МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ Економіки та бізнес освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність Комп'ютерні науки

Форма навчання Денна

КВАЛІФІКАЦІЙНА РОБОТА

Баніта Євгенія Руслановича

(прізвище, ім'я, по батькові здобувача)

на тему «Створення веб-додатка для порівняння даних з інтернет-магазинів »

(повна назва теми)

за матеріалами

(повна назва бази дослідження)

науковий керівник - Бєлінський А.О.

(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 09.06 20 25 р. №12

Завідувач кафедри

(підпис)

к.е.н., доцент Радько В.М.

Наук. ступінь, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра Економіки та цифрового бізнесу
Освітній ступінь Бакалавр
Спеціальність Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _____В.М. Радько

"07" квітня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА ЗДОБУВАЧУ

Баніту Євгенію Руслановичу

1. Тема роботи «Створення веб-додатка для порівняння даних з інтернет-магазинів»

науковий керівник роботи Бєлінський Андрій Олександрович

затверджені наказом вищого навчального закладу від «04» квітня 2025 р. № 224-ст (д/ф)
№ 151-ст (з/ф)

2. Строк подання здобувачем роботи 31.05.2025р.

3. Зміст кваліфікаційної роботи бакалавра, об'єкт, предмет та мета дослідження:

Розділ 1 Теоретичні засади розробки веб-застосунків для порівняння товарів в електронній комерції

Розділ 2 Концептуальне проєктування веб-додатку для порівняння товарів з Інтернет-магазинів

Розділ 3 Реалізація та тестування веб-додатку для порівняння товарів

Об'єкт дослідження – інформаційні технології автоматизованого збору, обробки та

РЕФЕРАТ

Робота складається з 71 сторінок, 45 рисунків, 13 джерел, 3 додатків.

Об’єктом дослідження є процес автоматизованого збору, обробки та порівняння інформації про товари з різних електронних комерційних платформ (інтернет-магазинів) з метою надання користувачам зручного інструменту для аналізу цін, характеристик та інших параметрів товарів.

Розробити та реалізувати веб-додаток, що забезпечує автоматизований збір, обробку та порівняння даних про товари з різних інтернет-магазинів з метою підвищення зручності для користувачів при здійсненні покупок онлайн в інтернет-магазинах.

Результатами нашого дослідження, виявлено аналіз існуючих рішень для порівняння товарів, виявлено їх переваги та недоліки, визначено напрям для вдосконалення. Спроєктовано архітектуру веб-додатку, що забезпечує взаємодію між основними компонентами системи, розроблено користувацький інтерфейс, який дозволяє ефективно та зручно використовувати розроблений додаток. Проведено тестування веб-додатку, що підтвердило його працездатність, зручність використання та відповідність вимогам.

Областю застосування можна назвати використання розробленого додатку у сфері електронної комерції, маркетингових досліджень та освітніх проєктах.

Основна сфера використання, це допомога споживачам у швидкому виборі товарів у найкращих умовах. Додатковими ж, є адаптованість додатку до збору даних про ціни, характеристики та доступність, а також як приклад реалізації сучасного веб-сервісу, для навчання студенів розробці.

Ключові слова: АРХІТЕКТУРНА МОДЕЛЬ СИСТЕМИ, ВЕБ-ДОДАТОК, ІНТЕРНЕТ-МАГАЗИН, ІНТЕРФЕЙС КОРИСТУВАЧА, ПАРСИНГ ДАНИХ, ПОРІВНЯННЯ ТОВАРІВ, РОЗРОБКА ДОДАТКУ.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface

XML – eXtensible Markup Language

ЗСУ – Збройні Сили України

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

XHTML – eXtensible HyperText Markup Language

JS – JavaScript

PHP – Hypertext Preprocessor

IP – Internet Protocol

URL – Uniform Resource Locator

MIT – Massachusetts Institute of Technology

EJS – Embedded JavaScript

SCSS – Sassy Cascading Style Sheets

UI – User Interface

JSON – JavaScript Object Notation

CMD – Command

DOM – Document Object Model

ВСТУП

В умовах стрімкого розвитку електронної комерції все більше користувачів віддають перевагу покупкам в інтернет-магазинах. Це пов’язано зі зручністю, широким асортиментом товарів та можливістю швидко порівнювати пропозиції. Однак зростання кількості інтернет-магазинів ускладнює процес вибору товару, оскільки інформація розпорошена по різних платформах, а кожен сайт має свій інтерфейс та методи представлення даних. Існуючі сервіси для порівняння товарів (наприклад, Hotline, Price.ua, Google Shopping) забезпечують користувачам певний рівень зручності, але мають низку обмежень:

- неповне охоплення інтернет-магазинів або категорій товарів;
- затримка в оновленні інформації про ціни та наявність;
- залежність від наявності офіційної інтеграції з магазинами через API або партнерські програми;
- обмежена гнучкість у фільтрації та аналізі параметрів товарів.

Крім того, не всі магазини відкрито надають свої дані для агрегації, що створює необхідність розробки рішень для збору даних шляхом парсингу веб-сторінок або альтернативних методів інтеграції. У цьому контексті особливої актуальності набуває розробка веб-застосунку, який дозволить автоматично збирати, обробляти та порівнювати дані з кількох інтернет-магазинів у режимі реального часу. Такий інструмент повинен забезпечувати:

- зручний та адаптивний інтерфейс користувача;
- актуальну та повну інформацію про товари;
- можливість гнучкого порівняння характеристик;
- незалежність від партнерських угод з магазинами.

Розробка такого веб-застосунку вимагає дослідження методів збору та структурування інформації, вибору ефективної архітектури програмного забезпечення та реалізації алгоритмів для коректного порівняння товарів з різних джерел. Таким чином, існує об’єктивна потреба в нових рішеннях, які можуть ефективно автоматизувати процес порівняння товарів з інтернет-магазинів,

підвищуючи зручність для користувачів та надаючи точну та актуальну інформацію.

Суть цієї роботи полягає в розробці веб-додатку, який автоматично збирає, обробляє та надає користувачеві можливість порівнювати товари з різних інтернет-магазинів. Основна ідея полягає в об’єднанні даних, отриманих із зовнішніх джерел (веб-сторінок, API), для подальшого представлення у вигляді зручного інтерфейсу, де користувач може порівнювати ціни, характеристики, рейтинги, акції тощо.

Такий додаток дозволяє:

- заощадити час користувача на пошук та порівняння товарів вручну;
- отримувати більш повну та об’єктивну інформацію;
- покращити досвід онлайн-шопінгу;
- сприяти прозорості цінової політики магазинів.

Основними компонентами веб-додатку є модулі збору та обробки даних, система порівняння та відображення результатів у зручному для користувача вигляді.

Станом на сьогоднішній день існує низка веб-сервісів, які частково реалізують функціональність порівняння товарів - зокрема, Hotline.ua, Price.ua, Google Shopping, E-katalog та інші. Однак ці сервіси здебільшого працюють на основі партнерських угод з інтернет-магазинами та використовують фіксовані джерела даних через API або XML-канали.

Основними обмеженнями таких сервісів є обмежена кількість інформації, затримка в її оновленні, відсутність гнучких фільтрів для більш точного пошуку за тонкими характеристиками та неможливість адаптуватися до потреб конкретного користувача.

У науковому та практичному плані відбувається активний розвиток напрямків, пов’язаних зі скрейпінгом веб-сторінок, машинним навчанням для порівняння даних, а також побудовою користувацьких інтерфейсів з високим рівнем зручності. Саме тому створення адаптивного та незалежного веб-застосунку для порівняння товарів залишається актуальним та науково

обґрунтованим завданням.

У сучасних умовах активного розвитку електронної комерції споживачі стикаються з проблемою вибору товарів серед великої кількості пропозицій в інтернет-магазинах та дуже великою різницею в ціні на той самий товар. Відсутність єдиного інструменту для швидкого, точного та об’єктивного порівняння товарів за ціною, характеристиками та відгуками створює потребу в ефективному рішенні. Розробка веб-додатку для порівняння даних з інтернет-магазинів дозволяє автоматизувати цей процес, підвищити зручність використання, заощадити час та сприяти прийняттю більш вигідних рішень споживачами. Саме це визначає актуальність теми дослідження.

Метою роботи є розробка веб-додатку для автоматизованого збору, обробки та порівняння даних про товари з різних інтернет-магазинів, що дозволить користувачам робити ефективний вибір товарів на основі порівняння цін, характеристик та відгуків.

Завданням дослідження для цього проєкту є:

- аналіз існуючих методів збору та обробки даних;
- оцінка технічних вимог до веб-застосунку для порівняння товарів;
- розробка архітектури веб-застосунку, яка забезпечує збір, обробку та порівняння даних;
- тестування веб-застосунку на наявність помилок.

Практичне значення роботи полягає у створенні інструменту, який дозволяє споживачам швидко та зручно порівнювати товари з різних інтернет-магазинів, що сприяє прийняттю більш обґрунтованих та вигідних рішень під час покупок. Розроблений веб-додаток може бути використаний:

- як основа для комерційного сервісу порівняння цін;
- у діяльності маркетингових агентств для аналізу ринку;
- у навчальному процесі як приклад впровадження систем збору та обробки даних з відкритим кодом;
- для розвитку технологій у сфері електронної комерції, зокрема, автоматизації процесів пошуку, фільтрації та аналізу інформації.

Завдяки масштабуванню, адаптивності архітектури та універсальному підходу до збору даних, система може легко підлаштовуватися до різного ринку.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ ДЛЯ ПОРІВНЯННЯ ТОВАРІВ В ЕЛЕКТРОННІЙ КОМЕРЦІЇ

1.1 Сучасний стан та проблеми порівняння даних про товари

У сучасному цифровому світі електронна комерція є одним із секторів економіки, що найшвидше розвиваються. Щодня велика кількість користувачів здійснюють онлайн-покупки, вибираючи з широкого асортименту товарів, доступних у різних інтернет-магазинах. У цих умовах прийняття раціонального рішення про покупку вимагає часу, зусиль та доступу до достовірної, актуальної та структурованої інформації.

Однак більшість інтернет-магазинів працюють незалежно один від одного, мають власні формати представлення товарів, унікальні назви, структури сторінок та способи опису характеристик. Це ускладнює порівняння товарів, особливо для користувачів, які хочуть знайти найвигіднішу пропозицію або вибрати товар, що найкраще відповідає їхнім потребам. Ручне порівняння таких даних є неефективним та призводить до втрати часу, а в деяких випадках – до невигідних покупок.

Саме тому існує об’єктивна потреба в розробці спеціалізованого програмного забезпечення – веб-додатку, який автоматизує процес збору, обробки та порівняння пропозицій товарів з різних онлайн-ресурсів. Такий інструмент дозволяє:

- централізувати інформацію з багатьох джерел;
- швидко знаходити товари за заданими параметрами;
- об’єктивно порівнювати ціни, характеристики, наявність знижок;
- покращити загальний користувацький досвід під час онлайн-шопінгу.

Крім того, розробка такого веб-застосунку є актуальною в професійному та науково-дослідному контексті. Вона передбачає використання сучасних

технологій: веб-скрейпінгу, REST API, обробки неструктурованих даних, побудови масштабованої архітектури веб-застосунку, фронтенд-розробки тощо. Це дає змогу продемонструвати знання та навички в різних галузях програмної інженерії.

Актуальність теми також посилюється відсутністю універсального, відкритого та адаптивного рішення, яке б дозволяло легко додавати нові джерела даних, оновлювати правила обробки сторінок без переписування всього застосунку, і водночас залишалося зручним для звичайного користувача.

Таким чином, створення такого веб-застосунку має як практичне, так і науково-технічне значення. Він відповідає потребам сучасного суспільства, підвищує ефективність онлайн-торгівлі та відкриває нові можливості для аналізу ринку електронної комерції.

Сайти порівняння цін (або системи порівняння покупок) – це онлайн-платформи, які дозволяють користувачам знаходити товари з різних інтернет-магазинів, порівнювати їх за ціною, характеристиками, рейтингами та іншими параметрами. Такі сервіси виступають посередниками між споживачами та продавцями, допомагаючи швидко знайти найвигіднішу пропозицію [4].

Основні функції:

- автоматичний збір даних про товари з багатьох інтернет-магазинів;
- обробка та об’єднання інформації (назв, характеристик, зображень);
- порівняння товарів за заданими параметрами (ціна, бренд, відгуки тощо)
- перенаправлення користувача на веб-сайт магазину для здійснення покупки.

Інтернет-магазин (англ. Internet shop, англ. Online shop) – місце в інтернеті, де відбувається прямий продаж товарів споживачеві (юридичній або фізичній особі), враховуючи доставку. При цьому розміщення споживацької інформації, замовлення товару і угода відбуваються там само, всередині мережі (на сайті інтернет-магазину) [4].

Існує кілька ефективних способів вирішення проблеми підвищення

швидкості онлайн-шопінгу, які можна реалізувати як з боку користувача, так і з боку розробників платформи. Ось ключові напрямки:

- Використання сервісів порівняння товарів:

автоматичне порівняння цін, характеристик та наявності товарів у різних магазинах.

користувач швидше знаходить найвигіднішу пропозицію, не переглядаючи десятки сайтів.

- Фільтрація та сортування:

надання гнучких фільтрів за брендом, ціною, рейтингом, категорією тощо.

- Персоналізовані рекомендації:

використання алгоритмів штучного інтелекту для вибору товарів на основі історії переглядів, покупок або уподобань.

- Зручні мобільні додатки:

оптимізовані для смартфонів інтерфейси зі швидким доступом до улюблених категорій та оплатою в один клік.

- Автоматизоване заповнення форм:

збереження платіжних та адресних даних для пришвидшення процесу замовлення.

- Використання веб-додатків зі збором даних:

спеціальні веб-сервіси, які автоматично збирають, нормалізують та відображають інформацію з різних джерел.

- Інтеграція з доставкою та відстеженням:

можливість одразу бачити умови доставки, терміни та відстежувати замовлення.

Під час аналізу проблемної області було встановлено, що на даний момент електронна комерція характеризується значним зростанням кількості онлайн-магазинів і товарних пропозицій. Це створює об’єктивну потребу в інструментах, які дозволяють швидко й ефективно орієнтуватися в інформаційному потоці та приймати рішення щодо купівлі.

Існуючі сервіси порівняння цін частково вирішують це завдання, однак мають низку обмежень: залежність від партнерських угод з магазинами, обмежений набір джерел, недостатню гнучкість та обмежену персоналізацію для користувача. Крім того, більшість таких платформ не надають актуальну інформацію в режимі реального часу або працюють лише з великими ретейлерами.

Таким чином, наявні рішення не повністю задовольняють потреби сучасних користувачів, що зумовлює актуальність створення нового веб-додатка, здатного автоматизовано збирати, обробляти й порівнювати дані з широкого кола інтернет-магазинів з урахуванням актуальних вимог до швидкості, точності та зручності користування.

1.2 Аналіз існуючих веб-сервісів для порівняння товарів

Одним із завдань даного розділу є вивчення наявних рішень у сфері веб-додатків, функціональність яких відповідає меті обраного проєкту. Основною ідеєю створення власного веб-додатку є оптимізація часу пошуку необхідного товару, забезпечення оперативного доступу до його характеристик, а також автоматизоване порівняння цінових пропозицій з метою уникнення переплати за ідентичний товар чи послугу.

Веб-додатки для порівняння даних – це онлайн-сервіси, які автоматично здійснюють збір, обробку та відображення інформації про однакові або подібні товари з різних торговельних майданчиків. Вони надають користувачу можливість здійснити обґрунтований вибір, орієнтуючись на найбільш вигідну пропозицію [4].

У результаті дослідження ринку було виділено три найбільш популярні платформи для порівняння цін в Україні:

- Hotline;
- Price.ua;
- Google Shopping.

Зокрема, Hotline.ua є одним із провідних українських сервісів для підбору товарів та порівняння цін. Історія ресурсу розпочалася у жовтні 1992 року з випуску каталогу прайс-листів на комп'ютерну техніку, який поширювався за допомогою модемного зв'язку. Згодом з'явилася його друкована версія, що регулярно публікувалася до 2005 року. Онлайн-версія сервісу почала функціонувати з 1 січня 2006 року. Станом на 2012 рік ресурс об'єднував понад 2 млн товарних пропозицій від приблизно 1700 інтернет-магазинів. У 2014 році була запущена україномовна версія сайту. Середньодобова аудиторія порталу становить близько 100 тисяч відвідувачів, щомісячна – понад 1,5 мільйона користувачів [4].

Для повноцінного аналізу функціональності кожного із зазначених сервісів у роботі представлено скріншоти головної сторінки та сторінки з результатами пошуку.

Кожен із розглянутих веб-сервісів має низку унікальних особливостей, які формують його впізнаваність та впливають на загальну популярність серед користувачів. У випадку з Hotline варто відзначити інтуїтивно зрозумілий інтерфейс, відсутність агресивної реклами, а також мінімалістичний дизайн, зразок якого подано на рис. 1.1.

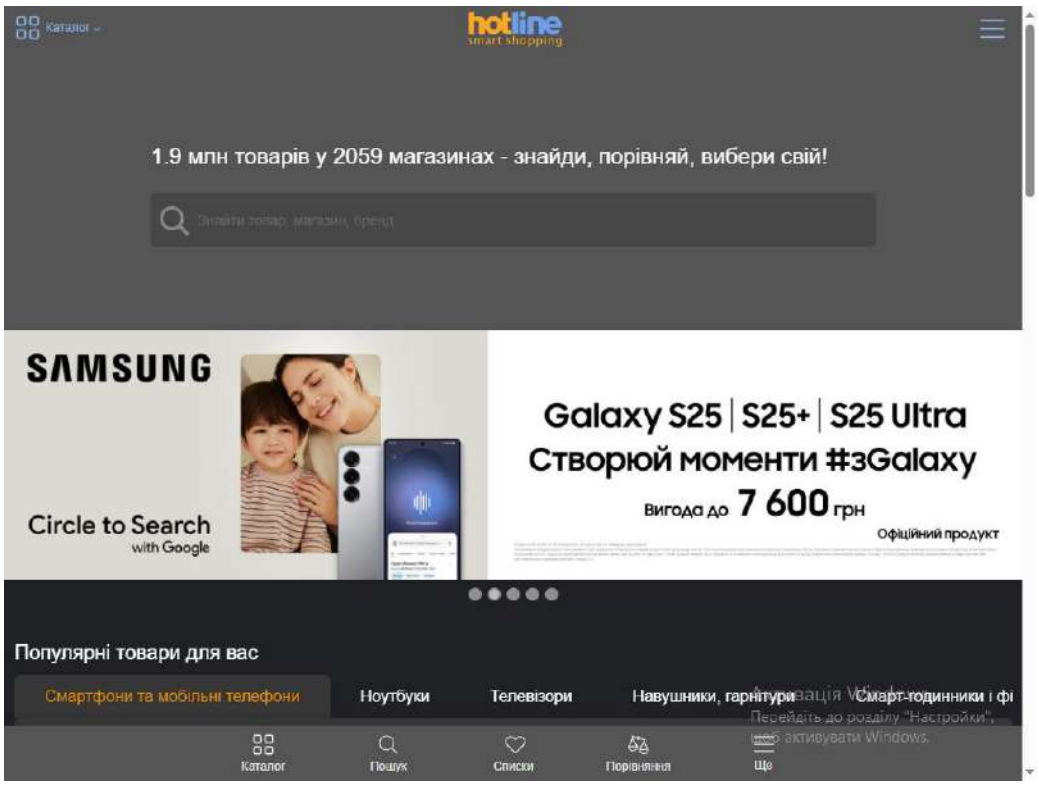


Рис 1.1. Інтерфейс головної сторінки сервісу Hotline

Примітка. Джерело: розроблено з використанням [5]

На веб-сайті реалізовано ряд додаткових функцій, які покликані підвищити зручність користування сервісом. Серед них варто відзначити списки побажань, таблиці для порівняння товарів, а також систему акаунтів, яка дозволяє користувачам зберігати обрані позиції та відстежувати динаміку цін.

Інтерфейс результатів пошуку має класичну структуру, зручною для користувача. Серед корисних функціональних особливостей можна виділити можливість перегляду альтернативних пропозицій за аналогічним запитом, а також відображення дати додавання товару до системи, що дає змогу оцінити актуальність інформації.

Зразок відображення результатів пошуку наведено на рис. 1.2.

