МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО

«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Застосування сучасних технологій розробки чат-ботів для планування та організації дня »

Виконавець: Володимир ДРОБОТУН

Керівник: к.т.н., доцент Сергій ВОДОП'ЯНОВ

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*
Кафедра *комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)
«____»_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Дроботуна Володимира Олександровича
(ПІБ випускника)

1. Тема роботи: «Застосування сучасних технологій розробки чат-ботів для планування та організації дня» затверджена наказом ректора № 623/ст від 01.05.2023р.
2. Термін виконання роботи: з 15 травня 2023 року по 25 червня 2023 року.
3. Вихідні дані до роботи: чат-бот для планування та організації дня.
4. Зміст пояснювальної записки: 1. Огляд та аналіз предметної області. 2. Проектування та архітектура системи. 3. Реалізація та функціональні можливості чат-бота. 4. Тестування та оцінка ефективності.
5. Перелік обов'язкового ілюстративного матеріалу: 1. Вибір програмного забезпечення для програмування. 2. Вихідні дані для роботи з чат-ботом. 3. Приклади відповідей та результатів чат-бота. 6. Тестування та оцінка ефективності.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Огляд та аналіз предметної області. Написання 1 розділу, представлення керівнику.	15.05.2023-20.05.2023	
2.	Вибір та опис використаних технологій. Написання 2 розділу, представлення керівнику.	21.05.2023-27.05.2023	
3.	Розробка «Чат-бота для планування та організації дня» Написання 3 розділу, представлення керівнику.	28.05.2023-04.06.2023	
4.	Написання 4 розділу, представлення керівнику.	05.06.2023-10.06.2023	
5.	Загальне редагування та друк пояснювальної записки.	10.06.2023-12.06.2023	
6.	Проходження нормоконтролю, перепліт пояснювальної записки.	12.06.2023-15.06.2023	
7.	Розробка тексту доповіді. Оформлення матеріалу для презентації	16.06.2023-18.06.2023	

7. Дата видачі завдання 15.05.2023р.

Керівник кваліфікаційної роботи _____ Сергій ВОДОП’ЯНОВ
(підпис керівника)

Завдання прийняв до виконання _____ Володимир ДРОБОТУН
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Застосування сучасних технологій розробки чат-ботів для планування та організації дня» містить: 46 сторінок, 34 рисунки, 11 інформаційних джерел.

Об’єкт дослідження – Чат-бот.

Предмет дослідження – Застосування сучасних технологій розробки чат-ботів для планування та організації дня

Мета кваліфікаційної роботи –розробка чат-бота з використанням обраних технологій.

Методи дослідження – логічний, синтезу, аналізу, порівняльний, обробка літературних джерел та моделювання.

Результати кваліфікаційної роботи рекомендується використовувати для ознайомлення із сучасними технологіями розробки чат-ботів

Для розробки телеграм бота знайдено та використано найефективніше програмне забезпечення.

ТЕЛЕГРАМ БОТ, PYTHON, TELEGRAM BOT API, БІБЛІОТЕКИ ДЛЯ РОБОТИ З ЧАТ-БОТАМИ, АЛГОРИТМИ ДЛЯ ОБРОБКИ КОМАНД ТА ПОВІДОМЛЕНЬ КОРИСТУВАЧА.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Чат-бот	−	Програмне забезпечення, здатне автоматично взаємодіяти з користувачем через текстові повідомлення.
Телеграм-бот	−	Чат-бот, який працює на платформі Telegram і надає функціональні можливості через цю платформу.
Технології розробки	−	Набір інструментів, методів та підходів, що використовуються для створення програмного забезпечення.
API(Application Programming Interface)	−	Набір правил і протоколів, що визначають спосіб взаємодії між компонентами програмного забезпечення.
NLP(Natural Language Processing)	−	Галузь комп’ютерної науки, що займається обробкою та аналізом прродної мови людей.
База даних	−	Структуроване сховище даних, що використовується для зберігання та організації інформації.
IT	−	Інформаційні технології
Планування та організація дня	−	Процес встановлення цілей, пріоритетів та розподілу часу для досягнення максимальної продуктивності і ефективності.

ВСТУП

В нашому сучасному світі, коли люди живуть постійним ритмом та швидко збільшується потреба в продуктивності та ефективності, планування та організація дня стають дедалі важливішими завданнями для кожної людини. Інформаційні технології допомагають нам розпланувати наші дії, і все більше користувачів віддають перевагу простоті та зручності використання телеграм-ботів для планування та організації свого дня.

Мета даного індивідульного завдання - створити телеграм-бота для планування та організації дня, який буде допомагати користувачам ефективно використовувати свій час та планувати свої завдання. Такий бот може бути особливо корисним для людей зі зайнятим графіком, дистанційних працівників, студентів, домогосподарок та багатьох інших.

Актуальність теми продиктована потребою в ефективному плануванні та організації дня для досягнення успіху в різних сферах життя. Телеграм-бот може стати цінним помічником у цьому процесі, допомагаючи користувачам створювати розклади, нагадувати про важливі події, контролювати виконання завдань та забезпечувати організованість у повсякденному житті.

Метою дослідження є створення практичного телеграм-боту для подальшого вивчення впливу різних функцій на продуктивність та організованість користувачів. Об'єктом дослідження є телеграм-бот з різними функціями (створення розкладів, нагадування про події, контроль виконання завдань тощо). Дослідження має на меті з'ясувати, які функції телеграм-боту є найбільш корисними для забезпечення ефективності та організованості у повсякденному житті користувачів.

Об’єктом дослідження є телеграм-бот з різними функціями планування та організації дня, такі як нагадування про зустрічі та завдання, календар для планування розкладу дня, можливість додавати та видаляти нотатки, а також відстежувати свій прогрес виконання завдань.

Наукова новизна кваліфікаційної роботи на тему «Застосування сучасних технологій розробки чат-ботів для планування та організації дня»

полягає у вивченні та впровадженні новітніх технологій розробки чат-ботів з використанням Python, Telegram Bot API та інших бібліотек.

Практичне значення даної роботи полягає у можливості створення зручного та функціонального чат-бота, який забезпечить користувачів зручним інтерфейсом для планування та організації їх дня. Використання чат-ботів дозволить персоналізувати досвід користувача, забезпечуючи йому зручний доступ до функцій планування та організації завдань. Такий підхід сприятиме ефективному управлінню часом і підвищенню продуктивності.

Такий підхід до планування та організації дня через телеграм-боти є особливо актуальним і важливим в наш час. Швидкий ритм життя, постійний потік інформації і багато завдань, які треба виконати, можуть створювати стрес та розпоряджати нашим часом. Телеграм-боти для планування та організації дня створюють зручний і доступний інструмент для управління часом та завданнями.

Також, важливо дослідити вплив особистих уподобань та потреб користувачів на використання телеграм-боту. Кожна людина має свої власні пріоритети та способи планування. Дослідження може допомогти визначити, як найкраще задіяти користувачів у процесі використання телеграм-боту та забезпечити їм індивідуальний підхід до планування та організації дня.

Отже, дане дослідження зосереджене на створенні практичного телеграм-бота для планування та організації дня, а також вивченні впливу різних функцій на продуктивність та організованість користувачів. Це дозволить забезпечити користувачів зручним та ефективним інструментом для управління часом та завданнями, сприяючи підвищенню продуктивності та досягненню поставлених цілей у різних сферах життя.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Технологічний контекст розробки чат-ботів в Telegram

У цьому розділі розглянемо загальну технологічну архітектуру чат-ботів в Telegram(рис. 1). Розробка чат-ботів вимагає використання різних компонентів та технологій для забезпечення їх функціональності та взаємодії з користувачами.

Рис. 1.1. Логотип Telegram

Одним з головних компонентів є серверні аплікації. Це програми, які виконуються на сервері та забезпечують обробку та відправку повідомлень користувачам. Серверні аплікації зазвичай програмуються за допомогою мов програмування, таких як Python, Node.js, Java або PHP. Вони отримують вхідні повідомлення від користувачів, обробляють їх та відправляють зворотні відповіді.

Ще одним важливим компонентом є бази даних. Чат-боти можуть зберігати інформацію про користувачів, їх налаштування, історію повідомлень та інші дані. Це дозволяє створювати персоналізовані відповіді та зберігати стан бота між взаємодіями з користувачами. Для зберігання даних можуть використовуватися різні типи баз даних, включаючи реляційні бази даних, NoSQL бази даних або спеціалізовані системи кешування.

Для забезпечення зручного та ефективного взаємодії з користувачами чат-ботів використовуються інші технології, такі як натуральна мова обробки (NLP) і штучний інтелект (AI). NLP (рис. 1.2) дозволяє ботам розуміти та інтерпретувати повідомлення користувачів, розпізнавати команди та інструкції, а також генерувати природну мову для відповідей.

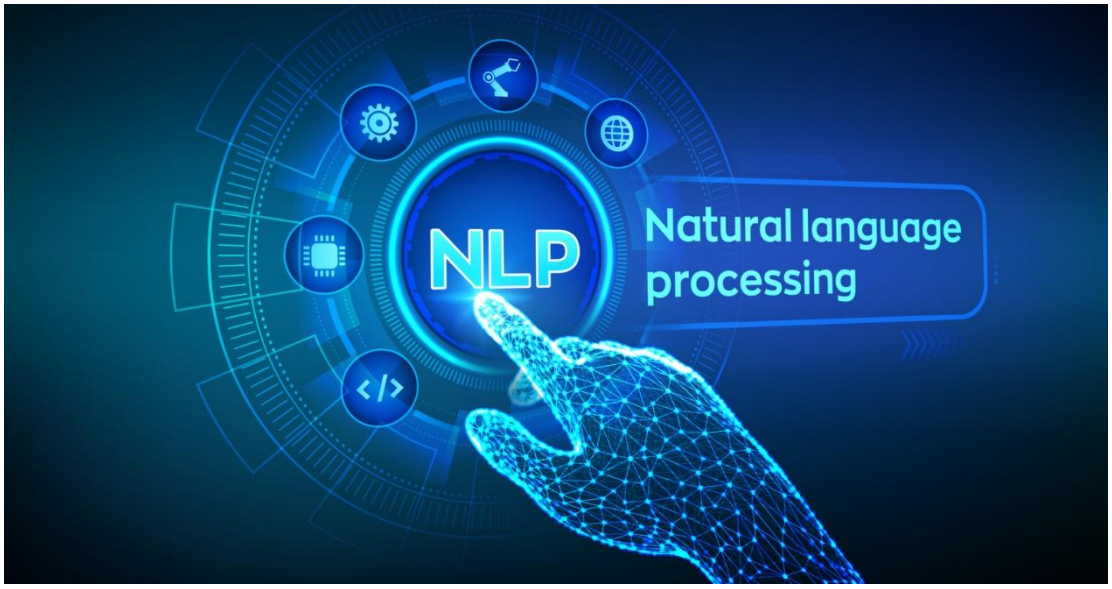


Рис. 1.2. Натуральна мова обробки

Це включає в себе використання алгоритмів обробки мови, таких як токенізація, стемінг, виявлення синтаксичних структур та інше. Завдяки штучному інтелекту, чат-боти можуть навчатися зі своїх даних та вдосконалювати свої відповіді з часом.

