗАМОВЛЕННЮ ДОСТАВКИ ЇЖІ

3.1 Архітектура програмного продукту

Весь програмний код розділений на папки : папка «DipCode» складається з папки «dataBase» яка містить файли «__init__.py» та «db_forRapid.py» які відповідають за приєднану базу даних за запити які використовуються для управління даними. Папка «handlers» складається з файлів «__init__.py», «admin.py», «client.py», які містять команди (які виконуватимуться після натискання кнопки з меню) та перевірку введених даних та функції для запису та обробки даних. Папка «keyboards» містить файли «__init__.py», «admin_kb.py», «client_kb.py», які містять програмний код для кнопок меню бота для адміністратора та клієнта відповідно. Файл «bot_telegram.py» містить хенлери, а файл «create_bot.py» містить реєстрацію бота.

Користувач під час використання чат-боту натискатиме кнопки меню, за якими в свою чергу відповідають функції для обробки інформації. Після запиту через кнопки-команди йде повідомлення для власного боту через Telegram до сервера. Сервер Telegram взаємодіє з API Telegram Bot (перелік повідомлень для власного бота) за допомогою HTTPS. Потім виконуються певні дії з даними (пошук, додавання чи видалення) в базі даних. Після отримання результату запиту з бази даних готова оброблена інформація повертається через бота та сервера (та є перевірка чи є повідомлення для власного бота), і в кінці до користувача. Цей процес відображений на рисунку 3.1

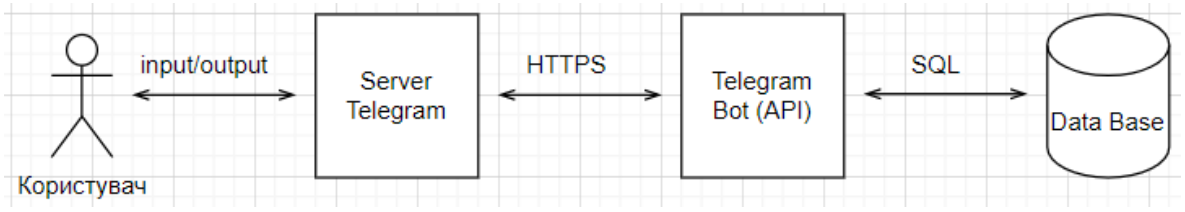

Рисунок 3.1 – Архітектура чат-бота для месенджеру Телеграм

Сервери чат-ботів працюють за допомогою Telegram Bot API, який дозволяє створювати та керувати чат-ботами на платформі обміну повідомленнями Telegram. Щоб створити бота, необхідно зареєструватися в Telegram BotFather. Telegram BotFather надасть маркер API, який буде необхідно використовувати для взаємодії з сервером.

Для роботи з базою даних використовується мова SQL та додаток DB Browser(SQLite). SQLite — бібліотека, яка надає легку дискову базу даних, яка не потребує окремого серверного процесу і дозволяє отримати доступ до бази даних за допомогою нестандартного варіанту мови запитів SQL [18].

3.2 Реалізація бази даних

Запити до бази даних написані мовою MySQL. Всі запити містяться в папці проекту, у файлі db_forRapid.py. Сама база даних – файл проекту, під назвою menu_Rapid.db. Для перегляду вмісту таблиць баз даних використано DB Browser (SQLite).

Загальний вигляд таблиць відображений на рисунку 3.2

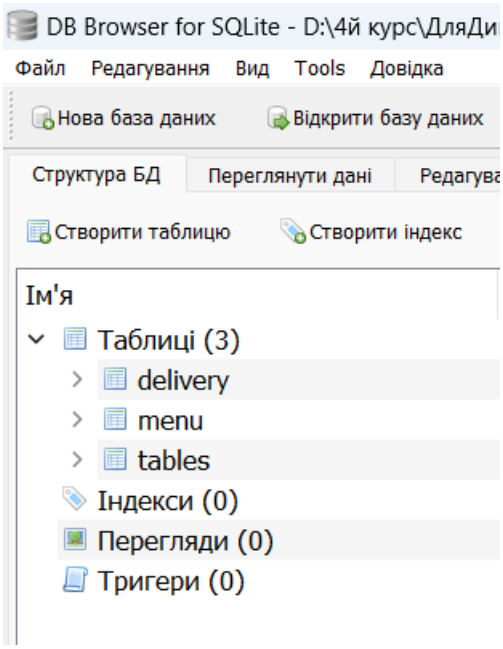

Рисунок 3.2 – Створені таблиці delivery, menu, tables

Приклад відображення даних в DB Browser з таблиці delivery відображено на рисунку 3.3

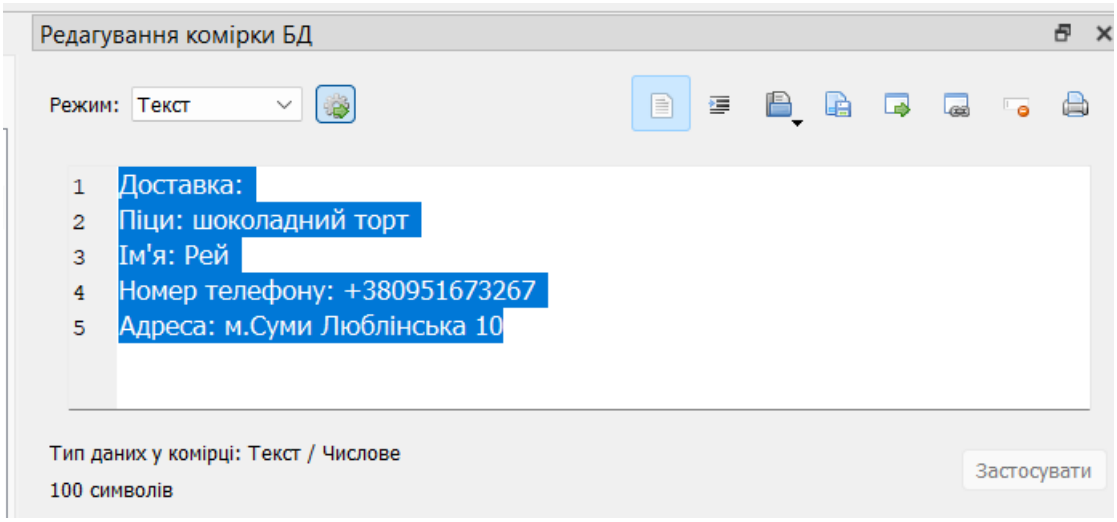

Рисунок 3.3 – Дані з таблиці delivery

Таблиця menu зі стравами відображено на рисунку 3.4.

Таблиця: menu Filter in any column

	img	name	description	price	dish_type
	Фільтр	Фільтр	Фільтр	Філ...	Фільтр
1	AgACAgIAAxkBAAIF5GSFlr6qQm76H...	Маргарита	томати, сир, базилік	110.0	звичайна
2	AgACAgIAAxkBAAIF72SFlvmrlKtCEV...	Барбекю	Прошутто, цибуля, томати, сир, ...	150.0	звичайна
3	AgACAgIAAxkBAAIF_GSFl4zHziKQiax...	М'ясна	Прошутто, бекон, перець, оливки, ...	200.0	звичайна
4	AgACAgIAAxkBAAIGB2SFl9MYimXgq...	Сирна	томати, чотири сири, базилік	130.0	звичайна
5	AgACAgIAAxkBAAIGEmSFmNlbVXBN...	Шоколадний кекс	Сирний крем, шоколадний бісквіт, ...	200.0	спеціальна
6	AgACAgIAAxkBAAIGHWSFmRP1RGJ...	Маковий	Бісквіт з полуницею,бісквіт з маком...	210.0	спеціальна
7	AgACAgIAAxkBAAIGKGSFmWfZ0YH...	Нуга	Молочний та білий шоколад, горіхи...	250.0	спеціальна
8	AgACAgIAAxkBAAIGM2SFmZ1pHJBft...	Шоколадний торт	Темний та молочний шоколад	270.0	спеціальна
9	AgACAgIAAxkBAAIHPGSG8fk_7UbKZ...	Неаполітанська	Саламі, два види солодкого перцю,...	180.0	звичайна
10	AgACAgIAAxkBAAIHdWSN3zDDJb5C...	Оливка	Оливки, томати, базилік, салямі	140.0	звичайна

Рисунок 3.4 – Таблиця menu з заповненими даними про страви

Приклад відображення даних DB Browser з таблиці tables відображено на рисунку 3.5

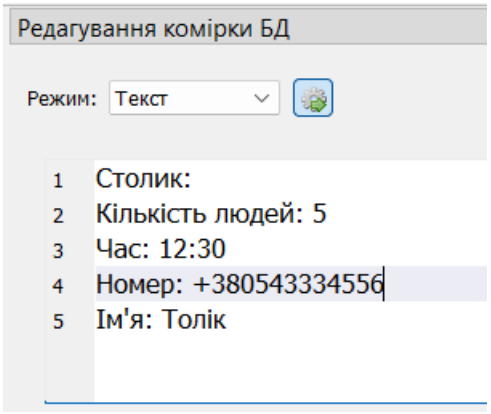

Рисунок 3.5 – Дані з таблиці tables

3.3 Програмна реалізація

Для створення чат-бота спочатку необхідно було зареєструвати Телеграм бот. Для цього необхідно знайти окремий Телеграм бот «https://t.me/BotFather», який допоможе створити власного бота. Опис дій бота зображено на рисунку 3.6.

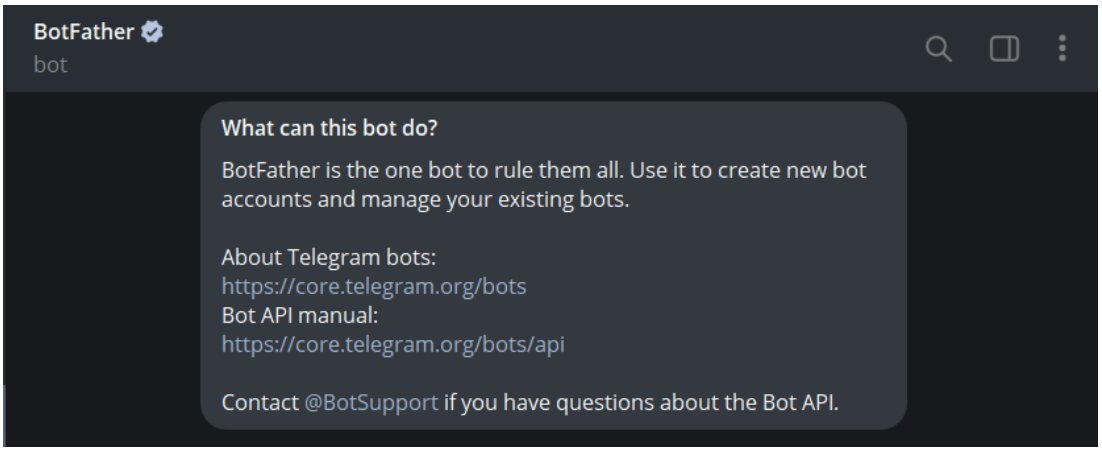

Рисунок 3.6 – Опис бота BotFather

Після запуску бота з’являється повідомлення з переліком всіх команд для управління/створення власним ботом. Далі для створення боту була використана команда «/newbot», потім необхідно було придумати ім’я бота. Після цих дій необхідно придумати юзернейм, який закінчується на «bot». Опис команд для додавання головного фото бота, його опису, перейменування і тд відображено на рисунку 3.7

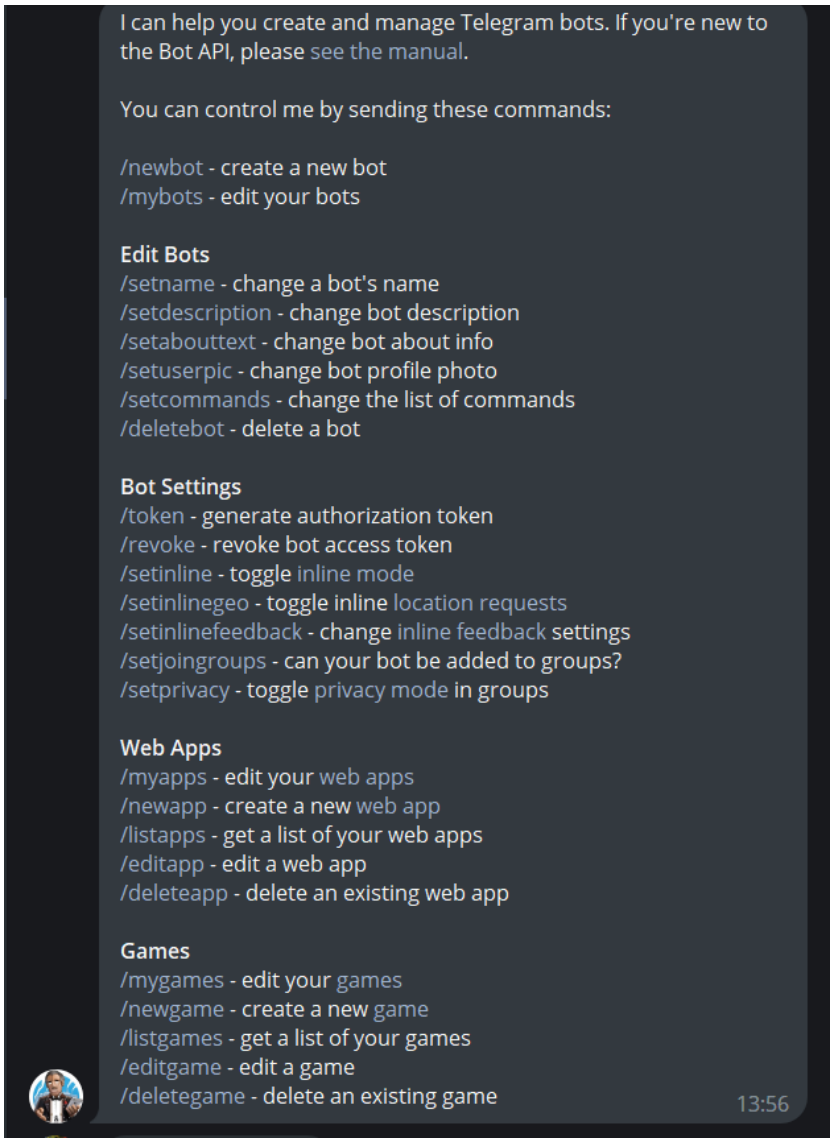

Рисунок 3.7 – Опис команд бота BotFather, які використовувалися для створення власного бота

Після введення всіх необхідних даних був сформований унікальний токен до HTTP API, рисунок 3.8

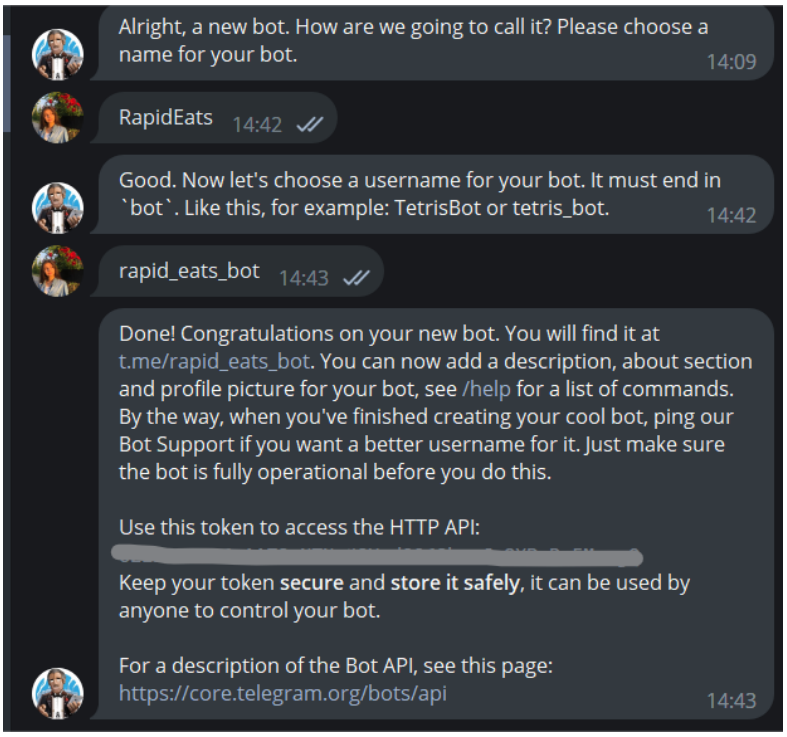

Рисунок 3.8 – Сформований токен

Під час написання коду були використані такі засоби : aiogram, sqlite3, re, string. Всі додані бібліотеки[19] описані в таблиці 3.1

Таблиця 3.1 – Засоби (класи, модулі), використані для написання коду

aiogram	Bot	Bot – це клас з модуля aiogram, який представляє екземпляр бота. Ви можете створити новий об'єкт Bot, імпортувавши клас Bot з модуля aiogram і надавши йому токен API, який ви можете отримати з @BotFather
	Dispatcher	Dispatcher - це клас з модуля aiogram, який забезпечує простий спосіб обробки вхідних оновлень, таких як повідомлення, відредаговані повідомлення, повідомлення каналів, відредаговані повідомлення каналу, вбудовані запити, вибрані вбудовані результати, запити зворотного виклику, запити на доставку та запити перед оформленням замовлення 1. Створити новий об'єкт Dispatcher можна, імпортувавши клас Dispatcher з модуля aiogram та надавши йому екземпляр Bot

Продовження таблиці 3.1

	MemoryStorage	MemoryStorage – клас з модуля aiogram.contrib.fsm_storage.memory, що забезпечує сховище станів на основі in-memory. Цей тип сховища не рекомендується використовувати в ботах, оскільки втрачаються стани після перезапуску(але так як у мене створена база даних, всі дані зберігаються туди). Створюється новий об'єкт MemoryStorage, імпортувавши клас MemoryStorage з модуля aiogram.contrib.fsm_storage.memory і надається його екземпляру Dispatcher
	Executor	Executor — клас з модуля aiogram.utils.executor, який надає можливість розпочати встановити довготривалий полінг (послійний зв’язок із сервером) для екземпляра Dispatcher. Створити новий об'єкт Executor можна, імпортувавши клас Executor з модуля aiogram.utils.executor та надавши йому екземпляр Dispatcher
	ReplyKeyboardMarkup	ReplyKeyboardMarkup - клас з модуля aiogram.types.reply_keyboard, який представляє користувацьку клавіатуру з параметрами відповіді. Можна створити новий об'єкт ReplyKeyboardMarkup, імпортувавши клас ReplyKeyboardMarkup з модуля aiogram.types.reply_keyboard та надати йому масив рядків кнопок, кожен з яких представлений масивом об'єктів KeyboardButton
	KeyboardButton	BotsKeyboardButton — клас модуля aiogram.types, що представляє одну кнопку клавіатури.
	InlineKeyboardMarkup	InlineKeyboardMarkup — це клас модуля aiogram.types, який представляє вбудовану клавіатуру, яка з'являється поруч із повідомленням, до якого вона належить. Створити об'єкт InlineKeyboardMarkup можна, викликавши його конструктор з потрібними параметрами.

