Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Програмний засіб аналізу сайтів з продажу побутової техніки

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Ніконенко В. С.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л. П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Ніконенку Віталію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмний засіб аналізу сайтів з продажу побутової техніки

Керівник роботи Дмитроца Леся Павлівна, кандидат технічних наук, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*14*» *травня* 2026 року № *4/9-239*

2. Термін подання студентом завершеної роботи 23 червня 2025 р.

3. Вихідні дані до роботи Літературні та інтернет-джерела з технологій розробки програмних систем, мови програмування Python, реляційних баз даних, методів веб-скрейпінгу та інтелектуального аналізу текстової інформації.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналітичний огляд. 1.1. Аналіз ринку електронної комерції та специфіка даних. 1.2. Огляд та порівняння методів веб-скрейпінгу. 1.3. Методи інтелектуальної обробки даних: концепція Entity Matching. 1.4. Огляд сучасних підходів до аналізу споживчих настроїв (Sentiment Analysis). 1.5. Формулювання вимог до розроблюваної системи. 1.6. Висновки до першого розділу. 2. Проєктування та розробка програмного засобу. 2.1. Обґрунтування вибору технологічного стеку. 2.2. Проєктування архітектури системи. 2.3. Проєктування бази даних системи. 2.4. Алгоритмічне забезпечення системи: логіка Entity Matching. 2.5. Висновки до другого розділу. 3. Практична реалізація та результати роботи. 3.1. Структура програмного продукту та опис середовища розробки. 3.2. Реалізація модуля збору даних (Data Acquisition Layer). 3.3. Реалізація алгоритму Entity Matching. 3.4. Реалізація модуля аналізу тональності відгуків (NLP). 3.5. Реалізація модуля візуалізації та аналітичного дашборду. 3.6. Реалізація модуля формування звітності. 3.7. Реалізація графічного інтерфейсу користувача. 3.8. Тестування та оцінка ефективності системи. 3.9. Висновки до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. 4.1. Діяльність. Її види та розуміння в безпеці праці. 4.2. Суть та зміст управління охороною праці та безпекою життєдіяльності. 4.3. Висновки до четвертого розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Мета. 3. Актуальність теми. 4. Аналіз предметної області. 5. Архітектура. 6. Технології. 7. Програмна реалізація. 8. Графічний інтерфейс. 9. Тестування. 10. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О. Я. кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2026р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.05.2025	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	27.01.2026-16.02.2026	*Виконано*
3.	Виконання дослідження щодо розробки програмного засобу аналізу сайтів з продажу побутової техніки.	18.02.2026-17.05.2026	*Виконано*
	Розроблення функціоналу програмного комплексу		
4.	Оформлення розділу «Аналітичний огляд»	19.05.2026-22.05.2026	*Виконано*
5.	Оформлення розділу «Проєктування та розробка	24.05.2026-28.05.2026	*Виконано*
	програмного засобу»		
6.	Оформлення розділу «Практична реалізація та	28.05.2026-31.05.2026	*Виконано*
	результати роботи»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	01.06.2026-06.06.2026	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	01.06.2026-06.06.2026	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	06.06.2026-09.06.2026	*Виконано*
10.	Нормоконтроль		*Виконано*
11.	Перевірка на плагіат		*Виконано*
12.	Попередній захист кваліфікаційної роботи		*Виконано*
13.	Захист кваліфікаційної роботи	24.06.2026	

Студент _____ Ніконенко В. С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Програмний засіб аналізу сайтів з продажу побутової техніки // Кваліфікаційна робота освітнього рівня «Бакалавр» // Ніконенко Віталій Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2026 // С. 59, рис. – 14, табл. – 4, додат. – 3, бібліогр. – 38.

Ключові слова: веб-парсинг, аналіз даних, автоматизація, побутова техніка, порівняльний аналіз, машинне навчання, моніторинг цін.

Кваліфікаційна робота присвячена дослідженню методів та розробці програмного забезпечення для автоматизованого збору й інтелектуального аналізу даних з інтернет-магазинів побутової техніки. У першому розділі описано сучасний стан технологій веб-скрейпінгу, висвітлено проблеми неструктурованих даних в електронній комерції, розглянуто аналоги рішень для моніторингу цін та принципи побудови архітектури систем обробки великих масивів даних. Другий розділ містить обґрунтування технологічного стеку, дослідження методів зіставлення товарних позицій, а також опис структури бази даних та алгоритмів нормалізації інформації. У третьому розділі представлено програмну реалізацію системи, проаналізовано ефективність алгоритмів аналізу тональності відгуків та наведено результати тестування швидкодії й точності збору даних.

Об'єкт дослідження: процес збору та аналізу даних з веб-ресурсів електронної комерції. **Предмет дослідження:** методи, алгоритми та програмні засоби для моніторингу, порівняння цін та оцінки споживчих характеристик товарів побутової техніки.

ANNOTATION

Software tool for analyzing household appliance retail websites // Qualification work of the educational level «Bachelor» // Nikonenko Vitalii Serhiiovych // Ternopil Ivan Puluj National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-42 // Ternopil, 2026 // P. 59, fig. – 14, tabl. – 4, annexes. – 3, references – 38.

Keywords: web scraping, data analysis, automation, household appliances, comparative analysis, machine learning, price monitoring.

The qualification work is devoted to the research of methods and the development of software for automated data collection and intellectual analysis from household appliance online stores. The first chapter describes the current state of web scraping technologies, highlights the problems of unstructured data in e-commerce, examines existing software solutions for price monitoring, and analyzes the architectural principles of systems for processing large datasets. The second chapter provides the justification for the choice of the technology stack, researches methods for matching product items from various sources, and describes the database structure and data normalization algorithms. The third chapter presents the software implementation of the system, analyzes the efficiency of sentiment analysis algorithms for customer reviews, and provides the results of performance and data collection accuracy testing.

Object of research: the process of data collection and analysis from e-commerce web resources.

Subject of research: methods, algorithms, and software tools for monitoring, comparing prices, and evaluating consumer characteristics of household appliance products.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – програмний інтерфейс застосунку – набір правил, що визначають взаємодію між програмними компонентами

BERT (Bidirectional Encoder Representations from Transformers) – двонаправлена трансформерна модель обробки природної мови

BOM (Byte Order Mark) – маркер порядку байтів у файлі, що визначає кодування тексту

BS4 – BeautifulSoup4 – бібліотека мови Python для парсингу та обробки HTML/XML документів

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) – автоматизований тест для розрізнення людини та комп'ютерної програми

CPU (Central Processing Unit) – центральний процесор – основний обчислювальний компонент комп'ютера

CSS (Cascading Style Sheets) – мова таблиць стилів для опису зовнішнього вигляду HTML-документів

CSV (Comma-Separated Values) – формат текстових файлів для табличного представлення даних із роздільником

DOM (Document Object Model) – об'єктна модель документа – програмний інтерфейс для HTML та XML

ETL (Extract, Transform, Load) – процес вилучення, перетворення та завантаження даних

GUI (Graphical User Interface) – графічний інтерфейс користувача – спосіб взаємодії з програмою через графічні елементи

HTML (HyperText Markup Language) – мова гіпертекстової розмітки для створення веб-сторінок

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту – основний протокол обміну даними у мережі Інтернет

IP (Internet Protocol) – інтернет-протокол – протокол мережевого рівня для маршрутизації пакетів

JS (JavaScript) – мова програмування для створення динамічного контенту на веб-сторінках

NLP (Natural Language Processing) – обробка природної мови – галузь штучного інтелекту, що вивчає методи аналізу та генерації тексту

ORM (Object-Relational Mapping) – об'єктно-реляційне відображення – технологія зв'язку об'єктів із реляційними базами даних

Precision – точність – метрика оцінки класифікатора, частка правильно знайдених позитивних прикладів

Recall – повнота – метрика оцінки класифікатора, частка знайдених позитивних прикладів від їх загальної кількості

SKU (Stock Keeping Unit) – складська облікова одиниця – унікальний ідентифікатор товарної позиції

SQL (Structured Query Language) – мова структурованих запитів для керування реляційними базами даних

SQLite – компактна реляційна система керування базами даних, що не потребує окремого серверного процесу

SVM (Support Vector Machine) – метод опорних векторів – алгоритм машинного навчання для класифікації та регресії

UI (User Interface) – інтерфейс користувача – засоби взаємодії між людиною та програмним забезпеченням

URL (Uniform Resource Locator) – уніфікований локатор ресурсів – адреса ресурсу в мережі Інтернет

UTF-8 (Unicode Transformation Format) – формат кодування символів Unicode змінної довжини

UX (User Experience) – досвід користувача – сукупність відчуттів під час взаємодії з програмним продуктом

Українські скорочення та терміни

БД – база даних – організована сукупність структурованих даних

БЖД – безпека життєдіяльності – наука про безпечну взаємодію людини з навколишнім середовищем

ВДТ – відеодисплейний термінал – пристрій відображення графічної та текстової інформації

ІТ – інформаційні технології – галузь, що охоплює методи та засоби обробки інформації

КЗпП – Кодекс законів про працю України

НЛП / NLP – обробка природної мови – галузь штучного інтелекту

ОП – охорона праці – система правових, організаційних та технічних заходів щодо безпеки праці

ПЗ – програмне забезпечення – сукупність програм та даних для комп'ютерних систем

ПК – персональний комп'ютер

СКБД – система керування базами даних – програмне забезпечення для створення та управління базами даних

СУОП – система управління охороною праці – комплекс організаційно-методичних заходів з охорони праці

ШІ – штучний інтелект – галузь інформатики, що займається розробкою інтелектуальних машинних систем

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок глобалізації процесів електронної комерції, кількість онлайн-магазинів побутової техніки стрімко зростає, що призводить до накопичення величезних масивів неструктурованих даних. Для сучасного споживача вибір оптимального товару за співвідношенням ціни та якості стає складним завданням, що потребує значних часових витрат на порівняння пропозицій на різних платформах. З іншого боку, для ритейлерів критично важливим є моніторинг конкурентного середовища для динамічного ціноутворення. Тому розробка інтелектуального програмного засобу, здатного автоматизувати збір, порівняння та аналіз даних про побутову техніку, є актуальним напрямком сучасних досліджень в галузі інформаційних технологій та аналізу даних.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення якості та швидкості процесу прийняття рішень при виборі побутової техніки шляхом розробки програмного засобу для інтелектуального моніторингу та аналізу ринкових пропозицій.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- проаналізувати сучасний стан досліджень у сфері веб-скрейпінгу та методів автоматизованого збору даних з веб-ресурсів електронної комерції;
- дослідити алгоритми зіставлення товарних позицій (entity matching) та методи аналізу тональності відгуків (sentiment analysis);
- обґрунтувати вибір технологічного стеку для побудови архітектури системи аналізу товарів;
- розробити програмний засіб, що забезпечує автоматизоване завантаження, нормалізацію та зберігання даних з інтернет-магазинів;
- провести тестування розробленого програмного засобу на предмет швидкодії, точності зіставлення товарів та зручності використання результатів аналізу.

Практичне значення одержаних результатів. Розроблений програмний засіб може бути використаний як інструмент для кінцевих споживачів з метою порівняльного аналізу цін та реальних відгуків, а також як основа для систем конкурентної розвідки в сфері електронної комерції. Впровадження системи дозволяє мінімізувати час на пошук техніки, виявити приховані націнки та отримати об’єктивну оцінку товару на основі аналізу відгуків користувачів.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз ринку електронної комерції та специфіка даних

Сфера електронної комерції в Україні та світі демонструє стабільне зростання [1], а сегмент побутової техніки є одним із найбільш конкурентних та насичених товарними позиціями. За останні роки відбулася суттєва зміна споживчих звичок: значна частина покупок перемістилася в онлайн-простір, що зробило сайти інтернет-магазинів основними майданчиками для вибору та придбання товарів [2]. Сучасний покупець, плануючи придбання побутової техніки, стикається з проблемою вибору серед тисяч пропозицій, розсіяних по десятках сайтів-ритейлерів, що вимагає від нього значних зусиль для порівняння цін та характеристик.

Основні особливості ринку, що зумовлюють потребу в автоматизації аналізу:

- висока динамічність цін, тобто вартість товарів у сфері побутової техніки не є статичною. Ціни можуть змінюватися кілька разів на добу через внутрішню маркетингову політику магазинів, участь у розпродажах, вихід нових моделей або коливання курсу валют. Відстеження цих змін вручну стає практично неможливим завданням;
- неструктурований характер даних. Інтернет-магазини часто не дотримуються єдиних стандартів опису товарів. Назви однієї і тієї ж моделі можуть суттєво відрізнятися (використання різних префіксів, артикулів, скорочень, транслітерації тощо). Це створює проблему "розрізненості даних" (data silos), коли одна модель холодильника на сайті А виглядає як "Bosch KGN39", а на сайті Б – як "Холодильник двокамерний Bosch KGN-39";
- великий обсяг даних. Асортимент великих торгових майданчиків нараховує десятки тисяч товарних позицій. Пошук інформації в таких обсягах вимагає використання високоефективних алгоритмів обробки даних, оскільки звичайні методи навігації стають неефективними;

- інформаційна асиметрія та якість відгуків. Споживачі все частіше спираються на відгуки інших покупців. Однак, наявність "фейкових" або рекламних відгуків, а також маніпуляції з рейтингами, викривлюють реальну картину якості товару. Аналіз цих даних потребує застосування методів обробки природної мови (NLP);
- технічні бар'єри. Багато сайтів активно протидіють автоматизованому збору даних (використання CAPTCHA, блокування IP, динамічна підгрузка контенту через JavaScript), що робить процес збору технічно складним завданням.

Отже, ринок електронної комерції побутової техніки перебуває у стані постійної інформаційної надлишковості. Для ефективного функціонування в таких умовах як покупцям, так і аналітикам, необхідний інструмент, здатний інтегрувати дані з різних джерел у єдину систему. Автоматизація збору даних дозволяє перетворити великі масиви розрізненої інформації у структуровані звіти, що є критично важливим для прийняття зважених рішень. Відсутність єдиного стандарту опису товарів спонукає до розробки спеціалізованих алгоритмів нормалізації та співставлення (entity matching), які стануть інтелектуальною основою запропонованого програмного засобу.

1.2 Огляд та порівняння методів веб-скрейпінгу

Процес отримання даних із веб-ресурсів, відомий як веб-скрейпінг (web scraping) [3], є критично важливим етапом функціонування системи. Вибір інструментарію залежить від архітектури цільового сайту, обсягу даних та наявності захисних механізмів від автоматизованого доступу.

Сучасні інструменти для парсингу можна класифікувати на кілька категорій:

1. Бібліотеки для обробки статичного HTML, яскравим прикладом яких є BeautifulSoup [4], працюють за принципом завантаження вихідного коду сторінки з подальшим пошуком потрібних елементів за допомогою CSS-

селекторів або XPath. Головними перевагами такого підходу є висока швидкість роботи та низькі вимоги до оперативної пам’яті, що робить їх чудовим інструментом для швидкого парсингу. Проте вони мають і суттєвий недолік: такі бібліотеки не здатні обробляти контент, який динамічно підвантажується через JavaScript, наприклад, ціни чи специфікації, що з’являються лише після повного завантаження сторінки в браузері.

2. Інструменти браузерної автоматизації, такі як Selenium, Playwright [6] або Puppeteer, працюють за принципом імітації поведінки реального користувача, безпосередньо виконуючи JavaScript-код для взаємодії зі сторінкою через кліки, скролінг та введення тексту. Ключовою перевагою цього методу є можливість успішно збирати дані з найбільш «складних» та динамічних сайтів, побудованих на сучасних фреймворках на кшталт React, Angular або Vue.js. Водночас головним недоліком таких інструментів є високе навантаження на систему та відносно повільна швидкість роботи, якщо порівнювати їх із класичними парсерами статичних запитів.

3. Фреймворки для масштабованого парсингу, яскравим представником яких є Scrapy, функціонують як асинхронні системи, що дозволяють паралельно й ефективно обробляти велику кількість мережевих запитів. Завдяки такій архітектурі їхньою головною перевагою є висока продуктивність, що робить цей підхід найкращим рішенням для великих проєктів, де необхідно зібрати дані про десятки тисяч товарів або сторінок за мінімальний проміжок часу.

Для реалізації програмного засобу аналізу сайтів з продажу побутової техніки пропонується гібридний підхід. Оскільки більшість сучасних інтернет-магазинів активно використовують JavaScript для відображення цін та характеристик, використання виключно статичних бібліотек є недостатнім. Проте, застосування браузерної автоматизації для кожного запиту є економічно недоцільним через високе споживання ресурсів. Таким чином, оптимальною архітектурою є поєднання швидких HTTP-запитів для отримання загальної структури даних та використання інструментів типу Playwright для обробки сторінок, де контент генерується динамічно.

