

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ТЕМИ ДЛЯ ПЛАТФОРМИ SHOPIFY З
ВИКОРИСТАННЯМ LIQUID

DEVELOPMENT OF A SHOPIFY THEME USING LIQUID

спеціальність 122 Комп'ютерні науки

освітня програма «Комп'ютерні науки»

Виконав: здобувач вищої освіти
групи КН-42
Літвінчук Андрій Андрійович

(підпис)

Керівник: асистент
Сачук Вікторія Олегівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«__» _____ 2026 р.
Гарант освітньої програми:
д.пед.н., професор
Тулашвілі Юрій Йосипович

(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра комп'ютерних наук

Ступінь вищої освіти: бакалавр

Галузь знань: 12 Інформаційні технології

Спеціальність: 122 Комп'ютерні науки

Освітня програма: «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Валерій ЛІЩИНА

« ____ » _____ 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ПЕРШОГО (БАКАЛАВРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Літвінчука Андрія Андрійовича

(прізвище, ім'я, по-батькові)

1. Тема кваліфікаційної роботи « Розробка теми для платформи Shopify з використанням Liquid ».

Керівник асистент Сачук Вікторія Олегівна,

затверджені наказом закладу вищої освіти від « ____ » _____ 2026 р. № 32/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи « ____ » червня 2026 р.

3. Вихідні дані до роботи: опублікувати зарубіжні та вітчизняні дослідження у сфері розробки інтернет-магазинів, документація платформи Shopify, документація мови шаблонів Liquid, технології веброзробки, принципи побудови клієнт-серверних застосунків, методи зберігання та обробки даних електронної комерції, протоколи безпеки вебзастосунків, методи адаптивної верстки, інструменти тестування працездатності та сумісності Shopify-теми.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): Аналіз предметної області та існуючих рішень електронної комерції. Аналіз і вибір технологій розробки. Проектування структури та функціонального наповнення інтернет-магазину. Розробка кастомної Shopify-теми та функціональних модулів системи. Побудова функціонально-структурної схеми роботи системи. Тестування, налагодження та оцінка ефективності вебзастосунку.

5. Перелік графічного матеріалу: 23 рисунків, презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
<i>Аналіз проблеми за темою роботи та постановка завдань дослідження</i>	<i>Сачук В. О</i>		
<i>Теоретичне дослідження</i>	<i>Сачук В. О</i>		
<i>Практична реалізація</i>	<i>Сачук В. О</i>		
<i>Спеціальні розрахунки</i>	<i>Сачук В. О</i>		
<i>Показник запозичень тексту</i>	_____ %		
<i>Інструментальна перевірка</i>	<i>Кошелюк В. А.</i>		
<i>Нормоконтроль</i>	<i>Сачук В. О.</i>		
<i>Гарант ОПП</i>	<i>Тулашвілі Ю. Й.</i>		

7. Дата видачі завдання « » 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	<i>Провести огляд літературних джерел по темі кваліфікаційної роботи</i>	<i>до 17.02.2026 р</i>	
2	<i>Провести аналіз загальної проблеми і вибір напрямків дослідження</i>	<i>до 28.02.2026 р.</i>	
3	<i>Розробити функціональну схему роботи програмного продукту</i>	<i>до 12.03.2026 р</i>	
4	<i>Описати засоби розробки об'єкта проектування</i>	<i>до 26.03.2026 р.</i>	
5	<i>Практична реалізація об'єкта проектування</i>	<i>до 30.04.2026 р.</i>	
6	<i>Провести спеціальні розрахунки</i>	<i>до 18.05.2026 р.</i>	
7	<i>Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі</i>	<i>до 02.06.2026 р.</i>	

Здобувач вищої освіти _____ Андрій ЛІТВІНЧУК

Керівник роботи _____ Вікторія САЧУК

АНОТАЦІЯ

Літвінчук А. А. Розробка інтернет-магазину електроніки на платформі Shopify з використанням кастомної теми. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерні науки». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі виконано аналіз сучасного стану електронної комерції, розглянуто проблеми розробки інтернет-магазинів та проведено порівняльний аналіз сучасних платформ для створення систем електронної торгівлі. Обґрунтовано вибір платформи Shopify та технологій розробки вебзастосунку. Другий розділ містить специфікацію функціональних і нефункціональних вимог до системи, опис архітектури інтернет-магазину, функціонально-структурної схеми та моделі даних платформи Shopify. Також наведено UML-діаграми та опис основних компонентів системи. У третьому розділі здійснено практичну реалізацію інтернет-магазину електроніки на платформі Shopify із використанням мови шаблонізації Liquid, HTML, CSS та JavaScript. Реалізовано кастомну Shopify-тему, каталог товарів, сторінку продукту, систему фільтрації, cart drawer, адаптивне мобільне меню та механізми пошуку товарів. Проведено тестування та налагодження вебзастосунку, а також описано процес інсталяції та використання програмного продукту. Четвертий розділ містить спеціальні розрахунки, пов'язані з SEO-оптимізацією інтернет-магазину, оптимізацією продуктивності вебзастосунку та забезпеченням адаптивності інтерфейсу. У висновках узагальнено результати виконаної роботи та визначено перспективи подальшого розвитку системи.

Ключові слова: Shopify, інтернет-магазин, електронна комерція, вебзастосунок, Liquid, HTML, CSS, JavaScript, SEO-оптимізація, адаптивний дизайн, cart drawer, e-commerce.

ANNOTATION

Andrii Litvinchuk. Development of an Electronics Online Store on the Shopify Platform Using a Custom Theme. Manuscript.

Bachelor's Qualification Thesis in the Educational Program "Computer Science". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, four chapters, conclusions, a list of references, and appendices.

The first chapter analyzes the current state of electronic commerce, examines the main challenges of online store development, and provides a comparative analysis of modern e-commerce platforms. The choice of the Shopify platform and web development technologies is substantiated. The second chapter contains the specification of functional and non-functional requirements for the system, a description of the online store architecture, the functional-structural scheme, and the Shopify data model. UML diagrams and descriptions of the main system components are also presented. The third chapter describes the practical implementation of an electronics online store on the Shopify platform using the Liquid templating language, HTML, CSS, and JavaScript. A custom Shopify theme, product catalog, product page, filtering system, cart drawer, adaptive mobile menu, and product search mechanisms were implemented. Testing and debugging of the web application were carried out, and the installation and usage process of the software product was described. The fourth chapter contains special calculations related to SEO optimization of the online store, web application performance optimization, and interface adaptability. The conclusions summarize the results of the completed work and define the prospects for further system development.

Keywords: Shopify, online store, electronic commerce, web application, Liquid, HTML, CSS, JavaScript, SEO optimization, responsive design, cart drawer, e-commerce.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1 Аналіз сучасного стану електронної комерції та проблеми розробки інтернет-магазинів.....	10
1.2 Аналіз існуючих платформ та аналогічних рішень	13
1.3 Аналіз і вибір технологій розробки інтернет-магазину	15
1.4 Постановка завдання на кваліфікаційну роботу бакалавра	18
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ	19
2.1 Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання.....	19
2.2 Функціонально-структурна схема роботи об'єкта проектування	21
2.3 Аналіз моделі даних платформи Shopify	26
РОЗДІЛ 3 РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ НА ПЛАТФОРМІ SHOPIFY .	28
3.1 Практична реалізація інтернет-магазину на платформі Shopify	28
3.2 Тестування та налагодження інтернет-магазину	55
3.3 Посібник з інсталяції та використання програмного продукту	62
РОЗДІЛ 4 СПЕЦІАЛЬНІ РОЗРАХУНКИ	67
ВИСНОВКИ.....	72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	74
ДОДАТКИ.....	77

ВСТУП

У сучасному світі інформаційні технології відіграють вирішальну роль у розвитку економіки та суспільства загалом. Одним із найбільш динамічних напрямків є електронна комерція (e-commerce), яка надає можливість проведення торговельних операцій через мережу Інтернет. Стрімке зростання популярності онлайн-купівель зумовлене низкою чинників, серед яких зручність доступу до товарів і послуг, збереження часу, широкий вибір продукції та можливість зіставлення цін.

З розвитком електронної комерції зростають і вимоги до інтернет-магазинів. Сучасні вебресурси повинні гарантувати не лише функціональність, але й високий ступінь зручності користування, швидкодію, адаптивність до різних пристроїв та привабливий дизайн. Особливо важливим є користувацький досвід (User Experience, UX), який прямо впливає на рівень конверсії та успішність бізнесу.

Звичний підхід до формування інтернет-магазинів передбачає створення багатокомпонентних програмних систем із застосуванням серверної частини, клієнтського інтерфейсу та сховищ даних. Такий метод вимагає суттєвих затрат часу, фінансів та високого рівня технічної обізнаності. З огляду на це, великого поширення набули платформи класу SaaS (Software as a Service), які дають змогу будувати інтернет-магазини без потреби розробки громіздкої інфраструктури.

Однією з найбільш популярних платформ у сфері електронної торгівлі є Shopify. Вона пропонує користувачам повний набір засобів для створення, адаптування та адміністрування інтернет-магазину. До головних переваг Shopify можна зарахувати легкість застосування, наявність готових зразків, з'єднання з платіжними сервісами, потенціал зростання та масштабування.

Водночас, типові теми Shopify мають певні обмеження, пов'язані з однорідністю дизайну та функціоналу. Це породжує потребу у створенні індивідуальних тем і секцій, які дають змогу втілити унікальний вигляд сайту та додаткові функціональні можливості. Саме формування персоналізованих

рішень є нагальним завданням для розробників та дизайнерів у сфері електронної комерції.

Актуальність роботи полягає у необхідності розробки сучасного інтернет-магазину з використанням платформи Shopify із впровадженням унікальних дизайнерських та функціональних рішень, що забезпечують підвищення якості користувацького досвіду та конкурентоспроможності вебресурсу.

Метою кваліфікаційної роботи є розробка інтернет-магазину електроніки на базі платформи Shopify шляхом створення власної теми та реалізації кастомних функціональних елементів.

Для досягнення поставленої мети необхідно вирішити такі завдання на кваліфікаційну роботу:

- провести аналіз сучасного стану електронної комерції;
- дослідити існуючі платформи для створення інтернет-магазинів;
- обґрунтувати вибір технологій розробки;
- спроектувати структуру інтернет-магазину;
- розробити кастомну тему Shopify;
- реалізувати унікальні функціональні секції;
- провести тестування та налагодження системи;
- оцінити ефективність розробленого рішення;

Об'єктом дослідження є процес створення інтернет-магазину у середовищі електронної комерції.

Предметом дослідження є методи, технології та засоби розробки Shopify-тем, а також шляхи вдосконалення споживчого досвіду завдяки впровадженню індивідуальних компонентів інтерфейсу.

Наукова новизна роботи полягає у розробці кастомної теми для платформи Shopify із використанням унікальних секцій та покращених підходів до організації користувацького інтерфейсу.

Практичне значення отриманих результатів полягає у можливості використання розробленого інтернет-магазину в реальних умовах ведення комерції. Запропоновані рішення можуть бути використані для формування

комерційних веб-проектів, що задовольняють сучасним запитам до якості та спроможності.

У ході виконання роботи передбачається застосування сучасних методів аналізу, проектування та створення вебзастосунка, а також інструментарію платформи Shopify, мови шаблонів Liquid, технологій HTML, CSS та JavaScript.

Структура роботи містить вступ, чотири розділи, висновки, перелік використаних джерел та додатки. У першому розділі буде здійснено аналіз предметної області електронної комерції, сучасних платформ для створення інтернет-магазинів та технологій розробки. У другому розділі будуть окреслені вимоги до системи та представлено її архітектуру, функціонально-структурну схему й модель даних. Третій розділ буде зосереджений на практичній розробці інтернет-магазину на платформі Shopify, створенні кастомної теми та реалізації функціональних компонентів системи. У четвертому розділі будуть подані результати SEO-оптимізації, оптимізації продуктивності вебзастосунку та оцінка ефективності розробленого рішення.

Результати дослідження були представлені на міжнародній науково-практичній конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці та виробництві (ІТОНВ-2026)» (додаток А).

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз сучасного стану електронної комерції та проблеми розробки інтернет-магазинів

Електронна комерція (e-commerce) є одним із найбільш динамічних напрямів розвитку сучасної цифрової економіки [1]. Вона охоплює процеси купівлі-продажу товарів і послуг із застосуванням інформаційно-комунікаційних технологій, зокрема мережі Інтернет. Упродовж останніх років спостерігається стабільне зростання обсягів електронної торгівлі [2], що зумовлено як розвитком технологій, так і зміною поведінки покупців.

Важливим чинником розвитку електронної комерції стало широке поширення мобільних пристроїв та доступу до високошвидкісного Інтернету [3]. Користувачі отримали можливість здійснювати покупки у будь-який час і з будь-якого місця, що значно підвищило популярність інтернет-магазинів. Окрім того, онлайн-торгівля надає низку переваг порівняно з традиційними формами ведення бізнесу, зокрема зниження витрат на утримання фізичних торгових точок, розширення географії збуту та можливість автоматизації бізнес-процесів.

Водночас, сучасні інтернет-магазини мусять відповідати високим вимогам як зі сторони користувачів, так і з боку підприємництва. До головних вимог можна віднести забезпечення швидкого завантаження сторінок, адаптованості до різних типів пристроїв, зрозумілого інтерфейсу, а також високого рівня захисту даних. Особливе значення набуває користувацький досвід (UX), який визначає зручність взаємодії користувача з вебресурсом та прямо впливає на рівень конверсії [4].

Попри суттєвий розвиток технологій, процес створення інтернет-магазину лишається непростим та багатоскладовим завданням [5]. Традиційний спосіб створення вебзастосунків передбачає застосування клієнт-серверної архітектури, яка охоплює фронтенд-частину (інтерфейс користувача), бекенд-частину (серверну логіку) та сховище даних. Втілення такого підходу вимагає чималих

витрат часу, ресурсів та наявності здібних фахівців у різних сферах створення програмного забезпечення.

До головних труднощів, що постають при створенні інтернет-магазинів, можна віднести наступне.

По-перше, це складність впровадження повної функціональності. Інтернет-магазин має надавати чималий спектр можливостей, зокрема перегляд продукції, фільтрування та пошук, оформлення замовлення, інтеграцію з платіжними системами, обробку замовлень та керування клієнтами. Впровадження всіх цих функцій з нуля є клопітким процесом.

По-друге, суттєвою проблемою є гарантування високої продуктивності вебресурсу. Швидкість завантаження сторінок прямо впливає на поведінку відвідувачів [6]. Затримки навіть у кілька секунд можуть спричинити втрату потенційних покупців. Тому треба вживати оптимізовані засоби, що гарантують швидкий доступ до вмісту.

По-третє, велику складність становить адаптація інтерфейсу до різних пристроїв. Сучасні користувачі застосовують як персональні комп'ютери, так і смартфони та планшети. Це вимагає реалізації адаптивного чи респонсивного дизайну, який забезпечує правильне відображення веб-сторінок незалежно від величини екрана.

По-четверте, важливим чинником є з'єднання із зовнішніми сервісами. Інтернет-магазини мають підтримувати взаємодію з платіжними системами, службами доставки, аналітичними інструментами та іншими сторонніми сервісами. Це ускладнює процес створення та потребує додаткових технічних знань.

Ще однією вагомою перешкодою є гарантування безпеки даних. Інтернет-магазини оперують особистими даними клієнтів, а також фінансовою інформацією, що робить їх потенційною мішенню для кібернебезпек [7]. Тому треба втілювати новітні методи захисту, охоплюючи кодування даних, охорону від неправомірного доступу та дотримання норм безпеки.

Зважаючи на названі труднощі, у сфері електронної торгівлі набрали великого поширення платформи класу SaaS (Software as a Service) [8], що пропонують готові рішення для побудови інтернет-магазинів. Застосування подібних платформ дає змогу суттєво полегшити процес створення [9], оскільки вони забезпечують основну інфраструктуру, включно з розміщенням, сховищем даних, системою керування товарами та сполученням із платіжними системами.

Використання SaaS-платформ дозволяє сконцентруватися на розробці вигляду для клієнтів та вдосконаленні вражень користувача, що є головними чинниками успіху сучасного онлайн-магазину. Проте навіть у цій ситуації з'являється потреба адаптації типових рішень під особливі запити комерції, що передбачає залучення допоміжних засобів та технологій.

Таким чином, оцінка теперішнього стану електронної торгівлі демонструє, що створення дієвого інтернет-магазину є непростим завданням, яке вимагає врахування значної кількості аспектів. Залучення сучасних платформ та технологій дозволяє значно спростити цей процес, однак не усуває потреби у створенні власних рішень, спрямованих на покращення вартості та конкурентної стійкості вебзастосунку.

У рамках даної роботи об'єктом розробки є інтернет-магазин електроніки, що реалізується на платформі Shopify. Система орієнтована на забезпечення зручного перегляду товарів, оформлення замовлень та ефективної взаємодії з користувачем.

Інформаційні потоки в системі включають взаємодію користувача з веб-інтерфейсом, обробку запитів на серверній стороні платформи Shopify, а також збереження та обробку даних про товари, користувачів і замовлення. Основними функціональними складовими системи є каталог товарів, сторінка товару, кошик та модуль оформлення замовлення.

З огляду на проведений аналіз, доцільність створення даної системи обумовлена необхідністю розробки сучасного, адаптивного та зручного у використанні інтернет-магазину з можливістю кастомізації інтерфейсу. Використання платформи Shopify дозволяє значно спростити реалізацію

серверної частини, зосередивши увагу на розробці користувацького інтерфейсу та покращенні користувацького досвіду.

1.2 Аналіз існуючих платформ та аналогічних рішень

Проведений аналіз сучасних систем для розробки інтернет-магазинів [10] засвідчив, що кожна з них володіє власними характеристиками, сильні сторони та вади, які обумовлюють доцільність їх застосування у певних обставинах. Вибір відповідного програмного рішення є вагомим кроком проєктування, адже він напряду впливає на подальшу розробку, функціональність системи, її здатність до розширення, роботу та комфорт експлуатації.

Актуальні платформи електронної торгівлі можна умовно розділити на дві головні групи: SaaS-пропозиції [11] та системи з відкритим кодом. SaaS-платформи, а саме Shopify та Wix eCommerce, надають клієнтам готову інфраструктуру, що охоплює хостинг, систему керування змістом, сполучення з фінансовими послугами та технічне забезпечення. Це суттєво полегшує процес формування інтернет-магазину та дозволяє сконцентруватися на налаштуванні вигляду та бізнес-логіки. Разом з тим, подібні варіанти можуть мати обмеження стосовно глибокої кастомізації та контролю над системою.

Системи з відкритим кодом, такі як WooCommerce [12] та Magento [13], надають значно ширші можливості для кастомізації та розширення функціоналу. Вони дозволяють втілити складні бізнес-процеси та інтегрувати додаткові модулі відповідно до потреб клієнта. Проте застосування таких платформ вимагає наявності технічних знань, а також самостійного налаштування серверного оточення та подальшого супроводу системи.

Платформа Magento є одним із найбільш потужних варіантів у сфері електронної комерції. Вона сфокусована на створенні великих інтернет-магазинів та корпоративних рішень із високими вимогами до функціональності та масштабованості. Magento підтримує складні робочі сценарії, включаючи багаторівневі переліки товарів, різні типи клієнтів та зв'язок із зовнішніми

сервісами. Однак її використання пов'язане зі значною заплутаністю розробки, високими потребами до серверних ресурсів та необхідністю залучення досвідчених фахівців.

