Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

Допущено до захисту

Завідувач кафедри комп'ютерних наук

доктор технічних наук, професор

Андрій БОНДАРЧУК

«01» червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
Спеціальність 122 Комп'ютерні науки
Освітня програма 122.00.01 Інформатика

Тема: ІНТЕЛЕКТУАЛЬНИЙ АГЕНТ МОНІТОРИНГУ ТА АНАЛІЗУ ЦІНОВИХ ТЕНДЕНЦІЙ НА МАРКЕТПЛЕЙСАХ

Виконав
студент групи ІНб-1-22-4.0д
Богоніс Максим Ярославович

_____
(підпис)

Науковий керівник
Доцент кафедри комп'ютерних наук
Варченко-Троценко Лілія Олександрівна

_____
(підпис)

Київ – 2026
Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики

Кафедра комп’ютерних наук

«Затверджую»
Завідувач кафедри комп’ютерних наук,
доктор технічних наук
Бондарчук Андрій Петрович

« 31 » жовтня 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Інтелектуальний агент моніторингу та аналізу цінових тенденцій на маркетплейсах»

Виконавець – студент групи ІНб-1-22-4.0д,
спеціальності 122 Комп’ютерні науки,
освітньої програми 122.00.01 Інформатика
Богоніс Максим Ярославович

1. Вихідні дані: законодавчо-нормативні акти України у сфері інформаційних технологій та електронної комерції; наукові й довідкові джерела з питань моніторингу цін, обробки вебданих, product matching і веброзробки; документація використаних програмних засобів.
2. Основні завдання: проаналізувати предметну область і сервіси порівняння цін; визначити проблеми збору, нормалізації, фільтрації та зіставлення товарних даних; спроєктувати архітектуру інтелектуального агента; реалізувати збір даних з Allo, MOYO, Stylus і Kibernetiki; розробити механізми обробки результатів і вебінтерфейс; провести тестування системи та визначити напрями розвитку.
3. Пояснювальна записка: обсяг – 79 с.; формат А4 комп’ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.
4. Графічні матеріали: презентація до захисту; структурна й архітектурна схеми системи; схема роботи інтелектуального агента; приклади інтерфейсу та результати тестування.
5. Додатки: структура програмного проєкту; фрагменти серверної реалізації; модулі колекторів онлайн-магазинів; приклади екранів вебсистеми; додаткові сценарії функціонального тестування.
6. Строк подання роботи на кафедру: «01» червня 2026 р.

Науковий керівник:
Доцент кафедри комп’ютерних наук
Варченко-Троценко Лілія Олександрівна

«31» жовтня 2025 р.

Виконавець:
Богоніс Максим Ярославович

«31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Етапи виконання роботи	Термін виконання	Примітка
1	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	Виконано
2	Аналіз проблеми, вивчення аналогів, формування цілей і завдань	01.11.25 - 11.01.26	Виконано
3	Аналіз предметної області моніторингу цін на маркетплейсах та існуючих сервісів порівняння цін	26.01.26 - 15.03.26	Виконано
4	Дослідження методів збору, нормалізації, фільтрації та зіставлення товарних пропозицій	16.03.26 - 31.03.26	Виконано
5	Розробка архітектури інтелектуального агента, структури даних та вебінтерфейсу	01.04.26 - 15.04.26	Виконано
6	Тестування роботи системи на різних типах пошукових запитів	16.04.26 - 30.04.26	Виконано
7	Впровадження механізмів збору даних, групування результатів і підготовка системи до демонстрації	01.05.26 - 15.05.26	Виконано
8	Підготовка пояснювальної записки, графічних матеріалів, додатків.	25.05.25 - 12.06.26	Виконано
9	Захист кваліфікаційної роботи	16.06.26	

«Затверджую»

Науковий керівник — доцент, Варченко-Троценко Л.О.

Індивідуальний план склав — Богоніс М.Я.

РЕФЕРАТ

Кваліфікаційна робота: 79 с., 12 рис., 9 табл., 4 дод., 36 джерел.

Актуальність теми зумовлена розвитком електронної комерції, великою кількістю онлайн-магазинів і постійною зміною цін. Ручне порівняння ускладнюється неоднорідністю назв товарів, наявністю аксесуарів у видачі та змішуванням нових і вживаних позицій.

Об’єктом дослідження є процес функціонування інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах.

Предметом дослідження є методи, моделі та програмні засоби збирання, нормалізації, фільтрації, зіставлення й відображення цінових даних з різних онлайн-магазинів.

Метою роботи є розроблення інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах, призначеного для автоматизованого збору, обробки, групування та наочного подання товарних пропозицій.

Основні результати. У роботі проаналізовано предметну область моніторингу цін, існуючі сервіси порівняння пропозицій і проблеми автоматизованого збору товарних даних. Спроєктовано архітектуру вебсистеми, реалізовано модулі збирання даних з Allo, MOYO, Stylus і Kibernetiki, а також механізми нормалізації, фільтрації, групування, сортування та відображення результатів.

Практичне значення роботи полягає у створенні прототипу вебсистеми для автоматизованої обробки цінових пропозицій: від пошукового запиту до формування зведених карток товарів. Систему можна використовувати як основу для сервісу порівняння цін, а її подальший розвиток може передбачати підключення нових джерел, історію цін, графіки та сповіщення про зміну вартості.

Ключові слова: інтелектуальний агент, моніторинг цін, маркетплейс, порівняння цін, вебсистема, нормалізація даних, фільтрація товарів, групування пропозицій, fastapi.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

Скорочення	Розшифрування
API	інтерфейс прикладного програмування
CSS	каскадні таблиці стилів
HTML	мова розмітки гіпертексту
HTTP	протокол передавання гіпертексту
JS	JavaScript
JSON	формат обміну структурованими даними
ORM	об’єктно-реляційне відображення
SQL	мова структурованих запитів
UI	користувацький інтерфейс
URL	уніфікований локатор ресурсу
UX	досвід взаємодії користувача із системою

ЗМІСТ

ВСТУП

Електронна комерція створила для користувача широкий вибір товарів, але водночас ускладнила пошук справді вигідної пропозиції. Для одного й того самого пристрою ціна може відрізнятися залежно від магазину, наявності акцій, залишків на складі, стану товару та часу оновлення сторінки. Тому ручне порівняння пропозицій у кількох джерелах часто перетворюється на тривалий процес, у якому користувач змушений самостійно перевіряти назви моделей, характеристики, ціни й актуальність знайдених результатів [22–25].

У цій роботі термін «маркетплейси» використано в широкому значенні – як сукупність вебплатформ електронної торгівлі та онлайн-магазинів, з яких можна отримувати товарні пропозиції. Практична частина роботи зосереджена на онлайн-магазинах Allo, MOYO, Stylus і Kibernetiki, що дало змогу перевірити роботу системи на кількох незалежних джерелах.

Для розв’язання цієї проблеми потрібна система, яка не обмежується простим зчитуванням цін зі сторінок магазинів. Вона має обробляти товарні назви, відрізняти основні товари від аксесуарів, зіставляти однакові або близькі моделі та подавати результати у структурованому вигляді. Саме така логіка покладена в основу розробленого інтелектуального агента [22, 23, 26].

Актуальність теми зумовлена також тим, що сучасні вебресурси мають різну структуру сторінок, різні формати представлення назв товарів і цін, а тому для побудови ефективної системи моніторингу недостатньо простого збирання інформації. Необхідно реалізувати логіку інтелектуального зіставлення товарних пропозицій, фільтрації нерелевантних результатів, нормалізації атрибутів товарів і коректного відображення зведених карток у користувацькому інтерфейсі [26–31].

Метою кваліфікаційної роботи є розроблення інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах, призначеного для автоматизованого збору, обробки, зіставлення та наочного подання товарних

пропозицій із різних онлайн-магазинів. Для досягнення поставленої мети необхідно розв’язати такі завдання:

1. Проаналізувати предметну область моніторингу цін на маркетплейсах та визначити основні проблеми автоматизованого збору цінової інформації;
2. Дослідити існуючі сервіси порівняння цін і підходи до обробки товарних пропозицій;
3. Спроєктувати архітектуру програмної системи, структуру її модулів та логіку взаємодії між ними;
4. Реалізувати механізми збору даних з кількох онлайн-магазинів, нормалізації атрибутів товарів, фільтрації нерелевантних позицій і групування однакових моделей;
5. Розробити вебінтерфейс для пошуку, фільтрації, сортування та перегляду зведених результатів;
6. Провести тестування системи на різних типах запитів і проаналізувати коректність отриманих результатів.

Об’єктом дослідження є процес функціонування інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах.

Предметом дослідження є методи, моделі та програмні засоби збирання, нормалізації, зіставлення та відображення цінових даних, а також алгоритми фільтрації й групування товарних пропозицій з різних онлайн-магазинів.

Методи дослідження: системний аналіз предметної області; порівняльний аналіз існуючих рішень; методи вебзбирання даних; нормалізація текстових атрибутів товарів; алгоритмічне групування та фільтрація даних; проєктування вебзастосунків; функціональне тестування програмного забезпечення [4, 26–31].

Практичне значення роботи полягає у створенні вебсистеми, яка автоматизує пошук і порівняння товарних пропозицій з кількох онлайн-магазинів. Розроблена система збирає дані з підтримуваних джерел, відсіює частину нерелевантних результатів, групує пропозиції однакових моделей і

подає їх у вигляді карток товарів. Такий прототип може бути використаний як основа для подальшого розвитку сервісу порівняння цін.

Галузь застосування результатів роботи охоплює сервіси електронної комерції, системи моніторингу цін, інформаційно-аналітичні вебплатформи, а також програмні рішення для підтримки прийняття рішень у сфері цифрових покупок і ринкового аналізу.

Результати роботи апробовано шляхом публікації тез доповіді на конференції Інформаційні технології – 2026 [5].

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПІДХОДІВ ДО МОНІТОРИНГУ ЦІН НА МАРКЕТПЛЕЙСАХ

1.1 Особливості моніторингу цін на сучасних маркетплейсах

Сучасний ринок електронної комерції характеризується високою динамічністю, великою кількістю продавців і постійною зміною асортименту товарів. Для більшості категорій техніки ціна одного й того самого або близького за характеристиками товару може відрізнятися залежно від магазину, часу оновлення сторінки, наявності акцій, програми лояльності, залишків на складі та інших чинників. У зв’язку з цим моніторинг цін на маркетплейсах і в онлайн-магазинах набуває важливого практичного значення як для кінцевого споживача, так і для розробників інформаційно-аналітичних систем [22–25].

Моніторинг цін можна розглядати як процес систематичного збирання, упорядкування, порівняння й аналізу цінової інформації з різних джерел з метою виявлення актуальної вартості товару, змін ціни в часі, цінових коливань між магазинами та загальних ринкових тенденцій. На відміну від звичайного пошуку товару, моніторинг передбачає не лише разове отримання значень цін, а й їх подальшу обробку, очищення, зіставлення та інтерпретацію [22–25].

Особливістю сучасних маркетплейсів є неоднорідність представлення товарних даних. Один і той самий товар може бути описаний у різних магазинах по-різному: з повною назвою моделі, з артикулом, з технічними характеристиками, з вказанням кольору, обсягу пам’яті, року випуску або без частини цих атрибутів. Крім того, у результатах пошуку поряд із основними товарами часто з’являються супутні позиції: чохли, адаптери, стилуси, захисне скло, клавіатури, навушники та інші аксесуари. Через це звичайне зіставлення рядків за назвою не дає достатньої точності та потребує додаткових механізмів фільтрації й нормалізації [26–31].

Ще однією важливою особливістю є різниця у структурі вебсторінок і способах розміщення інформації. Онлайн-магазини можуть використовувати різну HTML-структуру, різні класи елементів, різні шаблони назв і форматів цін. Часто одна й та сама сторінка містить як поточну ціну, так і стару ціну до знижки, технічну службову ціну, ціну для кредиту чи розстрочки, а також інформаційні блоки, які не мають безпосереднього відношення до результату пошуку. Для коректного моніторингу ці дані необхідно не просто зчитати, а правильно інтерпретувати [26, 27, 29].

Певні труднощі виникають і через наявність товарів у різному стані. На ринку техніки поряд із новими товарами часто зустрічаються вживані, відновлені або уцінені моделі. Такі пропозиції можуть суттєво відрізнятися за ціною, і якщо не враховувати цей фактор, користувач отримує викривлену картину ринку. У результаті система моніторингу має вміти розрізняти нові та вживані товари, коректно відображати їхній статус і давати можливість фільтрувати такі результати [22, 23, 31].

Окремою задачею є зіставлення однакових моделей між різними джерелами. Для цього система має виділяти ключові атрибути товару: бренд, модель, обсяг пам’яті, колір, покоління та модифікацію. Без такої нормалізації неможливо коректно об’єднати пропозиції різних магазинів у спільну картку товару [26–32].

Окрему роль у моніторингу відіграє чинник актуальності даних. Ціна є динамічною характеристикою, а отже система повинна або регулярно оновлювати інформацію, або працювати в режимі оперативного збору даних за запитом користувача. У другому випадку особливого значення набувають швидкість обробки запиту, стабільність колекторів даних і ефективність алгоритмів фільтрації результатів. Для користувача цінною є не будь-яка інформація, а саме така, що є актуальною на момент пошуку.

Моніторинг цін має також аналітичний вимір. Якщо система накопичує результати пошуку в часі, вона може бути основою для виявлення цінових тенденцій, сезонних змін, ефекту акційних кампаній, різниці між новими та

вживаними товарами, а також поведінки окремих магазинів щодо формування ціни. Таким чином, навіть базова система пошуку та порівняння цін є фундаментом для побудови ширшого інтелектуального агента, орієнтованого на аналіз цінових трендів [22, 23].

Для цієї роботи важливо, що моніторинг цін не зводиться до отримання числового значення з вебсторінки. У реальних онлайн-магазинах потрібно враховувати різні назви моделей, додаткові характеристики, стан товару, наявність аксесуарів у видачі та відмінності у структурі сторінок. Тому розроблювана система має виконувати не тільки збір даних, а й попередню обробку, фільтрацію та групування товарних пропозицій.

1.2 Аналіз існуючих сервісів порівняння та відстеження цін

Розвиток електронної комерції сприяв появі значної кількості сервісів, орієнтованих на пошук, порівняння та відстеження цін. Такі системи можуть мати різний формат: від класичних агрегаторів цін до маркетплейсів із вбудованими механізмами сортування пропозицій, а також спеціалізованих ресурсів для спостереження за змінами вартості в часі. Попри різницю у реалізації, усі вони мають спільну мету – надати користувачеві зручний доступ до інформації про товарні пропозиції з кількох джерел та допомогти вибрати найкращий варіант [22–25].

Однією з базових функцій таких сервісів є агрегація результатів з різних магазинів. У найпростішому випадку користувач вводить назву товару та отримує список пропозицій, відсортований за ціною, популярністю або умовами доставки. Такий підхід є корисним, але не завжди достатнім, оскільки товари з подібними назвами можуть відрізнятися за обсягом пам’яті, кольором, поколінням або станом. Відсутність глибшої нормалізації атрибутів часто призводить до того, що в одній видачі опиняються як релевантні моделі, так і супутні товари або варіанти іншої модифікації [22, 23, 25].

Більш розвинені сервіси порівняння цін намагаються представити користувачу не просто набір посилань, а структуровану інформацію: назву

моделі, ключові характеристики, діапазон цін, кількість магазинів, наявність відгуків, рейтинг продавців тощо. Такі рішення значно підвищують зручність користування. Водночас навіть вони часто мають обмеження, пов’язані з неоднорідністю джерел, дублюванням результатів або недостатньо точним розмежуванням між основними товарами та аксесуарами [26, 27, 29].

Окремий клас систем становлять сервіси відстеження цін. Їхня основна цінність полягає не в одноразовому порівнянні пропозицій, а в накопиченні історичних даних. Завдяки цьому користувач може спостерігати, як змінювалася вартість певної моделі в часі, коли відбувалося зниження ціни, чи є поточна пропозиція вигідною порівняно з попередніми періодами. Такі сервіси є корисними для аналітики, однак вони, як правило, потребують стабільної системи збору даних, регулярного оновлення та коректного зіставлення моделей між різними джерелами [22, 23].

Аналіз існуючих рішень дає змогу виділити низку типових переваг. По-перше, користувач отримує економію часу, оскільки не потрібно окремо переглядати кожен магазин. По-друге, сервіси порівняння цін підвищують прозорість ринку, дозволяючи швидко виявити ціновий розрив між продавцями. По-третє, у разі наявності системи фільтрів і категорій вони забезпечують зручну навігацію за типом товару, брендом, діапазоном цін та іншими параметрами.

Водночас існуючі сервіси мають і низку обмежень. Одним із ключових недоліків є недостатня якість узгодження товарних атрибутів. Якщо один магазин подає повну назву моделі з усіма характеристиками, а інший – скорочену або частково локалізовану версію, система може неправильно згрупувати пропозиції або, навпаки, помилково об’єднати різні товари в одну картку. Особливо це помітно для техніки, де навіть одна цифра у назві моделі, колір або обсяг пам’яті суттєво впливають на ідентифікацію товару [26–31].

Іншою поширеною проблемою є наявність нерелевантних результатів у пошуковій видачі. Якщо сервіс покладається переважно на текстове співпадіння запиту з назвою сторінки, до результатів можуть потрапляти

чохли, адаптери, клавіатури, скло та інші аксесуари, які містять назву цільового товару в описі сумісності. У підсумку користувач змушений вручну відокремлювати основні товари від допоміжних, що зменшує практичну цінність сервісу [27, 29, 30].

Ще одним суттєвим недоліком є нечітке відображення стану товару. Для кінцевого користувача різниця між новим, вживаним, відновленим та уціненим товаром має принципове значення, оскільки безпосередньо впливає на справедливість порівняння цін. Якщо система не виділяє ці категорії або не дозволяє їх фільтрувати, результати порівняння можуть бути некоректними з аналітичної точки зору [22, 23, 31].

Окремо варто звернути увагу на проблему прозорості алгоритмів групування. Багато існуючих сервісів не пояснюють, чому саме ті чи інші пропозиції були об’єднані в одну групу. Для користувача це може виглядати зручно, але для дослідницького або аналітичного застосування виникає потреба в більш контрольованій і зрозумілій логіці зіставлення. Саме тому під час розроблення власної системи варто приділити увагу алгоритмам нормалізації атрибутів і формуванню зведених карток на основі зрозумілих правил [26–36].

Аналіз наявних підходів показує, що сучасні сервіси порівняння та відстеження цін вирішують важливу прикладну задачу, однак не завжди достатньо добре справляються з глибокою нормалізацією даних, відсіканням аксесуарів, розмежуванням нових і вживаних товарів та точним групуванням пропозицій однакових моделей. Це формує підстави для розроблення власного інтелектуального агента, у якому увага буде зосереджена не лише на зборі цін, а й на якості обробки товарної інформації, точності зіставлення моделей та зручності кінцевого представлення результатів.

Порівняння наявних підходів показує, що для задачі моніторингу цін важливо поєднати кілька функцій в одному процесі: збір даних, перевірку релевантності, нормалізацію атрибутів і групування пропозицій. Саме ці функції були визначені як базові для розроблюваного інтелектуального агента.

Порівняльну характеристику основних типів сервісів порівняння та відстеження цін наведено в таблиці 1.1.

