

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:
«Інтернет-магазин комп'ютерних комплектуючих»

Виконав студент групи ПЗ-21-5
СЛИНЮК Богдан Вадимович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2025р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СЛИНІЮК Богдан Вадимович**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Інтернет-магазин комп'ютерних комплектуючих»**,
затверджена Наказом закладу вищої освіти від **14.02.2025р., №61/с**

Вихідні дані для роботи:

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	27.02.2025	27.02.2025
2	Власенко О.В.	27.02.2025	27.02.2025
3	Власенко О.В.	27.02.2025	27.02.2025

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «бакалавр» присвячена розробці інтернет-магазину комп'ютерних комплектуючих. Робота складається з програмного продукту та пояснювальної записки. Текстова частина пояснювальної записки до кваліфікаційної роботи містить 54 сторінки, 39 ілюстрацій та 10 таблиць. Загальний обсяг — 71 сторінок.

Метою роботи є створення сучасного, функціонального та зручного інтернет-магазину, який дозволяє користувачам переглядати асортимент комп'ютерних комплектуючих, здійснювати покупки, оформлювати замовлення, а адміністраторам — керувати товарною базою.

У процесі роботи були сформульовані основні завдання для створення системи: розробка модулів для користувачів і адміністраторів, реалізація каталогу з можливістю фільтрації, створення системи кошика, оформлення замовлень і управління наявністю товарів.

Було проведено аналіз існуючих аналогів для визначення функціональних переваг і недоліків. Для розробки обрано клієнт-серверну архітектуру. Серверна частина реалізована на платформі Node.js з використанням фреймворку Express.js для створення API. Для зберігання інформації про товари, замовлення та користувачів використано документо-орієнтовану базу даних MongoDB, що забезпечує гнучкість у роботі зі структурою даних.

У роботі описано основні сценарії використання системи, логіку взаємодії між модулями, структуру бази даних, приклади реалізації ключових компонентів інтерфейсу, а також розглянуто особливості адаптивного дизайну для різних типів пристроїв.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ-МАГАЗИН, КОМП'ЮТЕРНІ КОМПЛЕКТУЮЧІ, NODE.JS, EXPRESS.JS, MONGODB, API.

					ІПЗ.КР.Б-121 - 25 - ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Слинюк Б.В.			Інтернет-магазин комп'ютерних комплектуючих			Лім.		Арк.	Аркушів	
Керівник		Власенко О.В.								3	54	
								ФІКТ, гр. ІПЗ-21-5				
Н. контр.												
Затвердив												

ABSTRACT

This Bachelor's qualification thesis is dedicated to the development of an online store for computer components. The work consists of a software product and an explanatory note. The textual part of the explanatory note to the qualification thesis contains 54 pages, 39 illustrations, and 10 tables. The total volume is 71 pages.

The objective of the work is to create a modern, functional, and user-friendly online store that allows users to browse the range of computer components, make purchases, place orders, and allows administrators to manage the product database.

In the course of the work, the main tasks for creating the system were formulated: the development of modules for users and administrators, the implementation of a catalog with filtering capabilities, the creation of a shopping cart system, order processing, and product availability management.

An analysis of existing analogues was conducted to identify functional advantages and disadvantages. A client-server architecture was chosen for the development. The server-side is implemented on the Node.js platform using the Express.js framework to create an API. For storing information about products, orders, and users, the document-oriented database MongoDB was used, which provides flexibility in working with the data structure.

The thesis describes the main use-case scenarios of the system, the logic of interaction between modules, the database structure, examples of the implementation of key interface components, and also considers the features of responsive design for various types of devices.

KEYWORDS: ONLINE STORE, COMPUTER COMPONENTS, NODE.JS, EXPRESS.JS, MONGODB, API.

		Слинюк Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ САЙТУ	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	10
1.3. Вибір архітектури платформи	15
1.4. Обґрунтування вибору інструментальних засобів	16
Висновки до першого розділу	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ САЙТУ З ПРОДАЖУ КОМП'ЮТЕРНИХ КОМПЕКТУЮЧИХ.....	20
2.1. Визначення варіантів використання та структури системи	20
2.2. Розробка бази даних	22
2.3. Проектування та реалізація алгоритмів роботи	27
2.4 Реалізація функціоналу	30
Висновки до другого розділу	32
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУЗЗ	
3.1 Порядок встановлення та налаштування системи.....	33
3.2 Структура інтерфейсу та порядок взаємодії з інтернет-магазином.....	34
3.3 Тестування програмного продукту	48
Висновки до третього розділу	50
ВИСНОВКИ.....	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	52
ДОДАТКИ.....	55

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

CRUD - основні операції, що використовуються для управління даними (Create , Read , Update , Delete)

ПЗ – Програмне забезпечення

MVC – Модель–вигляд–контролер — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

		Слинюк Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. Стрімкий розвиток інформаційних технологій та їх широке впровадження в освіту, бізнес і побут стимулюють стабільно високий попит на комп'ютерне обладнання та комплектуючі. У цих умовах інтернет-магазини стали ключовим каналом продажів, що дозволяє споживачам швидко знаходити, порівнювати та купувати необхідні товари. Водночас високий рівень конкуренції у сфері електронної комерції вимагає від розробників створення не лише естетично привабливих, а й технічно ефективних, надійних та швидких веб-рішень. Тому розробка сучасного інтернет-магазину з використанням продуктивних технологій, таких як Node.js, є актуальним і практично значущим завданням.

Метою випускної кваліфікаційної роботи є проектування та розробка повнофункціонального інтернет-магазину комп'ютерних комплектуючих, який забезпечуватиме зручний інтерфейс для покупців та ефективні інструменти для адміністрування.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Провести аналітичну роботу та обґрунтувати проєктні рішення: Проаналізувати ринок електронної комерції, дослідити конкурентів, обрати архітектуру системи та визначити стек технологій (мови програмування, фреймворки, база даних).

2. Спроекувати архітектуру та структуру даних: Розробити логічну та модульну структуру застосунку, визначити основні сценарії використання (use-cases) та спроекувати структуру бази даних для зберігання інформації про товари, користувачів та замовлення.

3. Розробити основний функціонал для користувачів: Реалізувати ключові модулі для покупців, включаючи каталог товарів, пошук, розширену фільтрацію, конфігуратор ПК, кошик та процес оформлення замовлення з інтеграцією платіжних систем.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

4. Реалізувати адміністративну панель та інтерфейси системи: Створити панель для адміністратора для управління товарами, категоріями, замовленнями та користувачами. Розробити адаптивний дизайн для всіх сторінок сайту, забезпечивши зручну взаємодію на настільних та мобільних пристроях.

5. Провести комплексне тестування системи: Перевірити працездатність усіх розроблених модулів, виявити та усунути можливі помилки, а також оцінити відповідність системи вимогам щодо зручності, продуктивності та надійності.

Об'єкт дослідження це процеси проєктування та розробки програмних рішень для електронної комерції.

Предмет дослідження це методи, моделі та інструментальні засоби, що використовуються для створення інтернет-магазину комп'ютерних комплектуючих на платформі Node.js.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ САЙТУ

1.1. Постановка задачі

На сьогоднішній день інтернет-магазини стали незамінним інструментом для бізнесу, особливо у сфері торгівлі комп'ютерними комплектуючими. Власний вебсайт для підприємця є не лише засобом залучення клієнтів, а й ключовим елементом конкурентоспроможності на високотехнологічному ринку. Наявність функціонального та зручного інтернет-магазину дозволяє компаніям розширювати аудиторію, покращувати взаємодію з клієнтами, підвищувати рівень продажів і забезпечувати стабільну присутність на ринку.

Метою створення інтернет-магазину комп'ютерних комплектуючих є розробка платформи, яка надає користувачам можливість легкої навігації, швидкого пошуку товарів, зручного оформлення замовлень і отримання актуальної інформації про продукцію. Окрім того, система повинна забезпечувати високий рівень безпеки даних та відповідати сучасним вимогам дизайну й функціональності.

Інтернет-магазин комп'ютерних комплектуючих зорієнтований на задоволення потреб досвідчених користувачів ПК, геймерів та ентузіастів, що збирають комп'ютери самостійно, а також підприємців, котрі прагнуть автоматизувати продажі. Окрім зручного користувацького інтерфейсу, важливою складовою є багатфункціональна адміністративна панель, що дає змогу керувати товарами, категоріями, замовленнями та відгуками. Вона міститиме модуль для додавання, редагування й видалення позицій каталогу, інструменти обробки замовлень з можливістю зміни їх статусу, а також систему модерації.

Для оперативної роботи адміністрації заплановано створення Telegram-бота, який миттєво надсилатиме сповіщення про нові замовлення, щойно вони будуть оформлені на сайті.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Реалізація проєкту передбачає кілька етапів. Спершу відбудеться проєктування дизайну та користувацьких сценаріїв: розробка логічного та зрозумілого вигляду головної сторінки, сторінок каталогу з фільтрами, кошика та оформлення замовлення. Далі — проєктування та розробка архітектури: створення серверної частини для обробки замовлень і зберігання даних, клієнтської частини з акцентом на швидкість та інтерактивність. На третьому етапі будуть реалізовані ключові функціональні компоненти: реєстрація й авторизація користувачів з обов'язковою верифікацією через пошту або телефон, управління каталогом, конфігуратор ПК та інструмент для порівняння товарів. Також буде проведена інтеграція з платіжною системою LiqPay для онлайн-оплат та з API "Нової Пошти" для зручного вибору відділення доставки. Завершальним кроком стане тестування: перевірка працездатності основних функцій, виправлення помилок і оптимізація продуктивності всієї системи.

1.2. Аналіз аналогів програмного продукту

Перед розробкою інтернет-магазину комп'ютерних комплектуючих важливо проаналізувати існуючі аналоги, які вже функціонують на ринку, для визначення їхніх сильних і слабких сторін. Це дозволить врахувати найкращі практики, уникнути можливих недоліків та створити конкурентоспроможний продукт.

Brain (рис. 1.1) – один з провідних українських ритейлерів електроніки, комп'ютерних комплектуючих та побутової техніки. Компанія активно розвиває омніканальну модель, поєднуючи онлайн-платформу з розгалуженою мережею фізичних магазинів та точок видачі по всій Україні [1].

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

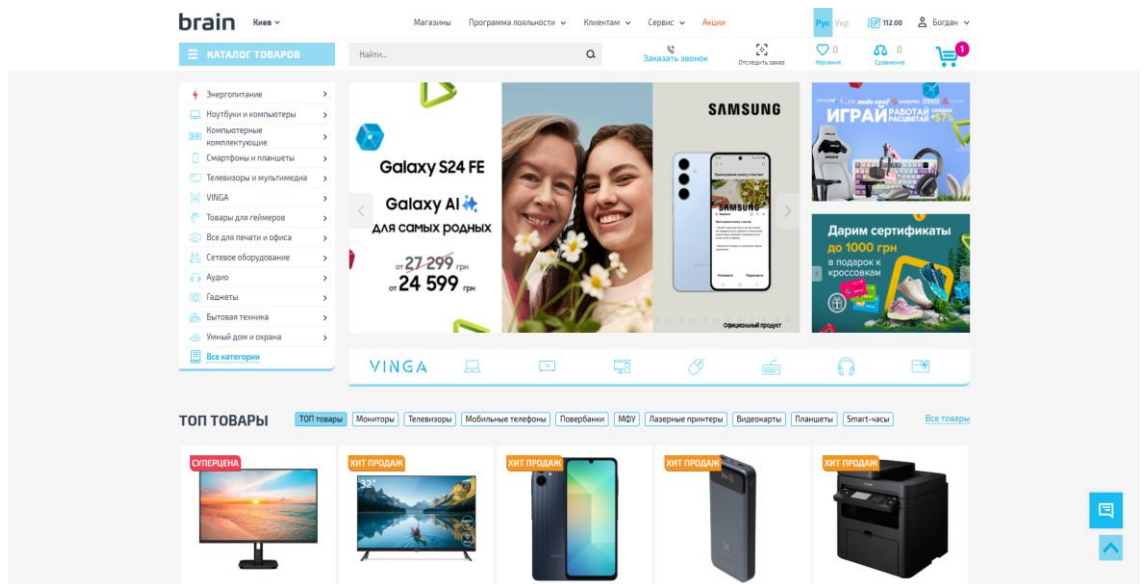


Рис. 1.1. Сайт Brain

Серед основних переваг інтернет-магазину варто відзначити надзвичайно широкий асортимент товарів, що виходить за межі лише ІТ-продукції; наявність фізичних магазинів, що дозволяє клієнтам оглянути товар перед покупкою, отримати консультацію та скористатися послугою самовивозу; розвинені додаткові сервіси, зокрема професійна збірка ПК та власні сервісні центри для гарантійного обслуговування; а також гнучкі умови оплати, що включають кредитування та оплату частинами від провідних банків. Водночас інтернет-магазин має й певні недоліки: інтерфейс сайту, особливо в мобільній версії, може здаватися дещо перевантаженим через велику кількість категорій та рекламних банерів; ціни на деякі позиції можуть бути вищими порівняно з меншими, вузькоспеціалізованими магазинами; крім того, у періоди високого навантаження час очікування відповіді від служби підтримки може збільшуватися.

ІТбох (рис. 1.2) – це великий український онлайн-ритейлер комп'ютерної техніки та електроніки. Платформа спрямована на широке коло споживачів, пропонуючи як ІТ-товари, так і продукцію для дому та відпочинку [2].

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

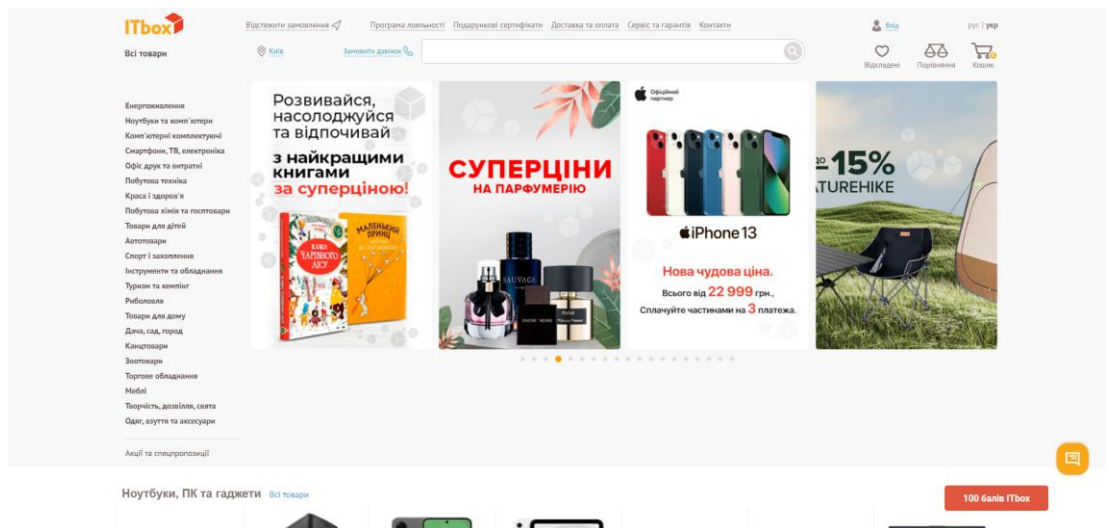


Рис. 1.2. Сайт ITbox

До сильних сторін платформи можна віднести глибоку інтеграцію з логістикою Rozetka, що надає покупцям доступ до величезної мережі точок видачі по всій країні; наявність унікальної програми «Як новий», яка пропонує уцінені товари (наприклад, з пошкодженою упаковкою або після ремонту), що є привабливим для бюджетно-орієнтованих клієнтів; а також активну спільноту користувачів з великою кількістю відгуків та запитань під товарами. Однак, існують і певні недоліки: через сильну інтеграцію бренд ITbox може сприйматися не як самостійний гравець, а як сателіт Rozetka, що розмиває його ідентичність; дизайн та структура сайту значною мірою наслідують материнську компанію, що робить користувацький досвід менш унікальним; окрім того, клієнтська підтримка є централізованою, що може призводити до стандартних затримок у вирішенні специфічних питань.