Продовження таблиці 3.1

	InlineKeyboardButton	InlineKeyboardButton — клас модуля aiogram.types, що представляє одну кнопку вбудованої клавіатури. Створити об'єкт InlineKeyboardButton можна, викликавши його конструктор з потрібними параметрами. Наприклад, створити об'єкт InlineKeyboardButton за допомогою button = InlineKeyboardButton('Текст кнопки', callback_data='my_callback_data').
	FSMContext	FSMContext - це клас в модулі aiogram.dispatcher, який представляє скінченний машинний контекст (контекст – коли машина або комп’ютер може розуміти текст або мову, яку вона обробляє). Він надає інтерфейс для роботи з поточним станом і пов'язаними з ним даними. Можна використовувати метод set_state для встановлення поточного стану, метод get_state для отримання поточного стану та метод завершення для скидання стану.
	Types	Types — модуль у пакеті aiogram, що містить класи для роботи з різними типами об'єктів Telegram
	State	State - клас, імпортований з модуля aiogram.dispatcher.filters.state. Використовується для представлення стану в скінченному автоматі (FSM) при побудові бота.
	StatesGroup	StatesGroup - клас, імпортований з модуля aiogram.dispatcher.filters.state. Він використовується для групування пов'язаних станів разом при побудові бота. Група станів може містити декілька об'єктів станів, кожен з яких представляє стан у скінченному автоматі (FSM).
	Text	Text – клас фільтра, імпортований з модуля aiogram.dispatcher.filters. З його допомогою можна перевірити, чи відповідає текст повідомлення певним умовам. Наприклад, за допомогою Text можна відповідати лише на повідомлення, які містять певне слово або фразу.

Продовження таблиці 3.1

sqlite3		Sqlite3 — модуль на мові Python, що надає інтерфейс DB-API для баз даних SQLite. SQLite — бібліотека, яка надає легку дискову базу даних, яка не потребує окремого серверного процесу і дозволяє отримати доступ до бази даних за допомогою нестандартного варіанту мови запитів SQL[18].
re		re — модуль в Python, що забезпечує підтримку регулярних виразів[19]. Регулярні вирази є потужним інструментом для зіставлення шаблонів у тексті. Він надає функції для пошуку шаблонів у рядках, розділення рядків на основі шаблонів та виконання замін на рядки[20].
string		Клас, який дозволяє використовувати різні методи string для маніпулювання рядками, такі як split(), join(), replace() та багато інших, також метод format() для форматування рядків[20].

3.4 Використання програмного додатку

Перед запуском бота на головному полі прикріплене повідомлення з описом чат-бота для месенджеру Телеграм по замовленню доставки їжі, це відображено на рисунку 3.9.

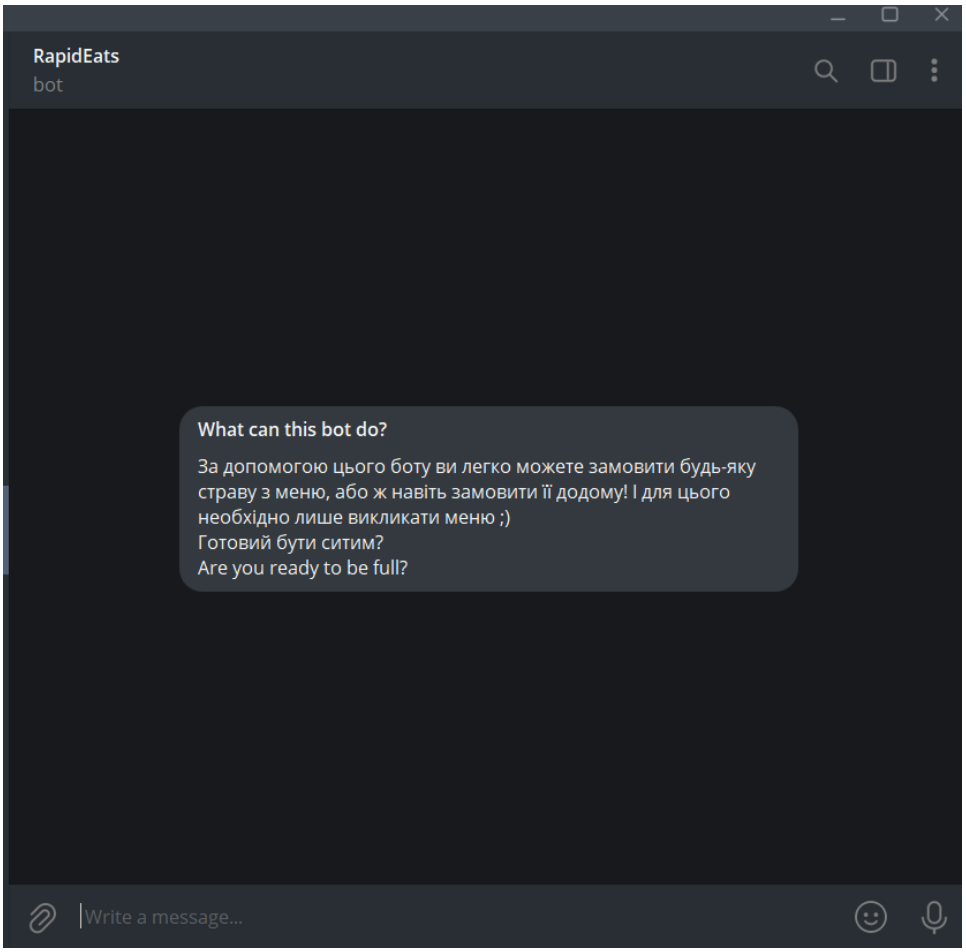

Рисунок 3.9– Опис чат-бота

Права розділені на дві групи: звичайний користувач та адміністратор. У функціях звичайного користувача є: перегляд меню звичайних страв, перегляд меню спеціальних страв, можливість замовити столик, оформити доставку та отримати контакти з працівниками закладу для зворотного зв’язку. У функціях авторизованого користувача(адміністратора) є : завантаження нового пункту меню (спеціальної чи звичайної страви), видалення пункту меню, перегляд замовлень на столики та опрацювання замовлень, перегляд замовлень на доставку та опрацювання замовлень.

Чат-бот є доступним для всіх користувачів месенджеру Telegram, але права адміністратора надаються не кожному користувачеві. Є створена спеціальна група в Telegram під назвою «Rapid_Eats», в яку додані лише обрані користувачі – працівники закладу, яка відображена на рисунку 3.10.

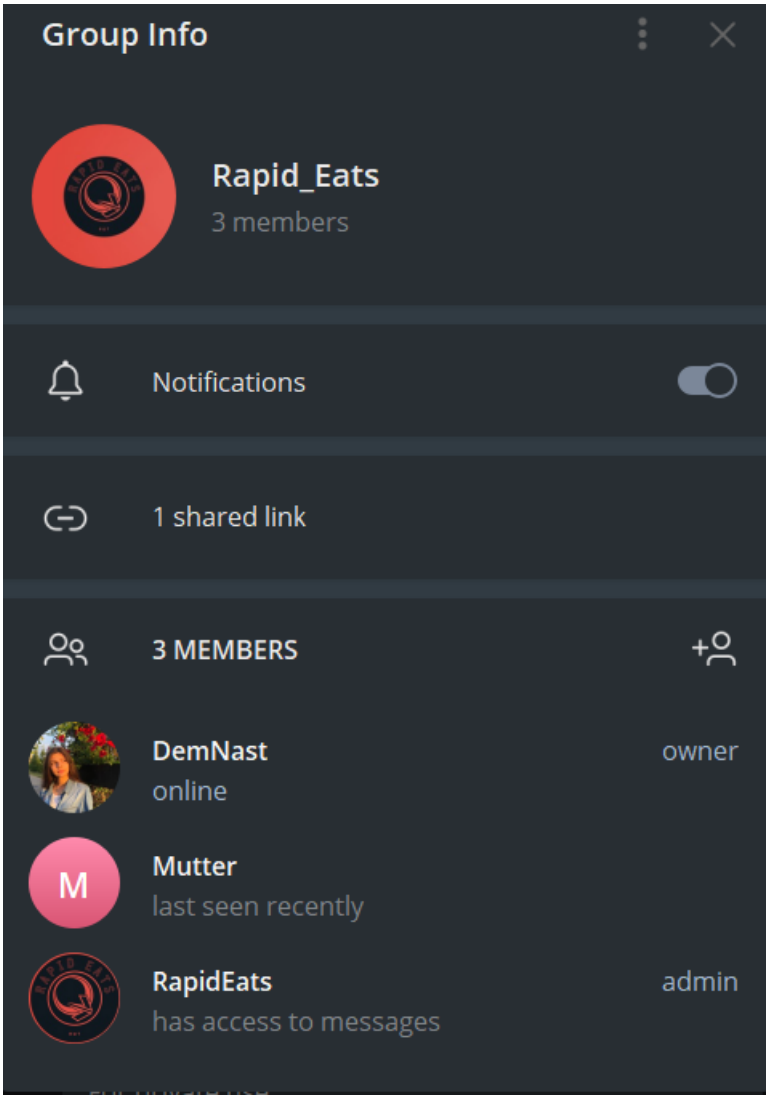

Рисунок 3.10 – Група працівників закладу, необхідна для авторизації адмінів

Головний адміністратор має право додавати нових користувачів до групи та надавати права адміністратора іншим. Для використання бота адміністратором вводиться команда «/admin» в групі «Rapid_Eats», після чого, сам бот надсилає повідомлення про готовність до роботи в приватні повідомлення з авторизованим користувачем, це відображено на рисунку 3.11

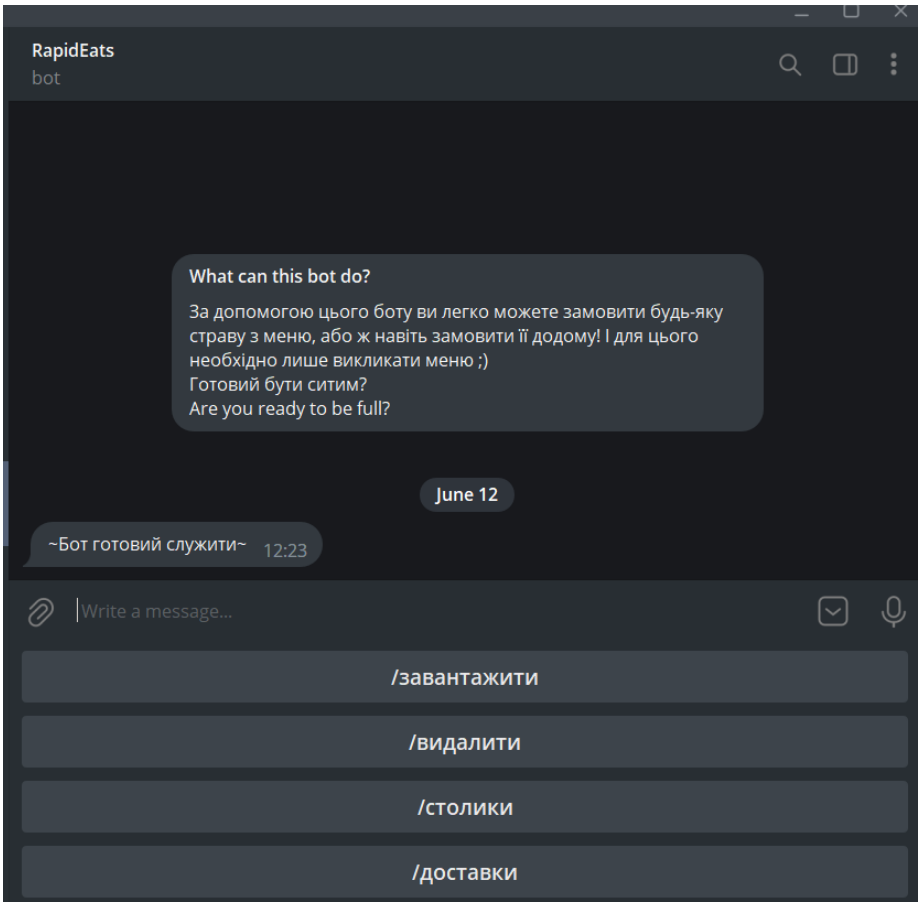

Рисунок 3.11 – Меню для авторизованого користувача (адміністратора)

Серед кнопок меню : завантажити, видалити, столики та доставки.

Після натискання кнопки «завантажити» з’являється повідомлення-прохання завантажити фото для нового пункту меню, рисунок 3.12

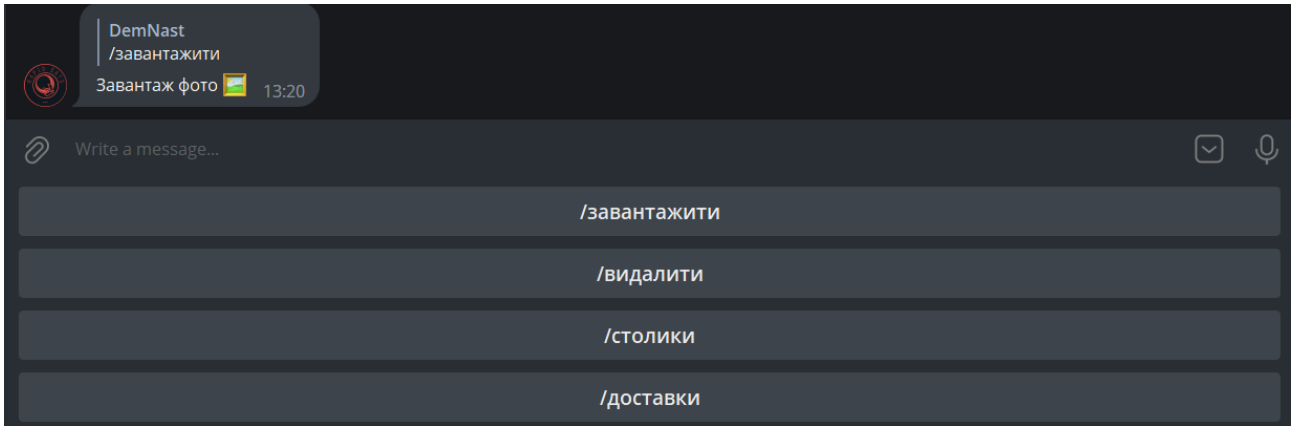

Рисунок 3.12 – Перший етап створення нового пункту меню – додавання фото адміністратором

Після завантаження фотографії, з’являється нове повідомлення-прохання про введення назви страви, рисунок 3.13

Рисунок 3.13 – Другий етап створення нового пункту меню – введення назви адміністратором

Наступними кроками у створенні нового елементу меню є повідомлення прохання про введення опису страви, її опису, ціни, та типу (звичайна чи спеціальна), рисунок 3.14.

