Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту)
Кафедра обчислювальної техніки
(повна назва кафедри)

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ОНЛАЙН-МАГАЗИНУ КОПЮТЕРНОЇ ТЕХНІКИ»

08-54.БКР.004.00.000 ПЗ

Виконав: студент 4 курсу, групи 1КІ-21б
спеціальності
123 Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
освітня програма «Комп'ютерна інженерія»
Годлевський Д.В.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ОТ
Тарновський М.Г.
(прізвище та ініціали)

Рецензент: к.т.н., доцент каф.МБІС
Грицак А. В.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«13» червня 2025 р.

Вінниця 2025

Вінницький національний технічний університет

Факультет Інформаційних технологій та комп'ютерної інженерії

Кафедра Обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 - Інформаційні технології

Спеціальність 123 - «Комп'ютерна інженерія»

Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н.проф. _____Азаров О. Д.
«21» березня 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Годлевському Дмитру Вадимовичу

1 Тема роботи «Інформаційна система для онлайн-магазину комп'ютерної техніки», керівник роботи к.т.н., доц. Тарновський М. Г, затверджені наказом ВНТУ від «20» березня 2025 року №97.

2 Термін подання студентом роботи 13.05.2025 року

3 Вихідні дані до роботи: підтримка типових функцій систем електронної комерції; підтримка інтеграції із зовнішніми сервісами (оплата, доставка); забезпечення адаптивного дизайну та кросбраузерної сумісності; інтеграція з інструментами аналітики, такими як Google Analytics, Hotjar; підтримувати механізми захисту персональних даних користувачів.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз предметної області; вибір та обгрунтування аналогів; вибір та обгрунтування інструментів для розробки інформаційної системи; розробка функціональних модулів інформаційної системи; розробка адміністративної панелі; реалізація безпеки, персоналізації, масштабування.

5 Перелік графічного матеріалу: загальна структура інформаційної системи онлайн-магазину; схема технічного середовища інформаційної системи; загальна схема взаємодії користувачів з інформаційною системою онлайн-магазину.

6 Консультанти розділів роботи приведений в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Спеціальна частина	Тарновський М. Г., доцент кафедри ОТ	21.03.25	13.05.25
Нормоконтроль	ас. каф. ОТ Швець С. Ш.	14.05.25	30.05.25

7 Дата видачі завдання «21» березня 2025 р.

8 Календарний план виконання БКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів виконання бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	21.03.25 – 28.03.25	виконано
2	Обгрунтування актуальності теми. Аналіз предметної області	29.03.25 – 04.04.25	виконано
3	Вибір та обгрунтування інструментів для розробки інформаційної системи	05.04.25 – 09.04.25	виконано
4	Розробка інформаційної системи для онлайн-магазину комп'ютерної техніки	11.04.25 – 13.05.25	виконано
5	Оформлення пояснювальної записки та ілюстративного матеріалу	14.05.25 – 21.05.25	виконано
6	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	22.05.25 – 02.06.25	виконано

Студент _____ (підпис) Годлевський Д.В.

Керівник роботи _____ (підпис) Тарновський М. Г.

АНОТАЦІЯ

УДК 004.382.4:621.382

Годлевський Д.В. Інформаційна система для онлайн-магазину комп’ютерної техніки. Бакалаврська кваліфікаційна робота зі спеціальності 123 – Комп’ютерна інженерія. Вінниця: ВНТУ, 2025. 98 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 27; таб.: 5.

У бакалаврській роботі розроблено інформаційну систему для онлайн-магазину комп’ютерної техніки. У межах дослідження проаналізовано сучасні вимоги до подібних систем, обґрунтовано вибір архітектури та інструментів розробки, розглянуто актуальні технології, що забезпечують адаптивність, безпеку та інтерактивність вебзастосунку.

Реалізовано функціальні можливості управління каталогом товарів, кошиком, замовленнями, вподобаними товарами, системою фільтрації, рекомендаціями, а також засоби збереження стану користувача. Особливу увагу приділено адаптивності інтерфейсу, валідації даних, інтеграції із зовнішніми сервісами та можливості масштабування системи. Робота містить приклади коду, структури даних, а також скріншоти реалізованого інтерфейсу.

Ключові слова: інформаційна система, онлайн-магазин, веброзробка, адаптивний інтерфейс, JavaScript, каталог, кошик, замовлення.

ABSTRACT

Hodlevskyi D.V. Information System for an Online Computer Hardware Store. Bachelor's Thesis in Specialty 123 – Computer Engineering. Vinnytsia: VNTU, 2025. 98 pages.

In Ukrainian. Bibliography: 21 sources; illustrations:27; tables 5.

This bachelor's thesis presents the development of an information system for an online computer hardware store. The research analyzes current requirements for such systems, justifies the choice of architecture and development tools, and reviews relevant technologies that ensure the adaptability, security, and interactivity of the web application.

Implemented features include product catalog management, shopping cart, order processing, favorites, filtering, recommendation system, and mechanisms for preserving user state. Special attention is given to interface adaptability, data validation, integration with external services, and system scalability. The work includes code examples, data structures, and interface screenshots.

Keywords: information system, online store, web development, responsive interface, JavaScript, catalog, cart, orders.

ЗМІСТ

ВСТУП

У XXI столітті цифрові технології виступають ключовим чинником трансформації глобальної економіки. Зокрема, електронна комерція є однією з найдинамічніших галузей, яка істотно змінює механізми взаємодії між суб'єктами ринку . Сучасний споживач очікує високого рівня зручності, персоналізації, доступності та швидкодії в процесі онлайн-покупок, що зумовлює необхідність адаптації інформаційних систем до нових вимог.

Актуальність роботи обумовлена тим, що Інтернет-магазини комп'ютерної техніки, як один із сегментів електронної комерції, демонструють стійке зростання. Попит на технічні засоби зумовлений постійною модернізацією апаратного та програмного забезпечення, а також розширенням сфер їх застосування. У зв'язку з цим зростає потреба у високоякісних інформаційних системах, здатних забезпечити ефективне управління товарними запасами, процесами замовлень, аналітикою продажів, взаємодією з клієнтами та іншими бізнес-процесами.

Інформаційна система онлайн-магазину виконує не лише облікову, а й стратегічну функцію, виступаючи інструментом оптимізації управлінської діяльності, підвищення ефективності взаємодії з клієнтами та забезпечення конкурентоспроможності підприємства. Сучасні технології, включаючи системи керування базами даних, хмарні рішення, веб-сервіси, засоби аналітики та штучного інтелекту, дозволяють створювати масштабовані, гнучкі та безпечні системи.

В умовах глобалізації навіть малі підприємства можуть виходити на міжнародні ринки, що, у свою чергу, вимагає врахування регуляторних вимог різних юрисдикцій, забезпечення захисту персональних даних, багатомовності та локалізації користувацького інтерфейсу.

Отже, вдосконалення інформаційних систем онлайн-магазинів комп'ютерної техніки є актуальним науково-практичним завданням, яке

охоплює комплекс технічних, організаційних, правових та економічних аспектів.

Метою дослідження є розширення функціональних можливостей інформаційної системи онлайн-магазину комп'ютерної техніки відповідно до сучасних вимог електронної комерції.

Для досягнення цієї мети в роботі розв'язано такі **задачі:**

— проаналізувати тенденції розвитку електронної комерції у сфері комп'ютерної техніки;

— визначити вимоги до інформаційної системи онлайн-магазину з урахуванням сучасних стандартів та очікувань користувачів;

— дослідити сучасні архітектурні рішення та компоненти, які реалізуються у подібних системах;

— здійснити аналіз функціональних можливостей існуючих рішень на ринку електронної комерції;

— сформулювати напрями вдосконалення функціональних модулів системи (каталог, кошик, система замовлень, захист даних, інтеграція);

— розробити рекомендації щодо розвитку та масштабування інформаційної системи.

Об'єктом дослідження є процеси функціонування та взаємодії інтернет-застосунків.

Предметом дослідження є програмні засоби інформаційної системи онлайн-магазину комп'ютерної техніки.

Апробація результатів роботи здійснена у доповіді на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [1].

Годлевський Д.В. Тарновський М.Г. Інформаційна система для онлайн-магазинун комп'ютерної техніки // Матеріали всеукраїнської науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25372

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні вимоги до інформаційних систем онлайн-магазинів

Інформаційна система онлайн-магазину комп’ютерної техніки у сучасних умовах повинна відповідати численним вимогам, які формуються як технічними стандартами, так і очікуваннями користувачів [2]. Конкуренція у сфері електронної комерції зумовлює необхідність постійного вдосконалення функціональності, безпеки та зручності використання подібних систем. Інформаційна система вже не є лише інструментом обліку, а стає критичним компонентом стратегічного розвитку бізнесу.

Однією з основних вимог є висока продуктивність та масштабованість системи. Навіть під час пікових навантажень, таких як святкові розпродажі чи акційні кампанії, сайт повинен залишатися швидким, надійним і безпечним. Цього можна досягти завдяки горизонтальному масштабуванню, автоматичному балансуванню навантаження, кешуванню даних та використанню розподілених баз даних.

Інтерфейс системи повинен бути адаптивним і зручним для користувача. З огляду на домінування мобільного трафіку, обов’язковою є підтримка адаптивного дизайну, що забезпечує комфортну роботу з сайтом на будь-якому пристрої. Актуальною є також концепція mobile-first design, коли саме мобільна версія є основою для розробки [3].

Сучасні користувачі очікують персоналізованого досвіду взаємодії. Рекомендаційні алгоритми, персональні акції, адаптовані пропозиції – все це створює відчуття індивідуального підходу. Для реалізації подібних функцій використовуються алгоритми машинного навчання, аналіз поведінки користувача та штучний інтелект [4].

Окрема увага приділяється безпеці персональних даних та фінансових транзакцій. Системи повинні дотримуватися норм GDPR, PCI DSS та інших міжнародних стандартів. Це передбачає шифрування даних, багаторівневу авторизацію, захист від атак типу XSS, SQL-ін’єкцій, CSRF та інше.

Впровадження SSL/TLS-сертифікатів, firewall, моніторинг активності користувачів — усе це частина комплексної стратегії кіберзахисту.

Інформаційна система має підтримувати інтеграцію з зовнішніми сервісами: платіжними системами (LiqPay, Stripe, PayPal), службами доставки (Нова Пошта, Meest), CRM-системами (Bitrix24, HubSpot), маркетинговими платформами (SendPulse, Mailchimp) [5]. Відкриті API, webhook-и, RESTful-архітектура дозволяють швидко масштабувати функціонал та зменшити ручну працю персоналу.

Не менш важливою є наявність аналітичних інструментів, які дозволяють відстежувати продажі, поведінку користувачів, ефективність рекламних кампаній. Вбудовані або інтегровані системи бізнес-аналітики (Power BI, Google Data Studio) дозволяють менеджменту приймати обґрунтовані рішення на основі реальних даних.

Програми лояльності, швидка реєстрація, чат-боти, one-click checkout, відгуки користувачів, багатомовність — це все стало стандартом, а не перевагою. Особливо для компаній, які прагнуть виходу на міжнародні ринки. У цьому випадку система повинна підтримувати різні валюти, локалізовані методи оплати та враховувати регіональні податкові особливості.

Необхідною є також гнучкість системи до змін – можливість оперативного внесення нових функцій, редагування контенту, запуску акцій без залучення розробників. Для цього використовуються візуальні CMS, шаблони, а також мікросервісна або headless-архітектура.

Підсумовуючи, сучасна інформаційна система онлайн-магазину – це не просто набір функцій, а багаторівнева екосистема, яка поєднує зручність, безпеку, аналітику, автоматизацію та здатність до масштабування. Її розробка вимагає системного підходу з урахуванням як технічних, так і бізнес-аспектів.

1.2 Основні технічні виклики при реалізації функціоналу

Розробка інформаційної системи онлайн-магазину стикається з великою кількістю технічних викликів, які охоплюють різні рівні — від баз даних до

інтерфейсу користувача, від архітектури до аналітики. Розуміння цих викликів є ключем до ефективної реалізації та подальшої підтримки системи.

Одним із базових завдань є проєктування структури каталогу товарів. Це складна задача, особливо в умовах багаторівневої категоризації, великої кількості атрибутів товарів та необхідності гнучких фільтрів. Реалізація «розумного» пошуку вимагає використання пошукових систем на кшталт Elasticsearch, які дозволяють здійснювати повнотекстовий пошук, сортування, агрегації та автопідказки [6].

Оптимізація запитів до бази даних — ще один критичний аспект. При високому навантаженні навіть невеликі затримки можуть призвести до втрати клієнтів. Застосовується індексація таблиць, кешування запитів (наприклад, через Redis), використання read replicas та шардинг для масштабування [7]. Також важливо обрати відповідну СУБД — для структурованих даних підійде PostgreSQL, для гнучких — MongoDB.

Реалізація функціоналу кошика замовлень має враховувати збереження даних навіть для неавторизованих користувачів. Це може реалізовуватись через сесії, cookies або локальне сховище браузера. Синхронізація стану кошика між пристроями після входу в акаунт — також непросте завдання, що потребує продуманої логіки бекенду.

Інтеграція з платіжними шлюзами — це не лише технічне, але й юридичне завдання. Потрібно забезпечити захист транзакцій, відповідність вимогам безпеки, обробку помилок та збоїв у транзакціях, верифікацію користувачів (3D Secure, OTP-коди).

Також слід реалізувати відстеження логістики в реальному часі. Це потребує взаємодії з API служб доставки, оновлення статусів замовлень, генерації номерів ТТН, а також сповіщення клієнта через email або месенджери.

Багаторівнева система авторизації забезпечує захист акаунтів клієнтів та менеджерів. Застосовуються OAuth 2.0, JWT-токени, CAPTCHA, SMS-підтвердження, а також обмеження IP-адрес для доступу до адміністративної панелі.

Важливим компонентом є адмінпанель, яка повинна забезпечити повний контроль над контентом магазину, управління товарами, замовленнями, клієнтами, аналітикою. Бажано впровадити гнучку систему прав доступу, щоб різні ролі (адмін, маркетолог, оператор) мали різні рівні доступу.

Необхідно також забезпечити захист від зовнішніх атак. Основні загрози — це SQL-ін’єкції, XSS-атаки, brute-force, CSRF, DDoS. Тут важлива як превентивна безпека (фільтрація запитів, обмеження частоти), так і постійний моніторинг активності через системи на кшталт Fail2Ban, WAF, IDS.

Ключовим етапом є вибір архітектури системи. Сучасні рішення часто реалізуються у вигляді мікросервісів — кожен модуль (авторизація, замовлення, аналітика) працює окремо. Це дозволяє масштабувати систему по частинах, оновлювати модулі незалежно один від одного, але потребує додаткових інструментів (Docker, Kubernetes, Kafka).

Крім того, варто впровадити DevOps-практики — CI/CD для швидких оновлень, автоматичне тестування, деплой через Git, контроль якості коду, staging-середовище. Це мінімізує ризики збоїв при впровадженні нових функцій.

Таким чином, технічна реалізація інформаційної системи — це багаторівневий процес, що вимагає злагодженої роботи команди розробників, DevOps, тестувальників та аналітиків. Правильне вирішення викликів на етапі проєктування забезпечує стійку роботу системи в умовах зростаючих вимог бізнесу та користувачів.

1.3 Вибір та обгрунтування аналогів

Одним з аналогів розроблюваної системи є інтернет-магазин Citrus (citrus.ua) — один із найбільших українських ритейлерів цифрової техніки (рис. 1.1). Магазин спеціалізується на продажу смартфонів, ноутбуків, аксесуарів та комп'ютерної техніки. Він має як фізичні торгові точки, так і зручний онлайн-майданчик, що дозволяє покупцям обирати товари з будь-якого місця.

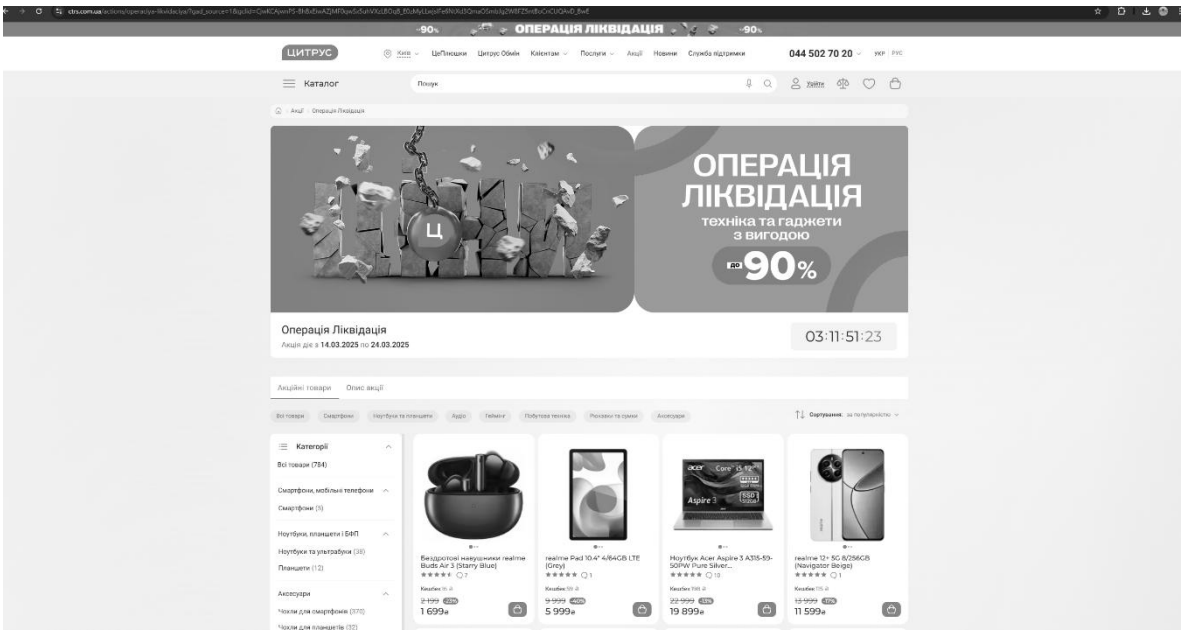

Рисунок 1.1 — Головна сторінка інтернет-магазину Citrus

Основні особливості:

— сучасний дизайн сайту: інтерфейс магазину виконаний у яскравих кольорах, що привертає увагу покупців; на головній сторінці представлені акційні пропозиції, новинки та популярні товари;

— розширена система фільтрів: користувачі можуть швидко знаходити потрібні товари за допомогою фільтрів за брендом, ціною, характеристиками тощо;

— функція Trade-In: магазин пропонує можливість обміну старої техніки на нову з доплатою, що є додатковим стимулом для покупців;

— онлайн-консультації: чат-боти та підтримка в реальному часі дозволяють користувачам отримувати відповіді на свої запитання безпосередньо на сайті;

— акційні пропозиції: магазин регулярно проводить сезонні знижки, розіграші та пропонує подарункові сертифікати.

Основні недоліки:

— високі ціни порівняно з конкурентами;

— обмежений асортимент бюджетної техніки;

— складна система повернення товару та гарантійного обслуговування.

Ще одним з аналогів є інтернет-магазин Brain (brain.com.ua) — спеціалізований магазин комп'ютерної техніки, комплектуючих та периферії (рис. 1.2). Він орієнтований на досвідчених користувачів, які шукають продуктивні комплектуючі для збірки ПК або професійного використання.

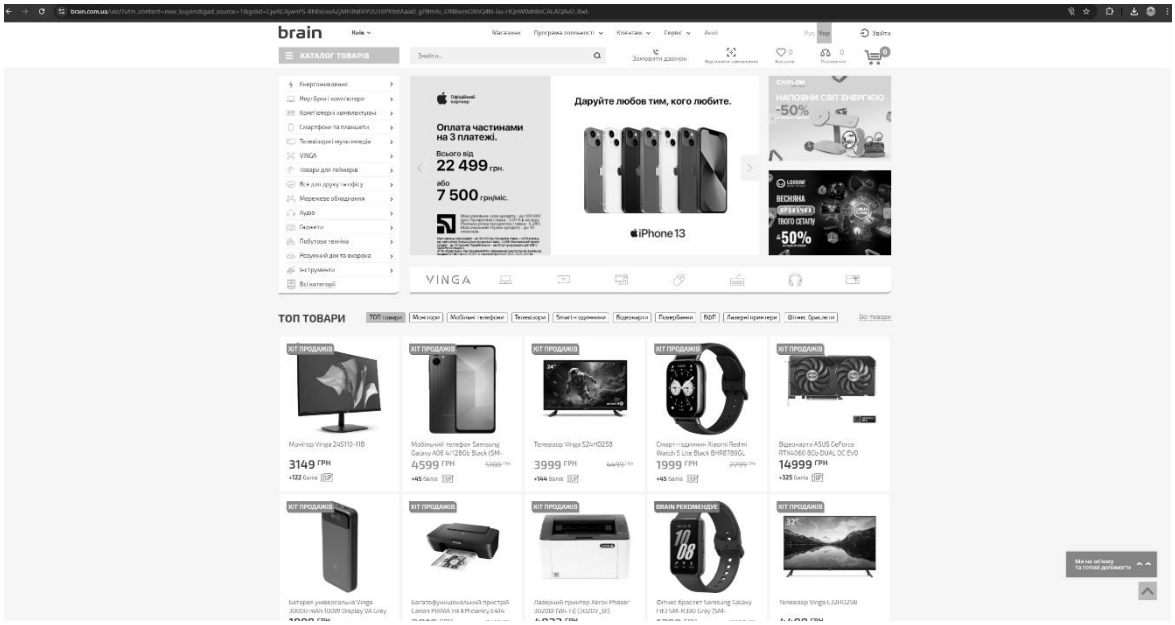

Рисунок 1.2 — Інтерфейс сайту інтернет-магазину Brain

Основні особливості:

— вузька спеціалізація; магазин пропонує широкий вибір комп'ютерних комплектуючих, таких як відеокарти, процесори, материнські плати, блоки живлення тощо;

— функція порівняння товарів; користувачі можуть порівнювати характеристики різних моделей, що допомагає їм приймати обґрунтовані рішення;

— інтеграція зі службами доставки; магазин співпрацює з такими службами, як Нова Пошта, Укрпошта та Meest, що забезпечує зручність доставки;

— гнучка система оплати; покупці можуть оплачувати товари онлайн-банкінгом, через розстрочку або оплату частинами;

— база оглядів і тестів; на сайті представлена велика кількість оглядів та тестів комп'ютерної техніки, що допомагає користувачам у виборі.

Недоліки:

— обмежена кількість фізичних магазинів для самовивозу;

— відсутність широкого вибору мобільних гаджетів і аксесуарів;

— інтерфейс сайту менш привабливий порівняно з конкурентами.

Rozetka (rozetka.ua) — найбільший український маркетплейс, який пропонує широкий асортимент товарів, включаючи комп'ютерну техніку, комплектуючі, ноутбуки та аксесуари. Магазин є одним із найпопулярніших серед українських покупців завдяки своїй зручності та великому вибору товарів.

Основні особливості:

— гігантський асортимент товарів; Rozetka пропонує товари від бюджетної техніки до преміальних моделей, що дозволяє задовольнити потреби різних категорій покупців;

— зручний пошук і категоризація; користувачі можуть швидко знаходити потрібні товари за допомогою пошуку та фільтрів;

— рейтингова система; покупці можуть залишати відгуки та ставити оцінки товарам, що допомагає іншим користувачам приймати рішення;

— великий вибір способів доставки; Rozetka пропонує кур'єрську доставку, поштові служби та самовивіз;

— програма Rozetka Premium; покупці можуть отримувати кешбек та знижки на доставку, що робить покупки ще вигіднішими.

Недоліки:

— високе навантаження на сайт у пікові періоди, що може впливати на швидкість роботи;

— деякі продавці на маркетплейсі пропонують неоригінальну продукцію;

— складна система повернення товарів;

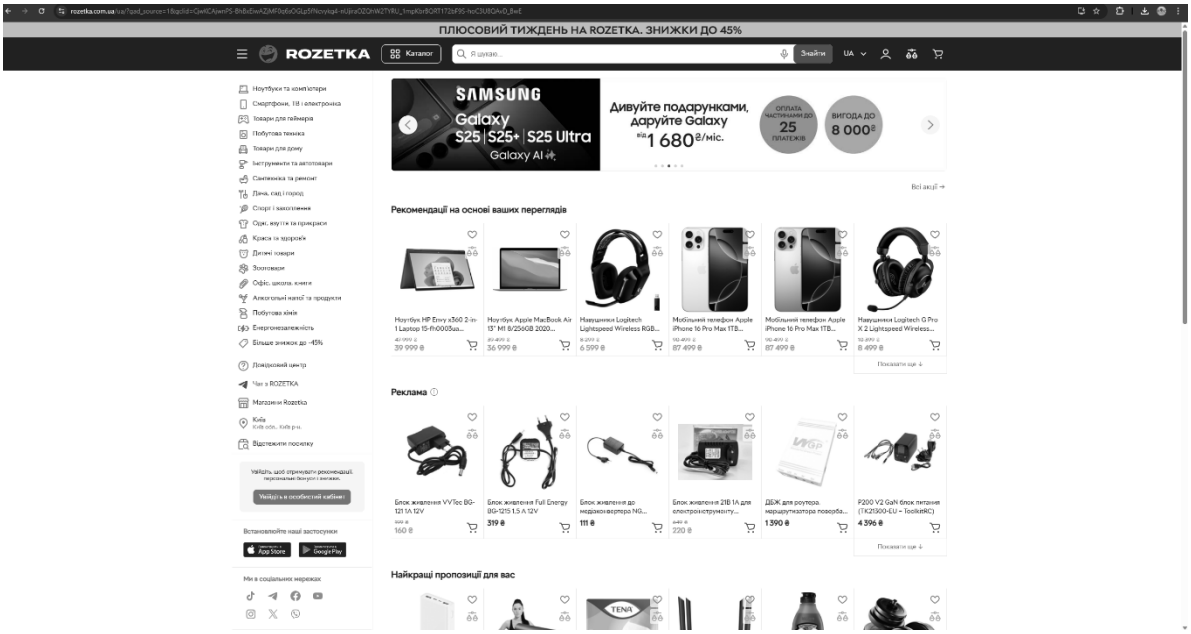

Рисунок 1.3 — Інтерфейс інтернет-магазину Rozetka

Усі ІС демонструють загальну тенденцію до автоматизації процесів, персоналізації взаємодії з користувачем, гнучкого масштабування функціоналу.

