Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка програмного забезпечення для чат-боту їжі»
за освітньою програмою: «12 Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ІІ31911»

_____ /Максим ТАР
(підпис студента) (Ім'я ПРІЗВ

Керівник: _____ /доц. Вадим АНДРЮЩЕН
(підпис) (посада, Ім'я ПРІЗВ

Нормоконтролер: _____ /доц. Світлана ВОЛК
(підпис) (посада, Ім'я ПРІЗВ

Засвідчую, що у цій роботі немає запозичень з
інших авторів без відповідних пос

Студент _____
(підпис)

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»
Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis
on the topic: «Software development for a food chatbot»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1911: /Maksym TARAN/
Scientific Supervisor: /Vadym ANDRIUSHCHENKO/
Normative controller: /Svitlana VOLKOVA/

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____/Вадим ГОРЯЧКІН/
(підпис)
Дата_____

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра
студенту Тарану Максиму Ігоревичу
1. Тема роботи: «Розробка програмного забезпечення для чат-боту їжі»
Керівник роботи: Андрющенко Вадим Олександрович, доцент
затверджені наказом № 1209 ст від 07.12.2022
2. Строк подання студентом роботи: 12.06.2023 р.
3. Вихідні дані до роботи: _____
_____

4. Пояснювальна записка, зміст (питання, для опрацювання):

4.1 Аналітична частина:

Для отримання якісної програми треба зібрати вимоги та ознайомитися з предметною областю, знайти та оглянути аналоги і визначити яку програму потрібно розробити.

4.2 Основна частина:

Спроектувати систему, яка буде виконувати зазначену задачу, визначити сутності необхідні для реалізації програмного додатка, розробити інтерфейс користувача.

Обрати мову програмування на котрій буде реалізовано додаток та розробити програму, що реалізує проект системи.

4.3 Тестування:

Протестувати та налагодити розроблену програму.

Висновки щодо роботи.

5. Перелік графічного матеріалу:

Скріншоти виконання програми.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір та аналіз вимог	22.01.2023	10%
2	Зовнішнє проектування	12.02.2023	20%
3	Внутрішнє проектування	16.03.2023	30%
4	Розробка алгоритмів	18 .04.2023	40%
5	Розробка програмного забезпечення	26.05.2023	65%
6	Тестування програмного забезпечення	10.06.2023	85%
7	Подання кваліфікаційної роботи до кафедри	12.06.2023	100%
8	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	26.06.2023	

Студент _____ _Максим ТАРАН_
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ _доц. Вадим АНДРЮЩЕНКО__
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра:

В сучасному світі замовлення їжі через ресторани та доставку є популярним та зручним способом задоволення гастрономічних потреб. Однак, часто користувачі мають питання щодо стану свого замовлення або потребують додаткової інформації. Для вирішення цих проблем та забезпечення зручності для користувачів, використання чат-ботів стає все більш популярним. Цей реферат присвячений розробці чат-бота для користувача, який дозволяє переглядати стан замовлення їжі та отримувати відповіді на питання, пов'язані з процесом замовлення.

Об'єктом дослідження та розробки є чат-бот для користувача, спеціально призначений для перегляду стану замовлення їжі та надання відповідей на питання, пов'язані з процесом замовлення.

Метою даної роботи є розробка чат-бота для користувача, який надає зручний та швидкий спосіб перегляду стану замовлення їжі та отримання відповідей на питання. Цей чат-бот спростить процес взаємодії користувачів з ресторанами та доставкою їжі, забезпечуючи зручну та ефективну комунікацію.

Для розв'язання поставленої задачі було використано мову програмування C# та Telegram Bot API. Чат-бот розробляється на платформі Telegram, яка забезпечує зручний інтерфейс для взаємодії з користувачами. Застосування мови програмування C# дозволяє створити потужну та функціональну систему, здатну обробляти запити користувачів та надавати необхідну інформацію.

Результатом проведеної роботи буде функціональний чат-бот для користувача, який забезпечуватиме можливість перегляду стану замовлення їжі та отримання відповідей на питання, пов'язані з процесом замовлення. Чат-бот буде забезпечувати швидку та зручну комунікацію, що дозволить користувачам отримати необхідну інформацію без зайвих зусиль.

Розробка чат-бота для користувача з переглядом стану замовлення їжі та відповідями на питання щодо замовлення має велике значення для покращення комунікації між користувачами та ресторанами/доставкою їжі. Цей чат-бот спростить процес замовлення та забезпечить зручну і ефективну взаємодію. Користувачі зможуть швидко отримувати необхідну інформацію та відповіді на свої запитання, а ресторани/доставка їжі зможуть поліпшити якість обслуговування та задовольнити потреби клієнтів.

Пояснювальна записка складається зі вступу, п’яти розділів, висновків, бібліографічного списку та додатків.

Вступ описує суть, мету та актуальність роботи (3 сторінки).

Перший розділ: збір та аналіз вимог (11 сторінок).

Другий розділ: проєктування системи (10 сторінок).

Третій розділ: розробка програми (13 сторінок).

Четвертий розділ: тестування та налагодження (22 сторінки).

П’ятий розділ: аналіз та висновки (3 сторінок).

Додатки: робочий проект. Таблиць – 3 , рисунків – 30, бібліографія – 23 джерела.

Ключові слова: чат-бот, замовлення їжі, штучний інтелект, обробка природної мови, автоматизація, ресторанний бізнес.

ЗМІСТ

ВСТУП

Актуальність роботи. У сучасному світі зростає популярність замовлення їжі через ресторани та доставку. Однак, з підвищенням попиту виникає потреба у поліпшенні комунікації між користувачами та ресторанами/доставкою їжі, а також у забезпеченні зручного та оперативного доступу до інформації про стан замовлення та відповіді на питання користувачів.

Таким чином, актуальність роботи полягає у вирішенні цієї проблеми шляхом розробки чат-бота для користувача, який дозволяє переглядати стан замовлення їжі та отримувати відповіді на питання. Цей чат-бот спростить процес взаємодії між користувачами та ресторанами/доставкою їжі, забезпечуючи зручну та ефективну комунікацію.

В порівнянні з відомими розв'язаннями проблеми, розробка чат-бота для користувача з переглядом стану замовлення їжі та відповідями на питання щодо замовлення є актуальною, оскільки існуючі методи комунікації (телефонні дзвінки, повідомлення електронною поштою тощо) можуть бути часо- та працезатратними, а також не забезпечувати миттєвої доступності інформації.

Обґрунтування актуальності обраної теми роботи підкреслює значущість вирішення проблеми комунікації та доступу до інформації в сфері замовлення їжі. Розробка чат-бота для користувача є важливим кроком у поліпшенні процесу замовлення та надання інформації, а також підвищенні задоволеності користувачів. Така тема свідчить про професійну підготовленість та розуміння автором її значення для сучасної практики та потреб користувачів.

Мета роботи. Мета роботи полягає в створенні функціонального і ефективного чат-бота, який забезпечить розробникам та користувачам значну користь. Для розробників чат-бот буде підвищенням їхнього професійного рівня, розширенням навичок у сфері розробки програмного забезпечення та пропозицією нового продукту для ринку. Користувачі ПЗ матимуть можливість зручно та швидко отримувати інформацію про стан свого замовлення та отримувати відповіді на

питання, що виникають у процесі замовлення. Це сприятиме покращенню їхнього досвіду використання послуг ресторанів та доставки їжі, економлячи їм час та зусилля.

Мета роботи сформульована з урахуванням отриманих результатів проведеної роботи та спрямована на розробку конкретного продукту, що приносить користь як розробникам, так і користувачам.

Експлуатаційне призначення розробленого чат-бота полягає в наданні зручного та швидкого доступу користувачам до інформації про стан їхніх замовлень та відповідей на їхні запитання, пов'язані з процесом замовлення їжі. Цей продукт має наступні практичні застосування:

1. Перегляд стану замовлення: Користувачі можуть за допомогою чат-бота швидко отримати інформацію про поточний статус свого замовлення. Вони можуть перевірити, чи було прийнято їхнє замовлення, чи виконується процес приготування їжі, які етапи доставки вже пройдені та коли очікується прибуття замовлення. Це дозволяє користувачам бути в курсі та планувати свій час.

2. Відповіді на питання користувачів: Чат-бот надає можливість задавати питання, пов'язані з процесом замовлення, меню, оплатою, доставкою тощо. Він забезпечує швидкі та точні відповіді на ці запитання, що допомагає користувачам усунути незрозумілості та знайти необхідну інформацію. Це покращує комунікацію між користувачами та ресторанами/службами доставки їжі, а також сприяє задоволенню їхніх потреб.

3. Оптимізація робочого процесу: Для ресторанів та служб доставки їжі чат-бот може виявитися корисним інструментом для оптимізації робочого процесу. Він дозволяє автоматизувати взаємодію з клієнтами, зменшуючи навантаження на робітників та забезпечуючи швидку відповідь на запитання. Крім того, чат-бот може збирати дані про популярні запити та проблеми клієнтів, що дозволить аналізувати цю інформацію та вдосконалювати сервіс [4].

Експлуатація розробленого чат-бота дозволяє покращити якість обслуговування клієнтів, зменшити час очікування та запити на підтримку, а також забезпечити ефективну комунікацію між ресторанами/службами доставки їжі та їхніми клієнтами.

1 ЗБІР ТА АНАЛІЗ ВИМОГ

1.1 Анкетування

Для отримання більш точних даних під час анкетування рекомендується повторити процес декілька разів. Це дозволяє збільшити достовірність результатів і знизити вплив випадкових факторів. Кількість повторень анкетування визначається методами математичної статистики, з урахуванням заданого рівня статистичної похибки.

Дані питання були використанні для того, що найчастіше використовують користувачі під час замовлення їжі, або те, що їх найбільше цікавить стосовно цього.

Таблиця 1. Приклади питань для збору вимог

№	Тип питання	Опис питання	Приклад відповіді на питання
1	Які основні функції чат-бота ви очікуєте?	Питання спрямоване на визначення очікуваних функцій та можливостей чат-бота.	Я б хотів, щоб чат-бот надавав інформацію про стан мого замовлення та відповідав на мої запитання щодо доставки.
2	Які види інформації про замовлення ви б хотіли переглядати через чат-бота?	Питання спрямоване на визначення конкретної інформації, яку користувачі хотіли б отримувати щодо свого замовлення.	Мене цікавить перегляд статусу замовлення, очікуваний час доставки та деталі про вміст замовлення
3	Як ви б хотіли взаємодіяти з чат-ботом? Через текстові повідомлення або кнопки?	Питання спрямоване на визначення уподобань користувачів стосовно способу комунікації з чат-ботом.	Мені зручно використовувати кнопки для швидкого вибору опцій та текстові повідомлення для детальнішого взаємодії.

Таблиця 1. Продовження

4	Як часто ви плануєте використовувати чат-бота для перегляду стану замовлення та отримання відповідей на запитання?	Питання спрямоване на визначення частоти використання чат-бота користувачами.	Я буду використовувати чат-бота кожного разу, коли потрібно перевірити статус мого замовлення або отримати інформацію.
5	Які додаткові можливості або функції ви б хотіли бачити у чат-боті?	Питання спрямоване на визначення додаткових можливостей чат-бота, які користувачі знаходять цікавими або корисними.	Було б чудово мати можливість змінити деталі замовлення через чат-бот або отримувати рекомендації стосовно нових страв
6	Як би ви оцінили зручність інтерфейсу чат-бота в Телеграмі?	Питання спрямоване на оцінку зручності та виконавчої якості інтерфейсу чат-бота в Телеграмі.	Інтерфейс досить зручний. Всі необхідні функції доступні легко і швидко через кнопки та текстові команди.
7	Як часто ви очікуєте отримувати відповіді на питання через чат-бота?	Питання спрямоване на визначення очікуваної швидкості та регулярності відповідей від чат-бота.	Я сподіваюся отримувати відповіді від чат-бота протягом декількох хвилин після задання запитання

Таблиця 1. Продовження

8	Як би ви оцінили розуміння та точність відповідей чат-бота на ваші питання?	Питання спрямоване на оцінку рівня розуміння та точності відповідей чат-бота на запитання користувачів.	Чат-бот зазвичай розуміє мої запитання та надає точні відповіді, але іноді можуть бути незрозумілість чи неточності.
9	Як би ви оцінили загальний рівень задоволення від використання чат-бота?	Питання спрямоване на оцінку загального задоволення користувачів від використання чат-бота.	Я задоволений використанням чат-бота. Він значно спрощує процес отримання інформації про замовлення
10	Які покращення або зміни ви запропонували б для поліпшення чат-бота?	Питання спрямоване на отримання пропозицій та ідей від користувачів щодо можливих покращень чат-бота.	Я б запропонував включити можливість створення замовлення безпосередньо через чат-бот або додати опцію отримання сповіщень про зміни статусу замовлення.
11	Як би ви оцінили швидкість виконання запитів чат-ботом?	Питання спрямоване на оцінку швидкості відповідей та обробки запитів користувачів чат-ботом.	Чат-бот відповідає на мої запити досить швидко, без значних затримок.
12	Чи було вам зрозуміло, як користуватися чат-ботом з першого разу?	Питання спрямоване на оцінку простоти та зрозумілості інтерфейсу чат-бота для нових користувачів.	На першому етапі мені знадобилось трохи часу, щоб зрозуміти, як користуватися чат-ботом, але потім все стало зрозуміліше

Таблиця 1. Продовження

13	Чи мали ви достатньо інформації про стан вашого замовлення після використання чат-бота?	Питання спрямоване на оцінку інформативності та повноти наданої інформації про статус замовлення через чат-бота.	Чат-бот надав достатньо деталей про стан мого замовлення, я був інформований про кожну зміну.
14	Чи були у вас труднощі чи проблеми під час використання чат-бота?	Питання спрямоване на виявлення можливих проблем або труднощів, з якими користувачі можуть зіткнутися під час використання чат-бота.	У мене виникли деякі проблеми з розумінням деяких команд чат-бота, але після звернення до довідкової інформації все стало зрозуміліше
15	Як ви оцінюєте якість відповідей, які надавав чат-бот?	Питання спрямоване на оцінку якості та релевантності відповідей, наданих чат-ботом на питання користувачів.	Більшість відповідей чат-бота були достатньо точними та допомагали мені з розумінням стану мого замовлення.

1.2 Аналіз конкурентів

Дослідження та аналіз існуючих українських чат-ботів, а також їхніх особливостей та переваг, варто розглядати як частину підготовки до створення нового чат-бота для перегляду інформації про замовлення та відповіді на питання користувачів.

Для початку, рекомендується провести аналіз популярних українських чат-ботів, які надають подібні функції. Здійснення дослідження можна розпочати зі збору інформації про такі чат-боти, їхні основні функції, інтерфейс, способи взаємодії з користувачами та можливості, які вони надають.