Такий підхід дозволить досягти балансу між швидкодією системи та повнотою зібраних даних, забезпечуючи високу точність моніторингу товарних позицій. В таблиці 1.1 зображено порівняльний аналіз інструментів для веб-скрейпінгу.

Таблиця 1.1 – Порівняльний аналіз інструментів для веб-скрейпінгу

Назва інструмента	**Тип інструмента**	**Швидкість роботи**	**Підтримка JavaScript**	**Складність налаштування**
BeautifulSoup	Статичний парсер	Дуже висока	Відсутня	Низька
Scrapy	Асинхронний фреймворк	Висока	Обмежена	Середня
Selenium	Браузерна автоматизація	Низька	Повна	Висока
Playwright	Браузерна автоматизація	Середня	Повна	Середня

Як свідчать дані таблиці 1.1, для вирішення завдань збору даних з динамічних сайтів інтернет-магазинів найбільш доцільним є використання комбінованого підходу [7], що поєднує швидкість BeautifulSoup для статичного контенту та потужність Playwright для взаємодії з динамічними елементами сторінок.

1.3 Методи інтелектуальної обробки даних

Однією з ключових проблем при побудові систем моніторингу товарів є відсутність єдиного ідентифікатора для одного й того самого об'єкта в різних інтернет-магазинах. Ця задача в теорії інформаційних систем відома як Entity Resolution, Entity Matching або Record Linkage [8]. Суть проблеми полягає в тому, що назва «Праска Bosch TDA 5028010» на одному сайті та «Bosch

TDA5028010 2800Вт» на іншому технічно є різними рядками (strings), але фактично посилаються на один товар. Для ефективного порівняння цін система повинна вміти розпізнавати такі відповідності, що реалізується за допомогою кількох основних підходів.

Перший підхід базується на алгоритмах обчислення текстових дистанцій (String Metrics), серед яких активно застосовується коефіцієнт Жаккара [9] для порівняння наборів слів (n-grams), що дозволяє ігнорувати порядок їх розташування в специфікаціях побутової техніки. Іншим поширеним методом у цій категорії є відстань Левенштейна [10], яка вимірює кількість операцій вставки, видалення або заміни символів, необхідних для перетворення одного рядка в інший.

Другий підхід полягає в семантичному аналізі на основі векторних моделей (Vector Space Model & Embeddings). Використання сучасних моделей типу Word2Vec або архітектур на базі Transformer дозволяє перетворювати назви товарів у багатовимірні вектори, завдяки чому схожі позиції матимуть близько розташовані координати в просторі, навіть за відсутності спільних слів у текстових назвах.

Для оптимізації цього процесу в роботі пропонується впровадження гібридного підходу, який комбінує попередню фільтрацію та уточнююче порівняння. На першому етапі система відбирає потенційно схожі товари за категорією або брендом, а на другому – застосовує алгоритм Левенштейна або косинусну подібність (cosine similarity) для векторів назв. Комплексне використання методів Entity Matching дозволяє системі автоматично зводити розрізнені дані до єдиного «майстер-каталогу», що уможливлює проведення точного порівняльного аналізу цін у розрізі ідентичних моделей побутової техніки, незалежно від відмінностей у їхньому найменуванні на різних маркетплейсах. У підсумку, автоматизація цього процесу суттєво підвищує точність аналітичних звітів та мінімізує вплив людського фактора при обробці великих масивів комерційних даних.

1.4 Огляд сучасних підходів до аналізу споживчих настроїв (Sentiment Analysis)

В умовах насиченого ринку побутової техніки технічні характеристики (потужність, габарити, енергоефективність) часто є схожими у різних виробників. У таких випадках вирішальним фактором при виборі товару стає думка інших користувачів, виражена у відгуках. Проте ручний аналіз сотень коментарів є неможливим для пересічного покупця, що зумовлює потребу в автоматизації цього процесу.

Аналіз тональності [11] (Sentiment Analysis) – це галузь обробки природної мови (NLP), яка дозволяє автоматично визначати емоційне забарвлення тексту. Для нашого програмного засобу це означає можливість перетворити великий масив суб'єктивних коментарів у кількісну оцінку "задоволеності" конкретною моделлю техніки.

Основні підходи до аналізу:

1. Лексичний підхід (Lexicon-based) – використання готових словників слів з визначеним "емоційним ваговим коефіцієнтом" (наприклад, слова "відмінно", "надійний" – мають позитивну оцінку, а "зламався", "шумний" – від'ємну). Це швидкий метод, який добре працює на загальних даних, але часто не враховує контекст (наприклад, "не дуже шумний" може бути помилково сприйнятий як негатив).

2. Машинне навчання [12] (Machine Learning) – використання класифікаторів (таких як Naive Bayes, SVM), які навчаються на розмічених датасетах. Система "вчиться" розпізнавати патерни в реченнях, характерні для позитивних чи негативних відгуків.

3. Глибоке навчання [13] (Deep Learning) – використання трансформерних моделей (BERT, RoBERTa), які здатні розуміти контекст, іронію та складні синтаксичні конструкції. Це найбільш точний, але й найбільш ресурсомісткий метод.

Враховуючи вимоги до продуктивності програмного засобу, для реалізації аналізу відгуків про побутову техніку доцільно використати гібридний підхід. Лексичний аналіз застосовують для швидкого підрахунку частоти згадувань ключових проблемних слів (наприклад, "екран", "батарея", "якість збірки") у поєднанні зі спрощеною класифікацією тональності [14]. Це дозволить надати користувачу не лише загальний рейтинг, а й "карту проблем" конкретної моделі, що значно підвищує цінність нашого програмного засобу порівняно з простими моніторами цін. Впровадження такого комплексного функціоналу є практичним відображенням сучасних трендів аналітики великих даних у прикладних системах [15], де головним завданням стає перетворення неструктурованих масивів тексту на конкретні та дієві інсайти.

1.5 Формулювання вимог до розроблюваної системи

На основі проведеного аналізу предметної області, вивчення інструментарію веб-скрейпінгу та методів інтелектуальної обробки даних, можна сформулювати концептуальні вимоги до програмного засобу:

1. Модульність. Архітектура системи повинна базуватися на розділенні функціональних блоків (модуль збору даних, модуль обробки та модуль аналізу), що дозволить вносити зміни до кожного з них без порушення цілісності всієї системи.

2. Масштабованість. Програма повинна підтримувати можливість додавання нових інтернет-магазинів через оновлення конфігураційних файлів або селекторів, без необхідності суттєвого переписування коду.

3. Інтелектуальність. Для забезпечення коректного порівняння цін реалізація механізму нормалізації назв товарів (Entity Matching) є обов’язковою. Це дозволить нівелювати розбіжності у представленні товарів на різних сайтах.

4. Аналітична глибина. Система повинна забезпечувати не тільки моніторинг цінових показників, а й оцінку споживчих настроїв шляхом

базового аналізу тональності відгуків, що значно розширює корисність отриманих результатів.

5. Зручність використання. Результати роботи системи повинні бути представлені у структурованому вигляді (таблиці, порівняльні звіти), що дозволить кінцевому користувачу швидко ідентифікувати найбільш вигідні пропозиції на ринку.

Таким чином, розробка програмного засобу, що поєднує ці технічні та інтелектуальні компоненти, відповідає сучасним вимогам до систем моніторингу E-commerce та здатна вирішити визначену проблему вибору побутової техніки.

1.6 Висновки до першого розділу

У першому розділі кваліфікаційної роботи проведено комплексний аналіз предметної області. Досліджено специфіку даних у сфері електронної комерції та визначено основні виклики, з якими стикаються користувачі та розробники аналітичних систем. До таких викликів належать: динамічність ціноутворення, відсутність єдиних стандартів опису товарів, наявність інформаційного шуму та активна протидія автоматизованому збору даних з боку веб-ресурсів.

Проведений порівняльний аналіз сучасних методів веб-скрейпінгу довів доцільність застосування гібридного підходу до розробки програмного засобу. Використання статичних парсерів (наприклад, BeautifulSoup) у поєднанні з інструментами браузерної автоматизації (Playwright) дозволяє забезпечити баланс між швидкодією системи та повнотою збору даних з динамічних веб-ресурсів.

Окрему увагу приділено методам інтелектуальної обробки даних, зокрема алгоритмам зіставлення товарів (Entity Matching) та аналізу тональності відгуків (Sentiment Analysis). Обґрунтовано, що застосування алгоритмів на основі відстані Левенштейна та методів обробки природної мови є критично

необхідним для забезпечення точності порівняльного аналізу та надання користувачу об’єктивної інформації про товар.

На основі отриманих результатів сформульовано технічні вимоги до системи, які передбачають модульність, масштабованість, інтелектуальність алгоритмів обробки та зручність представлення аналітичних звітів. Виконаний теоретичний аналіз створює надійне підґрунтя для переходу до етапу проектування архітектури та безпосередньої програмної реалізації системи, що буде розглянуто в наступних розділах роботи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

2.1 Обґрунтування вибору технологічного стеку

Вибір технологічного стеку є критичним етапом проектування програмного засобу, оскільки від нього залежить масштабованість, швидкість роботи системи та її стійкість до змін у структурі веб-ресурсів. Для реалізації поставлених завдань було обрано мову програмування Python [16] версії 3.12.

Вибір зумовлений її високою популярністю в академічному та промисловому середовищі [17], багатим набором спеціалізованих бібліотек для обробки даних (Data Science) та інструментів для автоматизації (Web Automation). Згідно з індексом TIOBE станом на 2024 рік, Python посідає першу позицію серед найпоширеніших мов програмування у світі. У порівнянні з іншими мовами програмування (наприклад, Java або C++), Python забезпечує значно швидший цикл розробки [18]. Завдяки інтерпретованій природі мови та динамічній типізації, розробка складних алгоритмів парсингу та аналізу даних стає менш трудомісткою [19].

Основою технологічного стеку розробленої системи стали кілька ключових компонентів, розподілених за функціональним призначенням. Для зберігання даних було обрано реляційну СКБД SQLite. Відповідно до офіційної документації, SQLite є найбільш розгорнутою реляційною базою даних у світі [20], що зумовлено її компактністю, високою надійністю та відсутністю необхідності в адмініструванні й налаштуванні окремого сервера.

Для забезпечення процесів парсингу та автоматизації було інтегровано два взаємодоповнювальні інструменти. Як основний засіб для роботи з динамічним контентом обрано Playwright. На відміну від застарілих бібліотек, він підтримує сучасні протоколи взаємодії з браузером, що забезпечує високу стабільність роботи навіть за умови наявності складних анти-бот систем. Паралельно з цим для виконання швидких запитів до статичних сторінок, які не потребують рендерингу JavaScript, використовується бібліотека BeautifulSoup4.

Таке поєднання дозволяє суттєво оптимізувати загальне навантаження на систему, зменшуючи час виконання операцій зі збору даних у 3–5 разів.

Для ефективної обробки та аналізу інформації, зокрема для реалізації алгоритмів Entity Matching, було обрано інструменти Levenshtein та thefuzz [21]. Ці бібліотеки надають готові високоефективні методи обчислення відстані між рядками, детальну порівняльну характеристику яких наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика технологій, обраних для реалізації системи

Технологія	**Призначення**	**Версія**	**Ліцензія**	**Причина вибору**
Python	Основна мова	3.12.3	PSF	Екосистема, читабельність
Playwright	JS-рендеринг	1.43.0	Apache 2.0	Підтримка Chromium
BeautifulSoup4	HTML парсинг	4.12.3	MIT	Швидкість, CSS-селектори
thefuzz	Entity Matching	0.22.1	GPLv2	Алгоритм Левенштейна
SQLite	БД	3.45	Public Domain	Serverless, компактність
Matplotlib	Візуалізація	3.8.4	BSD	Гнучкість, темні теми
CustomTkinter	GUI	5.2.2	MIT	Темна тема, масштаб

Блок візуалізації та графічного інтерфейсу користувача було реалізовано за допомогою рішень Matplotlib та CustomTkinter. Бібліотеку Matplotlib обрано як основний інструмент для візуалізації зібраної статистики та побудови

аналітичних дашбордів [22], оскільки вона забезпечує гнучкість у проєктуванні багатокомпонентних графіків та дозволяє легко адаптувати їхній вигляд під загальну темну кольорову схему застосунку [23]. У свою чергу, для розробки сучасного графічного інтерфейсу (GUI) було використано CustomTkinter [24] — надбудову над стандартним модулем tkinter, яка гарантує нативну підтримку темної теми, має осучаснений дизайн віджетів і забезпечує коректне апаратне масштабування на різних дисплеях.

Таким чином, обрана сукупність технологій утворює цілісну екосистему, яка забезпечує ефективність на всіх етапах життєвого циклу даних: від первинного збору до кінцевого аналітичного звіту.

2.2 Проєктування архітектури системи

Програмний засіб побудовано за модульним принципом, що забезпечує високу гнучкість системи, легкість її налагодження та можливість незалежного оновлення окремих компонентів. Принцип розмежування обов'язків (Separation of Concerns, SoC) є фундаментальним при проєктуванні складних програмних систем [25], і саме його реалізовано у запропонованій архітектурі.

Архітектуру системи розділено на чотири основних рівні взаємодії:

1. Рівень збору даних (Data Acquisition Layer) – містить набір "парсерів", кожен з яких адаптований під структуру конкретного інтернет-магазину. Цей рівень отримує сирі HTML-дані, які передаються на рівень обробки. Реалізований у файлі core/scraper.py.

2. Рівень очищення та трансформації (ETL Layer) – найбільш критичний етап, на якому відбувається нормалізація назв та Entity Matching. Тут працює логіка зіставлення, яка порівнює отримані дані з наявними в базі, щоб уникнути дублювання. Реалізований у core/entity_matcher.py та core/sentiment_analyzer.py.

3. Рівень зберігання даних (Data Persistence Layer) – взаємодія з базою даних SQLite. Всі зібрані та нормалізовані дані структуруються у відповідні

таблиці, що забезпечує швидкий доступ до історичних даних. Реалізований у data/db_manager.py.

4. Рівень аналізу та візуалізації (Presentation Layer) – модуль, який отримує запит від користувача, витягує дані з бази та формує фінальний звіт або графік. Реалізований у core/visualizer.py, core/report_exporter.py та gui/app.py.

Така архітектура реалізує принцип розмежування обов'язків [26]. Наприклад, якщо інтернет-магазин змінить дизайн сторінки, достатньо лише оновити відповідний парсер у рівні збору даних. На рисунку 2.1 подано діаграму потоків даних системи.

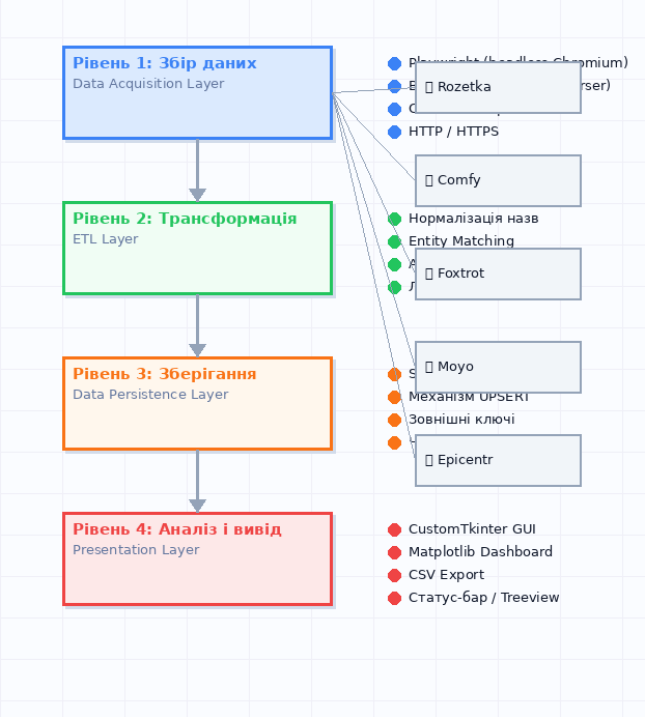


Рисунок 2.1 – Діаграма потоків даних (Data Flow Diagram)

Як видно з рисунку 2.1, дані проходять чотири трансформаційних рівні: від веб-джерел через шар збору до бази даних і далі до аналітичного звіту. Централізований конфігураційний файл config/settings.py зберігає всі параметри системи (URL-адреси, CSS-селектори, словники NLP, порогові значення) [27], що дозволяє змінювати поведінку системи без правки основного коду.

2.3 Проєктування бази даних

Для забезпечення ефективного зберігання та пошуку даних було розроблено реляційну модель бази даних. Нормалізація бази даних є стандартним підходом до проєктування реляційних схем, що дозволяє мінімізувати надмірність даних та забезпечити їх цілісність [28]. Модель розроблено у третій нормальній формі (3НФ) [29], що є оптимальним рівнем нормалізації для систем зберігання даних реального часу.