WooCommerce, зі свого боку, є популярним варіантом для створення інтернет-магазинів на базі системи управління вмістом WordPress. Його головною перевагою є відкритий код, що гарантує високу гнучкість та можливість повного керування системою. WooCommerce має велику кількість доповнень і тем, які допомагають розширити функціонал крамниці. Проте така гнучкість тягне за собою потребу самостійного налаштування хостингу, оновлення платформи та забезпечення захисту, що обтяжує процес розробки та експлуатації.

Wix eCommerce [14] є взірцем платформи, орієнтованої на швидке створення вебсайтів із мінімальними технічними знаннями. Вона надає зручний візуальний редактор та базовий набір засобів для реалізації інтернет-магазину. Це дозволяє стрімко створити простий вебресурс, проте звужує можливості кастомізації та масштабування, що робить цю платформу менш придатною для складних проєктів.

Окрему увагу варто приділити платформі Shopify [15], яка поєднує в собі переваги SaaS-рішень із достатнім рівнем гнучкості для впровадження індивідуальних рішень. Shopify забезпечує повний цикл створення інтернет-магазину, включаючи хостинг, систему управління товарами, інтеграцію з платіжними системами та інструменти аналітики. Крім того, платформа підтримує використання мови шаблонізації Liquid, що дозволяє змінювати структуру сторінок та впроваджувати кастомні функціональні елементи.

Важливою перевагою Shopify є можливість швидкого старту проєкту без потреби налаштування серверної частини. Це дозволяє суттєво скоротити час розробки та зменшити витрати на підтримку системи. Крім того, платформа забезпечує високий рівень безпеки та стабільності, що є значущим чинником для електронної комерції.

Порівняльний розгляд згаданих засобів дає змогу дійти висновку, що вибір оптимального рішення має ґрунтуватися на компромісі між складністю створення, набором функцій та реальними вимогами до проєкту. Для великих корпоративних систем слушно використовувати Magento, для гнучких варіантів із повним контролем – WooCommerce, а для нескладних розробок – Wix eCommerce.

У підсумку зробленого огляду було встановлено, що для впровадження онлайн-магазину у межах цієї кваліфікаційної роботи найбільш вдалим є застосування платформи Shopify. Це пояснюється її легкістю в експлуатації, наявністю готової інфраструктури, підтримкою новітніх технологій та здатністю створювати індивідуальні розробки. Застосування цього засобу дозволяє зосередитися на створенні інтерфейсу, поліпшенні досвіду користувача та втіленні функціональних аспектів без потреби будувати серверну частину від початку.

Порівняльний аналіз розглянутих платформ показує, що SaaS-рішення (Shopify, Wix eCommerce) забезпечують простоту використання та швидкий запуск проєкту, проте мають певні обмеження у кастомізації. У свою чергу, системи з відкритим кодом (WooCommerce, Magento) надають ширші можливості налаштування, але потребують значних технічних ресурсів та часу на розробку і підтримку.

Отже, обрана платформа відповідає поставленим завданням та забезпечує ефективну реалізацію інтернет-магазину з урахуванням сучасних вимог електронної комерції.

1.3 Аналіз і вибір технологій розробки інтернет-магазину

Після вибору платформи для реалізації інтернет-магазину наступним значущим етапом є визначення технологій, які будуть застосовані для розробки інтерфейсу користувача, впровадження функціональних можливостей та забезпечення належного рівня продуктивності системи. Вірний підбір

технологічного стеку дає змогу не лише спростити процес створення, але й забезпечити якісний фінальний результат, що відповідає сучасним вимогам електронної комерції.

Підґрунтям створення інтернет-магазину у межах цієї роботи є платформа Shopify, яка використовує власну мову шаблонізації Liquid. Liquid є серверною мовою шаблонів [16], що дозволяє динамічно створювати HTML-сторінки на підставі даних, отриманих із системи управління контентом Shopify. Завдяки цій мові можна реалізовувати показ товарів, колекцій, кошика, а також інші складові інтернет-магазину.

Важливою рисою Liquid є її безпечність та обмеженість у виконанні складної логіки, що допомагає уникнути виникнення помилок на стороні сервера. Водночас ця мова надає достатній ступінь гнучкості для створення індивідуальних шаблонів і частин, що є потрібним для реалізації унікального вигляду інтернет-магазину.

Для побудови структури веб-сторінок використовується мова розмітки HTML (HyperText Markup Language) [17]. Вона окреслює логічну структуру вмісту, включаючи заголовки, текстові блоки, зображення, кнопки та інші елементи інтерфейсу. Застосування сучасних стандартів HTML5 дозволяє забезпечити семантичність розмітки, що позитивно впливає на доступність сайту та його індексацію пошуковими системами.

Оформлення зовнішнього вигляду веб-сторінок виконується за допомогою каскадних таблиць стилів CSS (Cascading Style Sheets). CSS дає змогу задавати кольори, шрифти, відступи, розташування елементів та інші візуальні атрибути. Це дозволяє створити інтерфейс, що коректно відображається на різних пристроях, включаючи персональні комп'ютери, планшети та смартфони.

Для втілення інтерактивності застосовується мова програмування JavaScript [18]. Вона дає змогу забезпечити динамічну взаємодію користувача з веб-сторінкою без потреби її перезавантаження. Зокрема, JavaScript використовується для реалізації анімацій, обробки подій користувача, керування елементами інтерфейсу та підвищення загального досвіду користувача.

У контексті платформи Shopify JavaScript також залучається для взаємодії з кошиком, оновлення вмісту сторінки в реальному часі та реалізації індивідуальних функціональних елементів. Це дозволяє суттєво збільшити зручність користування інтернет-магазином.

Окрім головних технологій, важливе значення у розробці має система налаштування тем Shopify, яка дає змогу змінювати зміст і структуру сторінок без редагування коду. Це досягається завдяки використанню спеціальних конфігураційних файлів, що визначають параметри секцій та блоків.

Важливим аспектом є також забезпечення адаптивності інтерфейсу. Сучасні користувачі активно застосовують мобільні пристрої для доступу до інтернет-магазинів, тому потрібно забезпечити правильне відображення та комфортність використання на екранах різного розміру. Для цього використовуються медіа-запити CSS та гнучкі макети.

Зважаючи на вимоги до продуктивності, під час творення застосовуються методи оптимізації, як-от мінімізація CSS і JavaScript, використання оптимізованих зображень та зменшення числа звернень до сервера. Це дає змогу забезпечити швидке завантаження сторінок, що є значущим чинником для утримання користувачів.

Отже, обраний технологічний набір, який охоплює платформу Shopify, мову шаблонів Liquid, а також технології HTML, CSS і JavaScript, цілком відповідає потребам до творення сучасного інтернет-магазину. Застосування цих технологій дозволяє втілити функціональний, пристосований та зручний у користуванні вебресурс, що відповідає поточним тенденціям розвитку електронної комерції.

Окрім зазначених технологій, у процесі розробки використовуються сучасні інструменти програмування та тестування. Зокрема, для створення та редагування коду застосовується середовище розробки Visual Studio Code, яке забезпечує зручну роботу з файлами тем Shopify та підтримку необхідних мов програмування. Для взаємодії з платформою Shopify використовується адміністративна панель та інструменти керування темами.

Апаратно-програмні вимоги до розроблюваної системи є мінімальними. Для роботи достатньо персонального комп'ютера або ноутбука з доступом до мережі Інтернет та сучасного веб-браузера (Google Chrome, Mozilla Firefox тощо). Оскільки платформа Shopify є хмарним рішенням, відсутня необхідність у розгортанні серверної інфраструктури.

Таким чином, обрані технології та інструменти розробки забезпечують ефективне створення інтернет-магазину, спрощують процес реалізації та дозволяють отримати якісний програмний продукт, що відповідає сучасним вимогам електронної комерції.

1.4 Постановка завдання на кваліфікаційну роботу бакалавра

У сучасних умовах розвитку електронної комерції важливим завданням є створення ефективних, зручних та адаптивних інтернет-магазинів, які забезпечують високий рівень взаємодії з користувачем. Використання сучасних платформ, таких як Shopify, дозволяє значно спростити процес розробки вебзастосунків, однак потребує застосування додаткових підходів для створення унікального дизайну та функціональності.

У межах даної кваліфікаційної роботи передбачається розробка інтернет-магазину електроніки на платформі Shopify з використанням мови шаблонізації Liquid, сучасних технологій веб-розробки та підходів до створення адаптивного інтерфейсу. Основна увага приділяється реалізації кастомної теми, яка забезпечує зручний користувацький досвід та відповідає сучасним вимогам електронної комерції.

Результатом виконання кваліфікаційної роботи має стати функціонуючий інтернет-магазин, розроблений на платформі Shopify, який демонструє можливості створення кастомних тем та реалізації сучасного користувацького інтерфейсу. Розроблене рішення повинно відповідати вимогам зручності використання, продуктивності та адаптивності вебзастосунків.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання

На етапі проєктування інтернет-магазину важливим завданням є обґрунтування вибору підходів, технологій та засобів реалізації системи. Вибір технологічної бази безпосередньо впливає на ефективність розробки, складність підтримки програмного продукту, його продуктивність та можливості масштабування.

На основі проведеного аналізу предметної області та існуючих рішень, наведених у розділі 1, було визначено, що доцільним є використання платформи Shopify як основи для реалізації інтернет-магазину. Даний вибір обумовлений тим, що Shopify є сучасною SaaS-платформою, яка надає повністю готову інфраструктуру для створення, розгортання та підтримки вебзастосунків електронної комерції.

Порівняно з традиційним підходом, що передбачає розробку повноцінної клієнт-серверної системи із власною серверною логікою та базою даних, використання Shopify дозволяє суттєво спростити процес розробки [19]. Зокрема, відсутня необхідність налаштування серверного середовища, розробки API та реалізації механізмів обробки даних, оскільки ці функції вже реалізовані на рівні платформи. Це дозволяє зосередитися на створенні інтерфейсу користувача та реалізації бізнес-логіки інтернет-магазину.

Важливою перевагою Shopify є високий рівень надійності та безпеки. Платформа забезпечує захист даних користувачів, підтримує сучасні стандарти безпеки та має вбудовані механізми інтеграції з платіжними системами. Крім того, Shopify гарантує стабільну роботу системи навіть при значному навантаженні, що є важливим для сучасних інтернет-магазинів.

Як основний інструмент для формування структури веб-сторінок обрано мову шаблонізації Liquid. Дана мова використовується у середовищі Shopify для

динамічного формування HTML-сторінок на основі даних, що зберігаються у системі. Використання Liquid дозволяє гнучко керувати відображенням товарів, колекцій та інших елементів інтерфейсу.

Перевагою мови Liquid є її безпечність, оскільки вона не дозволяє виконувати довільний код на сервері, що зменшує ризик виникнення помилок та вразливостей. Водночас вона надає достатні можливості для реалізації складних шаблонів і логіки відображення даних.

Для створення інтерфейсу користувача зазвичай використовуються стандартні веб-технології HTML, CSS та JavaScript. Мова HTML застосовується для формування структури сторінок, CSS – для їх стилізації та створення адаптивного дизайну, а JavaScript – для реалізації інтерактивних елементів інтерфейсу.

Особливу увагу приділено забезпеченню адаптивності інтерфейсу та зручності використання системи на мобільних пристроях. Це дозволяє покращити взаємодію користувача з інтернет-магазином.

Використання сучасних можливостей CSS, таких як Flexbox та Grid, дозволяє створювати гнучкі та адаптивні макети, що забезпечують коректне відображення інтернет-магазину на різних пристроях. JavaScript, у свою чергу, використовується для обробки дій користувача, динамічного оновлення вмісту сторінки та реалізації інтерактивних компонентів.

Обрана схема розробки базується на поєднанні клієнтської частини, реалізованої за допомогою веб-технологій, та серверної частини, яка повністю забезпечується платформою Shopify. Такий підхід дозволяє чітко розмежувати відповідальність між компонентами системи та спростити процес розробки.

Додатковою перевагою використання платформи Shopify є наявність великої кількості готових інструментів для інтеграції сторонніх сервісів та розширення функціональності інтернет-магазину. Платформа підтримує підключення платіжних систем, служб доставки, маркетингових сервісів та аналітичних інструментів без необхідності реалізації складної серверної логіки.

Це дозволяє підвищити функціональність системи та забезпечити можливість її подальшого розвитку відповідно до потреб користувачів.

Таким чином, обрані технології та засоби реалізації є обґрунтованими з науково-технічної точки зору та відповідають сучасним вимогам до розробки інтернет-магазинів. Використання платформи Shopify, мови шаблонізації Liquid та стандартних веб-технологій дозволяє забезпечити ефективність розробки, зручність використання та високу якість кінцевого програмного продукту.

2.2 Функціонально-структурна схема роботи об'єкта проектування

Після обґрунтування вибору технологій та засобів реалізації системи наступним етапом є побудова функціонально-структурної схеми інтернет-магазину. Така схема дозволяє визначити цілі системи, її основні задачі, функціональні можливості, а також структуру та взаємодію між окремими компонентами.

Основною ціллю розроблюваної системи є забезпечення зручної, швидкої та ефективної взаємодії користувача з інтернет-магазином для перегляду товарів, їх вибору та оформлення замовлення.

Функціональні характеристики системи включають можливість перегляду товарів, їх пошуку та фільтрації, додавання товарів до кошика, редагування замовлення, введення користувацьких даних та обробку замовлень. Система також забезпечує збереження інформації та її подальше відображення у зручному для користувача вигляді.

Окрему роль у роботі системи відіграє клієнтська частина вебінтерфейсу та засоби навігації користувача.

Інформаційні потоки в системі реалізуються через взаємодію користувача, клієнтської частини веб-інтерфейсу та серверної частини платформи Shopify. Користувач надсилає запити через браузер, які обробляються сервером, після чого результати повертаються у вигляді веб-сторінок. Таким чином,

інформаційні потоки забезпечують безперервну взаємодію між користувачем та системою, що є основою ефективного функціонування інтернет-магазину.

Загальна архітектура системи представлена на рисунку 2.1.

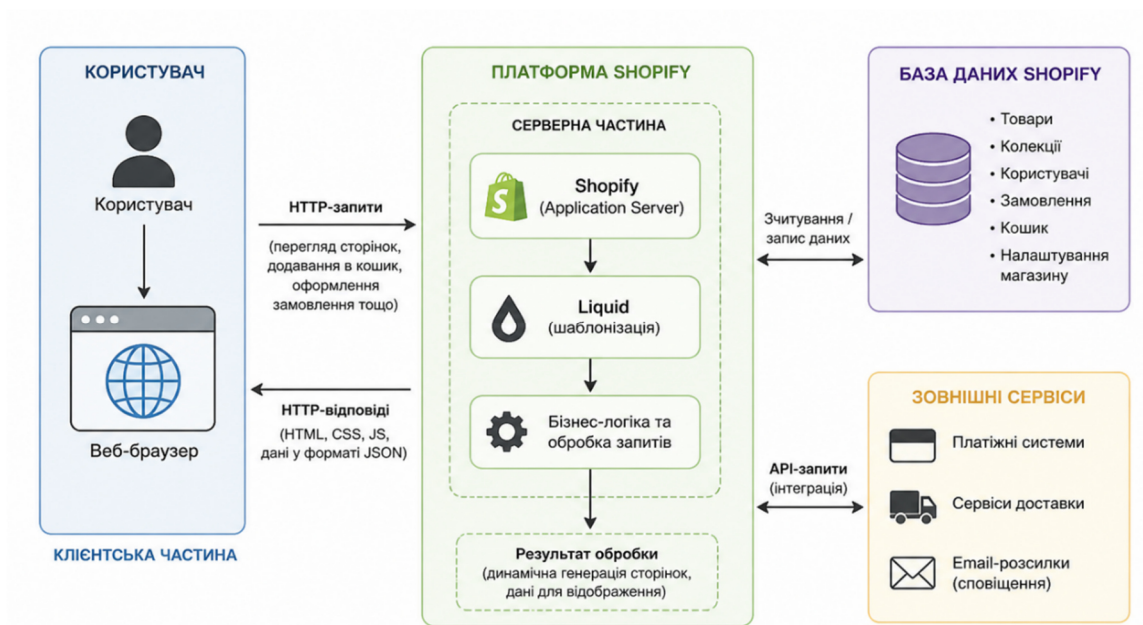


Рисунок 2.1 – Загальна архітектура інтернет-магазину

Джерело: розроблено автором

На поданій схемі зображено взаємодію користувача з веб-інтерфейсом, а також обробку запитів серверною частиною платформи Shopify та взаємодію з базою даних. Така архітектура відповідає сучасній клієнт-серверній моделі [20], де основна логіка обробки даних реалізована на стороні платформи.

Для відображення послідовності дій користувача використовується функціональна схема роботи інтернет-магазину, що наведена на рисунку 2.2.

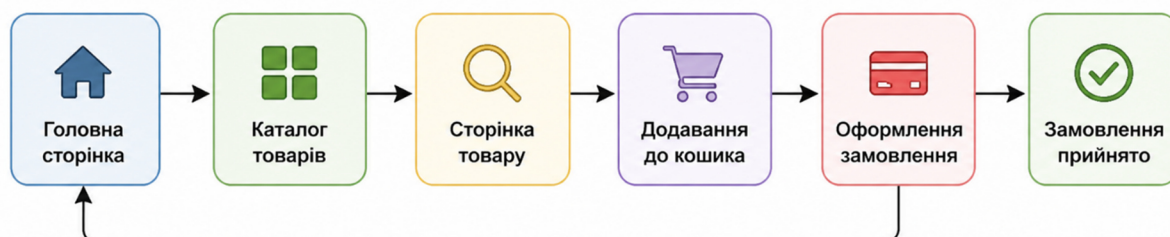


Рисунок 2.2 – Функціональна схема роботи інтернет-магазину

Джерело: розроблено автором

Дана схема демонструє основні етапи взаємодії користувача із системою: перехід на головну сторінку, перегляд каталогу товарів, вибір конкретного товару, додавання його до кошика та оформлення замовлення. Такий підхід дозволяє відобразити загальну логіку роботи інтернет-магазину та послідовність виконання основних дій користувача. Крім того, схема дає можливість визначити ключові точки взаємодії між користувачем і системою. Це спрощує подальше проєктування функціональних модулів та інтерфейсу користувача.

Наступним елементом опису є компонентна структура системи, що наведена на рисунку 2.3.