Далі слід вивчити переваги та недоліки цих чат-ботів. Для цього можна провести аналіз відгуків та оглядів користувачів, які вже скористалися такими сервісами. Основним завданням буде виділення основних переваг, які користувачі цінують, такі як зручність використання, швидкість та точність відповідей, доступність інформації, а також визначення недоліків, наприклад, недостатньо детальної або незрозумілої інформації, складнощів у взаємодії або повільної реакції [17].

Крім того, варто вивчити тенденції розвитку чат-ботів в Україні. Це можна зробити шляхом дослідження останніх новин та інновацій у сфері розробки чат-ботів, а також вивчення відгуків та рейтингів наукових статей, досліджень та конференцій, пов'язаних з цією темою. Таке дослідження допоможе виявити нові можливості та потенціал для покращення чат-ботів українською мовою.

Проведемо аналіз трьох чат-ботів для перегляду стану замовлення. Ось опис та аналіз кожного з них:

1. ChatBotX:
    - Опис: ChatBotX є чат-ботом, який надає користувачам можливість переглядати стан своїх замовлень.
    - Переваги:
        - Інтуїтивний і легкий у використанні інтерфейс.
        - Швидкий доступ до інформації про замовлення.
        - Можливість отримувати сповіщення про зміни статусу замовлення.
    - Недоліки:
        - Обмежений функціонал: забезпечує лише перегляд стану замовлення, без можливості взаємодії з ботом.
        - Обмежена підтримка мов: працює лише на англійській мові.
2. OrderTrackBot:

- Опис: OrderTrackBot - це чат-бот, призначений для користувачів, які хочуть переглянути стан своїх замовлень та отримати інформацію про їх доставку.
- Переваги:
    - Можливість перегляду детальної інформації про замовлення.
    - Інтерактивний інтерфейс з можливістю взаємодії з ботом.
    - Підтримка різних мов.
- Недоліки:
    - Відсутність сповіщень про зміни статусу замовлення.
    - Обмежений функціонал поза переглядом стану замовлення.

3. TrackMeBot:
    - Опис: TrackMeBot - це чат-бот, який допомагає користувачам переглядати стан їх замовлень та відстежувати їх доставку.
    - Переваги:
        - Зручний і простий у використанні інтерфейс.
        - Можливість перегляду стану замовлення в режимі реального часу.
        - Сповіщення про зміни статусу замовлення.
    - Недоліки:
        - Відсутність можливості взаємодії з ботом або задавати питання.
        - Обмежена функціональність поза переглядом стану замовлення.

Ці три чат-боти надають можливість користувачам переглядати стан їх замовлень. Кожен з них має свої переваги та недоліки. Вибір між ними залежить від конкретних потреб та вимог користувачів.

Висновки до першого розділу

У даному розділі було розглянуто питання збору та аналізу вимог. Ми провели пошук та огляд існуючих програмних аналогів, що дозволило виявити їх обмеження та визначити недоліки, які ми плануємо уникнути у нашому додатку. Після проведеного аналізу вимог ми чітко сформулювали завдання та напрямки розробки нашого чат-боту. Отримані вимоги стануть основою для подальшої реалізації та розвитку додатку з метою досягнення поставлених цілей та задоволення потреб користувачів.

2 ПРОЄКТУВАННЯ

2.1. Зовнішнє проєктування

2.1.1 Функціональне призначення

Функціональне призначення чат-бота відображає основну дію, яку він виконує для користувачів. Основна функція ПЗ нашого боту полягає у наданні користувачам можливості переглядати стан своїх замовлень та отримувати відповіді на питання, пов'язані з їх замовленнями. Це означає, що користувачі зможуть отримати доступ до інформації про свої замовлення, таку як статус, деталі, час доставки та інше, безпосередньо через чат-інтерфейс бота [1].

Розпишемо детально функціональне призначення нашого чат-бота:

1. Перегляд стану замовлення:
    - Користувачі зможуть запитувати статус своїх замовлень.
    - Бот надасть користувачам актуальну інформацію про стан замовлень, таку як прийнято, в процесі обробки, відправлено, доставлено і т.д.
    - Користувачі зможуть отримати детальні дані про кожне замовлення, включаючи номер замовлення, товари, кількість, ціну тощо.
2. Відповіді на питання користувачів:
    - Користувачі зможуть задавати питання про свої замовлення та отримувати відповіді від бота.
    - Бот буде надавати користувачам необхідну інформацію, яка відповідає на їх питання, наприклад, щодо зміни статусу, очікуваного часу доставки, способів оплати тощо.
3. Взаємодія з користувачем:
    - Бот буде забезпечувати зручний та простий у використанні інтерфейс для взаємодії з користувачами.

- Користувачі зможуть взаємодіяти з ботом, виконуючи різні команди або запити, що стосуються їх замовлень.

2.1.2 Експлуатаційне призначення

Експлуатаційне призначення нашого чат-бота полягає в наданні нематеріальної вигоди користувачам при його використанні. Приклади таких вигод включають:

1. Збільшення ефективності: Використання чат-бота для перегляду стану замовлення дозволяє користувачам отримувати швидкий доступ до необхідної інформації без необхідності контактувати з оператором чи перевіряти статус замовлення на інших платформах. Це зменшує часові витрати на отримання інформації та робить процес більш зручним та швидким.

2. Підвищення точності та надійності: Чат-бот може надати точну та актуальну інформацію про стан замовлення, оскільки вона базується на оновленій базі даних. Це дозволяє уникнути можливих помилок, що можуть виникати при людському втручанні або передачі інформації [7].

3. Зручність та доступність: Чат-бот доступний 24/7 і може бути використаний користувачами в будь-який зручний для них час. Він надає можливість отримувати потрібну інформацію без потреби в дзвінках чи особистій зустрічі, що робить процес більш зручним та ефективним.

4. Скорочення витрат: Використання чат-бота для перегляду стану замовлення дозволяє зменшити необхідність в інших ресурсах, таких як оператори чи додатковий персонал для надання інформації про замовлення. Це може привести до економії часу та грошей для бізнесу.

Таким чином, експлуатаційне призначення нашого чат-бота полягає в підвищенні точності і зручності отримання інформації про замовлення, зменшенні часових витрат та економії ресурсів, що приносить нематеріальну вигоду користувачам.

2.1.3 Функціональні вимоги

1. Перегляд стану замовлення:
    - Надання користувачеві інформації про поточний статус його замовлення.
    - Відображення деталей замовлення, таких як товари, кількість, ціна тощо.
    - Забезпечення можливості відстеження місцезнаходження замовлення (якщо є відповідна функціональність).
2. Відповіді на питання користувачів:
    - Аналіз запитів користувачів та надання відповідей на питання щодо замовлення.
    - Підтримка мовних команд та запитів різної форми.
    - Забезпечення точності та зрозумілості відповідей.
3. Взаємодія з користувачем:
    - Забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для взаємодії з користувачем.
    - Надання можливості користувачеві задавати питання, вводити команди або вибирати опції з меню.
    - Забезпечення можливості коригування та оновлення інформації про замовлення.

2.1.4 Вхідні дані

Вхідними даними програмного додатку є:

- програмний код на мові програмування C#;
- файли з розширенням .cs із програмним кодом на мові програмування C# (кожен файл відповідає за окрему сутність);

2.1.5 Вихідні дані

1. Інформація про замовлення: Відповідь на запит користувача, яка містить деталі замовлення, такі як статус, товари, кількість, ціна тощо.
2. Відповідь на питання користувача: Відповідь на конкретне запитання користувача щодо замовлення, наприклад, інформація про доставку, оплату або знижки.

Формат даних:

1. Запит користувача: Текстовий рядок відповідного формату.
2. Інформація про замовлення: Текстовий рядок або структуровані дані у форматі, що містять необхідну інформацію про замовлення.
3. Відповідь на питання користувача: Текстовий рядок або структуровані дані у форматі, що містять відповідь на питання користувача.

Область допустимих значень:

1. Запит користувача: Будь-який текстовий рядок, в якому користувач може висловити свій запит або питання.
2. Ідентифікатор користувача: Унікальний ідентифікатор, який може бути обмежений конкретними форматованими правилами або обмеженнями системи.
3. Інформація про замовлення: Значення та формат полів інформації про замовлення визначаються конкретною системою та можуть мати обмеження щодо типів даних або максимальних значень.
4. Відповідь на питання користувача: Значення та формат відповідей на питання можуть бути текстовими рядками, числовими значеннями, датами або іншими допустимими форматами відповідно до поставленого питання та системи.

2.1.6 Опис зовнішнього інформаційного середовища, в якому буде експлуатуватися ПЗ.

Зовнішнє інформаційне середовище, в якому буде експлуатуватися ПЗ нашого чат-боту для перегляду стану замовлення та відповіді на питання користувачів, включає наступні елементи:

1. Телеграм-месенджер: Чат-бот буде взаємодіяти з користувачами через платформу Телеграм, яка надає зручний інтерфейс для обміну повідомленнями.

2. Користувачі: Люди, які використовують чат-бот, будуть взаємодіяти з ним шляхом відправки текстових повідомлень та отримання відповідей.

3. Інформаційні об'єкти:

- Замовлення: Інформація про замовлення, включаючи статус, товари, кількість, ціну тощо.

- Питання користувачів: Повідомлення від користувачів, що містять запитання, коментарі або іншу інформацію, пов'язану з замовленням.

4. Телеграм API: Інтерфейс програмування застосунків (API), наданий Телеграм-платформою, який дозволяє отримувати та надсилати повідомлення через Телеграм-месенджер.

5. Серверна інфраструктура: Сервери та інфраструктура, необхідні для розміщення та функціонування чат-боту [12].

6. Додаткові програми та сервіси: Для коректного функціонування проектованого ПЗ можуть знадобитися додаткові програми або сервіси, наприклад, база даних для збереження інформації про замовлення, система аутентифікації користувачів або сервіси доставки повідомлень.

Отже, зовнішнє інформаційне середовище включає Телеграм-месенджер як основний канал взаємодії, користувачів як активних учасників, а також різні інформаційні об'єкти, які передаються та обробляються чат-ботом. Для функціонування ПЗ можуть знадобитися додаткові програми та сервіси, а також використовуватися Телеграм API для комунікації з платформою Телеграм.

2.2. Внутрішнє проєктування

Для реалізації вищезазначених функціональних характеристик програмного додатку, було спроектовано та розроблено класи і форми, які відповідають за користувацький інтерфейс користувача та визначення відповідності тексту програми графічному представленню алгоритму.

2.2.1 Проектування класів системи

Розроблені класи відобразимо на діаграмах класів, рис. 2.1-2.2

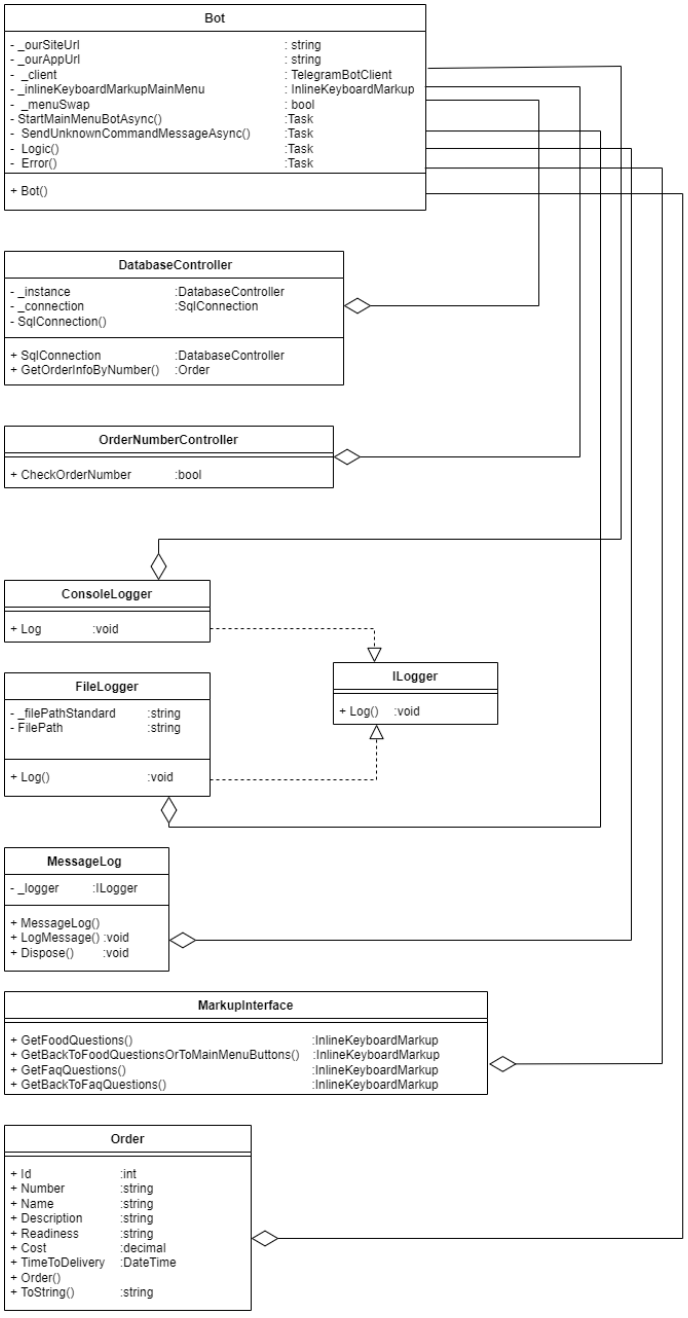


Рисунок 2.1 – Діаграма класів рівня логіки

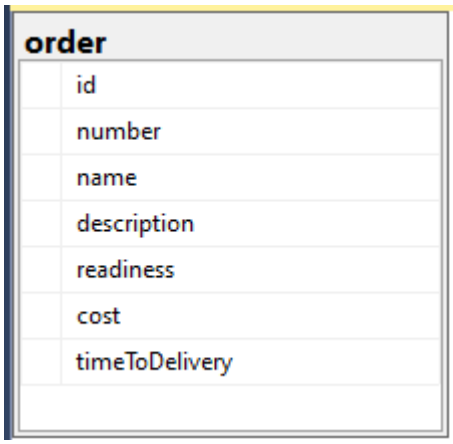


Рисунок 2.2 – Діаграма класів рівня бази даних для замовлень

Для розроблених діаграм класів було виділено наступні рівні:

- логіки – включає класи, що відповідають за логіку програми, а саме за визначення відповідності тексту програми графічному представленню алгоритмів;

- бази даних – включає базу даних програми. Схема бази даних визначає структуру таблиць та їх зв'язки. Вона описує поля кожної таблиці, їх типи даних та обмеження.

Розглянемо кожний розроблений клас більш детально:

- Bot – клас рівня логіки, який відповідає за клієнт чат-боту та нові повідомлення (кліки на кнопки) від користувача;
- DatabaseController – контроллер, для роботи з базою даних, який оснований на патерні Singleton;
- OrderNumberController – контроллер, який перевіряє правильність вводу номеру заказу користувачем;
- ConsoleLogger – логгер, який предназначений для того, щоб виводити помилки до консолі;
- FileLogger – логгер, який предназначений для того, щоб записувати помилки до файла;

- ILogger – інтерфейс, який предназначений для використання Dependency Injection у логгеров;
- MessageLog – сервіс, який приймає різні логери і викликає їх метод логування;
- MarkupInterface – клас для отримання клавіатур для користувача;
- Order – клас "Заказ" для опису інформації щодо замволення.