CompX (рис. 1.3) – це український ритейлер та виробник готових комп'ютерних систем, який має мережу спеціалізованих магазинів-шоурумів. Платформа орієнтована на геймерів та ентузіастів, пропонуючи широкий вибір комплектуючих та послуги з професійної збірки ПК [3].

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

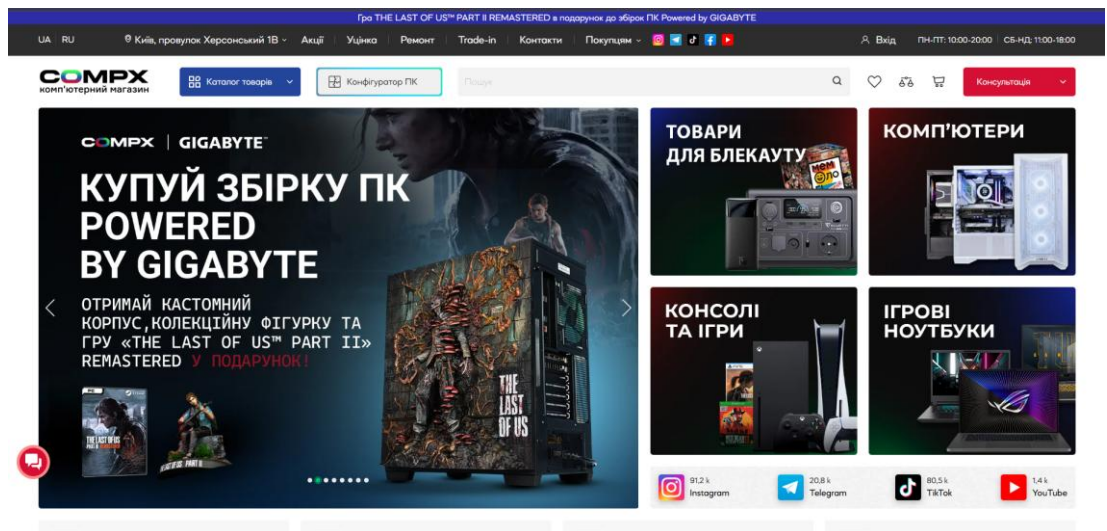


Рис. 1.3. Сайт CompX

Серед ключових переваг інтернет-магазину варто виокремити один з найкращих на ринку конфігураторів ПК, який дозволяє гнучко підбирати сумісні компоненти; сильний акцент на експертності та наданні професійних консультацій як онлайн, так і офлайн; а також наявність фізичних шоу-румів, де клієнти можуть наживо побачити готові збірки та оцінити якість компонентів. Утім, магазин має й низку недоліків: ціни на окремі комплектуючі можуть бути дещо вищими порівняно з великими універсальними ритейлерами; асортимент товарів є вузьким і сфокусованим переважно на продукції для ПК, що не задовольнить покупців, які шукають іншу електроніку; до того ж велика кількість технічної інформації та налаштувань у конфігураторі може бути складною для сприйняття користувачами-початківцями.

Telemart (рис. 1.4) – це один з найбільших в Україні спеціалізованих інтернет-магазинів, що фокусується на продажу комп'ютерних комплектуючих та ігрової периферії. Платформа спрямована насамперед на аудиторію геймерів та комп'ютерних ентузіастів, позиціонуючи себе не лише як магазин, а і як інформаційний хаб для спільноти [4].

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

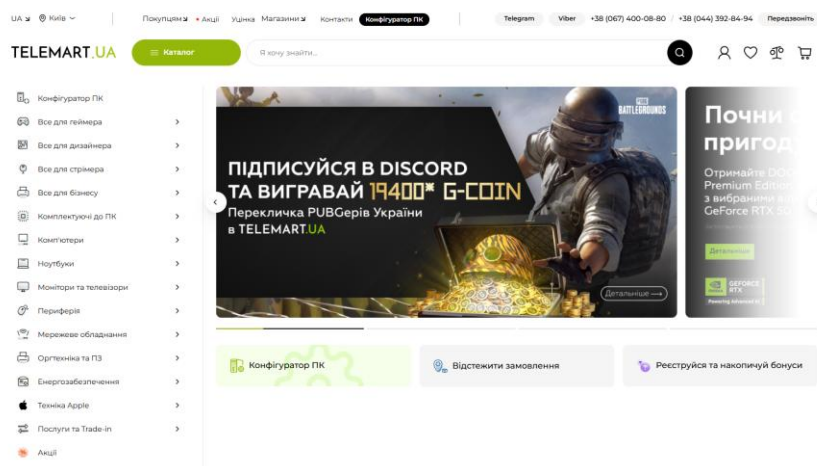


Рис. 1.4. Сайт Telemart

Серед беззаперечних переваг інтернет-магазину варто насамперед відзначити потужну інформаційну складову: на сайті представлені не лише товари, а й професійні огляди, новини індустрії та велика база відгуків, що перетворює його на повноцінний портал для ентузіастів. До плюсів також належить функціональний конфігуратор ПК, який допомагає підібрати сумісні компоненти, та наявність додаткових сервісів, як-от модернізація та діагностика комп'ютерів. Розвинена бонусна система та наявність шоу-румів у великих містах додатково підвищують лояльність клієнтів. Водночас, як програмний продукт, платформа має й недоліки. Через величезну кількість товарів, фільтрів та технічної інформації інтерфейс може здаватися складним та перевантаженим для користувачів-початківців. Яскраво виражений геймерський фокус у дизайні та комунікаціях може відштовхувати аудиторію, яка шукає техніку для офісних чи побутових потреб. Крім того, хоча мультимедійний контент присутній, він не завжди уніфікований: деякі товари мають детальні відеоогляди, тоді як інші обмежуються стандартними фото.

Таким чином, аналізуючи Brain, ITbox, CompX та Telemart, можна зробити висновок, що жоден із наявних на ринку продуктів не поєднує всі ключові переваги одночасно. Проект, що розробляється, вирішує проблему перевантаженості інтерфейсу, на відміну від Telemart чи Brain, пропонуючи інтуїтивно зрозумілу навігацію та чистий дизайн. Водночас, на відміну від деяких конкурентів, він із самого початку пропонує повноцінну адміністративну

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк. 14
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		

панель для легкого керування всіма аспектами магазину від товарів до відгуків. Як і провідні спеціалізовані гравці, проєкт містить конфігуратор ПК та інструмент порівняння товарів, але додатково підсилює зручність для адміністратора завдяки миттєвим сповіщенням у Telegram, а для клієнта — прямою інтеграцією з API "Нової Пошти" для вибору доставки та онлайн-оплатою через LiqPay[5]. Технічна архітектура на базі Node.js[6] та MongoDB забезпечує високу швидкодію та гнучкість у роботі з даними, що дозволяє легко масштабувати проєкт у майбутньому. Завдяки продуманій клієнтській частині та функціональному бекенду, розроблений продукт максимально враховує як потреби кінцевих користувачів, так і завдання ефективного управління бізнес-процесами. Порівняння інтернет-магазинів (Рис. 1.1)

Таблиця 1.1

Порівняльна характеристика інтернет-магазинів

Х-ка/ функція	Brain	ITbox	CompX	Telemart
Спеціалізований асортимент (ПК)	-	-	+	+
Наявність фізичних магазинів	+	+	+	+
Функціональний конфігуратор ПК	+	-	+	+
Інтуїтивно зрозумілий інтерфейс	-	-	-	-
Наявність експертного контенту	-	-	+	+
Інтеграція з "Новою Поштою"	+		+	+
Telegram-сповіщення для адміна	+	+	+	+
Онлайн-оплата (LiqPay)	-	-	-	-

1.3. Вибір архітектури платформи

Для розробки інтернет-магазину комп'ютерних комплектуючих було обрано класичну клієнт-серверну архітектуру, яка забезпечує ефективний розподіл ролей між клієнтською та серверною частинами. Серверна частина, організована за патерном MVC (Model-View-Controller), відповідає за всю бізнес-логіку, обробку запитів та взаємодію з базою даних. Клієнтська частина, у свою чергу, відповідає за представлення даних та користувацький інтерфейс.

Взаємодія між клієнтською та серверною частинами побудована на основі REST API. Такий підхід дозволяє чітко розмежувати логіку: клієнт надсилає

HTTP-запити на визначені ендпоінти, а сервер на базі Node.js та фреймворку Express.js їх обробляє, виконує необхідні операції та повертає дані у відповідь.

Клієнтська частина проєкту відповідає за інтерфейс користувача та побудована з використанням шаблонізатора Handlebars[7] та CSS-фреймворку Bootstrap. Вона реалізує відображення каталогу товарів, кошика, сторінок оформлення замовлення та особистого кабінету. Клієнт збирає дані від користувача — від вибраних позицій до особистої інформації для доставки і надсилає їх на сервер для обробки.

Серверна частина обробляє всі клієнтські запити, виконує бізнес-логіку (зокрема, розрахунок підсумкової вартості замовлення, перевірка наявності товарів) та забезпечує безпеку. Вся взаємодія з базою даних MongoDB відбувається виключно через серверну частину з використанням ODM-бібліотеки Mongoose. Ця бібліотека забезпечує зручну роботу з документами через моделі та схеми, а також надає механізми валідації та middleware для обробки даних, що значно підвищує надійність системи.

Застосування клієнт-серверної архітектури патерном MVC гарантує одразу кілька принципових переваг. По-перше, ефективність: взаємодія між клієнтом і сервером є стандартизованою, що спрощує розробку та підтримку. По - друге, масштабованість: чітке розділення логіки на Модель, Вигляд і Контролер дає змогу легко розширювати функціонал, додаючи нові можливості без радикальних змін у вже працюючих модулях. Нарешті, безпека: чутлива інформація та логіка роботи з базою даних інкапсульовані на сервері, тож ризик несанкціонованого доступу мінімізується. У такій архітектурі сервер відіграє ключову роль у реалізації бізнес-логіки, тоді як клієнтська частина лишається основним інструментом взаємодії з користувачем.

1.4. Обґрунтування вибору інструментальних засобів

Для створення інтернет-магазину комп'ютерних комплектуючих було обрано комплексний та гнучкий стек технологій на платформі Node.js, що у сукупності забезпечує високу продуктивність, безпеку та масштабованість. В

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

якості основи серверної частини виступає веб-фреймворк Express.js, який керує маршрутизацією та обробкою запитів, а за динамічне відображення сторінок на стороні сервера відповідає шаблонізатор express-handlebars.

Особливу увагу приділено безпеці та автентифікації. Для цього використано бібліотеку Passport.js, що реалізує як локальну стратегію (логін/пароль), так і вхід через Google. Безпека паролів гарантується їх хешуванням через bcryptjs, а управління сесіями — за допомогою express-session з адаптером connect-mongo для зберігання сесій у базі даних. Для захисту від поширених веб-вразливостей інтегровано бібліотеку Helmet.

Взаємодія з базою даних MongoDB реалізована через ODM-бібліотеку Mongoose, яка надає зручні інструменти для моделювання даних та валідації схем. Для реалізації ключового бізнес-функціоналу та комунікацій залучено набір спеціалізованих інструментів: nodemailer для надсилання транзакційних емейлів, Twilio для SMS-повідомлень, node-telegram-bot-api для сповіщень адміністратора, LiqPay та Stripe для прийому онлайн-платежів. Обробка завантаження файлів (наприклад, зображень товарів) реалізована через multer. Для забезпечення високої продуктивності та зручності розробки використовуються compression для стиснення HTTP-відповідей, morgan для логування запитів та node-cache для кешування даних.

Важливим етапом стало визначення бази даних, яка забезпечить зберігання та управління даними. Для цього було проведено аналіз двох основних підходів: реляційного PostgreSQL[7], MySQL[8], та документо-орієнтованого MongoDB[9], з урахуванням критеріїв гнучкості, продуктивності та масштабованості.

Порівняння обсягів даних - одним із важливих критеріїв стало визначення максимального обсягу даних, які можуть зберігатися у кожній із розглянутих СУБД (табл. 1.2).

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.2

Порівняння обмежень на розміри даних у різних СУБД

Параметр	PostgreSQL	MySQL	MongoDB
Максимальний розмір БД	Необмежено	Необмежено	Необмежено
Максимальний розмір таблиці/колекції	32 ТБ	256 ТБ	Необмежено
Максимальний розмір рядка/документа	1.6 ТБ	64 КБ	16 МБ

Як видно з таблиці, всі сучасні СУБД пропонують надійні базові механізми. Хоча вбудований захист від brute-force атак не є стандартною функцією MongoDB, цей аспект безпеки ефективно реалізується на рівні застосунку. У даному проєкті це забезпечується обмеженням кількості спроб входу та використанням надійних паролів, хешування яких виконується бібліотекою bcryptjs. Водночас MongoDB надає потужну та гнучку систему контролю доступу на основі ролей (RBAC), що є достатнім для вимог проєкту.

Функціональні можливості - для вибору оптимальної СУБД також було враховано функціональні можливості, що є ключовими для проєкту (табл. 1.3).

Таблиця 1.3

Порівняльна характеристика функціональних можливостей СУБД

Параметр	PostgreSQL	MySQL	MongoDB
Тип даних	Об'єктно-реляційний	Реляційний	Документоорієнтований
Підтримка JSON	Повна	Обмежена	Повна
Масштабованість	Висока	Середня	Дуже висока
Робота з транзакціями	Повна	Повна	Часткова
Гнучкість структури дани	Середня	Низька	Дуже висока

Порівняння безпеки - для інтернет-магазину безпека даних є критично важливим фактором, оскільки система працює з конфіденційною інформацією клієнтів (табл. 1.4).

Порівняння основних механізмів безпеки в розглянутих СУБД.

Параметр	PostgreSQL	MySQL	MongoDB
Аутентифікація користувачів	+	+	+
Захист від brute-force атак	+	+	-
Вбудоване шифрування даних	+	-	+
Система ролей і прав доступу (RBAC)	+	Обмежена	+

Після проведення аналізу було визначено, що MongoDB є оптимальним вибором для інтернет-магазину комп'ютерних комплектуючих. Ключовою перевагою стала гнучкість схеми даних: для каталогу, де товари різних категорій (процесори, відеокарти) мають унікальні набори характеристик, MongoDB дозволяє зберігати їх в одній колекції без створення надлишкових полів.

