**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ**

**ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

# КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: *«***Розробка телеграм боту для підвищення комунікативності підприємств-замовників та постачальників***»*

Виконав: студент 6 курсу групи Іт-61

Спеціальності 126 «Інформаційні систем технології»

(шифр і назва)

Демчук Павло Віталійович

(Прізвище ім'я по-батькові)

Керівник: к.т.н., в.о. доцента Падюка Р.І.

(Прізвище та ініціали)

Рецензент: к.е.н., доц. Іваницький І. Є

(Прізвище та ініціали)

**Дубляни 2024**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ПРИРОДОКОРИСТУВАННЯ**

**ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Другий (магістерський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

д.т.н., проф. А. М. Тригуба

«_____»_____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу здобувачу

Демчуку Павлу Віталійовичу

1.Тема роботи: «Розробка телеграм боту для підвищення комунікативності підприємств-замовників та постачальників»

Керівник роботи Падюка Роман Іванович, к.т.н., в.о. доцента.
затверджені наказом по університету від 28 квітня 2023 року №133/к-с.

2. Строк подання здобувачем роботи 15.01.2024р.
3. Вихідні дані до роботи: Google cloud; Google sheets; telegram bots.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхіднорозробити)

*1. Аналіз стану питання в теорії та практики*
*2. Обґрунтування, вибору та реалізація інструментарію для підвищення ефективності комунікації з постачальниками та замовниками*
*3. Результати вирішення задачі з підвищення ефективності комунікації з постачальниками та замовниками*
*4. Охорона праці*

*5. Визначення розробки проекту комунікації між замовником та постачальником*
*Висновки та пропозиції*
*Список використаної літератури*

5. Перелік ілюстраційного матеріалу : *графік росту користувачів у месенджері WhatsApp та Facebook; приклади чат-ботів для замовлення товарів; інтерфейс еористувачів, використання та ініціалізація бібліотек функцій; вивід графіків.*

6. Консультанти з розділів:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1, 2, 3, 5 | *Падюка Р.І., в.о. доцента кафедри інформаційних технологій* | | |
| 4 | *Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва* | | |

7. Дата видачі завдання 01 травня 2023 р.

Календарний план

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Приміт-ка |
|---|---|---|---|
| 1 | Написання першого розділу та означення головних завдань роботи | 01.05.-30.05.23 | |
| 2 | Виконання другого розділу та вибір інтрументарію для виконання роботи | 01.06.-30.06.23 | |
| 3. | Виконання третього розділу та демонстрація окремих функцій | 01.09.-30.09.23 | |
| 4. | Виконання четвертого розділу та узагальнення отриманих результатів магістерської роботи | 01.10.-30.10.23 | |
| 6. | Виконання п’ятого розділу та реалізація графіків для аналізу | 01.11.-30.11.23 | |
| 7. | Завершення оформлення розрахунково-пояснювальної записки та аркушів графічної частини | 01.12.-30.12.23 | |
| 8. | Завершення роботи в цілому | 01.01.-10.01.24 | |

Здобувач_____Демчук П. В.
(підпис)

Керівник роботи_____Падюка Р. І.
(підпис)

УДК 004.9 : 631.1

Розробка телеграм боту для підвищення комунікативності підприємств-замовників та постачальників

Демчук П. В. Кафедра ІТ – Дубляни, Львівський НАУ, 2024.

Кваліфікаційна робота: 73 с. текст. част. 24 рис., 11 арк. ілюстраційного матеріалу, 15 джерел.

Проведено огляд чат-ботів аналогів. Наведено різні бібліотеки та інструменти для виконання проекту. Сформульовано завдання кваліфікаційної роботи.

Проведено аналіз месенджерів, інструментів, бібліотек та API. Оглянуто пропозиції для збереження даних.

Окреслено задачі розробки чат-боту для підвищення комунікації між постачальником та замовником.Вибрано засоби реалізації.

Подано особливості реалізації даного чат-боту із системою аналізу постачальників.

Розроблено заходи із охорони праці та безпеки у надзвичайних ситуаціях підчас використання чат-боту.

Визначено показники ефективності розробленого чат-боту.

**Зміст**

# Вступ

В епоху сучасності, де бізнес-процеси стають все більш складними та потребують надшвидкої та ефективної комунікації, використання інноваційних технологій стає необхідністю. Розробка та впровадження технологічних рішень, спрямованих на поліпшення взаємодії між підприємствами-замовниками та їх постачальниками, визначається як стратегічно важливе завдання для успішного функціонування підприємств у глобальному бізнес-середовищі.

Протягом останнього десятиліття виявляється тенденція до посилення конкуренції та швидкої зміни умов ринку. Підприємства перебувають під тиском не лише у вдосконаленні своїх продуктів та послуг, але й постійному вдосконаленні своїх бізнес-процесів. Ключовим аспектом успіху є здатність до ефективної взаємодії між підприємствами та їх постачальниками. Комунікація, спираючись на швидкість, точність та взаєморозуміння, стає визначальним елементом для забезпечення конкурентоспроможності.

У умовах сталого розвитку технологій виникають нові можливості для вирішення проблем ефективності взаємодії між підприємствами та їх постачальниками. Однією з перспективних технологій, яка пропонується для застосування у бізнес-середовищі, є розробка та використання телеграм ботів. Телеграм боти розглядаються як програмні агенти, здатні автоматизувати комунікацію та виконувати завдання за визначеними алгоритмами, що подаються від користувача. Вони надають можливість створення індивідуалізованих рішень для різноманітних сфер діяльності.

Об’єктом даного дослідження стає взаємодія між підприємствами-замовниками та їх постачальниками в умовах сучасного бізнес-середовища. Наша спрямованість дослідження полягає в оптимізації цієї взаємодії за допомогою телеграм боту, спрямованого на оптимізацію процесів комунікації та підвищення продуктивності обміну інформацією між сторонами.

Мета нашого дослідження полягає в створенні та впровадженні телеграм боту, який слугуватиме інструментом для покращення ефективності комунікації між підприємствами-замовниками та їх постачальниками. Наша мета – розробити інноваційний продукт, спрямований на автоматизацію процесів обміну інформацією, зниження ризику помилок та підвищення загальної продуктивності взаємодії бізнес-партнерів.

Результати нашого дослідження мають потенціал відкрити нові перспективи для оптимізації ланцюгів постачання та забезпечення сталого росту бізнесу в умовах сучасного економічного середовища. Застосування телеграм боту в бізнес-процесах може стати важливим фактором у підвищенні конкурентоспроможності підприємств та їх адаптації до викликів сучасного ринку.

Усе вищеописане обгрунтовує актуальність нашого дослідження та визначає його теоретичну та практичну значущість. У наступних розділах ми зосередимо увагу на теоретичних основах використання телеграм ботів у бізнес-середовищі, проведемо аналіз існуючих рішень та визначимо напрямки для подальших досліджень.

# Розділ 1

# Аналіз стану питання в теорії та практики

## 1.1. Характеристика ботів

Чат-боти стали неодмінною частиною сучасної технологічної парадигми, здатні поліпшувати взаємодію між людьми та комп'ютерами. Вони ціняться за здатність надавати швидку та ефективну підтримку, вирішувати прості завдання та взаємодіяти з користувачами в режимі реального часу[14]. Чат-боти також відомі своєю здатністю до адаптації до індивідуальних потреб користувачів, забезпечуючи персоналізований підхід у взаємодії.

Вони можуть мати інтеграцію зі штучним інтелектом, який може навчатися на основі взаємодії з користувачами, що робить їх гнучкими та здатними адаптуватися до змінних потреб аудиторії. Завдяки цьому, вони можуть надавати персоналізовану інформацію та рекомендації, сприяючи покращенню користувацького досвіду.

У бізнесі чат-боти також використовуються для збору аналітичних даних про споживачів. За допомогою алгоритмів машинного навчання, чат-боти можуть аналізувати величезні обсяги даних, визначаючи попит на конкретні товари чи послуги, що допомагає компаніям у формуванні стратегій маркетингу та розвитку продуктів.

Використання чат-ботів у сфері бізнесу може мати значущий вплив. Вони можуть служити ефективним інструментом для підтримки клієнтів, автоматизації обробки замовлень та збільшення конверсії. Занурення чат-ботів у сферу бізнесу вимагає ретельного планування та налаштування, а також постійного вдосконалення для врахування змін потреб користувачів та вирішення нових завдань. Успішне впровадження чат-ботів може забезпечити ефективну взаємодію з клієнтами, підвищити рівень задоволеності та сприяти росту бізнесу.

Популярність чат-ботів пояснюється їхньою надзвичайною доступністю та готовністю працювати цілодобово. Вони можуть автоматизувати рутинні операції, зменшуючи час, необхідний для вирішення питань клієнтів чи виконання завдань. Також, чат-боти дозволяють компаніям покращувати свою ефективність, вивчаючи питання користувачів і забезпечуючи аналітику для подальших вдосконалень.

Однією з ключових переваг чат-ботів є їхня здатність інтегруватися з різними платформами та системами. Це дозволяє їм ефективно взаємодіяти з існуючими й новими інформаційними технологіями у компанії, що спрощує впровадження та підтримку. Наприклад, чат-бот може бути інтегрований з CRM-системою для оптимізації обробки клієнтських запитань та покращення обслуговування клієнтів.

Загалом, чат-боти представляють собою потужний інструмент для підтримки бізнес-процесів, спрощення комунікації та вдосконалення взаємодії з клієнтами, що в результаті може призвести до зростання продуктивності та конкурентоспроможності компанії.

## 1.2. Аналіз існуючих методів комунікації між замовниками та постачальниками продукції чи послуг

У вимірах глобальної економічної динаміки та швидко змінюваного бізнес-середовища, важливість ефективної комунікації між замовниками та постачальниками продукції чи послуг набуває все більшого значення. Ця проблема стає ключовою для успішної інтеграції бізнес-взаємодії та визначає конкурентоспроможність підприємств у сучасному світі. Таким чином, дослідження існуючих методів комунікації в цьому контексті стає важливим завданням для вивчення оптимальних стратегій і покращення взаємодії між сторонами.

Сьогодні термін "комунікаційна діяльність" часто ототожнюється з просуванням товару, яке трактується як односпрямований інформаційний вплив виробників на споживачів з метою спонукання їх до придбання своєї продукції. Таким чином, основою комунікації в управлінні є обмін інформацією. Комунікація формується відповідно до завдань, тобто цілей, які ставить перед собою менеджмент (рис. 1.1).


*Рисунок 1.1 - Цілі комунікацій*

Однією з пріоритетних цілей є забезпечення ефективного обміну інформацією між суб'єктом управління та об'єктом управління. Ця мета спрямована на оптимізацію координації дій суб'єкта управління в процесах, що впливають на об'єкт управління [15].

Як малі, так і великі підприємства відіграють потужну роль у якісній комунікації. Як компанії, що працює на ринку, важливо спілкуватися не тільки зі співробітниками, а й з клієнтами. Для того, щоб покращити канали комунікації з клієнтами, необхідно успішно використовувати нові технології.

Не менш важливим аспектом є покращення взаємовідносин у процесі обміну інформацією, адже від того, як співробітники залучені до комунікаційного процесу, залежить їхня сприйнятливість та здатність надавати необхідну інформацію.

Регулювання та впорядкування потоку інформації допомагає фільтрувати дані та визначати релевантну інформацію. Це гарантує, що інформація, необхідна для прийняття управлінських рішень, буде доступна вчасно і в необхідному обсязі.

Налагодження інформаційних каналів для обміну інформацією між окремими працівниками і групами та координації їхніх завдань і дій відіграє важливу роль в організації процесу комунікації.

Перед початком аналізу існуючих методів комунікації важливо розглянути об'єкт вивчення. Замовники і постачальники входять у складну мережу взаємодії, яка обумовлена різноманітністю галузей бізнесу та різними видами послуг і продукції. Це може охоплювати виробництво товарів, надання послуг, логістичні процеси, технічну підтримку, інновації та інші аспекти. У кожній з цих галузей взаємодія вимагає специфічних підходів та методів комунікації.

Починаючи аналіз, слід звернутися до традиційних методів комунікації, таких як особисті зустрічі, телефонні розмови та електронна пошта.

Особистий контакт надає можливість виражати емоції та встановлювати більш особистий зв'язок між сторонами, але водночас може бути обмежений в географічному плані.

Телефонні розмови забезпечують швидкий обмін інформацією, але можуть бути менш ефективними в порівнянні з іншими засобами комунікації.

Електронна пошта, з своєю чергою, дозволяє відправляти документи та інші матеріали, але може породжувати питання конфіденційності та забезпечення безпеки.

Враховуючи стрімке розвиток технологій, дослідження також повинно охопити інноваційні методи комунікації, такі як віртуальні конференції, чат-боти, відеодзвінки та соціальні мережі.

Віртуальні конференції можуть стати ефективною альтернативою особистим зустрічам, забезпечуючи можливість обговорення питань в реальному часі.

Чат-боти стають все більш популярними для швидкого вирішення проблем та надання інформації.

Відеодзвінки розширюють можливості віртуального контакту, а соціальні мережі можуть бути ефективним інструментом для залучення уваги та збільшення взаєморозуміння між сторонами.

Під час аналізу слід також врахувати специфіку ринку, на якому діють замовники та постачальники. Різні галузі можуть вимагати різних підходів і методів комунікації через особливості їх виробництва, обсягу операцій та вимог клієнтів. Наприклад, в індустрії інформаційних технологій може бути більш актуальними віртуальні форми спілкування, тоді як в сфері важкої промисловості більше акценту може бути розміщено на особистих зустрічах та логістичних процесах[11].

Додатковим фактором для уваги є культурні особливості та різноманітність мов у бізнес-середовищі. Збереження взаєморозуміння в умовах різних культурних контекстів може виявитися важливою задачею, адже неправильне розуміння може спричинити конфлікти та непорозуміння, що негативно позначається на взаємодії між сторонами.

