МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
« 13 » грудня 2024р.	« 13 » грудня 2024р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності 122 «Комп'ютерні науки»
(код і назва спеціальності)

освітньо-професійної програми Управління інформацією та аналітика даних

на тему: Дослідження методів збору та аналізу даних для інформаційної підтримки формування асортименту продукції ПрАТ «Оболонь»

Виконав: здобувач 2 курсу, групи КН-2-4М

Ревенко Михайло Олександрович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Харкянен Олена Валеріївна _____
(прізвище, ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент Гефрій Бараков _____
(ім'я та прізвище) (підпис)

Я як здобувач Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав і не одержував недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач _____
(підпис)

Київ - 2024 р.

2

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь магістр
Спеціальність 122 «Комп'ютерні науки»
(код і назва)

Освітньо-професійна програма Управління інформацією та аналітика даних
(назва)

ЗАТВЕРДЖУЮ
Завідувач
кафедри Інформаційних технологій,
штучного інтелекту і кібербезпеки

Грибков С.В.
"07" жовтня 2024року

З А В Д А Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Ревенка Михайла Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів збору та аналізу даних для інформаційної підтримки формування асортименту продукції ПрАТ «Оболонь»

керівник роботи Харкянен Олена Валеріївна, кандидат технічних наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від 7 жовтня 2024 року №884-кс
2. Строк подання здобувачем роботи: 6 грудня 2024 року
3. Вихідні дані до роботи:
1) Відомості щодо ПрАТ «Оболонь», його структуру та історію.
2) Дані щодо асортименту продукції ПрАТ «Оболонь».
3) Способи збору даних підприємством, відомості щодо їхньої подальшої обробки та застосування.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Розділ 1. Дослідження методів збору та аналізу даних
Розділ 2. Огляд видів методів аналізу даних
Розділ 3. Огляд та обґрунтування вибору засобів для вирішення задач
Розділ 4. Розробка програмного забезпечення для збору та аналізу відгуків
5. Перелік графічного матеріалу:
1) Логотипи використовуваних засобів для реалізації програмного застосунку.
2) Приклади інтерфейсу ПЗ для збору та аналізу відгуків.
3) Функціональна модель діяльності ПрАТ «Оболонь» з формування рекомендацій щодо асортименту продукції.

3

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Харкянен О.В., доцент	07.10.2024	11.11.2024
2	Харкянен О.В., доцент	07.10.2024	15.11.2024
3	Харкянен О.В., доцент	07.10.2024	18.11.2024
4	Харкянен О.В., доцент	07.10.2024	29.11.2024

7. Дата видачі завдання: 7 жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження методів збору і аналізу даних, обґрунтування їхнього вибору.	07.10.2024-14.10.2024	Виконано
2	Дослідження діяльності ПрАТ «Оболонь».	15.10.2024-21.10.2024	Виконано
3	Постановка задач щодо виконання збору і аналізу відгуків.	22.10.2024-25.10.2024	Виконано
4	Дослідження та вибір методу аналізу тексту.	26.10.2024-04.11.2024	Виконано
5	Розробка застосунку для збору та обробки текстових відгуків для формування рекомендацій щодо формування асортименту ПрАТ «Оболонь».	05.11.2024-12.11.2024	Виконано
6	Оформлення роботи.	13.11.2024-25.11.2024	Виконано
7	Оформлення автореферату.	26.11.2024-1.12.2024	Виконано
8	Оформлення презентації.	02.12.2024-06.12.2024	Виконано

Здобувач _____ (підпис) Ревенко М.О. (прізвище та ініціали)

Керівник роботи _____ (підпис) Харкянен О.В. (прізвище та ініціали)

АНОТАЦІЯ

Ревенко Михайло Олександрович – Дослідження методів збору та аналізу даних для інформаційної підтримки формування асортименту продукції ПрАТ «Оболонь».

Кваліфікаційна робота: 73 сторінок, 38 рисунків, 3 таблиць, 5 додатків, 26 джерел.

Містить проведене дослідження в області методів збору та аналізу даних з використанням технологій, заснованих на використанні нейронних мереж та мовних моделей. У ході проведеного дослідження проаналізовано принципи роботи, переваги та недоліки описаних методів збору та аналізу даних. Серед досліджених методів було обрано найбільш актуальні та ефективні для роботи із текстовими масивами та здійснено їх реалізацію у складі програмного забезпечення, яке автоматично виконує збір та аналіз даних з метою здійснення інформаційної підтримки формування асортименту ПрАТ «Оболонь». Було описано програмний код та продемонстровано роботу застосунку у вигляді ілюстрацій.

Ключові слова: ЗБІР ДАНИХ, АНАЛІЗ ДАНИХ, АНАЛІЗ ТОНАЛЬНОСТІ, БАЗА ДАНИХ, МОВНА МОДЕЛЬ, КЛАСИФІКАЦІЯ ТЕКСТУ, АСОРТИМЕНТ ПРОДУКЦІЇ, ПРАТ «ОБОЛОНЬ».

ABSTRACT

Mykhailo Oleksandrovych Revenko – Research on methods of data collection and analysis for informational support of forming the product range of PJSC "Obolon".

Qualification work: 73 pages, 38 figures, 3 tables, 5 appendices, 26 sources.

Contains research in the field of methods of data collection and analysis using technologies based on the use of neural networks and language models. In the course of the research, the working principles, advantages and disadvantages of the described methods of data collection and analysis were analyzed. Among the researched methods, the most relevant and effective ones for working with text arrays were selected and implemented as part of software that automatically collects and analyzes data in order to provide informational support for the formation of the Obolon PJSC assortment. The program code was described and the operation of the application was demonstrated in the form of illustrations.

Keywords: DATA COLLECTION, DATA ANALYSIS, TONE ANALYSIS, DATABASE, LANGUAGE MODEL, TEXT CLASSIFICATION, PRODUCT RANGE, PJSC "OBOLON".

ЗМІСТ

ВСТУП

Написання кваліфікаційної роботи, будучи фінальною віхою до здобуття освітнього ступеню магістра, є відповідальною та комплексною задачею, яка включає в себе низку робіт, починаючи від теоретичного дослідження теми роботи та закінчуючи практичною реалізацією задач, що підкреслюють значущість теми та її актуальність. Робота над кваліфікаційною роботою полягає окрім дослідження ще у закріпленні набутих під час проходження виробничої та переддипломної практик на підприємстві знань. Дана кваліфікаційна робота не є винятком, вона, загалом, складається із розділів, у яких виконується дослідження з метою вибору найкращих для реалізації поставлених у роботі задач методів збору та аналізу даних.

Отже, зрозумівши важливість роботи та її загальний вміст, можемо перейти до теми цієї роботи та її обґрунтування.

Темою кваліфікаційної роботи є «Дослідження методів збору та аналізу даних для інформаційної підтримки формування асортименту продукції ПрАТ «Оболонь», її вибір зумовлений отриманими відомостями щодо підприємства, якого вона стосується, після проходження на ньому згаданих вище практик.

Актуальність теми. Актуальність теми полягає у тому, що в умовах сучасного становища на ринку напоїв все гостріш стоїть питання конкуренції, особливо для підприємств масштабами схожими з ПрАТ «Оболонь». Одним із рішень щодо питань підвищення ефективності роботи підприємства є впровадження нових інформаційних технологій у його внутрішні процеси, особливо стосовно обробки та збору даних. Очевидно, що питання часу на проведення аналізу інформації напряму пов’язане із прибутком, тому ключовим є впровадження інформаційних технологій, що сприятимуть підвищенню ефективності прийняття управлінських рішень.

Для підприємств є дуже важливим здійснювати збір даних пов’язаних із реакцією на свою продукцію і ПрАТ «Оболонь» не є винятком. Збір відгуків легко недооцінити порівнюючи із іншими процесами, проте саме пов’язані із цим

роботи дозволяють оперативним чином реагувати на потреби покупців та вносити зміни у асортимент продукції для уникнення непередбачуваних збитків. Не менш важливим є потреба підприємства у зберіганні даних, оскільки вони мають вагомий потенціал в плані аналізу, наприклад, підприємство може легко визначити сезонний фактор у продажах певних напоїв на основі кількості відповідних відгуків, або визначити певні кореляції між напоями та основними скаргами, наприклад стосовно якості тари чи непривабливості її дизайну.

Актуальність теми зумовлена необхідністю впровадження сучасних аналітичних методів, які дозволяють приймати обґрунтовані рішення при формуванні асортименту продукції. Збір та аналіз даних можуть надати змогу підприємству глибше зрозуміти потреби споживачів, передбачити зміни в їхніх вподобаннях та швидко адаптувати асортимент продукції до нових умов.

Стосовно ПрАТ «Оболонь», як однієї із найбільших компаній в Україні у сфері напоїв [1], можна повідомити про її високий потенціал щодо впровадження актуальних методів збору та аналізу даних, які допоможуть їй підвищити ефективність виробництва продукції, згідно вимог ринку напоїв.

Зв’язок роботи з науковими програмами. Проведене дослідження є актуальним для кафедри інформаційних технологій, штучного інтелекту і кібербезпеки факультету автоматизації і комп’ютерних систем Національного університету харчових технологій (НУХТ), оскільки його тематика узгоджується з планом наукової роботи кафедри.

Мета дослідження. Метою дослідження є визначення способів надання та здійснення інформаційної підтримки формування асортименту продукції ПрАТ «Оболонь» на основі використання сучасних підходів та технологій збору та аналізу даних.

Завдання дослідження. Для досягнення мети потрібно розв’язати наступні завдання:

- дослідження сучасних підходів до збору та аналізу даних за допомогою сучасних методів та платформ;

- аналіз настрою відгуків на продукцію та класифікації зібраних даних точним та швидким чином;
- здійснення обробки, підготовки та звернення до даних;
- забезпечення зберігання даних, дотримуючись їхньої цілісності;
- побудова рейтингу продукції для вивчення реакції споживачів на продукцію ПрАТ «Оболонь», забезпечивши таким чином інформаційну підтримку формування асортименту продукції підприємства.

Об’єкт дослідження. Об’єктом дослідження є відгуки на продукцію ПрАТ «Оболонь», що представляють собою великий масив текстових даних, яким притаманні певні настрої.

Предмет дослідження. Методи збору та аналізу текстових даних, якими є відгуки на продукцію ПрАТ «Оболонь».

Методи дослідження. До методів дослідження можна віднести системний аналіз, функціональне моделювання, метод штучного інтелекту для обробки природної мови – трансформер.

Наукова новизна одержаних результатів. Наукова новизна дослідження полягає у запропонованому підході до формування пропозицій щодо коригування асортименту продукції харчового підприємства на основі обробки відгуків покупців, отриманих із соціальних мереж та чат-ботів з використанням мовних моделей та нейронних мереж.

Практичне значення отриманих результатів. Практичне значення роботи полягає у застосуванні сучасних підходів щодо аналізу та збору даних в рамках задач інформаційної підтримки формування асортименту ПрАТ «Оболонь» за допомогою розробленого спеціалізованого застосунку.

Особистий внесок здобувача полягає у реалізації наступних завдань:

- визначено найдоцільніші для реалізації поставлених завдань стосовно інформаційної підтримки формування асортименту ПрАТ «Оболонь»

методи збору даних та аналізу, які засновані на сучасних технологіях, обґрунтовано доцільність вибору;

- запропоновано та реалізовано сучасні підходи щодо збору даних із соціальних мереж та чат-ботів;
- застосовано методи, які базуються на використанні нейронних мереж та мовних моделей для виконання аналізу та класифікації відгуків на продукцію ПрАТ «Оболонь»;
- адаптовано технології для вирішення задач обробки текстових відгуків на продукцію ПрАТ «Оболонь»;
- розроблено програмний продукт, який на основі оброблених відгуків формує рейтинг продукції, даючи якісну інформаційну підтримку формування асортименту ПрАТ «Оболонь», забезпечуючи при цьому необхідний для роботи функціонал.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ МЕТОДІВ ЗБОРУ ТА АНАЛІЗУ ДАНИХ

1.1. Мета і обґрунтування доцільності проведення дослідження

Процес збору та аналізу даних є однією із найважливіших для ПрАТ «Оболонь» діяльностей. Протягом даного розділу було виконано дослідження основних методів збору і аналізу даних та обґрунтування важливості їхньої ролі в діяльності підприємства. Даний розділ включає в себе класифікацію досліджуваних методів, визначення їхніх особливостей, переваг та недоліків та, засновуючись на результатах, вибір найоптимальніших з них для інформаційної підтримки формування асортименту.

Дані представляють собою відгуки споживачів щодо продукції підприємства, дослідження методів їхнього збору та аналізу дозволить також запропонувати нові оптимальні підходи щодо налагодження зворотного зв’язку із аудиторією, що у свою чергу дасть ПрАТ «Оболонь» здійснювати зміни свого асортименту оперативним чином. Врахувавши специфіку даних та можливості сучасних технологій, було обрано для дослідження методи аналізу природної мови, оскільки вони якнайкраще підходять для роботи із текстами, написані людиною.

Метою проведеного дослідження, є визначення найактуальніших методів та підходів щодо збору та аналізу даних, які у подальших розділах пояснювальної записки будуть реалізовані в рамках додатку, доцільність створення якого було обґрунтовано засновуючись на результатах аналізу діяльності ПрАТ «Оболонь» стосовно інформаційної підтримки формування свого асортименту. Також дослідження методів збору та аналізу даних може дати змогу визначити основні тенденції щодо роботи підприємств із власною аудиторією з урахуванням використання сучасних технологій та платформ.

1.2. Огляд основних методів збору даних

Розглянемо методи збору даних, виконаємо їхній опис задля визначення їхніх переваг та недоліків. Провівши дослідження стосовно методів збору даних, наводимо наступну класифікацію:

1. Традиційні методи.
   - опитування – цей метод засновано на використанні створених підприємством анкет, щоб за їх допомогою отримати зворотний зв’язок від цільової аудиторії. Дозволяє отримати конкретну інформацію щодо продуктів або послуг [2];
   - фокус-групи – у цьому методі проводиться дослідження групи респондентів, які дають змогу глибоко аналізувати споживчі настрої та їхні очікування. Метод є ефективним, але потребує багато ресурсів [3];
   - спостереження та інтерв’ю – цей метод підходить для детального вивчення поведінки споживачів, але він також вимагає значних ресурсів [4].
2. Онлайн-опитування.
   - електронні анкети – відправлення онлайн-анкет для отримання зворотного зв’язку, що є більш економічним порівняно з традиційними методами. Однак для досягнення високої точності важливо налаштувати методологічні аспекти (вибірка, спосіб запрошення) [5].
3. Соціальні мережі.
   - API платформ соціальних медіа (зокрема, X, Instagram, Facebook) – соціальні мережі надають можливість безпосередньо аналізувати пости, коментарі та відгуки користувачів. Платформи особливо корисні завдяки великій кількості відкритих даних [6];
   - аналіз настроїв та відгуків – використовуючи спеціалізовані інструменти, наприклад бібліотека Tweepy мови Python для X,

можливо автоматизувати збір та обробку даних для дослідження настроїв споживачів [7].

4. Месенджери (зокрема, Telegram).
   - чат-боти – можуть виконувати як опитування користувачів, так і збір даних через інтерактивні повідомлення. Чат-боти можуть слугувати додатковим каналом для зворотного зв’язку, наприклад, у вигляді рейтингу або системи скарг [8];
   - моніторинг груп та каналів – у більшості месенджерів можливо відслідковувати публічні обговорення продукції у відкритих групах і каналах. За допомогою спеціальних ботів можливо автоматично аналізувати популярні питання, запити або коментарі на платформі [9].

Отже, відокремивши та описавши (див. таблицю 1.1) найбільш застосовувані методи збору відгуків, здійснимо їхнє порівняння та вибір найкращих з них.

Таблиця 1.1. Порівняння методів збору даних

№	Назва методу	Переваги	Недоліки
1	Опитування	Точна інформація, фокус на конкретних питаннях [2]	Велика витрата часу та ресурсів [2]
2	Фокус-групи	Глибокий аналіз настроїв, взаємодія з аудиторією [3]	Високі витрати, обмежена вибірка [3]
3	Спостереження	Детальний аналіз поведінки [4]	Висока трудомісткість, суб’єктивність [4]
4	Соціальні мережі	Швидкий доступ до великої кількості даних, автоматизація збору та аналізу	Обмеження API [6], необхідність спеціальних навичок, залежність від активності користувачів

Продовження таблиці 1.2. Порівняння методів збору даних

№	Назва методу	Переваги	Недоліки
5	Чат-боти	Швидкість взаємодії, легкість налаштування	Залежність від активності користувачів

Отже, з огляду на проведений порівняльний аналіз, найбільш оптимальним за швидкістю збору даних є методи пов’язані з чат-ботами та соцмережами. Дані методи було обрано, оскільки, критерій швидкості роботи є надзвичайно важливим, особливо в умовах сучасного стану інформаційних технологій. Більше того, недоліки розглянутих методів стосовно активності користувачів не є дуже вагомими, оскільки їх легко нівелювати за допомогою засобів щодо залучення аудиторії. Більше того, обмеження API для роботи із соціальними мережами не є критичними.

Розглянувши методи збору даних та обравши з них найактуальніші, переходимо до визначення способів та засобів обробки отриманої за їх допомогою інформації.

1.3. Огляд методів аналізу природньої мови

Аналіз природної мови (Natural Language Processing, NLP) є доцільним для реалізації підходом стосовно обробки текстових даних, що представлені коментарями і повідомленнями. Сучасні методи NLP можуть надати змогу підприємству визначити настрої аудиторії та потреби. У цьому розділі розглянемо основні методи NLP.

Методи NLP мають наступну класифікацію:

- аналіз тональності – цей метод дозволяє визначити емоційне забарвлення тексту, яке може бути позитивним, негативним або нейтральним. Аналіз тональності може бути корисним для оцінки загального ставлення споживачів до продукції [10];
- тематичне моделювання – цей метод використовується для автоматичного визначення основних тем або категорій у великому обсязі текстових даних. За допомогою цього методу можна

виокремити питання або аспекти продукції, які найчастіше обговорюються користувачами, та в подальшому адаптувати свої маркетингові або виробничі стратегії [10];

- видобування ключових слів та фраз – цей метод передбачає виділення основних ключових слів або фраз, які характеризують зміст тексту. Це може бути корисно для виділення основних аспектів, про які згадують споживачі у своїх відгуках [10];
- класифікація тексту – текстова класифікація дозволяє класифікувати текст відповідно до визначених класів, наприклад відповідно їхнім настроям. [11];
- розпізнавання іменованих сутностей – даний метод дозволяє автоматично виділяти важливі об’єкти в тексті, такі як назви продуктів, бренди. Метод може допомогти відслідковувати конкретні продукти, на які споживачі залишають відгуки [12].

Розібравши основні методи NLP, перейдемо до їхнього порівняльного аналізу.

Таблиця 1.3. Порівняння методів NLP

№	Метод	Переваги	Недоліки
1	Аналіз тональності	Оцінка настроїв споживачів, швидка реакція на зміни	Не завжди точний для складних текстів
2	Тематичне моделювання	Виявлення основних тем обговорень	Може потребувати ручного коригування тем
3	Видобування ключових слів	Швидке визначення основних аспектів продукції	Втрата контексту при аналізі ізольованих слів
4	Класифікація тексту	Структурована сегментація відгуків	Потребує значного обсягу даних для навчання моделей

Продовження таблиці 1.4. Порівняння методів NLP

№	Метод	Переваги	Недоліки
5	Розпізнавання сутностей	Виявлення ключових назв, брендів, місць у тексті	Може бути складним для аналізу неформального тексту

Використання методів обробки природньої мови для аналізу відгуків і коментарів є ефективним інструментом для того щоб зрозуміти настрої своїх споживачів. Їхнє застосування може дозволити підприємству оперативним чином реагувати на відгуки, визначати потреби та вподобання аудиторії. Зважаючи на результати порівняльного аналізу наведені у таблиці 1.2 обираємо підхід у вигляді використання методів аналізу тональності та класифікації тексту, як найбільш оптимальний. Даний вибір зумовлений оптимальним поєднанням швидкості виконання роботи та її якості. Також обраний підхід є оптимальним, оскільки за його допомогою можна вирішити завдання у вигляді класифікації тексту за настроєм з метою його подальшої обробки. Обравши види методів збору та аналізу даних, виконаємо дослідження ПрАТ «Оболонь» із міркувань визначення його специфіки та потреб як предметної області.

1.4. Характеристика діяльності ПрАТ «Оболонь»

Перш ніж перейти до постановки задач та огляд можливих способів їх реалізації, розглянемо діяльність підприємства ПрАТ «Оболонь» якого вони безпосередньо стосуються.

Приватне акціонерне підприємство «Оболонь» має досить великі у порівнянні із іншими підприємствами масштаби, займає вагому частину українського ринку напоїв та має у своєму складі, окрім головного заводу, розташованого за адресом Київська область, м. Київ, вул. Богатирська, 3, ще декілька заводів в областях України [1]. Логотип компанії має вигляд, наведений на рисунку 1.1.

Рисунок 1.1 – Логотип ПрАТ «Оболонь»

Підприємство спеціалізується у першу чергу на випуску широкого спектру напоїв, починаючи від пива та закінчуючи серіями мінеральних вод та лимонадів, які вже давно представлені на полицях вітчизняних магазинів.

Підприємство, очевидно, є серйозним гравцем на продовольчому ринку та відіграє важливу роль в економіці країни. ПрАТ «Оболонь» має довгу історію, яка починається ще з 80-го року минулого сторіччя [1] та протягом якої воно збільшувало об’єми виробництва продукції, формуючи при цьому власні традиції.

Також важливість ПрАТ «Оболонь» зумовлена його співпрацею із іншими виробниками напоїв, реалізацією продукції на експорт та навіть підтримкою вітчизняного спорту [1]. Не дивно, що компанія у 2020 році увійшла до топу інноваційних компаній України [1].

Отже, ПрАТ «Оболонь» є великим за обсягами та роллю у сфері виробництва напоїв підприємством, що визначає важливість його корпоративних даних, особливо стосовно відгуків споживачів щодо продукції підприємства.

Здійснимо постановку задачі стосовно методів збору та аналізу відгуків для інформаційної підтримки формування асортименту на ПрАТ «Оболонь».

1.5. Визначення задач щодо збору та аналізу відгуків споживачів продукції підприємства

Сформуємо список задач, які стосуються методів збору та аналізу відгуків для інформаційної підтримки формування асортименту на ПрАТ «Оболонь» із урахуванням специфіки отримуваних даних.

Список задач має наступний вигляд:

- виконання збору даних – дані потрібно зібрати, враховуючи їхню специфіку, зібрані дані мають представляти собою відгуки користувачів, назву продукції, якої вони стосуються та інформацію щодо дати їхнього написання;
- виконання класифікації даних – дані у вигляді відгуків потрібно обробити та класифікувати;
- виконання зберігання даних – отримані дані мають бути збережені відповідно до вимог, зокрема середовище для зберігання має відповідати вимогам ACID (atomicity, consistency, isolation, durability – атомарність, узгодженість, ізольованість, стійкість);
- виконання обробки даних – дані можуть бути додані, переглянуті та видалені, тобто користувач повинен мати можливість виконувати перелік базових операцій стосовно даних, у першу чергу тих, що стосуються продукції компанії;
- виконання побудови рейтингу – зібрані та оброблені дані слугують для побудови рейтингу продукції.

Наведені вище основні задачі повинні бути виконані у визначеній вище послідовності. Очевидно, що швидкість впровадження даних задач у практичну діяльність підприємства позитивно вплине на ефективність і прибутковість його роботи.

Найкращим способом забезпечення даних вимог є створення спеціалізованого програмного забезпечення.

Отже, сформулювавши перелік основних задач дослідження та обґрунтувавши основний підхід щодо їх вирішення, розглянемо список завдань, покладених на розв’язання програмним забезпеченням.