Водночас, існує ще потенціал для покращення: поліпшення зворотного зв’язку, спрощення повернення товарів, розширення аналітичних можливостей для покупців і менеджерів, впровадження рекомендацій на основі ШІ в реальному часі.

Таким чином, практичний аналіз реалізації функціоналу в діючих інформаційних системах онлайн-магазинів дозволяє зробити висновок про ефективність використання окремих компонентів, виявити недоліки та сформувати перелік рекомендованих рішень для побудови вдосконаленої системи.

2 ВИБІР ТА ОБҐРУНТУВАННЯ ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектура та структура веб-сайту як складова інформаційної системи онлайн-магазину комп’ютерної техніки

У сучасних умовах веб-сайт онлайн-магазину не є просто інтернет-сторінкою з переліком товарів. Це складний інформаційний ресурс, що виступає фронт-ендом масштабної інформаційної системи, яка охоплює весь цикл електронної комерції: від презентації продукції до обробки замовлень, інтеграції з платіжними системами, логістичними службами та внутрішніми бізнес-процесами, такими як управління складом чи аналітика продажів. Тому в цій роботі акцентовано увагу на інформаційній системі (ІС) як ключовому компоненті, що забезпечує ефективність, безпеку, масштабованість та гнучкість онлайн-магазину.

Розгляд ІС, а не лише веб-сайту, зумовлений кількома причинами:

— веб-сайт — це лише інтерфейс для взаємодії користувача з системою за ним стоять серверні модулі, бази даних, логіка бізнес-процесів та аналітика;

— ІС забезпечує обробку та зберігання даних, таких як облікові записи, товари, замовлення, транзакції, логістика, історія покупок;

— автоматизація, безпека та аналітика — це функції ІС, а не лише візуального представлення;

— успішне функціонування онлайн-магазину залежить від якості проєктування ІС, наприклад, від швидкості обробки замовлень чи захисту даних клієнтів.

Типи структур у веб-системах інтернет-магазинів:

— деревоподібна (ієрархічна) структура;

— сіткова (гратчаста) структура;

Деревоподібна це найпоширеніший тип для онлайн-магазинів; головна сторінка переходить до категорій (наприклад, «Комп’ютери», «Ноутбуки», «Периферія»), які мають підкатегорії («Ігрові ноутбуки», «Монітори»,

«Принтери»); така структура, наприклад, використовується на сайті Rozetka, що дозволяє логічно організувати тисячі товарів і легко масштабувати каталог.

Сіткова (гратчаста) структура, цей підхід об’єднує розділи за різними критеріями — категоріями, новинами, акціями, відгуками. Наприклад, товар може бути одночасно в категорії «Ноутбуки» та в розділі «Знижки тижня» на маркетплейсах, це характерно для великих платформ.

Сучасні онлайн-магазини базуються на компонентному підході, де сайт складається з модулів:

— модуль каталогу (структурований список товарів із фільтрами);

— модуль пошуку (повнотекстовий або атрибутивний з автодоповненням);

— картка товару (з фото, характеристиками, відгуками);

— модуль кошика та оформлення замовлення;

— особистий кабінет користувача (перевірка замовлень, бонуси, обране);

— адміністративна панель (управління товарами, цінами, статистикою);

— інтеграційний модуль (зв’язок із CRM, API служб доставки, платіжними шлюзами).

Сайт онлайн-магазину комп’ютерної техніки має включати такі функціональні зони:

— інформаційну зону — статті, інструкції, FAQ;

— маркетингову зону — акційні банери, персоналізовані пропозиції;

— аналітичну зону — збір даних про дії користувачів через Google Analytics чи внутрішню аналітику:

— інтерактивну зону — чат-боти, форми відгуків, конфігуратори ПК;

— структура навігації.

Для магазинів техніки важлива розгалужена, але не перевантажена навігація, яка включає:

— основне меню з категоріями;

— «хлібні крихти» для зручного повернення;

— бокові фільтри за характеристиками;

— пошукову панель з підказками;

— сортування товарів (за ціною, популярністю, рейтингом).

Сайт має бути адаптивним для різних пристроїв (десктопи, планшети, смартфони) і доступним — з швидким завантаженням, підтримкою для людей з порушеннями зору (контрастність, текстові альтернативи зображенням).

Грамотно побудована структура ІС спрощує підтримку та масштабування системи, підвищує юзабіліті, що впливає на конверсію та полегшує SEO-оптимізацію завдяки логічним URL і шаблонам сторінок.

Детальніше загальна структура інформаційної системи онлайн-магазину представлена в додатку В рисунок В.1.

2.2 Функціональні можливості інформаційної системи онлайн-магазину комп'ютерної техніки

Інформаційна система, що забезпечує функціонування онлайн-магазину комп'ютерної техніки, охоплює повний спектр бізнес-процесів — від управління асортиментом до обробки платежів, доставки, аналітики та підтримки клієнтів. На відміну від простого веб-сайту, ІС виступає посередником між користувачем і внутрішніми сервісами, забезпечуючи автоматизацію, зручність і надійність взаємодії, що особливо важливо в умовах високої конкуренції.

Управління каталогом товарів є ключовою функцією ІС. Комп'ютерна техніка має численні атрибути (процесори, обсяг пам'яті, відеокарти, порти), тому система підтримує гнучке додавання, редагування та видалення товарів. Наприклад, каталог Rozetka дозволяє фільтрувати ноутбуки за діагоналлю екрана чи типом процесора, що спрощує пошук. Повнотекстовий пошук із автопідказками та технічні фільтри забезпечують швидку навігацію в великому асортименті. Механізми сортування (за ціною, популярністю, новизною) підвищують зручність для користувача.

Обробка замовлень охоплює шлях від натискання кнопки «Купити» до завершення транзакції. ІС перевіряє наявність товару, резервує його на складі,

формує рахунок, фіксує дані клієнта та спосіб доставки. Система автоматично передає інформацію менеджерам і логістам, а клієнт отримує сповіщення про статуси («Оформлено», «Відправлено») через email чи месенджери. Для забезпечення відмовостійкості ІС використовує кешування даних і балансування навантаження, що дозволяє обробляти замовлення навіть під час пікових навантажень, наприклад, у період розпродажів. Інтеграція з API служб доставки, таких як Нова Пошта, синхронізує дані в реальному часі.

Механізми оплати дозволяють клієнтам використовувати банківські картки, електронні гаманці, оплату частинами чи кредит. ІС забезпечує шифрування даних, обробку помилок і відповідність стандартам PCI DSS. Інтеграція з LiqPay чи Stripe підтримує верифікацію через 3D Secure, що підвищує безпеку. Повернення коштів у разі відмови від покупки обробляється автоматично, а клієнт отримує сповіщення про успішне повернення, що сприяє довірі до магазину.

Основні функціональні блоки ІС, такі як каталог, обробка замовлень і механізми оплати, взаємодіють для забезпечення безперебійної роботи магазину, як показано на рис. 2.2.

Рисунок 2.1 — Основні функціональні блоки інформаційної системи онлайн-магазину

Функціонал доставки включає формування транспортних накладних, передачу даних до служб доставки (Нова Пошта, Meest, Justin) і відстеження статусу відправлення. Клієнти можуть обирати точки самовивозу, розраховувати вартість доставки в реальному часі чи вказувати точну адресу

через інтеграцію з картографічними сервісами, що покращує користувацький досвід. Система зберігає історію доставок для аналізу та вирішення спірних ситуацій.

Особистий кабінет користувача забезпечує доступ до історії замовлень, програм лояльності, обраних товарів і персональних даних. Клієнти можуть змінювати спосіб оплати, повторювати замовлення одним кліком чи відстежувати терміни гарантійного обслуговування, що економить час і підвищує зручність. Підтримка багатомовності в кабінеті дозволяє обслуговувати іноземних клієнтів, що актуально для глобалізації бізнесу.

Адміністративна панель є інструментом для менеджерів, дозволяючи керувати товарами, цінами, акціями, відгуками та замовленнями. Система прав доступу розмежовує ролі: наприклад, маркетолог створює знижки, але не має доступу до фінансових звітів, що підвищує безпеку. Панель підтримує імпорт/експорт даних для швидкого оновлення асортименту.

Система аналітики збирає дані про поведінку користувачів, популярність товарів і ефективність маркетингових кампаній. Ключові метрики, такі як конверсія, середній чек, частота повернень чи відсоток покинутих кошиків, відображаються в дашбордах (наприклад, Google Data Studio), що допомагає оптимізувати продажі та планувати закупівлі. Аналітика підтримує прогнозування попиту, наприклад, визначення популярних моделей ноутбуків перед новим навчальним роком.

Персоналізація та інтерактивні функції стають стандартом сучасних ІС. Рекомендаційні системи на основі штучного інтелекту аналізують історію переглядів і покупок, пропонуючи клієнтам релевантні товари, як це реалізовано на Amazon. Чат-боти з обробкою природної мови відповідають на запитання клієнтів у реальному часі, а інтеграція з голосовими асистентами (наприклад, Google Assistant) дозволяє замовляти товари голосом, що підвищує доступність.

Типовий сценарій роботи ІС демонструє її комплексність. Наприклад, клієнт обирає ноутбук, застосовує фільтр за брендом і ціною, додає товар до кошика, обирає доставку Новою Поштою та оплачує через LiqPay. ІС перевіряє

наявність товару, резервує його, генерує накладну, надсилає клієнту сповіщення про статус і передає дані логістам. Аналітика фіксує, які фільтри використовував клієнт, щоб покращити каталог у майбутньому. Детальна схема взаємодії користувачів з інформаційною системою наведена в додатку В (рисунок В.3).

2.3 Технологічне забезпечення інформаційної системи онлайн-магазину

Сучасна інформаційна система (ІС) онлайн-магазину комп'ютерної техніки потребує ретельно підібраного технологічного стека, який забезпечує масштабованість, безперебійну роботу, безпеку та ефективну взаємодію підсистем. У сфері торгівлі комп'ютерною технікою, де обробляються великі обсяги даних і потрібні часті оновлення, вибір технологій є критично важливим для успіху проєкту.

На клієнтській частині (frontend) використовуються технології для створення інтуїтивно зрозумілого, адаптивного інтерфейсу, що працює на всіх пристроях — від смартфонів до великих екранів. Найпоширеніші фреймворки — React, Vue.js або Nuxt.js із серверним рендерингом, які застосовуються, наприклад, у магазинах типу Rozetka для швидкого завантаження сторінок [8]. Їх переваги — висока продуктивність, гнучкість і підтримка компонентного підходу, що спрощує додавання нових функцій. Адаптивний дизайн і підтримка Progressive Web Applications (PWA) забезпечують зручність для мобільних користувачів, підвищуючи конверсію.

На серверній частині (backend) ІС обробляє запити до бази даних, авторизацію, логіку замовлень, генерацію накладних і інтеграцію з зовнішніми сервісами. Платформи Node.js, Django або Laravel, які використовуються великими платформами, як Amazon, дозволяють будувати REST API або GraphQL API для комунікації з frontend. Backend забезпечує валідацію даних і захист від атак, детальніше про які йтиметься в розділі 2.4. Масштабованість досягається через асинхронну обробку запитів і використання серверів із балансуванням навантаження.

База даних є центральною ланкою ІС, зберігаючи дані про товари, користувачів, замовлення та транзакції. Для структурованих даних, як у каталогах Rozetka, використовуються реляційні СУБД, такі як PostgreSQL або MySQL, що підтримують складні SQL-запити. Для неструктурованих даних (наприклад, історія переглядів) застосовуються документоорієнтовані бази, як MongoDB. Оптимізація запитів через індексацію та кешування (Redis) забезпечує швидкий доступ до даних навіть при високих навантаженнях.

Технологічний стек, що включає frontend, backend і бази даних, формує основу ІС, як показано на рис. 2.3.

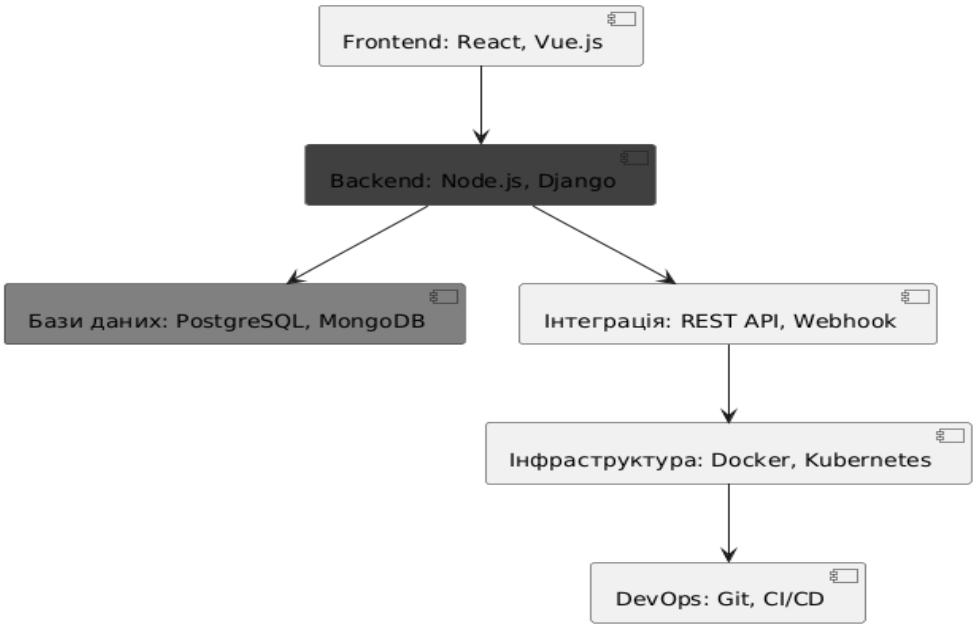

Рисунок 2.2 — Технологічний стек інформаційної системи

Інтеграція з зовнішніми сервісами — платіжними шлюзами (LiqPay, Stripe), CRM (Bitrix24), службами доставки (Нова Пошта) — реалізується через REST API або Webhook. Наприклад, після оплати через Stripe система отримує підтвердження транзакції в реальному часі, оновлюючи статус замовлення. Контейнеризація (Docker) ізольує модулі системи, а Kubernetes, який використовують великі магазини, як eBay, забезпечує оркестрацію, автоматичне відновлення та балансування навантаження для роботи 24/7.

Система контролю версій (Git) координує роботу розробників, дозволяючи повертатися до попередніх станів системи. CI/CD-пайплайни (наприклад, GitHub Actions) автоматизують тестування, деплой і моніторинг, що мінімізує ризики збоїв. Staging-середовища дозволяють тестувати нові функції перед релізом, забезпечуючи стабільність для користувачів.

Таким чином, технологічний стек ІС поєднує frontend, backend, бази даних, інтеграцію та DevOps-інструменти, забезпечуючи швидку, безпечну та масштабовану роботу магазину. Його правильний вибір визначає конкурентоспроможність бізнесу в умовах стрімкого розвитку електронної комерції.

2.4 Безпека та захист даних в інформаційній системі онлайн-магазину

Інформаційна безпека є критично важливою для онлайн-магазинів, які обробляють конфіденційні дані: персональну інформацію, платіжні реквізити, історію замовлень. Витік або компрометація даних може призвести до фінансових збитків, втрати довіри та юридичних наслідків. Захист ІС реалізується на мережевому, програмному, інфраструктурному та організаційному рівнях, створюючи багаторівневу систему безпеки.

На програмному рівні ІС захищається від поширених атак: SQL-ін’єкцій, XSS (міжсайтове скриптування), CSRF (підробка міжсайтових запитів). Наприклад, Rozetka використовує валідацію вхідних даних у формах пошуку, щоб запобігти SQL-ін’єкціям. Валідація запитів, фільтрація даних і захищена автентифікація мінімізують ризики. Сучасні системи також застосовують поведінкову аналітику, яка виявляє підозрілі дії користувачів, як спроби введення шкідливого коду.

Захист акаунтів клієнтів і менеджерів забезпечується системою автентифікації. Двофакторна автентифікація (2FA) через SMS чи email, як у платіжних системах типу LiqPay, забезпечує додатковий рівень безпеки. Використовуються JWT-токени, CAPTCHA та обмеження доступу за IP для адміністративної панелі. Ролі (адміністратор, оператор, клієнт)

розмежовуються, щоб маркетолог, наприклад, не мав доступу до платіжних даних. Підхід Zero Trust, який вимагає перевірки кожного запиту, набирає популярності в сучасних ІС.

Для захисту передаваних даних усі запити між браузером і сервером передаються через HTTPS із SSL/TLS-сертифікатами, що шифрують трафік. Наприклад, маркетплейси, як Amazon, використовують TLS для захисту платіжних транзакцій. Дані в базі (паролі, платіжні реквізити) шифруються алгоритмами bcrypt або Argon2. Локальні файли, як чеки чи договори, зберігаються в зашифрованому вигляді, щоб запобігти несанкціонованому доступу.

Багаторівневий захист, що охоплює програмний, автентифікаційний і шифрувальний рівні, формує основу безпеки ІС, як показано на рис. 2.4.

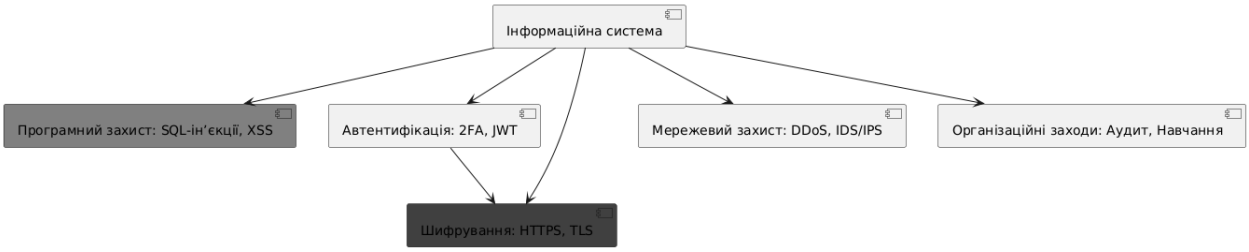

Рисунок 2.3 — Рівні захисту інформаційної системи

Захист від зовнішніх атак забезпечується мережевими фільтрами, системами виявлення вторгнень (IDS/IPS), проксі-серверами та обмеженням частоти запитів (rate limiting). Зазначені механізми блокують DDoS-атаки та вторгнення. Системи, як Cloudflare, використовуються великими магазинами для захисту від DDoS. Моніторинг логів через інструменти, як Fail2Ban, дозволяє виявляти аномалії. Регулярне сканування вразливостей забезпечує швидке реагування на нові загрози.

Підвищенню рівня безпеки сприяють організаційні заходи, які включають регулярну зміну паролів, резервне копіювання даних і аудит активності. Персонал проходить навчання з кібербезпеки, наприклад, щодо розпізнавання фішингових атак, що проводяться раз на квартал. Аудит системи безпеки, який

включає перевірку логів і тестування на проникнення, виконується щорічно для виявлення слабких місць. Обмеження доступу до чутливих даних мінімізує ризики внутрішніх порушень.

Відповідно до стандартів GDPR, PCI DSS та українському законодавству про захист даних клієнти мають право на доступ, видалення чи обмеження обробки своїх даних, що має бути реалізовано в особистому кабінеті. Політика конфіденційності повинна бути прозорою, а передача даних третім особам обмежена. Наприклад, платіжні шлюзи, як Stripe, забезпечують відповідність PCI DSS для безпечної обробки транзакцій.

Таким чином, безпека ІС є невід'ємною складовою, що забезпечує захист даних, довіру клієнтів і конкурентоспроможність бізнесу. Комплексний підхід, поєднуючи технічні та організаційні заходи, дозволяє мінімізувати ризики та відповідати сучасним вимогам.

2.5 Інтеграція інформаційної системи з зовнішніми сервісами

Інтеграція інформаційної системи (ІС) онлайн-магазину з зовнішніми сервісами є ключовою умовою її функціональності та конкурентоспроможності. Сучасні платформи взаємодіють із платіжними, логістичними, маркетинговими та аналітичними сервісами, що автоматизує процеси та покращує користувацький досвід.

Інтеграція з платіжними шлюзами забезпечує зручні та безпечні способи оплати — банківські картки, Google Pay, Apple Pay, оплата частинами. Через API LiqPay чи Stripe ІС передає дані транзакції (сума, валюта, ідентифікатор замовлення), отримує підтвердження оплати та оновлює статус. Система обробляє помилки, повернення коштів і підтримує верифікацію (3D Secure). Наприклад, Rozetka використовує Stripe для міжнародних платежів, забезпечуючи відповідність PCI DSS.

Інтеграція з логістичними компаніями автоматизує доставку. API Нової Пошти передає дані про відправлення (адреса, вага, тип доставки), генерує накладну та повертає трекінг-номер. ІС відстежує статуси («Відправлено»,

«Доставлено») і сповіщає клієнта. Інтеграція з картографічними сервісами, як Google Maps, дозволяє клієнтам обирати точки самовивозу чи вказувати точну адресу, що підвищує зручність.

CRM-системи (Bitrix24, HubSpot) систематизують дані про клієнтів, їх замовлення, вподобання та звернення. Через API ІС передає історію покупок і контакти, дозволяючи менеджерам створювати персоналізовані пропозиції. Наприклад, HubSpot аналізує поведінку клієнтів, щоб сегментувати аудиторію для маркетингових кампаній. Це підвищує конверсію та лояльність.

Схема інтеграції ІС із платіжними, логістичними та CRM-системами представлена на рис. 2.5.

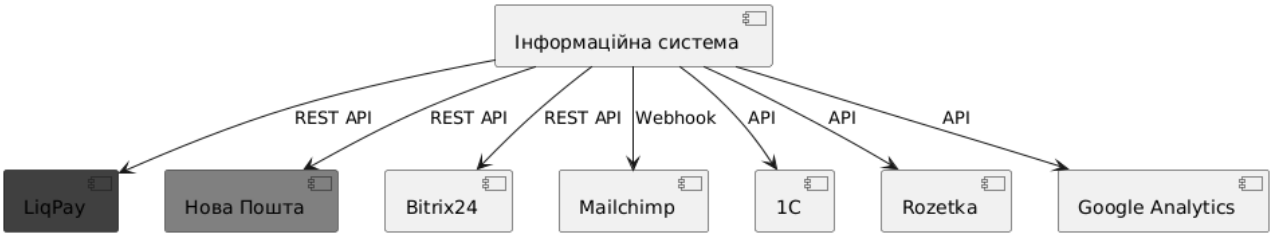

Рисунок 2.4 — Схема інтеграції інформаційної системи з зовнішніми сервісами

Email-маркетинг і push-сповіщення через Firebase інформують клієнтів про статус замовлення, що підвищує їх залученість. Платформи, як Mailchimp чи SendPulse, автоматизують розсилки про знижки, новинки чи покинуті кошики. Webhook надсилає подію (наприклад, «кошик залишено»), а ІС формує персоналізоване повідомлення. Аналітика відкриттів і кліків допомагає оптимізувати кампанії.

Інтеграція з ERP-системами (1С, SAP) синхронізує складський облік, фінанси та закупівлі. Наприклад, ІС передає дані про залишки товарів, автоматично оновлюючи каталог. Аналітичні платформи, як Google Analytics, отримують дані про поведінку користувачів (перегляди, кліки), що допомагає оцінити ефективність інтерфейсу чи реклами.

Для розширення продажів ІС інтегрується з маркетплейсами, як Rozetka чи Prom.ua. API передає дані про товари, ціни та залишки, синхронізуючи каталог і замовлення. Це дозволяє магазину охопити ширшу аудиторію без ручного дублювання даних.

Інтеграція може стикатися з проблемами, як затримки в API (наприклад, через перевантаження серверів Нової Пошти), несумісність форматів даних чи потреба в оновленні API. Для їх вирішення використовуються проміжні шари (middleware), моніторинг запитів і регулярне тестування.

Таким чином, інтеграція з зовнішніми сервісами автоматизує бізнес-процеси, підвищує швидкість обслуговування та забезпечує цифрову трансформацію. Її ефективність залежить від правильного вибору технологій (REST API, Webhook) і вирішення технічних викликів.

2.6 Масштабування інформаційної системи онлайн-магазину

Масштабування інформаційної системи (ІС) є ключовим для забезпечення її продуктивності, стабільності та здатності обробляти зростаючі обсяги користувачів і транзакцій. Особливо це актуально для онлайн-магазинів комп’ютерної техніки, де пікові навантаження виникають під час розпродажів чи сезонних акцій. Масштабування охоплює апаратні, програмні та архітектурні рішення. Розрізняють вертикальне та горизонтальне масштабування.