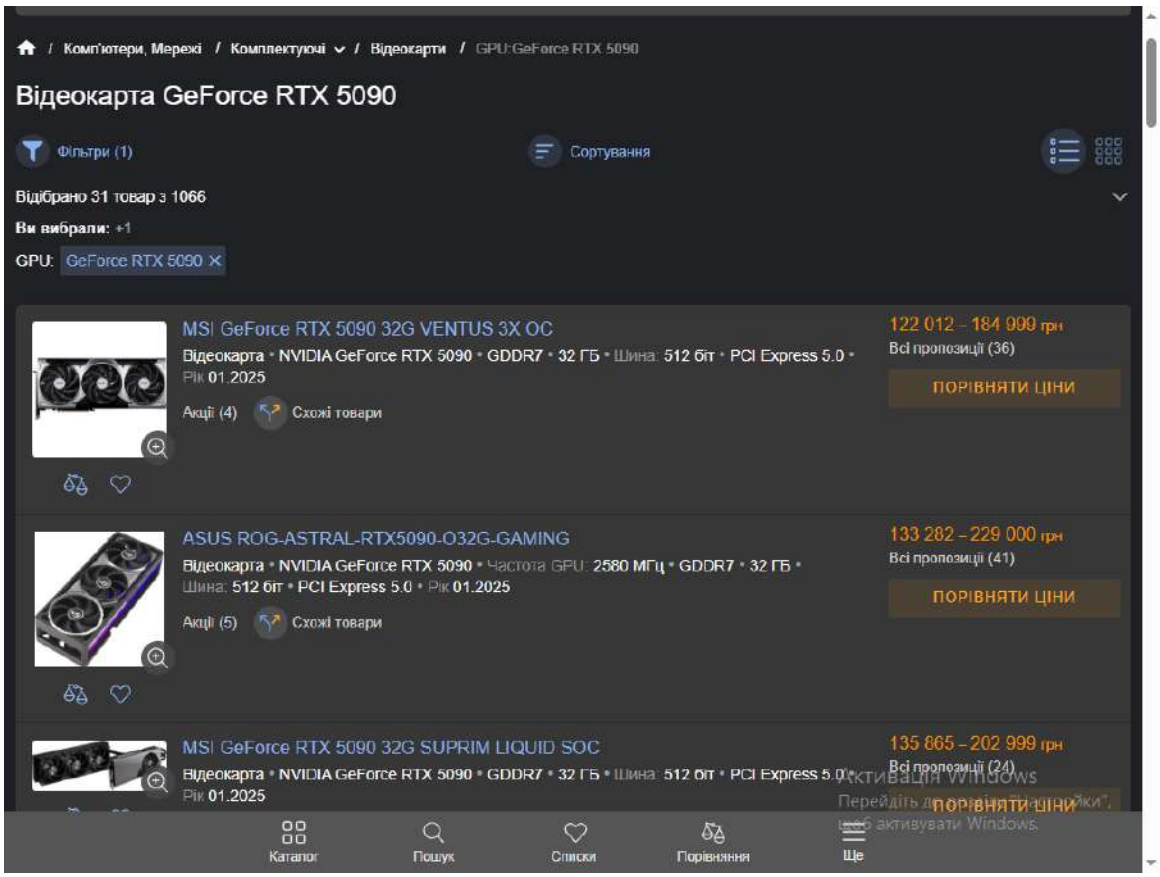


Рис 1.2. Скріншот сторінки з результатами пошуку на Hotline

Примітка. Джерело: розроблено з використанням [5]

Крім основного функціоналу пошуку, на сайті реалізовано каталог товарів, який представлено на рис. 1.3. Він забезпечує зручну навігацію за категоріями та дозволяє користувачеві швидко знаходити потрібні позиції. Каталог охоплює широкий спектр товарів, включаючи як загальновживані категорії, так і спеціалізовані, зокрема засоби енергозабезпечення та товари військового призначення, що свідчить про актуальність та адаптивність сервісу до сучасних потреб користувачів.

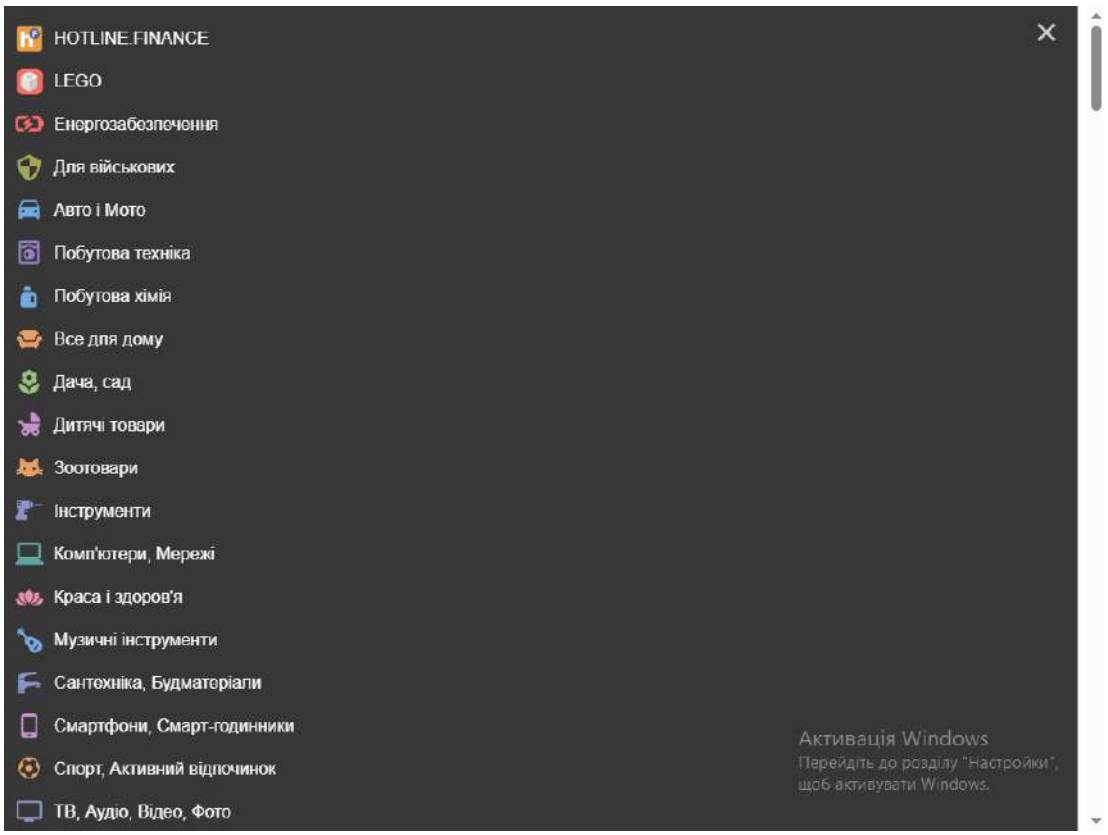


Рис. 1.3. Каталог товарів з сайту Hotline

Примітка. Джерело: розроблено з використанням [5]

Другим за популярністю серед українських онлайн-сервісів для порівняння цін є Price.ua. Це платформа, яка забезпечує пошук товарів і послуг у різних інтернет-магазинах із подальшим порівнянням їхніх цін та характеристик. Сервіс входить до складу міжнародного інтернет-холдингу Universal Commerce Group.

Price.ua функціонує за моделлю оплати за клік (PPC), отримуючи винагороду за кожен перехід користувача до інтернет-магазину, в якому представлений відповідний товар.

Окрім основного функціоналу пошуку, сайт містить каталог товарів, відгуки покупців та рейтинг продавців, який формується на основі користувацької активності. Такий підхід дозволяє потенційним покупцям оцінити надійність і репутацію торговельних майданчиків, що сприяє формуванню більш обґрунтованого вибору.

Головна сторінка ресурсу, представлена на рис. 1.4, хоча й має дещо застарілий візуальний стиль, все ж залишається функціонально повною та

зручною для навігації, не втрачаючи при цьому своєї інформативної привабливості.

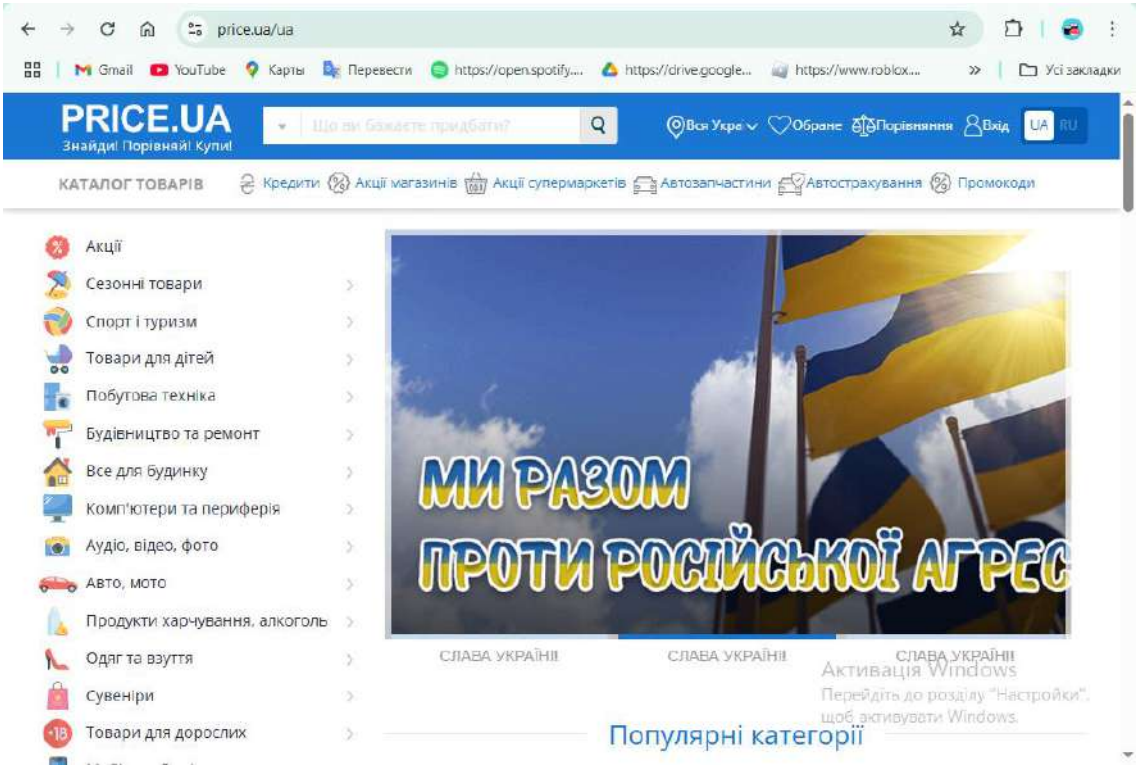


Рис. 1.4. Головна сторінка сайту Price.ua

Примітка. Джерело: розроблено з використанням [6]

Сайт вирізняється широким асортиментом категорій товарів, що забезпечує зручність у пошуку продукції різного призначення. Серед позитивних особливостей ресурсу варто також відзначити наявність посилання на донат для Збройних Сил України, що свідчить про соціальну відповідальність платформи.

Окремої уваги заслуговує функція, відсутня на більшості аналогічних сервісів, – можливість порівняння цін у супермаркетах. Це розширює сферу застосування платформи та робить її корисною не лише для придбання техніки чи побутових товарів, а й для щоденних покупок. Інтерфейс цієї функціональності наведено на рис. 1.5.

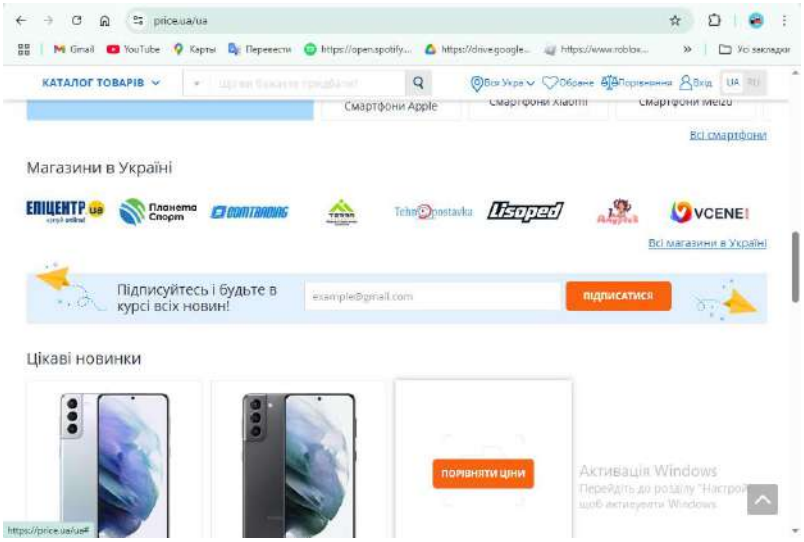


Рис. 1.5. Функція порівняння цін у фізичних магазинах, реалізована на Price.ua

Примітка. Джерело: розроблено з використанням [6]

Під час тестування було виявлено певне обмеження у функціональності пошуку на сайті: система не змогла знайти конкретний товар, введений у запит, причини чого залишаються незрозумілими. Крім того, результати, що були запропоновані, стосуються переважно офлайн-магазинів, які мають фізичні торговельні точки, тоді як інтернет-магазини без фізичної присутності в результатах не відображаються.

Це може свідчити про певні обмеження алгоритму пошуку або особливості співпраці ресурсу з окремими торговельними платформами. Відповідні результати пошуку наведено на рис. 1.6.

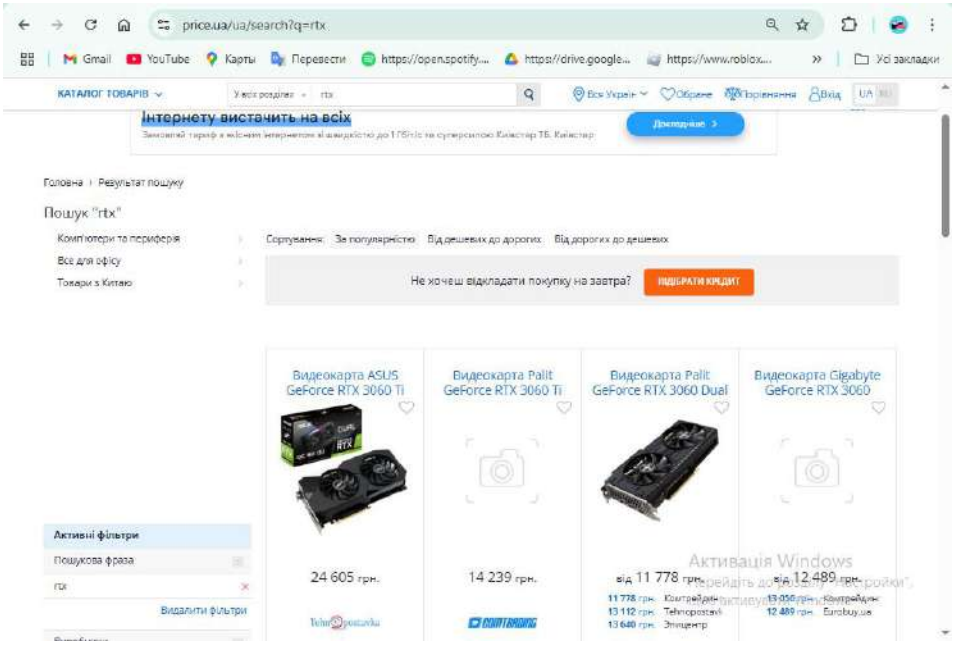


Рис. 1.6. Результати пошуку на сайті Price.ua

Примітка. Джерело: розроблено з використанням [6]

Загалом, сайт має достатньо розвинений функціонал і забезпечує широкий спектр можливостей для користувача. Водночас, у порівнянні з конкурентами, стає зрозуміло, чому цей ресурс посідає другу позицію за популярністю серед подібних сервісів в Україні.

Останнім із розглянутих рішень є Google Shopping, інтерфейс якого представлено на рис. 1.7.

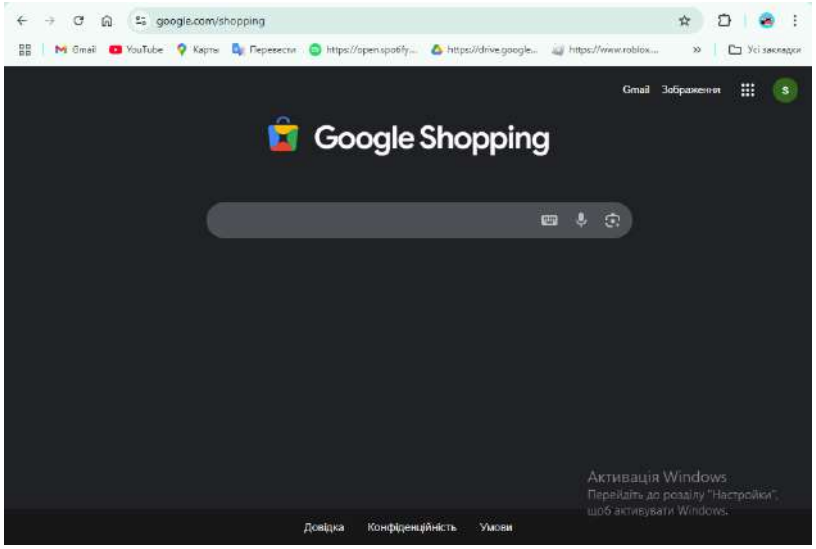


Рис. 1.7. Головна сторінка сервісу Google Shopping

Примітка. Джерело: розроблено з використанням [9]

Головна сторінка Google Shopping виконана у фірмовому стилі пошукової системи Google – вона має мінімалістичний та зручний дизайн, який забезпечує комфортну взаємодію та зосередженість користувача на основній функції сервісу – пошуку товарів. Інтерфейс містить лише необхідні елементи, що дозволяє швидко розпочати пошук без відволікання на другорядний функціонал.

Після введення запиту користувача система переадресовує на сторінку результатів, де кожна товарна позиція представлена у стислому вигляді у вигляді інформаційної картки, що містить фотографію товару, його назву, ціну, а також джерело (назву інтернет-магазину), з якого взято інформацію. Приклад відображення результатів пошуку наведено на рис. 1.8.

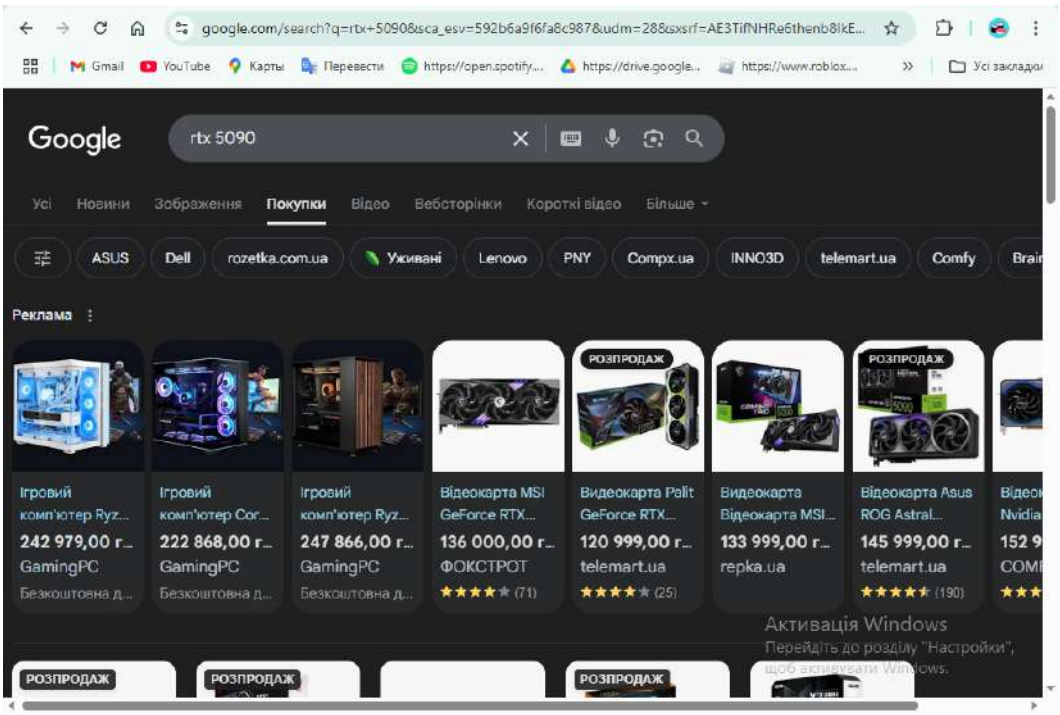


Рис. 1.8. Результати пошуку у сервісі Google shopping

Примітка. Джерело: розроблено з використанням [9]

Google Shopping надає користувачам основні інструменти фільтрації, серед яких – фільтри за магазинами, моделями, брендами та іншими індивідуальними характеристиками товарів. Проте, загальна функціональність сервісу є доволі

обмеженою: користувач отримує лише добірку товарних карток відповідно до заданих критеріїв, які містять базову інформацію (фото, назва, ціна, джерело). Водночас відсутні такі можливості, як відстеження цін, додавання товарів до списку улюблених, формування кошика або пряме порівняння обраних позицій.

На основі проведеного аналізу користувацької поведінки на подібних сервісах було сформульовано висновки, які стали підґрунтям для визначення функціональної концепції власного веб-додатку. Зокрема, було виявлено, що переважна більшість користувачів не використовує функції порівняння чи збереження товарів у списки улюбленого безпосередньо на сервісах-агрегаторах, віддаючи перевагу виконанню цих дій на сайтах конкретних інтернет-магазинів або не використовуючи їх узагалі. Це дозволяє дійти висновку, що виключення другорядного функціоналу (реєстрація, акаунти, улюблене, кошики тощо) сприятиме створенню більш простої, стабільної та зручної системи, орієнтованої на основну цінність – швидке порівняння товарних пропозицій.

Аналіз популярності функціональних елементів показав, що найбільш використовуваним фільтром є сортування за ціною. Це свідчить про те, що саме порівняння вартості є ключовою метою користувачів, і тому саме цей функціонал повинен стати центральною частиною майбутнього веб-сервісу.

Ще одним важливим критерієм, виявленим у процесі дослідження, є надійність продавців і прозорість їхньої взаємодії з покупцями. У зв’язку з цим було ухвалено рішення використовувати для порівняння лише перевірені інтернет-магазини, які виступають офіційними дистриб’юторами продукції. Наприклад, у випадку з платформою Rozetka, до аналізу включаються виключно ті товарні пропозиції, які розміщені безпосередньо самою Rozetka, без залучення сторонніх продавців або посередників.