- збір даних з соцмережі X (колишній Twitter) – вибір цієї соцмережі зумовлено її перевагами у порівнянні з Facebook та її популярності у своєму сегменті;
- збір даних з чат-боту – застосування чат-ботів є популярним способом отримання даних, оскільки забезпечує легкість їх збору та обробки. Надходження даних забезпечується платформою на якій вони розташовані, наприклад, месенджер «Telegram», оскільки він є найпопулярнішим серед месенджерів;
- обробка та аналіз даних, їх подальша класифікація – отримані дані у вигляді відгуків щодо певних товарів з асортименту підприємства потрібно обробити для визначення їхнього емоційного забарвлення. Також потрібно реалізувати відсіювання відгуків, які не мають достатнього емоційного забарвлення;
- розподіл отриманих даних за категоріями (позитивні та негативні відгуки) – дане завдання, що стосується обробки даних, є ключовим, тому потребує окремої уваги;
- коректне зберігання даних у необхідному в контексті поставлених задач вигляді – після обробки даних, потрібно виконати процедури, пов’язані з їх збереженням. Тобто, потрібно розробити відповідні структури баз або сховищ даних для накопичення відгуків про продукцію підприємства та конкурентів;
- зберігання даних у базі даних для подальшого до них звернення – дані мають зберігатись у базі або сховищі даних, яке відповідає вимогам ACID;
- реалізація доступу до бази або сховища даних, що містить відгуки – потрібно забезпечити безперебійний доступ до даних;

- реалізація роботи чат-боту для збору даних – чат-бот має коректним чином збирати та зберігати дані;
- реалізація формування рейтингу продукції – на основі накопичених даних потрібно виконати їхню обробку таким чином, щоб сформувати рейтинги продукції;
- реалізація привабливого та зрозумілого для користувача інтерфейсу – оскільки запропоновано вирішити задачі щодо збору та аналізу даних за допомогою застосунку, то він повинен мати простий та зрозумілий інтерфейс, що відповідає стандарту GUI для ефективної роботи користувачів;
- реалізація навігації по даних – потрібно організувати сортування та пошук даних для підвищення ефективності опрацювання відгуків споживачів продукції.

Узагальнений алгоритм для збору та обробки відгуків, наведений на рисунку 1.2.

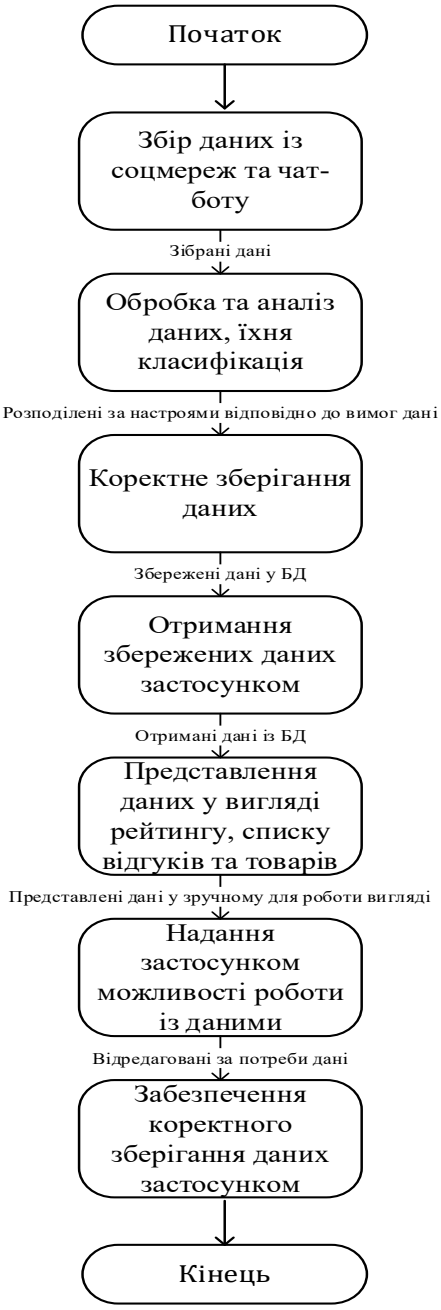


Рисунок 1.2 – Алгоритм роботи із даними, якій мають відповідати можливості майбутнього застосунку

1.6. Дослідження задач щодо інформаційної підтримки формування асортименту продукції ПрАТ «Оболонь»

На основі опису та алгоритму, наведених у пункті 1.5 здійснимо дослідження стану вирішення поставлених задач щодо обробки відгуків на продукцію компанії.

Спочатку дослідимо структуру ПрАТ «Оболонь» та визначимо, який відділ підприємства відповідає за формування рекомендацій щодо асортименту.

ПрАТ «Оболонь» має організаційну структуру, яка наведена на рисунку 1.3.

Рисунок 1.3 – Організаційна структура ПрАТ «Оболонь»

Підприємство складається із багатьох відділів, проте найбільшу потребу у інформаційній підтримці має планово-виробничий відділ.

Дослідження діяльності планово-виробничого відділу стосовно формування рекомендацій щодо асортименту надасть змогу зрозуміти які завдання потрібно вирішити для забезпечення ефективної роботи відділу.

Для дослідження діяльності планово-виробничого відділу підприємства, з використанням програмного забезпечення AllFusion Process Modeler (раніше відомого як BPWin) розроблено функціональну модель в нотації IDEF0. Вона описує процес збору відгуків та їх застосування у контексті формування рекомендацій щодо асортименту продукції ПрАТ «Оболонь». Функціональна модель відображає ключові процеси формування рекомендацій щодо формування асортименту продукції та наведена на рисунках 1.4-1.6.

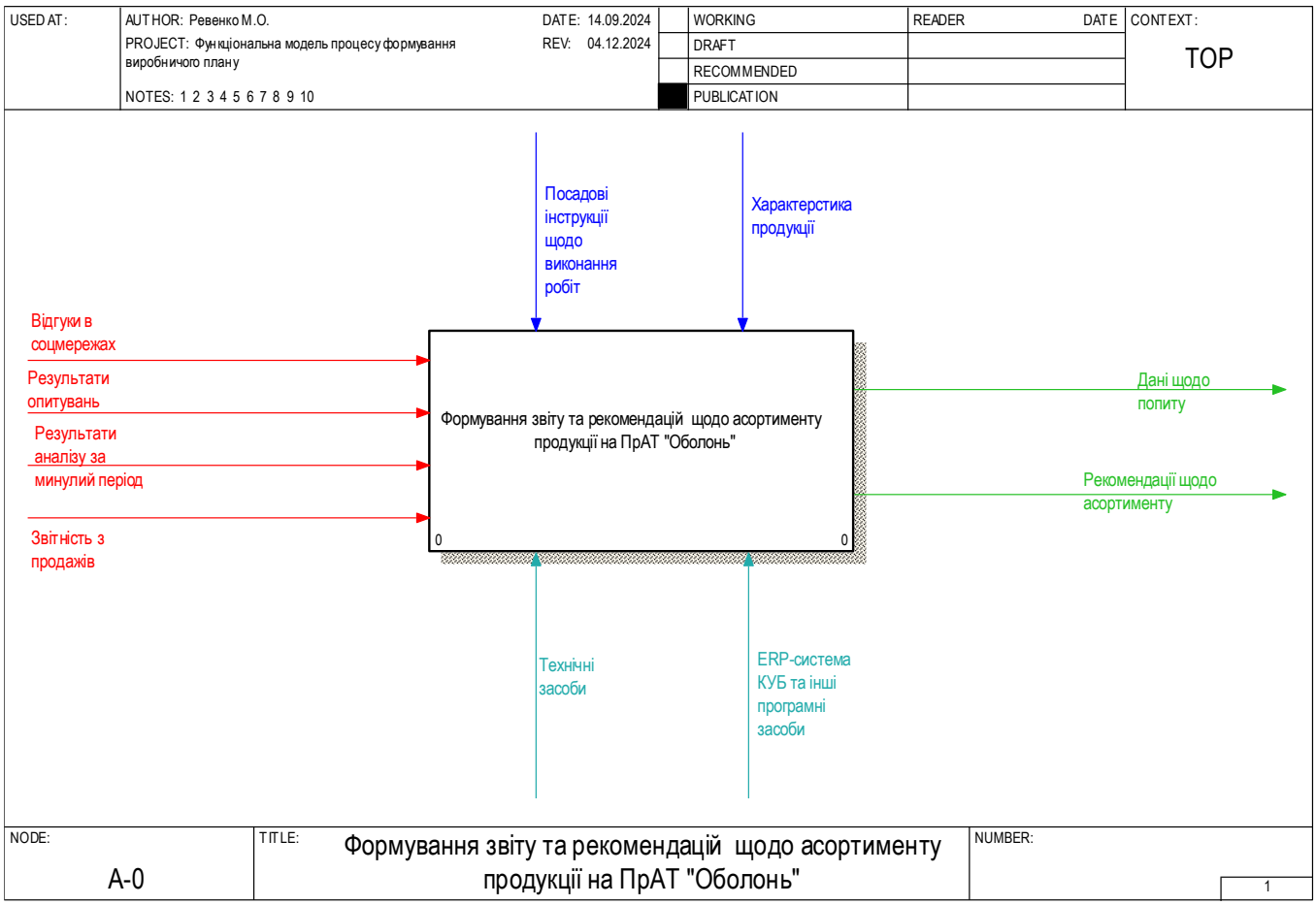


Рисунок 1.4 – Контекстна діаграма функціональної моделі процесу формування звіту та рекомендацій щодо асортименту продукції ПрАТ «Оболонь»

Перший рівень моделі дозволяє побачити дані, які представляють собою перелік відгуків із різних джерел, насамперед це відгуки із соцмереж та опитувань, які збираються компанією за допомогою офіційних сторінок у соцмережах чи шляхом проведення прямих опитувань споживачів. Процес збору даних регламентується посадовими інструкціями та містить характеристики зібраної продукції. При виконанні діяльності формування звіту та рекомендацій підприємство спирається на технічні та програмні засоби, проте вони в повній мірі не можуть забезпечити даний процес.

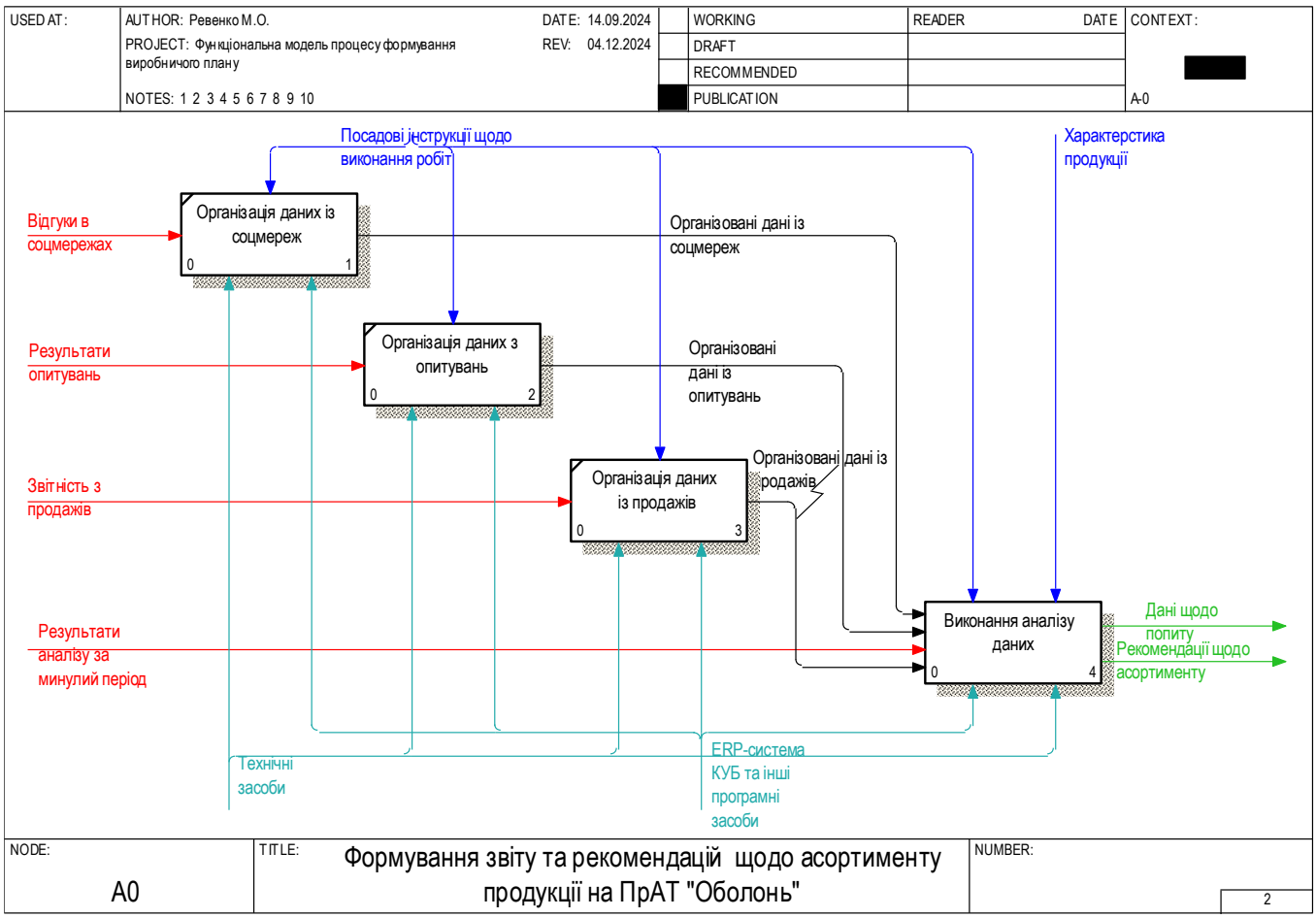


Рисунок 1.5 – Перший рівень декомпозиції - процес «Формування звіту та рекомендацій щодо асортименту продукції на ПрАТ «Оболонь»

Як видно із рисунку 1.5, здійснюється початкова обробка та організація отриманих даних для їхнього упорядкування, підготовки та обробки можливих в них помилок.

Після перевірки, дані піддаються аналізу, цей процес наведено на рисунку 1.6.

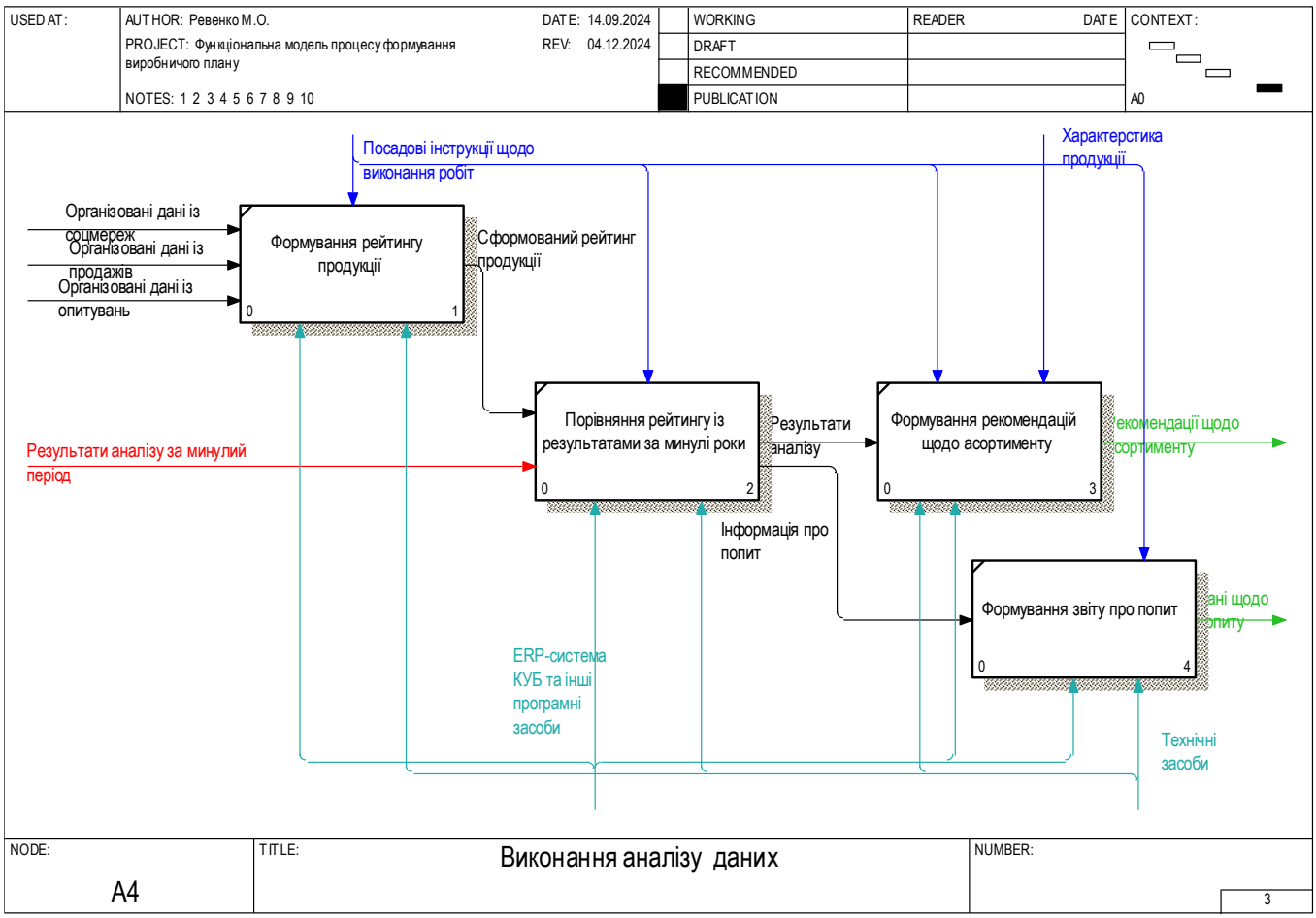


Рисунок 1.6 – Декомпозиція процесу «Виконання аналізу даних»

Згідно рисунку 1.6 спочатку формується рейтинг, який порівнюється із попередніми і вже після цього підприємство формує звіт та рекомендації.

Зважаючи на наведену на рисунках 1.4-1.6 функціональну модель, автоматизація формування рейтингу продукції є ключовою задачею. Практично повсякчасно підприємство опирається на програмні засоби, що представляють собою інструменти, які надають соцмережі та ERP-система КУБ, яка виконує зберігання даних.

При подальшому дослідженні діяльності підприємства було з’ясовано, що вона в основному опирається на взаємодію із великою за обсягами та складною за структурою базою даних ERP-системи КУБ. І хоча підприємство й веде активну модернізацію своїх процесів, система справляється із поточними задачами щодо організації виробництва, продажів продукції тощо. Але задачі щодо аналізу даних

доцільно вирішувати із застосуванням мовних моделей обробки тональності тексту на основі зібраних відгуків з їх подальшою класифікацією. Наявна на підприємстві ERP-система КУБ є надійним інструментом підтримки його поточної діяльності, реалізує частину аналітичних задач, але впровадження у її структуру мовних моделей є досить складним і довготривалим процесом на відміну від розробки додаткового програмного застосунку.

1.7. Висновки до першого розділу

Отже, ПрАТ «Оболонь» є, провідним українським харчовим підприємством з багатогранною історією, спеціалізується на виробництві напоїв та пов’язаної із ними продукції і протягом десятиліть займає суттєву частину цього ринку, відіграє важливу соціальну роль, зокрема за рахунок підтримки вітчизняного спорту, має стійкі традиції.

ПрАТ «Оболонь» використовує потужну ERP-систему КУБ, сфера використання якої найрізноманітніша, починаючи від організації виробництва та закінчуючи обробкою вакансій. Підприємство регулярно здійснює модернізацію своєї діяльності. Усе це призводить до того, що ПрАТ «Оболонь» є потужним гравцем на ринку продажів алкогольної та безалкогольної продукції.

Процес збору відгуків також виконується не без взаємодії із вже згаданою ERP-системою. Проте в умовах сучасності збір, організація, аналіз та зберігання відгуків споживачів щодо продукції підприємства краще всього реалізувати шляхом застосування чат-ботів, автоматичного парсингу соціальних мереж та методів, які здійснюють класифікацію тексту визначаючи його тональність з використанням відповідного програмного забезпечення. Усе попередньо сказане ERP-система КУБ здійснити не може, а її модернізація є занадто дорогою задачею у порівнянні із створенням окремого програмного забезпечення.

РОЗДІЛ 2. ОГЛЯД МЕТОДІВ АНАЛІЗУ ДАНИХ

2.1. Вибір методу класифікації тексту

Для вирішення завдань щодо аналізу даних розглянемо підхід, який базується на визначені тональності тексту для подальшої його класифікації.

Опишемо методи, які стосуються обраного підходу, визначимо їхні переваги та недоліки, виберемо найоптимальніший з них.

Класифікація методів аналізу тональності загалом має наступний вигляд:

1. Методи засновані на словниках – дані методи засновані на використанні словників, які вже попередньо мають оцінку тональності (позитивна, нейтральна та негативна), а тональність тексту вже у свою чергу визначається як їхня сума [13].

$$\text{Оцінка тональності} = \sum_{i=1}^{n} S(\omega_i), \tag{2.1}$$

де $S(\omega_i)$ – оцінка тональності слова ω_i, а n – кількість слів.

Також, можливе використання додаткових обробок, наприклад модифікація оцінок для заперечень або інтенсивних слів, які можуть застосовуватись у вигляді коригувальних коефіцієнтів;

2. Традиційні машинні методи – дані методи застосовують статистичні моделі, які навчаються на наданих даних для класифікації тексту. Дані методи включають наступні моделі та техніки:
   - метод наївного Баєса – заснований на припущені незалежності між словами та обчислені ймовірності для кожного класу [14]. Можна представити наступним чином:

$$P(Class|Text) \propto \frac{P(Text|Class) * P(Class)}{P(Text)}, \quad (2.2)$$

де P(Text|Class) – ймовірність тексту для даного класу,

P(Class) – апостеріорна ймовірність класу;

– методи опорних векторів (SVM) – ці методи розділяють класи гіперплощиною, оптимізуючи таким чином відстань між ними. Можна представити наступним чином:

$$H(x) = \omega^T x + b, \quad (2.3)$$

де ω – вектор ваг, x – вектор ознак тексту, b – зміщення [15];

– логістична регресія – базується на використанні сигмоїдної функції для прогнозу ймовірностей [16]:

$$P(Class|Text) = \frac{1}{1 + e^{-\omega^T x}}; \quad (2.4)$$

3. Методи засновані на застосуванні глибоких нейронних мереж – ці методи, як видно із назви, виконують класифікацію тексту за використанням нейронних мереж. Їх у свою чергу можна поділити на наступні методи:

– словникові вектори – представляють слова у вигляді багатовимірних векторів, які враховують їхнє семантичне оточення. Математична модель має наступний вигляд:

$$V(\omega) = f(\omega, context), \quad (2.5)$$

де $V(\omega)$ – вектор слова ω, а context – векторний контекст слова.

– рекурентні нейронні мережі (RNN), зокрема LSTM – здійснюють обробку тексту послідовно, запам'ятовуючи попередні слова через застосування спеціальних осередків пам'яті. LSTM (Long short-term memory) – є системою глибинного навчання, при реалізації якої вдалося обійти проблему зникнення або зашкалювання градієнтів у процесі навчання методом зворотного поширення помилки [17].

LSTM можна представити наступними формулами:

$$f_t = \sigma\left(W_f * [h_{t-1}, x_t] + b_f\right), \tag{2.6}$$

$$i_t = \sigma(W_i * [h_{i-1}, x_t] + b_i), \tag{2.7}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W_c * [h_{i-1}, x_t] + b_C), \tag{2.8}$$

де f_t – забуття, i_t – входи, C_t – оновлений стан;

– трансформери (BERT, GPT) – засновані на використанні механізму самоуваги для паралельної обробки слів у тексті [10, 18]. Їхня ключова формула має наступний вигляд:

$$Attetion(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.9}$$

де Q, K, V – матриці запитів ключів і значень, d_k – розмірність простору;

4. Гібридні методи – у своїй роботі поєднують декілька підходів, прикладом може бути використання словникових оцінок при машинному навчанні [19].

Визначимо почергово для наведених методів їхні переваги та недоліки згрупувавши їх у таблиці 2.1.