У подальшому аналізі можна вивчити досвід великих корпорацій та малих підприємств у сфері взаємодії з постачальниками та замовниками. Проведення кейс-студій та інтерв'ю з представниками різних компаній може розкрити ефективні практики та виклики, з якими вони стикаються в процесі комунікації. Такий підхід дозволить зрозуміти контекст застосування різних методів та визначити найбільш оптимальні стратегії.

Необхідно також врахувати вплив цифровізації на зміну підходів до комунікації. Впровадження цифрових технологій, таких як хмарні рішення, автоматизація процесів та інші інноваційні засоби, може значно полегшити взаємодію між сторонами та покращити комунікаційні процеси.

Крім того, слід звернути увагу на аспекти безпеки в комунікаційних процесах між замовниками та постачальниками. Цифрові засоби передачі інформації пов'язані із загрозами щодо конфіденційності та цілісності даних.

Розробка та впровадження ефективних заходів безпеки є критичним аспектом, оскільки вони забезпечують захист конфіденційної інформації та підтримують довіру між сторонами.

Загальний аналіз існуючих методів комунікації між замовниками та постачальниками продукції чи послуг вимагає комплексного підходу та врахування різноманітних факторів, таких як тип бізнесу, галузеві особливості, культурний контекст та технологічні інновації. Це дослідження може визначити оптимальні стратегії комунікації, сприяти покращенню взаємодії між сторонами та визначити напрямки подальших досліджень для розвитку цієї важливої сфери бізнесу.

## 1.3. Аналіз ринку месенджерів

В умовах стрімкого технологічного розвитку та постійного зростання інформаційного суспільства, месенджери стають невід'ємною частиною нашого щоденного життя. Ця сфера комунікаційних технологій переживає надзвичайно швидкі зміни, і аналіз ринку месенджерів визначається необхідністю зрозуміти динаміку цього сегменту та виявити фактори, що впливають на його розвиток.

Перед аналізом, слід розглянути широкий спектр месенджерських платформ, які присутні на ринку. Від популярних глобальних плеєрів, таких як WhatsApp, Telegram, та Viber, до спеціалізованих месенджерів, орієнтованих на бізнес, як Slack або Microsoft Teams, і до месенджерів, спрямованих на конкретні регіональні ринки. Дослідження різноманітності цих платформ дозволить визначити тенденції та особливості, які впливають на їхню конкурентоспроможність та популярність серед користувачів[1].

Відкриває рейтинг всім відомий додаток для спілкування WhatsApp. Це безкоштовний месенджер, за допомогою якого можна надсилати текстові та голосові повідомлення, здійснювати дзвінки, обмінюватись різними файлами, підходить і для відеодзвінків.

Згідно зі статистикою 2022 року, програмою користується понад 2 000 000 000 користувачів. Донедавна їх кількість неухильно зростала, але потім з’явилася інформація про передачу особистих даних. Хтось, злякавшись, перейшов до конкурентів. Але більшість, розуміючи всі плюси програми, так само використовує її для спілкування і для роботи.

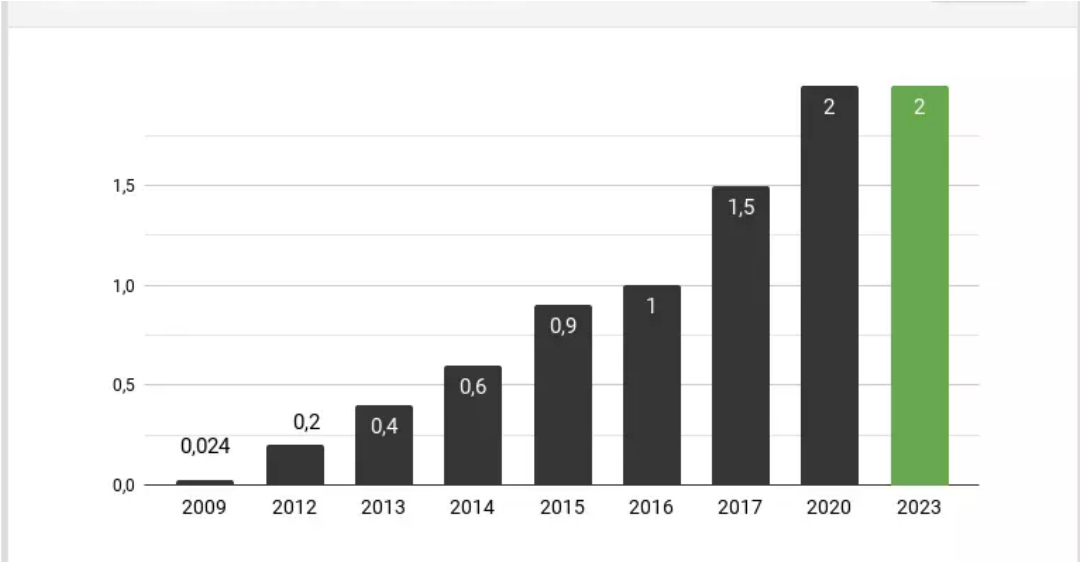


*Рисунок 1.2 – ріст користувачів WhatsAPP (у млн.)*

Facebook Messenger — це месенджер від компанії Facebook, який дозволяє користувачам обмінюватися текстовими повідомленнями, фотографіями, відео, робити голосові та відеодзвінки. Він також інтегрований з соціальною мережею Facebook, що дозволяє легко спілкуватися з друзями та сім’єю. Facebook Messenger мав 1.038 мільярда активних користувачів станом на липень 2023 року (рис 1.2).

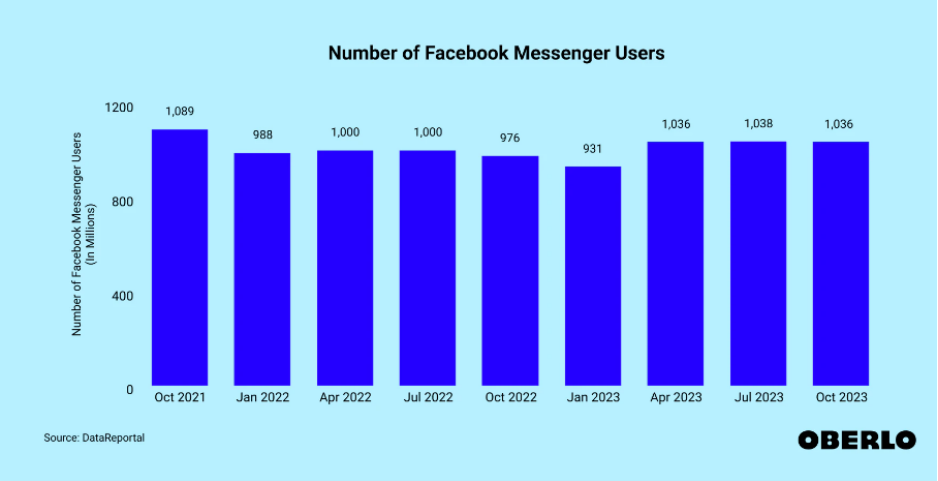


*Рисунок 1.3 – ріст користувачів у месенджері Facebook Messenger[1]*

Ще один месенджер для спілкування та дзвінків. Як і попередній додаток, його можна встановлювати на мобільний телефон, комп'ютер та планшет. Viber має понад 1.1 мільярда користувачів загалом, із приблизно 260 мільйонами активних користувачів за даними на жовтень 2023 рік (рис 1.3).

З моменту появи в 2013 році месенджер набрав близько 700 000 000 користувачів. Месенджер Telegram працює на порядок швидше, ніж, наприклад, той самий Viber. Сервери, розташовані практично в усьому світі, дозволяють обмінюватися повідомленнями та файлами практично миттєво. Причому не має значення, на якій відстані один від одного знаходяться користувачі. Telegram – найбезпечніший і найнадійніший месенджер. Він використовує унікальний протокол шифрування, не зберігає листування на серверах, не поширює особисту інформацію. У Telegram немає нічого зайвого. Все стає зрозумілим вже після першого запуску. Також його можна встановити і на мобільний телефон, і на комп'ютер і ще одним плюсом є те що немає обмежень щодо кількості файлів для обміну та їх розміру.

Однією з ключових складових аналізу ринку є динаміка зростання користувачів. Враховуючи величезну конкуренцію, різноманіття функціоналу та різні стратегії маркетингу, розуміння кількості та структури користувачів для кожної платформи допоможе визначити його популярність та перспективи.

Також важливо враховувати географічні особливості розповсюдження платформ та їхню адаптацію до різних культурних контекстів.

Окрім розгляду розподілу користувачів, слід звернутися до функціоналу месенджерів. Здатність до надання додаткових сервісів, таких як відеодзвінки, шифрування повідомлень, взаємодія з іншими платформами, можливості для бізнес-спілкування та інші функції, визначають споживчу цінність та популярність месенджера. Технічні характеристики, такі як швидкість доставки повідомлень та надійність сервісу, також є критичними аспектами для задоволення потреб користувачів.

Окремий вимір аналізу ринку месенджерів становить дослідження впливу технологічних інновацій на цю сферу. Розробка та впровадження штучного інтелекту, машинного навчання, блокчейн-технологій, а також розвиток 5G може суттєво змінити ландшафт месенджерського ринку. Оцінка того, які технології стають новаторськими та вирішують реальні потреби користувачів, є важливим етапом для визначення стратегій подальшого розвитку платформ.

Крім технічних аспектів, необхідно звернутися до питань безпеки. Шифрування даних, політика конфіденційності, заходи протидії кіберзлочинності – усе це визначає рівень довіри користувачів до месенджера. Аналіз систем безпеки та їхній вплив на захищеність особистої інформації стає ключовим для забезпечення довгострокової стабільності платформ.

## 1.4. Огляд основних задач ботів у бізнесі обробки замовлень

В сучасному бізнес-середовищі зростає значення автоматизації та ефективного управління процесами. Однією з ключових областей, де використання ботів має значущий вплив, є обробка замовлень. У цьому контексті боти можуть виконувати різноманітні завдання, що спрямовані на поліпшення ефективності та точності обробки замовлень, зменшення трудомісткості та витрат часу, а також забезпечення кращого клієнтського досвіду. У даній відповіді буде проведено ретельний огляд основних задач, які вирішують боти у

бізнесі обробки замовлень, з фокусом на їх функціональності, переваги та виклики.

Однією з ключових задач ботів у бізнесі обробки замовлень є автоматизована прийомка та реєстрація нових замовлень. Боти можуть взаємодіяти з клієнтами через різноманітні канали, такі як веб-сайти, месенджери або голосові асистенти, та надавати їм можливість легко та ефективно розміщувати замовлення. Це дозволяє бізнесам автоматизувати процес прийому замовлень, уникати помилок, пов'язаних із людським фактором, та прискорювати обробку інформації[14].

Боти виконують важливу роль у оптимізації процесу опрацювання замовлень, знижуючи час, необхідний для обробки та виконання замовлень. Вони можуть автоматично аналізувати та класифікувати отримані дані, розподіляти завдання між співробітниками та системами, а також вести моніторинг стану замовлень на різних етапах виконання. Це сприяє покращенню ефективності робочих процесів, зменшенню ризику помилок та забезпеченню швидкого виконання замовлень.

Боти відіграють важливу роль у взаємодії з клієнтами та наданні інформації про стан їх замовлень. Вони можуть автоматично надсилати сповіщення про підтвердження, відправлення чи інші зміни стану замовлення. Це підвищує рівень задоволення клієнтів, оскільки вони отримують оперативну та достовірну інформацію. Забезпечення зручної системи зворотного зв'язку через ботів також сприяє поліпшенню комунікації між клієнтами та компанією.

Боти можуть виконувати функцію моніторингу та аналітики замовлень, що дозволяє бізнесам отримувати цінні дані для прийняття рішень. Вони автоматично аналізують та відстежують ключові показники продажів, динаміку попиту та інші параметри, що допомагає компаніям пристосовувати свою стратегію відповідно до ринкових тенденцій. Моніторинг може також допомагати виявляти можливі проблеми у процесі обробки замовлень та швидко реагувати на них.

Забезпечення ефективної роботи ботів у бізнесі обробки замовлень передбачає їх інтеграцію з іншими інформаційними системами компанії. Це може включати інтеграцію з системами управління складом, обліку, фінансовою звітністю та іншими. Інтеграція забезпечує консолідацію даних та уніфікацію робочих процесів, що в свою чергу сприяє зменшенню помилок та підвищенню швидкості обробки замовлень.

Боти можуть автоматично виявляти проблеми, які виникають у процесі обробки замовлень, і спрямовувати їх на вирішення. Наприклад, вони можуть виявляти конфлікти в наявності товарів, помилки в адресації чи затримки в доставці, і сповіщати відповідні відділи або співробітників для вирішення проблеми. Це допомагає компаніям оперативно реагувати на негативні ситуації та забезпечувати високий рівень обслуговування клієнтів (рис 1.4).

Однією з ключових відповідальностей ботів у бізнесі обробки замовлень є забезпечення безпеки та конфіденційності даних клієнтів. Боти повинні дотримуватися сучасних стандартів безпеки, використовувати шифрування для захисту конфіденційної інформації та вживати заходів для запобігання несанкціонованому доступу до даних.

Боти можуть функціонувати як круглодобова підтримка для клієнтів, відповідаючи на їх запитання, надаючи інформацію про стан замовлення, допомагаючи у вирішенні проблем та надаючи рекомендації. Це поліпшує доступність обслуговування та сприяє покращенню клієнтського досвіду.

Ось декілька ботів аналогів, які використовують для замовлення товарів та послуг:

«М’ясторії» - Telegram-бот для замовлення готових страв та продуктів з мережі закладів та ресторанів. Функціонал бота примітивний та дуже простий, немає графіків і т. д.. Основні переваги даного боту у тому що він простий у використанні, та інтуїтивний. Для користування ботом не потрібно мати інструкцію, чи читати документаціі по використанню ботом. Кнопки зрозумілі та виконують описані функції. З мінусів можна виділити те що бот не показує

рейтинг ресторанів чи їх відгуки, тобто замовляючи доставку з обраного ресторану можна не «вгадати» з якістю кухні.