Другим важливим фактором є природна інтеграція з екосистемою Node.js: формат зберігання даних BSON легко перетворюється на JavaScript[10]-об'єкти, що спрощує та прискорює розробку. Висока продуктивність операцій читання є критичною для швидкого завантаження сторінок каталогу, а можливість горизонтального масштабування через шардинг закладає надійний фундамент для майбутнього зростання проєкту.

Висновки до першого розділу

У першому розділі було сформульовано мету та завдання для розробки інтернет-магазину комп'ютерних комплектуючих та визначено основні функціональні вимоги. Проведено аналіз існуючих аналогів, що дозволило виокремити ключові функції, зокрема каталог товарів, кошик, профіль користувача, конфігуратор ПК та адміністративну панель. Також було обґрунтовано вибір технологій для реалізації проєкту: Node.js для серверної частини та MongoDB як системи керування базами даних.

РОЗДІЛ 2. ПРОЕКТУВАННЯ САЙТУ З ПРОДАЖУ КОМП'ЮТЕРНИХ КОМПЕКТУЮЧИХ

2.1. Визначення варіантів використання та структури системи

Для забезпечення повноцінної роботи інтернет-магазину комп'ютерних комплектуючих було реалізовано низку модулів, які охоплюють управління товарним каталогом, обробку замовлень та взаємодію з покупцями. Платформа дозволяє зберігати та оновлювати інформацію про товари, користувачів, категорії та замовлення, що істотно спрощує процес онлайн-торгівлі. Основний функціонал включає перегляд та фільтрацію товарів, використання конфігуратора ПК, додавання до кошика та порівняння, оформлення замовлення, а також реєстрацію й керування особистим профілем. Для адміністрації реалізовано можливості керування асортиментом, категоріями, замовленнями, відгуками, користувачами та перегляд аналітики продажів.

У системі передбачено два основних типи користувачів: Покупець та Адміністратор. Роль Покупця охоплює як неавторизованих гостей, так і зареєстрованих користувачів. Будь-який відвідувач сайту, навіть без реєстрації, може вільно переглядати товари, використовувати фільтри та пошук, користуватися конфігуратором ПК, а також додавати товари до кошика чи порівняння. Проте, реєстрація надає додаткові переваги: зареєстрований користувач отримує доступ до особистого кабінету, де може керувати своєю інформацією та переглядати історію минулих замовлень. Адміністратор, у свою чергу, є користувачем з розширеними правами, що відповідає за повне управління платформою. До його повноважень входить керування каталогом товарів та категоріями, обробка всіх замовлень, модерація відгуків, а також управління акаунтами користувачів та доступ до статистики й аналітики сайту.¹

На (рис. 2.1) зображено побудовану діаграму базових варіантів використання для кожного типу користувачів.

		Слинюк Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

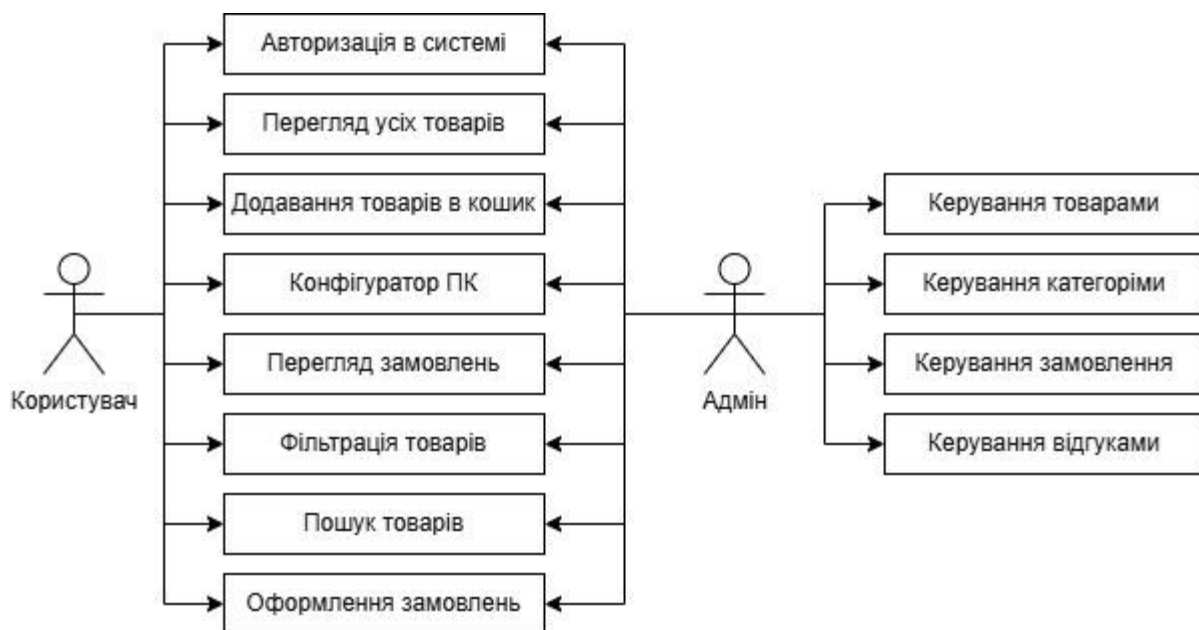


Рис 2.1. Варіанти використання сайту

У рамках розробки інтернет-магазину комп'ютерних комплектуючих було визначено низку ключових функціональних вимог, які мають забезпечити повноцінну роботу системи та задовольнити потреби як звичайних користувачів, так і адміністраторів. Передусім, реалізовано механізм авторизації з підтримкою реєстрації та входу з ролями "користувач" і "адміністратор". Кожен зареєстрований клієнт має можливість керувати власними даними. Система надає зручний інтерфейс для перегляду каталогу товарів із розширеними фільтрами за виробником (Intel, AMD, NVIDIA), ціновими діапазонами, а також за ключовими технічними характеристиками, такими як тип сокету для процесора, обсяг відеопам'яті чи форм-фактор материнської плати. Реалізовано пошук за ключовими словами і сортування результатів. Для комфортного шопінгу користувачам надано можливість додавати товари до кошика та порівняння, переглядати їх вміст і змінювати кількість чи видаляти позиції. Процес оформлення замовлення включає вибір місце доставки та оплату через платіжний сервіс LiqPay. Крім того, кожен клієнт має доступ до власної історії замовлень. Адміністраторам надано інструменти для управління товарними позиціями (описи, ціни, залишки), категоріями, а також функціонал для обробки

замовлень зі зміною їхнього статусу. Виконання цих вимог створює надійну та зручну платформу для онлайн-продажу.

Нефункціональні вимоги передбачають, що час навчання користувача роботі із системою не перевищуватиме однієї години. Час відгуку на типові дії має становити не більше десяти секунд, проте для ключових операцій бажаним є показник у секунд для забезпечення кращого користувацького досвіду. Система має бути надійною, забезпечувати доступність на рівні не менше 99% та підтримувати до ста одночасних користувачів на початковому етапі.

Аналіз функціональних вимог дозволив виокремити ключові сутності, необхідні для реалізації основної бізнес-логіки інтернет-магазину. До них належать користувачі (User), товари (Product), категорії (Category), замовлення (Order) та відгуки (Review).

На основі цього виділення було побудовано документо-орієнтовану модель даних, яка враховує взаємозв'язки між об'єктами та їхні атрибути. Ця модель стала фундаментом для розробки серверної частини застосунку з використанням ODM-бібліотеки Mongoose.

2.2. Розробка бази даних

Одним із ключових етапів реалізації інтернет-магазину комп'ютерних комплектуючих є проектування та створення документо-орієнтованої бази даних, яка забезпечує гнучке збереження, цілісність та ефективний доступ до інформації про користувачів, товари, замовлення та інші об'єкти системи. База даних повинна підтримувати всі функціональні вимоги, що були визначені на етапі аналізу, та забезпечувати масштабованість системи.

Для розробки бази даних була обрана система керування базами даних MongoDB, а опис структури колекцій виконано з використанням ODM-бібліотеки Mongoose. Такий підхід дозволяє декларативно задавати схеми даних, забезпечувати валідацію та зручно взаємодіяти з базою через JavaScript-об'єкти. У реалізованій моделі даних виділено такі основні колекції: Users для зберігання

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

інформації про користувачів, Products для товарів, Categories для ієрархічної структури каталогу, Orders для замовлень та Reviews для відгуків.

Використання Mongoose дозволило чітко реалізувати частину "Модель" (Model) у патерні MVC, інкапсулюючи всю логіку роботи з даними (пошук, створення, оновлення) у відповідних файлах моделей. Зв'язки між колекціями, наприклад, між Orders та Users, реалізовано за допомогою посилань на ObjectId, а для отримання пов'язаних даних при запитах активно використовується метод populate (), що спрощує розробку та зменшує кількість окремих запитів до бази даних. Крім того, для автоматичного відстеження часу створення та оновлення документів у всіх ключових схемах було активовано опцію timestamps, що є важливим для логування та аналізу даних.

Варто зазначити, що кошик для покупок реалізований на рівні сесії користувача для оптимізації, оскільки він є тимчасовим і не потребує постійного зберігання в базі даних. Адреса доставки зберігається як вбудований об'єкт безпосередньо в документі колекції Orders.

Ключова колекція Product має гнучку структуру: окрім базових полів (назва, ціна, залишки), вона містить масив посилань на зображення та вкладений об'єкт specs для технічних характеристик. Це дозволяє зберігати різні набори атрибутів для різних типів товарів (наприклад, сокет і кількість ядер для процесорів, обсяг і тип пам'яті для відеокарт) в одній колекції. Колекція Order, у свою чергу, включає посилання на користувача, масив придбаних товарів, загальну суму, статус замовлення та об'єкт customerInfo з даними клієнта для доставки.

Для наочного уявлення структури даних, які використовуються у системі, зображено ERD-діаграму(Рис 2.2) (діаграму "сутність-зв'язок") бази даних інтернет-магазину. Діаграма відображає основні колекції, їхні ключові поля та логіку організації інформації в системі.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

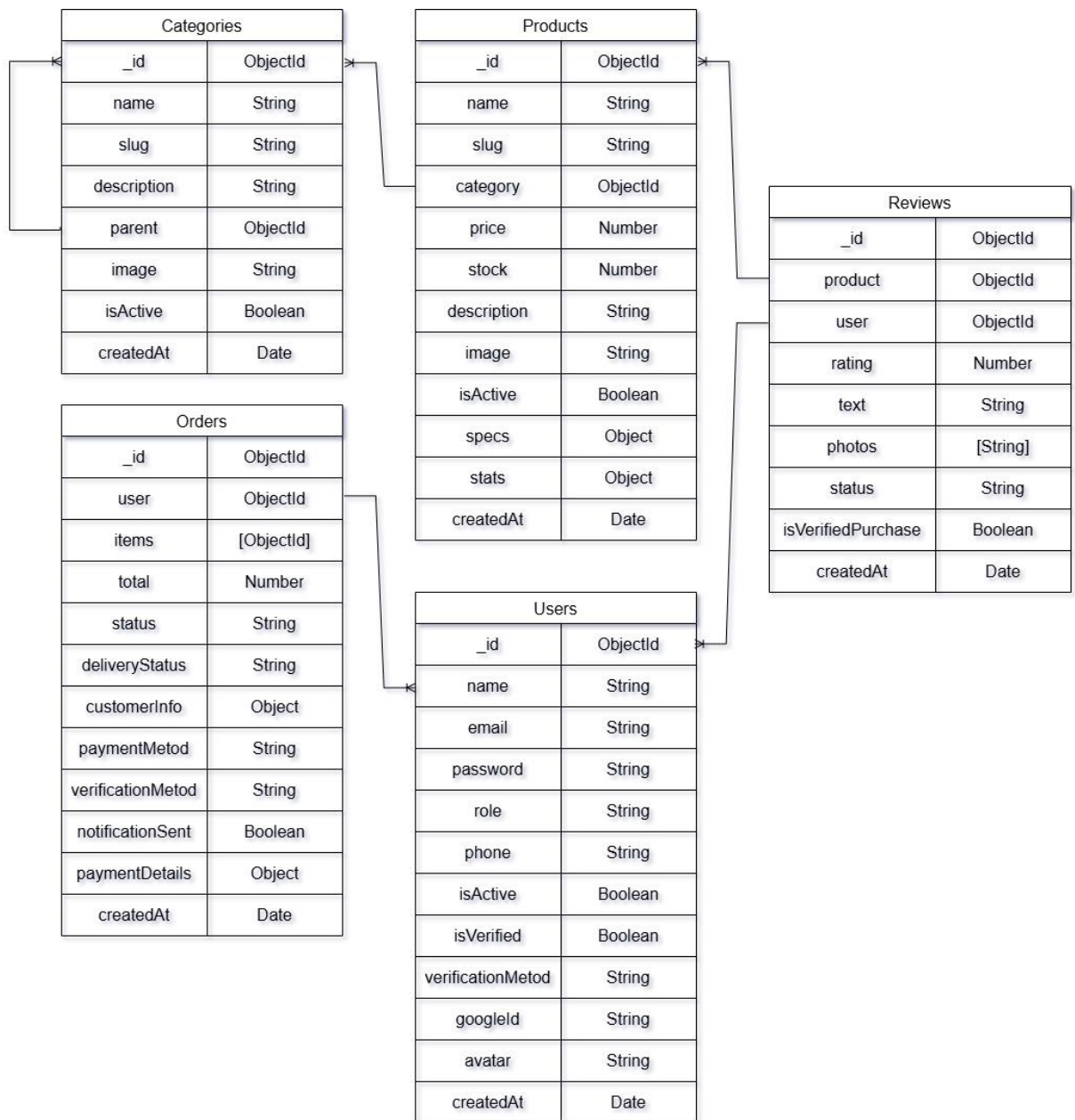


Рис. 2.2 ERD-діаграма “сутність-зв’язок”

Після загального представлення структури, у таблицях нижче наведено детальний опис кожної сутності (таблиці 2.1 – 2.5), включаючи назви полів, їхні типи даних у Mongoose, а також призначення. Зв’язки між колекціями реалізовано за допомогою посилань через поля типу ObjectId.