Таблиця 1.1

Порівняльна характеристика сервісів порівняння та відстеження цін

Тип сервісу	**Основне призначення**	**Переваги**	**Недоліки**	**Що враховано в розробленій системі**
Агрегатор цін	Порівняння вартості товару в кількох магазинах	Швидке зіставлення цін, економія часу користувача	Часто слабке групування моделей, можливі дублікати, нерелевантні результати	Реалізовано групування пропозицій у картки товарів і відображення діапазону цін
Маркетплейс із внутрішнім пошуком	Пошук товарів у межах однієї торгової платформи	Велика кількість пропозицій, зручний інтерфейс, вбудовані фільтри	Не дає повного міжмагазинного порівняння, часто змішує аксесуари й основні товари	Реалізовано пошук по кількох окремих магазинах із подальшим зведенням результатів
Сервіс відстеження цін	Аналіз зміни вартості товару в часі	Дає змогу виявляти цінові тенденції та періоди зниження ціни	Потребує накопичення історичних даних і стабільного збирання інформації	Поточна система створює основу для подальшого накопичення історії цін
Класичний пошук магазину	Пошук товару в одному магазині	Простота використання, висока швидкість доступу до товару	Потрібно вручну порівнювати кілька сайтів, важко зіставляти однакові моделі	Реалізовано централізований пошук із кількох джерел в одному інтерфейсі
Аналітичний сервіс моніторингу	Аналіз асортименту, доступності та цін на ринку	Дає структуровану картину ринку, підтримує аналітичні висновки	Висока складність реалізації, потреба в нормалізації даних	Реалізовано інтелектуальну фільтрацію, нормалізацію атрибутів і групування результатів

1.3 Проблеми автоматизованого збору та зіставлення товарних пропозицій

Під час розроблення системи моніторингу цін основна складність виникає не на етапі отримання сторінки магазину, а під час перетворення знайдених результатів у придатні для порівняння дані. Онлайн-магазини по-різному оформлюють назви товарів, ціни, доступність і технічні характеристики, тому однакові моделі не завжди можна зіставити простим порівнянням рядків [10, 14, 15].

Першою суттєвою проблемою є неоднорідність назв товарів. Один і той самий пристрій може бути представлений у скороченій, повній або частково локалізованій формі. Наприклад, у різних джерелах можуть використовуватися різні варіанти подання бренду, сімейства моделі, кольору, обсягу пам’яті, покоління процесора або артикулу. У результаті просте текстове порівняння назв не є достатнім для надійного об’єднання однакових пропозицій в одну картку товару, що вимагає переходу від роботи з сирими рядками до виділення структурованих атрибутів [26–31].

Не менш помітною проблемою є поява нерелевантних результатів у пошуковій видачі. Багато магазинів у відповідь на запит за назвою моделі повертають не лише сам товар, а й аксесуари, сумісні товари, витратні матеріали або супутні пристрої. Наприклад, для запитів за смартфонами або планшетами в результатах можуть з’являтися чохли, захисне скло, зарядні пристрої, стилуси, клавіатури, адаптери, кабелі. Якщо такі результати не відсіювати автоматично, якість зведеної видачі суттєво знижується, а користувач змушений вручну відокремлювати релевантні позиції від нерелевантних [27, 29, 30].

Окремої уваги потребує спосіб подання цінової інформації. На сторінці результатів пошуку або в картці товару магазин може виводити поточну ціну, стару ціну до знижки, ціну за акцією, службову ціну, кредитний платіж або інші числові значення, які не повинні потрапляти до підсумкової вибірки як

основна ціна товару. Крім того, іноді на сторінці присутні ціни на супутні товари або службові значення, які формально схожі на ціну, але не є нею по суті. Через це алгоритм повинен не лише знаходити числові значення, а й перевіряти їхній контекст [22, 23, 26].

На коректність порівняння також впливає стан товару. На ринку техніки можуть одночасно бути представлені нові, вживані, відновлені або уцінені товари. Такі пропозиції є корисними для користувача, однак вони не повинні безконтрольно змішуватися в одному наборі результатів. Якщо система не враховує стан товару, це спотворює цінову картину й ускладнює аналіз. Саме тому необхідно виявляти маркери вживаного чи відновленого товару та надавати користувачу можливість окремо працювати з такими пропозиціями [22, 23, 31].

П’ятою проблемою є складність групування однакових моделей між різними джерелами. Навіть якщо два магазини продають той самий пристрій, назви можуть відрізнятися за порядком слів, мовою написання, наявністю коду моделі чи службових уточнень. Додатково ситуацію ускладнює те, що окремі атрибути можуть бути подані неявно або взагалі бути відсутніми. Для розв’язання цієї задачі необхідно розробити механізм нормалізації, який виділяє бренд, сімейство, модель, обсяг пам’яті, колір та інші характеристики, а потім використовує їх для побудови зведених карток [26–32].

Шостою проблемою є наявність схожих, але не тотожних моделей. Наприклад, при пошуку однієї моделі смартфона можуть з’являтися версії Pro, Plus, Max, Air, mini, SE або інші модифікації. Якщо система не вміє розрізняти такі варіанти, вона може помилково об’єднати різні моделі або, навпаки, зайво розділити однакові товари. Алгоритм має враховувати не лише загальну назву, а й сильні варіанти моделі, які змінюють зміст пошукового запиту [31–36].

Сьомою проблемою є динамічність вебджерел. Онлайн-магазини періодично змінюють структуру сторінок, класи елементів, схеми відображення карток товарів і пошукових результатів. Через це навіть працездатний механізм збору даних потребує регулярної перевірки та

адаптації. У практичній реалізації це означає необхідність модульної архітектури колекторів, де кожен магазин обробляється окремим компонентом, а логіка парсингу може бути змінена без порушення роботи всієї системи [10, 14, 15].

Важливою є також проблема збереження балансу між точністю й універсальністю. Якщо правила фільтрації та зіставлення зробити надто жорсткими, система може втрачати частину релевантних пропозицій. Якщо ж правила будуть надто м’якими, у видачу потраплятиме багато шуму. Тому в інтелектуальному агенті необхідно реалізувати таку логіку обробки, яка буде достатньо гнучкою для різних типів техніки й різних джерел, але водночас забезпечуватиме прийнятну точність результатів [26–36].

У результаті автоматизований збір і зіставлення товарних пропозицій пов’язані з комплексом взаємопов’язаних проблем: неоднорідністю назв, наявністю аксесуарів у видачі, різними форматами цін, змішаними статусами товарів, труднощами групування моделей та нестабільністю вебструктур магазинів. Сукупність цих факторів підтверджує необхідність розроблення інтелектуального агента, у якому основна увага приділяється не лише збору даних, а й їх попередній обробці, нормалізації та логічному узгодженню.

Основні проблеми автоматизованого збору та зіставлення товарних пропозицій узагальнено в додатку А, таблиця А.1.

1.4 Постановка задачі розроблення інтелектуального агента

Аналіз предметної області, сучасних сервісів порівняння цін і типових проблем автоматизованого збору товарних даних дає підстави сформулювати задачу розроблення інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах як задачу створення програмної системи, здатної автоматизувати пошук, збирання, обробку, зіставлення та відображення цінової інформації з кількох джерел у зручному для користувача вигляді [4, 22, 26–31].

У межах цієї роботи інтелектуальний агент розглядається як програмний компонент, який діє за заданим запитом користувача, звертається до кількох онлайн-магазинів, збирає результати пошуку, очищує та нормалізує отримані дані, виявляє товарні атрибути, групує однакові або близькі пропозиції в зведені картки та надає можливість подальшої фільтрації і порівняння. Завдяки цьому розрізнені сторінки окремих магазинів перетворюються на єдине середовище аналізу товарних пропозицій.

Основною задачею системи є коректне зіставлення товарів з різних джерел. Для цього агент має розв’язувати низку підзадач: виділення сутностей із пошукового запиту, класифікацію товарної категорії, нормалізацію назв товарів, виявлення ключових атрибутів моделі та відсікання нерелевантних результатів. Лише після цього можливо побудувати підсумкові картки товарів, у яких різні пропозиції магазинів будуть віднесені до однієї моделі.

З урахуванням поставленої мети до системи висуваються такі функціональні вимоги:

- Прийом пошукового запиту користувача українською або англійською мовою;
- Визначення основних атрибутів запиту, зокрема бренду, сімейства моделі, модифікації, кольору, обсягу пам’яті та категорії товару;
- Збирання товарних пропозицій із кількох онлайн-магазинів;
- Виділення релевантних результатів і відсікання аксесуарів та сторонніх позицій;
- Виявлення й нормалізація характеристик товару;
- Групування однакових або близьких товарних пропозицій у спільні картки;
- Відображення магазинів, цін, доступності та стану товару;
- Фільтрація результатів за магазином, діапазоном цін та станом товару;
- Сортування карток за ціною;

- Забезпечення коректного відображення кількості видимих карток після застосування фільтрів.

Наведені функціональні вимоги відповідають логіці побудови прикладної вебсистеми для пошуку, обробки та подання інформації користувачеві [4, 5].

Поряд із функціональними вимогами важливими є й нефункціональні вимоги. До них належать: модульність архітектури, можливість додавання нових магазинів без повного перепроєктування системи, прийнятна швидкодія обробки запиту, читабельність інтерфейсу, зручність користування, а також стійкість до часткових змін у структурі вебсторінок окремих магазинів [4, 5, 7].

У межах кваліфікаційної роботи потрібно зосередити увагу на побудові працездатного прототипу вебсистеми, що реалізує повний цикл обробки пошукового запиту. Вхідними даними для агента є текстовий запит користувача та, за потреби, вибір категорії товару. Вихідними даними є набір зведених карток товарів, де для кожної моделі вказано основні атрибути, діапазон цін, перелік магазинів, наявність товару та позначка про вживаний стан, якщо вона є.

Для розв’язання поставленої задачі необхідно побудувати архітектуру системи, що складатиметься з кількох логічних рівнів. На першому рівні знаходиться інтерфейс користувача, який приймає запит і відображає результати. На другому рівні працює логіка пошуку та нормалізації, де виділяються сутності запиту, аналізуються назви товарів і виконується групування пропозицій. На третьому рівні розташовані колектори магазинів, які відповідають за збирання даних із конкретних джерел. За потреби така архітектура може бути доповнена рівнем збереження історичних даних для подальшого аналізу цінових тенденцій.

Для наочного подання загальної логіки роботи розроблюваної системи доцільно подати узагальнену схему задачі інтелектуального агента.

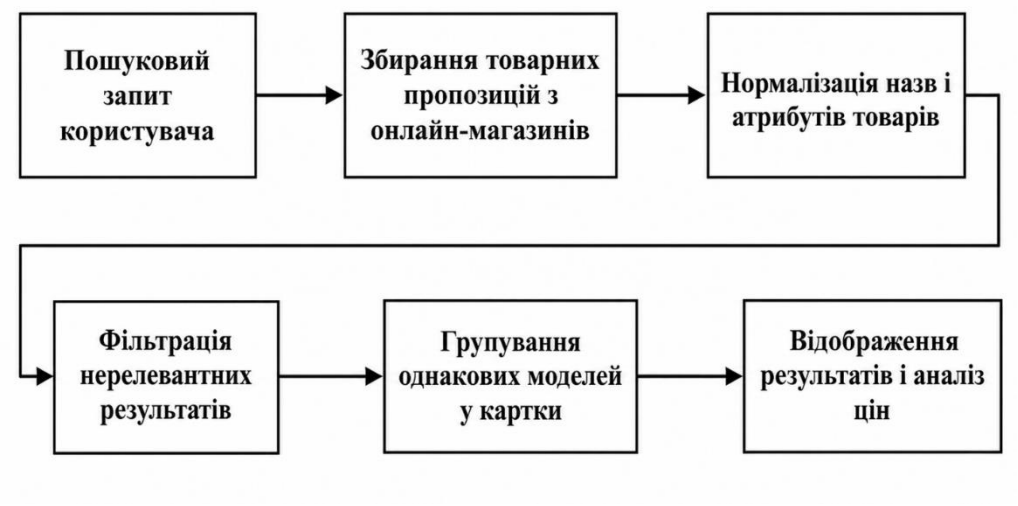

Рисунок 1.1. Узагальнена схема задачі розроблення інтелектуального агента

Як видно з рисунка 1.1, робота системи починається з обробки пошукового запиту користувача і завершується формуванням структурованого результату у вигляді згрупованих карток товарів для подальшого аналізу.

Постановка задачі також передбачає визначення критеріїв якості результату. Для цієї системи такими критеріями є:

- Релевантність знайдених товарів до пошукового запиту;
- Відсутність або мінімізація аксесуарів і нерелевантних позицій у видачі;
- Точність визначення характеристик товару;
- Правильність об’єднання пропозицій однакової моделі;
- Коректне розмежування нових і вживаних товарів;
- Зручність подальшої роботи з результатами через фільтри і сортування.

Зазначені критерії якості прямо пов’язані з підходами до product matching, нормалізації атрибутів і фільтрації релевантних результатів [26–36].

У межах кваліфікаційної роботи ставиться задача створення інтелектуального агента у вигляді вебсистеми, яка автоматизує пошук і порівняння товарних пропозицій з кількох онлайн-магазинів, виконує

нормалізацію та логічне зіставлення результатів, відфільтровує нерелевантні позиції і формує структуроване подання цінової інформації для подальшого аналізу. Саме розв’язання цієї задачі становить зміст практичної частини дослідження.

Висновки до розділу 1

У першому розділі розглянуто особливості моніторингу цін на маркетплейсах, проаналізовано існуючі сервіси порівняння та відстеження цін, а також визначено основні проблеми автоматизованого збору й зіставлення товарних пропозицій. Встановлено, що ключовими труднощами є неоднорідність назв товарів, різне подання цінової інформації, наявність аксесуарів у видачі, змішування нових і вживаних товарів та складність групування однакових моделей.

Проведений аналіз показав, що існуючі сервіси частково розв’язують задачу порівняння цін, однак мають обмеження щодо нормалізації атрибутів, фільтрації нерелевантних результатів і прозорості групування пропозицій. Це підтвердило доцільність створення власної системи, орієнтованої не лише на збір цін, а й на обробку товарної інформації.

У результаті сформульовано задачу розроблення інтелектуального агента як вебсистеми, що виконує збір, нормалізацію, фільтрацію, групування та відображення товарних пропозицій з кількох онлайн-магазинів. Отримані висновки стали основою для проєктування архітектури системи та визначення її функціональних вимог.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОГО АГЕНТА МОНІТОРИНГУ ТА АНАЛІЗУ ЦІНОВИХ ТЕНДЕНЦІЙ

2.1 Функціональні та нефункціональні вимоги до системи

Розроблення інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах потребує чіткого визначення вимог до майбутньої системи. Вимоги визначають межі функціонування програмного продукту, задають орієнтири для проєктування його архітектури та дозволяють оцінити, наскільки реалізоване рішення відповідає поставленій меті [4, 5, 7]. Для цієї роботи доцільно виділити функціональні та нефункціональні вимоги.

До функціональних вимог належать ті властивості системи, які безпосередньо пов’язані з її прикладним призначенням і забезпечують взаємодію з користувачем, обробку запиту та формування результатів.

Першою функціональною вимогою є можливість введення пошукового запиту користувачем. Система повинна приймати текстову назву товару українською або англійською мовою, а також дозволяти вибір категорії, якщо користувач бажає звузити область пошуку. Такий підхід підвищує гнучкість роботи системи й дає змогу використовувати як широкі, так і уточнені запити [19–21].

Другою вимогою є автоматизований збір результатів із кількох онлайн-магазинів. Інтелектуальний агент повинен звертатися до зовнішніх вебджерел, отримувати релевантні товарні пропозиції та формувати первинний набір даних для подальшого аналізу. У межах реалізованої системи таке збирання відбувається через окремі модулі-колектори для кожного підтримуваного магазину [10, 14, 15].

Третьою функціональною вимогою є виділення та нормалізація атрибутів товару. Система повинна вміти розпізнавати бренд, сімейство моделі, номер моделі, варіант виконання, колір, обсяг пам’яті, стан товару та інші характеристики, що впливають на точність порівняння. Саме ця вимога

дає змогу перейти від сирих назв товарів до структурованих даних, придатних для групування й фільтрації [26–31].

Четвертою вимогою є відсікання нерелевантних результатів. Пошукова видача онлайн-магазинів часто містить аксесуари, схожі моделі або товари суміжних категорій. Тому система повинна містити механізми фільтрації, які зменшують ризик потрапляння таких позицій до кінцевого списку карток.

П’ятою функціональною вимогою є групування однакових або близьких пропозицій. Якщо кілька магазинів пропонують одну й ту саму модель товару, система має об’єднати їх у спільну картку, у межах якої користувач бачить усі доступні пропозиції, діапазон цін і ключові характеристики. Це є однією з найважливіших функцій системи, оскільки вона робить міждерельне порівняння товарів зручним для користувача.

Шостою вимогою є коректне відображення цінової інформації та стану товару. Для кожної пропозиції повинні бути вказані ціна, магазин, доступність та, за наявності, позначка про вживаний стан. Для кожної картки повинні відображатися мінімальна та максимальна ціна серед доступних пропозицій, а також кількість магазинів або карток, що задовольняють поточні умови відбору.

Сьомою вимогою є підтримка користувацьких фільтрів і сортування. Система повинна надавати можливість відфільтровувати результати за магазином, діапазоном цін і ознакою вживаного товару, а також сортувати картки за зростанням або спаданням ціни. Такі засоби роблять результати пошуку більш гнучкими та придатними для аналізу.

Восьмою функціональною вимогою є збереження та оновлення даних у внутрішньому сховищі. Після обробки запиту система повинна мати можливість зберігати відомості про знайдені товари й пропозиції, що створює основу для повторного використання результатів та подальшого розширення системи до історичного аналізу цінових тенденцій.

Окрім функціональних, система повинна відповідати і нефункціональним вимогам, які визначають її якість, стабільність та придатність до подальшого розвитку.

Однією з основних нефункціональних вимог є модульність архітектури. Система має бути побудована так, щоб логіка збирання даних, обробки товарних назв, групування пропозицій та відображення результатів була розділена на окремі компоненти. Це спрощує підтримку програмного коду і дає змогу додавати нові магазини без перепроєктування всієї системи [10–18].

Другою важливою нефункціональною вимогою є розширюваність. Система повинна допускати подальше розширення функціоналу, зокрема підключення нових джерел даних, реалізацію історичного аналізу цін, додавання нових фільтрів або нових правил зіставлення товарів. Для дипломного проєкту це є важливою характеристикою, оскільки свідчить про практичну перспективність програмного рішення.

Третьою вимогою є прийнятна швидкодія. Обробка запиту користувача не повинна займати надто багато часу, інакше система втрачає зручність використання. Це означає, що механізми збирання даних, фільтрації та побудови карток мають бути реалізовані достатньо ефективно, а помилки окремих джерел не повинні блокувати роботу всієї системи.

Четвертою вимогою є стійкість до часткових збоїв. Якщо один із модулів збирання даних тимчасово не працює або зовнішнє джерело змінює структуру сторінки, система повинна зберігати працездатність для інших джерел і продовжувати формувати результати на основі доступних даних.

П’ятою нефункціональною вимогою є зручність користувацького інтерфейсу. Вебінтерфейс повинен бути інтуїтивно зрозумілим, містити логічно організовані блоки пошуку, фільтрів і результатів, а також забезпечувати наочне подання товарних карток і пропозицій магазинів.

Шостою вимогою є підтримуваність програмного коду. Структура проєкту повинна бути зрозумілою, з розподілом на шаблони, стилі, JavaScript-

логіку, модулі збирання даних, сервіси обробки та шари роботи з даними. Це полегшує виправлення помилок, тестування та подальший розвиток системи.