Таблиця 2.1. Порівняння методів класифікації тексту

№	Назва	Переваги	Недоліки
1	Методи засновані на словниках	Простота реалізації. Висока інтерпретованість результатів. [13]	Ігнорування контексту та послідовності слів. Неможливість працювати з новими словами або неформальними виразами. [13]
2	Традиційні машинні методи	Простота впровадження та хороша продуктивність на структурованих даних. [14] Підходять для середніх розмірів наборів даних.	Не враховують порядок слів. Залежність від інженерії ознак.
3	Глибокі нейронні мережі	Здатність уловлювати глибокі контекстуальні зв’язки. Висока точність на великих наборах даних.	Велика обчислювальна складність та, як результат, високі вимоги до апаратного забезпечення.
4	Гібридні методи	Можливість поєднання інтерпретованості та точності. Краще врахування специфічних мовних особливостей. [19]	Ускладнення впровадження. Вимога великих обчислювальних ресурсів. [19]

Отже, зважаючи на проведений аналіз та, зокрема, на останні тенденції на ринку IT-технологій, для виконання класифікації тексту краще всього обрати моделі засновані на глибоких нейронних мережах, а говорячи конкретно – трансформери. Цей вибір зумовлено тим, що моделі вибраного виду здатні до паралелізму в обчисленнях завдяки механізму самоуваги та є універсальними, тобто мають здатність до перекладу, генерації та, що важливо у нашому випадку, аналізу тональності та класифікації тексту. Більше того, обраний механізм

зарекомендував себе гарним чином, оскільки лежить в основі багатьох моделей, наприклад GPT та BERT [18].

2.2. Метод аналізу тональності та класифікація тексту на основі використанні моделі трансформерів

На основі наведеного вище дослідження для аналізу тональності та класифікації тексту обрано метод заснований на використанні трансформерів. Перш ніж перейти до безпосередньої розробки опишемо вибраний підхід детальніше.

Основний принцип трансформерів – це механізм самоуваги, зокрема його покращена версія – мультиголова самоувага. Розглянемо основні особливості моделі детальніше:

1. Самоувага – даний механізм призначений для визначення важливості інших слів для кожного почергово. Це дозволяє моделі враховувати залежності між словами незалежно від їхнього положення [19].

$$Attetion(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.10)$$

де Q, K, V – матриці запитів ключів і значень, d_k – розмірність простору.

Тобто, кожне слово представляється у вигляді трьох векторів – запитів (Q), ключів (K) та значень (V);

2. Мультиголова самоувага – є розширеним механізмом самоуваги, в якому декілька «голів» паралельним чином виконують обчислення значень уваги, що дозволяє моделі виявляти одночасно різні типи залежності між словами. Формули аналогічні, але обчислення виконується для кожної «голови» із подальшим об’єднанням результатів [20]:

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)W^o; \quad (2.11)$$

3. Архітектура трансформерів – архітектура містить послідовність шарів уваги та повнозв'язних шарів. Модель використовує позиційні закодування для врахування порядку слів у тексті. Формула позиційних закодувань має вигляд:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right); \quad (2.12)$$

4. Функція втрат – для задач класифікації, таких як класифікація тональності, найчастіше використовують крос-ентропію:

$$Loss = -\sum_i y_i \log \hat{y}_i, \quad (2.13)$$

де y_i – істинна ймовірність, а $\hat{y}_i$ – передбачена ймовірність.

2.3. Висновки до другого розділу

Отже, провівши дослідження методів призначених для аналізу тональності тексту та його класифікації, було обрано як найбільш доцільні до застосування методи засновані на використанні глибоких нейронних мереж та модель типу трансформерів.

Моделі типу трансформерів є досить складними за принципами своєї роботи, проте здатні забезпечити необхідну точність в аналізі тональності.

Досліджена модель покладена в основу розробки програмного застосунку, описаного в четвертому розділі, оскільки вона може забезпечити гарне поєднання швидкості роботи та її ефективності, яка вже була доведена відомими моделями типу BERT та GPT [21]. Моделі типу трансформерів є сучасним та потужним інструментом. Обрана модель реалізована у поєднанні із інструментом забезпечення низькорівневої інфраструктури для створення, навчання та оптимізації, засновуючись на операціях із тензорами – базовими математичними об'єктами для обчислень у нейромереж [22].

РОЗДІЛ 3. ОГЛЯД ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ ДЛЯ ВИРІШЕННЯ ЗАДАЧ

3.1. Обґрунтування вибору засобів реалізації додатку

Головним засобом реалізації додатку є мова програмування Python. Вибір цієї мови обумовлено її актуальністю та спектром інструментів, що мають стійку підтримку та популярність серед розробників. Очевидно, що дані плюси зумовлені, окрім популярності, ще й відносною простотою освоєння у порівнянні з іншими мовами програмування, наприклад, Python є мовою із динамічною типізацією на відміну від C++, що полегшує розробку програмного забезпечення аналогічного за масштабами до наведеного у даному розділі прикладу. Також Python містить вбудовані функції, які ще більш спрощують розробку.

Мова програмування Python має підтримку не тільки безпосередніх розробників а від спільноти програмістів, що призводить до активної розробки бібліотек та фреймворків. Найбільш відомими бібліотеками є Pandas, Matplotlib, Numpy, Sqlite, Seaborn та інші. Деякі з них, наприклад Pandas та Numpy використовуються при наукових дослідженнях, які потребують високої точності [23]. Слід також згадати про важливість мови у сфері нейронних мереж та Data Science, оскільки середовище Python надає можливість розробникам створювати та навчати нейронні мережі використовуючи для цього відповідні бібліотеки, зокрема SciPy, TensorFlow, Scikit-Learn [24].

Стосовно бібліотек, у даній роботі використано наступні:

Sqlite3, Datetime, Telegram, Transformers, Torch, Pandas, Tkinter, Tkcalendar, Matplotlib.

Рисунок 3.1 – Логотип обраної мови програмування для створення додатку

Також, важливо звернути увагу на інтегроване середовище розробки (далі IDE – Integrated Development Environment), яке використовувалось для розробки програмного забезпечення. Даним IDE є PyCharm Community Edition від компанії JetBrains, вибір якого обумовлено його інструментарієм, ефективністю та наявним досвідом роботи у ньому. Також PyCharm є підтримуваним середовищем розробки, що наприклад означає можливість встановлення плагінів на нього для підвищення ефективності роботи. Логотип та приклад середовища обраного IDE наведено на рисунках 3.2-3.3.

Рисунок 3.2 – Логотип обраного для практичної частини IDE

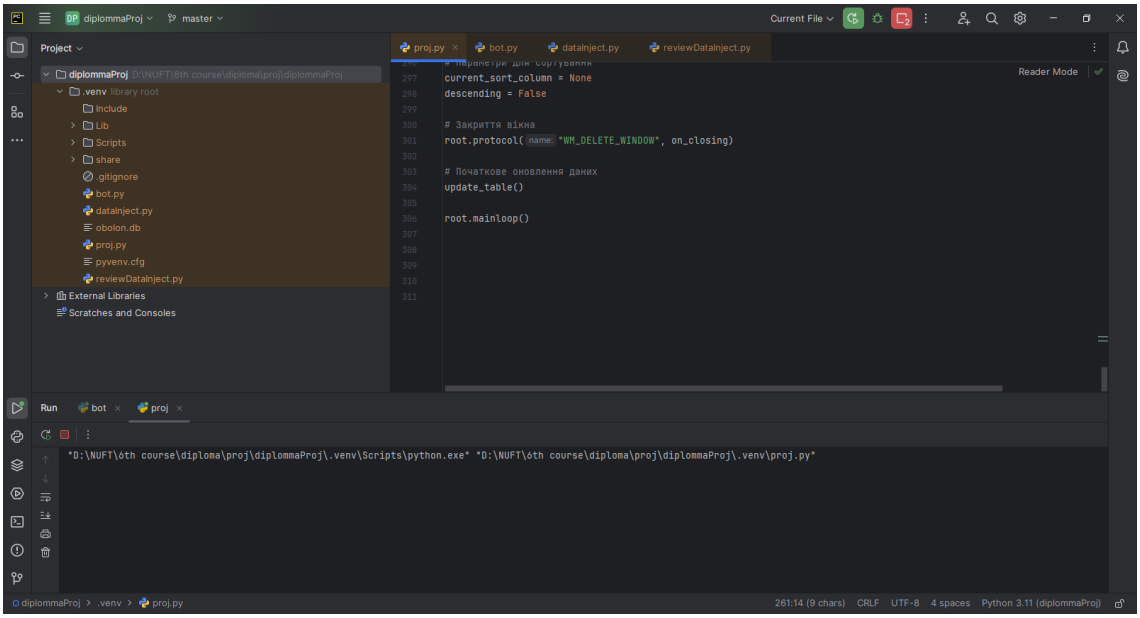


Рисунок 3.3 – Приклад інтерфейсу IDE PyCharm

Отже, мова програмування Python є потужним та водночас легким у засвоєнні інструментом, саме тому він є основним засобом розробки застосунку, розробку якого у подальшому буде описано в даному розділі.

3.2. Способи реалізації додатку

Розглянемо, основні бібліотеки мови Python для реалізації функціоналу додатку.

Для реалізації інтерфейсу користувача:

- tkinter – вікно керування застосунком, кнопки для виконання задач;
- tkcalendar – меню вибору періоду для аналізу даних;
- matplotlib – побудова графіків.

Для збору даних:

- telegram – бібліотека для реалізації функціоналу чат-боту у telegram;
- tweepy – для парсингу соціальних мереж;
- pandas – бібліотека для роботи із даними у вигляді набору.

Для аналізу даних:

- torch – низькорівнева інфраструктура для створення, навчання та оптимізації нейронних мереж;
- transformers – бібліотека для нейронної мережі для класифікації тексту.

Для роботи з даними:

- sqlite3 – бібліотека для зберігання даних та роботи з ними через запити мовою SQL. Обрання SQL як мови запитів до БД зумовлено, окрім досвіду роботи, ще й її відповідності стандартам ACID, що є дуже важливим при роботі із великими об’ємами даних.

Для реалізації чат-боту:

- telegram – бібліотека для реалізації функціоналу чат-боту у telegram.

Для внутрішніх обчислень:

- datetime – реалізація вибору даних за період.

Отже, визначивши способи вирішення основних завдань, покладених на додаток, можна перейти до безпосередньої його розробки.

3.3. Висновки до третього розділу

Отже, головним засобом для створення додатку для виконання завдань стосовно збору та аналізу даних є мова програмування Python. Обрана мова програмування є потужним та водночас легким в засвоєнні інструментом, сфера його застосувань найрізноманітніша. Для роботи із обраним засобом використано IDE PyCharm Community Edition від компанії JetBrains, в якому наявно усе необхідне для цього. Обрана мова програмування є популярним засобом, має велику кількість різних бібліотек, з яких було обрано для створення застосунку наступні: Sqlite3, Datetime, Telegram, Transformers, Torch, Pandas, Tkinter, Tkcalendar, Matplotlib.

Обрані засоби мають стійку підтримку, користуються попитом та є актуальними, що доводить їхню придатність до виконання задач стосовно розробки додатку.

РОЗДІЛ 4. ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБОРУ ТА АНАЛІЗУ ВІДГУКІВ

4.1. Опис головних моментів реалізації програмного забезпечення

Переходимо до огляду застосування обраних засобів для створення додатку, акцентуючи увагу на основних моментах.

Повний текст програми для збору та аналізу відгуків наведено у додатку Б даної роботи.

Першим кроком є проектування та підготовка бази даних проекту, яка буде містити у собі відгуки відповідно до їх тональності та у необхідному для проекту форматі.

Створення БД реалізовано шляхом написання скрипту мовою Python, частина якого із коментарями наведено далі.

Вхідні дані для перевірки роботи БД.

```
obolon_products = ["Оболонь Світле", "Оболонь Преміум", "Оболонь Біле", "Живчик", "Оболонь Безалкогольне", "Оболонь Квас", "Оболонь Чернігівське", "Оболонь Міллер","Оболонь Гіннес"]
```

Алгоритм функції «create_products_table_and_insert_data» можна пояснити наступним чином – спочатку підключаємося до БД проекту, створюємо таблицю для товарів та наповнюємо її вхідними даними (дані занесено з міркувань проведення тестування). Створення таблиці та додавання до неї даних реалізовано через застосування курсору та запиту у ньому (змінна «cursor»), зв’язок із базою даних через змінну «conn».

```
def create_products_table_and_insert_data():
    conn = sqlite3.connect('obolon.db')
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS Products (
```

```
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                product_name TEXT UNIQUE NOT NULL
            )
        """)
        cursor.executemany("INSERT OR IGNORE INTO Products (product_name)
VALUES (?)",
                           [(product,) for product in obolon_products])
        conn.commit()
        conn.close()
        print("Таблиця Products створена, і дані додані.")
```

Далі, аналогічно розроблено таблицю для відгуків із зв’язком до таблиці товарів, в даному випадку зв’язок можна описати як один товар – багато відгуків. Можна зауважити, що окрім мови програмування Python, лістинг містить запити мовою SQL. Повний код для даного скрипту наведено у Додатку Б даної роботи під пунктом «dataInject.py», що додатково місить функції для перевірки роботи БД.

```
    def create_reviews_table():
        conn = sqlite3.connect('obolon.db')
        cursor = conn.cursor()
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS Reviews (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                product_id INTEGER,
                review_text TEXT,
                sentiment TEXT,
                review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                FOREIGN KEY (product_id) REFERENCES Products (id)
            )
        """)
        conn.commit()
        conn.close()
```

```
    print("Таблиця Reviews створена.")
```

Отже, розглянувши скрипт для створення БД, розглянемо далі реалізацію функціоналу чат-боту аналогічним чином звертаючи увагу на ключові моменти.

Наведемо бібліотеки, які безпосередньо використано для створення боту та для виконання класифікації тексту.

```
from telegram import Update, ReplyKeyboardMarkup
from telegram.ext import Application, CommandHandler, MessageHandler, filters, ContextTypes, ConversationHandler
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
```

Із міркувань безпеки, токен боту приховано. Він представляє собою набір символів, який отримуються при розробці боту.

```
BOT_TOKEN = ''
```

Завантажуємо модель для аналізу тональності. Завантажена модель підтримує інші окрім української мови, тобто відгуки також можна залишати на інших мовах, а це у свою чергу розширює коло можливих опитуваних персон.

```
print("Завантаження моделі для аналізу тональності...")
tokenizer = AutoTokenizer.from_pretrained("nlptown/bert-base-multilingual-uncased-sentiment")
model = AutoModelForSequenceClassification.from_pretrained("nlptown/bert-base-multilingual-uncased-sentiment")
print("Модель завантажена.")
```

Ініціюємо базу даних та створюємо таблиці на випадок їх відсутності, тобто якщо користувач не матиме доступу до таблиць, бот все рівно зможе зберігати дані.

```
def init_db():
    conn = sqlite3.connect('obolon.db')
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS Products (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            product_name TEXT UNIQUE NOT NULL
        )
    """)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS Reviews (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            product_id INTEGER,
            review_text TEXT,
            sentiment TEXT,
            review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (product_id) REFERENCES Products (id)
        )
    """)
    conn.commit()
    conn.close()
```

Отримання даних про продукцію, та занесення їх у змінну «products» для подальшої роботи.

```
def get_product_list():
    conn = sqlite3.connect('obolon.db')
    cursor = conn.cursor()
    cursor.execute("SELECT product_name FROM Products")
```

```
        products = [row[0] for row in cursor.fetchall()]
        conn.close()
        return products
```

Розглянемо одну із найголовніших функцій – *функцію для аналізу тональності тексту.* Спочатку виконуємо токенізацію тексту, для його перетворення у зрозумілий для моделі формат, далі підготовлені дані передаються в модель, щоб отримати результати у вигляді необроблених чисел для кожного класу. Потім рядок із присвоєнням значення змінній «scores» значень логітів, що відображають ймовірності класів до нормалізації. Застосовуємо далі функцію «softmax» до логітів, перетворюючи їх у ймовірності для кожного класу. Потім знаходимо індекс класу з максимальною ймовірністю («argmax») і додаємо 1, щоб отримати рейтинг від 1 до 5. Та наостанок виконується класифікація за рейтингом, якщо рейтинг більше або дорівнює 4 то відгук позитивний, якщо більше або дорівнює 2 – негативний, інакше – нейтральний. Для наглядності програмний код представлено у вигляді блок-схеми в додатку В.

```
    def analyze_sentiment(review_text):
        inputs = tokenizer.encode_plus(review_text, return_tensors="pt",
truncation=True)
        with torch.no_grad():
            outputs = model(**inputs)
        scores = outputs.logits
        predictions = torch.softmax(scores, dim=1)
        rating = torch.argmax(predictions, dim=1).item() + 1
        if rating >= 4:
            return "positive"
        elif rating <= 2:
            return "negative"
        else:
            return "neutral"
```

Обробка виконання команди «/start» при роботі із чат-ботом. При її натисканні отримується список продуктів у вигляді змінної «products», якщо продуктів немає в БД, то виводиться відповідне повідомлення.

```
    async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -
> int:
        products = get_product_list()
        if not products:
            await update.message.reply_text("Немає доступних продуктів
для відгуків.")
            return ConversationHandler.END
        reply_keyboard = [products]
        await update.message.reply_text(
            "Вітаємо! Оберіть продукт, для якого ви хочете залишити
відгук.",
            reply_markup=ReplyKeyboardMarkup(reply_keyboard,
one_time_keyboard=True)
        )
        return SELECT_PRODUCT
```

Реалізація обробки вибору продукції, обробка випадку її відсутності.

```
    async def select_product(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        product_name = update.message.text
        products = get_product_list()
        if product_name not in products:
            await update.message.reply_text("Невірний вибір продукту.
Будь ласка, оберіть знову.")
            return SELECT_PRODUCT
        context.user_data['product_name'] = product_name
        await update.message.reply_text("Напишіть ваш відгук про цей
продукт.")
```

```
        return WRITE_REVIEW
```

Далі поетапно опишемо функцію «write_review». Спочатку отримуємо дані відгуку та продукту, потім передаємо відгук у функцію аналізу тональності, далі підключаємося до БД і заносимо відгук, маючи код товару та тональність. Алгоритм функції «write_review» наведено блок-схемою у Додатку В.

```
    async def write_review(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        review_text = update.message.text
        product_name = context.user_data['product_name']
        sentiment = analyze_sentiment(review_text)
        conn = sqlite3.connect('obolon.db')
        cursor = conn.cursor()
        cursor.execute("SELECT id FROM Products WHERE product_name = ?",
(product_name,))
        product_id = cursor.fetchone()[0]
        cursor.execute("""
            INSERT INTO Reviews (product_id, review_text, sentiment)
            VALUES (?, ?, ?)
        """, (product_id, review_text, sentiment))
        conn.commit()
        conn.close()
```

Надання користувачу відповіді на отриманий відгук, яка залежить від його настрою.

```
        if sentiment == "positive":
            sentiment_text = "Дякуємо за позитивний відгук!"
        elif sentiment == "negative":
            sentiment_text = "Дякуємо за ваш відгук. Нам прикро, що ви не
задоволені."
        else:
```

```
            sentiment_text = "Дякуємо за ваш відгук!"
        await update.message.reply_text(sentiment_text)
        return ConversationHandler.END
```

Обробка відміни операції.

```
    async def cancel(update: Update, context: ContextTypes.DEFAULT_TYPE)
-> int:
        await update.message.reply_text("Відгук скасовано.")
        return ConversationHandler.END
```

Важливим буде зауваження, що даний код виконує тільки обробку команд чат-боту але не створює його, доказом цього є використання токену. Тобто необхідно створити чат-бот через @BotFather та отримати у ньому токен. Робота боту буде показана далі протягом пояснювальної записки у підрозділі, присвяченому демонстрації роботи проекту та у додатку А.

Опишемо реалізацію процесу отримання даних із соцмереж. Спочатку необхідно пройти реєстрацію порталі розробників соцмережі «X», який знаходиться за наступним посиланням: «https://developer.x.com/en».

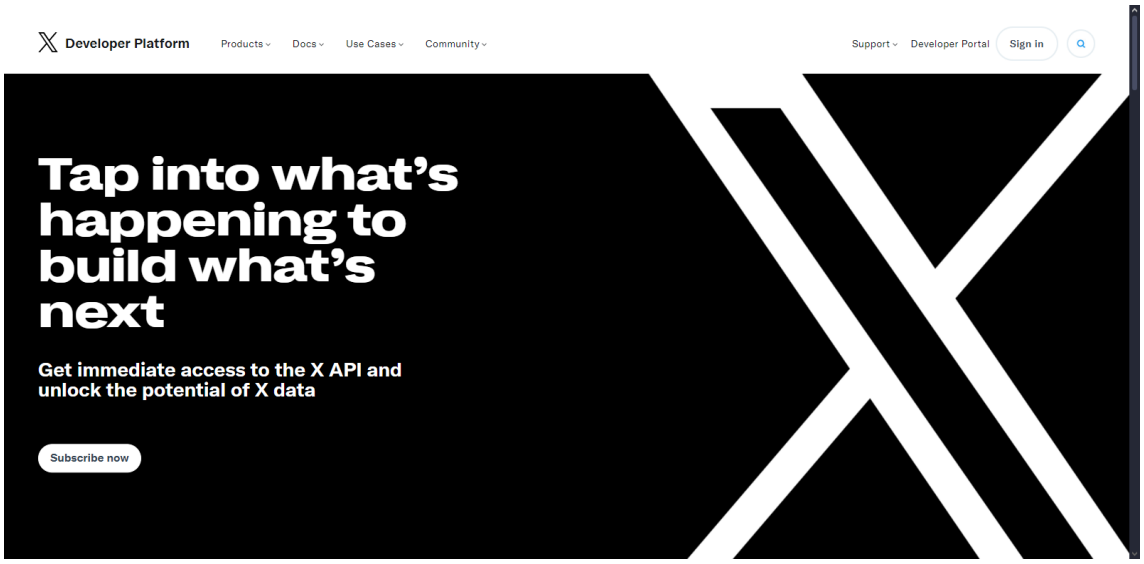


Рисунок 4.1 – Головна сторінка порталу

Далі створимо проект.