Вертикальне масштабування передбачає збільшення потужності серверів (додавання CPU, RAM, дискового простору). Наприклад, невеликі магазини можуть оновити сервер для обробки більшої кількості запитів до каталогу. Однак цей підхід обмежений вартістю апаратного забезпечення та неможливістю безкінечного нарощування ресурсів.

Горизонтальне масштабування додає нові сервери, розподіляючи навантаження між ними. Це дозволяє обробляти тисячі одночасних користувачів, як у випадку з Amazon під час Black Friday. Використовуються кластери серверів, які синхронізуються через розподілені бази даних (наприклад, PostgreSQL із реплікацією).

Для розподілу запитів між серверами для запобігання перевантаженню застосовується балансування навантаження. Найбільш відомими балансувальниками навантаження є Nginx та HAProxy. Наприклад, Rozetka використовує Nginx для рівномірного розподілу трафіку між серверами каталогу та оплати. Кешування через Redis або Memcached прискорює доступ до популярних товарів, зменшуючи навантаження на базу даних.

Модель масштабування, що включає вертикальне, горизонтальне масштабування та балансування навантаження, формує основу продуктивної ІС, як показано на рис. 2.6.

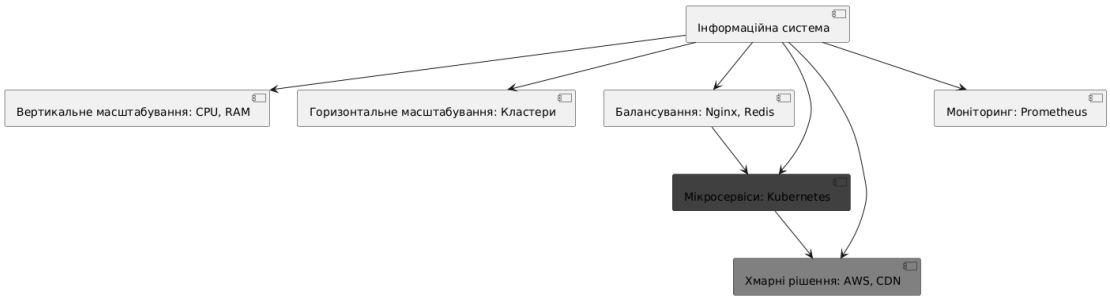

Рисунок 2.5 — Модель масштабування інформаційної системи

Гнучкість масштабування забезпечує мікросервісна архітектура. Мікросервіси розбивають ІС на незалежні модулі (каталог, оплата, доставка), кожен із яких масштабується окремо. Kubernetes, який використовує Amazon, оркеструє мікросервіси, забезпечуючи їх автоматичне розгортання та відновлення. Це дозволяє, наприклад, масштабувати модуль оплати під час пікових навантажень, не зачіпаючи каталог.

Ще одним підходом до забезпечення гнучкості масштабування є використання хмарних рішень. Хмарні платформи (AWS, Azure, Google Cloud) надають гнучке масштабування через віртуальні машини, контейнери та серверлес-архітектури. AWS Auto Scaling автоматично додає ресурси під час зростання трафіку, що використовують великі магазини. CDN (Cloudflare, Akamai) прискорює доставку статичного контенту, як зображення товарів, користувачам у різних регіонах.

Невід’ємною частиною масштабування є моніторинг продуктивності. Такі інструменти, як Prometheus і Grafana, відстежують метрики (час відгуку, використання CPU, кількість запитів), дозволяючи виявляти вузькі місця. Наприклад, якщо модуль оплати працює повільно, система сповіщає адміністратора, який додає сервери.

Слід зазначити, що масштабування супроводжується складнощами: управління мікросервісами вимагає координації, а хмарні сервіси збільшують витрати. Несумісність між модулями чи затримки в синхронізації баз даних можуть знижувати продуктивність. Для вирішення використовуються автоматизовані тести, моніторинг і резервні копії.

Таким чином, масштабування ІС поєднує апаратні, програмні та хмарні рішення, забезпечуючи продуктивність і стабільність. Його ефективність залежить від правильного вибору архітектури та інструментів моніторингу.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОНЛАЙН-МАГАЗИНУ КОМП'ЮТЕРНОЇ ТЕХНІКИ

3.1 Структура та загальна організація інформаційної системи

Інформаційна система (ІС) онлайн-магазину комп'ютерної техніки розроблена як сучасний веб-додаток, що базується на принципах Single Page Application (SPA). Використання SPA дозволяє забезпечити швидку та плавну взаємодію користувача з інтерфейсом шляхом динамічного оновлення вмісту сторінки без повного перезавантаження. Це зменшує затримки, покращує користувацький досвід і оптимізує продуктивність системи навіть при обробці великої кількості товарів, складних фільтрів чи пошукових запитів. Система побудована з використанням сучасних веб-технологій, зокрема HTML5 для семантичної розмітки, CSS3 для стилізації та JavaScript (ES6+) для реалізації логіки взаємодії, що забезпечує високу функціональність, адаптивність і відповідність стандартам веб-розробки.

Основна мета системи — створення зручного, інтуїтивно зрозумілого та ефективного інструменту для перегляду, пошуку, вибору та замовлення комп'ютерної техніки. ІС підтримує широкий спектр функцій, включаючи відображення каталогу товарів, пошук за ключовими словами, фільтрацію за технічними характеристиками, управління кошиком, збереження вподобаних товарів, оформлення замовлення та генерацію персоналізованих рекомендацій. Усі ці функції інтегровані в єдину модульну структуру, що полегшує підтримку, масштабування та подальший розвиток системи. Модульність досягається завдяки чіткому розподілу функціональних компонентів, кожен із яких відповідає за окрему задачу, але працює у взаємодії з іншими через JavaScript-функції та DOM-маніпуляції.

Основними модулями системи є:

— каталог товарів;

— система пошуку;

— фільтрація;

— кошик;

— список вподобаних товарів;

— оформлення замовлення;

— рекомендації;

— навігація.

Каталог товарів забезпечує ієрархічне відображення категорій (наприклад, «Зібрані ПК», «Ноутбуки», «Комплектуючі»), підкатегорій (наприклад, «Відеокарти», «Процесори») та окремих товарів із детальними характеристиками, такими як ціна, бренд, рік випуску, технічні параметри (обсяг пам’яті, тип процесора) та зображення; каталог є центральним елементом системи, через який користувач взаємодіє з асортиментом магазину;

Система пошуку дозволяє реалізувати швидкий і точний пошук товарів за назвою чи ключовими словами. Результати відображаються в реальному часі у спливаючому вікні ( ), що підвищує зручність і швидкість доступу до потрібних продуктів;

Фільтрація надає можливість налаштування відображення товарів за критеріями, такими як ціна, рік випуску, бренд, обсяг пам’яті чи тип відеокарти. Фільтри генеруються динамічно залежно від обраної категорії чи підкатегорії, що робить їх гнучкими та адаптивними;

Кошик дозволяє додавати товари, змінювати їх кількість, видаляти позиції та переглядати загальну суму замовлення. Кошик доступний у двох режимах: компактному (спливаюче вікно,) і повноекранному (renderFullCart()), що забезпечує гнучкість використання;

Список вподобаних товарів дозволяє користувачам зберігати товари, які їх зацікавили, у списку favorites для подальшого перегляду чи порівняння. Модуль підвищує зручність вибору продуктів, особливо при повторних відвідуваннях сайту;

Оформлення замовлення містить форму (<form id="checkout-form">) для введення особистих даних (ПІБ, телефон, email), вибору способу доставки та

оплати. Форма включає валідацію для перевірки коректності введених даних, що забезпечує надійність процесу;

Модуль рекомендацій дозволяє генерувати персоналізовані пропозиції на основі категорії товару чи випадкового вибору, відображаючи їх на головній сторінці чи сторінці товару. Цей модуль сприяє підвищенню продажів і покращенню користувацького досвіду;

Навігація реалізує інтуїтивну систему переходів між розділами (головна, каталог, про нас, контакти, кошик) за допомогою навігаційного меню (<nav>) і «хлібних крихт» (<nav id="breadcrumbs">), що полегшує орієнтацію в системі.

На рис. 3.1 зображено схему структури інформаційної системи онлайн-магазину комп'ютерної техніки. Схема ілюструє взаємозв'язок між основними модулями та їхню інтеграцію в єдину архітектуру. У центрі розташовано основний контейнер <main id="content">, який динамічно заповнюється вмістом залежно від дій користувача. Навколо нього розміщені модулі: каталог, пошук, фільтри, кошик, вподобані товари, оформлення замовлення та рекомендації. Кожен модуль пов'язаний із JavaScript-функціями, які керують його поведінкою, та HTML-елементами, що формують інтерфейс. Наприклад, модуль каталогу пов'язаний із функцією renderCatalog(), яка генерує список товарів, а модуль кошика — із функціями addToCart() і updateCart(). Стрілки на схемі показують потік даних, наприклад, як результати пошуку чи фільтрації оновлюють вміст каталогу. У верхній частині схеми зображено <header> із навігаційним меню та «хлібними крихтами», а в нижній — localStorage як механізм збереження даних між сеансами.

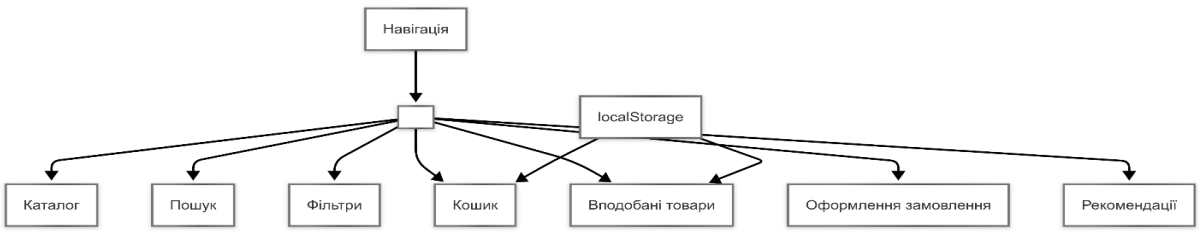

Рисунок 3.1 – Схема структури інформаційної системи онлайн-магазину комп'ютерної техніки

Для наочної демонстрації реалізації структури системи в додатку В (рисунок В.2) зображено скріншот сторінки каталогу товарів, яка є ключовим елементом інтерфейсу. Скріншот показує навігаційне меню в <header>, що містить посилання на розділи «Головна», «Каталог», «Про нас», «Контакти» та іконки кошика й вподобаних товарів. Під меню розташовані «хлібні крихти» (<nav id="breadcrumbs">), які відображають шлях, наприклад, «Головна > Каталог > Ноутбуки». У лівій частині екрана видно бічну панель із фільтрами (<aside id="filter">), що включають критерії, такі як ціна, бренд і рік випуску. Центральна частина (<main id="content">) містить сітку карток товарів, кожна з яких включає зображення, назву, ціну, технічні характеристики та кнопки «Додати до кошика» і «Вподобати». Скріншот ілюструє семантичну розмітку HTML5, модульну організацію та інтуїтивний інтерфейс, що відповідають описаній структурі системи.

Для підтвердження семантичної розмітки та структури інтерфейсу нище вставлено фрагмент HTML-коду, що відповідає за відображення каталогу товарів. Код включає основний контейнер <main id="content">, який заповнюється картками товарів, семантичні теги <header>, <nav> і <aside> для навігації та фільтрів, а також атрибути aria-label для забезпечення доступності. Наприклад, тег <nav id="breadcrumbs"> містить список посилань для «хлібних крихт», а <aside id="filter"> включає форму з чекбоксами для фільтрації. Цей фрагмент демонструє використання HTML5 для створення чіткої та доступної структури, що є основою модульної організації системи.

```
<header>
      <div class="header-content">
         <h1>Магазин Комп'ютерної Техніки</h1>
         <nav aria-label="Головна навігація">
<button class="button" onclick="navigate('home')">Головна</button>
<button class="button" onclick="navigate('catalog')">Каталог</button>
<button class="button" onclick="navigate('about')">Про нас</button>
<button class="button" onclick="navigate('contacts')">Контакти</button>
</nav>
  <div class="cart-favorites">
<button class="button" id="favorites-toggle" onclick="toggleFavorites()" aria-label="Вподобані">❤️ </button>
```

```
<button class="button" id="cart-toggle" onclick="toggleCart()" aria-label="Кошик">🛒
<span id="cart-count">0</span></button>
</div>
</div>
</header>
```

Структура системи базується на використанні HTML5, що забезпечує семантичну розмітку та підтримку стандартів доступності. Використовуються теги <header>, <nav>, <main>, <section>, <aside>, які чітко визначають призначення кожного блоку сторінки. Наприклад, <header> містить логотип, навігаційне меню та кнопки для доступу до кошика й вподобаних товарів, тоді як <main> відображає основний контент (каталог, сторінку товару чи форму замовлення). Атрибути aria-label застосовуються до навігаційних елементів і кнопок, що покращує доступність для користувачів із обмеженими можливостями. Наприклад, кнопка кошика має атрибут aria-label="Відкрити кошик", що полегшує її використання для програм зчитування екрана.

Основний принцип роботи системи — це динамічне управління DOM-деревом за допомогою JavaScript. Усі зміни вмісту сторінки, такі як відображення товарів, оновлення кошика чи застосування фільтрів, виконуються шляхом маніпуляцій із DOM без перезавантаження сторінки. Функція navigate() відповідає за перехід між розділами (головна сторінка, каталог, кошик тощо), оновлюючи вміст <main id="content"> і «хлібні крихти» (<nav id="breadcrumbs">). Наприклад, при виборі категорії «Ноутбуки» викликається renderCatalog("Ноутбуки"), яка генерує список товарів із відповідними характеристиками. Це забезпечує швидкість роботи та мінімізує звернення до сервера, що є важливим для локальних прототипів, таких як даний.

Дані про товари зберігаються у вигляді структурованого JavaScript-об’єкта products, який містить ієрархію категорій, підкатегорій і товарів. Кожен товар описаний атрибутами, такими як id, name, price, year, brand, технічні характеристики (наприклад, memory, processor) та шлях до зображення (image). Така структура дозволяє легко додавати нові товари чи категорії без зміни коду, що забезпечує гнучкість і масштабність. Наприклад, додавання нової категорії

«Периферія» потребує лише оновлення об’єкта products без модифікації функцій рендерингу. У перспективі products може бути замінений на API-запити до серверної бази даних, що підвищить ефективність роботи з великими обсягами даних.

Для збереження стану системи, зокрема вмісту кошика (cart) та списку вподобаних товарів (favorites), передбачається використання localStorage. Це дозволяє зберігати дані між сеансами, забезпечуючи безперервний досвід навіть після закриття браузера. Наприклад, товари, додані до кошика, зберігаються у localStorage у форматі JSON через функцію saveCart() (хоча в поточній реалізації вона ще не застосовується). При повторному відкритті сторінки функція loadCart() відновлює дані, дозволяючи користувачу продовжити роботу. Аналогічно, список вподобаних товарів зберігається через saveFavorites() і loadFavorites(), що підвищує зручність.

Інтерфейс системи побудований із урахуванням принципів UX/UI, що забезпечують простоту та зручність використання. Навігаційне меню в <header> містить чіткі посилання на основні розділи, а «хлібні крихти» дозволяють бачити поточне місце в структурі сайту та швидко повертатися до попередніх рівнів. Наприклад, при перегляді товару користувач бачить шлях «Головна > Каталог > Зібрані ПК > Геймерський ПК Titan X», що полегшує орієнтацію. Поле пошуку та фільтри розташовані в зручних місцях (у верхній частині сторінки та в бічній панелі), а спливаючі вікна кошика й вподобаних товарів економлять простір і не перевантажують інтерфейс.

Система враховує вимоги до адаптивності, що забезпечується CSS3-медіазапитами та гнучкими макетами (Flexbox, Grid). На екранах із шириною до 768 пікселів макет перебудовується: меню згортається в бургер-іконку, картки товарів відображаються в одній колонці, а поле пошуку займає 80% ширини. На екранах до 480 пікселів шрифти зменшуються, а відступи оптимізуються для економії простору. Це забезпечує комфортне використання на смартфонах, планшетах і настільних комп’ютерах, що є критично важливим для сучасних онлайн-магазинів.

Продуктивність системи оптимізована через використання асинхронного завантаження JavaScript (атрибут defer), мінімізацію маніпуляцій із DOM і кешування стилів. Хоча в поточній реалізації є простір для покращення (наприклад, заміна innerHTML на createElement), система працює швидко навіть із великим каталогом. У майбутньому передбачається додавання підтримки прогресивного веб-додатка (PWA), що дозволить працювати офлайн і швидше завантажувати сторінку за допомогою сервіс-воркерів.

Система розроблена з урахуванням масштабованості. Додавання нових категорій чи підкатегорій до каталогу не потребує зміни основного коду, оскільки структура products є гнучкою. Модуль оформлення замовлення може бути інтегрований із платіжними системами (наприклад, LiqPay) чи API логістики для реальної обробки замовлень. Логіка JavaScript побудована модульно, що дозволяє додавати нові функції, такі як порівняння товарів чи відгуки користувачів, без значних змін у коді. Наприклад, для реалізації порівняння товарів достатньо створити функцію compareProducts(), яка генеруватиме таблицю характеристик.

Отже, структура та організація інформаційної системи забезпечують її функціональність, гнучкість і зручність для користувачів. Використання SPA, семантичної HTML5-розмітки, модульного підходу та передбачення для масштабування роблять систему сучасною та готовою до подальшого розвитку. Скриншот на рисунку 3.2 і фрагмент коду на рисунку 3.3 підтверджують практичну реалізацію описаної структури, демонструючи її візуальну та технічну складові.

3.2 Стилізація та адаптація інтерфейсу користувача

Стилізація та адаптація інтерфейсу інформаційної системи (ІС) онлайн-магазину комп’ютерної техніки відіграють ключову роль у забезпеченні зручності, привабливості та функціональності користувацького досвіду. Інтерфейс системи розроблено з використанням сучасних технологій CSS3, включаючи Flexbox, Grid, анімації, переходи, градієнти та медіазапити, що

дозволяють створювати гнучкі макети, які адаптуються до різних пристроїв і розмірів екранів. Основна мета стилізації — створення візуально гармонійного, інтуїтивно зрозумілого та доступного інтерфейсу, який відповідає принципам UX/UI і забезпечує комфортну взаємодію для всіх категорій користувачів, від новачків до досвідчених покупців. Адаптивність інтерфейсу гарантує його коректне відображення на настільних комп'ютерах, планшетах і смартфонах, що є критично важливим у сучасних умовах, коли значна частка користувачів відвідує онлайн-магазини з мобільних пристроїв. Зовнішній вигляд головної сторінки магазину представлено в додатку Г (рисунок Г.1), а сторінки товару — в додатку Г (рисунок Г.3).

Стилізація інтерфейсу базується на сучасних принципах веб-дизайну, які поєднують естетику, функціональність і продуктивність. CSS3 дозволяє створювати складні макети, динамічні ефекти та адаптивні компоненти без надмірного ускладнення коду чи зниження швидкості завантаження сторінки. Інтерфейс системи включає ключові елементи, такі як навігаційне меню (<nav>), поле пошуку (<input id="search">), фільтри (<aside id="filter">), картки товарів (.product-card), кошик ( ), список вподобаних товарів ( ) і форму оформлення замовлення (<form id="checkout-form">). Кожен елемент оформлено в єдиній стилістичній концепції, що забезпечує цілісність дизайну та гармонійне сприйняття.

Основні принципи стилізації включають:

— єдину кольорова палітра; використовується комбінація нейтральних кольорів (білий #FFFFFF, сірий #F5F5F5) та акцентних (градієнт синього #007bff до зеленого #00cc99), що відповідає тематиці технологічного магазину, наприклад, кнопки мають градієнтний фон (background: linear-gradient(90deg, #007bff, #00cc99)), який додає сучасного вигляду та привертає увагу;

— чітка типографіка; застосовуються шрифти без засічок, такі як Roboto або Open Sans, із різними вагами (font-weight: 400 для тексту, 700 для заголовків) і розмірами (16px для основного тексту, 20px для назв товарів); розмір шрифту

адаптується через одиниці rem для забезпечення читабельності на різних екранах;

— гнучкий макет; використання Flexbox (display: flex) для вирівнювання елементів у навігаційному меню та фільтрах, а також Grid (display: grid) для створення сітки карток товарів, наприклад, каталог відображає три колонки на настільних комп'ютерах (grid-template-columns: repeat(3, 1fr)) і одну колонку на мобільних пристроях через медіазапити.

— динамічні ефектия; плавні переходи (transition: background-color 0.3s ease) застосовуються до кнопок і посилань, а анімації (@keyframes fadeIn) — до спливаючих вікон кошика чи результатів пошуку; ефект наведення на картки товарів (transform: scale(1.05)) додає інтерактивності;

— доступність (A11y); контрастні кольори (співвідношення контрасту не менше 4.5:1), атрибути aria-label (наприклад, aria-label="Пошук товарів" для поля пошуку) і достатні розміри елементів (кнопки не менше 44x44 пікселі) забезпечують зручність для користувачів із обмеженими можливостями.

Адаптивність інтерфейсу реалізована через CSS-медіазапити (@media), які змінюють стилі залежно від ширини екрана. На екранах із шириною до 768 пікселів навігаційне меню трансформується в бургер-меню (display: none для стандартного меню, display: block для бургер-іконки), поле пошуку займає 80% ширини (width: 80%), а фільтри відображаються вертикально (flex-direction: column). На екранах до 480 пікселів розмір шрифтів зменшується до 14px, відступи оптимізуються (padding: 8px), а кнопки збільшуються для зручності натискання пальцем. Це забезпечує коректне відображення та зручність використання на всіх типах пристроїв, що відповідає сучасним стандартам веб-дизайну.

На рис. 3.4 зображено схему стилізації та адаптації інтерфейсу інформаційної системи онлайн-магазину комп'ютерної техніки. Схема ілюструє, як CSS-технології застосовуються до основних елементів інтерфейсу та забезпечують адаптивність. У верхній частині рисунка показано макет для настільного комп'ютера: горизонтальне навігаційне меню, сітка карток товарів

(3 колонки, display: grid) і бічна панель фільтрів. У нижній частині зображено мобільний макет (ширина до 768 пікселів), де меню згорнуте в бургер-іконку, картки відображаються в одній колонці, а фільтри розташовані вертикально. Стрілки між макетами позначають медіазапити (@media), які керують адаптацією. Окремі блоки виділяють стилі для кнопок (градієнт, border-radius: 12px), карток (box-shadow, ефект наведення) і спливаючих вікон (анімація fadeIn). Схема також включає кольорову палітру (нейтральні та акцентні кольори) і типографіку (шрифти, розміри). Рисунок виконаний у мінімалістичному стилі з чіткими підписами для легкого сприйняття.

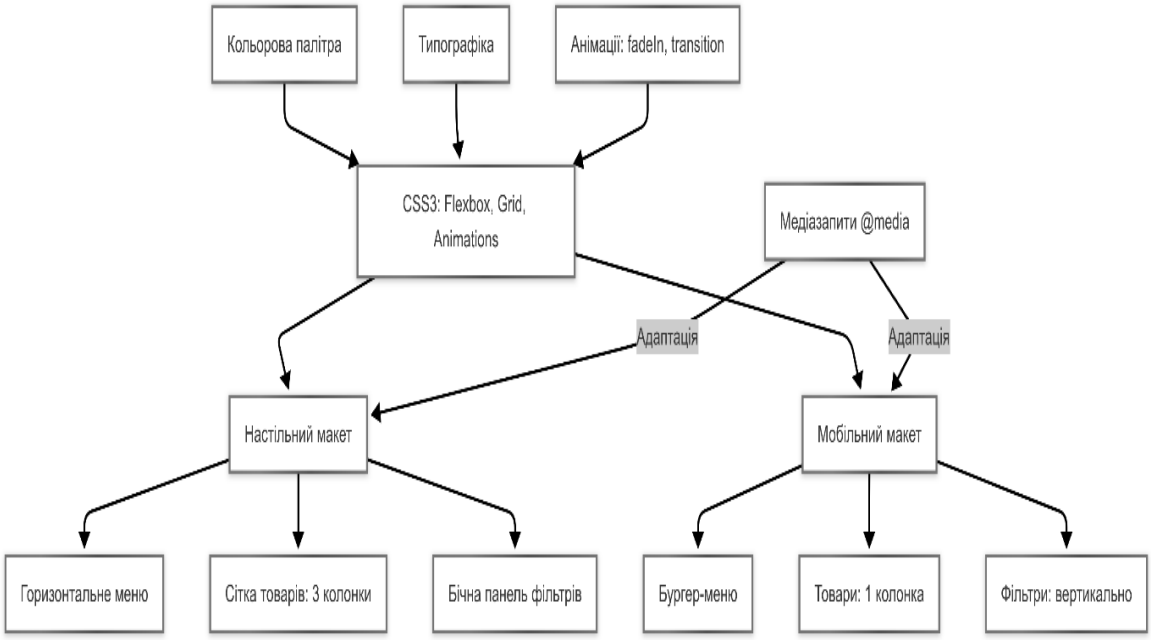

Рисунок 3.4 – Схема стилізації та адаптації інтерфейсу інформаційної системи

Для наочної демонстрації адаптивного дизайну на рис. 3.5 зображено скріншот сторінки каталогу товарів на мобільному пристрої з шириною екрана 375 пікселів (наприклад, iPhone 12). Скріншот показує, як медіазапити трансформують інтерфейс: навігаційне меню замінено бургер-іконкою, поле пошуку займає 80% ширини екрана, а картки товарів відображаються в одній колонці (grid-template-columns: 1fr). Фільтри розташовані вертикально під каталогом, що економить простір. Кнопки «Додати до кошика» і «Вподобати»

мають збільшений розмір (48x48 пікселів) і градієнтний фон, що полегшує взаємодію на сенсорних екранах. Спливаюче вікно кошика ( ) відображається в компактному вигляді з анімацією fadeIn. Скріншот ілюструє адаптивність, зручність і відповідність принципам UX/UI, описаним у цьому розділі.

