ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Веб-сайт з продажу комплектуючих для ПК»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

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

_____ Владислав КАРАЄВ
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр. КНД-41
Владислав КАРАЄВ

Керівник: Олена ШИКУЛА
науковий ступінь, вчене звання д.ф.-м.н., професор

Рецензент:
науковий ступінь, вчене звання _____
(Ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Караєву Владиславу Руслановичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Веб-сайт з продажу комплектуючих для ПК

керівник кваліфікаційної роботи Олена ШИКУЛА д.ф.-м.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» 02.2025р. № 56

2. Строк подання кваліфікаційної роботи «30» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література, веб-технології HTML, CSS, JavaScript, Node.js, СКБД MySQL

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   Аналіз сучасних веб-технологій для створення інтернет-магазинів
   Дослідження потреб користувача та типових функцій веб-магазинів
   Розробка веб-сайту для магазину комплектуючих ПК.
5. Перелік графічного матеріалу: *презентація*
   1. Інтерфейси популярних веб-магазинів комплектуючих ПК

2. Схема взаємодії JavaScript та DOM у браузері
3. Екранні форми веб-сайту.
4. Фрагменти програмного коду магазину.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз вимог до веб-магазинів	27.02-11.03.25	Виконано
2	Дослідження ринку комплектуючих ПК	12.03-18.03.25	Виконано
3	Аналіз веб-технологій	19.03-26.03.25	Виконано
4	Проєктування архітектури сайту та бази даних	27.03-01.04.25	Виконано
5	Розробка клієнтської частини	02.04-07.04.25	Виконано
6	Реалізація функціоналу	08.04-01.05.25	Виконано
7	Оформлення роботи: вступ, висновки, реферат	02.05-16.05.25	Виконано
8	Розробка демонстраційних матеріалів	17.05-21.05.25	Виконано

Здобувач вищої освіти _____ Владислав КАРАЄВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Олена ШИКУЛА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 51 стор., 3 табл., 18 рис., 20 джерел.

Мета роботи – створення веб-сайту для продажу комплектуючих, який задовільняє потреби користувачів, відповідає сучасним стандартам веб-розробки та забезпечує зручну взаємодію клієнта з товаром.

Об'єкт дослідження – магазин комплектуючих ПК.

Предмет дослідження – структура, функціональність та технічна реалізація веб-сайту з продажу комплектуючих для ПК.

Короткий зміст роботи:

Здійснено огляд та аналіз сучасних технологій веб-розробки, які застосовуються для створення інтернет-магазинів. Проаналізовано вимоги до структури та функціоналу веб-магазину, зокрема щодо зручності інтерфейсу, безпеки та обробки даних.

За допомогою HTML, CSS, JavaScript, Node.js та СКБД MySQL розроблено веб-сайт для продажу комп'ютерних комплектуючих. Реалізовано такі функції: каталог товарів з фільтрацією, кошик, авторизація, система знижок на день народження, можливість залишати відгук, базове зберігання даних.

Готове рішення може бути використане як основа для впровадження подібних систем в малому бізнесі, з можливістю подальшого розширення функціоналу.

КЛЮЧОВІ СЛОВА: ВЕБ-МАГАЗИН, КОМП'ЮТЕРНІ КОМПЛЕКТУЮЧІ, JAVASCRIPT, NODE.JS, СКБД MYSQL

ЗМІСТ

ВСТУП

У сучасному цифровому світі розвиток інформаційних технологій значною мірою впливає на всі сфери суспільного життя. Особливо стрімко змінюється галузь торгівлі, де дедалі більше компаній переходять на електронні засоби ведення бізнесу. В умовах високої конкуренції та швидких змін у поведінці споживачів традиційні канали продажу вже не є достатньо ефективними. Саме тому все більшу актуальність набуває створення зручних, інтуїтивно зрозумілих та функціональних веб-сайтів для онлайн-продажу товарів, зокрема й комплектуючих для персональних комп'ютерів.

Комплектуючі для ПК — це особливий сегмент ринку, що постійно оновлюється через швидкий розвиток апаратного забезпечення. Потенційні покупці, зазвичай, мають високі вимоги до точності технічних характеристик, наявності варіантів конфігурацій, сумісності компонентів тощо. Тому ефективна веб-платформа повинна не лише забезпечувати продаж товарів, але й надавати користувачам якісний інтерфейс для навігації, фільтрації, порівняння та детального ознайомлення з характеристиками продукції. В умовах розвитку електронної комерції веб-сайт є не лише інструментом продажу, а повноцінною платформою взаємодії з клієнтами. Він виконує функції вітрини, каталогу товарів, консультанта, сервісу підтримки, а також інструменту аналітики. Правильно реалізований веб-сайт дозволяє значно скоротити витрати на оренду торгових площ, зменшити потребу в персоналі, розширити географію продажу, а також збільшити доступність товарів для клієнтів. Це особливо важливо для малих та середніх підприємств, які прагнуть бути конкурентоспроможними.

Одним із головних аспектів створення ефективного веб-сайту є його технічна реалізація: вибір мови програмування, платформи, бази даних, системи керування вмістом (CMS), а також проєктування структури інтерфейсу, логіки обробки замовлень та інтеграції з платіжними системами. Важливу роль відіграє також

адаптивний дизайн, який дає змогу комфортно користуватись ресурсом як з комп’ютера, так і з мобільних пристроїв. Окремої уваги потребує забезпечення безпеки користувацьких даних, особливо під час обробки платежів та реєстрації користувачів.

В межах даної дипломної роботи розглядається процес розробки веб-сайту з продажу комплектуючих для ПК, що охоплює повний цикл — від збору вимог та проєктування до реалізації функціонального веб-ресурсу. У ході реалізації використовуються сучасні технології веб-розробки, що дозволяють створити інтерактивний інтерфейс користувача, забезпечити обробку запитів, а також організувати зберігання та обробку даних у базі даних. Актуальність теми зумовлена як високим попитом на комп’ютерну техніку та комплектуючі, так і потребою у швидкому, надійному, зручному способі їх придбання. Велика кількість користувачів віддають перевагу онлайн-шопінгу, оскільки він дозволяє швидко отримати необхідну інформацію, порівняти ціни, прочитати відгуки і здійснити покупку без фізичного відвідування магазину.

Об'єкт дослідження – магазин комплектуючих ПК.

Предмет дослідження – структура, функціональність та технічна реалізація веб-сайту з продажу комплектуючих для ПК.

Мета роботи – створення веб-сайту для продажу комплектуючих, який задовільняє потреби користувачів, відповідає сучасним стандартам веб-розробки та забезпечує зручну взаємодію клієнта з товаром.

Наукове завдання:

- Проаналізувати вимоги до сучасних інтернет-магазинів у сфері комп’ютерної техніки
- Розробити структуру веб-сайту: головну сторінку, каталог товарів, форму замовлення
- Реалізувати базу даних для зберігання інформації про товари
- Створити систему реєстрації та авторизації користувачів
- Забезпечити функціональність пошуку та фільтрування продукції

Практична значимість. Розроблений веб-сайт може бути використаний як готове рішення для малого бізнесу в сфері продажу комплектуючих ПК. Завдяки простій архітектурі, адаптивному дизайну та інтерактивному інтерфейсу він забезпечує комфортну роботу як для власника магазину, так і для користувача. Веб-додаток розроблений з використанням відкритих веб-технологій, що спрощує його обслуговування, впровадження і доопрацювання, а також дає можливість розгортання на будь-якому сучасному сервері або хмарному середовищі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Комп’ютерні комплектуючі як основа сучасного ПК

Комп’ютерні комплектуючі — це апаратні елементи, які разом формують повноцінну систему персонального комп’ютера. Кожен з них виконує окрему функцію: обробку даних, зберігання інформації, графічне відображення, зв’язок із користувачем або зовнішніми пристроями. Усі комплектуючі взаємодіють між собою через центральну плату — материнську.

Сучасні інформаційні технології стрімко розвиваються, тому ринок комплектуючих постійно оновлюється. З’являються нові стандарти, інтерфейси, підвищується енергоефективність і продуктивність компонентів. Це вимагає від продавців та користувачів обізнаності у технічних характеристиках товарів і правильного підбору сумісних пристроїв.

У веб-магазинах продаж комплектуючих має певні особливості: необхідна можливість гнучкої фільтрації, сортування за характеристиками, швидкий доступ до описів і технічної інформації. Тому веб-ресурси, присвячені цій тематиці, повинні бути адаптовані до специфіки продукції та вимог цільової аудиторії.

1.2 Основні вимоги до сучасних веб-магазинів

У нинішній час електронна комерція досягла високої позиції у світовому ринку. Дуже швидке зростання онлайн-торгівлі зумовлює необхідність створення ефективних веб-магазинів, які не лише виконують функцію платформи для продажу товарів, а й забезпечують комплексну взаємодію з користувачами. Веб-сайти, що реалізують електронну торгівлю, повинні бути зручними, швидкими, безпечними, адаптивними та функціональними.

Однією з найважливіших вимог до веб-магазину є інтуїтивна навігація. Відвідувач повинен легко орієнтуватися в структурі сайту: знаходити потрібні категорії, фільтрувати товари, шукати інформацію про наявність, характеристики, ціни. Для цього необхідна логічно організована структура каталогу, продумана система категорій та зручна реалізація функції пошуку. Інтерфейс користувача повинен бути простим, зрозумілим і доступним навіть для людей без технічних навичок. Успішний веб-магазин має передбачати мінімальну кількість кліків для здійснення покупки.

Наявність інструментів фільтрації (за ціною, брендом, категорією, наявністю знижок тощо), сортування за популярністю, рейтингом або ціною сприяє покращенню користувацького досвіду. Користувачам важливо мати змогу миттєво знайти бажаний товар серед всіх позицій. Такі інструменти дозволяють суттєво скоротити час пошуку й підвищити ефективність взаємодії з платформою.

Адаптивність веб-сайту один з ключових параметрів. Більшість користувачів здійснює покупки з мобільних пристроїв, тому сайт повинен коректно відображатися як на десктопах, так і на смартфонах чи планшетах. Використання технологій адаптивного дизайну та медіа-запитів дозволяє досягти цієї мети. Сторінки мають автоматично підлаштовуватись під розміри екрана, зберігаючи при цьому зручність користування і повну функціональність.

Дуже важливою є безпека даних. У процесі реєстрації, входу в обліковий запис, оформлення замовлення та проведення оплати веб-сайт обробляє персональні дані користувача. Відповідно, необхідно впровадити заходи шифрування, автентифікації та валідації. Для захисту від несанкціонованого доступу використовуються протоколи HTTPS, методи хешування паролів, захист від ботів, а також регулярні перевірки безпеки. Веб-магазин має бути захищеним від типових загроз, таких як міжсайтові скриптові атаки (XSS), міжсайтові запити (CSRF), фішинг та інші види шахрайства. Крім того, веб-сайт повинен бути оптимізованим щодо швидкодії. Дослідження показують, що навіть

незначна затримка в завантаженні сторінки може спровокувати втрату можливого клієнта. Швидке завантаження сторінок досягається за рахунок оптимізації графічного контенту, мінімізації CSS і JavaScript-файлів, використання кешування, CDN-серверів та лінійного кодування. Мінімізація кількості HTTP-запитів, правильна структура коду та відкладене завантаження ресурсів також позитивно впливають на продуктивність.

Пошукова оптимізація SEO є ще одним критичним аспектом. Веб-магазин повинен бути добре індексований пошуковими системами. Це досягається шляхом правильного використання заголовків, мета-тегів, структури URL, семантичної розмітки, наявності sitemap.xml і robots.txt. Високий рейтинг у пошукових системах забезпечує притік органічного трафіку без додаткових витрат на рекламу. Не менш важливою є інтеграція з платіжними системами. Веб-магазин має підтримувати популярні методи оплати такі як, банківські картки, онлайн-гаманці, оплата при отриманні тощо. Це вимагає надійної реалізації відповідних API, відповідності до вимог безпеки , а також тестування всіх можливих сценаріїв.

Окремо варто зазначити важливість аналітики. Успішні веб-магазини інтегрують інструменти для збору та детального аналізу даних: Google Analytics, внутрішні системи звітності, CRM. Це дозволяє відстежувати поведінку користувачів, ефективність маркетингових кампаній, популярність товарів, сезонність попиту й багато інших показників. Отримані дані можуть використовуватись для оптимізації структури сайту, підвищення конверсії, вдосконалення товарної політики.

Сучасний веб-магазин — це складна система, яка має задовольняти цілу низку технічних, функціональних та маркетингових вимог. Її ефективність визначається не лише технічними параметрами, але й загальним досвідом користувача, що, у свою чергу, напряму впливає на прибутковість та конкурентоспроможність бізнесу.

1.3 Актуальність розробки веб-сайту з продажу комплектуючих ПК

Сучасний ринок комп’ютерної техніки та комплектуючих характеризується високою швидкістю розвитку і постійними інноваціями, що спричиняє значну динаміку у виборі та оновленні апаратного забезпечення. Щороку на ринок виходять нові моделі процесорів, відеокарт, материнських плат, модулів оперативної пам’яті та інших компонентів, що підвищує рівень складності вибору для кінцевого споживача. В таких умовах традиційні способи продажу через фізичні магазини втрачають конкурентні переваги, адже вони не завжди можуть забезпечити достатній асортимент і своєчасне оновлення товарів. Це створює гостру потребу у високотехнологічних онлайн-платформах, які б поєднували в собі широкий вибір, актуальність інформації і зручність користування.

Паралельно з розвитком ринку комп’ютерних комплектуючих, суттєво зростає й кількість користувачів, які активно збирають і модернізують персональні комп’ютери самостійно. Для цієї категорії важливим є не лише широкий асортимент, але й точна і достовірна інформація про сумісність компонентів, їхні технічні характеристики і можливості. Це підкреслює необхідність створення надійних веб-сайтів із функціями інтелектуального підбору комплектуючих, які допомагають уникнути помилок при виборі, знижують ризик несумісності та оптимізують конфігурації з урахуванням різних сценаріїв використання — від офісних ПК до ігрових або робочих станцій.

Крім того, в умовах сучасної економіки споживачі дедалі більше цінують можливість отримати кваліфіковану підтримку і консультації онлайн. Розробка веб-сайту, що інтегрує не лише каталог товарів, а й інформаційно-довідкові матеріали, огляди, рейтинги, а також системи рекомендацій і допомоги у виборі, значно підвищує рівень довіри клієнтів і покращує їхній досвід користування цією платформою. Зручний інтерфейс, швидкий доступ до детальної інформації,

можливість порівняння різних моделей у режимі реального часу — все це сприяє більш обґрунтованому вибору і задоволенню користувачів.

Важливим аспектом є також забезпечення якісної логістики і сервісу доставки, що відіграє ключову роль у формуванні позитивного іміджу онлайн-магазину. Оскільки багато комплектуючих мають специфічні вимоги до транспортування, реалізація функцій вибору способу доставки, страхування товарів, можливості перевірки замовлення при отриманні та гарантійного обслуговування є необхідними елементами сучасного сервісу. Ці фактори значно впливають на рівень задоволеності покупців і повторність їхніх замовлень.

Загалом, розробка веб-сайту з продажу комп’ютерних комплектуючих є надзвичайно актуальним завданням, що відповідає сучасним тенденціям цифрової трансформації ринку. Такий проект дозволить не лише покращити доступність і якість послуг у сфері продажів комп’ютерної техніки, а й сприятиме підвищенню рівня цифрової грамотності населення, розвитку IT-галузі та стимулюватиме інноваційні процеси у сфері електронної комерції. В умовах зростаючої конкуренції і підвищених очікувань споживачів, впровадження передових технологічних рішень у створення і функціонування таких веб-сайтів стає ключовим фактором успішності бізнесу.

1.4 Аналіз популярних веб-магазинів комплектуючих ПК

У сучасних умовах розвитку інформаційних технологій попит на комп’ютерні комплектуючі невпинно зростає. Відповідно до цього спостерігається активне розширення ринку інтернет-магазинів, які спеціалізуються на продажу електроніки, зокрема комплектуючих до ПК. Конкуренція в цій сфері висока, що стимулює учасників ринку вдосконалювати функціональні можливості своїх платформ, значно покращувати рівень обслуговування клієнтів, оптимізувати логістику, а також впроваджувати новітні

технології у сфері веб-розробки. Аналіз сучасних веб-сайтів дозволяє виділити ключові тенденції та підходи до організації ефективного онлайн-продажу комплектуючих.

Одним із найяскравіших прикладів успішного українського інтернет-магазину є Rozetka Ця платформа вважається лідером ринку електронної комерції в Україні, зокрема у сфері продажу комп'ютерної техніки та комплектуючих. Розетка пропонує тисячі найменувань товарів — від процесорів, оперативної пам'яті, материнських плат, відеокарт і блоків живлення до різних аксесуарів. Структура сайту добре продумана: головна сторінка містить тематичні блоки, акційні пропозиції, популярні категорії та рекомендації, засновані на історії переглядів користувача. На рисунку 1.1 зображено інтерфейс каталогу комплектуючих ПК на сайті Rozetka.

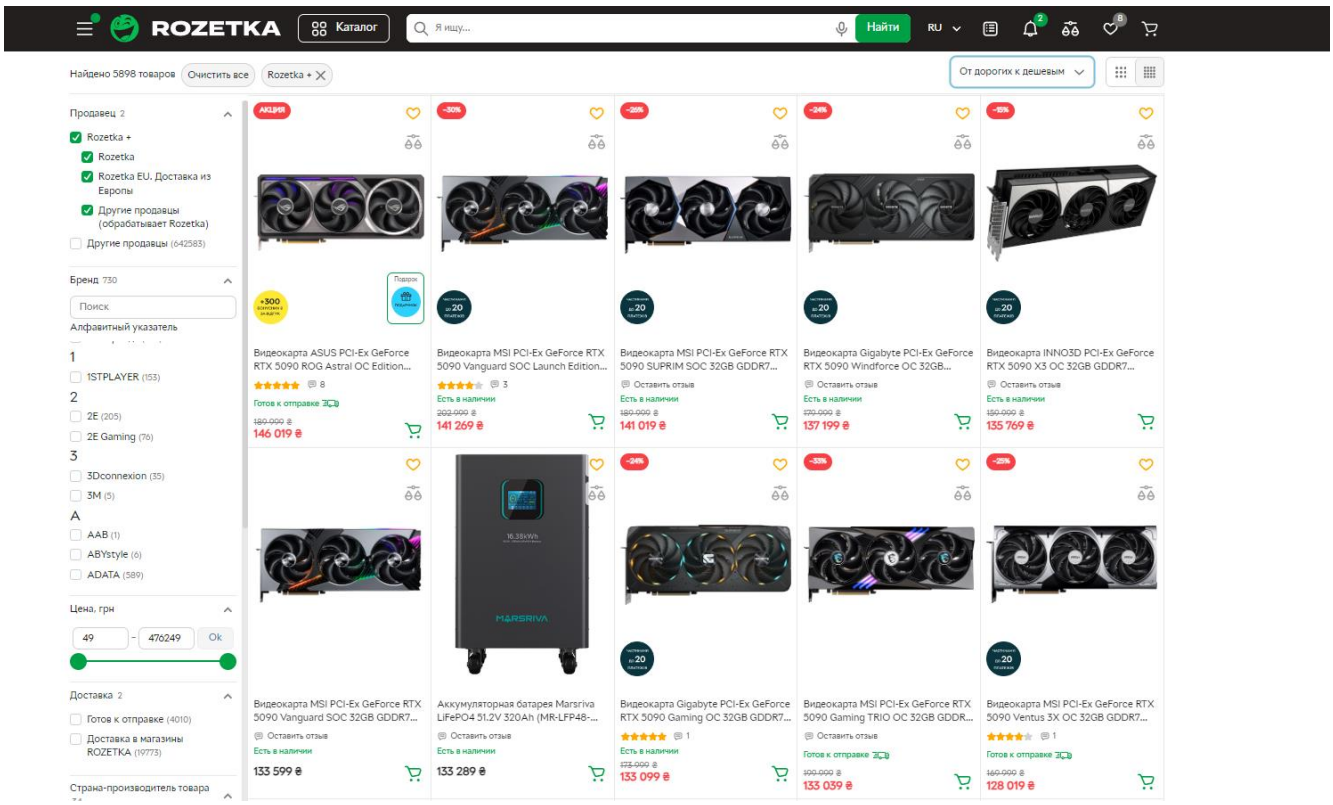


Рисунок 1.1 – Інтерфейс каталогу комплектуючих ПК на веб-сайті Rozetka

Важливою перевагою Rozetka є інтуїтивно зрозумілий інтерфейс, що дозволяє зручно орієнтуватися навіть недосвідченим користувачам. Користувач

може легко знайти потрібний товар за допомогою зручного пошуку, фільтрів за технічними характеристиками, виробником, ціною, рейтингом, наявністю на складі тощо. Реалізована система порівняння дозволяє зіставити параметри кількох товарів одночасно, що значно спрощує вибір. Додатково варто зазначити високий рівень опрацювання картки товару — детальний опис, велика кількість якісних фотографій, відеоогляди, технічні специфікації, а також відгуки та питання користувачів.