Сьомою нефункціональною вимогою є достатня точність зіставлення товарів. Для цієї системи точність є критично важливою, оскільки помилкове групування різних моделей або пропуск релевантних характеристик призводить до спотворення результатів порівняння. Саме тому логіка зіставлення товарів повинна враховувати сильні атрибути моделей і не обмежуватися лише простим текстовим пошуком [26–36].

Сформульовані вимоги визначили межі практичної реалізації системи. На їх основі було спроєктовано модульну структуру агента: окремі колектори для магазинів, серверну логіку обробки запиту, механізми нормалізації та групування пропозицій, а також вебінтерфейс для роботи з результатами.

Сформульовані функціональні та нефункціональні вимоги до системи подано в додатку А, таблиця А.2.

2.2 Архітектура програмної системи

Для реалізації інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах було обрано архітектуру вебсистеми клієнт-серверного типу. Такий підхід є доцільним, оскільки користувач взаємодіє із системою через браузер, а основна логіка збирання, обробки, нормалізації та групування даних виконується на серверній стороні. Обрана архітектура поєднує зручний інтерфейс користувача, централізовану логіку обробки та можливість підключення нових джерел даних [10, 13, 19–21].

Узагальнено архітектуру системи можна подати як сукупність чотирьох основних рівнів:

1. Рівень користувацького інтерфейсу;
2. Рівень прикладної логіки;
3. Рівень збирання та підготовки даних;
4. Рівень збереження даних.

На рівні користувацького інтерфейсу користувач вводить запит, вибирає категорію товару, переглядає результати пошуку, застосовує фільтри та сортування. Цей рівень реалізований у вигляді вебсторінок, що формуються на сервері та відображаються в браузері. До нього належать шаблони головної сторінки, сторінки результатів, базового макета, а також таблиці стилів і JavaScript-скрипти для клієнтської взаємодії. Основним завданням цього рівня є забезпечення зрозумілого й зручного подання даних [13, 19–21].

На рівні прикладної логіки виконується основна обробка запиту. Саме тут реалізовано механізми запуску пошуку, збирання результатів із зовнішніх джерел, нормалізації товарних назв, зіставлення пропозицій, побудови карток товарів і формування підсумкової відповіді для інтерфейсу. Цей рівень виступає центральною ланкою системи, оскільки об’єднує результати роботи всіх допоміжних модулів у єдиний логічний процес.

Ключову роль на цьому рівні відіграють сервіси обробки товарних даних. Один сервіс відповідає за інтерпретацію пошукового запиту та виділення атрибутів товару. Інший виконує збирання результатів із кількох джерел, їх попереднє очищення, групування та підготовку карток для відображення. Завдяки такому розподілу обов’язків система набуває більшої гнучкості та підтримуваності.

На рівні збирання та підготовки даних розташовані модулі-колектори для окремих онлайн-магазинів. Кожен колектор реалізує логіку звернення до конкретного джерела, обробку HTML-структури сторінок або інших доступних форматів даних, виділення назви товару, ціни, посилання, зображення, доступності та інших атрибутів. Таке рішення відокремлює особливості кожного магазину в окремий компонент [10, 14, 15].

Модульна організація колекторів є однією з важливих переваг архітектури. Вона дає змогу:

- Окремо тестувати кожен магазин;
- Локалізувати помилки;
- Додавати нові джерела даних;

- Підтримувати працездатність системи навіть у разі збою одного з колекторів.

Після збирання сирих даних система переходить до етапу їх нормалізації та логічного зіставлення. На цьому етапі із назв товарів виділяються бренд, сімейство, модель, колір, пам’ять, ознаки вживаного стану та інші характеристики. Далі система оцінює релевантність пропозицій щодо запиту користувача, відсікає аксесуари та нерелевантні товари, а після цього групує однакові або близькі моделі в зведені картки. Така логіка є основною інтелектуальною складовою розробленого агента.

На рівні збереження даних система працює з внутрішнім сховищем, у якому можуть бути зафіксовані продукти та знайдені для них пропозиції магазинів. База даних відокремлює тимчасові результати збирання від логіки їх подання, а також створює основу для подальшого розвитку системи в напрямі історичного накопичення цін і побудови аналітичних звітів [11, 12, 18]. У поточній реалізації рівень збереження також використовується для підтримки цілісності даних і оновлення результатів пошуку.

Маршрут обробки запиту в межах системи можна описати такою послідовністю дій. Спочатку користувач на головній сторінці вводить назву товару. Далі запит передається на сервер, де запускається механізм пошуку. Сервер активує колектори магазинів, збирає початковий список пропозицій, після чого результати проходять через модулі нормалізації та зіставлення. Потім система формує набір карток товарів, у кожній з яких зведено пропозиції кількох магазинів. На завершальному етапі підготовлені результати передаються у шаблон сторінки та відображаються користувачеві, а клієнтський JavaScript виконує локальну фільтрацію і сортування без повторного повного завантаження сторінки.

Архітектуру розробленої системи варто подати у вигляді багаторівневої схеми взаємодії основних компонентів.

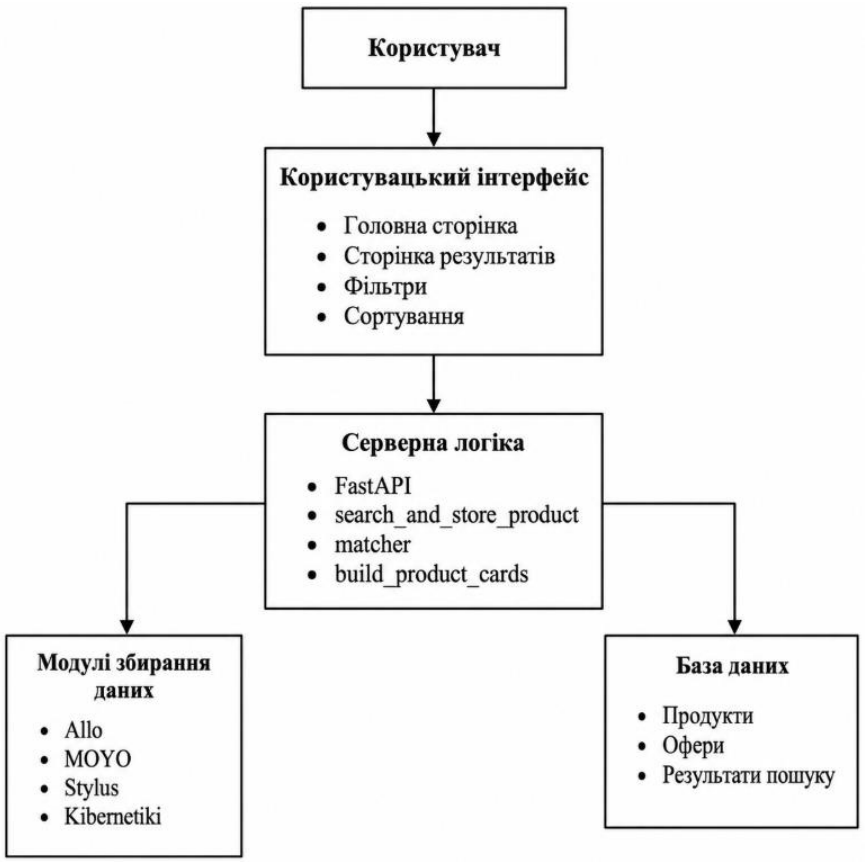

Рисунок 2.1. Архітектура програмної системи інтелектуального агента моніторингу цін

Рисунок 2.1 демонструє, що система побудована за клієнт-серверним принципом і включає рівень користувацького інтерфейсу, рівень прикладної логіки, модулі збирання даних з онлайн-магазинів і рівень збереження даних.

Обрана архітектура зручна для цієї задачі, оскільки кожен тип логіки винесено в окремий рівень. Колектори відповідають за особливості конкретних магазинів, серверна частина об’єднує й обробляє результати, а клієнтський інтерфейс відповідає за фільтрацію, сортування та перегляд карток товарів. Завдяки цьому зміна одного джерела даних не потребує повного перепроєктування системи.

Важливим архітектурним рішенням є також розмежування серверної та клієнтської логіки. Усі операції, пов’язані зі збиранням даних, нормалізацією, групуванням і підготовкою карток, виконуються на сервері. Натомість фільтрація за ціною, магазином і станом товару, а також сортування результатів виконуються на клієнтському рівні. Це зменшує навантаження на

сервер під час повторних дій користувача і водночас зберегти цілісність основної логіки обробки в одному місці [10, 13, 19–21].

Архітектура розробленої програмної системи є багаторівневою, модульною та орієнтованою на подальше розширення. Вона поєднує клієнтський вебінтерфейс, серверну бізнес-логіку, модулі збору товарних пропозицій і внутрішній рівень збереження даних. Саме така організація дає змогу практично реалізувати інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах.

Структуру програмного проєкту та основні модулі системи наведено в додатку А.

2.3 Проєктування структури даних і логіки обробки пропозицій

Під час проєктування інтелектуального агента одним із ключових завдань стало визначення такої структури даних, яка б дозволяла коректно зберігати сирі результати пошуку, перетворювати їх на нормалізовані сутності та надалі формувати зведені картки товарів для відображення у вебінтерфейсі. Для цього в системі було використано багаторівневий підхід до подання даних: від сирої товарної пропозиції конкретного магазину до нормалізованої моделі товару та підсумкової картки результату [11, 12, 18].

Базовою одиницею обробки в системі є окрема товарна пропозиція магазину. Така пропозиція містить щонайменше назву товару, ціну, посилання, доступність, джерело та додаткову службову інформацію, наприклад ознаку вживаного товару чи адресу зображення. Саме на цьому рівні дані надходять у систему після роботи модулів збирання. Оскільки пропозиції з різних магазинів можуть мати різний формат і різний ступінь деталізації, вони не можуть одразу відображатися як готовий результат пошуку. Спочатку їх потрібно уніфікувати та проаналізувати.

Наступним рівнем подання є набір нормалізованих атрибутів товару. Для кожної назви товару система виділяє такі характеристики, як бренд, сімейство моделі, модель, покоління, обсяг пам’яті, обсяг оперативної пам’яті,

тип накопичувача, процесор, графічний адаптер, розмір екрана, колір, стан товару, можливі варіанти моделі, коди модифікацій, а також ознаки аксесуарів. Саме така структура переводить текстовий опис, отриманий з магазину, у формалізовану модель, яку можна використовувати в алгоритмах порівняння та групування [26–31]. У коді ця логіка реалізована через окремі структури для атрибутів товару та сутності продукту, що містять як загальні ідентифікаційні поля, так і набір ключових характеристик.

Під час проєктування було важливо розділити поняття товарної пропозиції і сутності продукту. Пропозиція відображає конкретний екземпляр товару з певного магазину, тоді як сутність продукту є узагальненим уявленням про модель, яку шукає користувач. Такий поділ зберігає деталі кожного джерела і водночас дає змогу групувати пропозиції за ознаками однієї моделі.

Для задачі групування однакових моделей було передбачено формування зведених карток товарів. Кожна картка містить назву моделі, зображення, мінімальну та максимальну ціну, колір, обсяг пам’яті, обсяг оперативної пам’яті, доступність, перелік магазинів і вкладені пропозиції. У сервісі обробки результатів картки формуються шляхом групування пропозицій за набором ключових атрибутів, зокрема брендом, сімейством, моделлю, пам’яттю, оперативною пам’яттю та кольором. Після цього всередині картки відбувається об’єднання магазинних оферів, видалення дубльованих позицій і обчислення діапазону цін.

Окрему увагу під час проєктування було приділено логіці відображення неповних атрибутів. У реальних даних не всі магазини однаково повно вказують характеристики товару. Тому в системі використано механізм часткового доукомплектування атрибутів картки: якщо в одній пропозиції, що входить до групи, відсутній колір або пам’ять, але ці значення є в інших пропозиціях тієї самої моделі, система може використати найтиповіше або найчастіше значення для заповнення картки. Такий підхід підвищує повноту відображення й робить інтерфейс більш інформативним для користувача.

Ще одним важливим елементом структури даних є подання магазинних пропозицій усередині картки. Замість того щоб відображати одну “кращу” ціну від магазину, система зберігає кілька окремих оферів, якщо вони різняться за ціною або станом товару. Завдяки цьому можна коректно представляти ситуації, коли в одного магазину є кілька варіантів пропозиції на ту саму модель, у тому числі нові та вживані товари. Для уникнення дублювання використано окрему процедуру об’єднання оферів із перевіркою URL, магазину та ціни.

Під час проєктування логіки обробки було також враховано необхідність локальної фільтрації результатів у вебінтерфейсі. Картка товару повинна підтримувати зміну видимих магазинних пропозицій залежно від обраного магазину, діапазону цін і ознаки вживаного стану. Через це в моделі картки зберігаються не лише агреговані значення, а й повний список вкладених магазинних оферів. Така структура поєднує серверну логіку формування карток із клієнтською логікою зміни відображення результатів без повторного повного пошуку.

Крім структур зберігання результатів пошуку, у системі передбачено й збереження продуктів та оферів у базі даних. Для цього використовуються сутності продукту та пропозиції, де продукт зберігає канонічну назву, бренд, модель і категорію, а окремі офери містять посилання на джерело, назву товару в магазині, ціну, доступність і час збирання. Така модель встановлює логічний зв’язок між узагальненим продуктом і конкретними зібраними пропозиціями, а також створює основу для подальшого накопичення статистики змін цін [11, 12, 18].

Таким чином, структура даних у розробленій системі побудована за принципом послідовного переходу від сирої товарної пропозиції до нормалізованих атрибутів, далі – до сутності продукту і, нарешті, до зведеної картки товару. Такий підхід робить обробку даних гнучкою, спрощує групування, підтримує фільтрацію та залишає систему готовою до подальшого розширення.

Основні структури даних, що використовуються в системі, наведено в таблиці 2.2.

Таблиця 2.2

Основні структури даних системи інтелектуального агента моніторингу цін

Сутність	Основні поля	Призначення
Магазинна пропозиція	назва товару, ціна, URL, магазин, доступність, зображення, ознака вживаного стану	Представляє окремий товарний офер із конкретного магазину
Нормалізовані атрибути товару	бренд, сімейство, модель, пам’ять, оперативна пам’ять, колір, покоління, варіанти моделі, ознаки аксесуарів	Використовуються для зіставлення і групування результатів
Сутність продукту	канонічна назва, бренд, модель, категорія, ключові атрибути запиту	Є узагальненим представленням товару, який шукає користувач
Картка товару	назва моделі, пам’ять, колір, діапазон цін, список магазинних пропозицій, кількість оферів	Є підсумковою одиницею відображення результатів у вебінтерфейсі
Магазинний офер усередині картки	магазин, ціна, URL, доступність, стан товару	Відображає конкретну пропозицію в межах вже згрупованої картки
Запис продукту в БД	канонічна назва, бренд, модель, категорія	Використовується для збереження узагальненої інформації про продукт
Запис оферу в БД	посилання на продукт, джерело, назва оферу, ціна, доступність, час збирання	Зберігає конкретні зібрані пропозиції магазинів

2.4 Механізми збирання, нормалізації та зіставлення товарних даних

Розроблення інтелектуального агента потребувало побудови повного ланцюга обробки даних, який починається зі збирання товарних пропозицій із зовнішніх вебджерел і завершується формуванням узгоджених карток товарів для відображення в інтерфейсі. У межах цієї роботи такий ланцюг включає три основні етапи: збирання даних, нормалізацію атрибутів і зіставлення товарних пропозицій між собою.

На першому етапі реалізується збирання даних із кількох онлайн-магазинів. Для цього у системі використано окремі модулі-колектори, кожен із яких відповідає за конкретне джерело. У поточній реалізації підтримуються магазини Allo, MOYO, Stylus і Kibernetiki, а загальна система колекторів організована так, щоб активні джерела реєструвалися в окремому переліку та могли запускатися в єдиному циклі пошуку. Така модульність спрощує централізовану обробку результатів з різних джерел і за потреби підключати нові магазини без зміни основної логіки системи [10, 14, 15].

Кожен колектор виконує пошук за введеним користувачем запитом, аналізує HTML-сторінки відповідного магазину та намагається виділити назву товару, ціну, посилання, зображення й доступність. Для різних магазинів були реалізовані різні стратегії парсингу, що зумовлено відмінностями у структурі вебсторінок. Наприклад, для окремих джерел доцільно використовувати CSS-селектори для карток товарів, а для інших – комбінувати аналіз посилань, текстових вузлів, зображень і JSON-LD фрагментів. Такий підхід дозволив адаптувати систему до різних форматів пошукової видачі [14, 15].

Оскільки первинні дані з магазинів є неоднорідними, після збирання система переходить до етапу нормалізації текстової інформації. На цьому етапі назви товарів переводяться до стандартизованого вигляду: нормалізується регістр символів, уніфікуються одиниці вимірювання, прибираються маркетингові слова, службові позначення та частина шумових токенів. Додатково використовуються словники брендів, сімейств моделей, кольорів, сильних варіантів моделей і маркерів аксесуарів, що створює основу для подальшого виділення атрибутів і коректного порівняння назв товарів [26–31].

Одним із центральних етапів є виділення атрибутів товару. Для кожної назви система намагається визначити бренд, сімейство моделі, номер моделі, покоління, обсяг пам’яті, оперативну пам’ять, тип накопичувача, процесор, графічний адаптер, розмір екрана, колір, стан товару, а також можливі модифікації та коди моделі. Такий механізм ґрунтується на наборі правил і

шаблонів, орієнтованих на найпоширеніші категорії техніки: смартфони, ноутбуки, планшети, телевізори, консолі та аксесуари. Виділені атрибути використовуються як для оцінки релевантності результату щодо запиту користувача, так і для побудови підсумкових карток [26–36].

Окреме значення має визначення аксесуарів і нерелевантних результатів. Для цього в системі використовується набір маркерів, що вказують на чохли, адаптери, кабелі, стилуси, сумки, клавіатури, захисне скло та інші супутні товари. Якщо система розпізнає в назві чи характеристиках ознаки аксесуара, такий результат або відкидається, або отримує істотно нижчу оцінку релевантності. Це зменшує ризик потрапляння допоміжних позицій у картки основних товарів, пов’язані лише згадкою сумісної моделі [27, 29, 30].

Після нормалізації та виділення атрибутів система переходить до зіставлення знайдених пропозицій із сутністю продукту, сформованою на основі запиту користувача. Для цього обчислюється оцінка відповідності між характеристиками запиту та атрибутами конкретної товарної пропозиції. У розрахунку враховуються збіг бренду, сімейства, моделі, варіанта виконання, кольору, обсягу пам’яті, покоління та інших параметрів. Якщо пропозиція порушує сильні умови релевантності, наприклад належить до іншої моделі, іншого сімейства або іншої категорії, вона виключається з підсумкової видачі. Якщо ж збіг є достатнім, пропозиція залишається у списку для відображення.

Після відбору релевантних пропозицій виконується ґрупування однакових моделей. Групування ґрунтується на ключових нормалізованих атрибутах, зокрема бренді, сімействі, моделі, пам’яті, оперативній пам’яті та кольорі. На практиці завдяки цьому однакові моделі смартфона з кількох магазинів об’єднуються в одну картку, усередині якої вже відображається перелік доступних магазинних оферів. Додатково система виконує усунення дубльованих пропозицій за URL, магазином і ціною, щоб користувач не бачив технічних повторів.