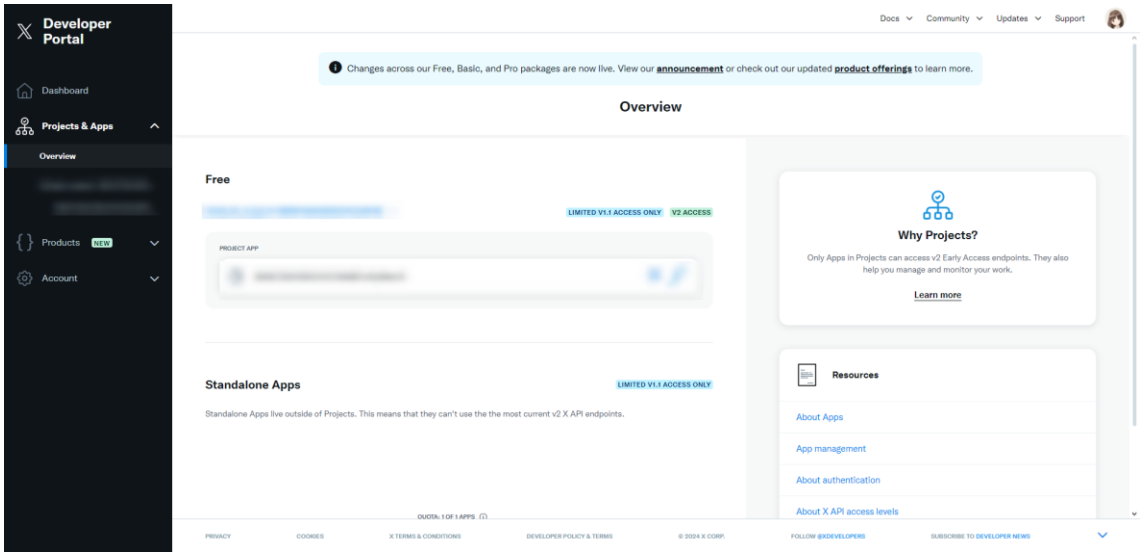


Рисунок 4.2 – Сторінка проектів на платформі

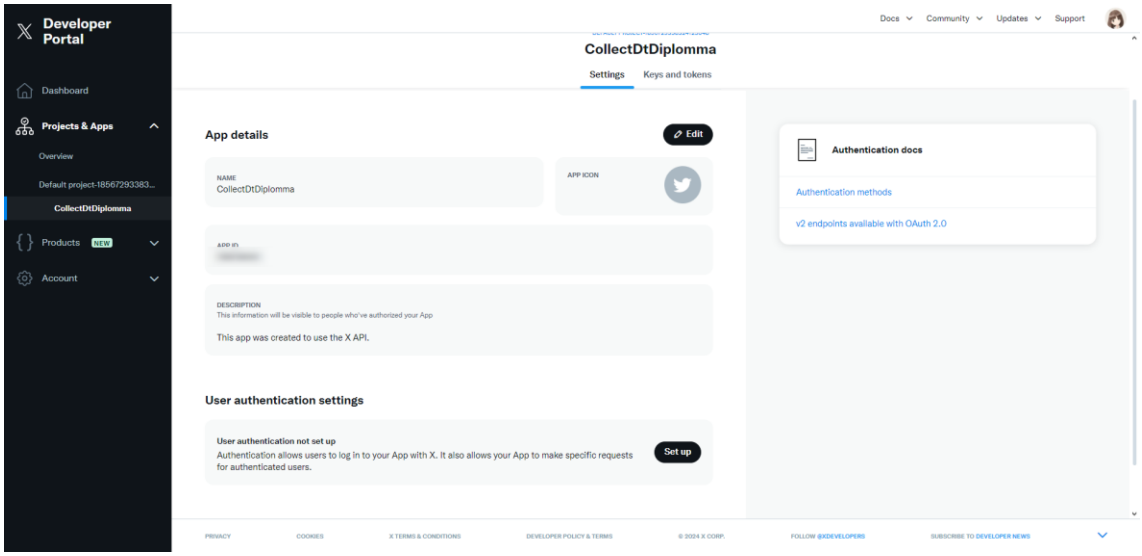


Рисунок 4.3 – Розроблений проект для реалізації збору даних із соцмереж

Отримавши потрібний токен, поетапно розглянемо основні частини програмного код для збору даних застосунком. Розглянемо тільки ту частину коду, яка призначена для збору даних, функції для ініціалізації БД та визначення тональності тексту аналогічні до тих, що наведені в коді реалізації роботи чат-боту.

Програмний код проекту наведено у Додатку Б.

Виконуємо імпорт потрібних бібліотек, зокрема «tweepy» для роботи із обраною соцмережею.

```
import tweepy
from transformers import AutoModelForSequenceClassification, AutoTokenizer
import torch
import sqlite3
from datetime import datetime
```

Наведений код не містить потрібного для роботи застосунку токену (змінна «BEARER_TOKEN») із аналогічних до попередньо розглянутого скрипту міркувань.

```
BEARER_TOKEN = ""
client = tweepy.Client(bearer_token=BEARER_TOKEN)
```

Далі код реалізації збору даних за хештегом, хештег передається в запит, який у свою чергу слугує для отримання списку твітів у змінну «tweets». Потім зберігаємо відгук у БД. Далі з міркувань перевірки роботи виводимо у консолі отримані дані та їхню тональність. Алгоритм отримання даних із соцмереж наведено у вигляді блок-схеми у додатку В.

```
def get_tweets_by_hashtag(hashtag, max_results=10):
    query = f"{hashtag} lang:uk"
    tweets = client.search_recent_tweets(query=query, max_results=max_results, tweet_fields=["created_at"])
    for tweet in tweets.data:
        text = tweet.text
        date_posted = tweet.created_at.strftime("%Y-%m-%d %H:%M:%S")
        sentiment = analyze_sentiment(text)
        save_review_to_db(text, sentiment, date_posted)
        print(f"Твит: {text}\nТональность: {sentiment}\nДата: {date_posted}\n")
```

Нажаль поки що не існує постів за хештегом «#оболоньвідгук», тому спробуємо проаналізувати твіти пов’язані з недавніми виборами в США за хештегом «#Election2024».

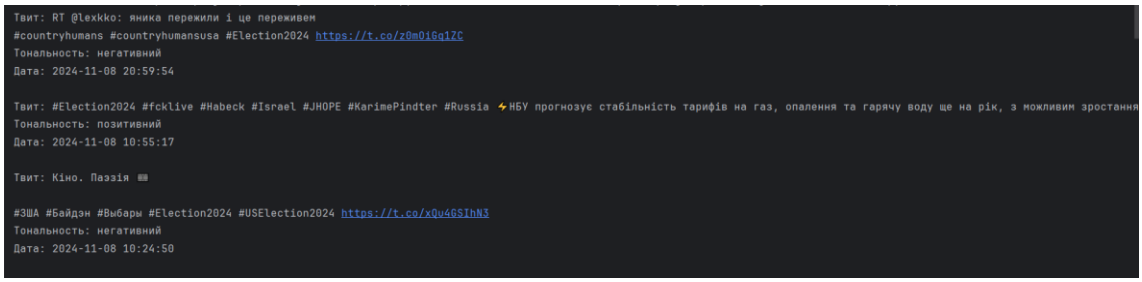


Рисунок 4.4 – Результати виконання

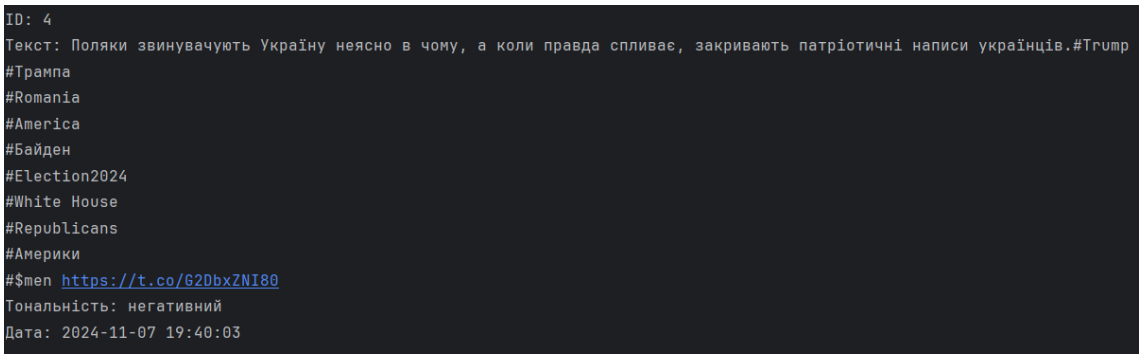


Рисунок 4.5 – Виведення отриманих даних

Нарешті розглянемо та прокоментуємо частини коду реалізації головної програми (повний код наведено у додатку Б).

Список бібліотек має наступний вигляд.

```
import pandas as pd
import tkinter as tk
import tkinter.filedialog as filedialog
from tkinter import ttk, messagebox
from tkcalendar import DateEntry
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
```

Бібліотека «pandas» слугує для роботи із даними, отриманими з БД, «tkinter» та її компоненти для інтерфейсу програми взагалі, «matplotlib» для побудови графіків.

Функція «get_data_from_db» слугує для отримання даних та формування на їхній основі рейтингу, який повертається як вікно даних. Також реалізовано обрання даних за певним періодом.

```
    def get_data_from_db(start_date, end_date):
        conn = sqlite3.connect('obolon.db')
        query = """
            SELECT P.product_name,
                   SUM(CASE WHEN R.sentiment = 'positive' THEN 1 ELSE 0
END) AS positive_reviews,
                   SUM(CASE WHEN R.sentiment = 'negative' THEN 1 ELSE 0
END) AS negative_reviews
            FROM Products P
            JOIN Reviews R ON P.id = R.product_id
            WHERE R.review_date BETWEEN ? AND ?
            GROUP BY P.product_name
        """
        df    =    pd.read_sql_query(query,    conn,    params=(start_date,
end_date))
        conn.close()
```

Виконується обчислення ретингу у результуючій таблиці наступним чином.

```
        df['rating'] = df['positive_reviews'] - df['negative_reviews']
        df['rating'] = df['rating'].apply(lambda x: max(x, 0))
        return df
```

Далі наведено код функції оновлення таблиці, обробка станів таблиці для оновлення (відфільтрована та відсортована). Оновлення призначене для

актуалізації даних без перезапуску додатку у випадку надходження нових відгуків чи зміни періоду.

```
    def update_table(filter_text="", sort_column=None, descending=False):
        start_date = start_date_entry.get_date().strftime('%Y-%m-%d')
        end_date = end_date_entry.get_date().strftime('%Y-%m-%d')
        data = get_data_from_db(start_date, end_date)
        if filter_text:
            data = data[data['product_name'].str.contains(filter_text,
case=False, na=False)]
        if sort_column:
            data = data.sort_values(by=sort_column, ascending=not
descending)
        for i in table.get_children():
            table.delete(i)
        if not data.empty:
            for _, row in data.iterrows():
                table.insert('', 'end',
                             values=(row['product_name'],
row['positive_reviews'], row['negative_reviews'], row['rating']))
```

Код для сортування даних.

```
    def sort_by_column(column):
        global current_sort_column, descending
        if current_sort_column == column:
            descending = not descending
        else:
            current_sort_column = column
            descending = False
        update_table(filter_text=search_entry.get(), sort_column=column,
descending=descending)
```

Функція «view_reviews» призначена для отримання відгуків на вибраний у таблиці продукт, наявна обробка відсутності обраного товару. При викликанні цієї функції відбувається перехід в окреме вікно для перегляду відгуків на обраний товар.

```
def view_reviews():
    selected_item = table.selection()
    if not selected_item:
        messagebox.showinfo("Вибір продукту", "Виберіть продукт для перегляду відгуків.")
        return
    product_name = table.item(selected_item[0])['values'][0]
    show_reviews_window(product_name)
```

Далі описано функцію для реалізації вікна перегляду відгуків.

```
def show_reviews_window(product_name):
    conn = sqlite3.connect('obolon.db')
    query = """
        SELECT sentiment, review_date, review_text
        FROM Reviews R
        JOIN Products P ON P.id = R.product_id
        WHERE P.product_name = ?
    """
    reviews = pd.read_sql_query(query, conn, params=(product_name,))
    conn.close()
```

Спочатку звернемо увагу що в основному нове вікно для перегляду відгуків представляє собою результат виконання запиту з параметром.

Ініціалізація вікна має вигляд.

```
    reviews_window = tk.Toplevel(root)
    reviews_window.title(f"Відгуки для продукту: {product_name}")
```

Задаються заголовки для таблиці перегляду її зовнішній вигляд та її вміст.

```
        columns = ('Тип відгуку', 'Дата', 'Текст відгуку')
        reviews_table = ttk.Treeview(reviews_window, columns=columns, show='headings')
        for col in columns:
            reviews_table.heading(col, text=col)
        reviews_table.pack(fill='both', expand=True, padx=10, pady=10)
        for _, review in reviews.iterrows():
            sentiment = "Позитивний" if review['sentiment'] == 'positive' else "Негативний"
            reviews_table.insert('', 'end', values=(sentiment, review['review_date'], review['review_text']))
```

Функція для відображення графіку рейтингів. Задаємо дані на основі яких його буде побудовано.

```
    def show_chart():
        start_date = start_date_entry.get_date().strftime('%Y-%m-%d')
        end_date = end_date_entry.get_date().strftime('%Y-%m-%d')
        data = get_data_from_db(start_date, end_date)
```

Обробка відсутності даних.

```
        if data.empty:
            messagebox.showinfo("Немає даних", "Дані для відображення графіку відсутні.")
            return
```

Задання вигляду діаграми.

```
        fig, ax = plt.subplots()
        ax.bar(data['product_name'], data['rating'], color='skyblue')
```

```
        ax.set_title('Рейтинг за продуктами')
        ax.set_xlabel('Продукт')
        ax.set_ylabel('Рейтинг')
        ax.set_xticklabels(data['product_name'], rotation=45, ha='right',
fontsize=8)
        fig.tight_layout()
        chart_window = tk.Toplevel(root)
        chart_window.title("Графік рейтингів продуктів")
        canvas = FigureCanvasTkAgg(fig, master=chart_window)
        canvas.draw()
        canvas.get_tk_widget().pack()
```

Функція «view_and_add_products» реалізує вікно перегляду продукції та її функціонал. Наступні функції «add_product» та «delete_product» описано усередині тіла «view_and_add_products», оскільки вони мають значення винятково всередині нього.

```
    def view_and_add_products():
        view_products_window = tk.Toplevel(root)
        view_products_window.title("Продукти")
        columns = ('Назва продукту',)
        products_table        =        ttk.Treeview(view_products_window,
columns=columns, show='headings')
        products_table.heading('Назва продукту', text="Назва продукту")
        products_table.pack(fill='both', expand=True, padx=10, pady=10)
        conn = sqlite3.connect('obolon.db')
        query = "SELECT product_name FROM Products"
        products = pd.read_sql_query(query, conn)
        conn.close()
        for _, row in products.iterrows():
            products_table.insert('',                                'end',
values=(row['product_name'],))
```

Обробка створення вікна для додавання продукції, його вигляд. «save_product» описано усередині цієї функції.

```
        def add_product():
            add_product_window = tk.Toplevel(view_products_window)
            add_product_window.title("Додати продукт")

            ttk.Label(add_product_window,                    text="Назва
продукту:").pack(padx=10, pady=5)
            product_name_entry = ttk.Entry(add_product_window, width=40)
            product_name_entry.pack(padx=10, pady=5)
```

Далі виконання додання нового продукту. Реалізовано обробку порожнього значення.

```
            def save_product():
                product_name = product_name_entry.get()
                if not product_name:
                    messagebox.showerror("Помилка",  "Назва  продукту  не
може бути порожньою.")
                    return
                conn = sqlite3.connect('obolon.db')
                cursor = conn.cursor()
                cursor.execute("INSERT   INTO   Products   (product_name)
VALUES (?)", (product_name,))
                conn.commit()
                conn.close()
                messagebox.showinfo("Успіх",  f"Продукт  '{product_name}'
успішно додано.")
                add_product_window.destroy()
                view_and_add_products()
```

Реалізація кнопок для додавання продуктів чи відміни операції.

```
            ttk.Button(add_product_window,                  text="Додати",
command=save_product).pack(padx=10, pady=10)
            ttk.Button(add_product_window,                 text="Відміна",
command=add_product_window.destroy).pack(padx=10, pady=5)
        ttk.Button(view_products_window,      text="Додати      продукт",
command=add_product).pack(pady=10)
```

Процедура видалення продукту. Обробка порожнього значення.

```
        def delete_product():
            selected_item = products_table.selection()
            if not selected_item:
                messagebox.showerror("Помилка",  "Виберіть  продукт  для
видалення.")
                return
```

Видалення продукту через використання запиту з параметром мовою SQL.

```
            product_name                                                =
products_table.item(selected_item[0])['values'][0]
            confirm  =  messagebox.askyesno("Підтвердження",  f"Ви  дійсно
хочете видалити продукт '{product_name}'?")
            if confirm:
                conn = sqlite3.connect('obolon.db')
                cursor = conn.cursor()
                cursor.execute("DELETE FROM Products WHERE product_name =
?", (product_name,))
                conn.commit()
                conn.close()
                messagebox.showinfo("Успіх",  f"Продукт  '{product_name}'
успішно видалено.")
                view_and_add_products()
```

Кнопка для видалення продукту.

```
    ttk.Button(view_products_window, text="Видалити продукт", command=delete_product).pack(pady=10)
```

Нарешті опишемо головне вікно програми. Спочатку задається його назва, розмір, стилі та шрифт.

```
root = tk.Tk()
root.title("Аналіз відгуків по продуктах Оболонь")
root.geometry("900x700")
style = ttk.Style(root)
style.theme_use("clam")
style.configure("TLabel", font=("Arial", 10))
style.configure("TButton", font=("Arial", 10, "bold"))
style.configure("Treeview.Heading", font=("Arial", 10, "bold"))
style.configure("Treeview", font=("Arial", 10), rowheight=25)
```

Задання вигляду та значення початкової дати для компоненту «start_date_entry». За замовченням це перше число поточного місяця (див. змінну «first_day_of_month»), кінцева дата у свою чергу це поточне число (див. «today»).

```
date_frame = ttk.LabelFrame(root, text="Оберіть діапазон дат", padding=(10, 5))
date_frame.pack(pady=10, padx=10, fill='x')
today = datetime.today()
first_day_of_month = today.replace(day=1)
ttk.Label(date_frame, text="Початкова дата:").pack(side=tk.LEFT, padx=5)
start_date_entry = DateEntry(date_frame, width=12, background='darkblue', foreground='white', borderwidth=2)
start_date_entry.set_date(first_day_of_month)
start_date_entry.pack(side=tk.LEFT, padx=5)
```

Далі задається значення компоненту «end_date_entry» у вигляді поточного числа.

```
    ttk.Label(date_frame, text="Кінцева дата:").pack(side=tk.LEFT, padx=5)
    end_date_entry = DateEntry(date_frame, width=12, background='darkblue', foreground='white', borderwidth=2)
    end_date_entry.set_date(today)
    end_date_entry.pack(side=tk.LEFT, padx=5)
```

Рядок пошуку за назвою продукту. Обробник події пошуку викликає функцію «update_table» із параметром тексту для пошуку.

```
    search_frame = ttk.Frame(root, padding=(10, 5))
    search_frame.pack(fill='x', padx=10)
    ttk.Label(search_frame, text="Пошук продукту:").pack(side=tk.LEFT, padx=5)
    search_entry = ttk.Entry(search_frame, width=20)
    search_entry.pack(side=tk.LEFT, padx=5)
    def on_search(event):
        filter_text = search_entry.get()
        update_table(filter_text)
    search_entry.bind("<KeyRelease>", on_search)
```

Розміщення кнопок для оновлення, перегляду відгуків, редагування списку товарів та виходу із програми.

```
    button_frame = ttk.Frame(root, padding=(10, 5))
    button_frame.pack(pady=10)
    update_button = ttk.Button(button_frame, text="Оновити дані", command=lambda: update_table(search_entry.get()))
    update_button.grid(row=0, column=0, padx=5)
```

```
    chart_button = ttk.Button(button_frame, text="Показати графік", command=show_chart)
    chart_button.grid(row=0, column=1, padx=5)
    view_reviews_button = ttk.Button(button_frame, text="Переглянути відгуки", command=view_reviews)
    view_reviews_button.grid(row=0, column=2, padx=5)
    view_and_add_products_button = ttk.Button(button_frame, text="Переглянути продукти", command=view_and_add_products)
    view_and_add_products_button.grid(row=0, column=3, padx=5)
    exit_button = ttk.Button(button_frame, text="Вихід", command=on_closing)
    exit_button.grid(row=0, column=4, padx=5)
```

Опис зовнішнього вигляду таблиці рейтингів.

```
    table_frame = ttk.LabelFrame(root, text="Рейтинг продуктів", padding=(10, 5))
    table_frame.pack(pady=10, padx=10, fill='both', expand=True)
    columns_en = ('product_name', 'positive_reviews', 'negative_reviews', 'rating')
    columns = ('Назва', 'Позитивні відгуки', 'Негативні відгуки', 'Рейтинг')
    table = ttk.Treeview(table_frame, columns=columns, show='headings')
    for col, col_en in zip(columns, columns_en):
        table.heading(col, text=col, command=lambda _col=col_en: sort_by_column(_col))
    table.pack(fill='both', expand=True)table.pack(fill='both', expand=True)
    current_sort_column = None
    descending = False
```

Тобто, після опису функціоналу, далі йде налаштування інтерфейсу користувача, його стилі та елементи. Ознайомимось із інтерфейсом користувача у наступному підрозділі.

4.2. Демонстрація роботи програми

Отже, розібравши вирішення поставлених завдань, можна перейти до покрокової демонстрації роботи застосунку та перевірки його роботи.

Головне вікно програми має наступний вигляд.

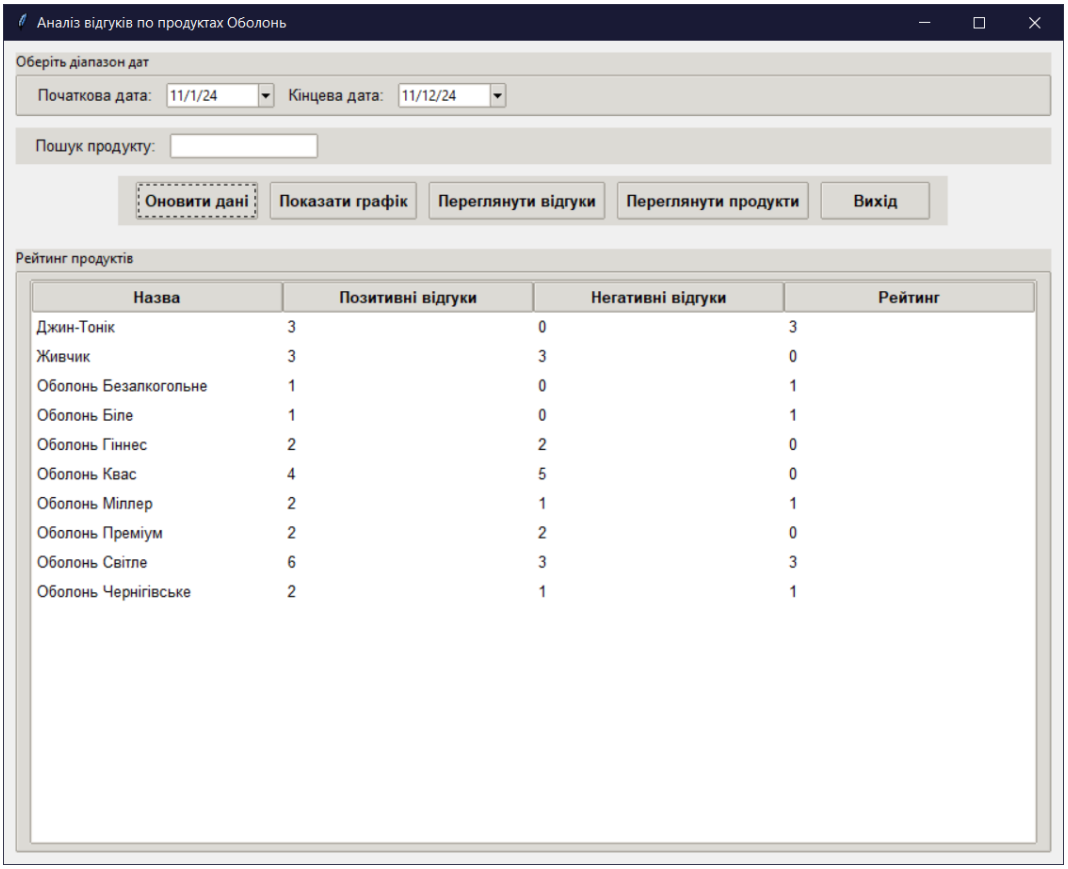
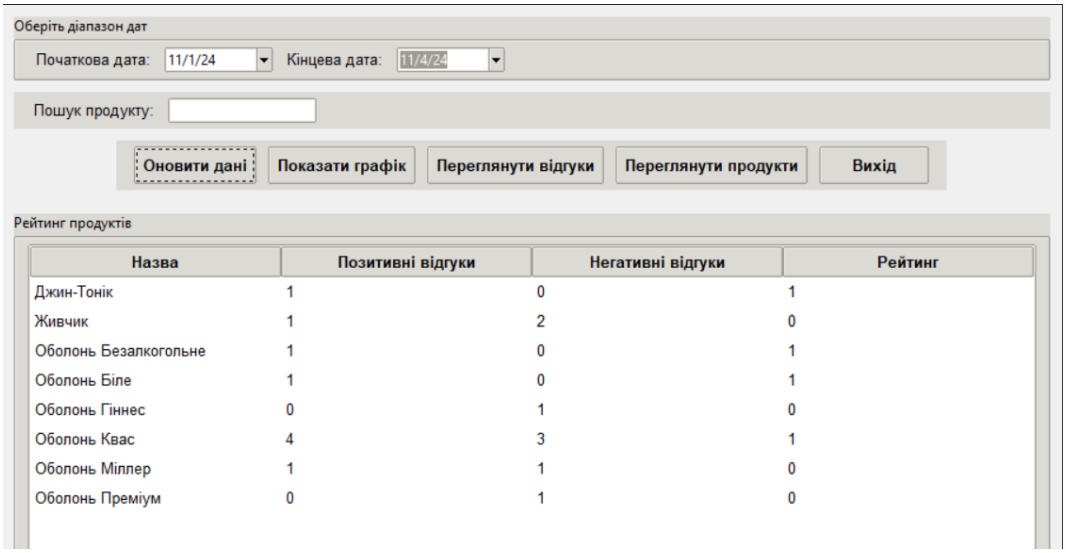


Рисунок 4.6 – Головне вікно програми

Як видно із рисунку 4.6, інтерфейс досить простий для опанування, є меню вибору діапазону дат для збору відгуків за певний період, за замовчуванням від початку місяця до поточної дати, є поле пошуку рейтингу за назвою продукції.

