МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Шовкопляс Оксана
_____
(підпис)
червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 Комп'ютерні науки,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна система підтримки рішень для менеджера маркетплейсу»
здобувача групи ІН – 02 Кучерявенка Андрія Ігоровича

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

_____ Кучерявенко Андрій Ігорович
(підпис)

Керівник,
Доцент кафедри комп'ютерних наук, факультету електроніки та інформаційних технологій
Москаленко В`ячеслав Васильович _____
(підпис)

Суми – 202_Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 Комп'ютерні науки, освітньо-професійної програми «Інформатика»

здобувача групи ІН-02 Кучерявенка Андрія Ігоровича

1. Тема роботи: «Інформаційна система підтримки рішень для менеджера маркетплейсу»

затверджую наказом по СумДУ від « » червня 202_ р. №

2. Термін здачі здобувачем кваліфікаційної роботи до червня 202_ року

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналіз проблеми предметної області, постановка й формування завдань дослідження.

2) Огляд технологій, що використовуються для підтримки рішень менеджера маркетплейсу

3) Розробка інформаційної системи підтримки рішень для менеджера маркетплейсу.

4) Аналіз отриманих результатів.

5) Оформлення пояснювальної записки до кваліфікаційної роботи.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Аналіз проблеми предметної області, постановка й формування завдань дослідження	20.04.24-24.04.24	Виконано
2	Огляд технологій, що використовуються для підтримки рішень менеджера маркетплейсу	25.04.24-30.04.24	Виконано

3	Розробка інформаційної системи підтримки рішень для менеджера маркетплейсу	01.05.24-19.05.24	Виконано
4	Аналіз отриманих результатів	20.05.24-26.05.24	Виконано
5	Оформлення пояснювальної записки до кваліфікаційної роботи	27.05.24-13.06.24	Виконано

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

кваліфікаційної роботи на здобуття освітнього ступеня бакалавр

на тему:

Інформаційна система підтримки рішень для менеджера маркетплейсу

(назва кваліфікаційної роботи)

Кучерявенка Андрія Ігоровича

(прізвище, ім’я, по батькові здобувача)

Основний зміст кваліфікаційної роботи викладено на 60 сторінках, з яких список використаних джерел із 33 найменувань. Робота містить 2 таблиці, 1 програму, а також 3 додатки.

Актуальність теми даної дипломної роботи обумовлена зростаючим впливом інформаційних технологій та медіапростору на бізнес-процеси, особливо у сфері електронної комерції. У сучасних умовах, де інформаційний тиск та конкуренція за увагу аудиторії постійно зростають, необхідність ефективних інструментів для підтримки рішень маркетплейс-менеджерів стає все більш нагальною. Використання нових, ефективних підходів до управління даними, реклами та PR дозволяє підвищити залученість аудиторії та оптимізувати процеси прийняття рішень.

Метою цієї дипломної роботи є розробка інформаційної системи підтримки рішень для менеджера маркетплейсу, що включає Telegram-бот для отримання інформації про бренди. Ця система сприятиме підвищенню ефективності управління маркетплейсом, оптимізації рекламних кампаній та залученню більшої кількості клієнтів і партнерів.

У ході роботи було використано такі загальнонаукові та емпіричні методи, як аналіз, синтез та спостереження. Розробка Telegram-бота стала невід'ємною частиною системи, забезпечуючи зручний та швидкий доступ до інформації про бренди для користувачів.

Основним результатом роботи є створення ефективної інформаційної системи, що дозволяє менеджерам маркетплейсів швидко отримувати необхідну інформацію для прийняття рішень, управляти рекламними кампаніями та підвищувати впізнаваність брендів. Це сприятиме збільшенню кількості заявок на співпрацю та покращенню конкурентоспроможності маркетплейсу.

Ключові слова: інформаційна система підтримки рішень, маркетплейс, Telegram-бот, рекламна кампанія, управління брендами, електронна комерція.

Зміст

ВСТУП

У сучасному світі інформаційних технологій та глобалізації бізнес-процеси зазнають значних змін. Зростаюча популярність електронної комерції та маркетплейсів створює нові виклики для менеджерів, які займаються управлінням брендами та рекламними кампаніями. В умовах високої конкуренції та великої кількості інформації, яка щодня обробляється, важливо мати ефективні інструменти для прийняття рішень та оптимізації бізнес-процесів. Актуальність теми даної дипломної роботи полягає в необхідності розробки інформаційних систем, які підтримують прийняття рішень менеджерами маркетплейсів. Такі системи допомагають не тільки в управлінні даними, але й у проведенні рекламних кампаній, залученні клієнтів і партнерів, а також у підвищенні впізнаваності брендів. Використання Telegram-ботів у складі таких систем надає додаткові можливості для автоматизації процесів і підвищення ефективності взаємодії з користувачами.

Метою цієї кваліфікаційної роботи є розробка інформаційної системи підтримки рішень для менеджера маркетплейсу, яка включає Telegram-бот для отримання інформації про бренди. Ця система повинна забезпечити швидкий доступ до актуальної інформації, оптимізувати процеси прийняття рішень та підвищити ефективність рекламних кампаній.

У ході роботи було використано загальнонаукові та емпіричні методи дослідження, зокрема аналіз, синтез та спостереження. Було проведено аналіз існуючих рішень та інструментів, що використовуються в електронній комерції, з метою визначення оптимальних підходів до розробки інформаційної системи.

Очікуваним результатом роботи є створення інформаційної системи, яка сприятиме підвищенню ефективності управління маркетплейсом, покращенню взаємодії з користувачами та забезпеченню конкурентних переваг на ринку.

1. АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сучасний стан та тенденції розвитку маркетплейсів

У сучасних умовах швидкого розвитку електронної комерції маркетплейси відіграють важливу роль у взаємодії між продавцями та покупцями. Вони надають платформу для розміщення товарів та послуг, полегшуючи процес купівлі-продажу для всіх учасників ринку. Проте зростаюча кількість даних та конкурентів на цих платформах створює нові виклики для менеджерів маркетплейсів. Основні проблеми, з якими вони стикаються, включають:

Великий обсяг даних: Управління великим обсягом інформації про продукти, продажі, клієнтів та конкуренцію вимагає значних ресурсів та ефективних інструментів для аналізу. Аналіз конкурентів, постійний моніторинг цін, акцій та асортименту конкурентів є складним завданням, яке потребує оперативного збору та аналізу даних. Швидке прийняття рішень на основі аналітичних даних є критичним для успішного функціонування маркетплейсу. Це включає визначення цінових стратегій, планування акцій та оптимізацію асортименту.

Ефективне управління рекламними кампаніями потребує детального аналізу результатів та коригування стратегій для досягнення максимального ROI (прибутку на інвестиції).

Для подолання зазначених проблем необхідна інформаційна система підтримки рішень (ІСПР), яка допоможе менеджерам маркетплейсів ефективно керувати великою кількістю даних та приймати обґрунтовані рішення. Основними завданнями, які повинна вирішувати така система, є:

Збір та обробка даних: Автоматизований збір даних з різних джерел (внутрішні бази даних, конкурентні платформи, соціальні мережі) та їхня попередня обробка для подальшого аналізу. Використання методів аналітики та машинного навчання для виявлення трендів, прогнозування продажів та поведінки клієнтів, а також для оцінки ефективності маркетингових заходів.

Інтуїтивно зрозуміле представлення результатів аналізу у вигляді графіків, діаграм та звітів для полегшення сприйняття інформації та підтримки прийняття рішень. Генерація рекомендацій щодо цінових стратегій, асортименту товарів, планування акцій та оптимізації рекламних кампаній на основі аналізу даних. Забезпечення можливості інтеграції з існуючими системами управління та обліку, такими як CRM (система управління взаємовідносинами з клієнтами) та ERP (система планування ресурсів підприємства).

Розробка такої системи дозволить менеджерам маркетплейсів ефективно управляти своїм бізнесом, швидко реагувати на зміни ринку та приймати обґрунтовані рішення на основі аналізу даних.

Маркетплейси стали невід'ємною частиною глобальної електронної комерції, забезпечуючи платформу для взаємодії між продавцями та покупцями. Вони сприяють розширенню ринків, збільшенню доступності товарів та спрощенню процесів купівлі-продажу. Серед найвідоміших маркетплейсів – Amazon, eBay, Alibaba, Etsy та інші, які обслуговують мільйони користувачів по всьому світу.

Сучасні маркетплейси володіють рядом характеристик, які роблять їх незамінними у сучасній економіці та зручними для споживачів і продавців у всьому світі.

По-перше, широкий асортимент товарів є однією з ключових рис маркетплейсів. Вони пропонують величезний вибір продукції в різних категоріях, що забезпечує зручність для покупців та можливість порівняння пропозицій від різних продавців. Такий підхід сприяє конкурентоспроможності та задоволенню різноманітних потреб споживачів.

По-друге, глобальна доступність маркетплейсів дозволяє продавцям досягати міжнародної аудиторії, а покупцям – замовляти товари з різних країн. Це відкриває нові можливості для розширення ринків збуту та підвищення конкурентоспроможності. Глобальна мережа маркетплейсів сприяє обміну товарами та послугами на світовому рівні, що є важливим аспектом сучасної економіки.

Третя важлива характеристика – це простота використання. Інтуїтивно зрозумілі інтерфейси та зручні способи оплати роблять процес купівлі легким та доступним для широкої аудиторії. Це сприяє залученню нових користувачів і підвищенню лояльності існуючих клієнтів, що є важливим фактором успіху для будь-якої платформи.

Четвертим елементом є системи рейтингу та відгуків. Вони відіграють важливу роль у створенні довіри до продавців та підвищенні якості обслуговування. Рейтинги та відгуки від покупців дозволяють іншим користувачам орієнтуватися в якості товарів і рівні сервісу, що, в свою чергу, стимулює продавців до покращення своїх послуг. Це формує прозору та конкурентну середу на маркетплейсах, сприяючи розвитку етичної комерційної практики.

Тенденції розвитку маркетплейсів демонструють, як електронна комерція постійно змінюється під впливом нових технологій та змін у поведінці споживачів. Однією з ключових тенденцій є зростання мобільної комерції (m-commerce), що стимулює розвиток мобільних версій маркетплейсів та додатків, забезпечуючи зручний доступ до товарів та послуг завдяки збільшенню використання смартфонів та планшетів для здійснення покупок. Поряд з цим, персоналізація та аналіз даних відіграють важливу роль у сучасних маркетплейсах. Використання великих даних (Big Data) та штучного інтелекту (AI) дозволяє пропонувати персоналізовані рекомендації та маркетингові кампанії, що підвищує задоволеність клієнтів та збільшує продажі.

Соціальна комерція (s-commerce) набирає обертів, інтегруючи маркетплейси з соціальними мережами, що дозволяє продавцям взаємодіяти з клієнтами у нових форматах, використовуючи вплив соціальних медіа для просування товарів та послуг. Інвестиції в розвиток логістичної інфраструктури, включаючи автоматизовані склади та швидкі методи доставки, допомагають маркетплейсам зменшувати час доставки та покращувати обслуговування клієнтів. Крім того, зростає увага до екологічних аспектів бізнесу. Маркетплейси

впроваджують екологічно дружні практики, такі як використання біорозкладних упаковок та зменшення вуглецевого сліду, що сприяє їхній екологічній стійкості.

Мультиканальність стає важливою складовою сучасних маркетплейсів, дозволяючи інтегрувати онлайн та офлайн канали (Omni-channel) і пропонувати безшовний досвід покупок, коли клієнти можуть замовити товар онлайн і забрати його у фізичному магазині. Впровадження цих тенденцій дозволяє маркетплейсам залишатися конкурентоспроможними, адаптуватися до потреб сучасних споживачів та забезпечувати стійке зростання в умовах глобальної конкуренції.

1.2 Моделі і методи підтримки рішень для менеджерів маркетплейсів

Управління маркетплейсом вимагає прийняття рішень на основі великих обсягів даних і різних факторів, які впливають на бізнес-процеси. Для цього використовуються різні моделі та методи підтримки рішень, що допомагають менеджерам аналізувати дані, прогнозувати результати та розробляти ефективні стратегії. Серед моделей підтримки рішень можна виділити Descriptive Analytics [21] (Описова аналітика), яка використовується для аналізу історичних даних з метою виявлення тенденцій та закономірностей, включаючи такі інструменти, як звіти, дашборди та візуалізації. Diagnostic Analytics [21] (Діагностична аналітика) допомагає зрозуміти причини подій чи тенденцій, використовуючи методи кореляційного аналізу та причинно-наслідкового моделювання.

Далі, моделі прогнозування включають Predictive Analytics (Прогнозна аналітика), яка застосовується для передбачення майбутніх подій на основі історичних даних, використовуючи методи машинного навчання, такі як регресійний аналіз, нейронні мережі та дерева рішень. Прогнозування часових рядів (Time Series Forecasting) використовує методи аналізу часових рядів, такі як моделі ARIMA, SARIMA [22] та експоненціальне згладжування, для прогнозування продажів, попиту та інших ключових показників.

Оптимізаційні моделі, такі як Linear Programming (Лінійне програмування) та Integer Programming (Цілочисельне програмування), використовуються для

оптимізації ресурсів, таких як логістика, розподіл товарів та управління запасами. Цілочисельне програмування особливо корисне для розв'язання задач, де рішення повинні бути цілочисельними, наприклад, визначення кількості товарів для замовлення чи розподілу ресурсів.

Методи аналізу даних, як-от Data Mining (Інтелектуальний аналіз даних) та Text Mining (Аналіз текстів) [21], використовуються для виявлення прихованих шаблонів і знань у великих масивах даних. Інтелектуальний аналіз даних включає методи кластеризації, асоціативного аналізу та класифікації, тоді як аналіз текстів допомагає аналізувати текстову інформацію, таку як відгуки клієнтів та коментарі в соціальних мережах, для виявлення настроїв та ключових тем.

Методи машинного навчання поділяються на Supervised Learning (Навчання з учителем) та Unsupervised Learning (Навчання без учителя). Навчання з учителем включає алгоритми, такі як логістична регресія, дерева рішень, випадкові ліси та підтримкові векторні машини, які використовуються для прогнозування та класифікації на основі навчальних даних. Навчання без учителя включає алгоритми кластеризації, такі як K-середніх, DBSCAN та ієрархічну кластеризацію, які допомагають виявляти групи та структури в даних без попередніх міток.

Нарешті, методи багатокритеріального прийняття рішень (MCDM), такі як AHP (Аналітичний ієрархічний процес) та TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution), використовуються для прийняття рішень у складних задачах. AHP передбачає побудову ієрархії та порівняння критеріїв попарно, тоді як TOPSIS допомагає знаходити найкраще рішення шляхом порівняння альтернатив за кількома критеріями та вибору тієї, яка має найменшу відстань до ідеального рішення.