У підсумку, одним із принципових рішень у розробці власного веб-додатку є відмова від системи акаунтів. Це дозволить знизити технічну складність реалізації, підвищити анонімність, забезпечити вищу стабільність роботи ресурсу, а також зробить використання сервісу інтуїтивно простим для користувачів з низьким рівнем цифрової грамотності. Проєкт орієнтований на

безкоштовне використання, що підвищує його доступність для широкої аудиторії.

1.3 Вибір інструментів розробки та обґрунтування технологічного стеку

Метою даного підрозділу є обґрунтування вибору оптимального технологічного стеку для реалізації веб-сервісу. Оскільки проєкт передбачає створення інтерактивної веб-сторінки, доцільним є використання базових веб-технологій, які забезпечують її структурне, стилістичне та функціональне наповнення: HTML, CSS та JavaScript.

HTML (*HyperText Markup Language*) є стандартизованою мовою розмітки, що використовується для створення структури веб-сторінок. Веб-браузери інтерпретують HTML-документи, отримані від сервера за протоколами HTTP/HTTPS або відкриті з локального сховища, та відображають їх у вигляді графічного інтерфейсу на екрані користувача.

CSS (*Cascading Style Sheets*, укр. каскадні таблиці стилів) – це мова стилів, яка дозволяє описувати зовнішній вигляд HTML-документів. Вона є однією з ключових технологій веб-розробки поряд із HTML і JavaScript. CSS широко використовується для візуальної презентації контенту, зокрема в аспектах кольору, розміщення, анімації та адаптивності інтерфейсу [4].

JavaScript – динамічна, об'єктно-орієнтована, прототипна мова програмування, яка реалізує стандарт ECMAScript. У контексті веб-розробки JavaScript забезпечує клієнтську логіку – інтерактивність, обробку подій, динамічну зміну вмісту сторінки та обмін даними з сервером у реальному часі (асинхронно). Мова підтримує декілька парадигм програмування, зокрема прототипну, імперативну й частково функціональну, а також характеризується динамічною та слабкою типізацією, автоматичним управлінням пам’яттю, використанням функцій як об’єктів першого класу [4].

Для реалізації серверної логіки обрано Node.js – платформу з відкритим програмним кодом, призначену для розробки високопродуктивних мережевих застосунків за допомогою мови JavaScript. Node.js дозволяє виконувати

JavaScript-код на сервері, що розширює сферу використання цієї мови за межі браузера. Платформа має активну спільноту, підтримує масштабування, забезпечує високу швидкодію та є сумісною з великою кількістю сучасних бібліотек і фреймворків [4].

Серед можливих альтернатив розглядалися такі технології, як PHP, Flask та Deno. Проте, порівняльний аналіз показав перевагу Node.js завдяки його вищій продуктивності, сучасності, зручності у використанні та відповідності набутим знанням, отриманим під час навчального курсу. Платформа Deno, хоча й вважається більш безпечною та функціонально розширеною, поступилася вибору через обмежену кількість практичних навичок і доступних навчальних ресурсів у рамках виконання цієї роботи.

Таким чином, вибраний технологічний стек (HTML, CSS, JavaScript, Node.js) забезпечує ефективну реалізацію як клієнтської, так і серверної частини веб-додатку відповідно до поставлених функціональних та технологічних вимог.

Одним із ключових аспектів у розробці веб-додатку є маршрутизація (англ. *routing*) – процес визначення шляху передавання інформації між вузлами мережі. У загальному технічному розумінні маршрутизатор (англ. *router*) приймає рішення щодо напрямку передавання пакетів даних на основі IP-адреси отримувача. Всі пристрої, що беруть участь у передаванні інформації, використовують ці адреси для визначення маршруту. Для прийняття відповідних рішень маршрутизатор повинен мати інформацію про доступні мережеві шляхи та віддалені вузли [4].

У контексті веб-розробки маршрутизація виконує функцію управління запитами користувача, забезпечуючи переходи між сторінками, обробку динамічних URL-адрес, доступ до API та інші функції. Вона є критично важливою як для клієнтської, так і для серверної частини додатку, дозволяючи підтримувати структуровану взаємодію між користувачем і системою.

Для реалізації серверної маршрутизації в рамках даного проєкту було обрано веб-фреймворк Express.js, який є одним із найбільш поширених рішень для Node.js. Express – це мінімалістичний та гнучкий фреймворк для створення

веб-застосунків і API, що поширюється на умовах ліцензії MIT. Його перевагами є висока сумісність з Node.js, зручність у використанні, активна спільнота підтримки та сучасність технологічної бази.

Крім маршрутизації, важливу роль у реалізації веб-додатку відіграє система шаблонізації. У даній роботі для генерації динамічних HTML-сторінок застосовано Embedded JavaScript (EJS) – мову шаблонів, яка дозволяє вставляти JavaScript-код безпосередньо в HTML-розмітку. EJS поєднує в собі логіку програмування та візуальну структуру, що робить її ефективним інструментом для інтеграції даних, отриманих із зовнішніх джерел.

Методи отримання даних: API та веб-скрейпінг

Оскільки основним завданням веб-додатку є збір та обробка даних з інтернет-магазинів, було розглянуто два можливих підходи: використання API та веб-скрейпінгу.

API (*Application Programming Interface*) – це набір інтерфейсів, протоколів і інструментів, які дозволяють взаємодіяти між різними програмними компонентами. У контексті веб-розробки API забезпечує стандартизований доступ до функціоналу сервісів третіх сторін, дозволяючи інтегрувати їхні дані без прямої взаємодії з інтерфейсом користувача [4]. Проте, у межах виконання дипломної роботи використання API виявилося недоцільним через наступні чинники:

- не всі інтернет-магазини надають відкриті API для отримання товарних даних;
- більшість API-рішень є платними, що суперечить бюджетним обмеженням освітнього проєкту.

У зв’язку з цим було прийнято рішення застосовувати веб-скрейпінг (*web scraping*) – технологію вилучення структурованих даних із веб-сторінок. Скрейпінг передбачає завантаження веб-сторінки, її подальший аналіз і вилучення необхідної інформації. Процес зазвичай автоматизується за допомогою програмного забезпечення, яке емулює поведінку користувача або взаємодіє з веб-сервером безпосередньо. Скрейпінг дає змогу ефективно збирати

дані з HTML-структури веб-сторінок навіть у разі відсутності офіційних інтерфейсів API [4].

У результаті проведеного аналізу, для реалізації веб-додатку було обрано наступний технологічний стек:

- Node.js – основна серверна платформа для обробки запитів і побудови логіки взаємодії;
- Express.js – маршрутизатор та серверний фреймворк для управління HTTP-запитами;
- EJS – шаблонізатор для створення динамічних HTML-сторінок;
- HTML – мова розмітки для побудови структури веб-сторінки;
- CSS / SCSS – стилізація інтерфейсу згідно з принципами адаптивного дизайну;
- JavaScript – мова програмування для реалізації інтерактивного функціоналу на стороні клієнта;
- Puppeteer (не згадувався в цьому фрагменті, але актуальний) – інструмент для автоматизації браузера й реалізації веб-скрейпінгу.

Обрані інструменти забезпечують повний цикл розробки веб-застосунку – від збору даних до їх візуалізації у браузері користувача. Рішення щодо застосування веб-скрейпінгу обґрунтоване практичними обмеженнями (відсутність API, економія ресурсів) та дозволяє автоматизувати отримання актуальної інформації з кількох джерел у реальному часі, що є ключовою функціональністю розробленого сервісу.

Висновки до розділу 1

У першому розділі було розглянуто теоретичні засади розробки веб-додатків, орієнтованих на вирішення задачі порівняння товарів в умовах стрімкого розвитку електронної комерції. Проведений аналіз сучасного стану галузі дозволив виявити низку проблем, що уповільнюють ефективність споживчого вибору, серед яких: фрагментованість інформації про товари,

відсутність уніфікованих стандартів представлення даних, а також складність здійснення оперативного порівняння пропозицій з різних джерел.

Оцінка функціональних можливостей наявних онлайн-сервісів для порівняння товарів дала змогу виявити їхні переваги й обмеження, що, у свою чергу, стало основою для формування вимог до розроблюваного програмного рішення та визначення можливих напрямів його вдосконалення.

У межах обґрунтування вибору технологій було встановлено доцільність використання платформи Node.js для реалізації серверної логіки, фреймворку Express для маршрутизації HTTP-запитів, шаблонізатора EJS для створення динамічних HTML-сторінок, а також бібліотеки Puppeteer як інструменту для автоматизованого збору даних з веб-ресурсів. Обраний технологічний стек забезпечує ефективну, масштабовану та економічно доцільну реалізацію веб-додатку без необхідності інтеграції з базами даних.

Таким чином, результати першого розділу сформували науково-методичне підґрунтя для подальшого проєктування архітектури системи та безпосередньої реалізації веб-додатку для порівняння товарів з інтернет-магазинів.

РОЗДІЛ 2
КОНЦЕПТУАЛЬНЕ ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ ПОРІВНЯННЯ ТОВАРІВ З ІНТЕРНЕТ-МАГАЗИНІВ

2.1 Архітектурна модель веб-додатку для збору та порівняння даних

Архітектура програмного забезпечення (англ. *software architecture*) – це концептуальна модель структурування програмної або обчислювальної системи, що передбачає абстрагування її складових на певному етапі життєвого циклу. Залежно від призначення та складності, система може включати кілька рівнів абстракції та функціонувати в різних режимах, для кожного з яких можлива окрема архітектурна схема [4].

У науково-технічній літературі неодноразово підкреслюється, що проєктування архітектури програмного забезпечення є не менш складним, а іноді й більш творчим та структурованим процесом, ніж архітектура фізичних об’єктів, таких як будівлі [3].

Архітектура веб-додатку, призначеного для порівняння товарних пропозицій з інтернет-магазинів, базується на принципах модульності та функціонального розділення компонентів. Основна ідея полягає у створенні чіткої структурної моделі, яка забезпечить прозорість, узгодженість та ефективність реалізації. Архітектурне проєктування спрямоване на впорядкування компонентів системи з метою досягнення високого рівня масштабованості, зручності супроводу та можливості подальшого розширення функціоналу.

У межах проєкту передбачено реалізацію багаторівневої архітектури, що включає такі ключові компоненти:

- Клієнтська частина (FrontEnd) – інтерфейс користувача (UI), який забезпечує взаємодію з системою;
- Серверна частина (BackEnd) – логіка обробки запитів, модулі порівняння та взаємодії з інтерфейсом;

- Модулі збору та оновлення даних – компоненти, відповідальні за вилучення актуальної інформації з веб-ресурсів;
- Шаблонізована система виводу результатів – механізми відображення даних у зручному та структурованому форматі.

Запропонована архітектурна модель відповідає принципам сучасного програмного проєктування та забезпечує:

- масштабованість – можливість додавання нових функцій без порушення існуючої структури;
- гнучкість – адаптацію під зміни зовнішніх умов або вимог;
- зручність супроводу – спрощення процесів тестування, оновлення й технічної підтримки.

В рамках розробки веб-додатку для порівняння товарів з інтернет-магазинів було сформовано логічну послідовність його роботи, яка ґрунтується на аналізі існуючих рішень та адаптована відповідно до вимог функціональності. Послідовність взаємодії системи має такий вигляд:

1. Користувач вводить назву товару у веб-інтерфейсі.
2. Серверна частина отримує запит через маршрутизатор.
3. Активується механізм парсингу, який ініціює збір даних з відповідних онлайн-магазинів.
4. Зібрана інформація структурується у форматі JSON-об’єктів.
5. Дані передаються на шаблонізатор, який формує сторінку з порівняльною таблицею.
6. Згенерована HTML-сторінка повертається користувачу як відповідь на запит.

Сформована логіка відображає мінімалістичний та зручний підхід до реалізації ключової функції – автоматизованого порівняння товарних пропозицій без потреби у додаткових діях з боку користувача (реєстрація, авторизація тощо).

Користувацький інтерфейс слугує точкою взаємодії між користувачем та функціоналом веб-додатку. Аналіз аналогічних сервісів та зворотного зв’язку з користувачами засвідчив, що основним очікуванням є швидкий доступ до

порівняльної таблиці з цінами та назвами сайтів без зайвих візуальних чи функціональних елементів. Саме тому було прийнято рішення реалізувати інтерфейс у максимально простій та інтуїтивно зрозумілій формі, забезпечивши лише візуальне представлення результатів запиту.

Інтерфейс буде реалізовано з використанням шаблонізатора EJS (Embedded JavaScript), що забезпечує динамічну генерацію HTML-сторінок на основі даних, отриманих із серверної частини. Завдяки поєднанню EJS та CSS буде створено адаптивний шаблон, здатний гнучко відображати інформацію про будь-який товар.

Серверна логіка відповідатиме за обробку запитів, взаємодію з модулями збору даних та формування відповіді у вигляді готової веб-сторінки. Для реалізації цієї частини застосовуються Node.js та Express.js — сучасні, масштабовані інструменти, що забезпечують ефективну маршрутизацію запитів, обробку даних та інтеграцію з шаблонізатором.

Цільовими критеріями при розробці серверної частини є:

- стабільність обробки запитів;
- гнучкість масштабування функціоналу;
- мінімальні затримки при формуванні відповіді;
- логічна ізоляція модулів.

Збір інформації з інтернет-магазинів реалізується за допомогою веб-скрейпінгу, що дозволяє вилучати структуровані дані без наявності API у сторонніх сервісів. Обраний підхід базується на автоматизованому аналізі HTML-коду сторінок за допомогою спеціальних бібліотек (зокрема, Puppeteer), що дозволяє імітувати дії користувача в браузері та отримувати необхідну інформацію (назву товару, ціну, URL джерела).

Семантично, процес веб-скрейпінгу передбачає побудову дерева розбору сторінки (DOM) з подальшим витягом елементів за заданими шаблонами. Такий підхід аналогічний побудові абстрактних синтаксичних дерев (AST) у компіляторах, що дозволяє точно інтерпретувати структуру сторінки та автоматизувати вилучення даних [1].

Серед переваг використання веб-скрейпінгу у межах даної роботи:

- відсутність потреби у підключенні до API сторонніх ресурсів;
- відсутність витрат на платні підключення;
- універсальність збору даних з будь-якого сайту, незалежно від політики доступу до їхніх API.

Парсинг (веб-скрейпінг) – це процес автоматизованого збору та аналізу інформації з веб-сайтів за допомогою спеціалізованого програмного забезпечення. Сутність цього процесу полягає в поетапному вилученні даних зі сторінки: програмний агент (бот) відкриває веб-ресурс, аналізує структуру HTML-коду, ідентифікує необхідні елементи та зберігає їх у структурованому вигляді для подальшої обробки.

Варто зазначити, що аналогічні принципи використовуються і великими пошуковими системами (наприклад, Google), де автоматизовані програми (веб-краулери) індексують вміст сайтів. Саме тому обмеження доступу для скрейперів часто призводить і до небажаних обмежень для пошукових систем, що ускладнює ефективний захист сайту від несанкціонованого збору даних.

У публічному дискурсі веб-скрейпінг іноді асоціюється з негативним використанням, однак важливо наголосити, що сам по собі парсинг не є протиправним явищем. Йдеться про обробку інформації, яка розміщена у відкритому доступі, а програмні засоби лише автоматизують і пришвидшують процес, який користувач теоретично міг би виконати вручну. За умови етичного та технічно обґрунтованого використання веб-скрейпінг може виступати потужним інструментом для аналізу цифрового контенту.

Для реалізації парсингу в межах цього проєкту було обрано бібліотеку Puppeteer – сучасний інструмент для керування браузером Google Chrome або Chromium через API. Puppeteer дає змогу імітувати поведінку реального користувача, включаючи прокрутку сторінки, кліки, затримки між діями, що робить процес скрейпінгу менш помітним для захисних систем сайтів. Хоча бібліотека потребує більше ресурсів у порівнянні з текстовими парсерами, її перевагами є:

- підтримка динамічного контенту (включаючи JavaScript-генеровані елементи);
- гнучке налаштування сценаріїв взаємодії з веб-сторінками;
- можливість обробки сайтів зі складною структурою.

2.2 Схема роботи додатка та прототип користувацького інтерфейсу

Схема – спрощене зображення в загальних рисах системи, будови чого-небудь або взаєморозташування, зв'язку частин чогось. Іншими словами, схема роботи веб-додатка – розклад його на частини, в порядку їх взаємодії між собою [4].

Провівши аналіз роботи вибраних нами технологій, була створена схема роботи веб-додатку, яка має оптимальну працездатність та швидкість, при цьому є максимально масштабованою.

1. Запит користувача. Користувач переходить на сайт, перше що він бачить, це пошукову сторінку а також кнопку «пошук». Інтерфейс інтуїтивно зрозумілий, приємний для зору і що зараз дуже актуально, мінімалістичний. Користувач вводить назву свого товару в стрічку для пошуку і тисне на кнопку поруч.

2. Обробка на сервері. Для маршрутизації веб-додаток використовуватиме Express.js, який буде приймати запит даних від клієнта, такий як пошук даних на сторінках інтернет-магазинів.

3. Збір даних. За допомогою модулю Puppeteer, веб-додаток збиратиме інформацію методом парсингу. Тобто, відкриватиме сайти інтернет-магазинів, виконуватиме там пошук товару та витягуватиме дані з HTML структури сайту, такі як назва товару, ціна, його зображення та назву сайту, а також залишатиме посилання на картку цього товару з оригінального сайту.

4. Обробка даних. Отримані дані нормалізуються, тобто переводяться у однаковий формат даних та чекають подальших маніпуляцій з собою.

5. Формування відповіді. За допомогою Ejs-шаблонізатора сервер

підставлятиме дані, зібрані до цього та створюватиме HTML-сторінку для відображення заповнених шаблонів. Також, під час цього процесу деякі дані, такі як ціна будуть згруповані та зібрані для використання у таблиці, яка потім також буде відображена.

6. Результат. Оброблені дані подаються у вигляді сторінки, на якій користувач бачить результат свого запиту у вигляді деякого списку карток, які мають в собі всю інформацію, включаючи ціни, посилання та фото. Також, на сторінці буде зображена порівняльна таблиця, для більш легкого зорового сприйняття інформації.

Зображення схеми роботи веб-додатку за допомогою діаграми можна побачити на рис 2.1.

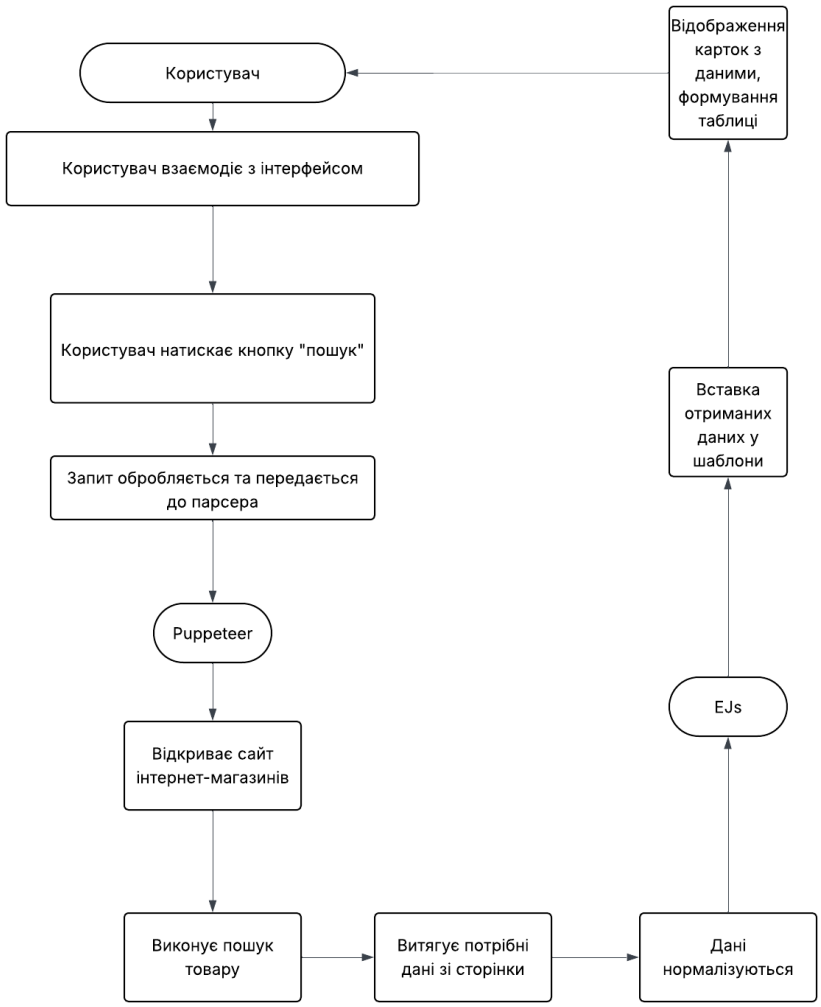


Рис. 2.1. Схема роботи веб-додатку для порівняння даних

Примітка. Джерело: розроблено автором

Ця схема показує повний цикл – від запиту до відповіді, з урахуванням серверної логіки, парсингу, і відображення на клієнті.