Кнопка «Оновити дані» виконує оновлення таблиці, може знадобитись у випадку змін дат вибірки.



Назва	Позитивні відгуки	Негативні відгуки	Рейтинг
Джин-Тонік	1	0	1
Живчик	1	2	0
Оболонь Безалкогольне	1	0	1
Оболонь Біле	1	0	1
Оболонь Гіннес	0	1	0
Оболонь Квас	4	3	1
Оболонь Міллер	1	1	0
Оболонь Преміум	0	1	0

Рисунок 4.7 – Приклад зміни діапазону дат вибірки

Кнопка «Показати графік» відповідає як видно із назви за побудову графіку на основі даних головної таблиці рейтингу продукції.

Візуалізація рейтингу дає змогу наочним чином порівняти рейтинги видів продукції.

Рисунок 4.8 – Приклад діаграми за рейтинговим балом

Кнопка «Переглянути відгуки» спрацьовує у випадку якщо користувач вибере у таблиці рейтингів потрібний продукт.

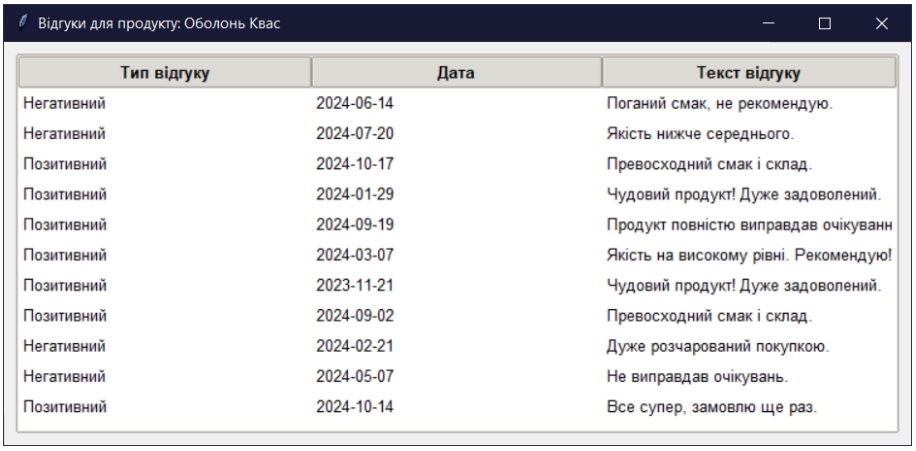


Рисунок 4.9 – Список відгуків для обраного продукту

Кнопка «Переглянути продукти» викликає вікно для перегляду, додавання та видалення товару. Додамо новий товар, напишемо для нього відгук та потім видалимо.

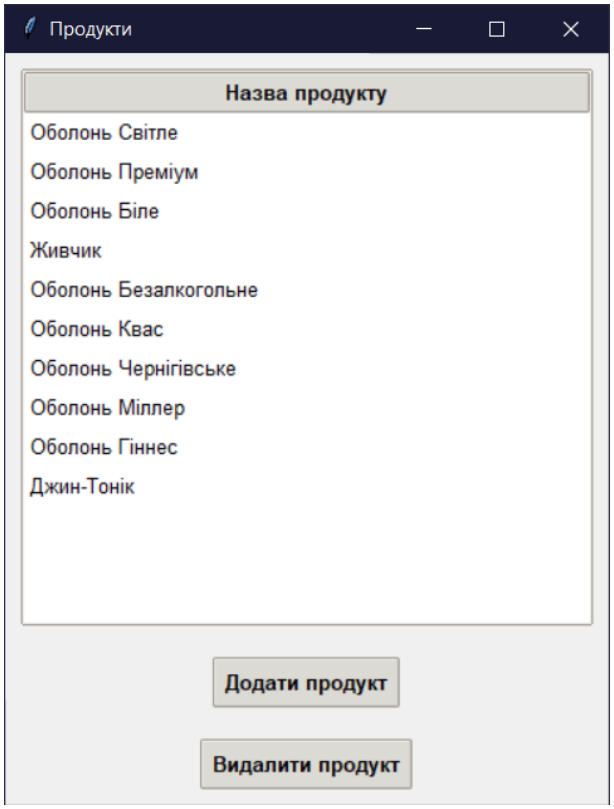


Рисунок 4.10 – Список продуктів

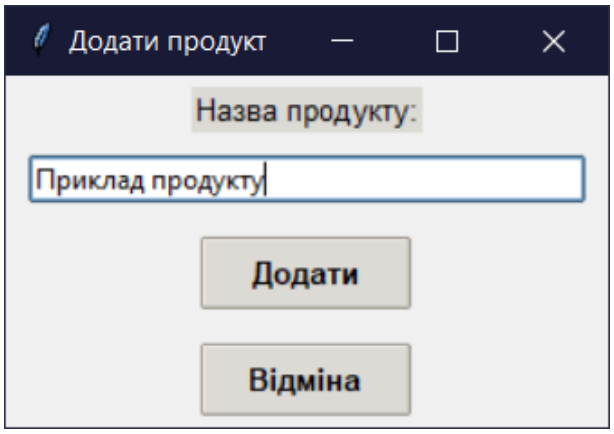


Рисунок 4.11 – Вікно додавання нового продукту

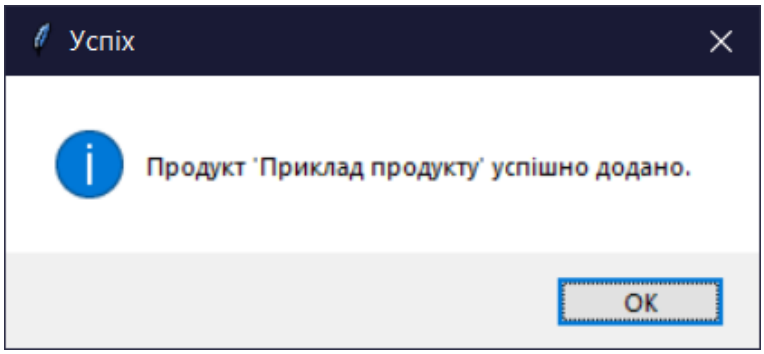


Рисунок 4.12 – Повідомлення про успішне внесення нового товару

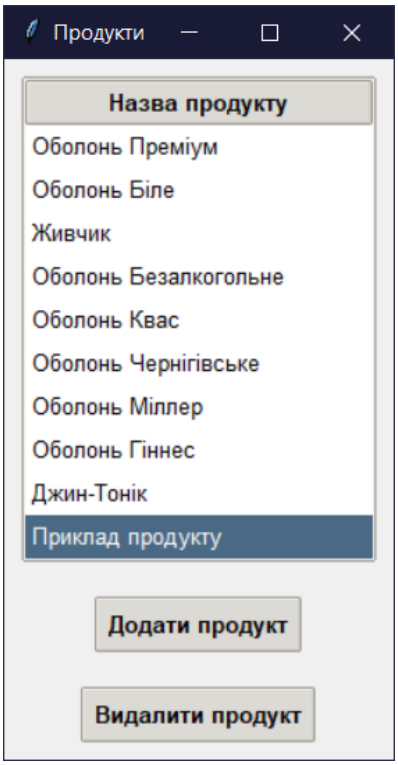


Рисунок 4.13 – Новий товар у списку

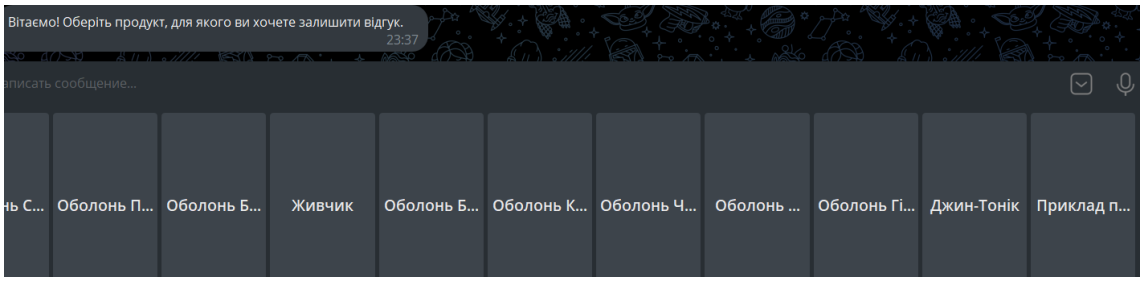


Рисунок 4.14 – Новий товар у переліку товарів при написанні відгуку

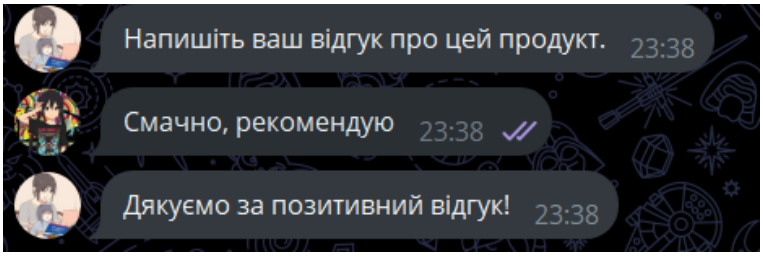
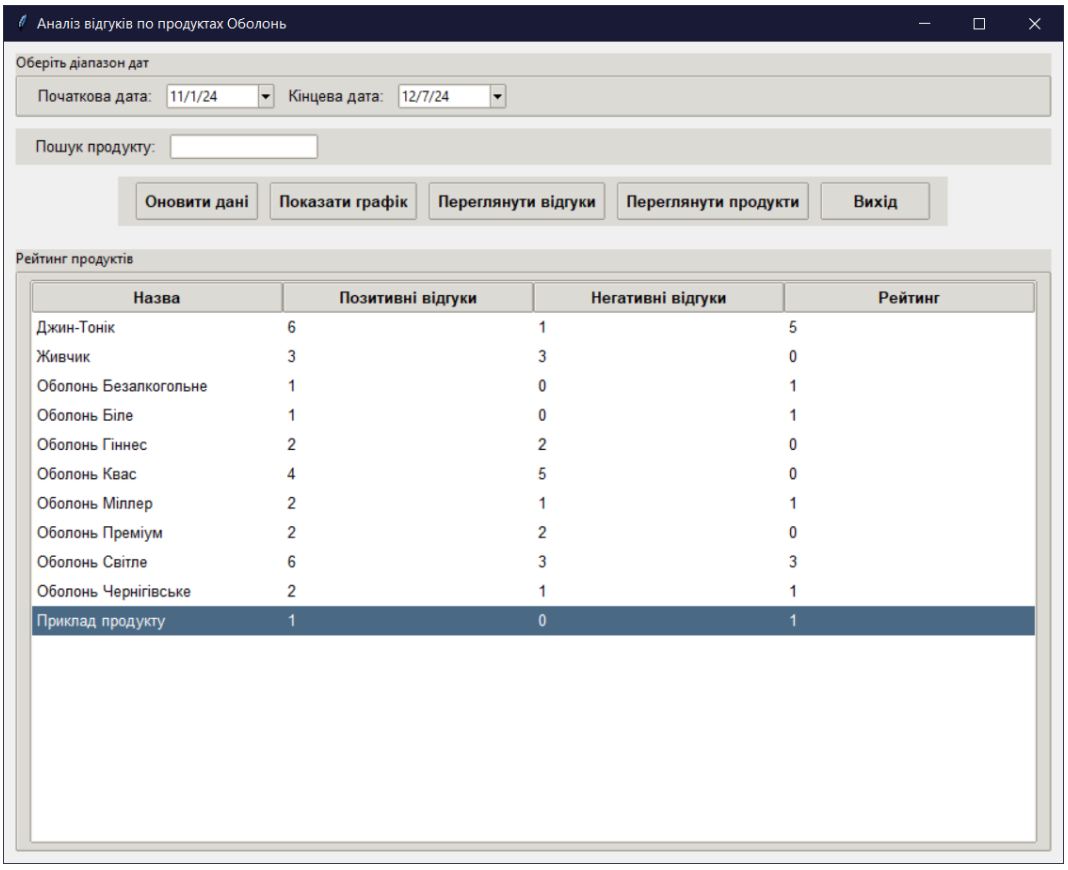


Рисунок 4.15 – Написання відгуку для нового товару



Назва	Позитивні відгуки	Негативні відгуки	Рейтинг
Джин-Тонік	6	1	5
Живчик	3	3	0
Оболонь Безалкогольне	1	0	1
Оболонь Біле	1	0	1
Оболонь Гіннес	2	2	0
Оболонь Квас	4	5	0
Оболонь Міллер	2	1	1
Оболонь Преміум	2	2	0
Оболонь Світле	6	3	3
Оболонь Чернігівське	2	1	1
Приклад продукту	1	0	1

Рисунок 4.16 – Новий товар у рейтинговому списку

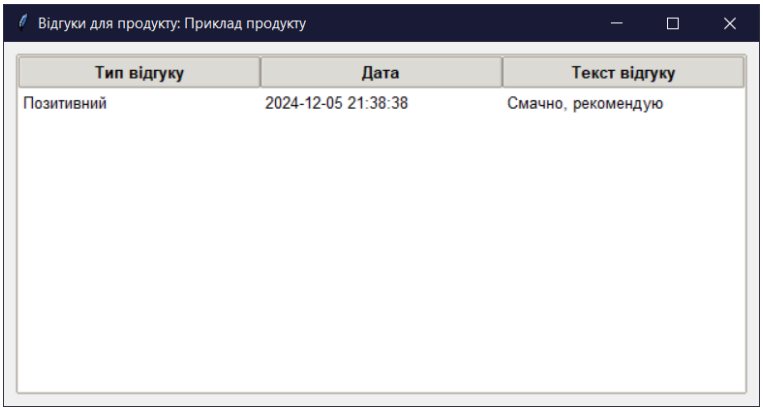


Рисунок 4.17 – Написаний для цього товару відгук у списку відгуків

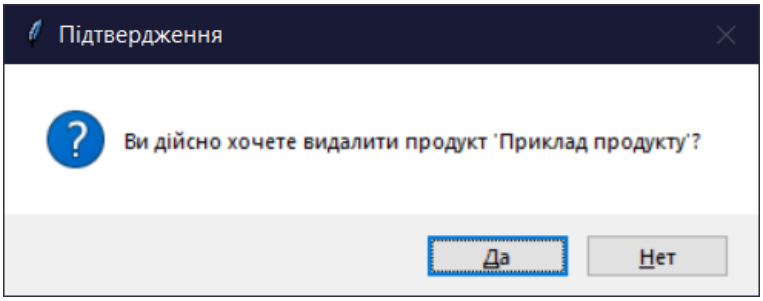


Рисунок 4.18 – Повідомлення про підтвердження видалення

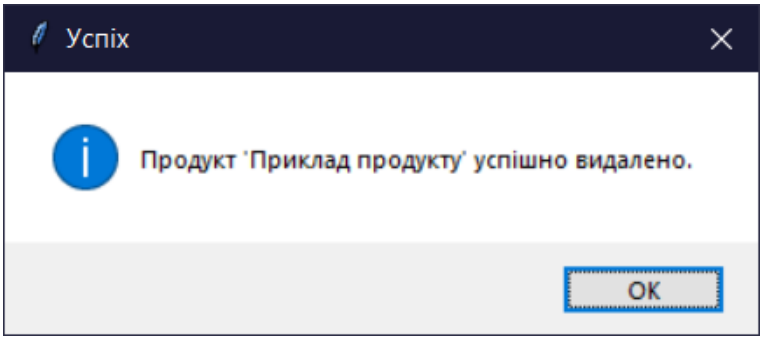


Рисунок 4.19 – Повідомлення про видалення

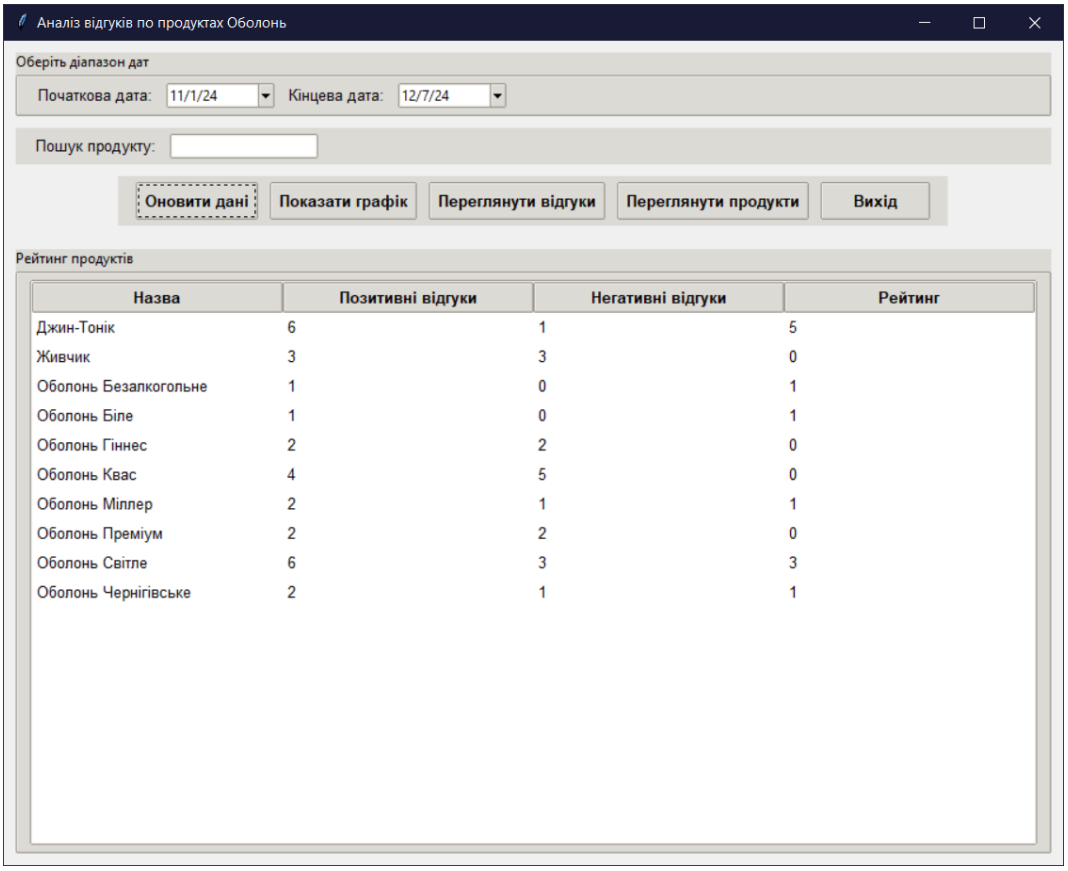


Рисунок 4.20 – Вигляд списку після видалення прикладу продукту

Розглянемо сортування, наприклад за рейтингом, натиснувши потрібний заголовок головної таблиці.

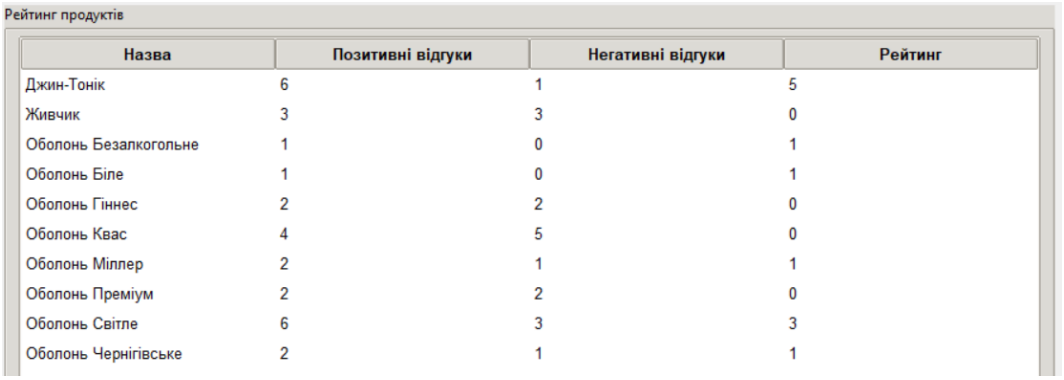


Рисунок 4.21 – Вигляд таблиці до сортування

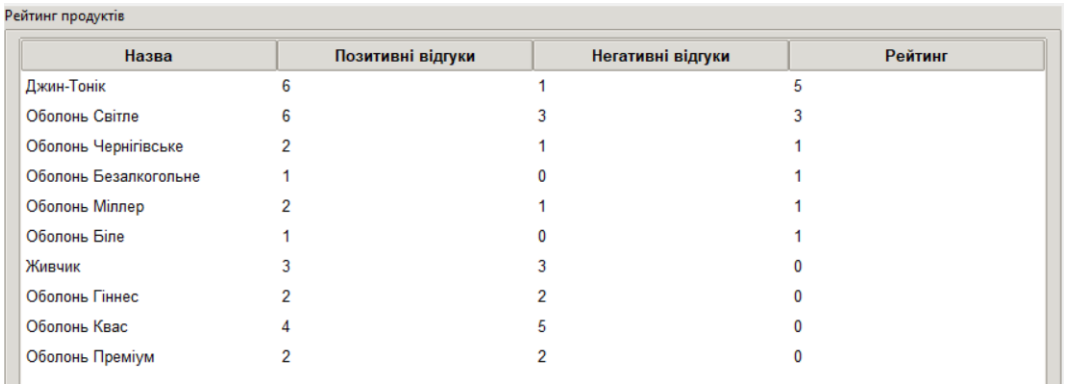


Рисунок 4.22 – Вигляд таблиці після сортування

Спробуємо віднайти рейтинг та кількість відгуків для конкретного продукту, наприклад для «Живчик».

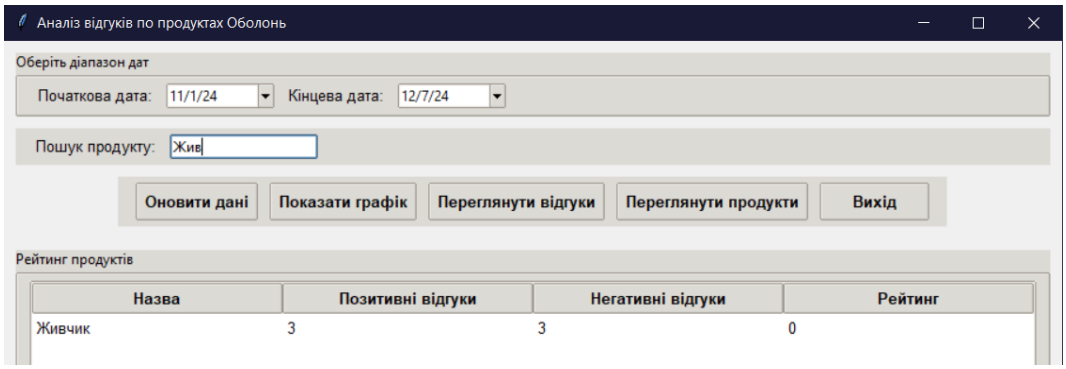


Рисунок 4.23 – Приклад пошуку товару за назвою

Функція пошуку працює одночасно із введенням назви, тобто у випадку, якщо користувач хочу віднайти товари, які починаються з літер «П», він просто може її ввести у рядок пошуку.