Рисунок 3.14 – Наступний етап створення нового пункту меню – введення назви страви, опису, ціни та типу страви адміністратором

Після створення нової страви в меню, вона додається також у базу даних, конкретніше у таблицю – меню зі стравами, у даному випадку це пункт 9 на рисунку 3.15

	img	name	description	price	dish_type
	Фильтр	Фильтр	Фильтр	Фи...	Фильтр
1	AgACAgIAAxkBAAIF5GSFlr6qQm76H...	Маргарита	томати, сир, базилік	110.0	звичайна
2	AgACAgIAAxkBAAIF72SFlvmrlKtCEV...	Барбекю	Прошутто, цибуля, томати, сир, ...	150.0	звичайна
3	AgACAgIAAxkBAAIF_GSFl4zHziKQiax...	М'ясна	Прошутто, бекон, перець, оливки, ...	200.0	звичайна
4	AgACAgIAAxkBAAIGB2SFl9MYimXgq...	Сирна	томати, чотири сири, базилік	130.0	звичайна
5	AgACAgIAAxkBAAIGEmSFmNlbVXBN...	Шоколадний кекс	Сирний крем, шоколадний бісквіт, ...	200.0	спеціальна
6	AgACAgIAAxkBAAIGHWSFmRP1RGJ...	Маковий	Бісквіт з полуницею,бісквіт з маком...	210.0	спеціальна
7	AgACAgIAAxkBAAIGKGSFmWfZ0YH...	Нуга	Молочний та білий шоколад, горіхи...	250.0	спеціальна
8	AgACAgIAAxkBAAIGM2SFmZ1pHJBft...	Шоколадний торт	Темний та молочний шоколад	270.0	спеціальна
9	AgACAgIAAxkBAAIHPGSG8fk_7UbKZ...	Неаполітанська	Саламі, два види солодкого перцю,...	180.0	звичайна

Рисунок 3.15 – Нова страва з меню додана до таблиці бази даних

Після натискання кнопки «видалення» відображуються всі страви, і звичайне меню, і страви зі спеціального меню, та з’являється інлайн кнопки під кожним пунктом меню для зручного видалення, це відображено на рисунку 3.16.

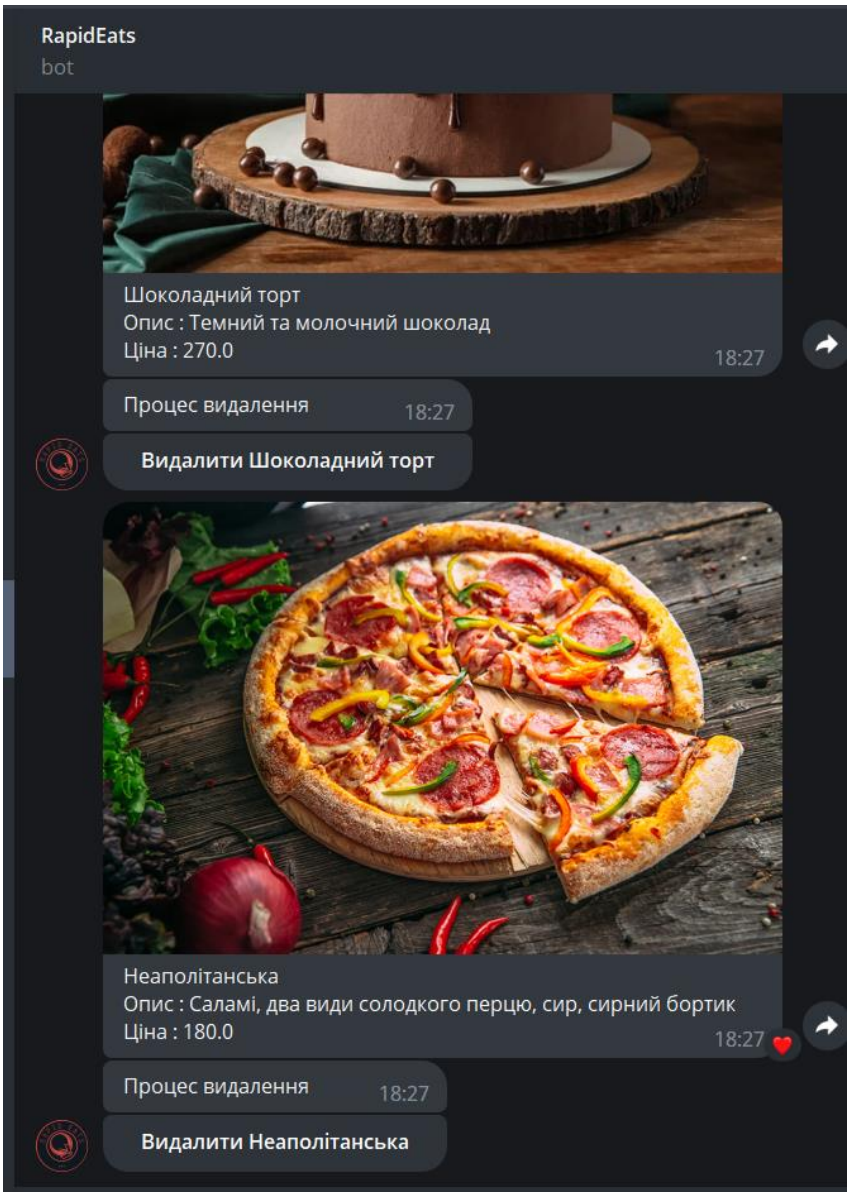

Рисунок 3.16 – Видалення страв з меню адміністратором

Після натискання адміністратором кнопки «столики» відображуються всі замовлення на столики з інлайн кнопкою для опрацювання замовлень. Після опрацювання замовлення воно видаляється з бази даних, рисунок 3.17

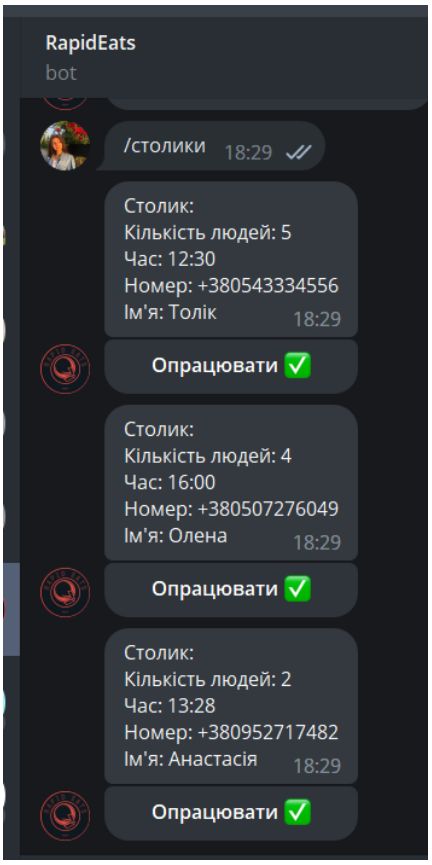

Рисунок 3.17 – Відображення замовлень на столики

Після натискання адміністратором кнопки «доставки» відображуються всі замовлення на доставку страв з інлайн кнопкою для опрацювання замовлень. Після опрацювання замовлення воно видаляється з бази даних, рисунок 3.18.

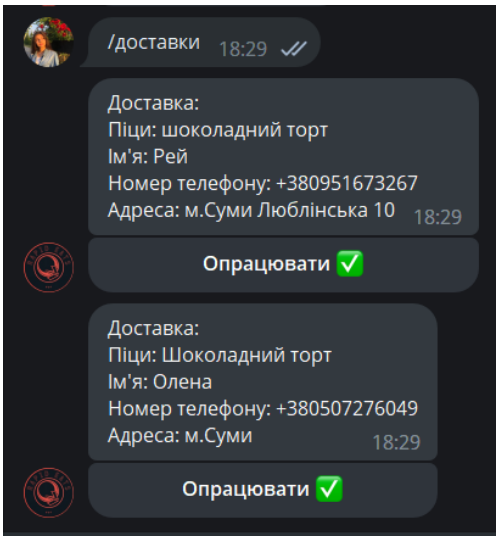

Рисунок 3.18 – Відображення доставок в режимі адміністратора

Для використання бота звичайним користувачем вводиться команда «/start» в групі «Rapid_Eats», або кнопка «Старт», після чого, сам бот надсилає повідомлення про готовність до роботи та меню з кнопками , це відображено на рисунку 3.19.

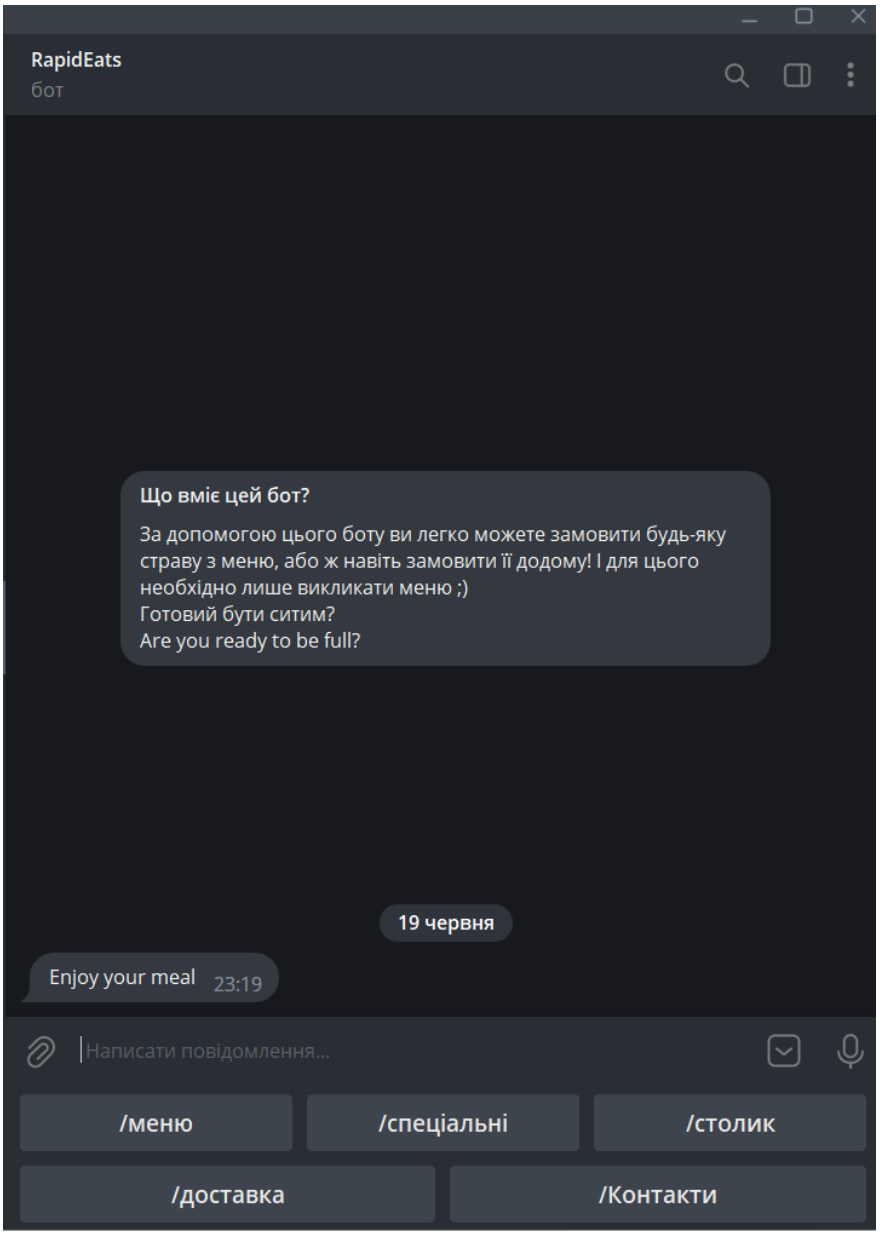

Рисунок 3.19 – Вигляд бота після натискання кнопки «Старт»

Кнопка «меню» представляє собою функцію перегляду меню звичайних страв, це відображено на рисунку 3.20

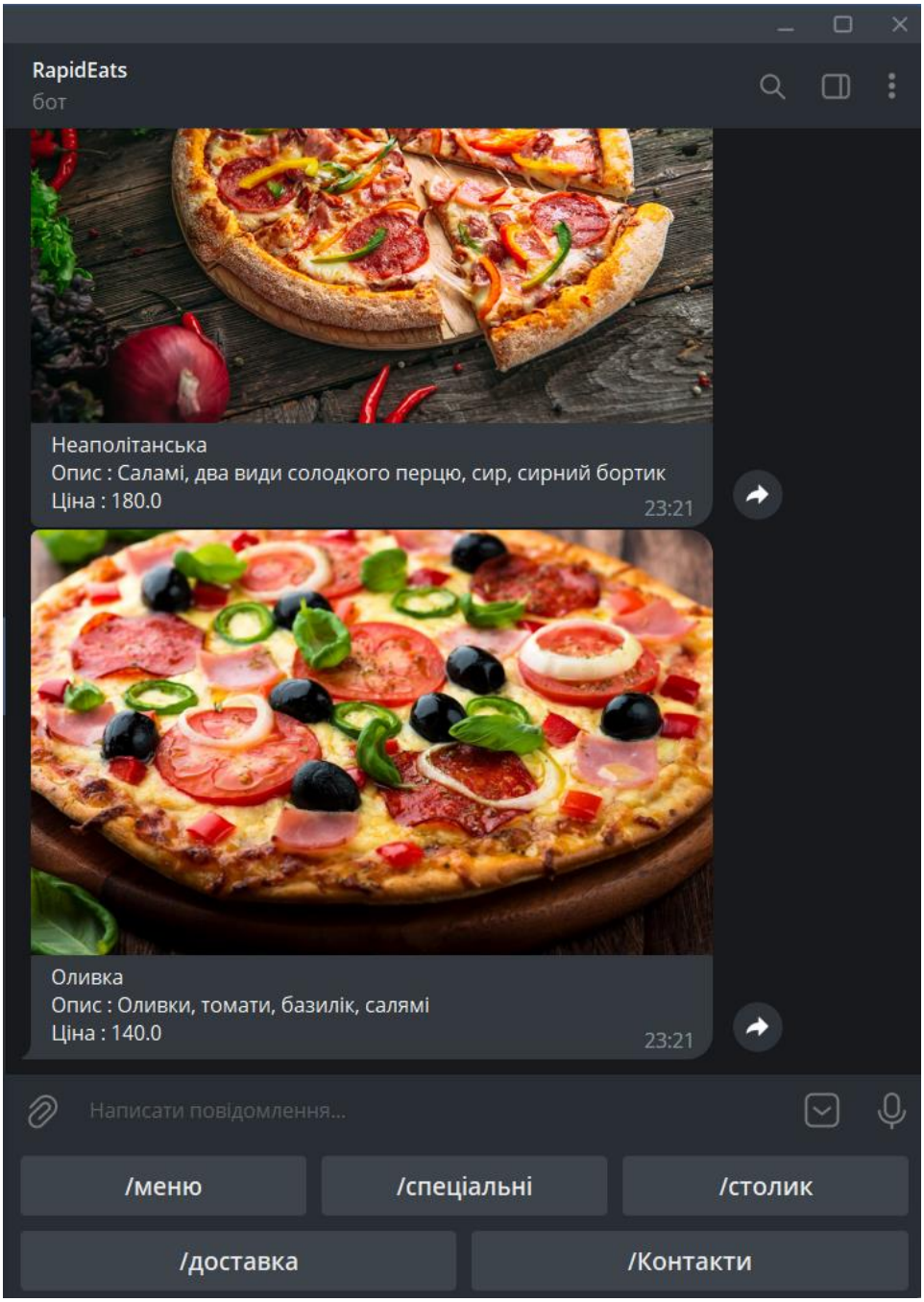

Рисунок 3.20 – Відображення меню звичайних страв

Кнопка «спеціальні» представляє собою функцію перегляду меню спеціальних страв, це відображено на рисунку 3.21.