Rozetka має багато варіантів оплати та інтегрована з великою кількістю служб доставки, включно з власною кур'єрською мережею. Підтримка користувачів працює 24/7, що дозволяє оперативно вирішувати питання, пов'язані з покупками або доставкою. Важливою особливістю є адаптивна верстка, що забезпечує повноцінну роботу сайту на мобільних пристроях. Це дуже важливо з огляду на те, що велика частина покупок зараз здійснюється саме через смартфони.

Ще одним сильним гравцем на українському ринку є інтернет-магазин «Цитрус». Він, окрім продажу смартфонів, гаджетів і аксесуарів, пропонує широкий вибір комплектуючих. Цитрус орієнтований на молодшу аудиторію і активно використовує сучасні дизайнерські рішення — темна тема, великі інтерактивні банери, яскраві кольори. Інтерфейс сайту відносно простий, із мінімалістичним дизайном, де акцент робиться на візуальні ефекти й зручність перегляду. Платформа підтримує адаптивність, має добре реалізовану мобільну версію, активну бонусну систему, партнерські програми та сервіси з розстрочення. Також варто відзначити інтеграцію з сервісами авторизації через Google чи Facebook, що пришвидшує реєстрацію та покупку. На рисунку 1.2 ілюстровано головну сторінку веб-магазину Цитрус.

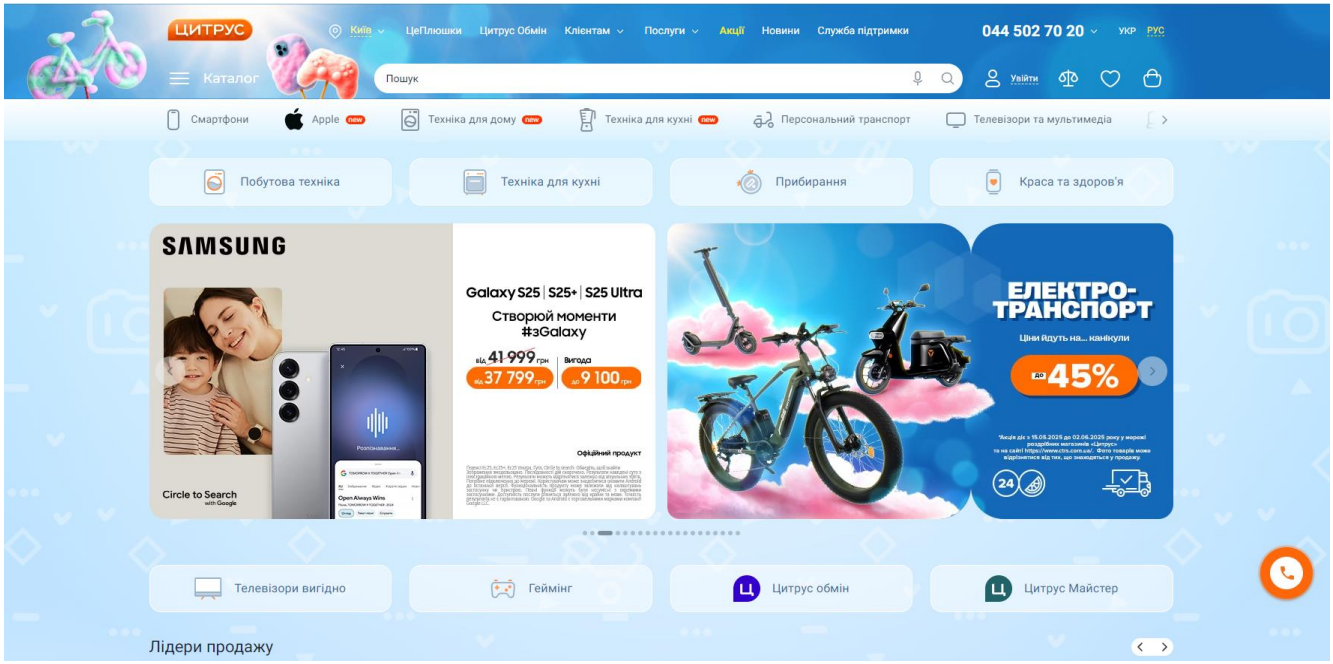


Рисунок 1.2 – Головна сторінка веб-магазину Цитрус

Серед світових платформ Newegg є одним із найбільших спеціалізованих інтернет-магазинів, орієнтованих саме на продаж комп'ютерних комплектуючих та електроніки. Це американська платформа, що обслуговує клієнтів у різних країнах. Її головною особливістю є поглиблена категоризація: на сайті представлено сотні підкатегорій, фільтрів, брендів і конфігурацій, що робить пошук максимально гнучким. Крім того, Newegg активно використовує мультимовність, підтримку різних валют, локалізовані методи оплати та доставки. На платформі доступна можливість створення конфігурацій ПК, де користувач може вибрати компоненти, що підходять одне до одного, і зразу побачити загальну суму. На рисунку 1.3 зображено інтерфейс вибору компонентів для індивідуальної збірки ПК у веб-магазині Newegg. Також на сайті розміщено багато технічних блогів, відеоогляді, таблиць сумісності, що підвищує цінність ресурсу не тільки як магазину, але і як інформаційної бази.

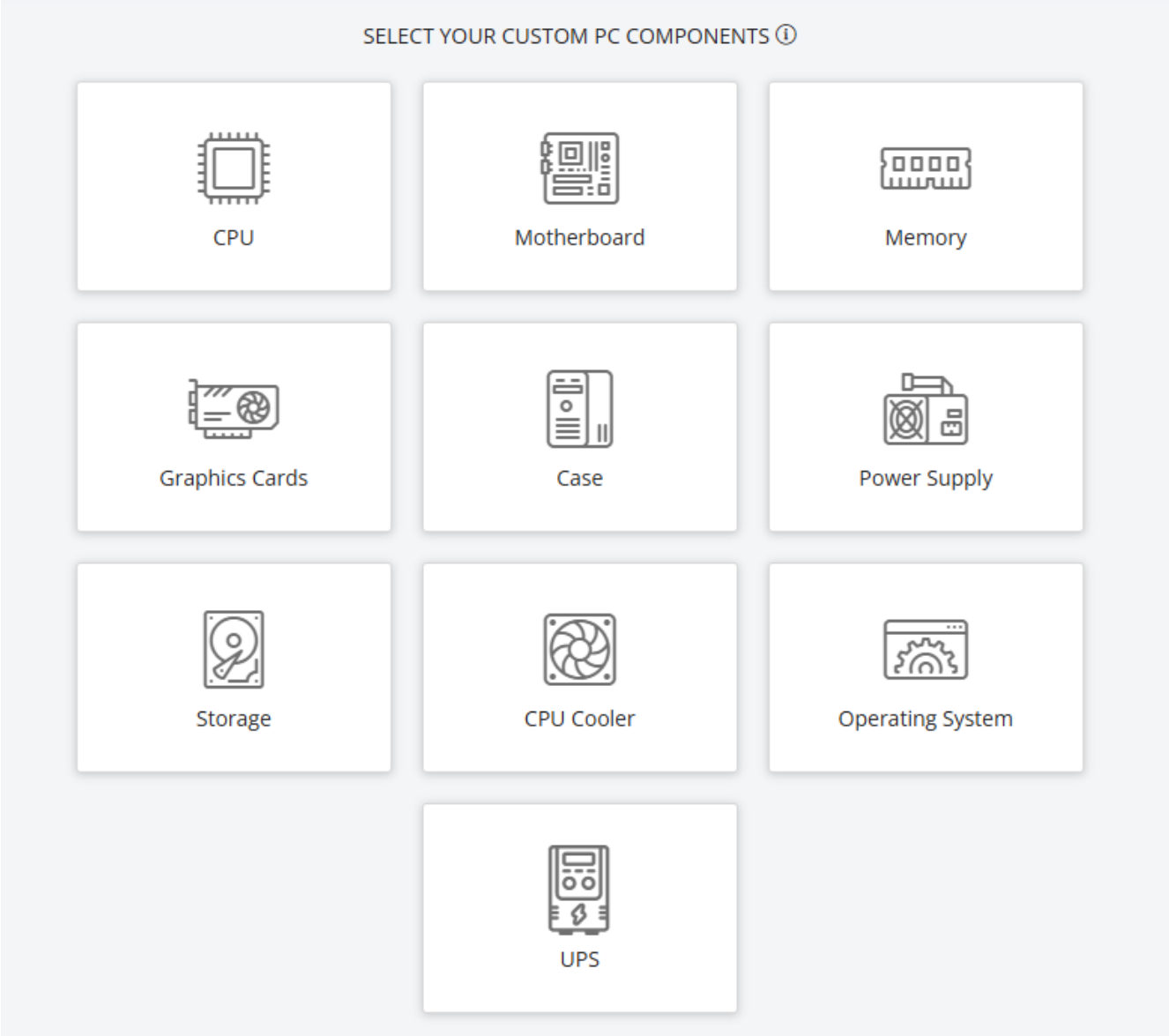


Рисунок 1.3 – Інтерфейс вибору компонентів для індивідуальної збірки ПК на веб-сайті Newegg

Цей аналіз веб-магазинів дозволяє сформувати загальні вимоги до сучасної платформи з продажу комплектуючих ПК: адаптивний інтерфейс, багаторівнева система фільтрації, інформативна картка товару, різноманіття методів оплати, технічна підтримка користувачів, дуже висока швидкість завантаження, інтеграція з логістичними службами, а також інструменти персоналізації та рекомендацій.

1.5 Тренди у веб-розробці для інтернет-магазинів

У сфері веб-розробки електронної комерції щороку з’являються нові підходи, фреймворки, архітектурні рішення та інструменти. Знання сучасних тенденцій дозволяє створювати більш адаптивні, масштабовані, безпечні та конкурентоспроможні веб-магазини. У цьому підрозділі розглянуто основні технологічні тренди, що мають практичне значення для реалізації сучасних веб-проєктів у сфері онлайн-торгівлі.

Одним із провідних напрямів є застосування архітектури Headless CMS, яка дозволяє розділити фронтенд і бекенд частини веб-застосунку. Це дає змогу гнучко змінювати зовнішній інтерфейс без впливу на логіку обробки даних, що особливо важливо для адаптації веб-магазинів під мобільні застосунки або PWA (Progressive Web App). Такі CMS, як Strapi, Contentful або Sanity, активно використовуються як backend-рішення для інтернет-магазинів.

Також набуває популярності концепція JAMstack — це підхід, що поєднує JavaScript, API та розмітку (Markup). Завдяки попередній генерації сторінок (pre-rendering) та винесенню динамічних даних у зовнішні API, сайти стають надзвичайно швидкими, безпечними та масштабованими. JAMstack ідеально підходить для інтернет-магазинів, де важлива швидкість завантаження, SEO-оптимізація та безперервність роботи.

Ще одним важливим напрямом є застосування штучного інтелекту та машинного навчання. Вони дають змогу реалізувати персоналізовані рекомендації товарів, інтелектуальні чат-боти для підтримки, прогнозування попиту та аналіз поведінки користувачів. Наприклад, за допомогою AI можна адаптувати вміст головної сторінки відповідно до інтересів відвідувача, що підвищує конверсію та середній чек покупки.

Широке розповсюдження отримали інструменти CI/CD (Continuous Integration Continuous Deployment), які дозволяють автоматизувати процес розгортання та тестування веб-застосунків. Такі сервіси, як GitHub Actions, GitLab CI, Netlify або Vercel, забезпечують швидке впровадження оновлень,

зменшуючи ризик помилок при розробці. Це особливо актуально для інтернет-магазинів, де критично важливо зберігати безперервну доступність ресурсу.

Значну роль також відіграє концепція PWA — прогресивних веб-додатків. Вони поєднують переваги класичних веб-сайтів і мобільних застосунків: швидкість, офлайн-доступ, push-повідомлення, можливість встановлення на головний екран. У сфері електронної комерції це дозволяє залучити більше мобільних користувачів без необхідності створення окремого застосунку.

Не менш важливою є тенденція переходу на безсерверні архітектури (serverless), коли частину бізнес-логіки обробляють хмарні функції (наприклад, AWS Lambda, Azure Functions). Це дозволяє економити ресурси, спрощує масштабування та забезпечує високу надійність системи.

Застосування цих трендів дозволяє не лише створювати технічно досконалі веб-рішення, але й покращити користувацький досвід, підвищити безпеку, швидкість та доступність веб-магазину. Врахування новітніх тенденцій на етапі проєктування дає змогу створити веб-платформу, здатну конкурувати на сучасному цифровому ринку та адаптуватися до змін у поведінці споживачів і розвитку технологій.

1.6 Огляд комп’ютерних комплектуючих

1.6.1 Процесори (CPU)

Центральний процесор (Central Processing Unit, або CPU) є важливим обчислювальним елементом персонального комп’ютера, що виконує більшість обчислень і керує роботою інших компонентів. Саме процесор обробляє інструкції програм, контролює потоки даних, проводить логічні й арифметичні операції, а також відповідає за загальну швидкодію системи. Якість і потужність процесора безпосередньо впливають на продуктивність ПК у різноманітних задачах — від веб-серфінгу й офісної роботи до професійної обробки відео чи сучасних відеоігор.

На відміну від старих однокристальних чипів, сучасні CPU мають кілька ядер, що працюють паралельно. Це означає, що один фізичний чип містить кілька ядер, кожне з яких може працювати незалежно або паралельно з іншими, обробляючи окремі потоки даних. Наприклад, чотириядерний процесор здатен одночасно виконувати декілька задач, що істотно підвищує загальну ефективність комп'ютера. Крім фізичних ядер, деякі моделі підтримують технологію багатопоточності (наприклад, Intel Hyper-Threading), яка дозволяє одному ядру обробляти два потоки одночасно.

Основними виробниками центральних процесорів у сучасній індустрії є компанії Intel і AMD. Intel традиційно асоціюється з високою енергоефективністю та надійною продуктивністю в офісному й мобільному сегментах. AMD, у свою чергу, останніми роками демонструє значне зростання популярності завдяки своїй лінійці Ryzen, яка поєднує в собі високу продуктивність, хорошу енергоефективність та конкурентну ціну, що робить ці процесори привабливими для ігрових ПК та робочих станцій.

Ключові технічні характеристики процесора включають:

- Тактову частоту — швидкість виконання інструкцій, вимірюється в гігагерцах (ГГц);
- Кількість ядер і потоків — показник багатозадачності;
- Об'єм кеш-пам'яті — проміжна пам'ять для швидкого доступу до даних;
- Тепловий пакет (TDP) — кількість тепла, яку процесор виділяє при роботі, важливий параметр для вибору системи охолодження;
- Сокет (Socket) — фізичний роз'єм, через який процесор встановлюється на материнську плату. Важливо забезпечити сумісність процесора і материнської плати.

Вибір процесора багато в чому залежить від цільового використання комп'ютера. Для виконання легких завдань, як-от перегляд документів, електронної пошти чи робота з текстом, буде достатньо моделей із двома або чотирма ядрами, що мають низьке енергоспоживання. Для ігрових систем або

редагування мультимедійного контенту необхідні продуктивніші процесори з шістьма, вісьмома або більше ядрами та високою тактовою частотою. У сегменті професійної роботи, наприклад, 3D-моделювання чи машинного навчання, використовуються багатоядерні процесори з розширеними функціями обробки.

1.6.2 Відеокарти (GPU)

Графічний процесор (GPU) один із ключових та найважливіших компонентів сучасного ПК, який має виконувати функції обробки графічних даних та виведення зображень на монітор. Основним завданням GPU є рендеринг 2D та 3D-графіки, відтворення відео високої чіткості, а також виконання складних обчислювальних задач, які безпосередньо пов’язані з візуалізацією. На противагу центральному процесору, який розрахований на виконання різнотипних і послідовних обчислень, графічний процесор спеціалізується на паралельній обробці величезної кількості однотипних операцій. Завдяки цій архітектурі GPU є надзвичайно ефективним у задачах, що потребують масового паралелізму, таких як обробка зображень, застосування графічних ефектів, анімація та симуляції.

Існують два основні типи графічних рішень, що використовуються в комп’ютерних системах — інтегровані (вбудовані) та дискретні відеокарти. Інтегровані графічні процесори розміщуються на центральному процесорі або на материнській платі, що дозволяє набагто знизити загальну вартість системи і зменшити енергоспоживання. Такі рішення найкраще підходять для базових задач, таких як перегляд відео, робота з документами, веб-серфінг та невимогливі ігри. Основною перевагою інтегрованих GPU є їхня енергоефективність, низька ціна і компактність, що робить їх оптимальним вибором для ноутбуків, бюджетних ПК та офісних комп’ютерів. Проте інтегровані відеокарти мають обмежені можливості, зокрема вони зазвичай використовують оперативну

пам’ять системи, що суттєво знижує їх продуктивність у порівнянні з дискретними відеокартами.

Дискретні відеокарти являють собою окремі апаратні модулі, які підключаються до материнської плати через високошвидкісний інтерфейс PCI Express. Вони мають власну відеопам’ять (VRAM), що дозволяє працювати незалежно від системної оперативної пам’яті і забезпечує значно більшу пропускну здатність при обробці графічних даних. Обсяг і тип відеопам’яті безпосередньо впливають на якість виведеного зображення, здатність відтворювати високодеталізовані текстури, складні тіні, освітлення та інші візуальні ефекти. Сучасні дискретні відеокарти можуть бути оснащені від 4 до 24 і більше гігабайт високошвидкісної VRAM (наприклад, GDDR6, GDDR6X, HBM2), що робить їх недоторканими не тільки для геймерів, але й для фахівців у галузі комп’ютерної графіки, анімації, відеомонтажу та інших професійних задач.

Найбільш відомими і поширеними виробниками дискретних відеокарт є компанії NVIDIA і AMD, які займають передові позиції на світовому ринку. NVIDIA пропонує серії відеокарт GeForce, орієнтовані на геймерів і звичайних користувачів, а також професійні рішення Quadro (нині відомі як NVIDIA RTX) для CAD/CAM-проектування, 3D-моделювання, візуалізації та наукових симуляцій. AMD, у свою чергу, виробляє лінійки Radeon, які мають успіх як серед геймерів, так і серед професіоналів, а також пропонує Radeon Pro для робочих станцій. Обидві компанії активно розробляють драйвери і програмне забезпечення, яке оптимізує роботу їхніх продуктів, забезпечує сумісність з сучасними іграми і додатками, а також розширені функції, такі як трасування променів, технології адаптивної синхронізації та інші.

Окрім традиційних графічних завдань, сучасні GPU все більше використовуються для обчислень загального призначення (GPGPU). Це стало можливим завдяки розвитку паралельних обчислень і появі відповідних платформ, таких як CUDA від NVIDIA та OpenCL, які дозволяють

використовувати обчислювальні ресурси відеокарти для задач штучного інтелекту, машинного навчання, обробки великих даних, наукових досліджень і симуляцій. Наприклад, популярні фреймворки машинного навчання, такі як TensorFlow та PyTorch, активно використовують GPU для навчання нейронних мереж, що значно прискорює процес обробки у порівнянні із використанням центрального процесора.

Ключовою частиною ефективного використання GPU є підтримка сучасних графічних API, таких як DirectX, OpenGL, Vulkan, а також специфічних технологій на кшталт CUDA для NVIDIA. Ці інтерфейси забезпечують розробникам програмного забезпечення можливість максимально ефективно використовувати обчислювальні ресурси графічних процесорів, створювати якісну графіку і складні візуальні ефекти, а також підтримувати сумісність із широким спектром пристроїв.

При виборі відеокарти необхідно враховувати цілу низку технічних параметрів, серед яких: обсяг і тип відеопам’яті (GDDR5, GDDR6, HBM), тактова частота графічного ядра, кількість обчислювальних блоків, архітектуру GPU, тип системи охолодження, а також сумісність із материнською платою, процесором та корпусом комп’ютера. Дуже важливо оцінити потреби користувача — від офісних завдань до складних професійних робіт або ігор із високими вимогами до графіки.

Отже, графічний процесор є не лише засобом виведення зображень на екран, а й потужним багатофункціональним інструментом, здатним вирішувати широкий спектр задач, пов’язаних із графікою та обчисленнями. Його правильний вибір і оптимальна інтеграція в систему мають стратегічне значення для побудови сучасного персонального комп’ютера, особливо в тих випадках, коли ключовими факторами є продуктивність, якість графіки та ефективність виконання складних обчислювальних процесів.