Наостанок, кнопка «Вихід» виконує закриття застосунку аналогічно до кнопки закриття вікна.

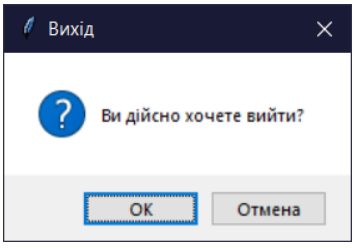


Рисунок 4.24 – Вікно виходу із додатку

На рисунках 4.14 - 4.15 можна побачити процедуру написання відгуку через застосування чат-боту, спробуємо написати негативний та позитивний відгуки для «Джин-Тонік» та переглянути їх. Першою командою для активації є «/start», отримуємо вітання від боту, та список продуктів, з якого можемо обрати потрібний або ввести самостійно.

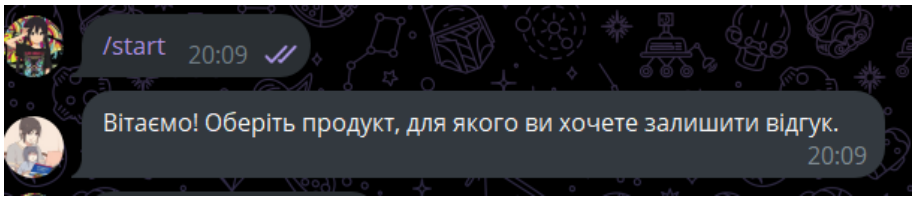


Рисунок 4.25 – Початок написання відгуку через чат-бот

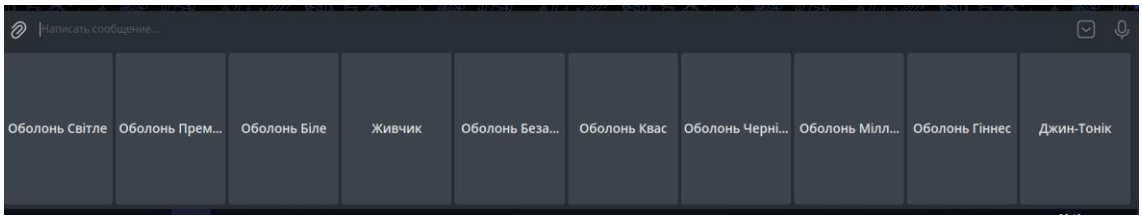


Рисунок 4.26 – Перелік товарів для написання відгуку

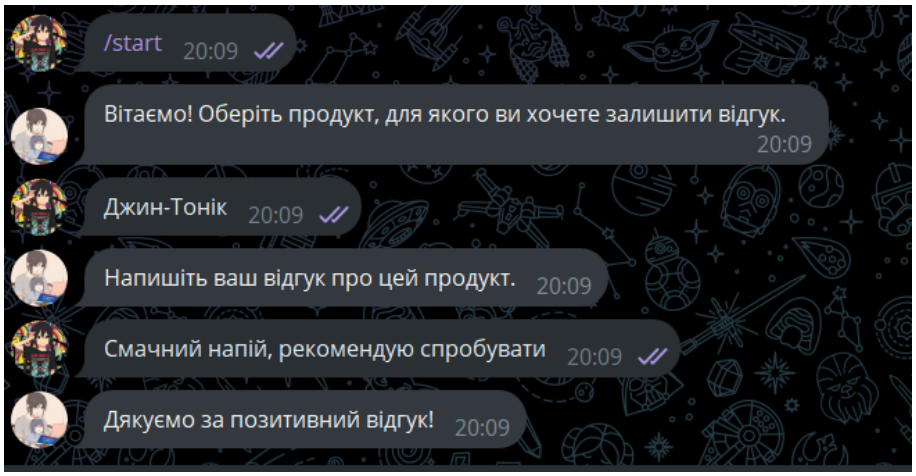


Рисунок 4.27 – Приклад написання позитивного відгуку

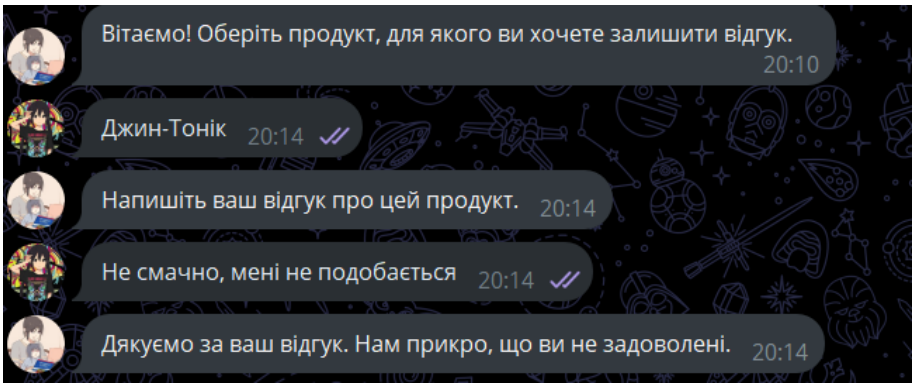


Рисунок 4.28 – Приклад написання негативного відгуку

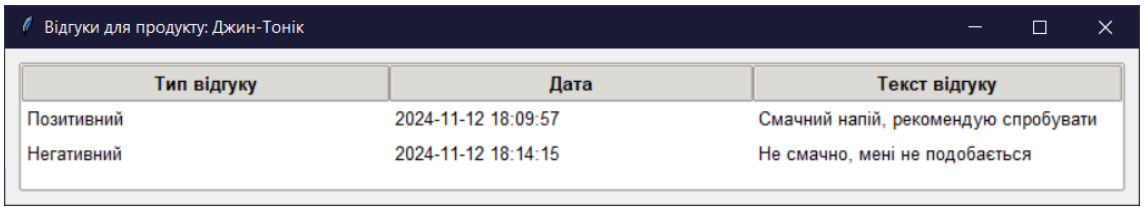


Рисунок 4.29 – Написані розподілені відгуки

Отже, розглянувши роботу застосунку, перевіривши його роботу перейдемо до підведення підсумків щодо даного розділу.

4.3. Висновки до четвертого розділу

Отже, створений додаток виконує поставлені завдання, надає можливість обробки та збору даних, які представлені відгуками. В рамках додатку реалізовано аналіз тональності тексту та його класифікацію обраними після проведення дослідження методами. Додаток має простий та зрозумілий інтерфейс, надає необхідний функціонал та може слугувати для виконання завдань щодо інформаційної підтримки процесу формування асортименту ПрАТ «Оболонь».

Обраний засіб реалізації протягом розділу 3 довів свою ефективність шляхом свого застосування стосовно задач збору та аналізу відгуків за використанням сучасних платформ у якості джерел даних та моделей, заснованих на використанні трансформерів, для аналізу тональності та класифікації тексту.

Python як мова програмування для вирішення поставлених завдань надала потужний набір бібліотек для аналізу, обробки та візуалізації даних, наприклад Pandas, Tkinter, Sqlite та Matplotlib, бібліотеки Telegram та Tweepy, використані

для збору даних, а для реалізації обробки природної мови Torch та Transformers. Обраний метод реалізації також є найбільш доцільним із інструментів, оскільки гарно зарекомендував себе для вирішення аналітичних завдань [25]. Завдяки простоті синтаксису та розвиненій екосистемі, обраний засіб дозволив швидким та ефективним чином реалізувати задачі щодо інформаційної підтримки формування асортименту ПрАТ «Оболонь».

ВИСНОВКИ

Методи збору та аналізу даних є важливою складовою процесу прийняття рішень для формування асортименту продукції ПрАТ «Оболонь». Завдяки сучасним інструментам обробки інформації, підприємство має можливість оперативним чином реагувати на зміни у вподобаннях споживачів та адаптувати свій асортимент відповідно до ринкових тенденцій.

Після дослідження підходів до збору та аналізу даних за допомогою сучасних методів та платформ визначено, що найкращим засобами для збору даних є застосування чат-ботів та парсингу соцмереж, у зв'язку із їхньою швидкістю та економністю роботи, а для аналізу – технології обробки природної мови, оскільки вони є найбільш оптимальними для роботи із текстовими даними у вигляді відгуків і дозволяють підприємству глибше розуміти потреби клієнтів задля своєчасного пристосування до них, що дозволяє уникати додаткових збитків та зміцнювати свої позиції на ринку. Збирання відгуків з соцмереж та через чат-боти є оптимальним поєднанням вартості та ефективності, процес збору даних за їх застосуванням є швидким та дешевим.

Для виконання завдання аналізу настрою відгуків на продукцію та класифікації зібраних даних точним та швидким чином, досліджено методи обробки природної мови та обрано для цієї цілі методи засновані на використанні моделі трансформерів, оскільки вони можуть забезпечити високу точність та швидкість роботи при обробці великих об'ємів тексту.

Після проведення дослідження діяльності ПрАТ «Оболонь» визначено його роботи щодо розроблення рекомендацій для формування асортименту, визначено, що інформаційна підтримка представляє собою рейтинг, який ілюструє реакцію споживачів на продукцію ПрАТ «Оболонь». На основі дослідження підприємства, запропоновано рішення у вигляді окремого програмного продукту для реалізації методів збору та аналізу відгуків.

Створене мовою Python програмне рішення дозволяє виконувати обробку підготовку та звернення до даних, при цьому зберігаючи їх належним чином.

Застосунок на основі зібраних даних здійснює побудову рейтингу продукції для вивчення реакції споживачів, здійснюючи інформаційну підтримку формування асортименту продукції ПрАТ «Оболонь».

Застосування методів збору та аналізу даних значно спрощує управління асортиментом ПрАТ «Оболонь», оскільки забезпечує більш структурований та інформативний підхід до роботи з відгуками клієнтів, що має потенціал не тільки для побудови рейтингів, як у створеному застосунку, а ще й у сфері Data Science взагалі. Тобто імплементація цих технологій має відчутний потенціал для ПрАТ «Оболонь».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Історія – Оболонь [Електронний ресурс]. – Дані. Текст. – Режим доступу: https://obolon.ua/ua/about/history (дата звернення: 08.10.2024).
2. Роль види та методи соціологічного опитування [Електронний ресурс]. – Дані. Текст. – Режим доступу: https://osvita.ua/vnz/reports/sociology/12369/ (дата звернення: 08.10.2024).
3. Фокус-група як метод маркетингових досліджень [Електронний ресурс]. – Дані. Текст. – Режим доступу: http://market.avianua.com/?p=4381 (дата звернення: 10.10.2024).
4. Якими бувають інтерв'ю в маркетингу та як з їхньою допомогою зрозуміти клієнтів [Електронний ресурс]. – Дані. Текст. – Режим доступу: https://aboutmarketing.info/internet-marketynh/yakymy-buvayut-intervyu-v-marketynhu-ta-yak-z-yikhnoyu-dopomohoyu-zrozumity-kliyentiv/ (дата звернення: 12.10.2024).
5. Переваги і недоліки інтернет-опитувань [Електронний ресурс]. – Дані. Текст. – Режим доступу: https://stud.com.ua/115575/marketing/perevagi_nedoliki_internet_opituvan (дата звернення: 12.10.2024).
6. Що таке API у додатках та сайтах [Електронний ресурс]. – Дані. Текст. – Режим доступу: https://wezom.com.ua/ua/blog/chto-takoe-api-v-prilozhenijah-i-sajtah (дата звернення: 12.10.2024).
7. Examples – tweepy 4.14.0 [Електронний ресурс]. – Дані. – Режим доступу: https://docs.tweepy.org/en/stable/examples.html (дата звернення: 12.10.2024).
8. Чат-бот для бізнесу [Електронний ресурс]. – Дані. – Режим доступу: https://www.promodo.ua/blog/chat-boti-dlya-riznih-potreb-biznesu-rekomendaciyi-fahivciv-promodo (дата звернення: 12.10.2024).

9. Парсинг Соціальних Мереж [Електронний ресурс]. – Дані. – Режим доступу: https://datamining.com.ua/parsing-socialnih-merezh/ (дата звернення: 12.10.2024).
10. What is NLP. [Електронний ресурс]. – Дані. – Режим доступу: www.ibm.com/topics/natural-language-processing (дата звернення: 13.10.2024).
11. Understanding Text Classification In Natural Language Processing—A Beginners' Guide [Електронний ресурс]. – Дані. – Режим доступу: https://www.linkedin.com/pulse/understanding-text-classification-natural-language-david-adamson-mbcs (дата звернення: 13.10.2024).
12. What is named entity recognition (NER)? [Електронний ресурс]. – Дані. – Режим доступу: https://www.techtarget.com/whatis/definition/named-entity-recognition-NER (дата звернення: 14.10.2024).
13. Baeldung. Sentiment Analysis Dictionaries [Електронний ресурс]. – Дані. – Режим доступу: https://www.baeldung.com/cs/sentiment-analysis-dictionaries (дата звернення: 26.10.2024).
14. Anote AI. Naive Bayes Bag of Words Approach: A Powerful Text Classification Technique [Електронний ресурс]. – Дані. – Режим доступу: https://anote-ai.medium.com/naive-bayes-bag-of-words-approach-a-powerful-text-classification-technique-8ccfb07ac2be/ (дата звернення: 26.10.2024).
15. QuantStart. Support Vector Machines: A Guide for Beginners [Електронний ресурс]. – Дані. – Режим доступу: https://www.quantstart.com/articles/Support-Vector-Machines-A-Guide-for-Beginners/ (дата звернення: 27.10.2024).
16. Spot Intelligence. Logistic Regression for Text Classification in Python [Електронний ресурс]. – Дані. – Режим доступу: https://spotintelligence.com/2023/02/22/logistic-regression-text-classification-python/ (дата звернення: 27.10.2024).

17. Colah, C. Understanding LSTMs [Електронний ресурс]. – Дані. – Режим доступу: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення: 28.10.2024).
18. Goel, A. From BERT to GPT: The Transformative Power of Transformers in NLP [Електронний ресурс]. – Дані. – Режим доступу: https://medium.com/@ashu.goel_9925/from-bert-to-gpt-the-transformative-power-of-transformers-in-nlp-68666428aa55 (дата звернення: 29.10.2024).
19. Wiley Online Library. Use of Transformers in Text Processing and NLP [Електронний ресурс]. – Дані. – Режим доступу: https://onlinelibrary.wiley.com/doi/10.1155/2021/9986920 (дата звернення: 03.11.2024).
20. Vaswani, A., Shazeer, N., Parmar, N., et al. Attention is All You Need // Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS). – 2017. – С. 4–5.
21. DataCamp. How Transformers Work [Електронний ресурс]. – Дані. – Режим доступу: https://www.datacamp.com/tutorial/how-transformers-work (дата звернення: 03.11.2024).
22. Tavva, P. BERT vs GPT: A Tale of Two Transformers That Revolutionized NLP [Електронний ресурс]. – Дані. – Режим доступу: https://medium.com/@prudhvithtavva/bert-vs-gpt-a-tale-of-two-transformers-that-revolutionized-nlp-11fff8e61984 (дата звернення: 03.11.2024).
23. Де використовується Python і чому вам потрібно знати цю мову – Genius space [Електронний ресурс] – Дані. – Режим доступу: https://genius.space/lab/de-vikoristovuyetsya-python-i-chomu-vam-potribno-znati-tsyu-movu/ (дата звернення: 06.11.2024).
24. Мова програмування Python для машинного навчання та штучного інтелекту [Електронний ресурс] – Дані. – Режим доступу: https://goit.global/ua/articles/mova-prohramuvannia-python-dlia-mashynnoho-navchannia-ta-shtuchnoho-intelektu/ (дата звернення: 06.11.2024).

25. Найкращі бібліотеки Python у 2023 році [Електронний ресурс]. – Дані. Текст. – Режим доступу: https://spacelab.ua/articles/biblioteki-python-u-2023-rosi/ (дата звернення: 06.11.2024).
26. Методичні рекомендації до виконання кваліфікаційної роботи на здобуття освітнього ступеня «Магістр» спеціальності 122 «Комп'ютерні науки» освітньо-професійної програми «Управління інформацією та аналітика даних» денної форми здобуття освіти [Електрон. ресурс] / уклад. М. П. Костіков, С. В. Грибков, Н. В. Ліманська. К.: НУХТ, 2024. – 29 с.