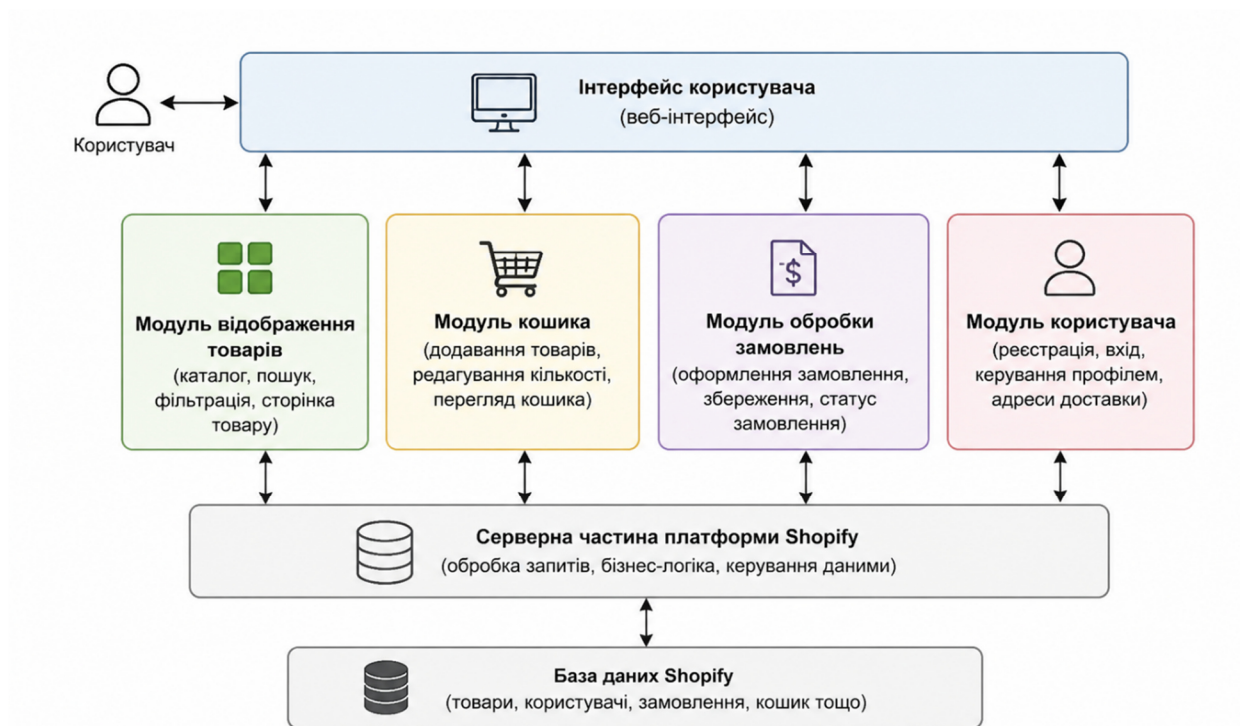


Рисунок 2.3 – Компонентна структура інтернет-магазину

Джерело: розроблено автором

Компонентна структура включає інтерфейс користувача, модуль відображення товарів, модуль кошика та модуль обробки замовлень. Кожен із компонентів виконує окрему функцію, а їх взаємодія забезпечує повноцінну роботу системи. Важливою особливістю структури є модульність компонентів, що дозволяє спростити підтримку та подальше розширення функціональних можливостей інтернет-магазину.

Для більш детального аналізу структури системи наведена UML-діаграма класів, яка зображена на рисунку 2.4.

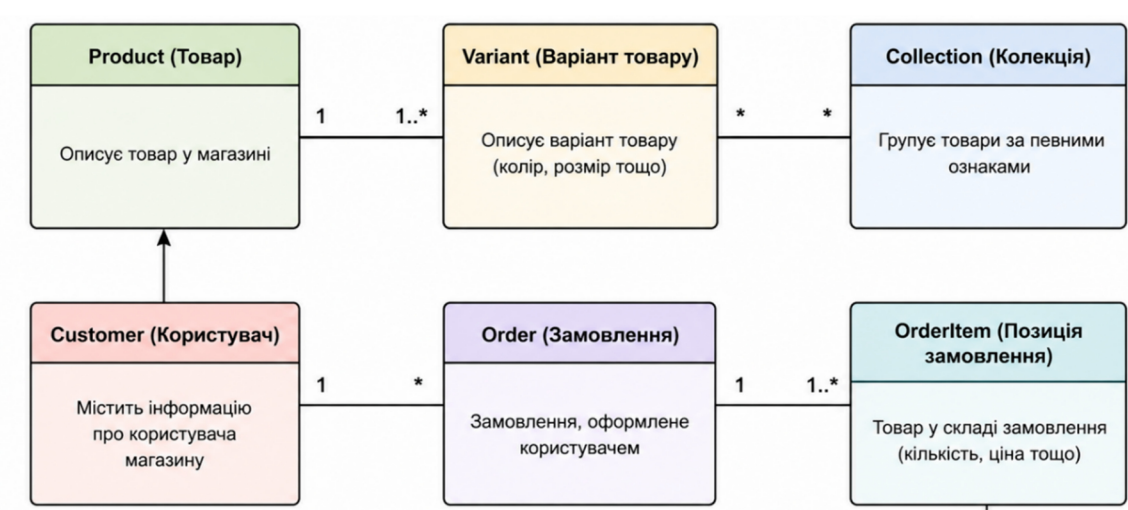


Рисунок 2.4 – UML-діаграма класів інтернет-магазину

Джерело: розроблено автором

На діаграмі відображено основні сутності системи, зокрема товар, варіант товару, користувач та замовлення, а також зв’язки між ними. Це дозволяє наочно представити логіку організації даних у системі.

Для представлення взаємодії між основними частинами системи використовується діаграма компонентів, що наведена на рисунку 2.5.

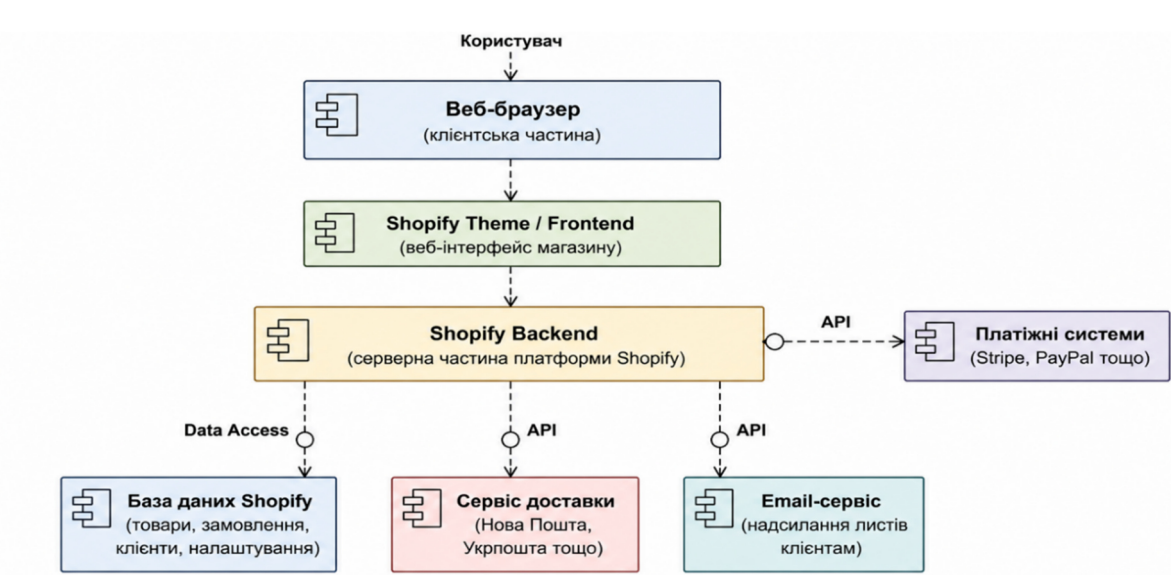


Рисунок 2.5 – Діаграма компонентів інтернет-магазину

Джерело: розроблено автором

На діаграмі показано взаємодію між інтерфейсом користувача, серверною частиною платформи Shopify та зовнішніми сервісами, такими як платіжні системи. Це дозволяє оцінити розподіл функціональності між компонентами системи.

Для детальнішого опису процесу оформлення замовлення доцільно використати діаграму послідовності, яка наведена на рисунку 2.6.

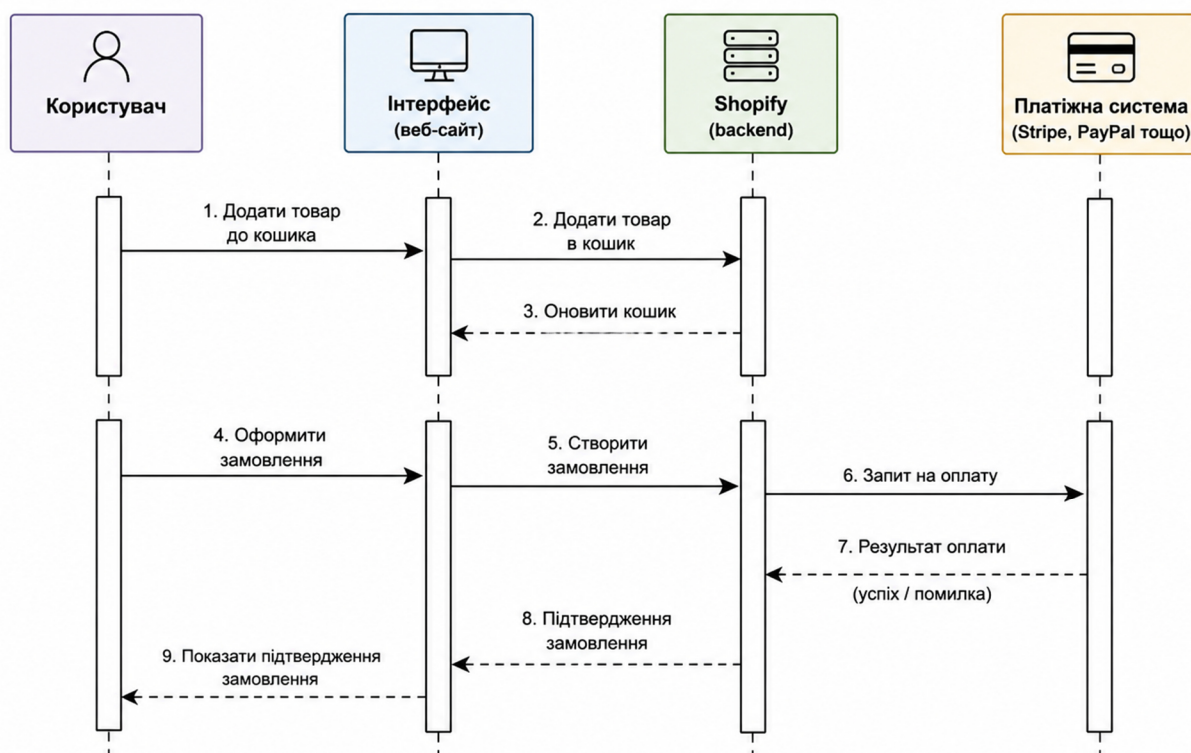


Рисунок 2.6 – Діаграма послідовності оформлення замовлення

Джерело: розроблено автором

Дана діаграма ілюструє обмін повідомленнями між користувачем, інтерфейсом та серверною частиною системи під час додавання товару до кошика та оформлення замовлення.

Таким чином, функціонально-структурна схема дозволяє сформулювати цілісне уявлення про архітектуру інтернет-магазину, його основні компоненти та логіку роботи. Використання платформи Shopify значно спрощує реалізацію системи, оскільки більшість серверних процесів вже реалізовані на рівні

платформи, що дозволяє зосередитися на створенні інтерфейсу користувача та реалізації функціональних можливостей.

2.3 Аналіз моделі даних платформи Shopify

Важливим етапом проєктування інтернет-магазину є аналіз моделі даних, яка використовується для зберігання, обробки та відображення інформації. У випадку розробки системи на базі платформи Shopify відсутня необхідність створення власної бази даних, оскільки платформа надає готову модель даних, що реалізована на рівні сервісу.

Shopify використовує централізовану систему збереження даних, яка забезпечує роботу з основними об'єктами інтернет-магазину, такими як товари, колекції, користувачі та замовлення.

Основною сутністю системи є товар (Product), який представляє окрему одиницю продукції, доступну для придбання. Товар містить базову інформацію, таку як назва, опис, ціна та зображення. Для представлення різних варіантів одного товару використовується сутність варіанта товару (Variant), яка дозволяє задавати додаткові характеристики, наприклад колір або розмір.

Ще однією важливою сутністю є колекція (Collection), яка використовується для групування товарів за певними ознаками. Це дозволяє організувати каталог продукції та забезпечити зручну навігацію для користувача. Один товар може належати до кількох колекцій, що забезпечує гнучкість структури.

Сутність користувача (Customer) використовується для збереження інформації про клієнтів інтернет-магазину. Вона містить дані, необхідні для оформлення замовлення, такі як ім'я, контактна інформація та адреса доставки.

Замовлення (Order) є ключовою сутністю, яка фіксує факт покупки. Воно містить інформацію про обрані товари, їх кількість, загальну суму та дані користувача. Замовлення пов'язане з користувачем і товарами, що забезпечує цілісність даних у системі.

Додатково у системі використовується сутність кошика (Cart), яка дозволяє тимчасово зберігати обрані товари перед оформленням замовлення. Кошик забезпечує можливість редагування списку товарів та їх кількості.

Зв'язки між основними сутностями моделі даних наведено на рисунку 2.7.



Рисунок 2.7 – Логічна модель даних інтернет-магазину на платформі Shopify

Джерело: розроблено автором

Користувач може оформлювати декілька замовлень, кожне з яких містить інформацію про вибрані товари. Таким чином, реалізується зв'язок між користувачем та товарами через сутність замовлення.

Доступ до даних у системі здійснюється за допомогою вбудованих механізмів платформи Shopify, зокрема через мову шаблонізації Liquid, яка дозволяє отримувати та відображати дані у веб-інтерфейсі. Це усуває необхідність прямої взаємодії з базою даних та спрощує процес розробки.

Таким чином, модель даних платформи Shopify забезпечує повноцінну підтримку функціонування інтернет-магазину та дозволяє реалізувати всі необхідні операції з товарами, користувачами та замовленнями без створення власної бази даних. Це значно спрощує розробку та підвищує надійність системи.

РОЗДІЛ 3

РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ НА ПЛАТФОРМІ SHOPIFY

3.1 Практична реалізація інтернет-магазину на платформі Shopify

Практична реалізація інтернет-магазину виконувалася на платформі Shopify із використанням мови шаблонізації Liquid та сучасних вебтехнологій HTML, CSS і JavaScript. Основною метою розробки було створення адаптивного та функціонального інтернет-магазину електроніки, який забезпечує зручну взаємодію користувача із системою та підтримує можливість подальшого масштабування й кастомізації.

На відміну від класичних клієнт-серверних вебзастосунків, платформа Shopify надає готову серверну інфраструктуру для реалізації електронної комерції. Завдяки цьому процес розробки зосереджується переважно на створенні користувацького інтерфейсу, реалізації структури сторінок та адаптації функціоналу магазину під потреби користувача. Такий підхід дозволяє значно скоротити час розробки та спростити підтримку системи. Крім того, використання Shopify забезпечує високу стабільність роботи системи та підтримку сучасних засобів електронної комерції. Платформа також надає можливість інтеграції із зовнішніми сервісами та платіжними системами.

Архітектура Shopify базується на використанні тем, які складаються з окремих шаблонів, секцій, фрагментів коду та ресурсів інтерфейсу. У процесі розробки було створено кастомну Shopify-тему, структура якої організована відповідно до принципів модульного проєктування. Це дозволяє ізолювати окремі елементи системи, повторно використовувати компоненти та спрощує подальше внесення змін у програмний код.

Основними директоріями теми є layout, templates, sections, snippets, assets та config. Кожна із них виконує окрему функцію та відповідає за певний етап формування інтерфейсу користувача. Зокрема, директорія layout містить глобальні шаблони сторінок, templates відповідає за структуру окремих типів сторінок, sections використовується для створення динамічних блоків

інтерфейсу, а snippets дозволяє повторно використовувати окремі фрагменти коду, структуру директорій проєкту зображено на рисунку 3.1.

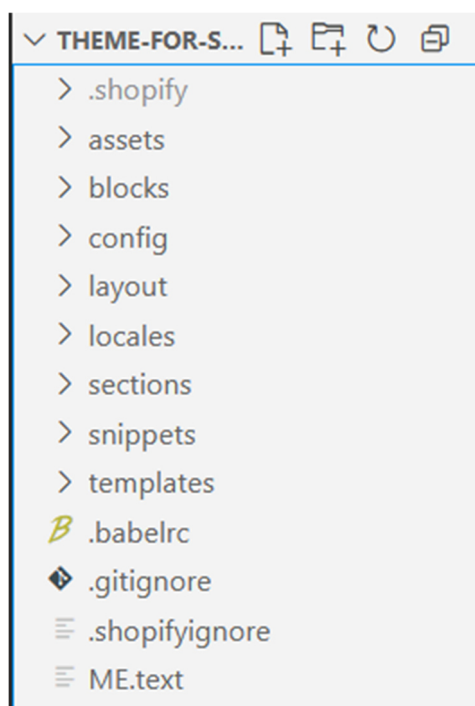


Рисунок 3.1 – Структура директорій Shopify-теми

Джерело: розроблено автором

Директорія `assets` використовується для зберігання файлів стилів, JavaScript-коду, шрифтів та графічних ресурсів. Це дозволяє централізовано керувати всіма статичними файлами інтерфейсу та спрощує процес оптимізації продуктивності вебзастосунку. Папка `config` містить конфігураційні файли теми, які визначають параметри кастомізації інтерфейсу через адміністративну панель Shopify.

Основою всієї Shopify-теми є файл `theme.liquid`, який розміщується у директорії `layout`. Саме цей файл виконує роль головного шаблону системи та відповідає за формування базової HTML-структури всіх сторінок інтернет-магазину. Через нього здійснюється підключення стилів, JavaScript-файлів, службових компонентів Shopify та динамічного контенту.

Файл `theme.liquid` також забезпечує інтеграцію окремих секцій інтерфейсу, таких як `header`, `footer`, `navigation` та інших компонентів, що використовуються

на різних сторінках системи (ліст. 3.1). Використання єдиного шаблону дозволяє забезпечити однакову структуру всіх сторінок магазину та спрощує підтримку проєкту.

Лістинг 3.1 – Підключення основних компонентів у файлі theme.liquid

```
<!doctype html>
<html lang="{{ request.locale.iso_code }}">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>{{ page_title }}</title>
  {{ 'base.css' | asset_url | stylesheet_tag }}
  {{ 'custom.css' | asset_url | stylesheet_tag }}
  {{ 'animations.css' | asset_url | stylesheet_tag }}
  {{ 'custom-slider.js' | asset_url | script_tag }}
  {{ content_for_header }}
</head>
<body>
  {% sections 'header-group' %}
  <main id="MainContent">
    {{ content_for_layout }}
  </main>
  {% sections 'footer-group' %}
</body>
</html>;
```

кінець лістингу 3.1

У процесі формування сторінки Shopify автоматично підставляє вміст шаблонів та секцій у контейнер «content_for_layout». Завдяки цьому кожна сторінка динамічно отримує власний контент залежно від типу ресурсу та дій користувача. Наприклад, сторінка товару відображає інформацію про конкретний продукт, тоді як сторінка колекції містить перелік товарів певної категорії.

Використання глобального шаблону забезпечує централізоване керування інтерфейсом системи. У разі зміни структури header, footer або глобальних стилів достатньо внести зміни лише в одному файлі, після чого оновлення автоматично застосуються до всіх сторінок інтернет-магазину. Це значно спрощує підтримку та модернізацію теми.

Під час розробки інтерфейсу використовувалася мова шаблонізації Liquid, яка є стандартним інструментом платформи Shopify для формування динамічного контенту. Liquid дозволяє отримувати дані про товари, колекції, користувачів та інші об'єкти системи без необхідності реалізації власної серверної логіки.