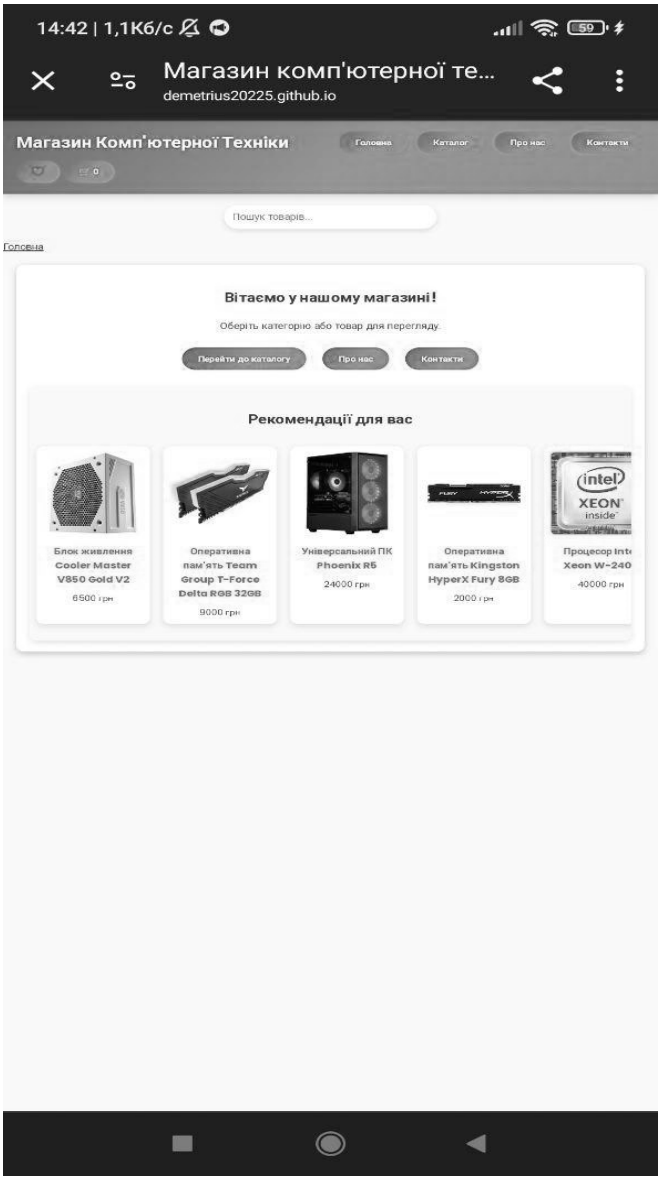

Рисунок 3.5 – Скріншот каталогу товарів на мобільному пристрої

Для підтвердження реалізації стилізації було вставлено нище фрагмент CSS-коду, що відповідає за оформлення карток товарів (.product-card) і кнопок (.button). Код включає використання Flexbox (display: flex, flex-direction: column) для вирівнювання вмісту картки, градієнтного фону для кнопок (linear-gradient), тіні (box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2)), закруглених кутів (border-radius:

12px) і ефекту наведення (transform: scale(1.05)). Анімація для спливаючих вікон визначена через @keyframes fadeIn, що забезпечує плавну появу. Фрагмент також містить медіазапит (@media (max-width: 768px)), який змінює сітку карток на одну колонку. Цей код демонструє сучасні CSS-технології та їхню роль у створенні привабливого й адаптивного інтерфейсу.

```
.product-card {
    background: white;
    border-radius: 12px;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    transition: all 0.3s ease;
    position: relative;
}
.product-card:hover {
    transform: translateY(-5px);
    box-shadow: 0px 6px 15px rgba(0, 0, 0, 0.2);
}
.product-image {
    width: 250px;
    height: 300px;
    object-fit: cover;
    border-radius: 12px;
    margin-bottom: 10px;
}
```

Стилізація системи враховує принципи UX/UI, які забезпечують інтуїтивність і зручність. Для навігації використовуються навігаційні елементи типу «Хлібні крихти» (<nav id="breadcrumbs">), що відображають шлях користувача (наприклад, «Головна > Каталог > Ноутбуки»), полегшуючи орієнтацію.

Для забезпечення високої інтерактивності використані кнопки з чітким дизайном, з градієнтним фоном, закругленими кутами і тінню, що робить їх помітними. Наприклад, кнопка «Додати до кошика» змінює колір при натисканні (:active { background: linear-gradient(90deg, #005bb5, #009973) }).

Кошик і список вподобаних товарів відображаються у фіксованих вікнах (position: fixed), що економить простір і дозволяє швидко переглядати вміст.

Поле пошуку розширюється при фокусі (transition: width 0.3s ease), а результати з’являються у спливаючому вікні з анімацією.

Доступність (A11y) забезпечується через використання контрастних кольорів (наприклад, білий текст #FFFFFF на синьому градієнті), атрибутів aria-label і достатніх розмірів шрифтів (не менше 16px). Наприклад, кнопка закриття кошика має атрибут aria-label="Закрити кошик", а поле пошуку — aria-label="Пошук товарів". Клавіатурна навігація підтримується через коректний порядок фокусу (tabindex), що робить систему зручною для користувачів без миші.

Продуктивність стилізації оптимізована шляхом:

— мінімізації CSS, забезпечена спільним класами (наприклад, .popup для спливаючих вікон) зменшують дублювання коду;

— використання для анімацій transform і opacity замість width чи height, що знижує навантаження на браузер;

— зображення товарів мають атрибут loading="lazy", що прискорює завантаження сторінки; у перспективі CSS може бути оптимізований через препроцесори (SASS) або методологію BEM для підвищення читабельності та масштабованості.

Адаптивність підтримується гнучкими одиницями вимірювання (rem, vw, %). Наприклад, ширина карток товарів задається в vw для масштабування, а відступи — у rem для пропорційності. Медіазапити дозволяють адаптувати макет до різних роздільних здатностей, включаючи Retina-дисплеї, де застосовується min-resolution: 2dppx для чіткіших зображень.

Система передбачає можливість розширення стилізації. Наприклад, додавання темного режиму може бути реалізовано через CSS-змінні (:root { --primary-color: #007bff; }) і перемикач теми. Локалізація для мов із письмом справа наліво (наприклад, арабської) підтримується через direction: rtl. Мікроанімації, такі як пульсація кнопок чи розгортання фільтрів, додають зворотний зв’язок і підвищують залученість користувача.

У майбутньому стилізацію можна вдосконалити шляхом інтеграції бібліотек, таких як Tailwind CSS, для швидшого створення компонентів, або використання CSS-in-JS для динамічної стилізації у фреймворках, таких як

React. Автоматизоване тестування стилів через Stylelint гарантуватиме відповідність стандартам і відсутність помилок. Скриншот на рис. 3.5 і фрагмент коду на рис. 3.6 підтверджують практичну реалізацію адаптивного дизайну та стилізації, демонструючи їхню візуальну та технічну складові.

Отже, стилізація та адаптація інтерфейсу забезпечують його привабливість, зручність і доступність. Використання CSS3, Flexbox, Grid, анімацій і медіазапитів створює гнучкий і сучасний дизайн, який відповідає вимогам різних пристроїв і сценаріїв використання. Система готова до масштабування та подальшого вдосконалення, що робить її перспективною для комерційного використання.

3.3 Реалізація логіки взаємодії за допомогою JavaScript

Реалізація логіки взаємодії інформаційної системи (ІС) онлайн-магазину комп’ютерної техніки є центральним компонентом, що забезпечує динамічну роботу веб-додатка, інтерактивність і безперебійну взаємодію користувача з інтерфейсом. Логіка побудована на основі JavaScript (ES6+), який відповідає за обробку подій, управління станом системи, рендеринг вмісту, оновлення DOM і збереження даних між сеансами. Використання сучасних можливостей JavaScript, таких як стрілкові функції, методи масивів (map, filter), асинхронне програмування та модульна структура, дозволяє створювати ефективний і гнучкий код, що відповідає стандартам веб-розробки. Основна мета цього розділу — описати механізми реалізації ключових функцій системи, таких як відображення каталогу, пошук, фільтрація, управління кошиком, збереження вподобаних товарів і оформлення замовлення, а також продемонструвати їхню практичну реалізацію через код і візуальні елементи.

Логіка взаємодії системи базується на концепції Single Page Application (SPA), де всі дії користувача (переходи між розділами, додавання товарів до кошика, застосування фільтрів) обробляються без перезавантаження сторінки. Це досягається через динамічне оновлення вмісту основного контейнера <main id="content"> за допомогою JavaScript-функцій, які реагують на події, такі як

кліки, введення тексту чи зміна стану форми. Наприклад, функція navigate() керує переходами між розділами (головна сторінка, каталог, кошик, оформлення замовлення), викликаючи відповідні функції рендерингу, такі як renderCatalog() чи renderCart(). Такий підхід забезпечує швидкість роботи, мінімізує затримки та створює плавний користувацький досвід.

Ключові функції системи, реалізовані через JavaScript, включають такі функції.

Функція renderCatalog(category, subcategory) для відображення каталогу товарів. Функція генерує HTML-розмітку для карток товарів на основі даних із об’єкта products, використовуючи метод map для ітерації та шаблонні рядки для створення DOM-елементів. Каталог оновлюється при виборі категорії чи застосуванні фільтрів.

Функціональність системи реалізує ключові етапи взаємодії користувача з онлайн-магазином. Пошук товарів здійснюється за допомогою функції searchProducts(query), яка фільтрує масив products відповідно до введених ключових слів. Результати пошуку виводяться в режимі реального часу у спливаючому вікні, що оновлюється через обробник події input, прив’язаний до відповідного поля.

Фільтрація товарів реалізована функцією applyFilters(filters), яка застосовує вибрані користувачем критерії, такі як ціна, бренд або рік випуску. Вона використовує метод filter для формування нового підмасиву products і викликає функцію renderCatalog() для оновлення відображення на сторінці.

Управління кошиком включає додавання, видалення товарів, а також зміну їх кількості. Для цього використовуються функції addToCart(item), removeFromCart(id) та updateCartQuantity(id, quantity). Загальна вартість обчислюється методом reduce, а інтерфейс оновлюється через функцію updateCart().

Список вподобаних товарів формується за допомогою функції toggleFavorite(id), яка додає або видаляє товар із масиву favorites. Стан

зберігається в localStorage за допомогою saveFavorites(), а відображення реалізується через renderFavorites().

Процес оформлення замовлення обробляється подією submit, прив’язаною до форми з ідентифікатором checkout-form. При цьому створюється об’єкт order, генерується унікальний номер замовлення, а вміст кошика очищується через виклик функції clearCart().

На рис. 3.7 зображено схему логіки взаємодії інформаційної системи онлайн-магазину комп’ютерної техніки. Схема ілюструє, як JavaScript-функції взаємодіють із DOM, даними та подіями користувача. У центрі розташовано блок «JavaScript-логіка», від якого відходять стрілки до основних функцій: navigate(), renderCatalog(), searchProducts(), applyFilters(), addToCart(), toggleFavorite(), renderCart() і обробника submit. Кожен блок функції пов’язаний із відповідним елементом інтерфейсу: <main id="content">, <input id="search">, <aside id="filter">, , , <form id="checkout-form">. Стрілки показують потік даних, наприклад, як products фільтруються в searchProducts() і відображаються через renderCatalog(). У нижній частині схеми зображено localStorage для збереження кошика та вподобаних товарів. Окремі блоки виділяють події (click, input, submit), які запускають функції. Схема виконана в мінімалістичному стилі з чіткими підписами та стрілками для легкого сприйняття.

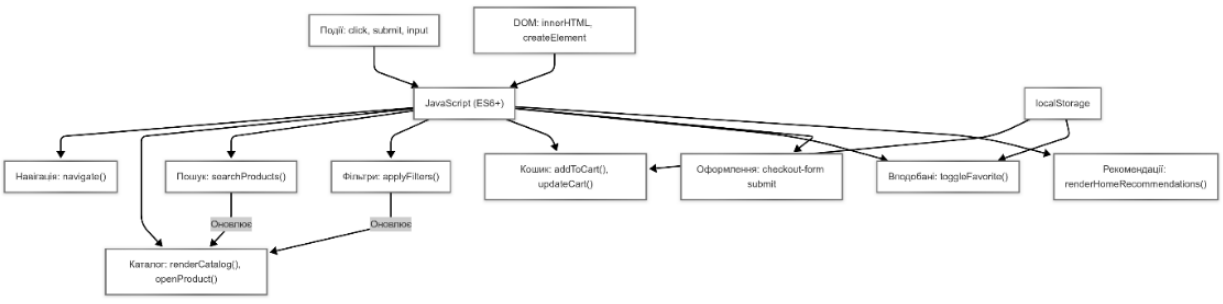

Рисунок 3.7 – Схема логіки взаємодії інформаційної системи

Для демонстрації практичної реалізації логіки в додатку Г (рисунок Г.4) зображено скріншот сторінки кошика, яка відображається при виборі відповідного розділу через navigate('cart'). Скріншот показує компактне

спливаюче вікно ( ) з переліком доданих товарів, їхньою кількістю, ціною за одиницю та загальною сумою. Кожен товар має кнопки для зміни кількості (+/–) і видалення (іконка кошика). У нижній частині вікна відображається загальна сума та кнопка «Оформити замовлення», яка веде до форми замовлення. Кошик оформлено з градієнтним фоном і анімацією появи (fadeIn), що відповідає стилізації, описаній у розділі 3.2. Скріншот ілюструє динамічне оновлення вмісту, інтерактивність і зручність управління кошиком, що забезпечуються JavaScript-логікою.

Для підтвердження реалізації логіки взаємодії нище зображено фрагмент JavaScript-коду функції addToCart() та пов’язаної функції updateCart(). Код демонструє, як товар додається до масиву cart, оновлюється кількість (quantity), обчислюється загальна сума через reduce і оновлюється інтерфейс через маніпуляції з DOM. Функція addToCart() перевіряє, чи товар уже є в кошику, і відповідно збільшує кількість або додає новий запис. Функція updateCart() генерує HTML-розмітку для відображення кошика, включаючи назви товарів, ціни, кількість і кнопки керування. Цей фрагмент підкреслює модульність, ефективність і динамічність JavaScript-логіки, описаної в розділі.

```
        function addToCart(category, subCategory, index) {
            let продукт;
            if (subCategory) {
продукт = products["Каталог"][category].товари[subCategory].товари[index];
            } else {
                продукт = products["Каталог"][category].товари[index];
            }
            if (!продукт) {
                alert("Товар не знайдено!");
                return;
            }
            продукт.category = category;
            продукт.subCategory = subCategory || '';
            let existingProduct = cart.find(item => item.назва === продукт.назва);
            if (existingProduct) {
                existingProduct.кількість++;
            } else {
                cart.push({ ...продукт, кількість: 1 });
            }
            updateCart();
        }
```

```
function updateCart() {
    const cartElement = document.getElementById("cart");
    const cartItems = document.getElementById("cart-items");
    const cartTotal = document.getElementById("cart-total");
    const cartIcon = document.getElementById("cart-count");
    cartItems.innerHTML = "";
    let total = 0;
    if (cart.length === 0) {
  cartElement.style.display = "none"; // Приховати кошик, якщо він порожній
        cartIcon.textContent = "0"; // Оновити індикатор кошика
        cartTotal.textContent = "Загальна сума: 0 грн"; // Оновити загальну суму
        return;
    }
    cart.forEach((item, index) => {
        total += item.ціна * item.кількість;
        cartItems.innerHTML += `
            <li>
<img src="${item.зображення}" alt="${item.назва}" style="width: 50px; height: 50px;">
                <span>${item.назва} - ${item.ціна} грн</span>
                <button onclick="changeQuantity(${index}, -1)">➖</button>
                ${item.кількість}
                <button onclick="changeQuantity(${index}, 1)">➕</button>
                <button onclick="removeFromCart(${index})">❌</button>
            </li>`;
    });
cartTotal.textContent = `Загальна сума: ${total} грн`; // Оновити загальну суму
    cartIcon.textContent = cart.length; // Оновити індикатор кошика
}
```

Логіка взаємодії в системі побудована з урахуванням вимог до продуктивності, зручності та масштабованості. Особливу увагу приділено мінімізації маніпуляцій із DOM — оновлення вмісту відбувається лише для змінених елементів, наприклад, елементу #cart-count, який відображає кількість товарів у кошику. Такий підхід дозволяє уникнути надмірного перерендерення сторінки (reflow), що позитивно впливає на швидкодію.

Дані каталогу (products) зберігаються у пам’яті браузера, тоді як кошик та список вподобаних товарів — у localStorage, що зменшує обчислювальне навантаження при повторному доступі до інформації. Для оптимізації пошуку реалізовано дебонсинг: подія input, пов’язана з полем пошуку, викликає функцію searchProducts() із затримкою (наприклад, 300 мс), що дозволяє зменшити кількість викликів при швидкому введенні. Також система передбачає

використання асинхронних функцій (async/await) для потенційних API-запитів у майбутньому, аби не блокувати головний потік виконання.

З точки зору UX/UI логіка взаємодії орієнтована на зручність користувача. Додавання товару до кошика супроводжується візуальним зворотним зв’язком у вигляді сповіщення або спливаючого вікна, а зміни кількості товарів у кошику миттєво відображаються в інтерфейсі. Кнопки керування, такі як зміна кількості чи видалення товару, мають зрозумілі іконки та анімовані ефекти натискання, що робить взаємодію інтуїтивною. Основні функції — такі як renderCatalog() і searchProducts() — оптимізовані таким чином, щоб забезпечити швидке виконання навіть при великому обсязі масиву products.

Розробка також враховує принципи доступності (A11y). Для динамічних елементів, таких як результати пошуку, використано атрибут aria-live, що дозволяє програмам зчитування екрана повідомляти користувачам про зміни. Клавіатурна навігація реалізована за допомогою атрибутів tabindex та обробників подій keydown, а інтерактивні елементи мають чіткі підписи, наприклад aria-label="Збільшити кількість" для кнопки "+", що забезпечує повну підтримку користувачів з обмеженими можливостями.

Уся логіка системи створена з урахуванням подальшої масштабованості. Зокрема, функція renderCatalog() може бути легко адаптована для роботи з API, замінивши внутрішній масив products на динамічні дані з сервера. Планується реалізація функціоналу порівняння товарів через окрему функцію compareProducts(), яка формуватиме таблиці характеристик. Крім того, інтеграція з платіжними системами, такими як LiqPay або Stripe, може бути реалізована через асинхронні обробники у функції оформлення замовлення.

Для подальшого вдосконалення логіки взаємодії передбачається впровадження сучасних фреймворків, таких як React, що дозволить ефективно керувати станом додатку та забезпечить повторне використання компонентів. Також можливе використання бібліотек, на кшталт Lodash, для оптимізації роботи з масивами, а також впровадження модульного автоматизованого

тестування за допомогою Jest для перевірки ключових функцій, таких як addToCart() чи searchProducts().

Додаткову оптимізацію продуктивності можна досягти шляхом заміни innerHTML на createElement, що зменшує кількість перерендерень. Для обробки великих обсягів даних можна використати Web Workers, а для кешування результатів фільтрації чи пошуку — структури Map або Set. Візуальне підтвердження реалізації цієї логіки наведено на рисунках 3.8 і 3.9, які демонструють фрагменти коду та інтерфейс системи.

Таким чином, реалізована логіка забезпечує швидку, зручну та адаптивну роботу інформаційної системи, відповідає сучасним стандартам веб-розробки та легко масштабовується відповідно до зростаючих потреб онлайн-магазину.

3.4 Оформлення замовлення та валідація введених даних

Оформлення замовлення та валідація є завершальними етапами взаємодії користувача з інформаційною системою (ІС) онлайн-магазину комп’ютерної техніки, забезпечуючи створення замовлення на основі товарів у кошику та перевірку коректності введених даних. Цей процес реалізовано через інтерактивну форму (<form id="checkout-form">), яка дозволяє користувачу ввести особисті дані, обрати спосіб доставки та оплати, а також отримати підтвердження замовлення. Валідація даних здійснюється на клієнтській стороні за допомогою HTML5 і JavaScript, що гарантує надійність, безпеку та зручність. Використання сучасних веб-технологій, таких як регулярні вирази, атрибути HTML5 (required, pattern) і динамічні повідомлення про помилки, забезпечує ефективну обробку даних і відповідність стандартам UX/UI. Основна мета цього розділу — описати механізми реалізації форми замовлення, валідації даних і підтвердження замовлення, а також продемонструвати їхню практичну реалізацію через візуальні елементи та код.

Процес оформлення замовлення починається з переходу користувача до форми замовлення через кнопку «Оформити замовлення» у кошику ( ), що викликає функцію navigate('checkout'). Форма містить поля для

введення особистих даних (ПІБ, телефон, email), випадаючі списки для вибору способу доставки («Нова Пошта», «Укрпошта», «Самовивіз») і способу оплати («Карткою», «Готівкою при отриманні»), а також кнопку подачі форми. Після заповнення форми та проходження валідації створюється об’єкт order, який містить дані про товари, користувача, спосіб доставки, оплати та унікальний номер замовлення. Далі кошик очищається через функцію clearCart(), а користувачу відображається сповіщення про успішне замовлення з номером.

Валідація даних реалізована на двох рівнях: HTML5 та JavaScript.

При HTML5-валідації використовуються атрибути required, type="email", type="tel", pattern для перевірки формату введених даних безпосередньо в браузері. Наприклад, поле email має атрибут type="email", який перевіряє наявність символу @ і домену, а поле телефону — pattern="\+380\d{9}" для формату «+380XXXXXXXXX».

JavaScript-валідація реалізується функцією validateForm(), що перевіряє коректність даних за допомогою регулярних виразів і додаткових умов. Наприклад, ПІБ перевіряється на наявність лише літер і пробілів (/^[А-Яа-яA-Za-z\s]+$/), а email — на відповідність формату (/^[^\s@]+@[^\s@]+\.[^\s@]+$/). Якщо дані некоректні, користувачу відображаються повідомлення про помилки (наприклад, «Введіть коректний номер телефону»), які генеруються динамічно через innerText.

На рис. 3.10 зображено схему оформлення замовлення та валідації інформаційної системи онлайн-магазину комп’ютерної техніки. Схема ілюструє послідовність дій і взаємодію компонентів. У центрі розташовано блок «Форма замовлення ()», від якого відходять стрілки до етапів: «Введення даних», «HTML5-валідація», «JavaScript-валідація», «Створення об’єкта order», «Очищення кошика» і «Сповіщення про замовлення». Блок «Введення даних» включає поля (ПІБ, телефон, email, доставка, оплата), а блоки валідації показують атрибути (required, pattern) і функцію validateForm(). Стрілки вказують потік даних: від форми до валідації, створення об’єкта order і сповіщення. У нижній частині схеми зображено localStorage для збереження

кошика до моменту оформлення. Окремі блоки виділяють повідомлення про помилки та підтвердження. Схема виконана в мінімалістичному стилі з чіткими підписами та стрілками для легкого сприйняття.