ДОДАТКИ

Додаток А. Приклади інтерфейсу програми

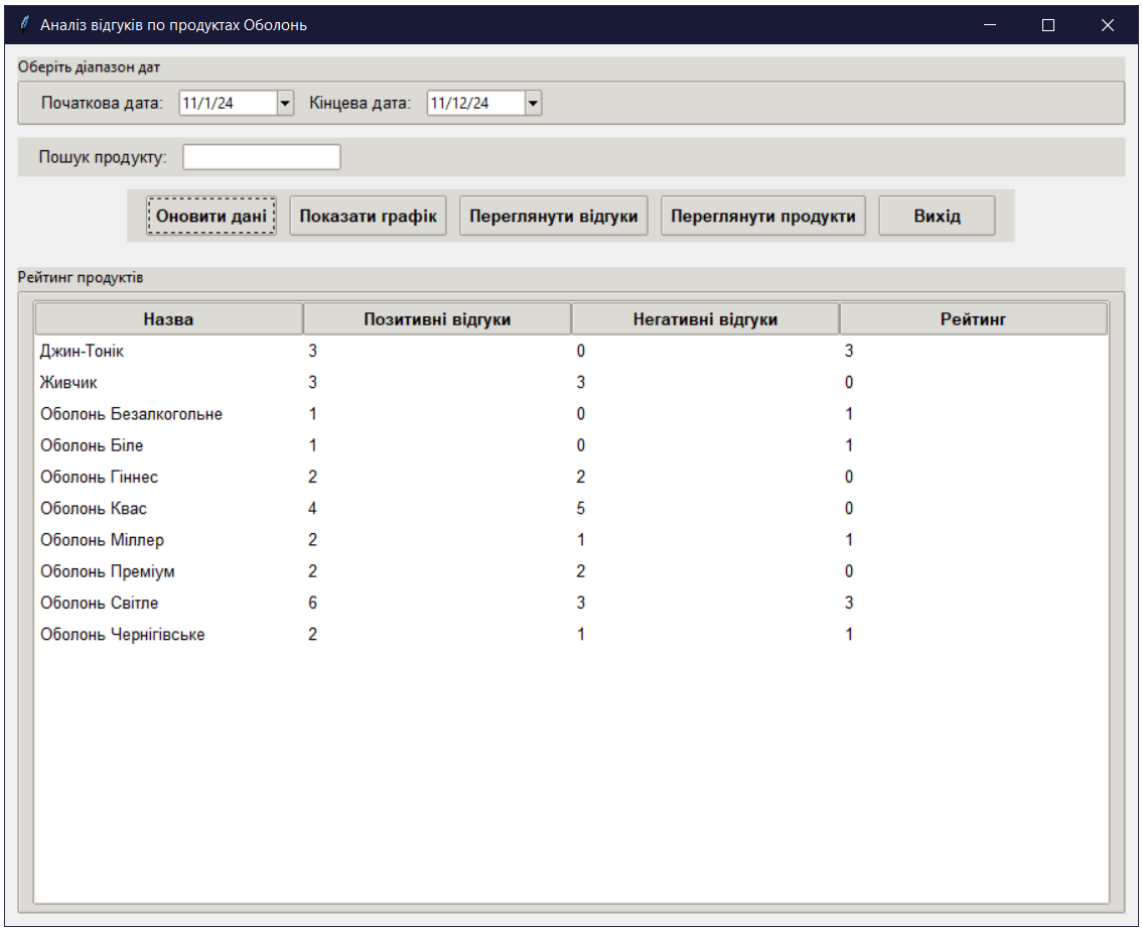


Рисунок А.1 – Головне вікно додатку

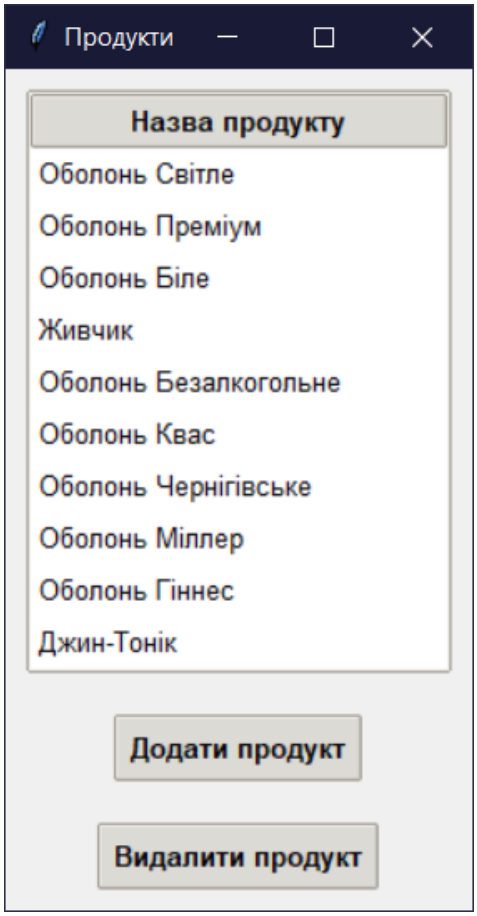


Рисунок А.2 – Меню перегляду та редагування списку продуктів

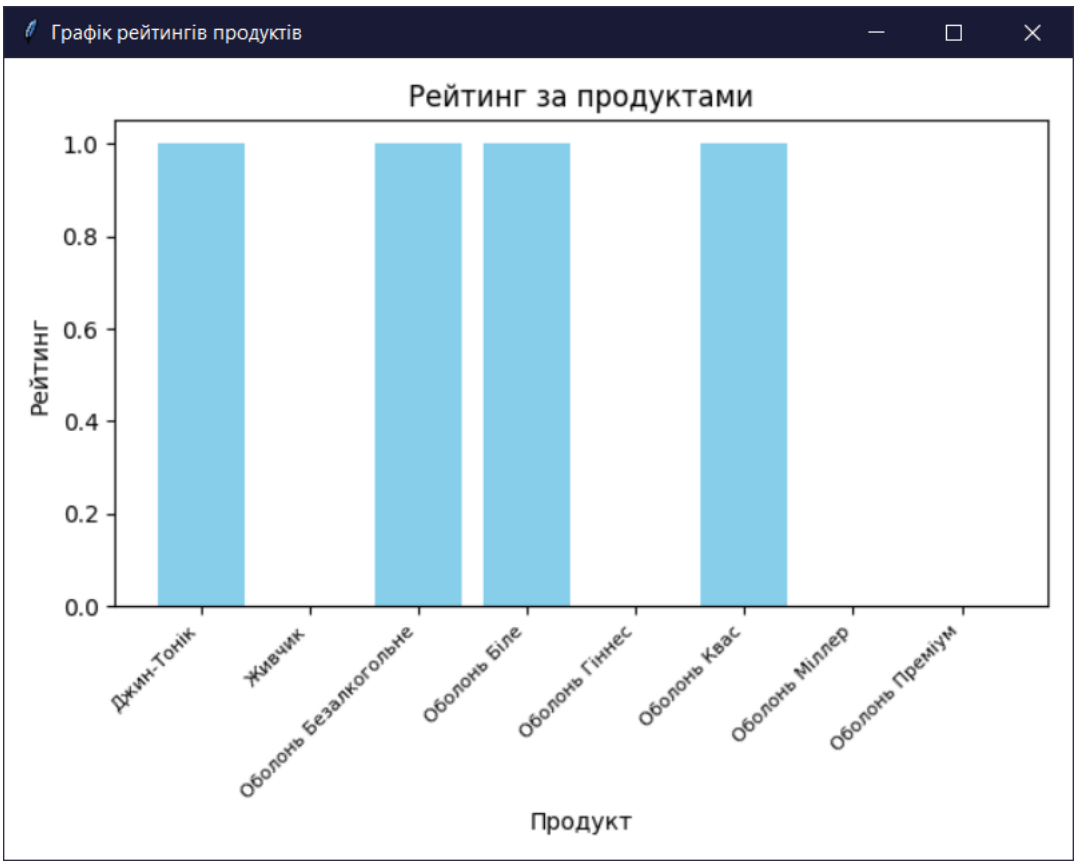


Рисунок А.3 – Приклад графіку рейтингів продукції

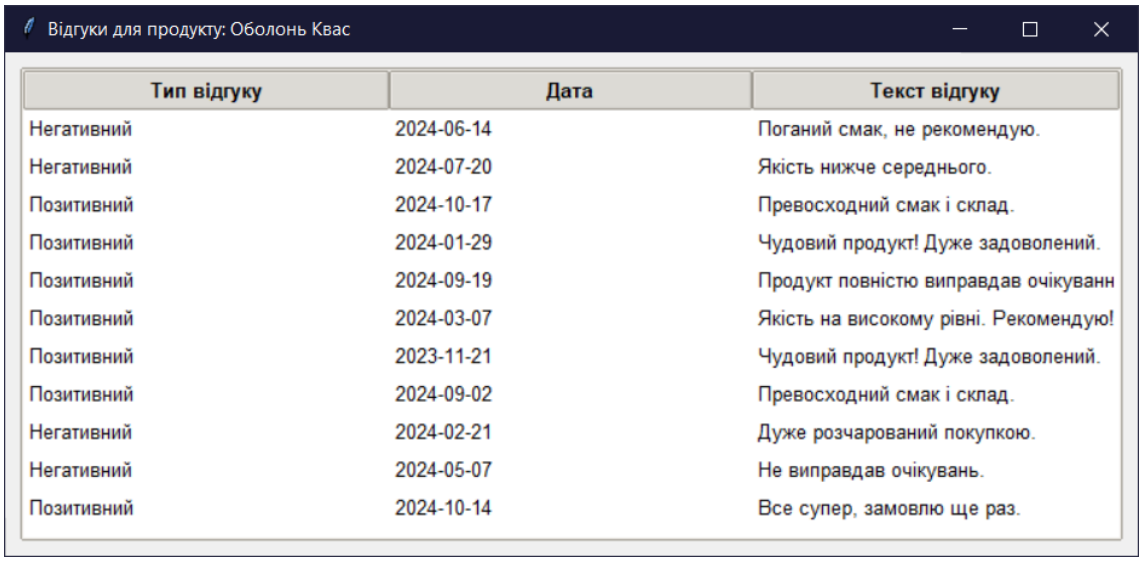


Рисунок А.4 – Вікно перегляду відгуків

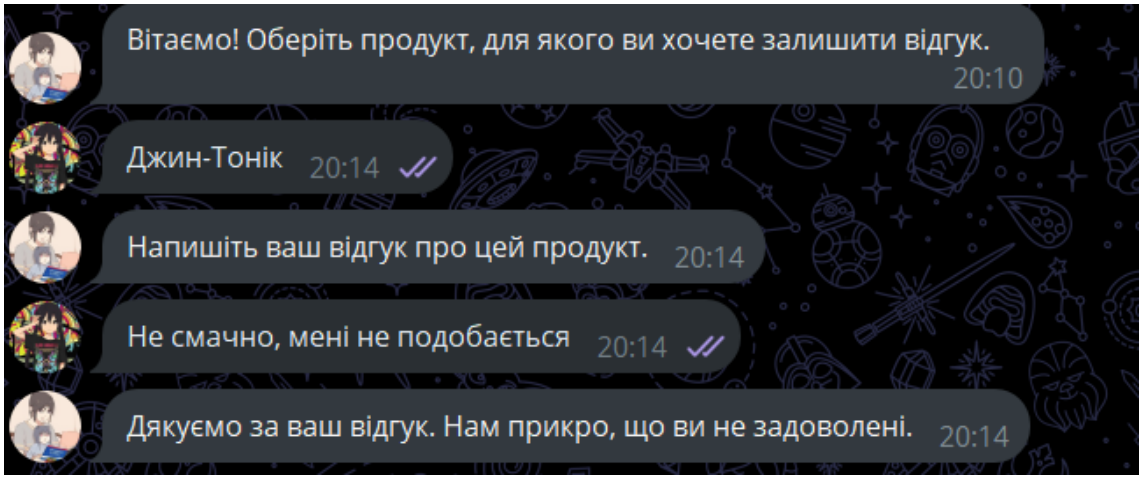


Рисунок А.5 – Приклад написання відгуку через розроблений бот

Додаток Б. Програмний код застосунку

Б.1. Код «proj.py» для реалізації функціоналу віконного додатку:

```
import sqlite3
import pandas as pd
import tkinter as tk
import tkinter.filedialog as filedialog
from tkinter import ttk, messagebox
from tkcalendar import DateEntry
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from datetime import datetime
def get_data_from_db(start_date, end_date):
    conn = sqlite3.connect('obolon.db')
    query = """
        SELECT P.product_name,
               SUM(CASE WHEN R.sentiment = 'positive' THEN 1 ELSE 0
END) AS positive_reviews,
               SUM(CASE WHEN R.sentiment = 'negative' THEN 1 ELSE 0
END) AS negative_reviews
        FROM Products P
        JOIN Reviews R ON P.id = R.product_id
        WHERE R.review_date BETWEEN ? AND ?
        GROUP BY P.product_name
    """
    df    =    pd.read_sql_query(query,    conn,    params=(start_date,
end_date))
    conn.close()
    df['rating'] = df['positive_reviews'] - df['negative_reviews']
    df['rating'] = df['rating'].apply(lambda x: max(x, 0))
    return df
def update_table(filter_text="", sort_column=None, descending=False):
```

```
        start_date = start_date_entry.get_date().strftime('%Y-%m-%d')
        end_date = end_date_entry.get_date().strftime('%Y-%m-%d')
        data = get_data_from_db(start_date, end_date)
        if filter_text:
            data = data[data['product_name'].str.contains(filter_text,
case=False, na=False)]
        if sort_column:
            data = data.sort_values(by=sort_column, ascending=not
descending)
        for i in table.get_children():
            table.delete(i)
        if not data.empty:
            for _, row in data.iterrows():
                table.insert('', 'end',
                             values=(row['product_name'],
row['positive_reviews'], row['negative_reviews'], row['rating']))
    def sort_by_column(column):
        global current_sort_column, descending
        if current_sort_column == column:
            descending = not descending
        else:
            current_sort_column = column
            descending = False
        update_table(filter_text=search_entry.get(), sort_column=column,
descending=descending)
    def view_reviews():
        selected_item = table.selection()
        if not selected_item:
            messagebox.showinfo("Вибір продукту", "Виберіть продукт для
перегляду відгуків.")
            return
        product_name = table.item(selected_item[0])['values'][0]
        show_reviews_window(product_name)
    def show_reviews_window(product_name):
```

```
        conn = sqlite3.connect('obolon.db')
        query = """
            SELECT sentiment, review_date, review_text
            FROM Reviews R
            JOIN Products P ON P.id = R.product_id
            WHERE P.product_name = ?
        """
        reviews = pd.read_sql_query(query, conn, params=(product_name,))
        conn.close()
        reviews_window = tk.Toplevel(root)
        reviews_window.title(f"Відгуки для продукту: {product_name}")
        columns = ('Тип відгуку', 'Дата', 'Текст відгуку')
        reviews_table = ttk.Treeview(reviews_window, columns=columns,
show='headings')
        for col in columns:
            reviews_table.heading(col, text=col)
        reviews_table.pack(fill='both', expand=True, padx=10, pady=10)
        for _, review in reviews.iterrows():
            sentiment = "Позитивний" if review['sentiment'] == 'positive'
else "Негативний"
            reviews_table.insert('', 'end', values=(sentiment,
review['review_date'], review['review_text']))
    def show_chart():
        start_date = start_date_entry.get_date().strftime('%Y-%m-%d')
        end_date = end_date_entry.get_date().strftime('%Y-%m-%d')
        data = get_data_from_db(start_date, end_date)
        if data.empty:
            messagebox.showinfo("Немає даних", "Дані для відображення
графіку відсутні.")
            return
        fig, ax = plt.subplots()
        ax.bar(data['product_name'], data['rating'], color='skyblue')
        ax.set_title('Рейтинг за продуктами')
        ax.set_xlabel('Продукт')
```

```
        ax.set_ylabel('Рейтинг')
        ax.set_xticklabels(data['product_name'], rotation=45, ha='right',
fontsize=8)
        fig.tight_layout()
        chart_window = tk.Toplevel(root)
        chart_window.title("Графік рейтингів продуктів")
        canvas = FigureCanvasTkAgg(fig, master=chart_window)
        canvas.draw()
        canvas.get_tk_widget().pack()
    def on_closing():
        if messagebox.askokcancel("Вихід", "Ви дійсно хочете вийти?"):
            root.destroy()
    def view_and_add_products():
        view_products_window = tk.Toplevel(root)
        view_products_window.title("Продукти")
        columns = ('Назва продукту',)
        products_table        =        ttk.Treeview(view_products_window,
columns=columns, show='headings')
        products_table.heading('Назва продукту', text="Назва продукту")
        products_table.pack(fill='both', expand=True, padx=10, pady=10)
        conn = sqlite3.connect('obolon.db')
        query = "SELECT product_name FROM Products"
        products = pd.read_sql_query(query, conn)
        conn.close()
        for _, row in products.iterrows():
            products_table.insert('',                                'end',
values=(row['product_name'],))
        def add_product():
            add_product_window = tk.Toplevel(view_products_window)
            add_product_window.title("Додати продукт")
            ttk.Label(add_product_window,                        text="Назва
продукту:").pack(padx=10, pady=5)
            product_name_entry = ttk.Entry(add_product_window, width=40)
            product_name_entry.pack(padx=10, pady=5)
```

```
            def save_product():
                product_name = product_name_entry.get()
                if not product_name:
                    messagebox.showerror("Помилка", "Назва продукту не
може бути порожньою.")
                    return
                conn = sqlite3.connect('obolon.db')
                cursor = conn.cursor()
                cursor.execute("INSERT INTO Products (product_name)
VALUES (?)", (product_name,))
                conn.commit()
                conn.close()
                messagebox.showinfo("Успіх", f"Продукт '{product_name}'
успішно додано.")
                add_product_window.destroy()
                view_and_add_products()
            ttk.Button(add_product_window, text="Додати",
command=save_product).pack(padx=10, pady=10)
            ttk.Button(add_product_window, text="Відміна",
command=add_product_window.destroy).pack(padx=10, pady=5)
        ttk.Button(view_products_window, text="Додати продукт",
command=add_product).pack(pady=10)
        def delete_product():
            selected_item = products_table.selection()
            if not selected_item:
                messagebox.showerror("Помилка", "Виберіть продукт для
видалення.")
                return
            product_name =
products_table.item(selected_item[0])['values'][0]
            confirm = messagebox.askyesno("Підтвердження", f"Ви дійсно
хочете видалити продукт '{product_name}'?")
            if confirm:
                conn = sqlite3.connect('obolon.db')
```

```
                cursor = conn.cursor()
                cursor.execute("DELETE FROM Products WHERE product_name =
?", (product_name,))
                conn.commit()
                conn.close()
                messagebox.showinfo("Успіх",  f"Продукт  '{product_name}'
успішно видалено.")
                view_and_add_products()
        ttk.Button(view_products_window,     text="Видалити     продукт",
command=delete_product).pack(pady=10)
    root = tk.Tk()
    root.title("Аналіз відгуків по продуктах Оболонь")
    root.geometry("900x700")
    style = ttk.Style(root)
    style.theme_use("clam")
    style.configure("TLabel", font=("Arial", 10))
    style.configure("TButton", font=("Arial", 10, "bold"))
    style.configure("Treeview.Heading", font=("Arial", 10, "bold"))
    style.configure("Treeview", font=("Arial", 10), rowheight=25)
    date_frame  =  ttk.LabelFrame(root,  text="Оберіть  діапазон  дат",
padding=(10, 5))
    date_frame.pack(pady=10, padx=10, fill='x')
    today = datetime.today()
    first_day_of_month = today.replace(day=1)
    ttk.Label(date_frame,    text="Початкова    дата:").pack(side=tk.LEFT,
padx=5)
    start_date_entry       =       DateEntry(date_frame,       width=12,
background='darkblue', foreground='white', borderwidth=2)
    start_date_entry.set_date(first_day_of_month)
    start_date_entry.pack(side=tk.LEFT, padx=5)
    ttk.Label(date_frame,     text="Кінцева     дата:").pack(side=tk.LEFT,
padx=5)
    end_date_entry        =        DateEntry(date_frame,        width=12,
background='darkblue', foreground='white', borderwidth=2)
```

```
    end_date_entry.set_date(today)
    end_date_entry.pack(side=tk.LEFT, padx=5)
    search_frame = ttk.Frame(root, padding=(10, 5))
    search_frame.pack(fill='x', padx=10)
    ttk.Label(search_frame, text="Пошук продукту:").pack(side=tk.LEFT,
padx=5)
    search_entry = ttk.Entry(search_frame, width=20)
    search_entry.pack(side=tk.LEFT, padx=5)
    def on_search(event):
        filter_text = search_entry.get()
        update_table(filter_text)
    search_entry.bind("<KeyRelease>", on_search)
    button_frame = ttk.Frame(root, padding=(10, 5))
    button_frame.pack(pady=10)
    update_button = ttk.Button(button_frame, text="Оновити дані",
command=lambda: update_table(search_entry.get()))
    update_button.grid(row=0, column=0, padx=5)
    chart_button = ttk.Button(button_frame, text="Показати графік",
command=show_chart)
    chart_button.grid(row=0, column=1, padx=5)
    view_reviews_button = ttk.Button(button_frame, text="Переглянути
відгуки", command=view_reviews)
    view_reviews_button.grid(row=0, column=2, padx=5)
    view_and_add_products_button = ttk.Button(button_frame,
text="Переглянути продукти", command=view_and_add_products)
    view_and_add_products_button.grid(row=0, column=3, padx=5)
    exit_button = ttk.Button(button_frame, text="Вихід",
command=on_closing)
    exit_button.grid(row=0, column=4, padx=5)
    table_frame = ttk.LabelFrame(root, text="Рейтинг продуктів",
padding=(10, 5))
    table_frame.pack(pady=10, padx=10, fill='both', expand=True)
    columns_en = ('product_name', 'positive_reviews', 'negative_reviews',
'rating')
```

```
    columns = ('Назва', 'Позитивні відгуки', 'Негативні відгуки',
'Рейтинг')
    table = ttk.Treeview(table_frame, columns=columns, show='headings')
    for col, col_en in zip(columns, columns_en):
        table.heading(col, text=col, command=lambda _col=col_en:
sort_by_column(_col))
    table.pack(fill='both', expand=True)table.pack(fill='both',
expand=True)
    current_sort_column = None
    descending = False
    root.protocol("WM_DELETE_WINDOW", on_closing)
    update_table()
    root.mainloop()
```

Б.2. Код «bot.py» для реалізації збору та аналізу через застосування чат-боту.

```
    import sqlite3
    from datetime import datetime
    from telegram import Update, ReplyKeyboardMarkup
    from telegram.ext import Application, CommandHandler, MessageHandler,
filters, ContextTypes, ConversationHandler
    from transformers import AutoTokenizer,
AutoModelForSequenceClassification
    import torch
    BOT_TOKEN = ''
    SELECT_PRODUCT, WRITE_REVIEW = range(2)
    print("Завантаження моделі для аналізу тональності...")
    tokenizer = AutoTokenizer.from_pretrained("nlptown/bert-base-
multilingual-uncased-sentiment")
    model =
AutoModelForSequenceClassification.from_pretrained("nlptown/bert-base-
multilingual-uncased-sentiment")
```

```
print("Модель завантажена.")
def init_db():
    conn = sqlite3.connect('obolon.db')
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS Products (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            product_name TEXT UNIQUE NOT NULL
        )
    """)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS Reviews (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            product_id INTEGER,
            review_text TEXT,
            sentiment TEXT,
            review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (product_id) REFERENCES Products (id)
        )
    """)
    conn.commit()
    conn.close()
def get_product_list():
    conn = sqlite3.connect('obolon.db')
    cursor = conn.cursor()
    cursor.execute("SELECT product_name FROM Products")
    products = [row[0] for row in cursor.fetchall()]
    conn.close()
    return products
def analyze_sentiment(review_text):
    inputs = tokenizer.encode_plus(review_text, return_tensors="pt",
truncation=True)
    with torch.no_grad():
        outputs = model(**inputs)
```

```
        scores = outputs.logits
        predictions = torch.softmax(scores, dim=1)
        rating = torch.argmax(predictions, dim=1).item() + 1
        if rating >= 4:
            return "positive"
        elif rating <= 2:
            return "negative"
        else:
            return "neutral"
    async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -
> int:
        products = get_product_list()
        if not products:
            await update.message.reply_text("Немає доступних продуктів
для відгуків.")
            return ConversationHandler.END
        reply_keyboard = [products]
        await update.message.reply_text(
            "Вітаємо! Оберіть продукт, для якого ви хочете залишити
відгук.",
            reply_markup=ReplyKeyboardMarkup(reply_keyboard,
one_time_keyboard=True)
        )
        return SELECT_PRODUCT
    async def select_product(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        product_name = update.message.text
        products = get_product_list()
        if product_name not in products:
            await update.message.reply_text("Невірний вибір продукту.
Будь ласка, оберіть знову.")
            return SELECT_PRODUCT
        context.user_data['product_name'] = product_name
```

```
        await update.message.reply_text("Напишіть ваш відгук про цей продукт.")
        return WRITE_REVIEW
    async def write_review(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int:
        review_text = update.message.text
        product_name = context.user_data['product_name']
        sentiment = analyze_sentiment(review_text)
        conn = sqlite3.connect('obolon.db')
        cursor = conn.cursor()
        cursor.execute("SELECT id FROM Products WHERE product_name = ?", (product_name,))
        product_id = cursor.fetchone()[0]
        cursor.execute("""
            INSERT INTO Reviews (product_id, review_text, sentiment)
            VALUES (?, ?, ?)
        """, (product_id, review_text, sentiment))
        conn.commit()
        conn.close()
        if sentiment == "positive":
            sentiment_text = "Дякуємо за позитивний відгук!"
        elif sentiment == "negative":
            sentiment_text = "Дякуємо за ваш відгук. Нам прикро, що ви не задоволені."
        else:
            sentiment_text = "Дякуємо за ваш відгук!"
        await update.message.reply_text(sentiment_text)
        return ConversationHandler.END
    async def cancel(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int:
        await update.message.reply_text("Відгук скасовано.")
        return ConversationHandler.END
    def main():
        init_db()
```

```
        app = Application.builder().token(BOT_TOKEN).build()
        conv_handler = ConversationHandler(
            entry_points=[CommandHandler("start", start)],
            states={
                SELECT_PRODUCT:       [MessageHandler(filters.TEXT       &
~filters.COMMAND, select_product)],
                WRITE_REVIEW:        [MessageHandler(filters.TEXT        &
~filters.COMMAND, write_review)],
            },
            fallbacks=[CommandHandler("cancel", cancel)],
        )
        app.add_handler(conv_handler)
        print("Бот запущений...")
        app.run_polling()
    if __name__ == "__main__":
        main()
```

Б.3. Код «x_extract.py» для здійснення збору та аналізу даних із соцмережі:

```
    import tweepy
    from    transformers    import    AutoModelForSequenceClassification,
AutoTokenizer
    import torch
    import sqlite3
    from datetime import datetime
    BEARER_TOKEN = ""
    client = tweepy.Client(bearer_token=BEARER_TOKEN)
    MODEL_NAME = "nlptown/bert-base-multilingual-uncased-sentiment"
    model                                                                =
AutoModelForSequenceClassification.from_pretrained(MODEL_NAME)
    tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
    conn = sqlite3.connect("obolon.db")
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS reviews (
```

```
                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                        review_text TEXT,
                        sentiment TEXT,
                        date_posted TEXT
                    )''')
    conn.commit()
    def analyze_sentiment(text):
        inputs = tokenizer(text, return_tensors="pt", truncation=True,
padding=True)
        with torch.no_grad():
            outputs = model(**inputs)
        scores = outputs.logits[0].softmax(dim=0)
        sentiment_score = torch.argmax(scores).item()
        return "позитивний" if sentiment_score >= 3 else "негативний"
    def save_review_to_db(text, sentiment, date_posted):
        cursor.execute("INSERT INTO reviews (review_text, sentiment,
date_posted) VALUES (?, ?, ?)",
                       (text, sentiment, date_posted))
        conn.commit()
    def get_review_dt(max_results = 10):
        cursor.execute("SELECT * FROM reviews LIMIT ?", (max_results, ))
        reviews = cursor.fetchall()
        for review in reviews:
            review_id, review_text, sentiment, date_posted = review
            print(f"ID: {review_id}\nТекст: {review_text}\nТональність:
{sentiment}\nДата: {date_posted}\n")
    def get_tweets_by_hashtag(hashtag, max_results=10):
        query = f"{hashtag} lang:uk"
        tweets = client.search_recent_tweets(query=query,
max_results=max_results, tweet_fields=["created_at"])
        for tweet in tweets.data:
            text = tweet.text
            date_posted = tweet.created_at.strftime("%Y-%m-%d %H:%M:%S")
            sentiment = analyze_sentiment(text)
```

```
            save_review_to_db(text, sentiment, date_posted)
            print(f"Твит:     {text}\nТональность:     {sentiment}\nДата:
{date_posted}\n")
    hashtag = "#оболоньвідгук"
    get_tweets_by_hashtag(hashtag)
    get_review_dt()
    conn.close()
```

Б.4. Код «dataInject.py» для створення БД проекту та її тестування:

```
    import sqlite3
    from datetime import datetime
    obolon_products = [
        "Оболонь Світле",
        "Оболонь Преміум",
        "Оболонь Біле",
        "Живчик",
        "Оболонь Безалкогольне",
        "Оболонь Квас",
        "Оболонь Чернігівське",
        "Оболонь Міллер",
        "Оболонь Гіннес"
    ]
    def create_products_table_and_insert_data():
        conn = sqlite3.connect('obolon.db')
        cursor = conn.cursor()
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS Products (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                product_name TEXT UNIQUE NOT NULL
            )
        """)
        cursor.executemany("INSERT OR IGNORE INTO Products (product_name)
VALUES (?)",
```

```
                        [(product,) for product in obolon_products])
    conn.commit()
    conn.close()
    print("Таблиця Products створена, і дані додані.")
def create_reviews_table():
    conn = sqlite3.connect('obolon.db')
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS Reviews (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            product_id INTEGER,
            review_text TEXT,
            sentiment TEXT,
            review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (product_id) REFERENCES Products (id)
        )
    """)
    conn.commit()
    conn.close()
    print("Таблиця Reviews створена.")
create_products_table_and_insert_data()
create_reviews_table()
```