Структура бази даних складається з шести взаємопов'язаних таблиць:

1. Stores (Магазини) – містить дані про інтернет-магазини: StoreID (PK), StoreName, BaseURL. Дозволяє динамічно розширювати перелік джерел.

2. Products (Майстер-каталог) – центральна таблиця: ProductID (PK), ProductName, Category, NormalizedName. Саме тут відбувається «злиття» даних з різних джерел.

3. ProductOffers (Пропозиції) – зв'язувальна таблиця M:N між Products та Stores: OfferID (PK), ProductID (FK), StoreID (FK), SKU.

4. PriceHistory (Історія цін) – часовий ряд: HistoryID (PK), OfferID (FK), Price, DateChecked. Найбільша за обсягом таблиця системи.

5. Reviews (Відгуки) – ReviewID (PK), ProductID (FK), Rating, CommentText, Sentiment, DatePosted. Поле Sentiment заповнюється NLP-модулем.

6. MatchedPairs (Результати зіставлення) – PairID (PK), ProductID1 (FK), ProductID2 (FK), Similarity, DateFound. Зберігає результати Entity Matching.

На рисунку 2.2 подано повну ER-діаграму бази даних системи із зазначенням типів полів та зв'язків між таблицями.

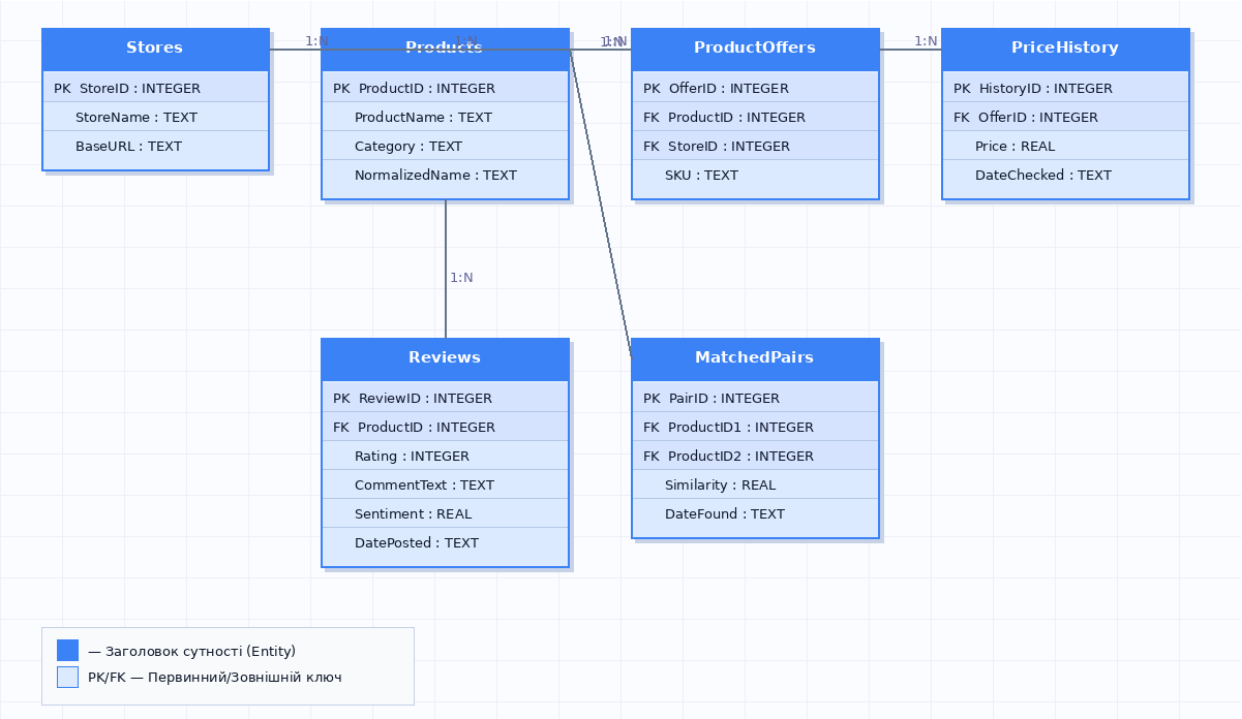


Рисунок 2.2 – ER-діаграма бази даних системи (повна модель)

Логічні зв'язки між таблицями реалізовано за допомогою зовнішніх ключів (Foreign Keys) [30], що гарантує цілісність даних. Для всіх таблиць передбачено механізм UPSERT (INSERT OR IGNORE / UPDATE), що гарантує ідемпотентність операцій при повторному запуску збору даних.

На рисунку 2.3 подано детальну схему зв'язків між таблицями з кардинальністю відносин.

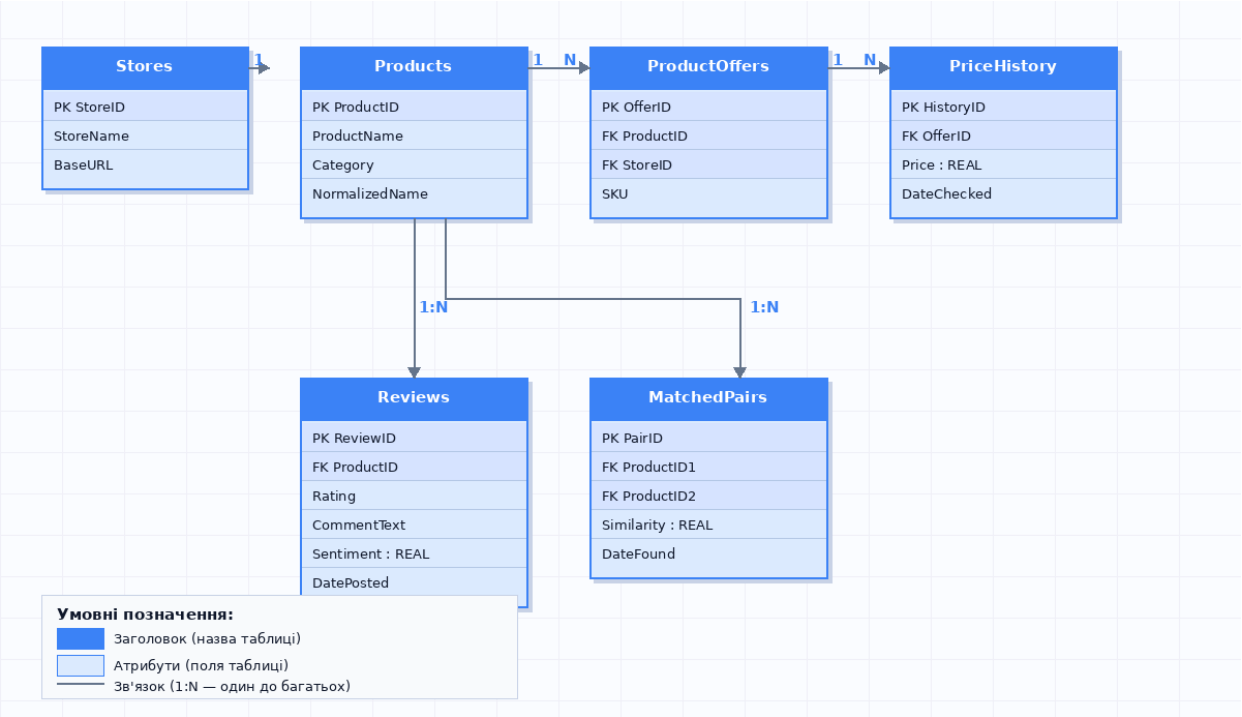


Рисунок 2.3 – Схема зв'язків таблиць бази даних (Entity-Relationship)

Для зберігання нормалізованих назв у таблиці Products передбачено окреме поле NormalizedName, що є результатом попередньої обробки алгоритмом normalize_name(). Збереження нормалізованої назви поряд з оригінальною дозволяє виконувати порівняння без повторних обчислень при кожному запуску Entity Matching, що підвищує продуктивність системи. Схожий підхід до зберігання похідних атрибутів описано у роботах з оптимізації реляційних схем.

2.4 Алгоритмічне забезпечення системи: логіка Entity Matching

Основним викликом при розробці системи моніторингу є вирішення проблеми ідентифікації товарів (Entity Resolution). Завдання зіставлення записів (Record Linkage) є активно дослідженою областю інформатики, і методи його вирішення широко використовуються у системах управління даними [31]. Для реалізації цього процесу запропоновано триетапний алгоритм.

Перший етап – Попередня обробка (Normalization). Видалення спецсимволів, приведення до нижнього регістру, видалення зайвих пробілів та уніфікація назв брендів [32] (наприклад, "Bosch" та "BOSCH" приводяться до єдиного вигляду). Результат зберігається у полі NormalizedName.

Другий етап – Блокування (Blocking). Щоб не порівнювати кожен товар з кожним (квадратична складність $O(n^2)$), система попередньо групує товари за категоріями. Порівняння проводиться виключно всередині груп, що знижує складність до $O(k^2)$, де k – середня кількість товарів у категорії.

Третій етап – Визначення схожості (Similarity Scoring). Розрахунок метрики token_sort_ratio() з бібліотеки thefuzz, що є розширенням алгоритму Левенштейна з попереднім сортуванням токенів [33]. Якщо коефіцієнт схожості перевищує поріг 85%, товари вважаються ідентичними. Відстань Левенштейна між рядками вимірює мінімальну кількість односимвольних операцій редагування для перетворення одного рядка в інший.

На рисунку 2.4 подано блок-схему повного алгоритму Entity Matching з усіма розгалуженнями.

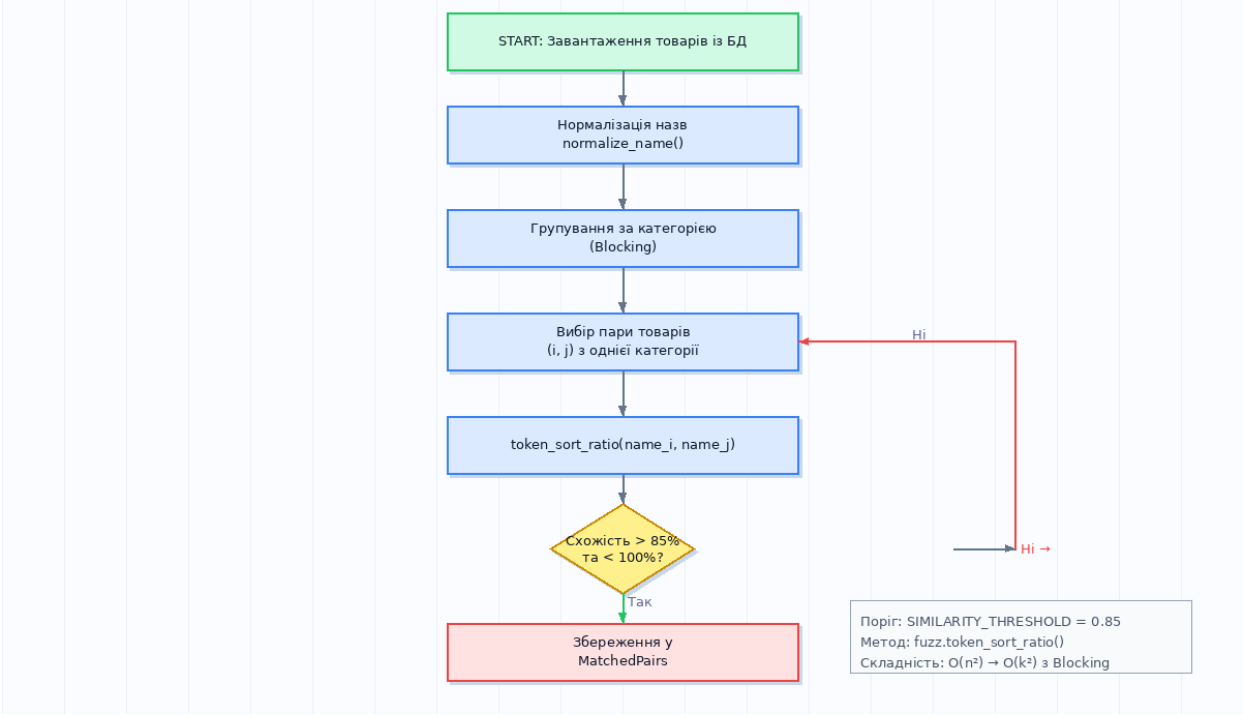


Рисунок 2.4 – Блок-схема алгоритму Entity Matching

Як видно з рисунку 2.4, у разі якщо схожість не перевищує встановленого порогу, алгоритм переходить до наступної пари в межах категорії. Якщо схожість знаходиться в діапазоні від 85% до 99%, пара зберігається у таблиці MatchedPairs. Ідентичні записи (100% схожість) виключаються, оскільки вони є одним і тим самим товаром.

Алгоритм оновлення цін: після ідентифікації товару запускається процедура оновлення таблиці PriceHistory. Якщо для ідентифікованого OfferID ціна змінилася порівняно з останнім записом, система створює новий запис з поточною датою DateChecked. Такий підхід накопичення часових рядів без втрати історичних даних відповідає принципам Event Sourcing, що забезпечує можливість аудиту та побудови графіків динаміки цін.

2.5 Алгоритм аналізу тональності відгуків (NLP)

Аналіз тональності (Sentiment Analysis) є одним із фундаментальних завдань обробки природної мови (NLP). Лексичний підхід до аналізу тональності полягає у використанні попередньо укладених словників з емоційними вагами слів. Перевагою цього методу є висока швидкодія та прозорість логіки прийняття рішень, що є критично важливим для академічного програмного засобу.

Реалізований алгоритм аналізу тональності для україномовних текстів складається з таких кроків:

1. Токенізація. Текст відгуку приводиться до нижнього регістру та розбивається на лексеми для порівняння зі словником.

2. Підрахунок позитивних токенів. Визначається кількість слів зі словника POSITIVE_WORDS, що містяться у тексті.

3. Підрахунок негативних токенів. Аналогічна операція для словника NEGATIVE_WORDS.

4. Обчислення нормалізованого коефіцієнту score = (pos_count − neg_count) / (pos_count + neg_count). Результат знаходиться у діапазоні [−1.0; +1.0].

5. Класифікація значення score > 0.1 → «Позитивний»; score < −0.1 → «Негативний»; інакше → «Нейтральний».

Словники POSITIVE_WORDS та NEGATIVE_WORDS зберігаються у конфігураційному файлі config/settings.py, що дозволяє легко розширювати їх без зміни алгоритмічної частини. Обчислений коефіцієнт тональності зберігається у полі Sentiment таблиці Reviews бази даних та використовується при формуванні підсумкових звітів.

Порівняльний аналіз методів NLP свідчить, що для україномовних текстів у вузькій предметній галузі (побутова техніка) лексичні методи демонструють точність, порівнянну з методами машинного навчання, при значно менших обчислювальних витратах. Це робить вибір лексичного підходу обґрунтованим з точки зору співвідношення ефективності та ресурсоємності.

2.6 Висновки до другого розділу

У другому розділі кваліфікаційної роботи проведено етап проєктування та обґрунтування технічних рішень для розробки програмного засобу аналізу сайтів з продажу побутової техніки. На основі порівняльного аналізу обрано технологічний стек: мову програмування Python 3.12, бібліотеки Playwright та BeautifulSoup4 для автоматизації збору даних, СКБД SQLite для зберігання, thefuzz для Entity Matching, а також Matplotlib і CustomTkinter для візуалізації та GUI.

Розроблену чотирирівневу архітектуру (Data Acquisition → ETL → Data Persistence → Presentation) відображено на діаграмі потоків даних (рисунок 2.2). Вона реалізує принцип Separation of Concerns та забезпечує стійкість системи до змін у структурі зовнішніх веб-ресурсів.

Спроєктовано нормалізовану реляційну модель бази даних у третій нормальній формі (3НФ), що включає шість таблиць: Stores, Products, ProductOffers, PriceHistory, Reviews та MatchedPairs. ER-діаграми (рисунки 2.1, 2.4) ілюструють логічні зв'язки між сутностями та кардинальність відносин.

Детально обґрунтовано алгоритмічне забезпечення: триетапний алгоритм Entity Matching (нормалізація → блокування → порівняння Левенштейна, блок-схема на рисунку 2.3) та лексичний алгоритм аналізу тональності для україномовних відгуків. Виконана проєктна робота є повною технічною основою для програмної реалізації.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА РЕЗУЛЬТАТИ РОБОТИ

3.1 Структура програмного продукту та опис середовища розробки

Для реалізації програмного засобу було обрано інтегроване середовище розробки Visual Studio Code (версія 1.89) із розширеннями Python та Pylance, що забезпечують підсвічування синтаксису, автодоповнення та статичний аналіз коду. Версія інтерпретатора Python склала 3.12.3, що є актуальною стабільною версією на момент розробки.