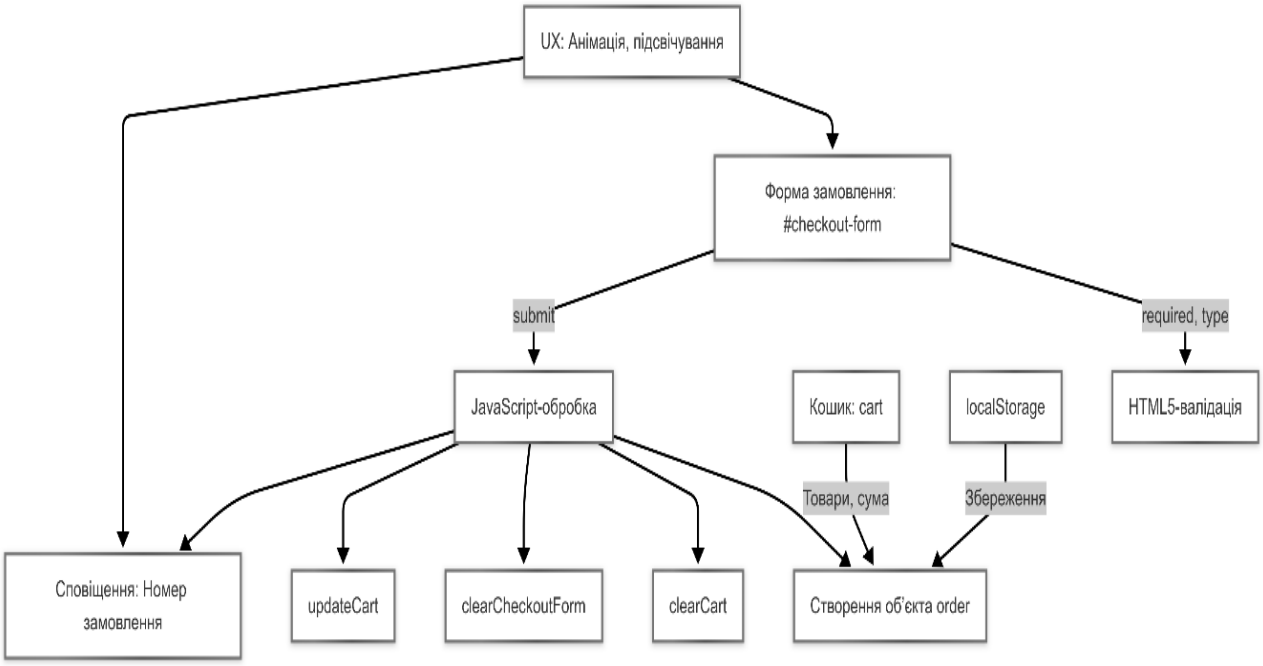

Рисунок 3.10 – Схема оформлення замовлення та валідації інформаційної системи

Для демонстрації практичної реалізації процесу оформлення замовлення на рис. 3.11 зображено скріншот форми замовлення, розташованої в <main id="content">. Форма включає текстові поля для ПІБ, телефону, email, випадаючі списки для способу доставки («Нова Пошта», «Укрпошта», «Самовивіз») і оплати («Карткою», «Готівкою»), а також кнопку «Оформити замовлення» з градієнтним фоном (linear-gradient(90deg, #007bff, #00cc99)) і анімацією натискання. Скріншот показує повідомлення про помилку для поля email («Введіть коректний email»), виділене червоною рамкою (border: 2px solid red), що ілюструє роботу HTML5- і JavaScript-валідації. Поля мають підписи (<label>) і атрибути aria-label (наприклад, aria-label="Введіть ваш email"), що забезпечують доступність. Скриншот демонструє зручність, інтуїтивність і відповідність принципам UX/UI, описаним у розділі 3.2.

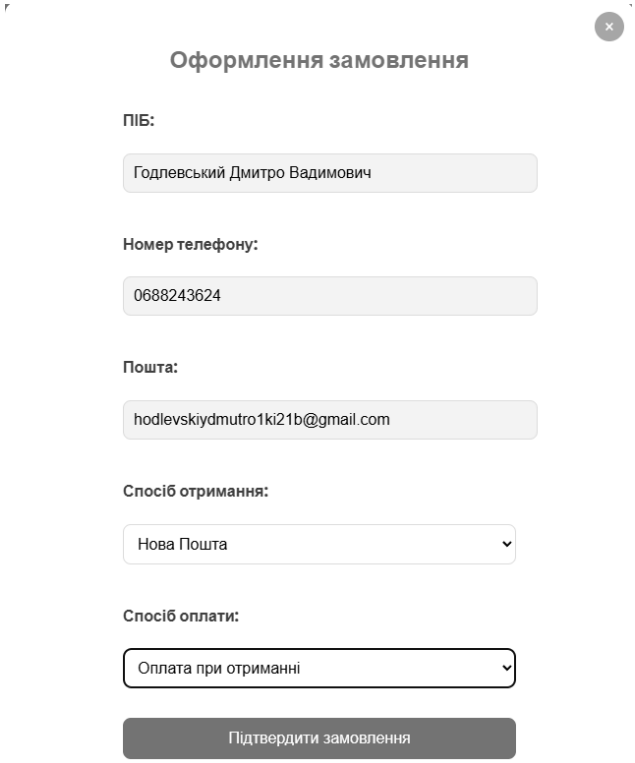

Рисунок 3.11 – Скріншот форми оформлення замовлення

Нище буде наведено фрагмент JavaScript-коду обробника події submit для форми #checkout-form, який реалізує логіку оформлення замовлення. Код перевіряє наявність товарів у кошику, отримує дані форми через об’єкт FormData, створює об’єкт order із контактними даними користувача, списком товарів і загальною сумою. Генерація випадкового номера замовлення здійснюється через Math.floor(Math.random() * 10000). Після обробки відображається сповіщення (alert) із номером замовлення та сумою, форма закривається через toggleCheckout(), кошик очищається через clearCart(), а сторінка перенаправляється на вкладку кошика за допомогою navigate("full-cart"). Цей фрагмент ілюструє простоту, ефективність і зручність обробки замовлення, покладаючись на HTML5-атрибути для базової валідації.

```
document.getElementById("checkout-form").addEventListener("submit", function (e) {
    e.preventDefault();
    if (cart.length === 0) {
        alert("Кошик порожній! Додайте товари перед оформленням замовлення.");
        return;
    }
```

```
    const formData = new FormData(this);
    const order = {
      fullName: formData.get("full-name"),
      phone: formData.get("phone"),
      email: formData.get("email"),
      delivery: formData.get("delivery"),
      payment: formData.get("payment"),
      items: cart,
      total: cart.reduce((sum, item) => sum + item.ціна * item.кількість, 0)
    };
    const orderNumber = Math.floor(Math.random() * 10000);
    alert(`Дякуємо за ваше замовлення!
Номер замовлення: #${orderNumber}
Сума: ${order.total} грн
Ми зв'яжемося з вами найближчим часом.`);
    toggleCheckout();
    clearCart();
    clearCheckoutForm();
    navigate("full-cart");
function clearCheckoutForm() {
    const form = document.getElementById("checkout-form");
    form.reset();
}
```

Форма замовлення розроблена з урахуванням сучасних UX/UI-принципів, що забезпечують зручність, інтуїтивність та ефективність користувацької взаємодії. Усі поля мають чіткі підписи, а випадаючі списки згруповані логічно, що полегшує вибір необхідних опцій. Для підвищення зручності заповнення форма містить лише базові поля — такі як ПІБ, номер телефону, email, спосіб доставки та оплати. Це дозволяє мінімізувати час взаємодії з формою та підвищити ймовірність завершення покупки. Помилки валідації відображаються в реальному часі: відповідні поля підсвічуються червоним кольором, а поруч виводяться текстові повідомлення про виявлені проблеми. Візуально форма оформлена з урахуванням єдиного стилю: кнопка надсилання має градієнтне тло, закруглені кути (border-radius: 12px) та анімований ефект натискання, що відповідає загальній стилізації, описаній у розділі 3.2.

Питання доступності (A11y) також було враховано. Для кожного поля форми використовуються елементи <label> або атрибути aria-label, які роблять форму зручною для користувачів із вадами зору. Для повідомлень про помилки додано атрибути aria-describedby, що забезпечує зв’язок між полем і відповідним

текстом. Клавіатурна навігація реалізована через правильне використання tabindex та обробників подій keydown, зокрема для випадаючих списків. Повідомлення про помилки мають контрастне поєднання кольорів (наприклад, червоний #FF0000 на білому тлі), що відповідає рекомендованому співвідношенню контрасту 4.5:1 згідно з WCAG.

Форма також враховує вимоги безпеки. Зокрема, передбачено очищення введених даних за допомогою функції sanitizeInput(), яка видаляє HTML-теги й скрипти, запобігаючи XSS-атакам. Крім того, для перевірки правильності введення реалізовано перевірку формату за допомогою регулярних виразів: для телефону, email та ПІБ встановлено відповідні патерни, а також обмеження на довжину введення (наприклад, maxlength="12" для номера телефону, або pattern="[А-Яа-яA-Za-z\s]+" для ПІБ, що запобігає введенню цифр чи спеціальних символів).

Продуктивність форми оптимізовано кількома способами. Використовується вбудована HTML5-валідація, яка дозволяє зменшити обсяг JavaScript-обробки. Крім того, реалізовано дебонсинг під час введення даних у поля: перевірка викликається з затримкою (наприклад, 300 мс), що дозволяє уникнути надмірного навантаження на браузер. DOM-маніпуляції мінімізовано: повідомлення про помилки оновлюються лише для тих полів, у яких відбулися зміни.

Процес оформлення замовлення спроектовано з можливістю подальшого масштабування. Зокрема, форма може бути легко інтегрована з платіжними системами, такими як LiqPay або Stripe, за допомогою API, додавши асинхронний обробник до події submit. За потреби також можливе додавання нових полів (наприклад, для адреси доставки або коментарів), що вимагатиме лише незначного розширення функції validateForm() та структури об’єкта order. У майбутньому можливе підключення CRM-систем, які збиратимуть і зберігатимуть замовлення через запити типу fetch.

Подальше вдосконалення процесу може відбуватися через впровадження серверної валідації як додаткового рівня захисту, використання функції

автозаповнення браузера (через атрибути autocomplete="name", autocomplete="tel"), або інтеграцію з бібліотеками типу Formik для зручного управління формами у React-проєктах. Скриншот форми на рисунку 3.11 та фрагмент JavaScript-коду на рисунку 3.12 ілюструють технічну реалізацію механізму оформлення замовлення.

Використання HTML5-атрибутів (required, type="email", type="tel") у поєднанні з обробником submit дозволяє забезпечити як зручність користування, так і надійність обробки даних. Такий підхід гарантує відповідність форми сучасним стандартам веб-розробки, а також відкриває можливості для подальшого розширення функціональності системи.

3.5 Реалізація персоналізованого функціоналу: рекомендації, вподобані товари та збереження стану

Персоналізація підвищує зручність e-commerce платформ, адаптуючи вміст до вподобань користувачів. У розробленій інформаційній системі (ІС) онлайн-магазину комп’ютерної техніки реалізовані рекомендаційна система, список вподобаних товарів, збереження стану кошика та фільтрів, а також інтерактивні елементи інтерфейсу. Ці функції забезпечують інтуїтивний користувацький досвід. Реалізація базується на JavaScript (ES6+), HTML5, CSS3 та localStorage, використовуючи клієнтську обробку для швидких взаємодій. Цей розділ описує технічну реалізацію, аспекти користувацького досвіду (UX), продуктивність і перспективи вдосконалення, підкріплюючи пояснення фрагментами коду та візуальними ілюстраціями.

Блок рекомендацій відображається на головній сторінці в контейнері під заголовком «Рекомендації для вас» та на сторінці товару в блоці . Функція renderHomeRecommendations() випадково відбирає до десяти товарів із каталогу (products), включаючи категорії «Зібрані ПК», «Ноутбуки» та підкатегорії «Комплектуючі» (наприклад, «Відеокарти»). Картки містять зображення, назву, ціну, стилізовані через CSS (recommendation-item) із адаптивним дизайном

(object-fit: contain). На сторінці товару openProduct() формує рекомендації з тієї ж категорії, виключаючи поточний товар, із обмеженням до чотирьох позицій. Картки дозволяють перейти до товару через onclick="openProduct()". На рис. 3.13 зображено блок рекомендацій на головній сторінці з картками товарів.

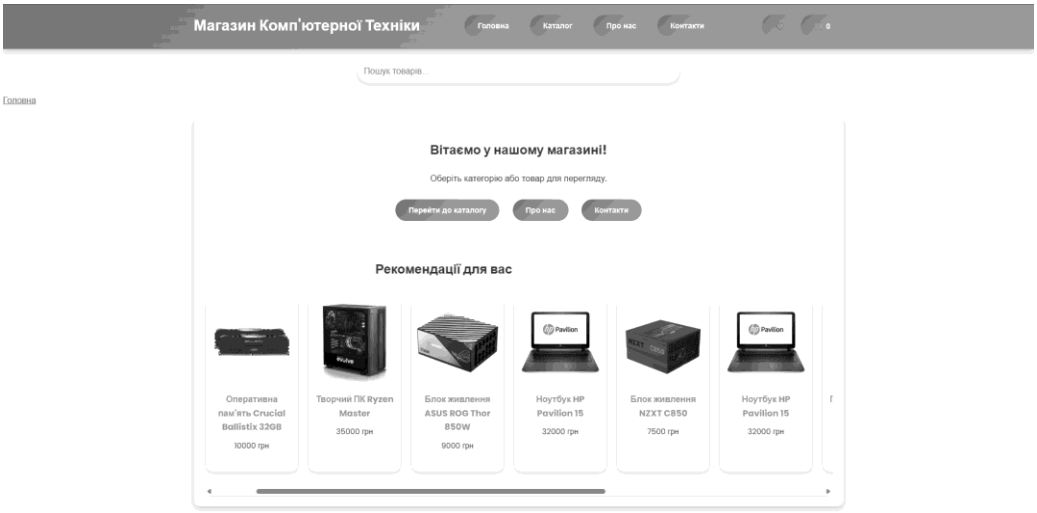

Рисунок 3.13 – Блок рекомендацій на головній

```
function renderHomeRecommendations() {
  const recommendationsContainer = document.getElementById("home-recommendations");
  if (!recommendationsContainer) return;
  const allProducts = [];
  for (let category in products["Каталог"]) {
    const categoryData = products["Каталог"][category];
    if (Array.isArray(categoryData.товари)) {
      categoryData.товари.forEach((product, index) => {
        allProducts.push({ ...product, category, index });
      });
    } else if (typeof categoryData.товари === "object") {
      for (let subCategory in categoryData.товари) {
        categoryData.товари[subCategory].товари.forEach((product, index) => {
          allProducts.push({ ...product, category, subCategory, index });
        });
      }
    }
  }
  const randomRecommendations = [];
  for (let i = 0; i < 10; i++) {
    const randomIndex = Math.floor(Math.random() * allProducts.length);
    randomRecommendations.push(allProducts[randomIndex]);
  }
  recommendationsContainer.innerHTML = randomRecommendations.map(product => `
```

```
            <div class="recommendation-item" onclick="openProduct('${product.category}',
'${product.subCategory || ''}', ${product.index})">
                <img src="${product.зображення}" alt="${product.назва}">
                <h3>${product.назва}</h3>
                <p>${product.ціна} грн</p>
            </div>
        `).join('');
    }
```

Список вподобаних товарів дозволяє зберігати продукти через масив favorites, який синхронізується з localStorage. Функція toggleFavorite(category, subCategory, index) додає або видаляє товар, перевіряючи його за назвою. Наприклад, «Геймерський ПК Titan X» додається з полями category, subCategory, назва, ціна. Збереження виконується через localStorage.setItem('favorites', JSON.stringify(favorites)), а завантаження — через JSON.parse(localStorage.getItem('favorites')) || []. Інтерфейс відображається в <aside id="favorites">, яке відкривається кнопкою <button id="favorites-toggle"> (іконка ❤️). Позиції включають зображення, назву, ціну та кнопки для переходу (openProductFromFavorites) або видалення (toggleFavorite). Можна побачити в додатку Г (рисунок Г.5) модальне вікно зі списком вподобаних товарів.

```
function toggleFavorite(category, subCategory, index) {
    let product = subCategory
        ? products["Каталог"][category].товари[subCategory].товари[index]
        : products["Каталог"][category].товари[index];
    if (!product) {
        alert("Товар не знайдено!");
        return;
    }
    product.category = category;
    product.subCategory = subCategory || '';
    const favoriteIndex = favorites.findIndex(fav => fav.назва === product.назва);
    if (favoriteIndex === -1) {
        favorites.push(product);
    } else {
        favorites.splice(favoriteIndex, 1);
    }
    updateFavorites();
    updateFavoriteButtons();
}
function updateFavorites() {
    const favoritesItems = document.getElementById("favorites-items");
    favoritesItems.innerHTML = favorites.map((item, index) => `
```

```
            <li>
              <img src="${item.зображення}" alt="${item.назва}" style="width: 50px; height: 50px;">
              <span>${item.назва} - ${item.ціна} грн</span>
              <button onclick="openProductFromFavorites(${index})">Перейти</button>
              <button onclick="toggleFavorite('${item.category}', '${item.subCategory}', ${item.index})">❌</button>
            </li>
          `).join('') || '<p>Список вподобаних порожній</p>';
          localStorage.setItem('favorites', JSON.stringify(favorites));
        }
```

Збереження стану забезпечує безперервність для кошика, вподобаних товарів і фільтрів. Кошик (cart) — масив із полями назва, ціна, кількість — зберігається в localStorage після змін (addToCart, changeQuantity, removeFromCart). Функція updateCart() оновлює <aside id="cart"> і записує дані через localStorage.setItem('cart', JSON.stringify(cart)). Фільтри (ціна, бренд) зберігаються в об’єкті filters і синхронізуються через localStorage. Поле пошуку (<input id="search-box">) зберігає історію в localStorage. Як показано на рис. 3.16, поле пошуку відображає результати у списку товарів.

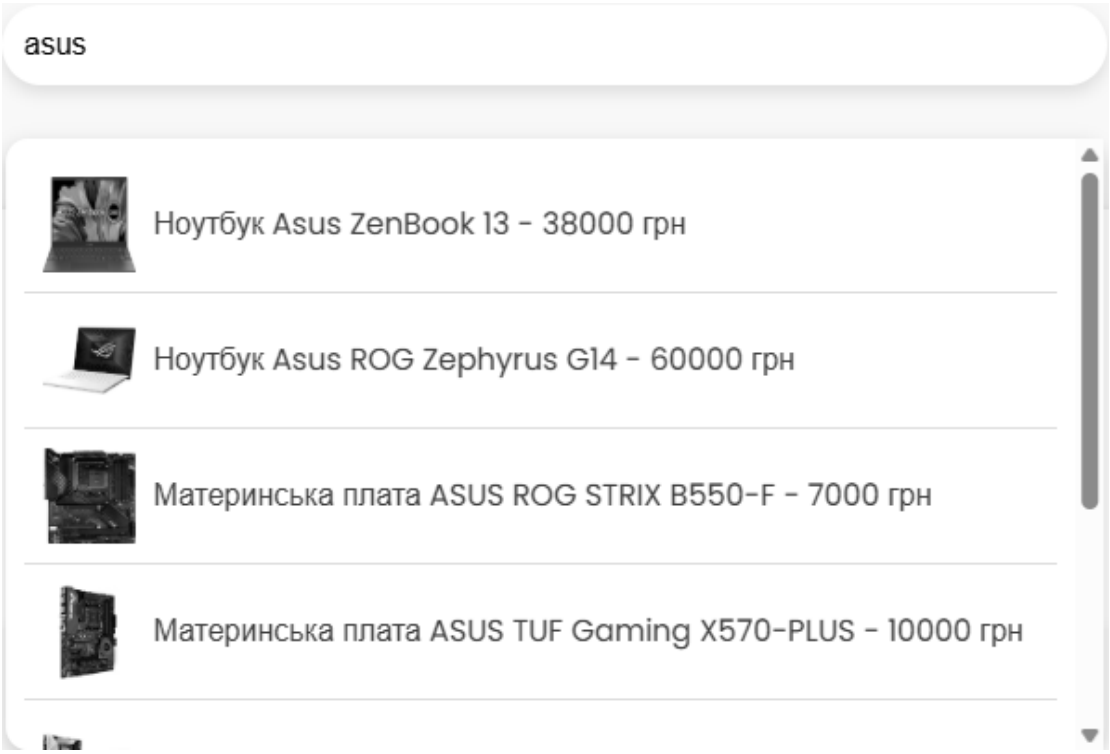

Рисунок 3.16 – Поле пошуку з результатами.

```
function updateCart() {
    const cartItems = document.getElementById("cart-items");
    const cartTotal = document.getElementById("cart-total");
    const cartIcon = document.getElementById("cart-count");
    cartItems.innerHTML = "";
    let total = 0;
    if (cart.length === 0) {
        cartItems.innerHTML = '<p>Кошик порожній</p>';
        cartIcon.textContent = "0";
        cartTotal.textContent = "Загальна сума: 0 грн";
        return;
    }
    cart.forEach((item, index) => {
        total += item.ціна * item.кількість;
        cartItems.innerHTML += `
            <li>
                <img src="${item.зображення}" alt="${item.назва}" style="width: 50px; height:
50px;">
                <span>${item.назва} - ${item.ціна} грн</span>
                <button onclick="changeQuantity(${index}, -1)">➖</button>
                ${item.кількість}
                <button onclick="changeQuantity(${index}, 1)">➕</button>
                <button onclick="removeFromCart(${index})">❌</button>
            </li>`;
    });
    cartTotal.textContent = `Загальна сума: ${total} грн`;
    cartIcon.textContent = cart.length;
    localStorage.setItem('cart', JSON.stringify(cart));
}
```

Елементи UX: Поле пошуку підтримує динамічні результати через searchProducts(). Спливаючі вікна кошика та вподобаних адаптивні (width: 90% на мобільних). Кнопки дій мають градієнтний фон. Атрибути aria-label (наприклад, aria-label="Додати до кошика") забезпечують доступність.

Для масштабованості та вдосконалення проєкту було додано інтеграцію з сервером, що дозволяє синхронізувати кошик і обране з профілем користувача через OAuth 2.0. Система рекомендацій аналізує вибрані бренди у списку обраного за допомогою TensorFlow.js, щоб пропонувати релевантні товари. Продуктивність покращено завдяки використанню дебонсингу при збереженні в localStorage (затримка 300 мс), створенню елементів через createElement замість innerHTML, а також кешуванню рекомендацій у sessionStorage та завантаженню зображень через loading="lazy", що дозволяє швидше відкривати сторінки.

Безпеку підвищено за рахунок очищення введених даних за допомогою функції sanitizeInput(), яка захищає від XSS-атак. Для тестування використовуються Jest — для функції toggleFavorite() та Stylelint — для перевірки стилів CSS.

3.6 Реалізація інтерактивної взаємодії з користувачем: пошук, фільтрація, навігація

Інтерактивна взаємодія з користувачем забезпечує швидкий пошук товарів, їх фільтрацію та інтуїтивну навігацію в e-commerce платформах. У розробленій інформаційній системі (ІС) онлайн-магазину комп'ютерної техніки реалізовані пошукова система, фільтрація товарів і навігація з хлібними крихтами. Функції дозволяють знаходити продукти, налаштовувати параметри відбору та орієнтуватися в структурі сайту. Реалізація базується на JavaScript (ES6+), HTML5, CSS3 та localStorage, забезпечуючи швидкість, адаптивність і доступність. Розділ описує технічну реалізацію, аспекти користувацького досвіду (UX), продуктивність і перспективи вдосконалення, підкріплюючи пояснення фрагментами коду та візуальними ілюстраціями.

Пошукова система дозволяє знаходити товари за назвою через <input id="search-box"> у блоці <section class="search-container">. На рис. 3.17 зображено поле пошуку з результатами на мобільному пристрої, що демонструє адаптивний дизайн.

Функція searchProducts() викликається при введенні (oninput), порівнюючи запит у нижньому регістрі з назвами товарів у каталозі (products) через includes(). Пошук охоплює категорії («Зібрані ПК», «Ноутбуки», «Комплектуючі») та підкатегорії (наприклад, «Відеокарти»). Результати відображаються в як список (search-result-item), із зображенням (50x50 пікселів), назвою, ціною та клікабельною областю для openProduct(). Порожній запит приховує результати, відсутність збігів показує «Нічого не знайдено».

Рисунок 3.17 – Поле пошуку на мобільному пристрої

```
function searchProducts() {
    const searchBox = document.getElementById("search-box");
    const searchResults = document.getElementById("search-results");
    const query = searchBox.value.toLowerCase();
    if (query.length === 0) {
        searchResults.style.display = "none";
        return;
    }
    let results = [];
```

```
            for (let category in products["Каталог"]) {
                const categoryData = products["Каталог"][category];
                if (Array.isArray(categoryData.товари)) {
                    categoryData.товари.forEach((product, index) => {
                        if (product.назва.toLowerCase().includes(query)) {
                            results.push({ category, subCategory: '', index, product });
                        }
                    });
                } else if (typeof categoryData.товари === "object") {
                    for (let subCategory in categoryData.товари) {
                        categoryData.товари[subCategory].товари.forEach((product, index) => {
                            if (product.назва.toLowerCase().includes(query)) {
                                results.push({ category, subCategory, index, product });
                            }
                        });
                    }
                }
            }
            if (results.length > 0) {
                searchResults.innerHTML = results.map(result => `
                    <div class="search-result-item" onclick="openProduct('${result.category}', '${result.subCategory || ''}', ${result.index})">
                        <img src="${result.product.зображення}" alt="${result.product.назва}" style="width: 50px; height: 50px;">
                        <p>${result.product.назва} - ${result.product.ціна} грн</p>
                    </div>
                `).join('');
                searchResults.style.display = "block";
            } else {
                searchResults.innerHTML = "<p>Нічого не знайдено</p>";
                searchResults.style.display = "block";
            }
        }
```

Фільтрація товарів реалізована через <aside id="filter"> у каталозі, із чекбоксами для параметрів (ціна, бренд, рік). Функція applyFilters(category, subCategory) збирає значення в об’єкт filters і фільтрує товари через filter, враховуючи числові діапазони (ціна) і категорійні параметри (бренд). Відфільтровані товари відображаються в через renderFilteredProducts(), із картками, що містять зображення, назву, ціну та кнопки («Детальніше», «Вподобати»). Скидання фільтрів виконує resetFilters(). Стан фільтрів зберігається в localStorage. Можна побачити на рис. 3.18 панель фільтрів із чекбоксами для вибору параметрів.