Інтерфейс користувача, (англ. user interface, UI) – засіб зручної взаємодії користувача (людини) з інформаційною системою. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача; у графічних системах інтерфейс користувача, втілюється багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їх розташуванням, сортуванням елементів вікон, гнучкими налагодженнями як самих вікон, так і окремих їх елементів (файли, теки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань. Все що користувач бачить, перейшовши на сайт – це і є користувацьким інтерфейсом.

Основними принципами прийнятного UI є простота, інтуїтивність і адаптивність. Для нашого додатку, є доцільним використання мінімальної кількості відображених елементів для зменшення смислового навантаження та збільшення інтуїтивної зрозумілості та призначення веб-додатку. Планується реалізація пошукової стрічки, для вводу користувачем даних для пошуку, а також великої контрастної кнопки для виконання пошуку даних.

Для розробки було проаналізовано найпопулярніші сервіси для пошуку даних та порівняння цін, серед них можна виділити три найкращих претенденти, таких як Hotline, Price.ua та Magazilla. Користувацький інтерфейс зображено на рис 2.2.

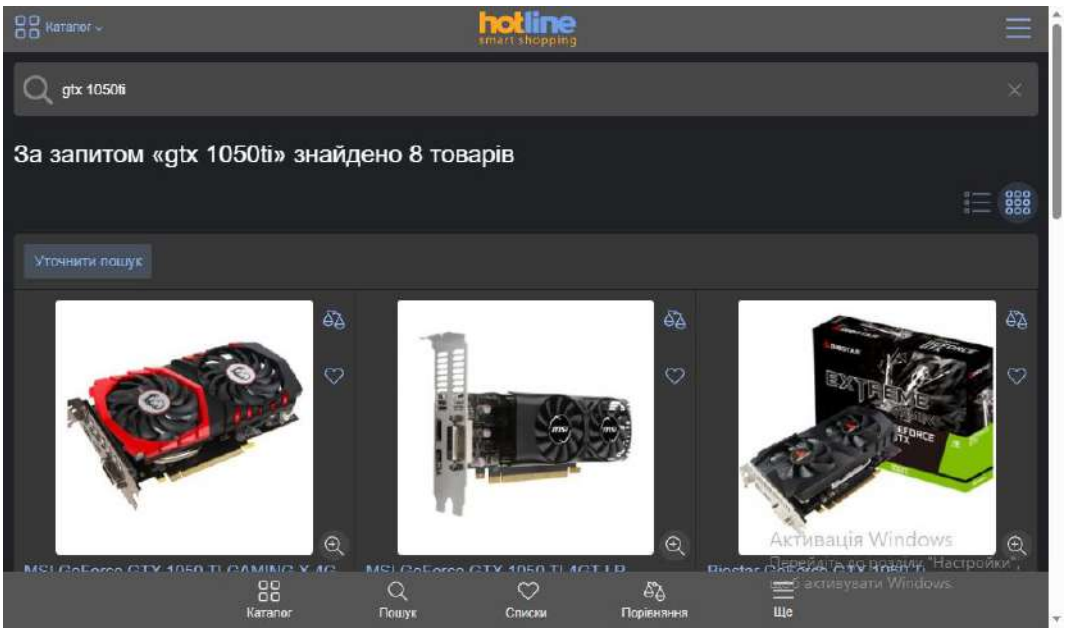


Рис. 2.2. Приклад користувацького інтерфейсу з Hotline

Примітка. Джерело: розроблено з використанням [5]

Інтерфейс цього інтернет-магазину є інтуїтивно зрозумілим, адаптивним, але має багато елементів, які звертають на себе свою увагу, таких як таргетна реклама товарів, у яких з сервісом підписано контракт, можна побачити на рис. 2.3. Вона забирає на себе усю увагу, та об’єктивно відводить її від основного функціоналу сайту.

Рис. 2.3. Таргетна реклама з Hotline

Примітка. Джерело: розроблено з використанням [5]

Price.ua – непоганий сервіс, але в ньому також є деякі недоліки. Інтерфейс сайту зображено на рис. 2.4.

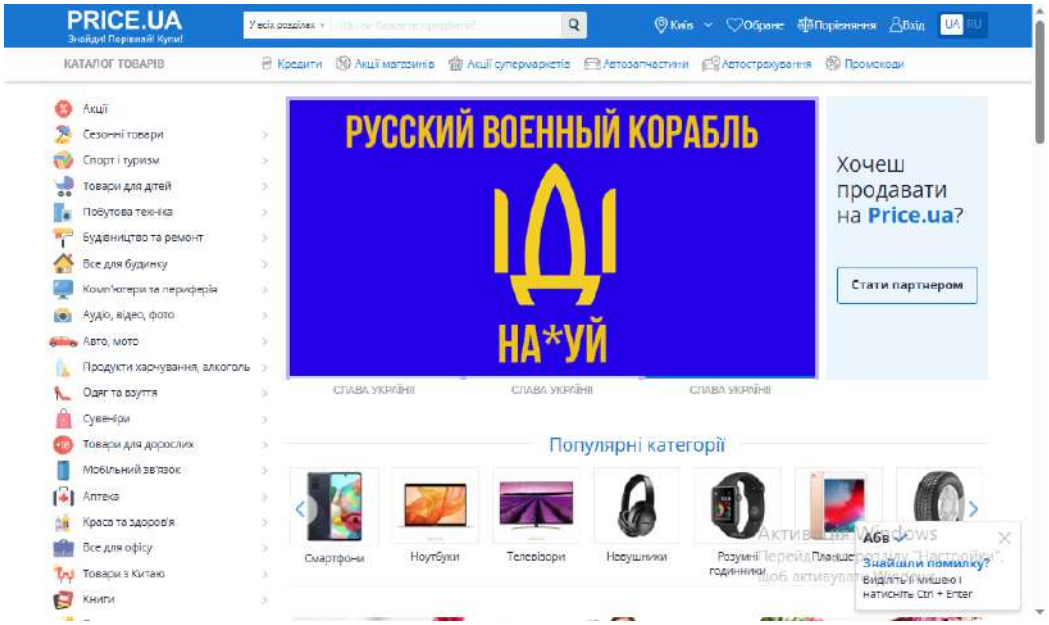


Рис. 2.4. Інтерфейс сайту Price.ua

Примітка. Джерело: розроблено з використанням [6]

Перше що привернуло мою увагу – інтерфейс цього сайту дуже схожий на інтерфейс сайту для електронного поштування – Mail.ru. Він зроблений в схожих тонах, ключові кнопки розміщені в тому ж місці, що й на тому сайті. Через це, інтерфейс виглядає старомодно та занадто нагромаджено, меню товарів зліва можна було б приховати за спливаючим контекстним меню або взагалі переміщенням на іншу сторінку, спеціально з категоріями. Після формування запиту, сервіс видав нам результати пошуку, зображені на рис. 2.5.

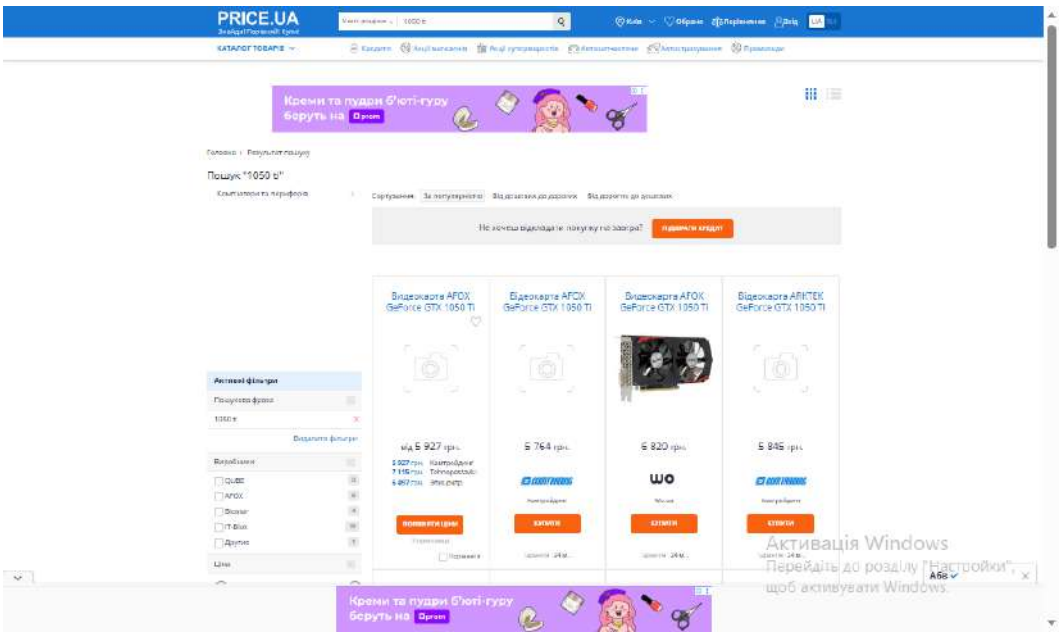


Рис. 2.5. Результати пошуку з Price.ua

Примітка. Джерело: розроблено з використанням [6]

З невідомої нам причини, тут немає популярних сайтів інтернет-магазинів. Скоріш за все, сайт збирає інформацію за допомогою API-запитів, завдяки чому користувачеві подається доступ до найбільш актуальної інформації, але тільки з тих магазинів, які погодилися подати свій API цьому сервісу. Скоріш за все оплата для інтернет-магазинів здійснюється за рахунок партнерської програми на основі парного піару – Price.ua подає інформацію щодо товарів, які шукає користувач тільки з тих сайтів, з якими має угоду, вони в свою чергу мають з цього прибуток, через те що створюється ілюзія того, що більше вибору немає, тому користувач повинен вибирати між чимось з того, що йому було представлено. Як на мене – це не дуже корректно та етично. За допомогою парсингу реалізувати роботу легше, а також він не є незаконною діяльністю, тому його можна використовувати в своїх цілях. Парсинг надає більший асортимент товару а також більш гнучкий до збільшення магазинів, з якими можна проводити маніпуляції.

Також не менш важливою складовою методу збору даних, використаного на сервісі Price.ua, є ціна його використання. Лише деякі сайти готові співпрацювати з такими сервісами, тим паче за умовну безкоштовність.

Переважна більшість з них буде стягувати плату за використання їх API даних, що може негативно відобразитися на доцільності їх використання. Розміщаючи свою рекламу на сайті, Price.ua збирає гроші, які потім витрачає на оплату хостингу та API-запитів, тому такий сервіс не буде приносити прибуток.

Останнній з виділених нами сервіс – Magazilla. Інтерфейс зображено на рис. 2.6.

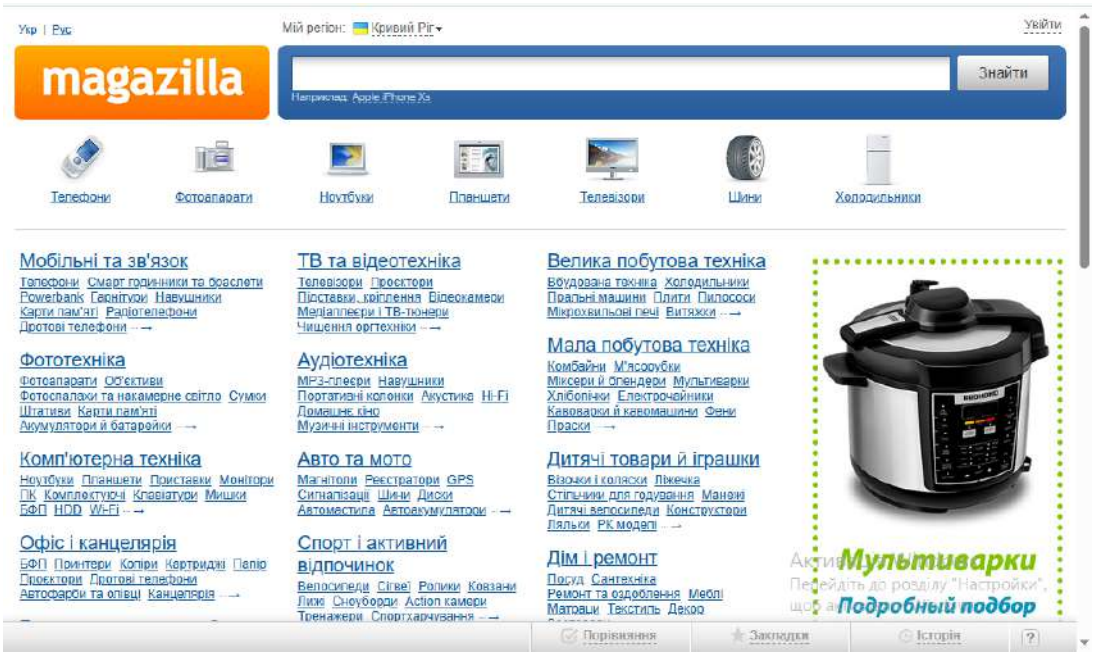


Рис. 2.6. Користувацький інтерфейс сайту Magazilla

Примітка. Джерело: розроблено з використанням [6]

Інтерфейс цього сайту є найбільш старим, через те що сайт давно не отримував оновлень або актуалізацій. Знову ж таки, на головній сторінці дуже багато нагромадженого тексту, який можна було сховати за контекстними меню, також дизайн іконок виглядає застаріло, через довгу відсутність оновлень. На нашу думку сайт закинуто, через що його фактично неможливо оцінювати на одному рівні з іншими.

Скріншот сторінки з результатами пошуку зображено на рис. 2.7.

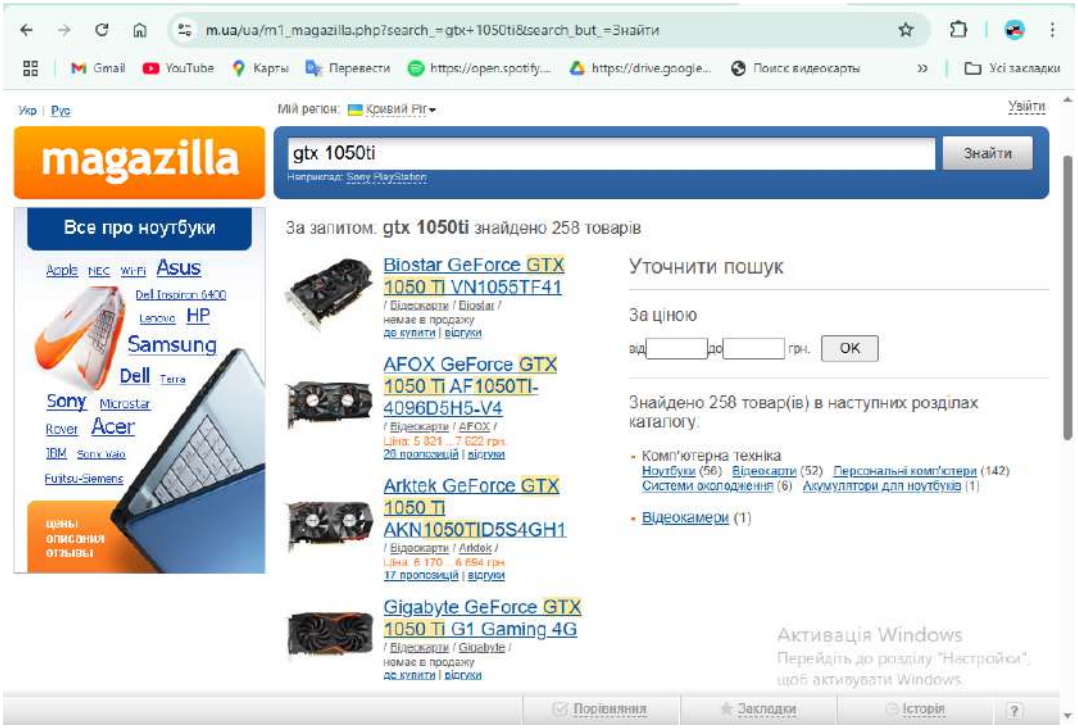


Рис. 2.7. Результат пошуку на сайті Magazilla

Примітка. Джерело: розроблено з використанням [6]

Сторінка з результатами виглядає неякісно, елементи розміщені так, що це «ріже око», тобто вони не залежать від один одного, ніяк не відділені, здається ніби вони накладаються один на одного. На сайті підібрано автоматично фільтр по виробникам, що може спричинити дискомфорт деяким користувачам, тому що цей фільтр є абсолютним і його неможливо вимкнути.

Проаналізувавши існуючі рішення проблеми, можна побачити ряд деяких проблем, кожна з яких має свою значимість і так чи інакше впливає на загальне враження від використання сервісу. Кожен з сервісів мав би мати унікальний інтерфейс, який легко запам’ятати, але є й такі, які запозичують ідеї або дизайнерські рішення у інших. Для веб-додатку дуже важливо мати свій, унікальний зовнішній вигляд, тому що від нього залежить те, наскільки ваш сервіс буде користуватися попитом та буде взнаваний серед користувачів. Також, не останнім по значущості йде показник контрастності, тому що це базова складова інтерфейсу. Кнопки не мають зливатися з іншими елементами, вони мають якось виділятися на їх тлі. Також весь текст на сайті має бути читабельним,

без синтаксичних помилок а також зрозумілий для користувача.

Висновки до розділу 2

У другому розділі було проведено проєктування веб-додатку для порівняння товарів з інтернет-магазинів, що є важливим етапом у процесі впровадження. Розглянуто архітектурну модель системи, яка базується на взаємодії клієнт-сервер і не потребує збереження даних у базі даних, що підвищує продуктивність та зменшує складність реалізації.

Запропоновано схему роботи веб-додатку, яка охоплює основні етапи: відправлення запиту користувача, парсинг цін з інтернет-магазинів за допомогою Puppeteer, обробка зібраних даних на сервері за допомогою Node.js та Express, а також генерація динамічної сторінки за допомогою EJS для відображення результатів.

Окрім технічного обґрунтування, розроблено прототип інтерфейсу користувача, який враховує зручність навігації, доступність інформації та швидкість взаємодії з додатком. Такий підхід забезпечує не тільки технічну ефективність, але й орієнтацію на потреби кінцевого користувача.

Таким чином, результати цього розділу заклали чітку структурну основу для реалізації веб-додатку, адаптованого до завдань електронної комерції.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ ПОРІВНЯННЯ ТОВАРІВ

3.1 Розробка інтерфейсу користувача

Для початку, так як я вибрав основною технологією для серверної частини – Node.js, то почати хотілося б з створення нового проекту. Щоб створити його, потрібно спочатку скачати Node.js з офіційного сайту, та встановити його за інструкцією звідти ж.

Після цього, створити папку будь де, та бажано дати їй ім’я на англійській. Наступний крок – відкрити консоль саме у цій директорії. Щоб довго не писати шлях до неї, просто натисніть на шлях, перебуваючи у папці, в яку хочете встановити її і впишіть там команду cmd (рис. 3.1).

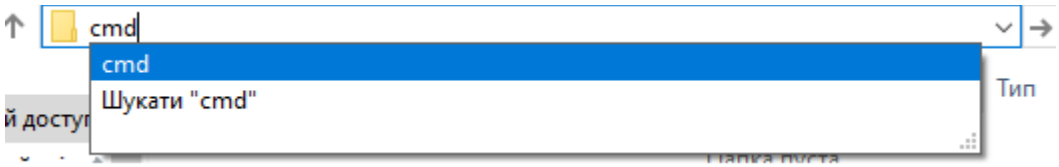


Рис. 3.1. Вписана команда у шлях директорії

Примітка. Джерело: розроблено автором

Після цього, вписуємо у консоль команди:

mkdir node-parser

cd node-parser

npm init -y

Ця команда створить файл package.json в якому будуть знаходитися всі залежності та налаштування проекту.

Чекаємо поки пройде обробка, і вписуємо ще одну команду туди ж:

npm install express ejs puppeteer

Після цього, в нашій директорії будуть створені файли, зображені на рис.

3.2, які будуть основою для нашої роботи.

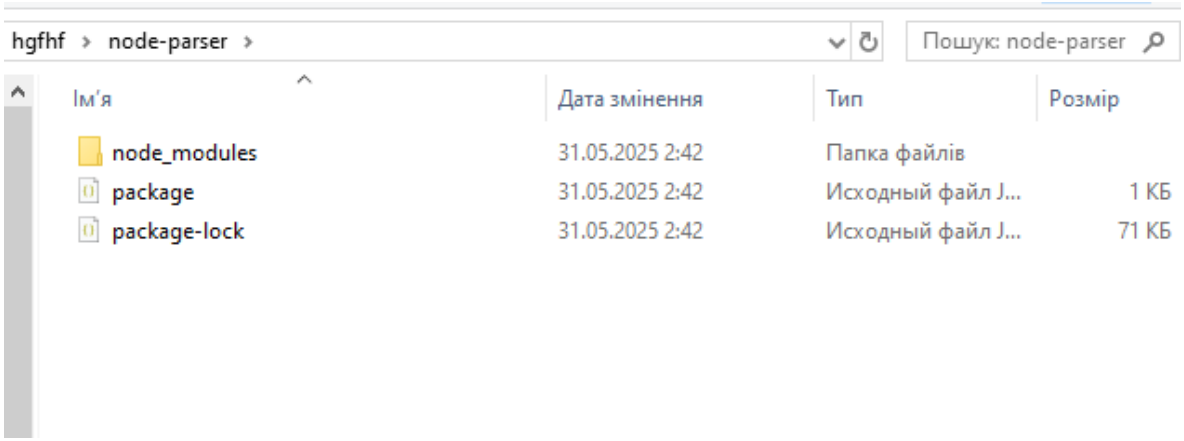


Рис. 3.2. Файли, створені після налаштування проєкту

Примітка. Джерело: розроблено автором

Другий набір команд потрібен для завантаження потрібних модулів вібраних нами інструментів, а саму Express.js, Ejs та puppeteer.