Окрім цього, для забезпечення спілкування з користувачами в реальному часі чат-боти використовують механізми передачі повідомлень, такі як WebSocket (рис. 1.3) або Webhook. Це дозволяє ботам отримувати повідомлення негайно і відправляти відповіді в реальному часі, що робить взаємодію з користувачами більш динамічною і ефективною.

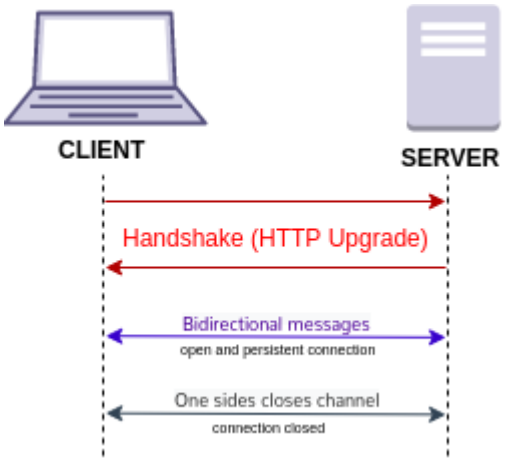


Рис. 1.3. З'єднання завдяки WebSocket

Технологічний контекст розробки чат-ботів в Telegram також включає в себе інтеграцію з іншими сервісами та API. Боти можуть взаємодіяти з зовнішніми додатками, сервісами платіжних систем, соціальними мережами та іншими зовнішніми ресурсами. Це дозволяє розширити функціональність бота і надати користувачам доступ до різноманітних сервісів та можливостей.

Загалом, розробка чат-ботів в Telegram вимагає поєднання різних технологій та компонентів. Відправлення та отримання повідомлень, обробка та зберігання даних, використання штучного інтелекту та інтеграція з іншими сервісами - це лише деякі з ключових аспектів, які потрібно враховувати при розробці чат-ботів в Telegram. Зрозуміння цього технологічного контексту допоможе розробникам ефективно створювати потужні та корисні боти для спілкування з користувачами у месенджері Telegram.

1.2. Переваги використання Telegram Bot API

У цьому розділі ми розглянемо широкий спектр переваг, які надає використання Telegram Bot API для розробки чат-ботів. Telegram Bot API є потужним інструментом, який забезпечує розробникам гнучкість та можливості для створення різноманітних та інтерактивних ботів в Telegram (рис. 1.4).

Рис. 1.4. Офіційний бот для створення інших ботів

Однією з основних переваг Telegram Bot API є його простота та зручність використання. API надає зрозумілу та докладну документацію, яка пояснює функціональність, параметри та методи. Це дозволяє розробникам швидко оволодіти API та легко створювати потужні чат-боти без зайвих зусиль.

Telegram Bot API також пропонує розширені можливості для взаємодії з користувачами. API дозволяє створювати клавіатури, які полегшують навігацію та взаємодію з ботом. Крім того, API підтримує роботу з різними типами медіафайлів, включаючи зображення, відео, аудіо та документи, що дозволяє ботам передавати та обробляти різноманітну інформацію.

Іншою важливою перевагою є доступність Telegram Bot API для різних мов програмування та розробницьких інструментів. Це означає, що розробники можуть використовувати свою улюблену мову програмування, таку як Python, JavaScript, Java, C# тощо, для створення чат-ботів. Крім того, Telegram надає різноманітні бібліотеки та SDK, які спрощують процес розробки та надають додаткові функціональні можливості.

Однією з ключових переваг Telegram Bot API є можливість використання кастомних клавіатур. Це дозволяє розробникам створювати власні інтерфейси взаємодії з користувачами, включаючи кнопки, меню та інші елементи управління. За допомогою кастомних клавіатур можна створювати зручний та

інтуїтивно зрозумілий інтерфейс для користувачів, що полегшує взаємодію з ботом та робить його більш привабливим для використання (рис. 1.5).

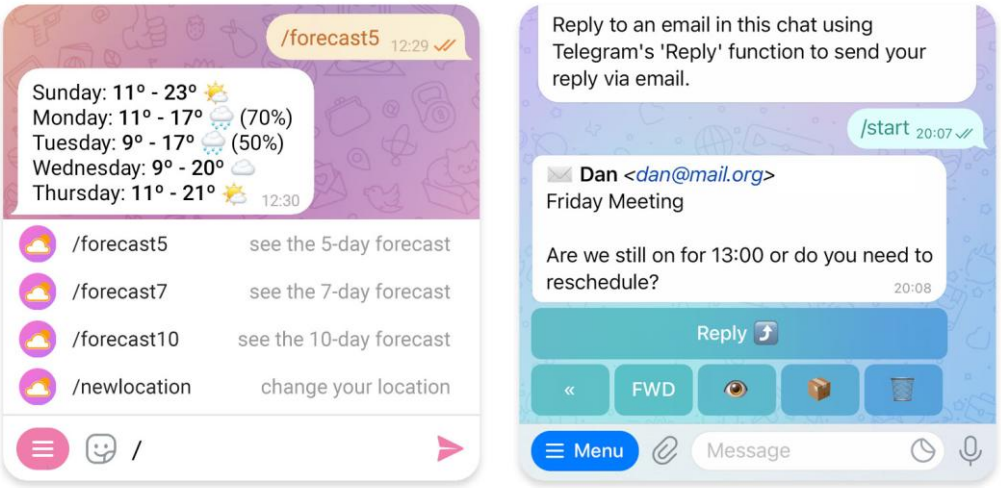


Рис. 1.5. Можливості телеграм-ботів

Також важливою перевагою Telegram Bot API є можливість обробки команд. Розробники можуть створювати власні команди, які активують певні функції чат-бота. Наприклад, команда "/start" може відкривати вітальне повідомлення або надавати корисну інформацію. Це дозволяє зручно управляти ботом та надавати користувачам швидкий доступ до потрібної функціональності.

Однією з потужних можливостей Telegram Bot API є підтримка інтеграції зі сторонніми сервісами. Розробники можуть легко інтегрувати свої чат-боти зі зовнішніми додатками, API та сервісами. Це дозволяє розширити функціональність бота та надати користувачам додаткові можливості. Наприклад, чат-бот може взаємодіяти зі сервісами оплати, доставки, соціальних мереж та багатьма іншими, щоб надати користувачам більш широкий спектр послуг.

Крім того, Telegram постійно вдосконалює свою платформу та впроваджує нові функції для чат-ботів. У майбутньому можна очікувати покращення штучного інтелекту, що дозволить ботам ставати ще більш розумними та контекстно-орієнтованими. Також, Telegram може розширити можливості взаємодії з користувачами, додати нові типи повідомлень та функціонал для роботи з медіафайлами. Такі оновлення роблять Telegram Bot API ще привабливішим для розробників та користувачів чат-ботів.

Загалом, використання Telegram Bot API виявляється вигідним варіантом для розробки чат-ботів. Зручність використання, розширені можливості взаємодії з користувачами, доступність для різних мов програмування та інструментів, безпека та можливість інтеграції зі сторонніми сервісами роблять Telegram Bot API потужним інструментом для створення інноваційних та функціональних чат-ботів в Telegram.

1.3. Виклики та ризики розробки чат-ботів в Telegram

Розробка чат-ботів в Telegram неминуче пов'язана з певними викликами та ризиками, які потребують уваги та вирішення. Одним із викликів є шкалування та продуктивність. При зростанні кількості користувачів, бот повинен бути здатен ефективно обробляти великий потік запитів та забезпечувати миттєву відповідь. Недостатня продуктивність може призвести до затримок у відповідях, що негативно позначиться на користувачах. Розробники повинні виявляти та оптимізувати проблемні елементи, використовувати кешування та інші стратегії для покращення продуктивності(рис.6).

Рис. 1.6. Серверна кімната

Безпека та приватність даних також є важливими аспектами розробки чат-ботів. Розробники повинні забезпечувати захист конфіденційної інформації, яку обробляє бот. Це включає захист від несанкціонованого доступу до даних (рис.

1.7), шифрування комунікації та правильне використання автентифікації та авторизації. Важливо також розуміти та дотримуватися політик безпеки Telegram, щоб уникнути можливих порушень.

Рис.1.7. Порушення політики безпеки

Управління помилками та збоями є ще одним викликом при розробці чат-ботів. Виникнення помилок та збоїв може призвести до некоректної роботи бота або його повного відмови. Розробники повинні передбачити можливі ситуації помилок, виявляти їх та вирішувати, а також реагувати на збої та відновлювати нормальну роботу бота якомога швидше.

Інтерфейс користувача та взаємодія зі сторонніми сервісами також можуть становити виклики при розробці чат-ботів в Telegram. Бот повинен мати зрозумілий та зручний інтерфейс, що дозволяє користувачам легко взаємодіяти з ним і здійснювати бажані дії. Важливо забезпечити зручне введення даних, чітке відображення інформації та можливість швидко знаходити потрібну функціональність. Потрібно також розглянути можливість інтеграції зі сторонніми сервісами, такими як платіжні системи, соціальні мережі або інші додатки, що можуть збагатити функціональність бота (рис. 1.8).