Спроектований інтерфейс користувача відображено на рис. 2.3.

Інтерфейс користувача містить наступні елементи:

- віконце з текстовою інформацією;
- кнопка навігації назад;
- кнопка навігації до чату;
- кнопка навігації до головного меню;
- кнопки навігації між пунктами меню;

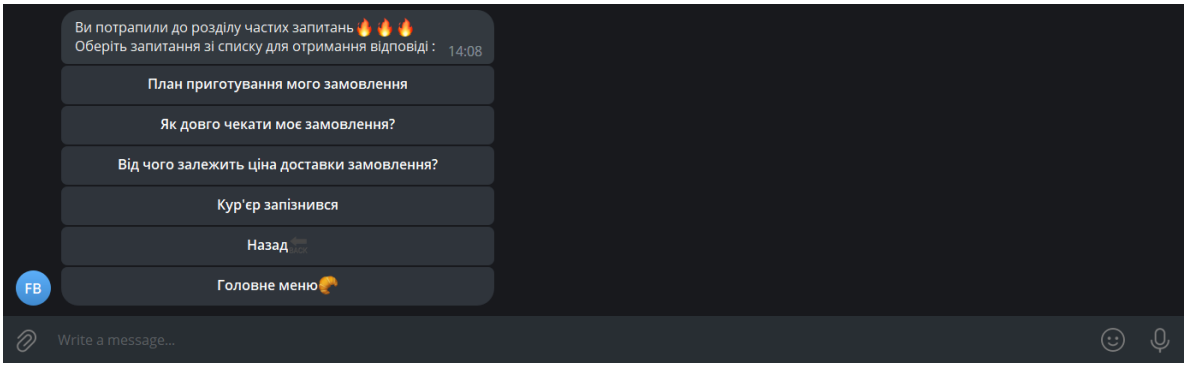


Рисунок 2.3– Ескіз екранної форми

Взаємодія користувача з графічним інтерфейсом відображена на діаграмі станів (рис. 2.4)

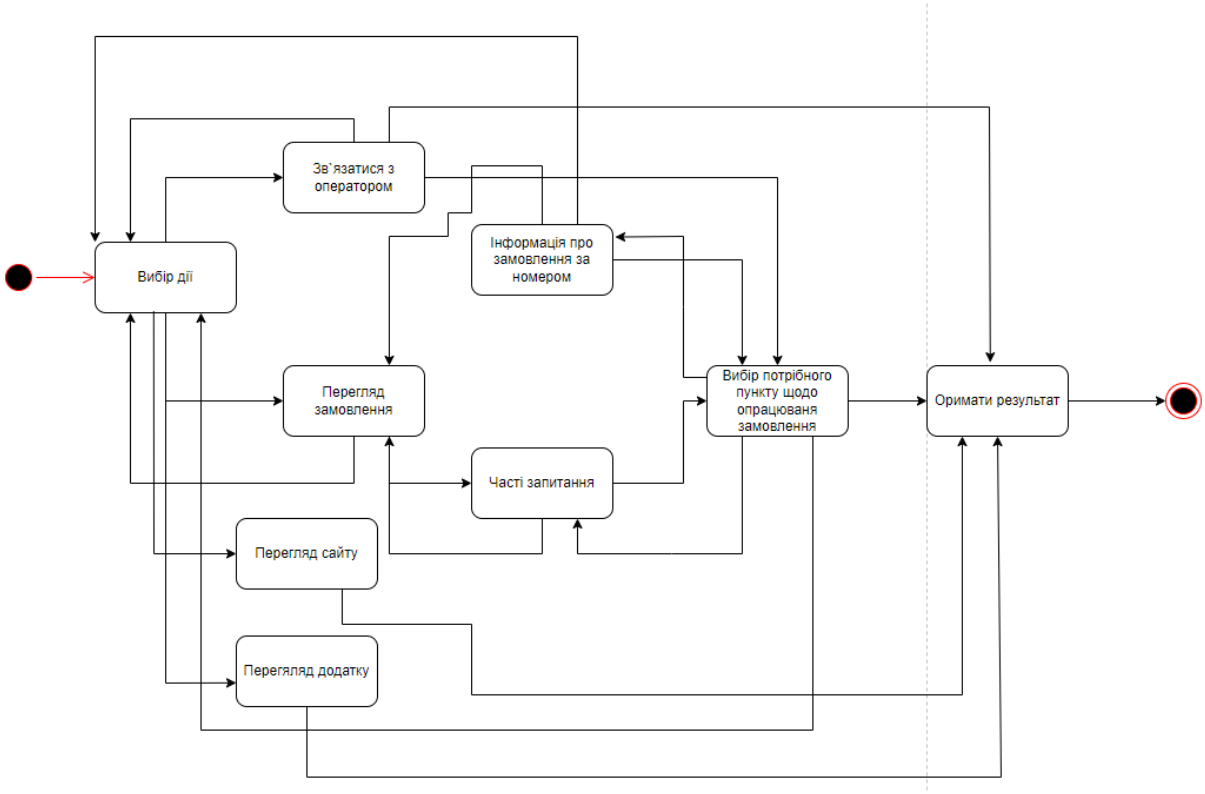


Рисунок 3.4 – Діаграма станів взаємодії користувача з вікном меню

Висновки до другого розділу

Підсумовуючи висновки, можна зазначити наступне:

1. Функціональні вимоги, сформульовані для розробки чат-боту для перегляду стану замовлення їжі та відповіді на питання користувачів, чітко визначають основні дії та функціональності, які має реалізувати програмний засіб. Вони включають можливість перегляду статусу замовлення, отримання детальної інформації про замовлення, надання відповідей на типові запитання та інші функції, необхідні для задоволення потреб користувачів.

2. Виконання функціональних вимог передбачає використання певних методів та алгоритмів, що можуть бути описані та пояснені з посиланням на літературні джерела. Це забезпечить якість та ефективність розробленого програмного засобу.

3. Вхідні та вихідні дані, які використовуються в чат-боті, включають назви, суть та формат інформаційних об'єктів, а також область допустимих значень. Це

допоможе у взаємодії з іншими програмами та системами, а також забезпечить правильну обробку та виведення інформації для користувачів.

4. Зовнішнє інформаційне середовище, в якому експлуатуватиметься програмний засіб, включає канали вводу-виводу та інформаційні об'єкти, з якими він взаємодіє. Для успішної роботи чат-бота можуть знадобитись додаткові програми або засоби, що забезпечать його функціонування.

Висновки, зроблені на основі функціональних вимог, вхідних та вихідних даних та опису зовнішнього інформаційного середовища, допоможуть нам в подальшій розробці та реалізації програмного засобу. Вони визначають напрямки діяльності, ключові функції та цілі, які ми плануємо досягти з метою задоволення потреб наших користувачів та забезпечення ефективної роботи чат-боту.

3 РОЗРОБКА ПРОГРАМИ

Використання мови програмування C# є високоякісним варіантом для розробки такого типу системи. C# володіє вражаючою потужністю та розширюваністю, і має широкий спектр функціональностей та бібліотек, що значно спрощують процес розробки [3].

У процесі розробки було проведено визначення суті методів та конструювання алгоритмів, необхідних для досягнення поставлених функціональних вимог. Методи були розроблені з урахуванням основних завдань бота, таких як перегляд стану замовлення та надання відповідей на питання користувачів.

Алгоритми були створені з метою оптимального та ефективного виконання потрібних операцій. Вони орієнтовані на забезпечення швидкості та точності обробки даних, а також забезпечення зручного та інтуїтивно зрозумілого взаємодії з користувачем.

3.1 Метод покрокової деталізації

Метод покрокової деталізації є важливим етапом у процесі розробки програмного забезпечення. Його головна мета - розкрити деталі функцій та процесів системи, розбити їх на окремі кроки та деталізувати кожен крок для більш точного розуміння та виконання.

Цей метод дозволяє розбити складні функції на менші, більш керовані елементи, що спрощує розробку та підтримку програмного забезпечення. Покрокова деталізація дозволяє команді розробників чітко спланувати послідовність дій, визначити необхідні ресурси та забезпечити ефективну взаємодію між різними компонентами системи.

Переваги методу покрокової деталізації включають:

1. Краще розуміння функціональних вимог: Деталізація кроків дозволяє зрозуміти всі етапи виконання функцій системи, що полегшує розробку та тестування.

2. Ефективна робота команди: Розбиття функцій на окремі кроки сприяє розділенню роботи між розробниками та забезпечує більшу ефективність колективної роботи.
3. Легша налагодження та підтримка: Детальна документація кроків спрощує процес налагодження та підтримки системи, дозволяючи швидше виявляти та виправляти помилки.
4. Зручне взаємодія зі замовником: Покрокова деталізація дозволяє замовнику краще зрозуміти, як саме будуть виконуватися його вимоги та як буде функціонувати система [11].

Запишемо алгоритм за методом покрокової деталізації. Перепишемо подану задачу у вигляді специфікації вхідних та вихідних даних.

Вхід: запит користувача – текстовий рядок, який містить питання від користувача.

Вихід: відповідь чат-бота – текстовий рядок, який містить відповідь на запит користувача.

Розіб’ємо задачу на підзадачі:

1. Прийняття запиту:

   1.1. Отримання текстового повідомлення від користувача.

   1.2. Аналіз введеного тексту для визначення наміру користувача.

2. Розпізнавання команд:

   2.1. Порівняння наміру користувача зі списком підтримуваних команд.

   2.2. Визначення необхідної дії на основі розпізнаної команди.

3. Обробка команди:

   3.1. Виконання відповідних операцій, пов'язаних з командою.

   3.2. Передача необхідних даних та параметрів для виконання дії.

4. Генерація відповіді:

4.1. Створення текстової відповіді на основі виконаної дії.

4.2. Форматування відповіді для відображення користувачу.

5. Відправка відповіді:

5.1. Надсилання сформованої відповіді користувачеві через відповідний канал.

6. Завершення запиту:

6.1. Очікування нового запиту від користувача або завершення роботи чат-бота.

Цей алгоритм дозволяє розкрити всі етапи обробки запиту від користувача до надсилання відповіді. Він дозволяє систематично розбити задачу на підзадачі, що спрощує процес розробки та забезпечує чіткість та структурованість роботи.

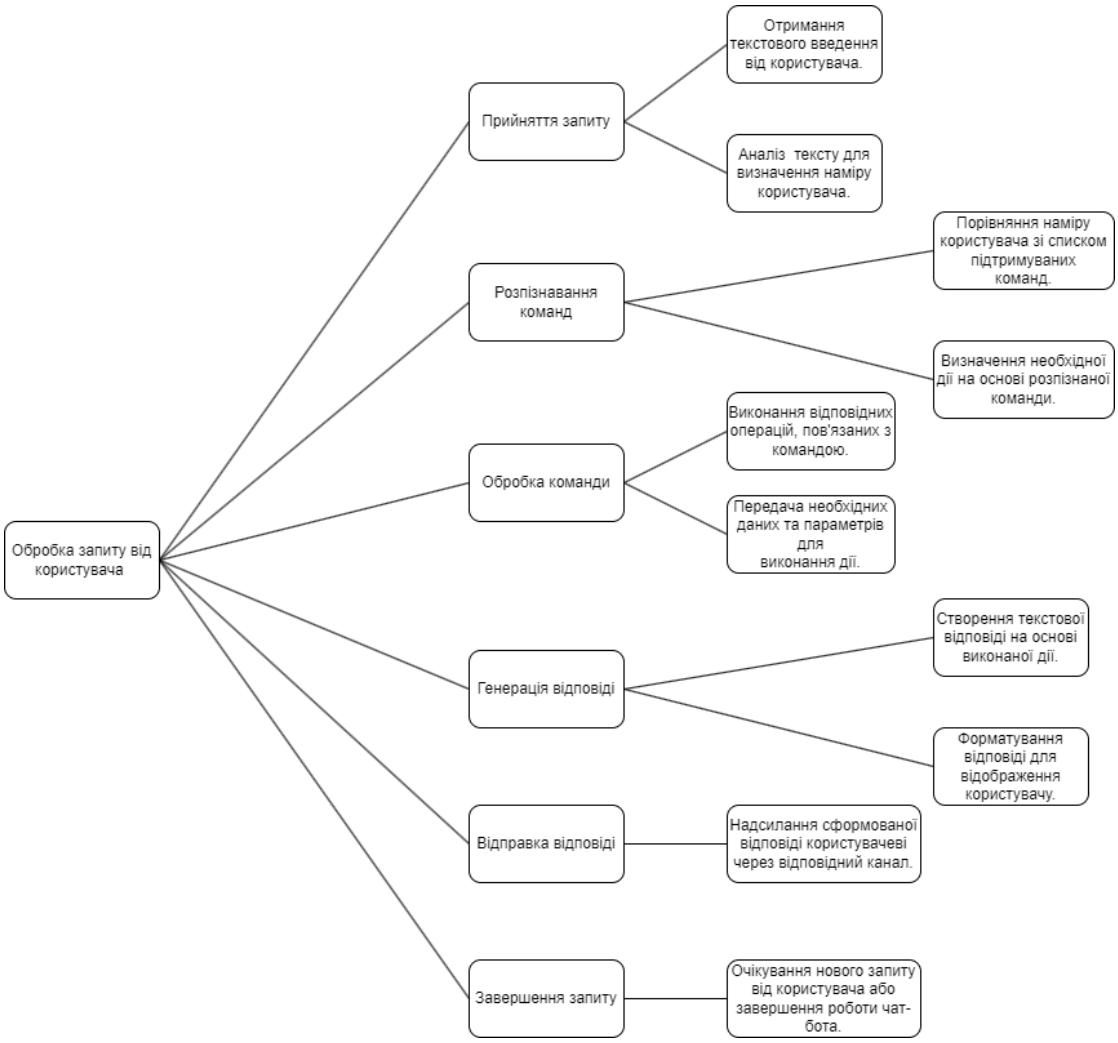
Кожна з під задач є значно простішою, ніж задача в цілому. Підзадачі можуть бути реалізовані окремими функціями (наприклад, 2.1, 2.2., 2.3), або бути частиною однієї 3.1, 3.2.

Цю ж задачу представимо деревом рис. 3.1 та таблицею каскадів (табл. 2). У каскадному представлені зробимо на один рівень деталізації більше, щоб наблизитися до безпосередньої реалізації.

Таблиця 2 – Каскадне представлення алгоритму

Задача	Підзадачі	
Обробка запиту ввід користувача	Прийняття запиту	Отримання текстового введення від користувача.
		Аналіз тексту для визначення наміру користувача.
	Розпізнавання команд	Порівняння наміру користувача зі списком підтримуваних команд.
		Визначення необхідної дії на основі розпізнаної команди.