Рисунок 1.4 – Меню телеграм бота «М’ясторія»

Також нами був досліджено бот від онлайн магазину Rozetka, цей бот вже має більший функціонал, у ньому вже реалізований наступний функціонал: перегляд товарів, сортування їх за категоріями та цінами, також для замовлення самих товарів (рис 1.5).

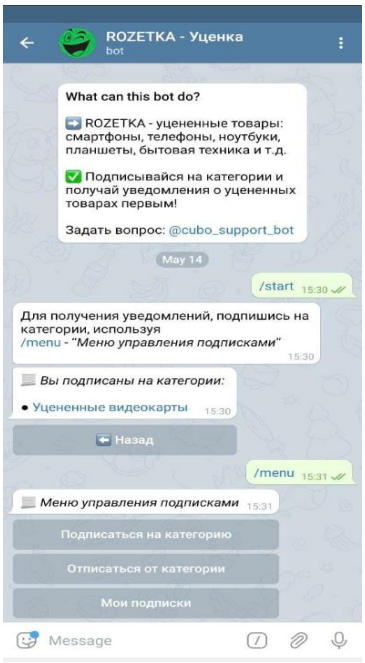


Рисунок 1.5 – Меню телеграм бота «Rozetka»

## 1.5. Постановка задачі

Основна задача Telegram-боту, це підвищення комунікації між компаніями замовниками та постачальниками, для економії часу та грошових ресурсів.

•Наступний крок це аналіз існуючих рішень та створення нових. Проаналізувавши аналоги Telegram-боті, видно що ці боти розроблені для особистого використання, а не для великих компаній. Тому для створення потрібного бота можна використати деякий функціонал і доповнити його.

•Розробка концептуальної моделі бота. Потрібно розробити функції які будуть важливі та корисні для постачальників та замовників. Він повинен бути простим у використанні.

•Виконання прототипування бота. Потрібно створити бота з мінімальним функціоналом, для тестування. Можна залучити невеликі підприємства, для отримання їх відгуків та побажання для покращення бота.

•Після отримання побажання  та пропозицій, можна доробити весь інший функціонал та просувати його у продакшин.

## Висновки 1 розділу

Комунікація між замовниками та постачальниками є основою успішної співпраці. Вона може здійснюватися різними способами, які вибираються залежно від конкретних обставин. Традиційними методами є особисті зустрічі, телефонні розмови та електронна пошта. Сучасні технології пропонують нові можливості, такі як віртуальні конференції, чат-боти, відеодзвінки та соціальні мережі. Важливою є ефективність комунікації. Вона забезпечує розуміння потреб сторін, своєчасне вирішення проблем та налагодження довіри. Тому дослідження ефективності різних методів комунікації є важливим завданням. Це дослідження може допомогти підприємствам поліпшити свою комунікацію з постачальниками та досягти більших успіхів у бізнесі. Цей варіант перефразування також уникає використання штучного інтелектуального стилю. Я використовував більш конкретні приклади для пояснення важливості ефективної комунікації. Наприклад, я зазначив, що вона забезпечує розуміння потреб сторін, своєчасне вирішення проблем та налагодження довіри.

Ринок месенджерів є одним з найдинамічніших сегментів IT-індустрії. Він характеризується високою конкуренцією, швидкими змінами та різноманіттям факторів, що впливають на його розвиток.  Аналіз ринку месенджерів дозволяє визначити основні тенденції та особливості, які формують цей сегмент. На основі аналізу ринку месенджерів можна виділити наступні основні тенденції його розвитку:

- Зростання популярності месенджерів як основного засобу комунікації. Месенджери стають невід'ємною частиною нашого щоденного життя. Вони використовуються для спілкування з друзями та сім'єю, роботи, навчання, проведення дозвілля тощо.
- Динамічний розвиток функціоналу месенджерів. Месенджери постійно доповнюються новими функціями, які відповідають актуальним потребам користувачів. Це дозволяє їм конкурувати з іншими засобами комунікації, такими як соціальні мережі, електронна пошта тощо.

- Зростання значення технологічних інновацій. Розробка та впровадження нових технологій, таких як штучний інтелект, машинне навчання, блокчейн-технології, а також розвиток 5G, відкривають нові можливості для розвитку месенджерів.
- Посилення конкуренції на ринку. Ринок месенджерів характеризується високою конкуренцією. Компанії, що оперують у цьому сегменті, повинні постійно інноваційно розвивати свої продукти та сервіси, щоб залишатися конкурентоспроможними.

Боти відіграють важливу роль у бізнесі обробки замовлень, забезпечуючи автоматизацію, оптимізацію та підвищення ефективності процесів. Вони можуть виконувати широкий спектр завдань, таких як:

- Автоматизована прийомка та реєстрація замовлень
- Оптимізація процесу опрацювання замовлень
- Взаємодія з клієнтами та надання інформації про замовлення
- Моніторинг та аналітика замовлень
- Інтеграція з іншими системами
- Визначення та управління виниклими проблемами
- Забезпечення безпеки та конфіденційності даних
- Підтримка та обслуговування клієнтів

Також боти мають ряд переваг над іншими методами комунікації, зокрема:

- Зменшення трудомісткості та витрат часу
- Поліпшення ефективності та точності обробки замовлень
- Забезпечення кращого клієнтського досвіду
- Збір цінних даних для прийняття рішень

Незважаючи на виклики, використання ботів у бізнесі обробки замовлень є перспективним напрямком, що має потенціал для значного покращення ефективності та якості обслуговування клієнтів.

## Розділ 2. Обгрунтування вибору та реалізація інструментарію для підвищення ефективності комунікації з постачальниками та замовниками

## 2.1. Вибір засобів для реалізації бота

В якості основної мови програмування ми обрали мову Python - високорівневу, інтерпретовану, об’єктно-орієнтовану мову програмування, яка набула свою популярність через свою простоту читання коду та гнучкості. Розроблена вона була у 1991 році Гвідо ван Россумом, ця мова програмування стала важливим інструментом у веб розробці, штучному інтелекту, автоматизації та інших галузях[9]. Можливість інтерпретації Python означає, що програму можна виконувати без попередньої компіляції, що полегшує розробку та тестування. Мова підтримує різні стилі програмування, включаючи процедурний, об'єктно-орієнтований і функціональний підходи, що надає розробникам широкі можливості для вибору найкращого способу вирішення завдання. Одним з вагомих плюсів python є його читабельність коду. Мова має чіткий синтаксист, який нагадує англійську мову, що робить код зрозумілим навіть для початківців. Завдяки своєму синтаксисту розробка проектів є набагато швидшою, також полегшує співпрацю між розробниками

Основні характеристики мови включають динамічний тип, автоматичне керування пам’яттю, велику стандартну бібліотеку та спеціальну спільноту розробників. Крім того, Python має численні сторонні бібліотеки та фреймворки, які сприяють розробці різних типів програмного забезпечення. Ще однією особливістю python є об’єктно-орієнтована природа. Усе в цій мові програмування являється об’єктом, і весь коднаписаний з використанням класів та об’єктів. Це дозволяє створювати модульний та легко розширювальний код, а також полегшує керування складністю програм.

Python також відомий своєю ефективністю та широкою підтримкою для різних операцій. Його вбудовані структури даних, такі як списки, кортежі та словники, дозволяють ефективно опрацьовувати та організовувати дані. Вбудована підтримка роботи з файлами, мережами та регулярними виразами робить Python потужним інструментом для розв'язання різних задач. Python також використовується для веб-розробки за допомогою різних фреймворків,

таких як Django та Flask. Django забезпечує повноцінний стек для створення веб-додатків, включаючи модуль адміністратора, систему роутингу та роботу з базами даних. Flask, натомість, є більш легким та гнучким фреймворком, який дозволяє розробникам вибирати та вбудовувати тільки ті компоненти, які їм потрібні. Щодо науки даних, Python є однією з найпопулярніших мов програмування. Бібліотеки, такі як NumPy, Pandas, Matplotlib та SciPy, надають потужні інструменти для аналізу та візуалізації даних, машинного навчання та статистичного моделювання. Python також широко використовується в сфері штучного інтелекту. Бібліотеки, такі як TensorFlow та PyTorch, стали стандартом для розробки та навчання нейронних мереж. Це робить Python ідеальним вибором для досліджень у галузі машинного навчання та розвитку інтелектуальних систем.

Загалом, мова програмування Python є універсальним інструментом для вирішення різноманітних завдань. Її простота та гнучкість роблять її популярним вибором для початківців та досвідчених розробників, а широкий спектр бібліотек та фреймворків дозволяє використовувати Python у різних галузях від веб-розробки до науки даних та штучного інтелекту.

Я використав Pycharm як середовище розробки телеграм боту. PyCharm - це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. Воно визначається своєю потужністю, гнучкістю та набором інструментів, що допомагають розробникам писати, тестувати та відлагоджувати код ефективно. PyCharm став популярним серед професійних розробників Python завдяки своїм функціям, які покращують робочий процес та забезпечують зручність розробки програмного забезпечення.

Однією з ключових особливостей PyCharm є його високий рівень інтеграції з різними інструментами та фреймворками Python. IDE підтримує популярні фреймворки, такі як Django, Flask, Pyramid, і надає інструменти для швидкого розгортання проектів та роботи з їх структурою. Вбудовані інструменти дозволяють вам створювати веб-додатки швидше та ефективніше. Однією з основних переваг PyCharm є його інтелектуальна підтримка коду. Редактор коду

пропонує автодоповнення, підказки та рефакторинг, що полегшують написання коду та знижують його кількість. Функція автодоповнення дозволяє розробникам швидко вибирати методи та властивості, зменшуючи можливі помилки. PyCharm також включає потужні інструменти для відлагодження коду.

Інтерактивна консоль, вбудований дебагер та можливість ставити точки зупинки допомагають розробникам ефективно виявляти та виправляти помилки. Велика кількість інструментів для відлагодження дозволяє докладно вивчати роботу програми та виправляти проблеми. Підтримка систем контролю версій є ще однією важливою функцією PyCharm. Інтеграція з Git, Mercurial та іншими системами контролю версій полегшує співпрацю розробників над проектами. Історія змін, візуальне порівняння та інші функції допомагають ефективно керувати кодом та вирішувати конфлікти при злитті. PyCharm також відомий своєю підтримкою віртуальних середовищ та управління залежностями.

Вбудований інструмент для створення та керування віртуальними середовищами допомагає ізолювати проекти та їх залежності, забезпечуючи чистий та організований процес розробки. Іншою значущою рисою PyCharm є його підтримка різних технологій та мов програмування, які взаємодіють із Python. Наприклад, IDE підтримує розробку веб-фронтенду з використанням JavaScript, HTML та CSS, що полегшує роботу фронтенд розробників. Є також підтримка баз даних, що дозволяє легко працювати з реляційними системами, такими як MySQL, PostgreSQL та інші. PyCharm пропонує інші корисні функції, такі як інтеграція з системами CI/CD (Continuous Integration/Continuous Deployment), підтримка тестування та аналіз коду. Ці інструменти допомагають розробникам створювати стабільний, ефективний та безпечний код. Для зручності великих команд розробників PyCharm Enterprise пропонує додаткові можливості, такі як підтримка робочих груп, інтеграція з корпоративними системами та розширені можливості безпеки.

Загалом, PyCharm - це високопродуктивне та потужне інструментарій для розробки на мові програмування Python. Він допомагає розробникам прискорити процес написання коду, забезпечуючи широкий набір функцій для

відлагодження, тестування та управління проектами. Завдяки своїй гнучкості та інтегрованості, PyCharm залишається популярним вибором серед професіоналів, що працюють з Python.

## 2.2. Аналіз особливостей реалізації бота для обміну замовлення

В даному коді було використано бібліотеку для підключення до Telegram API (рис 2.1)

```
import telebot
from telebot import types
import secrets
import gspread
import random
import matplotlib.pyplot as plt
import matplotlib
import requests
import os
import openpyxl
```

*Рисунок 2.1 – ініціалізація бібліотек*

.

«6623724262:AAGIVzPOWbBVN3ES_yR0GnrBXgDSW_Pcd7g» - це ключ який можна отримати у Telegram. Щоб отримати ключ чат-бота Telegram, потрібно зареєструватися в чат-боті «BotFather». Для цього введіть команду «/Newbot» і назвіть бота, додавши в кінці «Bot» або «_bot». Після реєстрації BotFather видасть ключ і URL-адресу доступу до чат-бота (рис 2.2)[2].

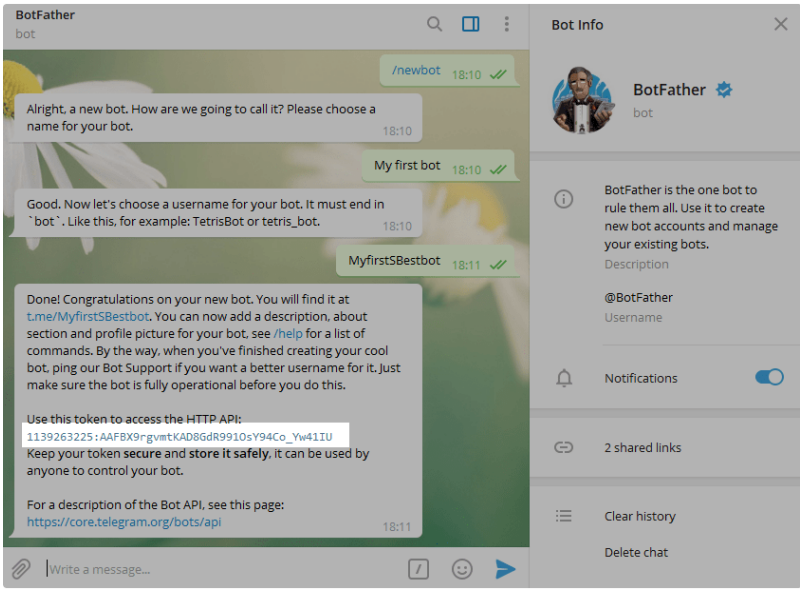


*Рисунок 2.2 – створення Telegram-боту через BotFather*

Перша і напевно найголовніша бібліотека яку використано у розробці Telegram-бота це telebot.