Після всіх маніпуляцій, було створено структуру файлів і папок яка потрібна нам для зручності. На рис. 3.3 можна побачити поточний вигляд нашої директорії.

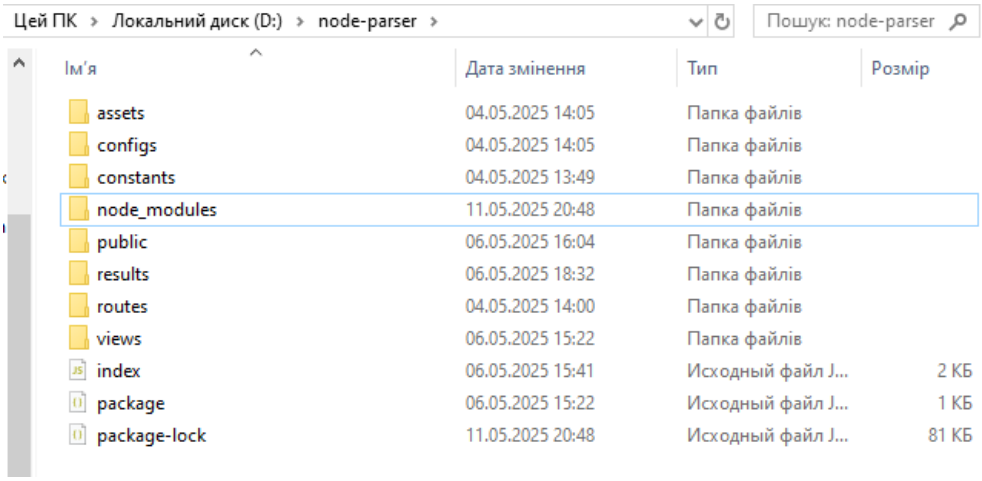


Рис. 3.3. Система файлів директорії з проєктом

Примітка. Джерело: розроблено автором

Давайте більш детально розглянемо що це за файли та папки.

Файл index.js – Головний серверний файл. Запускає Express, підключає маршрути, налаштовує шаблонізатор EJS.

Наступний кореневий файл проекту – package.json. Містить залежності проєкту, назву, скрипти для запуску.

Папка assets – має в собі дані, у форматі JSON, в яких записано список товарів для підказок під час пошуку товару.

public – папка в якій знаходятся потрібні під час розробки іконки сайтів, документ у форматі .css який відповідає за стилізацію сторінки. Також тут знаходится Sorce map для SCC. Основний код JavaScript для фронтенду та взаємодії з DOM, допоміжні функції для фронтенду а також деякі додаткові конфігурації.

Папка routes – створена для маршрутів Express.js, також містить в собі маршрут для запуску парсеру, файли для приймання запиту та передачі їх до парсеру, а також деякі файли для обробки головної сторінки.

І остання папка – папка views – створена для шаблонів Ejs. Містить в собі головну сторінку з формою пошуку. Також містить файли EJS які використовуються для рендерингу динамічного контенту.

Для початку створимо стартовий файл. В ньому прописуємо пару статичних виразів, які використовуються для того, щоб задати серверу параметри для взаємодії з іншими частинами коду а також команду для старту. Саме цей файл буде початком для нашої роботи, завдяки якому всі директорії і файли будуть пов’язані між собою. Код основного файлу роботи зображений на рис. 3.4.

Рис. 3.4. Код файлу index.js з нашої основної директорії проєкту

Примітка. Джерело: розроблено автором

Жодний сервіс чи веб-додаток для порівняння даних не обійдеться без своєї сторінки, завдяки якій користувач буде взаємодіяти з додатком. Проєкт буде реалізовано з використанням однієї сторінки, яка і буде головною. Код головної сторінки закріплено на рис. 3.5.

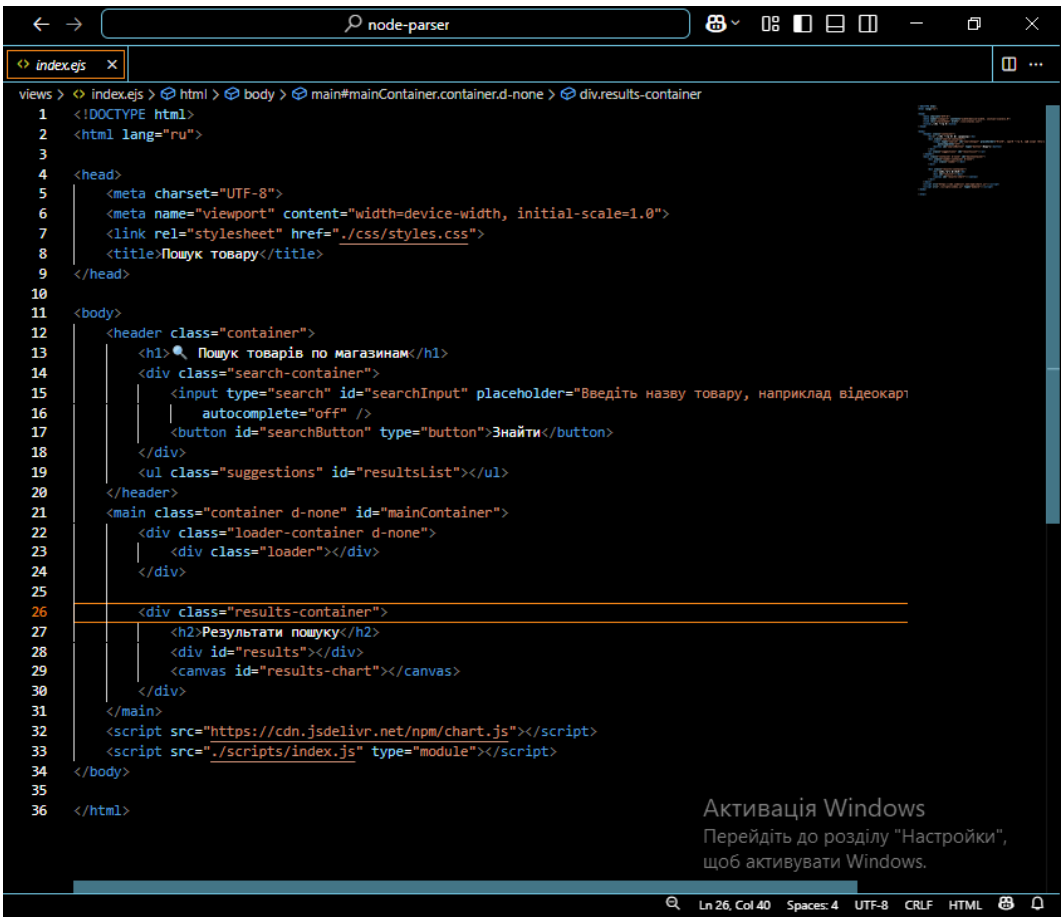
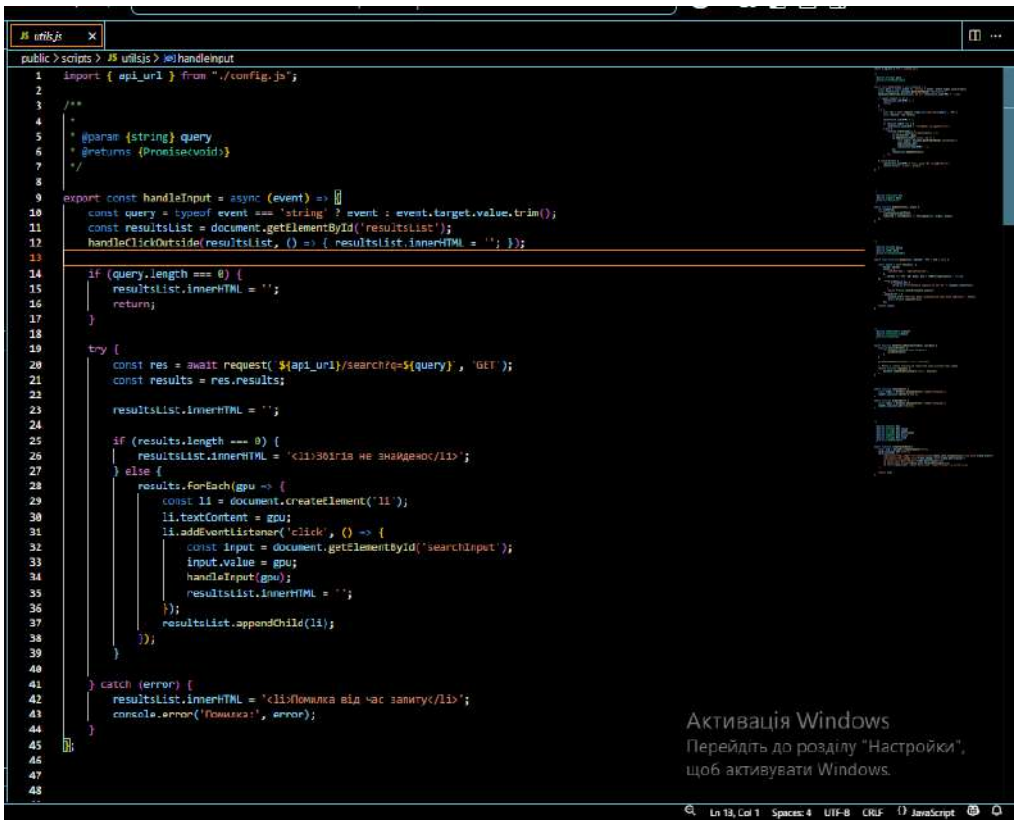


Рис. 3.5. Текст файлу index.ejs

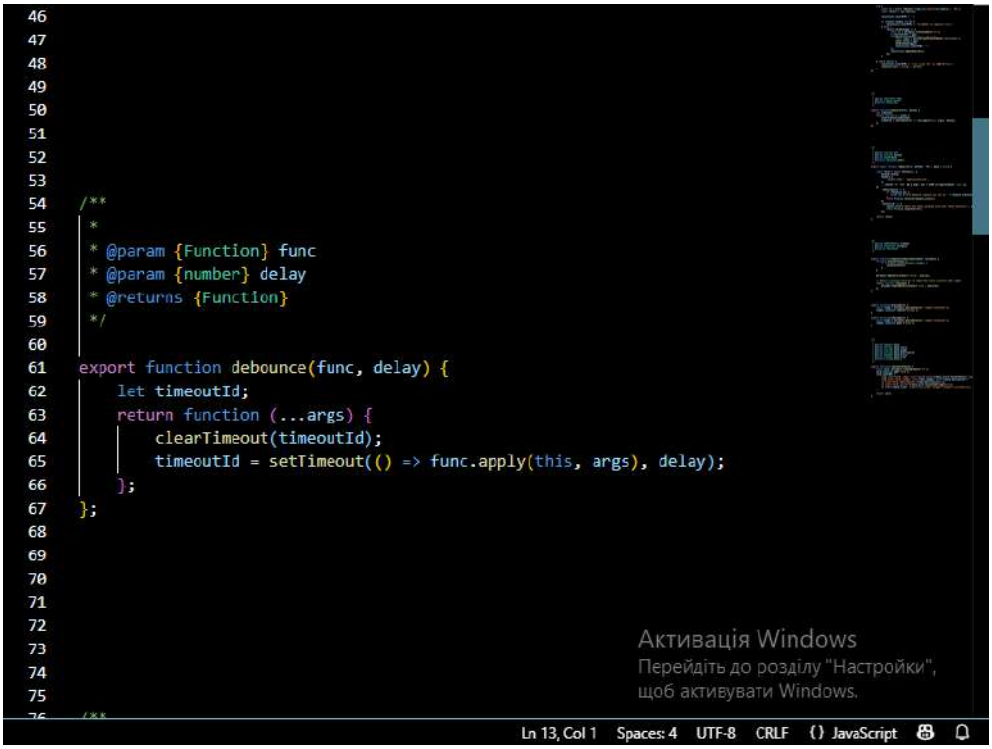
Примітка. Джерело: розроблено автором

У файлі проглядається структура сторінки, на якій реалізовано перший контейнер, який включає в себе пошукову стрічку та кнопку пошуку, а також другий контейнер, який включає в себе результати пошуку.

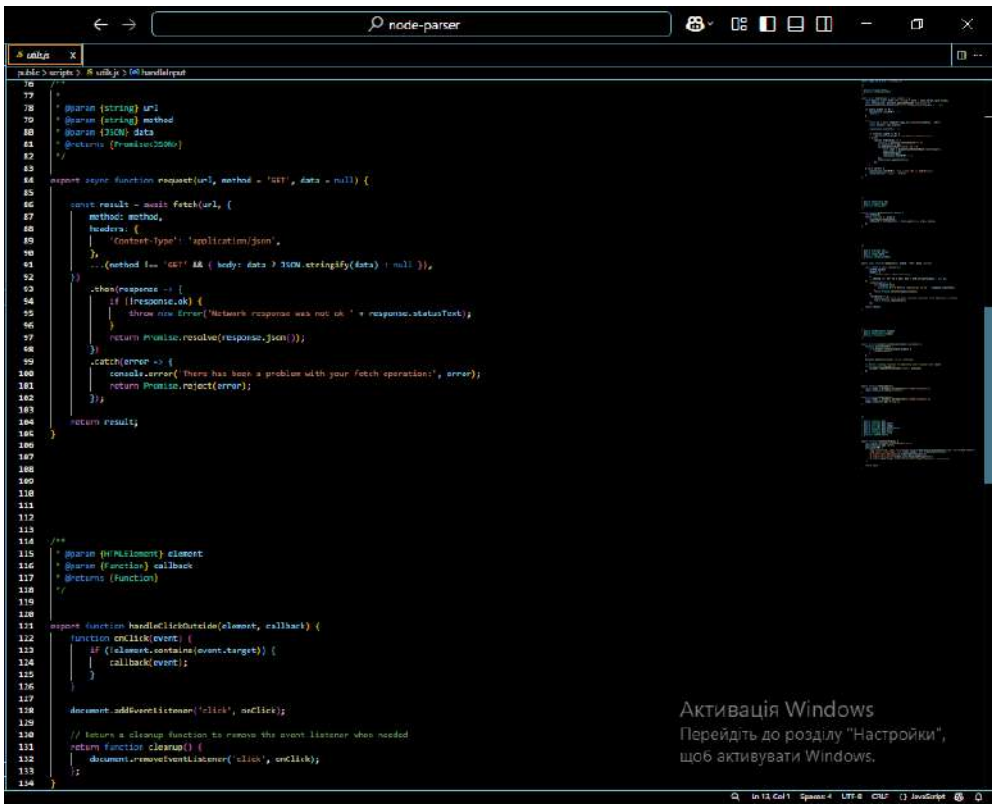
Для реалізації деяких додаткових функцій, було створено цей файл, зображений на рис. 3.6.


utils.js
public > scripts > utils.js > handleInput
import { api_url } from "./config.js";
/**
* @param {string} query
* @returns {Promise<void>}
*/
export const handleInput = async (event) => {
const query = typeof event === 'string' ? event : event.target.value.trim();
const resultsList = document.getElementById('resultsList');
handleClickOutside(resultsList, () => { resultsList.innerHTML = ''; });
if (query.length === 0) {
resultsList.innerHTML = '';
return;
}
try {
const res = await request(`${api_url}/search?q=${query}`, 'GET');
const results = res.results;
resultsList.innerHTML = '';
if (results.length === 0) {
resultsList.innerHTML = '
Збігів не знайдено
';
} else {
results.forEach(gpu => {
const li = document.createElement('li');
li.textContent = gpu;
li.addEventListener('click', () => {
const input = document.getElementById('searchInput');
input.value = gpu;
handleInput(gpu);
resultsList.innerHTML = '';
});
resultsList.appendChild(li);
});
}
} catch (error) {
resultsList.innerHTML = '
Помилка від час запиту
';
console.error('Помилка:', error);
}
};
Активація Windows
Перейдіть до розділу "Настройки",
щоб активувати Windows.
Ln 13, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript

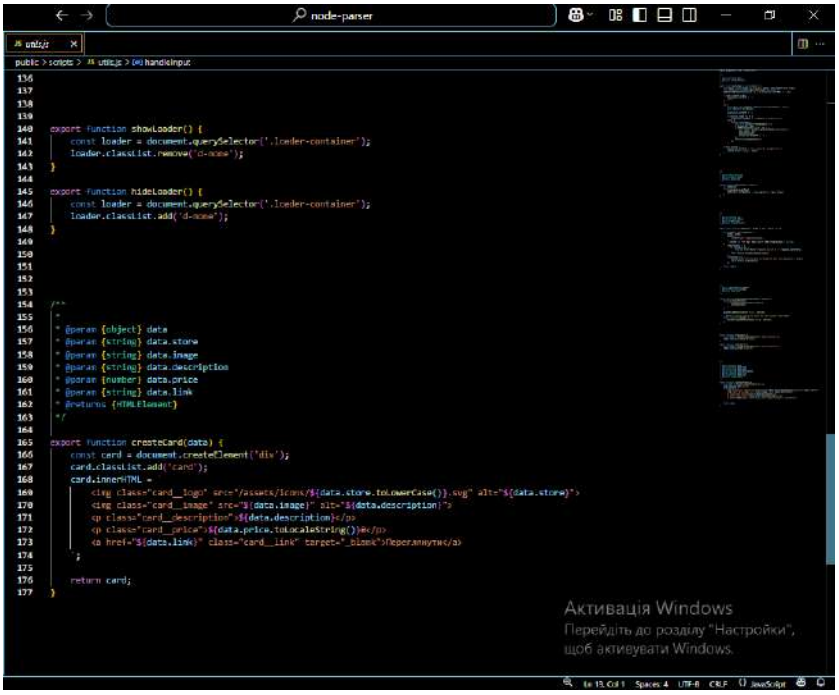
(a)


/**
* @param {Function} func
* @param {number} delay
* @returns {Function}
*/
export function debounce(func, delay) {
let timeoutId;
return function (...args) {
clearTimeout(timeoutId);
timeoutId = setTimeout(() => func.apply(this, args), delay);
};
};
Активація Windows
Перейдіть до розділу "Настройки",
щоб активувати Windows.
Ln 13, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript

(b)



(c)



(d)