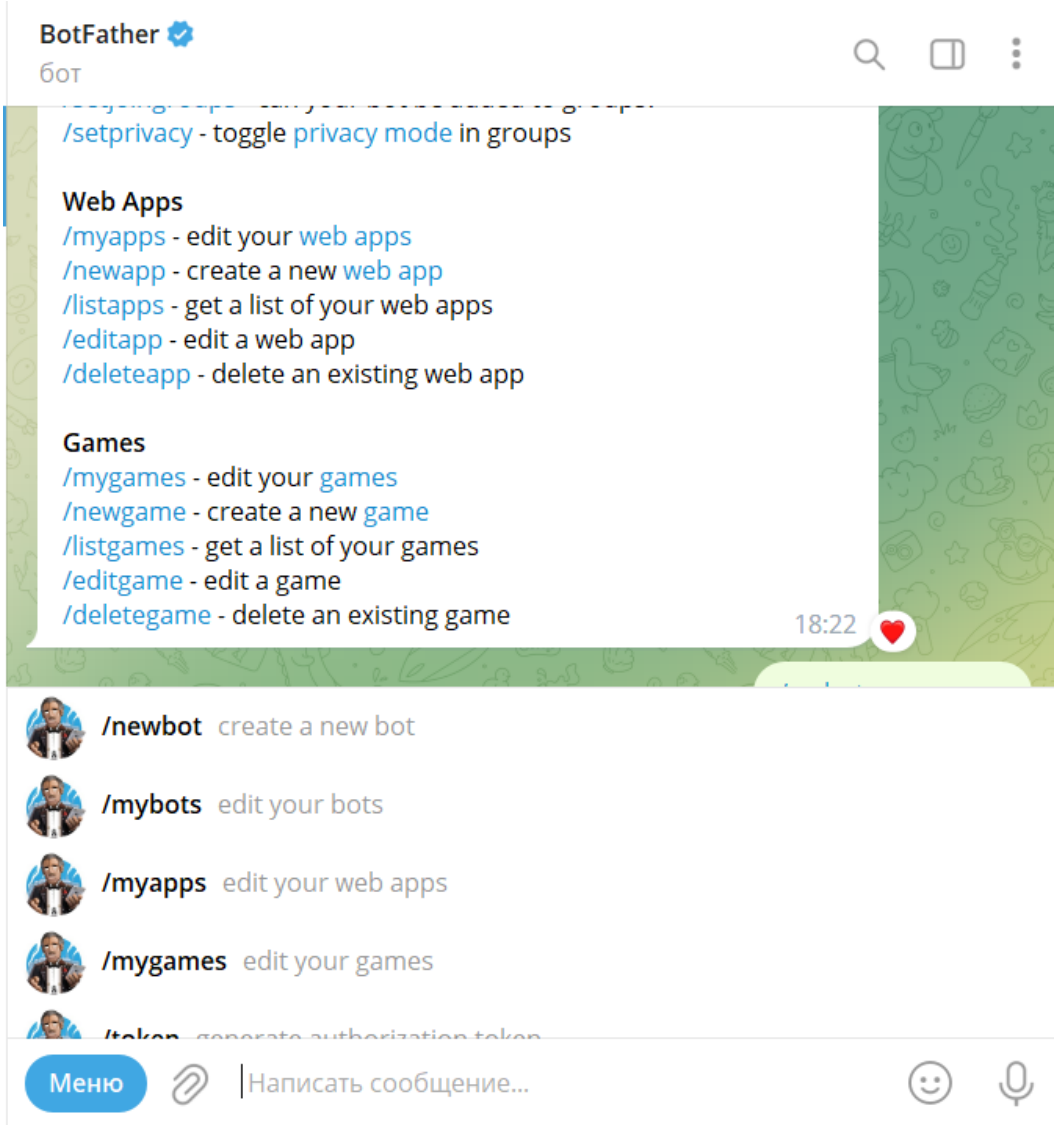


Рис. 1.8. Приклад інтерфейсу телеграм-бота

Одним із ризиків розробки чат-ботів є відповідальність за контент, що надсилається та обробляється ботом. Розробники повинні бути уважними та контролювати вміст, щоб уникнути поширення неправдивої інформації, образливого вмісту або порушення авторських прав. Також важливо мати механізми модерації та відповідати на скарги користувачів швидко та ефективно.

Ще одним ризиком є залежність від зовнішніх факторів, таких як доступність Telegram-серверів, стабільність мережі та інші технічні проблеми. В разі технічних збоїв або недоступності сервісу Telegram, робота бота може бути обмежена або неможлива. Розробники повинні розглянути такі сценарії та вжити заходів для забезпечення резервного варіанту дії, наприклад, шляхом використання альтернативних каналів комунікації або кешування даних.

Окрім того, при розробці чат-ботів важливо забезпечувати постійну підтримку та оновлення. Telegram постійно вдосконалює свої можливості та API, тому розробники повинні бути в курсі останніх змін і адаптувати свої боти до нових вимог та функціональності. Крім того, розробники повинні бути готові відповідати на запити та питання користувачів, вирішувати проблеми та надавати технічну підтримку (рис.1.9).

Рис. 1.9. Розробник

Узагальнюючи, розробка чат-ботів в Telegram має свої виклики та ризики, такі як шкалування та продуктивність, безпека та приватність даних, управління помилками та збоями, інтерфейс користувача, відповідальність за контент, залежність від зовнішніх факторів та необхідність постійної підтримки. Розробники повинні бути готові вирішувати ці виклики, враховувати ризики та вдосконалювати свої боти, щоб забезпечити найкращий досвід користувачів та відповідати їх потребам.

1.4. Тенденції розвитку чат-ботів в Telegram

Тенденції розвитку чат-ботів в Telegram постійно еволюціонують і впливають на спосіб, яким ми взаємодіємо зі світом навколо нас. З кожним роком все більше компаній, організацій і розробників використовують Telegram для створення інноваційних та потужних чат-ботів, що надають користувачам нові можливості та зручність в комунікації.

Однією з ключових тенденцій є зростання популярності і використання штучного інтелекту (ШІ) та машинного навчання у розробці чат-ботів. Завдяки цим технологіям, боти стають все більш інтелектуальними та здатними адаптуватися до потреб користувачів. Вони здатні розпізнавати мову, розуміти запити, аналізувати контекст та надавати персоналізовані відповіді. Застосування ШІ та машинного навчання робить чат-ботів більш ефективними та удосконаленими інструментами для комунікації та взаємодії.

Іншою важливою тенденцією є інтеграція з іншими сервісами та додатками. Telegram надає можливість зв'язку зі сторонніми API, що дозволяє розробникам інтегрувати чат-ботів з платіжними системами, соціальними мережами, CRM-системами та багатьма іншими сервісами. Це дозволяє створювати ботів, які можуть виконувати широкий спектр завдань, включаючи прийом платежів, надання інформації про статус замовлення, розсилку сповіщень та багато іншого (рис. 1.10).

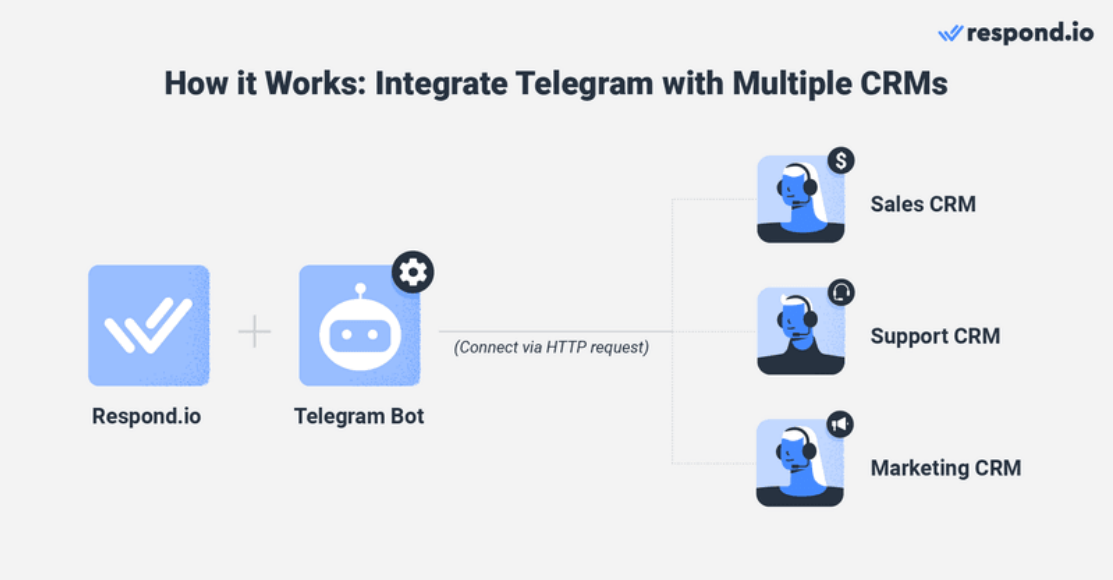


Рис. 1.10. Інтеграція CRM-систем в телеграм-боті

Також помітною тенденцією є розширення можливостей чат-ботів за допомогою використання голосових технологій. З'являються боти, які можуть розпізнавати голосові команди та надавати відповіді у голосовому форматі. Це робить комунікацію з ботами ще зручнішою, особливо в ситуаціях, коли користувачам зручніше говорити, ніж писати.

Зростання інтерактивності та використання багатомовності є ще однією тенденцією, яка впливає на розвиток чат-ботів в Telegram. Сучасні боти можуть

надавати користувачам можливість взаємодіяти з ними за допомогою кнопок, меню, каруселей та інших елементів управління. Вони можуть відтворювати різні ролі, такі як покупець або технічна підтримка, і надавати користувачам персоналізовані та унікальні досвіди(рис.1.11).