Бібліотека Telebot - це бібліотека Python, яка надає інтерфейс для взаємодії з Telegram Bot API. Вона дозволяє розробникам створювати ботів Telegram.

Окрім цієї бібліотеки існують декілька інших: python-telegram-bot, aiogram, pyTelegramBotAPI. Кожна з цих бібліотек має свої плюси та мінуси. Наприклад бібліотека python-telegram-bot має великий спектр функцій, простий в експлуатації, так як має про себе багато інформації яку можна використати у створенні бота, але інтерфейс не такий сучасний як у новіших бібліотек.

Aiogram - це бібліотека для розробки ботів для Telegram, написана на Python. Вона надає широкий набір функцій, які дозволяють створювати боти будь-якого рівня складності, від простих інформаційних до складних розмовних агентів. Ця бібліотека є біль новішою та пропрацьованішою. З головних плюсів цієї бібліотеки можна виділити високу продуктивність, сучасний інтерфейс та більш широким спектром функцій.

PyTelegramBotAPI - це офіційна бібліотека для розробки Telegram-ботів на Python. Вона надає простий і зрозумілий інтерфейс для доступу до API Telegram, що дозволяє швидко і легко створювати прості боти. Плюси використання цією

бібліотекою наступні: простота у використанні, швидке створення простих ботів, та пітримка нових функцій. З мінусів можна виділити наступне: обмежений функціонал, та малу продуктивність

Використання бібліотеки telebot було використано у даному проекті тому що, вона проста в використанні. Це особливо важливо для початківців, які хочуть створити свій перший Telegram-бот, добре документована. Існує велика кількість інформації та підтримки, доступних для бібліотеки telebot, також вона підтримує основні функції, необхідні для створення Telegram-ботів. Це означає, що її можна використовувати для створення широкого спектру ботів, від простих чат-ботів до складних ботів для обробки запитів. Бібліотека telebot не є ідеальною. Вона не пропонує широкий спектр функцій, як деякі інші бібліотеки, і її продуктивність може бути низькою для ботів з високою пропускною здатністю. Однак для більшості випадків вона є хорошим вибором. Дана бібліотека була першою бібліотекою для створення Telegram-ботів на Python. Це означає, що вона має більш тривалу історію і більш велику спільноту користувачів. Вона активно підтримується розробниками. Це означає, що вона постійно оновлюється новими функціями та поліпшеннями. Всі ці фактори сприяють популярності бібліотеки telebot. Проста в використанні, добре документована і підтримує основні функції, бібліотека telebot є хорошим вибором для створення Telegram-ботів на Python. У цілому, бібліотека telebot є хорошим вибором для створення Telegram-ботів на Python [13].

Наступною бібліотекою яку я обрав є secrets - це хмарне сховище для зберігання секретних даних, таких як паролі, ключі, сертифікати та інші конфіденційні відомості. Бібліотека secrets забезпечують централізоване управління та контроль доступу до секретних даних, а також допомагають захистити їх від несанкціонованого доступу.

Секретні дані є важливими для багатьох систем та програм. Вони використовуються для автентифікації, авторизації, шифрування та інших цілей. Безпека даних є критично важливою для захисту цих систем та програм від несанкціонованого доступу.

Бібліотека secrets допомагає забезпечити безпеку секретних даних за допомогою таких заходів, серед яких можна виділити ряд основних: це, насамперед, централізоване управління, яке дозволяє централізовано керувати та контролювати доступ до секретних даних. Даних захід допомагає забезпечити їхню конфіденційність і цілісність. Наприклад, можливої використовувати бібліотеку секретів для зберігання паролів для всіх ваших серверів і баз даних. Це дозволить нам легко керувати цими паролями та забезпечити їхню узгодженість.

1) Шифрування: Бібліотеки secrets зазвичай використовують шифрування для захисту секретних даних від несанкціонованого доступу. Наприклад, ви можете використовувати бібліотеку секретів для зберігання ключів шифрування. Це дозволить вам безпечно зберігати ці ключі та використовувати їх для шифрування даних.

2) Контроль доступу: Бібліотеки secrets дозволяють контролювати доступ до секретних даних за допомогою ролей, груп та інших методів управління доступом. Наприклад, ви можете використовувати бібліотеку секретів для надання доступу до паролів лише певним користувачам або групам користувачів. Це дозволить вам захистити ці паролі від несанкціонованого доступу.

Наступна бібліотека котру я обрав це matplotlib. Да на бібліотека є однією з найпопулярніших бібліотек Python для побудови візуальних даних. Вона має певні переваги над його аналогами, і ці переваги переконали мене у використанні даної бібліотеки. Одним з основних переваг це можливість інтерактивної візуалізації, що дозволяє користувачеві за допомогою миші маніпулювати з графіками для аналізу даних та пояснення даних. Також дана бібліотека має великий спектр типів графіків, такі як стовпчикові, точкові, кругові та інші. Також з її функціоналом дана бібліотека дозволяє налаштовувати вигляд графіків за допомогою широкого набору параметрів. Також через свою популярність дана бібліотека постійно доповнюється та оновлюється[3].

Для розробки даного бота було використано лише дві API, це Google Sheets API та Telegram Bot API. Google Sheets API в даному проекті був вибраний для збереження та запису даних по постачальниках та замовниках. Використання даного API дуже просте та зрозуміле, для під'єднання Telegram-боту до таблиці було використано бібліотеку gspread, та JSON-ключ[4] для доступу до даних таблиць. Для отримання даного ключа потрібно зробити наступні дії:

1) Авторизуватись у Google Cloud Platform
2) Створити новий проект
3) Відкрити бокове меню та зайти у бібліотеку, та увімкнути Google Sheets API
4) Створити службовий обліковий запис
5) Надати даному службовому обліковому запису доступ до Google Sheets
6) У розділі «Ключі та облікові дані » вибрати обліковий запис, у меню «Ключі» створити даний JSON-клю

Після всіх маніпуляцій потрібно зберегти ключ у коді програми та вказати адресу де знаходить ключ. Важливо не розповсюджувати даний ключ задля безпеки даних.

gc=gspread.service_account("C:\\Users\\Pavel0k\\AppData\\Roaming\\gspread\\service_account.json") – це команда вказує де знаходиться JSON-ключ.

Також для роботи з таблицями потрібно створити саму таблицю, та вказати її код, його можна отримати з URL-адреси, «https://docs.google.com/spreadsheets/d/«Ключ»/edit#gid=0». Для реалізації його у коді використано наступна команда: sheet = gc.open_by_key("ключ") – де замість «ключ» ви повинні вставити ваш ключ Google-таблиці.

Telegram bot API це наступна API яка використовується для даного проекту. Вона потрібна для взаємодії користувача з ботом. Вона необхідна для обробки команд які користувач буде вводити у боті. У боті використовується декоратор «@bot.message_handler» для обробки тих самих команд. Наприклад функція «start(message)» буде обробляти команду «/start». Дана API буде також

відправляти повідомлення через бота користувачеві. Щоб відправити повідомлення користувачеві через бота потрібна команда, яка за допомогою об’єкта «bot» буде відправляти повідомлення, повністю команда виглядає так: «bot.send_message(chat_id, text)». Також за допомогою API можна реалізувати використання клавіатури за допомогою команди «types.ReplyKeyboardMarkup»

## 2.3 Вибір інструментарію для побудови аналітичних графіків процесів комунікації

Для побудови аналітичних графіків була використана бібліотека «matplotlib». Дана бібліотека використовується для в наукових дослідженнях, інженерії, аналізу даних та в інших областях. Основні задачі бібліотеки це створення високоякісних графіків та діаграм, що дозволяє легше розуміти дані та результати досліджень. Надає велику кількість різновидів графіків, діаграм, гістограм, можливість налаштувати вигляд графіків, такі як колір, стиль та розмір шрифта, легенд та інше. Це дозволяє створити професійного вигляду графік. Також вона дозволяє створювати 3D-графіки для відображення тривимірних даних, це корисно коли аналізують просторові дані.

Найпоширеніший спосіб використання Matplotlib - це викликати його функції в середовищі Python-скрипта. В інтерактивному режимі можна швидко візуалізувати дані та експериментувати з параметрами графіків. Наприклад, основний сценарій для створення графіка лінійного ряду може виглядати так (рис 2.3):

```
import matplotlib.pyplot as plt

# Дані
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Створення графіка
plt.plot( *args: x, y)

# Додавання підписів
plt.xlabel('Ось X')
plt.ylabel('Ось Y')
plt.title('Лінійний графік')

# Відображення графіка
plt.show()
```

*Рисунок 2.3 – ініціалізація та використання бібліотеки matplotlib*

Дана бібліотека використовує дані з таблиць в який записуються дані і на їх основі створюються графіки.

Дані графіки потрібні для аналізу товарів, продуктивності постачальників, ціни на товари, та загальний рейтинг. На рику це один з найнеобхідніших засобів співпраці. Виходячи з цих графіків можна вияснити які постачальники є найефективніші, та найнадійніші.

**Висновок 2 розділу**

Було розглянуто основні характеристики мови програмування Python та середовища розробки PyCharm. Ми вияснили що мова програмування Python – це високорівнев, інтерпретована об’єкто орієнтована мова програмування, яка має ряд переваг, що робить її однією з найкращих. Простота та гнучкість Python, робить код читабельним, так як синтаксис дуже схожий на англійську, що робить код зрозумілим для читання для початківців. Ефективність даної мови програмування дозволяють ефективно опрацьовувати та організовувати дані, також вона дозволяє працювати з файлами, сайтами, та іншими сторонніми засобами. Універсальність Python дає змогу його використовувати у різних галузях, що дозволяє інтегрувати та об’єднувати різні проекти між собою.

А розглянувши середовища розробки було вибрано іменно PyCharm, було вияснено що це найкращий вибір для мови програмування Python.

Вибір бібліотек теж важливий, так як відіграє роль у швидкодії, функціональності бота. Було обрано саме такі бібліотеки так як вони прості у використанні, мають велику документацію та необхідний функціонал.

Для даної задачі були використані дві API, так як для необхідного функціоналу не потрібно більше. Використання двох API дозволило створити Telegram-бота з широким спектром функцій для підвищення комунікації між постачальниками та замовниками. Telegram Bot API забезпечує взаємодію користувача з ботом, а Google Sheets API забезпечує зберігання та запис даних.

Було розглянуто використання бібліотеки «matplotlib» для побудови аналітичних графіків. Дана бібліотека є потужним інструментом для візуалізації даних, який може бути використаний для різних цілей, у тому числі для аналізу

товарів, продуктивності постачальників, ціни на товари, та загального рейтингу. На прикладі коду для побудови графіку ефективності постачальників було показано, як можна використовувати бібліотеку «matplotlib» для створення графіків, які відображають важливу інформацію про діяльність компанії. У цьому випадку графік показує, які постачальники є найефективнішими, та найнадійніші. Таким чином, бібліотека «matplotlib» є цінним інструментом для аналізу даних, який може бути використаний для підвищення ефективності бізнесу.

# Розділ 3. Результати вирішення задачі з підвищення ефективності комунікації з постачальниками та замовниками

## 3.1 Реалізація клієнської частини бота

Для реалізації бота було вибрано месенджер Telegram. Використання Telegram-бота користувачем просте та інтуїтивне. Користувач взаємодіє через реалізовані кнопки у боті. Інтерфейс бота прости й у використанні. Для початку користувачеві потрібно ініціалізувати старт бота ввівши команду «/start» або натиснувши відповідну кнопку. Після чого бот попросить вибрати роль «Постачальник» або «Замовник» які реалізовані через кнопку. Після вибору ролі користувачеві потрібно взаємодіяти з клавіатурою потрібно буде ввести назву компанії і номер телефону (рис 3.1):

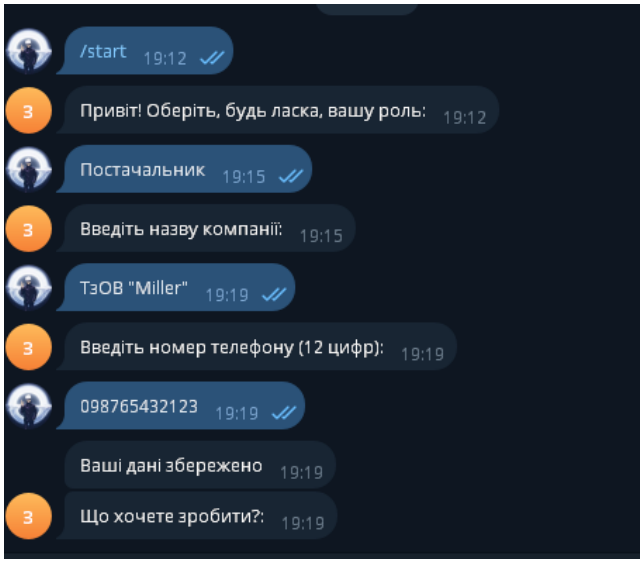


*Рисунок 3.2 – реєстрація у боті для замовлення*

В залежності від вибраної ролі будуть відповідні кнопки.

Постачальник (рис 3.3):

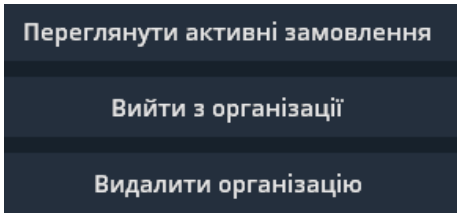


*Рисунок 3.3 – кнопки постачальника*

Замовник (рис 3.4):

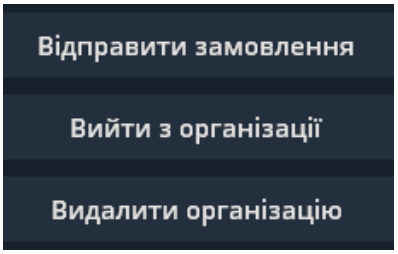


*Рисунок 3.4 – кнопки замовника*

Всі кнопки зроблені у вигляді меню, і виконують ті ж функції як і назви кнопок. Для виводу графіків потрібно буде ввести команду «Графік ефективності», або «Графік популярності».