Таблиця 2.1

Структура колекції Users

Поле	Тип даних Mongoose	Обов'язкове?	Призначення
_id	ObjectId	+	Унікальний ідентифікатор користувача
name	String	+	Ім'я користувача

Продовження таблиці 2.1

email	String	+	Електронна пошта
password	String	+	Хешований пароль
role	String	+	Роль
phone	String	-	Контактний номер телефону
isActive	Boolean	+	Статус активності акаунту
isVerified	Boolean	+	Статус верифікації
verificationMethod	String	-	Метод, яким було верифіковано акаунт
googleId	String	-	Ідентифікатор для входу через Google
avatar	String	-	Посилання на зображення профілю
createdAt	Date	+	Дата та час створення акаунту

Таблиця 2.2

Структура колекції Categories

Поле	Тип даних Mongoose	Обов'язкове?	Призначення
id	ObjectId	+	Унікальний ідентифікатор категорії
name	String	+	Назва категорії
slug	String	+	Унікальний ідентифікатор для URL
description	String	-	Опис категорії
parent	ObjectId	-	Посилання на батьківську категорію
image	String	-	Посилання на зображення категорії
isActive	Boolean	+	Статус активності категорії
createdAt	Date	+	Дата та час створення категорії

Таблиця 2.3

Структура колекції Products

Поля	Тип даних в Mongoose	Обов'язкове?	Призначення
id	ObjectId	+	Унікальний ідентифікатор товару
name	String	+	Назва товару
slug	String	+	Унікальний ідентифікатор для URL
category	ObjectId	+	Посилання на основну категорію товару
subcategory	ObjectId	-	Посилання на підкатегорію
price	Number	+	Ціна товару
stock	Number	+	Кількість товару на складі
description	String	+	Детальний опис товару
image	String	+	Посилання на головне зображення товару
isActive	Boolean	+	Чи відображається товар у каталозі
specs	Object	-	Вкладений об'єкт з технічними характеристиками
stats	Object	-	Статистика (рейтинг, кількість відгуків)
createdAt	Date	+	Дата та час створення товару

Таблиця 2.4

Структура колекції Reviews

Поля	Тип даних в Mongoose	Обов'язкове?	Призначення
<u>id</u>	ObjectId	+	Унікальний ідентифікатор відгуку
product	ObjectId	+	Посилання на товар, до якого залишено відгук
user	Number	+	Посилання на автора відгуку
rating	String	+	Оцінка товару (від 1 до 5)
text	String	+	Текст відгуку
photos	[String]	-	Масив посилань на фото, додані до відгуку
status	String	+	Статус модерації ('pending', 'approved')
isVerifiedPurchase	Boolean	+	Чи залишено відгук після покупки товару
createdAt	Date	+	Дата та час створення відгуку

Таблиця 2.5

Структура колекції Orders

Назва поля	Тип даних в Mongoose	Обов'язкове?	Призначення
<u>id</u>	ObjectId	+	Унікальний ідентифікатор замовлення
user	ObjectId	-	Посилання на користувача (якщо авторизований)
items	[Object]	+	Масив об'єктів із замовленими товарами
total	Number	+	Загальна вартість замовлення
status	String	+	Статус замовлення
deliveryStatus	String	+	Статус доставки
customerInfo	Object	+	Вкладений об'єкт з даними покупця
paymentMethod	String	+	Обраний спосіб оплати
deliveryMethod	String	+	Обраний спосіб доставки
notificationSent	Boolean	+	Статус відправки смс телеграм
paymentDetails	Object	-	Вкладений об'єкт з деталями транзакції
createdAt	Date	+	Дата та час створення замовлення

Використовуючи ключові колекції, як-от Users, Products, Categories, Orders та Reviews, було сформовано логіку зберігання основної інформації про покупців, асортимент товарів та їхні замовлення. Рольову модель побудовано на основі поля role в колекції Users, що дозволяє розмежовувати доступ до адміністративних функцій відповідно до типу користувача. Завдяки колекції

Reviews реалізовано можливість для користувачів залишати відгуки та оцінки до товарів. Колекція Categories із підтримкою ієрархічної структури забезпечує гнучку класифікацію товарів у каталозі. Така документо-орієнтована структура дозволяє легко масштабувати систему, додавати нові характеристики до товарів без зміни схеми та є зручною основою для впровадження персоналізованих рекомендацій у майбутньому.

2.3. Проєктування та реалізація алгоритмів роботи

Для реалізації вебзастосунку було обрано класичну клієнт-серверну архітектуру, а структура серверної частини побудована на основі патерну MVC (Model-View-Controller). Такий підхід є галузевим стандартом для розробки складних веб-систем, оскільки він забезпечує чітке розділення відповідальності між трьома основними компонентами. Це дозволяє незалежно розробляти логіку роботи з даними (Модель), користувацький інтерфейс (Вигляд) та логіку керування (Контролер), що значно підвищує гнучкість, спрощує тестування окремих частин системи та полегшує її подальшу підтримку й масштабування.

Хоча проєкт реалізовано переважно у функціональному та модульному стилі, що є ідіоматичним для екосистеми Node.js, для наочності його архітектуру представлено у вигляді структурної схеми ключових логічних компонентів (рис. 2.3). Ці компоненти, хоч і не завжди є класами в класичному розумінні ООП, виконують аналогічні ролі, інкапсулюючи дані та бізнес-логіку. Основними будівельними блоками системи є Моделі Mongoose, що представляють собою об'єктні оболонки над колекціями MongoDB, Контролери, реалізовані як модулі з набором функцій для обробки маршрутів та Сервіси, які ізолюють логіку взаємодії із зовнішніми API. Такий підхід дозволяє досягти високого рівня модульності та слабкої зв'язності компонентів, що є ключовим для створення надійної системи.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

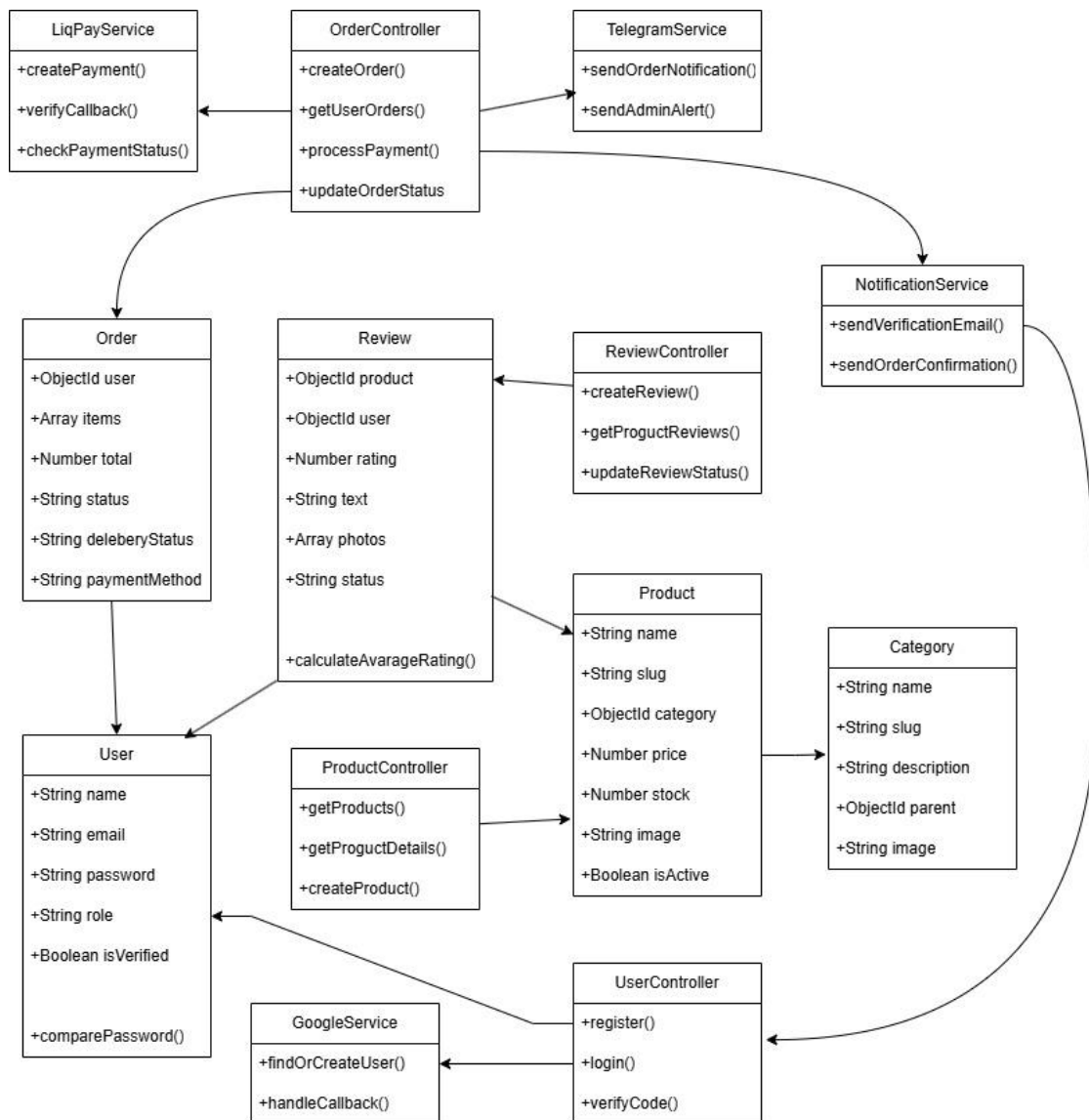


Рис. 2.3. Структурна схема основних компонентів системи

Ключовий алгоритм роботи системи полягає в обробці HTTP-запитів у рамках циклу "запит-відповідь". Коли клієнт (браузер) надсилає запит на певний URL, маршрутизатор Express.js [11] перехоплює його і передає відповідному контролеру. Перед цим запит може проходити через низку проміжних обробників (middleware), які виконують допоміжні завдання, наприклад, перевірку автентифікації. Контролер, отримавши запит, викликає необхідні методи з Моделі для отримання чи зміни даних у MongoDB [9]. Після отримання даних від Моделі, контролер передає їх у Вигляд (HBS-шаблон) для рендеринга фінальної HTML-сторінки. У випадку API-запитів, контролер повертає дані у форматі JSON.

Весь функціонал реалізовано з урахуванням ролей користувачів. Розмежування доступу побудовано на базі middleware-функцій, які перевіряють наявність активної сесії та роль користувача (user або admin) перед виконанням коду контролера. Такий механізм надійно захищає адміністративні маршрути та особисті дані користувачів від несанкціонованого доступу. Загальний алгоритм взаємодії користувача з системою, від входу на сайт до оформлення замовлення(Рис 2.4)

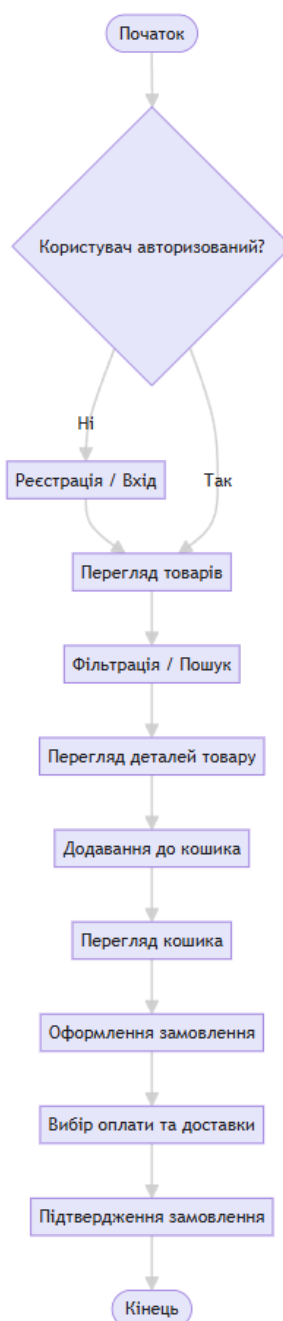


Рис. 2.4. Загальний алгоритм роботи інтернет-магазину

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Для більш глибокого технічного аналізу ключового бізнес-процесу - оформлення замовлення було розроблено діаграму послідовності (Рис. 2.5). Вона детально ілюструє ланцюжок взаємодій між сімома основними учасниками: Користувачем, Кошиком, Контролером замовлень, Базою даних, Платіжною системою, Системою сповіщень та Адміністратором.

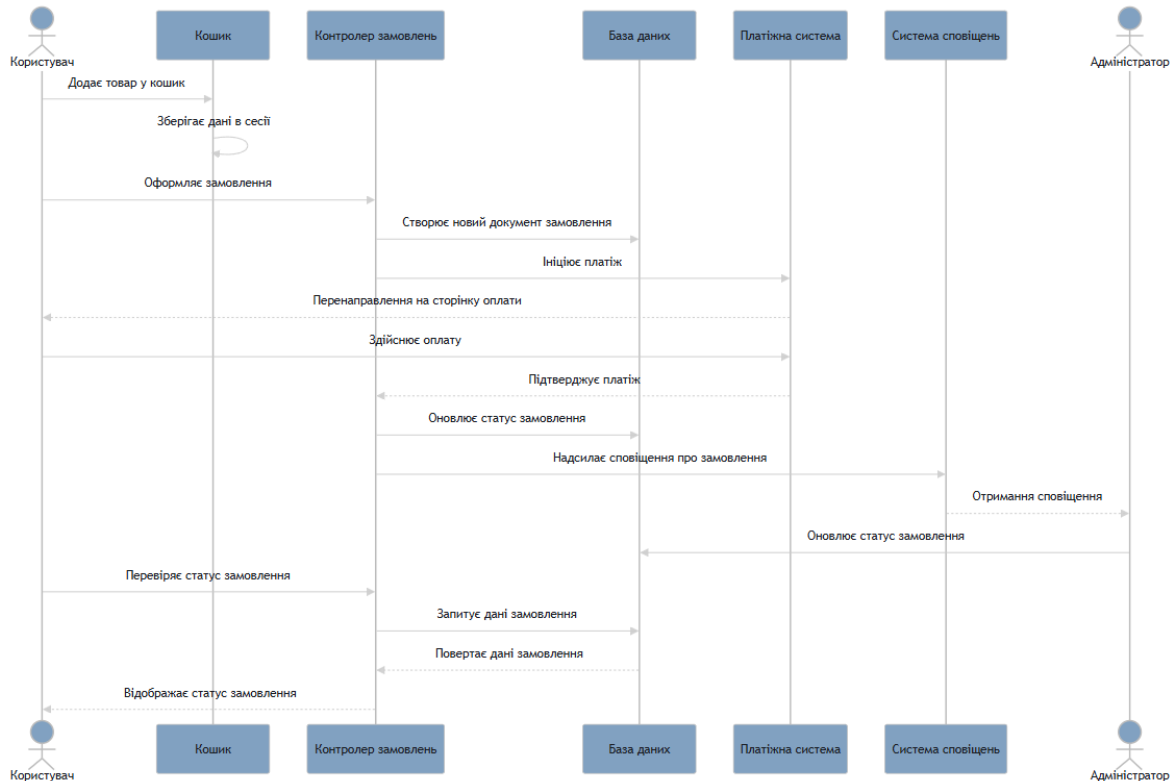


Рис. 2.5. Діаграма послідовності: оформлення замовлення

Процес, зображений на діаграмі, розпочинається з того, що Користувач додає товар до Кошика, який зберігається в сесії. Коли користувач переходить до оформлення та заповнює свої дані, Контролер замовлень валідує інформацію та створює новий документ у колекції Orders в Базі даних, присвоюючи йому початковий статус "pending". На наступному етапі система ініціює платіж, перенаправляючи користувача на сторінку Платіжної системи LiqPay. Після успішної оплати LiqPay надсилає підтвердження, контролер оновлює статус замовлення в базі даних на "paid" та очищує кошик у сесії. Паралельно, одразу після створення замовлення, Система сповіщень автоматично відправляє

Адміністратору повідомлення в Telegram. Цей детальний ланцюжок демонструє чіткий розподіл відповідальності між спроектованими компонентами.