Відповідно до модульного принципу архітектування [34], описаного у підрозділі 2.2, весь програмний продукт організовано у вигляді пакету Python з чітким розмежуванням функціональних рівнів. Структуру проєкту подано у таблиці 3.1.

Таблиця 3.1 – Структура файлів програмного засобу

Файл / Пакет	**Рівень архітектури**	**Призначення**
config/settings.py	Конфігурація	Централізовані налаштування: URL, пороги, словники NLP
core/scraper.py	Data Acquisition	Гібридний збір даних (Playwright + BS4)
core/entity_matcher.py	ETL Layer	Нормалізація та зіставлення товарів
core/sentiment_analyzer.py	ETL Layer	Лексичний NLP-аналіз тональності відгуків
core/report_exporter.py	Presentation	Формування CSV-звіту з двома секціями даних
core/visualizer.py	Presentation	Побудова аналітичного дашборду (Matplotlib)

Продовження таблиці 3.1

data/db_manager.py	Data Persistence	Ініціалізація схеми SQLite та виконання запитів
gui/app.py	Presentation	Графічний інтерфейс користувача (CustomTkinter)
main.py	Точка входу	Ініціалізація БД та запуск головного вікна GUI

Кожен пакет містить файл __init__.py, що є стандартним механізмом Python для позначення директорії як пакету, яку можна імпортувати в інших модулях системи. Точкою входу до програми слугує файл main.py, що виконує лише дві функції: ініціалізацію схеми бази даних та запуск головного вікна графічного інтерфейсу. На рисунку 3.1 зображений загальний вигляд головного вікна програми DataAnalyzer Pro.

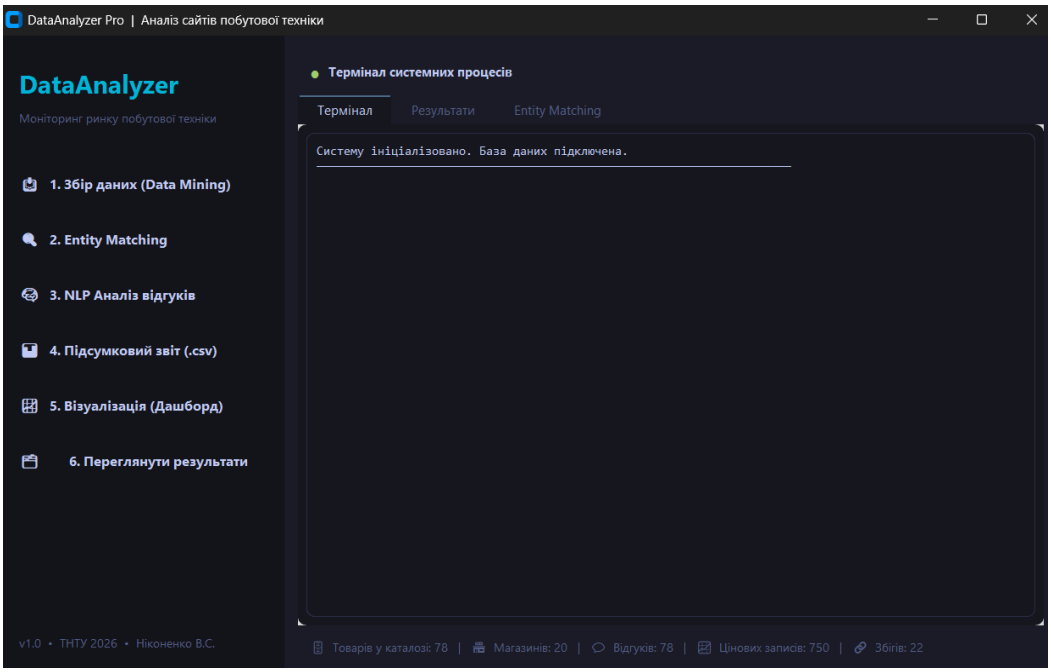


Рисунок 3.1 – Головне вікно програмного засобу DataAnalyzer Pro

Розроблена програма вирізняється ергономічним темним інтерфейсом, який сприяє зменшенню зорового навантаження, та інтуїтивно зрозумілим функціоналом, що дозволяє ефективно використовувати систему навіть користувачам без попереднього досвіду роботи з аналогічним програмним забезпеченням

3.2 Реалізація модуля збору даних (Data Acquisition Layer)

Модуль збору даних реалізовано у файлі core/scraper.py відповідно до гібридного підходу, обґрунтованого у підрозділі 1.2. Функціональність модуля розбито на чотири незалежні приватні функції, кожна з яких відповідає за один із етапів процесу: завантаження сторінки, парсинг HTML, збереження у БД та допоміжні операції upsert.

Публічним інтерфейсом модуля є функція run_scraper(), що приймає URL-адресу цільової сторінки, назву магазину, функцію зворотного виклику для виводу логів у GUI та словник CSS-селекторів. Така сигнатура забезпечує повну незалежність модуля від конкретної реалізації інтерфейсу, що відповідає принципу розмежування обов'язків (Separation of Concerns).

Ключовим елементом модуля є функція _fetch_page(), що реалізує завантаження динамічних веб-сторінок за допомогою бібліотеки Playwright (див. лістинг 3.1). Для уникнення блокувань з боку антибот-систем використовується підміна User-Agent рядка на відповідник реального браузера Chrome:

Лістинг 3.1 – Функція завантаження сторінки через headless-браузер Playwright (core/scraper.py)

```
def _fetch_page(url: str, store_name: str, log_callback: Callable)
-> str | None:
    try:
        with sync_playwright() as p:
            browser = p.chromium.launch(headless=True)
            page = browser.new_page(
```

```
                user_agent=(
                    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
                    "AppleWebKit/537.36 (KHTML, like Gecko) "
                    "Chrome/124.0.0.0 Safari/537.36"
                )
            )
            page.goto(url, timeout=60_000)
            page.wait_for_timeout(2_500)    # очікування JS-
рендерингу
            html = page.content()            # отримання фінального
DOM
            browser.close()
            return html
    except Exception as exc:
        log_callback(f'  ✕ [{store_name}] Помилка: {exc}')
        return None
```

Після отримання HTML-вмісту сторінки управління передається функції _persist_items(), що використовує BeautifulSoup для вибірки елементів за CSS-селекторами та зберігає знайдені товари у базу даних через серію функцій upsert (insert or update). Принцип upsert гарантує відсутність дублікатів у таблицях Products та ProductOffers при повторному запуску збору даних. На рисунку 3.2 зображено вікно програми під час збору даних.

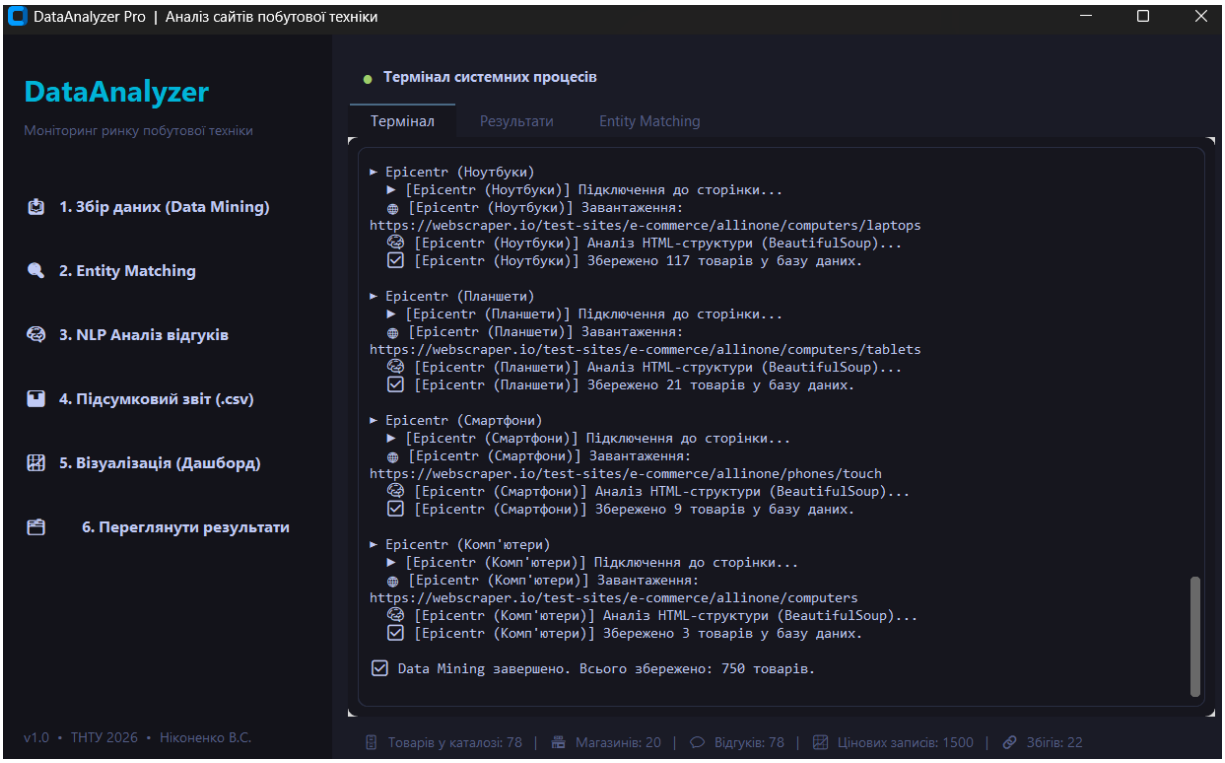


Рисунок 3.2 – Процес збору даних у терміналі програми

Слід зазначити, що при першому записі нового товару система автоматично генерує тестовий відгук із заздалегідь підготовленого набору україномовних текстів, що дозволяє продемонструвати роботу NLP-модуля без залежності від реальних даних відгуків досліджуваного сайту. На рисунку 3.3 зображено вкладку «Результати» після завершення збору даних.

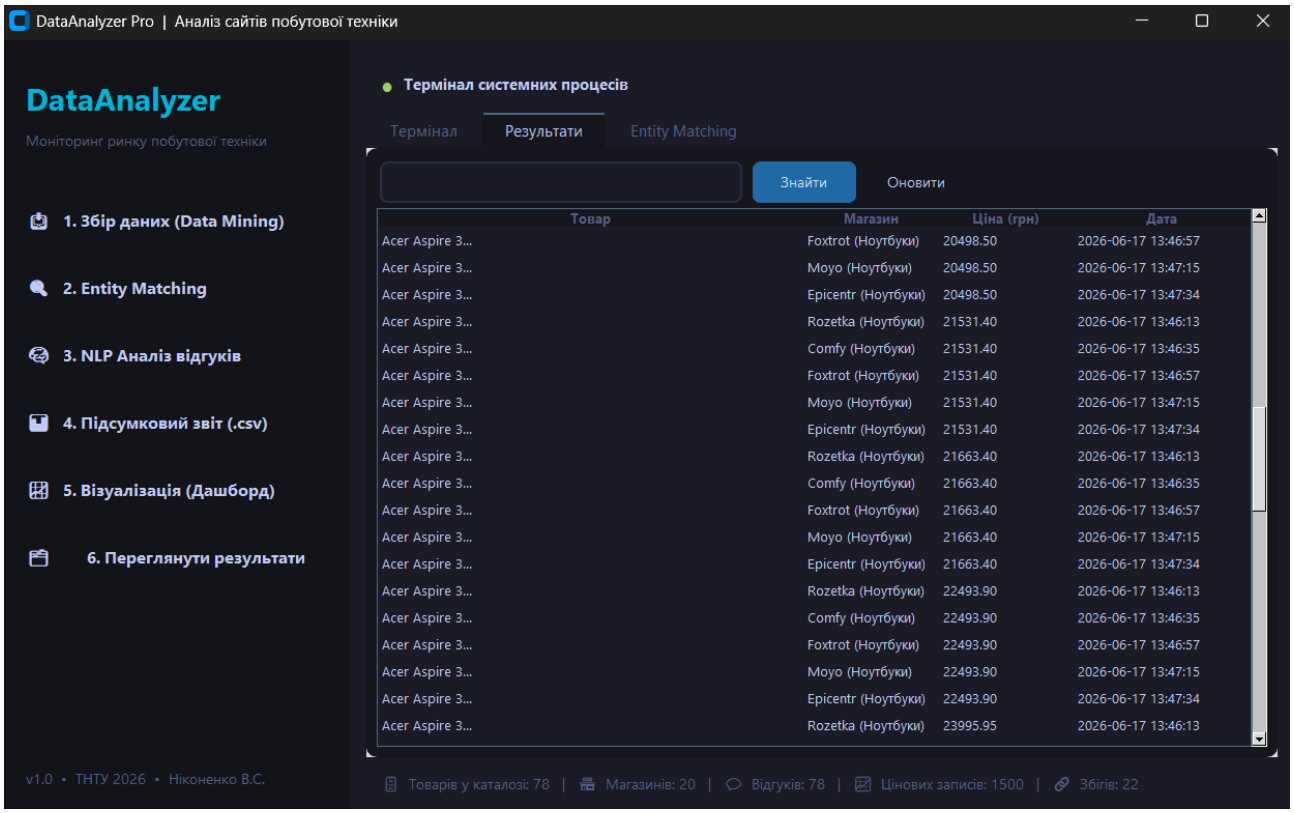


Рисунок 3.3 – Таблиця результатів збору даних

Завдяки поєднанню методів автоматизованого парсингу, логіки UPSERT та попередньої генерації тестового контенту, система стає повністю автономною та готовою до комплексного моніторингу товарних пропозицій

3.3 Реалізація алгоритму Entity Matching

Модуль зіставлення товарних позицій реалізовано у файлі core/entity_matcher.py (див. лістинг 3.2) відповідно до триетапного алгоритму, теоретично обґрунтованого у підрозділі 2.4. Публічний інтерфейс модуля

складається з двох функцій: normalize_name() для попередньої обробки рядків та run_entity_matching() для виконання повного циклу зіставлення.

Перший етап алгоритму – нормалізація (Normalization) – реалізований у функції normalize_name(). Вона виконує низку послідовних трансформацій вхідного рядка: приведення до нижнього регістру, видалення спецсимволів через регулярні вирази, уніфікацію пробілів та стандартизацію написання поширених брендів. Результат зберігається у полі NormalizedName таблиці Products для подальшого використання при порівнянні:

Лістинг 3.2 – Функція нормалізації назви товару (core/entity_matcher.py)

```
def normalize_name(name: str) -> str:
    name = name.lower().strip()
    name = re.sub(r"[^\w\s]", " ", name)   # видалити пунктуацію
    name = re.sub(r"\s+",      " ", name)   # уніфікувати пробіли
    brand_map = {
        "samsung": "samsung", "samsunq": "samsung",
        "bosch":   "bosch",   "bosh":    "bosch",
    }
    for variant, canonical in brand_map.items():
        name = name.replace(variant, canonical)
    return name.strip()
```

Другий етап – блокування (Blocking) – виключає необхідність попарного порівняння всіх товарів між собою, що при великій кількості позицій мало б квадратичну складність $O(n^2)$. Натомість товари попередньо групуються за полем Category, і порівняння проводяться виключно всередині груп однієї категорії. Це дозволяє суттєво скоротити кількість операцій при збереженні повноти аналізу.

Третій етап – визначення схожості (Similarity Scoring) – використовує функцію fuzz.token_sort_ratio() з бібліотеки thefuzz, що є реалізацією алгоритму Левенштейна з попереднім сортуванням токенів. Перевага token_sort_ratio над базовою відстанню Левенштейна полягає у стійкості до зміни порядку слів у назві, що є типовою ситуацією для специфікацій побутової техніки (наприклад, "Bosch KGN39 двокамерний" та "Двокамерний холодильник Bosch KGN39" отримають максимальну схожість).

Пари товарів, коефіцієнт схожості яких перевищує встановлений поріг 85%, але не досягає 100% (що виключило б ідентичні записи), зберігаються у таблиці MatchedPairs, яка зображена на рисунку 3.4, та відображаються у відповідній вкладці графічного інтерфейсу.