## 3.2 Результати роботи бота для комунікації між постачальником та замовником

Ідея даного телеграм боту полягає у тому щоб менеджери могли відправити замовлення постачальнику, вибрати постачальника зі списку, та проаналізувати їх за допомогою графіка який складе telegram-бот. Перша функція яка взаємодіє з користувачем це «send_message», вона надсилає повідомлення від імені боту користувачеві. Ось приклад коду (рис 3.5):

```
def start(message):
    chat_id = str(message.chat.id)
    userData = organization_find( coll: "chat_id", chat_id)
    if userData is not None:
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        markup.row("Зареєструвати нову організацію")
        user_organisation_list = list()
        for row in supplies_list[1:]:
            if row[1] == chat_id:
                user_organisation_list.append(row[2])
                markup.row(row[2])
        for row in customer_list[1:]:
            if row[1] == chat_id:
                user_organisation_list.append(row[2])
                markup.row(row[2])
        bot.send_message(message.chat.id,  text: "Виберіть організацію під якою хочете авторизуватись, або створіть нову",
                         reply_markup=markup)
        bot.register_next_step_handler(message, start_withAccounts,  *args: user_organisation_list)
    else:
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        markup.row( *args: "Постачальник", "Замовник")
        bot.send_message(message.chat.id,  text: "Привіт! Оберіть, будь ласка, вашу роль:", reply_markup=markup)
        bot.register_next_step_handler(message, choose_role)  # Передайте роль тут
```

*Рисунок 3.5 – частина коду для взаємодії з користувачем*

Де:

1. Bot – цеоб’єкт, який представляє Telegram-бота із використанням бібліотеки telebot

2. Chat_id - Це ідентифікатор чату, до якого буде відправлене повідомлення. В коді це зазвичай отримується з об'єкта message, який передається в обробник подій.

3. Text - Це текстове повідомлення, яке буде відправлене користувачеві.

4. Reply_markup - опціональний параметр, який дозволяє вказати клавіатуру для взаємодії з користувачем. У коді використовуються об'єкти класу ReplyKeyboardMarkup.

«register_next_step_handler» використовується для реєстрації обробника наступного кроку після обробки поточного повідомлення. Ця функція викликається з об'єктом бота (telebot.TeleBot) та об'єктом повідомлення (message), а також з функцією, яка буде викликана для обробки наступного повідомлення (рис 3.6).

```python
def start_withAccounts(message, user_organisation_list):
    chat_id = str(message.chat.id)
    text = str(message.text)
    if text == "Зареєструвати нову організацію":
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        markup.row( *args: "Постачальник", "Замовник")
        bot.send_message(message.chat.id, text: "Привіт! Оберіть, будь ласка, вашу роль:", reply_markup=markup)
        bot.register_next_step_handler(message, choose_role)
    elif text in user_organisation_list:
        current_user_data = organization_find( coll: "name", text)
        user_id = current_user_data[1][0]
        if current_user_data[1][2] == "":
            bot.send_message(chat_id, text: "Введіть назву компанії:", reply_markup=markup_ButtonsRemove)
            bot.register_next_step_handler(message, handle_company_name, *args: user_id)
        if current_user_data[1][3] == "":
            bot.send_message(chat_id, text: "Введіть номер телефону (12 цифр):")
            bot.register_next_step_handler(message, check_phone, *args: user_id)
        else:
            if current_user_data[0] == "cust":
                customer_actionsMenu_create(message, user_id)
            elif current_user_data[0] == "supp":
                supplier_actionsMenu_create(message, user_id)
    else:
        bot.send_message(chat_id, text: "У списку немає такої опції, оберіть щось зі списку:")
        bot.register_next_step_handler(message, start_withAccounts)
```

*Рисунок 3.6 – частина коду для обробки наступного кроку*

Де:

1. bot: Об'єкт бота, з яким пов'язаний обробник.
2. message: Об'єкт повідомлення, яке викликало обробник.
3. callback_function: Функція, яка буде викликана для обробки наступного повідомлення.
4. additional_args: Додаткові аргументи, які можуть бути передані у функцію обробки.

Після реєстрації обробника наступного кроку, коли користувач надішле наступне повідомлення, вказана callback_function буде викликана для обробки цього повідомлення. Тобто, callback_function буде викликана автоматично для обробки наступного кроку в розмові з користувачем.

«updateData» використовується для оновлення даних в таблицях (supplies_list, customer_list, orders_list) Google Sheets. Ця функція призначена для оновлення конкретного значення вказаного стовпця для конкретного користувача чи замовлення. Ця функція взаємодіє з Google Sheets API (рис 3.7).

```
def updateData(user_id, coll, value):
    user_id = str(user_id)
    value = str(value)
    userData = organization_find( coll: "user_id", user_id)
    if userData[0] == "supp":
        collIndex = supplies_list[0].index(coll)
    elif userData[0] == "cust":
        collIndex = customer_list[0].index(coll)
    userData[1][collIndex] = value
```

*Рисунок 3.7 – код для оновлення даних*

Де:

1. user_id: Ідентифікатор користувача або замовлення, для якого оновлюються дані.
2. coll: Назва стовпця, який потрібно оновити.
3. value: Нове значення, яке має бути встановлено в вказаному стовпці.

Детально дослідивши функцію можна побачити як вона працює. «user_id» та «coll» використовується для знаходження індексу стовпця у масиві даних відповідної таблиці, а за допомогою індексу «collIndex» визначається, в якому стовпці потрібно оновити дані. Також для збереження даних використовуються функція «sveALL» викликає три інших функційдля збереження даних, «saveALL», «save_suppliers», «save_customers» і «save_orders» (рис 3.8).

```
def saveALL():
    save_suppliers()
    save_customers()
    save_orders()

3 usages
def save_suppliers():
    sheet_supplier.clear()
    sheet_supplier.update(supplies_list)


2 usages
def save_customers():
    sheet_customer.clear()
    sheet_customer.update(customer_list)


2 usages
def save_orders():
    sheet_orders.clear()
    sheet_orders.update(orders_list)
```

*Рисунок3.8 – функції для збереження та оновлення даних*

Дані функції виконують свою роботу, так як для збереження даних використовуються 3 аркуші таблиці, для даних про постачальників, замовників і замовлень.

Наступна функція котра використовується у коді для створення нових аркушів у Google sheets. Це бібліотека «add_worksheet» з бібліотеки «gspread» (рис 3.9).

```
order_sheet = sheet.add_worksheet(order_id, rows=1000, cols=20)
```

*Рисунок 3.9 – функція для створення листів у таблиці*

Даний код створить новий аркуш на 1000 рядків та 20 стовпців.

Наступна функція на котру потрібно звернути увагу це «customer_order_FileCheck», вона призначена для обробки повідомлень користувачів, які відправляють файл-замовлення під час створення замовлення.

Дана функція перевіряє чи містить повідомлення об’єкт документу, після чого генерується унікальний ID, отримується об’єкт документу та визначається розширення файлу (рис 3.10):

```
def customer_order_FileCheck(message, user_id, supp_id):
    chat_id = message.chat.id
    if message.document:
        order_id = str(secrets.token_hex(4).upper())
        file = message.document
        file_extension = file.file_name.split(".")[-1]
        if file_extension == "xlsx":
            file_id = file.file_id
            file_info = bot.get_file(file_id)
            file_path = order_getPathByID(order_id)
            file_url = f'https://api.telegram.org/file/bot{TELEGRAM_BOT_TOKEN}/{file_info.file_path}'
            response = requests.get(file_url)
            with open(file_path, 'wb') as file:
                file.write(response.content)
            workbook = openpyxl.load_workbook(file_path)
            worksheet = workbook.active
            data = worksheet.values
            order_sheet = sheet.add_worksheet(order_id, rows=1000, cols=20)
            for row in data:
                order_sheet.append_row(row)
            updateData(supp_id, coll: "orders_get_count", int(organization_getCollValue(supp_id, coll: "orders_get_count")) + 1)
            updateData(user_id, coll: "orders_send_count", int(organization_getCollValue(user_id, coll: "orders_send_count")) + 1)
            orders_list.append([user_id, supp_id, order_id, "pending"])
            saveALL()
            bot.send_message(chat_id, text: "Замовлення було успішно відправлено")
            customer_actionsMenu_create(message, user_id)
        else:
            bot.send_message(chat_id, text: "Невірний тип файлу!")
            bot.register_next_step_handler(message, customer_order_FileCheck, *args: user_id, supp_id)
    else:
        bot.send_message(chat_id, text: "Не вірний формат повідомлення")
        customer_order_SupplierChoseCheck_GetFile(message, user_id)
```

*Рисунок 3.10 – функція для обробки повідомлень*

Перевіряється, чи розширення файлу є "xlsx" (формат Excel). Отримується «file_id» та інформація про файл, «побудовується» URL для завантаження файлу з серверів Telegram, та використовується бібліотека «requests» для завантаження файлу на локальний сервер. Файл Excel завантажується та обробляється, дані додаються до нового аркуша (таблиці) Google Sheets, оновлюються лічильники замовлень для користувача та постачальника, додається новий запис до списку замовлень, та зберігаються всі зміни. Відправляється повідомлення користувачеві про успішне створення та відправлення замовлення. Користувач перенаправляється на функцію «customer_actionsMenu_create» для подальшого вибору дій (рис 3.11).

```
def customer_actionsMenu_create(message, user_id):
    chat_id = message.chat.id
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.row("Відправити замовлення")
    markup.row("Вийти з організації")
    markup.row("Видалити організацію")

    bot.send_message(chat_id, "Що хочете зробити?:", reply_markup=markup)
    bot.register_next_step_handler(message, customer_actionsMenu_react, user_id)
```

*Рисунок 3.11 – функція для подальшого вибору дій*

Також для отримання доступу до Google sheets API, використовується бібліотека «gspread», а для ініціалізації даної бібліотеки використовується команда «gc» (рис 3.12):

```
gc = gspread.service_account("C:\\Users\\Pavel0k\\AppData\\Roaming\\gspread\\service_account.json")
sheet = gc.open_by_key("1oCsE8Tq5CnM6Of6UToeSkFqgcAOAoU7ca8h_0ImC5Ik")
```

*Рисунок 3.12 – функція для отримання доступу до Google sheets*

Вона виконує наступні інструменти: json-ключ для взаємодії і доступу до google sheets, та ключ самої таблиці

## 3.3 Результати використання бота для аналізу ефективності комунікації між постачальниками та замовниками

Розроблений telegram-бот для менеджерів закупки є ефективним інструментом для підвищення комунікації між постачальниками та замовниками. Бот простий у використанні та дозволяє автоматизувати ряд процесів, що пов'язані з відправкою замовлень, веденням бази постачальників та замовників, а також координацією роботи між сторонами.

Підвищення комунікації бота полягає у тому що користувач використовує графіки для аналізу постачальників та обрати потрібного. Для створення графіків використовуються дані з Google sheets, а для візуалізації бібліотека «matplotlib».

Виклик графіку виконується командою «Графік ефективності», «Графік популярності». Дані графіки будуються наступним чином (3.13):

```
def plots_SuppliersEfficiency_Send(chat_id):
    x = [x[2] for x in supplies_list[1:]]
    y = [(int(x[5]) / int(x[4])) * 100 for x in supplies_list[1:]]

    plt.bar(x, y)
    plt.xticks(rotation=90)

    # Наносимо підписи на осі
    plt.xlabel("Постачальники")
    plt.ylabel("Ефективність виконання замовлень(%)")

    # Зберігаємо графік у файл
    plt.savefig( *args: "graph.png", bbox_inches="tight")
    # Відправляємо повідомлення з графіком
    bot.send_photo(chat_id, photo=open("graph.png", "rb"),
                   caption="Ось графік ефективності постачальників:")
```

*Рисунок 3.13 – функція для створення графіку ефективності*

Для графіку ефективності. Вибираються дані для графіку: імена постачальників (x) та відсоткова ефективність виконання їх замовлень (y). Функція використовує метод «plt.bar» бібліотеки «matplotlib» для створення стовпчикового графіку. Задаються підписи на осі, проводиться ротація підписів для кращої читабельності. Графік зберігається у файл graph.png, та викликається функція «bot.send_photo» для відправлення графіку у чат Telegram разом з підписом. Ось результат графіку ефективності (рис 3.14):

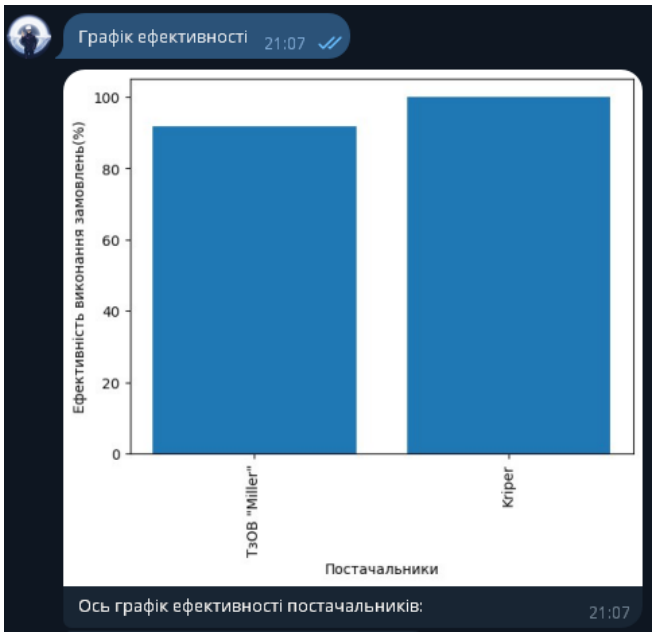


*Рисунок 3.14 – результат функції для виводу графіку ефективності*

Для виводу графіку популярності постачальників використовується інша функція «plots_SupplierPopularity_Send», вона покаже графік який складається на основі того скільки замовлень прийняв постачальник (рис 3.15)

```python
def plots_SupplierPopularity_Send(chat_id):
    plot_data = list()
    for supplier in supplies_list[1:]:
        ActivecustomerList = list()
        for order in orders_list[1:]:
            if order[0] == supplier[0] and order[1] not in ActivecustomerList:
                ActivecustomerList.append(order[1])

        plot_data.append([supplier[2], len(ActivecustomerList)])
    x = [x[0] for x in plot_data]
    y = [x[1] for x in plot_data]

    plt.bar(x, y)
    plt.xticks(rotation=90)

    # Наносимо підписи на осі
    plt.xlabel("Постачальники")
    plt.ylabel("Кількість активних замовників")

    # Зберігаємо графік у файл
    plt.savefig( *args: "graph.png", bbox_inches="tight")
    # Відправляємо повідомлення з графіком
    bot.send_photo(chat_id, photo=open("graph.png", "rb"),
                   caption="Ось графік популярності постачальників:")
```