Важливим складником механізму зіставлення є обробка стану товару. У системі реалізовано виявлення маркерів, що відповідають вживаному, відновленому або уціненому товару. Такі пропозиції позначаються окремо, а в інтерфейсі користувач може обирати, чи включати їх у результати пошуку. Це зберігає аналітичну коректність порівняння й одночасно залишити можливість працювати з ширшим спектром пропозицій [22, 23, 31].

На завершальному етапі обробки система формує картки товарів для інтерфейсу. Для кожної картки обчислюється мінімальна і максимальна ціна, кількість магазинів, доступність, а також заповнюються найважливіші атрибути: назва, колір, пам’ять, оперативна пам’ять і зображення. Якщо частина атрибутів відсутня в одній пропозиції, вони можуть бути уточнені за рахунок інших пропозицій цієї самої моделі. Такий механізм робить кінцевий результат більш стабільним і зрозумілим для користувача.

Для відображення логіки внутрішньої обробки товарних пропозицій подано послідовність основних етапів перетворення вхідних даних.

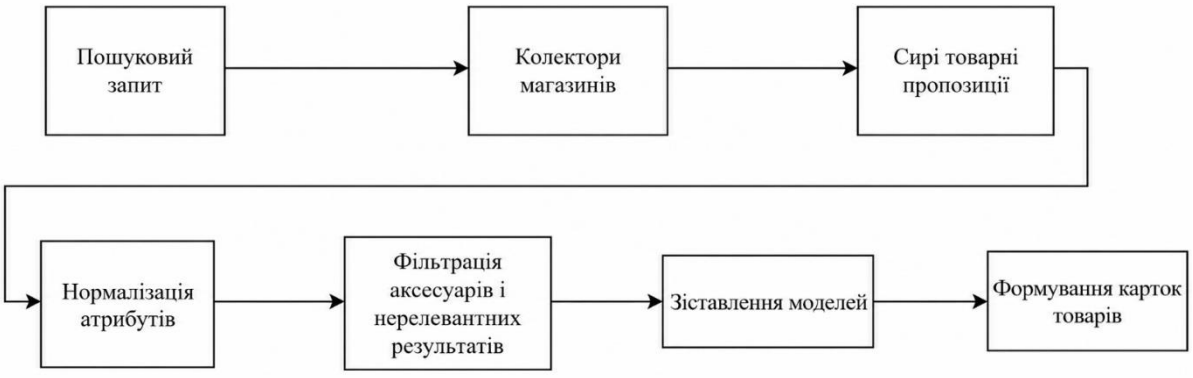

Рисунок 2.2. Послідовність збирання, нормалізації та зіставлення товарних даних в інтелектуальній системі

Як показано на рисунку 2.2, після отримання пошукового запиту система проходить етапи збирання даних, нормалізації атрибутів, фільтрації нерелевантних результатів, зіставлення моделей і побудови карток товарів.

Отже, механізми збирання, нормалізації та зіставлення товарних даних у розробленій системі утворюють єдиний інтелектуальний контур обробки інформації. Він поєднує модулі збору даних з різних онлайн-магазинів,

алгоритми виділення атрибутів, фільтрацію нерелевантних позицій, оцінювання релевантності та групування однакових моделей. Саме завдяки цьому користувач отримує не просто список знайдених сторінок магазинів, а структурований і практично корисний результат пошуку.

2.5 Проєктування користувацького інтерфейсу вебсистеми

Одним із важливих етапів розроблення інтелектуального агента стало проєктування користувацького інтерфейсу вебсистеми. Оскільки програмний продукт орієнтований на швидкий пошук, порівняння та аналіз товарних пропозицій, інтерфейс повинен був поєднати простоту взаємодії, наочність результатів та можливість гнучкого відбору знайдених даних. Саме тому під час проєктування було обрано підхід, за якого інтерфейс складається з двох основних сценаріїв використання: початкового введення запиту на головній сторінці та подальшої аналітичної роботи з результатами на окремій сторінці пошуку [19–21].

Базою для всіх сторінок системи є спільний шаблон, який визначає верхню панель із брендом сервісу, загальну структуру сторінки та підключення таблиці стилів. Такий підхід зберігає візуальну цілісність усієї системи та спрощує підтримку інтерфейсу, оскільки загальні елементи не дублюються в кожному шаблоні окремо. У макеті верхньої частини сторінки передбачено назву сервісу та короткий опис його призначення, що формує для користувача зрозумілий контекст ще до взаємодії з пошуковими функціями.

Головна сторінка системи побудована за принципом геройського стартового екрана, де в лівій частині розміщено короткий опис призначення сервісу, а в правій – пошукову панель. Така композиція одразу акцентує увагу користувача на основній функції системи – введенні запиту для пошуку товару. У складі пошукового блоку передбачено текстове поле для назви товару, кнопку запуску пошуку та список категорій, що дає змогу одразу звузити область пошуку. Окрім цього, на головній сторінці реалізовано набір швидких кнопок категорій, натискання на які автоматично підставляє

відповідну категорію в поле вибору. Це підвищує зручність роботи та скорочує кількість дій з боку користувача [19, 20]. На головній сторінці користувач отримує доступ до основного сценарію роботи системи – введення пошукового запиту та вибору категорії товару.

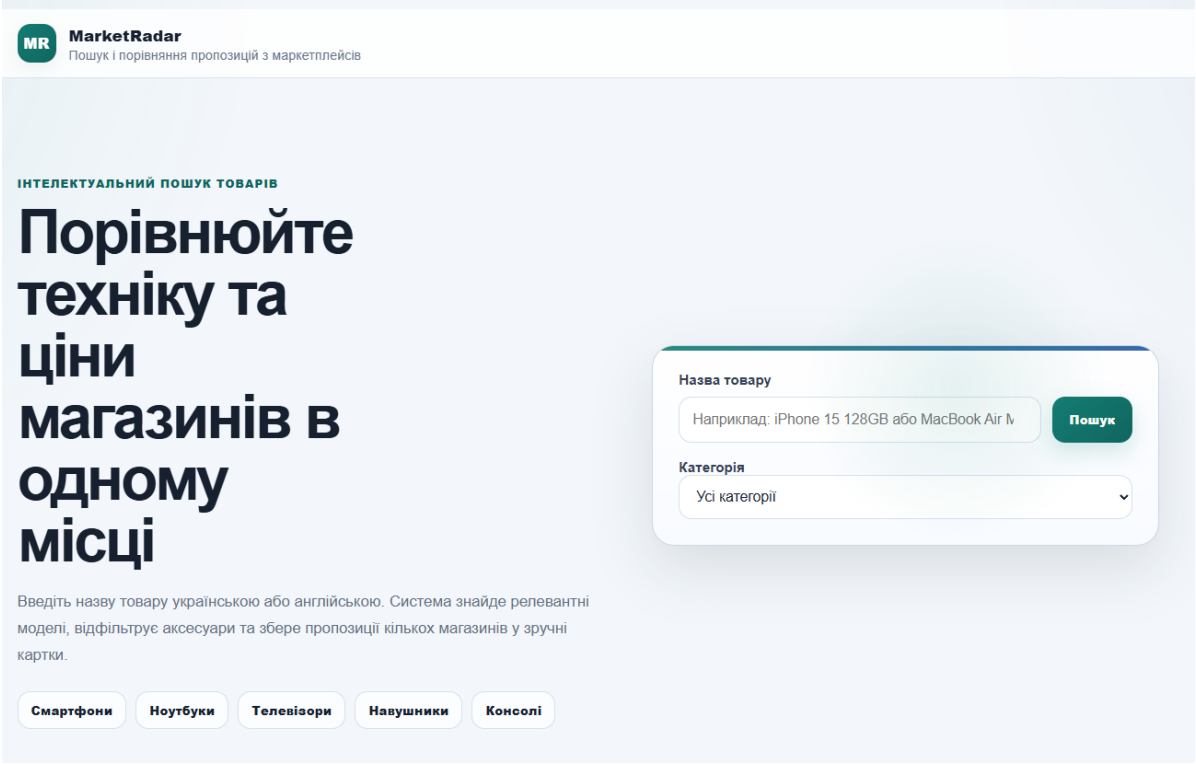

Рисунок 2.3. Головна сторінка вебсистеми

На рисунку 2.3 наведено зовнішній вигляд головної сторінки вебсистеми, яка містить заголовний блок, форму пошуку та елементи швидкого доступу до основних категорій.

Сторінка результатів побудована за багатоблочною схемою. У верхній частині повторно розміщується компактна пошукова форма, щоб користувач міг швидко змінити запит без повернення на головну сторінку. Нижче розташовано підсумковий інформаційний блок, який містить канонічну назву знайденого продукту, бренд, модель, категорію та кількість карток, що формують загальне уявлення про результат пошуку. Такий блок виконує роль короткого резюме і дає користувачеві розуміння, як саме система інтерпретувала його запит [19–21].

Основна частина сторінки результатів має двоколонкову структуру, що відокремлює інструменти уточнення видачі від самих знайдених даних. Панель фільтрів містить вибір магазину, поля для мінімальної та максимальної ціни, а також перемикач відображення вживаних товарів. Завдяки цьому користувач може одразу налаштувати видачу відповідно до власних потреб, не втрачаючи з поля зору самі результати.

У правій частині сторінки реалізовано панель керування результатами, яка включає заголовок, короткий пояснювальний текст і елемент сортування за ціною. Після цього відображається список карток товарів. Кожна картка є окремою візуально відокремленою одиницею інтерфейсу і містить зображення товару, назву моделі, короткі атрибутивні позначки, ціновий діапазон та кількість доступних пропозицій. У реалізації картки використовуються окремі поля для пам’яті, кольору, оперативної пам’яті та інших характеристик, що дозволяє користувачеві швидко оцінити різницю між моделями без відкриття додаткових сторінок.

Важливою частиною інтерфейсу є відображення магазинних оферів усередині картки. Для кожної моделі система показує список магазинів, у яких знайдено відповідну пропозицію, із зазначенням назви магазину, стану товару, доступності та ціни. Для переходу до конкретної пропозиції передбачено окрему кнопку переходу до магазину. Якщо кількість оферів у картці є великою, частина з них може бути прихована до натискання кнопки розгортання. Це рішення дає змогу зменшити візуальне перевантаження сторінки і водночас зберегти доступ до повного переліку магазинів. Після виконання пошукового запиту система формує сторінку результатів, у межах якої користувач може аналізувати знайдені моделі та керувати видачею.

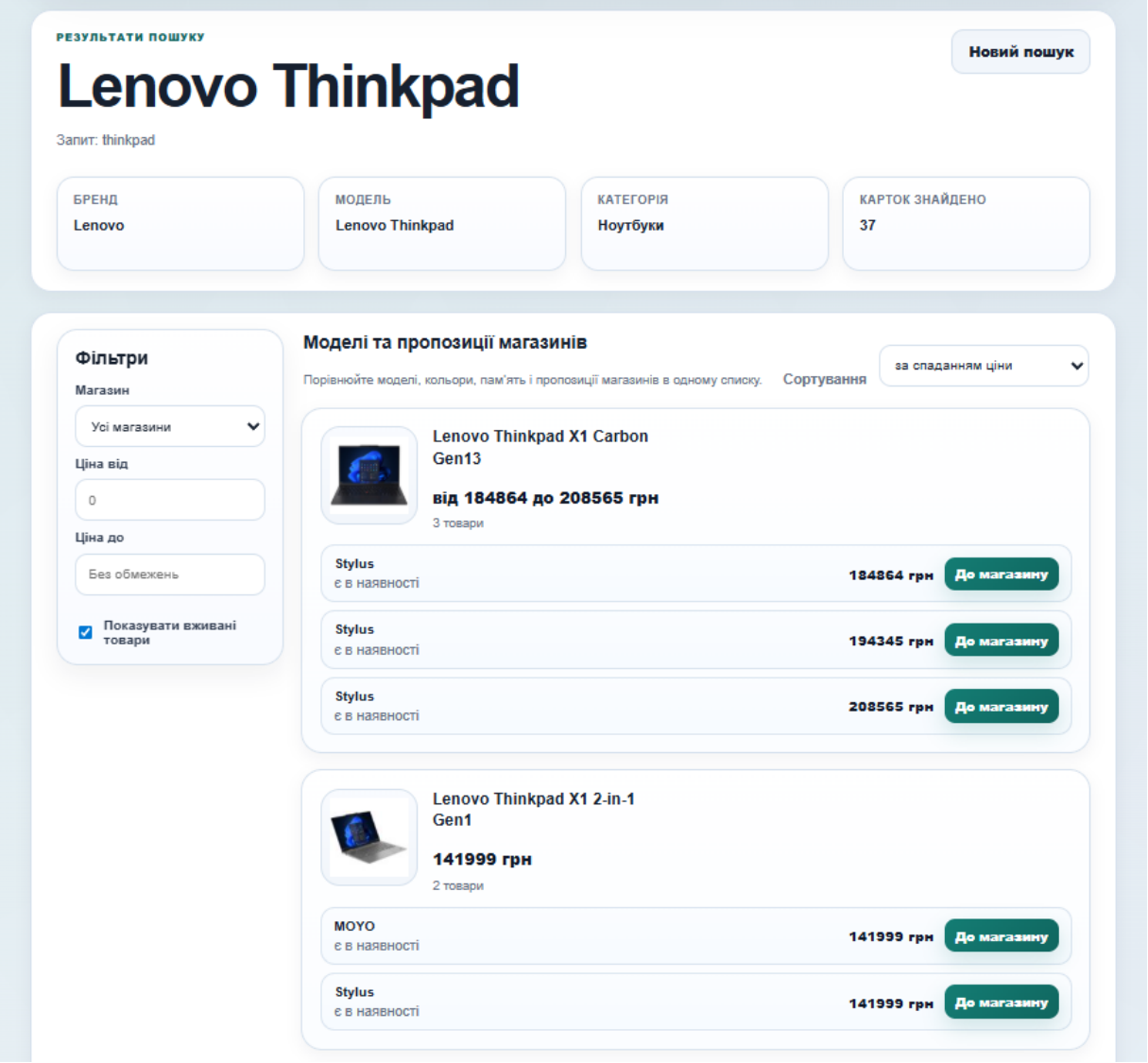

Рисунок 2.4. Сторінка результатів пошуку товарів

Рисунок 2.4 ілюструє сторінку результатів пошуку, де відображаються інформаційний блок про знайдений продукт, панель фільтрів і картки товарів із пропозиціями магазинів.

Для візуальної організації інтерфейсу використовується окрема таблиця стилів, у якій реалізовано єдину кольорову палітру, систему відступів, рамок, тіней, радіусів заокруглення та типографічних параметрів. У стилях передбачено окремі змінні для основного фону, поверхонь, акцентного кольору, кольорів попереджень і статусів. Це підтримує узгоджений візуальний стиль між головною сторінкою, сторінкою результатів, картками товарів, кнопками, бейджами і панелями фільтрів. Такий підхід відповідає

принципам сучасного вебпроєктування та підвищує якість сприйняття системи користувачем.

Окремо слід відзначити адаптивність інтерфейсу. У таблиці стилів передбачено правила для вузьких екранів, за яких двоколонкова структура перебудовується в одноколонкову, панель фільтрів втрачає режим фіксованого позиціювання, а блоки результатів автоматично підлаштовуються під меншу ширину. Це важливо, оскільки система має залишатися зручною не лише на настільних пристроях, а й на ноутбуках або пристроях із меншим розміром екрана [19, 20].

Користувацький інтерфейс доповнюється клієнтською логікою взаємодії, реалізованою мовою JavaScript. На клієнтському рівні система виконує локальну фільтрацію оферів усередині карток, оновлює відображуваний ціновий діапазон, кількість видимих товарів, сортує картки за ціною, а також керує логікою розгортання й згортання списку магазинних пропозицій. Завдяки цьому користувач змінює параметри перегляду без повторного повного запиту до сервера, що підвищує швидкодію інтерфейсу та покращує загальний досвід використання системи [21].

Під час проєктування інтерфейсу особлива увага приділялася зрозумілості та практичності. У системі використовується мінімалістичний візуальний стиль без перевантаження другорядними елементами. Основний акцент зроблено на пошуковому полі, структурованих картках товарів, діапазоні цін і можливості швидко перейти до магазину. Для системи порівняння товарних пропозицій така логіка є доцільною, оскільки користувачеві важливо швидко отримати відповідь на запит і порівняти кілька варіантів без зайвого навігаційного шуму.

Таким чином, користувацький інтерфейс розробленої вебсистеми спроєктовано як сукупність взаємопов’язаних блоків, кожен із яких відповідає окремому етапу роботи з даними: введенню запиту, перегляду узагальненої інформації, застосуванню фільтрів, аналізу карток товарів і переходу до

конкретних магазинних пропозицій. Таке проєктне рішення забезпечує поєднання функціональності, зручності та зрозумілості інтерфейсу.

Висновки до розділу 2

У другому розділі було визначено функціональні та нефункціональні вимоги до інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах, а також описано архітектуру програмної системи, структуру її даних, логіку обробки товарних пропозицій і принципи побудови користувацького інтерфейсу.

У результаті проєктування встановлено, що ефективна реалізація системи потребує поєднання кількох взаємопов’язаних компонентів: модулів збору даних з онлайн-магазинів, механізмів нормалізації текстової інформації, алгоритмів зіставлення товарних атрибутів, сервісу побудови карток товарів, клієнтського інтерфейсу фільтрації та внутрішнього рівня збереження даних. У результаті товарні пропозиції з різних джерел перетворюються на структурований і практично корисний результат.

Окрему увагу в розділі приділено проєктуванню структури даних, у межах якої розмежовано окрему магазинну пропозицію, нормалізований набір атрибутів товару, узагальнену сутність продукту та підсумкову картку для відображення. Це створило основу для коректного групування однакових моделей, фільтрації аксесуарів, обробки вживаних товарів і підтримки клієнтської взаємодії через фільтри та сортування.

Спроєктований користувацький інтерфейс вибудовує логічну послідовність роботи із системою: від введення запиту на головній сторінці до перегляду, уточнення та аналізу результатів на сторінці пошуку.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ПРОГРАМНОЇ СИСТЕМИ

3.1 Вибір інструментальних засобів і технологій реалізації

Реалізація інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах вимагала вибору таких інструментальних засобів, які б забезпечили поєднання вебінтерфейсу, серверної логіки обробки запитів, інтеграції з зовнішніми джерелами даних та внутрішнього збереження результатів. З урахуванням поставленої мети для розроблення системи було обрано стек технологій, орієнтований на створення серверного вебзастосунку з модульною структурою та можливістю подальшого розширення.

Основною мовою програмування для реалізації системи обрано Python. Такий вибір зумовлений кількома чинниками. По-перше, Python має велику кількість бібліотек для веброзробки, обробки текстових даних і роботи з HTML-сторінками. По-друге, ця мова зручна для побудови прототипів і практичних інформаційних систем, у яких важливими є швидкість розроблення, читабельність коду і гнучкість логіки обробки даних. По-третє, Python добре підходить для реалізації правил нормалізації назв товарів, зіставлення атрибутів і фільтрації результатів [8, 9].

Для серверної частини застосунку використано FastAPI. Це підтверджується структурою головного модуля, у якому створюється об’єкт застосунку FastAPI, налаштовується підключення статичних файлів і визначаються маршрути головної сторінки та сторінки результатів пошуку. Вибір FastAPI є доцільним, оскільки цей фреймворк дозволяє будувати легкі та зрозумілі вебзастосунки, підтримує зручну маршрутизацію, роботу з формами і залежностями, а також добре поєднується з шаблонізатором для серверного рендерингу HTML [10].