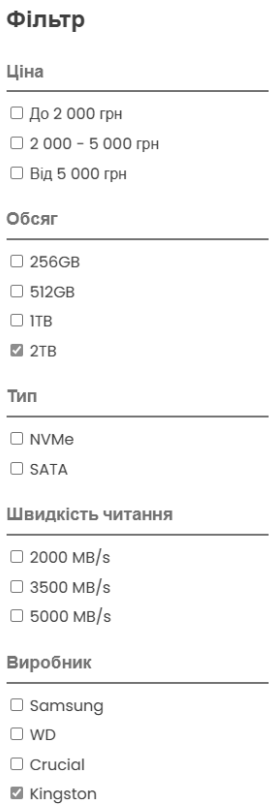

Рисунок 3.18 – Панель фільтрів із чекбоксами

```
function applyFilters(category, subCategory = '') {
const productsData = subCategory
? products["Каталог"][category].товари[subCategory].товари
: products["Каталог"][category].товари;
const filteredProducts = productsData.map((product, index) => ({ ...product, originalIndex:
index })).filter(product => {
const matchesPrice = filters.ціна.length === 0 || filters.ціна.some(range => {
    const [min, max] = range.split('-').map(Number);
    return product.ціна >= min && (max ? product.ціна <= max : true);
  });
  const matchesBrand = filters.бренд.length === 0 ||
filters.бренд.includes(product.бренд);
  return matchesPrice && matchesBrand;
});
renderFilteredProducts(filteredProducts, category, subCategory);
}
function renderFilteredProducts(filteredProducts, category, subCategory = '') {
  const content = document.getElementById("content");
  content.innerHTML = `<h2>${subCategory || category}</h2><div class="product-grid">`
+
    filteredProducts.map((product, index) => `
      <div class="product-item">
        <img src="${product.зображення}" alt="${product.назва}" style="width: 100px;
height: 100px;">
```

```
                <h3>${product.назва}</h3>
                <p>${product.ціна} грн</p>
                <div class="product-actions">
                  <button onclick="openProduct('${category}', '${subCategory || ''}', ${product.originalIndex || index})">Детальніше</button>
                  <button class="favorite-button" onclick="toggleFavorite('${category}', '${subCategory || ''}', ${product.originalIndex || index})">
                    ${favorites.some(fav => fav.назва === product.назва) ? '❤️' : '🤍'}
                  </button>
                </div>
              </div>
            `).join('') + `</div>`;
        }
```

Навігаційна система включає меню (<nav> у <header>), хлібні крихти (<nav id="breadcrumbs">) і кнопки переходу. Меню містить пункти «Головна», «Каталог», «Про нас», «Контакти», які викликають navigate(page). Функція оновлює <main id="content">, відображаючи рекомендації для home або категорії для catalog. Хлібні крихти, керовані масивом breadcrumbs, показують шлях (наприклад, «Головна > Каталог > Зібрані ПК») через updateBreadcrumbs(). Елементи крихт клікабельні, а goBack() повертає на попередній рівень. Як показано на рис. 3.19, хлібні крихти відображаються над контентом.

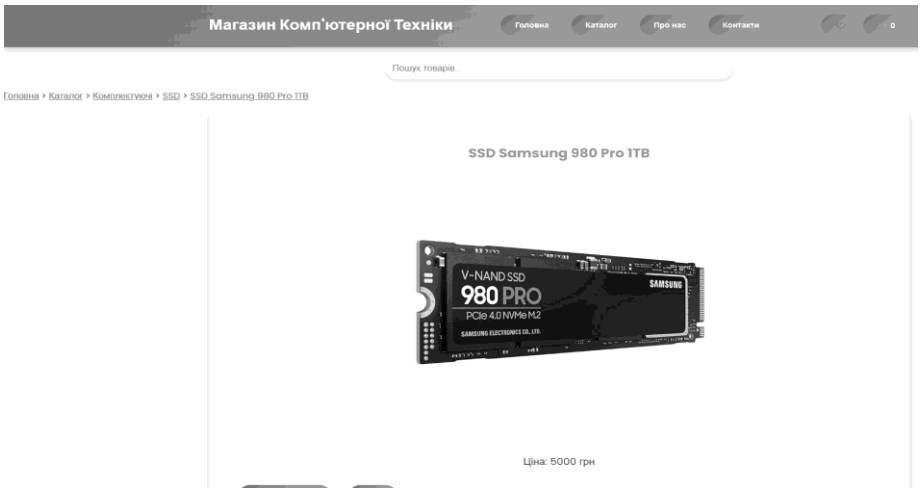

Рисунок 3.19 – Хлібні крихти над контентом

```
function navigate(page, addToBreadcrumbs = true) {
  const content = document.getElementById("content");
  const filter = document.getElementById("filter");
  content.classList.add('hidden');
  setTimeout(() => {
    if (page === "home") {
```

```
                breadcrumbs = [{ name: "Головна", page: "home" }];
                content.innerHTML = `
                    <h2>Вітаємо у нашому магазині!</h2>
                    <div class="recommendations-container">
                        <h2>Рекомендації для вас</h2>
                        <div class="recommendations" id="home-recommendations"></div>
                    </div>`;
                renderHomeRecommendations();
                filter.style.display = "none";
            } else if (page === "catalog") {
                if (addToBreadcrumbs) breadcrumbs = [{ name: "Головна", page: "home" }, { name: "Каталог", page: "catalog" }];
                content.innerHTML = `
                    <h2>Каталог</h2>
                    <div class="category-grid">
                        ${Object.keys(products["Каталог"]).map(category => `
                            <div class="category-card" onclick="navigate('${category}')">
                                <img src="${products["Каталог"][category].зображення}" alt="${category}">
                                <h3>${category}</h3>
                            </div>
                        `).join('')}
                    </div>`;
                filter.style.display = "none";
            }
            updateBreadcrumbs();
            content.classList.remove('hidden');
        }, 300);
    }
    function updateBreadcrumbs() {
        let breadcrumbsNav = document.getElementById("breadcrumbs");
        breadcrumbsNav.innerHTML = breadcrumbs
            .map((crumb, index) => `<a href="#" onclick="navigate('${crumb.page}', false)">${crumb.name}</a>${index < breadcrumbs.length - 1 ? ' > ' : ''}`)
            .join("");
    }
```

Елементи UX: Поле пошуку адаптивне (width: 80% на мобільних) із тінню при фокусі. Результати пошуку підтримують гортання (overflow-y: auto). Фільтри мають чіткі мітки, а їхнє застосування оновлює сітку товарів. Хлібні крихти відображаються горизонтально (display: flex) із сепаратором «>». Атрибути aria-label забезпечують доступність. Медіа-запити (@media (max-width: 768px)) адаптують інтерфейс.

Для покращення масштабованості і вдосконалення проєкту було реалізовано кілька важливих рішень. Пошук і фільтрація були перенесені на

сервер через API, наприклад /search?q=query, а URL синхронізовано з хлібними крихтами. Для оптимізації пошуку додано дебонсинг із затримкою 300 мс, а також реалізовано нечіткий пошук за допомогою бібліотеки Fuse.js. Фільтри створюються динамічно на основі метаданих, а для числових значень використовуються слайдери. Навігація побудована на клієнтському маршрутизаторі Navigo, з кешуванням у sessionStorage, що пришвидшує переходи між сторінками. Для захисту від XSS-атак усі запити очищуються за допомогою функції sanitizeInput(). Тестування проводиться за допомогою Jest для функцій пошуку та фільтрації, Cypress — для перевірки навігації, а Stylelint використовується для контролю якості CSS.

У плані продуктивності клієнтська обробка забезпечує високу швидкість для невеликих каталогів. Додаткове кешування фільтрів у sessionStorage та використання loading="lazy" для зображень допомагають оптимізувати час завантаження сторінки.

3.7 Визначення вимог до апаратної частини та технічної інфраструктури

Ключову роль у забезпеченні продуктивності, масштабованості та надійності інформаційної системи онлайн-магазину комп’ютерної техніки відіграють апаратні засоби, що забезпечують її функціонування. Розглядувана інформаційна система будується із застосуванням хмарних технологій, серверної інфраструктури та клієнтських пристроїв, що у сукупності повинні забезпечувати швидкий доступ до системи, безперебійну обробку запитів і захист даних.

В основі розроблюваної інформаційної системи онлайн-магазину комп’ютерної техніки лежить клієнт-серверна архітектура. Як сервер передбачається використання хмарної серверної інфраструктура, що наддасть можливість масштабувати ресурси залежно від навантаження. Основні апаратні вимоги до серверів включають:

— багатоядерний серверний процесор (наприклад, Intel Xeon або AMD EPYC) з частотою не менше 2.5 ГГц, що забезпечує швидку обробку запитів до

бази даних і виконання серверних скриптів; для пікових навантажень (наприклад, під час розпродажів) рекомендується щонайменше 8 ядер;

— оперативна пам’ять об’ємом від 16 ГБ для обробки одночасних запитів, кешування даних (наприклад, через Redis) і роботи з базами даних; для великих каталогів і високого трафіку рекомендується 32 ГБ або більше;

— кілька SSD-дисків сумарною ємністю не менше 500 ГБ, об’єднаних у RAID-масив (RAID 5 або RAID 10) для забезпечення швидкого доступу до бази даних (наприклад, PostgreSQL) та відмовостійкого зберігання медіафайлів (зображень, відео та ін.);

— мережевий інтерфейс Gigabit Ethernet (1 Гбіт/с) для швидкої передачі даних між сервером, клієнтами та зовнішніми сервісами (наприклад, API Нової Пошти чи Stripe).

Для масштабованості використовується хмарна платформа, наприклад, Amazon Web Services (AWS) EC2 або Google Cloud Platform (GCP). Контейнеризація через Docker і оркестрація через Kubernetes, як описано в розділі 2.3, дозволяють розподіляти навантаження між кількома серверами. Наприклад, окремий сервер для обробки транзакцій (LiqPay, Stripe) ізольовано від основного сервера каталогу, що підвищує безпеку та продуктивність.

Основні вимоги до налаштувань конфігурації хмарної платформи на прикладі AWS:

— веб-сервер: Nginx для обробки статичних файлів (CSS, зображення) і балансування навантаження між Node.js-серверами;

— контейнеризація: Docker-контейнери для ізоляції модулів (frontend, backend, база даних), що спрощує масштабування та деплой;

— оркестрація: Kubernetes для автоматичного розподілу навантаження, відновлення після збоїв і моніторингу;

— моніторинг: Prometheus і Grafana для відстеження метрик продуктивності (час відгуку, використання CPU, помилки API); логування через ELK Stack (Elasticsearch, Logstash, Kibana) дозволяє аналізувати журнали запитів і виявляти аномалії;

— шифрування: HTTPS із SSL/TLS-сертифікатами (Let’s Encrypt) для захисту трафіку, як описано в розділі 2.4;

— захист від атак: Cloudflare для фільтрації DDoS-атак і WAF (Web Application Firewall) для блокування SQL-ін’єкцій і XSS;

— CDN: використання Content Delivery Network (наприклад, Cloudflare CDN) для кешування статичних ресурсів (зображень, CSS, JavaScript) у різних регіонах, що прискорює завантаження для користувачів;

— регулярне резервне копіювання бази даних (щоденно) на окремий сервер або хмарне сховище (AWS S3);

— реплікація бази даних (master-slave) для швидкого відновлення у разі збою;

— автоматичне створення снапшотів контейнерів для відновлення системи після критичних помилок.

В додатку Г рисунок Г.2 зображено схему технічного середовища ІС онлайн-магазину, яка включає:

— серверний рівень: хмарний сервер (AWS EC2) із Nginx, Node.js, PostgreSQL, Redis і Docker-контейнерами;

— клієнтський рівень: смартфони, планшети, ПК із браузерами, підключені через HTTPS;

— мережевий рівень: Cloudflare для DDoS-захисту і CDN для кешування.

Резервне копіювання: AWS S3 для зберігання бекапів. Стрілки показують потік даних: від клієнтських запитів через Cloudflare до серверів, із обробкою в базах даних і поверненням відповіді.

Робота інформаційної системи онлайн-магазину підтримується персоналом, серед якого основні завдання покладаються на адміністраторів та менеджерів. Адміністратор відповідає за технічне управління ІС, включаючи налаштування серверів, адміністрування баз даних, оновлення програмного забезпечення, забезпечення безпеки та моніторинг продуктивності. Відповідно пристрій адміністратора повинен задовольняти вимогам, що наведені у таблиці 2.1.

Таблиця 2.1 – Вимоги до пристрою адміністратора

Компонент	Мінімальні вимоги	Рекомендовані вимоги	Примітки
Процесор	Чотириядерний, 2.8 ГГц (Intel Core i5, AMD Ryzen 5)	Чотириядерний, 3.2+ ГГц (Intel Core i7, AMD Ryzen 7)	Для обробки логів, звітів і багатозадачності.
Оперативна пам’ять	16 ГБ RAM	32 ГБ RAM	Для одночасної роботи IDE, SSH-клієнтів, браузерів і систем моніторингу.
Сховище	SSD, 512 ГБ	SSD, 1 ТБ	Для локальних копій баз даних, логів і конфігураційних файлів.
Монітор	1920x1080 пікселів, 24 дюйми	2560x1440 пікселів, 27+ дюймів	Для роботи з кількома вікнами (термінал, документація, моніторинг).
Інтернет-з’єднання	50 Мбіт/с	100+ Мбіт/с	Для віддаленого доступу до серверів (SSH), хмарних сервісів (AWS, Cloudflare).

Адміністратор має доступ до серверної інфраструктури через SSH і адмін-панель із розширеними правами. Наприклад, може налаштувати Nginx або оновити Docker-контейнери. Він працює через захищене VPN-з’єднання або SSH-тунель для безпечного доступу до серверів. Наприклад, для оновлення каталогу товарів адміністратор підключається до сервера через SSH, редагує базу даних PostgreSQL і аналізує логи. Пристрій підтримує багатозадачність, забезпечуючи одночасне керування серверами, моніторинг продуктивності та оновлення конфігурацій.

Менеджер відповідає за операційну діяльність: обробку замовлень, комунікацію з клієнтами, управління каталогом товарів і аналіз звітів продажів. Робота зосереджена на веб-інтерфейсі ІС, що знижує вимоги до пристрою.

Менеджер працює через модуль адмін-панелі, який дозволяє обробляти замовлення, редагувати товари та переглядати звіти. Наприклад, менеджер може змінити статус замовлення на «Обробляється», зв’язатися з клієнтом через інтегрований чат і відправити запит на доставку. Для вирішення цих завдань підійде ноутбук, настільний комп’ютер або планшет. Основні вимоги до пристрою менеджера наведені у таблиці 2.2.

Таблиця 2.2 – Вимоги до пристрою менеджера

Компонент	Мінімальні вимоги	Рекомендовані вимоги	Примітки
Процесор	Двоядерний, 1.8 ГГц (Intel Core i3 або екв.)	Чотириядерний, 2.0+ ГГц (Intel Core i5)	Для роботи з веб-інтерфейсом і браузером.
Оперативна пам’ять	8 ГБ RAM	16 ГБ RAM	Для браузера з кількома вкладками і офісних програм (Google Sheets).
Сховище	SSD, 256 ГБ	SSD, 512 ГБ	Для тимчасових файлів (експортовані звіти в CSV).
Монітор	1366x768 пікселів	1920x1080 пікселів	Для роботи з адмін-панеллю; менший екран для ноутбуків чи планшетів.
Інтернет-з’єднання	10 Мбіт/с	20+ Мбіт/с	Для швидкого завантаження сторінок і комунікації з клієнтами.

Клієнтська частина інформаційної системи доступна через веб-браузери на різних пристроях: настільних комп’ютерах, ноутбуках, планшетах і смартфонах. Мінімальні вимоги до апаратного забезпечення клієнтів:

— двоядерний процесор із частотою 1.5 ГГц (наприклад, Intel Core i3 або еквівалент для мобільних пристроїв);

— оперативна пам’ять 4 ГБ RAM для комфортної роботи браузера з JavaScript-додатками (наприклад, React для frontend);

— дисплей з роздільною здатністю від 320x568 пікселів (для мобільних пристроїв) до 1920x1080 пікселів (для десктопів), з підтримкою адаптивного дизайну через CSS-медіазапити, як описано в розділі 3.2;

— :швидкість Інтернет-з’єднання від 5 Мбіт/с.

Система підтримує сучасні браузери (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) з увімкненим JavaScript і підтримкою стандартів HTML5/CSS3. Для оптимізації продуктивності використовується ліниве завантаження зображень (loading="lazy") і кешування статичних ресурсів, що зменшує вимоги до клієнтського обладнання.

ВИСНОВКИ

Системи електронної комерції набувають усе більшої популярності в умовах цифровізації та зростання онлайн-покупок, що зумовлює актуальність розробки інноваційних та зручних вебрішень для онлайн-магазинів. Вони стають не лише інструментом продажу, а й важливим чинником конкурентоспроможності бізнесу.

У процесі роботи було проаналізовано сучасні вебтехнології, які дають змогу створювати адаптивні, зручні та функціональні сайти. Для реалізації клієнтської частини застосовувались мови HTML і CSS, а також JavaScript. Було використано інструменти стилізації, зокрема Flexbox і Grid, для побудови зрозумілого і привабливого інтерфейсу. Було враховано психологічні аспекти вибору кольорової гами, що підсилює зорове сприйняття й комфорт користувача. Правильно підібрані кольори позитивно впливають на загальну візуальну привабливість сайту і сприяють кращій навігації.

У рамках реалізації було створено логічну структуру сайту, яка включає сторінки з категоріями товарів, описами, кошиком, профілем користувача, а також формою оформлення замовлення. Продумано не лише зовнішній вигляд, а й внутрішня логіка взаємодії елементів. Окрему увагу приділено модулю рекомендацій, що дозволяє відображати персоналізовані пропозиції на основі активності користувача. Це підвищує зручність використання та сприяє зростанню зацікавленості.

Додатково до функціональної частини було реалізовано елементи гейміфікації, зокрема систему рівнів, очок досвіду, бейджів і віртуальних монет, які можна витрачати в магазині. Такий підхід не лише урізноманітнює взаємодію з сайтом, але й мотивує користувачів повертатися, що може бути важливим чинником у реальному комерційному використанні.

Розроблена інформаційна система орієнтована на покупців електроніки й комп’ютерних комплектуючих та дозволяє швидко отримати потрібну інформацію, здійснити пошук, порівняння, додавання до кошика, вподобань

тощо. Вона може бути використана як основа для реального вебмагазину, з можливістю подальшого розвитку, інтеграції з серверною частиною, базами даних та платіжними системами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Годлевський Д.В. Тарновський М.Г. Інформаційна система для онлайн-магазинун комп'ютерної техніки // Матеріали всеукраїнської науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців

«Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» (15 листопада 2024 р . - 14 червня 2025 р., Вінниця) : URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25372

2. Гороховський О.І., Азаров О.Д., Трояновська Т.І. Інформаційна технологія доставки контенту у системі комп'ютеризованої підготовки спеціалістів: монографія. Вінниця: ВНТУ, 2016. 160 с.

3. Як працює інтернет-магазин: основні процеси. URL: https://wezom.com.ua/ua/blog/biznes-processy-internet-magazina-i-ih-optimizaciya

4. Етапи розробки веб-сайту. URL: https://my-master.net.ua/ua/etapi-stvorennya-sajtu-z-nulya/

5. Методи розробки сайтів. URL: https://webstudio2u.net/ua/webdesign/354-site-developmethods.html

6. React Documentation. URL: https://react.dev/

7. Node.js Documentation. URL: https://nodejs.org/uk

8. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

9. Docker Documentation. URL: https://docs.docker.com/

10. HTML. URL: https://uk.wikipedia.org/wiki/HTML

11. CSS. URL: https://uk.wikipedia.org/wiki/CSS

12. JavaScript. URL: https://uk.wikipedia.org/wiki/JavaScript

13. База даних. URL: https://uk.wikipedia.org/wiki/База_даних

14. UX Design Principles. URL: https://www.nngroup.com/articles/ten-usability-heuristics/

15. Персоналізація в e-commerce: як підвищити конверсію. URL: https://turumburum.ua/blog/personalizaciya-v-marketingu-zakordonni-keysi-ta-praktichni-poradi-dlya-ecommerce

16. Інтернет-магазин Rozetka. URL: https://rozetka.com.ua/

17. Інтернет-магазин Brain. URL: https://brain.com.ua/

18. Інтернет-магазин Citrus. URL: https://www.citrus.ua/

19. W3C Web Accessibility Initiative (WAI). Web Content Accessibility Guidelines (WCAG) 2.1. URL:
https://www.w3.org/WAI/standards-guidelines/wcag/

20. OWASP Top Ten Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/

21. E-commerce Trends 2025: Personalization and AI in Online Retail. URL: https://www.shopify.com/enterprise/blog/global-ecommerce-statistics

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д.

20.03.2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

“ Інформаційна система для онлайн-магазину комп'ютерної техніки ”

08-54.БКР.004.00.000 ПЗ

Керівник роботи к.т.н. доц. каф. ОТ

Тарновський М. Г

Студент групи 1КІ–21б

Годлевський Д. В.

ВНТУ 2025

1 Підстава для виконання комплексного бакалаврської кваліфікаційної роботи (БКР)

1.1 Актуальність роботи обумовлена тим, що роботи обумовлена тим, що Інтернет-магазини комп’ютерної техніки, як один із сегментів електронної комерції, демонструють стійке зростання. Попит на технічні засоби зумовлений постійною модернізацією апаратного та програмного забезпечення, а також розширенням сфер їх застосування. У зв’язку з цим зростає потреба у високоякісних інформаційних системах, здатних забезпечити ефективне управління товарними запасами, процесами замовлень, аналітикою продажів, взаємодією з клієнтами та іншими бізнес-процесами.

1.2 Наказ про затвердження теми БКР.

2 Мета БКР і призначення розробки

2.1 Мета роботи — розширення функціональних можливостей інформаційної системи онлайн-магазину комп’ютерної техніки відповідно до сучасних вимог електронної комерції.

2.2 Призначення розробки — інформаційна система призначена для автоматизації основних процесів онлайн-магазину комп’ютерної техніки: управління товарами, оформлення замовлень, взаємодії з клієнтами та інтеграції з платіжними й логістичними сервісами. Вона має забезпечити зручність, швидкодію, безпеку та можливість масштабування.

3 Вихідні дані для виконання БКР

3.1 Підтримка типових функцій систем електронної комерції.

3.2 Підтримка інтеграції із зовнішніми сервісами (оплата, доставка).

3.3 Забезпечення адаптивного дизайну та кросбраузерної сумісності (hrome, Firefox, Safari, Edge).

3.4 Інтеграція з інструментами аналітики таким як Google Analytics, Hotjar.

3.5 Захист персональних даних користувачів і фінансових транзакцій відповідно до міжнародних стандартів, таких як GDPR, PCI DSS.

4 Вимоги до виконання БКР

4.1 Провести аналіз предметної області.

4.2 Вибрати та обгрунтувати аналогів.

4.3 Вибрати та обґрунтувати інструменти для розробки інформаційної системи.

4.4 Розробити функціональні модулі інформаційної системи.

4.5 Розробити адміністративну панель.

5 Етапи БКП та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи БКП

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз предметної області.	22.03.25	26.03.25	Вступ, Розділ 1
2	Вибір та обгрунтування аналогів	27.03.25	03.04.25	Розділ 1
3	Підбір та аргументація засобів розробки інформаційної системи	04.04.25	11.04.25	Розділ 2
4	Створення інформаційної системи для онлайн-магазину комп’ютерної техніки	12.04.25	27.04.25	Розділ 3
5	Оформлення пояснювальної записки,графічного матеріалу і презентації	28.04.25	8.05.25	ПЗ, графічний матеріал і презентація
6	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	09.05.25	13.05.25	Оформленні документи

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21.