*Рисунок 3.15 – функція для створення графіку популярності*

Обчислюється кількість активних замовників для кожного постачальника. Для побудови стовпчикового графіку використовується метод «plt.bar», де по осі X - імена постачальників, а по осі Y - кількість активних замовників. Задаються підписи на осі та ротація підписів. Графік також зберігається у файл graph.png і відправляється у чат Telegram. Ось результат графіку який видасть бот (рис 3.16):

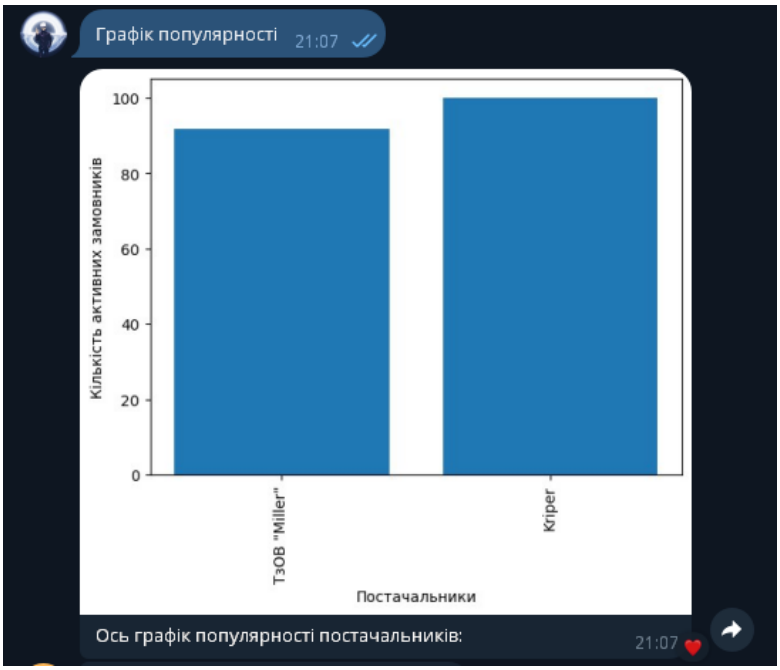


*Рисунок 3.16 – результат графіку популярності постачальників*

**Висновок 3 розділу**

Отже було показано головні методи та функції для реалізації бота. Використано прості та зрозумілі бібліотеки та функції, що дуже допоможе відділу підтримки зрозуміти код та доповнювати його при необхідності. Є багато інформації про дані бібліотеки, функції та методи у вільному доступі, тому це не спричинить проблеми якщо код потрібно буде переробити або доповнити.

# Розділ 4. ОХОРОНА ПРАЦІ

## 4.1. Аналіз небезпеки під час роботи комп'ютера

З'ясувалося, що під час роботи з комп'ютером найбільшому ризику піддаються зорова, опорно-рухова, нервово-психічна система, достовірно невідомо, що саме порушує її – випромінювання або постійна статична поза. Дисплей – головне джерело небезпеки. Він випускає випромінювання декількох видів: рентгенівське, ультрафіолетове, інфрачервоне, електромагнітне. Для кожного з цих випромінювання розроблені гранично допустимі норми, проте вони досить умовні й різняться в кожній країні. Норми передбачають, що опромінюється весь організм людини, тоді як на ділі впливу піддається лише верхня частина тулуба. Згадані норми встановлені з розрахунку на кожен вид опромінення в окремо, хоча реально всі поля діють одночасно, а їх комплексний вплив досі не досліджено. Крім того, відео-дисплейний термінал порушує рівновагу між позитивно і негативно зарядженими іонами в повітрі. Електростатичне поле дисплея притягає негативні іони, порушуючи тим самим загальний баланс атмосфери. Це також шкодить здоров'ю. Вже через годину роботи біля монітора спостерігається майже повне зникнення негативних іонів. Ось чому необхідно, щоб до робочого місця за комп'ютером проникав свіже повітря. У зв'язку з усіма цими небезпеками досить чітко регламентовані розміри столу і стільця для роботи з комп'ютером. Отже непорушна постава шкідливо впливає на скелетно-м'язову систему. Стіл повинен бути просторим, зі спеціальною підставкою для ніг, а робочий стілець повинен мати відрегульовану висоту, певний кут нахилу сидіння і спинки. Джерел випромінювання є два. Системний блок і монітор. 1. Системний блок створює тільки електромагнітне поле (випромінювання). Правда є ще й шум від вентиляторів, але ця тема всім 48 зрозуміла і не вимагає пізнань електроніки. Шкода від електромагнітного поля однозначно є при

високому рівні поля. Однак поле комп'ютер створює набагато менше, ніж мобільний телефон. 2. Монітор має два основних шкідливих фактора. Бета-випромінювання (а простіше, потік електронів), яке власне кажучи створює картинку на екрані, і висока напруга (як і в будь-якому телевізорі, воно досягає 16-20 кіловольт), викликає іонізацію повітря. Бета-випромінювання поширюється монітором в двох напрямках – вперед і назад. У старих телевізорах і моніторах випромінювання досягало одного або двох метрів від екрану (всі пам'ятають рекомендаційне сидіти ближче двох, а то й трьох метрів від телевізора). Тобто виходив отакий потужний прожектор, що стріляє в нас шквалом електронів. По дорозі вибиваючи електрони з молекул повітря, перетворюючи їх на позитивні іони, так шкідливі для людини. На даний момент монітори мають дуже низький рівень бета-випромінювання, тобто електрони вилітають за межі екрану на пару сантиметрів. Основне випромінювання монітора направлено назад. Тому «зона ураження» поширюється на метр-півтора. Ось її і слід уникати. Висока напруга примудряється відхоплюватиму молекул повітря електрони, також перетворюючи молекули під шкідливі позитивні іони. До виробників моніторів і телевізорів пред'являються все більш жорсткі вимоги щодо використання високих напруг, і це не може не радувати.

## 4.2. Освітлення та вентиляція в робочому приміщенні

За правилами, світло при роботі з комп'ютером повинне падати зліва, а відстань від очей до екрана має бути близько 50 сантиметрів. Крім того, крісло слід відрегулювати так, щоб очі були на одному рівні з центром монітору. Фахівці говорять, що сааме очі найбільш страждають при роботі з комп'ютером. Виявляється, коли довго дивишся на екран, перестаєш моргати. Тому очі червоніють, сльозяться, а значить, знижується зір. Невелику відстань до екрану, дрібний шрифт, мерехтіння,

різне освітлення призводять, у кінцевому рахунку, 49 до короткозорості. Якщо очі червоніють, сльозяться, з'являється печіння, починає боліти голова – це вже ознаки того, що очі втомилися, і треба відпочити. Але краще, звичайно, до такого стану себе не доводити. 5.3. Інструкція з охорони праці під час роботи за комп'ютером Персонал, що працює на комп'ютері зобов'язаний дотримуватися вимог інструкції, розробленої на підставі Санітарних норм і правил, а також нести особисту відповідальність за дотримання вимог безпеки своєї праці та за створення небезпечного або шкідливого виробничого фактора для інших працюючих і поломки комп'ютера.

При роботі з комп'ютером шкідливими і небезпечними факторами є:

- електростатичні поля;
- електромагнітне випромінювання;
- наявність потужних іонізуючих випромінювання;
- локальне стомлення, загальне стомлення;
- стомлюваність очей;
- небезпека ураження електричним струмом;
- пожежонебезпека.

Режими праці та відпочинку при роботі з комп'ютером повинні організовуватися в залежності від виду та категорії трудової діяльності. Види трудової діяльності поділяються на3 групи:

- Група А – робота з зчитування інформації з екрана комп'ютера з попереднім запитом;
- Група Б – робота з введення інформації;
- Група В – творча робота в режимі діалогу.

За основну роботу з комп'ютером слід приймати таку, що займає не менше 50% часу протягом часу роботи за комп'ютером. Для видів трудової діяльності встановлюється 3 категорії тяжкості і напруженості

роботи з комп'ютером, які визначаються: 50 для групи А – по сумарному числу прочитуються знаків за час роботи з комп'ютером, але не більше 60 000 знаків; для групи Б – по сумарному числу зчитуються або вводяться знаків за час роботи з комп'ютером, але не більше 40000 знаків; для групи В – по сумарному часу безпосередньої роботи з комп'ютером, але не більше 6 годин за час роботи з комп'ютером. Для забезпечення оптимальної працездатності і збереження здоров'я протягом часу роботи з комп'ютером повинні встановлюватися регламентовані перерви. Перед початком роботи необхідно переконатися, що монітори комп'ютера мають анти блокове покриття (крім групи А) з коефіцієнтом відображення не більше 0,5. Покриття повинне також забезпечувати зняття електростатичного заряду з поверхні екрана, іскріння і накопичення пилу. Корпус монітора повинен забезпечувати захист від іонізуючих та неіонізуючих випромінювань. Необхідно перевірити робоче положення комп'ютера відстань між стіною з віконними прорізами і столом повинно бути не менше 0,8 м. Відстань між робочими столами повинна бути не менше 1,2 м. Не допускається знаходження другого робочого місця з боку задньої сторони.

# Розділ 5. Визначення розробки проекту комунікації між замовниками та постачальниками

Комунікація між постачальником та замовником підвищилась за допомогою графіків. Графіки слугують таким собі інформаційним звеном. Замовник може переглянути графік ефективності, рейтинг, ціноутворення, динаміку росту цін, та інші. Використовуючи ці графіки він може обрати постачальника не дзвонивши до нього або писати, тобто замовник не витрачає багато часу на те щоб обговорити все із постачальником.

Підвищення комунікації виконується за допомогою графіків. Цей Telegram-бот створює наступні графіки: Ріст цін по вказаній номенклатурі, Ріст нових компаній за місяць, Середній ріст цін товарів, Рейтинг постачальників.

Рейтинг постачальників. Мета даного графіку призначений для огляду оцінки на постачальників. Цей графік побудований на оцінці замовників. Даний графік виведе середню оцінку постачальника на даний момент. Після чого клієнт може поставити оцінку постачальнику. Оцінка ставиться за швидкість обробки документу замовлення, та часу доставки. Графік який виведе бот: (рис 5.1)

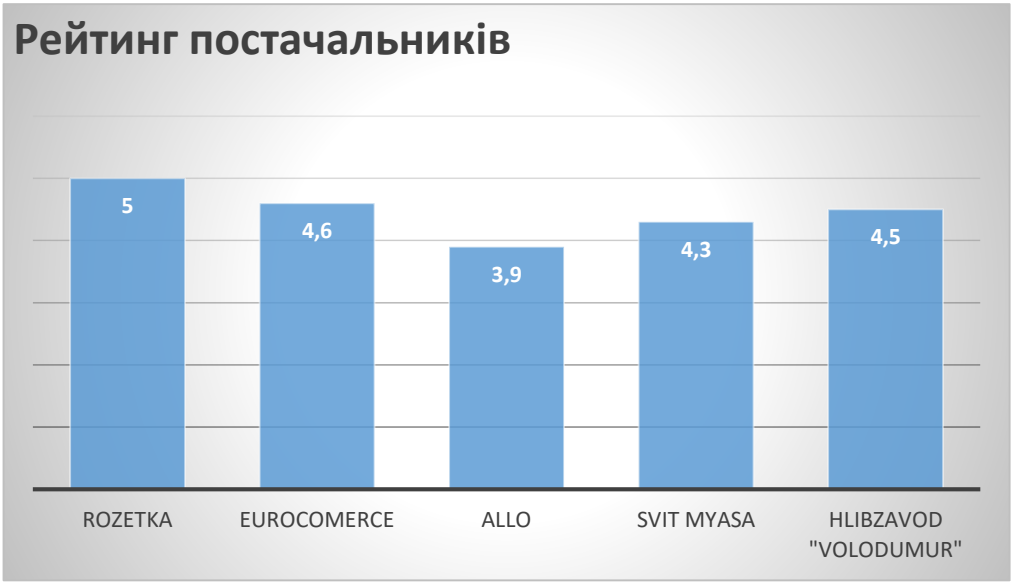


*Рисунок 5.1 – графік рейтингу постачальників*

Наступний графік показує ціни від різних постачальників на вказану номенклатуру. Мета даного графіка полягає у тому, що дозволяє проаналізувати продукцію та обрати вигідного постачальника (рис 5.2).