Мова Liquid використовує спеціальні конструкції для роботи з даними, серед яких особливе місце займають цикли, умови та фільтри. Це дозволяє гнучко керувати структурою сторінок і формувати HTML-вміст залежно від інформації, що надходить із серверної частини Shopify (ліст. 3.2).

Лістинг 3.2 – Приклад використання циклу Liquid для відображення товарів

```
{%- for product in collection.products -%}
  {% assign lazy_load = false %}
  {% if forloop.index > 2 -%}
    {% assign lazy_load = true -%}
  {% endif -%}
  <li
    class="grid__item"
    {% if settings.animations_reveal_on_scroll %}
      data-cascade
      style="--animation-order: {{ forloop.index }};"
    {% endif %}
  >
    {% render 'card-product', card_product: product %}
  </li>
  {% assign skip_card_product_styles = true -%}
{% endfor -%}
```

кінець лістингу 3.2

Під час виконання циклу система автоматично отримує інформацію про товари, включаючи їх назву, вартість, зображення та посилання на сторінку продукту. Отримані дані використовуються для формування карток товарів у каталозі та на головній сторінці інтернет-магазину.

Важливою перевагою Liquid є високий рівень безпеки. На відміну від серверних мов програмування загального призначення, Liquid не дозволяє виконувати довільний програмний код на сервері. Це мінімізує ризик виникнення помилок та потенційних вразливостей у системі.

Після реалізації базової структури теми було створено основні секції головної сторінки інтернет-магазину. Головна сторінка містить банер, секції популярних товарів, блоки категорій та інформаційні елементи, які забезпечують навігацію користувача по магазину.

Для створення гнучкого інтерфейсу використовувалися Shopify Sections, які дозволяють адміністратору змінювати структуру сторінки через Theme Editor без необхідності редагування програмного коду. Кожна секція реалізується у вигляді окремого Liquid-файлу та може містити власні параметри налаштування. Результат використання секцій у структурі головної сторінки інтернет-магазину наведено на рисунку 3.2.

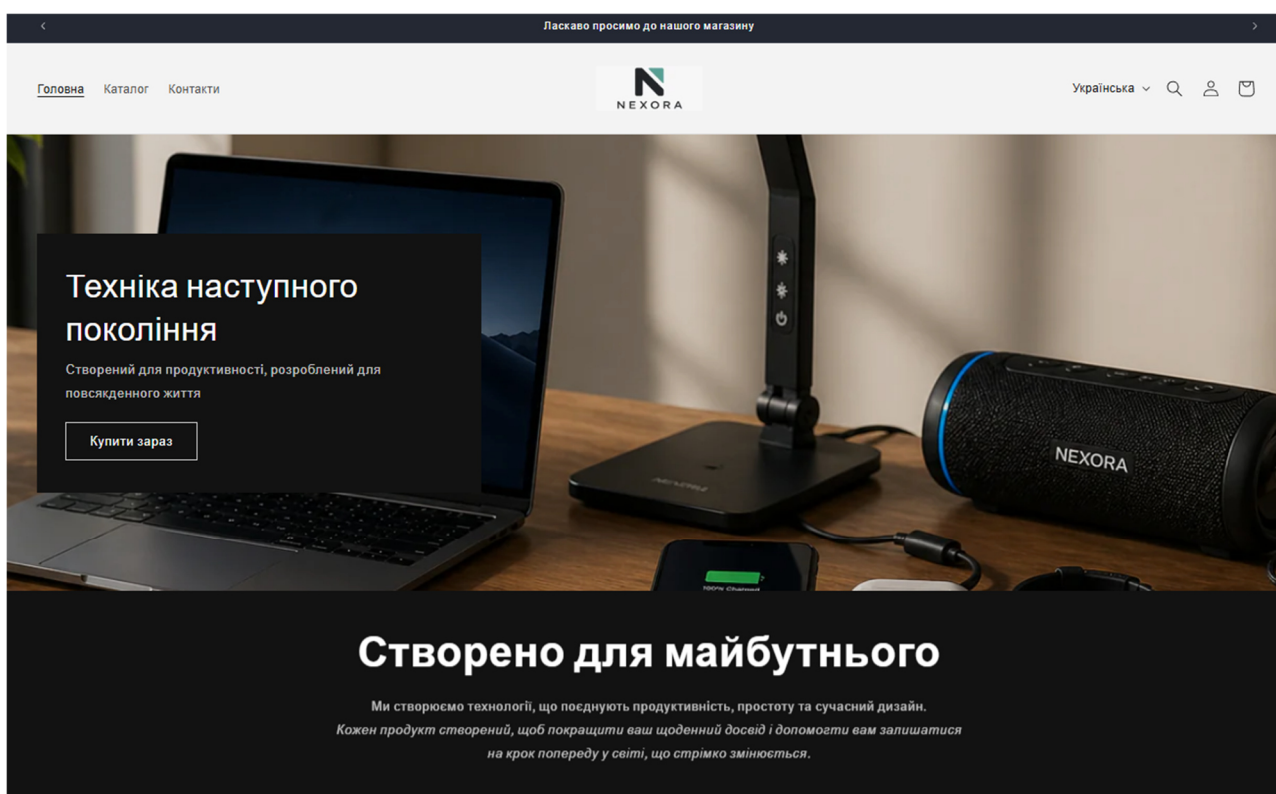


Рисунок 3.2 – Головна сторінка розробленого інтернет-магазину

Джерело: розроблено автором

Використання секцій дозволяє формувати динамічний інтерфейс, структура якого може змінюватися залежно від потреб адміністратора магазину. Наприклад, окремі блоки можуть додаватися, видалятися або змінювати порядок відображення без внесення змін у вихідний код теми.

Під час реалізації секцій використовувалася schema-конфігурація Shopify, яка визначає параметри налаштування компонентів через графічний інтерфейс адміністративної панелі. Завдяки цьому адміністратор може змінювати текстові елементи, зображення, кнопки та інші параметри секцій (ліст. 3.3). Повна schema-конфігурація секції наведена у додатку Б.

Лістинг 3.3 – Schema-конфігурація Shopify-секції

```
{% schema %}
{
  "name": "Benefits",
  "settings": [
    {
      "type": "header",
      "content": "Section"
    },
    ...
  ],
  "blocks": [
    {
      "name": "Benefit",
      "type": "benefit",
      "settings": [
        {
          "type": "checkbox",
          "id": "benefit_enabled",
          "label": "Enable",
          "default": true
        },
        ...
      ]
    }
  ]
}
{% endschema %}
```

кінець лістингу 3.3

Schema-конфігурація дозволяє створювати повністю налаштовувані компоненти інтерфейсу. Це значно підвищує гнучкість системи та дозволяє адаптувати зовнішній вигляд інтернет-магазину під різні потреби без втручання у програмний код.

Для стилізації сторінок використовувалися сучасні можливості CSS, зокрема Flexbox та Grid Layout. Використання цих технологій дозволило

створити адаптивний інтерфейс, який коректно відображається на персональних комп'ютерах, планшетах та мобільних пристроях.

Grid Layout використовувався для побудови структури каталогу товарів та сітки контенту, тоді як Flexbox застосовувався для вирівнювання елементів навігації, кнопок та окремих компонентів інтерфейсу (ліст. 3.4).

Лістинг 3.4 – Використання CSS Grid для побудови каталогу товарів

```
.how-it-works-steps {  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));  
  position: relative;  
}  
  
.how-it-works-step-number {  
  display: inline-flex;  
  align-items: center;  
  justify-content: center;  
  border-radius: 50%;  
  font-weight: 700;  
  position: relative;  
  z-index: 2;  
}  
  
.how-it-works-step-image {  
  width: 100%;  
  overflow: hidden;  
}
```

кінець лістингу 3.4

Використання Grid Layout дозволяє автоматично адаптувати кількість колонок каталогу залежно від розміру екрана користувача. Це забезпечує коректне відображення товарів на різних типах пристроїв та покращує загальний користувацький досвід.

Для мобільних пристроїв додатково використовувалися media queries, які змінюють структуру сторінки та розміри окремих елементів інтерфейсу. Такий підхід забезпечує адаптивність вебзастосунку та зручну взаємодію із системою на різних пристроях.

Після реалізації базової структури інтерфейсу було виконано розробку каталогу товарів та сторінки окремого продукту. Дані компоненти є ключовими елементами інтернет-магазину, оскільки саме через них користувач взаємодіє із товарами та оформлює замовлення.

Для реалізації каталогу використовувалися стандартні можливості Shopify Collections, які дозволяють об'єднувати товари за категоріями та автоматично формувати сторінки каталогу.

Кожен товар у каталозі відображається у вигляді окремої картки, яка містить зображення, назву, ціну та кнопку переходу до сторінки продукту. Для формування карток товарів використовувалися snippets-компоненти, що дозволило уникнути дублювання коду (рис. 3.3).

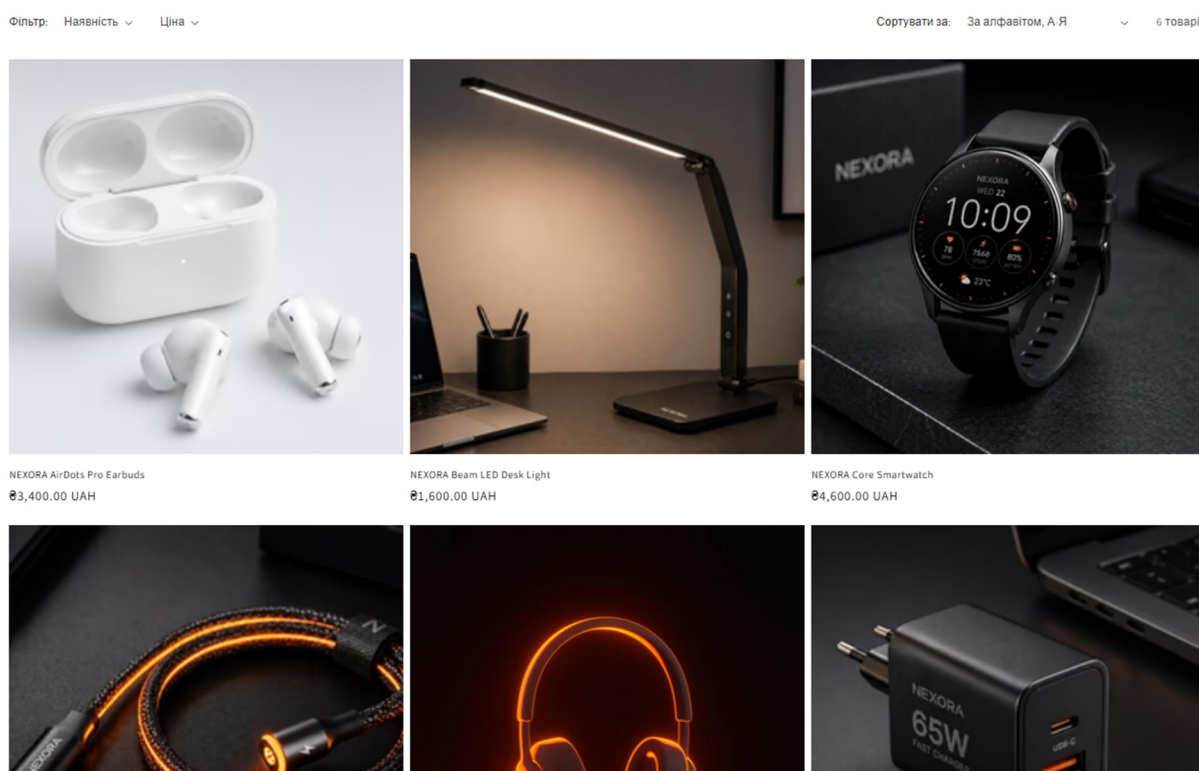


Рисунок 3.3 – Каталог товарів інтернет-магазину

Джерело: розроблено автором

Під час реалізації каталогу особлива увага приділялася адаптивності структури сторінки. Для цього використовувалася комбінація CSS Grid та Flexbox, що дозволило автоматично змінювати кількість колонок залежно від

ширини екрана користувача. Такий підхід забезпечує коректне відображення каталогу як на персональних комп'ютерах, так і на мобільних пристроях.

Для оптимізації швидкодії використовувалося відкладене завантаження зображень товарів. Це дозволяє зменшити обсяг даних, які завантажуються під час відкриття сторінки, та позитивно впливає на швидкість роботи інтернет-магазину і загальну продуктивність системи (ліст. 3.5).

Лістинг 3.5 – Реалізація картки товару у каталозі

```
<div class="card-wrapper product-card-wrapper">
  ...
  <div class="card card--media">
    <div class="card__media">
      
    </div>
    ...
    <div class="card__content">
      <div class="card__information">
        <h3 class="card__heading">
          <a href="{{ card_product.url }}">
            {{ card_product.title | escape }}
          </a>
        </h3>
        ...
        {% render 'price',
          product: card_product,
          show_compare_at_price: true
        %}
      </div>
    </div>
  </div>
</div>
```

кінець лістингу 3.5

У наведеному фрагменті коду використовується об'єкт «card_product», який містить інформацію про товар. З його допомогою система автоматично отримує необхідні дані та формує HTML-структуру картки продукту.

Використання параметра «loading="lazy"» забезпечує відкладене завантаження зображень, що дозволяє покращити продуктивність сторінки.

Для підвищення зручності користування в каталозі було реалізовано систему фільтрації товарів. Користувач може швидко переглядати продукцію певної категорії та сортувати товари за різними параметрами. Це дозволяє значно спростити процес пошуку необхідної продукції (ліст. 3.6).

Лістинг 3.6 – Реалізація фільтрації товарів

```
{%- for filter in results.filters -%}
  <details
    id="Details-{{ filter.param_name | escape }}-{{ section.id }}"
    class="facets__disclosure-vertical js-filter"
  >
    <summary class="facets__summary caption-large">
      <span>{{ filter.label | escape }}</span>
    </summary>
    <ul class="facets__list list-unstyled" role="list">
      {%- for value in filter.values -%}
        <li class="facets__item">
          <label class="facets__label">
            <input
              type="checkbox"
              name="{{ value.param_name }}"
              value="{{ value.value }}"
              {% if value.active %}checked{% endif %}
              {% if value.count == 0 and ... %}disabled{% endif %}
            >
          </label>
        </li>
      {%- endfor -%}
    </ul>
  </details>
{%- endfor -%}
```

кінець лістингу 3.6

Реалізована система фільтрації дозволяє динамічно змінювати перелік товарів залежно від обраних параметрів. Shopify автоматично обробляє запити фільтрації та формує оновлений список товарів без необхідності реалізації окремої серверної логіки.

Додатково система фільтрації підтримує автоматичне оновлення активних параметрів без необхідності ручного перезавантаження сторінки користувачем.

Крім того, реалізований підхід спрощує підтримку каталогу та дозволяє швидко додавати нові параметри фільтрації через адміністративну панель. Це позитивно впливає на зручність навігації та загальний користувацький досвід під час роботи з інтернет-магазином.

Після реалізації каталогу було виконано створення сторінки окремого товару. Дана сторінка містить розширену інформацію про продукцію, зображення, характеристики, варіанти товару та функціонал додавання продукції до кошика. Загальний вигляд сторінки окремого товару наведено на рисунку 3.4.

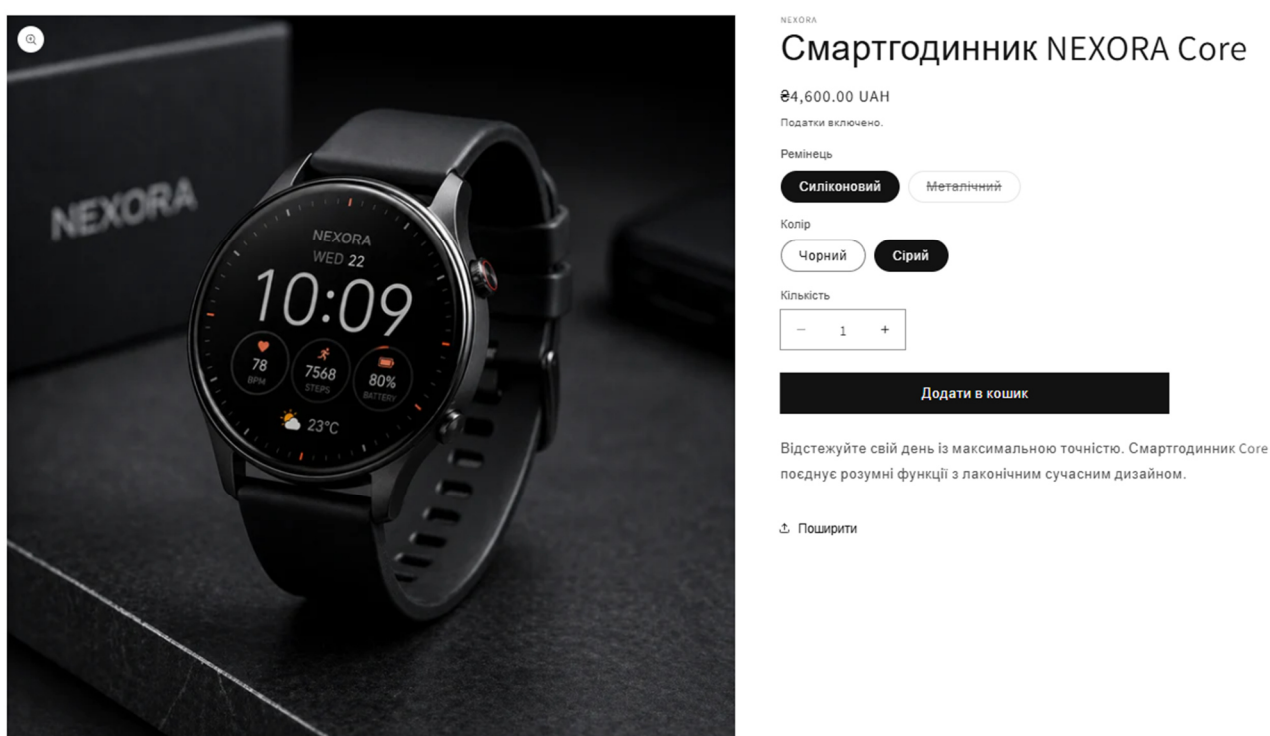


Рисунок 3.4 – Сторінка окремого товару

Джерело: розроблено автором

Для реалізації сторінки продукту використовувався шаблон `product.liquid`, який автоматично отримує інформацію про товар із системи Shopify. На сторінці реалізовано галерею зображень, блок характеристик та механізм вибору варіантів товару.

Однією з важливих функцій сторінки є можливість вибору модифікацій товару. Наприклад, користувач може обирати колір, обсяг пам'яті або інші

характеристики продукції. Для цього використовувалися Shopify Variants, які дозволяють зберігати декілька конфігурацій одного товару (ліст. 3.7).

Лістинг 3.7 – Відображення варіантів товару

```
{%- unless product.has_only_default_variant -%}
<variant-selects id="variant-selects-{{ section.id }}"
  data-section="{{ section.id }}"
>
  {%- for option in product.options_with_values -%}
    <div class="product-form__input product-form__input--dropdown">
      <label class="form__label" for="...">
        {{ option.name }}
      </label>
      <div class="select">
        <select
          id="Option-{{ section.id }}-{{ forloop.index0 }}"
          class="select__select"
          name="options[{{ option.name | escape }}"
          form="{{ product_form_id }}"
        >
          {% render 'product-variant-options',
            product: product, option: option,
            block: block, picker_type: picker_type
          %}
        </select>
      </div>
    </div>
  {%- endfor -%}
  <script type="application/json" data-selected-variant>
    {{ product.selected_or_first_available_variant | json }}
  </script>
</variant-selects>
{%- endunless -%}
```

кінець лістингу 3.7

Використання variants дозволяє автоматично змінювати ціну, доступність та інші параметри товару залежно від вибраної конфігурації. Це забезпечує більш зручну взаємодію користувача із системою та спрощує процес оформлення замовлення.