Для формування HTML-сторінок використано Jinja2Templates. У проєкті шаблони головної сторінки та сторінки результатів передаються до

браузера через TemplateResponse, а дані для відображення, зокрема категорії, пошуковий запит, сутність продукту, картки товарів і перелік пропозицій, формуються на сервері. Для системи порівняння цін це рішення є зручним, оскільки кінцеве подання результатів формується централізовано і зменшує складність клієнтської логіки [13].

Для роботи зі статичними ресурсами використано стандартний механізм StaticFiles, через який підключаються таблиці стилів і JavaScript-файли. Завдяки цьому інтерфейс системи поділяється на шаблони HTML, файл стилів та окрему клієнтську логіку взаємодії на JavaScript. Це робить організацію фронтенд-частини чистішою та полегшує підтримку інтерфейсу.

Для рівня збереження даних застосовано SQLAlchemy ORM, що видно з використання Session та операцій із моделями продуктів і пропозицій у сервісі обробки результатів пошуку. SQLAlchemy використовується для створення або оновлення продукту, збереження актуальних оферів, видалення попередніх записів перед оновленням та підтримки зв’язку між узагальненим продуктом і знайденими магазинними пропозиціями. Таке рішення відокремлює бізнес-логіку системи від деталей роботи з конкретною базою даних [11, 12].

Ключовим елементом серверної логіки є сервіс product_service, який об’єднує процес збору результатів із магазинів, побудову сутності продукту, фільтрацію релевантних пропозицій, дедуплікацію, сортування та формування карток для відображення. Саме цей сервіс виступає центральною точкою обробки пошукового запиту користувача. Після запуску пошуку він збирає пропозиції з усіх активних джерел, створює узагальнену сутність продукту, виконує відбір релевантних оферів і формує список карток товарів. Це дозволяє реалізувати наскрізний цикл обробки запиту в одному логічно зв’язаному модулі.

Для інтелектуальної обробки назв товарів і побудови узагальненого уявлення про модель використано окремий модуль matcher. У ньому реалізовано виділення брендів, сімейств моделей, варіантів виконання,

кольорів, маркерів аксесуарів, стану товару, а також алгоритми обчислення релевантності пропозицій щодо пошукового запиту. У цьому модулі визначено словники брендів, сімейств моделей, сильних варіантів, аксесуарів, кольорів і категорій, а також функції для побудови сутності продукту, виділення атрибутів і відсіювання нерелевантних результатів. Вибір окремого модуля для цієї логіки є виправданим, оскільки саме він містить найбільш складну частину інтелектуальної обробки даних.

На клієнтському рівні для взаємодії з уже отриманими результатами використано JavaScript. У файлі логіки сторінки результатів реалізовано фільтрацію пропозицій за магазином, мінімальною та максимальною ціною, ознакою вживаного товару, оновлення діапазону цін усередині картки, сортування карток та керування відображенням розгорнутих пропозицій магазину. Завдяки цьому частина взаємодії виконується без повторного звернення до сервера, що позитивно впливає на швидкодію і зручність користування системою [21].

Для оформлення інтерфейсу використано HTML та CSS. Шаблони головної сторінки й сторінки результатів побудовані з використанням базового макета, що задає спільну верхню панель і контейнер сторінки. У CSS-файлі реалізовано єдину палітру кольорів, стилі для hero-блоку, пошукової панелі, карток товарів, панелі фільтрів, кнопок, магазинних оферів і адаптивного відображення на вузьких екранах. Це дозволило сформувати узгоджений сучасний вигляд вебсистеми [19, 20].

Окремо слід відзначити організацію системи збирання даних через модуль collectors. У конфігурації активних джерел пошуку визначено чотири магазини: Allo, MOYO, Stylus і Kibernetiki. Така структура спрощує централізоване керування списком підтримуваних джерел і виконувати пошук одразу за всіма доступними колекторами. Модульність такого підходу спрощує розширення системи новими онлайн-магазинами в майбутньому [10, 14, 15].

Вибраний стек технологій складається з Python як основної мови програмування, FastAPI як серверного вебфреймворку, Jinja2 для серверного рендерингу шаблонів, SQLAlchemy ORM для роботи з даними, HTML/CSS для структури та стилізації інтерфейсу й JavaScript для клієнтської взаємодії. Такий набір засобів є достатнім для реалізації поставленої задачі та водночас підтримує модульність, читабельність і розширюваність програмної системи.

Інструментальні засоби та технології, використані для реалізації системи, подано в таблиці 3.1.

Таблиця 2.1

Інструментальні засоби та технології реалізації системи

Технологія	**Призначення в системі**
Python	Основна мова програмування серверної логіки
FastAPI	Реалізація серверного вебзастосунку та маршрутизації
Jinja2	Формування HTML-сторінок на сервері
SQLAlchemy	Робота з даними через ORM-рівень
SQLite	Збереження продуктів і оферів у базі даних
HTML	Побудова структури сторінок вебсистеми
CSS	Оформлення інтерфейсу користувача
JavaScript	Клієнтська фільтрація, сортування і взаємодія зі сторінкою результатів
Requests	Виконання HTTP-запитів до сторінок магазинів
Beautiful Soup	Аналіз HTML-сторінок і виділення даних
Uvicorn	Запуск ASGI-сервера під час розробки
Pydantic	Валідація та подання структурованих даних

Фрагменти серверної реалізації системи подано в додатку Б.

3.2 Реалізація модулів збирання даних з онлайн-магазинів

Одним із найважливіших практичних етапів розроблення системи стала реалізація модулів збирання даних з онлайн-магазинів. Саме ці модулі відповідають за отримання первинної інформації про товари, без якої неможливо побудувати ані механізм порівняння цін, ані систему аналізу цінових тенденцій. У межах роботи було реалізовано модульний підхід, за якого кожне джерело обробляється окремим колектором, а всі колектори підключаються через загальну систему реєстрації активних джерел.

У поточній реалізації система працює з чотирма онлайн-магазинами: Allo, MOYO, Stylus і Kibernetiki. Для кожного з них створено окремий модуль збирання даних, який реалізує логіку пошуку, завантаження сторінки, аналізу HTML-структури та формування нормалізованої товарної пропозиції. Така організація є принципово важливою, оскільки різні магазини мають різну структуру сторінок пошуку, різний спосіб подання назв і цін, а також різні маркери доступності чи стану товару. Наявність окремих колекторів дозволяє адаптуватися до цих відмінностей без ускладнення центральної логіки системи.

Підключення підтримуваних онлайн-магазинів у системі реалізовано через єдину конфігурацію активних джерел пошуку.

```
COLLECTORS = {
    "allo": collect_allo_offers,
    "moyo": collect_moyo_offers,
    "stylus": collect_stylus_offers,
    "kibernetiki": collect_kibernetiki_offers,
}
```

Лістинг 3.1. Фрагмент конфігурації активних джерел пошуку

Лістинг 3.1 демонструє, що підключення магазинів до системи виконується централізовано, що спрощує розширення застосунку новими джерелами даних.

Загальний механізм запуску колекторів реалізовано в сервісі пошуку. Після отримання запиту користувача сервіс проходить по списку активних джерел, бере відповідний колектор із таблиці реєстрації та виконує його. Якщо один із колекторів завершується помилкою, вона не блокує всю систему: пошук продовжується за іншими джерелами, а в підсумковому наборі результатів враховуються лише ті пропозиції, які вдалося успішно зібрати. Це рішення підвищує стійкість системи до часткових збоїв і є важливою перевагою модульної архітектури [10, 14, 15].

Кожен модуль збирання даних формує на виході уніфіковану структуру товарної пропозиції. Незалежно від того, з якого магазину надійшли дані, на подальший етап обробки передаються однакові поля: назва товару, ціна, URL

сторінки, назва магазину, джерело, доступність, зображення й додаткові ознаки, зокрема інформація про вживаний стан. Саме така уніфікація робить подальшу обробку всіх результатів однаковою, не прив’язуючи логіку matcher і побудови карток до конкретного магазину [11, 12].

Під час реалізації модулів збирання було важливо врахувати, що сторінки магазинів не є однорідними. Одні джерела містять відносно стабільні блоки товарних карток із чіткими селекторами назв, цін і зображень. Інші використовують більш складну або динамічну структуру, де доводиться шукати елементи не лише за класами, а й за посиланнями, текстовими вузлами, зображеннями та допоміжними маркерами. Практика інтеграції різних джерел показала, що універсального методу парсингу для всіх магазинів немає, тому модульний підхід виявився найдоцільнішим.

Реалізація колектора Allo орієнтована на витягування товарних карток із пошукової сторінки та виділення з них основних атрибутів. Водночас під час практичної інтеграції з цим джерелом було важливо додатково враховувати ознаки вживаних товарів і специфіку назв, щоб система не змішувала нові та вживані позиції в одній нерозділеній видачі. У підсумку пропозиції від Allo стали повноцінною частиною системи порівняння, у тому числі з підтримкою правильного групування в межах однієї товарної картки.

Колектор MOYO також реалізує збирання результатів пошуку, однак для нього особливо важливою виявилася коректна обробка назв моделей і фільтрація нерелевантних результатів. У процесі розроблення було враховано специфіку вмісту сторінок MOYO, що дозволило отримувати релевантні товари без включення зайвих позицій і коректно інтегрувати їх у спільний набір оферів. Для цього в системі також використовувалися допоміжні діагностичні маршрути, які давали змогу перевіряти проміжні результати пошуку, кількість сирих оферів та фінальний набір пропозицій після фільтрації.

Інтеграція джерела Stylus виявилася більш складною через особливості побудови сторінки пошуку. У процесі розроблення було встановлено, що

сторінка Stylus має структуру, у якій результати пошуку не завжди зручно зчитуються через звичайні селектори карток. Для цього джерела було використано інший підхід: орієнтація на товарні посилання та цінові вузли, а також допоміжні діагностичні маршрути для аналізу сторінки пошуку, пошуку товарних зображень і визначення правильного контейнера ціни. Це дало змогу адаптувати колектор до специфічної структури сторінки Stylus і включити цей магазин до загальної системи.

Джерело Kibernetiki також вимагало окремої логіки інтеграції. У його випадку важливо було фільтрувати не лише нерелевантні результати, а й аксесуари та розпродані товари, а також запобігати потраплянню технічних або службових цін до фінальної видачі. Практична реалізація цього колектора підтвердила, що для реальних вебджерел необхідно не просто зчитувати всі знайдені числові значення, а й перевіряти контекст, доступність товару та відповідність назви запиту користувача. Саме це забезпечило стабільну роботу Kibernetiki в межах загальної системи порівняння.

Усі колектори інтегруються в систему через єдину конфігурацію джерел, де зареєстровано активні магазини і відповідні функції збирання даних. Це дає змогу централізовано керувати набором підтримуваних магазинів і спрощує сценарій розширення: для додавання нового джерела достатньо створити новий модуль-колектор і зареєструвати його у спільному переліку джерел.

Важливим практичним інструментом під час реалізації модулів збирання стали допоміжні діагностичні маршрути. У системі передбачено окремі debug-маршрути для аналізу результатів пошуку, перевірки роботи окремих колекторів і перегляду проміжних етапів обробки. Такі маршрути використовувалися для відлагодження структури сторінок магазинів, перевірки релевантності сирих оферів і коректності інтеграції результатів у підсумкову систему. Їхня наявність значно спростила процес розроблення й адаптації колекторів до різних магазинів.

Ще однією важливою рисою реалізації є те, що колектори працюють не ізольовано, а в поєднанні з наступним етапом нормалізації. Зібрані ними назви

товарів не вважаються кінцевим результатом. Навпаки, вони є лише сирим матеріалом для подальшої обробки у matcher і product service. Саме тому під час реалізації модулів збирання було важливо забезпечити не ідеальну “людську” назву товару, а достатньо якісне первинне подання, з якого можна reliably виділити бренд, модель, колір, пам’ять та інші характеристики.

Модулі збирання даних з онлайн-магазинів реалізовано за модульним принципом: кожне джерело має власний колектор і власну логіку парсингу. Модульна організація підвищує гнучкість, стійкість і розширюваність системи. У результаті сформовано працездатний механізм отримання товарних пропозицій із чотирьох магазинів, який став основою для подальшої нормалізації, зіставлення та візуального подання даних.

Фрагменти реалізації модулів збирання даних з онлайн-магазинів наведено в додатку В.

3.3 Реалізація механізмів фільтрації, групування та відображення результатів

Після реалізації модулів збирання даних із зовнішніх джерел наступним важливим етапом стало впровадження механізмів, які перетворюють множину сирих магазинних пропозицій у структурований, зручний для користувача результат. У розробленій системі цей етап охоплює три взаємопов’язані процеси: фільтрацію релевантних оферів, групування однакових моделей та подальше відображення результатів у вебінтерфейсі. Центральну роль у цьому процесі відіграє сервіс обробки продуктів, який після збору сирих оферів формує сутність продукту, виконує відбір релевантних пропозицій, дедуплікацію та побудову карток для інтерфейсу.

Першим етапом є фільтрація релевантних результатів. Після запуску пошуку система збирає пропозиції з усіх активних джерел і формує первинну сутність продукту на основі пошукового запиту та назв знайдених товарів. Далі виконується фільтрація через механізм filter_display_offers, який порівнює атрибути знайдених пропозицій із побудованою сутністю продукту.

Після первинної фільтрації формується уточнена сутність, а потім виконується повторний відбір результатів, що підвищує точність зіставлення. Після цього пропозиції проходять дедуплікацію за URL, очищуються від порожніх посилань і сортуються за ціною. Саме така багатокрокова схема дозволяє зменшити кількість шуму у видачі та залишити лише пропозиції, що відповідають змісту запиту користувача.

Наступним етапом є групування пропозицій у картки товарів. У системі реалізовано функцію побудови карток, яка для кожного офера спочатку виділяє атрибути з назви товару, а потім визначає ключ групування. У поточній реалізації групування спирається на заголовок моделі та ключові атрибути, зокрема пам’ять, оперативну пам’ять і колір. Якщо кілька магазинних пропозицій мають однаковий ключ, вони потрапляють до однієї картки. Після цього для картки формується зведений набір магазинних оферів, обчислюється мінімальна та максимальна ціна, встановлюється доступність, а також визначається ознака наявності вживаних товарів у межах групи. Такий підхід дозволяє показати користувачеві не розрізнені результати пошуку, а структуровані моделі з кількома джерелами пропозицій усередині [11, 12].

Групування однакових товарних пропозицій у межах однієї картки реалізовано на серверному рівні шляхом формування ключа на основі основних атрибутів товару.

```
for offer in offers:
        attrs = extract_attributes(offer.source_product_name)
        canonical_brand = _canonical_card_brand(attrs)
        canonical_family = attrs.family or ""
        canonical_model = attrs.model or ""

        key = (canonical_brand, canonical_family, canonical_model,
attrs.storage,
           attrs.ram, attrs.color,)
        grouped.setdefault(key, []).append(offer)
```

Лістинг 3.2. Фрагмент реалізації групування товарних пропозицій у картки

Як видно з лістингу 3.2, система виділяє атрибути товару, формує ключ групування та об’єднує релевантні магазинні пропозиції в єдину картку моделі.

Однією з ключових особливостей розробленої системи є представлення кількох магазинних пропозицій у межах однієї картки товару.

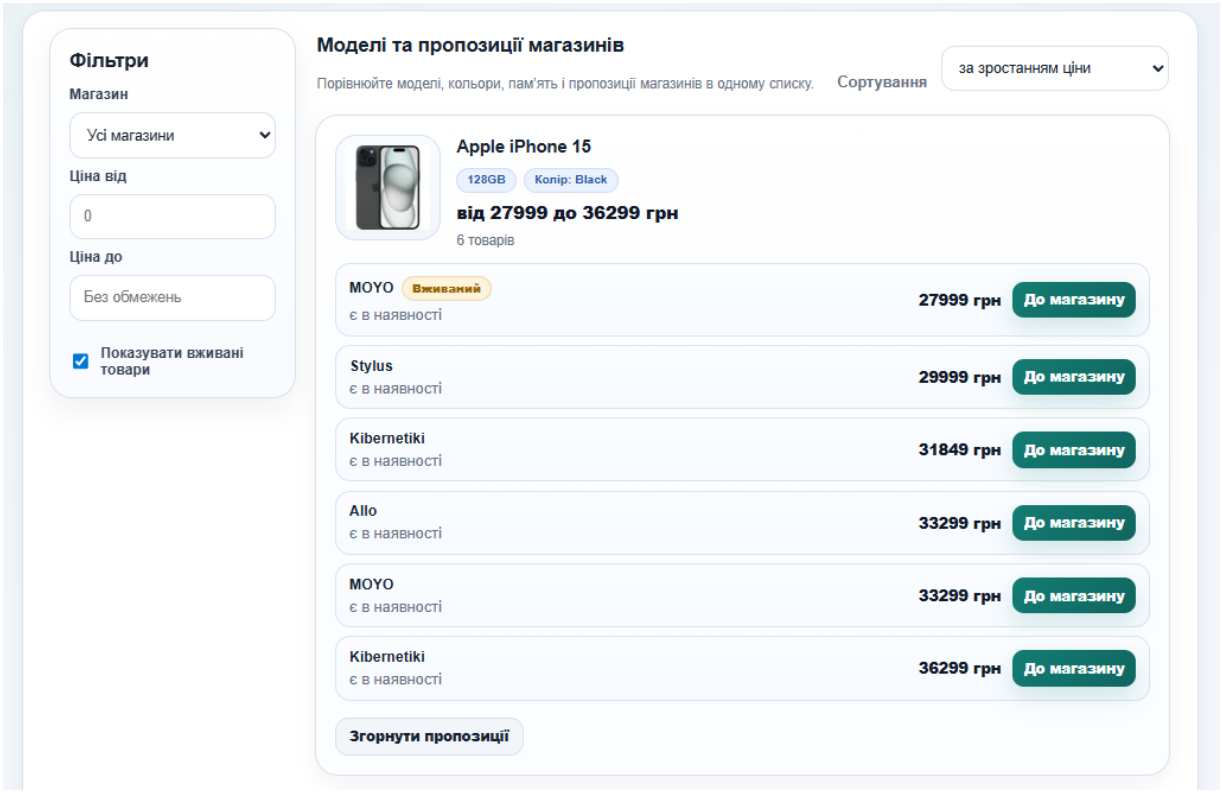

Рисунок 3.1. Відображення магазинних пропозицій у межах картки товару

Рисунок 3.1 демонструє приклад картки товару, всередині якої згруповано пропозиції різних магазинів із зазначенням ціни, доступності та стану товару.

Важливою частиною реалізації стало усунення дублювань магазинних оферів. Для цього в сервісі використовується окремий механізм об’єднання пропозицій магазину, який перевіряє джерело, URL та ціну. Якщо збіг за цими ознаками повторюється, дубльований елемент не додається повторно. Крім того, з підсумкового набору усуваються офери з некоректною ціною або відсутнім посиланням. Завдяки цьому в кожній картці користувач бачить лише

осмислений набір пропозицій, а не повторювані або технічно помилкові позиції.

Окремим аспектом реалізації стала обробка вживаних товарів. У системі вживаний стан визначається на основі маркерів у назві товару, після чого ця ознака передається як на рівень магазинного офера, так і на рівень картки товару. Завдяки цьому інтерфейс може позначати такі пропозиції окремим бейджем, а користувач отримує можливість вмикати або вимикати їх відображення. Така реалізація була важливою, оскільки без розмежування нових і вживаних пропозицій порівняння цін втрачало б аналітичну коректність.