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Інформаційна система для онлайн-магазину комп'ютерної техніки

Тип роботи: _____ бакалаврська кваліфікаційна робота _____

(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism __3.5__ %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Тарновський М.Г., к.т.н., доц. каф. ОТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Годлевський Д.В.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Графічна частина

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ОНЛАЙН-МАГАЗИНУ КОМПЮ’ТЕРНОЇ ТЕХНІКИ

КОМПЛЕКСНІ СХЕМИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОНЛАЙН-МАГАЗИНУ: СТРУКТУРА, ТЕХНІЧНЕ СЕРЕДОВИЩЕ ТА ВЗАЄМОДІЯ КОРИСТУВАЧІВ

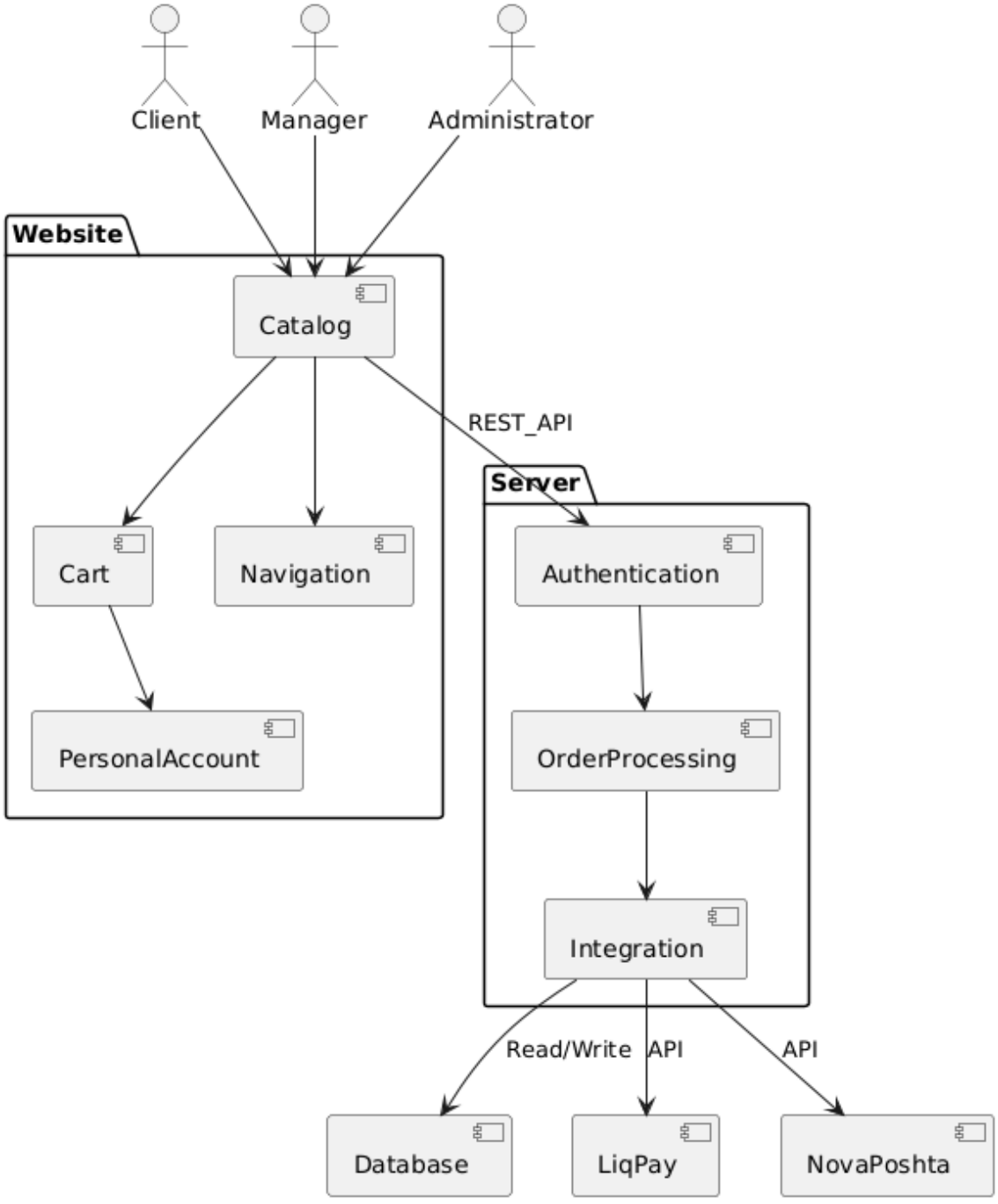

Рисунок В.1 — Загальна структура інформаційної системи онлайн-магазину

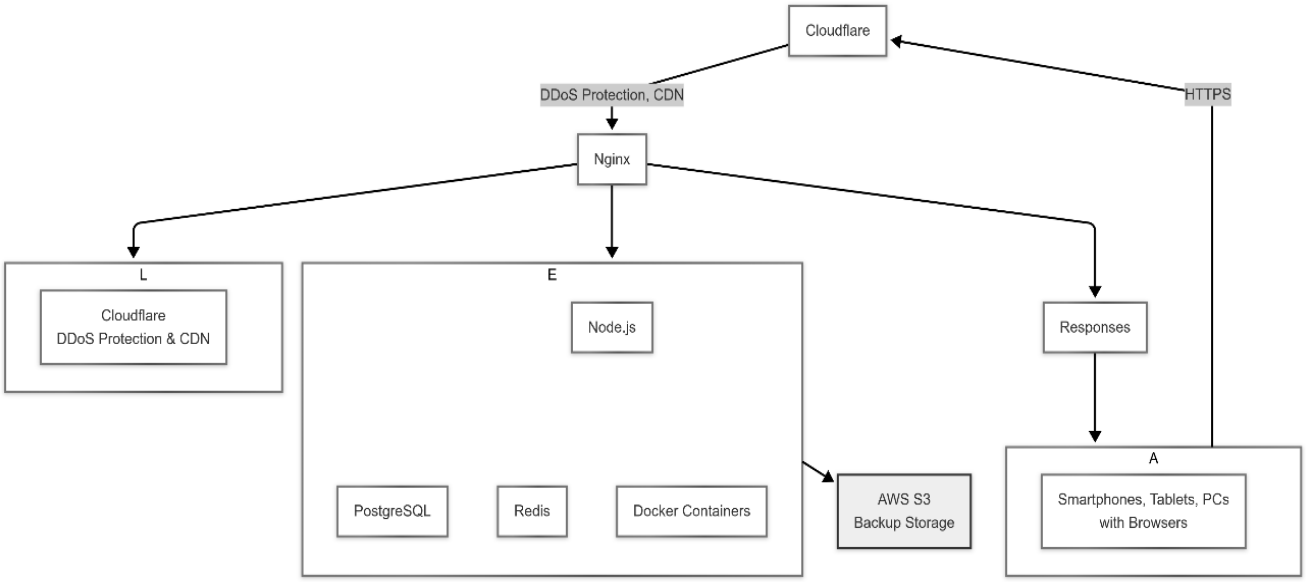

Рисунок В.2 — Схема технічного середовища інформаційної системи

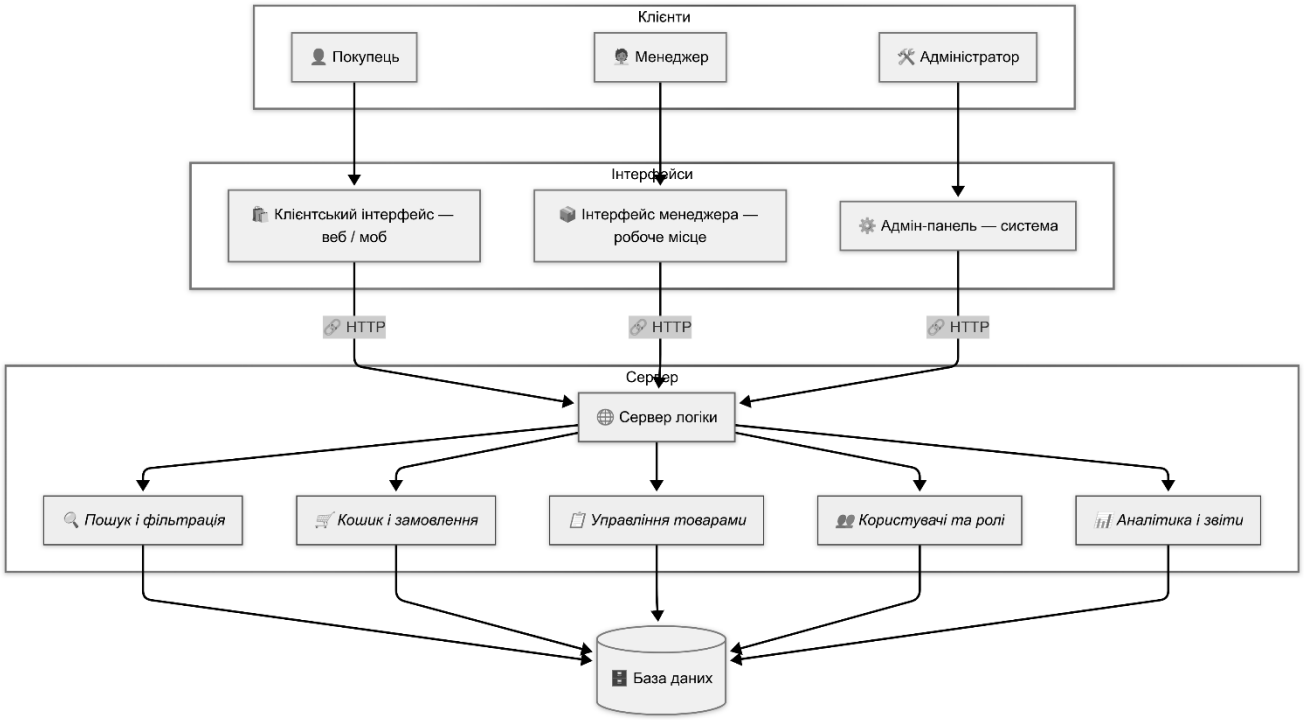

Рисунок В.3— Загальна схема взаємодії користувачів з інформаційною системою онлайн-магазину

ДОДАТОК Г

Ілюстративна частина

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ОНЛАЙН-МАГАЗИНУ КОМПЮ’ТЕРНОЇ ТЕХНІКИ