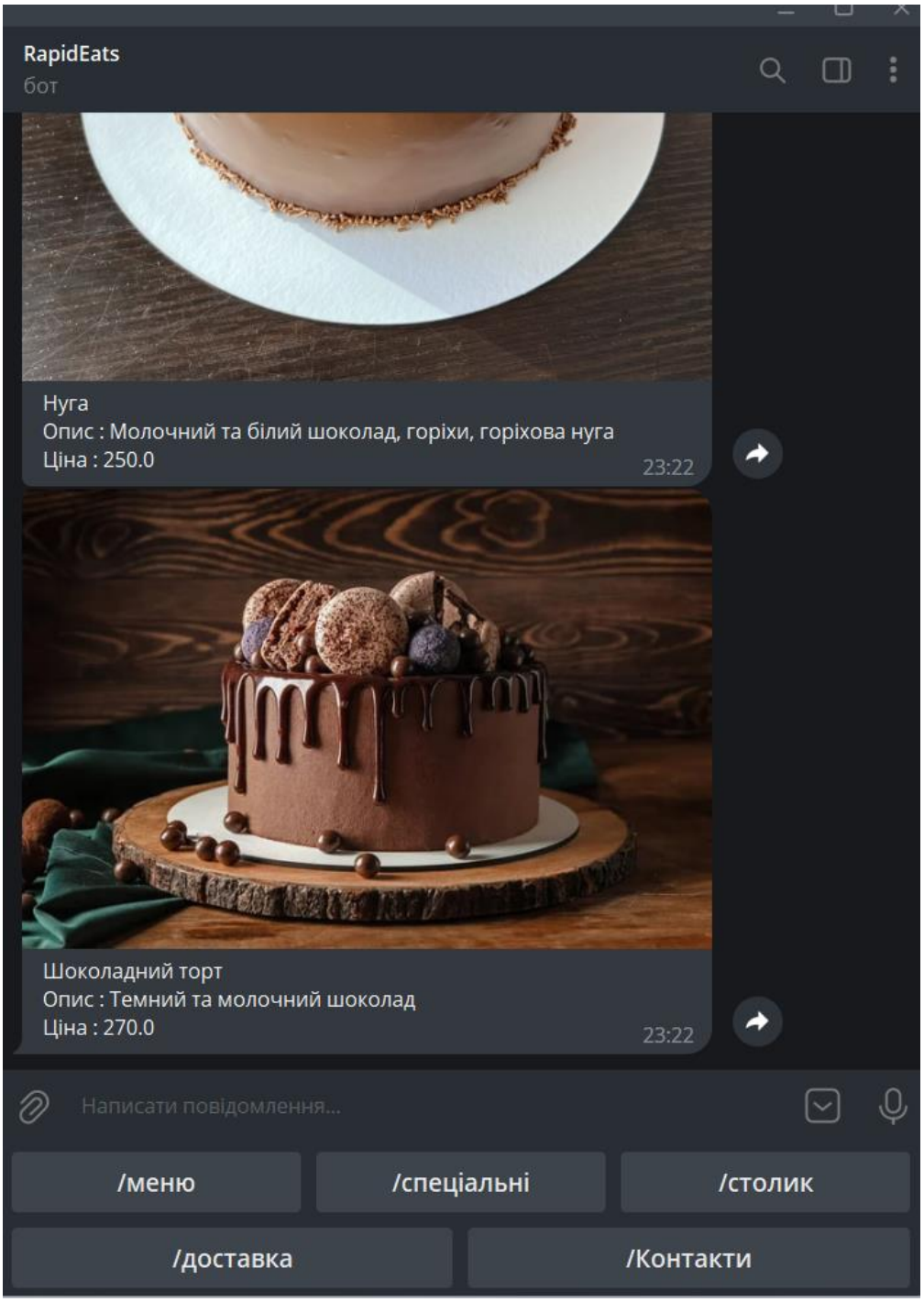

Рисунок 3.21 – Відображення меню спеціальних страв

Через кнопку «столик» користувач має можливість замовити столик, у результаті його дані будуть записані до бази даних, за допомогою чого адміністратор зможе зателефонувати відвідувачу та уточнити інформацію про бронювання, рисунок 3.22

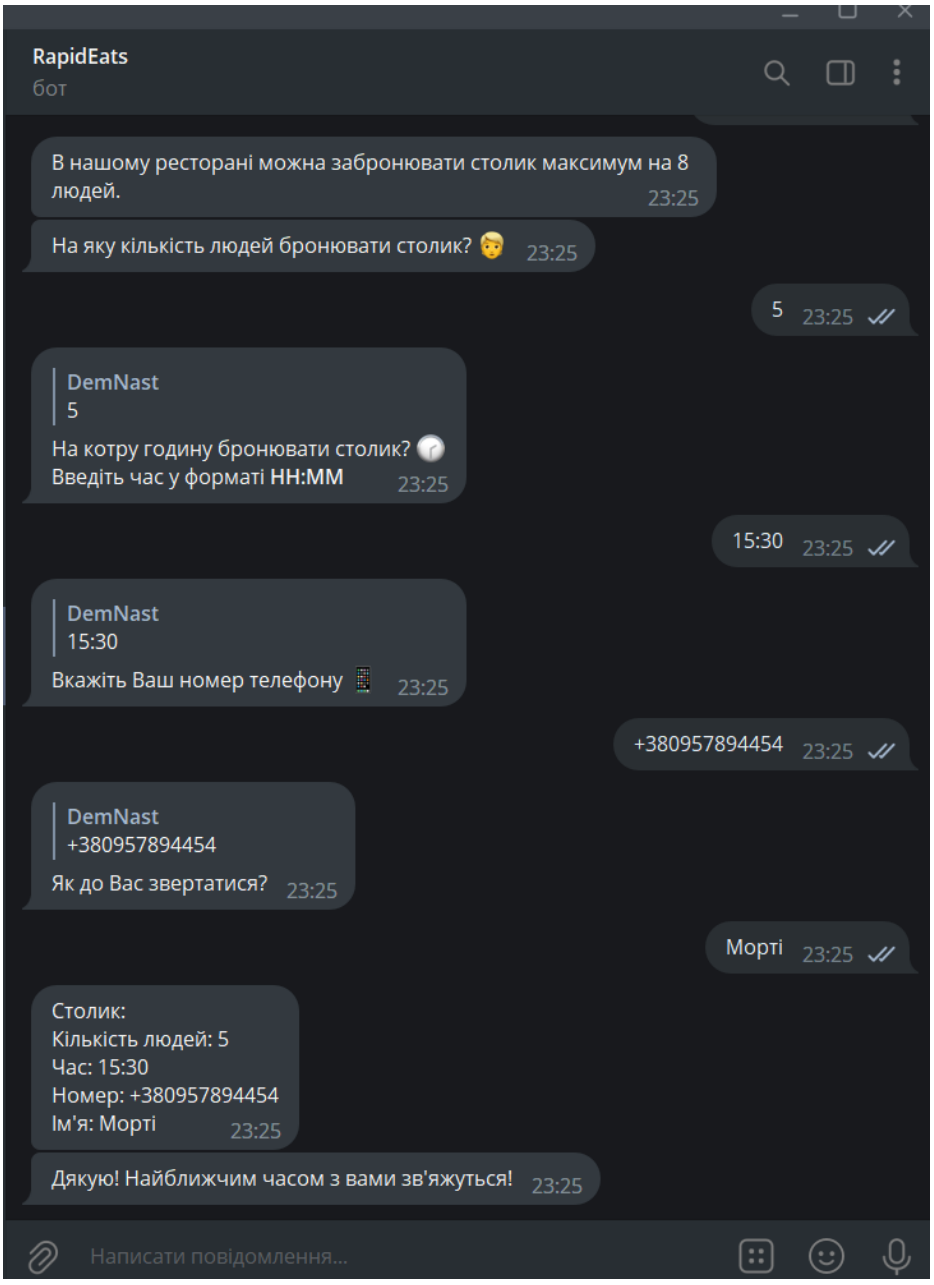

Рисунок 3.22 – Бронювання столика клієнтом

Кнопка «доставка» відповідає за замовлення доставки клієнтом. Після натискання будуть по черзі з’являтися повідомлення з вказівками для подальших дій користувача : вказати страви для замовлення, вказати ім’я, вказати номер телефону, вказати адресу. В результаті користувач отримає повідомлення з введеними даними для доставки та вказівкою чекати зворотного зв’язку для прийняття замовлення, це відображено на рисунку 3.23.

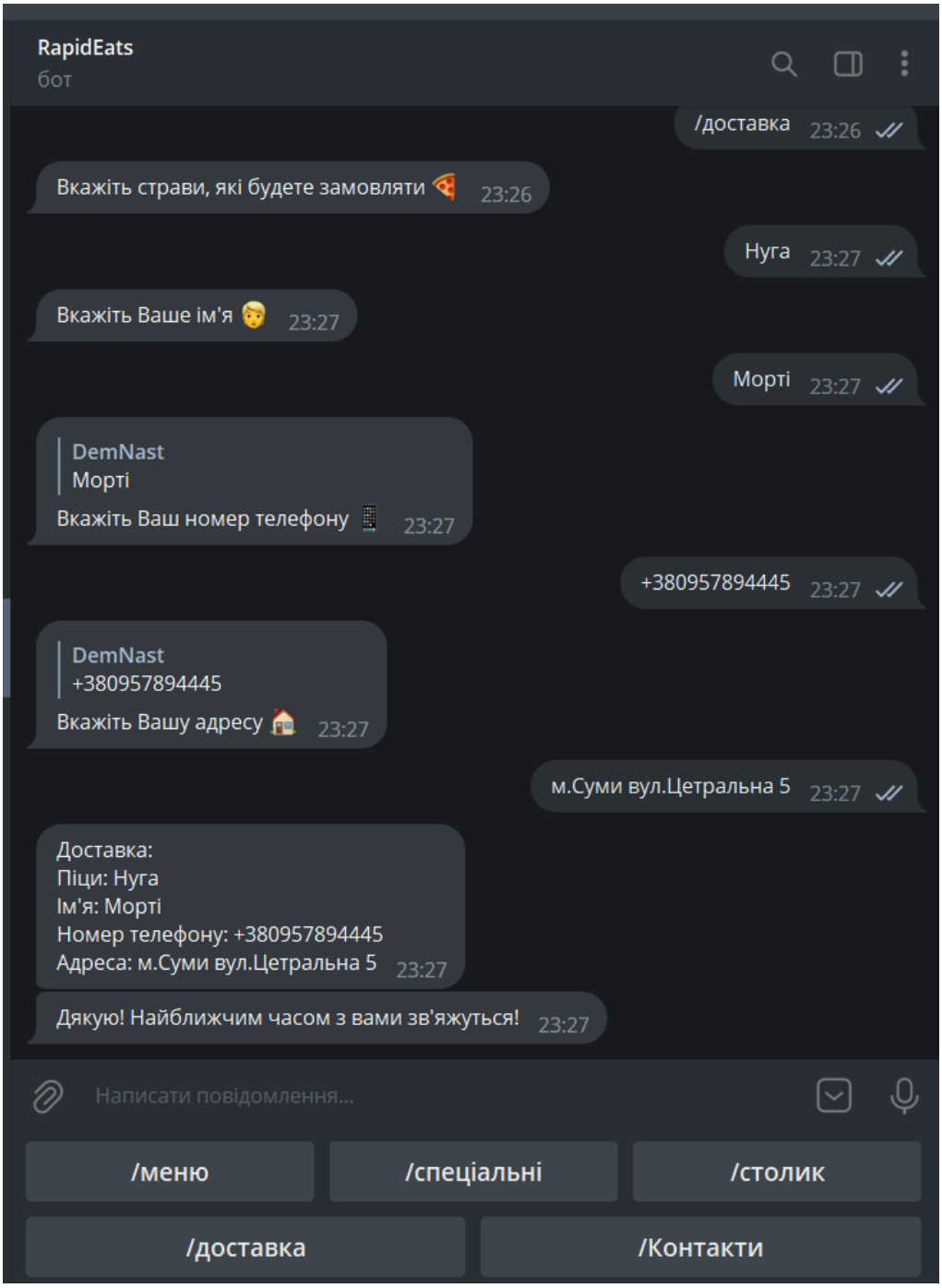

Рисунок 3.23 – Замовлення доставки клієнтом

Після натискання кнопки «контакти» є надіслане ботом повідомлення з варіантами зв’язку з закладом, рисунок 3.24.

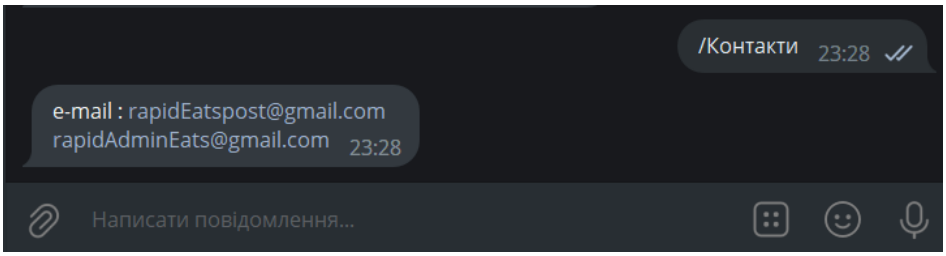

Рисунок 3.24 – Використання кнопки контакти

ВИСНОВКИ

На даний момент у розвитку інформаційних технологій помітний значний прогрес у всіх сферах діяльності людини. Прогрес не обійшов і сферу харчування. Відтепер можна з легкістю отримати вже готові страви, лише використовуючи чат-бот в Телеграмі. Особливо приємно те, що страви можна замовити з телефону не виходячи з дому, та у разі виникнення помилки можна зв’язатися з менеджером. Однак, при використанні створеного чат-бота оплата за покупку здійснюється тільки готівкою.

У ході виконання роботи були сформульовані мета і задачі проекту, також проведено аналіз аналогів проекту, визначені переваги та недоліки програмних продуктів. Були сформульовані функціональні вимоги: можливість перегляду звичайного меню та меню спеціальних страв для звичайного клієнта, та можливість редагування меню (додавання та видалення страв) для адміністратора, бронювання столика для клієнта та можливість перегляду вже замовлених столиків для адміністратора, замовити доставку, та дізнатися контакти для зворотного зв’язку у разі непорозуміння.

Було проведено моделювання контекстної діаграми в нотації IDEF0 та її декомпозицію на першому рівні, створено діаграми варіантів використання, діаграму послідовності та діаграму діяльності. За складеними моделями та технічним завданням виконано програмну реалізацію.

Результатом роботи над бакалаврською роботою є чат-бот для месенджеру Телеграм по замовленню доставки їжі. Результати даної роботи представлені у вигляді тез для доповіді на конференції «ІМА – 2023» [4], які можна переглянути у додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Manisha S. Story of ELIZA, the first chatbot developed in 1966 [Електронний ресурс] / Salecha Manisha – Режим доступу до ресурсу: https://analyticsindiamag.com/story-eliza-first-chatbot-developed-1966.

2. The History Of Chatbots – From ELIZA to ChatGPT [Електронний ресурс] // ONLIM. – 2022. – Режим доступу до ресурсу: https://onlim.com/en/the-history-of-chatbots/ .

3. Prissadang Suta. An Overview of Machine Learning in Chatbots [Електронний ресурс] / Prissadang Suta, Xi Lan, Biting Wu // International Journal of Mechanical Engineering and Robotics Research. – 2020. – Режим доступу до ресурсу: http://www.ijmerr.com/uploadfile/2020/0312/20200312023706525.pdf.

4. Деменко А.М., Ващенко С.М. Telegram чат- бот для мережі швидкого харчування. // «Інформатика, математика, автоматика»: матеріали та програма науково-технічної конференції, м. Суми, 24 – 28 квітня 2023 р. – Суми: Сумський державний університет, 2023. – С. 161.

5. 56 FACTS IN COMPARISON Facebook Messenger vs Telegram Messenger [Електронний ресурс] // versus. – 2022. – Режим доступу до ресурсу: https://versus.com/en/facebook-messenger-vs-telegram-messenger.

6. ВзаємоДія. Офіційні українські боти Telegram [Електронний ресурс] / ВзаємоДія – Режим доступу до ресурсу: https://viyna.net/ukrayinski-telegram-kanali-2.

7. Національна мережа продуктових магазинів "АТБ-Маркет". АТБ Телеграм Бот [Електронний ресурс] / Національна мережа продуктових магазинів "АТБ-Маркет" – Режим доступу до ресурсу: https://t.me/atb_market_bot.

8. Privat Bank. PrivatBank Bot [Електронний ресурс] / Privat Bank – Режим доступу до ресурсу: https://t.me/PrivatBank_help_bot.

9. Олеся Мельніченко. Що таке чат-бот: секрети використання та основні переваги для бізнесу [Електронний ресурс] / Олеся Мельніченко // HELPCRUNCH. – 2022. – Режим доступу до ресурсу: https://helpcrunch.com/blog/uk/shcho-take-chat-bot/.

10. State of ChatBots The 2018 Report [Електронний ресурс] / Drift, Audience, Salesforce та ін.]. – 2018. – Режим доступу до ресурсу: https://www.drift.com/wp-content/uploads/2018/01/2018-state-of-chatbots-report.pdf.