Для відображення результатів пошуку використано шаблон сторінки результатів, у якому кожна картка містить назву моделі, зображення, атрибутивні позначки, діапазон цін, кількість пропозицій і перелік магазинних оферів. У верхній частині сторінки розміщується повторна форма пошуку, а також блок стислої інформації про знайдений продукт, де відображаються бренд, модель, категорія та кількість карток. Нижче розміщується двоколонкова структура: зліва панель фільтрів, справа список карток товарів. Таке рішення дає користувачеві одночасний доступ до інструментів уточнення видачі та до самих результатів.

Значна частина механізмів фільтрації та взаємодії з результатами реалізована на клієнтському рівні за допомогою JavaScript. Після завантаження сторінки скрипт отримує доступ до елементів інтерфейсу, зокрема селектора магазину, полів мінімальної та максимальної ціни, перемикача вживаних товарів і елемента сортування. На основі цих параметрів функція applyControls проходить по всіх картках, викликає процедуру оновлення видимих оферів усередині картки, визначає, чи має картка залишатися видимою, оновлює порожній стан сторінки і запускає повторне сортування. Такий підхід дозволяє змінювати подання результатів без повторного звернення до сервера [21].

Усередині кожної картки реалізовано локальну фільтрацію магазинних оферів. Функція updateCardOffers перевіряє офери за магазином, ціною та ознакою вживаного товару, приховує невідповідні елементи, оновлює ціновий діапазон картки та перераховує кількість видимих пропозицій. Якщо кількість оферів у картці перевищує певне значення, частина з них приховується за кнопкою розгортання, а при натисканні відкривається повний список. Це рішення дає змогу зберігати компактність інтерфейсу і водночас не втрачати доступ до всіх доступних пропозицій.

Окремо реалізовано сортування карток за ціною. Для цього використовується функція sortVisibleCards, яка сортує елементи за поточною активною ціною – або за базовою ціною картки, або за уточненою ціною після застосування фільтрів усередині картки. Завдяки цьому логічний порядок результатів зберігається навіть після зміни магазину, діапазону цін або стану товару. Окрім того, в клієнтській логіці реалізовано оновлення кількості видимих карток, що відображає актуальний результат після застосування фільтрів.

Клієнтська логіка сторінки результатів оновлює видимі картки відповідно до встановлених користувачем параметрів фільтрації.

```
function applyControls() {
    if (!list) return;
    const cards = Array.from(list.querySelectorAll(".product-card"));
    const min = Number(minPrice?.value || 0), max = Number(maxPrice?.value ||
0);
    const store = storeFilter?.value || "all", showUsed =
Boolean(includeUsed?.checked);
    cards.forEach((card) => {
        const state = updateCardOffers(card, store, min, max, showUsed);
        card.classList.toggle("hidden", state.visibleCount === 0);
    });
    sortVisibleCards(); updateVisibleCardsCount();
}
```

Лістинг 3.3. Фрагмент клієнтської логіки фільтрації результатів

Лістинг 3.3 показує, що після зміни параметрів фільтрації система повторно обчислює видимість карток, а також оновлює їх порядок і кількість без повторного серверного пошуку.

У підсумку в системі реалізовано повний цикл перетворення результатів пошуку: від відбору релевантних магазинних пропозицій до побудови зведених карток і клієнтської взаємодії з ними. Поєднання серверної логіки фільтрації та групування з клієнтськими засобами сортування, локального відбору оферів і розгортання пропозицій забезпечило зручне, гнучке й практично корисне подання результатів пошуку для користувача.

3.4 Тестування роботи системи на різних типах пошукових запитів

Тестові сценарії перевірки роботи системи наведено в таблиці 3.2.

Таблиця 3.2

Тестові сценарії перевірки роботи системи

№	Пошуковий запит	Мета перевірки	Очікуваний результат
1	2	3	4
1	iPhone 15 128 blue	Перевірка точного запиту з моделлю, пам’яттю та кольором	Видача релевантних моделей без аксесуарів і з правильним групуванням
2	iPhone 16	Перевірка загального запиту без уточнених атрибутів	Відображення кількох релевантних карток сімейства без звуження до однієї моделі
3	iPhone 15 Pro	Перевірка розмежування сильної модифікації моделі	У видачі тільки Pro-версії без звичайної моделі
4	MacBook Air	Перевірка широкого запиту для ноутбуків	Показ усіх релевантних моделей MacBook Air без сторонніх товарів
5	MacBook Air M4	Перевірка уточненої моделі ноутбука	Видача звужується до відповідної модифікації
6	iPad Air	Перевірка пошуку планшета та відсіювання аксесуарів	У видачі залишаються планшети без чохлів, стилусів та інших аксесуарів
7	ThinkPad	Перевірка відсіювання нерелевантних суміжних товарів	У видачі не повинно бути рюкзаків, сумок або сторонніх позицій
8	Galaxy Tab	Перевірка пошуку іншої категорії планшетів	Коректне відображення релевантних моделей і магазинних оферів

Продовження Таблиці 3.2

1	2	3	4
9	iPhone 15 з фільтром від 30000 грн	Перевірка роботи фільтра мінімальної ціни	У видачі залишаються лише картки, що відповідають заданому порогу
10	iPhone 15 без показу вживаних товарів	Перевірка логіки приховування вживаних оферів	У видачі відсутні вживані товари, а кількість карток оновлюється коректно

Після реалізації основних модулів системи було проведено функціональне тестування, спрямоване на перевірку коректності пошуку, фільтрації, групування та відображення результатів для різних типів товарних запитів. Метою тестування було встановити, наскільки розроблена система здатна обробляти як точні, так і узагальнені запити, коректно відсіювати нерелевантні товари, правильно об’єднувати пропозиції однакових моделей та забезпечувати стабільну взаємодію користувача з результатами через фільтри і сортування. У методичці до кваліфікаційної роботи наголошено, що в проєктній роботі мають бути продемонстровані результати практичної реалізації та перевірки функціонування системи, що повністю відповідає структурі даного етапу дослідження [4].

Першу групу тестів становили **точні пошукові запити**, у яких користувач задає бренд, модель і одну або кілька характеристик товару. До цієї групи належать запити на кшталт смартфона з конкретною моделлю, обсягом пам’яті та кольором або ноутбука з уточненням серії. Під час таких тестів перевірялося, чи здатна система коректно виділити атрибути запиту, знайти відповідні пропозиції в усіх підключених магазинах, відсіяти аксесуари та об’єднати однакові моделі в одну картку. Очікуваним результатом був показ лише релевантних моделей із правильним поділом за кольором, пам’яттю та іншими сильними атрибутами. Такий сценарій безпосередньо підтримується

серверною логікою побудови сутності продукту, фільтрації оферів і формування карток товарів.

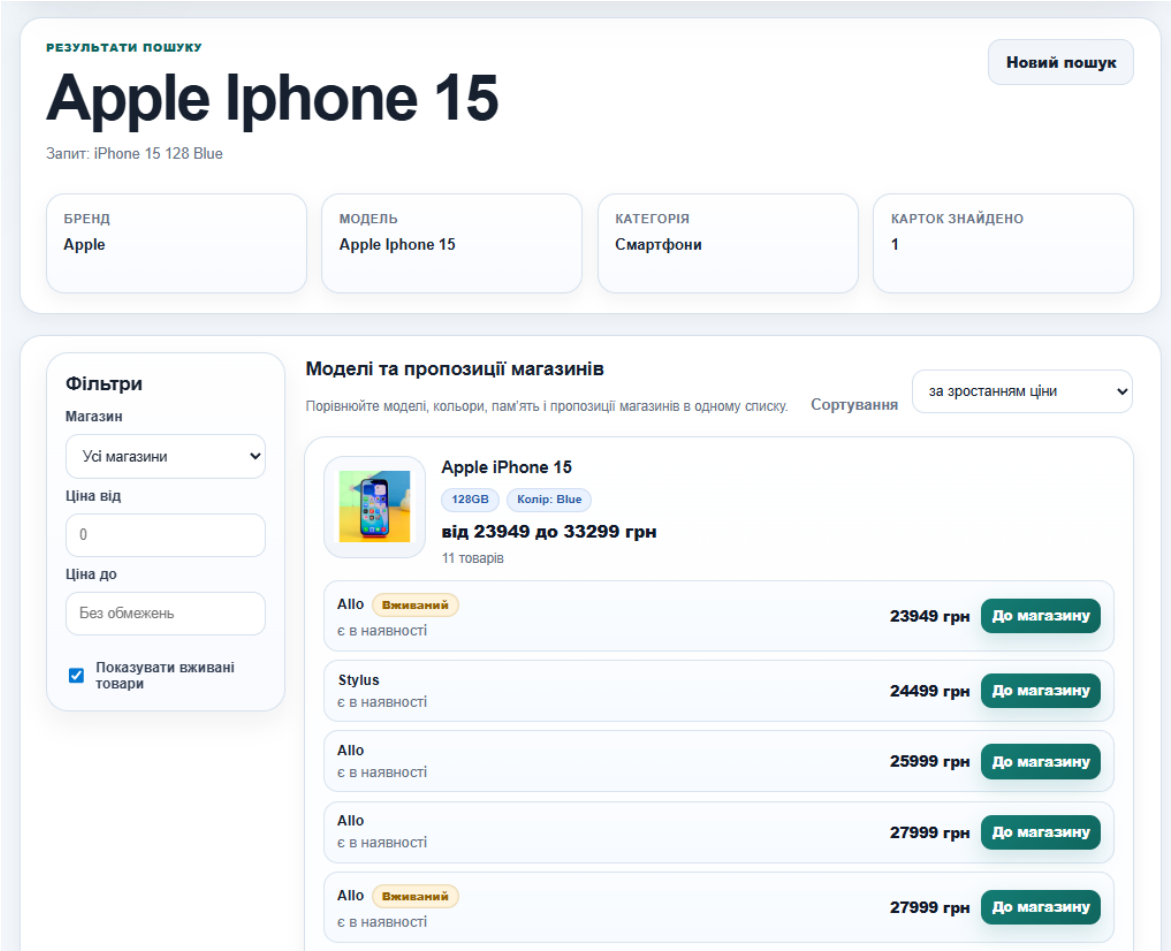

Рисунок 3.2. Результат обробки точного пошукового запиту

Рисунок 3.2 демонструє результат обробки точного пошукового запиту, за якого система коректно визначає модель, відсіює нерелевантні товари та формує релевантні картки.

Друга група тестів охоплювала узагальнені пошукові запити, коли користувач вводить лише базову назву моделі або сімейства пристроїв без детального уточнення характеристик. У цьому випадку система повинна була не звужувати видачу до однієї конкретної модифікації, а показувати кілька релевантних карток у межах одного сімейства. Під час таких тестів оцінювалася коректність роботи механізму побудови сутності продукту та логіки matcher, яка має зберігати баланс між широтою пошуку й точністю результатів. Успішним результатом вважалося таке подання, за якого система

не домислює надто вузькі характеристики, але при цьому не наповнює видачу сторонніми моделями.

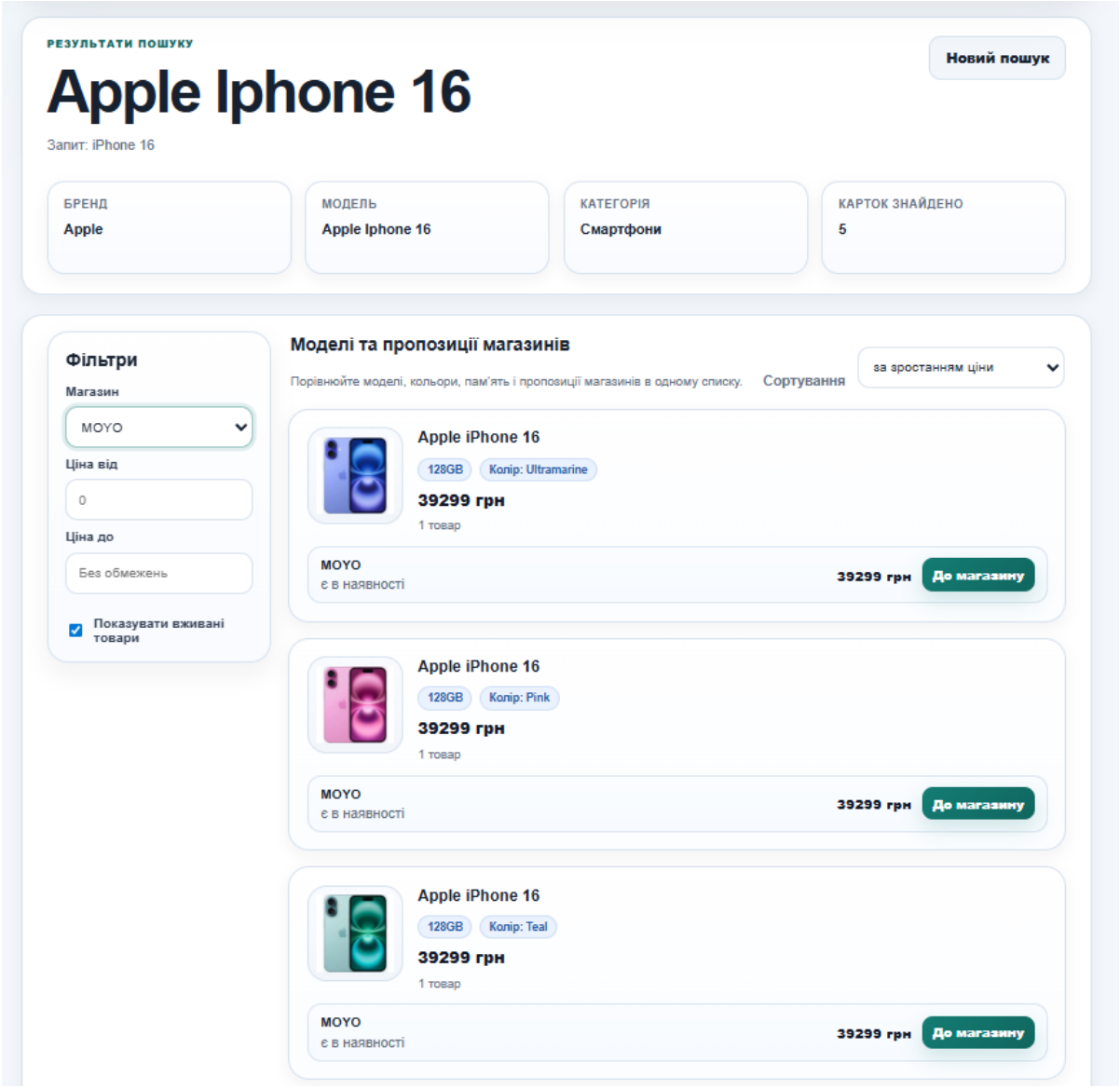

Рисунок 3.3. Результат обробки узагальненого пошукового запиту

На рисунку 3.3 показано, що при узагальненому пошуковому запиті система не звужує видачу до однієї конкретної моделі, а формує набір релевантних карток у межах відповідного сімейства товарів.

Третя група тестів була пов’язана з відокремленням основних товарів від аксесуарів. Для цього використовувалися пошукові запити за популярними моделями смартфонів, планшетів і ноутбуків, оскільки саме для них магазини часто повертають чохли, скло, адаптери, стилуси, сумки та інші супутні

товари. Перевірялося, чи працює механізм фільтрації аксесуарів у назвах магазинних пропозицій і чи не потрапляють такі результати до фінальних карток. Успішним вважався сценарій, за якого у видачі залишалися лише основні моделі техніки, а нерелевантні аксесуари виключалися ще до формування підсумкових карток. Необхідність такого виду тестування прямо випливає з реалізованої логіки виділення атрибутів і фільтрації оферів.

Четверта група тестів стосувалася розмежування нових і вживаних товарів. Для цього перевірялося, чи здатна система позначати вживані пропозиції окремо, чи не змішуються вони непомітно з новими товарами і чи правильно працює користувацький перемикач відображення вживаних оферів. У таких сценаріях важливо було не лише виявити стан товару на серверному рівні, а й переконатися, що клієнтська логіка сторінки результатів коректно приховує або показує відповідні офери й оновлює кількість видимих карток. Реалізація цього механізму перевіряється на стику сервісу побудови карток і JavaScript-фільтрації у сторінці результатів.

П’ята група тестів охоплювала фільтрацію за ціною та магазином. Для цього перевірялося, чи змінюється видимий набір карток після введення мінімальної або максимальної ціни, а також після вибору конкретного магазину. Окремо оцінювалася коректність перерахунку відображуваного діапазону цін усередині картки та оновлення кількості видимих карток на сторінці. Такі сценарії є важливими для користувацької взаємодії, оскільки саме вони дозволяють перетворити список знайдених карток на інструмент вибору оптимальної пропозиції. У реалізації це забезпечується функціями updateCardSummary, updateCardOffers, applyControls і механізмом оновлення кількості карток.

Шоста група тестів стосувалася групування однакових моделей з кількома магазинними оферами. У межах цих перевірок оцінювалося, чи система правильно об’єднує пропозиції тієї самої моделі з різних джерел в одну картку, чи не створює зайвих дубльованих карток та чи не втрачає окремі магазинні офери в процесі об’єднання. Окремо перевірялося збереження

кількох пропозицій від одного магазину, якщо вони фактично є різними за ціною або станом. Коректність такого сценарію забезпечується логікою групування карток та процедурою _merge_store_offers, яка усуває лише технічні дублікати, а не всі повтори як такі.

Сьома група тестів була пов’язана з різними категоріями товарів. Система підтримує категорії смартфонів, ноутбуків, планшетів, телевізорів, консолей, навушників, аксесуарів і побутової техніки, а також набір швидких категорій на стартовій сторінці. Під час перевірки оцінювалося, чи не виникає збоїв у роботі пошуку при переході між категоріями та чи узгоджено працює логіка інтерфейсу в межах різних типів запитів. Це має значення не лише для широти застосування системи, а й для оцінки її стійкості як багатокатегорійного сервісу порівняння техніки.

Окремим практичним інструментом тестування стали вбудовані діагностичні маршрути, реалізовані в серверному застосунку. У поточній версії системи наявні маршрути для перевірки результатів пошуку, проміжних станів побудови карток, окремих магазинних колекторів і етапів обробки даних. Їх використання дало змогу локалізувати проблеми в колекторах магазинів, у механізмах нормалізації назв, у клієнтській фільтрації результатів та в підрахунку видимих карток. Наявність таких маршрутів суттєво спростила процес функціонального тестування та пришвидшила усунення виявлених помилок.

За результатами проведених перевірок можна зробити висновок, що система загалом коректно обробляє як точні, так і широкі запити, відсіює нерелевантні результати, групує однакові моделі в спільні картки, підтримує розмежування нових і вживаних товарів, а також підтримує коректну роботу фільтрів і сортування на рівні інтерфейсу. Виявлені під час розробки й тестування неточності були поетапно усунені завдяки модульній архітектурі системи, що дозволяло локально виправляти окремі механізми без повного перепроєктування застосунку. У підсумку функціональне тестування

підтвердило працездатність розробленої вебсистеми та її придатність до подальшого розвитку.

Узагальнені результати функціонального тестування системи подано в таблиці 3.3.