	Обробка команди	Виконання відповідних операцій, пов'язаних з командою.
		Передача необхідних даних та параметрів для виконання дії.
	Генерація відповіді	Створення текстової відповіді на основі виконаної дії.
		Форматування відповіді для відображення користувачу.
	Відправка відповіді	Надсилання сформованої відповіді користувачеві через відповідний канал.
	Завершення запиту	Очікування нового запиту від користувача або завершення роботи чат-бота.



3.1 Дерево каскадів

Використання блок-схем в даній роботі має кілька переваг і може значно сприяти процесу розробки чат-боту. Ось декілька причин, чому використання блок-схем може бути корисним:

1. Візуалізація алгоритму: Блок-схеми надають можливість візуально представити послідовність дій та логіку програмного алгоритму. Вони допомагають зрозуміти структуру програми та взаємозв'язок між різними кроками обробки запиту [6].

2. Структурованість та чіткість: Блок-схеми дозволяють послідовно розбити алгоритм на окремі блоки та зв'язати їх зрозумілими стрілками. Це допомагає зберегти структурованість та чіткість в розробці програмного забезпечення.

3. Виявлення помилок та оптимізація: Використання блок-схем дає можливість аналізувати алгоритм на ранніх етапах розробки та виявляти потенційні помилки або неефективні рішення. Це сприяє оптимізації алгоритму та покращенню його продуктивності.
4. Комунікація з командою: Блок-схеми можуть служити засобом комунікації з іншими учасниками проекту. Вони дозволяють легко передавати та обговорювати логіку програмного алгоритму, спрощуючи спілкування та розуміння між членами команди.
5. Документація проекту: Блок-схеми можуть бути використані як складова частина документації проекту. Вони надають зрозуміле візуальне представлення алгоритму та можуть служити довідковим матеріалом для майбутніх розробників або інших зацікавлених сторін.

Загалом, використання блок-схем у розробці чат-боту допомагає зрозуміти, організувати та візуалізувати логіку роботи програмного алгоритму, спрощує комунікацію та сприяє оптимізації процесу розробки.

Визначмо види алгоритмів та їх реалізацій:

- лінійний алгоритм;
- алгоритм розгалуження;
- циклічний алгоритм з постумовою;
- вкладеність алгоритмів розгалуження;

До представлених тем не було додано лістинг програми, оскільки дані алгоритми були складені природньою мовою для більшого розуміння користувачам, які не знаються на програмних мовах.

1. Лінійний алгоритм (рис.3.1):

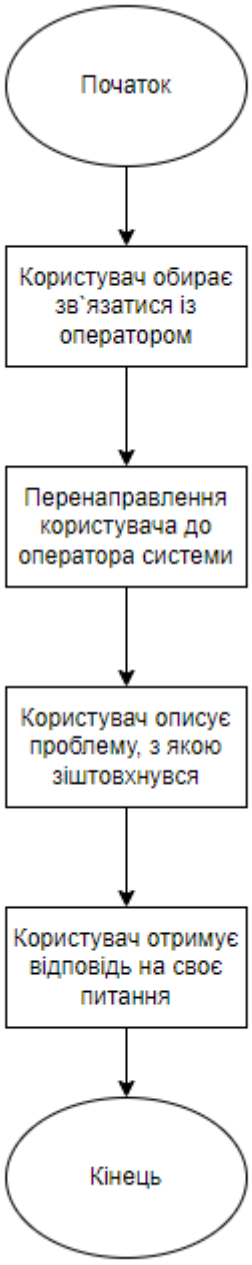


Рисунок 3.2 – Блок-схема лінійного алгоритму

2. Алгоритм розгалуження (рис 3.3):

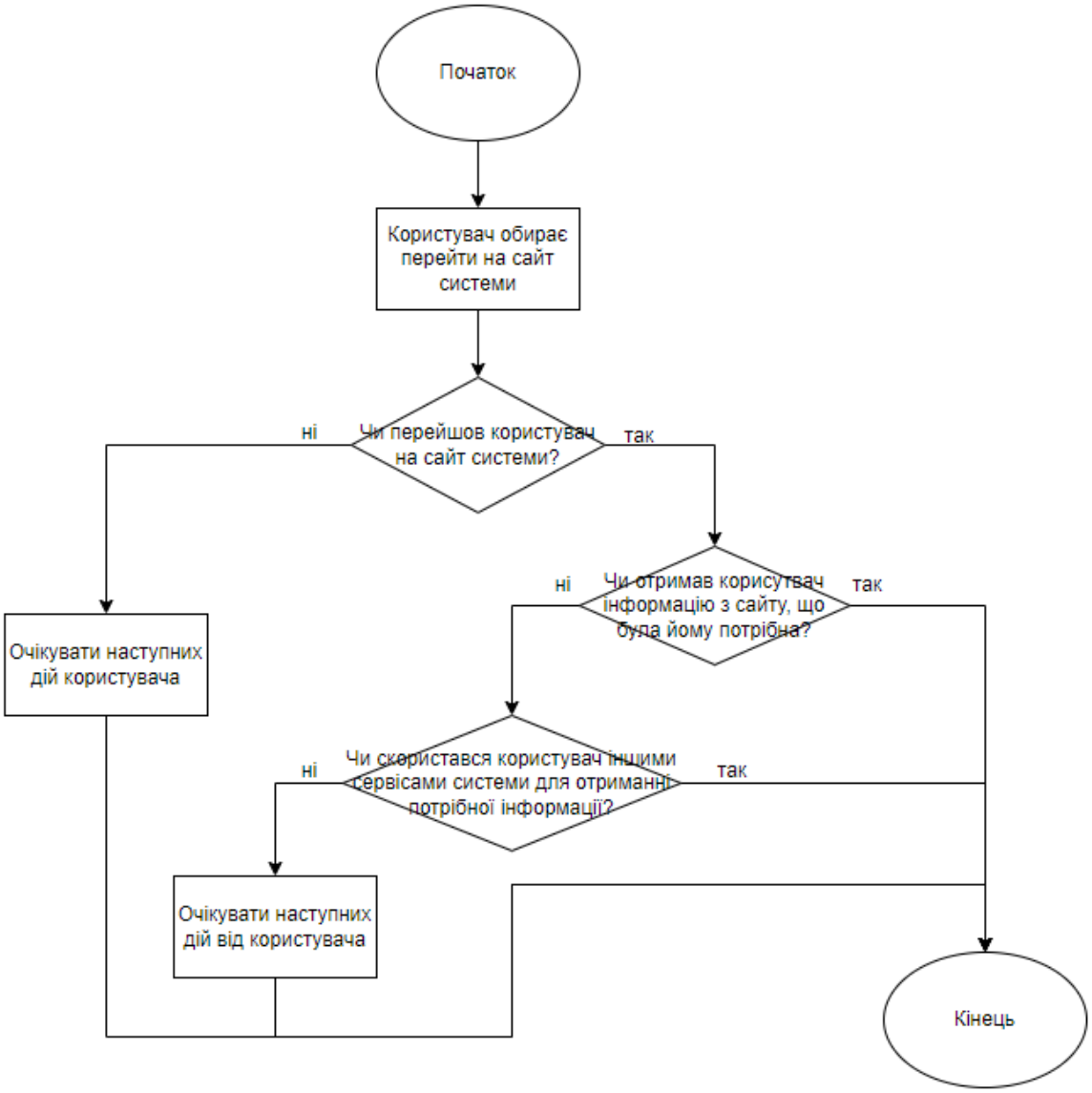


Рисунок 3.3 – Блок-схема алгоритму розгалуження з вкладеними конструкціями

3. Циклічний алгоритм з пост умовою (рис 3.4):

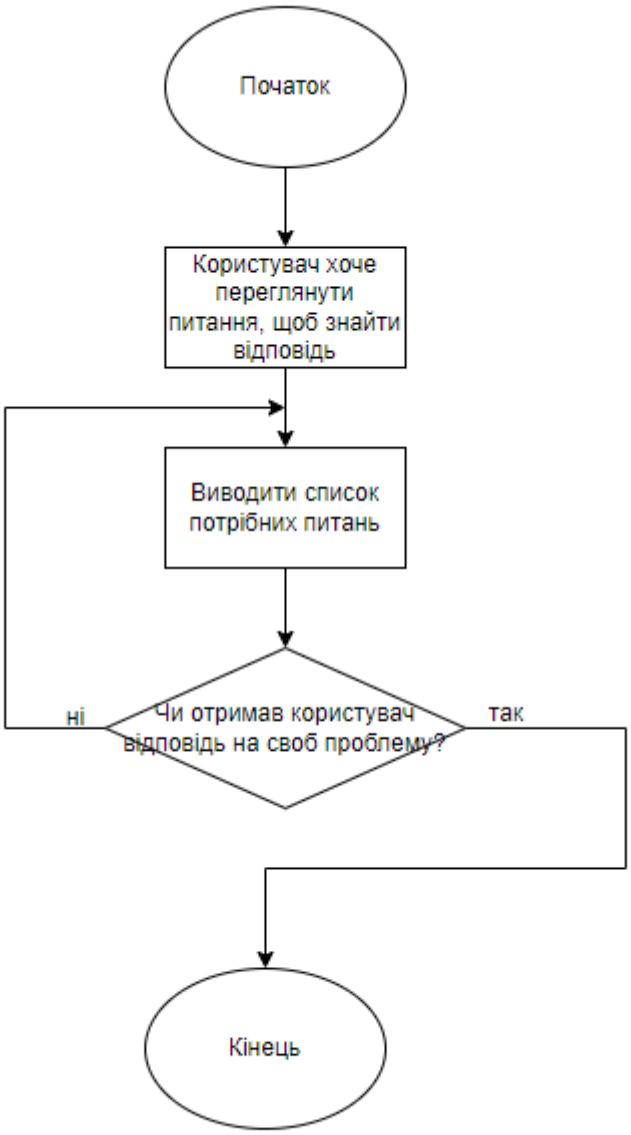


Рисунок 3.4 – Блок-схема циклічного алгоритму з пост умовою

Висновок до третього розділу

Під час виконання третього розділу пояснювальної записки було проведено визначення сутності методів та конструювання алгоритмів. Метод покрокової деталізації дозволив нам докладно розкрити кожен крок виконання програми, розглянути взаємозв'язки та відповідальність модулів системи. Цей підхід значно підвищив чіткість та структурованість програмної реалізації.

В результаті виконання цього розділу ми зробили важливий крок у програмному проектуванні. Ми чітко зрозуміли вибір мови програмування, методів та алгоритмів, що будуть використовуватися в розробці чат-боту. Це дозволяє нам мати чітке уявлення про те, як буде функціонувати наша система та які кроки потрібно зробити для досягнення поставлених цілей.

Загальною метою цього розділу було створення бази для подальшої реалізації та розвитку чат-боту. Визначення сутності методів та конструювання алгоритмів допомагає нам стати краще підготовленими для наступних етапів розробки, забезпечуючи чітке розуміння програмного проекту та його функціональних можливостей.

4 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

Тестування - це процес перевірки та оцінки програмного продукту з метою виявлення помилок, недоліків і переконання в його відповідності вимогам та очікуванням користувачів. Це важлива складова частина розробки програмного додатку, оскільки дозволяє забезпечити якість та надійність програми перед її випуском в експлуатацію.

Тестування має декілька цілей. Перш за все, воно допомагає виявити помилки та недоліки в програмі, що можуть призвести до некоректної роботи системи або незадоволення користувачів. Тестування також допомагає перевірити, чи відповідає програма вимогам та специфікаціям, що були поставлені перед нею. Крім того, воно дозволяє перевірити продуктивність та швидкодію програми, а також виявити можливі проблеми з безпекою.

В результаті тестування програми ми отримаємо декілька важливих переваг. По-перше, ми зможемо виявити та виправити помилки та недоліки, що підвищить якість та надійність програми. По-друге, тестування дозволить перевірити відповідність програми вимогам та очікуванням користувачів, що допоможе задовольнити їх потреби. Крім того, тестування допоможе виявити й вирішити проблеми з продуктивністю та безпекою програми, забезпечуючи оптимальне функціонування системи [8].

Загалом, тестування є важливим етапом розробки програмного додатку, який допомагає забезпечити якість, надійність та відповідність програми вимогам. Відповідне тестування дозволяє знизити ризик помилок та проблем підчас експлуатації, забезпечує задоволення потреб користувачів та сприяє успішному впровадженню програмного додатку.

1. Впевненість у коректній роботі: Тестування допомагає перевірити, чи розпізнає чат-бот правильно команди користувачів і чи надає вірні відповіді на запити. Це дозволяє забезпечити впевненість у тому, що чат-бот працює згідно з очікуваннями користувачів.

2. Виявлення помилок та вдосконалення: Тестування допомагає виявити потенційні помилки в роботі чат-бота, такі як неправильні відповіді, несправний аналізатор команд або проблеми зі зберіганням даних. Це дозволяє вносити відповідні виправлення та вдосконалення, забезпечуючи високу якість роботи чат-бота

3. Перевірка взаємодії з іншими системами: Чат-бот може взаємодіяти з іншими програмними компонентами або зовнішніми системами, такими як бази даних або API. Тестування дозволяє перевірити правильність цієї взаємодії і впевнитись, що чат-бот коректно обмінюється даними з іншими компонентами системи.

4. Валідація вимог: Тестування допомагає перевірити, чи відповідає чат-бот вимогам, які були поставлені перед ним. Це дозволяє переконатись, що функціональність, продуктивність та інші характеристики чат-бота відповідають встановленим стандартам і очікуванням.

5. Оптимізація та швидкодія: Тестування може допомогти виявити проблеми з продуктивністю чат-бота, такі як затримки у відповіді або неефективне використання ресурсів. Це дозволяє виявити можливі причини й удосконалити роботу чат-бота, забезпечуючи оптимальну швидкодію та використання ресурсів.

Загалом, завдяки тестуванню нашого чат-бота ми отримаємо впевненість у його якості та надійності, забезпечимо правильну взаємодію з користувачами та іншими компонентами системи, а також зможемо виявити та виправити помилки для покращення функціональності та продуктивності. Тестування є важливою складовою процесу розробки чат-бота та гарантує його успішне функціонування.

Таблиця 3 – Тестування та налагодження.

Номер тесту	Опис ситуації	Очікуваний результат
1	Запуск системи для початку тестування (рис. 4.1)	Виведення головного меню.