1.6.3 Оперативна пам’ять (RAM)

Оперативна пам’ять, або RAM (англ. Random Access Memory), є одним із найважливіших компонентів комп’ютерної системи, який несе відповідальність за короткотривале зберігання даних, що активно використовуються під час роботи комп’ютера. Основна роль оперативної пам’яті полягає в забезпеченні швидкого доступу до інформації, необхідної для виконання програм. На відміну від постійної пам’яті, такої як жорсткий диск (HDD) чи твердотільний накопичувач (SSD), RAM зберігає дані лише протягом часу роботи пристрою — як тільки живлення вимикається, вся інформація, яка знаходиться в оперативній пам’яті, автоматично стирається. Саме ця особливість робить RAM тимчасовим сховищем, що істотно впливає на продуктивність та швидкість роботи комп’ютера.

Функціонування оперативної пам’яті має безпосередній вплив на загальну продуктивність системи, оскільки саме RAM визначає обсяг інформації, яку комп’ютер може обробляти одночасно, не звертаючись до повільніших засобів збереження даних. При недостатньому обсязі оперативної пам’яті система змушена використовувати файл підкачки або сторінковий файл, який розташовується на жорсткому диску або SSD і значно повільніший, що веде до зниження загальної швидкодії. Це особливо помітно при виконанні ресурсоємних завдань — наприклад, при обробці відео, редагуванні високоякісних зображень, запуску віртуальних машин, великих баз даних або сучасних ігрових застосунків з високими вимогами до ресурсів.

Крім обсягу оперативної пам’яті, важливими характеристиками є також частота роботи модулів, пропускна здатність та таймінги. Частота визначає, з якою швидкістю модулі RAM можуть обмінюватися даними з центральним процесором і іншими компонентами системи. Пропускна здатність — це обсяг даних, який може бути переданий за одиницю часу, і вона залежить від частоти та ширини шини пам’яті. Таймінги (затримки) — це час, який потрібен для виконання певних операцій обміну даними, і чим вони менші, тим краще. Тому

при виборі оперативної пам’яті варто враховувати не лише її обсяг, а й якість і характеристики, що впливають на загальну продуктивність системи.

На сьогодні серед найпоширеніших стандартів оперативної пам’яті виділяють DDR4 і DDR5. Стандарт DDR4 є широко розповсюдженим у більшості сучасних комп’ютерних систем, включаючи як десктопи, так і ноутбуки, і забезпечує достатній рівень швидкодії для більшості завдань — від офісної роботи до ігор середнього рівня та мультимедійної обробки. DDR4 підтримує частоти від 1600 до 3200 МГц і володіє оптимальним співвідношенням ціни і продуктивності, що робить її доступною і зручною для широкого кола користувачів.

Проте з розвитком комп’ютерних технологій і збільшенням вимог до продуктивності, набирає популярність новіший стандарт DDR5. Він характеризується значно більшою пропускною здатністю, вищими тактовими частотами (починаючи з 4800 МГц і вище), а також покращеною енергоефективністю завдяки зниженій напрузі живлення. DDR5 також підтримує більший обсяг пам’яті на один модуль, що дозволяє збільшувати загальний обсяг оперативної пам’яті у системі до неймовірних значень, необхідних для високопродуктивних робочих станцій, серверів і топових ігрових ПК. Завдяки цим характеристикам DDR5 є ідеальним вибором для тих, хто працює з великими обчислювальними навантаженнями, вимагає максимальної швидкості обробки даних або планує майбутнє оновлення системи.

Варто також зазначити, що при виборі оперативної пам’яті необхідно ретельно враховувати її сумісність із материнською платою та процесором. Не всі плати підтримують новітні стандарти пам’яті, а деякі процесори можуть обмежувати максимально допустиму частоту роботи модулів RAM. Тому перед придбанням оперативної пам’яті варто уважно ознайомитись із технічними характеристиками платформи, переглянути рекомендації виробників материнської плати та процесора, а також перевірити можливість оновлення BIOS — це іноді критично для стабільної роботи сучасних модулів. Окрему

увагу слід звернути на форм-фактор: модулі типу DIMM використовуються в настільних ПК, тоді як SO-DIMM — у ноутбуках та компактних системах.

Оперативна пам’ять є одним із ключових компонентів комп’ютера, що безпосередньо впливає на загальну швидкодію та ефективність системи. З огляду на зростання обсягів оброблюваних даних і ускладнення програмного забезпечення, роль оперативної пам’яті лише посилюється. Дивлячись на нинішні вимоги до продуктивності, треба орієнтуватися можливість подальшого розширення обсягу пам’яті, що дозволить забезпечити довготривалу актуальність і гнучкість конфігурації системи.

1.6.4 Материнські плати

Материнська плата (англ. motherboard) виконує роль центрального елемента апаратної архітектури комп’ютера, об'єднуючи між собою всі ключові компоненти як на фізичному рівні, так і через системні інтерфейси. Саме на ній розташовані центральний процесор (CPU), модулі оперативної пам’яті (RAM), графічний процесор (GPU), накопичувачі інформації, система живлення, а також інтерфейси для підключення периферійних пристроїв. Материнська плата виконує ключову роль не лише у забезпеченні електроживлення для всіх цих компонентів, але й відповідає за організацію їхньої взаємодії через спеціальні системні шини, контролери та мости, що дозволяють ефективно передавати дані між різними компонентами ПК.

Однією з найважливіших характеристик материнської плати є тип роз’єму процесора, або сокета, який визначає сумісність плати з конкретними моделями процесорів. Відомі типи сокетів, такі як Intel LGA1700 або AMD AM5, підтримують лише відповідні покоління процесорів. Це означає, що при збірці комп’ютера необхідно ретельно узгодити вибір материнської плати з процесором, аби уникнути проблем із сумісністю. Також дуже важливим

аспектом є підтримка стандарту оперативної пам’яті, наприклад DDR4 або DDR5, а також максимально допустимий обсяг і частота модулів RAM, що безпосередньо має вплив на продуктивність системи та швидкість обробки даних.

Материнська плата обладнана різноманітними слотами і портами, що забезпечують підключення як внутрішніх, так і зовнішніх пристроїв. Основним інтерфейсом для розширення є слоти PCI Express (PCIe), які дозволяють підключати відеокарти, мережеві адаптери, звукові карти та інші розширювальні модулі. Для накопичувачів інформації застосовуються різні типи роз’ємів: традиційні SATA III забезпечують підключення жорстких дисків і SSD, а більш сучасні інтерфейси M.2 призначені для підключення швидких твердотільних накопичувачів NVMe, що значно підвищує швидкість обміну даними. Крім того, материнська плата містить численні порти USB для зовнішніх пристроїв, аудіоінтерфейси для підключення звукової апаратури, мережеві роз’єми Ethernet для підключення до локальних і глобальних мереж, а також інші роз’єми для комп’ютерної периферії.

Розмір і форма материнської плати, відомі як форм-фактор, досить важлива роль у визначенні її сумісності з корпусом комп’ютера та кількості підтримуваних слотів і портів. Основні форм-фактори — це ATX, Micro-ATX і Mini-ITX. Форм-фактор ATX є найбільш поширеним і універсальним, підходить для створення потужних ігрових та робочих систем, оскільки забезпечує достатньо місця для розміщення великої кількості слотів розширення і роз’ємів. Micro-ATX має менші розміри, що робить його зручним для компактних систем, але з меншою кількістю розширювальних можливостей. Mini-ITX є найменшим форм-фактором і використовується у компактних, часто мобільних рішеннях, хоча кількість слотів і портів у таких платах значно обмежена.

Вибір конкретної моделі материнської плати повинен базуватися на запланованому сценарії використання комп’ютера. Наприклад, для ігрових систем або робочих станцій, які виконують обробку відео чи графіки, доцільно

обирати материнські плати з потужною системою охолодження, підтримкою технологій розгону процесора і оперативної пам’яті, а також з великою кількістю слотів для встановлення додаткових карт розширення. Водночас для офісних або побутових комп’ютерів достатньо обрати доступну модель з базовим набором функцій, що забезпечить стабільну і ефективну роботу без зайвих витрат. Зважаючи на сучасні вимоги до продуктивності, важливо враховувати не лише сумісність, а й потенціал для апгрейду, навіть у звичайних системах.

Материнська плата є ключовим компонентом, який визначає сумісність усієї системи, можливість її подальшого оновлення і розширення. Від правильного вибору плати залежить не лише поточна продуктивність, а й перспективи розвитку комп’ютера. Тому при виборі материнської плати варто детально вчити технічні характеристики, сумісність із процесором і пам’яттю, а також потреби користувача, щоб у відповідь отримати надійну та ефективну роботу системи протягом тривалого часу.

1.6.5 Накопичувачі даних (SSD та HDD)

Накопичувачі даних є важливою складовою будь-якого комп’ютера, оскільки вони відповідають за зберігання операційної системи, програмного забезпечення, документів, мультимедійних файлів та інших даних користувача.

Станом на сьогоднішній день найбільш розповсюдженими видами накопичувачів даних залишаються HDD (жорсткі диски) та SSD (твердотільні накопичувачі). Жорсткі диски мають механічну конструкцію: дані зчитуються та записуються за допомогою рухомої головки, яка працює з магнітними дисками. Основною перевагою HDD є велика ємність за маленькою ціною, що робить їх привабливими для зберігання великого обсягу інформації, наприклад, фільмів, архівів або резервних копій.

У свою чергу, SSD працюють на основі флеш-пам’яті та не мають рухомих частин. Завдяки цьому вони забезпечують набагато вищу швидкість доступу до

даних, а також меншу затримку при читанні й запису. Це дозволяє значно скоротити час завантаження операційної системи, запуску програм і загальної реакції системи. Твердотільні накопичувачі найчастіше підключаються до комп’ютера через інтерфейси SATA або більш сучасний і швидкісний NVMe, який використовує шину PCIe.

Вибір між HDD та SSD залежить, перш за все, від потреб користувача та бюджету. У багатьох сучасних системах використовують комбінований підхід: SSD — для встановлення операційної системи та важливих програм, а HDD — для зберігання великої кількості файлів. У таблиці 1.1 наведено порівняння характеристик HDD та SSD.

Таблиця 1.1 – Порівняння характеристик HDD та SSD

Параметр	HDD	SSD
Тип пам’яті	Магнітні диски	Флеш-пам’ять
Рухомі частини	Так	Ні
Швидкість передачі даних	~100-150 МБ/с	SATA SSD: ~500 МБ/с M.2 NVMe: до 3500+ МБ/с
Стійкість до пошкоджень	Низька	Висока
Рівень шуму	Досить шумний	Безшумний
Типове використання	Архіви, великі файли	Операційна система, програми, ігри

2 АНАЛІЗ ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ТА ШЛЯХИ СТВОРЕННЯ ВЕБ-САЙТУ

2.1 Аналіз використаних технологій у проекті

2.1.1 Основи структурування та стилізації веб-сторінок

Мова розмітки HTML (HyperText Markup Language) відповідає за побудову структурного каркаса веб-сторінки Завдяки HTML можна логічно розбити сторінку на семантичні блоки, що полегшує сприйняття вмісту як користувачами, так і пошуковими системами. В сучасному веб-розробленні застосовується стандарт HTML5, який містить нові семантичні теги та покращує доступність і стандартизацію.

В данному проєкті було використано широкий спектр HTML5-тегів для організації контенту:

- <!DOCTYPE html> — оголошення типу документа, що вказує браузеру використовувати останній стандарт HTML5.
- <html lang="uk"> — кореневий тег сторінки з атрибутом, який задає мову вмісту.
- <head> — містить метадані сторінки: кодування (<meta charset="UTF-8">), адаптивність (<meta name="viewport">), підключення стилів (<link rel="stylesheet">), а також заголовок сторінки (<title>).
- <body> — основний вміст, який відображається користувачу.
- Семантичні блоки: <header>, <nav>, <section>, <main>, <footer> використовуються для структурування сайту відповідно до логіки.
- Контейнерні теги і застосовуються для групування та стилізації елементів, які не мають власної семантики.
- Текстові теги: заголовки (<h1>, <h2>), параграфи (<p>), списки (<ul>, <li>), посилання (<a>), зображення (<img>), поля вводу (<input>) та кнопки

(<button>).

Застосування семантичних тегів покращує доступність сайту для користувачів із обмеженими можливостями, а також підвищує SEO-оптимізацію, оскільки пошукові системи краще індексують чітко структуровану інформацію.

CSS (Cascading Style Sheets) відповідає за візуальне оформлення веб-сторінок. За допомогою CSS визначаються кольори, шрифти, розміщення елементів, відступи, межі та інші стилістичні параметри.

У проєкті було застосовано сучасний підхід до стилізації, що включає:

- Адаптивний дизайн: використання медіа-запитів (media queries) дозволяє підлаштовувати відображення сайту під різні розміри екранів — від мобільних телефонів до великих моніторів. Це забезпечує комфорт користувачам на будь-яких пристроях.
- Flexbox і Grid: сучасні CSS-модулі для гнучкого і структурованого розміщення блоків. Flexbox застосовується для розташування навігаційних елементів у горизонтальний ряд, Grid — для формування сітки товарів із автоматичним підлаштуванням під ширину вікна.
- Кольорова палітра: поєднання темних відтінків синього та сірого фону з яскравими зеленими та червоними акцентами на кнопках і знижках створює контрастний і привабливий дизайн.
- Візуальні ефекти: тіні, скруглені кути, ефекти наведення курсора (hover), плавні переходи підвищують естетику і покращують взаємодію користувача з інтерфейсом.

Дотримання стандартів і правильна організація CSS-файлів сприяють легкості підтримки проекту, а також зменшенню часу завантаження сторінок за рахунок мінімізації і кешування стилів. Важливим аспектом є також забезпечення кросбраузерної сумісності, що гарантує коректне відображення сайту у провідних браузерах. Для цього застосовуються префікси для CSS-властивостей та тестування у різних середовищах.

HTML та CSS формують фундамент будь-якого веб-сайту, забезпечуючи його структуру та привабливий вигляд.

2.1.2 Динаміка та інтерактивність

JavaScript є найпопулярнішою мовою програмування для створення динамічного і інтерактивного веб-контенту. Його основна функція — забезпечувати логіку роботи веб-сайту на стороні клієнта, реагувати на дії користувача, взаємодіяти з сервером та змінювати вміст сторінки без перезавантаження.

Завдяки цій мові програмування реалізується логіка, яка робить сайт «живим»: він реагує на події користувача, оновлює дані в режимі реального часу, змінює вміст DOM (Document Object Model) та забезпечує загальну динаміку інтерфейсу. На рисунку 2.1 зображено схему, що демонструє, як JavaScript взаємодіє з DOM та як інтерфейс браузера оновлюється внаслідок дій користувача.

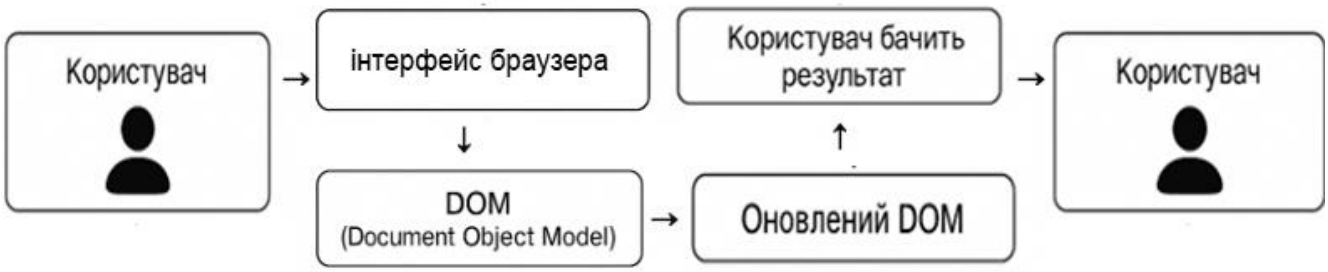


Рисунок 2.1 — Принцип взаємодії JavaScript та DOM у браузері

У проєкті JavaScript виконує ключові завдання. За його допомогою реалізовано відображення списку товарів, який динамічно завантажується з сервера через API-запити (AJAX або fetch). Це дозволяє демонструвати актуальний асортимент і оновлювати дані без повного перезавантаження

сторінки, що позитивно впливає на швидкість роботи сайту та зручність користування.

Також передбачено функцію фільтрації, яка надає можливість вводити назву товару або встановлювати межі цінового діапазону. JavaScript виконує фільтрацію відповідно до зазначених параметрів. Частина логіки працює на клієнтській стороні для забезпечення оперативного реагування, при цьому реалізована серверна перевірка для точності отриманих результатів.

Для максимальної зручності користувачів було успішно впроваджено інтуїтивно зрозумілий віртуальний кошик, що дозволяє легко додавати, видаляти та оперативно змінювати кількість бажаних товарів, надійно зберігаючи їх поточний стан у localStorage для безперервності покупок, що не зникнуть навіть після оновлення сторінки чи повторного входу на сайт.

Крім того, JavaScript забезпечує обробку процесу входу та оформлення замовлення безпосередньо на клієнтській стороні, що дозволяє значно підвищити швидкодію та комфорт взаємодії користувача із сайтом. Завдяки цьому знижується навантаження на сервер, а у випадку коректного введення даних інформація оперативно передається на сервер для подальшої обробки та збереження.

2.1.3 Серверна платформа веб-магазин

У процесі реалізації серверної частини веб-сайту було обрано платформу Node.js, яка забезпечує виконання JavaScript-коду поза межами браузера. Це середовище побудоване на рушії V8 від Google і відоме своєю швидкодією, асинхронною обробкою запитів та широкими можливостями масштабування. Саме завдяки цим характеристикам Node.js ідеально підходить для веб-застосунків, орієнтованих на велику кількість одночасних запитів.

У розробленому проєкті Node.js використовується для обробки клієнтських запитів, взаємодії з базою даних MySQL, а також реалізації логіки, пов’язаної з аутентифікацією користувачів, обробкою замовлень та управлінням каталогом товарів. Застосування JavaScript як на клієнтській, так і на серверній стороні дозволяє підтримувати єдину мову програмування в усьому стеку, що суттєво спрощує синхронізацію логіки між фронтендом і бекендом, а також покращує читаність і підтримку коду.

Для реалізації веб-сервера був використаний фреймворк Express.js, що спрощує обробку HTTP-запитів, маршрутизацію, а також дозволяє ефективно використовувати проміжні обробники (middleware) для валідації, логування чи обробки помилок. Зв’язок із базою даних було організовано через ORM-бібліотеку Sequelize, яка надає можливість працювати з MySQL на об'єктному рівні, приховуючи складність SQL-запитів і водночас дозволяючи будувати гнучкі моделі даних.

Серверна логіка сайту забезпечує швидке та стабільне опрацювання запитів, передачу даних у форматі JSON, інтеграцію з клієнтською частиною та масштабованість функціональності в майбутньому.

2.1.4 Система керування базами даних MySQL

У межах реалізації проєкту для зберігання товарів використано реляційну систему керування базами даних MySQL, яка є стабільним, надійним і широко підтримуваним рішенням для веб-застосунків. Обрана технологія дозволяє ефективно працювати зі структурованою інформацією та забезпечує високу швидкодію при взаємодії з великим каталогом продукції.

Серед альтернативних рішень також розглядаються PostgreSQL — потужна реляційна СКБД із розширеними можливостями для складних транзакцій та перевірок, — і MongoDB, яка належить до документоорієнтованих

баз даних і краще підходить для проєктів із гнучкою або непостійною структурою даних. Детальне порівняння особливостей цих систем наведено у таблиці 2.1

Таблиця 2.1 – порівняння MySQL, PostgreSQL та MongoDB

Характеристика	MySQL	PostgreSQL	MongoDB
Тип	Реляційна	Реляційна	Документоорієнтована
Структура даних	Таблиці SQL	Таблиці SQL + розширення	Документи (BSON/JSON)
Масштабованість	Вертикальна	Вертикальна	Горизонтальна
Транзакції	Так	Так (ACID повністю)	Обмежено
Гнучкість структури	Низька	Середня	Висока
Складність налаштування	Низька	Середня/Висока	Середня

Підключення до бази даних MySQL реалізовано за допомогою бібліотеки Sequelize — ORM (Object-Relational Mapping), яка дозволяє працювати з базою у вигляді JavaScript-об'єктів. Замість написання SQL-запитів вручну, структура таблиць детально описується в наведеному коді відображеному на рисунку 2.2

```
1   const { DataTypes } = require('sequelize');
2   const sequelize = require('../connect-to-db');
3
4   const Product = sequelize.define('Product', {
5     name: { type: DataTypes.STRING, allowNull: false },
6     imgUrl: { type: DataTypes.STRING, allowNull: false },
7     price: { type: DataTypes.FLOAT, allowNull: false },
8     discountPrice: { type: DataTypes.FLOAT, allowNull: true },
9     description: { type: DataTypes.TEXT, allowNull: false },
10  });
11
12  module.exports = Product;
```

Рисунок 2.2 – Структура таблиці "Products" у базі даних

Використання Sequelize дає змогу швидко створювати моделі, синхронізувати структуру бази даних і працювати з нею без потреби писати SQL-запити вручну. У проєкті база містить таблицю з інформацією про товари: назва, опис, ціна, знижена ціна, посилання на зображення. Дані використовуються для формування каталогу, відображення позицій на головній сторінці та реалізації пошуку.