Таблиця 3.3

Результати функціонального тестування системи

Сценарій перевірки	**Результат роботи системи**	**Виявлені особливості**	**Підсумок**
Точний запит із моделлю, пам’яттю та кольором	Система коректно знаходить релевантні моделі	Потрібна точна обробка атрибутів назви	Пройдено
Узагальнений запит без уточнення характеристик	Система не звужує видачу до однієї моделі	Важливий баланс між точністю і широтою пошуку	Пройдено
Пошук модифікацій Pro / Plus / Max / Air	Система розрізняє сильні варіанти моделей	Потрібен контроль сильних атрибутів	Пройдено
Відсіювання аксесуарів	Більшість нерелевантних аксесуарів виключається з видачі	Критично важливі словники маркерів аксесуарів	Пройдено
Розмежування нових і вживаних товарів	Вживані офери позначаються і можуть приховуватися	Важлива коректність маркерів стану товару	Пройдено
Фільтрація за ціною	Картки коректно оновлюються після зміни меж ціни	Потрібно оновлювати кількість видимих карток	Пройдено
Фільтрація за магазином	Видача обмежується оферами обраного джерела	Перераховується діапазон цін усередині картки	Пройдено
Сортування за ціною	Картки переставляються відповідно до активної ціни	Має враховуватися поточний стан фільтрів	Пройдено
Групування однакових моделей	Офери з різних джерел об’єднуються в одну картку	Важлива дедуплікація URL і магазинних оферів	Пройдено
Робота кількох магазинів одночасно	Система стабільно працює з 4 активними джерелами	Частковий збій одного джерела не зупиняє систему	Пройдено

3.5 Аналіз результатів функціонування системи

Після реалізації основних модулів було проведено аналіз результатів функціонування розробленої вебсистеми. Оцінювання здійснювалося за кількома критеріями: коректність збирання товарних пропозицій, якість фільтрації нерелевантних результатів, правильність групування однакових моделей, зручність роботи з інтерфейсом і можливість подальшого розширення системи [10, 14, 15, 26–31].

Першою перевагою розробленої системи є підтримка кількох джерел пошуку. У поточній реалізації система працює з онлайн-магазинами Allo, MOYO, Stylus і Kibernetiki. Це дало змогу перевірити роботу інтелектуального агента не на одному джерелі, а на кількох магазинах з різною структурою сторінок і різним поданням товарних назв. Такий результат підтверджує доцільність модульної організації колекторів, оскільки логіка роботи з кожним магазином винесена в окремий компонент [10, 14, 15].

Другою перевагою є перетворення сирих результатів пошуку на структуровані картки товарів. Замість простого списку посилань система формує картку моделі, у межах якої відображаються пропозиції з різних магазинів, діапазон цін, характеристики товару та доступні офери. Це спрощує порівняння цін, оскільки користувач бачить не розрізнені сторінки магазинів, а зведений результат для конкретної моделі.

Важливим результатом є робота механізмів фільтрації. Під час тестування встановлено, що система здатна відсікати частину аксесуарів, супутніх товарів і нерелевантних позицій, які часто потрапляють у пошукову видачу магазинів. Це підвищує якість кінцевих результатів і зменшує кількість ручних дій з боку користувача. Разом з тим повністю усунути всі нерелевантні позиції лише на основі правил складно, оскільки назви товарів у різних магазинах можуть бути неповними або неоднозначними [26–36].

Окремо було проаналізовано роботу механізму групування моделей. Система враховує назву, бренд, модель, окремі атрибути та сильні варіанти

модифікацій [26–36]. Це дає змогу об’єднувати однакові або близькі пропозиції в одну картку. Перевагою такого підходу є зручність порівняння, однак обмеженням залишається залежність від якості вихідних назв. Якщо магазин подає неповну або неточну назву товару, система може потребувати додаткової перевірки або уточнення правил зіставлення.

Результат роботи інтерфейсу наведено на рисунку 3.4. На прикладі пошукового запиту Apple Macbook Air видно, що система формує зведену картку товару, відображає знайдені магазинні пропозиції, показує діапазон цін і надає користувачу інструменти фільтрації та сортування. Користувач може уточнювати результати за магазином, ціною та станом товару, а також змінювати порядок відображення карток.

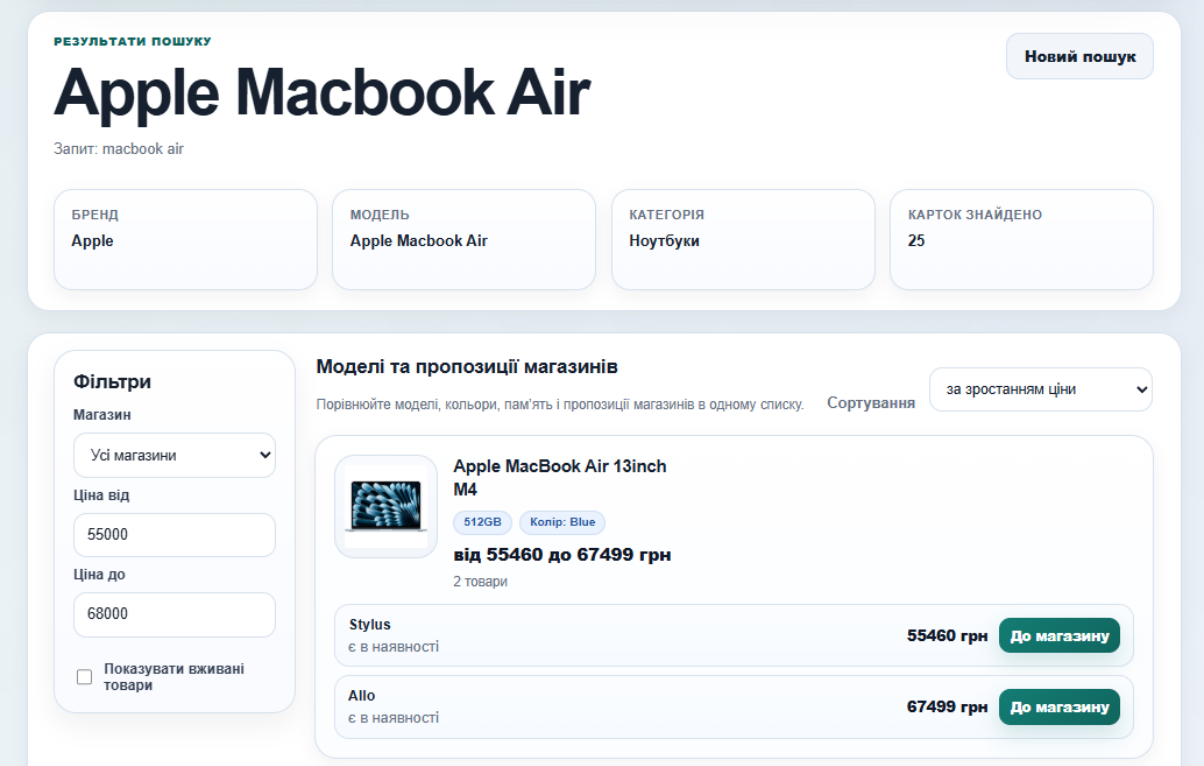

Рисунок 3.4. Приклад застосування фільтрів і сортування на сторінці результатів

Інтерфейс системи можна оцінити як достатньо зручний для базового сценарію порівняння цін. Його перевагою є те, що фільтрація та сортування виконуються без повторного повного пошуку на сервері. Це зменшує кількість зайвих запитів і робить роботу з результатами швидшою. Водночас інтерфейс

може бути вдосконалений через додавання графіків зміни цін, окремого перегляду історії пропозицій і сповіщень про зміну вартості.

До переваг реалізованої системи можна віднести:

1. Підтримку кількох онлайн-магазинів у межах одного пошукового сценарію.
2. Модульну структуру колекторів, що спрощує додавання нових джерел.
3. Формування зведених карток товарів замість простого списку посилань.
4. Фільтрацію аксесуарів і нерелевантних результатів.
5. Підтримку сортування та фільтрів у вебінтерфейсі.
6. Можливість подальшого розширення системи.

Разом з перевагами система має певні обмеження. По-перше, у поточній версії підтримується обмежена кількість джерел. По-друге, робота колекторів залежить від структури сторінок магазинів, тому зміна HTML-розмітки може потребувати оновлення відповідного модуля. По-третє, аналіз цінових тенденцій у часі реалізовано лише на рівні підготовки структури системи, без повноцінного накопичення історії цін, побудови графіків і автоматичних сповіщень. По-четверте, правила фільтрації та групування можуть потребувати уточнення для складних або неоднозначних назв товарів.

Проведений аналіз показав, що розроблена система виконує основні функції, визначені під час проєктування. Вона збирає пропозиції з кількох онлайн-магазинів, обробляє товарні назви, відсіює частину нерелевантних результатів, групує однакові моделі та подає результати у зручному вебінтерфейсі. Найбільш цінним результатом є перетворення розрізнених магазинних пропозицій на структуроване подання, придатне для порівняння.

Додаткові екрани вебсистеми та матеріали тестування подано в додатку Г.

Висновки до розділу 3

У третьому розділі подано практичну реалізацію інтелектуального агента моніторингу та аналізу цінових тенденцій. Основну увагу приділено серверній частині на Python і FastAPI, модулям збирання даних з онлайн-магазинів, механізмам нормалізації, фільтрації, групування пропозицій та клієнтській логіці роботи з результатами.

У ході реалізації створено вебсистему, що працює з чотирма джерелами пошуку: Allo, MOYO, Stylus і Kibernetiki. Модульна організація колекторів дала змогу врахувати особливості кожного магазину та залишити можливість подальшого розширення системи. Серверна логіка охоплює повний цикл обробки запиту: від збирання сирих оферів до фільтрації, групування та формування карток товарів.

Тестування показало, що система коректно обробляє різні типи пошукових запитів, відсіює нерелевантні результати, групує однакові моделі, розмежовує нові й вживані товари та дає змогу зручно порівнювати пропозиції в межах одного інтерфейсу. Отже, практична реалізація відповідає поставленій меті й може розглядатися як працездатний прототип системи моніторингу та порівняння цін.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано прикладне завдання розроблення інтелектуального агента моніторингу та аналізу цінових тенденцій на маркетплейсах. За результатами виконання роботи можна сформулювати такі висновки:

1. Проаналізовано предметну область моніторингу цін на маркетплейсах та онлайн-магазинах. Встановлено, що основними труднощами цієї сфери є постійна зміна цін, неоднорідність назв товарів, різне подання характеристик, наявність аксесуарів у пошуковій видачі, змішування нових і вживаних товарів, а також складність коректного зіставлення однакових моделей з різних джерел.

2. Досліджено існуючі сервіси порівняння та відстеження цін. Визначено, що такі сервіси спрощують пошук товарних пропозицій, однак не завжди достатньо точно виконують нормалізацію атрибутів, фільтрацію нерелевантних результатів і групування однакових моделей. Це підтвердило доцільність розроблення власної системи, орієнтованої не лише на збір цін, а й на попередню обробку товарних даних.

3. Сформульовано вимоги до інтелектуального агента та спроєктовано архітектуру програмної системи. Система побудована за клієнт-серверним принципом і складається з рівня користувацького інтерфейсу, рівня прикладної логіки, модулів збирання даних з онлайн-магазинів та рівня збереження даних. Така архітектура дала змогу розділити логіку збору, обробки, групування та відображення результатів.

4. Розроблено структуру даних і логіку обробки товарних пропозицій. У роботі передбачено перехід від сирих результатів пошуку до структурованих карток товарів. Для цього використано виділення ключових атрибутів товару, нормалізацію назв, фільтрацію аксесуарів і нерелевантних позицій, розмежування схожих модифікацій та групування однакових або близьких моделей.

5. Реалізовано прототип вебсистеми інтелектуального агента. У практичній частині створено модулі збирання даних з онлайн-магазинів Allo, MOYO, Stylus і Kibernetiki, реалізовано серверну логіку на Python і FastAPI, механізми обробки результатів, формування карток товарів, а також вебінтерфейс із фільтрацією, сортуванням і переглядом магазинних пропозицій.

6. Проведено тестування роботи системи на різних типах пошукових запитів. Перевірка показала, що система коректно обробляє точні й узагальнені запити, відсіює значну частину нерелевантних результатів, групує пропозиції однакових моделей, підтримує розмежування нових і вживаних товарів та дає користувачу змогу зручно працювати з результатами через фільтри й сортування.

Отже, мету кваліфікаційної роботи досягнуто. Розроблений інтелектуальний агент виконує автоматизований пошук, збирання, обробку, зіставлення та наочне подання товарних пропозицій з різних онлайн-магазинів. Практичне значення роботи полягає у створенні працездатного прототипу вебсистеми, який може бути використаний як основа для подальшого розвитку сервісу порівняння цін.

Подальший розвиток системи доцільно пов’язати з підключенням нових джерел даних, накопиченням історії цін, побудовою графіків зміни вартості, реалізацією сповіщень про зниження ціни та вдосконаленням алгоритмів зіставлення товарних моделей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Машкіна І. В., Абрамов В. О., Носенко Т. І., Іваніченко Є. В., Яскевич В. О. Навчально-методичний посібник до виконання та захисту кваліфікаційної роботи бакалавра для студентів спеціальності 122 «Комп'ютерні науки». Київ, 2024. 43 с.

2. Про вищу освіту : Закон України від 01.07.2014 № 1556-VII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/1556-18 (дата звернення: 10.03.2026).

3. Про освіту : Закон України від 05.09.2017 № 2145-VIII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2145-19 (дата звернення: 10.03.2026).

4. Стандарт вищої освіти за спеціальністю 122 «Комп'ютерні науки» для першого (бакалаврського) рівня вищої освіти : затв. наказом МОН України від 10.07.2019 № 962 // Міністерство освіти і науки України. URL: https://mon.gov.ua/static-objects/mon/sites/1/vishcha-osvita/zatverdzeni%20standarty/2019/07/12/122-kompyut.nauk.bakalavr-1.pdf (дата звернення: 12.03.2026).

5. Збірник тез XII Всеукраїнської науково-практичної конференції молодих учених, 14 трав. 2026 р., м. Київ / Київський столичний університет імені Бориса Грінченка. с. 70–78.

6. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Чинний від 2016-07-01]. Вид. офіц. Київ : ДП «УкрНДНЦ», 2016. 16 с.

7. ДСТУ ISO 5807:2016. Обробляння інформації. Символи та угоди щодо документації стосовно даних, програм та системних блок-схем, схем мережевих програм та схем системних ресурсів (ISO 5807:1985, IDT). [Чинний від 2016-10-10]. Київ : ДП «УкрНДНЦ», 2016.

8. Python Software Foundation. The Python Tutorial [Електронний ресурс] // Python 3 Documentation. URL: https://docs.python.org/3/tutorial/index.html (дата звернення: 15.03.2026).

9. Python Software Foundation. sqlite3 – DB-API 2.0 interface for SQLite databases [Електронний ресурс] // Python 3 Documentation. URL: https://docs.python.org/3/library/sqlite3.html (дата звернення: 15.03.2026).

10. FastAPI [Електронний ресурс] : documentation. URL: https://fastapi.tiangolo.com/ (дата звернення: 16.03.2026).

11. SQLAlchemy Documentation [Електронний ресурс]. URL: https://docs.sqlalchemy.org/ (дата звернення: 16.03.2026).

12. SQLAlchemy ORM [Електронний ресурс]. URL: https://docs.sqlalchemy.org/orm/ (дата звернення: 16.03.2026).

13. Pallets. Jinja Documentation [Електронний ресурс]. URL: https://jinja.palletsprojects.com/ (дата звернення: 18.03.2026).

14. Reitz K. Requests: HTTP for Humans [Електронний ресурс] // Requests Documentation. URL: https://requests.readthedocs.io/en/latest/ (дата звернення: 18.03.2026).

15. Richardson L. Beautiful Soup Documentation [Електронний ресурс]. URL: https://beautiful-soup-4.readthedocs.io/en/latest/ (дата звернення: 19.03.2026).

16. Uvicorn Documentation [Електронний ресурс]. URL: https://uvicorn.dev/ (дата звернення: 20.03.2026).

17. Pydantic Documentation [Електронний ресурс]. URL: https://pydantic.dev/docs/ (дата звернення: 21.03.2026).

18. SQLite Documentation [Електронний ресурс]. URL: https://sqlite.org/docs.html (дата звернення: 21.03.2026).

19. MDN Web Docs. HTML: HyperText Markup Language [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 22.03.2026).

20. MDN Web Docs. CSS: Cascading Style Sheets [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 22.03.2026).

21. MDN Web Docs. JavaScript Guide [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 22.03.2026).

22. OECD. Unpacking E-Commerce: Business Models, Trends and Policies [Електронний ресурс]. Paris : OECD Publishing, 2019. URL: https://www.oecd.org/en/publications/unpacking-e-commerce_23561431-en.html (дата звернення: 24.03.2026).

23. UNCTAD. E-commerce and the Digital Economy [Електронний ресурс]. URL: https://unctad.org/topic/ecommerce-and-digital-economy (дата звернення: 25.03.2026).

24. Eurostat. E-commerce statistics for individuals [Електронний ресурс]. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=E-commerce_statistics_for_individuals (дата звернення: 26.03.2026).

25. Ковальова О. М., Кірсанова В. В. Основні форми інтернет-торгівлі: особливості, переваги, недоліки // Економіка та держава. 2020. № 7. С. 85–92. DOI: 10.32702/2306-6806.2020.7.85.

26. Ristoski P., Petrovski P., Mika P., Paulheim H. A Machine Learning Approach for Product Matching and Categorization: Use Case: Enriching Product Ads with Semantic Structured Data // Semantic Web. 2018. Vol. 9, No. 5. P. 707–728. DOI: 10.3233/SW-180300.

27. Petrovski P., Primpeli A., Meusel R., Bizer C. The WDC Gold Standards for Product Feature Extraction and Product Matching // E-Commerce and Web Technologies. 2017. P. 73–86. DOI: 10.1007/978-3-319-53676-7_6.

28. Kannan A., Givoni I. E., Agrawal R., Fuxman A. Matching Unstructured Product Offers to Structured Product Specifications // Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2011. P. 384–392. DOI: 10.1145/2020408.2020474.

29. Gopalakrishnan V., Iyengar S. P., Madaan A., Rastogi R., Sengamedu S. H. Matching Product Titles Using Web-Based Enrichment // Proceedings of the 21st ACM International Conference on Information and Knowledge Management. 2012. P. 605–614. DOI: 10.1145/2396761.2396839.

30. Köpcke H., Thor A., Thomas S., Rahm E. Tailoring Entity Resolution for Matching Product Offers // Proceedings of the 15th International Conference on Extending Database Technology. 2012. P. 545–550. DOI: 10.1145/2247596.2247662.

31. Li J., Dou Z., Zhu Y., Zuo X., Wen J.-R. Deep Cross-Platform Product Matching in E-commerce // Information Retrieval Journal. 2020. Vol. 23, No. 2. P. 136–158. DOI: 10.1007/s10791-019-09360-1.

32. Peeters R., Bizer C. Supervised Contrastive Learning for Product Matching // Companion Proceedings of the Web Conference 2022. 2022. P. 248–251. DOI: 10.1145/3487553.3524254.

33. Choi J. I., Kallumadi S., Mitra B., Agichtein E., Javed F. Semantic Product Search for Matching Structured Product Catalogs in E-Commerce [Електронний ресурс] // arXiv preprint. 2020. arXiv:2008.08180. URL: https://arxiv.org/abs/2008.08180 (дата звернення: 28.03.2026).

34. Nguyen T. V., Rao N., Subbian K. Learning Robust Models for e-Commerce Product Search // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020. P. 6861–6869. DOI: 10.18653/v1/2020.acl-main.614.

35. Martinek A., Łukasik S., Gandomi A. H. Text-Based Product Matching: Semi-Supervised Clustering Approach [Електронний ресурс] // arXiv preprint. 2024. arXiv:2402.10091. URL: https://arxiv.org/abs/2402.10091 (дата звернення: 04.04.2026).