11. М'ясторія. Чат-бот Telegram [Електронний ресурс] / М'ясторія – Режим доступу до ресурсу: https://t.me/myastoriya_chatbot.

12. MR.POPS. TRUE ICE BOT [Електронний ресурс] / MR.POPS – Режим доступу до ресурсу: https://t.me/mr_pops_bot.

13. Oracle. MySQL [Електронний ресурс] / Oracle. – 2023. – Режим доступу до ресурсу: https://www.mysql.com/.

14. Python [Електронний ресурс] // 3.11. – 2022. – Режим доступу до ресурсу: https://www.python.org/.

15. JetBrains. PyCharm [Електронний ресурс] / JetBrains // The Python IDE for Professional Developers. – 2023. – Режим доступу до ресурсу: https://www.jetbrains.com/pycharm/.

16. Flathub. DB Browser for SQLite [Електронний ресурс] / Flathub. – 2021. – Режим доступу до ресурсу: https://flathub.org/de/apps/org.sqlitebrowser.sqlitebrowser.

17. stud.com.ua. МЕТОДОЛОГІЯ IDEF0 [Електронний ресурс] / stud.com.ua. – 2023. – Режим доступу до ресурсу: https://stud.com.ua/87184/ekonomika/metodologiya_idef0.

18. Каграманова Ю. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти [Електронний ресурс] / Юлія Каграманова // DOU. – 2022. – Режим доступу до ресурсу: https://dou.ua/forums/topic/40575/.

19. aiogram. Quick start [Електронний ресурс] / aiogram // AsyncIOteleGRAM. – 2023. – Режим доступу до ресурсу: https://docs.aiogram.dev/en/latest/quick_start.html

20. sqlite3 — DB-API 2.0 interface for SQLite databases [Електронний ресурс] // Python. – 2023. – Режим доступу до ресурсу: https://docs.python.org/3/library/sqlite3.html.

21. Python Regular Expressions [Електронний ресурс] // Google for Education. – 2023. – Режим доступу до ресурсу: https://developers.google.com/edu/python/regular-expressions?hl=en.

22. Python Documentation contents [Електронний ресурс] // Python. – 2023. – Режим доступу до ресурсу: https://docs.python.org/3/contents.html.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

Чат-боту для месенджеру Телеграм по замовленню доставки їжі

1 Призначення й мета створення чат-боту

1.1 Призначення чат-боту

Головними перевагами є не лише доступність 24/7 для користувача, а ще й швидка відповідь, постійна доступність необхідної інформації, легке використання, замовлення їжі віддалено, не виходячи з дому.

1.2 Мета створення

Головна мета проекту – розробка чат-боту для месенджеру Телеграм по замовленню доставки їжі.

1.3 Цільова аудиторія

Цільовою аудиторією даного проекту є робітники закладу, які зацікавлені у полегшеному процесі обслуговування, та люди які хочуть скористатися послугами закладу харчування – звичайні користувачі.

2 Вимоги до чат-боту

2.1 Вимоги в цілому

2.1.1 Вимоги до структури й функціонування чат-бота

Під час формування функціоналу програмного продукту було сформовано такі функціональні вимоги : доступний опис програмного продукту (що робить цей бот), зрозуміла навігація, читабельний шрифт (кнопки меню), зрозуміла послідовність дій.

Кінцевий продукт має бути представлений чат-ботом для месенджеру Телеграм по замовленню доставки їжі.

2.1.2 Вимоги до персоналу

Додаткові технічні навички необхідні для адміністраторів та базові навички з роботою в Телеграмі, бо необхідно вміти управляти групою, додавати нових користувачів, розмежовувати права користувачів. Для звичайних користувачів боту поглибленні знання не є необхідними, і якщо виникнуть запитання, то у боті представлені контакти для зворотнього зв’язку.

2.1.3 Вимоги до збереження інформації

Уся необхідна інформація зберігається до бази даних, яка приєднана до бота. Для роботи з базою була використана система управління DB Browser SQLite.

2.1.4 Вимоги до розмежування доступу

Права чат-бота розмежовані на дві групи : користувач, адміністратор.

Для звичайного користувача бот доступний за посиланням, для входу з правами адміністратора необхідно бути доданим в приватну групу та для запуску використати команду /admin. Крім того, що робітник закладу повинен бути доданим в групу, йому повинні бути надані права адміністратора групи, так як сам чат бот реагуватиме на надані ролі користувачів.

2.2 Структура чат-боту

2.1.1 Загальна інформація про структуру продукту

В шапці чат-бота відображається його ім’я та вказаний тип «бот».

Головне (робоче поле) – велика область, перед запуском в ньому міститься опис чат – бота. Після повторного запуску відображені попередньо відправлені повідомлення та команди. Можуть відображатись : страви головного меню, спеціальні страви, інформація про доставку, столики, чи контакти.

В нижній частині – кнопка «старт» при першому запуску, після повторного використання – меню з кнопками.

2.1.2 Навігація

В нижній частині головного поля знаходиться меню, яке навіть при виконанні певних функцій завжди доступне. Якщо користувачу під час роботи з ботом заважатиме меню, наявна кнопка для його згортання. При натисканні іншої кнопки меню, виконуватимуться функції натиснутої кнопки меню. Для звичайного користувача при оформленні доставки чи запиту на бронювання столика замість всіх кнопок головного меню підсвічується кнопка відміни.

2.1.3 Контент

У чат-боті здебільшого використовуються дані текстового формату : назви страв, опис страв, ім’я користувача, номер телефону(зберігається в БД як текст) та інші. Але також містяться картинки – фотографії страв, які оброблюються Телеграмом.

2.1.4 Дизайн та структура бота

Дизайн чат-бота обумовлений самому Телеграмом. Для розробника є можливість розташувати кнопки меню та оформити надписи на кнопках. Змінити оформлення, задній фон наприклад, користувач самостійно може зробити через налаштування самого Telegram. Загальна структура бота зображені на рисунку А.1-А.6

Навігація повинна бути зрозумілою для користувача ботом і зрозумілою для користувачів без гарного знання комп’ютерних технологій.

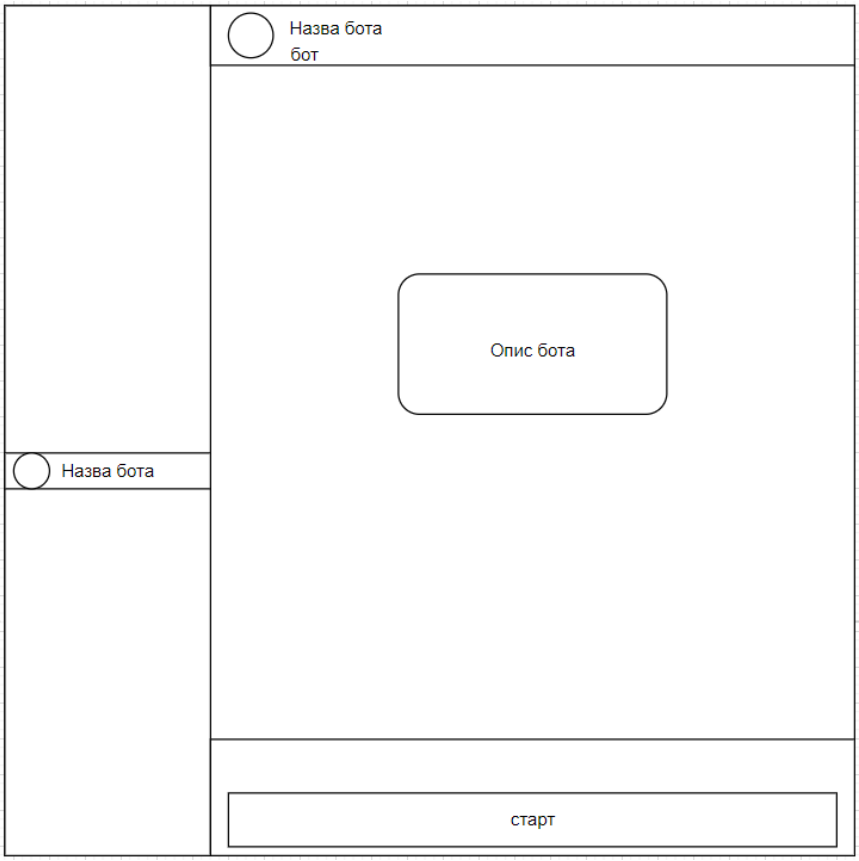

Рисунок А.1 – Дизайн бота, вигляд до запуску

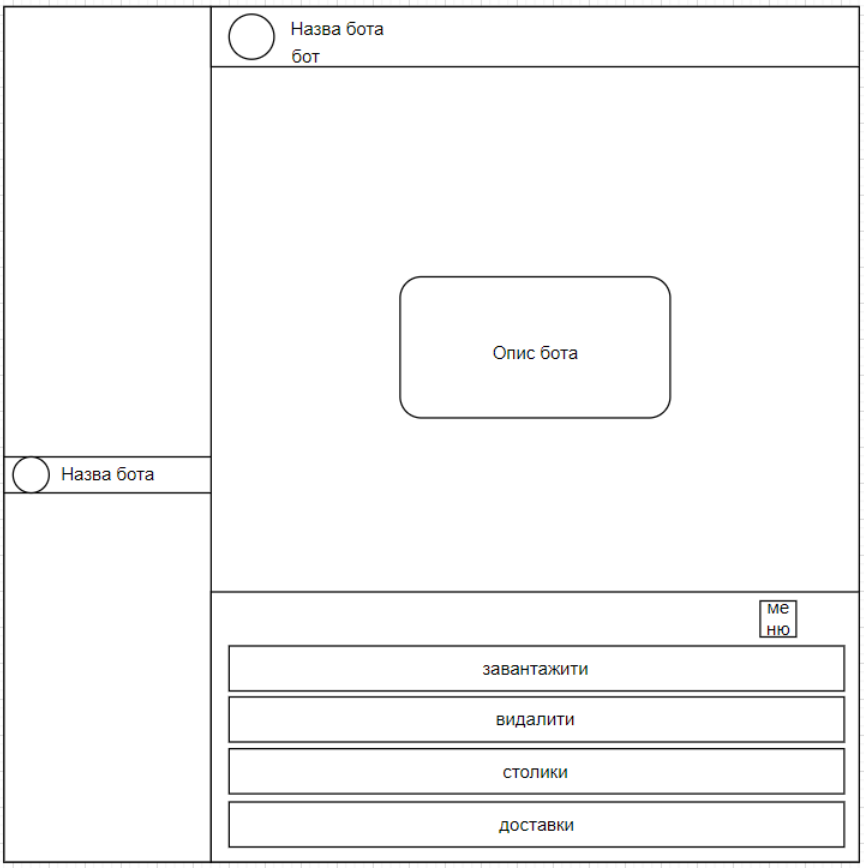

Рисунок А.2 – Дизайн бота, вигляд для адміністратора

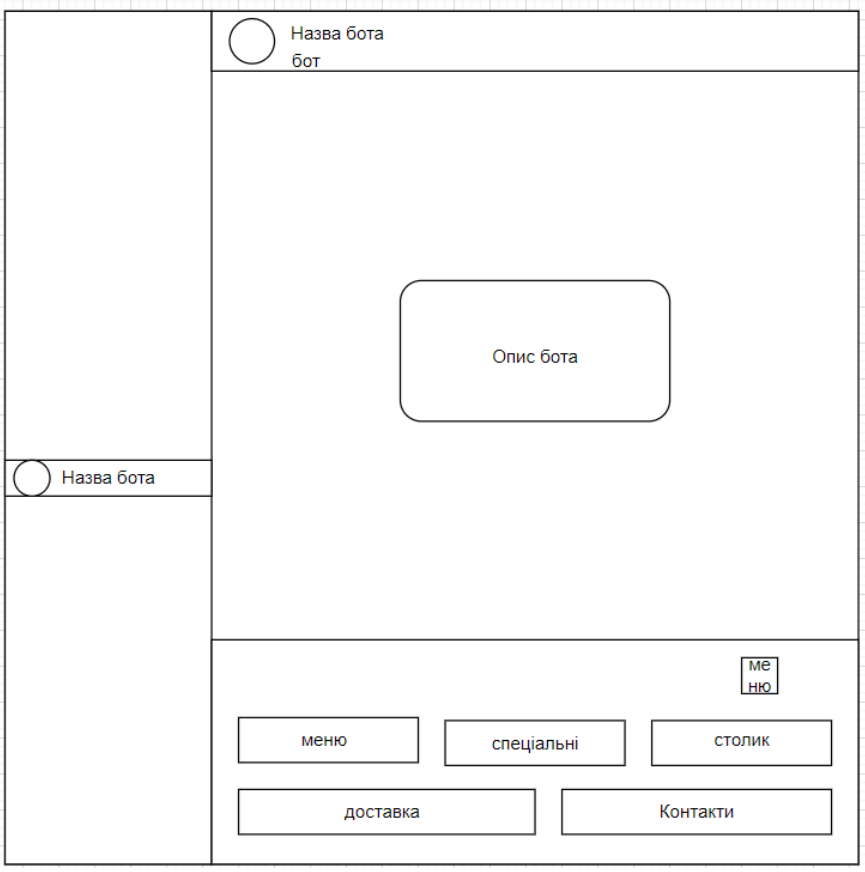

Рисунок А.3 – Дизайн бота, вигляд для клієнта

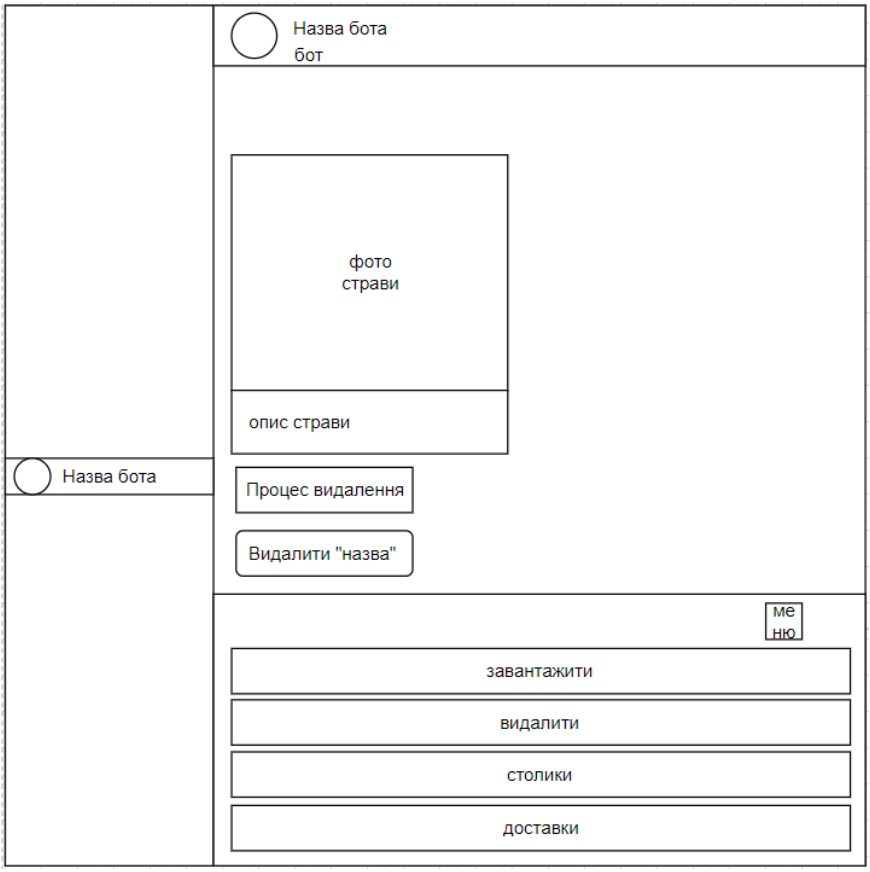

Рисунок А.4 – Дизайн бота, вигляд процесу видалення для адміністратора

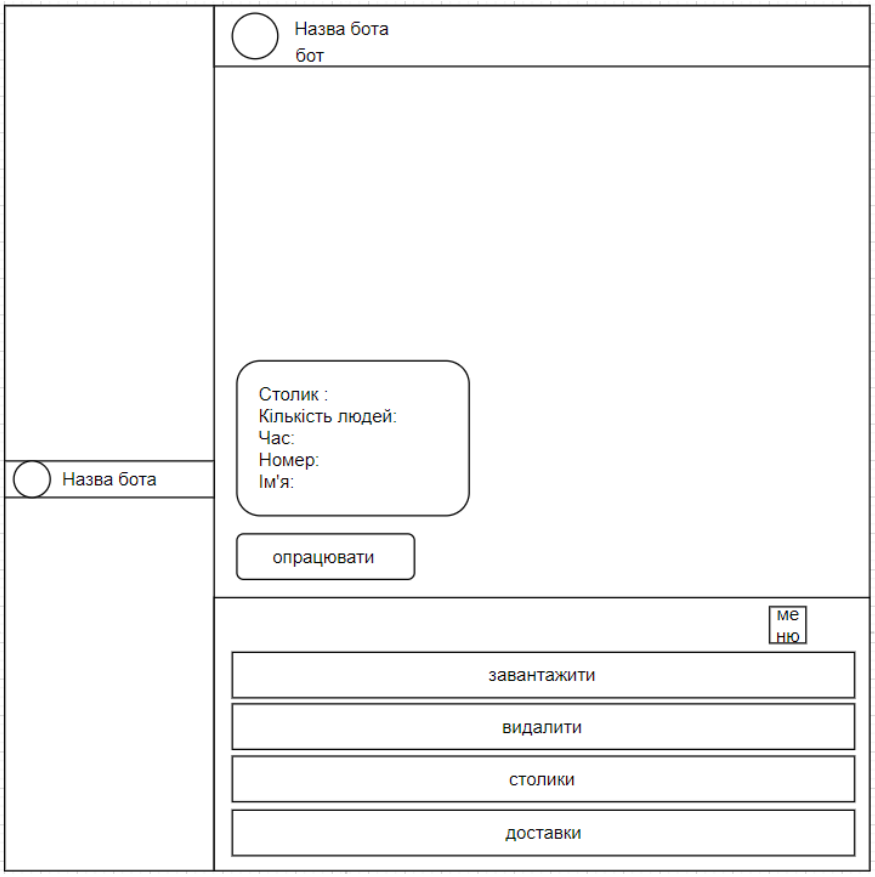

Рисунок А.5 – Дизайн бота, вигляд процесу перегляду/опрацювання столиків для адміністратора

Процес перегляду/опрацювання доставки має вигляд, такий як відображено на рисунку А.6, але відображатиметься саме доставка.