Запити до бази виконуються зі сторони сервера на Node.js, після чого результати передаються у клієнтську частину у форматі JSON. Такий підхід забезпечує динамічну взаємодію між клієнтом і сервером та дає змогу оновлювати вміст сторінки без перезавантаження. Наповнення товарів здійснювалося через функцію сидування, яка автоматично додає готовий список продукції до бази при запуску.

Додатковою перевагою реалізованого рішення є використання SSL-з'єднання для підключення до бази даних, що забезпечує захищену передачу даних. У цілому, MySQL у поєднанні з Node.js та Sequelize дозволяє створити надійний і масштабований механізм зберігання інформації, який легко адаптується до потреб інтернет-магазину та може бути розширений у майбутньому — наприклад, для обліку користувачів, замовлень або відгуків. Така структура є оптимальною для невеликих та середніх комерційних веб-рішень, забезпечуючи баланс між продуктивністю, простотою реалізації та функціональністю.

2.1.5 Зберігання локальних даних

Однією з ключових особливостей клієнтської частини веб-сайту є використання технології локального збереження даних у браузері користувача, що реалізовано через Web API localStorage. Цей інструмент дозволяє зберігати інформацію на боці клієнта без встановленого строку дії, що дає змогу

підтримувати стан користувача навіть після оновлення сторінки чи повторного відкриття сайту.

У межах реалізованого проєкту localStorage використовується для кількох основних задач. Насамперед — для збереження стану кошика, тобто списку доданих товарів, їх кількості та інших параметрів, які користувач вибирає під час формування замовлення. Крім того, за допомогою цього механізму зберігається базова інформація про користувача, зокрема ім’я, введене при вході, що дозволяє персоналізувати інтерфейс без повторного введення даних. Також у локальній пам’яті зберігаються вибрані налаштування фільтрів, завдяки чому після оновлення або повернення на сайт користувач бачить саме той набір товарів, який його цікавив.

Використання localStorage дозволяє значно зменшити кількість звернень до сервера, що покращує продуктивність та загальну швидкість взаємодії з веб-додатком. Завдяки простому програмному інтерфейсу на JavaScript збереження та отримання даних реалізуються швидко й ефективно. Разом із тим, важливо розуміти, що обсяг localStorage обмежується приблизно 5 мегабайтами на домен, а всі збережені дані доступні лише на стороні клієнта. Це означає, що зберігати конфіденційну або чутливу інформацію у цьому середовищі не рекомендується, оскільки вона може бути прочитана іншими скриптами у браузері.

З технічної точки зору, localStorage відрізняється від альтернативних технологій, таких як sessionStorage або cookies. Якщо перший зберігає дані лише на час однієї сесії, а другий постійно передає їх із кожним запитом до сервера, то localStorage залишається ефективним рішенням для локального кешування некритичної інформації, що використовується в інтерфейсі.

У контексті даного проєкту застосування localStorage є обґрунтованим і доцільним: з його допомогою реалізовано ключову функціональність для збереження кошика, персоналізації взаємодії з користувачем та підтримки налаштувань фільтрації, що покращує загальний користувацький досвід та

підвищує зручність користування веб-сайтом без додаткового навантаження на сервер.

2.2 Архітектура та шляхи реалізації веб-сайту

2.2.1 Клієнтська частина: взаємодія та інтерфейс

Клієнтська частина веб-сайту є безпосереднім середовищем взаємодії з користувачем і виконує ключову роль у забезпеченні зручності, швидкодії та функціональності інтернет-магазину. Візуальне оформлення, логіка поведінки сторінки та можливість динамічної роботи з даними — усе це визначає загальне враження користувача й ефективність роботи ресурсу в цілому.

Архітектура фронтенду побудована за принципом односторінкового застосунку (SPA), що дозволяє уникати повного перезавантаження сторінок при переході між розділами. Завдяки цьому користувач отримує плавний досвід роботи з інтерфейсом — контент оновлюється миттєво, а взаємодія з елементами сайту не супроводжується помітними затримками. HTML-структура сторінки формує каркас інтерфейсу з розмежуванням основних зон, таких як навігація, головна частина та футер. Систему стилів реалізовано через CSS із застосуванням медіа-запитів, що дозволяє забезпечити адаптивне відображення сайту на різних пристроях — від смартфонів до настільних ПК.

Клієнтська логіка реалізована на мові JavaScript, яка відповідає за обробку подій, управління контентом, фільтрацію товарів, роботу кошика та валідацію форм. Усі взаємодії з DOM-структурою відбуваються в режимі реального часу, що створює відчуття живого, реактивного інтерфейсу. Наприклад, додавання товару до кошика, відображення повідомлення про успішну дію або застосування фільтра відбуваються миттєво без оновлення всієї сторінки. Також у клієнтській частині активно використовується технологія localStorage — з її допомогою зберігається стан кошика та базова інформація про користувача, що

дозволяє підтримувати персоналізацію навіть після закриття або оновлення вкладки.

Інтерфейс спроєктовано з дотриманням сучасних принципів UX/UI-дизайну: структура сторінок інтуїтивно зрозуміла, навігація логічна, кнопки та інтерактивні елементи мають зрозумілі реакції на дії користувача. Форма пошуку дозволяє швидко знаходити товар за назвою, а фільтр за ціною дає змогу звузити результати до бажаного діапазону. Сайт коректно відображається на мобільних пристроях, що особливо важливо в умовах зростання мобільного трафіку.

З метою оптимізації продуктивності клієнтської частини застосовано низку технічних рішень. Файли стилів і скриптів мінімізовано для зменшення обсягу переданих даних. Зображення підвантажуються поступово, лише в міру їх появи в зоні видимості (lazy loading), що пришвидшує первинне завантаження сторінки. Крім того, дані кешуються на боці клієнта, що додатково знижує кількість звернень до сервера та покращує загальну швидкодію.

Клієнтська частина веб-сайту поєднує адаптивний дизайн, динамічну взаємодію та локальне збереження даних, що створює комфортний та ефективний досвід для користувача. Цей підхід повністю відповідає сучасним стандартам розробки інтерфейсів для інтернет-магазинів.

2.2.2 Серверна частина: обробка запитів і робота з базою даних

Серверна частина має ключову роль у функціонуванні веб-сайту, оскільки відповідає за реалізацію основної логіки, обробку різних запитів, взаємодію з базою даних і забезпечення узгодженості між клієнтом і сховищем інформації. Вона виступає посередником між користувачем, що взаємодіє з інтерфейсом, та серверними ресурсами, які містять структуру даних, ділову логіку і механізми безпеки.

У реалізованому проєкті сервер створено з використанням платформи Node.js, а основним інструментом для побудови маршрутизації та обробки

запитів став фреймворк Express.js. Це дозволило швидко й гнучко визначити REST API маршрути, через які клієнт може взаємодіяти з ресурсами — наприклад, отримувати перелік товарів. Комунікація відбувається через HTTP-запити типу GET, POST та інші, які відповідають за різні операції: отримання даних, створення нових записів, оновлення або видалення.

Для доступу до бази даних використовується ORM-бібліотека Sequelize, що забезпечує об'єктну абстракцію над реляційною моделлю. Завдяки цьому всі операції над таблицями (створення, читання, оновлення, видалення) виконуються в узгодженій та безпечній формі без необхідності прямого написання SQL-запитів. Моделі сутностей — наприклад, таблиця товарів — реалізовано у вигляді окремих структур з відповідними атрибутами, які потім синхронізуються з базою.

Крім основної логіки обробки запитів, у серверній частині передбачено обробку виняткових ситуацій. У разі помилки з'єднання з базою або виникнення проблем у процесі запиту сервер повертає користувачу відповідь із кодом помилки та поясненням, що забезпечує стабільність системи та полегшує діагностику на етапі розробки. Для безпеки та контролю доступу також реалізовано базові механізми захисту, зокрема CORS — для контролю дозволених доменів, що можуть звертатись до API, а також засоби обмеження частоти запитів, які можуть бути впроваджені для запобігання перевантаженню або зловживанням.

Така структура дає змогу легко масштабувати функціонал, додаючи нові маршрути або розширюючи базу даних. За потреби її можна доповнити механізмами авторизації, підтримкою замовлень і користувацькими кабінетами. Поточна реалізація вже створює стабільне та ефективне середовище для обслуговування основних запитів, зберігання товарів та інтеграції з клієнтською частиною сайту.

2.2.3 Використання API для динамічного завантаження та фільтрації

Для забезпечення інтерактивної взаємодії між клієнтською та серверною частинами веб-сайту було реалізовано REST API, яке виконує функцію посередника в обміні структурованими даними. Такий підхід дозволяє організувати логіку застосунку у вигляді чітко визначених запитів і відповідей, які відповідають стандартам сучасної веб-архітектури. У розробленому проєкті клієнтська частина звертається до серверної через API для отримання актуального списку товарів, фільтрації результатів за параметрами та оновлення вмісту сторінки в реальному часі без перезавантаження.

Основний обмін інформацією відбувається у форматі JSON. Зокрема, при першому завантаженні каталогу користувач отримує перелік усіх доступних товарів, а під час пошуку або застосування фільтрів клієнт формує динамічні запити, що містять параметри, такі як ключове слово в назві товару, мінімальна та максимальна ціна, категорія тощо. Сервер, у свою чергу, обробляє ці параметри та повертає лише ті позиції, які відповідають заданим критеріям, що дає змогу швидко та зручно взаємодіяти з великим обсягом даних. Запити обробляються за допомогою маршрутизатора, налаштованого на Express.js. Це дозволяє створювати логічно поділені точки доступу до інформації з можливістю передачі параметрів через URL. Така організація API спрощує розробку та підвищує гнучкість клієнтської частини, дозволяючи їй легко змінювати спосіб подачі інформації без переробки архітектури.

Задля безпеки сервер проводить базову валідацію запитів, перевіряючи, чи параметри фільтра є коректними, та обробляє помилки у випадку відсутності даних або технічних збоїв. Хоча в поточній версії реалізація авторизації або аутентифікації відсутня, структура побудована таким чином, щоб у майбутньому можна було легко інтегрувати механізми захисту, такі як токени доступу або обмеження частоти запитів.

Використання API у цьому проєкті не лише забезпечує ефективну взаємодію між фронтендом і бекендом, а й створює технічну основу для подальшого розвитку системи. Такий підхід дозволяє зосередитися на гнучкості, зручності користування та швидкодії, що є критично важливими факторами для сучасного інтернет-магазину.

2.2.4 Забезпечення безпеки даних і аутентифікації

Безпечне зберігання та передача інформації є однією з першочергових вимог до сучасного веб-середовища, особливо в контексті онлайн-торгівлі, де користувач взаємодіє з чутливими даними. Навіть якщо на початковому етапі система не оперує платіжною або персональною інформацією, підходи до її розробки повинні враховувати потенційні загрози та бути готовими до подальшого масштабування.

У поточній реалізації захист починається з обробки вхідних запитів на сервері. Перед виконанням будь-якої дії параметри перевіряються на відповідність очікуваному формату, що унеможливлює виконання шкідливих SQL або XSS-ін’єкцій. Крім того, обробка винятків побудована так, щоби користувач не отримував критичну інформацію про структуру системи у випадку помилки, що також зменшує потенційні вектори атак.

Архітектура бекенду побудована з урахуванням можливості підключення механізмів авторизації. Залежно від майбутніх потреб, тут легко реалізовується система на базі JSON Web Token або сесійного зберігання, що відкриває можливості для захисту приватних сторінок і обмеження доступу за ролями. У подальшому це дозволить додати функціонал для автентифікованих користувачів, таких як історія замовлень або керування профілем.

Для контролю зовнішніх запитів застосовано обмеження доступу на рівні доменів за допомогою CORS. Таким чином, запити приймаються лише з

дозволених джерел, що знижує ризик зловмисної взаємодії. Передбачається також, що фінальна версія сайту функціонуватиме виключно через захищений HTTPS-протокол, який гарантує шифрування даних при їх передачі між браузером і сервером.

Окремої уваги потребує захист даних користувача у базі. У разі впровадження реєстрації паролі не зберігатимуться у відкритому вигляді — замість цього застосовуватиметься хешування за допомогою надійних алгоритмів, таких як bcrypt. Подібна практика дозволяє зменшити наслідки потенційного витоку даних.

Отже, з самого початку закладено основи для побудови безпечного середовища такі як, перевірка запитів і структури доступу до збереження даних і майбутньої автентифікації. Це сприяє створенню надійної та стійкої до загроз системи, яка зможе адаптуватися до зростаючих вимог у процесі подальшого розвитку.

3 ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1 Аналіз функціональності веб-сайту

3.1.1 Пошук і фільтрація товарів

Пошук і фільтрація — це невід’ємні елементи функціональності сучасного інтернет-магазину, які безпосередньо впливають на зручність взаємодії користувача з каталогом. Особливо це актуально в нішах, де асортимент складається з десятків або сотень найменувань із близькими характеристиками, як-от у сфері комп’ютерних комплектуючих. У такому середовищі ефективна система навігації по товару набуває ключового значення.

У рамках реалізованого інтерфейсу пошук здійснюється через текстове поле, куди користувач вводить назву товару. На основі цього запиту клієнтська частина формує звернення до API, яке надсилається на сервер. Сервер, у свою чергу, здійснює пошук у базі даних за частковим збігом введеного тексту з назвами товарів.

Крім текстового пошуку, реалізовано базову фільтрацію за ціновим діапазоном. Це дає змогу звузити вибір до товарів, які відповідають фінансовим очікуванням відвідувача сайту. Користувач може вручну задати мінімальну та максимальну межу ціни, після чого відображаються лише ті позиції, що потрапляють у вказаний діапазон. Обидві функції можуть використовуватися одночасно, що підвищує точність результатів.

Поєднання цих двох механізмів дозволяє швидко орієнтуватися у великому переліку товарів, не перевантажуючи користувача непотрібною інформацією. Станом на зараз уже створено зручний і цілком робочий інструмент для швидкого доступу до релевантних позицій, що є важливим фактором для збільшення конверсії та покращення загального досвіду користування.

3.1.2 Механізм знижок і акцій

Система акцій і знижок є одним із найефективніших інструментів підвищення зацікавленості клієнтів та стимулювання продажів в електронній комерції. Особливо це актуально для веб-магазинів, де персоналізовані пропозиції можуть бути реалізовані автоматично, без втручання адміністратора. Використання знижок дозволяє не лише залучати нових покупців, але й утримувати постійних, формуючи емоційний зв’язок з брендом та стимулюючи повторні покупки.

У межах розробленого веб-додатку реалізована базова система персоналізованих знижок, орієнтована на визначні дати — зокрема, день народження користувача. Механізм полягає в наступному: під час авторизації перевіряється, чи вказав користувач дату народження при реєстрації. Якщо ця дата збігається з поточною, на сторінках магазину активується спеціальна знижка на всі товари, вартість яких перевищує задану межу . Це створює ефект персонального привітання й підвищує ймовірність покупки саме в цей день.

З бізнесової точки зору такий підхід має кілька важливих переваг:

- Підвищення лояльності. Клієнт, що отримує знижку в день народження, відчуває турботу, що позитивно впливає на сприйняття бренду.
- Мотивація до швидкої покупки. Оскільки пропозиція діє лише один день, користувач схильний не відкладати рішення про покупку.
- Зростання середнього чека. Знижки застосовуються до товарів певної вартості, що спонукає обирати дорожчі позиції.

З технічної сторони реалізація цієї функціональності базується на засобах клієнтської логіки. Після входу в систему перевіряється дата народження користувача (яка зберігається у localStorage або sessionStorage), порівнюється з поточною датою за допомогою об’єкта Date у JavaScript. Якщо збіг підтверджено, активується функція, яка перераховує ціни у каталозі товарів з урахуванням знижки. Перетворення відбувається динамічно — ціна змінюється без перезавантаження сторінки, що забезпечує плавний та приємний досвід для

користувача. На рисунку 3.1 зображено застосування персоналізованої знижки.

```
function applyBirthdayDiscount(product) {
    const currentUser = JSON.parse(localStorage.getItem('currentUser'));
    if (!currentUser?.hasBirthdayToday) return product;

    if (product.price >= 3000) {
        return {
            ...product,
            discountPrice: product.price * 0.2,
            isBirthdayDiscount: true
        };
    }
    return product;
}
```

Рисунок 3.1 – Застосування персоналізованої знижки у JavaScript

Відображення нової ціни супроводжується додатковими стилями: стара ціна перекреслюється, нова підсвічується, додається бейджик «Знижка» або повідомлення «День народження!». Також, у разі необхідності, цю логіку легко розширити: додати знижки до інших подій (чорна п’ятниця, новорічні акції), реалізувати таймер зворотного відліку або бонусну систему.

Цей підхід не вимагає складної інтеграції з сервером або платіжними системами, що робить його простим у впровадженні, але дуже ефективним у плані взаємодії з аудиторією. Також він демонструє гнучкість платформи до розширення й є прикладом корисного використання client-side логіки.

Загалом, динамічні знижки у веб-магазинах не лише підвищують обсяги продажів, а й формують імідж клієнтоорієнтованого сервісу. У майбутньому ця система може бути розширена за допомогою серверної перевірки, історії покупок, сегментації клієнтів та інтеграції з CRM-системами.

3.1.3 Список покупок та обробка замовлень

Кошик у веб-магазині виконує роль тимчасового сховища вибраних товарів, що дозволяє користувачеві формувати замовлення у зручний для себе спосіб. Його інтеграція в інтерфейс значно підвищує практичність користування сайтом, оскільки забезпечує контроль над вмістом замовлення без переходу на окремі сторінки чи потреби в повторному пошуку товарів.

У розробленому застосунку список покупок реалізовано повністю на клієнтському боці за допомогою мови програмування JavaScript. Усі додані товари зберігаються у localStorage браузера, що забезпечує автоматичне збереження даних навіть після оновлення сторінки або її закриття. Такий підхід дозволяє користувачу повернутись до оформлення покупки в будь-який момент, без необхідності повторно додавати позиції до кошика.

Користувач може додати товар до кошика натисканням відповідної кнопки на картці товару. Об’єкт із даними про товар — назва, ціна, кількість, зображення — кодується у форматі JSON і зберігається у локальному сховищі. У разі повторного додавання того самого товару система не створює дубліката, а збільшує кількість одиниць товару. Це дозволяє зручно керувати замовленням у динаміці.

Відображення вмісту кошика реалізовано за допомогою стандартних інструментів DOM API. При кожній зміні — додаванні, видаленні чи зміні кількості товарів — відбувається оновлення вмісту кошика без перезавантаження сторінки. Крім того, реалізоване автоматичне обчислення загальної вартості замовлення у реальному часі, що дозволяє користувачеві бачити фінальну суму з урахуванням усіх позицій одразу.

Окрему увагу приділено зручності взаємодії: передбачена можливість швидкого видалення товару, очищення всього кошика та виведення повідомлення у випадку, якщо кошик порожній. Також реалізовано перевірку наявності даних у localStorage при завантаженні сторінки — якщо користувач

уже формував замовлення раніше, система автоматично відображає попередній стан кошика.

Подібна реалізація є ефективним рішенням для невеликих веб-магазинів або MVP-проєктів, оскільки не потребує складної інтеграції з серверною логікою. У майбутньому цю модель можна легко розширити — додати збереження кошика на сервері, авторизацію із прив'язкою до облікового запису, або синхронізацію між кількома пристроями.

У підсумку, локальний кошик, реалізований на JavaScript і localStorage, забезпечує простоту, швидкодію та зручність ,що саме ті характеристики, які важливі для користувача в контексті онлайн-покупок.

3.2 Структура та логіка побудови товарного каталогу

Товарний каталог у системі інтернет-магазину є ключовим елементом, що формує основу взаємодії користувача з платформою. Саме через каталог відвідувачі отримують доступ до продукції, переглядають характеристики, порівнюють ціни, додають товари до кошика та здійснюють замовлення. Відповідно, структура і представлення товарів мають бути логічно організованими, інформативними та адаптованими до вимог цільової аудиторії.

У межах реалізованого проєкту каталог комплектуючих для ПК подано у вигляді структурованої бази товарів, які включають такі категорії: процесори, відеокарти, материнські плати, оперативна пам’ять, SSD, блоки живлення, комп’ютерні корпуси, охолодження, монітори, гарнітури, мишки, клавіатури та інші супутні компоненти. Це відображає реальний асортимент техніки, що актуальна для збирання або модернізації персонального комп’ютера.