36. Macková K., Pilát M. ProMap: Datasets for Product Mapping in E-commerce [Електронний ресурс] // CoRR. 2023. abs/2309.06882. URL: https://arxiv.org/abs/2309.06882 (дата звернення: 07.04.2026).

ДОДАТОК А

Таблиця А.1

Основні проблеми автоматизованого збору та зіставлення товарних пропозицій

Проблема	**Прояв у пошуку**	**Наслідок для користувача**	**Потрібний механізм розв'язання**
Неоднорідність назв товарів	Одна модель має різні назви в різних магазинах	Неможливо автоматично зіставити однакові товари простим порівнянням рядків	Нормалізація назв і виділення структурованих атрибутів
Наявність аксесуарів у видачі	Разом із основним товаром з'являються чохли, кабелі, захисне скло, адаптери	Видача стає шумною й менш корисною	Фільтрація аксесуарів за маркерами і правилами релевантності
Різне представлення цін	На сторінці є поточна, стара, акційна або службова ціна	Можливий вибір неправильної ціни для аналізу	Перевірка контексту ціни та відбір коректного цінового значення
Змішування нових і вживаних товарів	У результатах одночасно відображаються нові, вживані та відновлені товари	Порівняння цін стає некоректним	Визначення стану товару та окреме керування такими пропозиціями
Схожі, але не тотожні моделі	У пошуку з'являються Pro, Plus, Max, Air, SE та інші модифікації	Різні моделі можуть бути помилково об'єднані	Урахування сильних варіантів моделі та атрибутів
Динамічність вебсторінок магазинів	Зміна HTML-структури або назв елементів	Ламається механізм парсингу окремого магазину	Модульна архітектура окремих колекторів
Неповнота атрибутів у назві	Не всі магазини вказують колір, пам'ять або модифікацію	Частина карток відображається неповно	Часткове доукомплектування атрибутів із суміжних оферів
Баланс між точністю і повнотою	Надто жорсткі або надто м'які правила відбору	Або втрачаються релевантні результати, або з'являється шум	Комбінація правил релевантності та повторної фільтрації

Таблиця А.2

Функціональні та нефункціональні вимоги до системи

Тип вимоги	Формулювання	Практичне значення
Функціональна	Прийом текстового пошукового запиту користувача	Запускає основний сценарій роботи системи
Функціональна	Вибір категорії товару	Дає змогу звузити область пошуку
Функціональна	Збирання пропозицій із кількох онлайн-магазинів	Формує початковий набір даних для порівняння
Функціональна	Виділення атрибутів товару з назв	Сприяє розпізнаванню бренду, моделі, пам’яті, кольору та інших характеристик
Функціональна	Відсікання аксесуарів і нерелевантних результатів	Підвищує точність видачі
Функціональна	Групування однакових моделей у спільні картки	Полегшує порівняння товарів між магазинами
Функціональна	Відображення цін, доступності та стану товару	Дає користувачу повну картину ринку
Функціональна	Фільтрація за магазином, ціною і станом товару	Дає змогу гнучко уточнювати результати
Функціональна	Сортування карток за ціною	Полегшує вибір вигідної пропозиції
Функціональна	Оновлення кількості видимих карток після фільтрів	Робить інтерфейс інформативним і коректним
Нефункціональна	Модульність архітектури	Спрощує підтримку та розширення системи
Нефункціональна	Розширюваність	Спрощує додавання нових магазинів і функцій
Нефункціональна	Прийнятна швидкодія	Підтримує комфортне використання системи
Нефункціональна	Стійкість до часткових збоїв	Один збій колектора не зупиняє всю систему
Нефункціональна	Зручність користувацького інтерфейсу	Полегшує взаємодію з результатами пошуку
Нефункціональна	Підтримуваність програмного коду	Спрощує виправлення помилок і подальшу розробку
Нефункціональна	Достатня точність зіставлення моделей	Визначає якість кінцевих результатів

Таблиця А.3

Основні файли та модулі програмного проєкту

Шлях до файла	Призначення
app/main.py	Головний модуль вебзастосунку. Містить маршрути сторінок, обробку пошукового запиту та передавання результатів у шаблони інтерфейсу.
app/services/matcher.py	Модуль інтелектуальної обробки назв товарів. Виконує виділення атрибутів, перевірку релевантності, нормалізацію та зіставлення моделей.
app/services/product_service.py	Центральний сервіс обробки результатів пошуку. Відповідає за збирання оферів, фільтрацію, дедуплікацію, групування та побудову карток товарів.
app/collectors/__init__.py	Файл реєстрації активних джерел пошуку та конфігурації колекторів магазинів.
app/collectors/base.py	Базові допоміжні функції для модулів збирання даних, спільні утиліти роботи з HTML і HTTP-запитами.
app/collectors/allo.py	Колектор збирання товарних пропозицій з онлайн-магазину Allo.
app/collectors/moyo.py	Колектор збирання товарних пропозицій з онлайн-магазину MOYO.
app/collectors/stylus.py	Колектор збирання товарних пропозицій з онлайн-магазину Stylus.
app/collectors/kibernetiki.py	Колектор збирання товарних пропозицій з онлайн-магазину Kibernetiki.
app/schemas.py	Опис структур даних, які використовуються для передавання нормалізованих оферів, карток товарів і результатів пошуку.
app/templates/base.html	Базовий HTML-шаблон, який визначає спільну структуру сторінок вебсистеми.
app/templates/home.html	Шаблон головної сторінки з формою введення пошукового запиту та вибору категорії.
app/templates/results.html	Шаблон сторінки результатів пошуку з картками товарів, фільтрами та списком магазинних оферів.
app/static/css/styles.css	Файл стилів, що визначає візуальне оформлення інтерфейсу вебсистеми.
app/static/js/results.js	Клієнтська логіка сторінки результатів: фільтрація, сортування, оновлення карток і кількості видимих товарів.

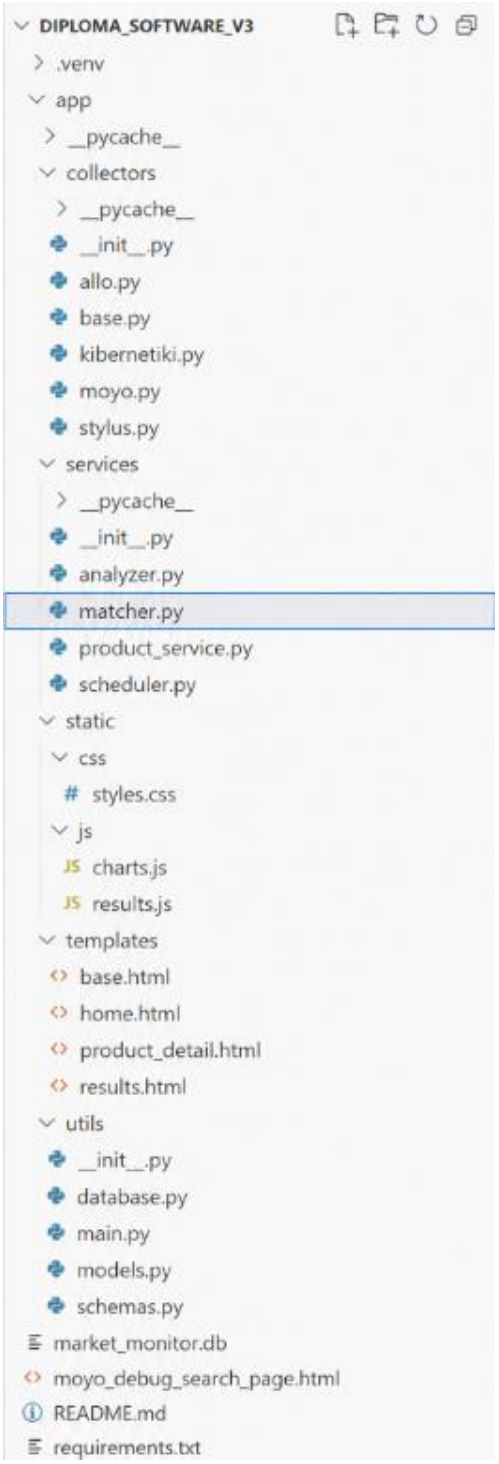

Рисунок А.1. Структура каталогів програмного проєкту

ДОДАТОК Б

```python
@app.post("/search")
def search(
    request: Request,
    query: str = Form(...),
    category: str = Form(default="all"),
    db: Session = Depends(get_db),
) -> object:
    result = search_and_store_product(db, query, category)
    return templates.TemplateResponse(
        "results.html",
        {
            "request": request,
            "query": query,
            "product": result.product,
            "cards": result.product_cards,
            "entity": result.entity,
        },
    )
```

Лістинг Б.1. Фрагмент реалізації маршруту пошуку

```python
def filter_display_offers(entity_or_query: ProductEntity | str, offers: list) -> list:
    entity = _ensure_entity(entity_or_query)
    display = []
    for offer in offers:
        attrs = extract_attributes(offer.source_product_name)
        scored = score_offer(entity, attrs)
        if _exclude_from_display(entity, attrs, scored):
            continue
        if scored["score"] >= DISPLAY_THRESHOLD:
            display.append(offer)
    return display
```

Лістинг Б.2. Фрагмент відбору релевантних пропозицій

```
def build_product_cards(offers: list[NormalizedOffer]) -> list[ProductOfferCard]:
    grouped: dict[tuple[str, str | None, str | None, str | None], list[NormalizedOffer]]
= {}

    for offer in offers:
        attrs = extract_attributes(offer.source_product_name)
        canonical_brand = _canonical_card_brand(attrs)
        canonical_family = attrs.family or ""
        canonical_model = attrs.model or ""
        key = (canonical_brand, canonical_family, canonical_model,
               attrs.storage, attrs.ram, attrs.color)
        grouped.setdefault(key, []).append(offer)
```

Лістинг Б.3. Фрагмент групування товарних пропозицій у картки

ДОДАТОК В

```
def collect_offers(query: str, limit: int = 12) -> list[NormalizedOffer]:
    for pattern in SEARCH_URLS:
        soup = fetch_soup(pattern.format(query=encoded_query(query)))
        if soup is None:
            continue
        offers = _parse_cards(soup, pattern.format(query=encoded_query(query)),
          limit)
        if offers:
            return offers
    return []
```

Лістинг В.1. Фрагмент реалізації модуля збирання даних з магазину Allo

```
def collect_offers(query: str, limit: int = 12) -> list[NormalizedOffer]:
    collected: list[NormalizedOffer] = []
    for pattern in SEARCH_URLS:
        search_url = pattern.format(query=encoded_query(query))
        soup = fetch_soup(search_url)
        if soup is None:
            continue
        offers = _parse_cards(soup, fallback_url=search_url, limit=limit * 4)
        if not offers:
            continue
        collected.extend(offers)
        unique = _unique_by_url(collected, limit * 4)
        non_refurb = [offer for offer in unique if not
_looks_like_refurb(offer.source_product_name.lower())]
        if non_refurb:
            return non_refurb[:limit]
    return []
```

Лістинг В.2. Фрагмент реалізації модуля збирання даних з магазину MOYO

```
def collect_offers(query: str, limit: int = 24) -> list[NormalizedOffer]:
    for pattern in SEARCH_URLS:
        search_url = pattern.format(query=encoded_query(query))
        soup = fetch_soup(search_url, headers=STYLUS_HEADERS)
        if soup is None:
            continue

        jsonld_offers = extract_jsonld_offers(soup, "stylus", BASE_URL, limit=limit
* 3)
        jsonld_offers = _filter_stylus_offers(jsonld_offers, query, limit)
        if jsonld_offers:
            return jsonld_offers

        offers = _parse_cards(soup, fallback_url=search_url, limit=limit)
        offers = _filter_stylus_offers(offers, query, limit)
        if offers:
            return offers
    return []
```

Лістинг В.3. Фрагмент реалізації модуля збирання даних з магазину Stylus

```
def collect_offers(query: str, limit: int = 24) -> list[NormalizedOffer]:
    for pattern in SEARCH_URLS:
        search_url = pattern.format(query=encoded_query(query))
        soup = fetch_soup(search_url, headers=KIBER_HEADERS)
        if soup is None:
            continue

        offers = _parse_cards(soup, fallback_url=search_url, limit=limit,
query=query)
        if offers:
            return offers

    return []
```

Лістинг В.4. Фрагмент реалізації модуля збирання даних з магазину Kibernetiki

ДОДАТОК Г

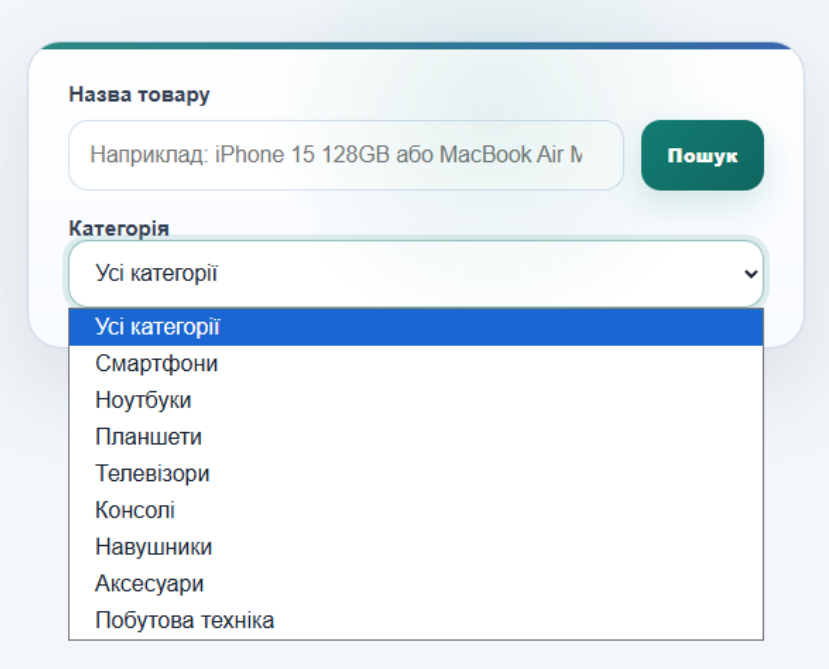

Рисунок Г.1. Додатковий приклад головної сторінки вебсистеми

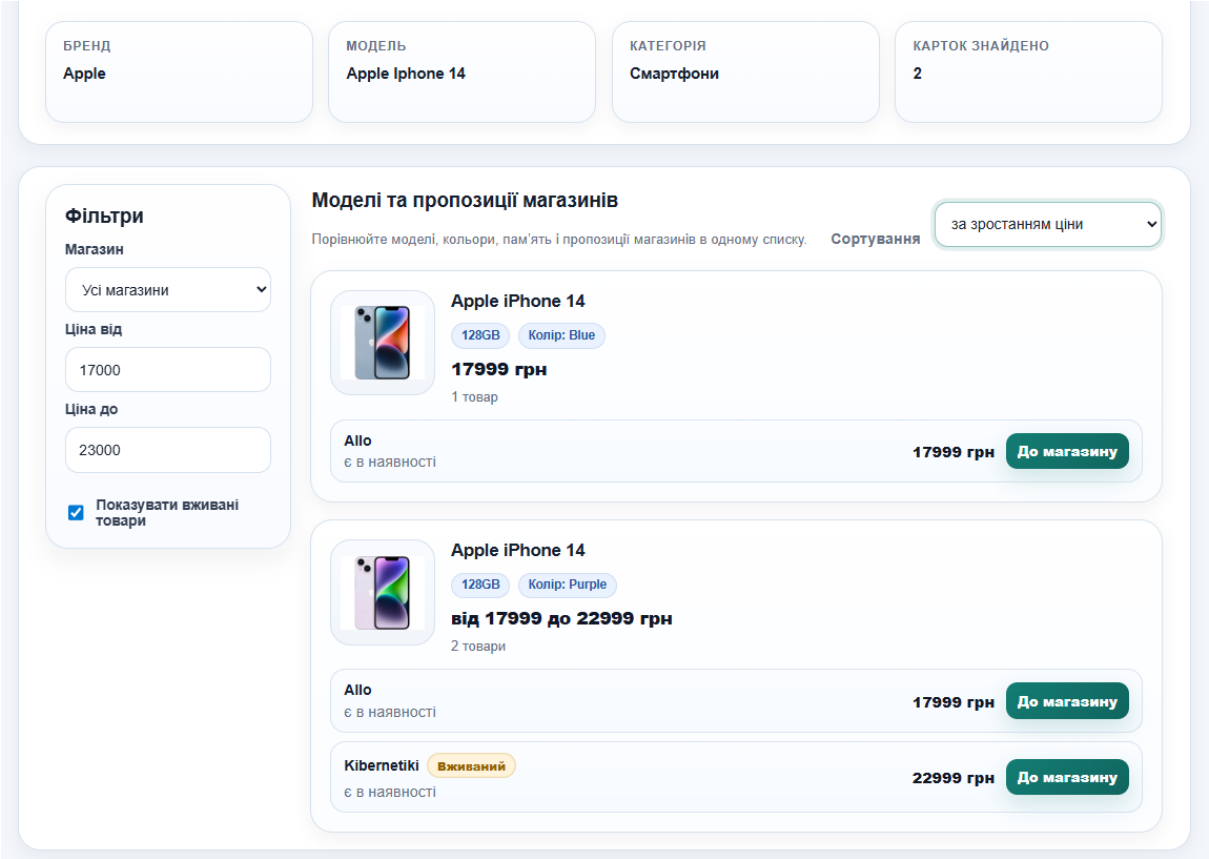

Рисунок Г.2. Додатковий приклад сторінки результатів пошуку

Таблиця Г.1

Додаткові сценарії функціонального тестування системи

Пошуковий запит	**Мета перевірки**	**Очікуваний результат**	**Фактичний результат**
iPhone 15	Перевірка базового пошуку смартфона	Відображення релевантних моделей без аксесуарів	Очікуваний результат досягнуто
iPhone 15 Pro	Перевірка сильної модифікації моделі	У видачі лише Pro-версії	Очікуваний результат досягнуто
iPhone 16	Перевірка широкого запиту без уточнення атрибутів	Видача кількох релевантних карток сімейства	Очікуваний результат досягнуто
MacBook Air	Перевірка узагальненого пошуку ноутбука	Показ релевантних моделей без надмірного звуження	Очікуваний результат досягнуто
MacBook Air M4	Перевірка уточненого запиту для ноутбука	Видача звужується до відповідної модифікації	Очікуваний результат досягнуто
iPad Air	Перевірка пошуку планшета	У видачі залишаються лише релевантні планшети	Очікуваний результат досягнуто
ThinkPad	Перевірка відсіювання нерелевантних товарів	Відсутність сумок, рюкзаків і сторонніх аксесуарів	Очікуваний результат досягнуто
Galaxy Tab	Перевірка роботи на іншій категорії планшетів	Коректне відображення карток і магазинних оферів	Очікуваний результат досягнуто

```
function applyControls() {
    if (!list) return;

    const cards = Array.from(list.querySelectorAll(".product-card"));
    const min = Number(minPrice?.value || 0);
    const max = Number(maxPrice?.value || 0);
    const store = storeFilter?.value || "all";
    const showUsed = Boolean(includeUsed?.checked);

    cards.forEach((card) => {
        const state = updateCardOffers(card, store, min, max, showUsed);
        card.classList.toggle("hidden", state.visibleCount === 0);
    });

    sortVisibleCards();
    updateVisibleCardsCount();
}
```

Лістинг Г.1. Фрагмент клієнтської логіки фільтрації результатів