Таблиця 3 – Продовження

2	Вибір пункту «Наш сайт» (рис.4.2)	Виведення посилання на сайт.
3	Вибір пункту «Наш додаток» (рис.4.3)	Виведення посилання на додаток.
4	Вибір пункту «Моє замовлення» (рис.4.4)	Виведення меню щодо замовлення клієнта.
5	Вибір пункту «Інформація про замовлення за номером» (рис.4.5)	Виведення повідомлення про подальші дії.
6	Введення номеру замовлення: 1234567890 (рис.4.6)	Виведення повної інформації про замовлення.
7	Вибір пункту «Часті запитання» (рис.4.7)	Виведення списку питань, які задають найчастіше.
8	Вибір пункту «План приготування мого замовлення» (рис.4.8)	Виведення повної інформації щодо цього.
9	Вибір пункту «Як довго чекати моє замовлення?» (рис.4.9)	Виведення повної інформації щодо цього.
10	Вибір пункту «Від чого залежить ціна доставки замовлення?» (рис.4.10)	Виведення повної інформації щодо цього.
11	Вибір пункту «Кур’єр запізнився» (рис.4.11)	Виведення повної інформації щодо цього.

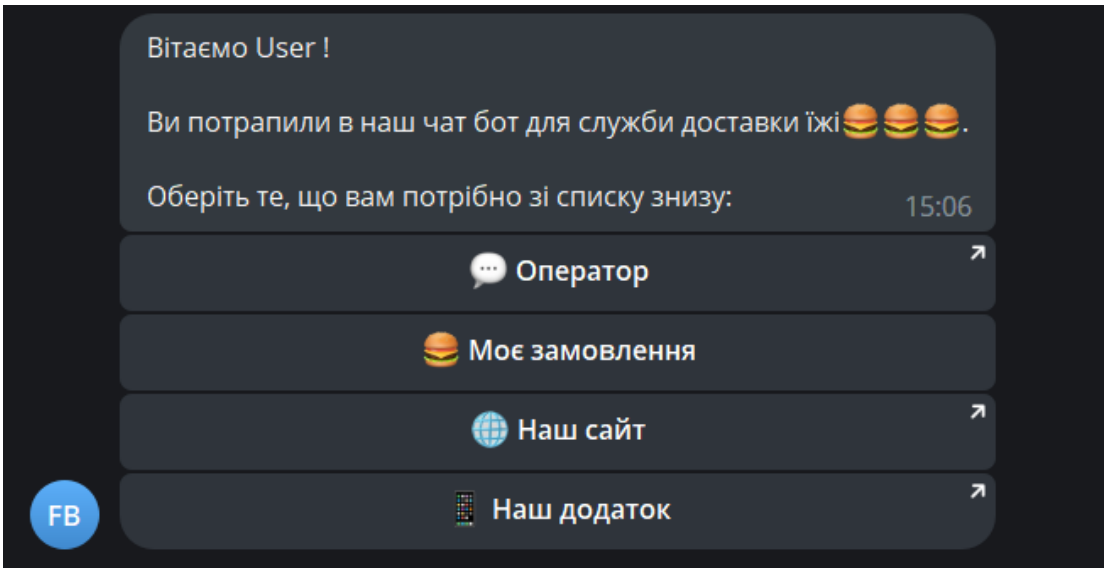


Рисунок 4.1 – Тест 1.

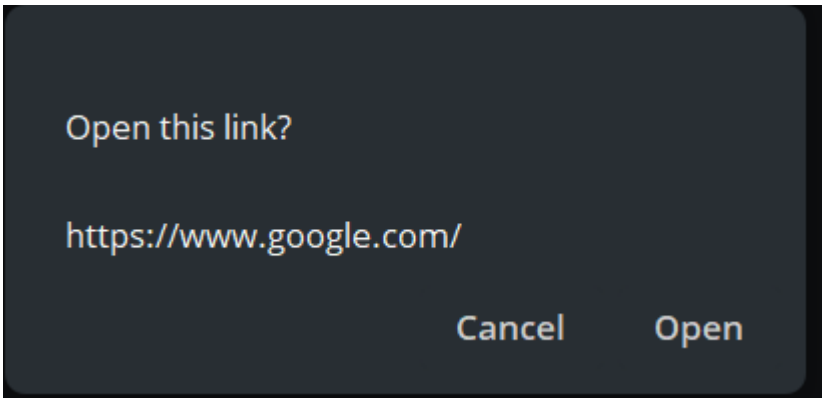


Рисунок 4.2 – Тест 2.