Кожен товар у базі представлено як окрема сутність із наступними атрибутами: назва, зображення, повний опис, ціна, акційна ціна (за наявності). Дані структуровано у відповідній таблиці бази даних, приклад якої наведено нижче у таблиці 3.1. Такий підхід дозволяє легко реалізовувати функції

фільтрації, пошуку, порівняння та сортування товарів за ціною або іншими критеріями.

Таблиця 3.1 – Приклад структури збереження товарів у базі даних MySQL

ID	Name	Price	DiscountPrice	Description
1	Intel Core i9-13900K	2749	1699	Потужний процесор 13-го покоління
2	Відеокарта GF RTX 5080 16GB	11299	11299	Топова відеокарта нового покоління
3	Corsair Vengeance LPX 32GB DDR4	1119	1109	Високоякісна оперативна пам’ять
4	Kingston A2000 1TB NVMe SSD	2109	1099	Швидкісний SSD-накопичувач

3.3 Перспективи розвитку веб-сайту

Подальший розвиток функціональних можливостей веб-застосунку є логічною і необхідною складовою його життєвого циклу. В умовах постійної зміни вимог користувачів, розвитку технологій та підвищення конкуренції на ринку електронної комерції, вдосконалення інтернет-магазину сприяє не лише підвищенню рівня надання різних послуг, а й зміцненню конкурентних позицій розробленого рішення.

На поточному етапі реалізації веб-сайту вдалося створити повноцінний інтернет-магазин із базовим функціоналом: каталог товарів із фільтрацією, динамічні сторінки товарів, кошик, система авторизації, персоналізована знижка до дня народження, а також є можливість додавання нових товарів у базу даних. Проте це лише початковий етап у розвитку повноцінної комерційної платформи. З огляду на перспективу масштабування, доцільно сформулювати перелік майбутніх напрямів вдосконалення, які сприятимуть розширенню

функціональності, покращенню зручності для користувача та підвищенню ефективності адміністративного управління сайтом.

Першочергово доцільним є впровадження повноцінного особистого кабінету користувача. У його межах може бути реалізовано функціонал перегляду історії замовлень, збереження обраних товарів (wish list), керування персональними даними та повторного замовлення. Це підвищить рівень персоналізації взаємодії з клієнтом, зробить процес користування сайтом зручнішим і покращить загальний користувацький досвід.

Ще одним важливим кроком є додання інструментів для адміністратора. Зокрема, варто реалізувати інтерфейс для керування товарами: додавання, редагування, видалення позицій, завантаження зображень, встановлення акцій і знижок. Додатково можна передбачити модулі керування категоріями товарів, оновленням бази даних, а також автоматичне формування звітів про продажі, запаси та популярність позицій.

Серед функціональних удосконалень для клієнтів варто відзначити впровадження інтерактивного чату з оператором або ботом-консультантом. Такий інструмент дозволить у режимі реального часу відповідати на запити клієнтів, допомагати з вибором товарів, вирішувати проблеми із замовленням чи оплатою. Це зменшить кількість незавершених покупок і підвищить загальну задоволеність клієнтів від взаємодії з платформою.

Також доцільно реалізувати систему відгуків і рейтингів. Відгуки на сторінках товарів формують довіру до магазину, покращують прозорість сервісу та допомагають новим клієнтам у виборі. Система рейтингів дозволяє виділяти найпопулярніші товари, що може бути використано також у модулях рекомендацій. З технічного боку перспективним є впровадження системи рекомендацій, яка ґрунтується на історії переглядів і покупок користувачів. Застосування елементарних алгоритмів машинного навчання або аналітичних правил дозволить формувати індивідуальні добірки товарів, що підвищить ймовірність покупки й середній чек.

У сфері безпеки та збереження даних подальший розвиток системи має включати реалізацію захищеної авторизації через JSON Web Token (JWT), двофакторну аутентифікацію, а також підключення сучасних сервісів для резервного копіювання бази даних. Це сприятиме підвищенню довіри до веб-сайту та дозволить масштабувати систему до обслуговування великої кількості користувачів.

Окрему увагу варто приділити аналітичному блоку. Підключення сторонніх або власноруч реалізованих модулів аналітики (наприклад, Google Analytics API) дозволить відстежувати поведінкові патерни користувачів, найпопулярніші товари, час на сторінці, джерела трафіку. На основі цих даних можна оптимізувати структуру каталогу, розміщення елементів на сторінці, покращити маркетингову стратегію та UX-дизайн.

Створена система має високий потенціал для розвитку. Запропоновані напрями вдосконалення дозволяють трансформувати базовий веб-сайт у повноцінну платформу електронної комерції, що відповідає актуальним ринковим стандартам і може розширюватися згідно з розвитком бізнесу. Реалізація описаних функціональних блоків у майбутньому дозволить не лише підвищити ефективність використання ресурсу, а й сформувати конкурентну перевагу серед аналогічних платформ.

3.4 Екранні форми та елементи інтерфейсу веб-сайту

У цьому розділі показано ключові екранні форми реалізованого веб-застосунку, що ілюструють основні функціональні можливості інтернет-магазину з продажу комп’ютерних комплектуючих.

На рисунку 3.2 зображено інтерфейс для створення облікового запису, що включає обов’язкові поля для введення імені користувача, пароля та іншої

інформації. Реалізовано базову валідацію та повідомлення про помилки при слабкому паролі.

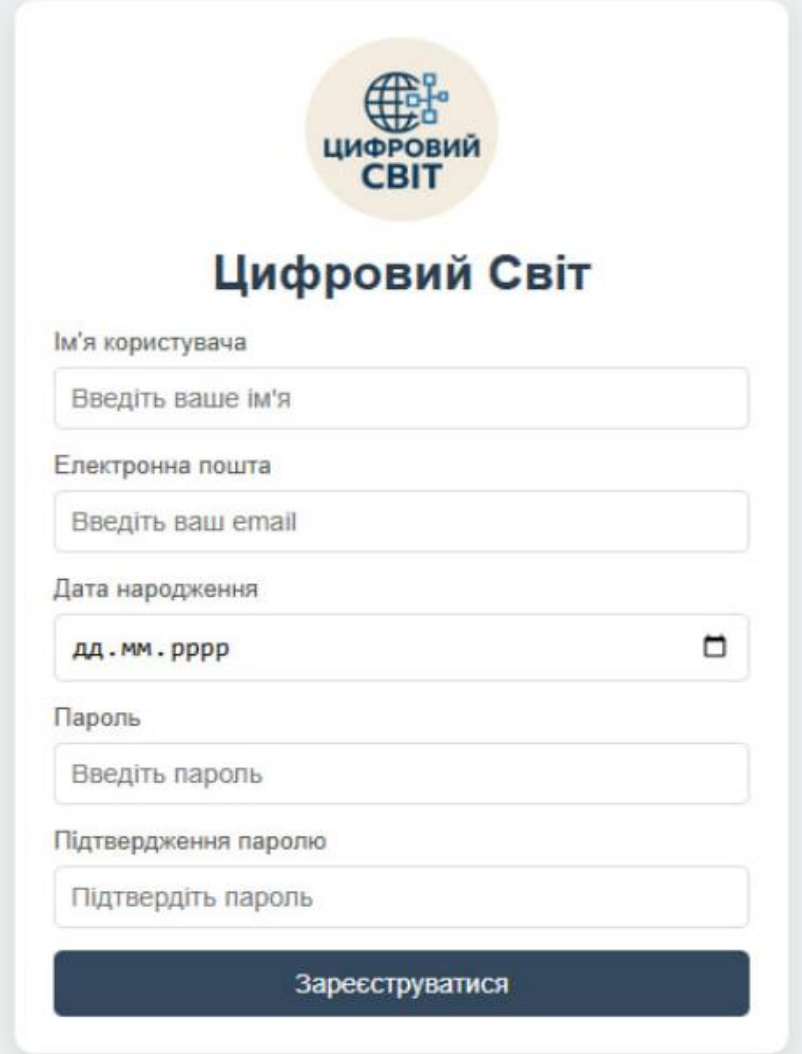


Рисунок 3.2 – Форма реєстрації нового користувача

На рисунку 3.3 зображено форму входу до системи. Після успішної авторизації користувач отримує доступ до персоналізованих функцій магазину, таких як залишення відгуків чи перегляд знижок.

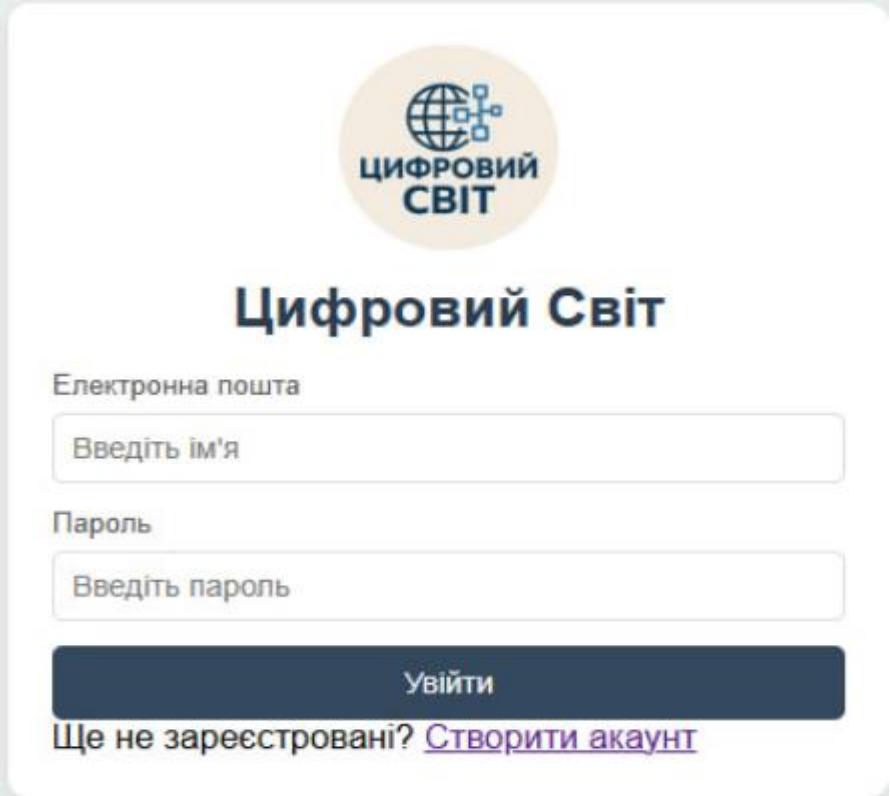


Рисунок 3.3 – Форма авторизації користувача

На рисунку 3.4 зображено головну сторінку магазину, яка містить навігаційне меню, банер або логотип, каталог товарів і доступ до основних функцій . Інтерфейс адаптивний і дозволяє користувачу швидко орієнтуватися у вмісті.

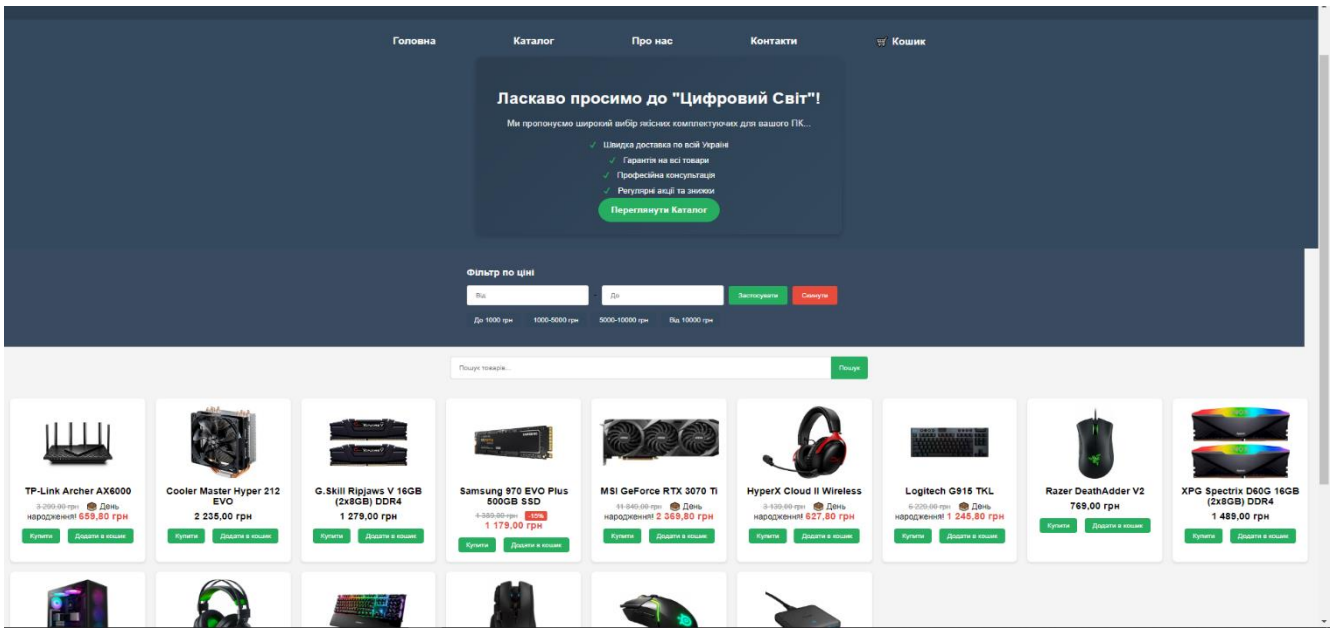


Рисунок 3.4 – Головна сторінка веб-сайту

На рисунку 3.5 зображено каталог товарів, де кожна позиція представлена у вигляді окремої картки з фотографією, назвою, описом, ціною та кнопкою «Додати в кошик». Також присутня персональна знижка на день народження. Контент формується динамічно з бази даних через API-запити.

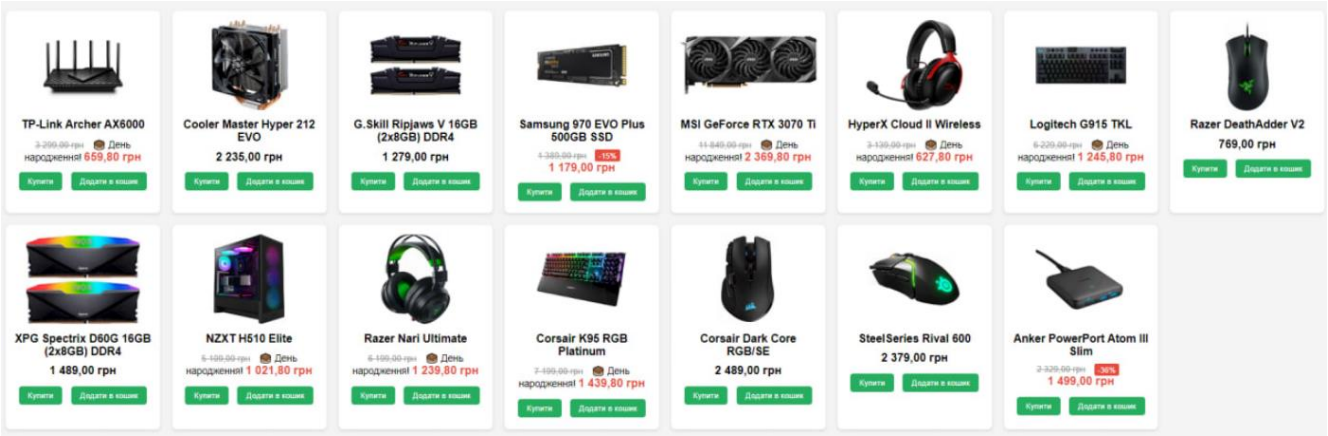


Рисунок 3.5 – Каталог товарів

На рисунку 3.6 зображено інтерфейс фільтрації товарів за ціновим діапазоном. Користувач вводить мінімальну та максимальну вартість, після чого результати оновлюються у реальному часі згідно з вибраними параметрами.

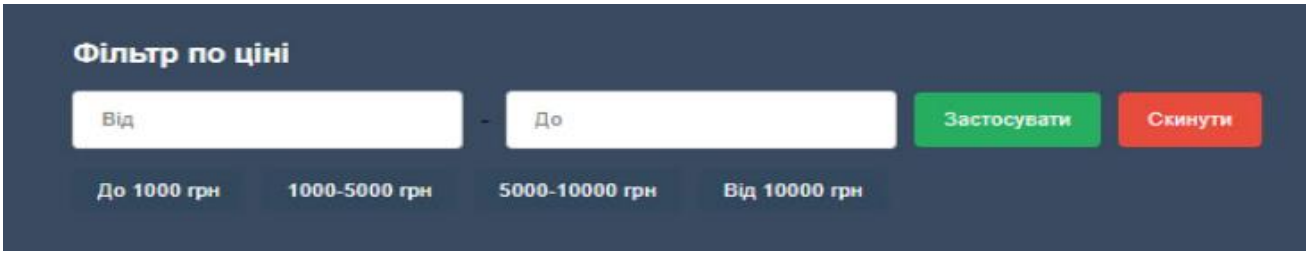


Рисунок 3.6 – Фільтрація за ціною

На рисунку 3.7 зображено поле, яке дає змогу здійснити пошук товарів за ключовими словами.

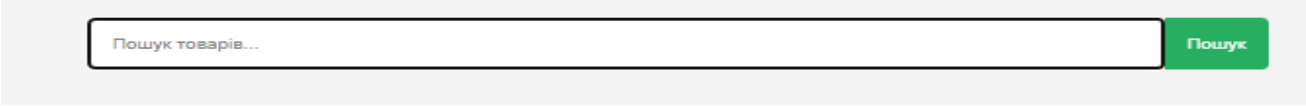


Рисунок 3.7 – Пошук за назвою товару

На рисунку 3.8 зображено сторінку товару, що відкривається після натискання на картку в каталозі. Форма містить велике зображення товару, опис, ціну, а також можливість залишити відгук.

Рисунок 3.8 – Перегляд окремого товару

На рисунку 3.9 зображено кошик покупок, у якому відображаються всі додані товари, їх кількість та підсумкова вартість. Передбачена можливість видалення позицій.

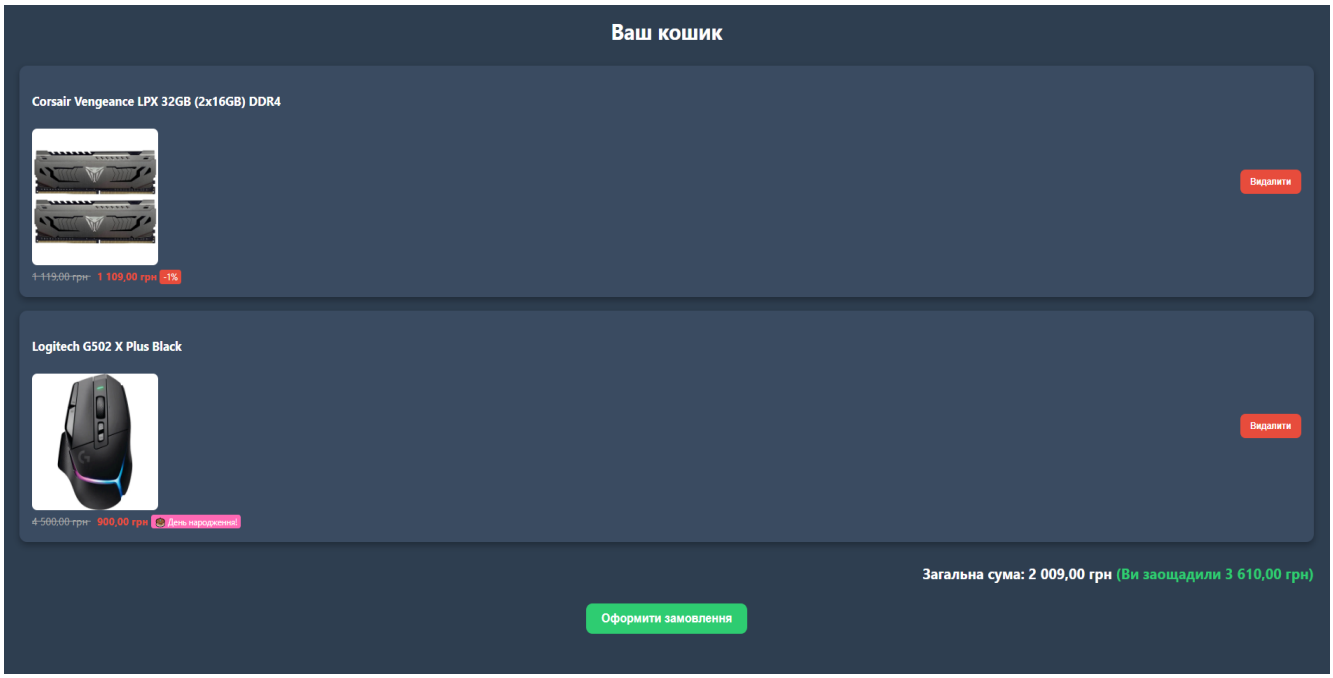


Рисунок 3.9 – Кошик покупок

На рисунку 3.10 відображено форму, що дозволяє користувачеві подати заявку на придбання товару в кредит. Передбачено вибір терміну кредитування та попереднє ознайомлення з умовами. Цей функціонал підвищує фінансову доступність продукції для широкого кола користувачів.