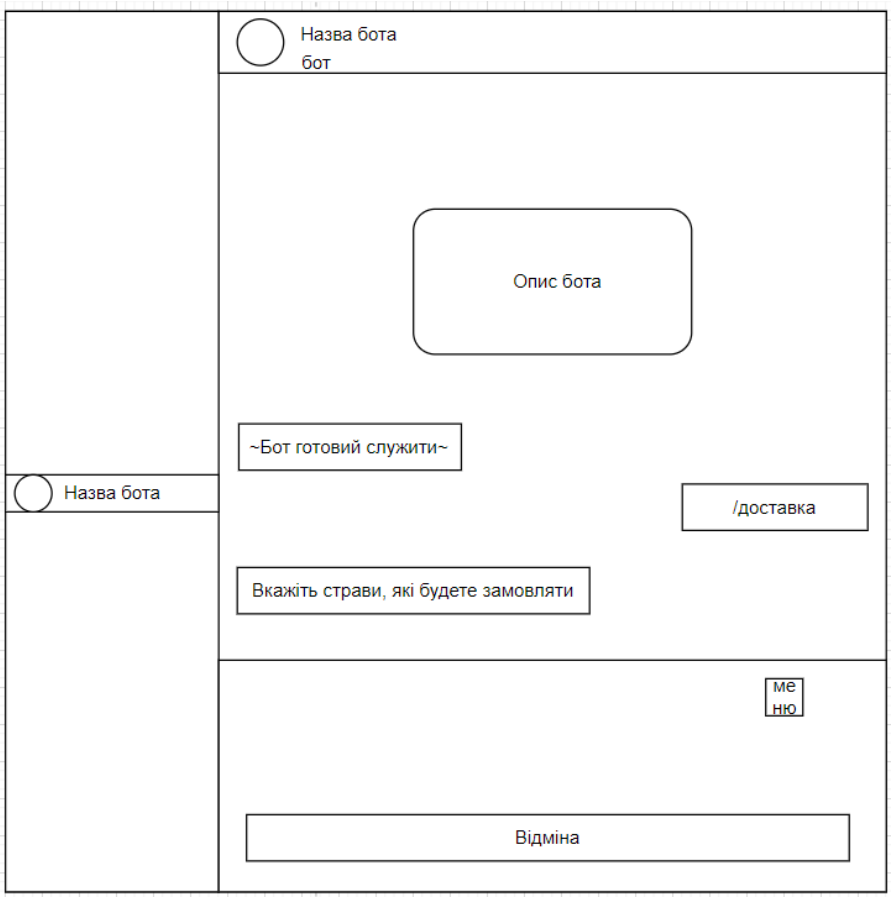

Рисунок А.6 – Дизайн бота, вигляд процесу оформлення доставки клієнтом

2.1.5 Система навігації (карта чат-боту)

Карта чат-бота Телеграм зображена на рисунку А.7

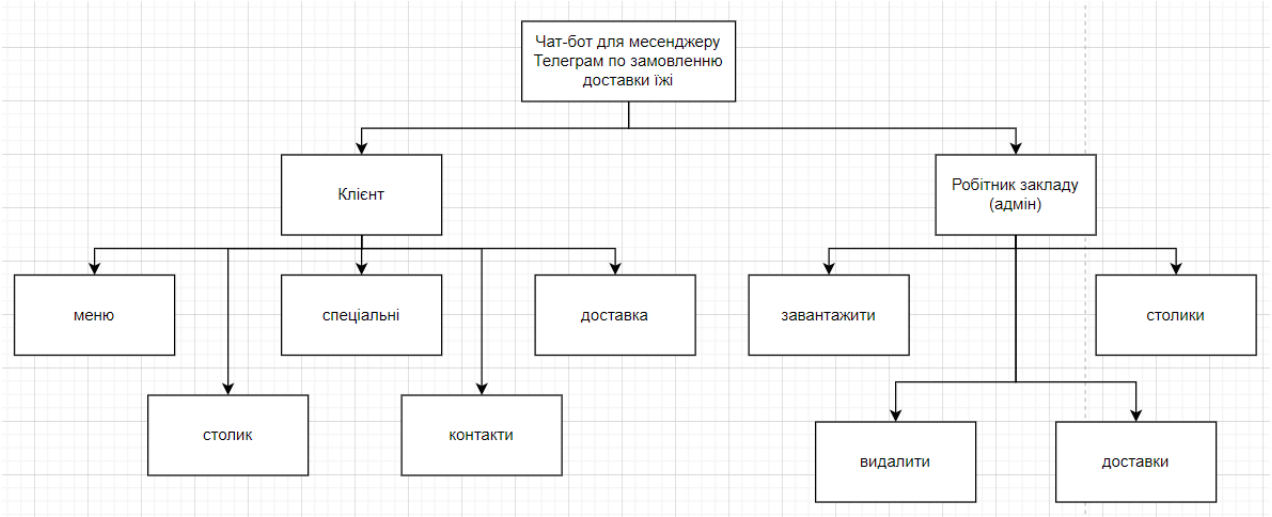

Рисунок А.7 – Карта чат-бота

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	**Потреби користувача**	**Джерело**
UN-01	Перегляд меню закладу	Гість, робітник закладу
UN-02	Меню спеціальних страв для замовлення	Гість, робітник закладу
UN - 03	Бронювання столика	Гість, робітник закладу
UN-4	Замовлення доставки	Гість
UN-5	Редагування даних	Робітник закладу
UN-6	Перегляд замовлень	Робітник закладу

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги:

- Реєстрація користувачів
- Відображення інформації про замовлення та доставку
- Запис адреси замовника для доставки
- Записи з інформацією про страви з меню

– Видалення даних про замовлення
– Оновлення даних про замовлення (статус виконано чи ні)
– Адміністрування інформації про користувачів, видалення, зміну користувацької групи, надання користувацьких прав.

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

ID	**Системні вимоги**	**Пріоритет**	**Опис**
SR-01	**База даних з інформацією про доставку**	M	Надає можливість відображати інформацію про існуючі замовлення
SR-02	**База даних - меню**	M	Надає можливість відображати інформацію про страви з меню
SR-05	**База даних - столики**	M	Надає можливість відображати інформацію про бронювання столиків
SR-04	**Меню з кнопками для гостя**	M	Перелік можливостей та доступних функцій для гостя закладу
SR-05	**Меню з кнопками для працівника закладу**	M	Перелік можливостей та доступних функцій для робітника закладу

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація бота відбувається з використанням:

- Мова програмування Python 3.11.1
- PyCharm :

System requirements

64-bit versions of Microsoft Windows 11, 10, 8

2 GB free RAM minimum, 8 GB of total system RAM recommended

2.5 GB hard disk space, SSD recommended

1024x768 minimum screen resolution

- Мова запитів MySQL 8.0
- DB Browser (SQLite) 3.12.2

2.4.2 Вимоги до лінгвістичного забезпечення

Інтерфейс бота написаний українською, хоча обмежень по вибору мови немає. Також деякі страви можуть мати назву та/чи опис англійською мовою.

2.4.3 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи бота рекомендовано:

– постійне інтернет-з’єднання
– для впевненості у роботі бота оновлений Телеграм до останньої версії
– будь-яка операційна система : Windows, Linux та інші

3 Склад і зміст робіт зі створення боту

Детальний опис етапів створення бота наведено в таблиці А.3.

Таблиця А.3 – Етапи розробки

№	Склад і зміст робіт	Строк розробки (к-ть днів)
0	**Створення Telegram-бота**	46
1	**Підготовка специфікації**	4
2	Визначення проблеми, які вирішить використання бота	2
3	Визначення цілей і задач	2
4	**Визначення властивостей бота**	6
5	Розробка ТЗ	4
6	Редагування та обговорення ТЗ	1
7	Затвердження ТЗ	1
8	**Процес написання**	24
9	Розробка шаблону	10
10	Розробка меню бота	7
11	Створення бази даних	5

Продовження таблиці А.3 – Етапи розробки

12	Наповнення бази даних	3
13	**Тестування**	8
14	Alpha - тестування	3
15	Beta - тестування	3
16	Додаткові виправлення	2
17	**Впровадження в дію**	2
18	Перевірка працездатності	2
19	Написання супровідної документації	2
20	Реліз Telegram-боту	1

4 Вимоги до складу й змісту робіт із введення в експлуатацію

Для використання чат-боту користувачем необхідно знайти його в Телеграм, це можна зробити, виконавши пошук за назвою бота, або використати ім’я юзера @rapid_eats_bot, або перейти за посиланням https://t.me/rapid_eats_bot . Для запуску бота використовується команда «/start». Для використання бота робітниками закладу, використовується спеціально створена група https://t.me/+W-nnnyI0-mZiZGNi . Головний адміністратор може надавати різні права доступу в групі, але саме для використання бота необхідно мати роль «адміністратор» а для запуску бота використовується команда «/admin».

ДОДАТОК Б
ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART. Для успішності проекту необхідно правильно визначити його мету за допомогою SMART-методу. Результати деталізації методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific (конкретна)	Написати чат-боту в крос-платформній системі Telegram для мережі швидкого харчування
Measurable (вимірювана)	Написати бот за 46 днів
Achievable (досяжна, узгоджена)	Необхідні знання програмування мовою Python, знання баз даних MySQL та написання документації
Relevant (реалістична)	Створений бот пришвидшить процес замовляння їжі, замовлення столиків, доставки
Time-framed (обмежена у часі)	Ціль має часове обмеження. Термін визначено за домовленістю між куратором та виконавцем – період навчального року (1 червня 2023 року).

Планування змісту робіт. WBS (Work Breakdown Structure – Ієрархічна структура робіт) - це перше завдання, яке потрібно виконати на стадії планування проекту. WBS - це графічне подання згрупованих елементів проекту у вигляді пакетів робіт, які ієрархічно пов’язані з продуктом проекту. Необхідна для

: забезпечення ефективного управління проектом, визначення і структурування переліку робіт, створення структури звітності, розуміння задач виконавцем. WBS є базовим засобом для створення організаційної структури (OBS) і системи управління проектом, оскільки дозволяє виявити проблеми організації робіт, визначення ієрархії проектних завдань (етапів робіт), підзавдань і пакетів робіт на всіх подальших фазах життєвого циклу проекту.

На найвищому (першому) рівні розміщений продукт проекту. Основні дії та заходи, що забезпечують досягнення мети проекту, зафіксовані на другому рівні декомпозиції. Декомпозиція робіт виконується до тих пір, поки вони не стануть елементарними (простими). Елементарні роботи – це дії, які мають однозначний чіткий результат, на які призначена відповідальному одна конкретна особа, для якої можна обчислити витрати праці і тривалість виконання.

На рисунку Б.1 представлено WBS з написання Telegram-боту.

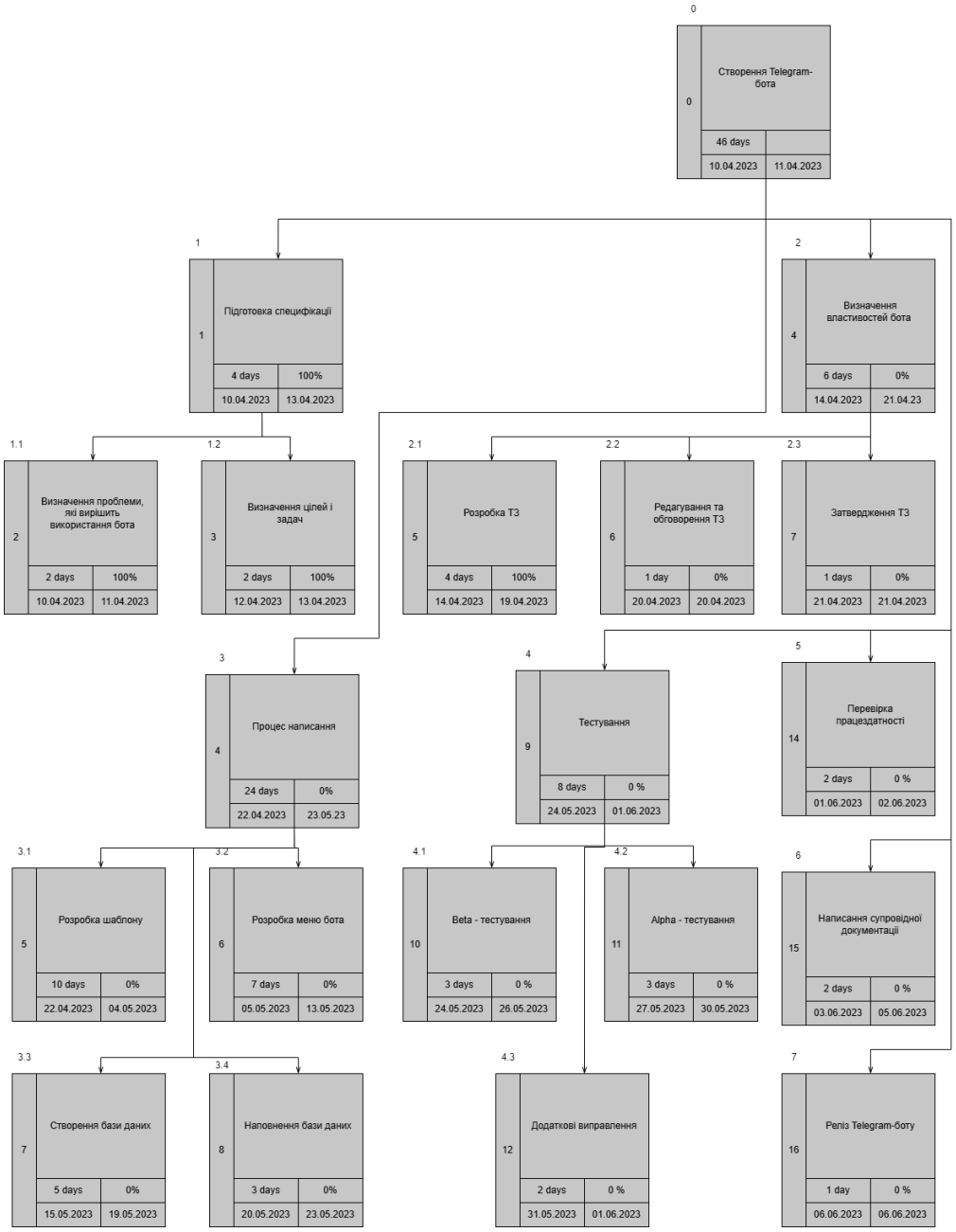

Рисунок Б.1 – WBS-структура робіт проекту

Планування структури виконавців. OBS - друге завдання, яке потрібно виконати на стадії планування проекту. Це графічне відображення учасників проекту (фізичних та юридичних осіб) та їхніх відповідальних осіб, залучених до реалізації проекту. Елементами OBS можуть бути : окремі виконавці (керівники, фахівці, службовці); організації, структурні підрозділи і служби, у яких зайнята та або інша кількість фахівців, що виконують певні функціональні обов'язки; зовнішні постачальники обладнання, послуг.