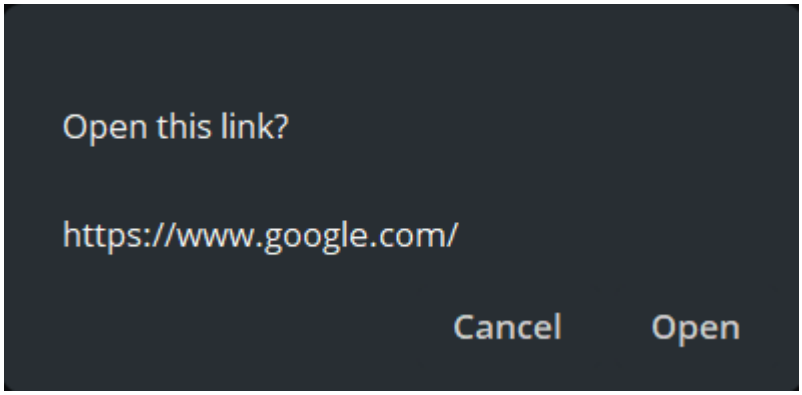


Рисунок 4.3 – Тест 3

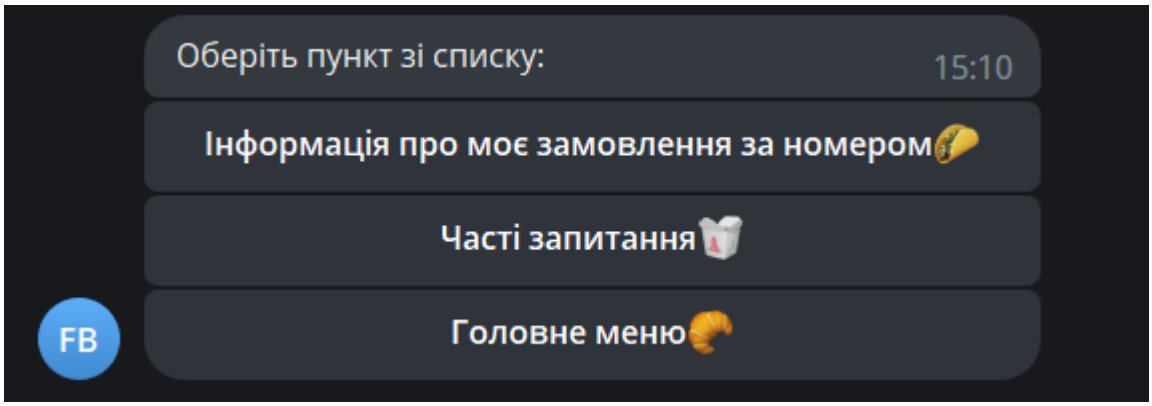


Рисунок 4.4 – Тест 4

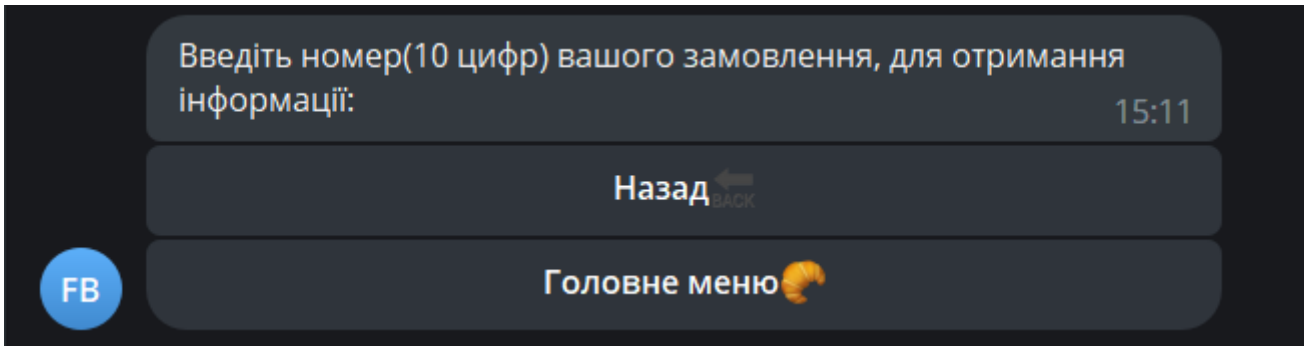


Рисунок 4.5 – Тест 5

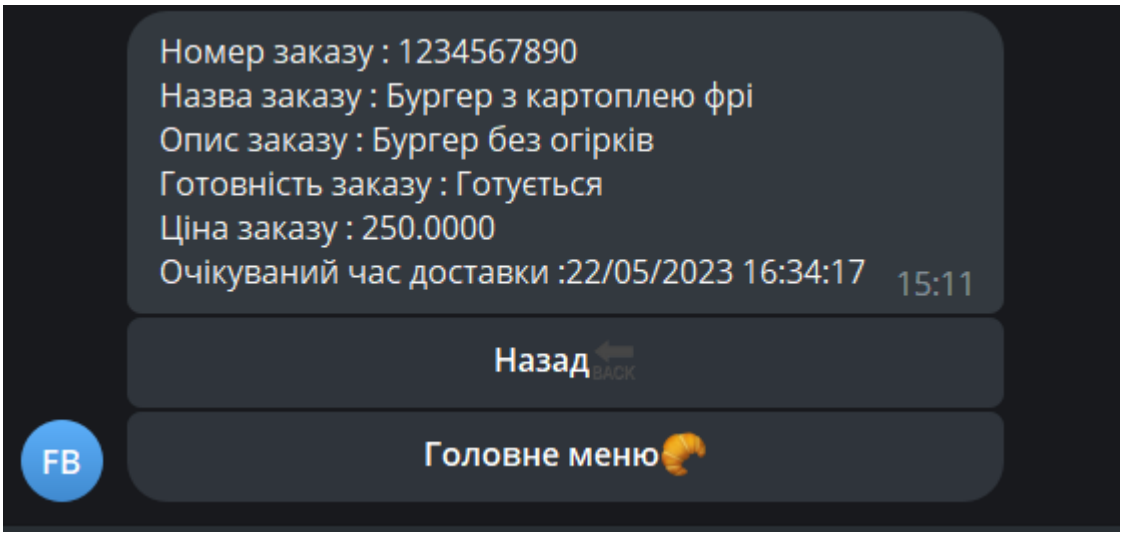


Рисунок 4.6 – Тест 6

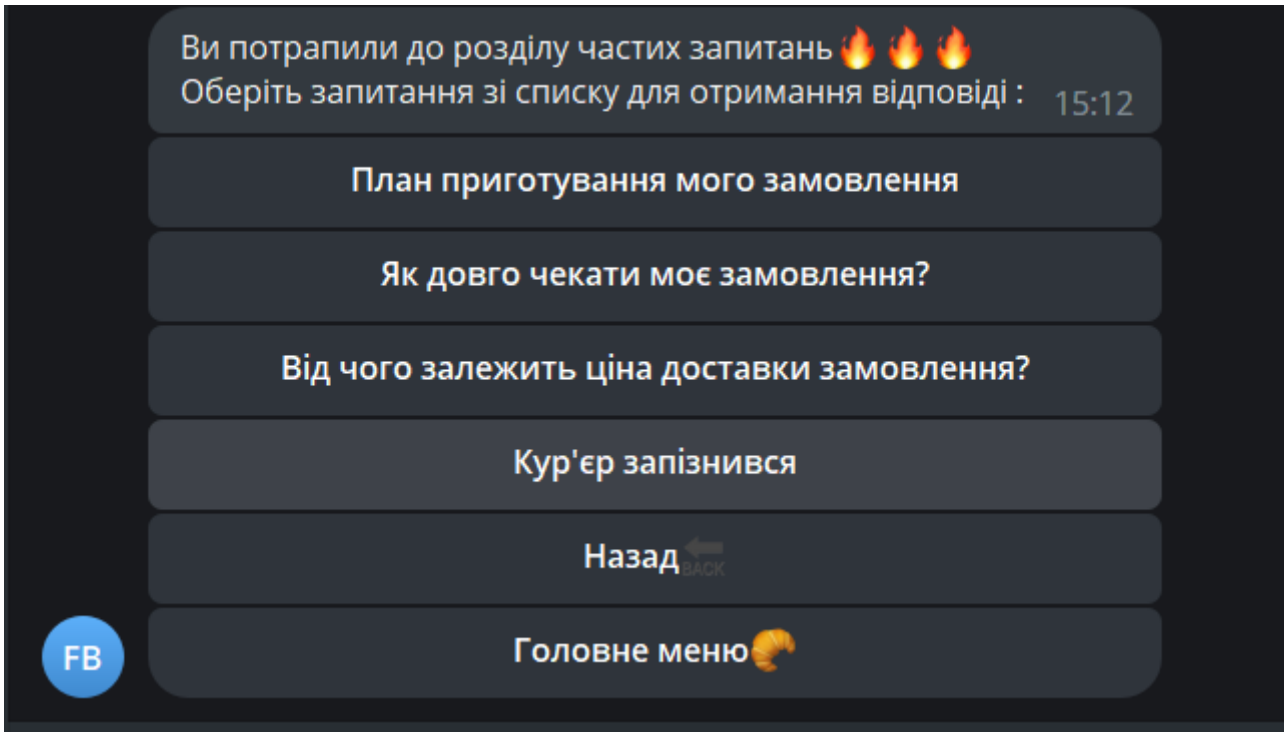


Рисунок 4.7 – Тест 7

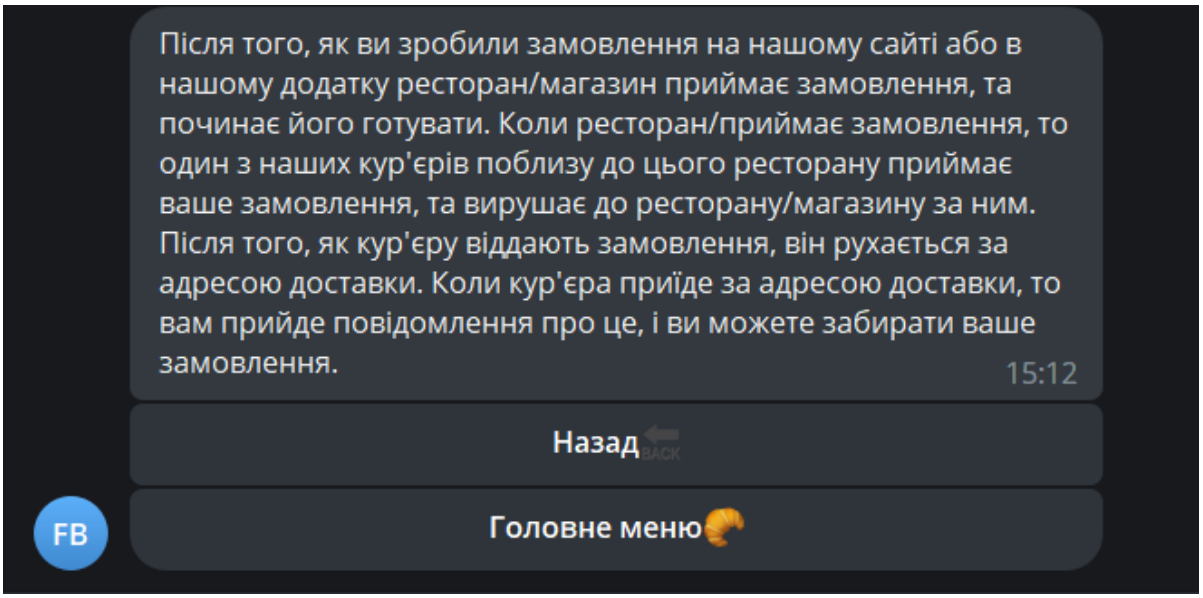


Рисунок 4.8 – Тест 8

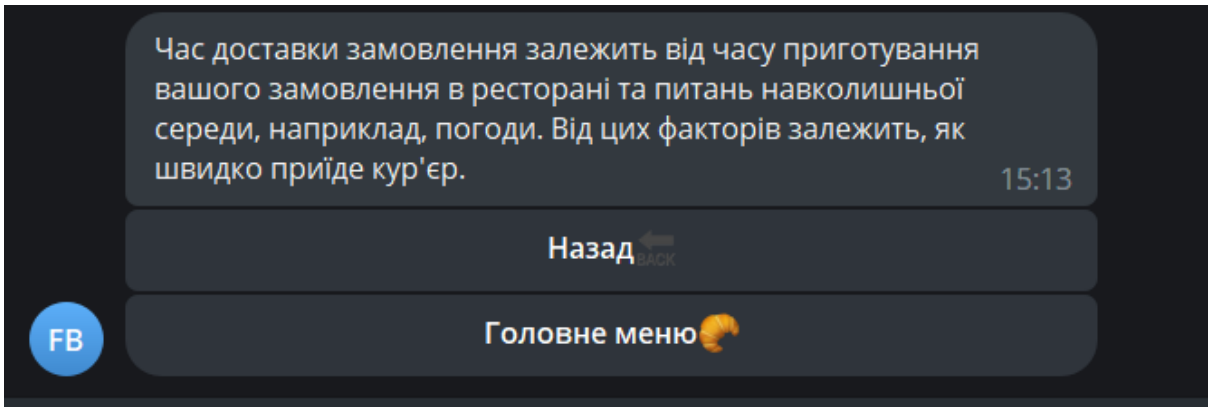


Рисунок 4.9 – Тест 9

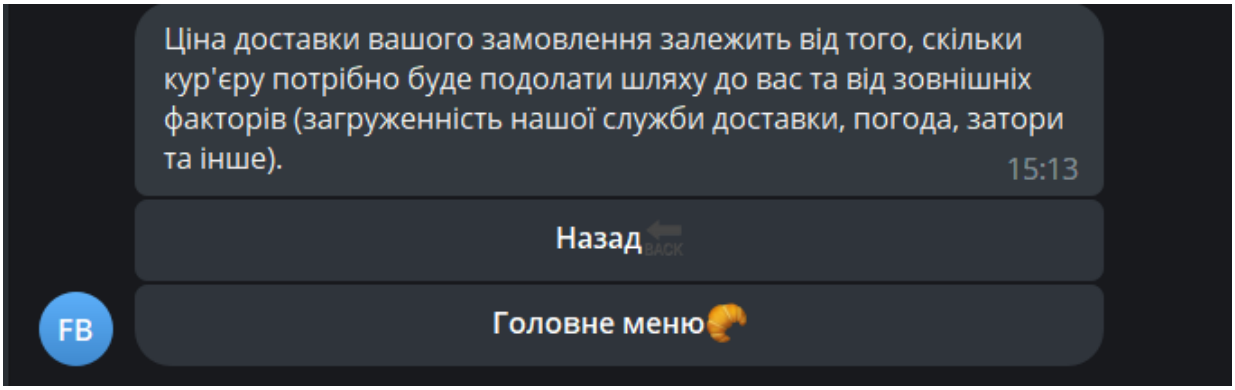


Рисунок 4.10 – Тест 10

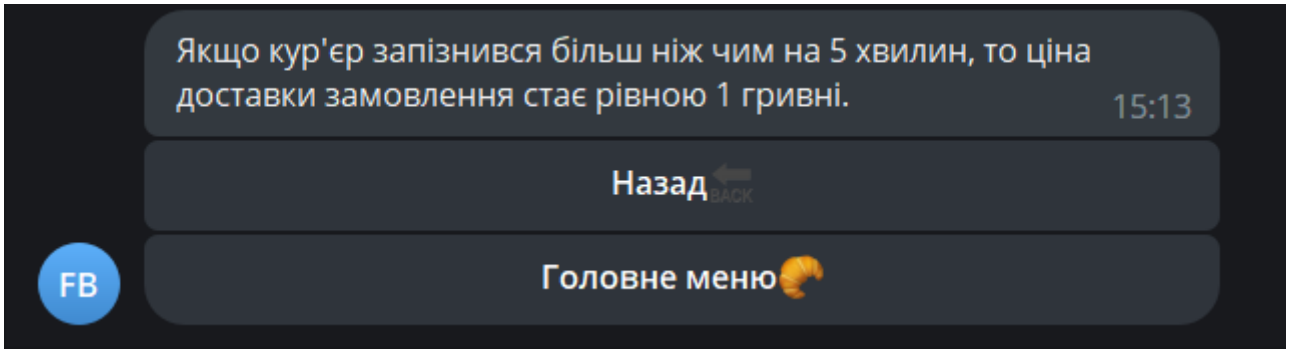


Рисунок 4.11 – Тест 11

Функціональність кнопки "Оператор" полягає у прив'язці до акаунту оператора, який відповідатиме за надання підтримки користувачам через чат. При натисканні на цю кнопку, відбувається відкриття чату з оператором, який буде доступний для спілкування. Для використання цієї функціональності компанія, яка

використовує додаток, повинна створити базу даних операторів, в якій будуть відображені дані про операторів та їх доступність для надання підтримки.

Також компанія повинна буде надати розробнику системи дійсні посилання на додаток та сайт компанії.

Тестування системи проводилося на різних пристроях з різними операційними системами з метою переконатись, що програма працює правильно та надає коректні результати на будь-якому з них. Основна мета тестування полягала в перевірці незалежності від типу пристрою та операційної системи та забезпеченні стабільної та надійної роботи програми для задоволення потреб користувачів.

Висновки до розділу 4

Після проведення тестування програми чат-боту ми зробили декілька висновків і визначень, які стануть основою для подальшої реалізації та вдосконалення системи. Основні висновки з тестування включають:

1. Функціональність: Під час тестування було підтверджено, що програма успішно виконує основні функції, такі як прийняття запитів користувачів, розпізнавання команд, обробка даних та генерація відповідей.

2. Сумісність: Тестування на різних пристроях з різними операційними системами підтвердило, що програма працює стабільно і надійно незалежно від типу пристрою або операційної системи.

3. Коректність: Програма відповідає на запити користувачів коректно та надає правильні результати. Тестування допомогло виявити та виправити можливі помилки та недоліки у роботі системи.

4. Надійність: Під час тестування було перевірено стабільність та надійність програми. Система працює без збоїв і забезпечує гарну продуктивність навіть при великій кількості одночасних запитів.

В цілому, тестування дозволило підтвердити працездатність програмного додатку чат-боту, виявити та виправити можливі проблеми та допомогти забезпечити якість та надійність системи. Отримані результати тестування стануть основою для подальшої розробки та вдосконалення додатку з метою досягнення поставлених цілей та задоволення потреб користувачів.

АНАЛІЗ ТА ВИСНОВКИ

В цій роботі ми провели детальний аналіз та проектування програмного додатку чат-боту для їжі. Пройшли кроки визначення вимог, вибору мови програмування та конструювання алгоритмів. Ми використали метод покрокової деталізації для розбиття задачі на підзадачі та визначення їх взаємозв'язків. Також, ми дослідили важливість використання блок-схем та їх переваги при проектуванні програм [8].

Ми провели тестування програми, щоб перевірити її функціональність, сумісність, коректність та надійність. Це дозволило нам підтвердити працездатність системи, виявити та виправити можливі помилки та недоліки, а також забезпечити якість та надійність додатку.

Наше дослідження також включало визначення бази даних для системи, що включає інформацію про операторів та їх зайнятість.

У результаті цієї роботи ми досягли таких важливих висновків:

1. Використання методу покрокової деталізації дозволило нам чітко визначити кроки виконання програми та взаємозв'язки між ними, що сприяє структурованості та зрозумілості програмного коду.

2. Вибір мови програмування C# був обґрунтованим рішенням, оскільки ця мова має потужний набір функціональностей та багато вбудованих бібліотек, що полегшують розробку.

3. Використання блок-схем під час проектування програми допомагає уявити весь процес виконання та виявити можливі проблемні моменти заздалегідь.

4. Тестування є важливою складовою процесу розробки програмного додатку, оскільки воно допомагає виявити та виправити помилки, забезпечити правильну роботу системи та задовольнити потреби користувачів.

5. База даних операторів є необхідною для ефективного використання функціоналу чат-боту та відстеження зайнятості операторів, що полегшує управління процесом обробки запитів користувачів.

Загальною метою нашої роботи було створення функціонального та надійного програмного додатку чат-боту для їжі. Завдяки проведеним дослідженням, проектуванню та тестуванню, ми змогли досягти цієї мети, забезпечивши структуровану та ефективну реалізацію системи, яка здатна ефективно взаємодіяти з користувачами та забезпечувати якісне обслуговування [3].

Крім того, завдяки нашим дослідженням та розробці, ми успішно реалізували чат-бота для їжі, який має наступні важливі особливості:

1. Здатність приймати текстові повідомлення від користувачів і розпізнавати їх намір. Це дозволяє користувачам взаємодіяти з чат-ботом шляхом надсилання запитів, замовлення їжі або отримання необхідної інформації.

2. Розпізнавання команд та виконання відповідних дій. Чат-бот може розпізнати команди користувача, замовити вам їжу, перевірити статус доставки, надати інформацію про ціни тощо.

3. Обробка команд та передача необхідних даних. Чат-бот виконує потрібні операції, пов'язані з отриманими командами, і передає відповідні дані для виконання замовлень або надання інформації.

4. Генерація текстових відповідей користувачам. Чат-бот створює зрозумілі та зрозумілі відповіді на запити користувачів, які включають деталі про замовлення, підтвердження операцій, вартість послуг та іншу важливу інформацію.

5. Відправлення відповідей через відповідний канал. Чат-бот надсилає сформовані відповіді користувачам через зручний канал зв'язку, такий як текстові повідомлення, електронна пошта або інші доступні канали.

Завдяки цим функціональностям і розробці чат-бота ми створили інноваційний та ефективний інструмент для отримання необхідної інформації.

Список використаних джерел

1. Microsoft Developer Network (MSDN). [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/en-us/

2. OpenAI Documentation. [Електронний ресурс]. Режим доступу: https://docs.openai.com/

3. Chatbots Magazine. [Електронний ресурс]. Режим доступу: https://chatbotsmagazine.com/

4. Towards Data Science. [Електронний ресурс]. Режим доступу: https://towardsdatascience.com/

5. Medium. [Електронний ресурс]. Режим доступу: https://medium.com/

6. Stack Overflow. [Електронний ресурс]. Режим доступу: https://stackoverflow.com/

7. ProgrammableWeb. [Електронний ресурс]. Режим доступу: https://www.programmableweb.com/

8. Botpress Blog. [Електронний ресурс]. Режим доступу: https://botpress.com/blog

9. Chatbot News Daily. [Електронний ресурс]. Режим доступу: https://chatbotnewsdaily.com/

10. IBM Developer. [Електронний ресурс]. Режим доступу: https://developer.ibm.com/

11. Dialogflow Documentation. [Електронний ресурс]. Режим доступу: https://cloud.google.com/dialogflow/docs/

12. Rasa Documentation. [Електронний ресурс]. Режим доступу: https://rasa.com/docs/

13. Botpress Documentation. [Електронний ресурс]. Режим доступу: https://botpress.com/docs/

14. Botsociety Blog. [Електронний ресурс]. Режим доступу: https://botsociety.io/blog/

15. HubSpot Blog. [Електронний ресурс]. Режим доступу: https://blog.hubspot.com/

16. ChatGrape Blog. [Електронний ресурс]. Режим доступу: https://www.chatgrape.com/blog/

17. SnatchBot Blog. [Електронний ресурс]. Режим доступу: https://snatchbot.me/blog/

18. Botanalytics Blog. [Електронний ресурс]. Режим доступу: https://botanalytics.co/blog

19. BotList. [Електронний ресурс]. Режим доступу: https://botlist.co/

20. Botfuel Blog. [Електронний ресурс]. Режим доступу: https://www.botfuel.io/blog/

21. "Розробка чат-ботів на платформі Facebook Messenger" [Електронний ресурс] / ProProgrammer. Режим доступу: https://proprogrammer.net/development-of-chat-bots-on-the-facebook-messenger-platform/

22. "Чат-боти в бізнесі: від ідеї до реалізації" [Електронний ресурс] / SoftServe. Режим доступу: https://www.slideshare.net/SoftServeInc/chat-81868280

23. "Розробка чат-ботів з використанням Dialogflow" [Електронний ресурс] / Ciklum. Режим доступу: https://www.ciklum.com.ua/blog/developing-chat-bots-with-dialogflow/

24. "Чат-боти на платформі Microsoft Bot Framework" [Електронний ресурс] / EPAM. Режим доступу: https://www.epam.ua/about/blog/chat-bots-on-the-microsoft-bot-framework

25. "Розробка інтелектуальних чат-ботів з використанням Microsoft Bot Framework" [Електронний ресурс] / IT-Ebooks. Режим доступу: https://it-ebooks.info/book/2423/

26. "Створення чат-ботів з використанням Python та Flask" [Електронний ресурс] / DOU. Режим доступу: https://dou.ua/lenta/articles/creating-chatbots-with-python-and-flask/

27. "Розробка чат-бота на Node.js з використанням Botpress" [Електронний ресурс] / GlobalLogic. Режим доступу: https://www.globallogic.com/ua/blog/how-to-develop-a-chatbot-with-botpress/

Додатки

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського державного університету науки і технологій

Анатолій РАДКЕВИЧ

18.02.22

МОБІЛЬНІЙ ДОДАТОК “ЧАТ-БОТ ЇЖІ”

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850. 01223-01-ЛЗ

Представники

підприємства-розробника

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

18.06.23

Керівник розробки

Вадим АНДРЮЩЕНКО

18.06.23

Виконавець

Максим ТАРАН

18.06.23

Норм-контролер

Олена КУРОП’ЯТНИК

18.06.23

2023

ЗАТВЕРДЖЕНО

44165850. 01223-01-ЛЗ

МОБІЛЬНІЙ ДОДАТОК “ ЧАТ-БОТ ЇЖІ ”

Технічне завдання

44165850.01223-01

Листів 16

2023

ЗМІСТ

1 ВВЕДЕННЯ

Мультиплатформенний додаток «Чат-бот їжі», що розробляється, має виконувати функції боту для відповіді на питання користувачів (замовників їжі) для полегшення їх роботи.