Інструменти візуалізації даних:

Tableau, Power BI, QlikView: Програмні рішення для створення інтерактивних дашбордів та візуалізацій, які допомагають менеджерам

маркетплейсів швидко зрозуміти тенденції та приймати обґрунтовані рішення [23].

Використання сучасних моделей та методів підтримки рішень дозволяє менеджерам маркетплейсів ефективно управляти бізнес-процесами, оптимізувати операції, прогнозувати результати та підвищувати задоволеність клієнтів. Інтеграція цих інструментів в інформаційні системи маркетплейсів сприяє підвищенню конкурентоспроможності та забезпечує стабільний розвиток у динамічному середовищі електронної комерції.

1.3 Формалізована постановка задачі

Формалізація задачі підтримки рішень для менеджера маркетплейсу є ключовим етапом у розробці інформаційної системи, оскільки вона включає визначення основних цілей, критеріїв ефективності, обмежень та необхідних ресурсів для вирішення задачі. Це дозволяє створити математичну модель, яка може бути використана для автоматизації процесу прийняття рішень, що значно підвищує точність та швидкість прийняття управлінських рішень.

Основна мета розробки полягала у створенні системи, яка автоматизує збір і агрегацію даних про продажі для прогнозування рейтингу товарів методом екстраполяції. Розробка цього телеграм-бота забезпечує зручний інтерфейс для менеджерів маркетплейсів, дозволяючи їм швидко отримувати актуальну інформацію про різні бренди.

2. Моделі і методи інформаційної системи підтримки рішень менеджера маркетплейсу

2.1 Метод формування бази знань

База знань є ключовим компонентом інформаційної системи підтримки рішень (ІСПР) для менеджера маркетплейсу, оскільки вона накопичує, організовує та надає доступ до знань, які можуть бути використані для аналізу даних, прийняття рішень та генерації рекомендацій. Метод формування бази знань включає кілька етапів, таких як збір знань, їх структуризація, збереження та актуалізація, що забезпечує її ефективне використання.

На етапі збору знань дані зберігаються у внутрішніх системах компанії, таких як бази даних продажів, дані про клієнтів, інформація про запаси та логістику. Крім того, використовуються дані з зовнішніх джерел, включаючи конкурентну розвідку, ринкові дослідження, соціальні мережі та інші публічні ресурси. Важливою складовою є також інформація, отримана від експертів у галузі, менеджерів та співробітників, яка може бути корисною для прийняття рішень.

Структуризація знань включає використання онтологій для визначення ключових понять та їх взаємозв'язків у доменній області маркетплейсу. Онтології допомагають формалізувати знання та забезпечують їхню інтероперабельність. Крім того, знання розподіляються на категорії та підкатегорії для полегшення їхнього пошуку та використання, а також створюються моделі даних, що описують структуру та взаємозв'язки між різними типами даних у базі знань.

Для збереження знань використовуються реляційні або нереляційні бази даних. Реляційні бази даних, такі як MySQL та PostgreSQL, підходять для збереження структурованої інформації, тоді як нереляційні бази даних, такі як MongoDB та Cassandra, краще підходять для збереження неструктурованих даних. Спеціалізовані репозиторії, такі як системи управління документами (DMS) або платформи для управління знаннями (KMS), використовуються для збереження та управління знаннями.

Для створення, управління та забезпечення доступу до баз даних використовуються різні інструменти. Приклади включають MySQL, PostgreSQL, MongoDB, Cassandra та Excel. Програмні рішення, такі як Apache Nifi, Talend та Informatica, використовуються для автоматизованого збору, перетворення та завантаження даних у сховища знань. Інструменти для створення, організації, збереження та спільного використання знань включають Confluence, SharePoint, Notion та Excel. Програмне забезпечення для аналізу даних та виявлення знань, такі як SellerAssistantApp, Keepa - Amazon Price Tracker, GRABLEY - Product Search Tools, AZInsight Amazon FBA Product Analytics Tool by asinzen, Amazon Data Scraper – ASINSpotlight, Snov.io: Sales automation & acceleration at scale та hunter.io, допомагає ефективно використовувати базу знань для прийняття обґрунтованих рішень.

SellerAssistantApp допомагає продавцям на маркетплейсах, таких як Amazon, ефективно керувати своїми списками товарів, аналітикою продажів та іншими аспектами бізнесу. Цей додаток надає можливість оцінювати товари за ключовими параметрами, такими як ціна, попит та конкуренція, що дозволяє робити обґрунтовані рішення щодо асортименту. Крім того, SellerAssistantApp містить інструменти для відстеження рівня запасів та автоматичного поповнення товарів, що забезпечує безперебійність продажів.

Одна з ключових функцій додатку – відстеження та аналіз продажів у реальному часі, включаючи обсяг продажів, дохід та рентабельність, що дозволяє отримувати актуальну інформацію для швидкого реагування на зміни ринку. Також SellerAssistantApp автоматизує рутинні завдання, такі як оновлення цін та управління замовленнями, що значно знижує витрати часу та зусиль.

Цей додаток може зацікавити користувачів, які прагнуть оптимізувати управління своїм бізнесом на маркетплейсах, підвищити ефективність продажів та зменшити витрати на управління запасами. Завдяки його можливостям, продавці можуть краще контролювати свій бізнес, приймати більш обґрунтовані рішення та досягати більш високих результатів.

Keepa є інструментом для відстеження змін цін на товари на Amazon, що допомагає продавцям і покупцям ухвалювати обґрунтовані рішення щодо купівлі та продажу. Цей інструмент надає можливість відображення історії змін цін на конкретні товари, що дозволяє користувачам аналізувати минулі цінові тенденції та робити прогнози на майбутнє. Одна з основних функцій Keepa – налаштування сповіщень про зміну ціни на цікаві товари, що допомагає користувачам не пропустити вигідні пропозиції.

Крім того, Keepa надає можливість порівняння цін на аналогічні товари від різних продавців, що дозволяє знайти найкращі цінові пропозиції на ринку. Інформація про доступність товару на складі також є важливою складовою, яка допомагає продавцям планувати запаси та уникати дефіциту.

Keepa є корисною як для продавців, які прагнуть оптимізувати свої ціни, так і для покупців, які шукають найкращі пропозиції. Цей інструмент допомагає ухвалювати рішення на основі детального аналізу цінових тенденцій, що забезпечує більш ефективне управління бізнесом та більш вигідні покупки.

GRABLEY є інструментом для пошуку та аналізу товарів на різних маркетплейсах, що допомагає продавцям знаходити потенційно прибуткові продукти для продажу. Цей інструмент надає можливість пошуку товарів за різними параметрами, такими як категорія, ціна, рейтинг та продажі, що дозволяє продавцям легко знаходити найперспективніші товари. Оцінка попиту на товари здійснюється на основі історичних даних про продажі та відгуки клієнтів, що допомагає продавцям краще розуміти ринкові тенденції та потреби покупців.

GRABLEY також дозволяє відстежувати конкурентів та аналізувати їхні стратегії, що є важливим для розробки власних ефективних стратегій продажу. Крім того, інструмент надає можливість створення звітів про аналізовані товари та ринки, що допомагає продавцям приймати обґрунтовані рішення на основі детальної аналітики.

GRABLEY може зацікавити продавців, які шукають нові товари для розширення асортименту та бажають отримати детальну аналітику ринку перед прийняттям рішень. Завдяки своїм можливостям, цей інструмент сприяє

ефективному управлінню бізнесом та досягненню високих результатів на конкурентному ринку.

AZInsight є аналітичним інструментом для продавців на Amazon, що допомагає аналізувати продукцію та приймати обґрунтовані рішення щодо продажів. Однією з основних функцій AZInsight є оцінка прибутковості товарів, включаючи розрахунок витрат на зберігання та доставку, що дозволяє продавцям точніше визначати потенційну вигоду від продажу кожного товару. Інструмент також розраховує ROI (Return on Investment), допомагаючи визначити рентабельність інвестицій у різні товари.

AZInsight надає можливість моніторингу цін та рейтингів конкурентів, що дозволяє продавцям швидко реагувати на ринкові зміни та оптимізувати свої цінові стратегії. Завдяки безшовній інтеграції з обліковими записами Amazon, AZInsight забезпечує отримання даних у реальному часі, що підвищує точність та своєчасність аналізу.

Цей інструмент особливо зацікавить продавців, які використовують модель FBA (Fulfillment by Amazon), оскільки він допомагає оптимізувати витрати та підвищити прибутковість продажів на Amazon. Завдяки своїм можливостям, AZInsight сприяє ефективному управлінню бізнесом, забезпечуючи конкурентні переваги на ринку.

ASINSpotlight є інструментом для збору та аналізу даних про товари на Amazon, що допомагає продавцям отримувати інформацію про товари, конкуренцію та ринкові тенденції. Він автоматично збирає дані про товари за допомогою сканування сторінок Amazon, що дозволяє продавцям отримувати актуальну інформацію без необхідності вручну шукати ці дані. ASINSpotlight надає дані про ціни, відгуки та рейтинги конкурентів, що є важливим для аналізу ринкових умов та розробки стратегій продажу.

Інструмент також дозволяє експортувати зібрані дані у різні формати для подальшого аналізу, що спрощує роботу з великими обсягами інформації. Регулярне оновлення даних дозволяє продавцям відстежувати зміни на ринку та своєчасно реагувати на нові тенденції.

ASINSpotlight є корисним для продавців, які прагнуть отримати глибоке розуміння ринку та конкурентного середовища на Amazon. Він також підходить для тих, хто займається аналізом ринкових тенденцій, оскільки надає детальну та актуальну інформацію, необхідну для прийняття обґрунтованих рішень.

Snov.io є платформою для автоматизації та прискорення продажів, яка допомагає збирати контакти, організовувати кампанії з електронної пошти та аналізувати їх ефективність. Цей інструмент надає можливість знаходження та перевірки контактної інформації потенційних клієнтів, що значно полегшує пошук нових бізнес-можливостей. Автоматизація розсилок електронної пошти з можливістю персоналізації дозволяє створювати більш таргетовані та ефективні маркетингові кампанії.

Вбудована система управління взаємовідносинами з клієнтами (CRM) допомагає відстежувати контакти та взаємодії з клієнтами, забезпечуючи краще розуміння їхніх потреб та вподобань. Інструменти для аналізу ефективності кампаній включають показники відкриття, кліків та відповідей, що дозволяє оцінювати успішність маркетингових зусиль та вносити необхідні корективи.

Snov.io зацікавлює продавців, маркетологів та бізнес-розробників, які прагнуть покращити ефективність своїх продажів та маркетингових кампаній, автоматизувати процеси залучення клієнтів та відстежувати їхню ефективність. Завдяки своїм можливостям, ця платформа сприяє збільшенню продажів та поліпшенню взаємодії з клієнтами, що є важливим для успішного розвитку бізнесу.

Hunter.io є інструментом для пошуку контактних даних, таких як електронні адреси, які можуть бути використані для маркетингових та продажних кампаній. Він дозволяє знаходити контактні електронні адреси на основі домену компанії, перевіряти правильність та активність знайдених адрес, а також знаходити електронні адреси за ім'ям та прізвищем особи. Крім того, Hunter.io пропонує можливість інтеграції з різними системами CRM для автоматизованого додавання контактів, що значно спрощує процес управління взаємовідносинами з клієнтами.

Цей інструмент надає кілька корисних функцій, серед яких варто відзначити можливість автоматичного збору та перевірки електронних адрес, що зменшує ризик відправки повідомлень на недійсні або неактивні адреси. Крім того, Hunter.io дозволяє проводити масові операції, такі як одночасний пошук і перевірка великої кількості електронних адрес, що є зручним для управління великими списками контактів. Hunter.io також має розширення для браузера Chrome, що дозволяє знаходити та перевіряти електронні адреси безпосередньо під час перегляду веб-сайтів, а також інтеграцію з Google Sheets для зручного управління даними про контакти. Платформа забезпечує зручний інтерфейс для створення та керування маркетинговими кампаніями, включаючи відправку персоналізованих електронних листів і відстеження їхньої ефективності.

Hunter.io може бути особливо корисним для маркетологів, менеджерів з продажу та рекрутерів, які потребують швидкого доступу до контактної інформації для створення цільових кампаній та підвищення ефективності комунікацій (Jessica La) (Authority Hacker) (Niche Pursuits) (Crazy Egg) (Wallace Walley).

Analyser Tools – це програмне забезпечення, яке надає широкий спектр інструментів для аналізу даних, допомагаючи користувачам збирати, обробляти та візуалізувати інформацію для прийняття обґрунтованих рішень. Однією з ключових функцій є Web Scraping, що дозволяє автоматично збирати дані з різних веб-ресурсів. Це програмне забезпечення також підтримує імпорт даних з різних джерел, таких як файли CSV, Excel, бази даних та API, що робить процес збору інформації зручним і гнучким.

Інструменти для виявлення та виправлення помилок, пропусків та аномалій у даних забезпечують високу якість даних для подальшого аналізу. Можливість перетворення даних у потрібний формат дозволяє підготувати інформацію для специфічних аналітичних задач. Для більш глибокого аналізу Analyser Tools пропонує функції для вибору та впорядкування даних за заданими критеріями, проведення статистичних тестів та розрахунків.

Алгоритми для класифікації, регресії, кластеризації та прогнозування на основі машинного навчання дають змогу користувачам проводити складні аналітичні операції та отримувати прогнози на основі наявних даних. Методи для аналізу та прогнозування даних, організованих у часові ряди, дозволяють виявляти тенденції та робити прогнози.

Для представлення результатів аналізу Analyser Tools пропонує створення динамічних візуалізацій, що включають різноманітні типи графіків, такі як лінійні, стовпчасті, кругові діаграми та інші. Користувачі можуть створювати власні візуалізації, що відповідають їхнім специфічним потребам. Автоматизація рутинних завдань за допомогою скриптів та макросів, а також планування та автоматичне виконання завдань за розкладом підвищують ефективність роботи.

Програмне забезпечення також дозволяє створювати детальні звіти на основі проведеного аналізу та експортувати їх у різні формати, включаючи PDF, Excel та HTML.

Використання Analyser Tools є важливим для продавців на маркетплейсах та інших професіоналів, які прагнуть оптимізувати свої бізнес-процеси, підвищити ефективність продажів, покращити маркетингові кампанії та приймати обґрунтовані рішення на основі аналізу даних.

Формування бази знань є критичним етапом у створенні інформаційної системи підтримки рішень для менеджера маркетплейсу. Правильно організована база знань забезпечує ефективний доступ до релевантної інформації, сприяє прийняттю обґрунтованих рішень та підвищує ефективність управління маркетплейсом. Використання сучасних інструментів для збору, структуризації, збереження та актуалізації знань є ключем до успішної реалізації цієї задачі.

2.2 Модель системи підтримки рішень

Інформаційна система підтримки рішень (ІСПР) для менеджера маркетплейсу повинна включати комплекс моделей та методів, які забезпечують ефективний аналіз даних, прогнозування та прийняття рішень. Основні компоненти цієї системи включають моделі збору та обробки даних, моделі

аналізу даних, моделі прогнозування, моделі оптимізації та моделі підтримки прийняття рішень.