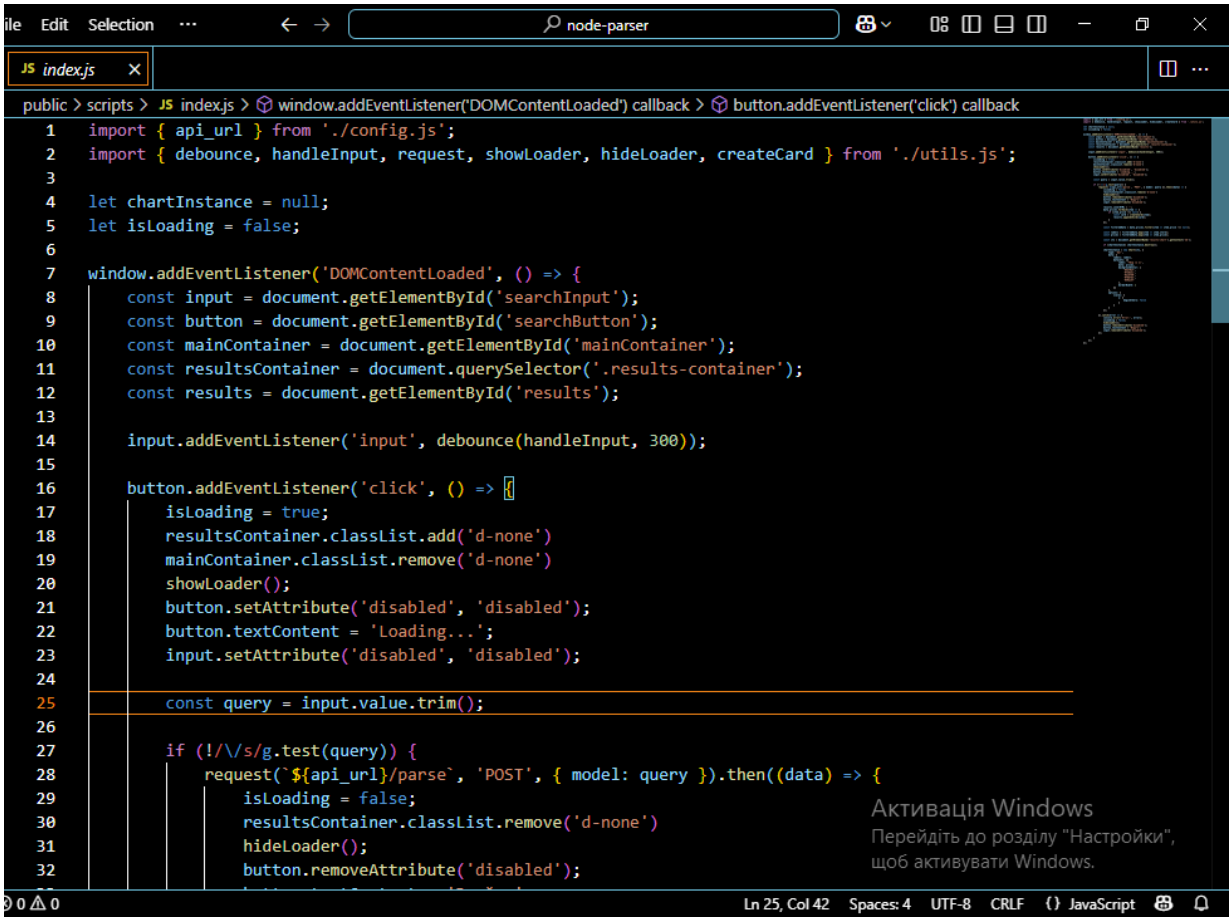
Рис. 3.6. Зміст файлу (a-d) utils.js

Примітка. Джерело: розроблено автором

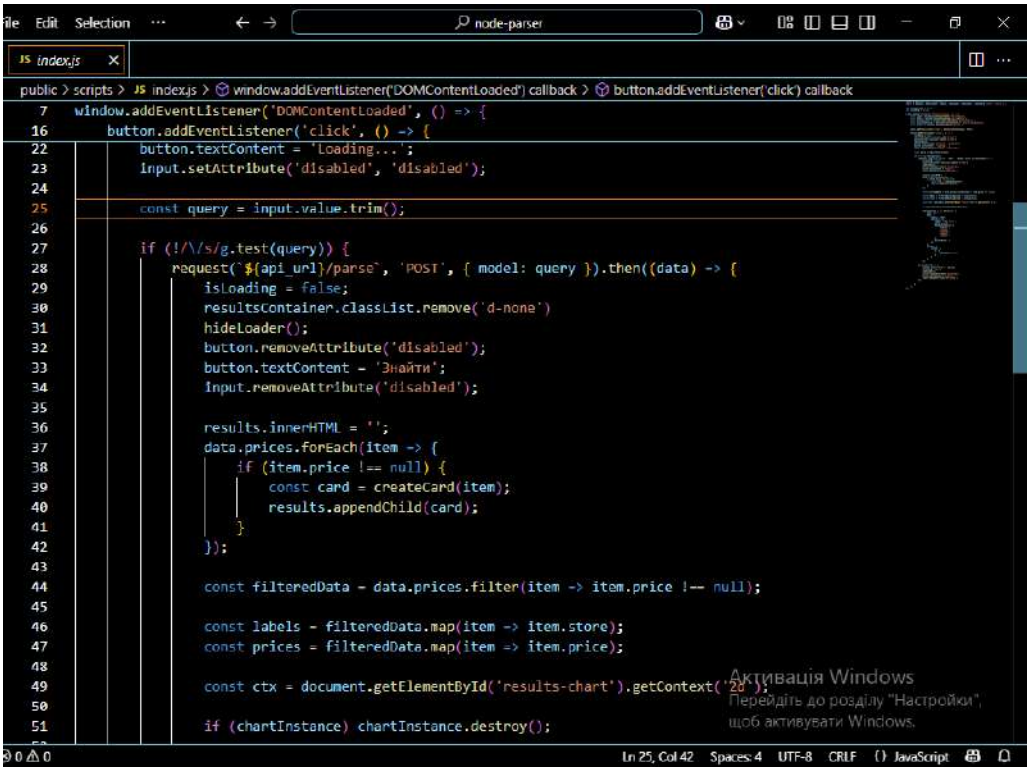
Для цього додатку, було реалізовано деякі функцій, такі як:

- меню підказок для пошуку деяких товарів, з обраного списку;
- вікно помилки, якщо у користувача щось не так зі з'єднанням до нашого сайту, а також вікно помилки, якщо щось не так з сайтом інтернет-магазину;
- зіставлення результатів парсингу з клієнтською частиною, для взаємодії з ними;
- анімація завантаження результатів парсингу.

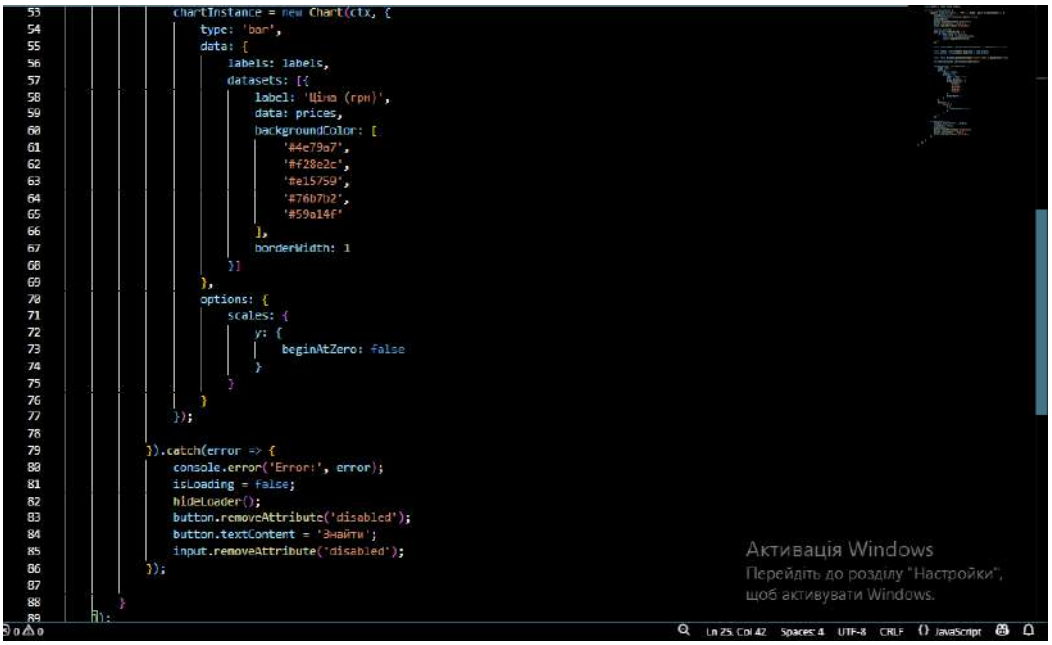
А тепер реалізуємо файл з основною взаємодією JavaScript-у з HTML-структурою нашого додатку. Структуру файлу зображено на рис. 3.7.



(a)



(b)



(c)

Рис. 3.7. Зміст файлу (a-c) index.js з директорії scripts

Примітка. Джерело: розроблено автором

Цей файл реалізує все, те, що було у файлі utils.js а також додатково опрацьовує дані та поміщає їх у таблицю.

Під час розробки інтерфейсу користувача було реалізовано зручну, інтуїтивно зрозумілу та адаптивну структуру взаємодії з веб-додатком. Основна увага була зосереджена на легкості пошуку товарів, швидкому доступі до результатів порівняння та візуальній привабливості наданих даних.

Інтерфейс було побудовано з використанням шаблонізатора EJS, який дозволив формувати динамічні HTML-сторінки залежно від запиту користувача та результатів парсингу. Адаптивний дизайн, реалізований за допомогою SCSS, забезпечив коректне відображення контенту на різних пристроях.

Особлива увага була приділена уніфікації стилів та використанню візуальних елементів (іконок, кольорових маркерів) для розпізнавання магазинів, що покращує зручність навігації та швидкість сприйняття інформації. Користувач отримує результати у вигляді структурованих блоків з назвою товару, ціною, зображенням та прямим посиланням на сторінку магазину.

Таким чином, інтерфейс повністю відповідає вимогам сучасних веб-додатків: він зручний, функціональний та зрозумілий для широкого кола користувачів, сприяючи ефективному виконанню основної функції – порівняння цін з різних інтернет-магазинів.

3.2 Розробка механізму пошуку та порівняння даних з інтернет-магазинів

Тепер, щодо основного функціоналу, до нього входить пошук даних на веб-ресурсах, його запозичення та використання в цілях порівняння. Для цього було створено файл, в якому задано веб-сайти, з яких буде проводитися парсинг. Зміст цього файлу зображений на рис. 3.8.

Рис. 3.8. Зміст файлу index.js з директорії constants

Примітка. Джерело: розроблено автором

Таким чином, для збору даних було обрано топ-5 провідних інтернет-магазинів в Україні. Додатково створюємо файл, в якому вказуємо кінцевий шлях парсеру, куди доставляти запозичені дані. Скріншот файлу на рис. 3.13.

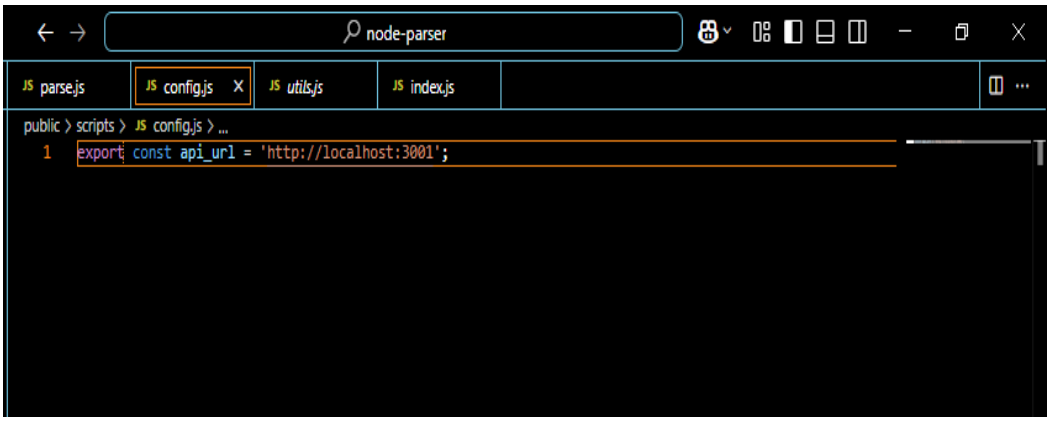


Рис. 3.9. Зміст файлу config.js

Примітка. Джерело: розроблено автором

Цей URL ми призначили для нашої головної сторінки, куди саме будуть експортуватися дані.

Наступним етапом розробки є маршрутизація. Навігації між сторінками немає, тому що веб-додаток реалізований на одній сторінці, без переходів, а ось маршрутизація запитів та передачі даних до парсеру, та й самого парсингу повинна бути. Вона є невід'ємною частиною будь якого додатку, тому що саме

завдяки ній, парсер буде розуміти звідки йому потрібно брати дані та які саме.

Для розгалудження системи створюю файл, в якому прописую пару сталих, які відповідають за файл з реалізацією маршрутизації пошуку, а також файл з маршрутизацією парсеру. Код цього файлу представлено на рис. 3.10.

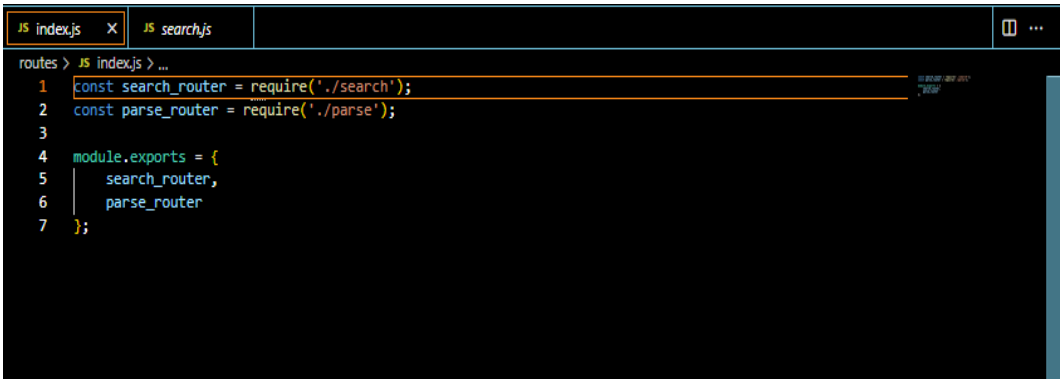
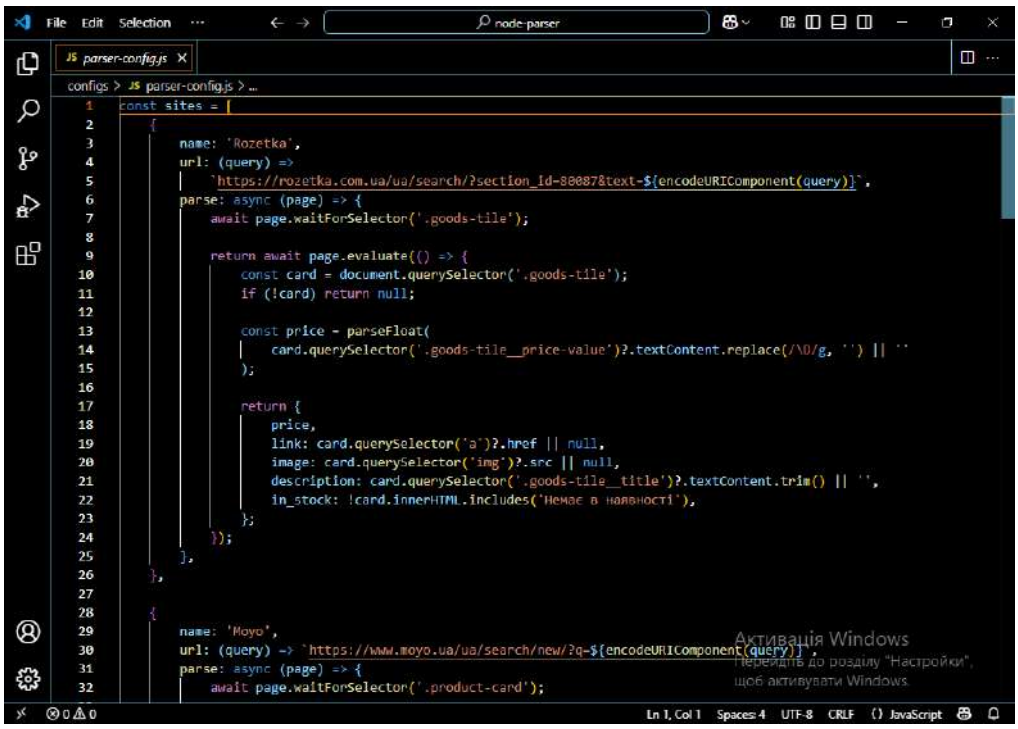


Рис. 3.10. Зміст файлу index.js з директорії routes

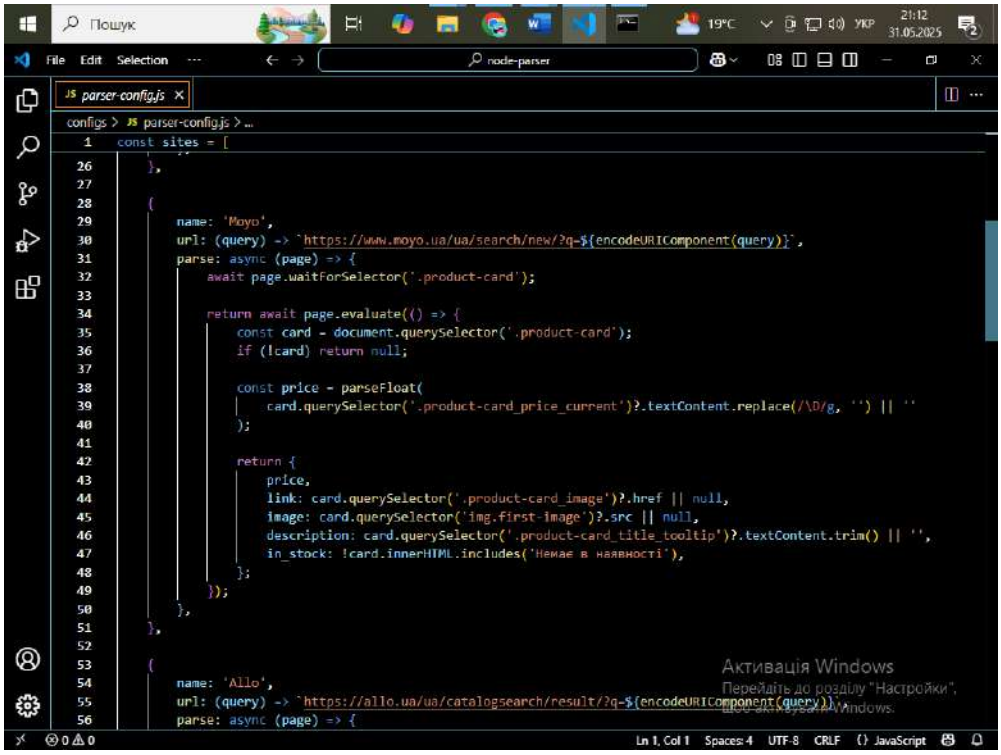
Примітка. Джерело: розроблено автором

Парсинг відбувається на основі css-селекторів, скрипт знаходить елементи у HTML-структурі сайту інтернет-магазину, які містять ціну, назву товару, його картинку, витягує з них атрибути. Після цього вони зберігаються в масив даних, після чого вони повертаються назад до серверу, обробляються і подаються до нього за допомогою Ejs. Ці взаємодії зображені у додатках Б і В.

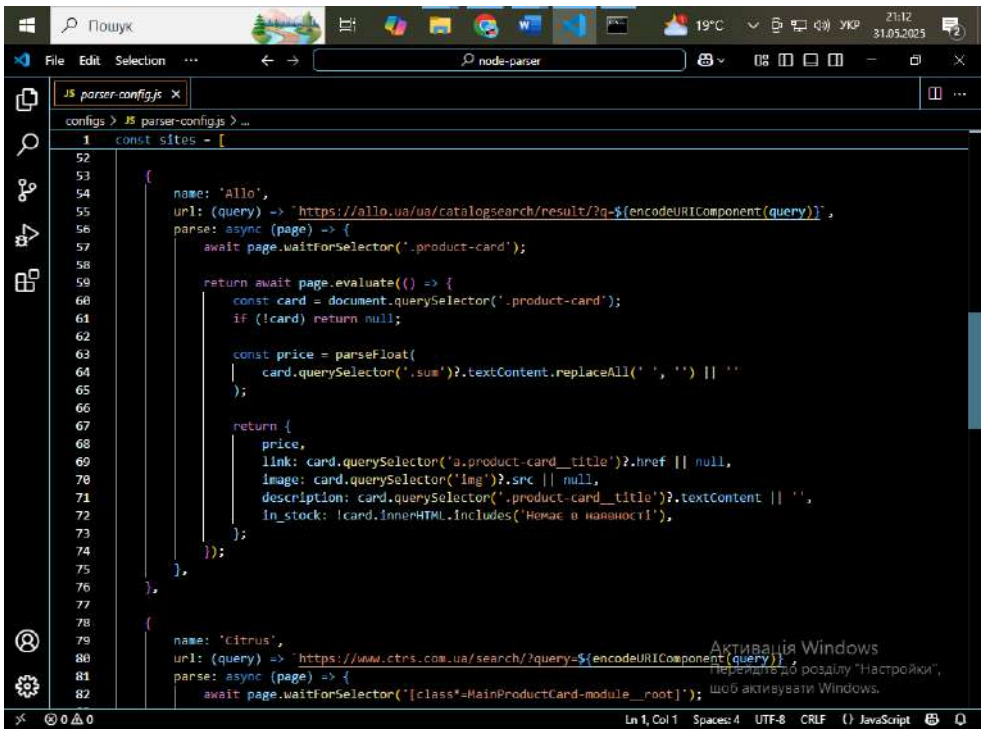
Також прикладаю код файлу, на рис. 3.11, з додатковими налаштуваннями парсеру, такими як URL сайтів для обробки, селектори, таймінги.