В залежності від вибраних користувачем питань або функції системи - мультиплатформенний додаток має генерувати різні відповіді на питання, які націлені на вирішення проблеми користувача.

Користувач, використовуючи цей мультиплатформенний додаток, може обрати питання із списку запропонованих та, за своїм бажанням, може написати адміністратору служби доставки їжі. Після отримання відповіді на поставлене питання, користувач може повернутись на початковий екран або розпочати пошук відповіді на свою проблему з самого початку, зберігаючи усі вибрані до цього запити.

Програмний продукт є безкоштовний, що робить його більш цікавим на фоні інших аналогів, але варто розуміти, що це рекламний продукт.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 08.12.21 №77ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи - “ЧАТ-БОТ ЇЖІ”. Керівник - доцент Андрющенко В. О.

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – програмний продукт має генерувати відповіді на питання замовників їжі для отримання оперативної відповіді на питання, пов'язані з роботою, такими як стан замовлення їжі.

Експлуатаційне призначення – додаток розроблений з метою надання зручного та швидкого способу отримання необхідної інформації для замовників їжі. Він призначений для полегшення комунікації та замовників їжі, допомагаючи їм вирішувати питання, що виникають під час замовлення потрібних страв.

4 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

4.1 Вимоги до функціональних характеристик

- Отримання інформації про стан їжі: Чат-бот повинен мати можливість отримувати дані про стан їжі, такі як статус замовлення, час приготування, інформацію про доставку та інші деталі. Це дозволить користувачам отримувати актуальну інформацію про свої замовлення.
- Відповіді на запитання користувачів: Чат-бот повинен мати можливість відповідати на питання користувачів щодо стану їжі, її складу, калорійності, алергенів та інших деталей. Відповіді повинні бути точними, зрозумілими та інформативними.
- Забезпечення спілкування: Чат-бот повинен мати можливість взаємодіяти з користувачами, приймати їх запитання та надавати на них відповіді. Взаємодія повинна бути зрозумілою, швидкою та ефективною.

Вимоги до вхідних даних:

- Запитання користувачів: Чат-бот повинен приймати запитання від користувачів щодо стану їжі, меню, акцій та інших пов'язаних питань. Вхідні дані повинні бути зрозумілі та коректні.
- Ідентифікаційні дані: При необхідності, чат-бот може вимагати введення ідентифікаційних даних користувача, таких як ім'я, номер телефону або адреса доставки.
- Інформація про стан їжі: В разі, якщо користувач запитує про конкретний стан їжі або замовлення, чат-бот може потребувати введення номера замовлення або інших ідентифікаторів для точної інформації.

Вхідні дані повинні бути зрозумілими, правильними та відповідати вимогам, що ставляться до функціональності та цілей додатку чат-боту таксі.

Вимоги до вихідних даних:

Вихідними даними є:

- Інформація про стан їжі: Чат-бот повинен забезпечувати вихідні дані щодо стану їжі, які можуть включати інформацію про підтвердження замовлення, статус приготування, час доставки та інші деталі, які допоможуть користувачеві бути в курсі стану свого замовлення.
- Відповіді на питання користувачів: Чат-бот повинен забезпечувати зрозумілі та точні вихідні дані відповідно до запитань, які користувачі ставлять. Це можуть бути інформація про акції, розклад роботи, контактні дані, або будь-яка інша інформація, що стосується послуги їжі.
- Інформація про меню та страви: Чат-бот повинен надавати вихідні дані про доступне меню та страви. Це можуть бути назви страв, опис, ціни, фотографії та інша інформація, яка допоможе користувачеві зробити вибір та замовити бажану їжу.
- Інформація про доставку: Якщо відповідний, чат-бот повинен надавати вихідні дані щодо доставки, такі як статус доставки, інформацію про кур'єра, очікуваний час прибуття та інші важливі деталі, які допоможуть користувачам відстежувати та контролювати процес доставки.

4.2 Вимоги до надійності

Вимоги до надійності наступні:

- при оновленні даних забезпечити показ відповідних повідомлень про стан роботи програми та результати виконання операцій;
- наявність архівної копії тексту програми на зовнішньому носії;
- наявність резервної копії бази даних на зовнішньому носії.

4.3 Вимоги експлуатації

Програмний продукт повинен використовуватись у приміщеннях які відповідають умовам роботи ЕОМ, а саме мають такі кліматичні, санітарні та гігієнічні умови, які відповідають ДНАОП 0.00-1.13-99 (див. табл. 1).

Таблиця 1. Кліматичні умови

Пора року	Категорія робіт згідно з ГОСТ 12.01-005-88	Температура повітря, град.С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
		Оптимальна	Оптимальна	Оптимальна
Холодна	легка-1-а	22-24	40-60	0,1
	легка-1-б	21-23	40-60	0,1
Тепла	легка-1-а	23-25	40-60	0,1
	легка-1-б	22-24	40-60	0,2

Працювати з програмою може людина, що має навички роботи з мобільними пристроями та ознайомлена з керівництвом користувача програмного продукту.

4.4 Вимоги до складу та параметрів технічних засобів

Продукт, що розробляється повинен використовуватись на пристроях, що мають наступні характеристики:

- діагональ екрану – 5.5;
- роздільна здатність дисплею – HD (1280x720);
- оперативна пам’ять – 2 ГБ;
- вбудована пам’ять – 16 ГБ;
- операційна система – ANDROID;
- частота процесора – 1.6 ГГц;
- комунікації – microUSB.

4.5 Вимоги до інформаційної та програмної сумісності

Програмний продукт розробляється для всіх видів операційних систем сімейства “ANDROID” починаючи від версії 6 та наступні версії.

4.6 Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних).

На упаковці повинно бути вказана назва продукту, номер версії, мінімальні системні вимоги.

На зворотній стороні упаковки вказується розробник та його юридична адреса.

Програмний продукт **МОБІЛЬНИЙ ДОДАТОК** **«ЧАТ-БОТ ЇЖІ»** • діагональ екрану – 5.5; • роздільна здатність дисплею – HD (1280x720); • оперативна пам’ять – 2 ГБ; • вбудована пам’ять – 16 ГБ; • операційна система – ANDROID; • частота процесора – 1.6 ГГц;	Програмний продукт **МОБІЛЬНИЙ ДОДАТОК** **«ЧАТ-БОТ ТАКСІ»** **Розробник: Таран М. І.** **Кафедра «КІТ», УДУНТ**

• комунікації – microUSB.	**м. Дніпро, вул. Лазаряна 2** **2023**

4.7 Вимоги до транспортування та зберігання

Транспортування повинне забезпечувати збереження програмного продукту його цілісність і запобігання несанкціонованого доступу до нього. Програмний виріб міститься на фізичному носії та переданий через microUSB порт.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача.

Вся документація програмного додатку повинна задовольняти вимоги до програмної документації.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиці 1. – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.23 - 18.06.23
Робочий проект	Програмування та відлагодження програми.	19.02.23 - 20.05.23
	Тестування програми	20.05.23 - 27.05.23
	Розробка, узгодження і затвердження програмної документації.	27.05.23 - 12.06.23

7 ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки доц. Клименко І. В.

Прийом здійснюється комісією у складі:

- Горячкін В. М. (керівник підрозділу);
- Андрющенко В. О. (керівник розробки).

8 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем: методичні вказівки до дипломного проектування та лабораторних робіт/уклад.: Ю.М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с

Додаток Б

Текст програми мобільного прототипу

Bot.cs

```
using CSharpFoodChatBot.Controllers;

using CSharpFoodChatBot.Loggers;

using CSharpFoodChatBot.MarkupInterfaces;

using CSharpFoodChatBot.Records;

using Telegram.Bot;

using Telegram.Bot.Types;

using Telegram.Bot.Types.Enums;

using Telegram.Bot.Types.ReplyMarkups;



namespace CSharpFoodChatBot.Bot

{

   //Клас боту

   internal class Bot

   {

      //Посилання на сайт служби доставки їжі

      private readonly string _ourSiteUrl;



      //Посилання на сайт для закачки додатку служби доставки їжі

      private readonly string _ourAppUrl;



      //Кліент боту

      private readonly TelegramBotClient _client;



      //Головне меню боту

      private readonly InlineKeyboardMarkup _inlineKeyboardMarkupMainMenu;



      //Флаг для перевірки введення номеру(перевірка на те, що ми точно знаходимося у меню вводу номеру)

      private bool _menuSwap;



      //Параметричний конструктор

      //Вхідні параметри

      //token - токен боту

      public Bot(string token)

      {

         //Ініціалізація посилань

         _ourSiteUrl = "https://www.google.com/";

         _ourAppUrl = "https://www.google.com/";



         //Спочатку в головному меню

         _menuSwap = false;
```

```
                //Головне меню програми
                _inlineKeyboardMarkupMainMenu = new InlineKeyboardMarkup(new[]
                {
                    new[]
                    {
                        InlineKeyboardButton.WithUrl("💬 Оператор", "tg://resolve?domain=thomasmangames") //Замість TELEGRAM_USERNAME ввести ім'я аккаунту оператору в телеграмі без @)
                    },
                    new[]
                    {
                        InlineKeyboardButton.WithCallbackData("🍔 Моє замовлення", "button_myOrder")
                    },
                    new[]
                    {
                        InlineKeyboardButton.WithUrl("🌐 Наш сайт", _ourSiteUrl)
                    },
                    new[]
                    {
                        InlineKeyboardButton.WithUrl("📱 Наш додаток", _ourAppUrl)
                    }
                });

                //Створення клієнту
                _client = new TelegramBotClient(token);

                //Початок прослуховування повідомлень
                _client.StartReceiving(Logic, Error);

                Console.ReadLine();
            }

            //Метод запуску головного меню боту
            //Вхідні дані
            //botClient - клієнт боту
            //mainMenuText - текст головного меню
            //chatId - айді чату
            //cancellationToken - токен для відміни дії, яка довго виконується
            private async Task StartMainMenuBotAsync(ITelegramBotClient botClient, string mainMenuText, long chatId, CancellationToken cancellationToken = default)
            {
                await botClient.SendTextMessageAsync(chatId, mainMenuText, replyMarkup: _inlineKeyboardMarkupMainMenu, cancellationToken: cancellationToken);
```

```
        }

        //Метод обробки невідомої команди до боту
        //Вхідні дані
        //botClient - клієнт боту
        //unknownCommandText - текст для відповіді на невідому команду
        //chatId - айді чату
        //cancellationToken - токен для відміни дії, яка довго виконується
        private static async Task SendUnknownCommandMessageAsync(ITelegramBotClient botClient, string unknownCommandText, long chatId, CancellationToken cancellationToken = default)
        {
            await botClient.SendTextMessageAsync(chatId, unknownCommandText, cancellationToken: cancellationToken);
        }

        //Логіка боту
        //Вхідні дані
        //botClient - клієнт боту
        //update - оновлення у боті(повідомлення, кліки користувача)
        //cancellationToken - токен для відміни дії, яка довго виконується
        private async Task Logic(ITelegramBotClient botClient, Update update, CancellationToken cancellationToken)
        {
            //Отримуємо повідомлення користувача
            var message = update.Message;

            //Це повідомлення та саме текстове?
            if (message != null && message.Type == MessageType.Text)
            {
                //Запуск боту
                if (message.Text.Equals("/start"))
                {
                    string mainMenuText = $"Вітаємо {message.From.FirstName + " " + message.From.LastName}!\n\n" +
                    $"Ви потрапили в наш чат бот для служби доставки їжі 🍔 🍔 🍔.\n\n" +
                    $"Оберіть те, що вам потрібно зі списку знизу:";

                    await StartMainMenuBotAsync(botClient, mainMenuText, message.Chat.Id, cancellationToken);
                }
                else if (OrderNumberController.CheckOrderNumber(message.Text) && _menuSwap) //Перевірка номеру заказу користувача
                {
                    //Отримуємо номер заказу від користувача
```

```
                    string orderNumber = message.Text;

                    //Робимо запит до бази даних на пошук заказу з таким номером
                    DatabaseController databaseController = DatabaseController.Instance;

                    Order order = databaseController.GetOrderInfoByNumber(orderNumber);

                    //Якщо заказу за таким номером не існує
                    if (order.Id == 0)
                    {
                        await botClient.SendTextMessageAsync(message.Chat.Id, "Заказу за таким номером не існує", replyMarkup: MarkupInterface.GetBackToFoodQuestionsOrToMainMenuButtons(), cancellationToken: cancellationToken);
                    }
                    else //Якщо існує
                    {
                        await botClient.SendTextMessageAsync(message.Chat.Id, order.ToString(), replyMarkup: MarkupInterface.GetBackToFoodQuestionsOrToMainMenuButtons(), cancellationToken: cancellationToken);
                    }
                }
                else //Якщо користувач спамить - виводимо повідомлення про невідому команду
                {
                    string unknownCommandText = "Невідома команда😢";

                    await SendUnknownCommandMessageAsync(botClient, unknownCommandText, message.Chat.Id, cancellationToken);
                }
            }
            else //Якщо це клік
            {
                //Отримуємо дію користувача
                var callbackQuery = update.CallbackQuery;

                if (callbackQuery != null)
                {
                    //Видаляємо попереднє повідомлення
                    await botClient.DeleteMessageAsync(callbackQuery.Message.Chat.Id, callbackQuery.Message.MessageId, cancellationToken: cancellationToken);

                    //На що саме клікнув користувач
                    switch (callbackQuery.Data)
                    {
                        //Кнопка "🍔Моє замовлення"
                        case "button_myOrder":
```

```
                {
                    string myOrderText = "Оберіть пункт зі списку:";

                    await botClient.SendTextMessageAsync(callbackQuery.Message.Chat.Id, myOrderText, replyMarkup: MarkupInterface.GetFoodQuestions(), cancellationToken: cancellationToken);

                    break;
                }
                //Кнопка "Інформація про моє замовлення за номером "
                case "button_infoAboutOrderByNumber":
                {
                    _menuSwap = true;

                    string enterNumberOfOrderText = "Введіть номер(10 цифр) вашого замовлення, для отримання інформації:";

                    await botClient.SendTextMessageAsync(callbackQuery.Message.Chat.Id, enterNumberOfOrderText, replyMarkup: MarkupInterface.GetBackToFoodQuestionsOrToMainMenuButtons(), cancellationToken: cancellationToken);

                    break;
                }
                //Кнопка "Часті запитання "
                case "button_faq":
                {
                    string faqQuestionsText = "Ви потрапили до розділу частих запитань \nОберіть запитання зі списку для отримання відповіді :";

                    await botClient.SendTextMessageAsync(callbackQuery.Message.Chat.Id, faqQuestionsText, replyMarkup: MarkupInterface.GetFaqQuestions(), cancellationToken: cancellationToken);

                    break;
                }
                //Кнопка "Назад ", для того, щоб повернутись назад до питань про замовлення
                case "button_goBackToFoodQuestions":
                {
                    _menuSwap = false;

                    string myOrderText = "Оберіть пункт зі списку:";

                    await botClient.SendTextMessageAsync(callbackQuery.Message.Chat.Id, myOrderText, replyMarkup: MarkupInterface.GetFoodQuestions(), cancellationToken: cancellationToken);

                    break;
```

```
                }

                //Кнопка "Головне меню🏠"

                case
"button_goBackToMainMenu":

                {

                    _menuSwap = false;



                    string mainMenuText =
$"Вітаємо {callbackQuery.Message.From.FirstName + " "
+ callbackQuery.Message.From.LastName}!\n\n" +

                    $"Ви
потрапили в наш чат бот для служби доставки
їжі🍔 🍔 🍔.\n\n" +

                    $"Оберіть те,
що вам потрібно зі списку знизу:";



                    await
StartMainMenuBotAsync(botClient, mainMenuText,
callbackQuery.Message.Chat.Id, cancellationToken);



                    break;

                }

                //Кнопка "План приготування
мого замовлення"

                case
"button_cookingPlanForMyOrder":

                {

                    string
cookingPlanForMyOrderAnswer = "Після того, як ви
зробили замовлення на нашому сайті або в нашому
додатку ресторан/магазин приймає замовлення," +

                    "
та
починає його готувати. Коли ресторан/приймає
замовлення, то один з наших кур'єрів поблизу до цього
ресторану приймає ваше замовлення," +

                    "
та
вирушає до ресторану/магазину за ним. Після того, як
кур'єру віддають замовлення, він рухається за адресою
доставки." +

                    "
Коли кур'єра приїде за адресою доставки, то вам прийде
повідомлення про це, і ви можете забирати ваше
замовлення.";



                    await
botClient.SendTextMessageAsync(callbackQuery.Message
.Chat.Id, cookingPlanForMyOrderAnswer, replyMarkup:
MarkupInterface.GetBackToFaqQuestions(),
cancellationToken: cancellationToken);



                    break;

                }

                //Кнопка "Як довго чекати моє
замовлення?"

                case
"button_howLongToWaitForAnOrder":

                {

                    string
howLongToWaitForAnOrderAnswer = "Час доставки
замовлення залежить від часу приготування вашого
замовлення в ресторані" +

                    "
та
питань навколишньої середи, наприклад, погоди. Від
цих факторів залежить, як швидко приїде кур'єр.";
```

```
                await botClient.SendTextMessageAsync(callbackQuery.Message.Chat.Id, howLongToWaitForAnOrderAnswer, replyMarkup: MarkupInterface.GetBackToFaqQuestions(), cancellationToken: cancellationToken);

                break;
            }

            //Кнопка "Від чого залежить ціна доставки замовлення?"
            case "button_whatDeterminesThePriceOfDelivery":
            {
                string whatDeterminesThePriceOfDeliveryAnswer = "Ціна доставки вашого замовлення залежить від того," +
                " скільки кур'єру потрібно буде подолати шляху до вас та від зовнішніх факторів " +
                "(загруженність нашої служби доставки, погода, затори та інше).";

                await botClient.SendTextMessageAsync(callbackQuery.Message.Chat.Id, whatDeterminesThePriceOfDeliveryAnswer, replyMarkup: MarkupInterface.GetBackToFaqQuestions(), cancellationToken: cancellationToken);

                break;
            }

            //Кнопка "Кур'єр запізнився"
            case "button_courierWasLate":
            {
                string courierWasLateAnswer = "Якщо кур'єр запізнився більш ніж чим на 5 хвилин, то ціна доставки замовлення стає рівною 1 гривні.";

                await botClient.SendTextMessageAsync(callbackQuery.Message.Chat.Id, courierWasLateAnswer, replyMarkup: MarkupInterface.GetBackToFaqQuestions(), cancellationToken: cancellationToken);

                break;
            }

            //Кнопка "Назад ⬅", для того, щоб повернутись до меню частих запитань
            case "button_goBackToFaqQuestions":
            {
                string faqQuestionsText = "Ви потрапили до розділу частих запитань 💧 💧 💧 \nОберіть запитання зі списку для отримання відповіді :";

                await botClient.SendTextMessageAsync(callbackQuery.Message.Chat.Id, faqQuestionsText, replyMarkup: MarkupInterface.GetFaqQuestions(), cancellationToken: cancellationToken);

                break;
            }
        }
```

```
            }

        }

    }



    //Обробка помилок боту

    //Вхідні дані

    //botClient - клієнт боту

    //exception - виключення

    //cancellationToken - токен для відміни дії, яка довго виконується

    private Task Error(ITelegramBotClient botClient, Exception exception, CancellationToken cancellationToken)

    {

        using MessageLog messageLog = new(new ConsoleLogger());


messageLog.LogMessage(exception.Message);



        return Task.CompletedTask;

    }

  }

}
```