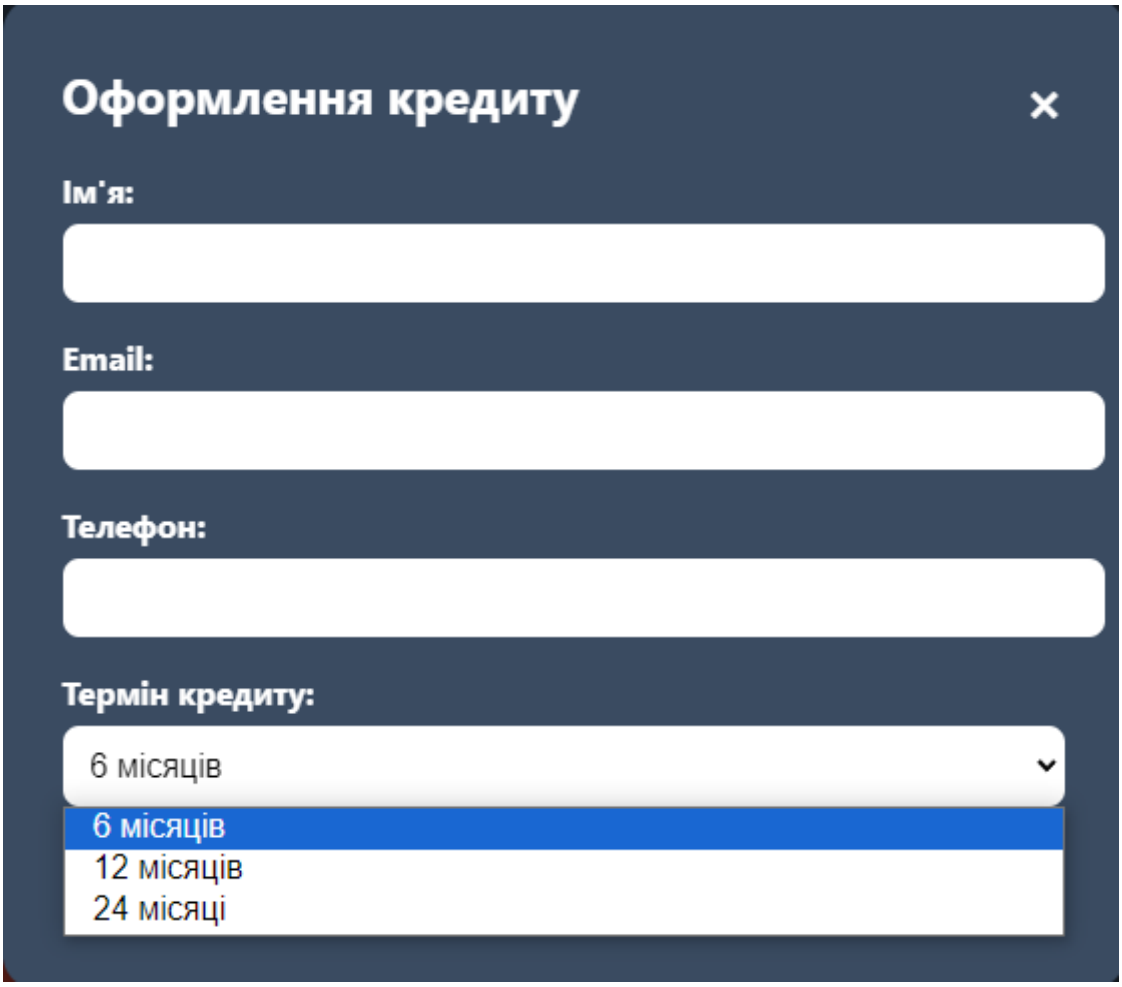


Рисунок 3.10 – Форма оформлення покупки в кредит

На рисунку 3.11 зображено інтерфейс, що дозволяє користувачеві залишити коментар або оцінку до переглянутого товару. Цей функціонал є важливою складовою користувацької взаємодії, оскільки забезпечує зворотний зв’язок і формує репутацію товару на основі реального досвіду покупців.

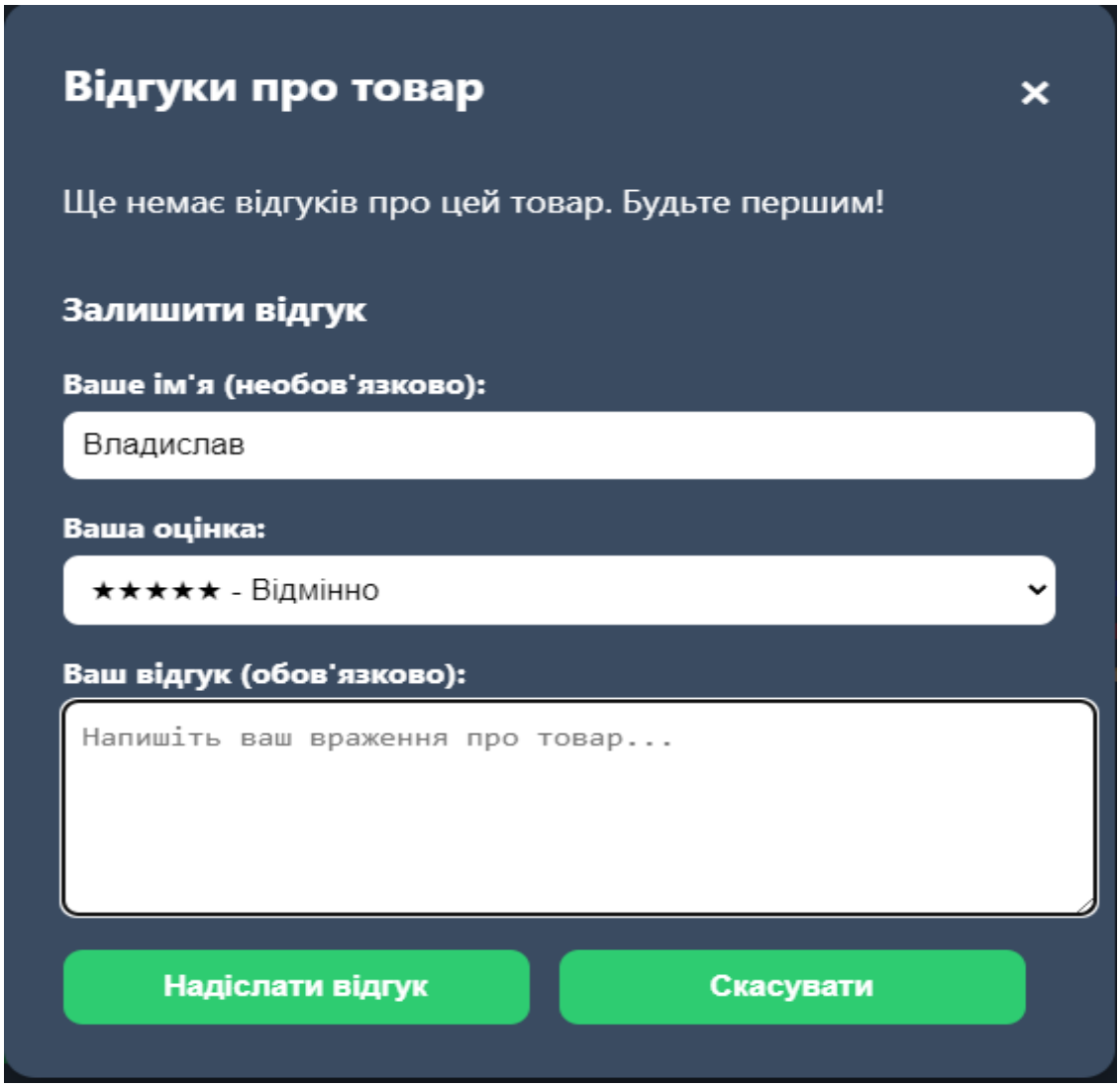


Рисунок 3.11 – Форма залишення відгуку про товар

На рисунку 3.12 ілюстрований приклад відображення відгуку на веб-сайті, що демонструє, як користувачі можуть залишати свої коментарі та оцінки щодо товарів або послуг.

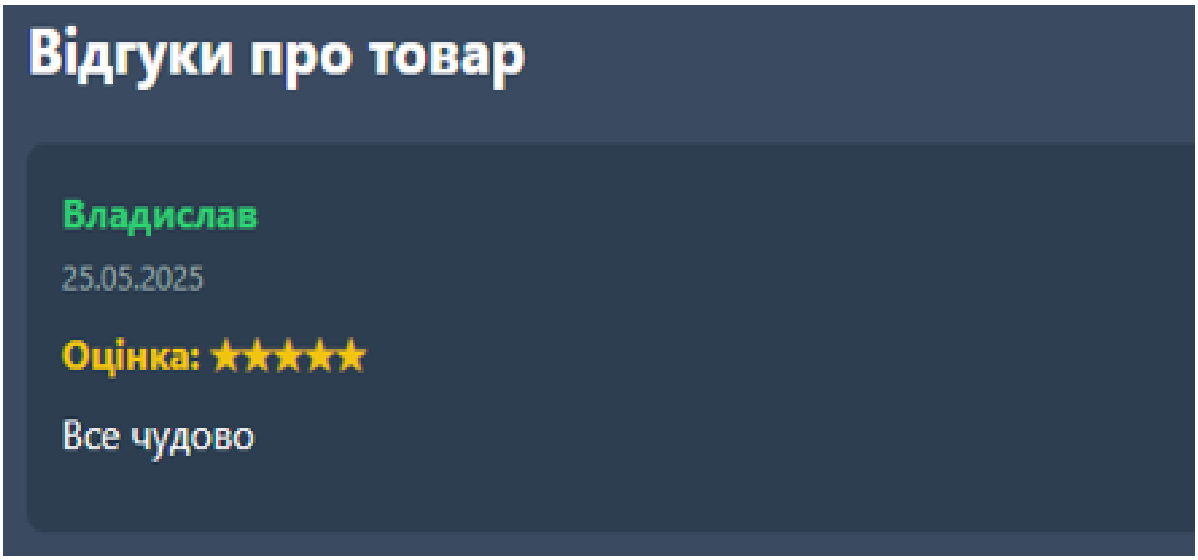


Рисунок 3.12 – Приклад відображення відгуку у веб-магазині

На рисунку 3.13 зображено повідомлення, що підтверджує успішне завершення оформлення покупки.

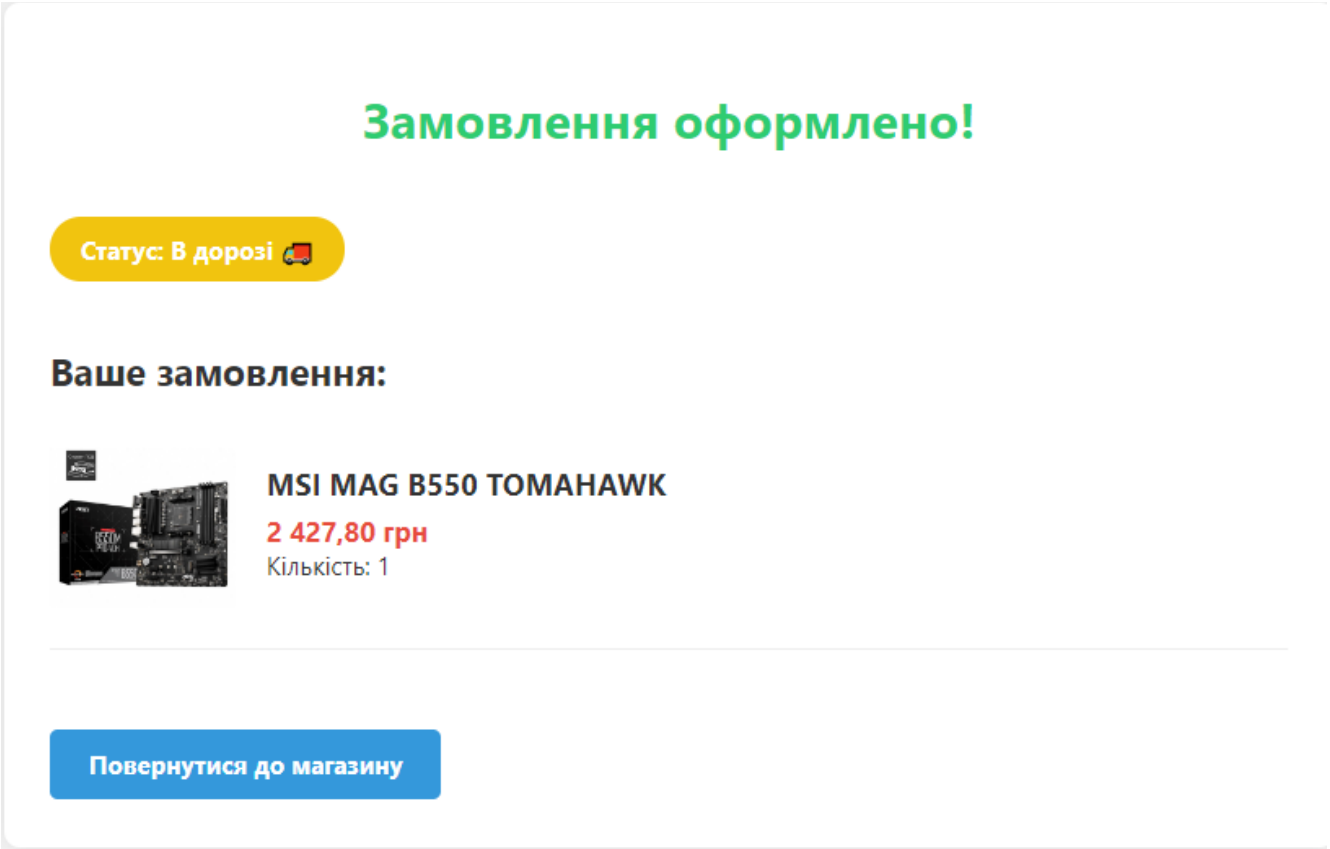


Рисунок 3.13 – Сторінка успішного оформлення замовлення

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було здійснено комплексну розробку веб-сайту для продажу комп'ютерних комплектуючих. Проєкт об'єднав як дослідження предметної області та аналіз існуючих аналогів сучасних веб-технологій, так і практичне втілення повноцінного функціонального веб-магазину з урахуванням вимог зручності, безпеки та ефективності.

На основі попереднього аналізу ринку електронної комерції та існуючих рішень було визначено ключові вимоги до структури, інтерфейсу та функціональності майбутнього сайту. У результаті реалізовано сайт, який дозволяє користувачам переглядати товари, здійснювати пошук і фільтрацію, додавати товари до кошика, авторизуватись, залишати відгуки та бачити персональні знижки в день народження.

Для реалізації веб-магазину були використані такі технології:

- HTML та CSS — для створення структури і стилю сторінок з адаптивною верстkою;
- JavaScript — для реалізації клієнтської логіки, динамічного оновлення інтерфейсу, обробки подій, фільтрації, кошика та взаємодії з користувачем;
- Node.js — як серверне середовище виконання JavaScript для обробки HTTP-запитів, підключення до бази даних та передачі даних до клієнта;
- MySQL — для збереження інформації про товари;
- LocalStorage — для збереження даних користувача, кошика та знижок без необхідності взаємодії з сервером.

У ході роботи було виконано такі завдання:

- розроблено архітектуру веб-додатку та базову модель бази даних;
- реалізовано інтерфейс із зручною навігацією, адаптивним дизайном та зрозумілою структурою;
- впроваджено персоналізовану знижку на день народження, що стимулює

активність користувачів;

- забезпечено локальне збереження кошика, що дозволяє формувати замовлення в декілька етапів;
- реалізовано фільтрацію товарів за ціною та пошук за назвою, що спрощує користувачам взаємодію з каталогом.

Практична цінність проєкту полягає в тому, що створена система може бути використана як основа або готове рішення для малого або середнього бізнесу. Завдяки своїй гнучкості та простоті у розгортанні, проєкт не потребує складного обслуговування, що дає змогу бути доступним для широкого кола користувачів.

Перспективи розвитку передбачають впровадження авторизації через базу даних, реалізацію особистого кабінету користувача, панелі адміністратора, інтеграцію з платіжними системами, додавання багатомовності та аналітики замовлень. Також можливе перенесення збереження кошика на сервер, що забезпечить синхронізацію між різними пристроями.

У процесі виконання роботи було досягнуто поставленої мети: створено сучасний, адаптивний, функціональний веб-сайт, який відповідає актуальним вимогам веб-розробки та демонструє ефективність використання відкритих технологій у сфері електронної комерції.

ПЕРЕЛІК ПОСИЛАНЬ

1. WebDevSimplified. JavaScript Shopping Cart Tutorial [Відео]. URL: https://www.youtube.com/watch?v=YeFzkC2awTM

2. Rozetka.ua – Інтернет-магазин техніки та електроніки [Електронний ресурс]. URL: https://rozetka.com.ua/

3. Citrus.ua – Інтернет-магазин електроніки [Електронний ресурс]. URL: https://www.citrus.ua/

4. Newegg.com – Online Tech Retailer [Електронний ресурс]. URL: https://www.newegg.com/

5. CodeAcademy. Learn JavaScript [Електронний ресурс]. URL: https://www.codecademy.com/learn/introduction-to-javascript

6. Node.js Documentation [Електронний ресурс]. URL: https://nodejs.org/en/docs/

7. Express.js Documentation [Електронний ресурс]. URL: https://expressjs.com/

8. MySQL 8.0 Reference Manual [Електронний ресурс]. URL: https://dev.mysql.com/doc/

9. GeeksforGeeks. Web Development Basics [Електронний ресурс]. URL: https://www.geeksforgeeks.org/web-development/

10. Wikipedia. JavaScript [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/JavaScript

11. Flanagan D. JavaScript Features To Forget [Електронний ресурс]. URL: https://davidflanagan.com/2020/05/12/javascript-to-forget.html

12. Дакетт Дж. HTML і CSS. Розробка та дизайн веб-сайтів. – К.: Вільямс, 2018. – 512 с.