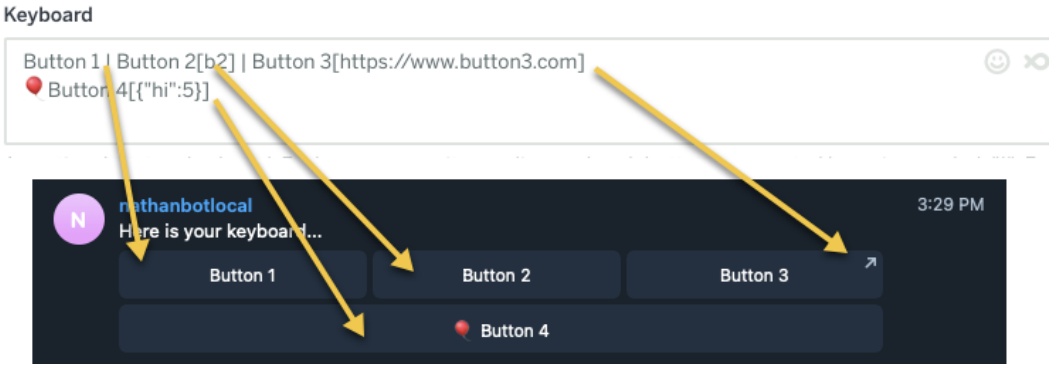


Рис.1.11 Взаємодія з ботом за допомогою кнопок

Не можна не зазначити і тенденцію до використання геолокації в чат-ботах. Вона дозволяє ботам надавати користувачам місцеву інформацію, знаходити найближчі до них послуги та об'єкти, розраховувати маршрути та багато іншого. Геолокація відкриває нові можливості для ботів у сферах туризму, гастрономії, торгівлі та багатьох інших.

Остаточно, розвиток чат-ботів в Telegram є динамічним та стимулюючим процесом. Штучний інтелект, інтеграція з іншими сервісами, використання голосових технологій, зростання інтерактивності, геолокація - всі ці тенденції сприяють створенню потужних та інтуїтивно зрозумілих чат-ботів, які відповідають нашим потребам у комунікації та взаємодії. У майбутньому можна очікувати ще більше інновацій та розширення можливостей чат-ботів в Telegram, які будуть перетворювати наш досвід взаємодії з цією платформою.

ВИСНОВКИ ДО РОЗДІЛУ 1

1. Розробка чат-ботів в Telegram вимагає використання різних компонентів та технологій, включаючи серверні аплікації, бази даних, технології обробки природної мови та штучний інтелект, механізми передачі повідомлень та інтеграцію з іншими сервісами і API.

2. Чат-боти в Telegram мають кілька переваг, включаючи простоту використання Telegram Bot API, розширені можливості для взаємодії з користувачами, доступність для різних мов програмування та кастомні клавіатури.

3. Telegram Bot API дозволяє розробникам створювати ботів з різноманітними функціями, включаючи обробку команд, роботу з медіафайлами та інтеграцію зі сторонніми сервісами.

4. Штучний інтелект та постійні оновлення платформи Telegram розширюють можливості чат-ботів, дозволяючи їм ставати розумнішими та контекстно-орієнтованими.

5. Використання Telegram Bot API є привабливим для розробників та користувачів чат-ботів через його гнучкість, широкі функціональні можливості та підтримку інтеграції з іншими сервісами.

6. Зрозуміння технологічного контексту розробки чат-ботів в Telegram допомагає розробникам ефективно створювати потужні та корисні боти для спілкування з користувачами у месенджері Telegram.

7. Виклики та ризики розробки чат-ботів в Telegram включають шкалування та продуктивність, безпеку та приватність даних, управління помилками та збоями, інтерфейс користувача, відповідальність за контент, залежність від зовнішніх факторів та необхідність постійної підтримки, і розробники повинні бути готові вирішувати ці виклики, враховувати ризики та вдосконалювати свої боти, щоб забезпечити найкращий досвід користувачів та відповідати їх потребам.

Виклики та ризики розробки чат-ботів в Telegram можна узагальнити наступними три реченнями:

1. Шкалування та продуктивність є одним з викликів розробки чат-ботів в Telegram. Зростання кількості користувачів вимагає ефективної обробки запитів та миттєвої відповіді, щоб уникнути затримок і незадоволення користувачів.

2. Безпека та приватність даних є важливими аспектами розробки чат-ботів. Розробники повинні забезпечити захист конфіденційної інформації, шифрування комунікації та дотримуватися політик безпеки Telegram для запобігання можливим порушенням.

3. Управління помилками та збоями є важливим викликом. Розробники повинні передбачати та вирішувати можливі ситуації помилок, а також реагувати на збої та швидко відновлювати нормальну роботу бота.

Тенденції розвитку чат-ботів в телеграм можна узагальнити наступними реченнями:

1. Розвиток чат-ботів в Telegram є постійним процесом, який впливає на спосіб, яким ми комунікуємося і взаємодіємо з оточуючим світом.

2. Використання штучного інтелекту, машинного навчання та інтеграція з іншими сервісами стають ключовими тенденціями у розвитку чат-ботів в Telegram, що дозволяє створювати інтелектуальні та ефективні інструменти для комунікації.

3. Впровадження голосових технологій, зростання інтерактивності та використання геолокації є іншими важливими тенденціями, що розширюють можливості чат-ботів в Telegram і покращують користувацький досвід.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА АРХІТЕКТУРА СИСТЕМИ

2.1. Вимоги до системи чат-бота в Telegram

Системи чат-ботів в Telegram мають відповідати певним вимогам, щоб забезпечити ефективну та задовільну взаємодію з користувачами. Основні вимоги до системи чат-бота в Telegram включають надійність, швидкість реакції, безпеку, функціональність та гнучкість. Нижче розглянемо кожну з цих вимог детальніше.

1. Надійність: Система чат-бота повинна бути надійною і стабільною, щоб забезпечити безперебійну роботу та уникнути відмов. Вона повинна витримувати велику кількість запитів від користувачів і відповідати на них швидко та ефективно. Також важливо мати моніторинг та системи реагування на помилки, щоб виявляти і виправляти проблеми якнайшвидше.

2. Швидкість реакції: Користувачі очікують миттєвих відповідей від чат-бота. Система повинна бути здатна обробляти запити швидко та надавати відповіді без помітної затримки. Важливо оптимізувати швидкодію системи, наприклад, шляхом використання кешування даних або оптимізації запитів до бази даних.

3. Безпека: Забезпечення безпеки є однією з ключових вимог до системи чат-бота в Telegram. Система повинна мати механізми аутентифікації та авторизації, щоб забезпечити захист особистої інформації користувачів. Також важливо захищати систему від атак, таких як SQL-ін'єкції або перехоплення сеансу.

4. Функціональність: Система повинна мати широкий спектр функцій і можливостей, які задовольняють потреби користувачів.

Вона повинна бути здатна взаємодіяти з користувачами, надавати інформацію, обробляти запити та виконувати завдання. Функціонал може включати створення опитувань, обробку платежів, надання консультацій, розсилку сповіщень та інші(рис.2.1).

Рис.2.1. Приклад функціонального телеграм-бота

5. Гнучкість: Система має бути гнучкою і здатною адаптуватися до різних ситуацій і потреб користувачів. Вона повинна мати конфігураційні налаштування, які дозволяють змінювати поведінку чат-бота без необхідності

втручання у вихідний код. Наприклад, зміна відповідей на певні запити або налаштування режимів роботи.

6. Інтеграція: Чат-боти в Telegram можуть бути інтегровані з іншими сервісами, такими як платіжні системи, CRM-системи або соціальні мережі. Система повинна надавати можливості взаємодії зі сторонніми API, щоб забезпечити цю інтеграцію(рис.2.2).

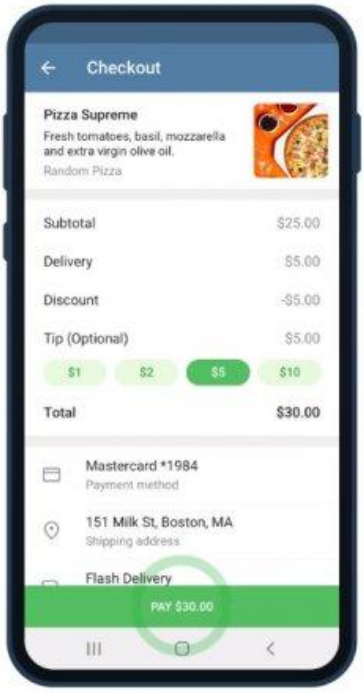


Рис.2.2.Інтеграція в телеграм-бот платіжних систем

7. Масштабованість: Чат-боти можуть зіткнутися зі збільшенням обсягу користувацького трафіку або запитів. Система повинна бути готовою масштабуватися, щоб впоратися з ростом навантаження. Це може включати використання горизонтального масштабування, використання кешування або облікових записів користувачів.

Усі ці вимоги є важливими для розробки та ефективної роботи системи чат-бота в Telegram. Враховуючи їх при проектуванні та розробці, можна забезпечити високу якість і задоволення користувачів, що сприятиме успіху та прийняттю чат-бота в Telegram.

2.2. Аналіз функціональних та нефункціональних вимог.

Аналіз функціональних та нефункціональних вимог є важливою складовою частини процесу розробки будь-якої системи, включаючи чат-бота в Telegram.

Функціональні вимоги описують те, що система повинна робити, які функції вона повинна виконувати, а нефункціональні вимоги визначають якість, продуктивність та характеристики системи. Давайте розглянемо ці вимоги більш детально.

Функціональні вимоги:

1. Реєстрація користувача: Система повинна мати можливість реєстрації нових користувачів. Це може включати збір обов'язкових даних, таких як ім'я, електронна пошта та пароль, а також опціональні дані, які можуть бути корисними для подальшої взаємодії з користувачем.

2. Автентифікація користувача: Після реєстрації користувач повинен мати можливість автентифікуватися в системі. Це може бути здійснено за допомогою електронної пошти та пароля або за допомогою інших механізмів, таких як двофакторна аутентифікація або автентифікація через соціальні мережі.

3. Взаємодія з користувачем: Основна функція чат-бота полягає взаємодії з користувачами через текстові повідомлення. Це може включати обробку текстових запитів, розпізнавання команд і надання відповідей на запити користувачів.

4. Надання інформації: Чат-бот повинен мати можливість надавати користувачам різноманітну інформацію. Це можуть бути відповіді на загальні запитання, роз'яснення процедур, надання керівництва або навіть надання рекомендацій.