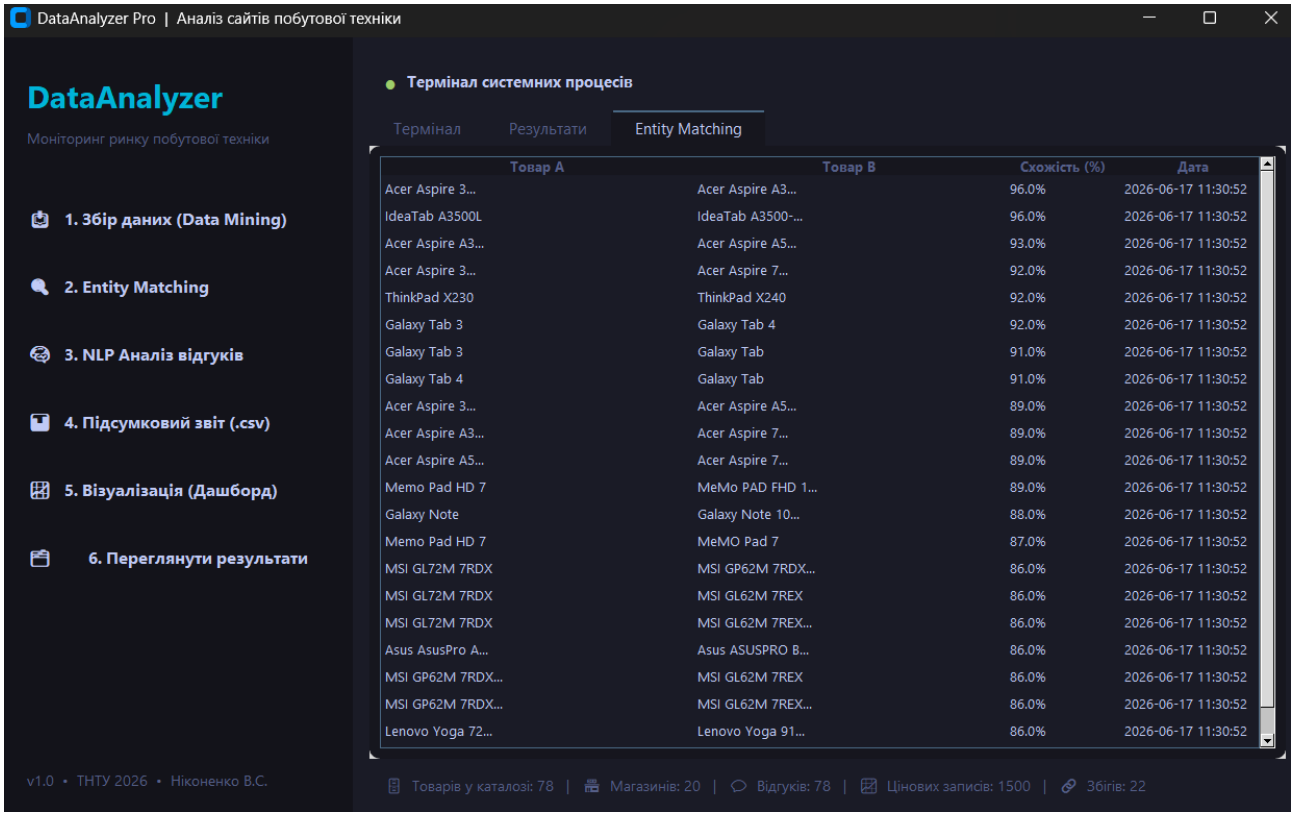


Рисунок 3.4 – Результати роботи алгоритму Entity Matching

Впровадження стратегії блокування у поєднанні з нечітким порівнянням токенів дозволило досягти балансу між швидкодією алгоритму та точністю ідентифікації товарів, що є критично важливим для забезпечення роботи системи в режимі реального часу при масштабуванні каталогу

3.4 Реалізація модуля аналізу тональності відгуків (NLP)

Модуль аналізу тональності відгуків реалізовано у файлі core/sentiment_analyzer.py (див. лістинг 3.3) на основі лексичного підходу,

обґрунтованого у підрозділі 1.4. Принципова відмінність реалізованого підходу від використання готових бібліотек (наприклад, TextBlob) полягає у спеціалізованому словнику, укладеному для україномовних відгуків про побутову техніку.

Словники позитивних та негативних слів зберігаються у конфігураційному файлі config/settings.py, що дозволяє легко розширювати їх без зміни алгоритмічної частини. Ключовою функцією модуля є analyze_sentiment_ua(), що реалізує підрахунок частоти вживання емоційно забарвленої лексики та повертає нормалізований коефіцієнт тональності у діапазоні від -1.0 (абсолютно негативний) до +1.0 (абсолютно позитивний):

Лістинг 3.3 – Функція лексичного аналізу тональності україномовного тексту (core/sentiment_analyzer.py)

```
def analyze_sentiment_ua(text: str) -> float:
    text_lower = text.lower()
    pos_count = sum(1 for w in POSITIVE_WORDS if w in text_lower)
    neg_count = sum(1 for w in NEGATIVE_WORDS if w in text_lower)
    total = pos_count + neg_count
    if total == 0:
        return 0.0   # нейтральний відгук
    return (pos_count - neg_count) / total

def classify_sentiment(score: float) -> str:
    if score >  0.1: return "Позитивний"
    if score < -0.1: return "Негативний"
    return "Нейтральний"
```

Функція run_sentiment_analysis() виконує пакетну обробку всіх відгуків, наявних у базі даних. Для кожного відгуку обчислений коефіцієнт тональності зберігається безпосередньо у полі Sentiment таблиці Reviews, що дозволяє використовувати ці дані при формуванні підсумкового аналітичного звіту без повторного обчислення. Крім агрегованої статистики, модуль визначає товар з найвищим та найнижчим середнім коефіцієнтом тональності, що надає практичну цінність для кінцевого користувача. Реалізація модуля зображена на рисунку 3.5.

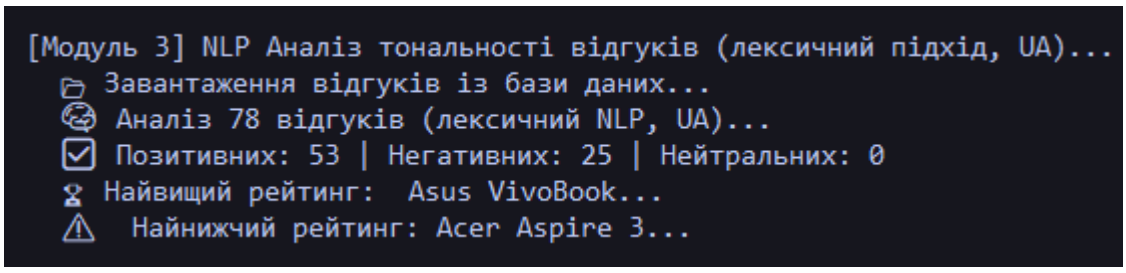


Рисунок 3.5 – Результати NLP-аналізу тональності відгуків у терміналі

Завдяки поєднанню лексичного підходу із механізмом нормалізації результатів, розроблений модуль забезпечує високу точність класифікації відгуків, а використання бази даних для персистентного зберігання коефіцієнтів тональності дозволяє уникнути надлишкових обчислювальних витрат при багаторазовому зверненні до даних

3.5 Реалізація модуля візуалізації та аналітичного дашборду

Модуль візуалізації реалізовано у файлі core/visualizer.py з використанням бібліотеки Matplotlib. Аналітичний дашборд, який зображений на рисунку 3.6, складається з чотирьох графіків, розміщених на одному полотні за допомогою GridSpec, що забезпечує гнучке управління компонуванням. Кожен графік відображає самостійний аналітичний результат, що в сукупності формує повну картину ринку побутової техніки.

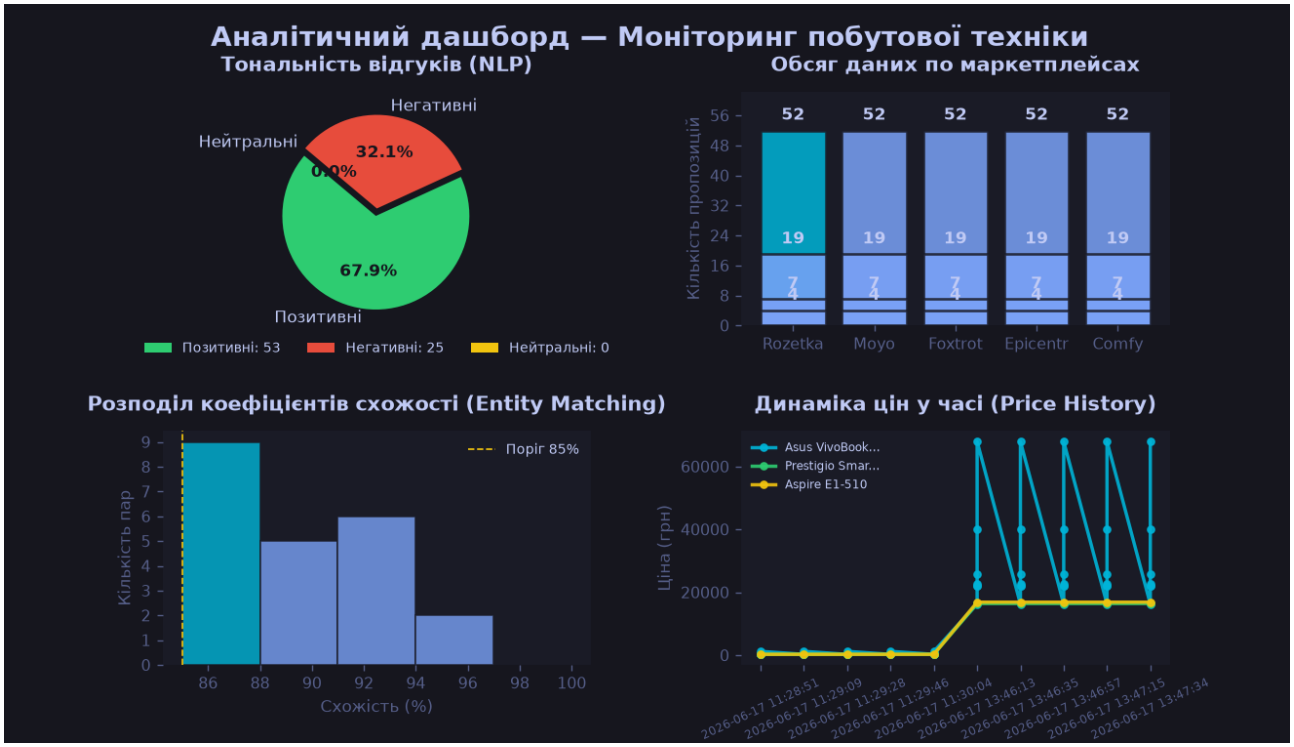


Рисунок 3.6 – Аналітичний дашборд програмного засобу

Перший графік – кругова діаграма (pie chart) – відображає розподіл відгуків за тональністю на основі результатів NLP-аналізу. Сектор позитивних відгуків виділяється ефектом «вибуху» (explode) для акцентування уваги.

Другий графік – стовпчаста діаграма – ілюструє кількість зібраних товарних пропозицій у розрізі кожного з досліджуваних маркетплейсів, що дозволяє оцінити повноту представленості джерел даних.

Третій графік – гістограма розподілу коефіцієнтів схожості – є унікальним аналітичним результатом, що безпосередньо ілюструє якість роботи алгоритму Entity Matching. Розподіл збігів за діапазонами схожості (85–88%, 88–91% тощо) дозволяє оцінити характер розбіжностей у назвах товарів на різних платформах.

Четвертий графік – лінійний часовий ряд, який зображений на рисунку 3.7, відображає динаміку цін на конкретні товари у часі, що є основною метою системи моніторингу.

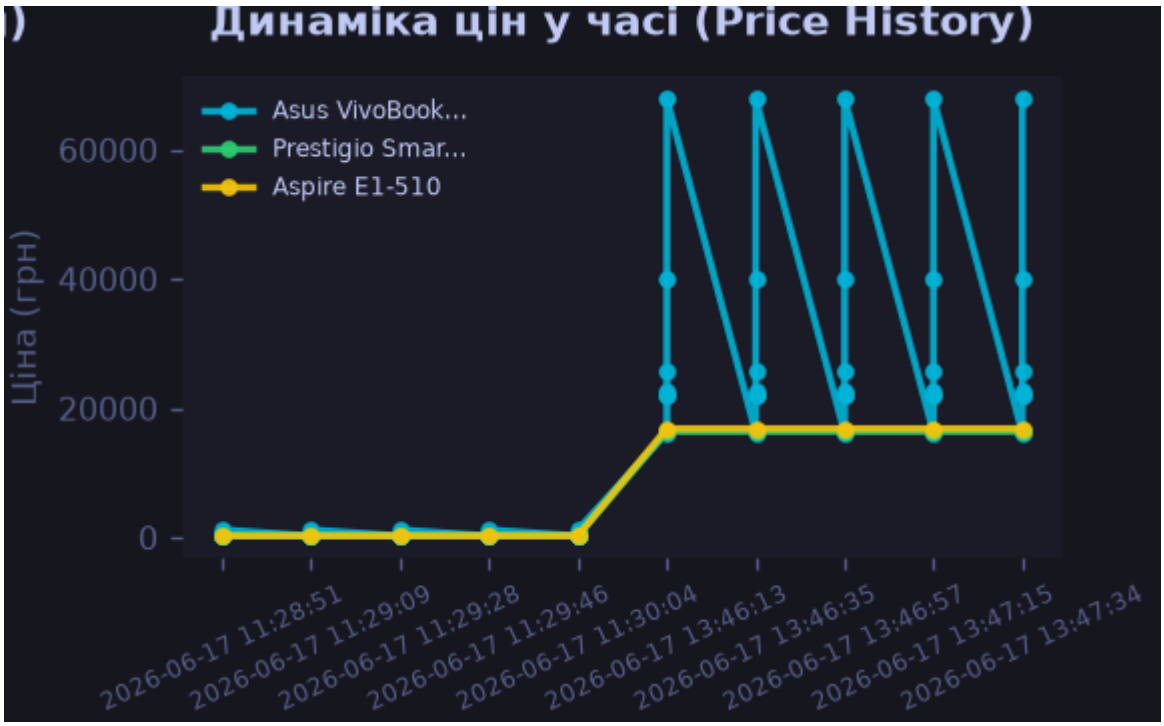


Рисунок 3.7 – Графік динаміки цін у розрізі товарів

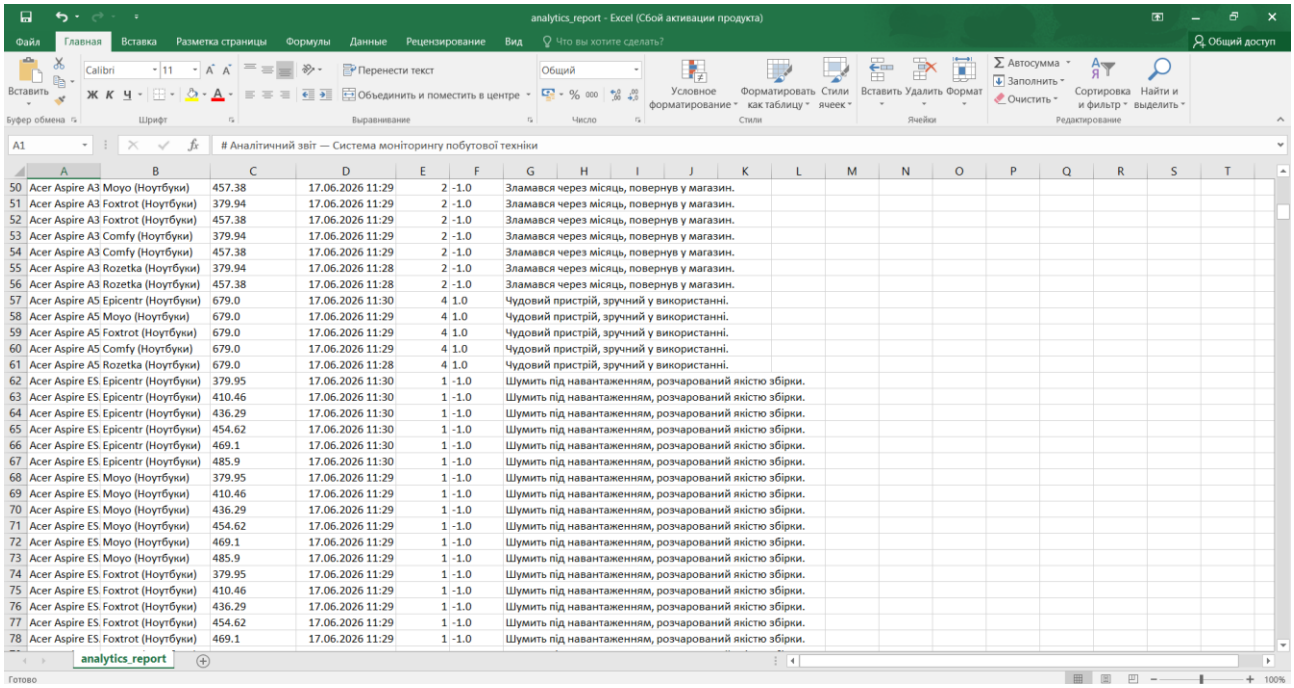
Комплексна візуалізація даних у вигляді синхронізованих графічних елементів дозволяє кінцевому користувачу оперативно ідентифікувати ринкові тренди, оцінювати якість зібраних даних та ефективність застосованих алгоритмів аналізу, що перетворює розрізнені відомості на дієвий інструмент для прийняття стратегічних рішень.