На рисунку Б.2 представлено організаційну структуру планування проекту.

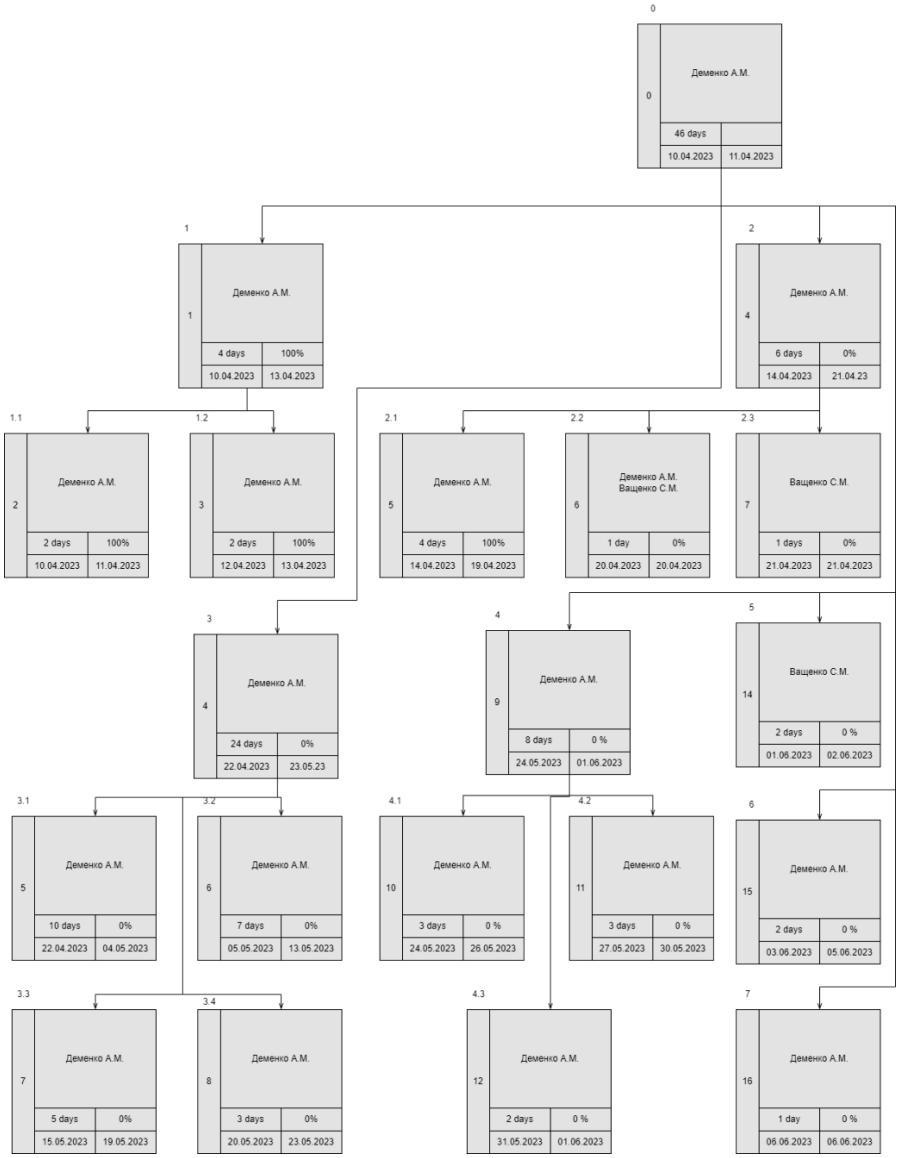

Рисунок Б.2 – OBS-структура робіт проекту

Список виконавців, що функціонують в проекті описано в таблиці Б.2.

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Керівник проекту	Ващенко С.М.	Формує завдання на розробку проекту

Продовження таблиці Б.2 – Виконавці проекту

Менеджер проекту	Деменко А.М.	Відповідає за виконання термінів, розподіл ресурсів за завдань між учасниками. Виконує збір та аналіз даних
Розробник	Деменко А.М.	Написання коду на Python, створення бази даних
Проектувальник	Деменко А.М.	Виконує проектування бази даних та розробляє структуру боту
Тестувальник	Деменко А.М.	Тестування функціоналу

Діаграма Ганта. На діаграмі Ганта всі роботи за проектом представлені у вигляді горизонтальних відрізків, паралельних осі часу. Використання моделі проекту, побудованої в програмному середовищі MS Project, дозволяє контролювати й оптимізувати план виконання робіт, наочно відстежувати хід його виконання.

Календарний графік проекту представлено на рисунку Б.3.

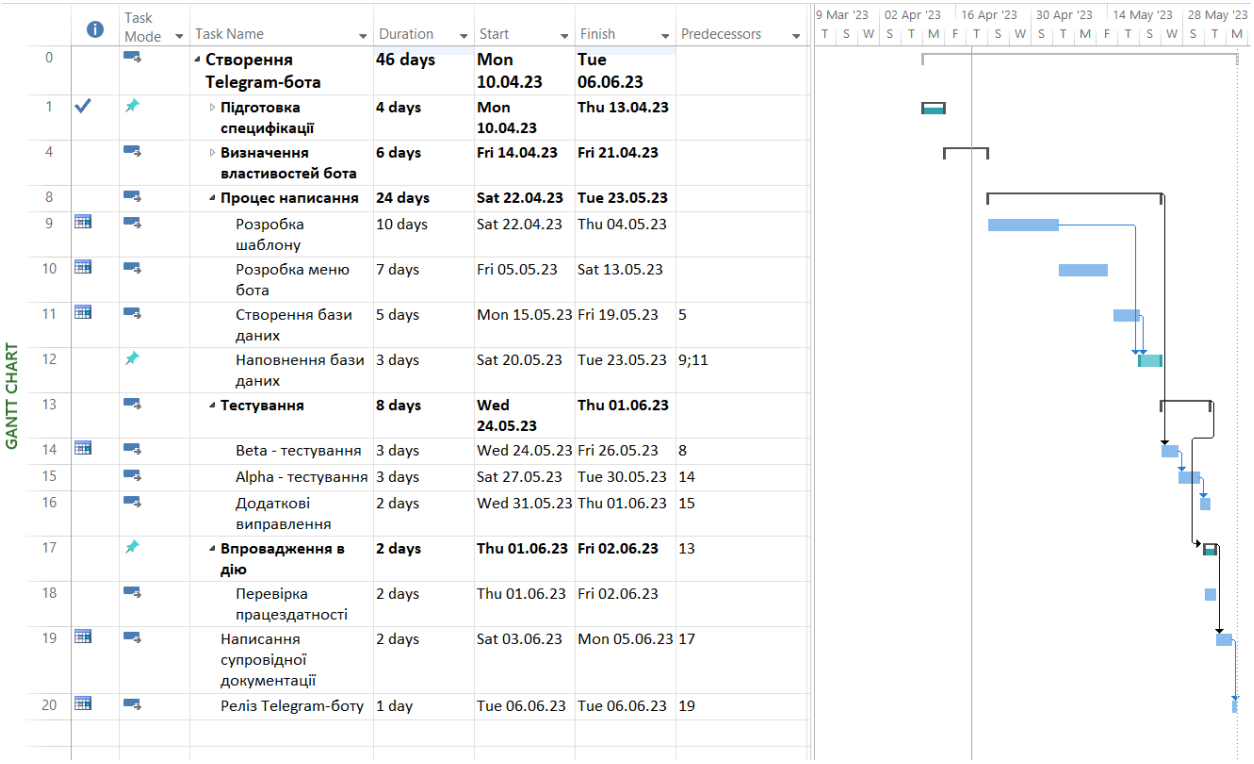

Рисунок Б.3 – Календарний графік проекту

Управління ризиками проекту. Ризик проекту - імовірнісна подія, яка у випадку своєї появи, позитивно або негативно вплине на хід проекту (хоча б на один із показників проекту: зміст, якість, вартість, час). Ідентифікація ризиків - це виявлення ризиків, здатних вплинути на проект, і документальне оформлення їх характеристик. Кожен ризик треба оцінити по 2 параметрах: Імовірність ризику та Вплив ризику на проект.

Оцінювання виконується за показниками, що описані в таблиці Б.3.

Таблиця Б.3 - Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику	Межі
1	Низька	Низький	Прийнятні	1≤R≤2
2	Середня	Середній	Виправдані	3≤R≤4

Продовження таблиці Б.3 - Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

3	Висока	Високий	Недопустимі	6≤R≤9

Була виконана оцінка проекту та визначені ймовірність, вплив і ранг кожного ризику. Упорядковані дані та отримані результати обчислення занесені в таблицю Б.4.

Таблиця Б.4 - Ймовірність, вплив і ранг кожного ризику.

№ ризику	Назва (опис) ризику	Ймовірність (0,1 – 0,9)	Вплив (0,05 – 0,8)	Ранг
1	Відключення світла	0,9	0,8	0,72
2	Загроза обстрілів	0,5	0,8	0,4
3	Хвороба учасників	0,3	0,4	0,12
4	Низька кваліфікація розробників	0,1	0,2	0,02
5	Неоптимальний розподіл часу	0,5	0,5	0,25
6	Часте внесення змін у ТЗ	0,2	0,2	0,04
7	Затримка у виправленні помилок в процесі тестування	0,5	0,2	0,1

Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Було побудовано матрицю ймовірності ризику та впливу загрози (ризику) – таблиця Б.5.

Таблиця Б.5 - Матриця ймовірності ризику та впливу.

Ймовірність ризику	Вплив загрози (ризику)				
	Дуже малий	Малий	Середній	Великий	Дуже великий
	0,05	0,1	0,2	0,4	0,8
0,9					R1(0,72)
0,7					
0,5			R7(0,1)		R2(0,4) R5(0,25)
0,3				R3(0,12)	
0,2			R6(0,04)		
0,1			R4(0,02)		

Класифікація ризиків за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.6.

Таблиця Б.6 – Класифікація ризиків.

№	Назва	Межі	Номера ризиків
1	Прийнятні	$0,005 \leq R \leq 0,05$	4,6
2	Виправдані	$0,05 < R \leq 0,14$	3,7
3	Недопустимі	$0,14 < R \leq 0,72$	1,2,5

У таблиці Б.7 описано ризики та стратегії реагування на кожен з них.

Таблиця Б.7 - Ризики та стратегії реагування.

D	Статус ризику	Опис	Ймовірність	Вплив	Ранг	План А (заходи запобігання виникненню ризику)	Тип стратегії реагування	План Б (заходи усунення наслідків ризику)
S_1	Відкритий	Відключення світла	0,9	0,8	0,72	Мати генератори, павербанки	Ухилення	Зміна графіку роботи в залежності від графіку вимикання світла
S_2	Відкритий	Загроза обстрілів	0,5	0,8	0,4	Працювати у захищеному, безпечному приміщенні	Зменшення	Працювати у безпечних місцях, обладнаними засобами першої допомоги, їжі
S_3	Відкритий	Хвороба учасника	0,3	0,4	0,12	Зміна обсягу роботи	Зменшення	Знайти людину, яка могла б виконати дану роботу, чи допомагати
S_4	Відкритий	Низька кваліфікація розробників	0,1	0,2	0,02	При відборі та виборі персоналу обирати людей, у яких є необхідні	Попередження	Знайти заміну некомпетентним співробітникам, або допомогу у виконанні роботи

						знання та навички		

Продовження таблиці Б.7 - Ризики та стратегії реагування.

S_5	Відкритий	Неоптимальний розподіл часу	0,5	0,5	0,25	Перерозраховувати терміни роботи та вказати нові дедлайни	Зменшення	Скорочення обсягу роботи, визначення нових термінів робіт
S_6	Відкритий	Часте внесення змін у ТЗ	0,2	0,2	0,04	Перевизначити мету, ціль проекту, обговорити неточності та знайти можливі вразливі місця в майбутньому, усунути їх	Ухилення	Зміна ТЗ
S_7	Відкритий	Затримка у виправленні помилок в процесі тестування	0,5	0,2	0,1	На етапі розробки враховувати можливі слабкі місця	Зменшення	Надати додатковий час для усунення помилок та виконання наступного кроку тестування

ДОДАТОК В

Лістинг програмного коду

В.1 DipCode\dataBase__init__.py

```
from dataBase import db_forRapid
```

В.2 DipCode\dataBase\db_forRapid.py

```
import sqlite3 as sq
from create_bot import bot

"""def sql_start():
    global base, cur
    base = sq.connect('menu_Rapid.db')
    cur = base.cursor()
    if base:
        print("~Base connected~")
    base.execute('CREATE TABLE IF NOT EXISTS menu(img TEXT, name TEXT PRIMARY KEY,
description TEXT, price TEXT)')
    base.execute('CREATE TABLE IF NOT EXISTS special_menu(img TEXT, name TEXT
PRIMARY KEY, description TEXT\
    , price TEXT)')#спеціальні страви
    base.commit()"""

def sql_start():
    global base, cur
    base = sq.connect('menu_Rapid.db')
    cur = base.cursor()
    if base:
        print("~Base connected~")
    base.execute('CREATE TABLE IF NOT EXISTS menu(img TEXT, name TEXT PRIMARY KEY,
description TEXT, price TEXT, dish_type TEXT)')
    base.execute('CREATE TABLE IF NOT EXISTS tables(table_data TEXT)')
    base.execute('CREATE TABLE IF NOT EXISTS delivery(delivery_data TEXT)')
```

```
     # спеціальні страви
    base.commit()

async def sql_add_command(state):
    async with state.proxy() as data:
        cur.execute('INSERT INTO menu VALUES (?, ?, ?, ?, ?)',
tuple(data.values()))
        base.commit()

async def sql_add_tables(state):
    async with state.proxy() as data:
        cur.execute('INSERT INTO tables VALUES (?, ?)', tuple(data.values()))
        base.commit()
"""
async def sql_read_all(message):
    for ret in cur.execute('SELECT * FROM menu').fetchall():
        await bot.send_photo(message.from_user.id, ret[0], f'{ret[1]}\nОпис :
{ret[2]}\nЦіна : {ret[3]}')
"""

async def sql_read_all():
    data = []
    for ret in cur.execute('SELECT * FROM menu').fetchall():
        data.append(ret)
    return data


async def sql_read(message, dish_type):
    print("works")
    print(dish_type)
    for ret in cur.execute('SELECT * FROM menu WHERE dish_type== ?',
(dish_type,)).fetchall():
        await bot.send_photo(message.from_user.id, ret[0], f'{ret[1]}\nОпис :
{ret[2]}\nЦіна : {ret[3]}')


async def sql_read2(message):
    print("works2")
    for ret in cur.execute('SELECT * FROM tables').fetchall():
        await bot.send_message(f'Замовник: {ret[0]}\nКількість людей : {ret[1]}')

async def sql_delete_command(data):
```

```
    cur.execute('DELETE FROM menu WHERE name == ?', (data,))
    base.commit()

def add_table(table_data):
    cur.execute("INSERT INTO tables (table_data) VALUES (?)", (table_data, ))
    base.commit()

def add_delivery(delivery_data):
    cur.execute("INSERT INTO delivery (delivery_data) VALUES (?)", (delivery_data,
))
    base.commit()

def read_tables():
    tables = []
    for ret in cur.execute("SELECT * FROM tables"):
        table_data = ret[0]
        tables.append(table_data)
    return tables

def read_delivery():
    delivery = []
    for ret in cur.execute("SELECT * FROM delivery"):
         delivery_data = ret[0]
         delivery.append(delivery_data)
    return delivery

def read_tables():
    tables = []
    for ret in cur.execute("SELECT * FROM tables"):
        table_data = ret[0]
        tables.append(table_data)
    return tables

def del_tables(table_data):
    cur.execute('DELETE FROM tables WHERE table_data = ?', (table_data,))
    base.commit()


def del_delivery(delivery_data):
    cur.execute('DELETE FROM delivery WHERE delivery_data = ?', (delivery_data,))
    base.commit()
```

B.3 DipCode\handlers__init__.py

```
from handlers import client
from handlers import admin
from handlers import other
```