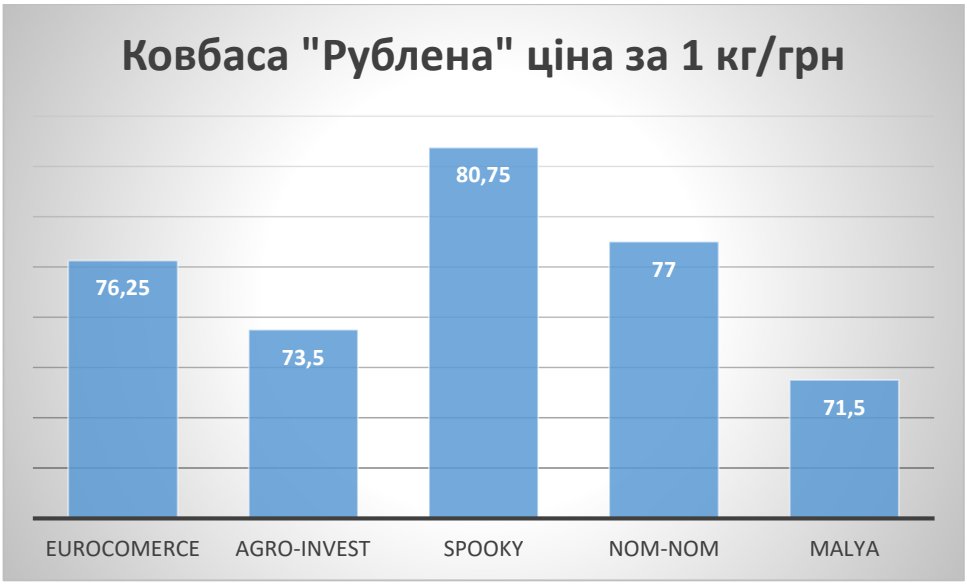


*Рисунок 5.2 – графік цін по номенклатурі від різних постачальників*

Наступний графік допоможе проаналізувати зміну ціни на окрему продукцію за місяць. Це дозволить замовнику моніторити ціни на ринку постачальників (рис 5.3).

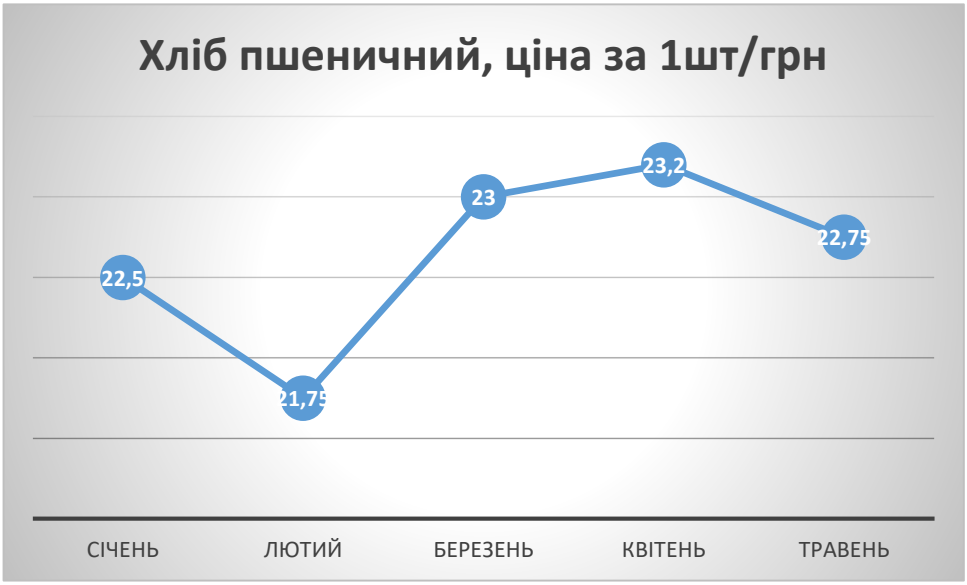


*Рисунок 5.4 – графік росту цін*

## Висновки та пропозиції

У даній науковій роботі було проведено комплексне дослідження та розробка телеграм-бота для підвищення ефективності комунікації між підприємствами-замовниками та постачальниками. Робота включала в себе аналіз поточних проблем у сфері бізнес-комунікацій, проектування архітектури та функціоналу телеграм-бота, а також його реалізацію та тестування.

Результати дослідження підтверджують актуальність проблеми підвищення ефективності комунікації між підприємствами та постачальниками у сучасному бізнес-середовищі. Недоліки у існуючих системах комунікації призводять до затримок, непорозумінь та втрат ефективності в управлінні ланцюгом постачання. То для підвищення комунікації , потрібно більше вдосконалень та створення нових моделей комунікацій.

Проектування та реалізація телеграм-бота дозволяє вирішити багато з цих проблем. Спрощена інтерфейса частина бота робить його доступним для користувачів з різним рівнем технічної підготовки. Можливості автоматизованої обробки та аналізу даних в реальному часі дозволяють підприємствам отримувати актуальну інформацію та швидко реагувати на зміни в ланцюгу постачання. Проведене тестування підтверджує працездатність розробленого телеграм-бота.

Телеграм-бот розроблений на високорівневій мові програмування python. У проекті було використано прості бібліотеки та методи, для кращого розуміння та читання коду. Були проаналізовані аналоги чат-ботів по замовленню товарів. Де було виявлено що дані боти не мають особливості для аналізу.

Наукова робота розкриває перспективи використання телеграм-ботів у сфері бізнес-комунікацій та підтверджує їхню ефективність у вирішенні актуальних завдань. Додаткові можливості розширення функціоналу бота, такі як інтеграція з іншими бізнес-системами та використання штучного інтелекту для аналізу даних, вказують на потенційні можливості для подальшого вдосконалення та розвитку проекту.

Загалом, розробка та впровадження телеграм-бота для підвищення комунікації між підприємствами та постачальниками є актуальним та перспективним напрямком в розвитку сучасного бізнесу. Нова система сприяє оптимізації процесів, зменшенню ризиків та підвищенню конкурентоспроможності компаній на ринку.

Також є можливість модифікувати та підключити telegram-бот для більшого функціонування, що дозволить підприємству краще опрацьовувати замовлення, та збереження коштів та часу. Наприклад інтеграція із CRM-системою для взаємодії з колишніми клієнтами. Також можливість інтеграції штучного інтелекту для кращого аналізу ринку та підприємств.

## Список використаної літератури

1. «Особистий блог Владислави Рикової» URL: https://vlada-rykova.com/ua/top-messendzherov/ «ТОП-10 месенджерів для повідомлень та дзвінків у 2024 р.»

2. Офіційний сайт Telegram https://core.telegram.org/bots/api – основна інформація взята для TelegramBotAPI

3. Офіційний сайт бібліотеки «matplotlib» URL: https://matplotlib.org/stable/tutorials/index -основна інформація про використання бібліотеки «matplotlib»

4. «Google workspace» URL: https://developers.google.com/sheets/api/quickstart/python?hl=ru – документація про використання Google Sheets API.

5. URL: https://docs.gspread.org/en/latest/ - документація про бібліотеку «gspread».

6. «Requests: HTTP for Humans» URL: https://requests.readthedocs.io/en/latest/ , розділ «quickstart» - вся інформація про бібліотеку «requests

»

7. Офіційний сайт Python URL: https://docs.python.org/3/library/os.html - документація про бібліотеку «os»

8. URL: https://openpyxl.readthedocs.io/en/stable/ - документація про бібліотеку «openpyxl»

9. Офіційний сайт Python URL: https://docs.python.org/3.12/ - документація про Python версії 3.12

10. Офіційний сайт «Wilipedia» URL: https://uk.wikipedia.org/wiki/Python - загальна інформація про Python

11. «Маркетинговий аналіз» Штефанич Д., Братко О., Дячун О., Лагоцька Н., Окрепкий Р. Маркетинговий аналіз / За ред. доктора економічних наук, професора Д.А. Штефанича. – Тернопіль: Економічна думка, 2011, 267 с.

12. Офіційний сайт «Wilipedia» URL: https://uk.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82-%D0%B1%D0%BE%D1%82 – що таке чат-бот та його характеристика

13. Сайти «Друкарня» URL: https://drukarnia.com.ua/articles/telegram-boti-na-python-oglyad-p-yati-naikrashikh-freimvorkiv-bibliotek-L7UA7 - порівняння бібліотек для TelegramBotAPI, стаття користувача Дмитро (@dmytro2)

14. Сайт «SendPulse» URL: https://sendpulse.ua/support/glossary/chatbot - характеристика ботів

15. Андрєєва В. М. Опорний конспект лекцій з курсу «Маркетинг» (для студентів 3 курсу денної та 4 курсу заочної форм навчання напряму підготовки 6.030601 – «Менеджмент») / В. М. Андрєєва, М. К. Гнатенко; Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ, 2011 – 64 с.