2.4 Реалізація функціоналу

Реалізація інтернет-магазину комп'ютерних комплектуючих виконана на платформі Node.js з використанням фреймворку Express.js. Архітектура застосунку базується на патерні MVC, що забезпечує чітке розділення бізнес-логіки, даних та їх представлення.

Дані зберігаються у документо-орієнтованій базі даних MongoDB, а для взаємодії з нею використовується ODM-бібліотека Mongoose. Такий підхід дозволяє працювати з даними у вигляді звичних JavaScript-об'єктів. Для кожної колекції визначено строгу схему, яка описує структуру документів та правила валідації.

Особливу увагу приділено автоматизації логіки на рівні моделі за допомогою проміжних обробників middleware Mongoose. Наприклад, `pre('save')` для моделі User автоматично хешує пароль за допомогою `bcryptjs` перед збереженням у базу даних. Для моделі Review `post('save')` автоматично перераховує середній рейтинг товару після додавання нового відгуку. Для оптимізації запитів використовуються індекси, зокрема текстові для пошуку та композитні для прискорення фільтрації.

Автентифікація користувачів реалізована за допомогою бібліотеки Passport.js, яка є гнучким middleware для Express.js. Щоб забезпечити зручність для різних користувачів, у проєкті було налаштовано дві основні стратегії. Перша, локальна стратегія (`passport-local`), відповідає за класичний вхід за email та паролем. При такій спробі авторизації система спочатку знаходить користувача в базі даних за його електронною поштою, а потім, за допомогою бібліотеки `bcryptjs`, безпечно порівнює наданий пароль із хешованим паролем, що зберігається в системі. Друга, стратегія Google (`passport-google-oauth20`), надає можливість для швидкої та безпечної автентифікації через існуючий обліковий запис Google, що значно спрощує процес входу для користувачів.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

Після успішної автентифікації дані про користувача (ID, роль) серіалізуються та зберігаються в сесії. Управління сесіями здійснюється за допомогою express-session, а для їх постійного зберігання використовується connect-mongo, що записує дані сесій у MongoDB. На кожному наступному запиті passportdeserializeUser отримує дані користувача з бази за ID із сесії, що дозволяє системі "впізнавати" користувача та перевіряти його права доступу за допомогою спеціальних middleware.

Інтеграція з платіжною системою - Фінансові операції реалізовані через інтеграцію з українською платіжною системою LiqPay. Процес обробки платежу складається з кількох етапів. Спочатку на сервері для кожного замовлення генерується унікальний набір параметрів, що включає публічний ключ, суму, валюту, опис, order_id, а також адреси для повернення клієнта та отримання серверного сповіщення.

Ці параметри кодуються у формат Base64 та підписуються за допомогою приватного ключа з використанням алгоритму SHA1. Отримані data та signature передаються на клієнтську частину, де вбудовуються у HTML-форму, яка автоматично перенаправляє користувача на сторінку оплати LiqPay. Після завершення транзакції LiqPay надсилає на вказаний server_url зворотний запит callback з результатом платежу. Серверна частина валідує підпис цього запиту, щоб переконатися в його автентичності, після чого оновлює статус замовлення в базі даних та надсилає адміністратору сповіщення про успішне замовлення в Telegram.

Висновки до другого розділу

У другому розділі було спроектовано архітектуру інтернет-магазину комп'ютерних комплектуючих на основі патерну MVC, визначено ролі користувачів та логіку їхньої взаємодії. Було розроблено структуру документо-орієнтованої бази даних MongoDB та описано ключові колекції. Також деталізовано алгоритми, відповідальні за обробку замовлень, автентифікацію користувачів та інтеграцію з платіжними сервісами.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1 Порядок встановлення та налаштування системи

Діаграма розгортання виступає як візуальне уявлення конфігурації вузлів обробки та компонентів системи під час їх встановлення. Це один із типів структурних діаграм, який дозволяє відобразити фізичне розміщення програмних компонентів у середовищі розгортання. На рисунку 3.1 представлено діаграму розгортання для інтернет-магазину комп'ютерних комплектуючих. Система реалізована за архітектурою клієнт-сервер, з чітким поділом на клієнтську частину, сервер застосунку, базу даних і зовнішні сервіси.

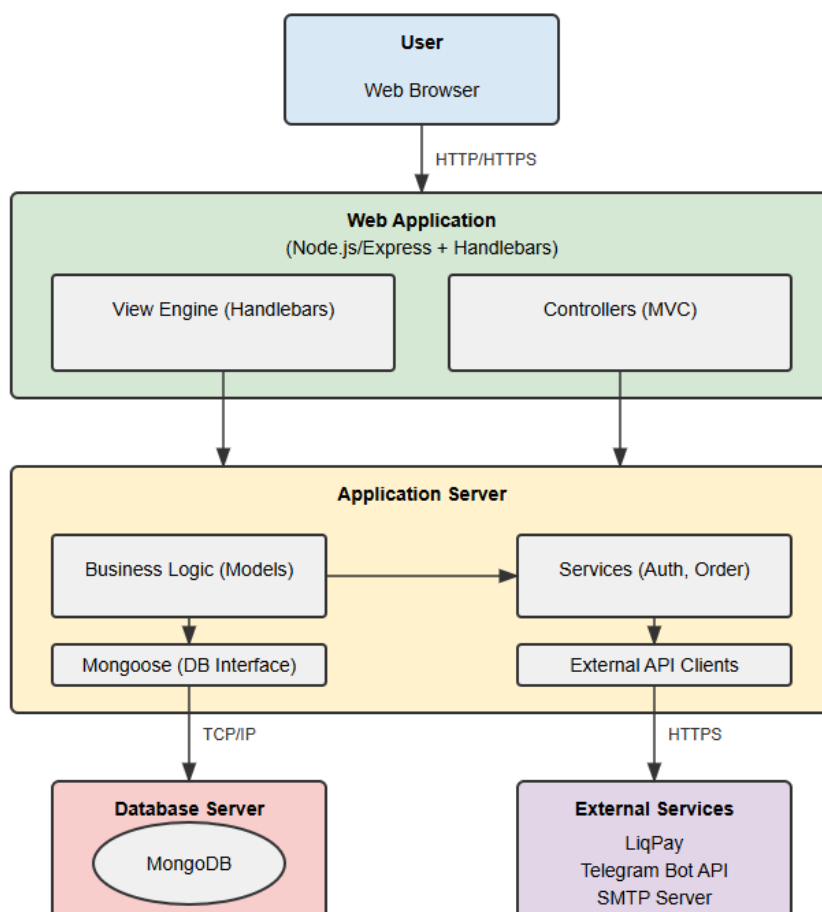


Рис. 3.1 Діаграму розгортання для інтернет-магазину комп'ютерних комплектуючих

Процес розгортання системи передбачає взаємодію кількох ключових вузлів. Клієнтська частина — це веббраузер користувача, який через захищений протокол HTTPS надсилає запити до сервера застосунку.

Сервер застосунку розгортається на хмарній платформі типу PaaS. Таке середовище автоматично керує інфраструктурою, а розгортання нових версій коду відбувається через систему контролю версій Git [12]. Додаток на Node.js та Express.js запускається в цьому середовищі, використовуючи змінні оточення (.env) для конфігурації. Цей вузол виконує всю бізнес-логіку, обробляє API-запити та взаємодіє з іншими сервісами. Для підвищення безпеки та продуктивності використовуються бібліотеки Helmet та Compression.

Сервер бази даних функціонує як окремий вузол. База даних MongoDB розгортається на спеціалізованій хмарній платформі, такій як MongoDB Atlas. Сервер застосунку з'єднується з нею через захищений протокол TCP/IP, використовуючи рядок підключення MONGO_URI (який знаходиться в env). Сесії користувачів також зберігаються в цій базі даних за допомогою MongoStore.

Для реалізації додаткового функціоналу система інтегрується із зовнішніми сервісами через HTTPS. До них належать платіжний шлюз LiqPay для обробки транзакцій, API Telegram для надсилання сповіщень адміністраторам та SMTP-сервіс для відправки електронних листів через Nodemailer.

3.2 Структура інтерфейсу та порядок взаємодії з інтернет-магазином

Для початку взаємодії з сайтом користувач має можливість авторизуватися, натиснувши кнопки «Вхід» або «Реєстрація» у верхній частині сторінки. Після цього відкривається форма авторизації, де користувач вводить свої облікові дані для доступу до особистого кабінету або може увійти за допомогою облікового запису Google(Рис 3.1.)

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

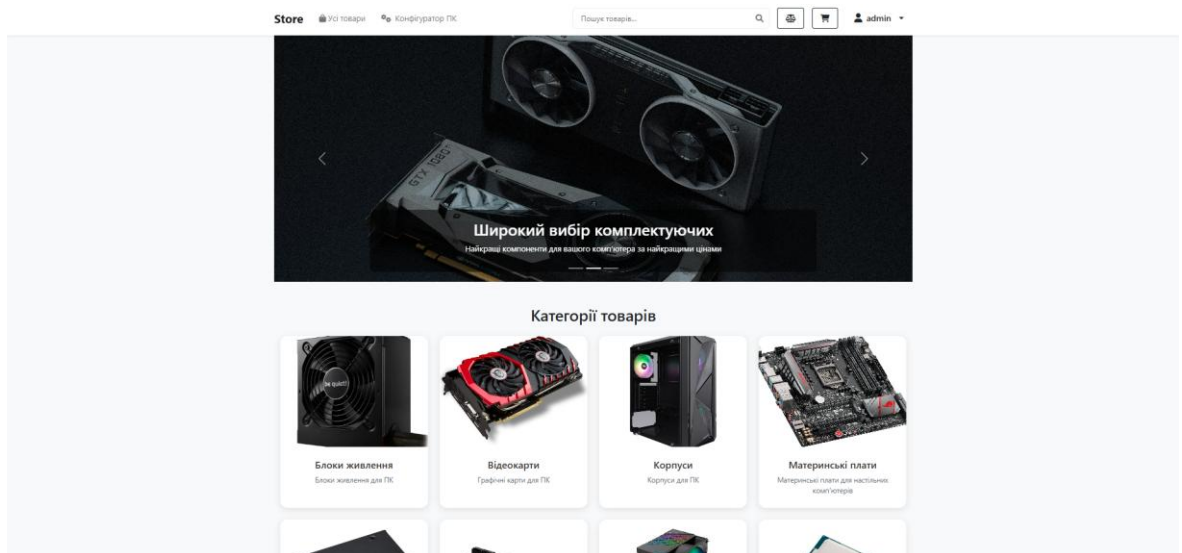


Рис. 3.2 Головна сторінка

Якщо користувач ще не має облікового запису, він може натиснути на кнопку “Зареєструватись”(Рис. 3.3.), що розташована під кнопкою входу через Google. Після цього відкриється форма реєстрації, де необхідно ввести ім'я користувача, електронну пошту, номер телефону, пароль, а також обрати спосіб верифікації (через email або номер телефону). У разі, якщо акаунт уже створено, можна скористатися кнопкою “Увійти” для входу(Рис. 3.4.)

Рис. 3.3. Форма реєстрації на сайті

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

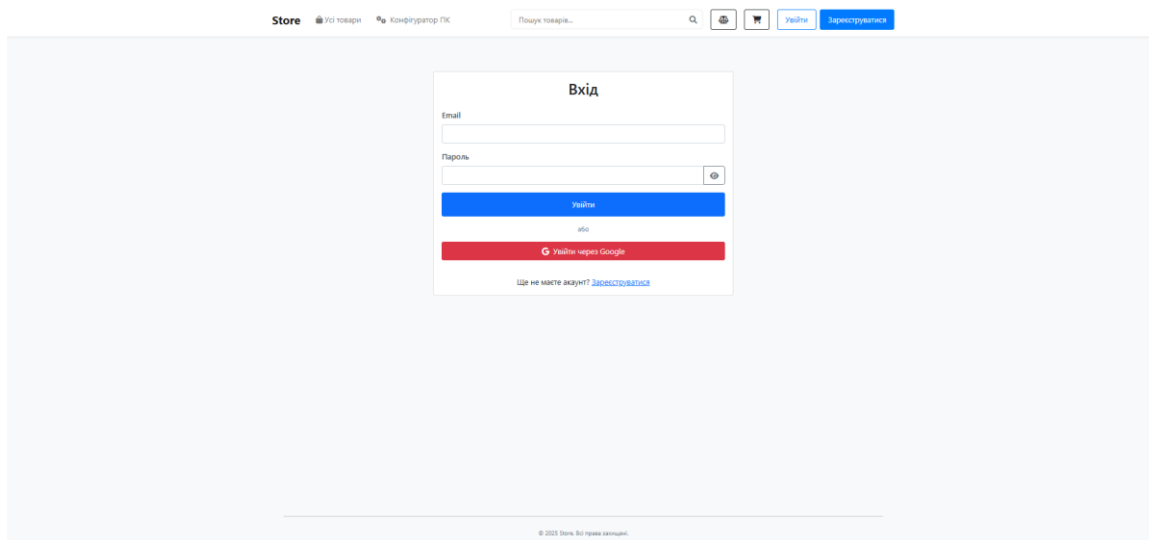


Рис. 3.4. Сторінка входу

Окрім можливості реєстрації чи входу, користувач також може ознайомитись з асортиментом товарів без авторизації. На головній сторінці відображаються всі категорії товарів з їх назвами та описами. Приклад інтерфейсу цієї сторінки наведено на рисунку 3.5.

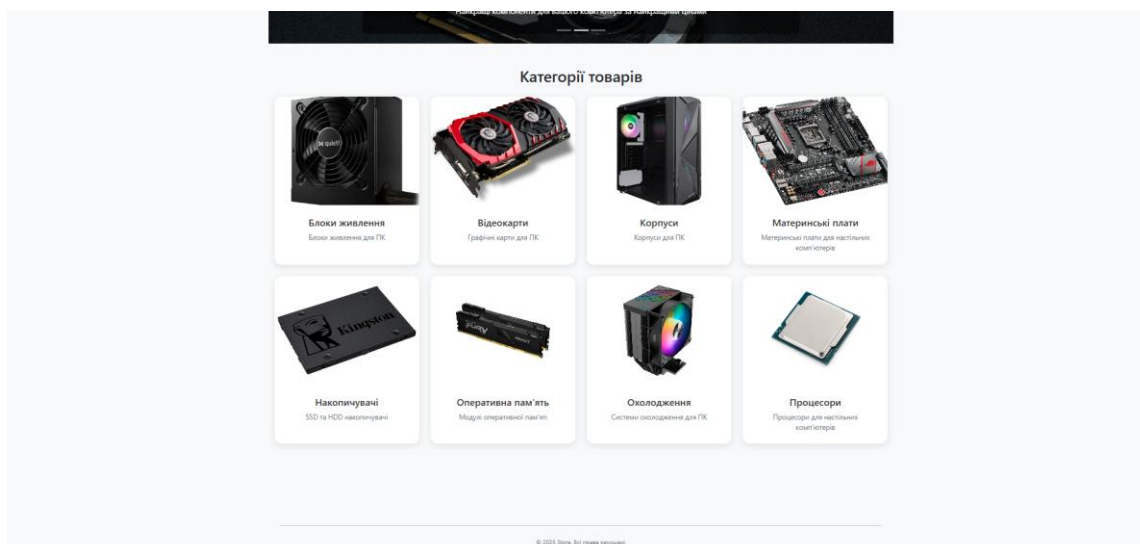


Рис. 3.5. Форма реєстрації на сайті

Далі можна обрати будь-яку категорію товарів, після чого відобразяться товари, що відповідають даній категорії. На картках товарів передбачена можливість додати товар до кошика, а також переглянути його детальніше. Також на картці вказано ціну, статус наявності та доступна функція фільтрації. (Рис 3.6.)