13. Prometheus. Основи програмування з HTML, CSS та JavaScript [Електронний ресурс].URL: https://prometheus.org.ua/prometheus-free/programming-fundamentals-html-css-javascript/

14. Prometheus. Основи Web UI розробки 2023 [Електронний ресурс]. URL: https://prometheus.org.ua/prometheus-free/web-ui-development-basics/

15. EdEra. Основи веб-розробки (HTML, CSS, JavaScript) [Електронний ресурс]. URL: https://study.ed-era.com/courses/course/5159

16. Better Stack Community. Monitoring Node.js Apps with Prometheus [Електронний ресурс]. URL: https://betterstack.com/community/guides/scaling-nodejs/nodejs-prometheus/

17. RisingStack Blog. Node.js Performance Monitoring with Prometheus [Електронний ресурс]. URL: https://blog.risingstack.com/node-js-performance-monitoring-with-prometheus/

18. Mozilla Developer Network. JavaScript Guide [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

19. W3Schools. Node.js Tutorial [Електронний ресурс]. URL: https://www.w3schools.com/nodejs/

20. DigitalOcean. MySQL Tutorials [Електронний ресурс]. URL: https://www.digitalocean.com/community/tags/mysql

ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

```
<!DOCTYPE html>
<html lang="uk">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Форма реєстрації</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="container">
        <div class="form-wrapper">
            <header>
                <img class="settingLogo" src="/images/logo.png"
alt="logo">
                <h1>Цифровий світ</h1>
            </header>
            <form id="registrationForm"
onsubmit="registration(event)">
                <div class="input-group">
                    <label for="username">Ім'я користувача</label>
                    <input type="text" id="username" name="username"
placeholder="Введіть ваше ім'я" required>
                </div>
                <div class="input-group">
                    <label for="email">Електронна пошта</label>
                    <input type="email" id="email" name="email"
placeholder="Введіть ваш email" required>
                </div>
                <div class="input-group">
                    <label for="date">Дата народження</label>
                    <input type="date" id="date" name="date"
placeholder="Введіть дату народження" required>
                </div>
                <div class="input-group">
                    <label for="password">Пароль</label>
                    <input type="password" id="password"
name="password" placeholder="Введіть пароль" required>
                </div>
```

```
                <div class="input-group">
                    <label for="confirmPassword">Підтвердження
паролю</label>
                    <input type="password" id="confirmPassword"
name="confirmPassword" placeholder="Підтвердіть пароль" required>
                </div>
                <button type="submit" class="submit-
btn">Зареєструватися</button>
            </form>
        </div>
    </div>

    <script src="form.js"></script>
</body>



</html>
<!DOCTYPE html>
<html lang="uk">

<head>
    <meta charset="UTF-8" />
    <title>Кошик</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-
serif;
            background-color: #2e3e50;
            color: #ffffff;
            margin: 0;
            padding: 40px;
        }

        h1 {
            text-align: center;
            margin-bottom: 30px;
        }

        .cart-item {
            background-color: #3a4b61;
            border-radius: 12px;
            padding: 20px;
            margin-bottom: 20px;
```

```
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);
}

.cart-item img {
    border-radius: 8px;
    margin-top: 10px;
}

.cart-item button {
    background-color: #e74c3c;
    color: white;
    border: none;
    padding: 10px 16px;
    border-radius: 8px;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.2s ease;
}

.cart-item button:hover {
    background-color: #c0392b;
}

.total {
    font-weight: bold;
    font-size: 1.4em;
    margin-top: 30px;
    text-align: right;
}

#checkout-button {
    display: block;
    margin: 30px auto 0;
    padding: 12px 24px;
    background-color: #2ecc71;
    color: white;
    border: none;
    border-radius: 10px;
    font-size: 18px;
    font-weight: bold;
    cursor: pointer;
```

```
    transition: background-color 0.2s ease;
}

#checkout-button:hover {
    background-color: #27ae60;
}

.modal {
    display: none;
    position: fixed;
    z-index: 999;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
    background-color: rgba(0, 0, 0, 0.6);
}

.modal-content {
    background-color: #3a4b61;
    margin: 10% auto;
    padding: 30px;
    border-radius: 16px;
    width: 90%;
    max-width: 500px;
    color: white;
    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.5);
}

.modal-content h2 {
    margin-top: 0;
    text-align: center;
}

.modal-content input {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border-radius: 8px;
    border: none;
    font-size: 16px;
}
```

```
        .modal-content button {
            margin-top: 15px;
            width: 100%;
            padding: 12px;
            background-color: #2ecc71;
            border: none;
            border-radius: 10px;
            color: white;
            font-weight: bold;
            font-size: 16px;
            cursor: pointer;
        }

        .modal-content button:hover {
            background-color: #27ae60;
        }

        .close {
            float: right;
            font-size: 28px;
            font-weight: bold;
            color: #ffffff;
            cursor: pointer;
        }

        .close:hover {
            color: #e74c3c;
        }
    </style>
</head>

<body>
    <h1>Ваш кошик</h1>
    <div id="cart-items"></div>
    <div class="total" id="total-price"></div>
    <button id="checkout-button">Оформити замовлення</button>

    <div id="checkout-modal" class="modal">
        <div class="modal-content">
            <span class="close" id="close-modal">&times;</span>
            <h2>Оформлення замовлення</h2>
            <input type="text" placeholder="Ваше ім'я" id="name"
required />
            <input type="tel" placeholder="Телефон" id="phone"
```

```
required />
            <input type="email" placeholder="Email" id="email"
required />
            <input type="number" placeholder="Номер картки"
id="cardNumber" required />
            <input type="number" placeholder="Термін картки"
id="cardData" required />
            <input type="number" placeholder="CVV код картки"
id="CVVcardData" required />


            <button onclick="submitOrder()">Підтвердити
замовлення</button>
        </div>
    </div>


    <script src="cart.js"></script>
</body>

</html>
document.addEventListener("DOMContentLoaded", () => {
    const cartContainer = document.getElementById("cart-items");
    const totalPriceEl = document.getElementById("total-price");

    let cart = JSON.parse(localStorage.getItem("cart")) || [];

    function applyBirthdayDiscount(product) {
        const currentUser =
JSON.parse(localStorage.getItem('currentUser'));
        if (!currentUser?.hasBirthdayToday) return product;

        if (product.price >= 3000) {
            return {
                ...product,
                discountPrice: product.price * 0.2,
                isBirthdayDiscount: true
            };
        }
        return product;
    }

    function formatPrice(price) {
        return new Intl.NumberFormat('uk-UA', { style: 'currency',
```

```
currency: 'UAH' }).format(price);
    }

    function updateCart() {
        cartContainer.innerHTML = "";
        let total = 0;
        let totalDiscount = 0;

        if (cart.length === 0) {
            cartContainer.innerHTML = "<p>Кошик порожній</p>";
            totalPriceEl.textContent = "";
            return;
        }

        cart = cart.map(applyBirthdayDiscount);

        cart.forEach((product, index) => {
            const item = document.createElement("div");
            item.classList.add("cart-item");

            const hasDiscount = product.discountPrice <
product.price;
            const priceToUse = hasDiscount ? product.discountPrice :
product.price;
            total += parseFloat(priceToUse);

            if (hasDiscount) {
                totalDiscount += (product.price -
product.discountPrice);
            }

            item.innerHTML = `
                <div>
                    <h3>${product.name}</h3>
                    <img src="${product.imgUrl}"
alt="${product.name}" width="200">
                    ${hasDiscount ?
                    `<div>
                            <span style="text-decoration: line-
through; color: #aaa; margin-right: 10px;">
                                ${formatPrice(product.price)}
                            </span>
                            <span style="color: #e74c3c; font-
weight: bold;">
```

```
${formatPrice(product.discountPrice)}
                            </span>
                            ${product.isBirthdayDiscount ?
                                '<span style="background-color:
#ff69b4; color: white; padding: 2px 6px; border-radius: 4px; font-
size: 0.8em;">🎂 День народження!</span>' :
                                `<span style="background-color:
#e74c3c; color: white; padding: 2px 6px; border-radius: 4px; font-
size: 0.8em;">
                                    -${Math.round(((product.price -
product.discountPrice) / product.price) * 100)}%
                                </span>`
                            }
                        </div>` :
                    `<p>${formatPrice(product.price)}</p>`
                }
                </div>
                <button
onclick="removeFromCart(${index})">Видалити</button>
            `;
            cartContainer.appendChild(item);
        });

        let totalText = `Загальна сума: ${formatPrice(total)}`;
        if (totalDiscount > 0) {
            totalText += ` <span style="color: #2ecc71;">(Ви
заощадили ${formatPrice(totalDiscount)})</span>`;
        }
        totalPriceEl.innerHTML = totalText;
    }

    window.removeFromCart = function (index) {
        cart.splice(index, 1);
        localStorage.setItem("cart", JSON.stringify(cart));
        updateCart();
    };

    updateCart();
});

const modal = document.getElementById("checkout-modal");
const closeModal = document.getElementById("close-modal");
const checkoutButton = document.getElementById("checkout-button");
```

```
checkoutButton.addEventListener("click", () => {
    modal.style.display = "block";
});

closeModal.addEventListener("click", () => {
    modal.style.display = "none";
});

window.addEventListener("click", (e) => {
    if (e.target === modal) {
        modal.style.display = "none";
    }
});

function submitOrder() {
    const name = document.getElementById("name").value.trim();
    const phone = document.getElementById("phone").value.trim();
    const email = document.getElementById("email").value.trim();
    const cardNumber =
document.getElementById("cardNumber").value.trim();
    const cardData =
document.getElementById("cardData").value.trim();
    const CVVcardData =
document.getElementById("CVVcardData").value.trim();

    if (!name || !phone || !email || !cardNumber || !cardData ||
!CVVcardData) {
        alert("Будь ласка, заповніть усі поля!");
        return;
    }

    const order = {
        date: new Date().toISOString(),
        customer: { name, phone, email },
        products: JSON.parse(localStorage.getItem('cart')) || [],
        status: "В дорозі"
    };

    const orders = JSON.parse(localStorage.getItem('orders')) || [];
    orders.push(order);
    localStorage.setItem('orders', JSON.stringify(orders));

    window.location.href = "order-confirmation.html";
```

```
}


let currentProduct = null;

window.showReviews = function() {
    const reviewsModal = document.getElementById("reviewsModal");
    if (reviewsModal) {
        reviewsModal.style.display = "flex";
        loadReviewsModal();
    }
};

window.closeModal = function(modalId) {
    document.getElementById(modalId).style.display = "none";
};

function loadReviewsModal() {
    const reviewsList = document.getElementById("reviews-list");
    const reviews =
JSON.parse(localStorage.getItem(`reviews_${currentProduct.id}`)) ||
[];

    reviewsList.innerHTML = reviews.length === 0
        ? '<p>Ще немає відгуків про цей товар. Будьте першим!</p>'
        : reviews.map(review => `
            <div class="review">
                <div class="review-author">${review.author}</div>
                <div class="review-date">${new
Date(review.date).toLocaleDateString('uk-UA')}</div>
                <div class="review-rating">Оцінка:
${'★'.repeat(review.rating)}${'☆'.repeat(5 - review.rating)}</div>
                <div class="review-text">${review.text}</div>
            </div>
        `).join('');
}

function addReview(event) {
    event.preventDefault();

    const form = event.target;
    const formData = new FormData(form);

    if (!formData.get('text')) {
```

```
        alert("Будь ласка, напишіть текст відгуку");
        return;
    }

    const review = {
        author: formData.get('author') || "Анонімний покупець",
        rating: parseInt(formData.get('rating')),
        text: formData.get('text'),
        date: new Date().toISOString()
    };

    const reviews =
JSON.parse(localStorage.getItem(`reviews_${currentProduct.id}`)) ||
[];
    reviews.push(review);
    localStorage.setItem(`reviews_${currentProduct.id}`,
JSON.stringify(reviews));

    form.reset();
    loadReviewsModal();
    loadReviewsPage();
    updateReviewButton();
    alert("Дякуємо за ваш відгук!");
}

function loadReviewsPage() {
    const reviewsContainer = document.getElementById("reviews-
container");
    if (!reviewsContainer) return;

    const reviews =
JSON.parse(localStorage.getItem(`reviews_${currentProduct.id}`)) ||
[];
    reviewsContainer.innerHTML = reviews.length === 0
        ? '<p>Ще немає відгуків про цей товар. Будьте першим!</p>'
        : `<h2>Відгуки про товар</h2>` + reviews.map(review => `
            <div class="review">
                <div class="review-author">${review.author}</div>
                <div class="review-date">${new
Date(review.date).toLocaleDateString('uk-UA')}</div>
                <div class="review-rating">Оцінка:
${'★'.repeat(review.rating)}${'☆'.repeat(5 - review.rating)}</div>
                <div class="review-text">${review.text}</div>
            </div>
```

```
        `).join('');
}

function updateReviewButton() {
    const button =
document.querySelector("button[onclick='showReviews()']");
    if (button) {
        const reviewsCount =
JSON.parse(localStorage.getItem(`reviews_${currentProduct.id}`))?.le
ngth || 0;
        button.textContent = `Відгуки (${reviewsCount})`;
    }
}

document.addEventListener("DOMContentLoaded", () => {
    try {
        const raw = localStorage.getItem("selectedProduct");
        const decoded = decodeURIComponent(raw);
        currentProduct = JSON.parse(decoded);

        if (currentProduct) {
            renderProduct();
            initEventListeners();
            loadReviewsPage();
            updateReviewButton();
        } else {
            document.getElementById("product-details").innerHTML =
"<p>Продукт не знайдено</p>";
        }
    } catch (err) {
        console.error("Помилка завантаження продукту:", err);
        document.getElementById("product-details").innerHTML =
"<p>Помилка завантаження продукту</p>";
    }
});

function renderProduct() {
    const container = document.getElementById("product-details");
    const productWithDiscount =
applyBirthdayDiscount(currentProduct);
    const hasDiscount = productWithDiscount.discountPrice <
productWithDiscount.price;
    const discountPercent = hasDiscount
        ? Math.round(((productWithDiscount.price -
```

```
productWithDiscount.discountPrice) / productWithDiscount.price) *
100)
        : 0;

    container.innerHTML = `
        <h1>${productWithDiscount.name}</h1>
        <img src="${productWithDiscount.imgUrl}"
alt="${productWithDiscount.name}" style="max-width:300px">
        <p>${productWithDiscount.description}</p>
        ${hasDiscount ?
            `<div style="margin-bottom: 20px;">
                <p style="text-decoration: line-through; color:
#aaa; margin-bottom: 5px;">
                    Стара ціна:
${formatPrice(productWithDiscount.price)}
                </p>
                <p style="color: #e74c3c; font-size: 24px; font-
weight: bold; margin-bottom: 5px;">
                    Нова ціна:
${formatPrice(productWithDiscount.discountPrice)}
                </p>
                ${productWithDiscount.isBirthdayDiscount ?
                    '<span style="background-color: #ff69b4; color:
white; padding: 5px 10px; border-radius: 4px;">🎂 Знижка 80% на День
народження!</span>' :
                    `<span style="background-color: #e74c3c; color:
white; padding: 5px 10px; border-radius: 4px;">
                        Знижка ${discountPercent}%
                    </span>`
                }
            </div>` :
            `<p style="font-size: 24px; font-weight: bold;">Ціна:
${formatPrice(productWithDiscount.price)}</p>`
        }
        <button onclick="addToCart()">Додати до кошика</button>
        <button onclick="buyCredit()">Купити в кредит</button>
        <button onclick="showReviews()">Відгуки
(${getReviewsCount()})</button>
    `;
}

function applyBirthdayDiscount(product) {
    const currentUser =
JSON.parse(localStorage.getItem('currentUser'));
```

```
    if (!currentUser?.hasBirthdayToday) return product;

    if (product.price >= 3000) {
        return {
            ...product,
            discountPrice: product.price * 0.2,
            isBirthdayDiscount: true
        };
    }
    return product;
}


function initEventListeners() {

document.getElementById("creditForm")?.addEventListener("submit",
function(e) {
        e.preventDefault();
        const formData = new FormData(e.target);
        alert(`Дякуємо! Ваша заявка на кредит прийнята. Ми
зв'яжемося з вами.`);
        closeModal('creditModal');
    });


document.getElementById("reviewForm")?.addEventListener("submit",
addReview);
}

function formatPrice(price) {
    return new Intl.NumberFormat('uk-UA', { style: 'currency',
currency: 'UAH' }).format(price);
}

function getReviewsCount() {
    return
JSON.parse(localStorage.getItem(`reviews_${currentProduct.id}`))?.le
ngth || 0;
}

window.addToCart = function() {
    let cart = JSON.parse(localStorage.getItem("cart")) || [];
    cart.push(currentProduct);
    localStorage.setItem("cart", JSON.stringify(cart));
```

```
    alert("Товар додано до кошика");
};

window.buyCredit = function() {
    document.getElementById("creditModal").style.display = "flex";
};
```

ДОДАТОК Б. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерних наук

ДИПЛОМНА РОБОТА
на ступінь вищої освіти бакалавр
із спеціальності 122 Комп'ютерні науки

Розробка веб-сайту з продажу комплектуючих для пк за допомогою JavaScript та СКБД MySQL

Виконав: студент 4 курсу, групи КНД-41
Караєв Владислав Русланович
Керівник: професор кафедри Комп'ютерних наук
д.ф.-м.н., професор Шикула О.М.

Київ - 2025

1

АКТУАЛЬНІСТЬ РОБОТИ

2

Впровадженню веб-технологій у торгівлю комплектуючими для ПК часто заважають недостатній рівень цифровізації малого бізнесу, обмежене фінансування та складність використання готових комерційних платформ. Багато підприємств стикаються з труднощами при впровадженні складних CMS або дорогих SaaS-рішень, які не відповідають їхнім реальним потребам.

У той же час сучасні веб-технології дозволяють створити простий, адаптивний та ефективний інтернет-магазин, яким зможе користуватись навіть підприємець без глибоких технічних знань. Саме тому розробка веб-сайту з продажу комплектуючих ПК із базовим, але функціональним інтерфейсом, є актуальною та затребуваною в умовах цифрової трансформації.

МЕТА ДОСЛІДЖЕННЯ

Об`єкт дослідження – магазин комплектуючих ПК.

Предмет дослідження – структура, функціональний та зручний веб-сайт для продажу комплектуючих ПК.

Мета роботи – розробити сучасний, функціональний та зручний магазин для продажу комп'ютерних комплектуючих, який відповідатиме вимогам електронної комерції та буде зрозумілим для користувача.

Наукове завдання – аналіз вимог до веб-магазинів; огляд сучасних технологій веб-розробки; дослідження архітектури веб-застосунків; розробка інтерфейсу користувача; створення серверної частини та бази даних для веб-магазину.

Методи дослідження – мови HTML, CSS, JavaScript, серверне середовище Node.js, система керування базами даних MySQL.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Комп'ютерні комплектуючі — це апаратні елементи, які разом формують повноцінну систему персонального комп'ютера. Кожен з них виконує окрему функцію: обробку даних, зберігання інформації, графічне відображення, зв'язок із користувачем або зовнішніми пристроями. Усі комплектуючі взаємодіють між собою через центральну плату — материнську. Сучасні інформаційні технології стрімко розвиваються, тому ринок комплектуючих постійно оновлюється. З'являються нові стандарти, інтерфейси, підвищується енергоефективність і продуктивність компонентів. Це вимагає від продавців та користувачів обізнаності у технічних характеристиках товарів і правильного підбору сумісних пристроїв.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Актуальність розробки веб-сайту з продажу комплектуючих ПК

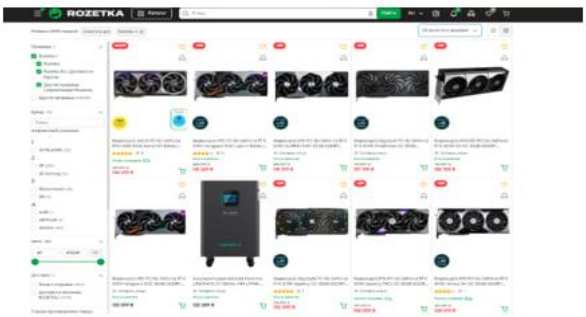
Сфера комп'ютерних комплектуючих вирізняється високим рівнем технічної складності та динаміки, що безпосередньо впливає на вимоги до функціональності онлайн-магазинів у цій галузі. Постійне оновлення асортименту, зміна цін, поява нових поколінь процесорів, відеокарт, оперативної пам'яті чи материнських плат потребують гнучкої структури веб-сайту та зручної системи керування вмістом.

Крім великого обсягу позицій у каталозі, викликом є наявність десятків технічних характеристик для кожного товару. Покупці в цій ніші часто приймають рішення на основі детального порівняння параметрів. Тому веб-сайт має надавати не лише базову інформацію про продукт, а й дозволяти швидко візуально співставити ключові технічні особливості — тактову частоту, обсяг пам'яті, сумісність з іншими компонентами тощо.

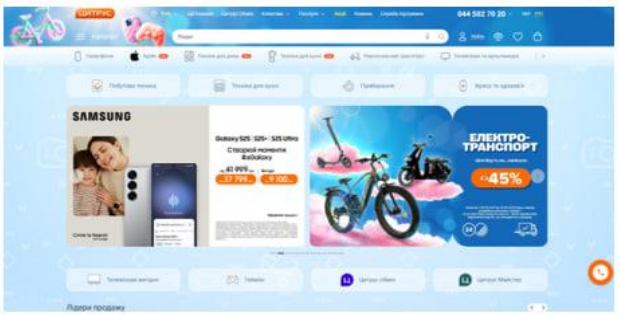
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Аналіз популярних веб-магазинів комплектуючих ПК

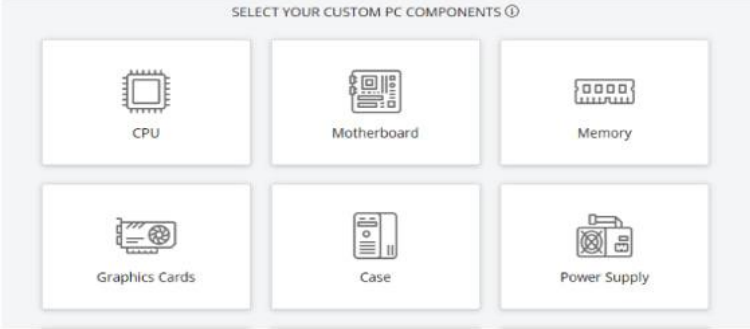
Каталог комплектуючих ПК у веб-магазині Rozetka



Основна сторінка веб-магазину "Цитрус"



Інтерфейс вибору компонентів для індивідуальної збірки ПК на веб-сайті Newegg



1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Огляд основних комп'ютерних комплектуючих

Процесори

Центральний процесор (Central Processing Unit, або CPU) є основним обчислювальним елементом персонального комп'ютера, що виконує більшість обчислень і керує роботою інших компонентів. Саме процесор обробляє інструкції програм, контролює потоки даних, проводить логічні й арифметичні операції

Відеокарти

Графічний процесор (GPU, або Graphics Processing Unit) є ключовим компонентом персонального комп'ютера, який відповідає за обробку графічних даних, виведення зображень на екран, рендеринг 3D-графіки, відео, а також за виконання обчислювальних задач, пов'язаних із візуалізацією.