Для реалізації функціоналу додавання товару до кошика використовувалися стандартні можливості Shopify Cart API та JavaScript. Після

натискання кнопки Add to Cart система формує асинхронний запит до сервера та додає товар у кошик без повного перезавантаження сторінки (ліст. 3.8).

Лістинг 3.8 – Додавання товару до кошика за допомогою JavaScript

```
onSubmitHandler(evt) {
  evt.preventDefault();
  if (this.submitButton.getAttribute('aria-disabled') === 'true') return;
  this.submitButton.setAttribute('aria-disabled', true);
  this.submitButton.classList.add('loading');
  const config = fetchConfig('javascript');
  config.headers['X-Requested-With'] = 'XMLHttpRequest';
  const formData = new FormData(this.form);
  config.body = formData;
  fetch(`${routes.cart_add_url}`, config)
    .then((response) => response.json())
    .then((response) => {
      publish(PUB_SUB_EVENTS.cartUpdate, {
        source: 'product-form',
        productVariantId: formData.get('id'),
        cartData: response,
      });
      this.cart.renderContents(response);
    })
    .finally(() => {
      this.submitButton.classList.remove('loading');
      this.submitButton.removeAttribute('aria-disabled');
    });
});
};
```

кінець лістингу 3.8

Використання асинхронних запитів дозволяє значно покращити користувацький досвід, оскільки оновлення кошика відбувається без повторного завантаження сторінки. Крім того, такий підхід забезпечує швидшу взаємодію користувача із системою та робить інтерфейс більш сучасним.

Для візуального відображення стану кошика було реалізовано спливаючий кошик (cart drawer), який автоматично оновлює кількість товарів та загальну суму замовлення після виконання операції додавання продукції. Даний компонент дозволяє користувачу швидко переглядати поточне замовлення без необхідності переходу на окрему сторінку кошика. Такий підхід покращує

зручність використання інтернет-магазину та прискорює процес оформлення покупки. Реалізацію спливаючого кошика наведено на рисунку 3.5.

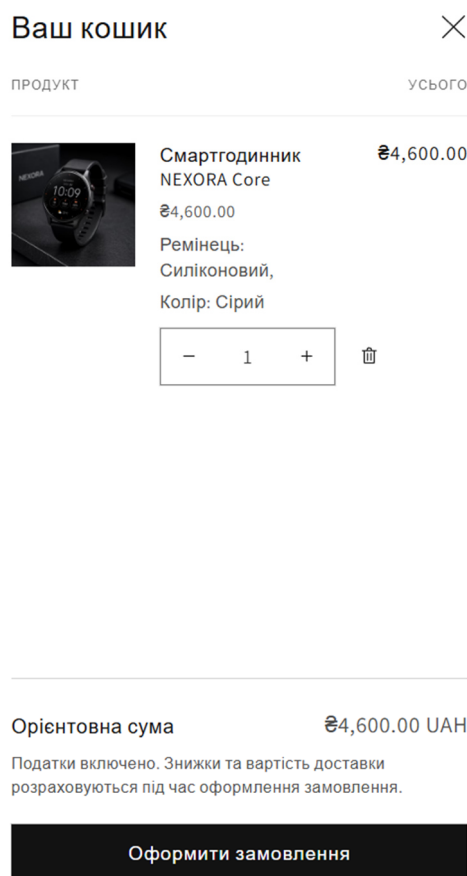


Рисунок 3.5 – Реалізація спливаючого кошика (cart drawer)

Джерело: розроблено автором

Для оновлення вмісту cart drawer використовувався JavaScript-механізм динамічного рендерингу компонентів. Після отримання відповіді від сервера система автоматично оновлює HTML-вміст окремих секцій кошика та відображає актуальну інформацію про замовлення. Такий механізм забезпечує синхронізацію стану кошика із серверною частиною Shopify та коректне відображення інформації про товари, їх кількість і загальну вартість замовлення. Крім того, динамічне оновлення компонентів підвищує інтерактивність інтерфейсу та покращує зручність використання інтернет-магазину. Також використання асинхронного оновлення дозволяє уникнути повного

перезавантаження сторінки під час взаємодії користувача з кошиком. Фрагмент реалізації механізму оновлення cart drawer наведено у лістингу 3.9.

Лістинг 3.9 – Оновлення вмісту cart drawer

```
renderContents(parsedState) {
  this.querySelector('.drawer__inner').classList.contains('is-empty') &&
    this.querySelector('.drawer__inner').classList.remove('is-empty');
  this.getSectionsToRender().forEach((section) => {
    const sectionElement = section.selector
      ? document.querySelector(section.selector)
      : document.getElementById(section.id);
    if (!sectionElement) return;
    sectionElement.innerHTML = this.getSectionInnerHTML(
      parsedState.sections[section.id],
      section.selector
    );
  });
  setTimeout(() => {
    this.open();
  });
}
```

кінець лістингу 3.9

У наведеному фрагменті коду реалізовано механізм оновлення окремих секцій кошика після додавання товару. Метод `renderContents()` проходить по всіх елементах, які необхідно оновити, та динамічно змінює їх HTML-вміст на основі відповіді сервера.

Після успішного оновлення інтерфейсу до cart drawer автоматично додається CSS-клас `active`, який відповідає за відображення спливаючого вікна кошика. Завдяки цьому користувач одразу отримує візуальне підтвердження успішного додавання товару до замовлення.

Для реалізації анімації відкриття кошика використовувалися CSS transition та JavaScript-обробники подій. Це дозволило зробити взаємодію із системою більш плавною та покращити візуальне сприйняття інтерфейсу користувачем.

Додатково у cart drawer реалізовано можливість зміни кількості товарів без переходу на окрему сторінку кошика. Після зміни кількості продукції система

автоматично виконує перерахунок вартості замовлення та оновлює відображувані дані у режимі реального часу (ліст. 3.10).

Лістинг 3.10 – Оновлення кількості товарів у кошику

```
<quantity-input class="quantity cart-quantity">
  <button
    class="quantity__button"
    name="minus" type="button"
  >
    <span class="svg-wrapper">
      {{- 'icon-minus.svg' | inline_asset_content -}}
    </span>
  </button>
  <input
    class="quantity__input"
    type="number" name="updates[]"
    value="{{ item.quantity }}" min="0"
    step="{{ item.variant.quantity_rule.increment }}"
    id="Drawer-quantity-{{ item.index | plus: 1 }}"
    data-index="{{ item.index | plus: 1 }}"
  >
  <button class="quantity__button" name="plus" type="button">
    <span class="svg-wrapper">
      {{- 'icon-plus.svg' | inline_asset_content -}}
    </span>
  </button>
</quantity-input>
```

кінець лістингу 3.10

У наведеному фрагменті коду реалізовано елемент керування кількістю товарів у кошику. Для цього використовуються кнопки збільшення та зменшення кількості продукції, а також поле введення числового значення quantity. Поточна кількість товару автоматично підставляється із об'єкта item, що містить інформацію про позицію у кошику.

Параметр step визначає крок зміни кількості товару відповідно до правил, заданих для конкретного варіанта продукції. Це дозволяє враховувати обмеження на мінімальну кількість або кратність замовлення окремих товарів. Крім того, використання data-атрибутів спрощує подальшу взаємодію JavaScript-коду із компонентами cart drawer.

Реалізований механізм забезпечує динамічне оновлення кошика без повторного завантаження сторінки та дозволяє користувачу швидко змінювати параметри замовлення. Це позитивно впливає на зручність користування інтернет-магазином та покращує загальний користувацький досвід.

Після реалізації функціоналу cart drawer було виконано розробку навігаційної системи інтернет-магазину. Основними елементами навігації є header, головне меню, пошук товарів та мобільне меню, які забезпечують швидкий доступ користувача до основних розділів вебзастосунку. Навігаційна система була реалізована таким чином, щоб забезпечити зручну взаємодію із магазином як на персональних комп'ютерах, так і на мобільних пристроях.

Header інтернет-магазину містить логотип, меню навігації, іконку кошика, пошук товарів та допоміжні елементи інтерфейсу. Для побудови структури header використовувалися Shopify Sections та Flexbox-розмітка, що дозволило забезпечити гнучке розташування компонентів інтерфейсу залежно від ширини екрана користувача (рис. 3.6).

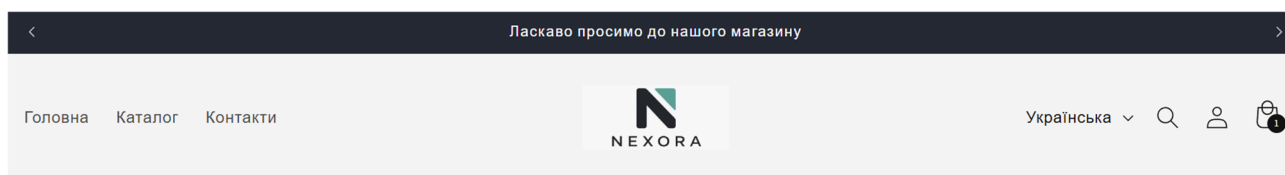


Рисунок 3.6 – Header та навігаційна система інтернет-магазину

Джерело: розроблено автором

Для реалізації меню навігації використовувався об'єкт linklists, який дозволяє автоматично отримувати структуру меню із адміністративної панелі Shopify. Це забезпечує можливість змінювати пункти навігації без необхідності редагування програмного коду теми (ліст. 3.11).

Лістинг 3.11 – Реалізація головного меню навігації

```
<nav class="header__inline-menu">
  <ul class="list-menu list-menu--inline" role="list">
    {% for link in section.settings.menu.links -%}
      <li>
```

```

    {%- if link.links != blank -%}
    <details id="Details-HeaderMenu-{{ forloop.index }}">
      <summary class="header__menu-item">
        <span>{{- link.title | escape -}}</span>
      </summary>
      <ul class="header__submenu list-menu" role="list">
        {%- for childlink in link.links -%}...{%- endfor -%}
      </ul>
    </details>
    {%- else -%}
    <a href="{{ link.url }}"
      class="header__menu-item list-menu__item" >
      {{- link.title | escape -}}
    </a>
    {%- endif -%}
  </li>
{%- endfor -%}
</ul>
</nav>

```

кінець лістингу 3.11

У наведеному фрагменті коду реалізовано механізм формування головного меню інтернет-магазину. Для цього використовується цикл `for`, який проходить по всіх елементах меню та автоматично формує HTML-структуру навігації.

Кожен пункт меню містить посилання на окремий розділ магазину, наприклад каталог товарів, сторінку акцій або контактну інформацію. Використання об'єкта `section.settings.menu.links` дозволяє динамічно змінювати структуру меню через адміністративну панель Shopify.

Під час розробки навігації також було реалізовано адаптивне мобільне меню. Для цього використовувався окремий JavaScript-механізм відкриття та закриття `mobile drawer`, який активується при натисканні на кнопку `hamburger-menu`.

Додатково мобільна навігація була оптимізована для коректної роботи на різних типах пристроїв та забезпечує зручну взаємодію користувача з елементами меню при використанні сенсорного керування. Для адаптації інтерфейсу під мобільні пристрої використовувалися CSS `media queries` та гнучкі `Flexbox`-макети. Також було реалізовано автоматичне приховування окремих

елементів навігації при зменшенні ширини екрана користувача. Приклад реалізації мобільного меню інтернет-магазину наведено на рисунку 3.7.

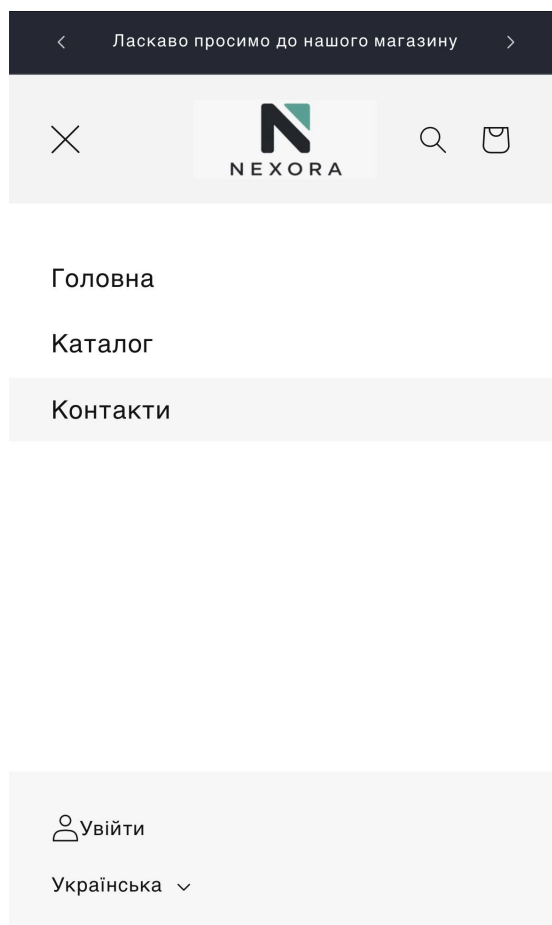


Рисунок 3.7 – Мобільне меню інтернет-магазину

Джерело: розроблено автором

Мобільне меню дозволяє зручно взаємодіяти із системою на пристроях із невеликою шириною екрана. Після відкриття drawer меню користувач отримує доступ до основних категорій товарів, сторінки кошика та інших функціональних елементів системи.

Додатково мобільне меню підтримує вкладену структуру навігації, що дозволяє групувати товари за категоріями та підкатегоріями. Такий підхід спрощує навігацію по каталогу та дозволяє швидше знаходити необхідну продукцію. Крім того, структура меню автоматично адаптується залежно від кількості доступних категорій та налаштувань Shopify. Для покращення взаємодії користувача із системою було реалізовано плавну анімацію відкриття

та закриття drawer-компонента. Механізм керування мобільним меню реалізовано за допомогою JavaScript та CSS-класів, які динамічно змінюють стан елементів інтерфейсу (ліст. 3.12).

Лістинг 3.12 – Реалізація відкриття мобільного меню

```

openMenuDrawer(summaryElement) {
  openMenuDrawer(summaryElement) {
    setTimeout(() => {
      this.mainDetailsToggle.classList.add('menu-opening');
    });
    summaryElement.setAttribute('aria-expanded', true);
    trapFocus(this.mainDetailsToggle, summaryElement);
    ...
  }
  closeMenuDrawer(event, elementToFocus = false) {
    if (event === undefined) return;
    this.mainDetailsToggle.classList.remove('menu-opening');
    ...
    this.mainDetailsToggle.querySelectorAll('.submenu-
open').forEach((submenu) => {
      submenu.classList.remove('submenu-open');
    });
    document.body.classList.remove(`overflow-hidden-
${this.dataset.breakpoint}`);
    removeTrapFocus(elementToFocus);
    this.closeAnimation(this.mainDetailsToggle);
    if (event instanceof KeyboardEvent)
      elementToFocus?.setAttribute('aria-expanded', false);
  }
}

```

кінець лістингу 3.12

У наведеному фрагменті коду реалізовано механізм відкриття та закриття мобільного меню. Після відкриття drawer до контейнера додається CSS-клас menu-opening, який відповідає за відображення меню та запуск анімації появи.

Додатково до елемента body додається клас overflow-hidden, який блокує прокрутку основної сторінки під час роботи мобільного меню. Це дозволяє покращити взаємодію користувача із системою та зробити інтерфейс більш зручним на мобільних пристроях.

Для покращення користувацького досвіду у header також було реалізовано пошук товарів. Пошуковий механізм дозволяє швидко знаходити продукцію за

назвою або ключовими словами та забезпечує швидку навігацію по каталогу інтернет-магазину. Інтерфейс реалізованого пошуку наведено на рисунку 3.8.

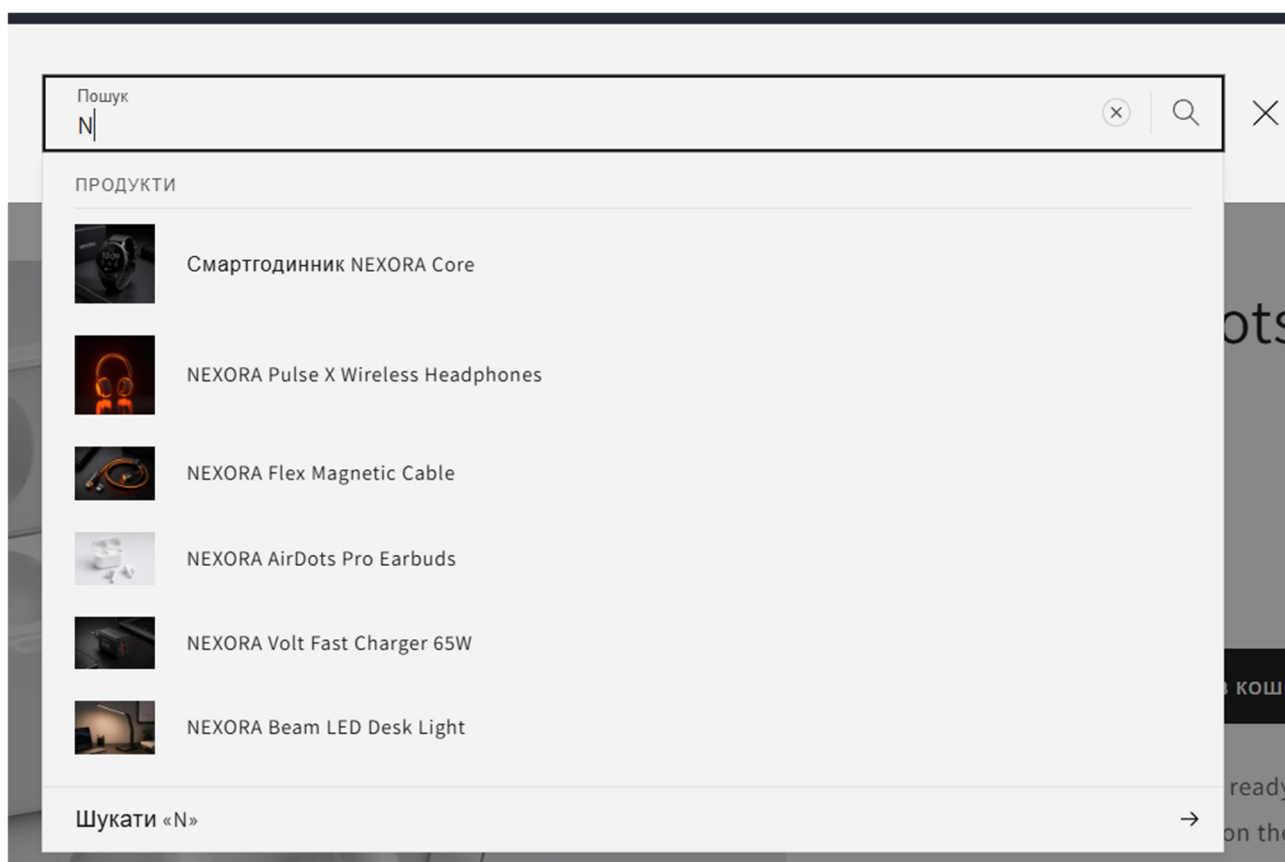


Рисунок 3.8 – Реалізація пошуку товарів в інтернет-магазині

Джерело: розроблено автором

Для реалізації пошукового поля використовувався окремий компонент `header-search`, який інтегрується у структуру `header` та взаємодіє із серверною частиною `Shopify`. Завдяки розміщенню пошуку у `header` користувач має швидкий доступ до цієї функції з будь-якої сторінки інтернет-магазину. Це особливо важливо для каталогів із великою кількістю товарів, де ручний перегляд категорій може займати більше часу. Пошуковий компонент містить поле введення тексту, кнопку активації пошуку та механізм обробки введених даних. Крім того, реалізований пошук дозволяє швидко знаходити необхідну продукцію за ключовими словами, що позитивно впливає на зручність навігації та загальний користувацький досвід. Використання вбудованих механізмів `Shopify` забезпечує коректну обробку пошукових запитів та автоматичне

формування сторінки результатів пошуку без необхідності реалізації додаткової серверної логіки (ліст. 3.13).