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

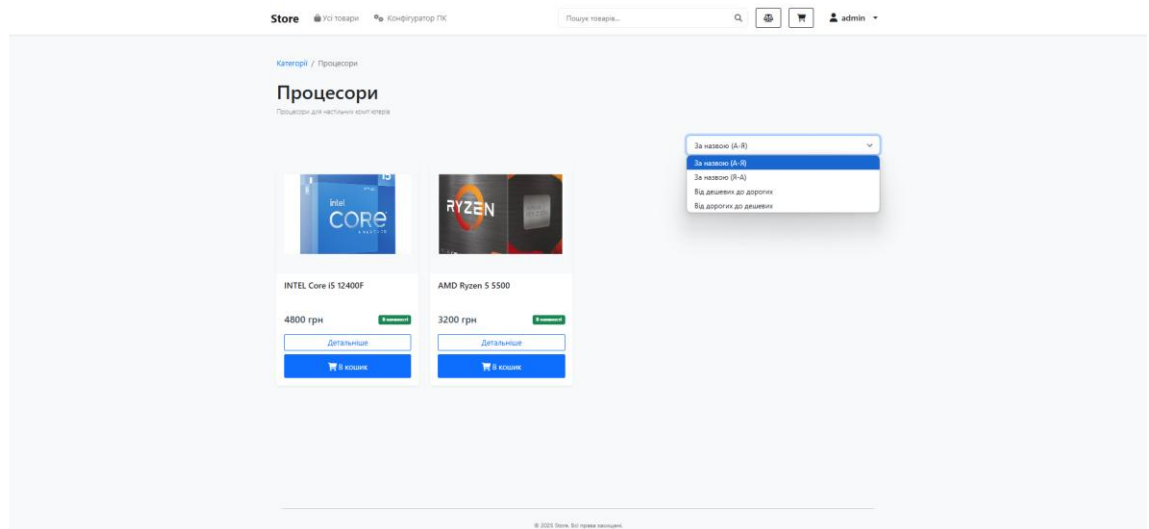


Рис 3.6 Товари категорії “Процесори”

Також є змога переглянути всі товари, які є в магазині, натиснувши кнопку “Усі товари”, що знаходиться у верхній частині сайту. На цій сторінці відображаються картки з назвами товарів, їх ціною, а також кнопки додавання до кошика та до порівняння (Рис. 3.7.).

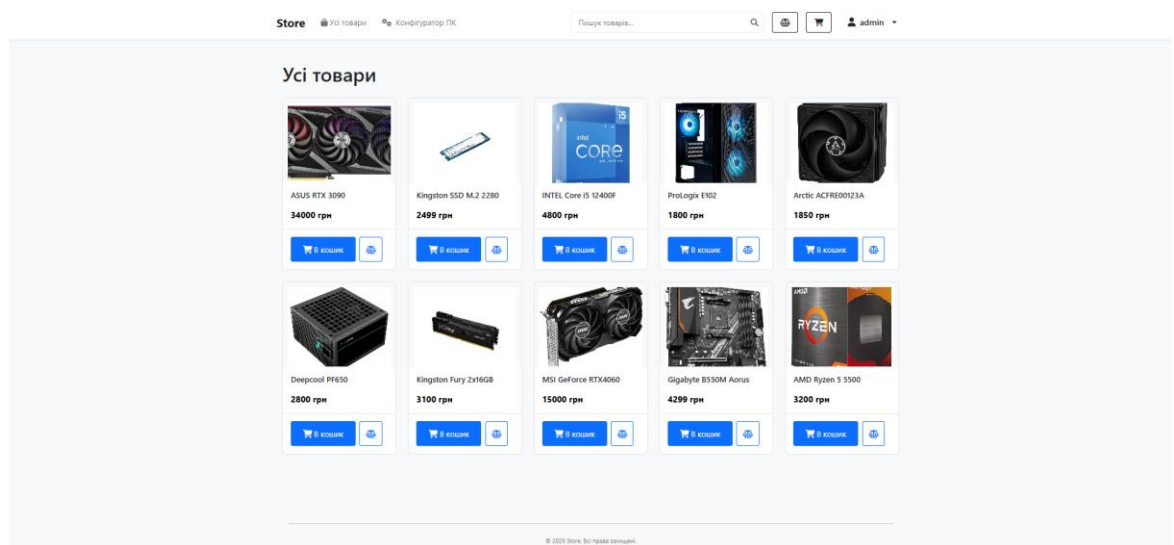


Рис 3.7 Сторінка “Усі товари”

Тепер можна натиснути на картку товару, щоб перейти до його детального опису (Рис. 3.8). На цій сторінці відображається фото обраного товару, цінник, статус наявності, кнопки додавання до кошика та до порівняння (Рис. 3.9), а також характеристики та опис. У самому низу сторінки можна переглянути відгуки (Рис. 3.10).

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

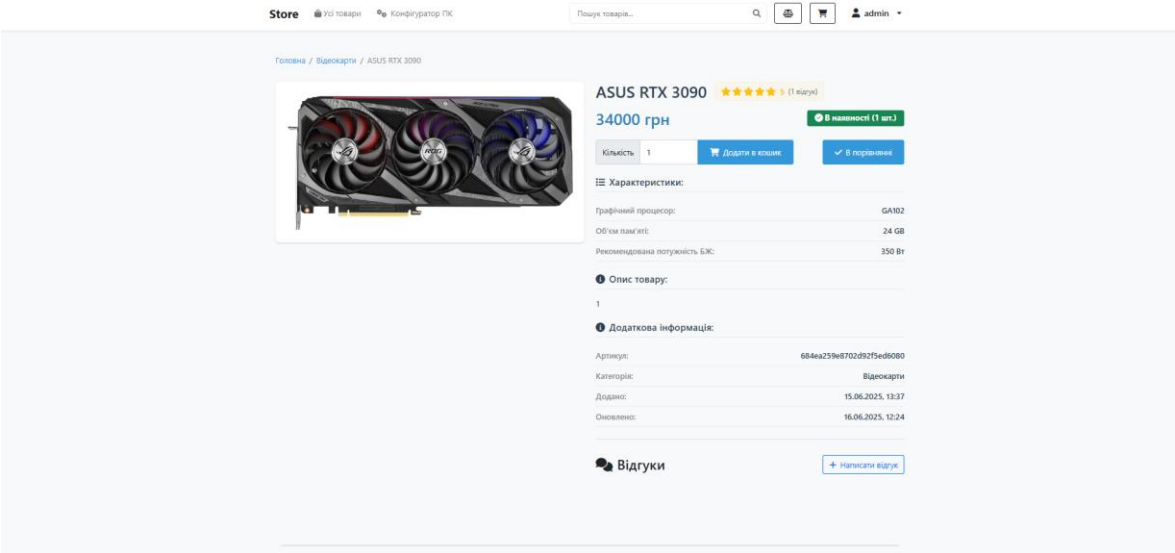


Рис. 3.8. Сторінка детального опису товару

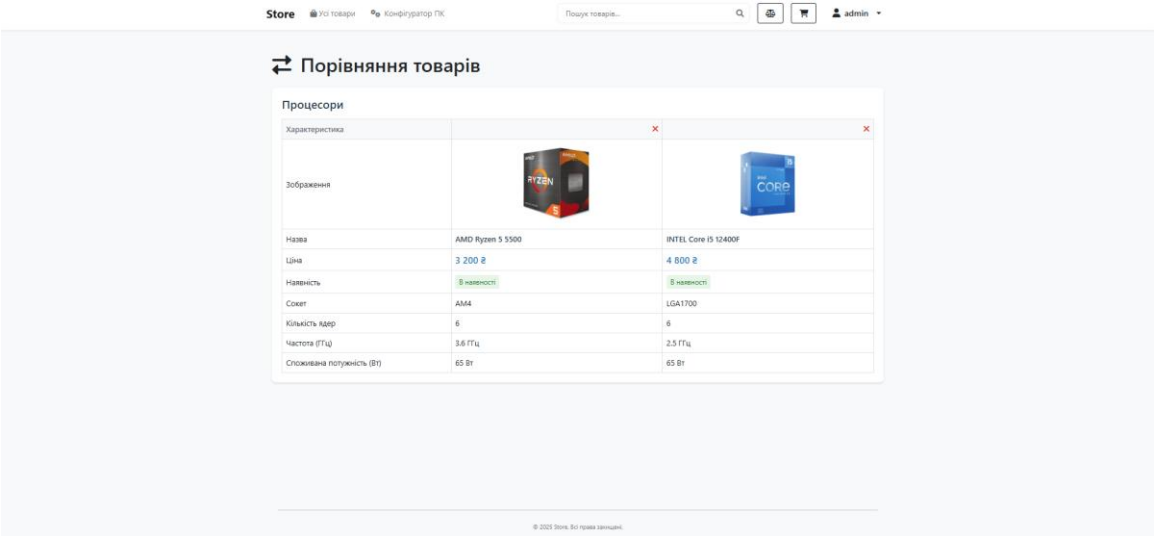


Рис. 3.9. Сторінка порівняння товарів

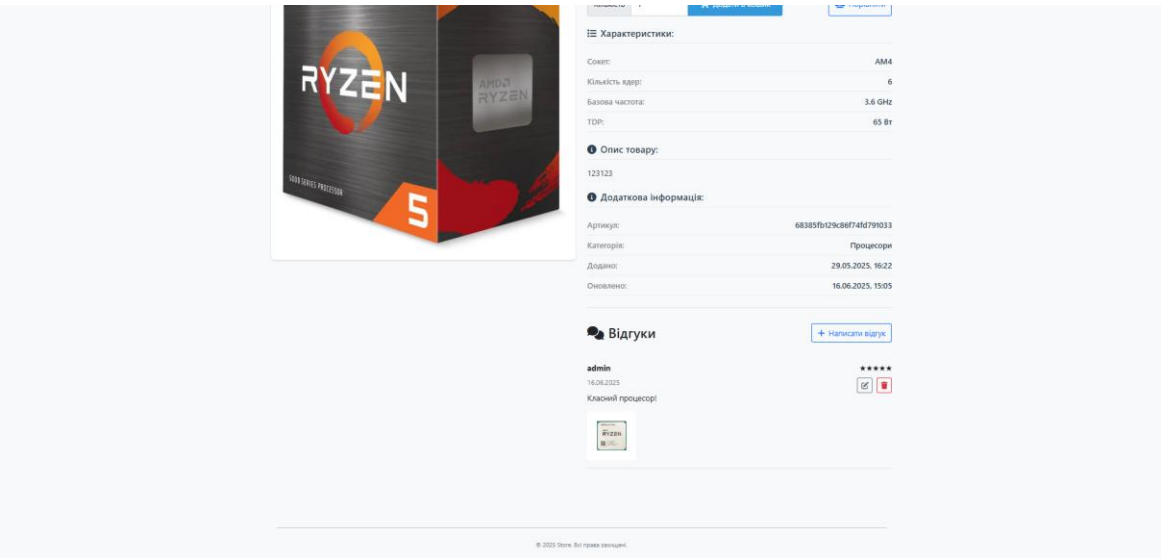


Рис. 3.10. Вигляд відгуків

На сайті також реалізовано пошук за ключовими специфікаціями та назвами. На сторінці результатів пошуку так само відображаються картки товарів та є можливість використовувати фільтрацію (Рис. 3.10.)

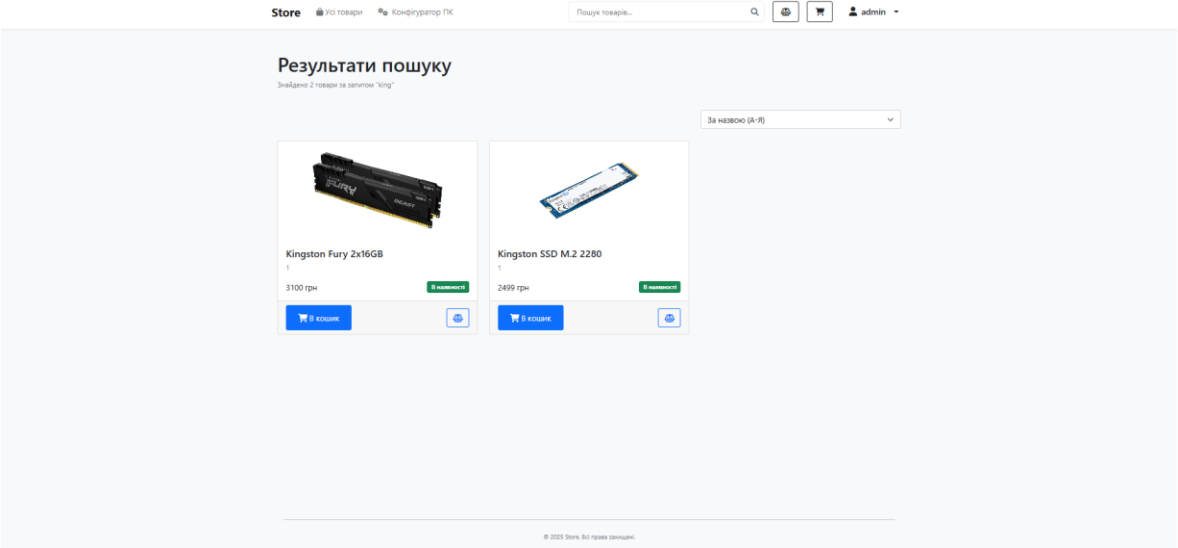


Рис. 3.10 Сторінка з результатами пошуку товарів

Користувачі, які розбираються в комп'ютерних комплектуючих, мають змогу скористатись функцією онлайн-збірки “Конфігуратор ПК” (Рис. 3.11). Це дозволяє самостійно обрати сумісні компоненти для майбутнього комп'ютера.