Оперативна пам'ять

Оперативна пам'ять, або RAM (англ. Random Access Memory), є одним із найважливіших елементів комп'ютерної системи, що відповідає за тимчасове зберігання даних, які активно використовуються під час роботи комп'ютера.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Огляд основних комп'ютерних комплектуючих

Материнські плати

Материнська плата (англ. motherboard) є базовим апаратним вузлом будь-якої комп'ютерної системи, оскільки забезпечує фізичне й логічне з'єднання всіх основних компонентів. Саме на ній розміщуються центральний процесор (CPU), модулі оперативної пам'яті (RAM), графічна карта (GPU), накопичувачі, система живлення, периферійні пристрої тощо.

Накопичувачі даних

Накопичувачі даних є важливою складовою будь-якого комп'ютера, оскільки вони відповідають за зберігання операційної системи, програмного забезпечення, документів, мультимедійних файлів та інших даних користувача. Найпоширеніші: HDD та SSD.

2 АНАЛІЗ ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ТА ШЛЯХИ СТВОРЕННЯ ВЕБ-САЙТУ

Основи структурування та стилізації веб-сторінок

HTML (HyperText Markup Language) є основною мовою розмітки веб-сторінок і служить для формування їхньої структури. Завдяки HTML можна логічно розбити сторінку на семантичні блоки, що полегшує сприйняття вмісту як користувачами, так і пошуковими системами.

CSS (Cascading Style Sheets) відповідає за візуальне оформлення веб-сторінок. За допомогою CSS визначається кольорова схема, шрифти, розміщення елементів, анімації, відступи, межі та інші стилістичні параметри.

2 АНАЛІЗ ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ТА ШЛЯХИ СТВОРЕННЯ ВЕБ-САЙТУ

Динаміка та інтерактивність

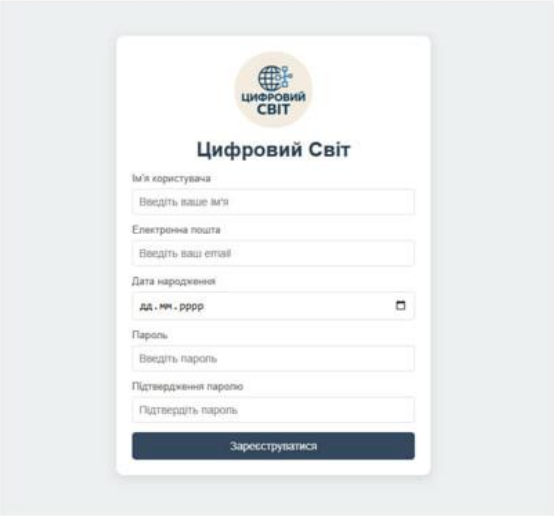
JavaScript є найпопулярнішою мовою програмування для створення динамічного і інтерактивного веб-контенту. Його основна функція — забезпечувати логіку роботи веб-сайту на стороні клієнта, реагувати на дії користувача, взаємодіяти з сервером та змінювати вміст сторінки без перезавантаження. Завдяки цій мові програмування реалізується логіка, яка робить сайт «живим»: він реагує на події користувача, оновлює дані в режимі реального часу, змінює вміст DOM (Document Object Model) та забезпечує загальну динаміку інтерфейсу.

2 АНАЛІЗ ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ТА ШЛЯХИ СТВОРЕННЯ ВЕБ-САЙТУ

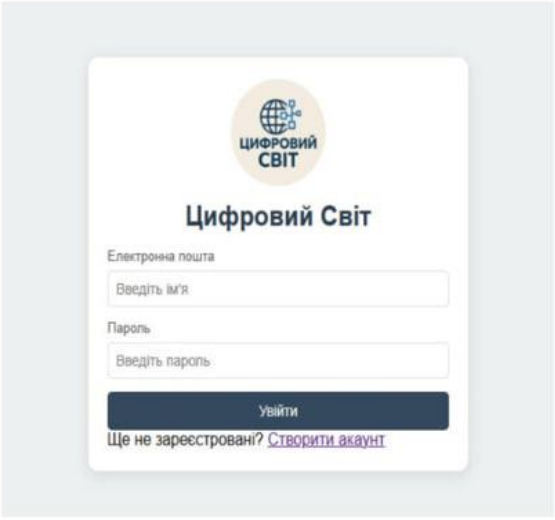
Система керування базами даних MySQL

У межах реалізації проєкту для зберігання товарів використано реляційну систему керування базами даних MySQL, яка є стабільним, надійним і широко підтримуваним рішенням для веб-застосунків. Обрана технологія дозволяє ефективно працювати зі структурованою інформацією та забезпечує високу швидкодію при взаємодії з великим каталогом продукції. Підключення до бази даних MySQL реалізовано за допомогою бібліотеки Sequelize — ORM (Object-Relational Mapping), яка дозволяє працювати з базою у вигляді JavaScript-об'єктів. Замість написання SQL-запитів вручну.

3. ОПИС ПРОГРАМНОГО ПРОДУКТУ

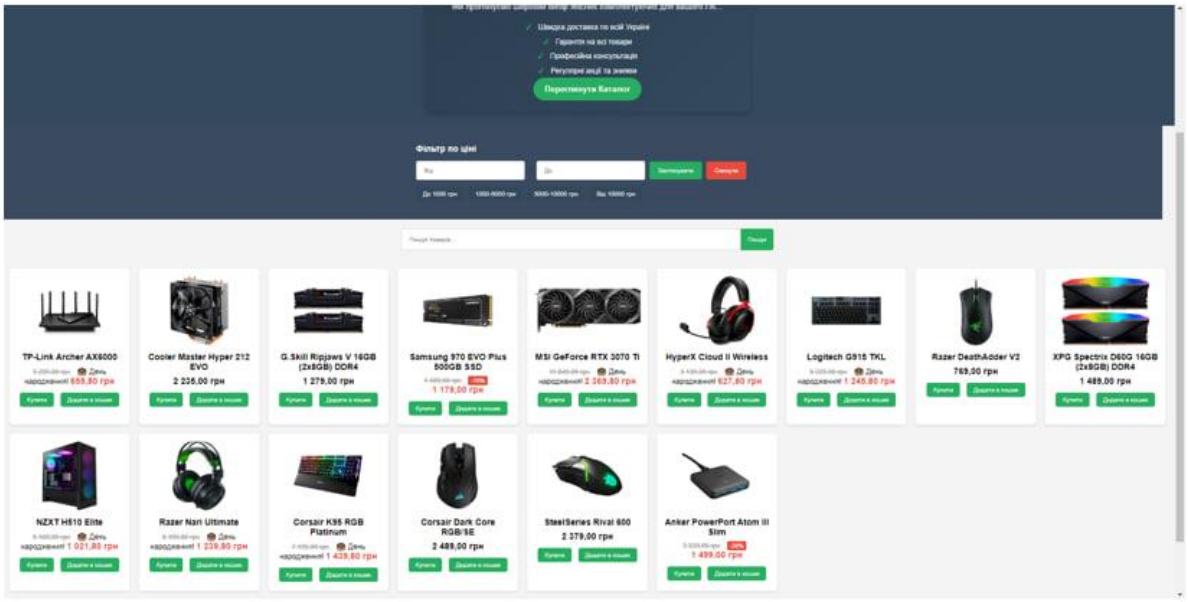


Форма реєстрації нового користувача



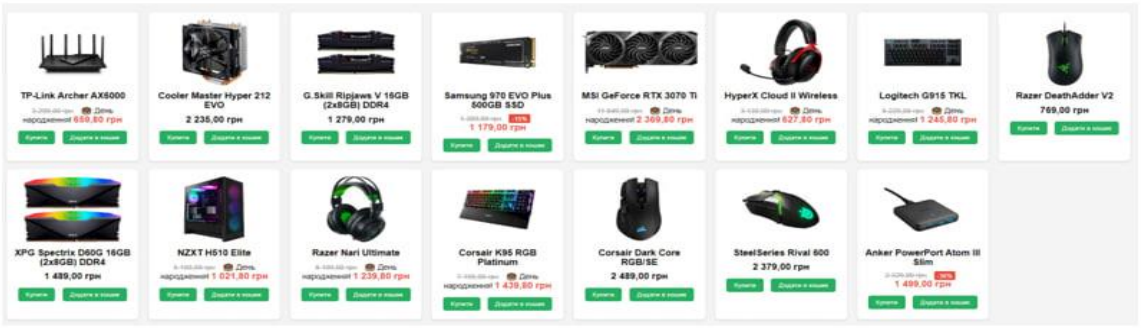
Форма авторизації користувача

3. ОПИС ПРОГРАМНОГО ПРОДУКТУ



Головна сторінка веб-сайту

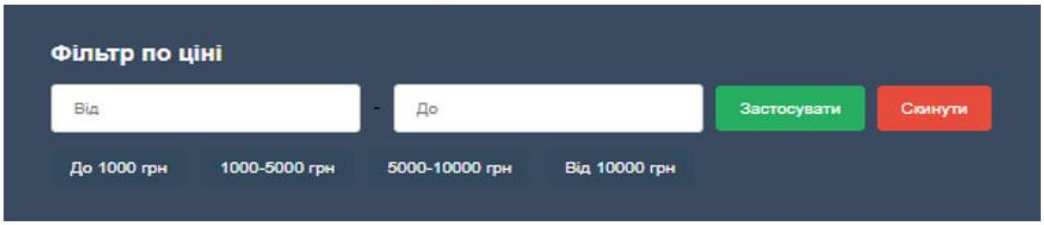
3. ОПИС ПРОГРАМНОГО ПРОДУКТУ



Каталог товарів

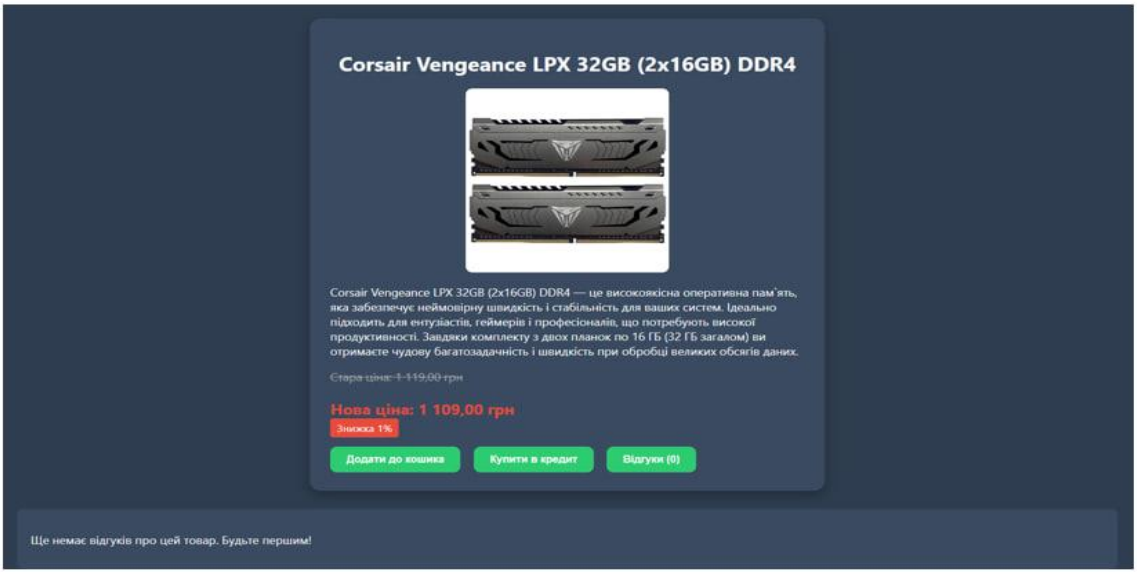
3. ОПИС ПРОГРАМНОГО ПРОДУКТУ

Фільтрація за ціною



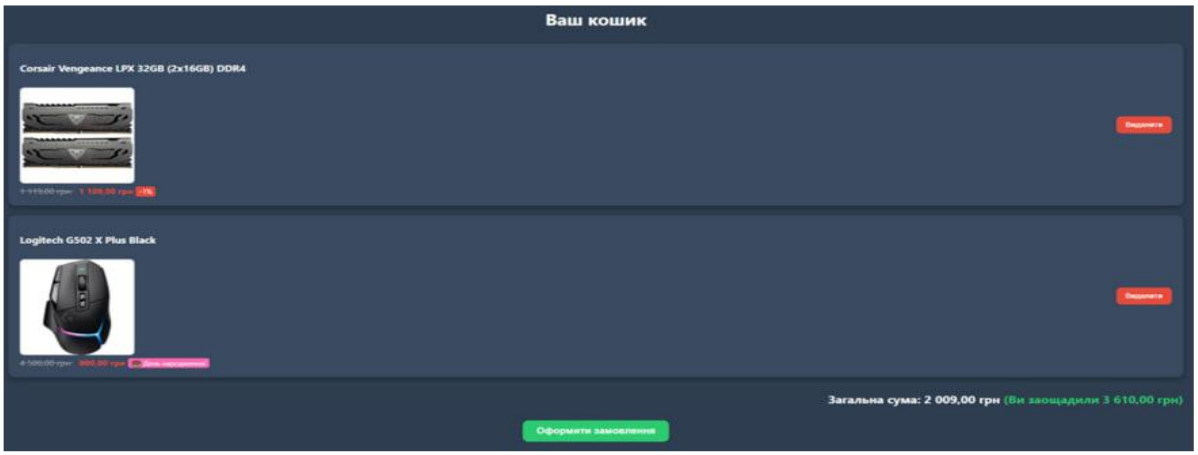
Фільтрація за пошуком

3. ОПИС ПРОГРАМНОГО ПРОДУКТУ



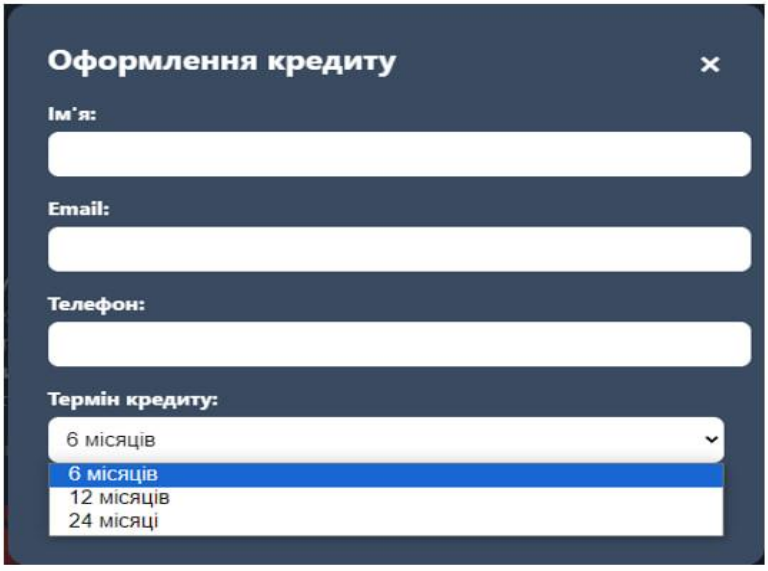
Перегляд окремого товару

3. ОПИС ПРОГРАМНОГО ПРОДУКТУ



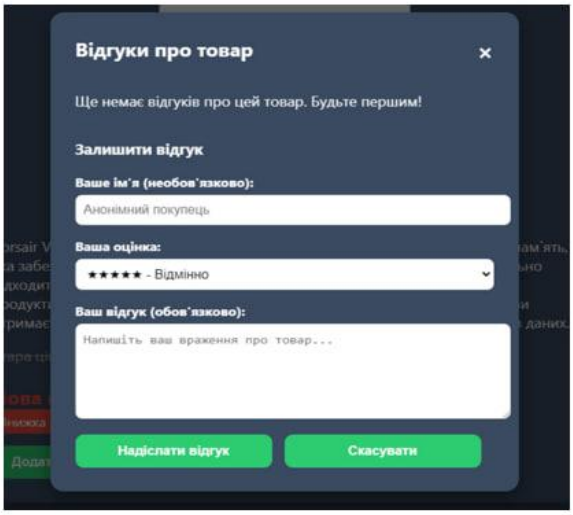
Кошик покупок

3. ОПИС ПРОГРАМНОГО ПРОДУКТУ

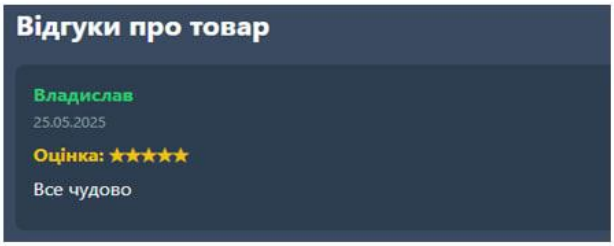


Форма оформлення покупки в кредит

3. ОПИС ПРОГРАМНОГО ПРОДУКТУ

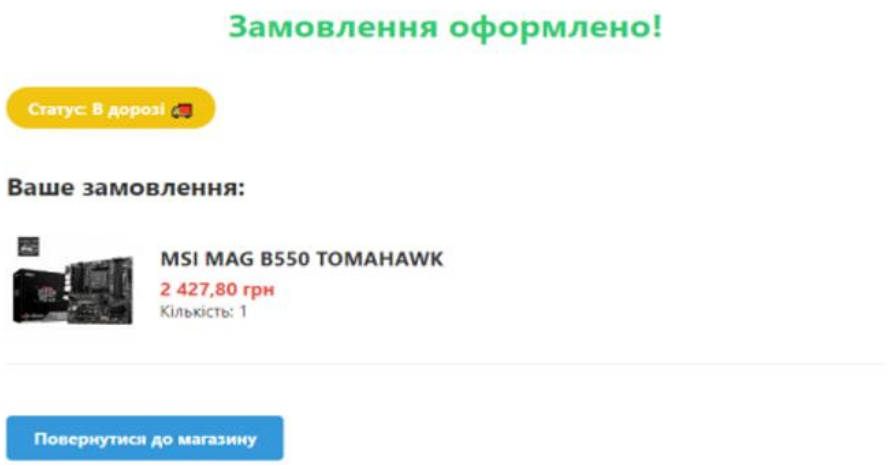


Форма залишення відгуку про товар



Приклад відгуку

3. ОПИС ПРОГРАМНОГО ПРОДУКТУ



Сторінка успішного оформлення замовлення

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було здійснено комплексну розробку веб-сайту для продажу комп'ютерних комплектуючих. Проєкт об'єднав як дослідження предметної області та аналіз існуючих аналогів сучасних веб-технологій, так і практичне втілення повноцінного функціонального веб-магазину з урахуванням вимог зручності, безпеки та ефективності.

На основі попереднього аналізу ринку електронної комерції та існуючих рішень було визначено ключові вимоги до структури, інтерфейсу та функціональності майбутнього сайту. У результаті реалізовано сайт, який дозволяє користувачам переглядати товари, здійснювати пошук і фільтрацію, додавати товари до кошика, авторизуватись, залишати відгуки та бачити персональні знижки в день народження.

ВИСНОВКИ

Для реалізації веб-магазину були використані такі технології:
• HTML та CSS — для створення структури і стилю сторінок з адаптивною верстkою;
• JavaScript — для реалізації клієнтської логіки, динамічного оновлення інтерфейсу, обробки подій, фільтрації, кошика та взаємодії з користувачем;
• Node.js — як серверне середовище виконання JavaScript для обробки HTTPзапитів, підключення до бази даних та передачі даних до клієнта;
• MySQL — для збереження інформації про товари;
• LocalStorage — для збереження даних користувача, кошика та знижок без необхідності взаємодії з сервером.

У ході роботи було виконано такі завдання:
• розроблено архітектуру веб-додатку та базову модель бази даних;
• реалізовано інтерфейс із зручною навігацією, адаптивним дизайном та зрозумілою структурою;
• впроваджено персоналізовану знижку на день народження, що стимулює активність користувачів;
• забезпечено локальне збереження кошика, що дозволяє формувати замовлення в декілька етапів;
• реалізовано фільтрацію товарів за ціною та пошук за назвою, що спрощує користувачам взаємодію з каталогом.

ВИСНОВКИ

Практична цінність проєкту полягає в тому, що створена система може бути використана як основа або готове рішення для малого або середнього бізнесу. Завдяки своїй гнучкості та простоті у розгортанні, проєкт не потребує складного обслуговування, що робить його доступним для широкого кола користувачів.

Перспективи розвитку передбачають впровадження авторизації через базу даних, реалізацію особистого кабінету користувача, панелі адміністратора, інтеграцію з платіжними системами, додавання багатомовності та аналітики замовлень. Також можливе перенесення збереження кошика на сервер, що забезпечить синхронізацію між різними пристроями.

У процесі виконання роботи було досягнуто поставленої мети: створено сучасний, адаптивний, функціональний веб-сайт, який відповідає актуальним вимогам веб-розробки та демонструє ефективність використання відкритих технологій у сфері електронної комерції.