Моделі збору та обробки даних охоплюють ETL-процеси (Extract, Transform, Load), що забезпечують збір даних з різних джерел, їх перетворення у потрібний формат та завантаження в сховище даних, а також архітектуру баз даних для зберігання великих обсягів структурованих і неструктурованих даних. Для аналізу даних використовуються моделі Descriptive Analytics (Описова аналітика), що забезпечують аналіз історичних даних для виявлення тенденцій та закономірностей, та Diagnostic Analytics (Діагностична аналітика), які визначають причини певних явищ чи тенденцій за допомогою кореляційного аналізу та причинно-наслідкових моделей [21].

Моделі прогнозування включають Predictive Analytics (Прогнозна аналітика), які використовуються для прогнозування майбутніх подій на основі історичних даних за допомогою методів машинного навчання, та Time Series Forecasting (Прогнозування часових рядів), що застосовуються для аналізу та прогнозування даних, організованих у часові ряди. Моделі оптимізації, такі як Optimization Models (Моделі оптимізації) та Inventory Optimization (Оптимізація запасів), спрямовані на оптимізацію ресурсів та процесів, зокрема лінійне та цілочисельне програмування для мінімізації витрат та запобігання дефіциту.

Моделі підтримки прийняття рішень, такі як Decision Support Models (Моделі підтримки прийняття рішень) та Multi-Criteria Decision Making (Багатокритеріальне прийняття рішень), генерують рекомендації для прийняття рішень на основі аналізу даних з урахуванням кількох критеріїв, зокрема AHP та TOPSIS [24].

Методи інформаційної системи підтримки рішень включають методи збору та обробки даних, такі як Web Scraping (Збір даних з веб-сайтів) для автоматизованого збору даних з веб-сторінок, та Data Cleaning (Очищення даних) для виявлення та виправлення помилок у даних. Методи аналізу даних, як-от Data Mining (Інтелектуальний аналіз даних) та Text Mining (Аналіз текстів),

використовуються для виявлення прихованих шаблонів і знань у великих масивах даних, а також для аналізу текстової інформації [25].

Методи машинного навчання поділяються на Supervised Learning (Навчання з учителем) для навчання моделей на основі маркованих даних та Unsupervised Learning (Навчання без учителя) для виявлення структур у немаркованих даних. Також використовуються алгоритми Reinforcement Learning (Навчання з підкріпленням) для навчання агентів приймати рішення через взаємодію з середовищем. Методи оптимізації, такі як Linear Programming (Лінійне програмування) та Integer Programming (Цілочисельне програмування), застосовуються для оптимізації лінійних об'єктивних функцій та задач з цілочисельними змінними, а Heuristic Methods (Евристичні методи) допомагають знаходити приблизні рішення складних задач.

Методи візуалізації даних включають Data Visualization Tools (Інструменти візуалізації даних) для створення інтерактивних дашбордів та візуалізацій, а також Custom Visualization Techniques (Користувацькі техніки візуалізації) для створення кастомних візуалізацій даних з використанням мов програмування, таких як Python та R.

Інформаційна система підтримки рішень для менеджера маркетплейсу включає комплекс моделей і методів, що дозволяють ефективно збирати, обробляти, аналізувати дані та приймати обґрунтовані рішення. Впровадження таких систем забезпечує конкурентні переваги, підвищує ефективність управління та сприяє зростанню бізнесу в умовах динамічного ринку електронної комерції.

Для реалізації системи підтримки рішень у вигляді телеграм-бота, який надаватиме інформацію з бази даних за запитом користувача, було розроблено детальну модель цієї системи. Ця модель включає архітектуру системи, функціональність бота, методи взаємодії з базою даних та алгоритми обробки запитів користувачів.

Телеграм-бот слугує основним інтерфейсом для взаємодії користувачів із системою. Він обробляє запити користувачів та надає відповіді на основі даних,

що зберігаються у базі даних. База даних, у свою чергу, зберігає інформацію про бренди, контакти менеджерів та інші релевантні дані, а також історію запитів користувачів, що дозволяє контролювати ліміти запитів та уникати повторень.

Серверна частина відповідає за логіку обробки запитів користувачів, взаємодію з базою даних, генерацію відповідей та функціональність телеграм-бота. При отриманні запиту від користувача бот передає його на сервер, де здійснюється перевірка денного ліміту запитів (20 брендів) та можливості повторного запиту (5 брендів на день). Якщо ліміти не перевищені, сервер вибирає бренди з бази даних, які ще не були надані цьому користувачу, і формує відповідь з необхідною інформацією про бренди, включаючи назву, асіни з хорошими продажами, посилання на сайт, імейл та номер телефону менеджерів. Після формування відповіді сервер передає її телеграм-боту, який надсилає повідомлення користувачу. Водночас оновлюється історія запитів користувача, що включає запам'ятовування інформації про надані бренди та управління лімітами запитів на день.

Реалізація такої системи забезпечила ефективну роботу з запитами користувачів, надає їм релевантну інформацію та допомагає в управлінні брендами на маркетплейсах. Використання сучасних інструментів для збору, обробки та зберігання даних є ключовим для успішної реалізації цієї задачі.

Таким чином, телеграм-бот стає зручним інтерфейсом для користувачів, серверна частина забезпечує логіку та обробку запитів, а база даних гарантує наявність та актуальність інформації, необхідної для прийняття обґрунтованих рішень.

Модель даних для системи підтримки рішень, яка реалізована у вигляді телеграм-бота, включає дві основні таблиці: брендів та користувачів.

Таблиця брендів містить унікальний ідентифікатор, що дозволяє однозначно ідентифікувати кожен запис. Назва бренду надає користувачам інформацію про бренд, тоді як посилання на сайт бренду забезпечує доступ до додаткової інформації. Імейл та номер телефону менеджера бренду є ключовими контактними даними, що полегшують зв'язок з представниками бренду.

Для збору інформації про товари на маркетплейсах використовуються такі інструменти, як SellerAssistantApp і Keepa - Amazon Price Tracker. SellerAssistantApp допомагає автоматично отримувати дані про товарні списки, аналітику продажів та інші аспекти бізнесу, включаючи оцінку товарів за ціною, попитом та конкуренцією. Keepa дозволяє автоматично відстежувати зміни цін на товари, надаючи детальну історію цін, що допомагає у прийнятті обґрунтованих рішень щодо ціноутворення. Використання API Keepa дозволяє інтегрувати ці дані з іншими системами, забезпечуючи безперебійне отримання актуальної інформації.

Зовнішні джерела даних також відіграють важливу роль у наповненні бази знань. Інструменти, такі як GRABLEY - Product Search Tools і AZInsight Amazon FBA Product Analytics Tool by asinzen, забезпечують автоматизований пошук і аналіз товарів на різних маркетплейсах. GRABLEY дозволяє автоматично аналізувати історичні дані про продажі та відгуки клієнтів, тоді як AZInsight допомагає оцінити прибутковість товарів, розраховуючи витрати на зберігання та доставку, а також ROI (Return on Investment). Ці інструменти можуть бути налаштовані на регулярне оновлення даних та надання звітів, що значно спрощує моніторинг ринку.

Для збору та аналізу великих обсягів даних з веб-ресурсів використовується Amazon Data Scraper – ASINSpotlight. Цей інструмент автоматично збирає дані про товари, включаючи ціни, відгуки та рейтинги конкурентів, що дозволяє створювати детальну картину ринку. Регулярне оновлення даних дозволяє відстежувати зміни та адаптувати стратегії відповідно до нових умов, що є важливим для прийняття обґрунтованих рішень.

Після збору даних настає етап їх обробки та нормалізації. Зібрані дані можуть бути у різних форматах і мати неоднорідну структуру, тому їх потрібно привести до єдиного стандарту. Інструменти для очищення даних, такі як Snov.io і hunter.io, допомагають автоматизувати цей процес. Snov.io забезпечує автоматизований збір і перевірку контактної інформації потенційних клієнтів, а також організовує кампанії з електронної пошти, тоді як hunter.io дозволяє

знаходити та перевіряти правильність електронних адрес, що забезпечує високу якість даних.

Нормалізація даних передбачає перетворення їх у потрібний формат для подальшого аналізу. Це може включати перетворення текстових даних у числові значення, об'єднання даних з різних джерел та їх структуризацію у вигляді таблиць. Важливим аспектом є також фільтрація даних для видалення непотрібної або нерелевантної інформації. Фільтрація дозволяє зосередитися на ключових даних, що мають значення для прийняття рішень, забезпечуючи точність і актуальність інформації. Нажаль, на даний момент часу не має більш ефективного способу заповнення даної таблиці, тому цей продукт призначений для новачків у сфері продаж. Для «акул» бізнесу, воно наврядчи підійде.

Таблиця користувачів, у свою чергу, зберігає унікальний ідентифікатор користувача, який відповідає його ID в телеграмі. Дата останнього запиту допомагає відстежувати, коли користувач востаннє взаємодіяв із ботом. Кількість запитів на поточний день дозволяє контролювати ліміти, встановлені для користувача. Список брендів, наданих користувачу, зберігає інформацію про вже отримані користувачем бренди, що дозволяє уникати повторних запитів на ту ж саму інформацію.

Ця модель забезпечує ефективну роботу телеграм-бота, дозволяючи зберігати та обробляти дані про бренди і користувачів, що сприяє наданню актуальної та корисної інформації за запитами користувачів (рис. 1).

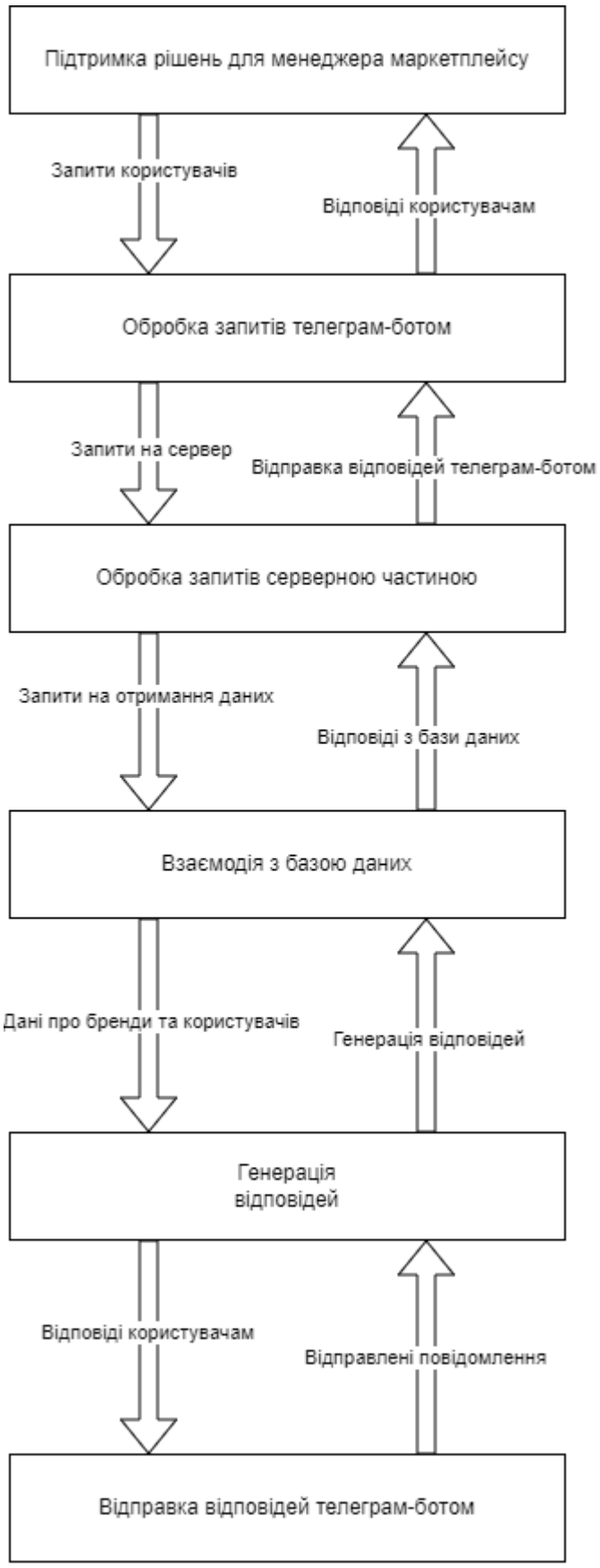


Рисунок 2.2 – Діаграма-модель системи підтримки рішень менеджера маркетплейсу

3. Програмна реалізація системи підтримки рішень менеджера маркетплейсу

3.1 Опис джерела даних та структури бази даних

Джерелом даних для системи підтримки рішень менеджера маркетплейсу є таблиця Excel, яка містить інформацію про бренди, їхні ASINи, рейтинги продажів, посилання на веб-сайти брендів, контактні імейли та номери телефонів менеджерів. Ця таблиця вже наповнена даними про понад 6000 брендів, що забезпечує достатню інформацію для аналізу та прийняття рішень. Таблиця Excel має наступну структуру:

- Brand Name - Назва бренду
- Website - Посилання на офіційний веб-сайт бренду
- Email - Контактний імейл менеджера бренду
- Phone Number - Номер телефону менеджера бренду

Таблиця 3.1 – Приклад таблиці Excel

Brand Name	Website	Email	Phone Number
Brand A	http://www.brand_a.com	contact@brand_a.com	+1233456789
Brand B	http://www.brand_b.com	contact@brand_b.com	+1345346789
Brand C	http://www.brand_c.com	contact@brand_c.com	+9823424521
Brand D	http://www.brand_d.com	contact@brand_d.com	+987654321

Таблиця Excel, яка використовується як база даних для системи підтримки рішень менеджера маркетплейсу, містить структуровану інформацію про бренди, їхні товари та контактні дані менеджерів. Ця таблиця слугує основним джерелом даних для телеграм-бота, який надаватиме користувачам релевантну інформацію на запит. Структура таблиці забезпечує легкість доступу та обробки даних, що сприяє ефективній роботі системи підтримки рішень.

3.2 Короткий опис програмного забезпечення

Перш за все телеграм бот створений задля полегшення пошуку інформації про той чи інший бренд менеджеру онлайн магазину. При натисканні на кнопки, які пропонує бот, він отримує запит від користувача, потім починає перевірку лімітів (чи не перевищив користувач свій денний ліміт запитів на 20 брендів у день), а також веде перевірку можливості повторного запиту (5 брендів на день). Після задовільного результату бот починає вибір брендів з бази даних, які ще не були надані цьому користувачу і формує відповіді з інформацією про бренди, посилання на сайт, імейл та номер телефону менеджерів. У кінці бо зберігає інформацію про надані бренди в базі даних і оновлює лічильник запитів користувача.