B.4 DipCode\handlers\admin.py

```
from aiogram.dispatcher import FSMContext
from aiogram import types, Dispatcher
from aiogram.dispatcher.filters.state import State, StatesGroup
from create_bot import dp, bot
from aiogram.dispatcher.filters import Text
from dataBase import db_forRapid
from keyboards import admin_kb
from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton
import aiogram.utils.markdown as md

ID = None
class FSMAdmin(StatesGroup):
    photo = State()
    name = State()
    description = State()
    price = State()
    dish_type = State()
#адмін повинен написати в групу, щоб взяти ID та перевірити права доступу
#@dp.message_handler(commands=['admin'], is_chat_admin = True)
async def make_change_command(message: types.Message):
    global ID
    ID = message.from_user.id
    await bot.send_message(message.from_user.id, '~Бот готовий служити~',
reply_markup=admin_kb.button_case_admin)
    await message.delete()

#@dp.message_handler(commands='Завантажити', state=None)
async def cm_start(message: types.Message):
    if message.from_user.id == ID:
```

```
        await FSMAdmin.photo.set()
        await message.reply('Завантаж фото 🖼️')

#@dp.message_handler(state="*", commands='відміна')
#@dp.message_handler(Text(equals='відміна', ignore_case = True), state="*")
async def cancel_handler(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        current_state = await state.get_state()
        if current_state is None:
            return
        await state.finish()
        await message.reply('OK')

#@dp.message_handler(content_types=['photo'], state=FSMAdmin.photo)
async def load_photo(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['photo'] = message.photo[0].file_id
        await FSMAdmin.next()
        await message.reply('Введи назву ✍️')


#@dp.message_handler(state=FSMAdmin.name)
async def load_name(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['name'] = message.text
        await FSMAdmin.next()
        await message.reply('Введіть опис ✍️')


#@dp.message_handler(state=FSMAdmin.description)
async def load_description(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['description'] = message.text
        await FSMAdmin.next()
        return await message.reply('Вкажи ціну 💰')


#@dp.message_handler(state=FSMAdmin.price)
```

```
async def load_price(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['price'] = float(message.text)
        await FSMAdmin.next()
        return await message.reply('Введи тип страви 🍰 (спеціальна чи звичайна)')

async def load_dish_type(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['dish_type'] = message.text
        await db_forRapid.sql_add_command(state)

        await state.finish()

@dp.callback_query_handler(text='del_table')
async def del_table(callback_query: types.CallbackQuery):
    db_forRapid.del_tables(callback_query.message.text)
    await callback_query.answer('Запит на столик опрацьований!')

@dp.callback_query_handler(text='del_delivery')
async def del_delivery(callback_query: types.CallbackQuery):
    db_forRapid.del_delivery(callback_query.message.text)
    await callback_query.answer('Запит на доставку опрацьований!')

@dp.callback_query_handler(lambda x: x.data and x.data.startswith('del '))
async def del_callback_run(callback_query: types.CallbackQuery):
    await db_forRapid.sql_delete_command(callback_query.data.replace('del ',''))
    await callback_query.answer(text=f'{callback_query.data.replace("del ","")}
видалена.', show_alert=True)

@dp.message_handler(commands='видалити')
async def delete_item(message: types.Message):
    if message.from_user.id==ID:
        read = await db_forRapid.sql_read_all()
        for ret in read:
            await bot.send_photo(message.from_user.id,ret[0], f'{ret[1]}\nОпис :
{ret[2]}\nЦіна : {ret[3]}')
            await bot.send_message(message.from_user.id, text='Процес видалення',
reply_markup=InlineKeyboardMarkup().\
                                    add(InlineKeyboardButton(f'Видалити {ret[1]}',
callback_data=f'del {ret[1]}')))
```

```
async def get_tables(message: types.Message):
    tables = db_forRapid.read_tables()
    if not tables:
        await message.answer("Ніхто ще не замовляв столики!")
    else:
        for table in tables:
            await message.answer(table,
reply_markup=InlineKeyboardMarkup().add(InlineKeyboardButton(text='Опрацювати ✅',
callback_data='del_table')))


async def get_delivery(message: types.Message):
    deliveries = db_forRapid.read_delivery()
    if not deliveries:
        await message.answer("Ніхто ще не замовляв достаку!")
    else:
        for delivery in deliveries:
            await message.answer(delivery,
reply_markup=InlineKeyboardMarkup().add(InlineKeyboardButton(text='Опрацювати ✅',
callback_data='del_delivery')))


def register_handlers_admin(dp: Dispatcher):
    dp.register_message_handler(cm_start, commands=['завантажити'], state=None)
    dp.register_message_handler(cancel_handler, Text(equals='відміна',
ignore_case=True), state="*")
    dp.register_message_handler(make_change_command, commands=['admin'],
is_chat_admin=True)
    dp.register_message_handler(load_photo, content_types=['photo'],
state=FSMAdmin.photo)
    dp.register_message_handler(load_name, state=FSMAdmin.name)
    dp.register_message_handler(load_description, state=FSMAdmin.description)
    dp.register_message_handler(load_price, state=FSMAdmin.price)
    dp.register_message_handler(load_dish_type, state=FSMAdmin.dish_type)
    dp.register_message_handler(cancel_handler, state="*", commands='відміна')
    dp.register_message_handler(get_tables, commands=['столики'], state=None)
    dp.register_message_handler(get_delivery, commands=['доставки'], state=None)
    #dp.register_message_handler(del_callback_run)
    #dp.register_message_handler(delete_item, commands='видалити')
```

B.5 DipCode\handlers\client.py

```
from aiogram import types, Dispatcher
from create_bot import dp, bot
from keyboards import kb_client
from keyboards import rmm_client
from dataBase import db_forRapid
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup
from aiogram.types import ReplyKeyboardRemove
import re


#@dp.message_handler(commands=['start,help'])
class BookTable(StatesGroup):
    p_amount = State()
    time = State()
    b_number = State()
    b_name = State()

class MakeOrder(StatesGroup):
    order = State()
    name = State()
    number = State()
    adress = State()

async def cancel_action(message: types.Message, state: FSMContext):
    if state is None:
        return

    await state.finish()
    await message.answer("Ви повернулися в головне меню!",
                         reply_markup=kb_client)

async def command_start(message: types.Message):
    try:
        await bot.send_message(message.from_user.id, 'Enjoy your meal',
reply_markup=kb_client)
        await message.delete()
    except:
        await message.reply('For private use : \nhttps://t.me/+eIGeK8h-XsAzNzhi')
```

```
#@dp.message_handler(commands=['working_time'])
async def work_time_command(message: types.Message):
    await bot.send_message(message.from_user.id, "Пн-Пт з 10:00 до 22:00")

async def seeContacts_command(message: types.Message):
    await bot.send_message(message.from_user.id, "e-mail :
rapidEatspost@gmail.com\nrapidAdminEats@gmail.com")#,
reply_markup=ReplyKeyboardRemove())

#@dp.message_handler(commands=['меню'])
async def food_menu_command(message: types.Message):
    dish_type = 'звичайна'
    await db_forRapid.sql_read(message, dish_type)


#@dp.message_handler(commands=['спеціальні'])
async def special_food_menu_command(message: types.Message):
    dish_type = 'спеціальна'
    await db_forRapid.sql_read(message, dish_type)


#@dp.message_handler(commands=['столик'])
async def book_table_command(message: types.Message):
    await message.answer('В нашому ресторані можна забронювати столик максимум на
8 людей.')
    await message.answer('На яку кількість людей бронювати столик? 👶',
reply_markup=rmm_client)
    await BookTable.p_amount.set()

async def valid_p_amount_1(message: types.Message, state: FSMContext):
    await message.reply('Бро, так не троллять 💀')

async def valid_p_amount_2(message: types.Message, state: FSMContext):
    await message.reply('Столик бронюється мінімум на одну і максимум на 8 людей
🙄')

async def load_p_amount(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['p_amount'] = message.text
```

```
    await message.reply('На котру годину бронювати столик? 🕐\nВведіть час у
форматі <b>HH:MM</b>', parse_mode="HTML")
    await BookTable.time.set()

async def valid_time_format(message: types.Message, state: FSMContext):
    await message.reply('Неправильний формат часу. Введіть час у форматі HH:MM.')

async def valid_hour_range(message: types.Message, state: FSMContext):
    await message.reply('Години повинні бути між 10 і 22.')

async def valid_minute_range(message: types.Message, state: FSMContext):
    await message.reply('Хвилини повинні бути між 00 і 59.')

async def load_time(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['time'] = message.text

    await message.reply('Вкажіть Ваш номер телефону 📱')
    await BookTable.b_number.set()

async def valid_number(message: types.Message, state: FSMContext):
    await message.reply('Вкажіть номер телефону у форматі +380XXXXXXXXX 🙏')

async def load_b_number(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['b_number'] = message.text

    await message.reply('Як до Вас звертатися?')
    await BookTable.b_name.set()

async def valid_name(message: types.Message, state: FSMContext):
    await message.reply('Круто 👍\nВведіть нормальне ім\'я!')

async def load_b_name(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['b_name'] = message.text
        bt_text = f"Столик:\nКількість людей: {data['p_amount']}\nЧас:
{data['time']}\nНомер: {data['b_number']}\nІм\'я: {data['b_name']}"
        db_forRapid.add_table(bt_text)
        await message.answer(bt_text)
```

```
    await message.answer('Дякую! Найближчим часом з вами зв\'яжуться!',
reply_markup=kb_client)
    await state.finish()


async def make_order_command(message: types.Message):
    await message.answer('Вкажіть піци, які будете замовляти 🍕',
reply_markup=rmm_client)
    await MakeOrder.order.set()

async def load_order(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['order'] = message.text

    await message.answer('Вкажіть Ваше ім\'я 👴')
    await MakeOrder.name.set()

async def load_name(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['name'] = message.text

    await message.reply('Вкажіть Ваш номер телефону 📱')
    await MakeOrder.number.set()

async def load_number(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['number'] = message.text

    await message.reply('Вкажіть Вашу адресу 🏠')
    await MakeOrder.adress.set()

async def load_adress(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['adress'] = message.text
        mo_text = f"Доставка:\nПіци: {data['order']}\nІм\'я: {data['name']}\nНомер
телефону: {data['number']}\nАдреса: {data['adress']}"
        db_forRapid.add_delivery(mo_text)
        await message.answer(mo_text)
```

```
    await message.answer('Дякую! Найближчим часом з вами зв\'яжуться!',
reply_markup=kb_client)
    await state.finish()


def register_handlers_client(dp: Dispatcher):
    dp.register_message_handler(cancel_action, text="Відміна 🚫", state='*')
    dp.register_message_handler(command_start, commands=['start','help'])
    dp.register_message_handler(work_time_command, commands=['Часи роботи'])
    dp.register_message_handler(seeContacts_command, commands=['контакти'])
    dp.register_message_handler(food_menu_command, commands=['меню'])
    dp.register_message_handler(special_food_menu_command,
commands=['спеціальні'])
    dp.register_message_handler(book_table_command, commands=['столик'])
    dp.register_message_handler(valid_p_amount_1, lambda message: not
message.text.isdigit(), state=BookTable.p_amount)
    dp.register_message_handler(valid_p_amount_2, lambda message: not 1 <=
int(message.text) <= 8, state=BookTable.p_amount)
    dp.register_message_handler(load_p_amount, state=BookTable.p_amount)
    dp.register_message_handler(valid_time_format, lambda message: not
re.match(r'^\d{2}:\d{2}$', message.text), state=BookTable.time)
    dp.register_message_handler(valid_hour_range, lambda message: not 10 <=
int(message.text.split(':')[0]) <= 22, state=BookTable.time)
    dp.register_message_handler(valid_minute_range, lambda message: not 0 <=
int(message.text.split(':')[1]) <= 59, state=BookTable.time)
    dp.register_message_handler(load_time, state=BookTable.time)
    dp.register_message_handler(valid_number, lambda message: not
re.compile(r'^\+380\d{9}$').match(message.text), state=BookTable.b_number)
    dp.register_message_handler(load_b_number, state=BookTable.b_number)
    dp.register_message_handler(valid_name, lambda message: not re.match(r'^[a-
яґєіїA-ЯҐЄІЇ]{1,50}$', message.text), state=BookTable.b_name)
    dp.register_message_handler(load_b_name, state=BookTable.b_name)
    dp.register_message_handler(make_order_command, commands=['доставка']
)#text="доставка"
    dp.register_message_handler(load_order, state=MakeOrder.order)
    dp.register_message_handler(valid_name, lambda message: not re.match(r'^[a-
яґєіїA-ЯҐЄІЇ]{1,50}$', message.text), state=MakeOrder.name)
    dp.register_message_handler(load_name, state=MakeOrder.name)
    dp.register_message_handler(valid_number, lambda message: not
re.compile(r'^\+380\d{9}$').match(message.text), state=MakeOrder.number)
    dp.register_message_handler(load_number, state=MakeOrder.number)
    dp.register_message_handler(load_adress, state=MakeOrder.adress)
```

B.6 DipCode\handlers\other.py

```
from aiogram import types, Dispatcher
import json,string
from create_bot import dp

#@dp.message_handler()
async def echo_send(message: types.Message):
    if {i.lower().translate(str.maketrans('','',string.punctuation)) for i in
message.text.split(' ')}\
        .intersection(set(json.load(open('Bad_words.json')))) != set():
            await message.reply('Censure!')
            await message.delete()

def register_handlers_other(dp : Dispatcher):
    dp.register_message_handler(echo_send)
```

B.7 DipCode\keyboards__init__.py

```
from keyboards.client_kb import kb_client
from keyboards.client_kb import rmm_client
```

B.8 DipCode\keyboards\admin_kb.py

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, ReplyKeyboardRemove

b_load = KeyboardButton('/завантажити')
b_delete = KeyboardButton('/видалити')
b_table = KeyboardButton('/столики')
b_delivery = KeyboardButton('/доставки')

button_case_admin =
ReplyKeyboardMarkup(resize_keyboard=True).add(b_load).add(b_delete).add(b_table).a
dd(b_delivery)
```

B.9 DipCode\keyboards\client_kb.py

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, ReplyKeyboardRemove

b_menu = KeyboardButton("/меню")
b_specialDish = KeyboardButton("/спеціальні")
b_bookTables = KeyboardButton("/столик")
b_delivery = KeyboardButton("/доставка")
b_seeContacts = KeyboardButton("/Контакти")

kb_client = ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)

kb_client.insert(b_menu).insert(b_specialDish).insert(b_bookTables).insert(b_deliv
ery).insert(b_seeContacts)#.row(b_sendMyNumber, b_sendGeolocation)
#add кожна кнопка з нового рядка
#insert де є місце
#row в рядок

return_main_menu = KeyboardButton("Відміна 🚫")
rmm_client = ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)
rmm_client.insert(return_main_menu)
```

B.10 DipCode\bot_telegram.py

```
from aiogram.utils import executor
from create_bot import dp
from handlers import client, admin, other
from dataBase import db_forRapid

#venv\Scripts\activate

async def on_startup(_):
    print('being online')
    db_forRapid.sql_start()

admin.register_handlers_admin(dp)
client.register_handlers_client(dp)
other.register_handlers_other(dp)


executor.start_polling(dp, skip_updates=True, on_startup=on_startup)
```

B.11 DipCode\create_bot.py

```
from aiogram import Bot
from aiogram.dispatcher import Dispatcher
import os
from aiogram.contrib.fsm_storage.memory import MemoryStorage

storage = MemoryStorage()

bot = Bot(token='TELEGRAM_BOT_TOKEN')
dp = Dispatcher(bot, storage = storage)
```

ДОДАТОК Г

Апробація

Telegram чат- бот для мережі швидкого харчування

Деменко А.М, *студент ІТ-91*; Ващенко С.М., *доцент*
Сумський державний університет, м. Суми, Україна

Потреба у харчуванні – основна потреба людства за всіх часів. З розвитком технологій з'явилося багато різних шляхів для купівлі чи заготівлі продуктів, а також замовлення готових страв. Сприяє цьому, в першу чергу, розвиток web-технологій. Сервіси, що надають такі послуги користувачеві, реалізовуються у вигляді сайтів, додатків, ботів. Перевагою бота, порівняно з іншими сервісами, є не лише доступність 24/7 для користувача, а ще й швидкі внутрішні процеси (питання користувача – відповідь), постійна доступність необхідної інформації, відсутня необхідність в поглиблених знаннях користування комп'ютерними технологіями, необмежений запас терпіння, яке іноді необхідне при спілкуванні.

Тому було вирішено реалізувати сервіс замовлення їжі у вигляді саме чат-боту. Вибір месенджеру Telegram грунтується на тому, що це один з найпопулярніших месенджерів в Україні. За розробленими вимогами інтерфейс має бути оформлений українською мовою. Текст повинен бути читабельним за легким для розуміння, кнопки меню достатнього розміру для зручного натискання.

При запуску бота користувачеві спершу надається короткий опис та технічна підтримка. Після запуску з'явиться кнопка меню, у якому відповідно до ролі користувача ботом відображуватимуться різні функції. Якщо користувач звичайний клієнт, то доступні функції такі: перегляд меню, спецзамовлення, звичайна доставка, перевірка столиків для бронювання. Для працівників закладу – перегляд замовлень, їх редагування.

Для реалізації Telegram-боту було використано такі технології: мову програмування Python, мову запитів SQL, середовище програмування PyCharm та інструмент роботи з базами даних Workbench.

Використання розробленого боту спростить організацію процесу спілкування «замовник-виконавець», а також пришвидшить виконання замовлень.