5. Обробка запитів: Чат-бот може бути програмований для обробки конкретних запитів користувачів. Наприклад, він може мати можливість здійснювати бронювання, розраховувати вартість послуг, відстежувати доставку товарів або проводити операції з платіжними системами.

6. Взаємодія з іншими сервісами: Чат-бот може бути інтегрований з іншими сервісами або API, щоб забезпечити додатковий функціонал. Наприклад, це може бути інтеграція з платіжними системами для здійснення платежів або з CRM-системами для зберігання та обробки даних користувачів.

Нефункціональні вимоги:

1. Надійність: Система повинна бути надійною і стабільною. Вона повинна

вміти відновлюватися в разі виникнення помилок або відмов. Надійність також може включати захист від несанкціонованого доступу та забезпечення цілісності даних.

2. Продуктивність: Чат-бот повинен бути продуктивним і здатним обробляти запити користувачів з прийнятною швидкістю. Затримки у відповіді можуть вплинути на користувацький досвід і задоволення користувачів.

3. Масштабованість: Система повинна бути готовою масштабуватися, щоб впоратися зі зростаючим обсягом користувацького трафіку або запитів. Вона повинна мати можливість горизонтального масштабування, використання кешування та оптимізацію запитів для підтримки більшого навантаження.

4. Безпека: Забезпечення безпеки і конфіденційності даних користувачів є важливою нефункціональною вимогою. Система повинна мати захист від несанкціонованого доступу, шифрування даних, аутентифікацію користувачів та інші заходи для забезпечення безпеки.

5. Інтерфейс: Користувацький інтерфейс повинен бути зручним та інтуїтивно зрозумілим. Чат-бот повинен мати зрозумілу структуру, логічний навігаційний шлях та зручні елементи керування для користувачів(рис.2.3).

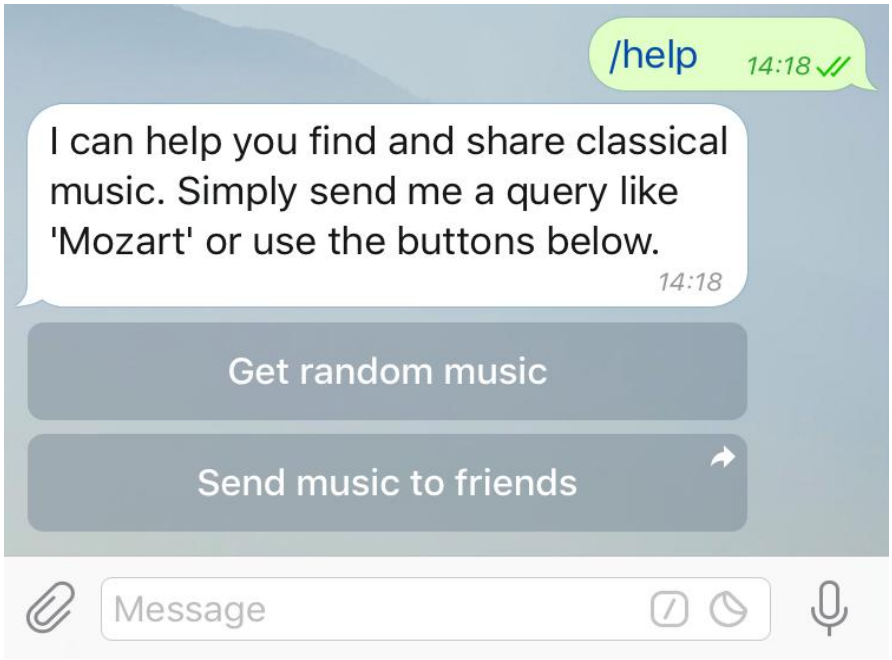


Рис.2.3.Приклад інтуїтивно зрозумілого інтерфейсу

6. Сумісність: Чат-бот повинен бути сумісним з різними пристроями та платформами. Він повинен працювати на різних версіях Telegram та різних операційних системах, щоб забезпечити доступність для широкої аудиторії

користувачів.

Аналіз функціональних та нефункціональних вимог допомагає зрозуміти обсяг роботи, який потрібно виконати для розробки чат-бота в Telegram, а також визначити ключові аспекти, які потрібно враховувати для успішного впровадження системи. Врахування цих вимог допомагає забезпечити високу якість та задоволення користувачів при використанні чат-бота.

2.3 Розробка архітектури системи чат бота в Telegram.

Розробка архітектури системи чат-бота в Telegram є важливим етапом у процесі реалізації функціональності та вимог, що ставляться до бота. Правильно спроектована архітектура дозволяє забезпечити ефективну та масштабовану роботу системи, зручну взаємодію з користувачами та можливість розширення функціоналу в майбутньому.

Перш за все, потрібно визначити, яку роль виконуватиме чат-бот в Telegram. Він може бути представлений як інтерфейс до певного сервісу або додатку, де користувачі зможуть здійснювати різноманітні дії через комунікацію з ботом. Наприклад, це може бути чат-бот для замовлення товарів, отримання інформації або підтримки клієнтів.

Для розробки архітектури можна використати підхід на основі модульної структури. В цьому випадку різні функції бота можна виділити в окремі модулі або компоненти, які взаємодіють між собою. Кожен модуль може відповідати за свою функціональність, наприклад, обробку запитів, взаємодію з базою даних або зовнішніми сервісами.

Один з ключових компонентів архітектури - це модуль комунікації з Telegram API. Він відповідає за отримання та відправку повідомлень через Telegram. Для забезпечення цієї функціональності можна використовувати Telegram Bot API, яке надає необхідні методи для обміну даними з Telegram.

Другим важливим компонентом є модуль обробки запитів користувачів. Він приймає вхідні повідомлення від Telegram API та виконує відповідні дії в залежності від змісту повідомлення. Цей модуль може містити логіку обробки команд, аналізу тексту, розпізнавання мови або інші алгоритми, необхідні для взаємодії з користувачами.

Третій компонент - модуль бази даних. Він зберігає інформацію про користувачів, їхні запити, статуси замовлень або будь-яку іншу потрібну інформацію. Для цього можна використовувати реляційну базу даних, таку як MySQL або PostgreSQL, або нереляційну базу даних, наприклад, MongoDB, залежно від потреб проекту.

Також слід розглянути можливість інтеграції з іншими сервісами або системами. Наприклад, якщо чат-бот призначений для підтримки клієнтів, важливо мати можливість інтеграції з системою керування відносинами з клієнтами (CRM) для зберігання та обробки інформації про клієнтів.

Архітектура системи чат-бота в Telegram може бути розширена за потреби. Наприклад, можна додати модуль аналізу даних для збору статистики, модуль аутентифікації користувачів для забезпечення безпеки або модуль інтеграції з платіжними системами для здійснення оплати послуг або товарів.

Важливо також врахувати фактори масштабованості та надійності системи. Розроблена архітектура повинна бути готова до збільшення кількості користувачів та обсягу оброблюваної інформації. Для цього можна використовувати принципи горизонтального масштабування та розподіленого зберігання даних.

ВИСНОВКИ ДО РОЗДІЛУ 2

Вимоги до системи чат-бота в Telegram є важливими для забезпечення ефективної та задовільної взаємодії з користувачами. Основні вимоги включають надійність, швидкість реакції, безпеку, функціональність, гнучкість, інтеграцію та масштабованість.

Надійність системи є важливою, оскільки чат-бот повинен працювати безперебійно та ефективно, витримувати велику кількість запитів та мати системи реагування на помилки.

Швидкість реакції є ключовою для задоволення очікувань користувачів. Система повинна бути здатна обробляти запити швидко та надавати миттєві відповіді.

Забезпечення безпеки є важливим аспектом, оскільки чат-боти мають обробляти особисту інформацію користувачів. Система повинна мати механізми аутентифікації, авторизації та захисту від різних атак.

Функціональність системи повинна задовольняти потреби користувачів і включати широкий спектр функцій, таких як надання інформації, обробка запитів, виконання завдань тощо.

Гнучкість системи дозволяє адаптуватися до різних потреб і ситуацій. Налаштування поведінки чат-бота мають бути доступними без втручання у вихідний код.

Інтеграція з іншими сервісами дозволяє розширити функціональність чат-бота. Система повинна надавати можливості взаємодії зі сторонніми API.

Після аналізу функціональних та нефункціональних вимог можна зробити такі висновки:

1. Функціональні вимоги: Вимоги щодо реєстрації та автентифікації користувачів, взаємодії з користувачем, надання інформації, обробки запитів та взаємодії з іншими сервісами дозволяють чат-боту виконувати основні функції комунікації та надання інформації користувачам.

2. Нефункціональні вимоги: Надійність системи забезпечує стабільну та

безперебійну роботу чат-бота, продуктивність гарантує швидку обробку запитів, масштабованість дозволяє системі зростати разом із збільшенням користувацького трафіку, безпека забезпечує захист даних користувачів,

зручний користувацький інтерфейс полегшує взаємодію користувачів з чат-ботом, а сумісність з різними пристроями та платформами забезпечує доступність системи для широкого кола користувачів.

Аналіз цих вимог дозволяє визначити ключові аспекти розробки та впровадження чат-бота, такі як реалізація системи реєстрації та автентифікації, обробка текстових запитів, інтеграція з іншими сервісами, заходи безпеки та оптимізація продуктивності. Дотримання цих вимог допомагає забезпечити високу якість та задоволення користувачів при використанні чат-бота в Telegram.

Розробка архітектури системи чат-бота в Telegram є ключовим етапом для забезпечення ефективної та масштабованої роботи бота. Важливо визначити роль чат-бота, яку він виконуватиме в Telegram, так як це вплине на функціонал та взаємодію з користувачами.

Модульна структура є ефективним підходом до розробки архітектури, де різні функції бота виділяються в окремі модулі або компоненти. Кожен модуль відповідає за свою функціональність, наприклад, обробку запитів, взаємодію з базою даних або зовнішніми сервісами.

Один з ключових компонентів - це модуль комунікації з Telegram API, який забезпечує отримання та відправку повідомлень через Telegram. Використання Telegram Bot API дозволяє зручно взаємодіяти з Telegram та обмінюватись даними.