Лістинг 3.13 – Реалізація пошукового поля

```
<form action="{{ routes.search_url }}" method="get" role="search">
  <div class="field">
    <input
      class="search__input field__input" id="{{ input_id }}"
      type="search" name="q" value="{{ search.terms | escape }}"
      placeholder="{{ 'general.search.search' | t }}"
    >
    <label class="field__label" for="{{ input_id }}">
      {{ 'general.search.search' | t }}</label>
    <input type="hidden" name="options[prefix]" value="last">
    <button type="reset"
      class="reset__button field__button"
      aria-label="{{ 'general.search.reset' | t }}"
    >
      ...
    </button>
    <button class="search__button field__button">
      ...
    </button>
  </div>
</form>
```

кінець лістингу 3.13

У наведеному фрагменті коду реалізовано форму пошуку товарів. Атрибут «action="{{ routes.search_url }}"» визначає адресу, на яку надсилається пошуковий запит користувача. Введене значення автоматично передається через параметр «q», який використовується Shopify для формування результатів пошуку.

Для покращення інтерфейсу користувача у header було реалізовано SVG-іконки, які використовуються для кнопок навігації, пошуку та кошика. Використання SVG дозволяє забезпечити високу якість графіки на різних типах екранів та зменшити обсяг графічних ресурсів порівняно із растровими зображеннями.

Додатково під час розробки інтерфейсу значна увага приділялася адаптивності вебзастосунку. Інтернет-магазин коректно відображається на

персональних комп'ютерах, планшетах та смартфонах. Для цього використовувалися media queries, адаптивні CSS-сітки та динамічна зміна розташування елементів інтерфейсу залежно від ширини екрана користувача. Приклад адаптивного відображення інтерфейсу на мобільному пристрої наведено на рисунку 3.9.

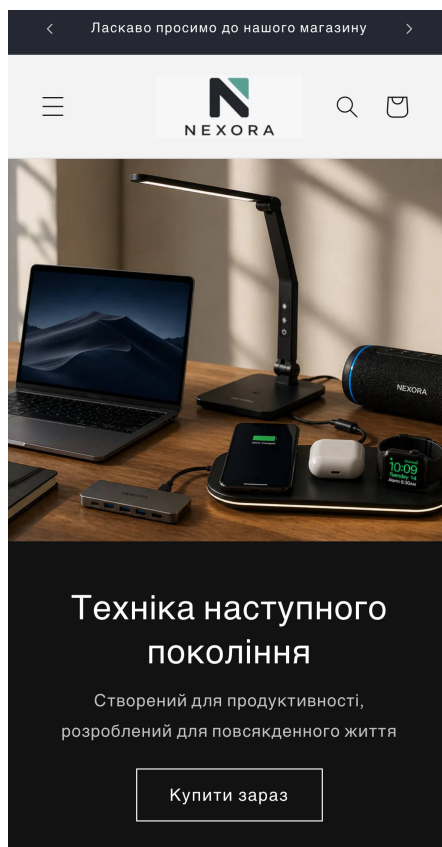


Рисунок 3.9 – Адаптивна версія інтернет-магазину на мобільному пристрої

Джерело: розроблено автором

Для адаптації інтерфейсу використовувалися CSS media queries, які дозволяють змінювати параметри стилізації залежно від розміру екрана пристрою. За допомогою media queries система автоматично перебудовує структуру сторінки для комфортного використання на смартфонах та планшетах. Це дозволяє приховувати окремі елементи, змінювати структуру блоків, розміри шрифтів та відступів, а також оптимізувати інтерфейс для мобільних пристроїв. Додатково адаптивний підхід забезпечує коректне розташування навігаційних елементів та контенту незалежно від орієнтації екрана користувача.

Використання `media queries` також дозволяє покращити читабельність інтерфейсу та забезпечити зручну взаємодію із системою на пристроях із невеликою шириною екрана (ліст. 3.14).

Лістинг 3.14 – Використання `media queries` для адаптивного дизайну

```
@media screen and (max-width: 768px) {
  .header__inline-menu {
    display: none;
  }
  .grid {
    grid-template-columns: 1fr;
  }
  .card__content {
    padding: 10px;
  }
}
```

кінець лістингу 3.14

У наведеному фрагменті коду реалізовано адаптацію інтерфейсу для мобільних пристроїв. При ширині екрана менше 768 пікселів приховується стандартне меню навігації, а структура каталогу автоматично перебудовується в одну колонку.

Такий підхід дозволяє покращити читабельність інтерфейсу та забезпечити зручну взаємодію користувача із системою навіть на невеликих екранах смартфонів. Крім того, зменшення кількості одночасно відображуваних елементів позитивно впливає на швидкість завантаження сторінки.

Під час реалізації інтернет-магазину також використовувалися JavaScript-компоненти для створення інтерактивних елементів інтерфейсу. Зокрема, було реалізовано слайдери зображень, анімації відкриття блоків та інтерактивні елементи навігації.

Для реалізації галереї товарів використовувався компонент `slider-component`, який дозволяє переглядати декілька зображень продукту без перезавантаження сторінки. Такий підхід покращує візуальне представлення товарів та забезпечує більш зручну взаємодію користувача із продукцією. Додатково реалізована галерея дозволяє швидко перемикатися між

зображеннями товару та переглядати різні варіанти продукції. Це забезпечує більш інформативне представлення товарів у вебінтерфейсі інтернет-магазину. Галерею зображень товару наведено на рисунку 3.10.

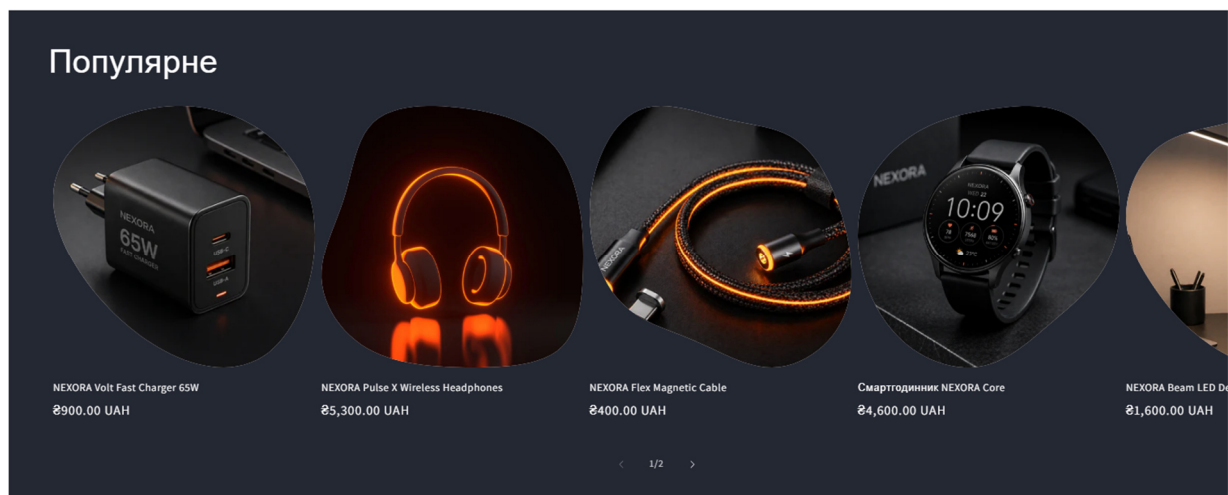


Рисунок 3.10 – Галерея зображень товару

Джерело: розроблено автором

Для керування слайдером використовувалися JavaScript-обробники подій, які змінюють активне зображення залежно від дій користувача. Перехід між елементами галереї відбувається без перезавантаження сторінки, що позитивно впливає на швидкодію вебзастосунку (ліст. 3.15).

Лістинг 3.15 – Реалізація JavaScript-слайдера

```
onButtonClick(event) {
    event.preventDefault();
    const nextElement = event.currentTarget.dataset.step === 'next'
        ? this.currentPage + 1
        : this.currentPage - 1;
    this.update();
    this.scrollToPage(nextElement);
}
```

кінець лістингу 3.15

У наведеному фрагменті коду реалізовано механізм перемикання елементів слайдера. Залежно від натиснутої кнопки система визначає наступний елемент галереї та виконує плавний перехід між зображеннями.

Для покращення продуктивності вебзастосунку використовувалися сучасні методи оптимізації ресурсів. Зокрема, реалізовано відкладене завантаження зображень, мінімізацію CSS та JavaScript-файлів, а також оптимізацію структури DOM-елементів.

Додатково використовувалися вбудовані механізми кешування Shopify CDN, які дозволяють швидко завантажувати статичні ресурси користувачам із різних регіонів. Це позитивно впливає на швидкість роботи інтернет-магазину та зменшує навантаження на серверну інфраструктуру (ліст.3.16).

Лістинг 3.16 – Відкладене завантаження зображень

```

```

кінець лістингу 3.16

Параметр «loading="lazy"» дозволяє завантажувати зображення лише в момент їх появи у видимій області екрана користувача. Це дозволяє зменшити початковий обсяг завантажуваних даних та прискорити відкриття сторінок інтернет-магазину.

Додатково для оптимізації адаптивних зображень використовувалися атрибути «srcset» та «sizes», які дозволяють браузеру автоматично вибирати найбільш оптимальний розмір зображення залежно від параметрів екрана користувача. Це дозволяє зменшити обсяг передаваних даних на мобільних пристроях та підвищити швидкість завантаження сторінок. Використання такого підходу позитивно впливає на продуктивність інтернет-магазину та забезпечує коректне відображення графічного контенту на різних типах пристроїв.

Завершальним етапом практичної реалізації стало налаштування параметрів кастомізації теми через Shopify Theme Editor. Адміністратор магазину отримує можливість змінювати структуру головної сторінки, налаштовувати кольорову схему, змінювати контент секцій та керувати окремими компонентами інтерфейсу без редагування програмного коду. Інтерфейс налаштування теми наведено на рисунку 3.11.

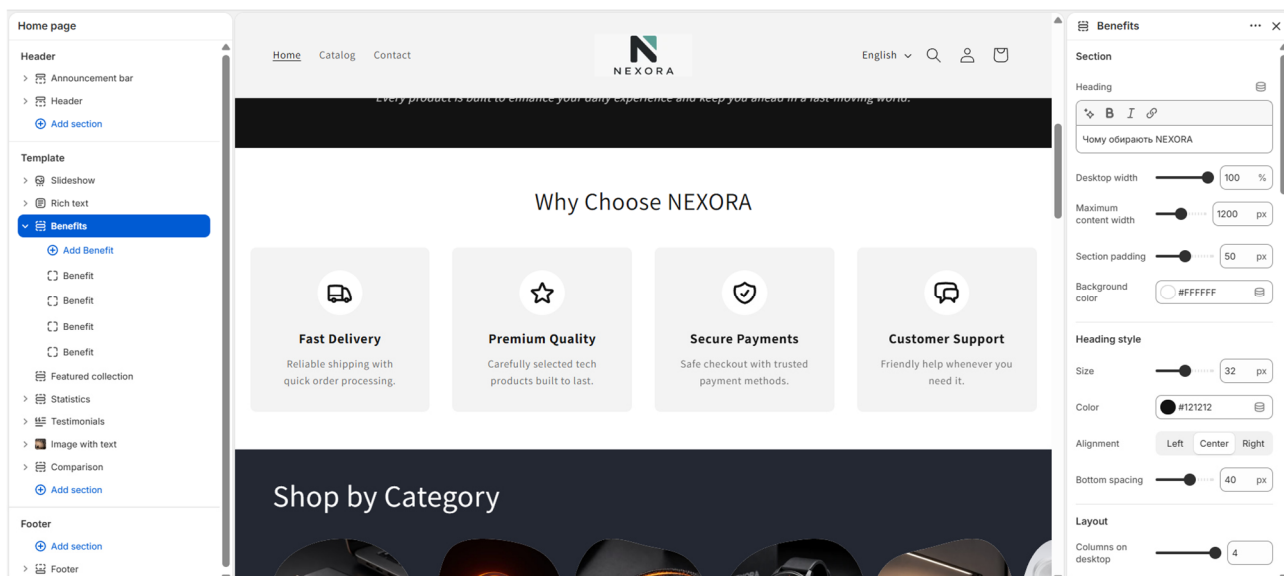


Рисунок 3.11 – Налаштування теми через Shopify Theme Editor [21]

Для забезпечення гнучкості кастомізації використовувалися параметри `settings_schema.json`, які визначають доступні налаштування теми. Це дозволяє створити зручний механізм керування інтерфейсом та адаптувати зовнішній вигляд інтернет-магазину під різні потреби адміністратора. Зокрема, через систему налаштувань адміністратор може змінювати кольорову схему, текстовий контент секцій, зображення та параметри відображення окремих компонентів інтерфейсу. Завдяки використанню `settings_schema.json` адміністратор може змінювати параметри теми через Shopify Theme Editor без необхідності редагування вихідного коду. Такий підхід спрощує подальшу підтримку інтернет-магазину та дозволяє швидко адаптувати інтерфейс відповідно до потреб користувачів або змін у структурі каталогу продукції. Додатково використання централізованої системи налаштувань забезпечує

зручне масштабування функціоналу теми та спрощує процес її подальшої модифікації. Фрагмент налаштування параметрів теми наведено у лістингу 3.17.

Лістинг 3.17 – Налаштування параметрів теми

```
[
  {
    "name": "Theme settings",
    "settings": [
      {
        "type": "color",
        "id": "background",
        "label": "Background color",
        "default": "#ffffff"
      }
    ]
  }
]
```

кінець лістингу 3.17

У наведеному фрагменті коду визначено параметри налаштування теми через Shopify Theme Editor. Адміністратор магазину може змінювати кольорову схему, параметри відображення окремих компонентів та інші характеристики інтерфейсу без внесення змін у вихідний код теми.

Таким чином, у результаті виконання практичної частини було реалізовано повноцінний інтернет-магазин на платформі Shopify із використанням Liquid, HTML, CSS та JavaScript. Розроблена система забезпечує адаптивний інтерфейс, динамічну взаємодію користувача із товарами, підтримку мобільних пристроїв та можливість подальшої кастомізації через адміністративну панель Shopify.

3.2 Тестування та налагодження інтернет-магазину

Після завершення основного етапу розробки інтернет-магазину було проведено тестування та налагодження системи. Основною метою тестування була перевірка коректності роботи функціональних модулів, адаптивності інтерфейсу, стабільності взаємодії користувача із системою та продуктивності вебзастосунку. Особлива увага приділялася перевірці компонентів, які

реалізують динамічну взаємодію із серверною частиною Shopify, оскільки саме вони найбільше впливають на загальний користувацький досвід.

У процесі тестування перевірялися всі основні модулі системи, зокрема каталог товарів, сторінка окремого продукту, система фільтрації, cart drawer, механізм додавання товарів до кошика, пошук товарів та адаптивне мобільне меню. Крім того, проводилася перевірка коректності відображення сторінок у різних браузерях та на різних типах пристроїв.

Тестування виконувалося поетапно. На початковому етапі проводилася перевірка окремих компонентів інтерфейсу, після чого здійснювалося комплексне тестування взаємодії всіх модулів системи між собою. Такий підхід дозволив своєчасно виявляти помилки та спростив процес налагодження вебзастосунку.

Одним із головних етапів тестування була перевірка роботи каталогу товарів та сторінки окремого продукту. Під час тестування перевірялося коректне відображення назв товарів, цін, зображень та варіантів продукції. Також контролювалася робота механізму переходу між сторінками товарів і правильність формування URL-адрес.

Окремо проводилася перевірка системи фільтрації товарів. Тестування включало перевірку вибору декількох параметрів одночасно, оновлення списку товарів після застосування фільтрів та правильність відображення активних параметрів фільтрації. Результати тестування показали, що система коректно обробляє запити користувача та динамічно оновлює каталог продукції без необхідності перезавантаження сторінки.

Важливу роль у процесі тестування відігравала перевірка функціоналу cart drawer. Для цього виконувалося додавання товарів до кошика, зміна кількості продукції та видалення окремих позицій із замовлення. Особлива увага приділялася перевірці асинхронних JavaScript-запитів та механізму оновлення HTML-вмісту окремих секцій інтерфейсу.

Під час тестування cart drawer перевірялося:

- коректне додавання товару до кошика;

- автоматичне оновлення загальної вартості замовлення;
- правильність зміни кількості продукції;
- стабільність роботи JavaScript-компонентів;
- коректність роботи анімацій відкриття та закриття drawer-компонента.

Результати перевірки показали, що після додавання товару система автоматично оновлює інформацію про кошик та відображає актуальні дані без повторного завантаження сторінки. Це дозволило забезпечити швидку взаємодію користувача із системою та покращити загальний користувацький досвід.

Додатково під час тестування перевірялася коректність синхронізації стану кошика між різними сторінками інтернет-магазину. Зокрема, після додавання товару у кошик виконувалася перевірка оновлення кількості товарів у header, cart drawer та сторінці кошика. Це дозволило переконатися у правильності роботи механізму динамічного оновлення секцій Shopify.

Окремо контролювалася стабільність роботи асинхронних JavaScript-запитів при швидкому повторному натисканні кнопки Add to Cart. Такий підхід дозволив перевірити поведінку системи при високій інтенсивності взаємодії користувача із вебзастосунком та уникнути дублювання товарів у кошику.

Для перевірки стабільності роботи JavaScript-компонентів використовувалися інструменти браузера Google Chrome DevTools. За допомогою вкладки Console перевірялася наявність JavaScript-помилки, а також виконувалося налагодження асинхронних запитів до Shopify Cart API. Це дозволило виявити окремі проблеми, пов'язані із повторним відкриттям cart drawer та оновленням DOM-елементів після динамічного рендерингу секцій.

Під час тестування також аналізувалася коректність оновлення HTML-секцій після виконання асинхронних запитів до Shopify Cart API. Особлива увага приділялася перевірці взаємодії JavaScript-компонентів із динамічно згенерованими DOM-елементами після повторного рендерингу сторінки. Це дозволило виявити випадки, коли оновлення інтерфейсу виконувалося раніше, ніж відповідний елемент з'являвся у структурі документа.

У процесі налагодження було реалізовано додаткові перевірки наявності DOM-елементів перед виконанням JavaScript-операцій. Це дозволило уникнути виникнення помилок під час оновлення окремих секцій інтерфейсу (ліст. 3.18).