Додаток В. Блок-схеми ключових функцій застосунку для збору та аналізу відгуків

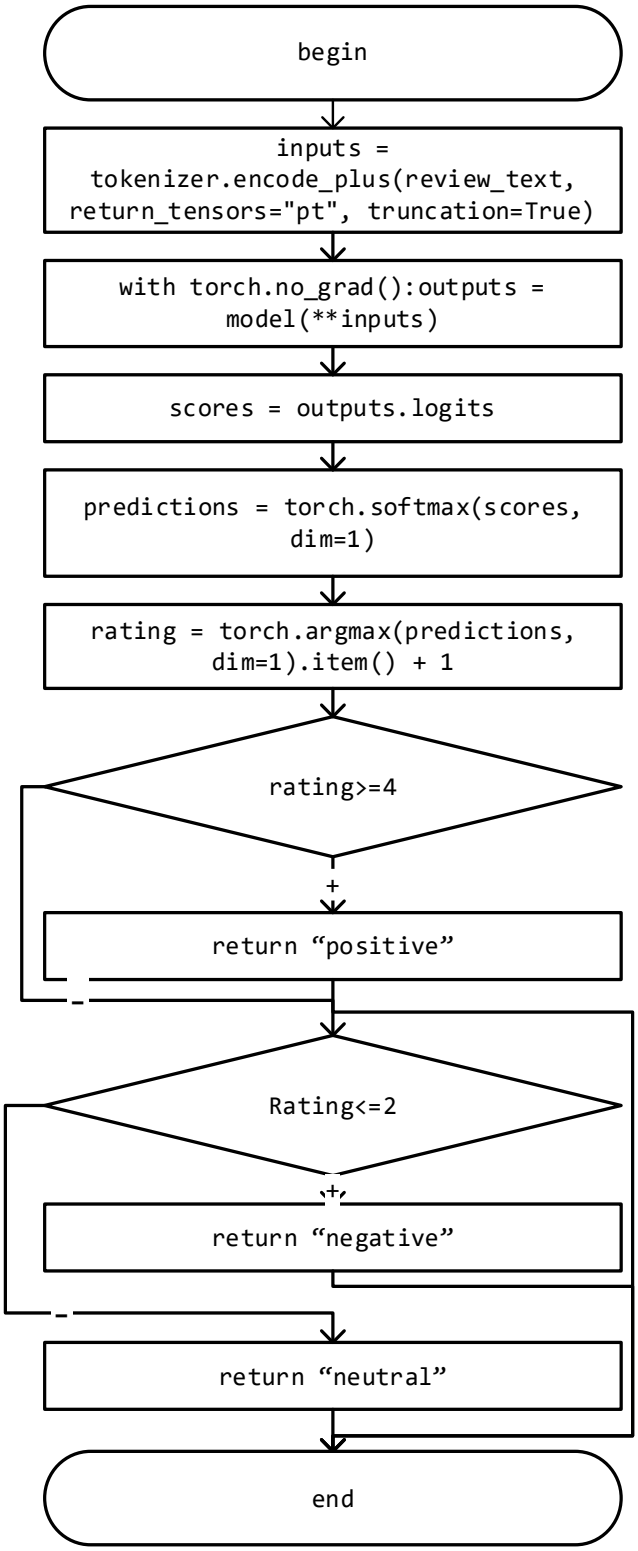


Рисунок В.1 – Блок-схема алгоритму аналізу тональності та класифікації тексту

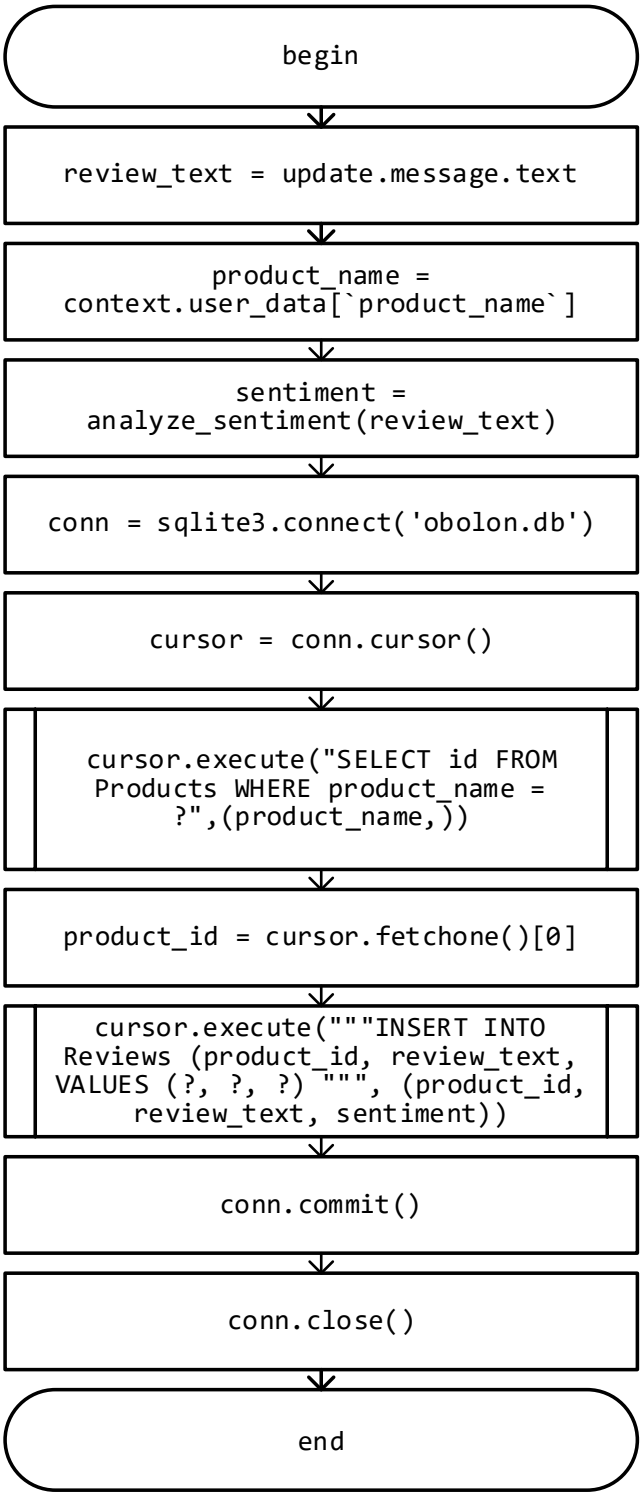


Рисунок В.2 – Блок-схема алгоритму збору відгуків за допомогою чат-боту

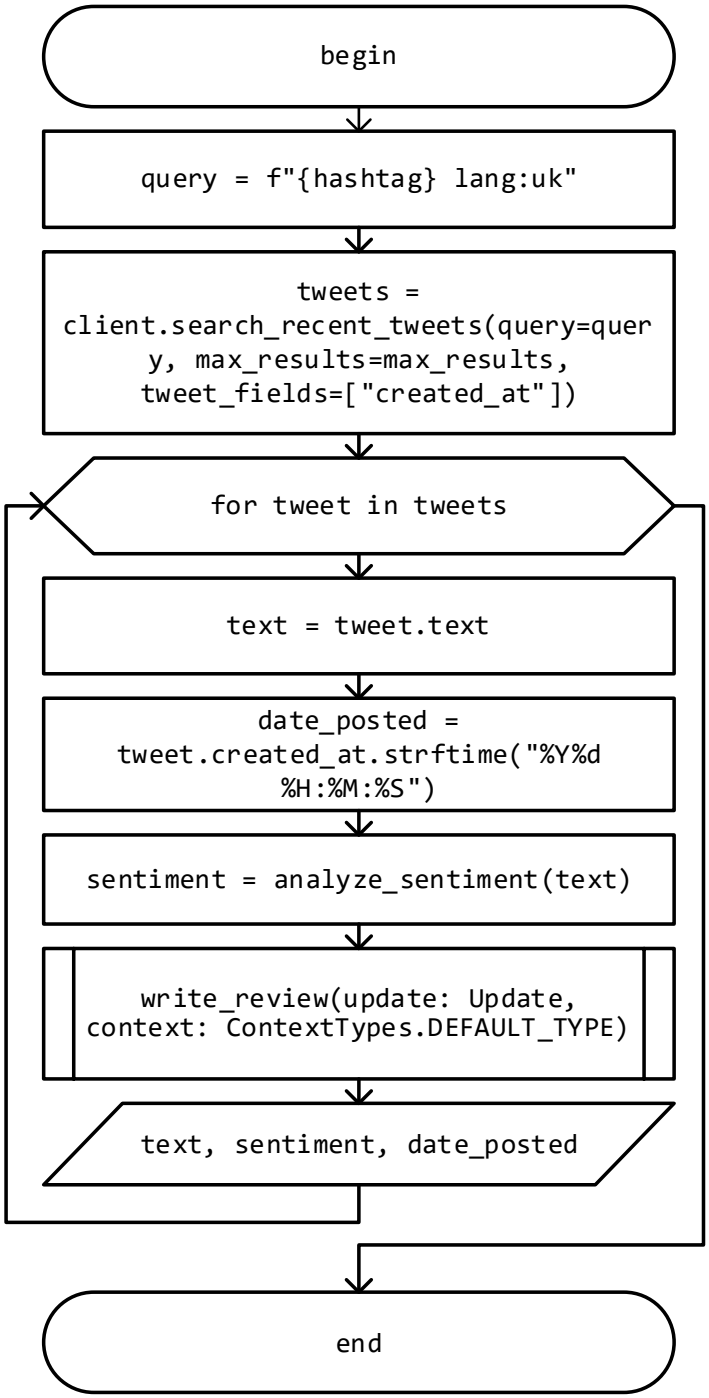


Рисунок В.3 – Блок-схема алгоритму збору відгуків за допомогою парсингу соціальної мережі

Додаток Г. Функціональна модель процесу формування звіту та рекомендацій щодо асортименту продукції на ПрАТ «Оболонь»

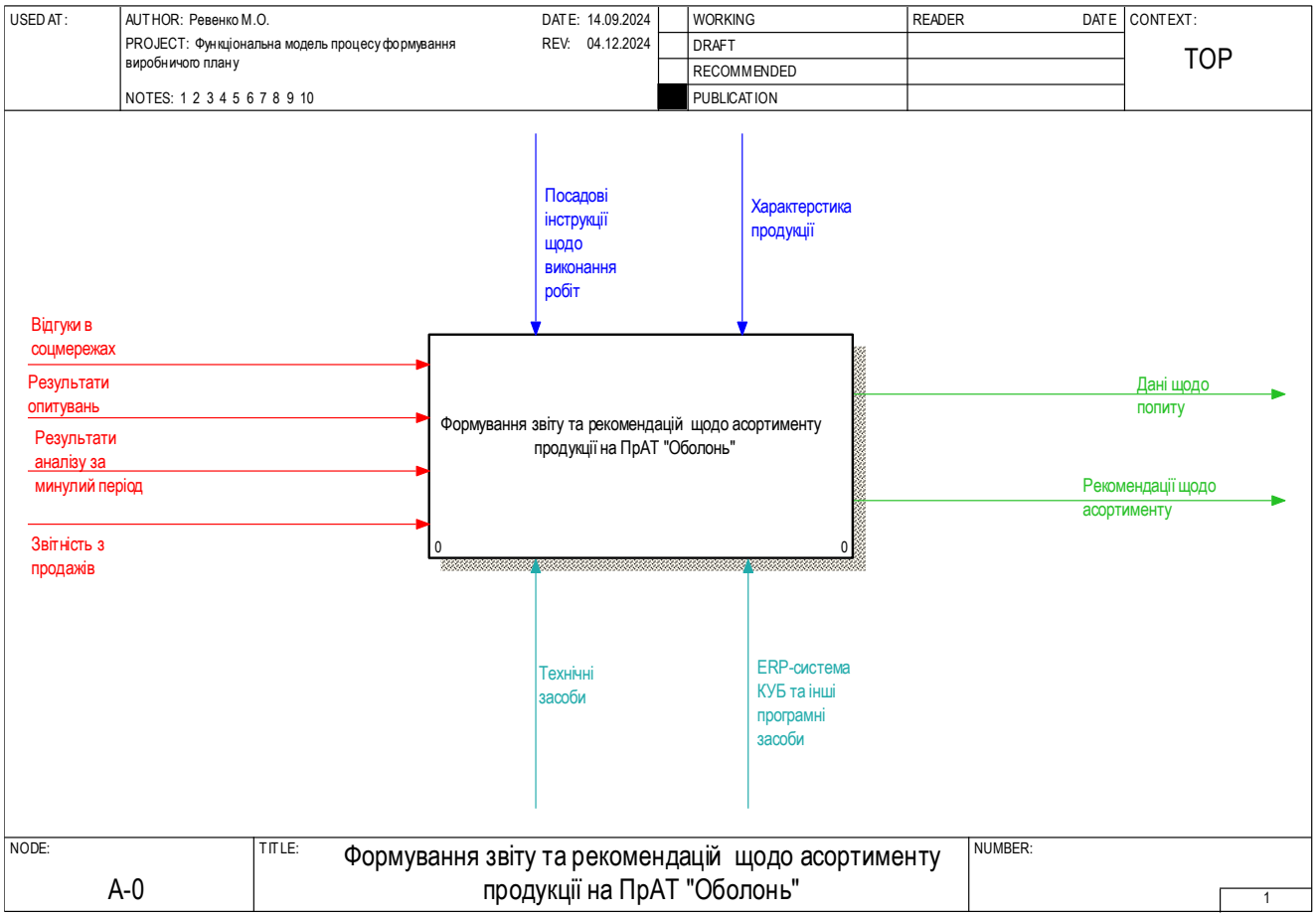


Рисунок Г.1 – Контекстна діаграма функціональної моделі процесу формування звіту та рекомендацій щодо асортименту продукції на ПрАТ «Оболонь»

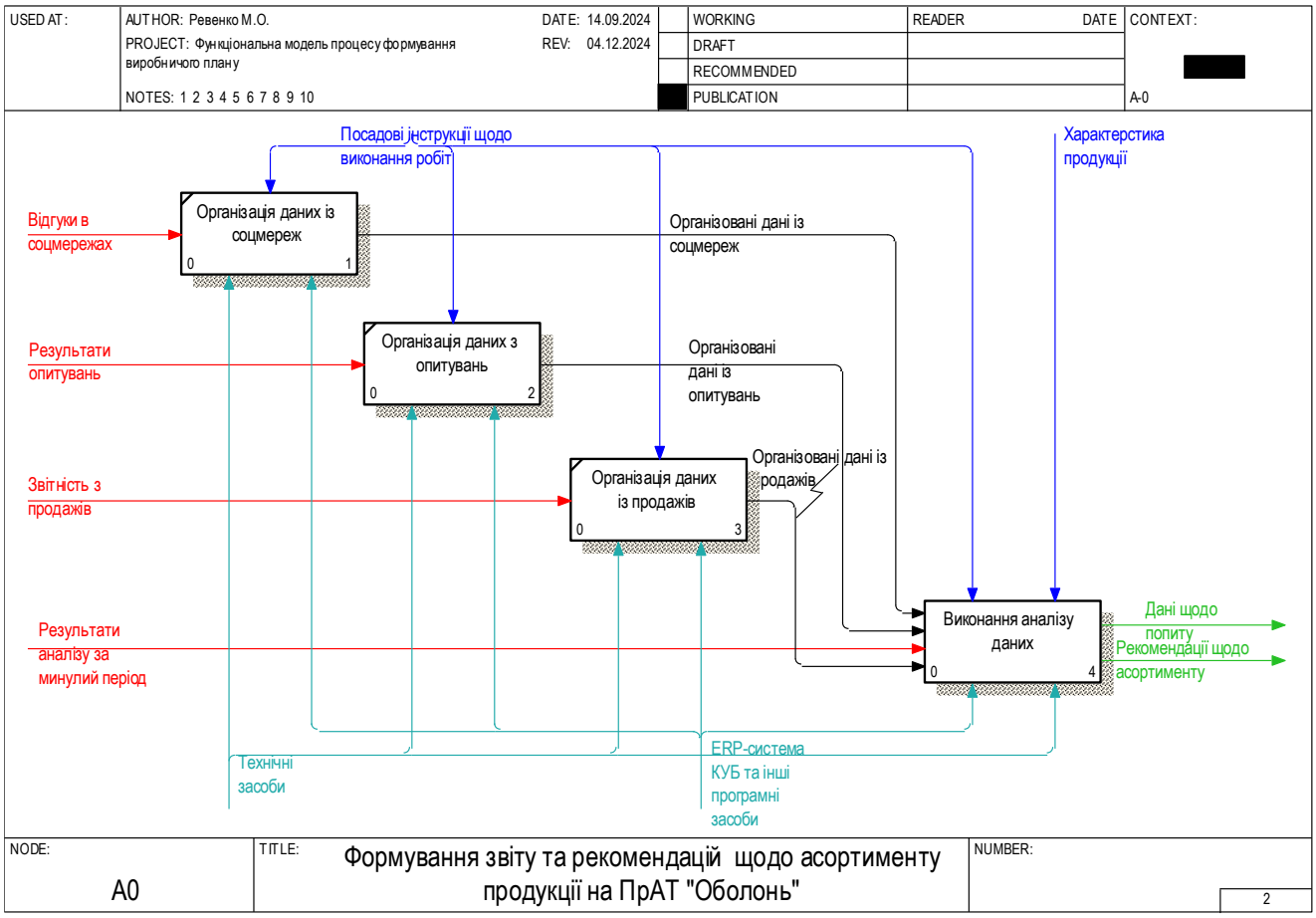


Рисунок Г.2 – Перший рівень декомпозиції – процесу «Формування звіту та рекомендацій щодо асортименту продукції на ПрАТ «Оболонь»

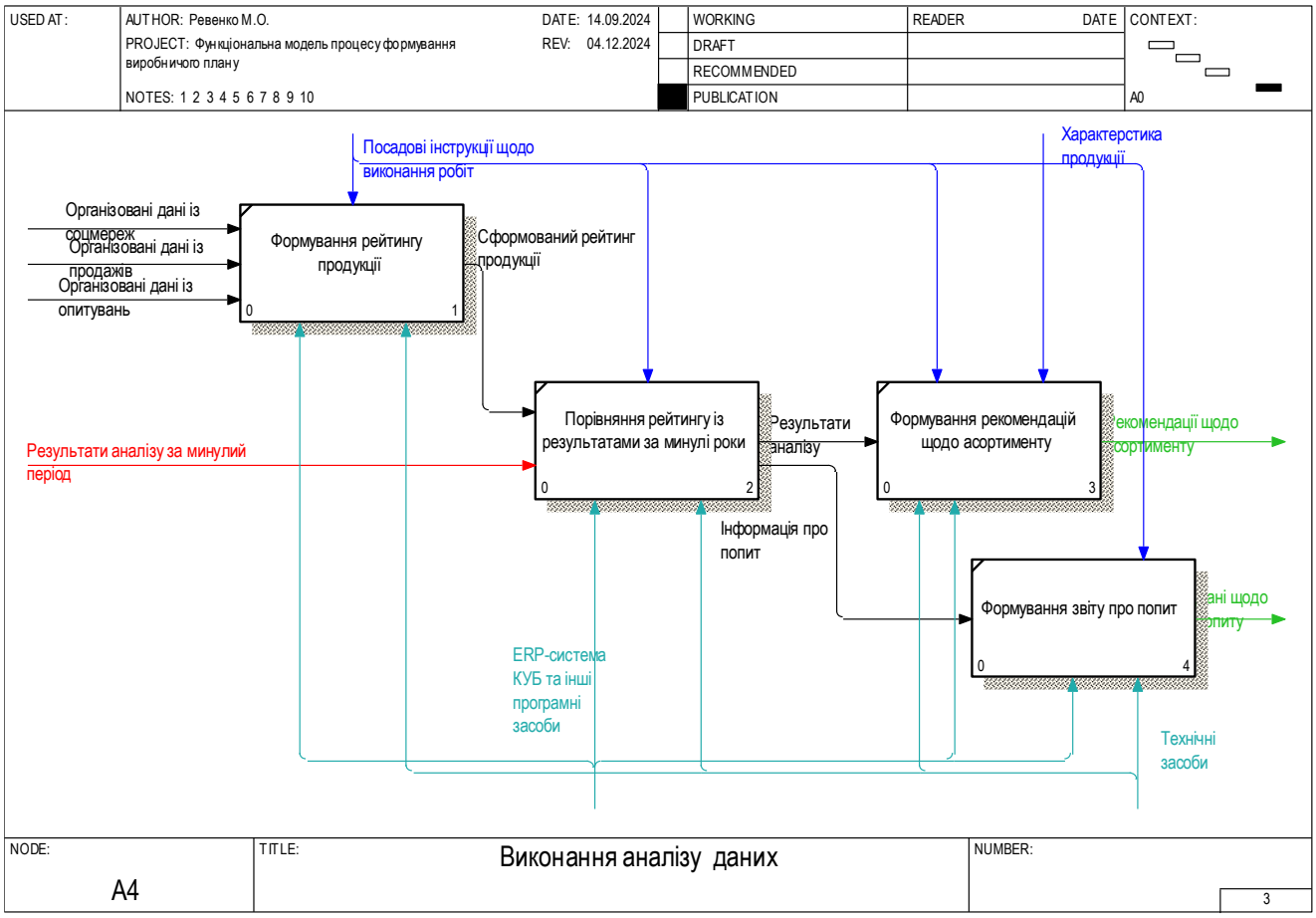


Рисунок Г.3 – Декомпозиція процесу «Виконання аналізу даних»

Додаток Д. Організаційна структура ПрАТ «Оболонь»

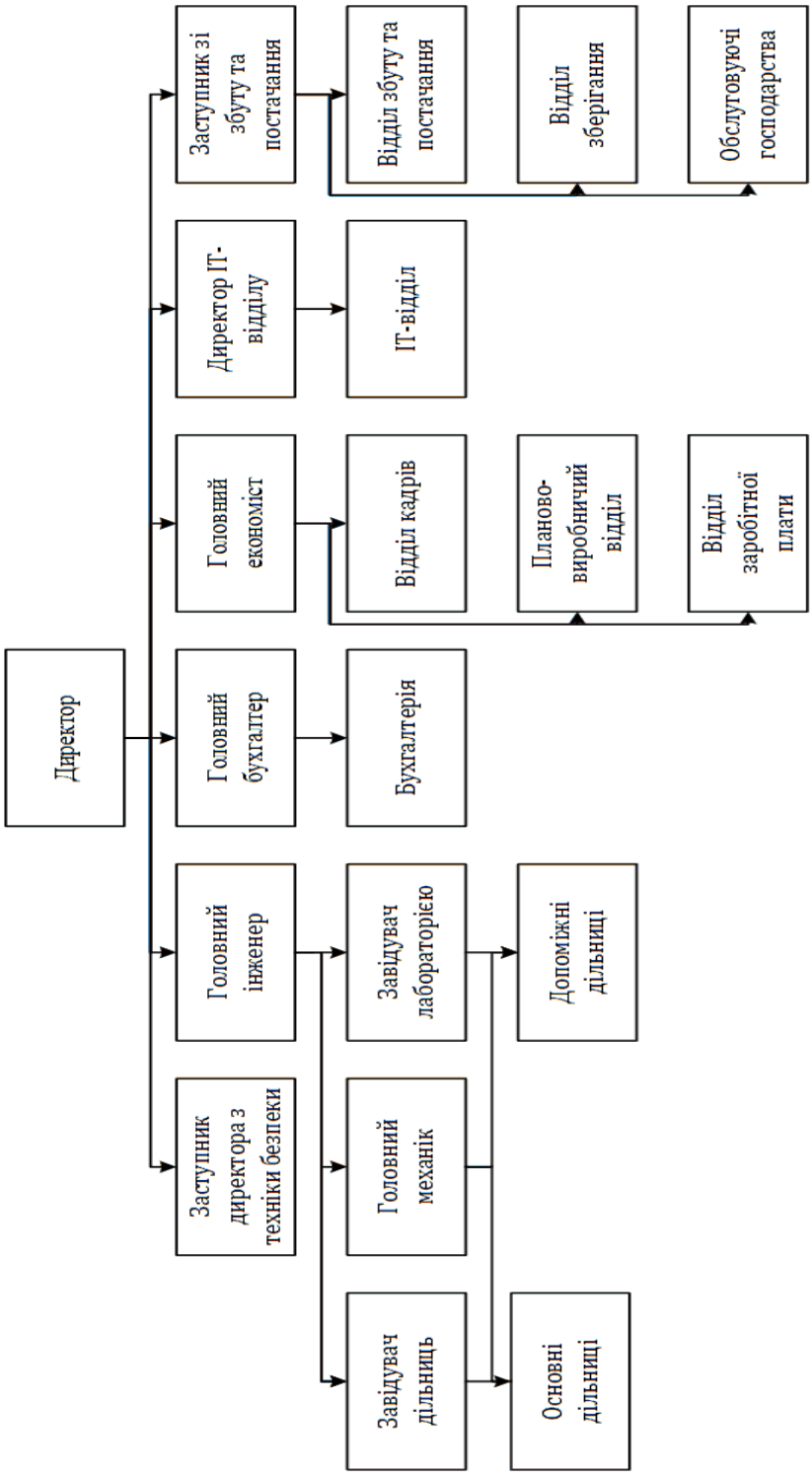


Рисунок Д.1 – Організаційна структура ПрАТ «Оболонь»