File Edit Selection
node-parser
parser-config.js
configs > parser-config.js > ...
const sites = [
{
name: 'Rozetka',
url: (query) =>
`https://rozetka.com.ua/ua/search/?section_id=80087&text=${encodeURIComponent(query)}`,
parse: async (page) => {
await page.waitForSelector('.goods-tile');
return await page.evaluate(() => {
const card = document.querySelector('.goods-tile');
if (!card) return null;
const price = parseFloat(
card.querySelector('.goods-tile__price-value')?.textContent.replace(/\D/g, '') || ''
);
return {
price,
link: card.querySelector('a')?.href || null,
image: card.querySelector('img')?.src || null,
description: card.querySelector('.goods-tile__title')?.textContent.trim() || '',
in_stock: !card.innerHTML.includes('Немає в наявності'),
};
});
},
},
{
name: 'Moyo',
url: (query) => `https://www.moyo.ua/ua/search/new/?q=${encodeURIComponent(query)}`,
parse: async (page) => {
await page.waitForSelector('.product-card');
Активація Windows
Перейдіть до розділу "Настройки",
щоб активувати Windows.
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript

(a)

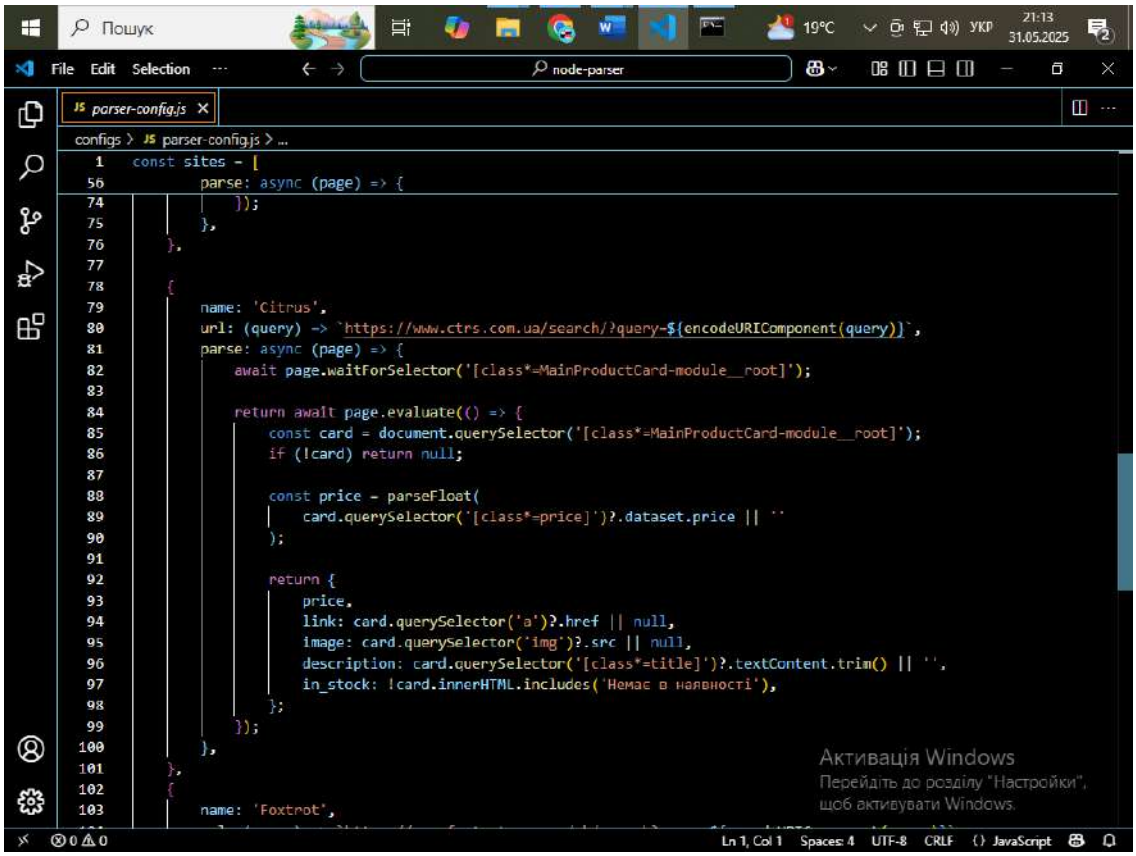

Пошук
19°C
УКР
21:12
31.05.2025
File Edit Selection
node-parser
parser-config.js
configs > parser-config.js > ...
const sites = [
name: 'Moyo',
url: (query) => `https://www.moyo.ua/ua/search/new/?q=${encodeURIComponent(query)}`,
parse: async (page) => {
await page.waitForSelector('.product-card');
return await page.evaluate(() => {
const card = document.querySelector('.product-card');
if (!card) return null;
const price = parseFloat(
card.querySelector('.product-card_price_current')?.textContent.replace(/\D/g, '') || ''
);
return {
price,
link: card.querySelector('.product-card_image')?.href || null,
image: card.querySelector('img.first-image')?.src || null,
description: card.querySelector('.product-card_title_tooltip')?.textContent.trim() || '',
in_stock: !card.innerHTML.includes('Немає в наявності'),
};
});
},
},
{
name: 'Allo',
url: (query) => `https://allo.ua/ua/catalogsearch/result/?q=${encodeURIComponent(query)}`,
parse: async (page) => {
Активація Windows
Перейдіть до розділу "Настройки",
щоб активувати Windows.
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript

(b)



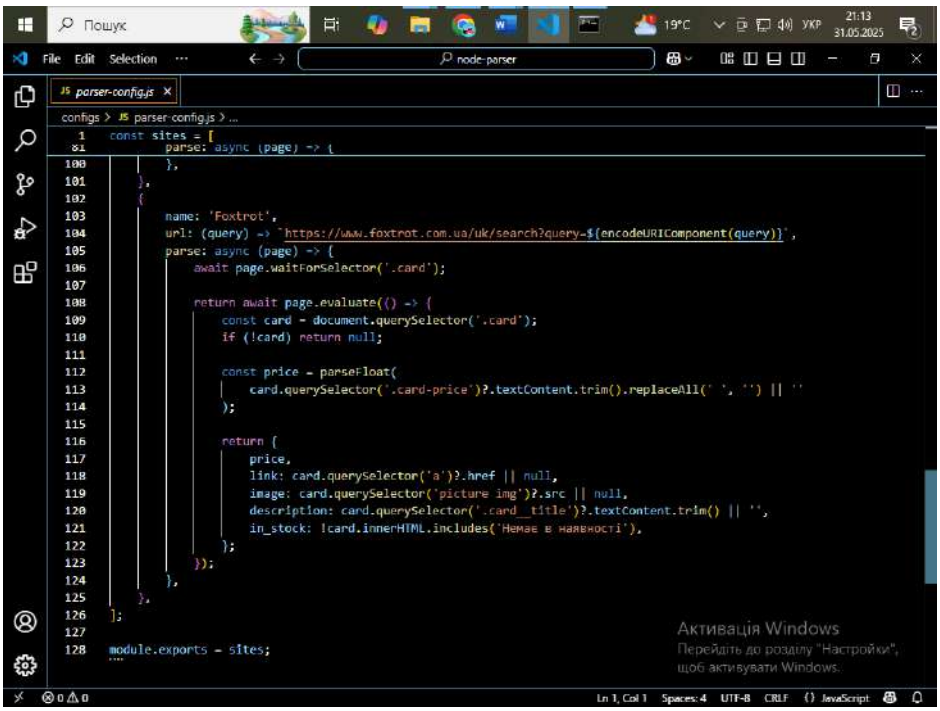
```
const sites = [

    {
        name: 'Allo',
        url: (query) => `https://allo.ua/ua/catalogsearch/result/?q=${encodeURIComponent(query)}`,
        parse: async (page) => {
            await page.waitForSelector('.product-card');

            return await page.evaluate(() => {
                const card = document.querySelector('.product-card');
                if (!card) return null;

                const price = parseFloat(
                    card.querySelector('.sum')?.textContent.replaceAll(' ', '') || ''
                );

                return {
                    price,
                    link: card.querySelector('a.product-card__title')?.href || null,
                    image: card.querySelector('img')?.src || null,
                    description: card.querySelector('.product-card__title')?.textContent || '',
                    in_stock: !card.innerHTML.includes('Немає в наявності'),
                };
            });
        },
    },

    {
        name: 'Citrus',
        url: (query) => `https://www.ctrs.com.ua/search/?query=${encodeURIComponent(query)}`,
        parse: async (page) => {
            await page.waitForSelector('[class*=MainProductCard-module__root]');
```

(c)



```
const sites = [
        parse: async (page) => {
            });
        },
    },

    {
        name: 'Citrus',
        url: (query) => `https://www.ctrs.com.ua/search/?query=${encodeURIComponent(query)}`,
        parse: async (page) => {
            await page.waitForSelector('[class*=MainProductCard-module__root]');

            return await page.evaluate(() => {
                const card = document.querySelector('[class*=MainProductCard-module__root]');
                if (!card) return null;

                const price = parseFloat(
                    card.querySelector('[class*=price]')?.dataset.price || ''
                );

                return {
                    price,
                    link: card.querySelector('a')?.href || null,
                    image: card.querySelector('img')?.src || null,
                    description: card.querySelector('[class*=title]')?.textContent.trim() || '',
                    in_stock: !card.innerHTML.includes('Немає в наявності'),
                };
            });
        },
    },
    {
        name: 'Foxtrot',
```

(d)



(е)

Рис. 3.11. зміст файлу parser-config.js

Примітка. Джерело: розроблено автором

У результаті реалізації механізму пошуку та порівняння даних з інтернет-магазинів було створено ефективний та гнучкий модуль для збору інформації з кількох торгових платформ. Використання бібліотеки Puppeteer дозволило отримувати дані безпосередньо з динамічних веб-сторінок, обробляти їх та конвертувати у зручний для аналізу формат.

Розроблений алгоритм підтримує масштабованість та налаштування, що дозволяє легко додавати нові джерела без суттєвих змін до кодової бази. Пошукові запити користувачів обробляються через веб-інтерфейс, після чого система автоматично порівнює знайдені товари за ціною, назвою та іншими параметрами, формуючи зручний результат для перегляду.

Таким чином, розроблений механізм є надійною основою для подальшого вдосконалення та розширення функціональності веб-застосунку, зокрема в напрямку підвищення точності пошуку, обробки знижок та персоналізації результатів для користувача.

3.3 Тестування веб-додатку для порівняння товарів

Тестування функціоналу є важливим етапом розробки, оскільки віно гарантує стабільність та ефективність додатку. Під час тестування важливо застосувати усі ключові функції веб-додатку, такі як коректне надсилання запиту, запуск парсера, повернення та відображення результатів та стуації. Скріншот головної сторінки на рис. 3.12.

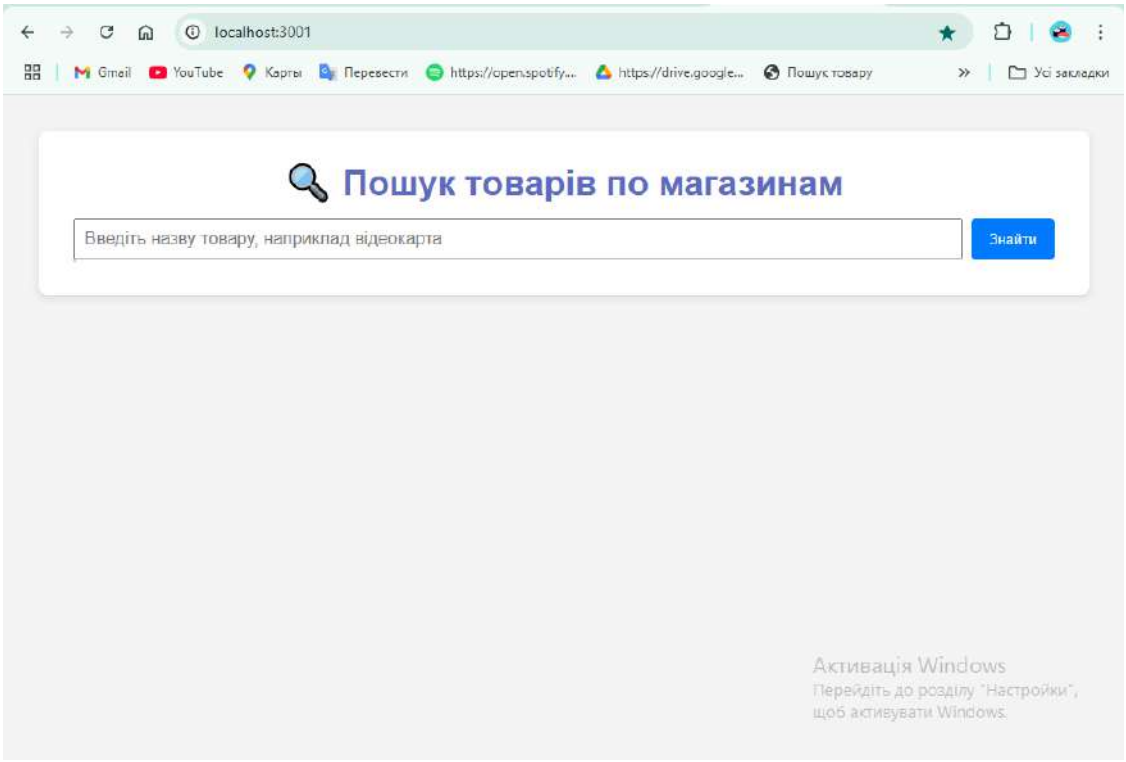


Рис. 3.12. Головна сторінка розробленого веб-додатку

Примітка. Джерело: розроблено автором

Головна сторінка відображена коректно, всі елементи в правильних пропорціях, не заважають один одному і не накладаються, а також – контрастують з фоновим кольором.

Наступною функцією на тестування є – відображення списку з переліком товарів, які мають ключові слова зі списку, який ми задавали раніше. На рис. 3.15, показано результат тестування, програмний код закріплено у додатку А.

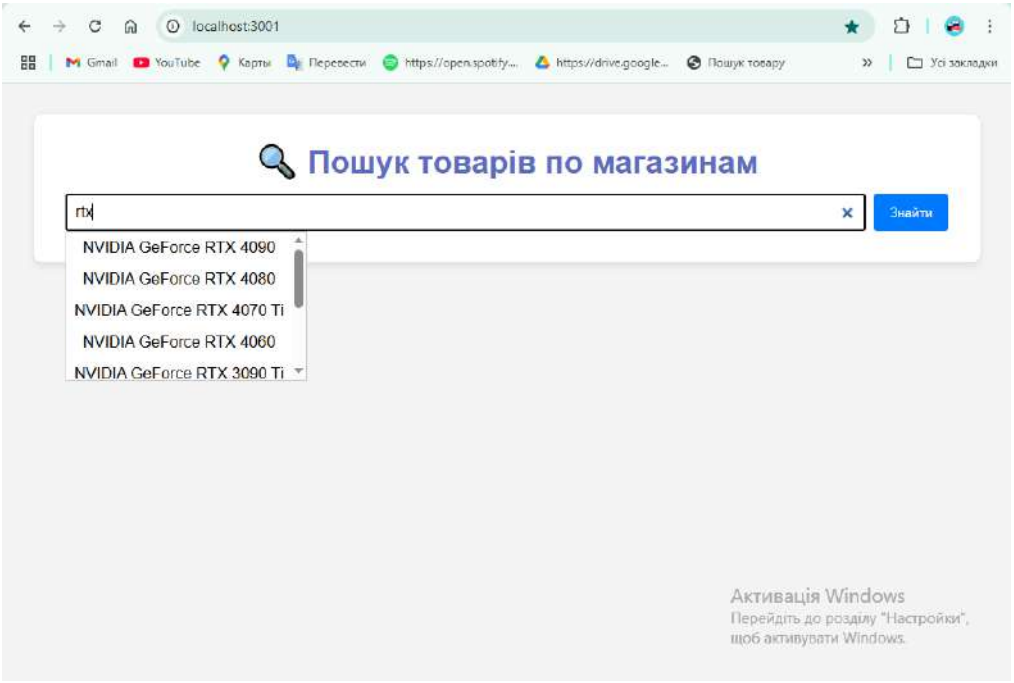


Рис. 3.13. Результат тестування та взаємодії з стрічкою пошуку та функції дозаповнення

Примітка. Джерело: розроблено автором

Як бачимо – результат задовільний, список відобразився коректно, нічого зайвого не показано, кожен з них дозаповнює наш запит.

Третьою функцією на тест, є відображення екрану завантаження. Під час цього жоден з елементів не має бути доступним, аби не переривати процеси а також має відобразитися спеціальна анімація, зображена на рис. 3.14.

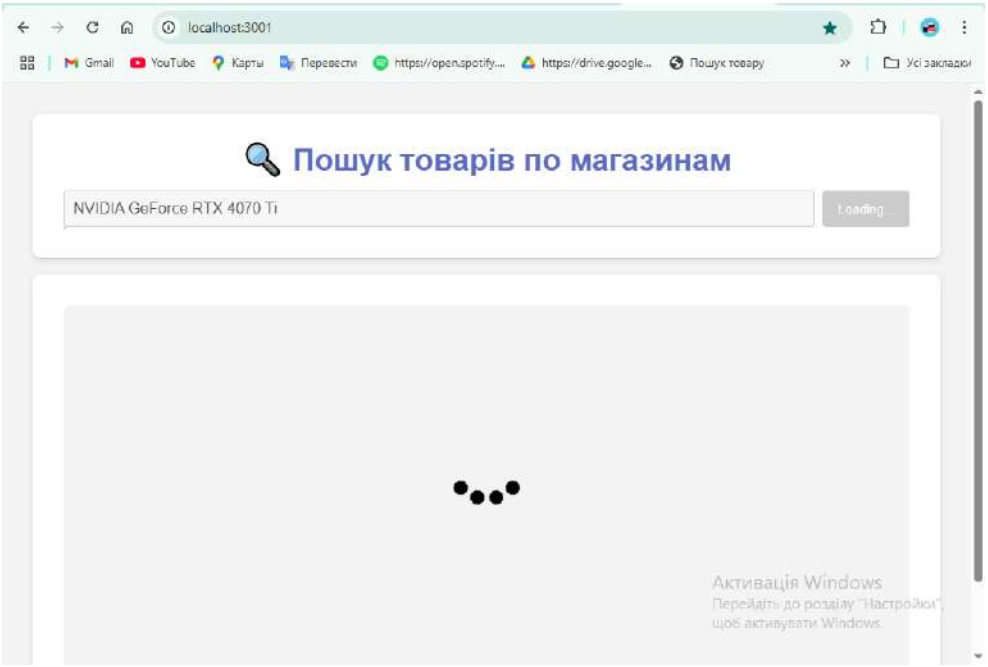


Рис. 3.14. Результат анімації під час пошуку та обробки даних

Примітка. Джерело: розроблено автором

Наступна функція – відображення списку результатів. Список відображається коректно, всі елементи на своєму місці, картка товару має в собі всі дані, зібрані за допомогою парсеру, такі як назва, ціна, назва та лого магазину, а також унікальне посилання на оригінальному сторінку цього товару. Результат зображено на рис. 3.15.

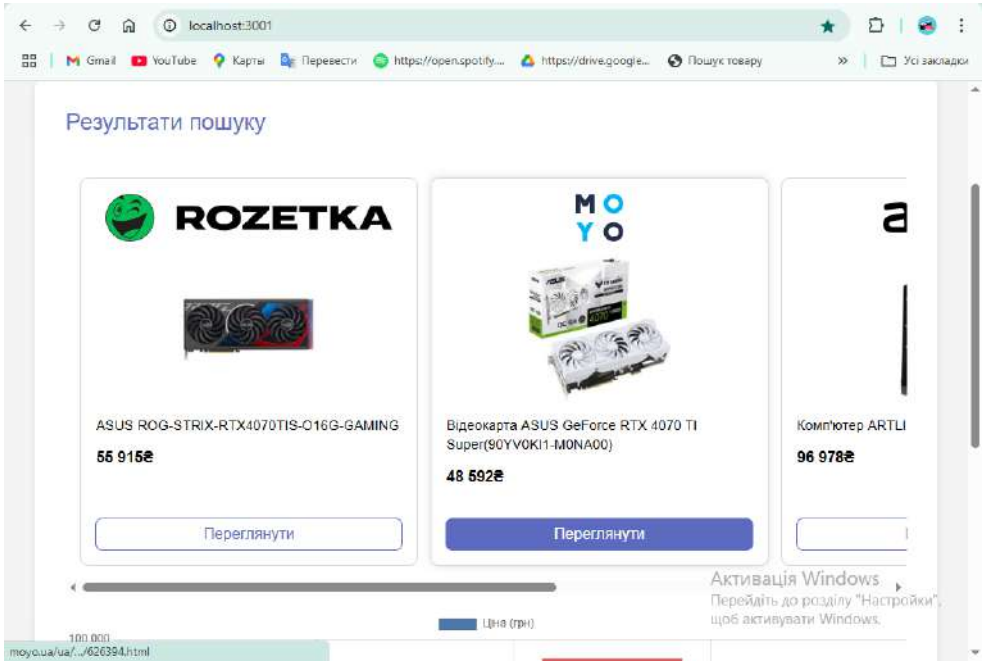


Рис. 3.15. Результати пошуку на розробленому сайті

Примітка. Джерело: розроблено автором

І останній елемент – порівняльна таблиця. Вона має відобразити співвідношення ціни між магазинами за допомогою стовпчикової діаграми. Також при наведенні на кожен стовпець має показуватися короткий звіт даних, з якого магазину та в яку ціну цей товар. Таблиця зображена на рис. 3.16.

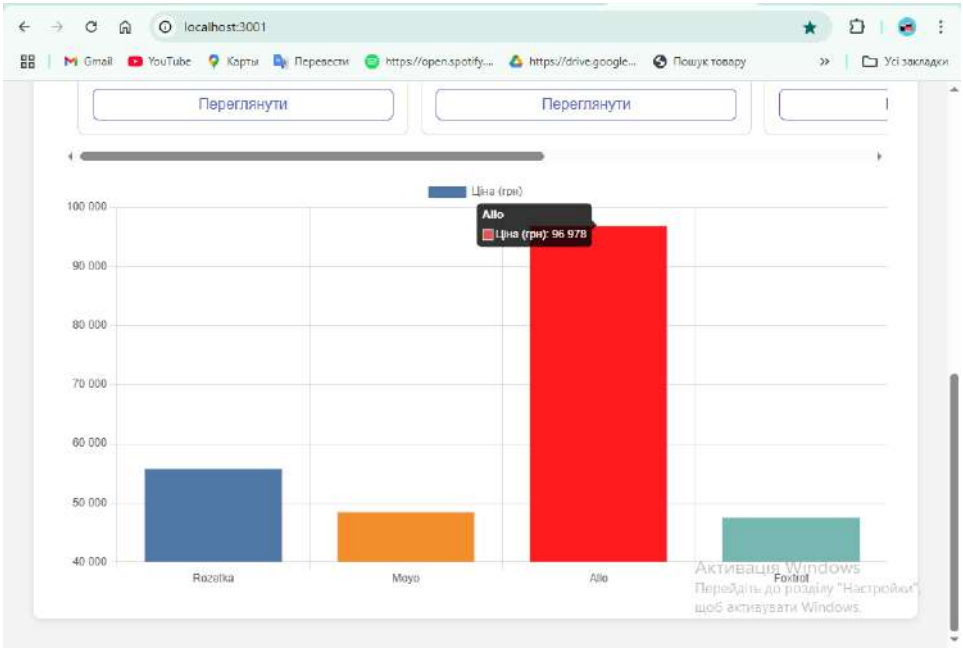


Рис. 3.16. Порівняльна таблиця цін з нашого сайту

Примітка. Джерело: розроблено автором

Додатково протестуємо, що буде якщо вибрати товар не зі списку запропонованих. Для прикладу спробуємо знайти товар «м’яка іграшка». Приклад зображено на рис. 3.17.