Лістинг 3.18 – Перевірка наявності DOM-елемента

```
const sectionElement = section.selector
  ? document.querySelector(section.selector)
  : document.getElementById(section.id);

if (!sectionElement) return;

sectionElement.innerHTML = this.getSectionInnerHTML(
  parsedState.sections[section.id],
  section.selector
);
```

кінець лістингу 3.18

У наведеному фрагменті коду реалізовано перевірку існування HTML-елемента перед його оновленням. Такий підхід дозволяє уникнути помилок JavaScript у випадках, коли окремий елемент інтерфейсу тимчасово відсутній у DOM-структурі сторінки.

Під час налагодження також перевірялася правильність роботи JavaScript-подій після повторного рендерингу HTML-компонентів. Оскільки Shopify динамічно оновлює окремі секції сторінки, виникає необхідність повторного підключення обробників подій до новостворених DOM-елементів. Реалізація такого механізму дозволила забезпечити стабільну роботу cart drawer, mobile menu та інших інтерактивних компонентів інтерфейсу.

Крім того, виконувалася перевірка коректності роботи CSS-анімацій та transition-ефектів під час відкриття drawer-компонентів. Це дозволило уникнути візуальних артефактів та покращити плавність роботи інтерфейсу користувача.

Окремий етап тестування був присвячений перевірці адаптивності вебзастосунку. Для цього використовувалися інструменти емуляції мобільних пристроїв у браузері Google Chrome, а також реальні смартфони та планшети з різними розмірами екранів.

Під час тестування перевірялися:

- коректність роботи мобільного меню;
- адаптація каталогу товарів;
- зручність перегляду сторінки товару;
- коректність роботи cart drawer;
- правильність масштабування зображень.

Особлива увага приділялася перевірці responsive layout при зміні ширини екрана. Для цього використовувалися CSS media queries та адаптивні Grid/Flexbox-макети. Під час тестування було встановлено, що система коректно перебудовує структуру сторінки залежно від типу пристрою та забезпечує зручну взаємодію користувача із вебзастосунком (рис. 3.12).

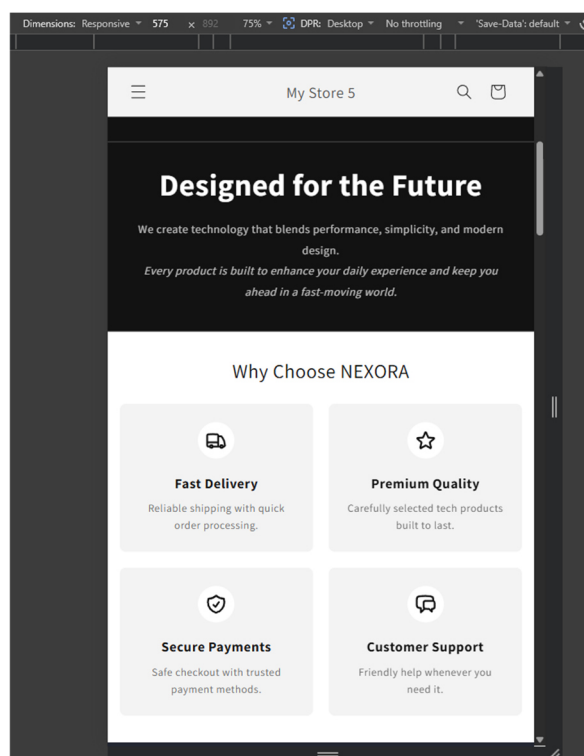


Рисунок 3.12 – Перевірка адаптивності інтерфейсу

Джерело: розроблено автором

Під час тестування також перевірялася коректність розташування навігаційних елементів та адаптація контенту при зміні орієнтації екрана пристрою. Особлива увага приділялася перевірці роботи touch-елементів

інтерфейсу, зокрема кнопок навігації, cart drawer та мобільного меню. Результати тестування підтвердили стабільну роботу адаптивного інтерфейсу на різних типах мобільних пристроїв.

Додатково проводилася перевірка роботи інтернет-магазину у різних веббраузерах. Тестування виконувалося у Google Chrome, Mozilla Firefox та Microsoft Edge. Основною метою такого тестування була перевірка сумісності CSS-стилів, JavaScript-функцій та коректності відображення HTML-компонентів.

Результати тестування показали, що система стабільно працює у всіх перевірених браузерах. Водночас у процесі налагодження було виконано оптимізацію окремих CSS-властивостей та JavaScript-анімацій для забезпечення однакової поведінки інтерфейсу в різних середовищах.

Під час міжбраузерного тестування окрема увага приділялася підтримці сучасних CSS-властивостей, зокрема Grid Layout, Flexbox та media queries. Також перевірялася коректність роботи SVG-іконок, адаптивних зображень та JavaScript-модулів у різних браузерних середовищах.

Додатково проводилася перевірка швидкодії вебзастосунку при низькій швидкості інтернет-з'єднання. Для цього використовувалися вбудовані інструменти Network Throttling у Chrome DevTools. Це дозволило оцінити поведінку системи в умовах обмеженої пропускної здатності мережі та перевірити ефективність оптимізації ресурсів.

Під час тестування також перевірялася продуктивність вебзастосунку. Для цього використовувалися вбудовані інструменти Lighthouse та PageSpeed Insights. Основна увага приділялася швидкості завантаження сторінок, оптимізації зображень та мінімізації обсягу ресурсів, що завантажуються браузером користувача.

Для підвищення продуктивності були використані:

- відкладене завантаження зображень;
- оптимізація CSS та JavaScript-файлів;
- використання Shopify CDN;

- адаптивні зображення;
- мінімізація кількості DOM-елементів.

Особливо ефективним виявилось використання атрибутів `loading="lazy"`, `srcset` та `sizes`, які дозволяють браузеру автоматично вибирати оптимальний розмір зображення залежно від параметрів пристрою користувача. Це дозволило зменшити обсяг передаваних даних та покращити швидкість відкриття сторінок інтернет-магазину.

Під час оптимізації також використовувалося кешування статичних ресурсів через Shopify CDN. Завдяки цьому стилі, JavaScript-файли та графічні ресурси завантажуються із серверів, розташованих найближче до користувача. Це дозволяє зменшити затримки при завантаженні сторінок та покращити загальну продуктивність інтернет-магазину.

Додатково виконувалася оптимізація структури DOM-дерева шляхом зменшення кількості вкладених HTML-елементів. Такий підхід позитивно впливає на швидкість рендерингу сторінок браузером та забезпечує більш стабільну роботу інтерфейсу на мобільних пристроях.

Крім тестування функціональних модулів, проводилася перевірка коректності роботи Theme Editor та механізму кастомізації Shopify-теми. Перевірялося редагування параметрів секцій, зміна текстових елементів, кольорової схеми та налаштувань окремих блоків інтерфейсу. Результати тестування показали, що адміністратор магазину може змінювати структуру сторінок без необхідності редагування вихідного коду теми.

Для коректної роботи інтернет-магазину достатньо сучасного веббраузера із підтримкою JavaScript, CSS3 та HTML5. Система коректно працює у браузерах Google Chrome, Mozilla Firefox та Microsoft Edge останніх версій. Оскільки платформа Shopify є хмарним рішенням, відсутня необхідність встановлення додаткового програмного забезпечення або розгортання серверної інфраструктури на стороні користувача.

Для комфортної роботи із системою рекомендовано використовувати пристрої з роздільною здатністю екрана не менше 1366×768 пікселів та

стабільним підключенням до мережі Інтернет. Інтернет-магазин також адаптований для роботи на мобільних пристроях під керуванням Android та iOS, що забезпечує зручну взаємодію користувача із системою незалежно від типу пристрою.

Таким чином, у результаті проведеного тестування та налагодження було підтверджено коректність роботи інтернет-магазину, стабільність взаємодії компонентів системи та відповідність сучасним вимогам до вебзастосунків електронної комерції.

3.3 Посібник з інсталяції та використання програмного продукту

Для успішної інсталяції та коректної роботи розробленого інтернет-магазину необхідно забезпечити відповідність мінімальним програмним вимогам. Оскільки програмний продукт реалізований на платформі Shopify та функціонує у хмарному середовищі, відсутня необхідність встановлення окремого серверного програмного забезпечення або розгортання локальної бази даних. Для роботи із системою необхідно забезпечити:

- наявність облікового запису Shopify;
- сучасний веббраузер із підтримкою HTML5, CSS3 та JavaScript;
- стабільне підключення до мережі Інтернет;
- доступ до адміністративної панелі Shopify.

Система коректно працює у браузерах Google Chrome, Mozilla Firefox та Microsoft Edge останніх версій. Для комфортної роботи рекомендується використовувати персональний комп'ютер або ноутбук із роздільною здатністю екрана не менше 1366×768 пікселів.

Розроблений програмний продукт реалізований у вигляді кастомної Shopify-теми, яка містить шаблони сторінок, секції, snippets-компоненти, файли стилів CSS, JavaScript-модулі та конфігураційні файли. Основними директоріями теми є layout, templates, sections, snippets, assets та config.

Для початку роботи необхідно увійти до адміністративної панелі Shopify та перейти до розділу Online Store далі до Themes. Саме через цей інтерфейс здійснюється встановлення та налаштування теми інтернет-магазину (рис. 3.13).

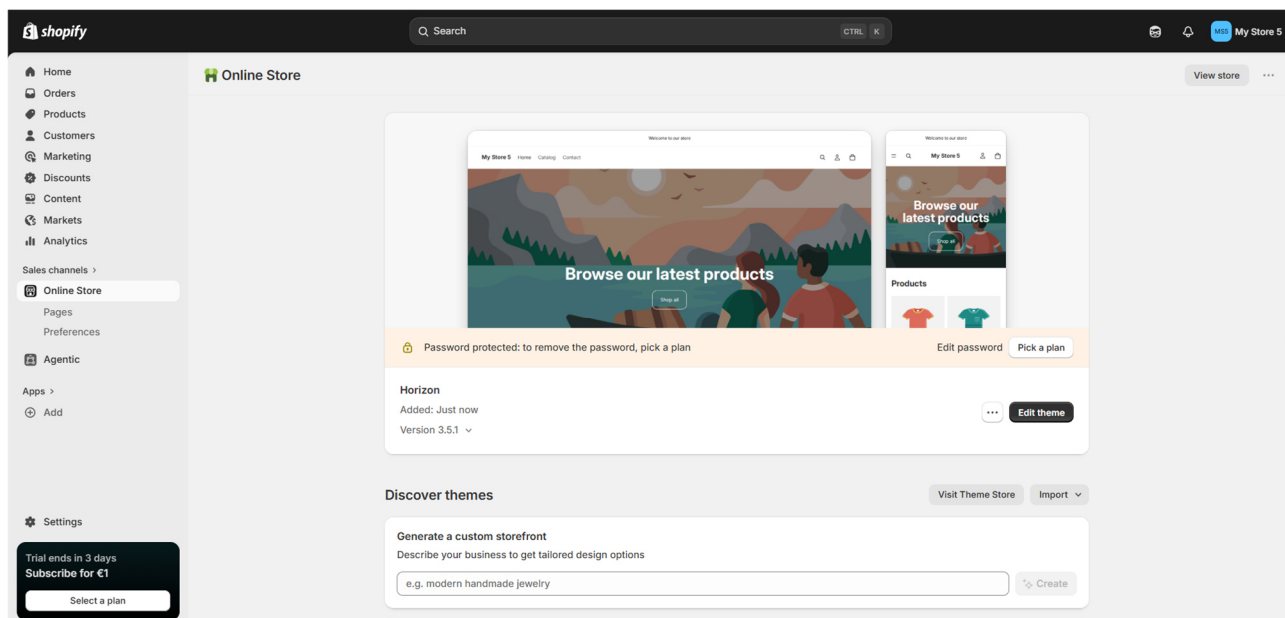


Рисунок 3.13 – Розділ керування темами у Shopify Admin [22]

Інсталяція теми виконується шляхом завантаження архіву з файлами теми у форматі .zip. Для цього необхідно натиснути кнопку Add theme далі Upload zip file та обрати архів із файлами розробленої Shopify-теми. Після підтвердження завантаження Shopify автоматично виконує обробку файлів та додає тему до списку доступних шаблонів магазину (рис. 3.14).

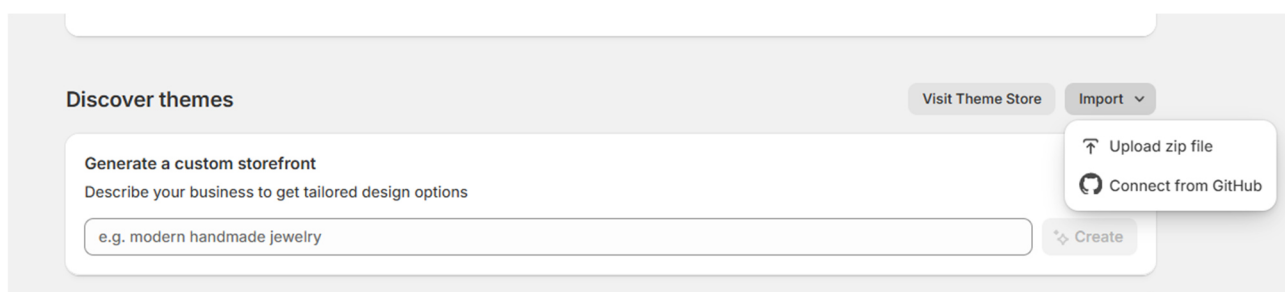


Рисунок 3.14 – Завантаження Shopify-теми через Upload zip file [23]

Після завершення встановлення тема може бути опублікована як основна тема інтернет-магазину. Для цього у списку встановлених тем необхідно обрати

пункт Publish. Після публікації тема автоматично застосовується до всіх сторінок магазину.

Коли встановлення теми завершилось адміністратор отримує можливість налаштовувати зовнішній вигляд інтернет-магазину через Shopify Theme Editor. Даний інструмент дозволяє змінювати структуру головної сторінки, редагувати секції, налаштовувати кольорову схему та керувати окремими компонентами інтерфейсу без необхідності редагування програмного коду.

Theme Editor підтримує роботу із Shopify Sections, які реалізовані у вигляді окремих модульних компонентів. Адміністратор може змінювати порядок секцій, редагувати текстові блоки, змінювати зображення, налаштовувати кнопки та керувати структурою сторінки через графічний інтерфейс.

Для переходу до режиму редагування необхідно натиснути кнопку Customize у розділі Themes. Після цього відкривається візуальний редактор теми, який дозволяє в режимі реального часу переглядати внесені зміни (рис. 3.15).

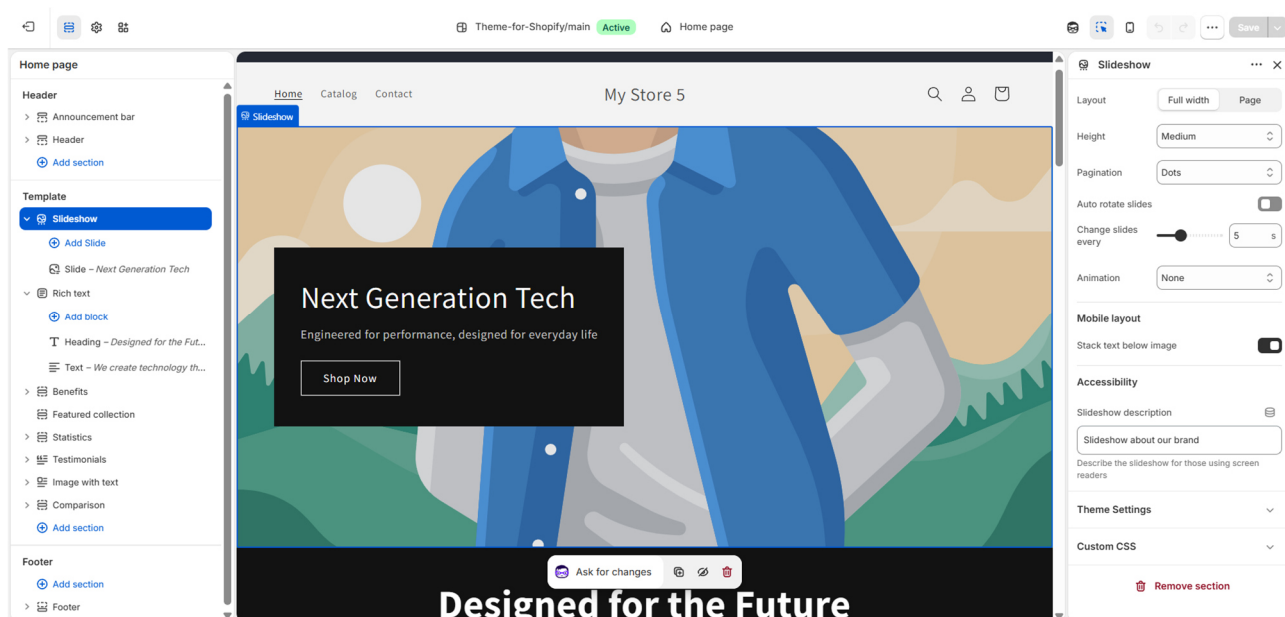


Рисунок 3.15 – Налаштування теми через Shopify Theme Editor [24]

Окрім налаштування структури сторінок, Theme Editor дозволяє змінювати глобальні параметри теми через розділ Theme Settings. У даному розділі

адміністратор може налаштовувати кольорову схему, типографіку, favicon, логотип та інші параметри оформлення інтернет-магазину.

Для коректної роботи інтернет-магазину необхідно додати товари до каталогу продукції. Керування товарами здійснюється через розділ Products адміністративної панелі Shopify. Під час створення товару адміністратор заповнює назву продукції, опис, вартість, характеристики та додає зображення товару.

Також система підтримує використання варіантів товару (variants), які дозволяють створювати декілька конфігурацій одного продукту, наприклад різні кольори, розміри або обсяг пам'яті. Shopify автоматично формує доступні комбінації варіантів та забезпечує їх відображення на сторінці продукту. Процес додавання нового товару через Shopify Admin наведено на рисунку 3.16.

The screenshot shows the 'Add Product' form in the Shopify Admin. The form is divided into several sections:

- Title:** A text input field containing 'NEXORA Hub Pro USB-C Adapter'.
- Description:** A rich text editor with a toolbar (bold, italic, underline, link, etc.) and a text area containing 'Expand your device connectivity with Hub Pro. Designed for laptops and modern devices.'
- Media:** A section for adding images, showing a preview of a USB hub connected to a laptop and a '+ ' button to add more media.
- Category:** A dropdown menu showing 'USB & FireWire Hubs in Computer Components' with a note '10 metafields'.
- Status:** A dropdown menu set to 'Active'.
- Publishing:** A section with a toggle for 'All channels'.
- Product organization:** A section with dropdowns for 'Type' (set to 'None') and 'Vendor' (set to 'None'), and buttons for 'Add collections' and 'Add tags'.
- Theme template:** A dropdown menu set to 'Default product'.

Рисунок 3.16 – Додавання нового товару через Shopify Admin [25]

Для організації структури каталогу використовуються колекції (Collections), які дозволяють групувати товари за категоріями. Це забезпечує

зручну навігацію по інтернет-магазину та спрощує пошук необхідної продукції користувачами.

Налаштування навігаційного меню здійснюється через розділ Navigation адміністративної панелі Shopify. У даному розділі адміністратор може створювати категорії, додавати посилання на сторінки та змінювати структуру головного меню інтернет-магазину.