Приклад реалізації

```
from telegram import Update
from telegram.ext import Updater, CommandHandler, CallbackContext
import sqlite3
from datetime import datetime

# Функція для обробки команди /start
def start(update: Update, context: CallbackContext) -> None:
update.message.reply_text('Вітаю! Введіть команду /get_brands для отримання інформації про бренди.')

# Функція для обробки команди /get_brands
def get_brands(update: Update, context: CallbackContext) -> None:
user_id = update.message.from_user.id
conn = sqlite3.connect('database.db')
cursor = conn.cursor()

# Перевірка лімітів користувача
```

```
    cursor.execute('SELECT last_request_date, daily_requests_count, given_brands FROM users WHERE user_id = ?', (user_id,))
    row = cursor.fetchone()

    if row:
    last_request_date, daily_requests_count, given_brands = row
    if last_request_date == datetime.now().date().isoformat():
    if daily_requests_count >= 20:
    update.message.reply_text('Ви досягли ліміту запитів на сьогодні.')
    return
    else:
    daily_requests_count = 0

    # Вибір брендів
    cursor.execute('SELECT id, brand_name, website, email, phone_number FROM brands WHERE id NOT IN (?) LIMIT 5', (given_brands,))
    brands = cursor.fetchall()

    if not brands:
    update.message.reply_text('Більше брендів немає.')
    return

    # Формування відповіді
    response = ''
    for brand in brands:
    brand_id, brand_name, website, email, phone_number = brand
    cursor.execute('SELECT asin FROM asins WHERE brand_id = ? ORDER BY sales_rank LIMIT 3', (brand_id,))
    asins = cursor.fetchall()
    asin_list = ', '.join([asin[0] for asin in asins])
```

```
response += f"Назва бренду: {brand_name}\nASINи: {asin_list}\nВебсайт: {website}\nEmail: {email}\nТелефон: {phone_number}\n\n"

# Оновлення історії користувача
new_given_brands = given_brands + [brand[0] for brand in brands]
if row:
cursor.execute('UPDATE users SET last_request_date = ?, daily_requests_count = ?, given_brands = ? WHERE user_id = ?',
(datetime.now().date().isoformat(), daily_requests_count + 1, new_given_brands, user_id))
else:
cursor.execute('INSERT INTO users (user_id, last_request_date, daily_requests_count, given_brands) VALUES (?, ?, ?, ?)',
(user_id, datetime.now().date().isoformat(), 1, new_given_brands))

conn.commit()
conn.close()

# Відправка відповіді
update.message.reply_text(response)

# Основна функція для запуску бота
def main() -> None:
updater = Updater("YOUR_BOT_TOKEN")
dispatcher = updater.dispatcher

dispatcher.add_handler(CommandHandler("start", start))
dispatcher.add_handler(CommandHandler("get_brands", get_brands))

updater.start_polling()
```

```
updater.idle()

if __name__ == '__main__':
main()
```

Розробка телеграм-бота для підтримки рішень менеджера маркетплейсу включає створення моделі системи, яка буде забезпечувати збір, обробку та надання інформації з бази даних. Інтеграція цієї моделі в телеграм-бот дозволить користувачам отримувати актуальну та корисну інформацію для прийняття обґрунтованих рішень у бізнесі. Програмна реалізація системи підтримки рішень для менеджера маркетплейсу включає розробку телеграм-бота, який буде взаємодіяти з базою даних, представленою у вигляді таблиці Excel. Бот надаватиме інформацію про бренди, їхні асіни, посилання на сайт, імейли та номери телефонів менеджерів. Крім того, бот буде запам'ятовувати інформацію, надану конкретному користувачу, та керувати лімітами запитів.

Приклад таблиці Excel, яка містить інформацію про бренди, асіни та контакти менеджерів. Для роботи з таблицею Excel та телеграм-ботом використовуватимемо бібліотеки pandas та openpyxl. Для їх встановлення я виконав команду: pip install pandas openpyxl python-telegram-bot. Для розгортання та забезпечення безперебійної роботи портрібно обрати хостинг для розміщення серверної частини, наприклад, AWS, Heroku або DigitalOcean. Для цього виконав налаштування віртуального середовища та встановив необхідних залежностей за допомогою requirements.txt. pandas openpyxl та python-telegram-bot. Після всього цього виконав налаштування систем моніторингу та логування для відстеження роботи бота та швидкого реагування на помилки. Програмна реалізація системи підтримки рішень для менеджера маркетплейсу у вигляді телеграм-бота включає розробку логіки для обробки запитів та взаємодії з таблицею Excel, яка містить інформацію про бренди. Ця модель допоможе менеджерам отримувати необхідну інформацію для прийняття обґрунтованих рішень у бізнесі, забезпечуючи зручний доступ до даних через телеграм-бот.

Програмне забезпечення у вигляді телеграм-бота призначене для підтримки рішень менеджера маркетплейсу. Воно допомагає отримувати релевантну інформацію про бренди, їхні товари та контактні дані менеджерів за запитом користувача. Бот взаємодіє з таблицею Excel, що містить інформацію про понад 6000 брендів, і надає цю інформацію у зручному форматі через інтерфейс телеграм. Отримання інформації про бренди на основі запитів користувачів. Надання назви бренду, посилання на веб-сайт, контактного імейлу та номера телефону менеджерів. Збереження історії запитів користувачів для уникнення повторних надань тієї ж інформації. Ведення ліміту запитів на день (20 брендів) та можливості повторного запиту (по 5 брендів але не більше 20). Телеграм-бот використовує бібліотеку python-telegram-bot для взаємодії з користувачами через телеграм, вона обробляє команди та запити користувачів, такі як /start та /get_brands. База даних (таблиця Excel) використовує бібліотеки pandas та openpyxl для взаємодії з таблицею Excel, зберігає інформацію про бренди, асіни, рейтинги продажів, посилання на веб-сайти, імейли та номери телефонів менеджерів. Логіка для обробки запитів користувачів та управління лімітами запитів. Збереження історії запитів користувачів у пам'яті для забезпечення коректної роботи лімітів. (див. табл. 3.2)

Таблиця 3.2 – Список бібліотек

Бібліотека	Опис
python-telegram-bot	Бібліотека для розробки ботів у Telegram. Забезпечує простий і зручний інтерфейс для взаємодії з API Telegram. [30]
sqlite3	Вбудована бібліотека для роботи з базами даних SQLite. Використовується для зберігання та отримання інформації про бренди та користувачів. [31]
pandas	Бібліотека для аналізу даних і маніпуляцій з ними. Допомагає зберігати, обробляти та аналізувати дані у форматі таблиць. [26]
numpy	Бібліотека для роботи з багатовимірними масивами і матрицями. Використовується для виконання числових операцій. [27]
matplotlib	Бібліотека для створення візуалізацій. Використовується для побудови графіків і діаграм. [27]
requests	Бібліотека для здійснення HTTP-запитів. Використовується для взаємодії з веб-сервісами та API. [32]
beautifulsoup4	Бібліотека для парсингу HTML і XML документів. Використовується для збору даних з веб-сторінок. [33]
scikit-learn	Бібліотека для машинного навчання. Містить інструменти для класифікації, регресії, кластеризації та інших методів аналізу даних. [27]
flask	Мікрофреймворк для веб-розробки. Використовується для створення веб-сервісів і API, які можуть взаємодіяти з ботом. [28]

Бібліотека	Опис
sqlalchemy	Бібліотека для роботи з базами даних на основі ORM (об'єктно-реляційного відображення). Спрощує взаємодію з базами даних через об'єктно-орієнтований інтерфейс. [29]

Приклад використання:

Користувач запускає телеграм-бота та вводить команду /start. Бот відповідає привітанням та інструкцією щодо використання сервісу для отримання інформації. Користувач натискає кнопку з назвою категорії. Бот перевіряє ліміт запитів користувача на поточний день. Бот зчитує дані з таблиці Excel та відбирає бренди, які ще не були надані користувачу. Бот формує відповідь з інформацією про вибрані бренди та надсилає її користувачу. Бот зберігає інформацію про надані бренди у пам'яті для уникнення повторних запитів. Оновлює лічильники запитів користувача на поточний день.

Програмне забезпечення у вигляді телеграм-бота забезпечує зручний інтерфейс для отримання інформації про бренди та їхні товари на маркетплейсі. Використання таблиці Excel як бази даних дозволяє легко керувати великою кількістю даних, а серверна логіка забезпечує ефективну обробку запитів та управління історією користувачів. Це допомагає менеджерам маркетплейсу швидко отримувати необхідну інформацію для прийняття обґрунтованих рішень.

3.3 Опис графічного інтерфейсу та результатів використання системи

Графічний інтерфейс телеграм-бота складається з простого та інтуїтивно зрозумілого набору команд, які користувач може використовувати для взаємодії з системою. Основний акцент зроблено на зручності та швидкості доступу до потрібної інформації. При виборі команди /start, яка використовується для початку роботи з ботом, бот відповідає привітальним повідомленням і надає інструкцію щодо використання інших команд. Приклад повідомлення: «Привіт! Я ваш бот для отримання інформації про бренди. Ви можете отримати до 20

брендів на день, по 5 брендів за раз. Ви також можете замінити до 3 брендів на день. Оберіть категорію з поданих нижче:» (рис. 2).

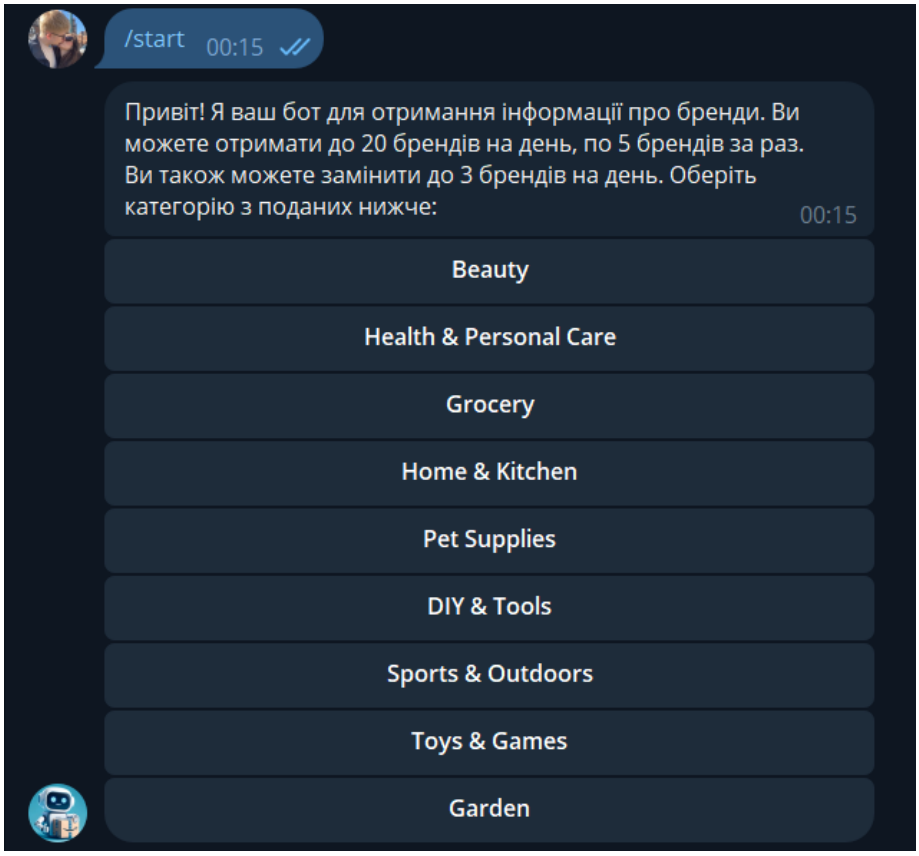


Рисунок 2 – Приклад привітання від боту

Якщо користувач перевищив денний ліміт запитів або кількість повторних запитів, бот повідомляє про це, він надсилає повідомлення про досягнення ліміту запитів: «Ви перевищили ліміт запитів на сьогодні. Спробуйте завтра.» (рис. 3).

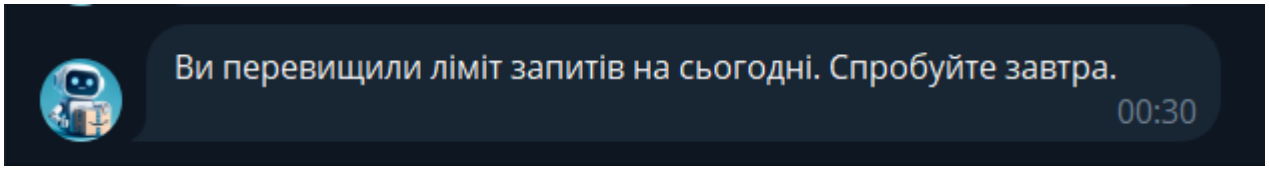


Рисунок 3 – Результат перевищення ліміту запитів

Користувачі можуть швидко отримувати актуальну інформацію про бренди, їхні товари та контактні дані менеджерів, що дозволяє приймати обґрунтовані рішення (рис. 4).