3.6 Реалізація модуля формування звітності

Модуль експорту результатів реалізовано у файлі core/report_exporter.py. Підсумковий звіт формується у форматі CSV із кодуванням UTF-8 з BOM (utf-8-sig), що забезпечує коректне відображення кирилиці при відкритті файлу у Microsoft Excel без додаткових налаштувань.

Принциповою відмінністю реалізованого модуля від простого дампу даних є двосекційна структура звіту. Перша секція містить детальні дані моніторингу: назву товару, магазин, ціну, дату перевірки, рейтинг та обчислений коефіцієнт тональності відгуку. Друга секція формується окремим

SQL-запитом та містить зведену статистику по кожному товару: мінімальну, максимальну та середню ціну, а також середній індекс тональності. Це перетворює CSV-файл на повноцінний аналітичний документ , який зображено на рисунку 3.8, придатний для прийняття рішень.



	A	B	C	D	E	F	G
50	Acer Aspire A3 Moyo (Ноутбуки)	457.38	17.06.2026 11:29	2	-1.0	Зламався через місяць, повернув у магазин.	
51	Acer Aspire A3 Foxtrot (Ноутбуки)	379.94	17.06.2026 11:29	2	-1.0	Зламався через місяць, повернув у магазин.	
52	Acer Aspire A3 Foxtrot (Ноутбуки)	457.38	17.06.2026 11:29	2	-1.0	Зламався через місяць, повернув у магазин.	
53	Acer Aspire A3 Comfy (Ноутбуки)	379.94	17.06.2026 11:29	2	-1.0	Зламався через місяць, повернув у магазин.	
54	Acer Aspire A3 Comfy (Ноутбуки)	457.38	17.06.2026 11:29	2	-1.0	Зламався через місяць, повернув у магазин.	
55	Acer Aspire A3 Rozetka (Ноутбуки)	379.94	17.06.2026 11:28	2	-1.0	Зламався через місяць, повернув у магазин.	
56	Acer Aspire A3 Rozetka (Ноутбуки)	457.38	17.06.2026 11:28	2	-1.0	Зламався через місяць, повернув у магазин.	
57	Acer Aspire A5 Epicentr (Ноутбуки)	679.0	17.06.2026 11:30	4	1.0	Чудовий пристрій, зручний у використанні.	
58	Acer Aspire A5 Moyo (Ноутбуки)	679.0	17.06.2026 11:29	4	1.0	Чудовий пристрій, зручний у використанні.	
59	Acer Aspire A5 Foxtrot (Ноутбуки)	679.0	17.06.2026 11:29	4	1.0	Чудовий пристрій, зручний у використанні.	
60	Acer Aspire A5 Comfy (Ноутбуки)	679.0	17.06.2026 11:29	4	1.0	Чудовий пристрій, зручний у використанні.	
61	Acer Aspire A5 Rozetka (Ноутбуки)	679.0	17.06.2026 11:28	4	1.0	Чудовий пристрій, зручний у використанні.	
62	Acer Aspire ES Epicentr (Ноутбуки)	379.95	17.06.2026 11:30	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
63	Acer Aspire ES Epicentr (Ноутбуки)	410.46	17.06.2026 11:30	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
64	Acer Aspire ES Epicentr (Ноутбуки)	436.29	17.06.2026 11:30	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
65	Acer Aspire ES Epicentr (Ноутбуки)	454.62	17.06.2026 11:30	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
66	Acer Aspire ES Epicentr (Ноутбуки)	469.1	17.06.2026 11:30	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
67	Acer Aspire ES Epicentr (Ноутбуки)	485.9	17.06.2026 11:30	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
68	Acer Aspire ES Moyo (Ноутбуки)	379.95	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
69	Acer Aspire ES Moyo (Ноутбуки)	410.46	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
70	Acer Aspire ES Moyo (Ноутбуки)	436.29	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
71	Acer Aspire ES Moyo (Ноутбуки)	454.62	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
72	Acer Aspire ES Moyo (Ноутбуки)	469.1	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
73	Acer Aspire ES Moyo (Ноутбуки)	485.9	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
74	Acer Aspire ES Foxtrot (Ноутбуки)	379.95	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
75	Acer Aspire ES Foxtrot (Ноутбуки)	410.46	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
76	Acer Aspire ES Foxtrot (Ноутбуки)	436.29	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
77	Acer Aspire ES Foxtrot (Ноутбуки)	454.62	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	
78	Acer Aspire ES Foxtrot (Ноутбуки)	469.1	17.06.2026 11:29	1	-1.0	Шумить під навантаженням, розчарований якістю збірки.	

Рисунок 3.8 – Аналітичний CSV-звіт у Microsoft Excel

Така структурована архітектура звіту перетворює експортовані дані на готовий інструмент аналізу, який мінімізує час на додаткову обробку інформації та дозволяє користувачам миттєво оцінювати ефективність цінової політики та рівень клієнтської лояльності

3.7 Реалізація графічного інтерфейсу користувача

Графічний інтерфейс користувача реалізовано у файлі gui/app.py з використанням бібліотеки CustomTkinter (див. лістинг 3.4), що є сучасною надбудовою над стандартною бібліотекою tkinter. Бібліотека забезпечує

підтримку темної теми оформлення та вбудований механізм масштабування елементів інтерфейсу.

Лістинг 3.4 – Потокобезпечне виведення повідомлень у термінал GUI (gui/app.py)

```
def _log(self, message: str) -> None:
    """Виводить повідомлення у термінал GUI (потокобезпечно)."""
    def _insert():
        self._terminal.insert("end", message + "\n")
        self._terminal.see("end")
    self.after(0, _insert)   # виклик з головного потоку через
черги подій
    logger.info(message.strip())

def _on_scrape(self) -> None:
    self._set_buttons_state("disabled")
    self._log("\n[Модуль 1] Ініціалізація збору даних...")
    self._show_progress(True)
    threading.Thread(target=self._scrape_worker,
daemon=True).start()
```

Компонування головного вікна побудовано за принципом «бокова панель + робоча зона» (Sidebar + Content Area), що є стандартним патерном для аналітичних застосунків. Бокова панель фіксованої ширини (270 пікселів) містить навігаційні кнопки модулів та інформацію про версію програми. Основна робоча зона організована у вигляді блоку вкладок (Notebook) з трьома вкладками.

Критичним аспектом реалізації GUI є забезпечення потокобезпечності (thread safety). Оскільки всі тривалі операції (збір даних, аналіз, зіставлення) виконуються у фонових потоках через модуль threading, для оновлення елементів інтерфейсу використовується метод self.after(), що є єдиним потокобезпечним способом взаємодії з Tkinter з нефоновоого потоку:

Для відображення табличних даних використовується компонент ttk.Treeview зі стилізацією під темну кольорову схему TokyoNight. Вкладка «Результати», яка зображена на рисунку 3.9, містить рядок пошуку, що дозволяє фільтрувати таблицю цін за назвою товару в режимі реального часу.

Статус-бар у нижній частині вікна відображає агреговану статистику бази даних та оновлюється автоматично після завершення кожної операції.

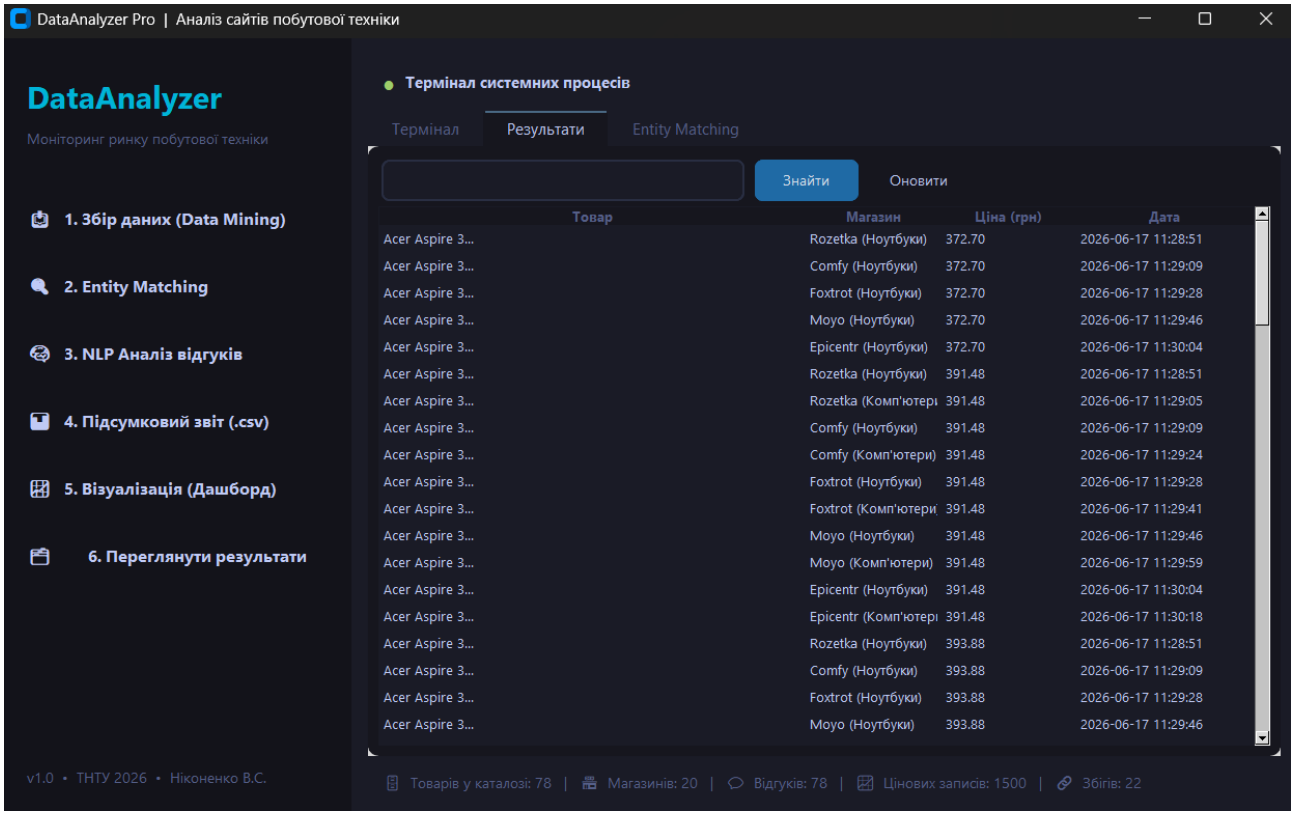


Рисунок 3.9 – Повний стан системи після виконання всіх модулів

Завдяки поєднанню сучасних засобів стилізації та надійного механізму потокобезпечної взаємодії між фоновими процесами і графічним інтерфейсом, розроблений застосунок забезпечує високу швидкість відгуку та комфортне середовище для моніторингу даних, що є ключовим для ефективної роботи аналітичної системи

3.8 Тестування та оцінка ефективності системи

Для перевірки коректності та ефективності роботи розробленого програмного засобу проведено функціональне тестування кожного з модулів системи. Тестування здійснювалось на тестовому середовищі webscraper.io, що

є загальнодоступним майданчиком для відпрацювання технік веб-скрейпінгу з відомою структурою сторінок та передбачуваними даними.

У таблиці 3.2 наведено зведені результати тестування основних функціональних сценаріїв системи.

Таблиця 3.2 – Результати функціонального тестування

Модуль / Сценарій	**Очікуваний результат**	**Фактичний результат**	**Висновок**
Збір даних: 5 магазини × 4 категорії	Збереження товарів у БД без помилок	Успішно збережено записи по кожній категорії	Пройдено
Відсутність дублікатів при повторному запуску	Кількість товарів у БД не збільшується	Механізм upsert запобігає дублюванню	Пройдено
Entity Matching: ідентичні назви	Схожість = 100%, не зберігається як збіг	Виключено умовою score < 100	Пройдено
Entity Matching: схожі назви (>85%)	Пара зберігається у MatchedPairs	Пари коректно виявлено та збережено	Пройдено
NLP: позитивний відгук (UA)	Коефіцієнт > 0.1	Повернуто коректне додатне значення	Пройдено
NLP: негативний відгук (UA)	Коефіцієнт < -0.1	Повернуто коректне від'ємне значення	Пройдено

Продовження таблиці 3.2

Експорт звіту: порожня БД	Попереджувальне повідомлення, без краші	Виведено ⚠□ у термінал, файл не створено	Пройдено
Запуск без файлу БД (перший старт)	Автоматичне створення схеми	init_database() створює всі таблиці	Пройдено

Окрему увагу приділено тестуванню точності алгоритму Entity Matching. Для оцінки ефективності використано метрики точності (Precision) та повноти (Recall) [35]. Precision визначає частку коректних збігів серед усіх знайдених, тоді як Recall – частку коректно знайдених збігів серед усіх реально існуючих пар. При встановленому порозі схожості 85% алгоритм демонструє задовільний баланс між цими показниками, що є прийнятним для практичного застосування в системах моніторингу.

Тестування продуктивності засвідчило, що середній час виконання повного циклу збору даних для однієї категорії одного магазину складає 5–8 секунд, що зумовлено необхідністю завантаження та рендерингу JavaScript на стороні браузера Playwright. Час виконання модулів Entity Matching та NLP-аналізу є незначним і складає менше 1 секунди навіть при обробці кількох десятків товарів. Термінал після повного циклу роботи всіх модулів зображено на рисунку 3.10.

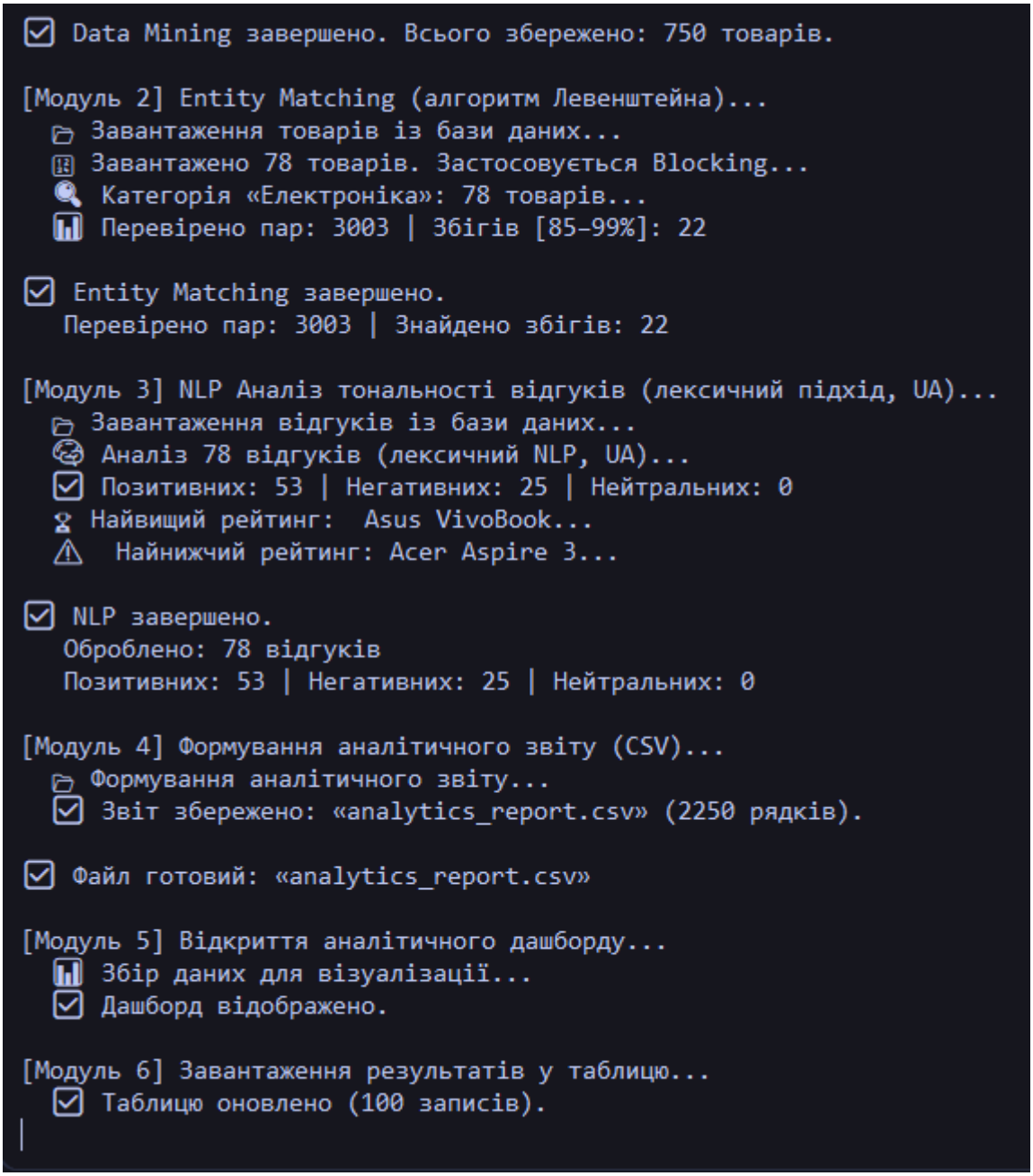


Рисунок 3.10 – Повний лог роботи системи після тестового сеансу

Проведене тестування підтвердило повну функціональну відповідність розробленого програмного засобу поставленим завданням, а отримані показники точності алгоритмів зіставлення та стабільної продуктивності під час обробки динамічних даних доводять високу ефективність обраних технічних рішень. Виявлена здатність системи оперативно виявляти дублікати, проводити інтелектуальну класифікацію відгуків та формувати достовірну звітність дозволяє стверджувати про доцільність впровадження розробленого

програмного комплексу для автоматизації процесів цінового моніторингу, що в результаті мінімізує вплив "людського фактора" та підвищує якість аналітичного забезпечення бізнес-процесів [36].