ЗОВНІШНІЙ ВИГЛЯД ОСНОВНИХ СТОРІНОК

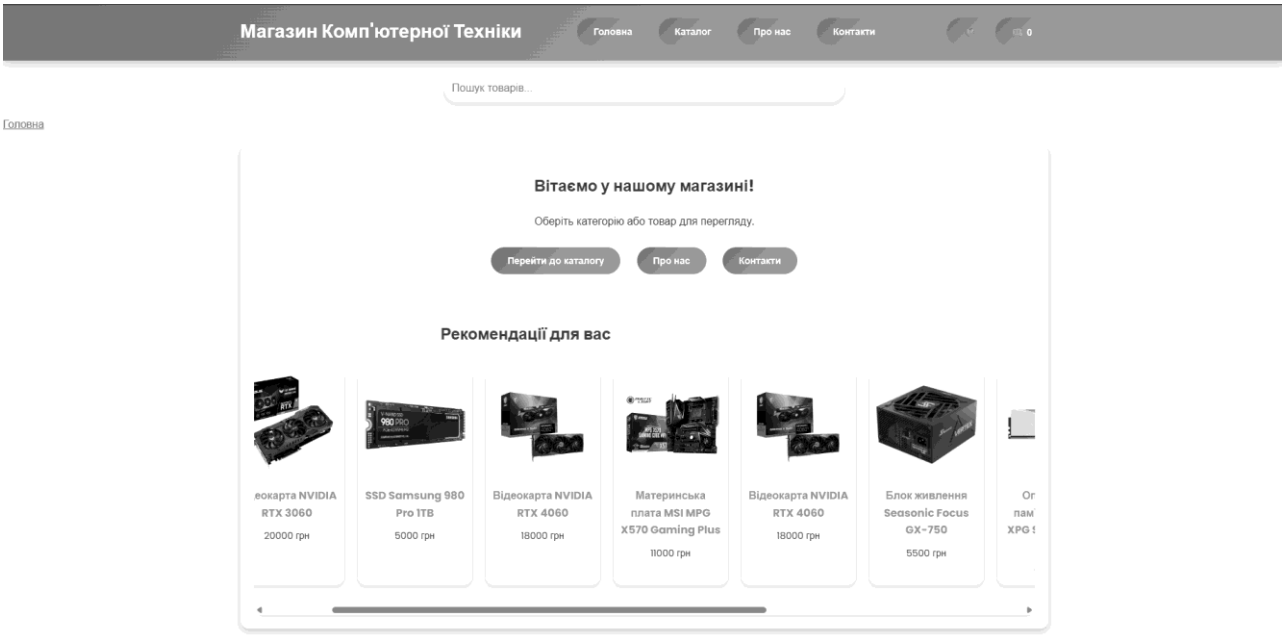

Рисунок Г.1— Головна сторінка магазину

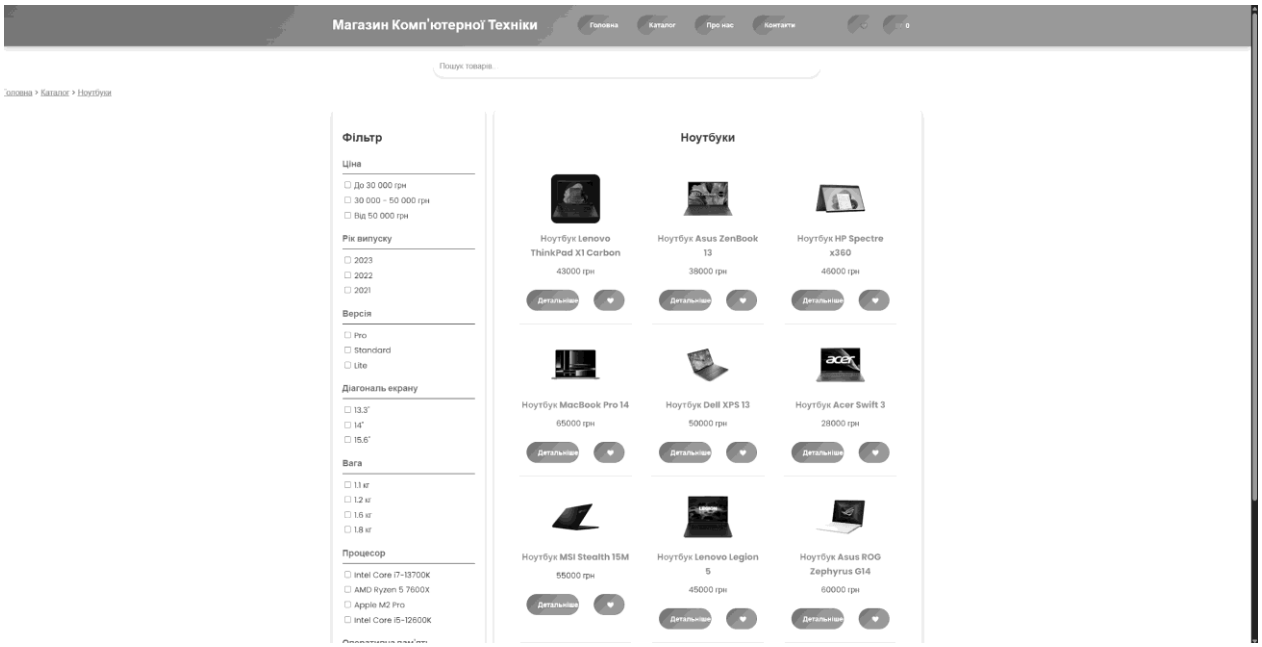

Рисунок Г.2 – Сторінка каталогу товарів

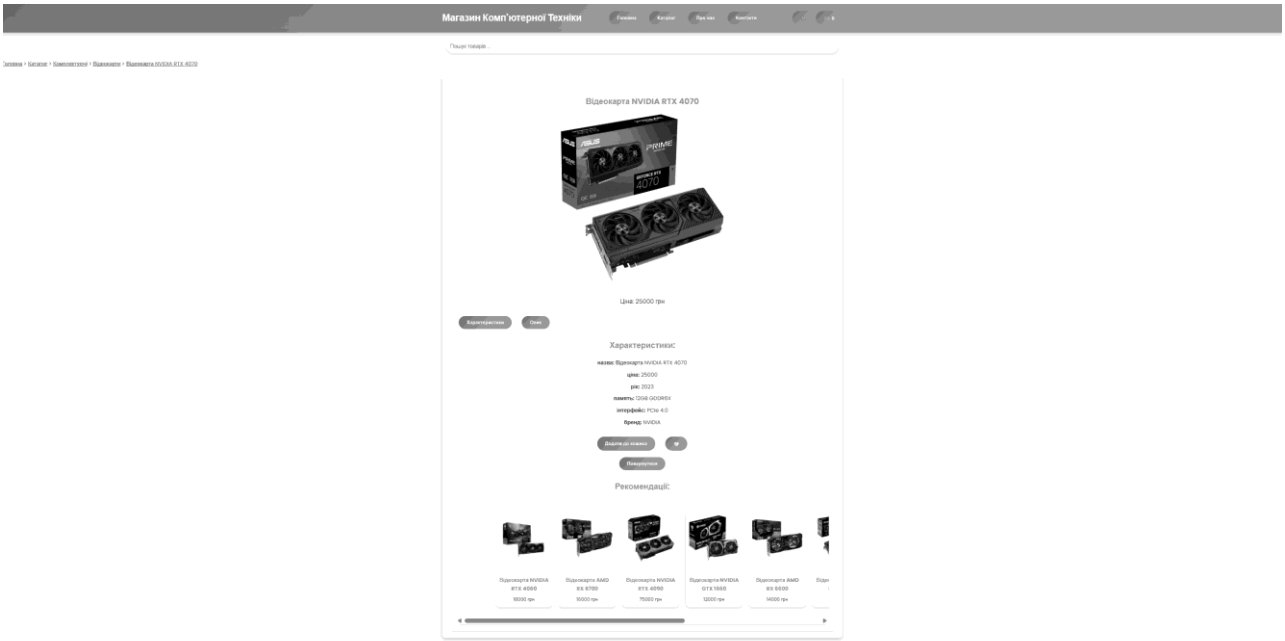

Рисунок Г.3— Сторінка товару

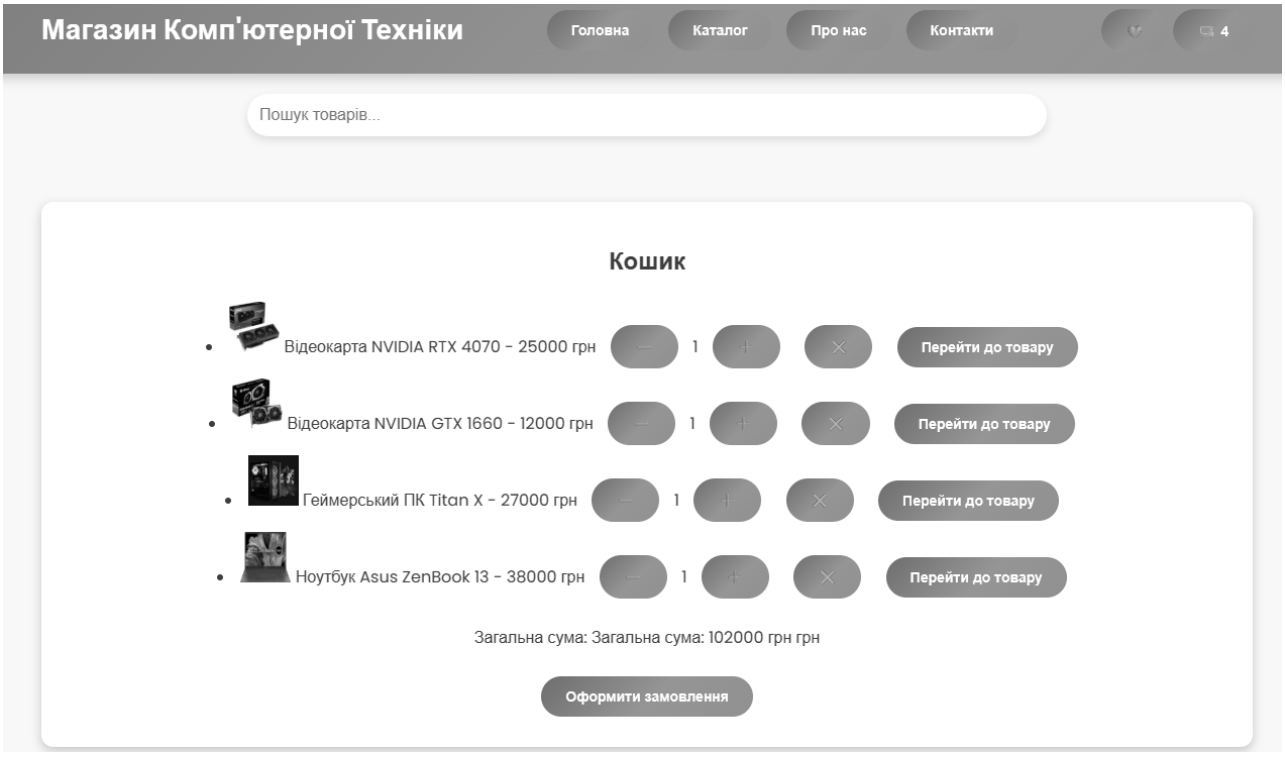

Рисунок Г.4 – Сторінки кошика

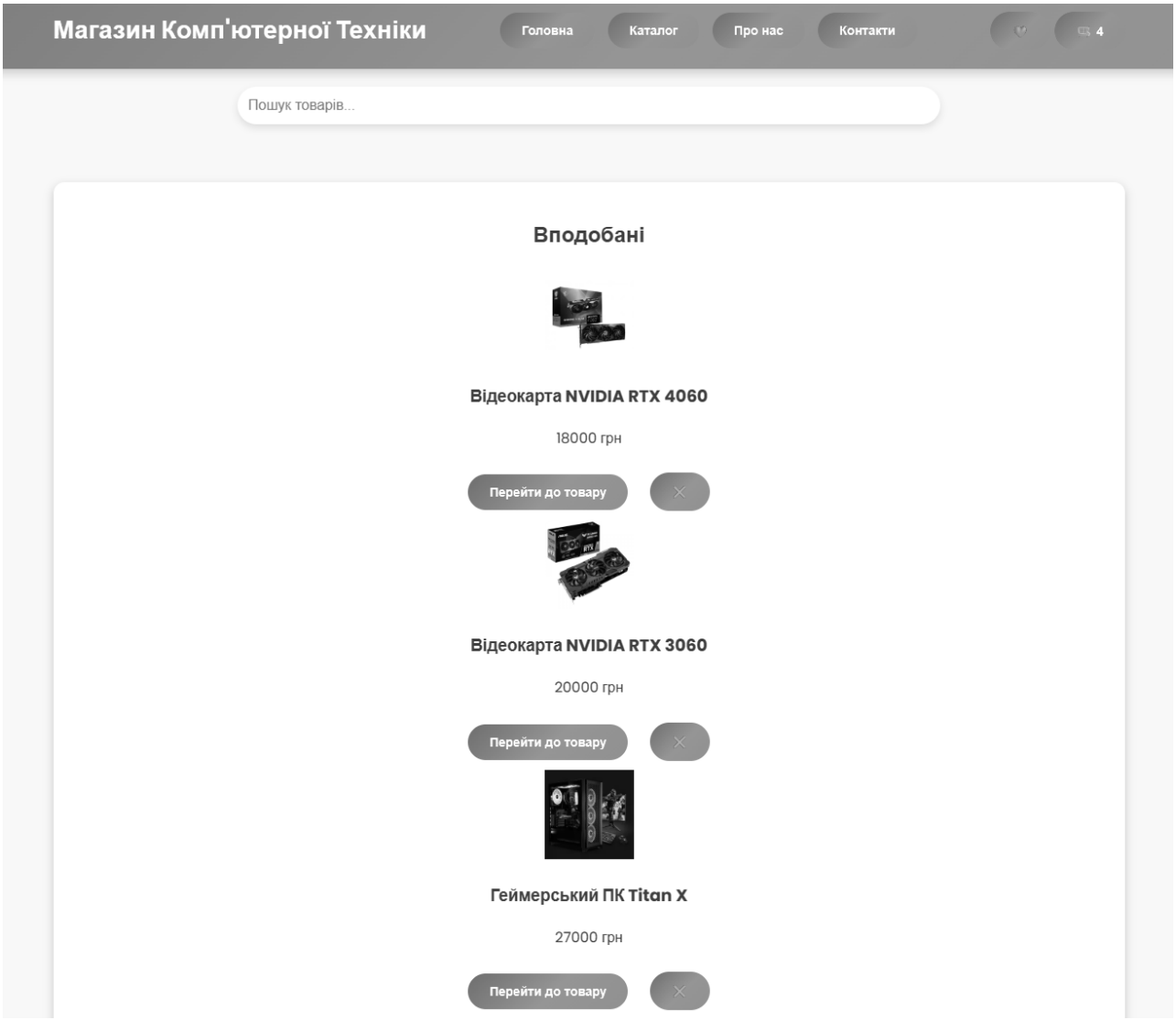

Рисунок Г.5 – Модальне вікно вподобаних

ДОДАТОК Д

Лістинг коду клієнтської частини застосунку

```
function navigate(page, addToHistory = true) {
   console.log("Navigating to:", page);
   if (addToHistory) {
      window.history.pushState({ page }, "", `#${page}`);
   }
function renderCatalog() {
   const content = document.getElementById("content");
   content.innerHTML = ""; // Очистити попередній контент

   let contentHTML = "<h2>Каталог</h2>";
   contentHTML += `<div class="catalog-grid">`;
   for (let category in products["Каталог"]) {
      const categoryData = products["Каталог"][category];
      contentHTML += `
         <div class="catalog-item">
            <img src="${categoryData.зображення}" alt="${category}">
            <h3>${category}</h3>
            <button onclick="renderCategory('${category}')">Переглянути</button>
         </div>
      `;
   }
function renderCategory(category) {
   const content = document.getElementById("content");
   const filter = document.getElementById("filter");
   content.classList.add('hidden');
   setTimeout(() => {
      breadcrumbs = breadcrumbs.slice(0, 2);
      breadcrumbs.push({ name: category, page: category });
      let contentHTML = `<h2>${category}</h2>`;
      let items = products["Каталог"][category].товари;
      if (Array.isArray(items)) {
         contentHTML += `<div class="product-grid">`;
         contentHTML += items.map((product, index) => `
            <div class="product-item">
                  <img src="${product.зображення}" alt="${product.назва}" style="width: 100px;
height: 100px;">
               <h3>${product.назва}</h3>
               <p>${product.ціна} грн</p>
               <div class="product-actions">
                  <button onclick="openProduct('${category}', '', ${index})">Детальніше</button>
                <button class="favorite-button" onclick="toggleFavorite('${category}', '', ${index})">
                     ${favorites.some(fav => fav.назва === product.назва) ? '❤️' : '🤍'}
                  </button>
               </div>
            </div>
         `).join('');
         contentHTML += `</div>`;
```

```
        filter.style.display = "block";
    }
function renderSubCategory(category, subCategory) {
  const content = document.getElementById("content");
  content.classList.add('hidden');
  setTimeout(() => {
    breadcrumbs = [
      { name: "Головна", page: "home" },
      { name: "Каталог", page: "catalog" },
      { name: category, page: category },
      { name: subCategory, page: subCategory }
    ];
    let contentHTML = `<h2>${subCategory}</h2>`;
    // Отримуємо дані для підкатегорії
    const subCategoryData = products["Каталог"][category].товари[subCategory];
    if (subCategoryData && Array.isArray(subCategoryData.товари)) {
      const items = subCategoryData.товари;

      contentHTML += `<div class="product-grid">`;
      contentHTML += items.map((product, index) => `
        <div class="product-item">
            <img src="${product.зображення}" alt="${product.назва}" style="width: 100px;
height: 100px;">
          <h3>${product.назва}</h3>
          <p>${product.ціна} грн</p>
          <div class="product-actions">
              <button onclick="openProduct('${category}', '${subCategory.replace(/'/g, "\\'")}',
${index})">Детальніше</button>
                  <button class="favorite-button" onclick="toggleFavorite('${category}',
'${subCategory.replace(/'/g, "\\'")}', ${index})">
                ${favorites.some(fav => fav.назва === product.назва) ? '❤️' : '🤍'}
              </button>
          </div>
        </div>
      `).join('');
      contentHTML += `</div>`;
    }
function toggleFavorite(category, subCategory, index) {
  let product;
  if (subCategory) {
    product = products["Каталог"][category].товари[subCategory].товари[index];
  } else {
    product = products["Каталог"][category].товари[index];
  }
  const favoriteIndex = favorites.findIndex(fav => fav.назва === product.назва);
  if (favoriteIndex === -1) {
    favorites.push(product);
  } else {
    favorites.splice(favoriteIndex, 1);
  }
  updateFavorites();
```

```
    updateFavoriteButtons();
}
function updateFavoriteButtons() {
    const favoriteButtons = document.querySelectorAll('.favorite-button');
    favoriteButtons.forEach(button => {
        const onclick = button.getAttribute('onclick');
        const match = onclick.match(/toggleFavorite\('([^']*)',\s*'([^']*)',\s*(\d+)\)/);
        if (match) {
            const category = match[1];
            const subCategory = match[2];
            const index = parseInt(match[3]);
            let продукт;
            if (subCategory) {
                продукт = products["Каталог"][category].товари[subCategory].товари[index];
            } else {
                продукт = products["Каталог"][category].товари[index];
            }
            button.textContent = favorites.some(fav => fav.назва === продукт.назва) ? '❤️' : '🤍';
        }
function removeFromFavorites(index) {
    favorites.splice(index, 1);
    updateFavorites();
    toggleFavorites();
}
function updateFavorites() {
    const favoritesItems = document.getElementById("favorites-items");
    favoritesItems.innerHTML = "";
    favorites.forEach((product, index) => {
        favoritesItems.innerHTML += `
            <li>
                <img src="${product.зображення}" alt="${product.назва}" style="width: 50px; height:
50px;">
                <span>${product.назва} - ${product.ціна} грн</span>
                <button onclick="removeFromFavorites(${index})">❌</button>
                <button onclick="openProductFromFavorites(${index})">Перейти до товару</button>
            </li>
        `;
    });
}
// Відкриття/закриття вподобаних
function toggleFavorites() {
    const content = document.getElementById("content");
    const filter = document.getElementById("filter");
    content.classList.add('hidden');
    setTimeout(() => {
        breadcrumbs = [{ name: "Головна", page: "home" }, { name: "Вподобані", page: "favorites"
}];
        let contentHTML = `<h2>Вподобані</h2>`;
        contentHTML += `<div class="favorites-grid">`;

        if (favorites.length > 0) {
```

```
        contentHTML += favorites.map((product, index) => `
          <div class="favorite-item">
                <img src="${product.зображення}" alt="${product.назва}" style="width: 100px;
height: 100px;">
            <h3>${product.назва}</h3>
            <p>${product.ціна} грн</p>
            <div class="favorite-actions">
                        <button onclick="openProductFromFavorites(${index})">Перейти до
товару</button>
              <button onclick="removeFromFavorites(${index})">❌</button>
            </div>
          </div>
        `).join('');
      }
function openProductFromFavorites(index) {
  const product = favorites[index];
  const category = product.category; // Додайте поле category до об'єкта товару
  const subCategory = product.subCategory || ''; // Додайте поле subCategory до об'єкта товару
    const productIndex = products["Каталог"][category].items.findIndex(p => p.name ===
product.name);

  if (productIndex !== -1) {
    openProduct(category, subCategory, productIndex);
  } else {
    alert("Товар не знайдено в каталозі!");
  }
}
function updateFilters(category, subCategory = '') {
  const filterContent = document.getElementById("filter-content");
  filterContent.innerHTML = ""; // Очистити попередні фільтри
  if (category === "Зібрані ПК") {
    filterContent.innerHTML = `
      <div class="filter-section">
        <h3>Ціна</h3>
        <label><input type="checkbox" name="ціна" value="0-20000"> До 20 000 грн</label>
          <label><input type="checkbox" name="ціна" value="20000-40000"> 20 000 - 40 000
грн</label>
        <label><input type="checkbox" name="ціна" value="40000+"> Від 40 000 грн</label>
      </div>
      <div class="filter-section">
        <h3>Рік випуску</h3>
        <label><input type="checkbox" name="рік" value="2023"> 2023</label>
        <label><input type="checkbox" name="рік" value="2022"> 2022</label>
        <label><input type="checkbox" name="рік" value="2021"> 2021</label>
      </div>
      <div class="filter-section">
        <h3>Версія</h3>
        <label><input type="checkbox" name="версія" value="Pro"> Pro</label>
        <label><input type="checkbox" name="версія" value="Standard"> Standard</label>
        <label><input type="checkbox" name="версія" value="Lite"> Lite</label>
      </div>
```

```
        <div class="filter-section">
          <h3>Оперативна пам'ять</h3>
                <label><input type="checkbox" name="оперативна_память" value="8GB">
8GB</label>
              <label><input type="checkbox" name="оперативна_память" value="16GB">
16GB</label>
              <label><input type="checkbox" name="оперативна_память" value="32GB">
32GB</label>
        </div>
        <div class="filter-section">
          <h3>Накопичувач</h3>
            <label><input type="checkbox" name="накопичувач" value="256GB SSD"> 256GB
SSD</label>
            <label><input type="checkbox" name="накопичувач" value="512GB SSD"> 512GB
SSD</label>
              <label><input type="checkbox" name="накопичувач" value="1TB SSD"> 1TB
SSD</label>
        </div>
        <div class="filter-section">
          <h3>Процесор</h3>
           <label><input type="checkbox" name="процесор" value="Intel Core i7-13700K"> Intel
Core i7-13700K</label>
          <label><input type="checkbox" name="процесор" value="AMD Ryzen 5 7600X"> AMD
Ryzen 5 7600X</label>
           <label><input type="checkbox" name="процесор" value="Intel Core i9-13900K"> Intel
Core i9-13900K</label>
          <label><input type="checkbox" name="процесор" value="AMD Ryzen 9 7950X"> AMD
Ryzen 9 7950X</label>
        </div>
        <div class="filter-section">
          <h3>Відеокарта</h3>
              <label><input type="checkbox" name="відеокарта" value="NVIDIA RTX 4060">
NVIDIA RTX 4060</label>
          <label><input type="checkbox" name="відеокарта" value="AMD RX 6700"> AMD RX
6700</label>
              <label><input type="checkbox" name="відеокарта" value="NVIDIA RTX 4090">
NVIDIA RTX 4090</label>
              <label><input type="checkbox" name="відеокарта" value="NVIDIA GTX 1660">
NVIDIA GTX 1660</label>
        </div>
    `;
    document.getElementById("filter").style.display = "block";
    // Додаємо обробники подій для чекбоксів
    const checkboxes = document.querySelectorAll('#filter-content input[type="checkbox"]');
    checkboxes.forEach(checkbox => {
      checkbox.addEventListener('change', filterAssembledPCs);
    });
  }
function filterAssembledPCs() {
  const filters = {
```

```
        ціна: Array.from(document.querySelectorAll('input[name="ціна"]:checked')).map(input => input.value),
        рік: Array.from(document.querySelectorAll('input[name="рік"]:checked')).map(input => input.value),
        версія: Array.from(document.querySelectorAll('input[name="версія"]:checked')).map(input => input.value), оперативна_память: Array.from(document.querySelectorAll('input[name="оперативна_память"]:checked')).map(input => input.value), накопичувач: Array.from(document.querySelectorAll('input[name="накопичувач"]:checked')).map(input => input.value), процесор: Array.from(document.querySelectorAll('input[name="процесор"]:checked')).map(input => input.value), відеокарта: Array.from(document.querySelectorAll('input[name="відеокарта"]:checked')).map(input => input.value)
    };
    const productsToFilter = products["Каталог"]["Зібрані ПК"].товари;
    const filteredProducts = productsToFilter.filter(product => {
        // Фільтр за ціною
        const matchesPrice = filters.ціна.length === 0 || filters.ціна.some(filter => {
            if (filter === "40000+") return product.ціна >= 40000;
            let [min, max] = filter.split("-");
            return product.ціна >= parseInt(min) && product.ціна <= parseInt(max);
        });
        // Фільтр за роком
        const matchesYear = filters.рік.length === 0 || filters.рік.includes(product.рік?.toString());
        // Фільтр за версією
        const matchesVersion = filters.версія.length === 0 || filters.версія.includes(product.версія);
        // Фільтр за оперативною пам'яттю
        const matchesRam = filters.оперативна_память.length === 0 || filters.оперативна_память.some(filter => {
            if (product.оперативна_память) {
                const ramValue = product.оперативна_память.match(/\d+/); // Виділяємо числове значення
                return ramValue && ramValue[0] === filter.replace(/\D/g, ''); // Порівнюємо числові значення
            }

function filterLaptops() {
    const filters = {
        ціна: Array.from(document.querySelectorAll('input[name="ціна"]:checked')).map(input => input.value),
        рік: Array.from(document.querySelectorAll('input[name="рік"]:checked')).map(input => input.value),
        версія: Array.from(document.querySelectorAll('input[name="версія"]:checked')).map(input => input.value), діагональ_екрану: Array.from(document.querySelectorAll('input[name="діагональ_екрану"]:checked')).map(input => input.value),
        вага: Array.from(document.querySelectorAll('input[name="вага"]:checked')).map(input => input.value), процесор: Array.from(document.querySelectorAll('input[name="процесор"]:checked')).map(input => input.value), оперативна_память:
```

```
Array.from(document.querySelectorAll('input[name="оперативна_память"]:checked')).map(input
=>                input.value),                                                    накопичувач:
Array.from(document.querySelectorAll('input[name="накопичувач"]:checked')).map(input      =>
input.value)
  };
  const productsToFilter = products["Каталог"]["Ноутбуки"].товари;
  const filteredProducts = productsToFilter.filter(product => {
    // Фільтр за ціною
    const matchesPrice = filters.ціна.length === 0 || filters.ціна.some(filter => {
      if (filter === "50000+") return product.ціна >= 50000;
      let [min, max] = filter.split("-");
      return product.ціна >= parseInt(min) && product.ціна <= parseInt(max);
    });
    // Фільтр за роком
    const matchesYear = filters.рік.length === 0 || filters.рік.includes(product.рік?.toString());
    // Фільтр за версією
    const matchesVersion = filters.версія.length === 0 || filters.версія.includes(product.версія);
    // Фільтр за діагоналлю екрану
                const matchesScreenSize = filters.діагональ_екрану.length ===  0  ||
filters.діагональ_екрану.includes(product.діагональ_екрану);
    // Фільтр за вагою
    const matchesWeight = filters.вага.length === 0 || filters.вага.includes(product.вага);
    // Фільтр за процесором
                    const matchesProcessor = filters.процесор.length ===  0  ||
filters.процесор.includes(product.процесор);
    // Фільтр за оперативною пам'яттю
                  const matchesRam = filters.оперативна_память.length ===  0  ||
filters.оперативна_память.includes(product.оперативна_память);
    // Фільтр за накопичувачем
    const matchesStorage = filters.накопичувач.length === 0 || filters.накопичувач.some(filter =>
{
      if (product.накопичувач) {
        const storageValue = product.накопичувач.match(/\d+/); // Виділяємо числове значення
          return storageValue && storageValue[0] === filter.replace(/\D/g, ''); // Порівнюємо
числові значення
      }
function filterComponents(subCategory) {
  const filters = {
      ціна: Array.from(document.querySelectorAll('input[name="ціна"]:checked')).map(input =>
input.value),
       рік: Array.from(document.querySelectorAll('input[name="рік"]:checked')).map(input =>
input.value),
    память: Array.from(document.querySelectorAll('input[name="память"]:checked')).map(input
=>                input.value),                                                    інтерфейс:
Array.from(document.querySelectorAll('input[name="інтерфейс"]:checked')).map(input      =>
input.value),
     ядра: Array.from(document.querySelectorAll('input[name="ядра"]:checked')).map(input =>
input.value),
    сокет: Array.from(document.querySelectorAll('input[name="сокет"]:checked')).map(input =>
input.value),                                                                  форм_фактор:
Array.from(document.querySelectorAll('input[name="форм_фактор"]:checked')).map(input      =>
```

```
input.value),                                                                    швидкість:
Array.from(document.querySelectorAll('input[name="швидкість"]:checked')).map(input =>
input.value),
        тип:  Array.from(document.querySelectorAll('input[name="тип"]:checked')).map(input =>
input.value),
      обсяг: Array.from(document.querySelectorAll('input[name="обсяг"]:checked')).map(input =>
input.value)                                                                    потужність:
Array.from(document.querySelectorAll('input[name="потужність"]:checked')).map(input =>
input.value),                                                                   сертифікація:
Array.from(document.querySelectorAll('input[name="сертифікація"]:checked')).map(input =>
input.value),
       розмір: Array.from(document.querySelectorAll('input[name="розмір"]:checked')).map(input
=> input.value),
      колір: Array.from(document.querySelectorAll('input[name="колір"]:checked')).map(input =>
input.value),
        бренд:  Array.from(document.querySelectorAll('input[name="бренд"]:checked')).map(input
=> input.value)
   };
   const productsToFilter = products["Каталог"]["Комплектуючі"].товари[subCategory].товари;
   const filteredProducts = productsToFilter.filter(product => {
      // Фільтр за ціною
      const matchesPrice = filters.ціна.length === 0 || filters.ціна.some(filter => {
         if (filter === "50000+") return product.ціна >= 50000;
         let [min, max] = filter.split("-");
         return product.ціна >= parseInt(min) && product.ціна <= parseInt(max);
      });
      // Фільтр за роком
      const matchesYear = filters.рік.length === 0 || filters.рік.includes(product.рік?.toString());
      // Фільтр за пам'яттю (для відеокарт)
      const matchesMemory = filters.память.length === 0 || filters.память.some(filter => {
         if (product.память) {
            const memoryValue = product.память.match(/\d+/); // Виділяємо числове значення
              return memoryValue && memoryValue[0] === filter.replace(/\D/g, ''); // Порівнюємо
числові значення
         }
function renderFilteredProducts(filteredProducts, category, subCategory = '') {
   const content = document.getElementById("content");
   content.innerHTML = "";
   let contentHTML = `<h2>${subCategory || category}</h2>`;
   contentHTML += `<div class="product-grid">`;
   contentHTML += filteredProducts.map((product, index) => `
      <div class="product-item">
          <img src="${product.зображення}" alt="${product.назва}" style="width: 100px; height:
100px;">
         <h3>${product.назва}</h3>
         <p>${product.ціна} грн</p>
         <div class="product-actions">
                        <button onclick="openProduct('${category}', '${subCategory || ''},
${product.originalIndex || index})">Детальніше</button>
            <button class="favorite-button" onclick="toggleFavorite('${category}', '${subCategory ||
''}', ${product.originalIndex || index})">
```

```
                ${favorites.some(fav => fav.назва === product.назва) ? '❤️' : '🤍'}
            </button>
        </div>
    </div>
  `).join('');
  contentHTML += `</div>`;
  content.innerHTML = contentHTML;
}
function searchProducts() {
  const searchBox = document.getElementById("search-box");
  const searchResults = document.getElementById("search-results");
  const query = searchBox.value.toLowerCase();

  if (query.length === 0) {
    searchResults.style.display = "none";
    return;
  }

function openProduct(category, subCategory, index) {
  const content = document.getElementById("content");
  const filter = document.getElementById("filter");
  content.classList.add('hidden');
  setTimeout(() => {
    let продукт;
    if (subCategory) {
      продукт = products["Каталог"][category].товари[subCategory].товари[index];
    }
function toggleTab(tabName) {
  // Отримуємо всі вкладки та кнопки вкладок
  const tabs = document.querySelectorAll('.tab-content');
  const tabButtons = document.querySelectorAll('.tab-button');
  // Закриваємо всі вкладки та знімаємо активний клас з кнопок
  tabs.forEach(tab => {
    tab.classList.remove('active');
  });
  tabButtons.forEach(button => {
    button.classList.remove('active');
  });
  // Знаходимо вкладку, яку потрібно показати
  const tabToShow = document.getElementById(tabName);
  if (tabToShow) {
    tabToShow.classList.add('active');
  }
  // Знаходимо кнопку, яка відповідає цій вкладці, і робимо її активною
                    const activeButton = document.querySelector(`.tab-
button[onclick="toggleTab('${tabName}')"]`);
  if (activeButton) {
    activeButton.classList.add('active');
  }
}
function renderHomeRecommendations() {
```

```
    const recommendationsContainer = document.getElementById("home-recommendations");
    if (!recommendationsContainer) return;
    const allProducts = [];
    for (let category in products["Каталог"]) {
        const categoryData = products["Каталог"][category];
        if (Array.isArray(categoryData.товари)) {
            categoryData.товари.forEach((product, index) => {
                allProducts.push({ ...product, category, index });
            });
        } else if (typeof categoryData.товари === "object") {
            for (let subCategory in categoryData.товари) {
                const subCategoryData = categoryData.товари[subCategory];
                if (Array.isArray(subCategoryData.товари)) {
                    subCategoryData.товари.forEach((product, index) => {
                        allProducts.push({ ...product, category, subCategory, index });
                    });
                }
            }
        }
    }
    const randomRecommendations = [];
    for (let i = 0; i < 10; i++) {
        const randomIndex = Math.floor(Math.random() * allProducts.length);
        randomRecommendations.push(allProducts[randomIndex]);
    }
function renderProductRecommendations(recommendations) {
    const recommendationsContainer = document.querySelector('.recommendations');
    recommendationsContainer.innerHTML = '';

    // Відображаємо лише 4 товари
    const limitedRecommendations = recommendations.slice(0, 4);
    limitedRecommendations.forEach(product => {
        recommendationsContainer.innerHTML += `
                    <div   class="recommendation-item"   onclick="openProduct('${product.category}',
'${product.subCategory || ''}', ${product.originalIndex || 0})">
            <img src="${product.зображення}" alt="${product.назва}">
            <h3>${product.назва}</h3>
            <p>${product.ціна} грн</p>
        </div>
        `;
    });
}
function showTab(tabName) {
    const tabs = document.querySelectorAll('.tab-content');
    const tabButtons = document.querySelectorAll('.tab-button');
    // Закрити всі вкладки
    tabs.forEach(tab => {
        tab.classList.remove('active');
    });
    // Зняти активний клас з усіх кнопок
    tabButtons.forEach(button => {
```

```
        button.classList.remove('active');
    });
    // Показати потрібну вкладку
    const tabToShow = document.getElementById(tabName);
    if (tabToShow) {
        tabToShow.classList.add('active');
    }
    // Додати активний клас до кнопки
    const activeButton = document.querySelector(`.tab-button[onclick="showTab('${tabName}')"]`);
    if (activeButton) {
        activeButton.classList.add('active');
    }
}
function goBack() {
    console.log("Breadcrumbs before going back:", breadcrumbs);
    if (breadcrumbs.length > 1) {
        const previousPage = breadcrumbs[breadcrumbs.length - 2].page;
        console.log("Navigating to:", previousPage);
         navigate(previousPage, false); // Використовуємо false, щоб не додавати новий запис до
історії
    } else {
        console.log("Navigating to home");
        navigate("home", false); // Повертаємося на головну, якщо немає попередньої сторінки
    }
}
function addToCart(category, subCategory, index) {
    let продукт;
    if (subCategory) {
        продукт = products["Каталог"][category].товари[subCategory].товари[index];
    }
function toggleFavorite(category, subCategory, index) {
    let product;
    if (subCategory) {
        product = products["Каталог"][category].товари[subCategory].товари[index];
    } else {
        product = products["Каталог"][category].товари[index];
    }

    if (!product) {
        alert("Товар не знайдено!");
        return;
    }
    // Додаємо категорію та підкатегорію до товару
    product.category = category;
    product.subCategory = subCategory || '';
    const favoriteIndex = favorites.findIndex(fav => fav.назва === product.назва);
    if (favoriteIndex === -1) {
        favorites.push(product);
    } else {
        favorites.splice(favoriteIndex, 1);
    }
```

```
    updateFavorites();
    updateFavoriteButtons();
}
function updateProductPageFavoriteButton(category, subCategory, index) {
    const productPage = document.querySelector(".product-details");
    if (productPage) {
        const favoriteButton = productPage.querySelector(".favorite-button");
        if (favoriteButton) {
            const product = subCategory
                ? products["Каталог"][category].items[subCategory].items[index]
                : products["Каталог"][category].items[index];
             favoriteButton.innerHTML = favorites.some(fav => fav.name === product.name) ? '❤️' :
'🤍';
        }
    }
function updateCart() {
    const cartElement = document.getElementById("cart");
    const cartItems = document.getElementById("cart-items");
    const cartTotal = document.getElementById("cart-total");
    const cartIcon = document.getElementById("cart-count");
    cartItems.innerHTML = "";
    let total = 0;
    if (cart.length === 0) {
        cartElement.style.display = "none"; // Приховати кошик, якщо він порожній
        cartIcon.textContent = "0"; // Оновити індикатор кошика
        cartTotal.textContent = "Загальна сума: 0 грн"; // Оновити загальну суму
        return;
    }
function renderFullCart() {
    const fullCartItems = document.getElementById("full-cart-items");
    const fullCartTotal = document.getElementById("full-cart-total");
    fullCartItems.innerHTML = "";
    let total = 0;
    cart.forEach((item, index) => {
        total += item.ціна * item.кількість;
        fullCartItems.innerHTML += `
            <li>
              <img src="${item.зображення}" alt="${item.назва}" style="width: 50px; height: 50px;">
               <span>${item.назва} - ${item.ціна} грн</span>
               <button onclick="changeQuantity(${index}, -1)">➖</button>
               ${item.кількість}
               <button onclick="changeQuantity(${index}, 1)">➕</button>
               <button onclick="removeFromCart(${index})">❌</button>
                        <button   onclick="openProduct('${item.category}',   '${item.subCategory   ||   ''}',
${item.originalIndex || 0})">Перейти до товару</button>
            </li>`;
    });
    fullCartTotal.textContent = `Загальна сума: ${total} грн`;
}
function toggleCheckout() {
    const checkoutSection = document.getElementById("checkout");
```

```
    if (cart.length === 0) {
        alert("Кошик порожній! Додайте товари перед оформленням замовлення.");
        return;
    }
    if (checkoutSection.style.display === "none" || checkoutSection.style.display === "") {
        checkoutSection.style.display = "block"; // Показати секцію
    } else {
        checkoutSection.style.display = "none"; // Приховати секцію
    }
}
function updateCheckoutButton() {
    const checkoutButton = document.querySelector("#cart-footer button");
    if (cart.length === 0) {
        checkoutButton.disabled = true; // Зробити кнопку неактивною
        checkoutButton.textContent = "Кошик порожній";
    } else {
        checkoutButton.disabled = false; // Зробити кнопку активною
        checkoutButton.textContent = "Оформити замовлення";
    }
}
function clearCheckoutForm() {
    const form = document.getElementById("checkout-form");
    form.reset(); // Очистити всі поля форми
}
function clearCheckoutForm() {
    const form = document.getElementById("checkout-form");
    form.reset(); // Очистити всі поля форми
}
function navigateToFullCart() {
    toggleCart(); // Закрити спливаюче вікно кошика
    navigate("full-cart"); // Відкрити повноекранний кошик
}
function openProductFromCart(index) {
    const продукт = cart[index];
    const category = продукт.category;
    const subCategory = продукт.subCategory || '';
    let productIndex;
    if (subCategory) {
        productIndex = products["Каталог"][category].товари[subCategory].товари.findIndex(p =>
p.назва === продукт.назва);
    } else {
        productIndex = products["Каталог"][category].товари.findIndex(p => p.назва ===
продукт.назва);
    }
    if (productIndex !== -1) {
        openProduct(category, subCategory, productIndex);
    } else {
        alert("Товар не знайдено в каталозі!");
    }
}
function openProductFromFavorites(index) {
```

```
const product = favorites[index];
const category = product.category; // Категорія товару
const subCategory = product.subCategory || ''; // Підкатегорія товару (якщо є)
// Знаходимо індекс товару в каталозі
let productIndex;
if (subCategory) {
```