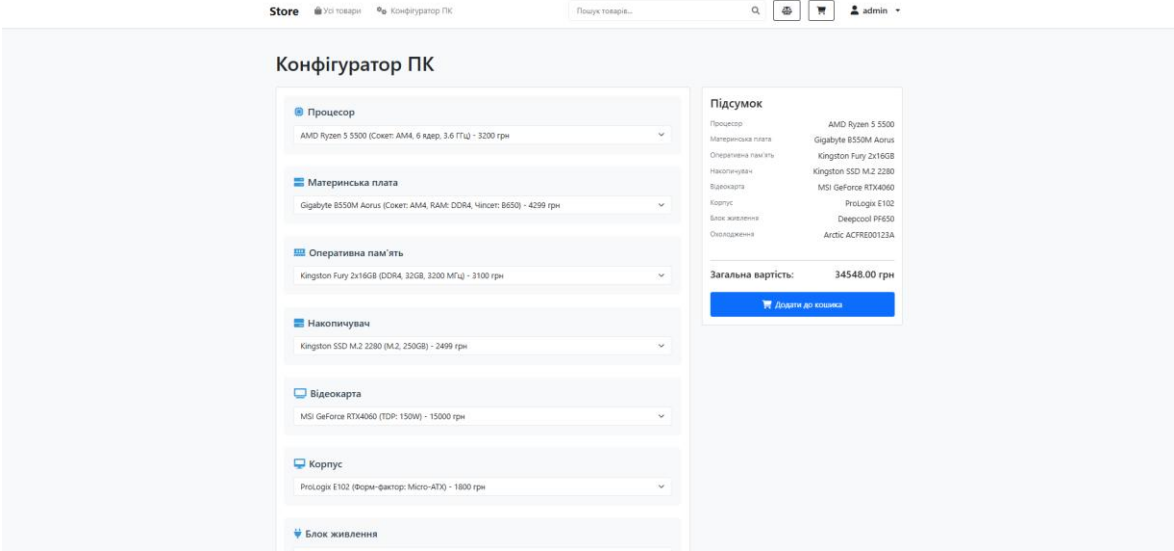


Рис. 3.11. Конфігуратор ПК

У кожного користувача є особистий кабінет (Рис. 3.12), де він може переглянути свою інформацію (ім'я та електронну пошту), змінити пароль та переглянути свої замовлення (Рис. 3.13).

Рис. 3.12. Особистий кабінет

На сторінці “Мої замовлення” (Рис. 3.13) користувач може переглянути історію своїх замовлень з даними доставки та інформацією про отримувача. Натиснувши на картку, можна переглянути деталі замовлення (Рис. 3.14) та його статус, який адміністратор може змінювати залежно від етапу обробки посилки.

Order ID	Date	Time	Payment	Delivery	Address	Contact	Items	Total
#68500b903d64ac775c62e47	16.06.2025	15:20	Online payment	New Post	Житомир, Відділення №11 (до 30 кг на одне місто): вул. Вільський Шлях, 14	Телефон: 0677105129 Email: admin@admin.com	ASUS RTX 3090 x1	34000 грн
#685008c203d64ac775c62d7a	16.06.2025	15:06	Online payment	New Post	Житомир, Відділення №18 (до 30 кг): вул. Шевченка, 18а	Телефон: 0677105129 Email: admin@admin.com	Конфігурація ПК x1	34548 грн
#685003b978c897357d056017	16.06.2025	14:44	Online payment	New Post	Житомир, Відділення №11 (до 30 кг на одне місто): вул. Вільський Шлях, 14	Телефон: 0677105129 Email: admin@admin.com	ASUS RTX 3090 x1	34000 грн

Рис. 3.13. Мої замовлення

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

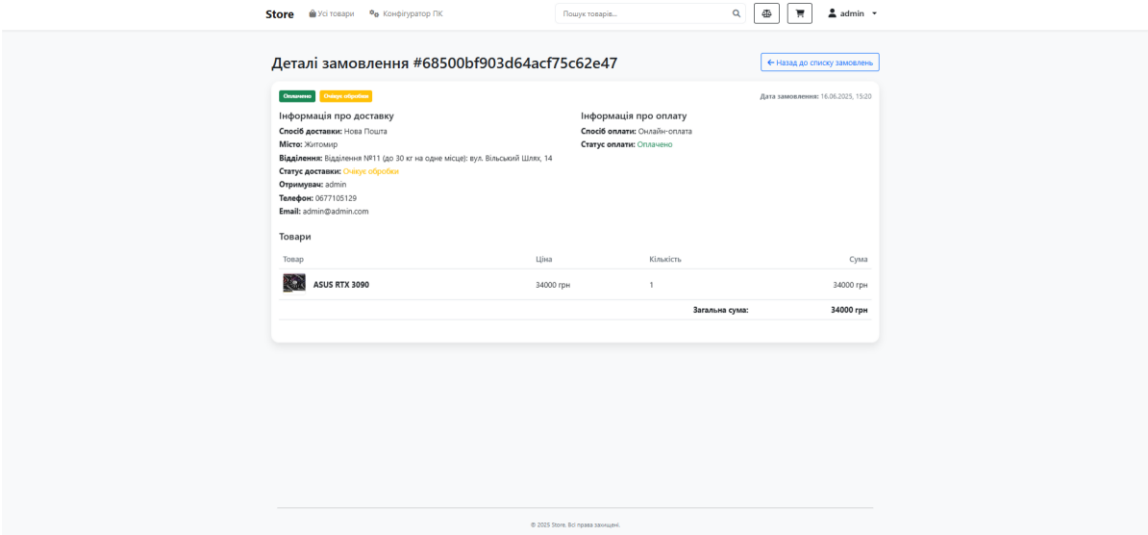


Рис 3.14 Сторінка деталі замовлення

Зареєстрований користувач має змогу додавати товари до кошика (Рис. 3.15.) та оформлювати замовлення (Рис. 3.16.), вносячи свої дані та обираючи місто та відділення для доставки, яке підтягується за допомогою API “Нової пошти”.

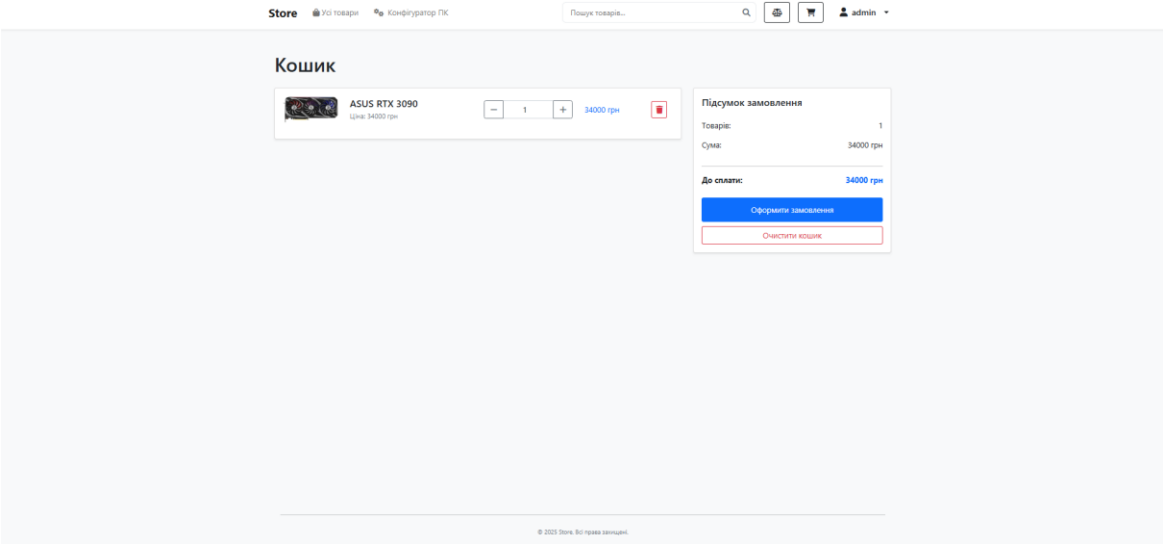


Рис. 3.15. Корзина

Рис 3.16. Оформлення замовлення

Після успішного оформлення та введення коректних даних користувача перенаправляє на сторінку оплати LiqPay (Рис. 3.17), де відображається сума, назва товару та поля для введення платіжних даних.

Рис 3.18 Сторінка успішної оплати від LiqPay

Після спроби оплати є два варіанти розвитку подій. У разі успішної оплати користувача перенаправляє на сторінку підтвердження від LiqPay (Рис. 3.18) та на створену на сайті сторінку успішної оплати (Рис. 3.19). Звідти можна перейти на головну сторінку або на сторінку замовлень. Водночас адміністратори отримують сповіщення в Telegram-групу про нове успішне замовлення (Рис. 3.20).

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

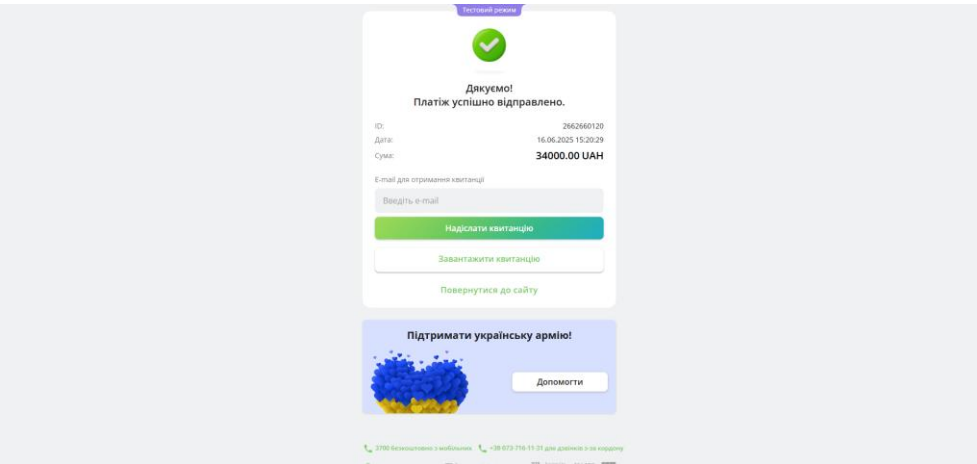


Рис 3.18 Сторінка успішної оплати від LiqPay

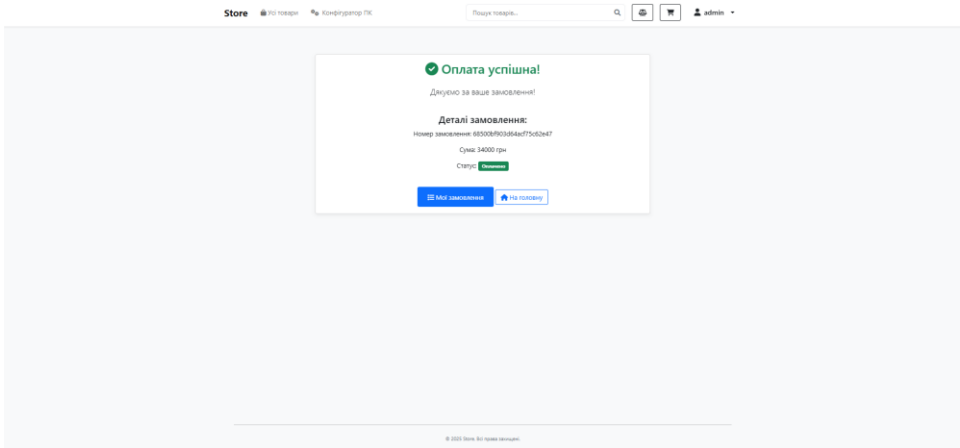


Рис 3.19 Сторінка успішної оплати на сайті

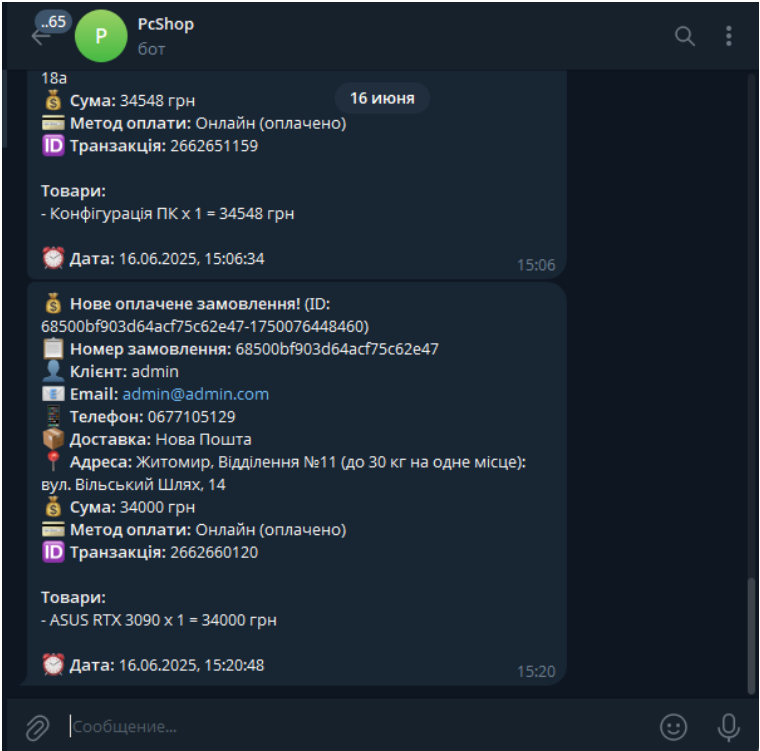


Рис 3.20 Телеграм повідомлення про успішне замовлення користувачем

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

Інший варіант — коли користувачу не вдалося оплатити товар. У такому випадку сповіщення в Telegram не надходить, а після сторінки неуспішної оплати від LiqPay (Рис. 3.21) замовлення не зберігається в історії. Користувача повертає до кошика з повідомленням «Невдалий платіж» (Рис. 3.22).