2) DatabaseController.cs

```
using CSharpFoodChatBot.Loggers;
using CSharpFoodChatBot.Records;
using System.Data.SqlClient;

namespace CSharpFoodChatBot.Controllers
{
  //Контроллер, для роботи з базою даних, який оснований на патерні Singleton
  internal class DatabaseController
  {
    //Об'єкт контроллеру
    private static DatabaseController _instance;

    //Підключення до бази даних
    private static SqlConnection _connection;

    //Конструктор з підключенням до бази даних
    private DatabaseController()
    {
      string connectionString = "Server=MEGATRON;" +
        "Database=orders_db;" +
        "Trusted_Connection=True;" +
        "TrustServerCertificate=True;";

      _connection = new SqlConnection(connectionString);
    }

    //Отримання об'єкту контроллеру
    public static DatabaseController Instance
    {
      get
      {
        return _instance ??= new DatabaseController();
      }
    }

    //Отримання заказу за номером
    //Вхідні дані
    //number - номер заказу
    //Вихідні дані
    //new Order - якщо заказ існує, то заказ із заповненними полями, та навпаки
    public Order GetOrderInfoByNumber(string number)
    {
      //Шукаємо замовлення
      try
      {
        //Відкриваємо підключення до бази даних
        _connection.Open();

        //Запит
        string queryText = $"SELECT * FROM [order] WHERE [number]='{number}';";

        //Команда
        SqlCommand command = new(queryText, _connection);

        //Читач даних
        SqlDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
          //Заказ знайдено?
          if (reader[1].Equals(number))
          {
            return new Order((int)reader[0], (string)reader[1], (string)reader[2], (string)reader[3], (string)reader[4], (decimal)reader[5], (DateTime)reader[6]);
```

```
                }
            }

            //Якщо не знайдено, то повертаємо пустий заказ
            return new Order();
        }
        catch (Exception ex) //Обробка помилок
        {
            using MessageLog messageLog = new(new ConsoleLogger());

messageLog.LogMessage(ex.Message);

            return new Order();
        }
        finally //Після блоку try-catch закриваємо підключення до бази даних
        {
            _connection.Close();
        }
    }
  }
}
```

3) OrderNumberController.cs

```
namespace CSharpFoodChatBot.Controllers
{
  //Контроллер, який перевіряє правильність вводу номеру заказу користувачем
    internal class OrderNumberController
    {
      //Метод перевірки номеру заказу
      //Вхідні дані
      //orderNumber - номер заказу
      //Вихідні дані
      //true/false - в залежності від результатів перевірки номеру
        public static bool CheckOrderNumber(string orderNumber)
        {
          //Якщо номер не null, довжиною в 10 символів та всі знаки номеру цифри - то він проходить перевірку
            if (orderNumber == null || orderNumber.Length < 10 || orderNumber.Length > 10 || !orderNumber.All(x => char.IsNumber(x)))
            {
              return false;
            }

            return true;
        }
    }
}
```

4) ConsoleLogger.cs

```
namespace CSharpFoodChatBot.Loggers
{
  //Логгер, який предназначений для того, щоб виводити помилки до консолі
    internal class ConsoleLogger : ILogger
    {
      //Метод Log, який виводить помилку до консолі
      //Вхідні дані
      //message - повідомлення помилки
      //Вихідні дані
      //Повідомлення у консолі з текстом помилки та датою цієї помилки
        public void Log(string message) => Console.WriteLine($"Error - {DateTime.Now}\n{message}\n");
    }
}
```

5) FileLogger.cs

```
namespace CSharpFoodChatBot.Loggers
{
  //Логгер, який предназначений для того, щоб записувати помилки до файлу
    internal class FileLogger : ILogger
    {
      //Шлях до файлу з помилками за замовченням
      private readonly string _filePathStandard;

      //Шлях до файлу, який можна встановлювати
      public string FilePath { get; set; }

      //Параметричний конструктор
      public FileLogger(string filePath)
      {
        FilePath = filePath;
        _filePathStandard = "Log.txt";
      }

      //Метод Log, який виводить помилку до консолі
      //Вхідні дані
      //message - повідомлення помилки
      //Вихідні дані
      //Текст у текстовому файлі з помилкою та датою цієї помилки
      public void Log(string message)
      {
        //Файл існує? - записати у вказаний користувачем шлях
        if (File.Exists(FilePath))
        {
          File.AppendAllText(FilePath, $"Error - {DateTime.Now}\n{message}\n");
        }
        else // - записати у шлях вказаний за замовченням
        {

File.AppendAllText(_filePathStandard, $"Error - {DateTime.Now}\n{message}\n");
        }
      }
    }
}
```

6) ILogger.cs

```
namespace CSharpFoodChatBot.Loggers
{
  //Інтерфейс, який предназначений для використання Dependency Injection у логгерах
    internal interface ILogger
    {
```

```
        //Прототипу методу логування помилки
        void Log(string message);
    }
}
```

7) MessageLog.cs

```
namespace CSharpFoodChatBot.Loggers
{
    //Сервіс, який приймає різні логери і викликає їх метод логування
    internal class MessageLog : IDisposable
    {
        //Логгер
        private readonly ILogger _logger;

        //Параметричний конструктор
        public MessageLog(ILogger logger)
        {
            _logger = logger;
        }

        //Метод для виклику логування з логгеру
        //Вхідні дані
        //messageForLog - текст помилки для логування
        //Вихідні дані
        //Помилка в консолі, або у файлі, в залежності від логеру
        public void LogMessage(string messageForLog) => _logger.Log(messageForLog);

        public void Dispose() { }
    }
}
```

8) MarkupInterface.cs

```
using Telegram.Bot.Types.ReplyMarkups;

namespace CSharpFoodChatBot.MarkupInterfaces
{
    //Клас для отримання клавіатур для користувача
    internal class MarkupInterface
    {
        //Отримати меню запитань про замовлення
        public static InlineKeyboardMarkup GetFoodQuestions()
        {
            InlineKeyboardMarkup inlineKeyboardMarkupFoodQuestions = new(new[]
            {
                new[]
                {

InlineKeyboardButton.WithCallbackData("Інформація про моє замовлення за номером ","button_infoAboutOrderByNumber")
                },
                new[]
                {

InlineKeyboardButton.WithCallbackData("Часті запитання ","button_faq")
                },
                new[]
                {

InlineKeyboardButton.WithCallbackData("Головне меню","button_goBackToMainMenu")
                }
            });

            return inlineKeyboardMarkupFoodQuestions;
        }

        //Повернутись до меню запитань про замовлення
        public static InlineKeyboardMarkup GetBackToFoodQuestionsOrToMainMenuButtons()
        {

            InlineKeyboardMarkup inlineKeyboardMarkupMainMenuButton = new(new[]
            {
                new[]
                {

InlineKeyboardButton.WithCallbackData("Назад ","button_goBackToFoodQuestions")
                },
                new[]
                {

InlineKeyboardButton.WithCallbackData("Головне меню","button_goBackToMainMenu")
                }
            });

            return inlineKeyboardMarkupMainMenuButton;
        }

        //Отримати часті запитання
        public static InlineKeyboardMarkup GetFaqQuestions()
        {
            InlineKeyboardMarkup inlineKeyboardMarkupFaqQuestions = new(new[]
            {
                new[]
                {

InlineKeyboardButton.WithCallbackData("План приготування мого замовлення","button_cookingPlanForMyOrder")
                },
                new[]
                {

InlineKeyboardButton.WithCallbackData("Як довго чекати моє замовлення?","button_howLongToWaitForAnOrder")
                },
                new[]
                {

InlineKeyboardButton.WithCallbackData("Від чого залежить ціна доставки замовлення?","button_whatDeterminesThePriceOfDelivery")
                },
```

```
                    new[]
                    {

InlineKeyboardButton.WithCallbackData("Кур'єр
запізнився","button_courierWasLate")
                    },
                    new[]
                    {

InlineKeyboardButton.WithCallbackData("Назад ","butt
on_goBackToFoodQuestions")
                    },
                    new[]
                    {

InlineKeyboardButton.WithCallbackData("Головне
меню ","button_goBackToMainMenu")
                    }
                });

                return
inlineKeyboardMarkupFaqQuestions;
            }

            //Повернутись до меню частих запитань
            public static InlineKeyboardMarkup
GetBackToFaqQuestions()
            {
                InlineKeyboardMarkup
inlineKeyboardMarkupFaqQuestions = new(new[]
                {
                    new[]
                    {

InlineKeyboardButton.WithCallbackData("Назад ","butt
on_goBackToFaqQuestions")
                    },
                    new[]
                    {

InlineKeyboardButton.WithCallbackData("Головне
меню ","button_goBackToMainMenu")
                    }
                });

                return
inlineKeyboardMarkupFaqQuestions;
            }
        }
    }
```

9) Order.cs

```
namespace CSharpFoodChatBot.Records
{
    //Клас "Заказ"
    internal class Order
    {
        //Айді
        public int Id { get; set; }
        //Номер заказу
        public string Number { get; set; }
        //Назва заказу
        public string Name { get; set; }
        //Опис заказу
        public string Description { get; set; }
        //Готовність
        public string Readiness { get; set; }
        //Оцінка
        public decimal Cost { get; set; }
        //Час доставки
        public DateTime TimeToDelivery { get;
set; }

        //Конструктор за замовчуванням
        public Order()
        {
            Id = 0;
            Number = string.Empty;
            Name = string.Empty;
            Description = string.Empty;
            Readiness = string.Empty;
            Cost = 0;
            TimeToDelivery = DateTime.MinValue;
        }

        //Параметричний конструктор
        public Order(int id, string number, string
name, string description, string readiness, decimal cost,
DateTime timeToDelivery)
        {
            Id = id;
            Number = number;
            Name = name;
            Description = description;
            Readiness = readiness;
            Cost = cost;
            TimeToDelivery = timeToDelivery;
        }

        //Перегружений toString
        public override string ToString()
        {
            return $"Номер заказу : {Number}\n" +
                $"Назва заказу : {Name}\n" +
                $"Опис заказу : {Description}\n" +
                $"Готовність заказу : {Readiness}\n"
+
                $"Ціна заказу : {Cost}\n" +
                $"Очікуваний час доставки
:{TimeToDelivery}";
        }
```