Після завершення налаштування теми та додавання товарів інтернет-магазин готовий до використання. Користувач отримує можливість переглядати каталог продукції, відкривати сторінки окремих товарів, використовувати систему пошуку та оформлювати замовлення через checkout-систему Shopify.

Для забезпечення високої продуктивності система використовує механізми кешування Shopify CDN, відкладене завантаження зображень та адаптивні графічні ресурси. Це дозволяє забезпечити швидке завантаження сторінок та стабільну роботу вебзастосунку на різних типах пристроїв.

Оскільки платформа Shopify є хмарним рішенням, адміністратор магазину не потребує налаштування серверного середовища або встановлення додаткового програмного забезпечення. Усі оновлення системи, збереження даних та робота серверної частини забезпечуються засобами платформи Shopify.

Таким чином, розроблений програмний продукт характеризується простотою інсталяції, зручністю адміністрування та можливістю гнучкого налаштування інтерфейсу без внесення змін у програмний код. Використання платформи Shopify дозволяє значно спростити процес розгортання інтернет-магазину та забезпечує ефективне керування системою через адміністративну панель.

РОЗДІЛ 4

СПЕЦІАЛЬНІ РОЗРАХУНКИ

У сучасних умовах розвитку електронної комерції важливим чинником успішного функціонування інтернет-магазину є його видимість у пошукових системах. Більшість користувачів здійснюють пошук товарів саме через Google та інші пошукові системи, тому ефективна SEO-оптимізація (Search Engine Optimization) безпосередньо впливає на кількість відвідувачів вебресурсу, рівень конверсії та загальну конкурентоспроможність інтернет-магазину.

SEO-оптимізація охоплює комплекс заходів, спрямованих на покращення структури вебзастосунку, швидкості його роботи, адаптивності інтерфейсу та коректності індексації сторінок пошуковими системами. Для інтернет-магазинів SEO має особливе значення, оскільки дозволяє забезпечити стабільний потік потенційних покупців без потреби постійного використання платної реклами.

У процесі розробки інтернет-магазину на платформі Shopify особлива увага приділялася технічній SEO-оптимізації, адаптивності інтерфейсу та швидкодії вебзастосунку. Платформа Shopify надає вбудовані механізми SEO, які дозволяють автоматично формувати sitemap.xml, robots.txt, canonical URL та SEO-friendly адреси сторінок. Це значно спрощує процес оптимізації вебресурсу для пошукових систем.

Одним із важливих елементів SEO є правильна структура HTML-документа. Під час розробки інтернет-магазину використовувалася семантична HTML-розмітка, що дозволяє пошуковим системам коректно аналізувати структуру сторінки та визначати її основний контент. Для цього використовувалися заголовки h1-h3, semantic tags та логічна структура контенту.

На сторінках товарів основна інформація про продукцію відображається за допомогою структурованих HTML-елементів, що позитивно впливає на індексацію сторінок пошуковими системами. Важливу роль також відіграє використання атрибутів alt для зображень, які дозволяють пошуковим системам

отримувати текстовий опис графічного контенту. Це покращує індексацію зображень та підвищує доступність вебзастосунку для користувачів.

Важливим аспектом SEO-оптимізації є швидкість завантаження сторінок. Пошукові системи враховують продуктивність вебресурсу під час ранжування результатів пошуку. Для забезпечення високої швидкодії у розробленому інтернет-магазині використовувалися сучасні методи оптимізації ресурсів.

Одним із таких методів є відкладене завантаження зображень (lazy loading). Даний механізм дозволяє завантажувати графічний контент лише у момент його появи у видимій області екрана користувача. Це дозволяє суттєво зменшити початковий обсяг даних, що завантажуються браузером, та прискорити відкриття сторінок.

Для реалізації адаптивного завантаження зображень використовувалися атрибути `srcset` та `sizes`. Їх використання дозволяє браузеру автоматично обирати найбільш оптимальний розмір зображення залежно від параметрів пристрою користувача. Такий підхід дозволяє зменшити обсяг передаваних даних та покращити швидкість завантаження сторінок, особливо на мобільних пристроях.

Для покращення швидкодії також використовувалися механізми кешування Shopify CDN. Усі статичні ресурси, зокрема CSS-файли, JavaScript-модулі та графічні елементи, завантажуються через глобальну мережу доставки контенту. Це дозволяє зменшити затримки при завантаженні сторінок та забезпечити стабільну роботу інтернет-магазину для користувачів із різних регіонів.

Також додатково оптимізовувалися JavaScript-скрипти та CSS-файли шляхом мінімізації та об'єднання, що дозволяло зменшити кількість запитів до сервера та підвищити швидкість завантаження сторінки.

Під час SEO-оптимізації значна увага приділялася адаптивності вебзастосунку. Сучасні пошукові системи використовують mobile-first indexing, тобто першочергово аналізують мобільну версію сайту. З огляду на це інтерфейс інтернет-магазину був реалізований із використанням responsive design та CSS media queries.

Адаптивний інтерфейс дозволяє забезпечити коректне відображення сторінок на персональних комп'ютерах, планшетах та смартфонах. Для цього використовувалися CSS Grid, Flexbox та media queries, які автоматично змінюють структуру сторінки залежно від ширини екрана користувача. Такий підхід дозволяє забезпечити комфортну взаємодію користувача із системою на мобільних пристроях та позитивно впливає на SEO-показники. Це також дозволяє забезпечити більш стабільну роботу вебзастосунку на різних пристроях і підвищує загальний рівень користувацького досвіду.

Важливим компонентом SEO є коректна структура URL-адрес сторінок. Shopify автоматично формує SEO-friendly URL для товарів, колекцій та сторінок інтернет-магазину. Такі адреси є зрозумілими для користувачів та пошукових систем, що позитивно впливає на індексацію контенту.

Додатково платформа Shopify автоматично генерує sitemap.xml та robots.txt, які використовуються пошуковими системами для індексації сторінок вебресурсу.

Для уникнення дублювання контенту Shopify використовує canonical URL. Canonical-посилання дозволяють пошуковим системам визначати основну версію сторінки та запобігають виникненню проблем із дубльованим контентом. Це є особливо важливим для сторінок товарів, які можуть відкриватися через різні URL-адреси.

У процесі оптимізації також використовувалися метадані сторінок, зокрема title та meta description. Дані елементи формують опис сторінки у результатах пошуку та впливають на CTR (Click Through Rate) користувачів. Shopify дозволяє налаштовувати SEO-параметри окремо для кожної сторінки товару, колекції або статті блогу.

Для оцінки ефективності SEO-оптимізації використовувалися інструменти Google Lighthouse та PageSpeed Insights. Під час аналізу перевірялися швидкість завантаження сторінок, оптимізація ресурсів, адаптивність інтерфейсу та доступність контенту.

Для оцінки ефективності оптимізації швидкодії інтернет-магазину використовувався показник зменшення часу завантаження сторінки, який обчислюється за формулою 4.1.

Відомо, що

$$E = \frac{T_1 - T_2}{T_1} * 100\%, \quad (4.1)$$

де E – відсоткове зменшення часу завантаження сторінки;

T_1 – середній час завантаження сторінки до впровадження оптимізаційних заходів (у секундах);

T_2 – середній час завантаження сторінки після впровадження оптимізаційних заходів (у секундах);

До впровадження оптимізаційних заходів середній час завантаження сторінки становив 3,8 с, а після використання lazy loading, адаптивних зображень та Shopify CDN – 2,4 с.

Отже, використання сучасних методів оптимізації дозволило зменшити час завантаження сторінки приблизно на 36,8 %, що позитивно впливає на SEO-показники та загальну продуктивність вебзастосунку.

Результати тестування показали, що використання lazy loading, адаптивних зображень, Shopify CDN та responsive design дозволило суттєво покращити продуктивність вебзастосунку та забезпечити відповідність сучасним вимогам SEO-оптимізації.

Крім технічної оптимізації, важливу роль відіграє внутрішня структура навігації інтернет-магазину. Правильна організація меню, колекцій та сторінок товарів спрощує навігацію для користувачів і покращує індексацію сайту пошуковими системами. Завдяки логічній структурі каталогу користувачі можуть швидше знаходити необхідну продукцію, що позитивно впливає на поведінкові фактори та тривалість перебування на сайті.

Під час розробки також враховувалися сучасні вимоги до доступності вебконтенту. Зокрема, використовувалися текстові описи зображень, коректна структура заголовків та достатній контраст елементів інтерфейсу. Це дозволяє покращити взаємодію користувачів із системою та позитивно впливає на оцінку вебресурсу пошуковими системами.

Таким чином, у процесі розробки інтернет-магазину було реалізовано комплекс заходів SEO-оптимізації, спрямованих на покращення індексації вебресурсу, підвищення швидкодії та забезпечення адаптивності інтерфейсу.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено інтернет-магазин електроніки на платформі Shopify із використанням сучасних вебтехнологій та кастомної теми. Під час виконання роботи проведено аналіз сучасного стану електронної комерції, досліджено основні проблеми створення інтернет-магазинів та розглянуто актуальні платформи для реалізації систем електронної торгівлі.

У процесі аналізу існуючих рішень було досліджено платформи Shopify, WooCommerce, Magento та Wix eCommerce. За результатами порівняльного аналізу визначено, що платформа Shopify є найбільш доцільною для реалізації поставленого завдання завдяки наявності готової серверної інфраструктури, високому рівню безпеки, можливостям масштабування та підтримці кастомізації інтерфейсу.

У роботі було обґрунтовано вибір технологій розробки, зокрема мови шаблонізації Liquid, а також HTML, CSS та JavaScript. Використання даного технологічного стеку дозволило реалізувати адаптивний та функціональний інтерфейс користувача, який відповідає сучасним вимогам електронної комерції.

Під час виконання практичної частини роботи було спроектовано архітектуру інтернет-магазину, визначено його основні функціональні модулі та реалізовано кастомну Shopify-тему. У процесі розробки створено каталог товарів, сторінку окремого продукту, систему фільтрації, cart drawer, адаптивне мобільне меню та механізм пошуку товарів.

Особлива увага приділялася реалізації динамічної взаємодії користувача із системою. Для цього використовувалися асинхронні JavaScript-запити та механізми динамічного оновлення HTML-вмісту окремих секцій інтерфейсу. Це дозволило забезпечити швидку взаємодію користувача із системою без необхідності повторного завантаження сторінок.

У процесі розробки також реалізовано адаптивний дизайн із використанням CSS Grid, Flexbox та media queries. Це дозволило забезпечити

коректне відображення інтернет-магазину на персональних комп'ютерах, планшетах та мобільних пристроях.

Важливим етапом роботи стало тестування та налагодження вебзастосунку. Під час тестування перевірялася коректність роботи функціональних модулів, стабільність JavaScript-компонентів, робота cart drawer та механізмів фільтрації товарів. Для налагодження використовувалися інструменти Google Chrome DevTools та Thunder Client, що дозволило виявити та усунути помилки взаємодії між компонентами системи.

У межах спеціальних розрахунків було проведено аналіз SEO-оптимізації інтернет-магазину. Реалізовано оптимізацію структури сторінок, адаптивне завантаження зображень, використання lazy loading, srcset та sizes, а також механізми кешування Shopify CDN. Це дозволило покращити швидкодію вебзастосунку та забезпечити відповідність сучасним вимогам пошукової оптимізації.

У результаті виконання роботи досягнуто поставленої мети – розроблено сучасний адаптивний інтернет-магазин електроніки на платформі Shopify із використанням кастомної теми та унікальних функціональних елементів. Створений вебресурс забезпечує зручну взаємодію користувача із системою, підтримує можливість подальшої кастомізації та може бути використаний у реальних умовах електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Ecommerce? Shopify. URL: <https://www.shopify.com/blog/what-is-ecommerce> (дата звернення: 26.02.2026)
2. Retail e-commerce sales worldwide. Statista. URL: <https://www.statista.com/topics/871/online-shopping/> (дата звернення: 27.02.2026)
3. Mobile commerce statistics. Statista. URL: <https://www.statista.com/statistics/806336/mobile-retail-commerce-share-worldwide/> (дата звернення: 28.02.2026)
4. User Experience Basics. Nielsen Norman Group. URL: <https://www.nngroup.com/articles/definition-user-experience/> (дата звернення: 01.03.2026)
5. Web application architecture. Microsoft Docs. URL: <https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/web-application> (дата звернення: 02.03.2026)
6. Mobile page speed and conversion. Think with Google. URL: <https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/> (дата звернення: 03.03.2026)
7. Ecommerce security: Best practices. Shopify. URL: <https://www.shopify.com/blog/ecommerce-security> (дата звернення: 04.03.2026)
8. What is SaaS? AWS. URL: <https://aws.amazon.com/what-is/saas/> (дата звернення: 05.03.2026)
9. Benefits of SaaS. IBM. URL: <https://www.ibm.com/topics/saas> (дата звернення: 06.03.2026)
10. Ecommerce platforms comparison. Shopify. URL: <https://www.shopify.com/blog/ecommerce-platforms> (дата звернення: 08.03.2026)
11. SaaS ecommerce explained. BigCommerce. URL: <https://www.bigcommerce.com/articles/ecommerce/saas-ecommerce/> (дата звернення: 10.03.2026)

12. WooCommerce official site. URL: <https://woocommerce.com/> (дата звернення: 12.03.2026)
13. Adobe Commerce (Magento). URL: <https://business.adobe.com/products/magento/magento-commerce.html> (дата звернення: 14.03.2026)
14. Wix eCommerce. URL: <https://www.wix.com/ecommerce> (дата звернення: 16.03.2026)
15. Shopify platform. URL: <https://www.shopify.com/> (дата звернення: 18.03.2026)
16. Liquid template language. Shopify. URL: <https://shopify.github.io/liquid/> (дата звернення: 20.03.2026)
17. HTML: HyperText Markup Language. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTML> (дата звернення: 22.03.2026)
18. JavaScript. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (дата звернення: 24.03.2026)
19. What is Shopify? Shopify. URL: <https://www.shopify.com/encyclopedia/shopify> (дата звернення: 28.03.2026)
20. Client-server architecture. IBM. URL: <https://www.ibm.com/topics/client-server-model> (дата звернення: 02.04.2026)
21. Shopify Theme Editor. Shopify Docs. URL: <https://help.shopify.com/en/manual/online-store/themes/customizing-themes> (дата звернення: 03.04.2026)
22. Manage Shopify themes. Shopify Docs. URL: <https://help.shopify.com/en/manual/online-store/themes/managing-themes> (дата звернення: 03.04.2026)
23. Upload Shopify theme files. Shopify Docs. URL: <https://help.shopify.com/en/manual/online-store/themes/add-theme> (дата звернення: 04.04.2026)

24. Customize your Shopify theme. Shopify Docs. URL: <https://help.shopify.com/en/manual/online-store/themes/theme-structure/theme-editor> (дата звернення: 05.04.2026)

25. Add and update products. Shopify Docs. URL: <https://help.shopify.com/en/manual/products/add-update-products> (дата звернення: 05.04.2026)

ДОДАТКИ

Додаток А

Сертифікат про участь у міжнародній науково-практичній конференції



CIHTEX 2026

СЕРТИФІКАТ

засвідчує, що

Літвінчук Андрій Андрійович

взяв участь у

I Міжнародній науково-практичній конференції:
Сучасні інформаційні технології: від теорії до практики

загальним обсягом 15 годин (0,5 кредитів ECTS),
 яка проводилася **22-23 травня 2026 року**
 в Луцькому національному технічному університеті



[mintech-conf.lntu.edu.ua](http://mintech-conf.lntu.edu.ua/cert/2026-180)
 /cert/2026-180

Ректор ЛНТУ

*Ірина ВАХОВИЧ*

Луцький національний технічний університет
 вул. Львівська, 75 м. Луцьк
 Волинська обл. 43018



Додаток Б

Schema-конфігурація секцій Shopify

```
{% schema %}
{
  "name": "Benefits",
  "settings": [
    {
      "type": "header",
      "content": "Section"
    },
    {
      "type": "inline_richtext",
      "id": "section_heading",
      "label": "Heading",
      "default": "Why shop with us"
    },
    {
      "type": "range",
      "id": "desktop_width_percent",
      "label": "Desktop width",
      "min": 50,
      "max": 100,
      "step": 5,
      "unit": "%",
      "default": 100
    },
    {
      "type": "range",
      "id": "max_width",
      "label": "Maximum content width",
      "min": 800,
      "max": 1600,
      "step": 100,
      "unit": "px",
      "default": 1200
    },
    {
      "type": "range",
      "id": "section_padding",
      "label": "Section padding",
      "min": 0,
      "max": 100,
      "step": 5,
      "unit": "px",
      "default": 50
    },
    {
      "type": "header",
```

```

    "content": "Layout"
  },
  {
    "type": "range",
    "id": "columns_desktop",
    "label": "Columns on desktop",
    "min": 2,
    "max": 4,
    "step": 1,
    "default": 4
  },
  {
    "type": "select",
    "id": "columns_mobile",
    "label": "Columns on mobile",
    "options": [
      {
        "value": "1",
        "label": "1"
      },
      {
        "value": "2",
        "label": "2"
      }
    ],
    "default": "1"
  },
  {
    "type": "header",
    "content": "Hover effects"
  },
  {
    "type": "range",
    "id": "hover_lift",
    "label": "Lift distance",
    "min": -10,
    "max": 0,
    "step": 1,
    "unit": "px",
    "default": -5
  },
  {
    "type": "range",
    "id": "hover_shadow_offset",
    "label": "Shadow offset",
    "min": 0,
    "max": 20,
    "step": 2,
    "unit": "px",
    "default": 10
  },

```

```

{
  "type": "range",
  "id": "hover_shadow_blur",
  "label": "Shadow blur",
  "min": 0,
  "max": 40,
  "step": 5,
  "unit": "px",
  "default": 20
},
{
  "type": "range",
  "id": "hover_shadow_opacity",
  "label": "Shadow opacity",
  "min": 0,
  "max": 0.5,
  "step": 0.1,
  "default": 0.2
},
],
"blocks": [
  {
    "name": "Benefit",
    "type": "benefit",
    "settings": [
      {
        "type": "checkbox",
        "id": "benefit_enabled",
        "label": "Enable",
        "default": true
      },
      {
        "type": "select",
        "id": "benefit_icon",
        "label": "Icon",
        "options": [
          {
            "value": "shipping",
            "label": "Free shipping"
          },
          {
            "value": "returns",
            "label": "Easy returns"
          },
          {
            "value": "support",
            "label": "24/7 support"
          },
          {
            "value": "secure",
            "label": "Secure checkout"
          }
        ]
      }
    ]
  }
]

```

```

    },
    {
      "value": "quality",
      "label": "Premium quality"
    },
    {
      "value": "gift",
      "label": "Gift wrapping"
    },
    {
      "value": "warranty",
      "label": "Warranty"
    },
    {
      "value": "eco",
      "label": "Eco-friendly"
    }
  ],
  "default": "shipping"
},
{
  "type": "inline_richtext",
  "id": "benefit_title",
  "label": "Title",
  "default": "Free shipping"
},
{
  "type": "text",
  "id": "benefit_description",
  "label": "Description",
  "default": "On all orders over $50"
},
{
  "type": "text",
  "id": "benefit_details",
  "label": "Additional details"
}
]
}
],
"presets": [
  {
    "name": "Benefits"
  }
]
}
{% endschema %}

```