Модуль обробки запитів користувачів приймає вхідні повідомлення від Telegram API і виконує відповідні дії залежно від їх змісту. Цей модуль містить логіку обробки команд, аналізу тексту, розпізнавання мови та інші алгоритми, необхідні для ефективної взаємодії з користувачами.

Модуль бази даних відповідає за зберігання інформації про користувачів, їх запити, статуси замовлень та іншу потрібну інформацію. Вибір реляційної або нереляційної бази даних залежить від потреб проекту.

Необхідно також розглянути можливість інтеграції з іншими сервісами або

системами, якщо це необхідно для функціоналу бота.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ ЧАТ-БОТА

У цьому розділі дипломної роботи ми представимо реалізацію та функціональні можливості нашого чат-бота, який призначений для планування та організації дня. Бот розроблений з використанням сучасних технологій розробки чат-ботів та Telegram API.

3.1 Огляд функціоналу.

Наш чат-бот має наступний функціонал:

- Додавання завдань: Користувач може додати нове завдання, вказавши опис, дату та час завершення.

Для додавання завдання користувач має ввести команду «/add_task», після чого бот запитує опис завдання, після введення опису завдання в користувача запитують дату до якої потрібно виконати завдання, після введення дати, користувач має ввести час, до якого потрібно виконати завдання. Після введення всіх необхідних бот виводить оновлений список з новим доданим завданням (рис.3.1).

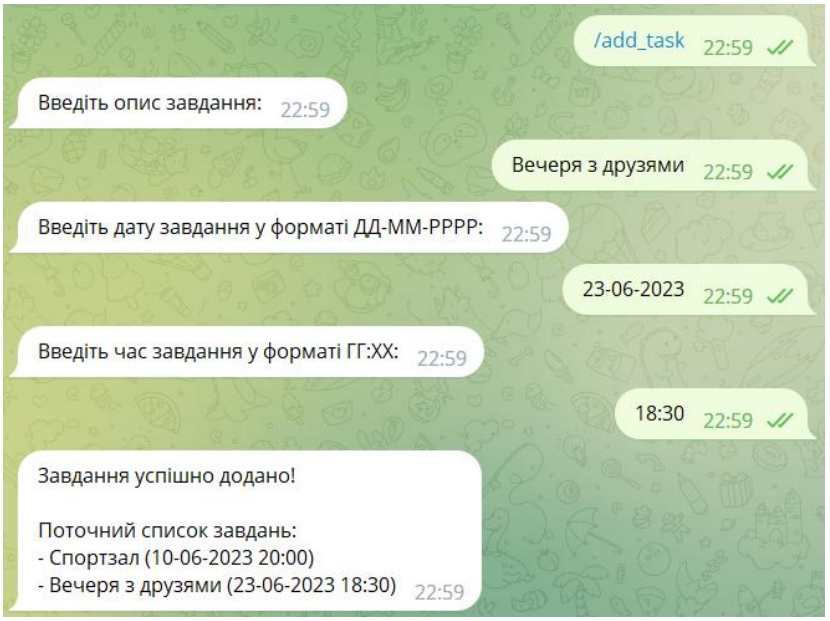


Рис.3.1. Приклад роботи чат-бота

• Перегляд завдань: Користувач може переглянути список своїх завдань. Для цього користувач поаинен ввести команду «/view_tasks», після чого йому буде виведено поточний список завдань (рис.3.2).

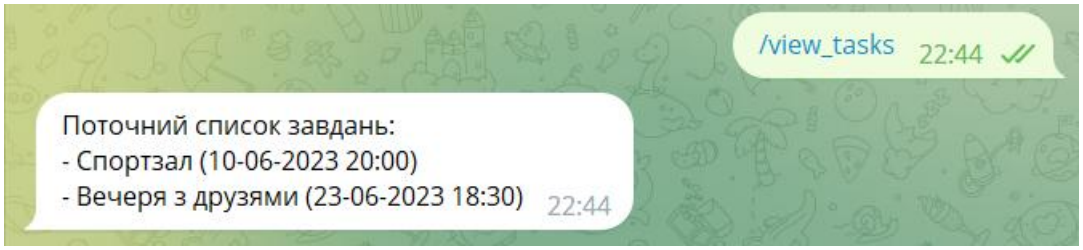


Рис.3.2. Перегляд завдань

• Видалення завдань: Користувач може видалити будь-яке завдання зі свого списку. Для цього він повинен ввести команду «/delete_task», після чого йому буде виведено список поточних завдань і він має натиснути на завдання, яке бажає видалити зі списку (рис.3.3).

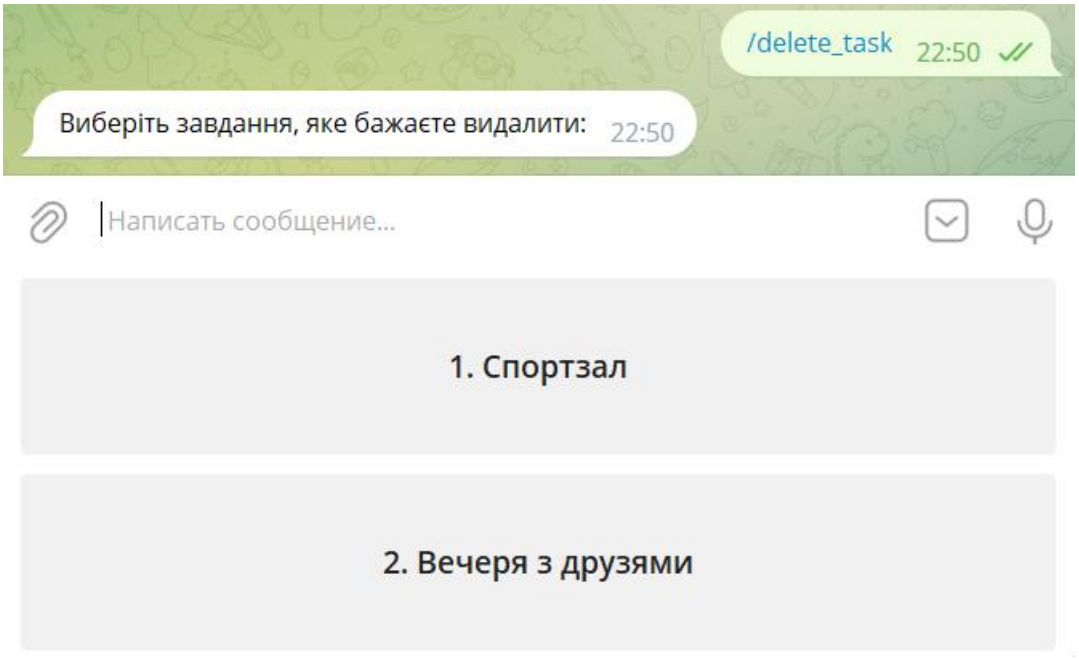


Рис.3.3. Видалення завдань

Після натискання на одне з завдань зі списку, завдання видаляється зі списку, і при наступному введенні команди «/view_tasks» видалене завдання вже відсутнє у списку (рис.3.4).

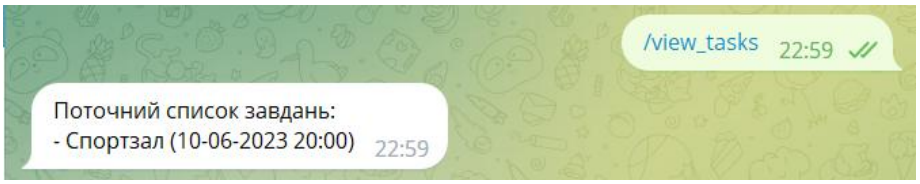


Рис.3.4. Результат видалення завдання

3.2 Завантаження та збереження завдань

Першим кроком в реалізації нашого чат-бота є завантаження та збереження завдань. Ми використовуємо файл JSON для збереження даних про завдання. Функція «load_tasks()» завантажує завдання з файлу JSON, а функція «save_tasks()» зберігає завдання у файл JSON(рис.3.5).

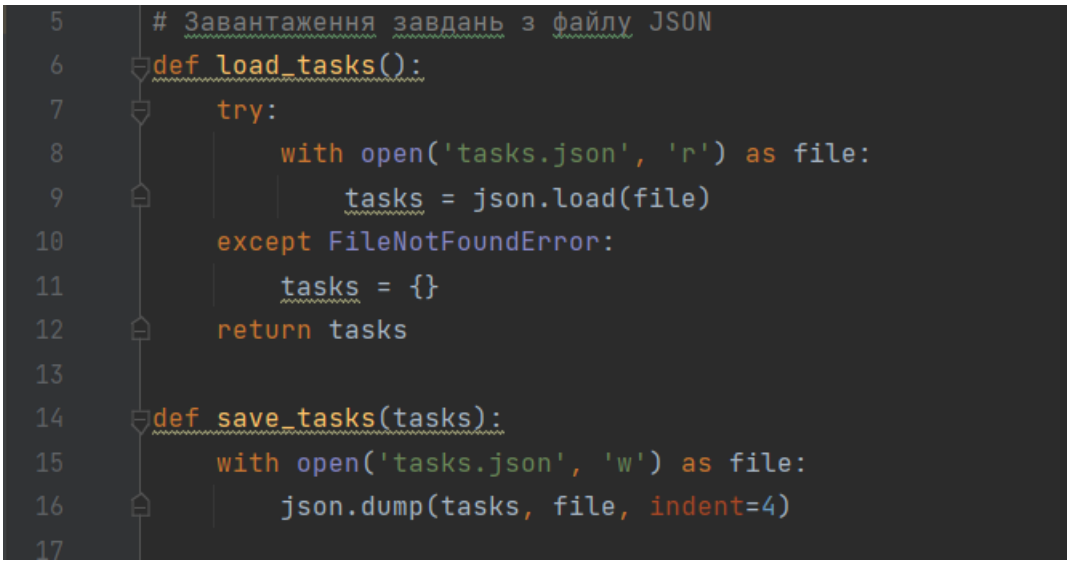
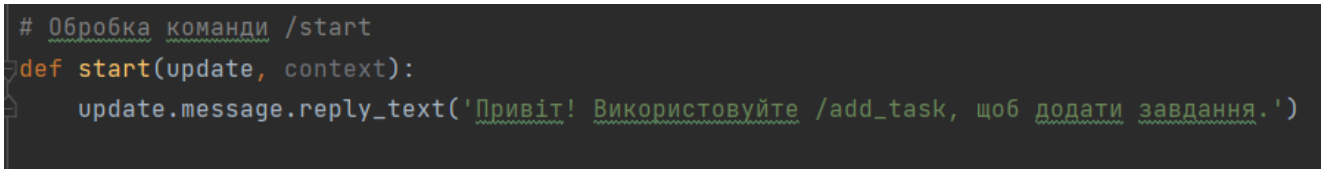


Рис.3.5. Функції для завантаження та збереження завдань

3.3 Обробка команд

Наш чат-бот реагує на кілька команд, такі як «**/start**», «**/help**», «**/add_task**», «**/view_tasks**» та «**/delete_task**». Кожна команда має свій власний обробник.