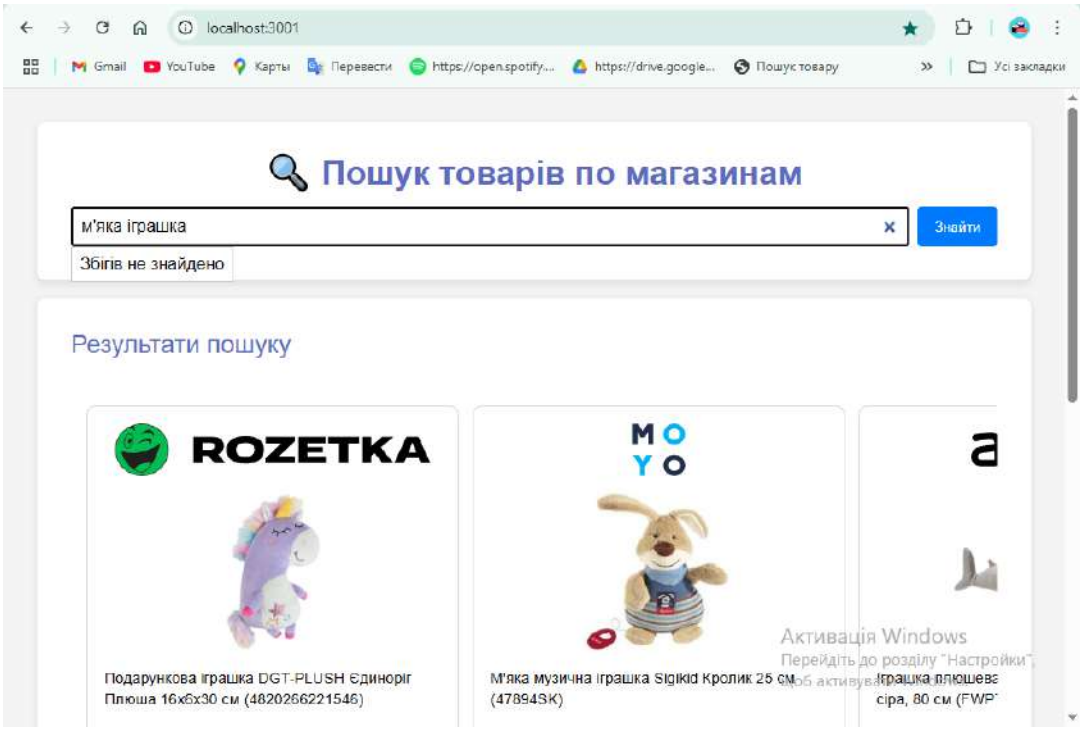


Рис. 3.17. Результат пошуку товару, не зі списку збігів

Примітка. Джерело: розроблено автором

Під час введення, список доповнень замінився на напис «Збігів не знайдено». Результати пошуку прикріпляю на рис. 3.18. та порівняльна таблиця за цим запитом, зображена на рис. 3.19.

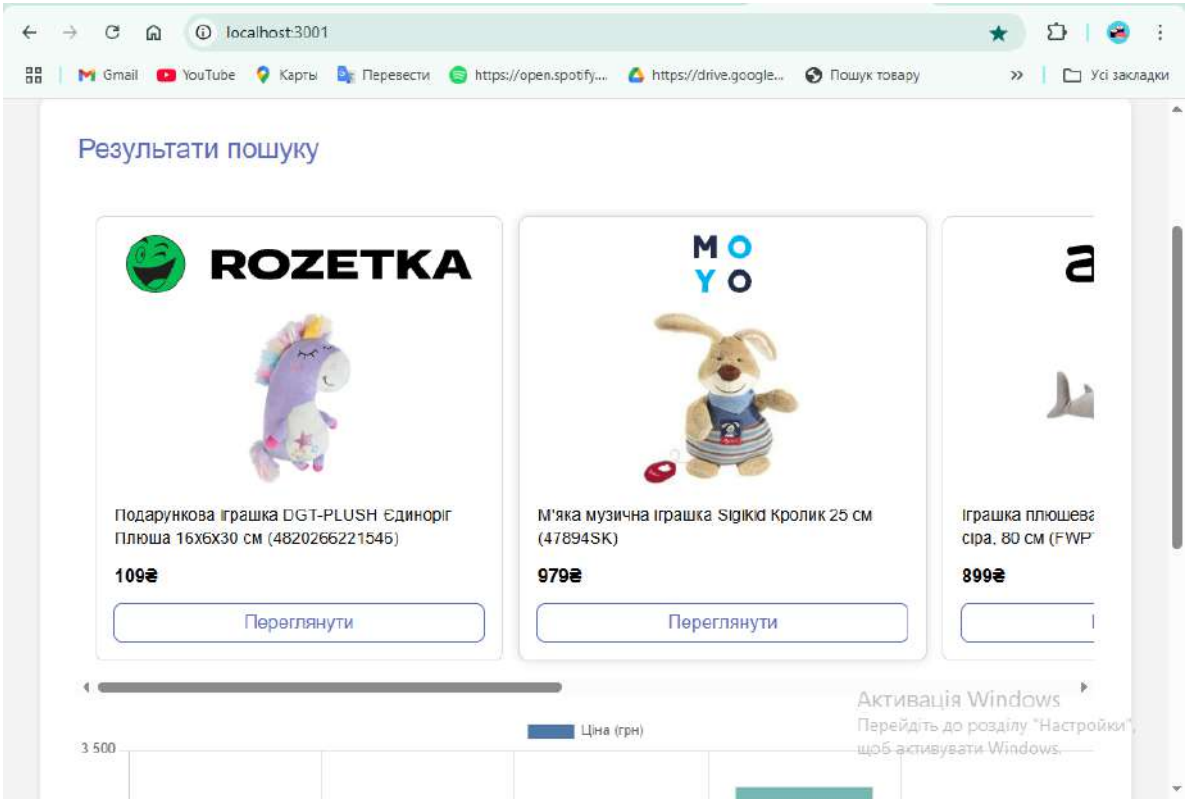


Рис. 3.18. Результати пошуку за запитом «м’яка іграшка»

Примітка. Джерело: розроблено автором

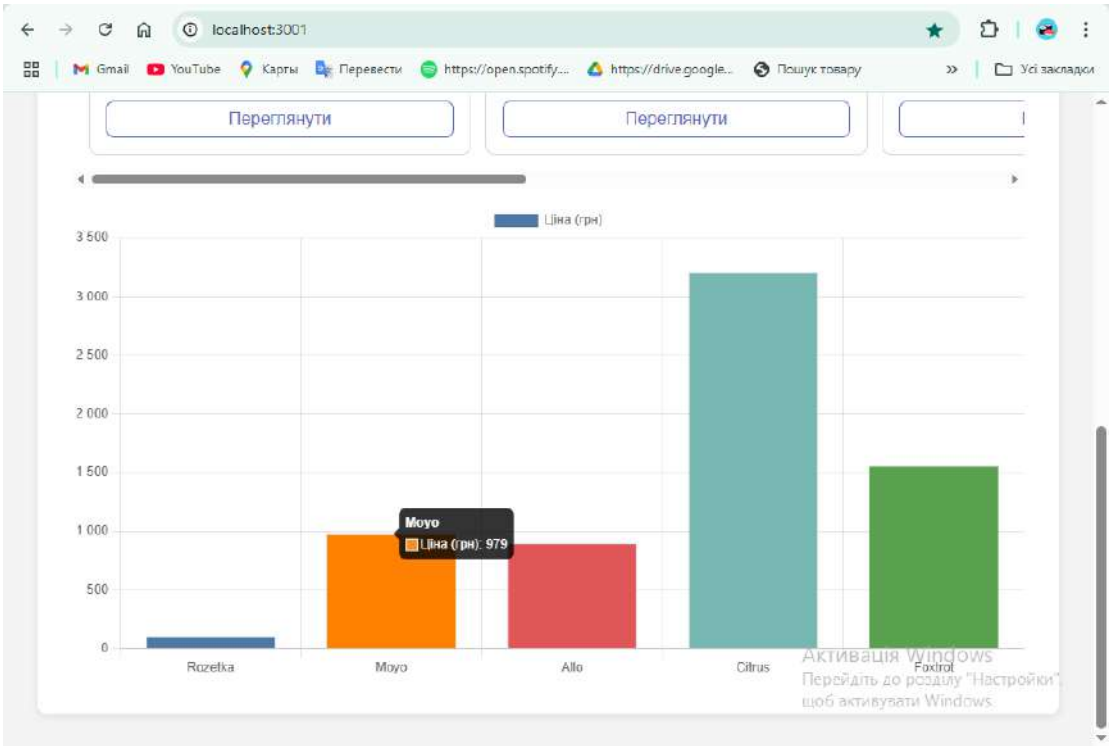


Рис. 3.19. Порівняльна таблиця за запитом «м’яка іграшка»

Примітка. Джерело: розроблено автором

Також, під час тестування, було підтверджено, що під якщо залишити стрічку пошуку пустою, то пошук просто не відбудеться, програма аналізує те, що стрічка пошуку пуста, і не починає пошук товару.

Останній тест, хотілося б провести на адаптивність. Нижче я прикріплю скріншоти з емулятора, в якому спеціально виберу розширення, яке відрізняється від оригінального. Результат зображено на рис. 3.20 та 3.21.

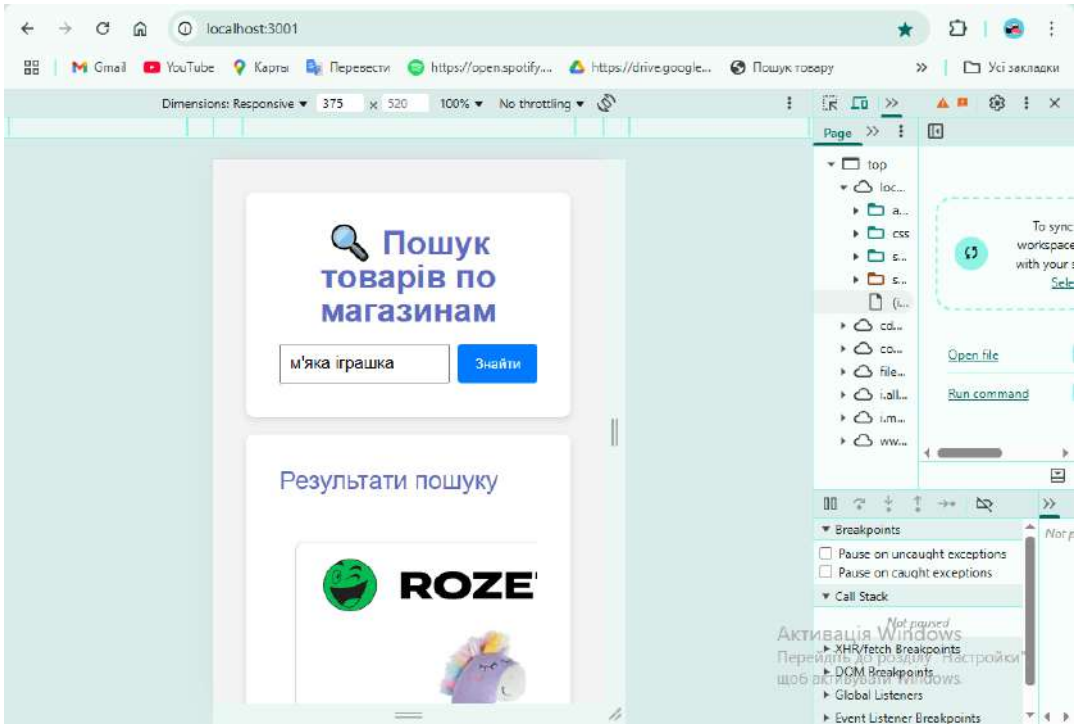


Рис. 3.20. Поведінка контейнеру зі стрічкою пошуку та контейнеру з його результатами під час зменшення розширення

Примітка. Джерело: розроблено автором

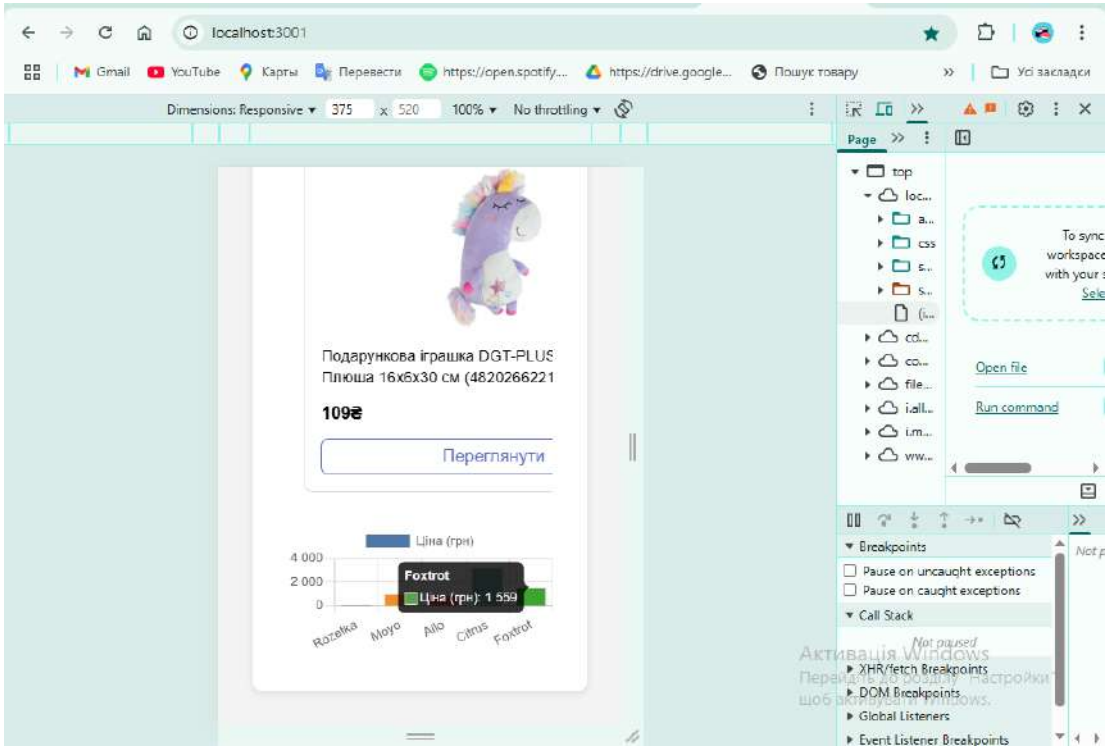


Рис. 3.21. Поведінка порівняльної таблиці під час зменшення розширення

Примітка. Джерело: розроблено автором

Земулювавши розширення 375x520 пікселів, ми бачимо, що інтерфейс перейшов у інший формат, в якому воно також коректно відображується.

Під час тестування веб-додатка для порівняння цін інтернет-магазинів підтверджено його працездатність, функціональність та відповідність вимогам. Проведено функціональні, інтеграційні та модульні тести, що охоплюють основні сценарії взаємодії користувача з системою – від введення пошукового запиту до відображення результатів у веб-інтерфейсі.

Парсинг даних із використанням бібліотеки Puppeteer показав стабільну роботу при отриманні даних із зовнішніх джерел. Система коректно реагує на помилки, пов’язані з недоступністю магазинів, відсутністю результатів чи некоректними запитами. Також проведено тест відображення результатів на сторінці, адаптивність дизайну та зручність взаємодії користувача з інтерфейсом.

Всі компоненти, що тестуються, взаємодіють коректно, що свідчить про правильну реалізацію архітектури додатку. Результати тестування підтвердили готовність системи до практичного використання та можливість подальшого

розширення її функціоналу.

Висновки до розділу 3

У третьому розділі здійснено практичну реалізацію веб-додатку для порівняння товарних пропозицій з інтернет-магазинів відповідно до попередньо спроєктованої архітектури. Розробка охопила всі ключові компоненти системи, включаючи користувацький інтерфейс, серверну логіку, а також механізми збору та порівняння даних.

Користувацький інтерфейс було створено з дотриманням сучасних принципів UX/UI-дизайну, з акцентом на зручність, доступність та візуальну лаконічність. Реалізоване рішення забезпечує інтуїтивно зрозумілу навігацію та оперативний доступ до релевантної інформації без надмірного візуального навантаження.

Серверна частина додатку реалізована з використанням платформи Node.js і фреймворку Express, що забезпечило ефективну маршрутизацію запитів і стабільну роботу серверної логіки. Збір даних з веб-ресурсів здійснювався за допомогою бібліотеки Puppeteer, яка дозволила реалізувати автоматизований парсинг динамічного контенту.

Для формування динамічних HTML-сторінок застосовано шаблонізатор EJS, що дало змогу інтегрувати та відображати отримані з різних джерел дані в узгодженому та зручному для користувача форматі.

У рамках тестування було проведено перевірку функціональної коректності основних модулів додатку, зіставлення результатів із очікуваними даними, а також оцінено стабільність роботи системи за типових сценаріїв використання. Тестові випробування підтвердили відповідність реалізованого функціоналу заявленим вимогам.

У підсумку, в результаті реалізації третього розділу створено повноцінний, функціонально завершений веб-додаток, здатний ефективно виконувати порівняння товарів із різних онлайн-магазинів у реальному часі.

ВИСНОВКИ

У результаті виконання дипломної роботи було досягнуто поставлену мету – розроблено функціональний веб-додаток для пошуку, збору та порівняння товарних пропозицій з кількох онлайн-магазинів у режимі реального часу.

У першому розділі здійснено комплексний аналіз предметної області електронної комерції. Розглянуто ключові аспекти порівняння товарної інформації в онлайн-середовищі, проведено огляд сучасних веб-сервісів із відповідною функціональністю. Обґрунтовано вибір інструментів і технологій, що були використані для реалізації розробленого рішення. За результатами аналізу сформовано оптимальний технологічний стек, до якого увійшли Node.js, Express, Puppeteer та EJS.

У другому розділі представлено концептуальне проєктування системи. Сформовано архітектурну модель веб-додатку, описано логіку його функціонування та створено прототип користувацького інтерфейсу. Запропонована структура забезпечує гнучкість, модульність і передбачає можливість масштабування функціоналу в майбутньому.

У третьому розділі наведено безпосередню реалізацію всіх основних компонентів системи, зокрема інтерфейсу користувача, модулів пошуку та порівняння товарів, а також механізмів веб-скрапінгу для збору інформації з сайтів інтернет-магазинів. Значну увагу приділено тестуванню розробленого програмного продукту: перевірено коректність роботи парсера, обробки запитів, відповідність отриманих результатів очікуваним, а також загальну стабільність системи за різних сценаріїв використання.

Розроблений веб-додаток підтвердив свою ефективність у вирішенні прикладної задачі автоматизованого порівняння товарів з різних джерел. Практичні результати дослідження можуть бути використані як основа для створення комерційного програмного забезпечення або для подальших наукових розвідок у сфері автоматизації онлайн-шопінгу й розвитку електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Appendix II, Crafting Interpreters, Robert Nystrom, 2021: https://timothya.com/pdfs/crafting-interpreters.pdf

2. Index | Node.js v24.2.0 Documentation. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 01.06.2025).

3. C M. R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson Education Asia, 2017.

4. Учасники проектів Вікімедіа. Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Головна_сторінка (дата звернення: 06.06.2025).

5. Hotline - порівняти ціни в інтернет-магазинах України. *Hotline.ua*. URL: https://hotline.ua/ (дата звернення: 11.06.2025).

6. Веб-сервіс для порівняння даних Price.ua: https://price.ua/ua

7. e-Katalog – каталог товарів і цін в інтернет-магазинах. *ek.ua - порівняння, відгуки, ціни в інтернет-магазинах*. URL: https://ek.ua/ua/ (дата звернення: 05.06.2025).

8. Magazilla – порівняння цін в інтернет-магазинах. *Magazilla - порівняння цін в інтернет-магазинах*. URL: https://m.ua/ua/ (дата звернення: 04.06.2025).

9. Веб-сервіс для порівняння даних Google Shopping: https://google-shopping.com.ua/

10. Teixeira P. Professional Node. js: Building Javascript Based Scalable Software. Wiley & Sons, Incorporated, John, 2012. 408 p.

11. Beginning Node.js, Express & MongoDB Development. Independently Published, 2019. 155 p.

12. Wandschneider M. Learning Node. js: A Hands-On Guide to Building Web Applications in JavaScript. Pearson Education, Limited, 2016. 320 p.

13. Екскурс до початку парсингу сайтів: https://www.promodo.ua/blog/parsing-saytiv-shcho-ce-yak-pracyuie-ta-navishcho-parsiti-dani

ЗГОДА здобувача вищої освіти

Державного університету економіки і технологій про перевірку кваліфікаційної роботи на прояви академічного плагіату та розміщення в Репозитарії Університету

Я, Баніт Євген Русланович, підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота «Створення веб-додатка для порівняння даних з інтернет-магазинів» виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений.

10.06.2025

Баніт Є.Р.

(підпис) (прізвище та ініціали)