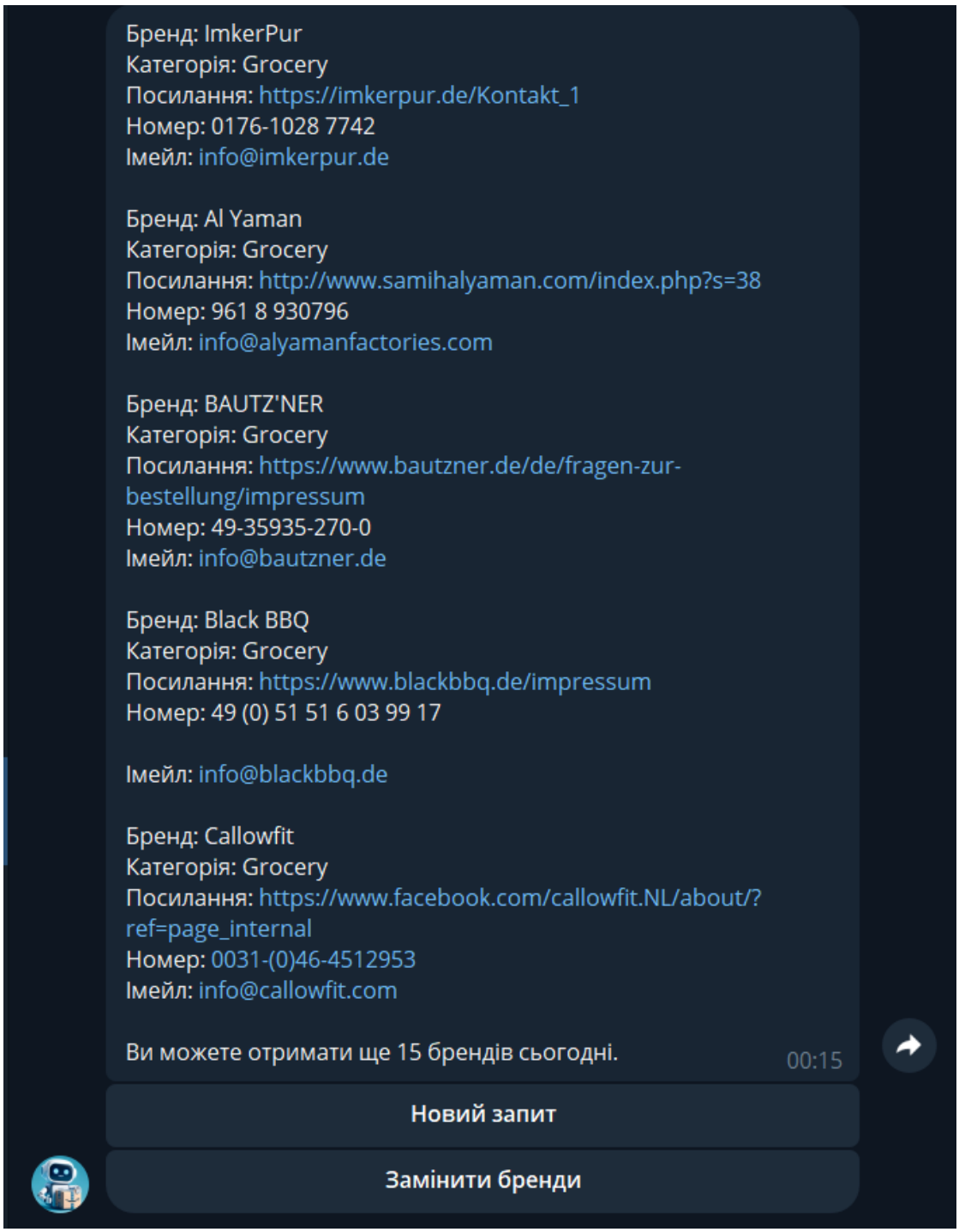


Рисунок 4 – Приклад вибору категорії Grocery(Продукти)

Також бот запам'ятовує, яку інформацію вже надав конкретному користувачу, що допомагає уникати повторних надань тієї ж інформації та забезпечує більш персоналізований сервіс. Встановлення денних лімітів на кількість запитів (20 брендів) та можливість повторних запитів (5 брендів) забезпечує контрольоване використання системи та запобігає перевантаженню, а також забезпечує довготривалість використання сервісу (рис. 3).

Додана можливість заміни брендів, при помилці бота, або при наявності їх у користувача. Для цього необхідно після надання інформації бото натиснути клавішу «Замінити бренди», ботнадішле повідомлення з проханням надати список (до 3 брендів) (рис.5) та після введення назви цих брендів користувач отримає оновлений список вже з заміною. (рис. 6) Якщо користувач введе більше ніж дозволено сервісом, то отримає помилку і повідомлення з проханням змінити запит (рис. 7).

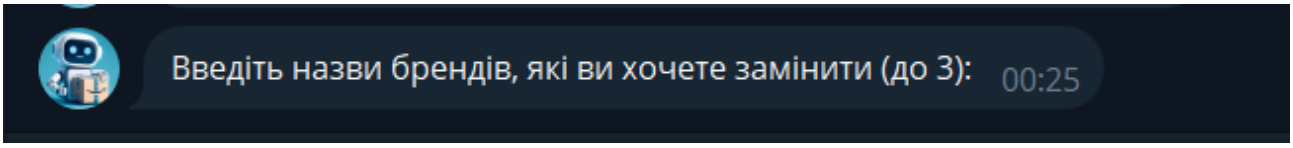


Рисунок 5 – Приклад результату натискання кнопки «Замінити бренди»

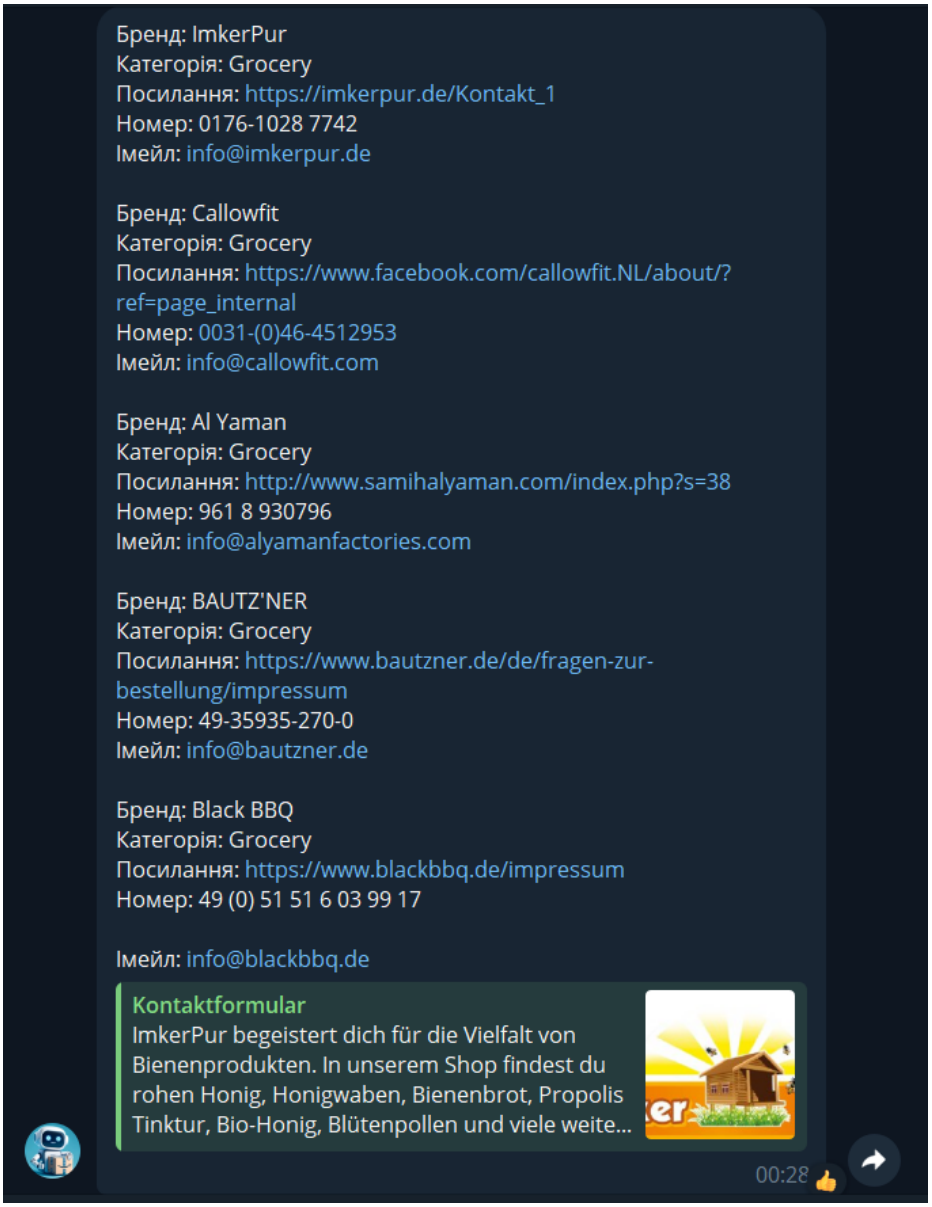


Рисунок 6 – Приклад результату після заміни

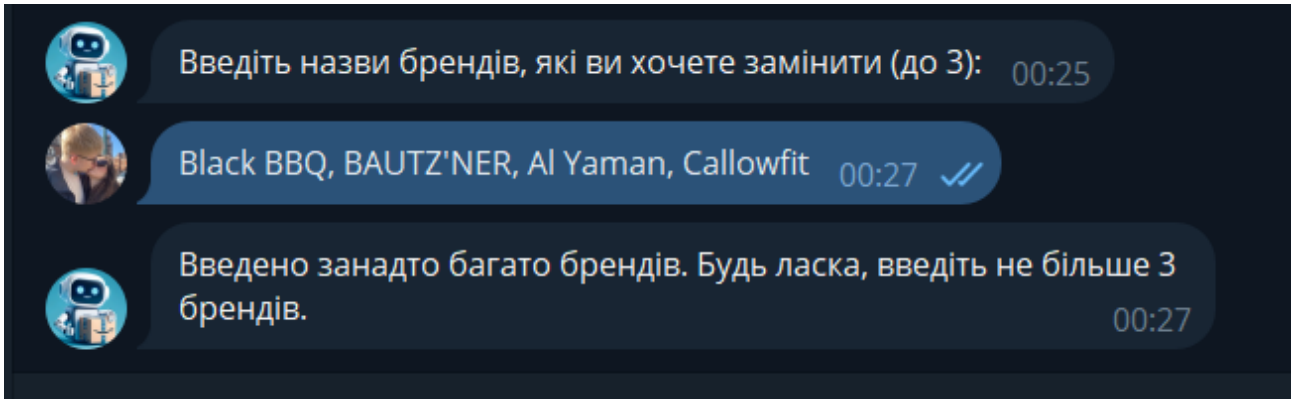


Рисунок 7 – Приклад перебільшення запитів

Графічний інтерфейс телеграм-бота є простим і інтуїтивно зрозумілим, що дозволяє користувачам легко взаємодіяти з системою без необхідності спеціальних знань або навичок. Використання таблиці Excel для зберігання даних дозволяє легко оновлювати інформацію про бренди та їхні товари, забезпечуючи актуальність наданих даних.

ВИСНОВОК

У ході виконання дипломної роботи було розроблено Telegram-бот, призначений для автоматизації збору та надання інформації про бренди в різних категоріях. Основні етапи розробки включали аналіз вимог, проєктування архітектури, реалізацію функціоналу, тестування та впровадження бота.

Розроблений бот має кілька ключових функцій. Він пропонує користувачам інтерактивний інтерфейс, що дозволяє вибирати категорії товарів та отримувати інформацію про бренди в обраних категоріях. Обмеження на кількість запитів дозволяє користувачеві отримати до 20 брендів на день, по 5 брендів за раз, що допомагає уникнути перевантаження системи та забезпечує рівномірний доступ до інформації для всіх користувачів. Крім того, бот надає можливість замінити до трьох брендів на день, дозволяючи користувачам отримувати більш актуальну інформацію відповідно до їхніх потреб.

Збереження контексту запитів та вибраних категорій для кожного користувача є ще однією важливою функцією бота. Це дозволяє уникнути повторів та забезпечити індивідуальний підхід до кожного користувача. Також бот оповіщає користувача про перевищення денного ліміту запитів та замін, пропонуючи відповідні дії для уникнення подальших перевищень.

Автоматизація збору даних реалізована через інтеграцію з Google Sheets, що дозволяє оперативно зчитувати необхідні дані з таблиць. Зібрані дані агрегуються для подальшого аналізу і прогнозування. Зокрема, використовуються методи екстраполяції для прогнозування рейтингу товарів, що дозволяє менеджерам маркетплейсів приймати обґрунтовані рішення щодо управління запасами та маркетинговими стратегіями.

У процесі розробки були використані сучасні технології та інструменти, такі як мова програмування Python, бібліотеки для роботи з Telegram API (python-telegram-bot), Google Sheets API для доступу до даних, а також інші допоміжні бібліотеки (pandas, gspread). Це дозволило створити надійний і масштабований сервіс, який легко адаптується до нових вимог і може бути розширений у

майбутньому. Проведене тестування показало, що бот працює стабільно, швидко обробляє запити користувачів і коректно реагує на всі передбачені ситуації.

Впровадження бота дозволило автоматизувати процес отримання інформації про бренди, що значно полегшило користувачам доступ до необхідних даних і підвищило ефективність їхньої роботи. На основі отриманих результатів можна зробити висновок, що поставлені завдання виконано успішно, і розроблений Telegram-бот повністю відповідає вимогам технічного завдання. Подальші дослідження та розвиток можуть бути спрямовані на розширення функціоналу бота, інтеграцію з іншими сервісами та вдосконалення алгоритмів обробки даних для надання ще більш точних і корисних результатів.

Список використаних джерел

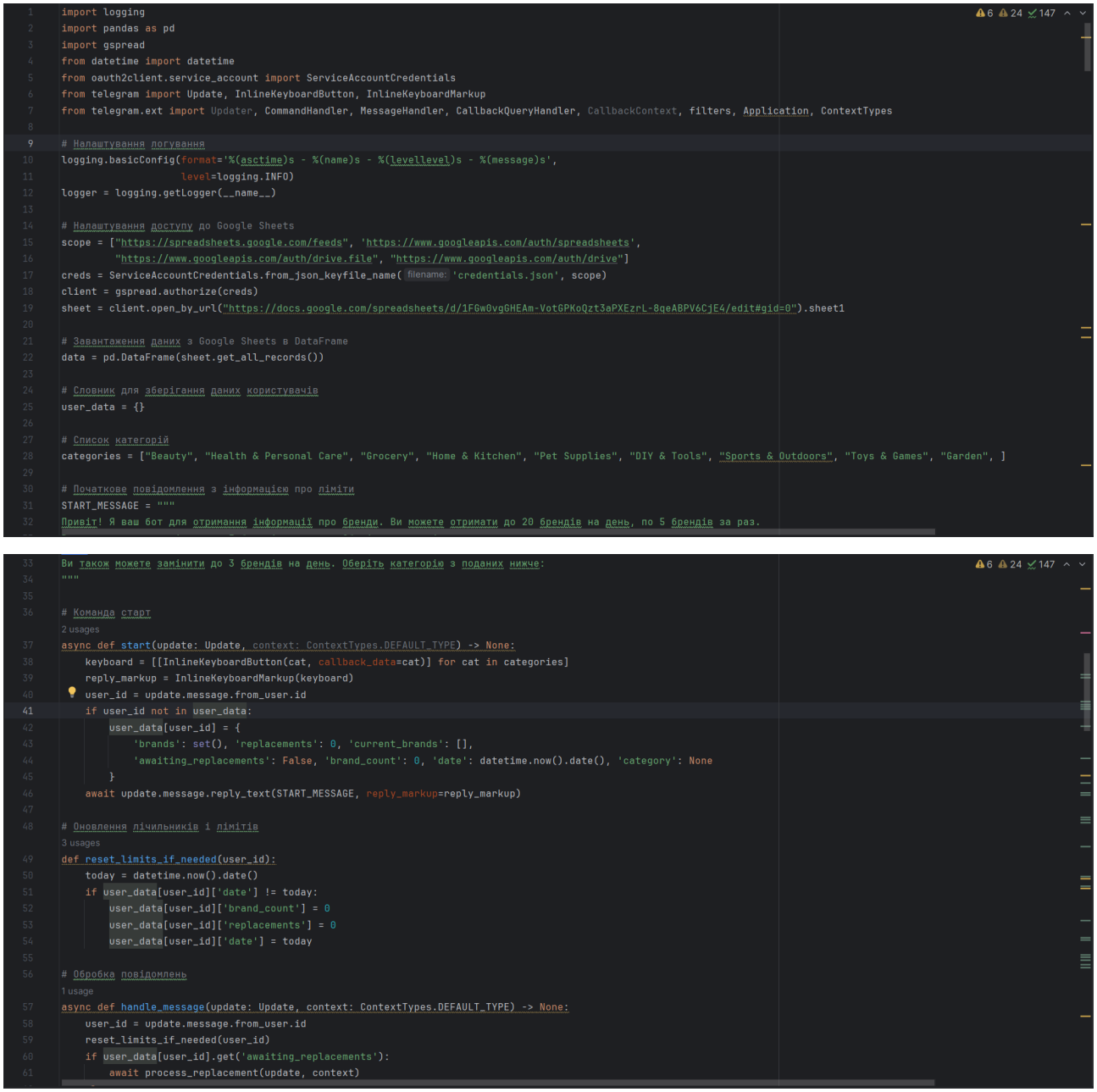
1. Python Software Foundation. Python 3.12.0 final now available. URL: https://web.archive.org/web/20231010192149/https://blog.python.org/2023/10/python-3120-final-now-available.html (дата звернення: 04.03.2024). (дата публікації: 10.10.2023).
2. Stack Overflow. The final Python 2 release marks the end of an era. URL: https://web.archive.org/web/20201016224330/https://stackoverflow.blog/2020/04/23/the-final-python-2-release-marks-the-end-of-an-era/ (дата звернення: 04.03.2024). (дата публікації: 23.04.2020).
3. Python Software Foundation. Python 3.13.0 alpha 1 is now available. URL: https://web.archive.org/web/20231022115655/https://blog.python.org/2023/10/python-3130-alpha-1-is-now-available.html (дата звернення: 24.05.2024). (дата публікації: 22.10.2023).
4. Open Hub. Python language summary. URL: https://openhub.net/p/python/analyses/latest/languages_summary (дата звернення: 11.05.2024).
5. GitHub. Search results for Python repository. URL: https://github.com/search?q=repo%3Apython%2Fcpython++language%3AC (дата звернення: 11.05.2024).
6. Artima. Interview with Guido van Rossum. URL: https://web.archive.org/web/20160901183332/http://www.artima.com/intv/pythonP.html (дата звернення: 30.06.2024).
7. PyInline. PyInline homepage. URL: https://web.archive.org/web/20070115141930/http://pyinline.sourceforge.net/ (дата звернення: 30.04.2024).
8. Shed Skin. Shed Skin blog. URL: https://web.archive.org/web/20110811024224/http://shed-skin.blogspot.com/ (дата звернення: 28.05.2024).