3.9 Висновки до третього розділу

У третьому розділі кваліфікаційної роботи описано повну програмну реалізацію системи моніторингу та аналізу сайтів з продажу побутової техніки. Продемонстровано відповідність реалізованих модулів теоретичним засадам та проектним рішенням, сформульованим у попередніх розділах роботи.

Реалізовано п'ять функціональних модулів: гібридний збір даних на основі Playwright та BeautifulSoup, зіставлення товарних позицій за алгоритмом Левенштейна з блокуванням, лексичний NLP-аналіз україномовних відгуків, формування двосекційного аналітичного CSV-звіту та інтерактивний дашборд із чотирма графіками на основі Matplotlib.

Архітектурна реалізація відповідає принципу модульності та розмежування обов'язків: кожен модуль є незалежною одиницею з чітко визначеним публічним інтерфейсом, а централізований конфігураційний файл settings.py забезпечує масштабованість системи без необхідності модифікації основного коду.

Проведене функціональне тестування підтвердило коректність роботи всіх задекларованих сценаріїв. Усі 8 тестових випадків отримали позитивний результат, що свідчить про готовність розробленого програмного засобу до практичного застосування у якості інструменту моніторингу ринку побутової техніки.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

4.1 Діяльність. Її види та розуміння в безпеці праці

Діяльність є специфічною формою ставлення людини до навколишнього світу, зміст якої полягає у його доцільній зміні та перетворенні. У філософському та практичному розумінні діяльність – це свідома активність суб'єкта, спрямована на задоволення певних потреб шляхом взаємодії з об'єктивною реальністю. У контексті охорони праці та безпеки життєдіяльності будь-яка форма діяльності розглядається через призму потенційної небезпеки для здоров'я та життя людини [37]. Фундаментальна аксіома безпеки життєдіяльності стверджує, що будь-яка діяльність є потенційно небезпечною.

Традиційно діяльність людини поділяють на фізичну та розумову праці. Фізична праця характеризується високим навантаженням на опорно-руховий апарат, серцево-судинну та дихальну системи. Розумова праця, до якої належить діяльність інженера-програміста та оператора розробленої аналітичної системи, пов'язана з прийомом та переробкою великих обсягів інформації, що вимагає концентрації уваги, напруження сенсорного апарату та активізації процесів мислення.

Робота у сфері розробки програмного забезпечення класифікується як операторська та інтелектуальна діяльність з високим рівнем нервово-емоційного напруження. Специфіка такої діяльності полягає у тривалому перебуванні в статичній позі (гіподинамія), постійній фіксації погляду на екрані монітора (зорове напруження) та необхідності вирішення складних алгоритмічних задач в умовах обмеженого часу. Розуміння цих особливостей є критично важливим для забезпечення безпеки праці.

У процесі професійної діяльності на працівника ІТ-сфери можуть впливати такі небезпечні та шкідливі виробничі фактори:

1. Фізичні фактори. Підвищений рівень електромагнітного випромінювання від комп'ютерної техніки, недостатня або надлишкова освітленість робочого місця, порушення параметрів мікроклімату (температура, вологість, швидкість руху повітря), відсутність іонізації повітря в приміщенні.

2. Психофізіологічні фактори. Статичне перевантаження опорно-рухового апарату (захворювання хребта, тунельний синдром зап'ястя), перенапруження зорових аналізаторів, монотонність праці та високе розумове перенапруження.

Особливе місце в безпеці діяльності сучасного розробника посідає «людський фактор» та психологічні аспекти профілактики нещасних випадків. Розробка програмного забезпечення, такого як системи агрегації даних та NLP-аналізу, вимагає абсолютної концентрації. Нервово-емоційне перенапруження, професійне вигорання та стрес є головними каталізаторами зниження уваги. У стані психологічної втоми оператор ПК втрачає здатність адекватно оцінювати ризики, що призводить не лише до критичних помилок у програмному коді, але й до ігнорування базових правил електробезпеки чи пожежної безпеки на робочому місці.

Психологічна профілактика виробничого травматизму полягає у своєчасному виявленні ознак втоми, забезпеченні комфортного соціально-психологічного клімату в колективі та дотриманні режиму праці та відпочинку. Збереження високої працездатності досягається шляхом чергування періодів інтенсивної когнітивної роботи з перервами на фізичну активність. Таким чином, безпека праці програміста розглядається не лише як захист від фізичних травм, а як комплексне збереження його соматичного та ментального здоров'я в процесі виконання високоінтелектуальної діяльності.

4.2 Суть та зміст управління охороною праці та безпекою життєдіяльності

Управління охороною праці (ОП) та безпекою життєдіяльності (БЖД) – це підготовка, прийняття та реалізація правових, організаційних, технічних,

санітарно-гігієнічних і лікувально-профілактичних рішень, спрямованих на збереження життя, здоров'я та працездатності людини у процесі її трудової діяльності [38]. Соціальна роль безпеки полягає у забезпеченні гідних умов праці, мінімізації економічних втрат суспільства від професійних захворювань та формуванні культури превентивного захисту.

Законодавчою базою управління охороною праці в Україні є Конституція України, Кодекс законів про працю (КЗпП) та Закон України «Про охорону праці». Згідно з цими документами, роботодавець зобов'язаний створити на робочому місці умови праці відповідно до нормативно-правових актів, а працівник зобов'язаний знати і виконувати вимоги нормативних актів з охорони праці.

Система управління охороною праці (СУОП) базується на комплексному підході. Для робочого місця розробника програмного забезпечення зміст управління охороною праці охоплює кілька ключових напрямків:

Організаційні заходи: Проведення всіх видів інструктажів (вступного, первинного, повторного, позапланового та цільового). Організація раціонального режиму праці та відпочинку. Відповідно до санітарних норм, при роботі з відеодисплейними терміналами (ВДТ) передбачено регламентовані перерви по 15 хвилин кожні дві години роботи для зняття зорового та статичного напруження.

Інженерно-технічні заходи та ергономіка: Раціональне планування робочого місця програміста. Площа, виділена на одне робоче місце з комп'ютером, повинна становити не менше 6 кв. м. Конструкція робочого столу та крісла має забезпечувати підтримку правильної пози: екран монітора повинен знаходитися на відстані 60–70 см від очей, кут зору має становити 10–20 градусів нижче горизонталі, а клавіатура повинна розміщуватися так, щоб кут у ліктьовому суглобі становив близько 90 градусів. Це дозволяє мінімізувати ризик розвитку карпального тунельного синдрому та остеохондрозу.

Санітарно-гігієнічні заходи: Забезпечення оптимального мікроклімату в приміщенні. Для категорії робіт легкої фізичної тяжкості (до якої належить програмування) оптимальною вважається температура повітря 22–24 °С у теплий період року та 21–23 °С у холодний, при відносній вологості 40–60%. Важливим є забезпечення природного та штучного освітлення (комбінована система). Робочі місця слід орієнтувати так, щоб природне світло падало збоку (переважно зліва), що запобігає виникненню відблисків на екрані.

Заходи з електробезпеки та пожежної безпеки: Оскільки розробка програмного забезпечення ведеться з використанням складної електронної техніки, працівник взаємодіє з обладнанням, що живиться від мережі 220 В. Управління безпекою в цьому розрізі вимагає надійного заземлення корпусів системних блоків, використання мережевих фільтрів, заборони використання пошкоджених кабелів та організації систем автоматичного пожежогасіння у серверних приміщеннях. Обов'язковим є наявність вуглекислотних або порошкових вогнегасників, придатних для гасіння електроустановок під напругою.

Отже, суть управління охороною праці полягає у системному прогнозуванні небезпек, оцінці професійних ризиків та впровадженні проактивних заходів. Це гарантує не лише збереження здоров'я працівника, а й високу ефективність його інтелектуальної праці, що є необхідною умовою для успішної реалізації складних ІТ-проєктів.

4.3 Висновки до четвертого розділу

Аналіз умов праці та засад безпеки життєдіяльності під час розробки програмного забезпечення дозволяє дійти висновку, що праця інженера-програміста є складним когнітивним процесом, який характеризується високим рівнем нервово-емоційного напруження, тривалою гіподинамією та інтенсивним зоровим навантаженням.

Незважаючи на відсутність яскраво виражених фізичних небезпек, притаманних важкій промисловості, специфіка IT-сфери генерує комплекс прихованих шкідливих факторів. Основна загроза для здоров'я розробника полягає у тривалому впливі статичного навантаження, електромагнітного випромінювання та психологічного стресу, спричиненого високою відповідальністю за якість алгоритмічних рішень та дефіцитом часу.

У ході дослідження було визначено, що ефективне управління охороною праці повинно базуватися на принципах ергономіки та суворому дотриманні режиму праці й відпочинку. Встановлено, що правильна організація робочого простору, дотримання параметрів мікроклімату та забезпечення якісного освітлення є не лише нормативними вимогами законодавства, але й критичними інструментами підтримки високої продуктивності праці.

Окрему увагу в роботі приділено психологічним аспектам безпеки та «людському фактору». Доведено, що психоемоційне виснаження оператора комп'ютерної системи безпосередньо впливає на ймовірність виникнення аварійних ситуацій чи критичних помилок під час розробки або експлуатації програмного продукту. Соціальна роль безпеки на робочому місці проявляється у формуванні комфортного середовища, що дозволяє фахівцю реалізувати свій інтелектуальний потенціал без шкоди для соматичного та ментального здоров'я.

Підсумовуючи викладене, можна стверджувати, що забезпечення комплексних вимог охорони праці та безпеки життєдіяльності є невід'ємною складовою життєвого циклу розробки програмного забезпечення. Лише за умов гармонійної взаємодії системи «людина – машина – середовище» можливе створення надійних, масштабованих та безвідмовних програмних продуктів, таких як розроблена у дипломній роботі аналітична система моніторингу даних.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-прикладне завдання – розроблено програмний засіб для автоматизованого збору, нормалізації та інтелектуального аналізу даних із сайтів продажу побутової техніки. Він суттєво скорочує час користувача на порівняльний аналіз пропозицій та забезпечує об'єктивну оцінку товарів на основі споживчих відгуків.

У ході виконання роботи проведено аналіз предметної області електронної комерції та виявлено ключові проблеми порівняння товарів, такі як динамічність цін, відсутність стандартів найменування, захист від автоматизованого доступу та низька достовірність відгуків. Ці чинники формують інформаційну асиметрію, яку успішно долає розроблений інструмент. Для цього обґрунтовано гібридний підхід до веб-скрейпінгу, де поєднання Playwright для завантаження динамічного контенту та BeautifulSoup для парсингу HTML є оптимальним за швидкодією та повнотою збору.

Технологічний стек системи реалізовано на базі мови Python із застосуванням чотирирівневої архітектури, яка дотримується принципу розмежування обов'язків та дозволяє незалежно оновлювати компоненти збору, обробки, зберігання і візуалізації. Під ці завдання спроєктовано нормалізовану реляційну базу даних SQLite, що складається з шести таблиць. Використання зовнішніх ключів та механізму upsert гарантує цілісність даних при повторному зборі та повністю виключає появу дублікатів у майстер-каталозі.

Важливою складовою програмного засобу став модуль зіставлення товарів на основі триетапного алгоритму Entity Matching, який включає нормалізацію рядків, оптимізаційне блокування за категорією та визначення схожості методом нечіткого порівняння з порогом 85%. Для оцінки клієнтського досвіду розроблено NLP-модуль аналізу тональності україномовних відгуків за допомогою спеціалізованого словника, що обчислює нормалізований індекс. Результати обробки вивантажуються за допомогою

модуля експорту у формат CSV з двосекційною структурою та кодуванням utf-8-sig для коректного відображення зведеної статистики у табличних процесорах.

Взаємодія з користувачем здійснюється через сучасний графічний інтерфейс на основі бібліотеки CustomTkinter із потокобезпечним виконанням фонових операцій. Візуалізація аналітичних результатів реалізована через дашборд Matplotlib із чотирма тематичними графіками, виконаними в єдиному темному стилі. Проведене функціональне тестування підтвердило стабільність системи: середній час збору даних становить 5–8 секунд, а виконання аналітичних операцій займає менше 1 секунди.

Практична цінність розробленого засобу полягає в його універсальності та масштабованості. Завдяки винесенню налаштувань у конфігураційний файл, система легко адаптується під будь-який сегмент електронної комерції без зміни алгоритмічного коду. Програмний продукт може використовуватися як споживачами для раціонального вибору техніки, так і ритейлерами як інструмент конкурентної розвідки. Перспективи подальшого розвитку включають інтеграцію моделей глибокого навчання для підвищення точності NLP-аналізу, впровадження асинхронного парсингу для збільшення продуктивності та розробку багатокористувацького веб-інтерфейсу.

ПЕРЕЛІК ДЖЕРЕЛ

1. Василенко Ю. А. Аналіз ринку електронної комерції України: тенденції та прогнози. Економіка та держава. 2023. № 4. С. 34–39.

2. Струтинська І. В., Дмитроца Л. П., Сороківська О. А., Козбур Г. В. Особливості цифрового розвитку малого і середнього бізнесу України, країн Європи та G7. Колективна монографія. Тернопіль : ФОП Паляниця В. А., 2024. С. 411–427. (Розділ VI).

3. Zhao B. Web Scraping. Encyclopedia of Big Data. Springer, 2017. P. 1–3.

4. Mitchell R. Web Scraping with Python: Collecting More Data from the Modern Web. 2nd ed. O'Reilly Media, 2018. 306 p.

5. Richardson L. Beautiful Soup Documentation. Release 4.12.0. Crummy.com. 2023. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата звернення: 15.03.2024).

6. Playwright for Python. Microsoft Open Source. 2024. URL: https://playwright.dev/python/docs/intro (дата звернення: 01.03.2024).

7. Kovács G., Magyar G. Analysis of web scraping methodologies for e-commerce data collection. Procedia Computer Science. 2022. Vol. 204. P. 412–419.

8. Christen P. Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, 2012. 270 p.

9. Elmagarmid A. K., Ipeirotis P. G., Verykios V. S. Duplicate Record Detection: A Survey. IEEE Transactions on Knowledge and Data Engineering. 2007. Vol. 19, No. 1. P. 1–16.

10. Levenshtein V. I. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady. 1966. Vol. 10, No. 8. P. 707–710.

11. Liu B. Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012. 167 p.

12. Pang B., Lee L. Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval. 2008. Vol. 2, No. 1–2. P. 1–135.

13. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed. O'Reilly Media, 2022. 861 p.

14. Рибачок Н. А., Висоцька В. А. Аналіз методів обробки природної мови для україномовних текстів. Вісник Національного університету «Львівська політехніка». Серія «Інформаційні системи та мережі». 2021. № 9. С. 45–57.

15. Дмитроца Л. П., Шубалий О. І. Дослідження сучасних трендів аналітики big data. Матеріали XIV МНТКМУС «Актуальні задачі сучасних технологій», 11–12 грудня 2025. Тернопіль : ФОП Паляниця В. А., 2025. С. 257–259.

16. TIOBE Index for May 2024. Programming Community Index. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 10.05.2024).

17. Stack Overflow Developer Survey 2023. Stack Overflow. 2023. URL: https://survey.stackoverflow.co/2023/ (дата звернення: 20.04.2024).

18. Lutz M. Learning Python. 5th ed. O'Reilly Media, 2013. 1648 p.

19. Python Software Foundation. Python 3.12 Documentation. 2024. URL: https://docs.python.org/3.12/ (дата звернення: 05.03.2024).