- Обробник команди «**/start**» надсилає вітальне повідомлення користувачу(рис.3.6)).



- Рис.3.6. Функція «/start»

- Обробник команди **/help** надсилає список доступних команд користувачеві(рис.3.7).

```python
# Обробка команди /help
def help(update, context):
    update.message.reply_text('Доступні команди:\n'
                              '/start - почати розмову\n'
                              '/help - показати список команд\n'
                              '/add_task - додати завдання\n'
                              '/view_tasks - переглянути список завдань\n'
                              '/delete_task - видалити завдання')
```

- Рис.3.7. Функція «/help»

- Обробник команди «**/add_task**» починає розмову для додавання нового завдання(рис.3.8).

```python
# Обробка команди /add_task
def add_task_command(update, context):
    update.message.reply_text('Введіть опис завдання:')
    return 'waiting_description'
```

Рис.3.8. Додавання опису завдання

- Обробник команди «**/view_tasks**» відображає список завдань користувача(рис.3.9).

```python
# Обробка команди /view_tasks
def view_tasks(update, context):
    tasks = load_tasks()
    chat_id = str(update.effective_user.id)  # Змінити chat_id на str(chat_id)

    if chat_id in tasks:
        task_list = '\n'.join([f'- {task["description"]} ({task["datetime"]})' for task in tasks[chat_id]])
        response_text = f'Поточний список завдань:\n{task_list}'
        update.message.reply_text(response_text)
    else:
        update.message.reply_text('Список завдань порожній.')
```

Рис.3.9. Функція «/view_tasks»

- Обробник команди **/delete_task** дозволяє користувачу видалити завдання зі свого списку(рис.3.10).

```python
# Обробка команди /delete_task
def delete_task(update, context):
    tasks = load_tasks()
    chat_id = str(update.effective_user.id)

    if chat_id in tasks:
        task_list = tasks[chat_id]
        if task_list:
            buttons = [[f'{i + 1}. {task["description"]}'] for i, task in enumerate(task_list)]
            reply_markup = ReplyKeyboardMarkup(buttons, one_time_keyboard=True)
            update.message.reply_text('Виберіть завдання, яке бажаєте видалити:', reply_markup=reply_markup)
            return 'waiting_task_number'
        else:
            update.message.reply_text('Список завдань порожній.')
    else:
        update.message.reply_text('Список завдань порожній.')
```

Рис.3.10. Функція «/delete_task»

3.4 Додавання завдань

Додавання нового завдання відбувається через розмову з користувачем. Кожна частина завдання (опис(рис.3.11), дата(рис.3.12), час(рис.3.13)) обробляється окремо, і після успішного завершення всіх кроків завдання додається до списку завдань(рис.3.14).

```python
# Обробка опису завдання
def process_task_description(update, context):
    print('Processing task description...')
    context.user_data['task_description'] = update.message.text
    update.message.reply_text('Введіть дату завдання у форматі ДД-ММ-РРРР:')
    return 'waiting_date'
```

Рис.3.11. Оброблення опису завдання

```python
# Обробка дати завдання
def process_task_date(update, context):
    print('Processing task date...')
    context.user_data['task_date'] = update.message.text
    update.message.reply_text('Введіть час завдання у форматі ГГ:ХХ:')
    return 'waiting_time'
```

Рис.3.12. Оброблення дати завдання

```python
# Обробка часу завдання
def process_task_time(update, context):
    print('Processing task time...')
    context.user_data['task_time'] = update.message.text
    add_task(update, context)
    return ConversationHandler.END  # Завершення розмови
```

Рис.3.13. Оброблення часу завдання

```python
# Додавання завдання
def add_task(update, context):
    chat_id = update.effective_user.id
    task_description = context.user_data.get('task_description')
    task_date = context.user_data.get('task_date')
    task_time = context.user_data.get('task_time')

    if task_description and task_date and task_time:
        if chat_id not in tasks:
            tasks[chat_id] = []

        task_datetime = f'{task_date} {task_time}'
        task = {'description': task_description, 'datetime': task_datetime}
        tasks[chat_id].append(task)
        save_tasks(tasks)

        task_list = '\n'.join([f'- {task["description"]} ({task["datetime"]})' for task in tasks[chat_id]])
        response_text = f'Завдання успішно додано!\n\nПоточний список завдань:\n{task_list}'
        update.message.reply_text(response_text, reply_markup=ReplyKeyboardRemove())

        # Очистити дані про завдання з контексту
        del context.user_data['task_description']
        del context.user_data['task_date']
        del context.user_data['task_time']
    else:
        update.message.reply_text('Помилка: неповна інформація про завдання.')
```

Рис.3.14. Додавання завдання

3.5 Видалення завдань

Видалення завдань відбувається через команду «**/delete_task**». Користувач може вибрати номер завдання для видалення. Після видалення завдання оновлюється список завдань(рис.3.15).

```python
def process_task_number(update, context):
    tasks = load_tasks()
    chat_id = str(update.effective_user.id)
    task_list = tasks[chat_id]

    selected_task_index = int(update.message.text.split('.')[0]) - 1

    if selected_task_index >= 0 and selected_task_index < len(task_list):
        selected_task = task_list[selected_task_index]
        task_list.pop(selected_task_index)
        tasks[chat_id] = task_list
        save_tasks(tasks)

        response_text = f'Завдання "{selected_task["description"]} ({selected_task["datetime"]})" видалено.'
        update.message.reply_text(response_text, reply_markup=ReplyKeyboardRemove())

        # Оновлення списку завдань після видалення
        updated_task_list = tasks[chat_id]
        if updated_task_list:
            updated_buttons = [[f'{i + 1}. {task["description"]}'] for i, task in enumerate(updated_task_list)]
            reply_markup = ReplyKeyboardMarkup(updated_buttons, one_time_keyboard=True)
            update.message.reply_text('Оновлений список завдань:', reply_markup=reply_markup)
        else:
            update.message.reply_text('Список завдань порожній.')

    else:
        update.message.reply_text('Неправильний номер завдання.')

    return ConversationHandler.END  # Завершення розмови
```

Рис.3.15. Видалення завдань

3.6 Підключення до Telegram API

Останнім кроком є підключення до Telegram API. Ми використовуємо модуль python-telegram-bot для спрощення цього процесу. Він надає зручний інтерфейс для роботи з Telegram API.

Створюється об'єкт «Updater», який використовує токен бота, створюється об'єкт «Dispatcher», який буде керувати обробкою повідомлень та команд(рис.3.16).

```python
# Ініціалізація токену бота та створення об'єкта Updater
updater = Updater('TOKEN', use_context=True)
dispatcher = updater.dispatcher
```

Рис.3.16. Підключення до Telegram API

Далі створюється `ConversationHandler`, який дозволяє обробляти різні стани та перехід між ними при спілкуванні з ботом. Для використання `ConversationHandler` необхідно встановити деякі точки входу (entry points), вказати стани (states) та задати дії, які відбуваються в кожному стані.

Основні компоненти `ConversationHandler` в даному фрагменті:

- «entry_points»: Вказує точку входу у розмову, у даному випадку команда «/add_task» викликає функцію «add_task_command».

- «states»: Визначає різні стани розмови та відповідні функції-обробники, які виконуються в кожному стані. В даному випадку є чотири стани: «waiting_description», «waiting_date», «waiting_time», «waiting_task_number», і для кожного стану вказані функції-обробники «process_task_description», «process_task_date», «process_task_time», «process_task_number», відповідно.

- «fallbacks»: Вказує додаткові команди, які можуть бути викликані в будь-який час для виходу з розмови або відміни поточної дії. У даному випадку команда «/cancel» викликає функцію «cancel».

Таким чином, `ConversationHandler` дозволяє створити послідовність дій для обробки розмови з користувачем, включаючи збереження стану між повідомленнями та можливість виклику спеціальних команд для виходу або відміни(рис.3.17).

```
conv_handler = ConversationHandler(
    entry_points=[CommandHandler('add_task', add_task_command)],
    states={
        'waiting_description': [MessageHandler(Filters.text, process_task_description)],
        'waiting_date': [MessageHandler(Filters.text, process_task_date)],
        'waiting_time': [MessageHandler(Filters.text, process_task_time)],
        'waiting_task_number': [MessageHandler(Filters.text, process_task_number)]
    },
    fallbacks=[CommandHandler('cancel', cancel)]
)
```

Рис.3.17. ConversationHandler

Далі додаються обробники команд до диспетчера (dispatcher) за допомогою функції «add_handler».

Кожен обробник команди («CommandHandler») приймає два параметри: назву команди та функцію-обробник, яка буде виконуватися при виклику цієї команди.

Основні компоненти цього фрагменту:

- «CommandHandler('start', start)»: Вказує, що при виклику команди `/start`, виконується функція «start».

- «CommandHandler('help', help)»: Вказує, що при виклику команди `/help`, виконується функція «help».

- «CommandHandler('view_tasks', view_tasks)»: Вказує, що при виклику команди «/view_tasks», виконується функція «view_tasks».

- «CommandHandler('delete_task', delete_task)»: Вказує, що при виклику команди `/delete_task`, виконується функція «delete_task».

- «conv_handler»: Вказує, що використовується «ConversationHandler» для обробки послідовних повідомлень та станів розмови.