9. ActiveState. Python Cookbook: Recipe 426123. URL: https://web.archive.org/web/20080213130748/http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/426123 (дата звернення: 19.05.2024).
10. Python Documentation. What's New In Python 3.0. URL: https://web.archive.org/web/20081004052720/http://docs.python.org/dev/3.0/whatsnew/3.0.html (дата звернення: 13.03.2024). (дата публікації: 04.10.2008).
11. SPEKA. Microsoft додала Python до Excel. URL: https://speka.media/microsoft-dodala-python-do-excel-p1glxp (дата звернення: 13.03.2024).
12. Programiz. Python IDEs. URL: https://www.programiz.com/python-programming/ide (дата звернення: 13.03.2024).
13. JetBrains. PyCharm: the Python IDE for Professional Developers. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 19.04.2024).
14. Keepa. Keepa: Amazon Price Tracker. URL: https://keepa.com/#! (дата звернення: 25.05.2024).
15. ASIN Spotlight. Home page. URL: https://www.asinspotlight.com/ru/home (дата звернення: 25.05.2024).
16. ASINZen. Home page. URL: https://asinzen.com/ (дата звернення: 24.05.2024).
17. Grabley. Extension home page. URL: https://extension.grabley.net/ (дата звернення: 25.05.2024).
18. Snov.io. Snov.io home page. URL: https://snov.io/ (дата звернення: 24.05.2024).
19. Hunter.io. Hunter.io home page. URL: https://hunter.io/ (дата звернення: 25.05.2024).
20. Analyzer.tools. Analyzer.tools home page. URL: https://www.analyzer.tools/ (дата звернення: 25.05.2024).
21. Ігнатьєв В. Аналітика даних для успіху в бізнесі | LC Work. *Сайт по поиску работы в сфере игорного бизнеса | LC Work.*

URL: https://logincasino.work/ua/blog/4-tipi-analizu-danikh-yaki-prinesut-korist-biznesu (дата звернення: 22.04.2024).
22. ARIMA & SARIMA: Real-World Time Series Forecasting. *neptune.ai*. URL: https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide (дата звернення: 16.05.2024).
23. Biermann I. BI-Tool-Vergleich: Tableau vs. Power BI vs. QlikView. *Compamind*. URL: https://compamind.de/business-intelligence/bi-tool-vergleich/ (дата звернення: 10.05.2024).
24. Soczewica R. What is TOPSIS?. *Medium*. URL: https://robertsoczewica.medium.com/what-is-topsis-b05c50b3cd05 (дата звернення: 03.05.2024).
25. How to clean web scraping data using python beautifulsoup. *Web Data Extractor & Scraper Tool | Try for FREE*. URL: https://webautomation.io/blog/how-to-clean-web-scraping-data-using-python-beautifulsoup/ (дата звернення: 16.03.2024).
26. McKinney W. Python for Data Analysis. O'Reilly Media, Incorporated, 2022. [Розділ 5, с. 129-230]
27. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data / ред. D. Schanafelt. O'Reilly Media, 2016. 548 с. [Розділ 2, Сторінка 45-140, Розділ 4, Сторінка 255-310, Розділ 5, Сторінка 311-410]
28. Grinberg M. Desenvolvimento web com Flask: Desenvolvendo Aplicações web com Python. Novatec Editora, 2018. [Розділ: 1-3, Сторінка: 1-70]
29. Ramm M., Bayer M., Rhodes B. SQLAlchemy: Database Access Using Python. Pearson Education, Limited, 2021. 504 с. [Розділ: 2-3, Сторінка: 25-90]
30. python-telegram-bot v21.2. python-telegram-bot v21.2. URL: https://docs.python-telegram-bot.org/en/stable/ (дата звернення: 24.03.2024).
31. sqlite3 DB-API 2.0 interface for SQLite databases. Python documentation. URL: https://docs.python.org/3/library/sqlite3.html (дата звернення: 05.04.2024).

32. Requests: HTTP for Humans™ – Requests 2.32.3 documentation. Requests: HTTP for Humans™ – Requests 2.32.3 documentation. URL: https://requests.readthedocs.io/en/latest/ (дата звернення: 16.05.2024).
33. Beautiful Soup: We called him Tortoise because he taught us. Swear not by the wiki, the fickle wiki, the inconstant wiki. URL: https://www.crummy.com/software/BeautifulSoup/ (дата звернення: 31.05.2024).

Додаток А.1 Фрагменти програмного коду у PYCharm

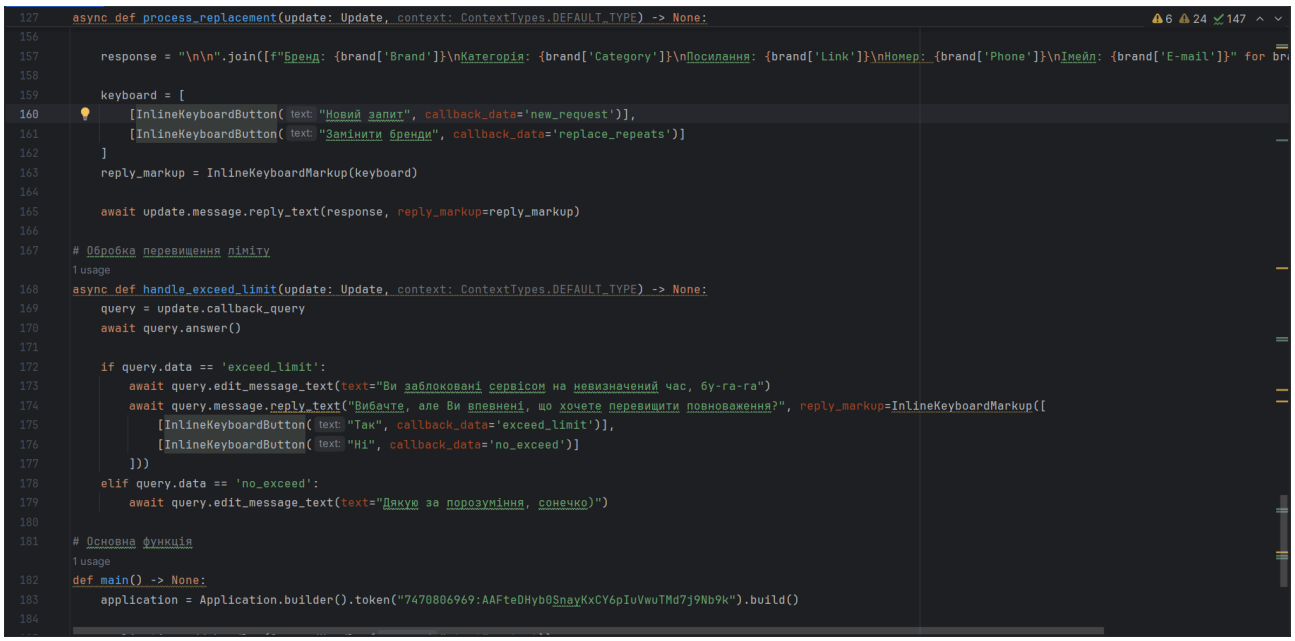


```python
import logging
import pandas as pd
import gspread
from datetime import datetime
from oauth2client.service_account import ServiceAccountCredentials
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Updater, CommandHandler, MessageHandler, CallbackQueryHandler, CallbackContext, filters, Application, ContextTypes

# Налаштування логування
logging.basicConfig(format='%(asctime)s - %(name)s - %(levellevel)s - %(message)s',
                    level=logging.INFO)
logger = logging.getLogger(__name__)

# Налаштування доступу до Google Sheets
scope = ["https://spreadsheets.google.com/feeds", 'https://www.googleapis.com/auth/spreadsheets',
         "https://www.googleapis.com/auth/drive.file", "https://www.googleapis.com/auth/drive"]
creds = ServiceAccountCredentials.from_json_keyfile_name( filename: 'credentials.json', scope)
client = gspread.authorize(creds)
sheet = client.open_by_url("https://docs.google.com/spreadsheets/d/1FGw0vgGHEAm-VotGPKoQzt3aPXEzrL-8qeABPV6CjE4/edit#gid=0").sheet1

# Завантаження даних з Google Sheets в DataFrame
data = pd.DataFrame(sheet.get_all_records())

# Словник для зберігання даних користувачів
user_data = {}

# Список категорій
categories = ["Beauty", "Health & Personal Care", "Grocery", "Home & Kitchen", "Pet Supplies", "DIY & Tools", "Sports & Outdoors", "Toys & Games", "Garden", ]

# Початкове повідомлення з інформацією про ліміти
START_MESSAGE = """
Привіт! Я ваш бот для отримання інформації про бренди. Ви можете отримати до 20 брендів на день, по 5 брендів за раз.
Ви також можете замінити до 3 брендів на день. Оберіть категорію з поданих нижче:
"""

# Команда старт
2 usages
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    keyboard = [[InlineKeyboardButton(cat, callback_data=cat)] for cat in categories]
    reply_markup = InlineKeyboardMarkup(keyboard)
    user_id = update.message.from_user.id
    if user_id not in user_data:
        user_data[user_id] = {
            'brands': set(), 'replacements': 0, 'current_brands': [],
            'awaiting_replacements': False, 'brand_count': 0, 'date': datetime.now().date(), 'category': None
        }
    await update.message.reply_text(START_MESSAGE, reply_markup=reply_markup)

# Оновлення лічильників і лімітів
3 usages
def reset_limits_if_needed(user_id):
    today = datetime.now().date()
    if user_data[user_id]['date'] != today:
        user_data[user_id]['brand_count'] = 0
        user_data[user_id]['replacements'] = 0
        user_data[user_id]['date'] = today

# Обробка повідомлень
1 usage
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    user_id = update.message.from_user.id
    reset_limits_if_needed(user_id)
    if user_data[user_id].get('awaiting_replacements'):
        await process_replacement(update, context)
```