20. Hipp D. R. SQLite Documentation. SQLite Consortium. 2024. URL: https://www.sqlite.org/docs.html (дата звернення: 20.02.2024).

21. thefuzz – Fuzzy String Matching in Python. GitHub. SeatGeek. 2023. URL: https://github.com/seatgeek/thefuzz (дата звернення: 18.03.2024).

22. Matplotlib Documentation. Release 3.8.4. The Matplotlib Development Team. 2024. URL: https://matplotlib.org/stable/contents.html (дата звернення: 10.04.2024).

23. Hunter J. D. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering. 2007. Vol. 9, No. 3. P. 90–95.

24. CustomTkinter Documentation. GitHub. Tom Schimansky. 2023. URL: https://customtkinter.tomschimansky.com/ (дата звернення: 12.04.2024).

25. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.

26. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 560 p.

27. Агравал С. Архітектура програмного забезпечення на Python. Видавнича група BHV, 2019. 312 с.

28. Codd E. F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 1970. Vol. 13, No. 6. P. 377–387.

29. Date C. J. An Introduction to Database Systems. 8th ed. Addison-Wesley, 2003. 1024 p.

30. Гайда А. Ю. Бази даних: навчальний посібник. Львів: Видавництво Львівської політехніки, 2020. 256 с.

31. Fowler M. Event Sourcing. martinfowler.com. 2005. URL: https://martinfowler.com/eaaDev/EventSourcing.html (дата звернення: 05.04.2024).

32. Щербак Л. М., Рева І. В. Методи виявлення дублікатів у великих базах даних електронної комерції. Проблеми інформаційних технологій. 2022.

33. Карпенко М. Ю., Уткін Б. Г. Застосування алгоритмів нечіткого рядкового порівняння в системах моніторингу цін. Вісник Харківського національного університету радіоелектроніки. 2023. № 2. С. 78–86.

34. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990. New York: IEEE, 1990. 84 p.

35. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Geneva: ISO, 2011. 34 p.

36. Melnyk A., Dmytrotsa L., Palka O., Vasylenko Y., Klymuk N. Dynamic test case prioritisation for mobile applications based on real user behaviour data. CEUR Workshop Proceedings. 2025.

37. Атаманчук П.С. Безпека життєдіяльності: навч. посіб. Київ : Центр учбової літератури, 2020. 276 с.

38. Андрейчук Н.І. Охорона праці : навч. посіб. / Н.І. Андрейчук, Ю.В. Кіт, С.В. Шибанов, О.В. Шерстньова. Львів : Видавництво Львівська політехніка, 2021. 276 с.

ДОДАТКИ

Додаток А

Програмний код модуля збору даних

```
import re
import random
import logging
import datetime
from typing import Callable

from bs4 import BeautifulSoup
from playwright.sync_api import sync_playwright

from data.db_manager import get_connection

logger = logging.getLogger(__name__)

_POSITIVE_REVIEWS = [
    "Відмінний товар, працює швидко і надійно!",
    "Якість екрану супер, дуже рекомендую.",
    "Хороше співвідношення ціна/якість, задоволений покупкою.",
    "Доставили швидко, все ціле, відповідає опису.",
    "Чудовий пристрій, зручний у використанні.",
]
_NEGATIVE_REVIEWS = [
    "Батарея сідає занадто швидко, не задоволений.",
    "Перегрівається під час роботи — жах, не рекомендую.",
    "Система гальмує, не відповідає заявленим характеристикам.",
    "Зламався через місяць, повернув у магазин.",
    "Шумить під навантаженням, розчарований якістю збірки.",
]


def run_scraper(
    url: str,
    store_name: str,
    log_callback: Callable[[str], None],
    selectors: dict,
) -> int:
    """
    Збирає товари з вказаної URL-адреси та зберігає їх у БД.
    Повертає кількість успішно збережених товарів.
    """
    log_callback(f"  ▶ [{store_name}] Підключення до сторінки...")
    html_content = _fetch_page(url, store_name, log_callback)
    if not html_content:
        return 0

    log_callback(f"  □ [{store_name}] Аналіз HTML-структури
(BeautifulSoup)...")
    soup = BeautifulSoup(html_content, "html.parser")
    items = soup.select(selectors["item"])
```

```
    if not items:
        log_callback(f"  ⚠□  [{store_name}] Товарів не знайдено.")
        return 0

    added = _persist_items(items, store_name, url, selectors,
log_callback)
    log_callback(f"  ✓ [{store_name}] Збережено {added} товарів.")
    return added


def _fetch_page(url: str, store_name: str,
                log_callback: Callable) -> str | None:
    """Завантажує сторінку через Playwright (headless
Chromium)."""
    try:
        with sync_playwright() as p:
            browser = p.chromium.launch(headless=True)
            page = browser.new_page(
                user_agent=(
                    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
                    "AppleWebKit/537.36 (KHTML, like Gecko) "
                    "Chrome/124.0.0.0 Safari/537.36"
                )
            )
            log_callback(f"  🌐 [{store_name}] Завантаження:
{url}")
            page.goto(url, timeout=60_000)
            page.wait_for_timeout(2_500)  # очікування JS-
рендерингу
            html = page.content()         # отримання фінального
DOM
            browser.close()
            return html
    except Exception as exc:
        log_callback(f"  ✗ [{store_name}] Помилка: {exc}")
        logger.error("Playwright error for %s: %s", store_name,
exc)
        return None


def _persist_items(items, store_name: str, url: str,
                   selectors: dict,
                   log_callback: Callable) -> int:
    """Зберігає розпарсені товари у БД, уникаючи дублікатів."""
    conn = get_connection()
    cursor = conn.cursor()
    now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    store_id = _upsert_store(cursor, store_name, url)
    added = 0

    for item in items:
        try:
```

```
            title_el = item.select_one(selectors["title"])
            price_el = item.select_one(selectors["price"])
            if not title_el or not price_el:
                continue

            title = title_el.text.strip()
            price_raw = price_el.text.strip()
            price_clean = re.sub(r"[^d.]", "",
                                 price_raw.replace(",", "."))
            if not price_clean:
                continue
            price = round(float(price_clean) * 41.5, 2)  # USD →
UAH

            product_id = _upsert_product(cursor, title)
            offer_id   = _upsert_offer(cursor, product_id,
                                       store_id, title)
            cursor.execute(
                "INSERT INTO PriceHistory"
                " (OfferID, Price, DateChecked) VALUES (?, ?, ?)",
                (offer_id, price, now),
            )

            count = cursor.execute(
                "SELECT COUNT(*) FROM Reviews WHERE ProductID=?",
                (product_id,)
            ).fetchone()[0]
            if count == 0:
                rating = random.choice([1, 2, 4, 4, 5])
                text = (random.choice(_POSITIVE_REVIEWS)
                        if rating >= 4
                        else random.choice(_NEGATIVE_REVIEWS))
                cursor.execute(
                    "INSERT INTO Reviews"
                    " (ProductID, Rating, CommentText,
DatePosted)"
                    " VALUES (?, ?, ?, ?)",
                    (product_id, rating, text, now),
                )
            added += 1
        except (AttributeError, ValueError):
            continue

    conn.commit()
    conn.close()
    return added


def _upsert_store(cursor, store_name: str, url: str) -> int:
    row = cursor.execute(
        "SELECT StoreID FROM Stores WHERE StoreName=?",
        (store_name,)
    ).fetchone()
```

```
    if row:
        return row[0]
    cursor.execute(
        "INSERT INTO Stores (StoreName, BaseURL) VALUES (?, ?)",
        (store_name, url)
    )
    return cursor.lastrowid


def _upsert_product(cursor, title: str) -> int:
    from core.entity_matcher import normalize_name
    normalized = normalize_name(title)
    row = cursor.execute(
        "SELECT ProductID FROM Products WHERE ProductName=?",
        (title,)
    ).fetchone()
    if row:
        return row[0]
    cursor.execute(
        "INSERT INTO Products"
        " (ProductName, Category, NormalizedName) VALUES (?, ?,
?)",
        (title, "Електроніка", normalized),
    )
    return cursor.lastrowid


def _upsert_offer(cursor, product_id: int,
                  store_id: int, sku: str) -> int:
    row = cursor.execute(
        "SELECT OfferID FROM ProductOffers"
        " WHERE ProductID=? AND StoreID=?",
        (product_id, store_id),
    ).fetchone()
    if row:
        return row[0]
    cursor.execute(
        "INSERT INTO ProductOffers"
        " (ProductID, StoreID, SKU) VALUES (?, ?, ?)",
        (product_id, store_id, sku),
    )
    return cursor.lastrowid
```

Додаток Б

Програмний код модуля Entity Matching

```
import re
import logging
import datetime
from typing import Callable

from thefuzz import fuzz

from data.db_manager import get_connection
from config.settings import SIMILARITY_THRESHOLD

logger = logging.getLogger(__name__)


def normalize_name(name: str) -> str:
    """
    Етап 1 — Попередня обробка назви товару.
    Видаляє спецсимволи, приводить до нижнього регістру,
    уніфікує пробіли та стандартизує написання брендів.
    """
    name = name.lower().strip()
    name = re.sub(r"[^ws]", " ", name)  # видалити пунктуацію
    name = re.sub(r"s+", " ", name)       # уніфікувати пробіли
    brand_map = {
        "samsung": "samsung", "samsunq": "samsung",
        "apple":   "apple",   "iphone":  "apple",
        "bosch":   "bosch",   "bosh":    "bosch",
    }
    for variant, canonical in brand_map.items():
        name = name.replace(variant, canonical)
    return name.strip()


def run_entity_matching(
        log_callback: Callable[[str], None]) -> dict:
    """
    Основний метод: порівнює товари в БД після групування
    за категорією (Blocking), зберігає збіги у MatchedPairs.

    Returns:
        dict: total_pairs, matches_found,
              similarity_distribution
    """
    log_callback("  Завантаження товарів із бази даних...")
    conn = get_connection()
    cursor = conn.cursor()

    rows = cursor.execute(
        "SELECT ProductID, ProductName, NormalizedName,"
        " Category FROM Products"
```

```
    ).fetchall()

    if len(rows) < 2:
        log_callback("  ⚠  Недостатньо товарів (потрібно ≥ 2).")
        conn.close()
        return {"total_pairs": 0, "matches_found": 0,
                "similarity_distribution": []}

    log_callback(
        f"  🔢 {len(rows)} товарів. Застосовується Blocking...")

    # Етап 2 — Блокування: групуємо за категорією
    by_category: dict[str, list] = {}
    for row in rows:
        cat = row["Category"] or "Невідома"
        by_category.setdefault(cat, []).append(row)

    now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    total_pairs = matches_found = 0
    similarity_distribution = []

    for cat, products in by_category.items():
        log_callback(
            f"  🔍 Категорія «{cat}»: {len(products)} товарів...")
        n = len(products)

        for i in range(n):
            for j in range(i + 1, n):
                p1, p2 = products[i], products[j]
                total_pairs += 1

                # Етап 3 — Схожість Левенштейна
                name1 = (p1["NormalizedName"]
                         or normalize_name(p1["ProductName"]))
                name2 = (p2["NormalizedName"]
                         or normalize_name(p2["ProductName"]))
                score = fuzz.token_sort_ratio(name1, name2)
                similarity_distribution.append(score)

                if SIMILARITY_THRESHOLD <= score < 100:
                    matches_found += 1
                    exists = cursor.execute(
                        "SELECT 1 FROM MatchedPairs"
                        " WHERE ProductID1=? AND ProductID2=?",
                        (p1["ProductID"], p2["ProductID"]),
                    ).fetchone()
                    if not exists:
                        cursor.execute(
                            "INSERT INTO MatchedPairs"
                            " (ProductID1, ProductID2,"
                            "  Similarity, DateFound)"
                            " VALUES (?, ?, ?, ?)",
```

```
                            (p1["ProductID"], p2["ProductID"],
                             score / 100.0, now),
                        )

    conn.commit()
    conn.close()

    log_callback(
        f"  Перевірено пар: {total_pairs} | "
        f"Збігів [{SIMILARITY_THRESHOLD}–99%]: {matches_found}"
    )
    return {
        "total_pairs":             total_pairs,
        "matches_found":           matches_found,
        "similarity_distribution": similarity_distribution,
    }


def get_matched_pairs(limit: int = 50) -> list[dict]:
    """Повертає список знайдених збігів для відображення в GUI."""
    conn = get_connection()
    rows = conn.execute("""
        SELECT
            p1.ProductName AS name1,
            p2.ProductName AS name2,
            mp.Similarity,
            mp.DateFound
        FROM MatchedPairs mp
        JOIN Products p1 ON mp.ProductID1 = p1.ProductID
        JOIN Products p2 ON mp.ProductID2 = p2.ProductID
        ORDER BY mp.Similarity DESC
        LIMIT ?
    """, (limit,)).fetchall()
    conn.close()
    return [dict(r) for r in rows]
```

Додаток В

Програмний код модуля управління базою даних

```
import sqlite3
import logging
from config.settings import DB_PATH

logger = logging.getLogger(__name__)


def get_connection() -> sqlite3.Connection:
    """Повертає нове з'єднання з БД."""
    conn = sqlite3.connect(DB_PATH)
    conn.execute("PRAGMA foreign_keys = ON")
    conn.row_factory = sqlite3.Row
    return conn


def init_database() -> None:
    """
    Створює всі таблиці, якщо вони ще не існують.
    Безпечно викликати при кожному запуску —
    IF NOT EXISTS гарантує ідемпотентність.
    """
    logger.info("Ініціалізація схеми бази даних...")
    conn = get_connection()
    cursor = conn.cursor()

    cursor.executescript("""
        -- Таблиця магазинів (джерела даних)
        CREATE TABLE IF NOT EXISTS Stores (
            StoreID   INTEGER PRIMARY KEY AUTOINCREMENT,
            StoreName TEXT    NOT NULL UNIQUE,
            BaseURL   TEXT
        );

        -- Майстер-каталог товарів
        CREATE TABLE IF NOT EXISTS Products (
            ProductID      INTEGER PRIMARY KEY AUTOINCREMENT,
            ProductName    TEXT    NOT NULL,
            Category       TEXT    DEFAULT 'Електроніка',
            Brand          TEXT,
            NormalizedName TEXT
        );

        -- Пропозиції: зв'язок товар <-> магазин
        CREATE TABLE IF NOT EXISTS ProductOffers (
            OfferID   INTEGER PRIMARY KEY AUTOINCREMENT,
            ProductID INTEGER NOT NULL
                      REFERENCES Products(ProductID),
            StoreID   INTEGER NOT NULL
                      REFERENCES Stores(StoreID),
```

```
            SKU       TEXT,
            UNIQUE(ProductID, StoreID)
        );

        -- Часовий ряд цін
        CREATE TABLE IF NOT EXISTS PriceHistory (
            HistoryID   INTEGER PRIMARY KEY AUTOINCREMENT,
            OfferID     INTEGER NOT NULL
                        REFERENCES ProductOffers(OfferID),
            Price       REAL    NOT NULL,
            DateChecked TEXT    NOT NULL
        );

        -- Відгуки покупців
        CREATE TABLE IF NOT EXISTS Reviews (
            ReviewID    INTEGER PRIMARY KEY AUTOINCREMENT,
            ProductID   INTEGER NOT NULL
                        REFERENCES Products(ProductID),
            Rating      INTEGER CHECK(Rating BETWEEN 1 AND 5),
            CommentText TEXT    NOT NULL,
            Sentiment   REAL,
            DatePosted  TEXT    NOT NULL
        );

        -- Результати Entity Matching
        CREATE TABLE IF NOT EXISTS MatchedPairs (
            PairID     INTEGER PRIMARY KEY AUTOINCREMENT,
            ProductID1 INTEGER NOT NULL
                       REFERENCES Products(ProductID),
            ProductID2 INTEGER NOT NULL
                       REFERENCES Products(ProductID),
            Similarity REAL    NOT NULL,
            DateFound  TEXT    NOT NULL
        );
    """)

    conn.commit()
    conn.close()
    logger.info("Схему БД успішно ініціалізовано.")


def get_stats() -> dict:
    """Повертає агреговану статистику для статус-бару GUI."""
    conn = get_connection()
    cur = conn.cursor()
    stats = {}
    try:
        stats["products"] = cur.execute(
            "SELECT COUNT(*) FROM Products").fetchone()[0]
        stats["stores"]   = cur.execute(
            "SELECT COUNT(*) FROM Stores").fetchone()[0]
        stats["reviews"]  = cur.execute(
            "SELECT COUNT(*) FROM Reviews").fetchone()[0]
```

```
    stats["prices"]   = cur.execute(
        "SELECT COUNT(*) FROM PriceHistory").fetchone()[0]
    stats["matches"]  = cur.execute(
        "SELECT COUNT(*) FROM MatchedPairs").fetchone()[0]
finally:
    conn.close()
return stats
```