Після додавання обробників команд до диспетчера, вони будуть викликатися та виконуватися відповідно до команд, які отримує бот в чаті(рис .3.18).

```
dispatcher.add_handler(CommandHandler('start', start))
dispatcher.add_handler(CommandHandler('help', help))
dispatcher.add_handler(CommandHandler('view_tasks', view_tasks))
dispatcher.add_handler(CommandHandler('delete_task', delete_task))
dispatcher.add_handler(conv_handler)
```

Рис.3.18. Додавання обробників команд

Після цього запускається бот за допомогою функції `start_polling()` з об'єкта `updater`.

`start_polling()` запускає процес отримання та обробки вхідних оновлень (повідомлень) від користувачів. Бот постійно слухає наявність нових оновлень і виконує відповідні функції-обробники, які були задані раніше.

Після виклику `start_polling()`, бот розпочинає виконання та очікує нових оновлень. Функція `updater.idle()` використовується для того, щоб утримувати бота активним та працюючим безперервно. Вона блокує виконання коду після `start_polling()` та дозволяє боту продовжувати працювати до тих пір, поки не буде зупинений вручну або не виникне якась помилка(рис.3.19).

```
updater.start_polling()
updater.idle()
```

Рис.3.19. Запуск бота

ВИСНОВКИ ДО РОЗДІЛУ 3

Розділ 3 "Реалізація та функціональні можливості чат-бота" описує конкретну реалізацію чат-бота з використанням сучасних технологій розробки. В цьому розділі була наведена прикладна реалізація чат-бота з можливостями планування та організації дня. Детально пояснено реалізацію та функціонал чат-бота.

Основна функціональність бота включає наступні можливості:

1. Додавання завдань: Користувач може додати нове завдання, введенням опису, дати та часу завдання.

2. Перегляд списку завдань: Користувач може переглянути поточний список завдань, які були додані раніше.

3. Видалення завдань: Користувач може видалити окремі завдання зі списку.

4. Допомога та початок розмови: Бот надає команду `/help`, щоб показати список доступних команд та надає початковий привітальний повідомлення за допомогою команди `/start`.

Код бота було організовано за допомогою `telegram.ext`, який забезпечує зручний спосіб роботи з різними типами повідомлень та командами. Використовується `ConversationHandler` для обробки послідовності повідомлень та станів, які дозволяють користувачу взаємодіяти з ботом для виконання конкретних завдань.

Загальною метою цього розділу є розробка функціонального чат-бота, який може допомогти користувачам планувати та організовувати свій день. Проаналізовано реалізацію функцій та структуру коду, що дозволяє легко розширювати функціональні можливості бота. Цей приклад може слугувати основою для створення ботів з іншими функціональними можливостями, залежно від потреб проекту.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

У цьому розділі описано тестування та оцінку ефективності розробленого чат-бота. Тестування було проведено для перевірки правильності роботи різних функцій бота та забезпечення відповідності до очікувань.

4.1 Тестування функцій бота

Для тестування функцій бота було використано різні сценарії взаємодії з користувачем. Було перевірено такі функції:

1.Додавання завдання: Виконано тестування додавання завдання з правильними даними, неповною інформацією та некоректним форматом дати та часу. Перевірено, чи було завдання додано до списку та чи виведено правильне повідомлення про успішне додавання або помилку.

2.Перегляд списку завдань: Виконано тестування перегляду списку завдань. Перевірено, чи виводиться правильний список завдань для кожного користувача, а також реакцію на порожній список.

3.Видалення завдання: Проведено тестування видалення завдання за вказаним номером. Перевірено, чи видаляється правильне завдання, чи оновлюється список після видалення та реакцію на некоректний номер завдання.

4.2 Оцінка ефективності

Оцінка ефективності бота може бути проведена з різних поглядів, включаючи швидкість відповіді, точність обробки команд, відсутність

помилок та інші аспекти.

Під час тестування було звернуто увагу на такі елементи ефективності:

1. Швидкість відповіді: Бот повинен реагувати швидко на команди користувача та обробляти їх без помітних затримок. Було перевірено час реакції бота на різні команди та спостережено, щоб впевнитися, що відповідь надходить вчасно.

2. Точність обробки команд: Бот повинен правильно розпізнавати та обробляти команди користувача. Під час тестування було перевірено, чи правильно обробляються команди додавання, перегляду та видалення завдань.

3. Реакція на помилки: Бот повинен відповідати на помилкові команди користувача зрозуміло та адекватно. Було перевірено, чи бот правильно реагує на неправильні команди та виводить зрозумілі повідомлення про помилку.

Розроблений чат-бот демонструє гарну ефективність та правильну функціональність під час тестування. Він швидко реагує на команди користувача, точно обробляє їх і надає зрозумілі повідомлення. Тестування допомогло виявити та виправити помилки, що сприяло покращенню якості та надійності бота.

Щоб подальше розвиток та вдосконалення бота, можна розглянути такі можливості:

1. Розширення функціоналу: Додати нові функції та можливості, які можуть бути корисними для користувачів. Наприклад, можливість нагадування про події, створення списків задач або планування подій.

2. Покращення інтерфейсу: Розробити інтуїтивно зрозумілий та привабливий інтерфейс для полегшення взаємодії з користувачем. Це може включати в себе використання графічних елементів, кольорів та інших дизайнерських ефектів.

3. Безпека та захист даних: Забезпечити високий рівень безпеки для персональних даних користувачів. Використовувати шифрування,

аутентифікацію та інші методи захисту для запобігання несанкціонованому доступу до інформації користувачів.

4. Підтримка багатомовності: Додати можливість розуміти та відповідати на запити користувачів різних мов. Це дозволить розширити базу користувачів та полегшити комунікацію з ними.

5. Підтримка для різних платформ: Розробити версії бота для різних платформ, таких як мобільні пристрої, веб-сайти або месенджери. Це дозволить користувачам отримувати доступ до бота зручним для них способом.

6. Постійне оновлення та вдосконалення: Забезпечити регулярні оновлення та вдосконалення бота, враховуючи зворотний зв'язок користувачів та нові технологічні можливості. Це дозволить тримати бота актуальним та забезпечити високу якість його роботи.

Враховуючи ці напрями розвитку, розроблений чат-бот може стати ще більш корисним та ефективним інструментом для користувачів.

ВИСНОВКИ ДО РОЗДІЛУ 4

У розділі "Тестування та оцінка ефективності" була показана важливість тестування чат-бота перед впровадженням і описані кроки для забезпечення правильної роботи програми. Основні висновки з цього розділу наступні:

1. Тестування є необхідним етапом у розробці чат-бота. Воно дозволяє виявити та виправити помилки та недоліки, підвищити якість та надійність програми перед впровадженням.

2. При тестуванні чат-бота варто переконатися, що він правильно обробляє вхідні дані та надає очікувані відповіді. Тестування повинно включати як позитивні, так і негативні сценарії взаємодії з ботом.

3. Використання збереження даних у файлі JSON дозволяє зберігати та отримувати завдання, які вводить користувач. Це дозволяє забезпечити постійність і доступність завдань навіть після перезапуску програми.

4. Розробка функцій додавання, перегляду та видалення завдань дозволяє користувачам зручно керувати своїми завданнями через чат-бота. Це покращує взаємодію з ботом та робить його більш корисним для користувачів.

5. Використання бібліотеки Telegram API та фреймворку python-telegram-bot спрощує розробку та інтеграцію чат-бота з платформою Telegram.

6. Регулярне оновлення та підтримка бота дозволяє розширювати його функціональність, покращувати ефективність та надійність, а також забезпечує задоволення потреб користувачів.

У цілому, цей розділ нагадує про важливість тестування та підтримки бота, які допомагають забезпечити високу якість та корисність чат-бота для його користувачів.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено дослідження та розробка чат-бота на платформі Telegram з використанням мови програмування Python та фреймворку python-telegram-bot. Робота була організована у чотири розділи, кожен з яких виконує певну функцію в процесі розробки чат-бота.

У першому розділі було проведено огляд та аналіз предметної області. Були досліджені існуючі методології розробки програмного забезпечення та вибрано найбільш відповідну для даного проекту. Було також проаналізовано Telegram API та фреймворк python-telegram-bot для визначення їх можливостей та обмежень.

У другому розділі було розроблено архітектуру системи чат-бота. Були визначені основні компоненти системи, взаємозв'язки між ними та принципи їх роботи. Було також визначено базові вимоги до функціональності чат-бота.

Третій розділ присвячений реалізації та функціональним можливостям чат-бота. Були створені необхідні функції та команди для додавання, перегляду та видалення завдань. Було розроблено інтерфейс користувача, який надає зручний спосіб взаємодії з ботом. Чат-бот здатний зберігати та обробляти дані завдань у форматі JSON.

Четвертий розділ присвячений тестуванню та оцінці ефективності розробленого чат-бота. Було проведено тестування різних сценаріїв взаємодії з ботом та перевірено правильність його роботи. Також була здійснена оцінка ефективності чат-бота з точки зору виконання завдань користувачів та зручності взаємодії.

В результаті роботи було успішно розроблено та протестовано чат-бот на платформі Telegram з функціональністю керування завданнями. Він надає зручний та простий спосіб ведення та організації завдань для користувачів. Розроблений чат-бот може бути використаний як в особистих, так і в професійних цілях, сприяючи підвищенню продуктивності та ефективності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1."Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming" by Eric Matthes, 2019.

2."Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners" by Al Sweigart , 2019.

3.“Telegram Bot API documentation”. URL: https://core.telegram.org/bots/api

4.“Python-telegram-bot documentation.” URL: https://python-telegram-bot.readthedocs.io/en/stable/

5.“IntelliJ IDEA”. URL: https://www.jetbrains.com/idea/

6.“BotFather”. URL: https://t.me/BotFather

7.Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020.

8. “Легкий спосіб вивчити Python 3 ще глибше”, Зед Шоу, 2020.

9.“Вивчаємо Python, 5-е видання”, Марк Лутц, 2020.

10. “Автоматизація рутинних завдань за допомогою Python. 2-ге видання.”, Е.Свейгарт, 2021

11.“Python 3.11 documentation”. URL: https://docs.python.org/3/