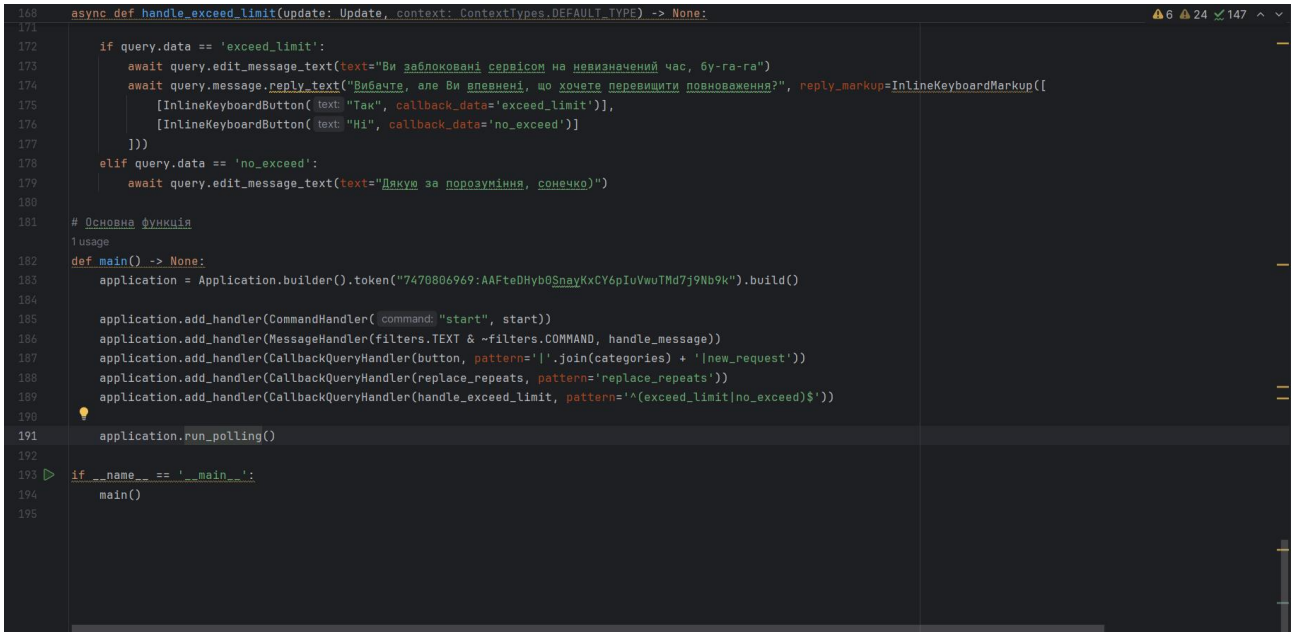
```python
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
        await update.message.reply_text("Будь ласка, скористайтесь командою /start для початку роботи.")

# Обробка натискань на кнопки
1 usage
async def button(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    query = update.callback_query
    await query.answer()
    user_id = query.from_user.id
    category = query.data.lower().strip()

    if user_id not in user_data:
        await query.edit_message_text(text="Будь ласка, скористайтесь командою /start для початку роботи.")
        return

    reset_limits_if_needed(user_id)

    if category == 'new_request':
        await start(update.callback_query, context)
        return

    if user_data[user_id]['brand_count'] >= 20:
        await query.edit_message_text(text="Ви перевищили ліміт запитів на сьогодні. Спробуйте завтра.")
        return

    user_data[user_id]['category'] = category
    user_brands = user_data[user_id]['brands']
    category_brands = data[data['Category'].str.lower() == category]

    new_brands = category_brands[~category_brands['Brand'].isin(user_brands)].head(5)

    if new_brands.empty:
```

```python
async def button(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
        return

    user_data[user_id]['brands'].update(new_brands['Brand'].values)
    user_data[user_id]['current_brands'] = new_brands[['Brand', 'Category', 'Link', 'Phone', 'E-mail']].to_dict('records')
    user_data[user_id]['brand_count'] += len(new_brands)

    response = "\n\n".join([f"Бренд: {row['Brand']}\nКатегорія: {row['Category']}\nПосилання: {row['Link']}\nНомер: {row['Phone']}\nІмейл: {row['E-mail']}" for _, row in ne

    keyboard = [
        [InlineKeyboardButton( text: "Новий запит", callback_data='new_request')],
        [InlineKeyboardButton( text: "Замінити бренди", callback_data='replace_repeats')]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)

    remaining = 20 - user_data[user_id]['brand_count']
    await query.edit_message_text( text: f"{response}\n\nВи можете отримати ще {remaining} брендів сьогодні.", reply_markup=reply_markup)

# Обробка натискань на кнопки для заміни повторів
1 usage
async def replace_repeats(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    query = update.callback_query
    await query.answer()
    user_id = query.from_user.id

    reset_limits_if_needed(user_id)

    if user_data[user_id]['replacements'] >= 3:
        await query.message.reply_text("Ви перевищили ліміт замін на сьогодні. Спробуйте завтра.")
        return

    user_data[user_id]['awaiting_replacements'] = True
```

```python
    await query.message.reply_text("Введіть назви брендів, які ви хочете замінити (до 3):")

# Обробка введення брендів для заміни
1 usage
async def process_replacement(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    user_id = update.message.from_user.id
    replacements = update.message.text.split(',')
    replacements = [r.strip() for r in replacements if r.strip()]

    if len(replacements) > 3:
        await update.message.reply_text("Введено занадто багато брендів. Будь ласка, введіть не більше 3 брендів.")
        return

    current_brands = {brand['Brand']: brand for brand in user_data[user_id]['current_brands']}
    invalid_brands = [r for r in replacements if r not in current_brands]

    if invalid_brands:
        await update.message.reply_text(f"Такого бренду не знайдено: {', '.join(invalid_brands)}. Повторіть запит.")
        return

    user_brands = user_data[user_id]['brands']
    category = user_data[user_id]['category']
    for replacement in replacements:
        if replacement in current_brands:
            user_brands.remove(replacement)

    new_brands = data[(~data['Brand'].isin(user_brands)) & (data['Category'].str.lower() == category)].head(len(replacements))
    user_data[user_id]['brands'].update(new_brands['Brand'].values)
    user_data[user_id]['replacements'] += 1
    user_data[user_id]['awaiting_replacements'] = False

    updated_brands = [brand for brand in user_data[user_id]['current_brands'] if brand['Brand'] not in replacements] + new_brands.to_dict('records')
```



```python
async def process_replacement(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:

    response = "\n\n".join([f"Бренд: {brand['Brand']}\nКатегорія: {brand['Category']}\nПосилання: {brand['Link']}\nНомер: {brand['Phone']}\nІмейл: {brand['E-mail']}" for br

    keyboard = [
        [InlineKeyboardButton( text: "Новий запит", callback_data='new_request')],
        [InlineKeyboardButton( text: "Замінити бренди", callback_data='replace_repeats')]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)

    await update.message.reply_text(response, reply_markup=reply_markup)

# Обробка перевищення ліміту
1 usage
async def handle_exceed_limit(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    query = update.callback_query
    await query.answer()

    if query.data == 'exceed_limit':
        await query.edit_message_text(text="Ви заблоковані сервісом на невизначений час, бу-га-га")
        await query.message.reply_text("Вибачте, але Ви впевнені, що хочете перевищити повноваження?", reply_markup=InlineKeyboardMarkup([
            [InlineKeyboardButton( text: "Так", callback_data='exceed_limit')],
            [InlineKeyboardButton( text: "Ні", callback_data='no_exceed')]
        ]))
    elif query.data == 'no_exceed':
        await query.edit_message_text(text="Дякую за порозуміння, сонечко)")

# Основна функція
1 usage
def main() -> None:
    application = Application.builder().token("7470806969:AAFteDHybOSnayKxCY6pIuVwuTMd7j9Nb9k").build()
```



```python
async def handle_exceed_limit(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:

    if query.data == 'exceed_limit':
        await query.edit_message_text(text="Ви заблоковані сервісом на невизначений час, бу-га-га")
        await query.message.reply_text("Вибачте, але Ви впевнені, що хочете перевищити повноваження?", reply_markup=InlineKeyboardMarkup([
            [InlineKeyboardButton( text: "Так", callback_data='exceed_limit')],
            [InlineKeyboardButton( text: "Ні", callback_data='no_exceed')]
        ]))
    elif query.data == 'no_exceed':
        await query.edit_message_text(text="Дякую за порозуміння, сонечко)")

# Основна функція
1 usage
def main() -> None:
    application = Application.builder().token("7470806969:AAFteDHybOSnayKxCY6pIuVwuTMd7j9Nb9k").build()

    application.add_handler(CommandHandler( command: "start", start))
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))
    application.add_handler(CallbackQueryHandler(button, pattern='|'.join(categories) + '|new_request'))
    application.add_handler(CallbackQueryHandler(replace_repeats, pattern='replace_repeats'))
    application.add_handler(CallbackQueryHandler(handle_exceed_limit, pattern='^(exceed_limit|no_exceed)$'))

    application.run_polling()

if __name__ == '__main__':
    main()
```

Додаток А.2 Фрагменти програмного коду

```
import logging
import pandas as pd
import gspread
from datetime import datetime
from oauth2client.service_account import ServiceAccountCredentials
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Updater, CommandHandler, MessageHandler, CallbackQueryHandler, CallbackContext, filters, Application, ContextTypes

# Налаштування логування
logging.basicConfig(format='%(asctime)s - %(name)s - %(levellevel)s - %(message)s',
level=logging.INFO)
logger = logging.getLogger(__name__)

# Налаштування доступу до Google Sheets
scope = ["https://spreadsheets.google.com/feeds",
'https://www.googleapis.com/auth/spreadsheets',
"https://www.googleapis.com/auth/drive.file",
"https://www.googleapis.com/auth/drive"]
creds = ServiceAccountCredentials.from_json_keyfile_name('credentials.json', scope)
client = gspread.authorize(creds)
sheet = client.open_by_url("https://docs.google.com/spreadsheets/d/1FGw0vgGHEAm-VotGPKoQzt3aPXEzrL-8qeABPV6CjE4/edit#gid=0").sheet1

# Завантаження даних з Google Sheets в DataFrame
data = pd.DataFrame(sheet.get_all_records())
```

```
# Словник для зберігання даних користувачів
user_data = {}

# Список категорій
categories = ["Beauty", "Health & Personal Care", "Grocery", "Home & Kitchen",
"Pet Supplies", "DIY & Tools", "Sports & Outdoors", "Toys & Games", "Garden",
]

# Початкове повідомлення з інформацією про ліміти
START_MESSAGE = """
Привіт! Я ваш бот для отримання інформації про бренди. Ви можете отримати
до 20 брендів на день, по 5 брендів за раз.
Ви також можете замінити до 3 брендів на день. Оберіть категорію з поданих
нижче:
"""

# Команда старт
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
None:
keyboard = [[InlineKeyboardButton(cat, callback_data=cat)] for cat in categories]
reply_markup = InlineKeyboardMarkup(keyboard)
user_id = update.message.from_user.id
if user_id not in user_data:
user_data[user_id] = {
'brands': set(), 'replacements': 0, 'current_brands': [],
'awaiting_replacements': False, 'brand_count': 0, 'date': datetime.now().date(),
'category': None
}
```

```
await update.message.reply_text(START_MESSAGE, reply_markup=reply_markup)

# Оновлення лічильників і лімітів
def reset_limits_if_needed(user_id):
today = datetime.now().date()
if user_data[user_id]['date'] != today:
user_data[user_id]['brand_count'] = 0
user_data[user_id]['replacements'] = 0
user_data[user_id]['date'] = today

# Обробка повідомлень
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
user_id = update.message.from_user.id
reset_limits_if_needed(user_id)
if user_data[user_id].get('awaiting_replacements'):
await process_replacement(update, context)
else:
await update.message.reply_text("Будь ласка, скористайтесь командою /start для початку роботи.")

# Обробка натискань на кнопки
async def button(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
query = update.callback_query
await query.answer()
user_id = query.from_user.id
category = query.data.lower().strip()
```

```
if user_id not in user_data:
await query.edit_message_text(text="Будь ласка, скористайтесь командою /start для початку роботи.")
return

reset_limits_if_needed(user_id)

if category == 'new_request':
await start(update.callback_query, context)
return

if user_data[user_id]['brand_count'] >= 20:
await query.edit_message_text(text="Ви перевищили ліміт запитів на сьогодні. Спробуйте завтра.")
return

user_data[user_id]['category'] = category
user_brands = user_data[user_id]['brands']
category_brands = data[data['Category'].str.lower() == category]

new_brands = category_brands[~category_brands['Brand'].isin(user_brands)].head(5)

if new_brands.empty:
await query.edit_message_text(text="Категорія не знайдена або немає нових брендів. Будь ласка, виберіть іншу категорію.")
return

user_data[user_id]['brands'].update(new_brands['Brand'].values)
```

```
user_data[user_id]['current_brands'] = new_brands[['Brand', 'Category', 'Link', 'Phone', 'E-mail']].to_dict('records')
user_data[user_id]['brand_count'] += len(new_brands)

response = "\n\n".join([f"Бренд: {row['Brand']}\nКатегорія: {row['Category']}\nПосилання: {row['Link']}\nНомер: {row['Phone']}\nІмейл: {row['E-mail']}" for _, row in new_brands.iterrows()])

keyboard = [
[InlineKeyboardButton("Новий запит", callback_data='new_request')],
[InlineKeyboardButton("Замінити бренди", callback_data='replace_repeats')]
]
reply_markup = InlineKeyboardMarkup(keyboard)

remaining = 20 - user_data[user_id]['brand_count']
await query.edit_message_text(f"{response}\n\nВи можете отримати ще {remaining} брендів сьогодні.", reply_markup=reply_markup)

# Обробка натискань на кнопки для заміни повторів
async def replace_repeats(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
query = update.callback_query
await query.answer()
user_id = query.from_user.id

reset_limits_if_needed(user_id)

if user_data[user_id]['replacements'] >= 3:
await query.message.reply_text("Ви перевищили ліміт замін на сьогодні. Спробуйте завтра.")
```

```
return

user_data[user_id]['awaiting_replacements'] = True
await query.message.reply_text("Введіть назви брендів, які ви хочете замінити (до 3):")

# Обробка введення брендів для заміни
async def process_replacement(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
user_id = update.message.from_user.id
replacements = update.message.text.split(',')
replacements = [r.strip() for r in replacements if r.strip()]

if len(replacements) > 3:
await update.message.reply_text("Введено занадто багато брендів. Будь ласка, введіть не більше 3 брендів.")
return

current_brands = {brand['Brand']: brand for brand in user_data[user_id]['current_brands']}
invalid_brands = [r for r in replacements if r not in current_brands]

if invalid_brands:
await update.message.reply_text(f"Такого бренду не знайдено: {', '.join(invalid_brands)}. Повторіть запит.")
return

user_brands = user_data[user_id]['brands']
category = user_data[user_id]['category']
for replacement in replacements:
```

```
if replacement in current_brands:
user_brands.remove(replacement)

new_brands = data[(~data['Brand'].isin(user_brands)) & (data['Category'].str.lower() == category)].head(len(replacements))
user_data[user_id]['brands'].update(new_brands['Brand'].values)
user_data[user_id]['replacements'] += 1
user_data[user_id]['awaiting_replacements'] = False

updated_brands = [brand for brand in user_data[user_id]['current_brands'] if brand['Brand'] not in replacements] + new_brands.to_dict('records')
user_data[user_id]['current_brands'] = updated_brands

response = "\n\n".join([f"Бренд: {brand['Brand']}\nКатегорія: {brand['Category']}\nПосилання: {brand['Link']}\nНомер: {brand['Phone']}\nІмейл: {brand['E-mail']}" for brand in updated_brands])

keyboard = [
[InlineKeyboardButton("Новий запит", callback_data='new_request')],
[InlineKeyboardButton("Замінити бренди", callback_data='replace_repeats')]
]
reply_markup = InlineKeyboardMarkup(keyboard)

await update.message.reply_text(response, reply_markup=reply_markup)

# Обробка перевищення ліміту
async def handle_exceed_limit(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
query = update.callback_query
await query.answer()
```

```
if query.data == 'exceed_limit':
await query.edit_message_text(text="Ви заблоковані сервісом на невизначений час, бу-га-га")
await query.message.reply_text("Вибачте, але Ви впевнені, що хочете перевищити повноваження?", reply_markup=InlineKeyboardMarkup([
[InlineKeyboardButton("Так", callback_data='exceed_limit')],
[InlineKeyboardButton("Ні", callback_data='no_exceed')]
]))
elif query.data == 'no_exceed':
await query.edit_message_text(text="Дякую за порозуміння, сонечко)")

# Основна функція
def main() -> None:
application = Application.builder().token("7470806969:AAFteDHyb0SnayKxCY6pIuVwuTMd7j9Nb9k").build()

application.add_handler(CommandHandler("start", start))
application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))
application.add_handler(CallbackQueryHandler(button,
pattern='|'.join(categories) + '|new_request'))
application.add_handler(CallbackQueryHandler(replace_repeats,
pattern='replace_repeats'))
application.add_handler(CallbackQueryHandler(handle_exceed_limit,
pattern='^(exceed_limit|no_exceed)$'))

application.run_polling()
```

```
if __name__ == '__main__':
main()
```

Додаток Б – Таблиця Excel

https://docs.google.com/spreadsheets/d/1FGw0vgGHEAm-VotGPKoQzt3aPXEzrL-8qeABPV6CjE4/edit#gid=0