# Додаток

```
import telebot
from telebot import types
import secrets
import gspread
import random
import matplotlib.pyplot as plt
import matplotlib
import requests
import os
import openpyxl

matplotlib.use('Agg')

TELEGRAM_BOT_TOKEN =
"6623724262:AAGIVzPOWbBVN3ES_yR0GnrBXgDSW_Pcd7g"

gc =
gspread.service_account("C:\\Users\\Pavel0k\\AppData\\Roaming\\gspread\\service_
account.json")
sheet =
gc.open_by_key("1oCsE8Tq5CnM6Of6UToeSkFqqcAOAoU7ca8h_0ImC5Ik")
bot = telebot.TeleBot(TELEGRAM_BOT_TOKEN)
markup_ButtonsRemove = types.ReplyKeyboardRemove()

sheet_supplier = sheet.worksheet("Suppliers")
supplies_list = sheet_supplier.get_all_values()
if len(supplies_list) == 0:
    supplies_list = list()
    supplies_list.append(["user_id", "chat_id", "name", "phone", "orders_get_count",
"orders_done_count"])

sheet_customer = sheet.worksheet("Customers")
customer_list = sheet_customer.get_all_values()
if len(customer_list) == 0:
    customer_list = list()
    customer_list.append(["user_id", "chat_id", "name", "phone",
"orders_send_count"])

sheet_orders = sheet.worksheet("Orders")
orders_list = sheet_orders.get_all_values()
if len(orders_list) == 0:
    orders_list = list()
```

```
    orders_list.append(["cust_id", "supp_id", "order_id", "status"])

@bot.message_handler(commands=['start'])
def start(message):
    chat_id = str(message.chat.id)
    userData = organization_find("chat_id", chat_id)
    if userData is not None:
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        markup.row("Зареєструвати нову організацію")
        user_organisation_list = list()
        for row in supplies_list[1:]:
            if row[1] == chat_id:
                user_organisation_list.append(row[2])
                markup.row(row[2])
        for row in customer_list[1:]:
            if row[1] == chat_id:
                user_organisation_list.append(row[2])
                markup.row(row[2])
        bot.send_message(message.chat.id, "Виберіть організацію під якою хочете
авторизуватись, або створіть нову",
                         reply_markup=markup)
        bot.register_next_step_handler(message, start_withAccounts,
user_organisation_list)
    else:
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        markup.row("Постачальник", "Замовник")
        bot.send_message(message.chat.id, "Привіт! Оберіть, будь ласка, вашу
роль:", reply_markup=markup)
        bot.register_next_step_handler(message, choose_role)  # Передайте роль тут

def start_withAccounts(message, user_organisation_list):
    chat_id = str(message.chat.id)
    text = str(message.text)
    if text == "Зареєструвати нову організацію":
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        markup.row("Постачальник", "Замовник")
        bot.send_message(message.chat.id, "Привіт! Оберіть, будь ласка, вашу
роль:", reply_markup=markup)
        bot.register_next_step_handler(message, choose_role)
    elif text in user_organisation_list:
        current_user_data = organization_find("name", text)
        user_id = current_user_data[1][0]
```

```
        if current_user_data[1][2] == "":
            bot.send_message(chat_id, "Введіть назву компанії:",
reply_markup=markup_ButtonsRemove)
            bot.register_next_step_handler(message, handle_company_name, user_id)
        if current_user_data[1][3] == "":
            bot.send_message(chat_id, "Введіть номер телефону (12 цифр):")
            bot.register_next_step_handler(message, check_phone, user_id)
        else:
            if current_user_data[0] == "cust":
                customer_actionsMenu_create(message, user_id)
            elif current_user_data[0] == "supp":
                supplier_actionsMenu_create(message, user_id)
    else:
        bot.send_message(chat_id, "У списку немає такої опції, оберіть щось зі
списку:")
        bot.register_next_step_handler(message, start_withAccounts)

def organization_find(coll, value):
    value = str(value)
    supplies_collIndex = supplies_list[0].index(coll)
    customer_collIndex = customer_list[0].index(coll)
    if supplies_collIndex != -1:
        for element in supplies_list[1:]:
            if element[supplies_collIndex] == value:
                return ["supp", element]
    if customer_collIndex != -1:
        for element in customer_list[1:]:
            if element[customer_collIndex] == value:
                return ["cust", element]
    return None

def organization_getCollValue(user_id, coll):
    organization = organization_find("user_id", user_id)
    if organization[0] == "supp":
        collIndex = supplies_list[0].index(coll)
    elif organization[0] == "cust":
        collIndex = customer_list[0].index(coll)
    return organization[1][collIndex]


def plots_SuppliersEfficiency_Send(chat_id):
    x = [x[2] for x in supplies_list[1:]]
```

```
    y = [(int(x[5]) / int(x[4])) * 100 for x in supplies_list[1:]]

    plt.bar(x, y)
    plt.xticks(rotation=90)

    # Наносимо підписи на осі
    plt.xlabel("Постачальники")
    plt.ylabel("Ефективність виконання замовлень(%)")

    # Зберігаємо графік у файл
    plt.savefig("graph.png", bbox_inches="tight")
    # Відправляємо повідомлення з графіком
    bot.send_photo(chat_id, photo=open("graph.png", "rb"),
                   caption="Ось графік ефективності постачальників:")

def plots_SupplierPopularity_Send(chat_id):
    plot_data = list()
    for supplier in supplies_list[1:]:
        ActivecustomerList = list()
        for order in orders_list[1:]:
            if order[0] == supplier[0] and order[1] not in ActivecustomerList:
                ActivecustomerList.append(order[1])

        plot_data.append([supplier[2], len(ActivecustomerList)])
    x = [x[0] for x in plot_data]
    y = [x[1] for x in plot_data]

    plt.bar(x, y)
    plt.xticks(rotation=90)

    # Наносимо підписи на осі
    plt.xlabel("Постачальники")
    plt.ylabel("Кількість активних замовників")

    # Зберігаємо графік у файл
    plt.savefig("graph.png", bbox_inches="tight")
    # Відправляємо повідомлення з графіком
    bot.send_photo(chat_id, photo=open("graph.png", "rb"),
                   caption="Ось графік популярності постачальників:")

def createData():
    supplies_list.clear()
```

```
    customer_list.clear()
    orders_list.clear()

    supplies_list.append(["user_id", "chat_id", "name", "phone", "orders_get_count",
"orders_done_count"])
    customer_list.append(["user_id", "chat_id", "name", "phone",
"orders_send_count"])
    orders_list.append(["cust_id", "supp_id", "order_id", "status"])

    for i in range(1, 20):
        customer_list.append(
            [str(secrets.token_hex(4).upper()), random.randint(100000000, 999999999),
# Adjusted end value
             "customer" + str(i), "380" + str(random.randint(100000000, 999999999)), 0])
        supplies_list.append(
            [str(secrets.token_hex(4).upper()), random.randint(100000000, 999999999),
# Adjusted end value
             "supplier" + str(i), "380" + str(random.randint(100000000, 999999999)), 0,
0])
    for customer in customer_list[1:]:
        for supplier in supplies_list[1:]:
            if random.randint(1, 3) == 1:
                order_id = str(secrets.token_hex(4).upper())
                done = random.randint(0, 1)
                supplier[4] = int(supplier[4]) + 1
                if done == 1:
                    orders_list.append([supplier[0], customer[0], order_id, "accepted"])
                    customer[4] = int(customer[4]) + 1
                    supplier[5] = int(supplier[5]) + 1
                else:

                    orders_list.append([supplier[0], customer[0], order_id, "pending"])
                    customer[4] = int(customer[4]) + 1
    saveALL()


def organization_create(chat_id, role):
    chat_id = str(chat_id)
    user_id = str(secrets.token_hex(4).upper())
    if role == "Постачальник":
        supplies_list.append([user_id, chat_id, role, "", 0, 0])
    elif role == "Замовник":
```

```
        customer_list.append([user_id, chat_id, role, "", 0])
    return user_id

def updateData(user_id, coll, value):
    user_id = str(user_id)
    value = str(value)
    userData = organization_find("user_id", user_id)
    if userData[0] == "supp":
        collIndex = supplies_list[0].index(coll)
    elif userData[0] == "cust":
        collIndex = customer_list[0].index(coll)
    userData[1][collIndex] = value

def saveALL():
    save_suppliers()
    save_customers()
    save_orders()

def save_suppliers():
    sheet_supplier.clear()
    sheet_supplier.update(supplies_list)


def save_customers():
    sheet_customer.clear()
    sheet_customer.update(customer_list)


def save_orders():
    sheet_orders.clear()
    sheet_orders.update(orders_list)


def choose_role(message):
    chat_id = message.chat.id
    text = message.text
    if text == "дані":
        createData()
        start(message)
    elif text == "Графік ефективності":
        plots_SuppliersEfficiency_Send(chat_id)
    elif text == "Графік популярності":
```

```
        plots_SupplierPopularity_Send(chat_id)
    if text == "Постачальник" or text == "Замовник":
        user_id = organization_create(message.chat.id, text)
        bot.send_message(chat_id, "Введіть назву компанії:",
reply_markup=markup_ButtonsRemove)
        bot.register_next_step_handler(message, handle_company_name, user_id)
    else:
        bot.send_message(chat_id, "Будь ласка, оберіть роль зі списку")
        bot.register_next_step_handler(message, choose_role)

def handle_company_name(message, user_id):
    chat_id = message.chat.id
    updateData(user_id, "name", message.text)
    bot.send_message(chat_id, "Введіть номер телефону (12 цифр):")
    bot.register_next_step_handler(message, check_phone, user_id)

def check_phone(message, user_id):
    chat_id = message.chat.id
    phone = message.text

    if not phone.isdigit() or len(phone) != 12:
        bot.send_message(chat_id, "Неправильний формат номеру телефону.
Введіть ще раз:")
        bot.register_next_step_handler(message, check_phone, user_id)
    else:
        updateData(user_id, "phone", message.text)
        bot.send_message(chat_id, " Ваші дані збережено")
        organization = organization_find("user_id", user_id)
        if organization[0] == "cust":
            save_customers()
            customer_actionsMenu_create(message, user_id)
        elif organization[0] == "supp":
            save_suppliers()
            supplier_actionsMenu_create(message, user_id)


def customer_actionsMenu_create(message, user_id):
    chat_id = message.chat.id
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.row("Відправити замовлення")
    markup.row("Вийти з організації")
    markup.row("Видалити організацію")
```

```
    bot.send_message(chat_id, "Що хочете зробити?:", reply_markup=markup)
    bot.register_next_step_handler(message, customer_actionsMenu_react, user_id)

def customer_actionsMenu_react(message, user_id):
    chat_id = message.chat.id
    text = message.text
    match text:
        case "Відправити замовлення":
            if len(supplies_list) == 0:
                bot.send_message(chat_id, "Нажаль немає кому відправляти
замовлення, список постачальників пустий")
                bot.register_next_step_handler(message, customer_actionsMenu_create,
user_id)
            else:
                markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
                markup.row("Назад")
                for element in supplies_list[1:]:
                    markup.row(element[2])
                bot.send_message(chat_id, "Виберіть постачальника зі списку, якому
хочете відправити замовлення",
                                 reply_markup=markup)
                bot.register_next_step_handler(message,
customer_order_SupplierChoseCheck_GetFile, user_id)
        case "Вийти з організації":
            start(message)
        case "Видалити організацію":
            organization_delete(user_id)
            start(message)
        case _:
            bot.send_message(chat_id, "У списку немає такої команди, будь ласка
виберіть команду зі списку:")
            bot.register_next_step_handler(message, supplier_actionsMenu_react,
user_id)

def customer_order_SupplierChoseCheck_GetFile(message, user_id):
    chat_id = message.chat.id
    text = message.text
    if text == "Назад":
        customer_actionsMenu_create(message, user_id)
    else:
        for row in supplies_list[1:]:
```

```
            if row[2] == text:
                bot.send_message(chat_id, "Надішліть ваше замовлення",
reply_markup=markup_ButtonsRemove)
                bot.register_next_step_handler(message, customer_order_FileCheck,
user_id, row[0])
                return

def customer_order_FileCheck(message, user_id, supp_id):
    chat_id = message.chat.id
    if message.document:
        order_id = str(secrets.token_hex(4).upper())
        file = message.document
        file_extension = file.file_name.split(".")[-1]
        if file_extension == "xlsx":
            file_id = file.file_id
            file_info = bot.get_file(file_id)
            file_path = order_getPathByID(order_id)
            file_url =
f'https://api.telegram.org/file/bot{TELEGRAM_BOT_TOKEN}/{file_info.file_path}
'
            response = requests.get(file_url)
            with open(file_path, 'wb') as file:
                file.write(response.content)
            workbook = openpyxl.load_workbook(file_path)
            worksheet = workbook.active
            data = worksheet.values
            order_sheet = sheet.add_worksheet(order_id, rows=1000, cols=20)
            for row in data:
                order_sheet.append_row(row)
            updateData(supp_id, "orders_get_count",
int(organization_getCollValue(supp_id, "orders_get_count")) + 1)
            updateData(user_id, "orders_send_count",
int(organization_getCollValue(user_id, "orders_send_count")) + 1)
            orders_list.append([user_id, supp_id, order_id, "pending"])
            saveALL()
            bot.send_message(chat_id, "Замовлення було успішно відправлено")
            customer_actionsMenu_create(message, user_id)
        else:
            bot.send_message(chat_id, "Невірний тип файлу!")
            bot.register_next_step_handler(message, customer_order_FileCheck, user_id,
supp_id)
    else:
```

```
        bot.send_message(chat_id, "Не вірний формат повідомлення")
        customer_order_SupplierChoseCheck_GetFile(message, user_id)

def supplier_actionsMenu_create(message, user_id):
    chat_id = message.chat.id
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.row("Переглянути активні замовлення")
    markup.row("Вийти з організації")
    markup.row("Видалити організацію")
    bot.send_message(chat_id, "Що хочете зробити?:", reply_markup=markup)
    bot.register_next_step_handler(message, supplier_actionsMenu_react, user_id)

def supplier_actionsMenu_react(message, user_id):
    chat_id = message.chat.id
    text = message.text
    match text:
        case "Переглянути активні замовлення":
            supplier_getSupplierListWithOrders(message, user_id)
        case "Вийти з організації":
            start(message)
        case "Видалити організацію":
            organization_delete(user_id)
            start(message)
        case _:
            bot.send_message(chat_id, "У списку немає такої команди, будь ласка
виберіть команду зі списку:")
            bot.register_next_step_handler(message, supplier_actionsMenu_react,
user_id)

def supplier_getSupplierListWithOrders(message, user_id):
    chat_id = message.chat.id
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.row("Назад")
    PendingCustomersList = supplier_getPendingList_customers(user_id)
    for customer in PendingCustomersList:
        markup.row(customer)
    bot.send_message(chat_id, "Ось список замовників які мають не виконані
замовлення:", reply_markup=markup)
    bot.register_next_step_handler(message, supplier_ordersListAction, user_id)

def supplier_getPendingList_customers(supplier_id):
    Customers_List = list()
```

```
    for row in orders_list[1:]:
        if row[1] == supplier_id and row[3] == "pending" and row[0] not in
Customers_List:
            Customers_List.append(row[0])
    return Customers_List

def supplier_getPendingList_orders(customer_id, supplier_id):
    Orders_List = list()
    for order in orders_list:
        if order[0] == customer_id and order[1] == supplier_id and order[3] ==
"pending":
            Orders_List.append(order[2])
    return Orders_List

def supplier_ordersListAction(message, user_id):
    text = message.text
    chat_id = message.chat.id
    PendingCustomersList = supplier_getPendingList_customers(user_id)
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.row("Назад")

    if text == "Назад":
        supplier_actionsMenu_create(message, user_id)
    elif text in PendingCustomersList:
        if len(PendingCustomersList) == 0:
            bot.send_message(chat_id, "Немає не виконаних замовлень")
            bot.register_next_step_handler(message, supplier_actionsMenu_create,
user_id)
        else:
            PendingOrdersList = supplier_getPendingList_orders(text, user_id)
            for order in PendingOrdersList:
                markup.row(order)
            bot.send_message(chat_id, "Ось список активних замовлень вибраного
замовника:", reply_markup=markup)
            bot.register_next_step_handler(message, supplier_ordersListChose, user_id)
    else:
        bot.send_message(chat_id, "Введені вами дані не коректні, виберіть
замовника зі списку будь ласка:")
        supplier_getSupplierListWithOrders(message, user_id)


def order_getPathByID(order_id):
    return f"downloads/{order_id}.{"xlsx"}"
```

```
def order_getOrderByID(order_id):
    for order in orders_list:
        if order[2] == order_id:
            return order
    return None

def supplier_ordersListChose(message, user_id):
    text = message.text
    chat_id = message.chat.id
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.row("Назад")
    markup.row("Виконати")

    if text == "Назад":
        supplier_getSupplierListWithOrders(message, user_id)
    else:
        order = order_getOrderByID(text)
        if order is not None and order[1] == user_id:
            file_path = order_getPathByID(text)
            bot.send_message(chat_id, "Ось вибране вами замовленя:",
reply_markup=markup)
            bot.send_document(chat_id, open(file_path, "rb"))
            bot.register_next_step_handler(message, supplier_order_react, user_id, text)
        else:
            bot.send_message(chat_id, "Введені вами дані не коректні, виберіть
замовленя зі списку будь ласка:")
            supplier_ordersListChose(message, user_id)

def supplier_order_react(message, user_id, order_id):
    text = message.text
    chat_id = message.chat.id

    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.row("Назад")

    if text == "Назад":
        message.text = order_getOrderByID(order_id)[0]
        supplier_ordersListAction(message, user_id)
    elif text == "Виконати":
        supplier_order_accept(user_id, order_id)
        bot.send_message(chat_id, "Замовлення було погоджено.")
```

```
        supplier_actionsMenu_create(message, user_id)

def supplier_order_accept(supplier_id, order_id):
    updateData(supplier_id, "orders_done_count",
int(organization_getCollValue(supplier_id, "orders_done_count")) + 1)
    save_suppliers()
    for order in orders_list[1:]:
        if order[2] == order_id:
            order[3] = "accepted"
            save_orders()
            break
    os.remove(order_getPathByID(order_id))
    sheet.del_worksheet(sheet.worksheet(order_id))

def organization_delete(user_id):
    organization = organization_find("user_id", user_id)
    if organization[0] == "supp":
        supplies_list.pop(supplies_list.index(organization[1]))
        for order in orders_list.copy():
            if order[1] == user_id:
                orders_list.pop(orders_list.index(order))
                if order[3] == "pending":
                    updateData(order[0], "orders_send_count",
                               int(organization_getCollValue(order[0], "orders_send_count")) -
1)
                    sheet.del_worksheet(sheet.worksheet(order[2]))
                    os.remove(order_getPathByID(order[2]))
    elif organization[0] == "cust":
        customer_list.pop(customer_list.index(organization[1]))
        for order in orders_list.copy():
            if order[0] == user_id:
                orders_list.pop(orders_list.index(order))
                if order[3] == "pending":
                    updateData(order[1], "orders_get_count",
                               int(organization_getCollValue(order[1], "orders_get_count")) - 1)
                    sheet.del_worksheet(sheet.worksheet(order[2]))
                    os.remove(order_getPathByID(order[2]))
    saveALL()

if __name__ == '__main__':
    bot.polling()
```

```
import matplotlib.pyplot as plt

# Дані
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
# Створення графіка
plt.plot(x, y)
# Додавання підписів
plt.xlabel('Ось X')
plt.ylabel('Ось Y')
plt.title('Лінійний графік')

# Відображення графіка
plt.show()
```