Додаток В – Приклад виконання програми

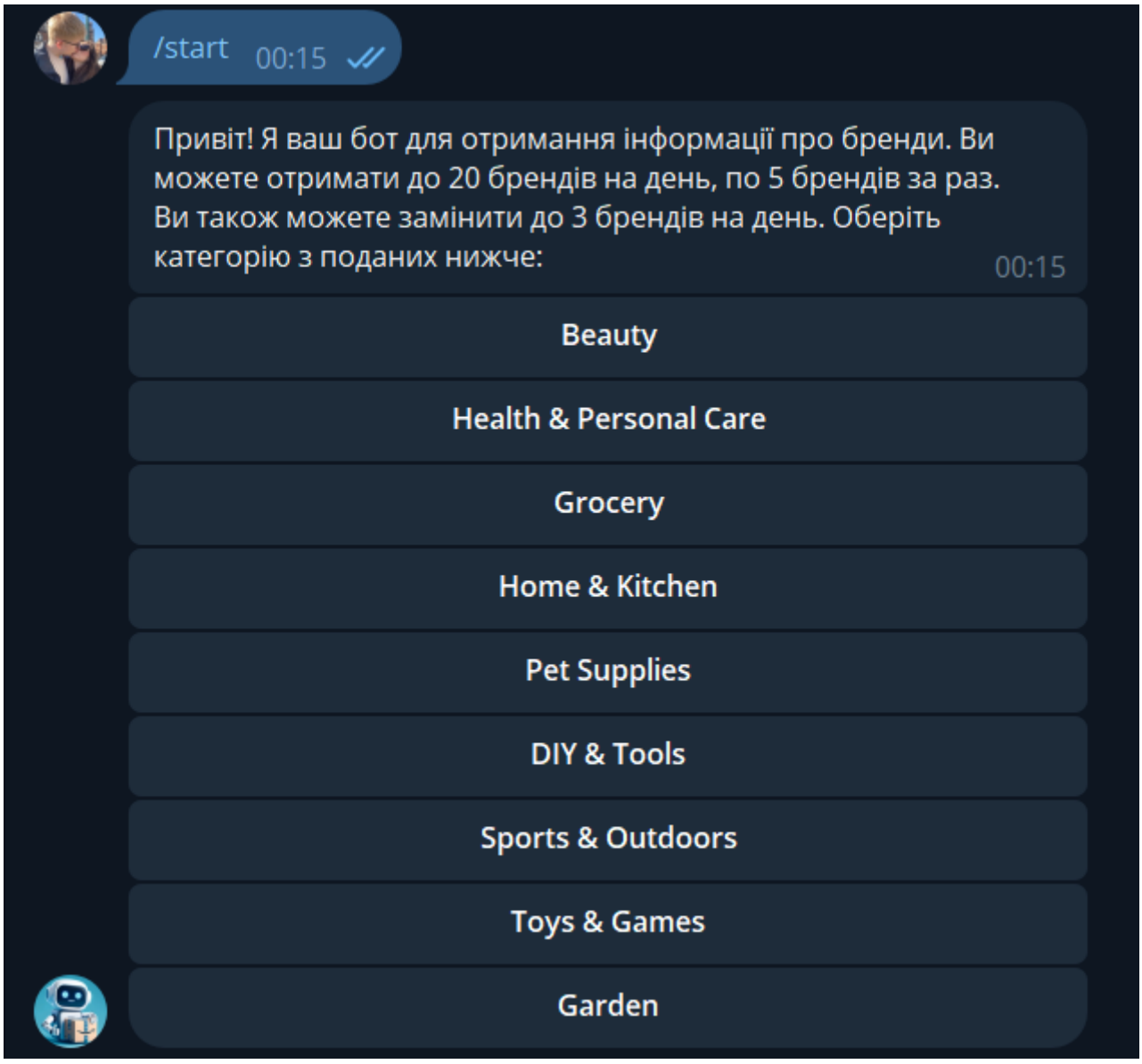


Рисунок 1 – Приклад привітання від боту

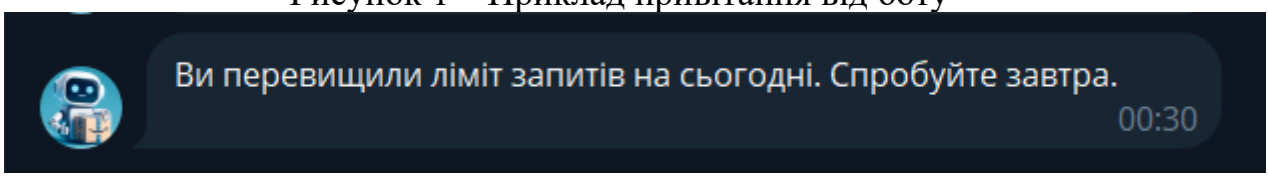


Рисунок 2 – Результат перевищення ліміту запитів

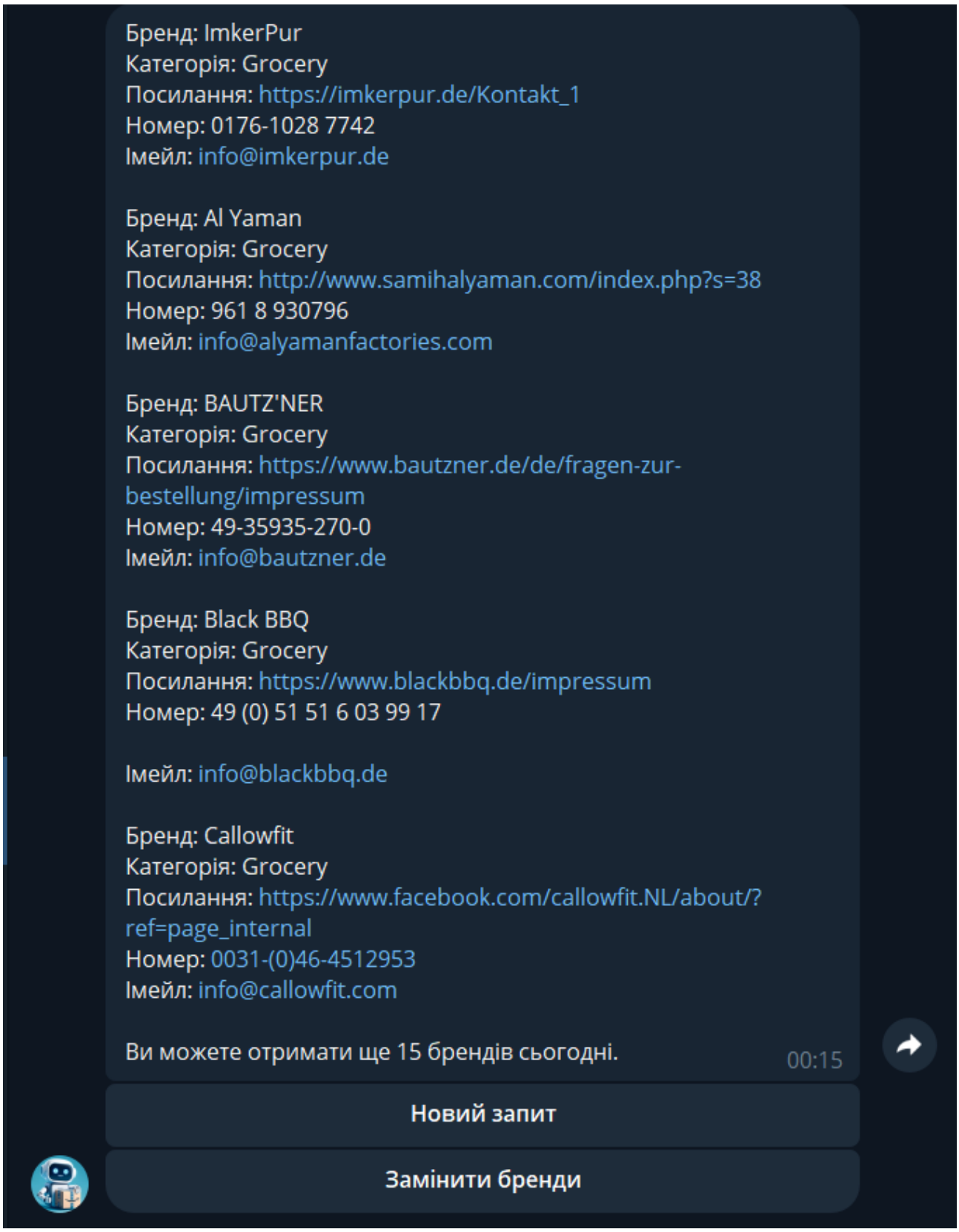


Рисунок 3 – Приклад вибору категорії Grocery(Продукти)

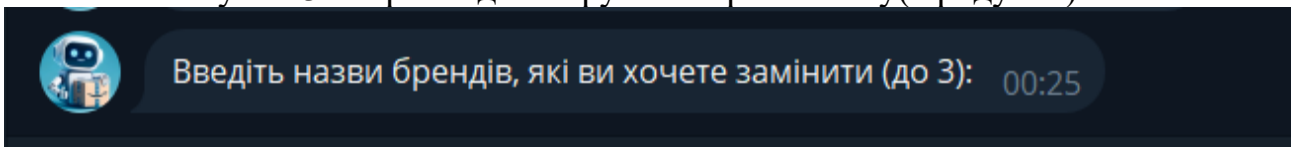


Рисунок 4 – Приклад результату натискання кнопки «Замінити бренди»

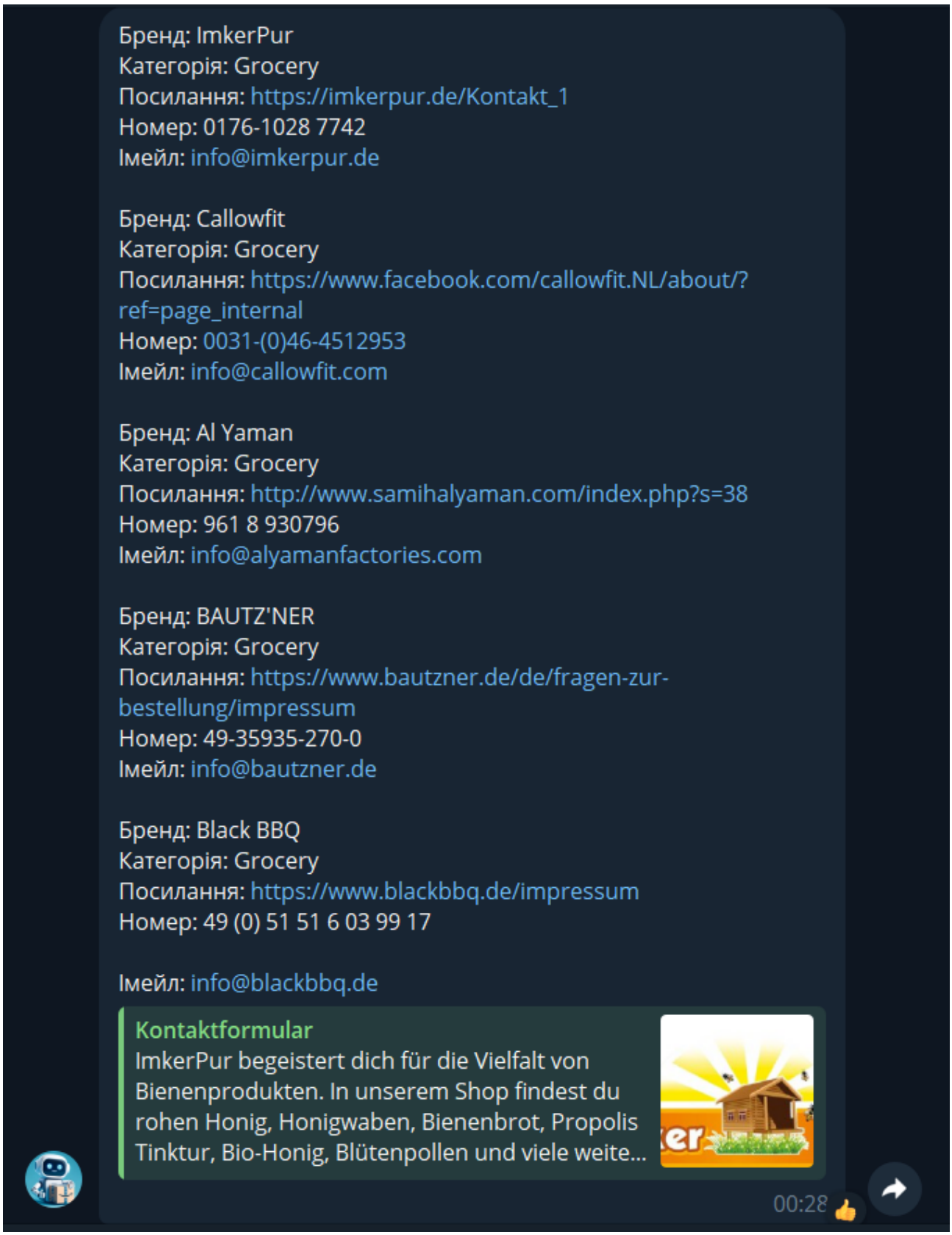


Рисунок 5 – Приклад результату після заміни

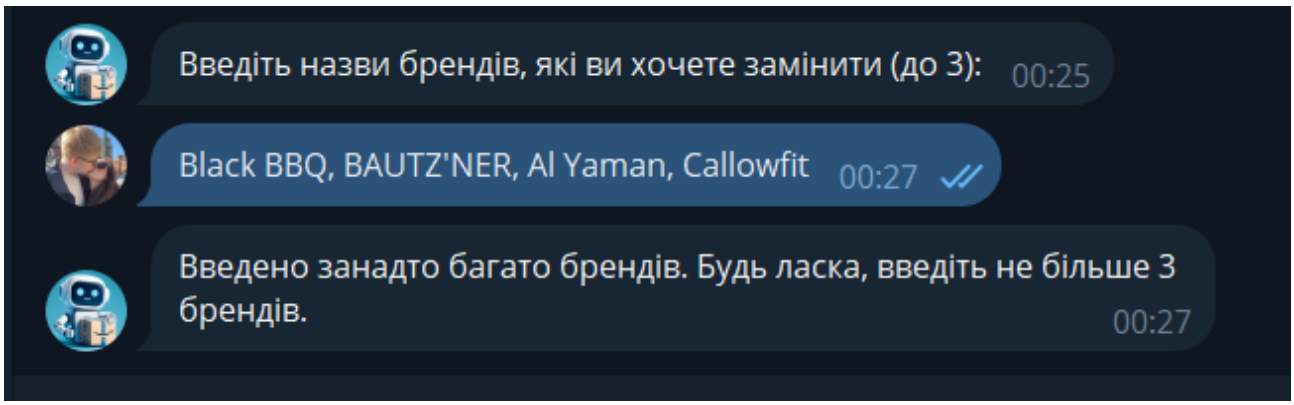


Рисунок 6 – Приклад перебільшення запитів