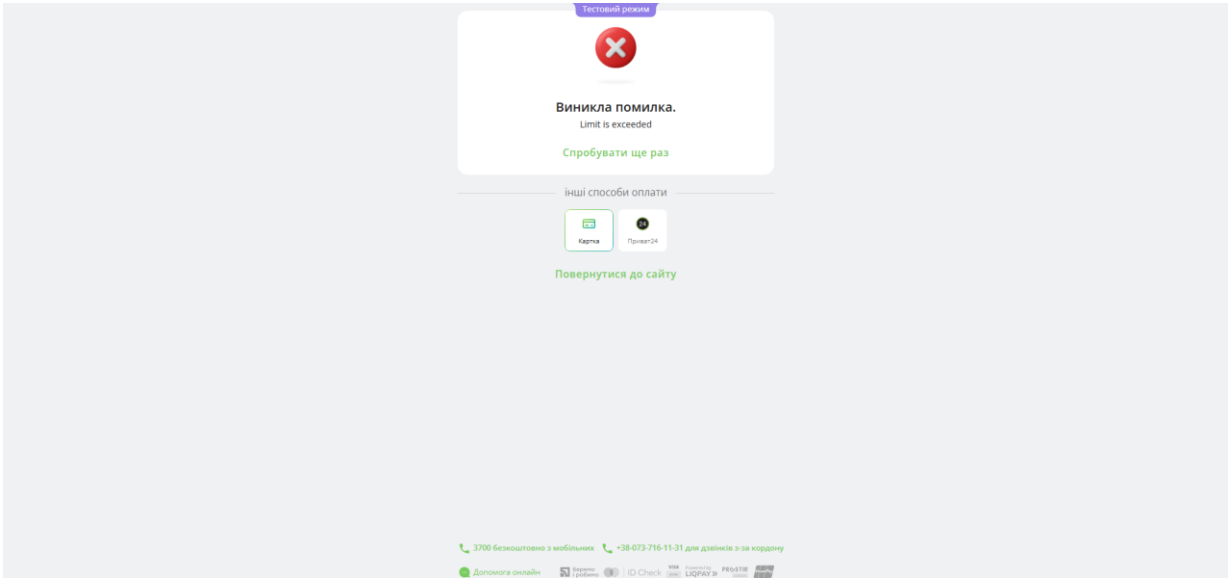


Рис. 3.21. Сторінка неуспішної оплати LiqPay

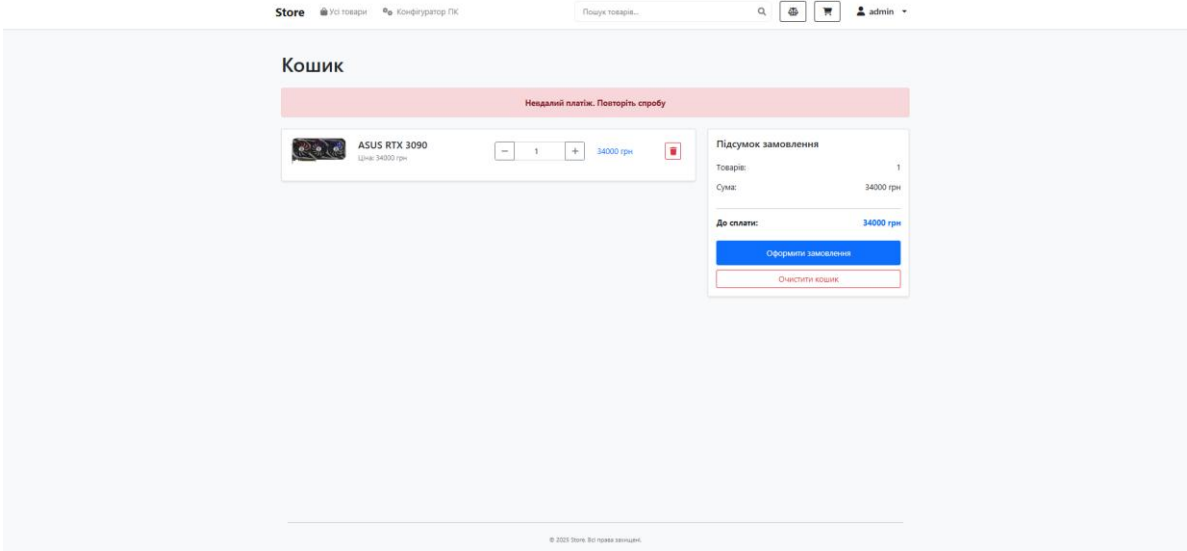


Рис 3.22 Реакція сайту на невдачу оплати

Якщо користувач має роль адміністратора, йому стає доступною панель адміністратора, де зібрано основні функції керування магазином. Перейшовши в адмін-панель, адміністратор одразу потрапляє на сторінку управління категоріями (Рис. 3.23), де має змогу видаляти, додавати та редагувати їх (Рис. 3.24).

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

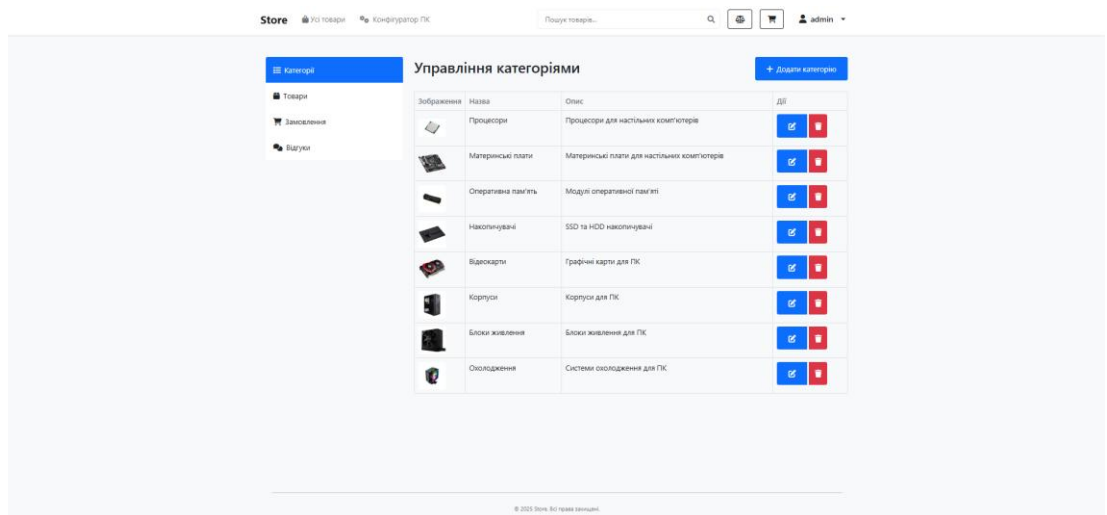


Рис 3.23. Сторінка управління категоріями

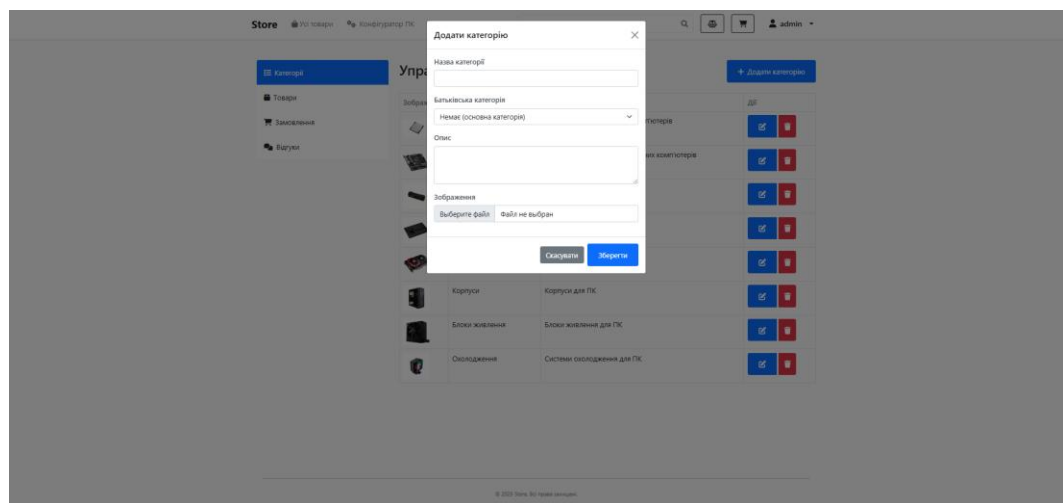


Рис 3.24. Модальне вікно додавання категорій

Далі в меню можна перейти на сторінку управління товарами (рис. 3.25), де також є можливість додавати, редагувати та видаляти товари (рис. 3.26).

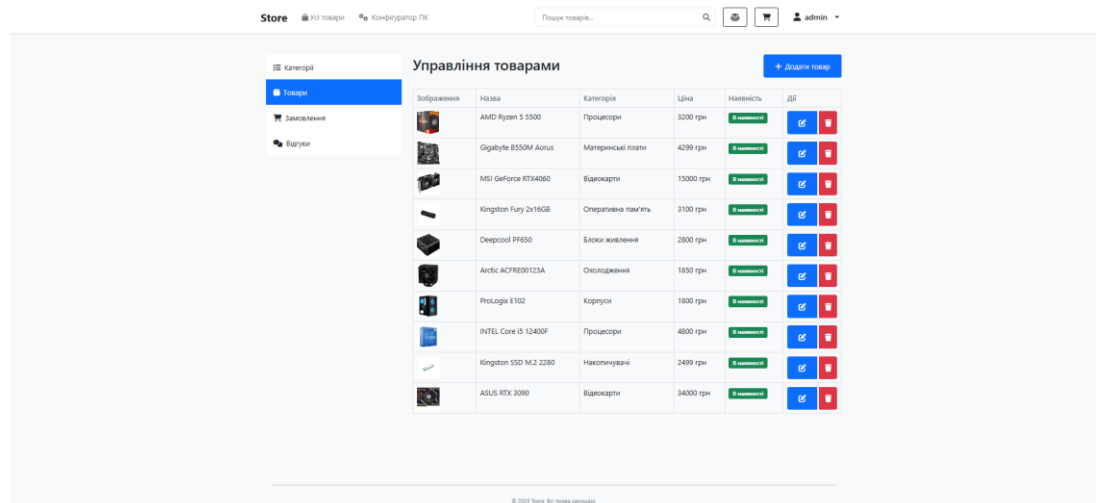


Рис 3.25. Сторінка управління товарами

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

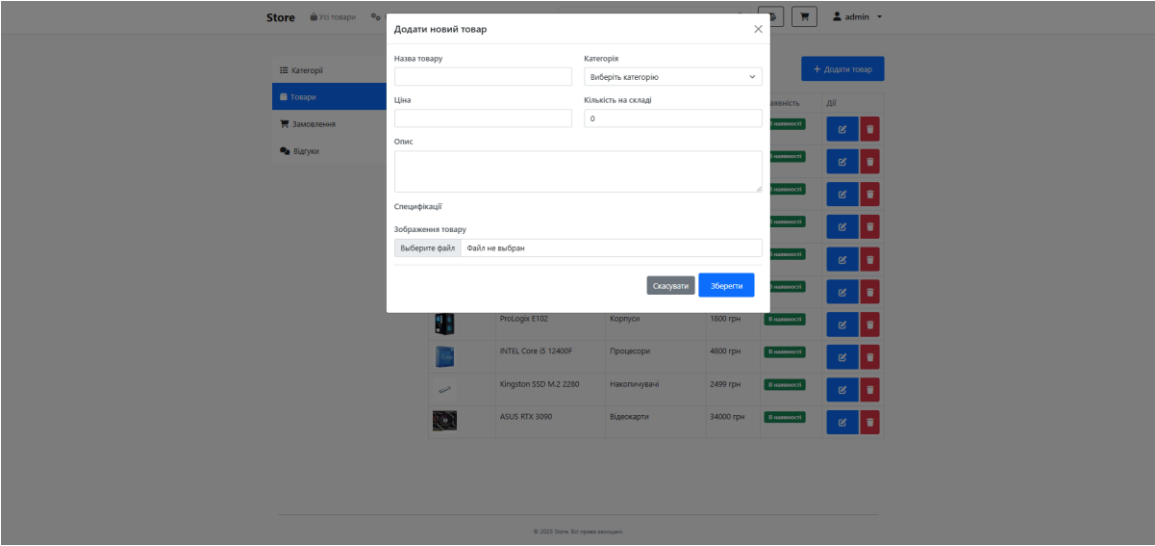


Рис 3.25 Модальне вікно додавання товарів

Адміністратор може обрати категорію, і після цього в модальному вікні з'являться поля для специфікацій, що відповідають цій категорії.

Наступною є сторінка “Управління замовленнями” (Рис. 3.26), де адміністратор може переглядати деталі замовлення (Рис. 3.27) та змінювати статус доставки.

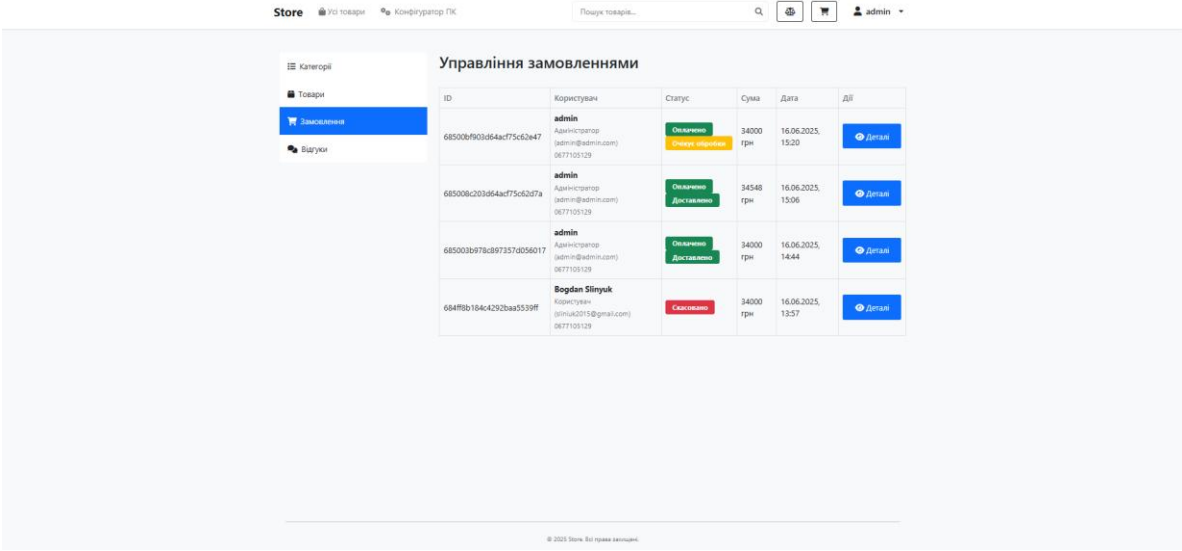


Рис 3.26 Сторінка управління замовленнями

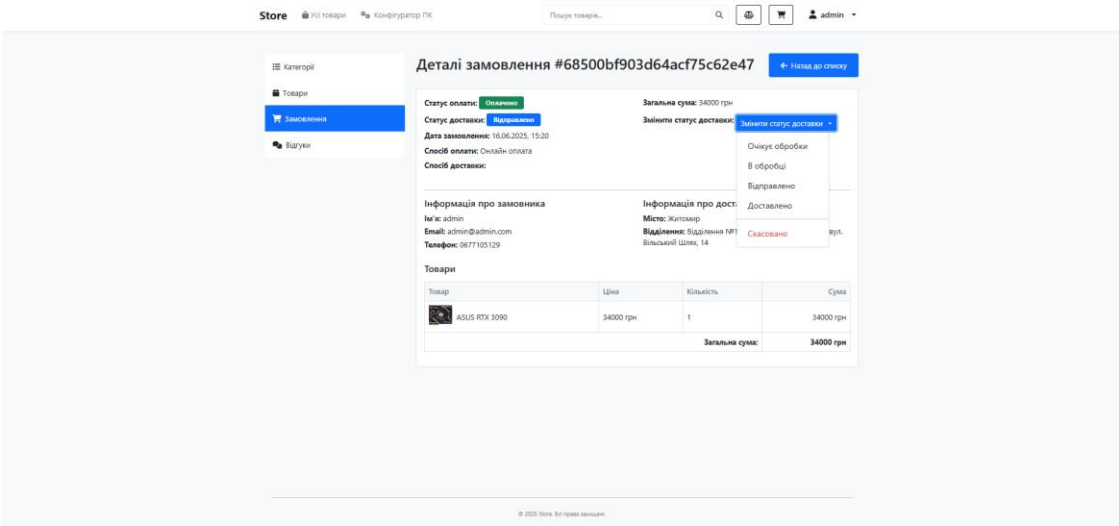


Рис 3.27 Сторінка деталей замовлень

Останньою сторінкою адмін-панелі є “Відгуки” (Рис. 3.28). Після натискання кнопки “Дії” з’являється модальне вікно (Рис. 3.29), де адміністратор може переглядати всі додані відгуки, видаляти їх, а також блокувати користувачам можливість залишати відгуки за порушення правил сайту.

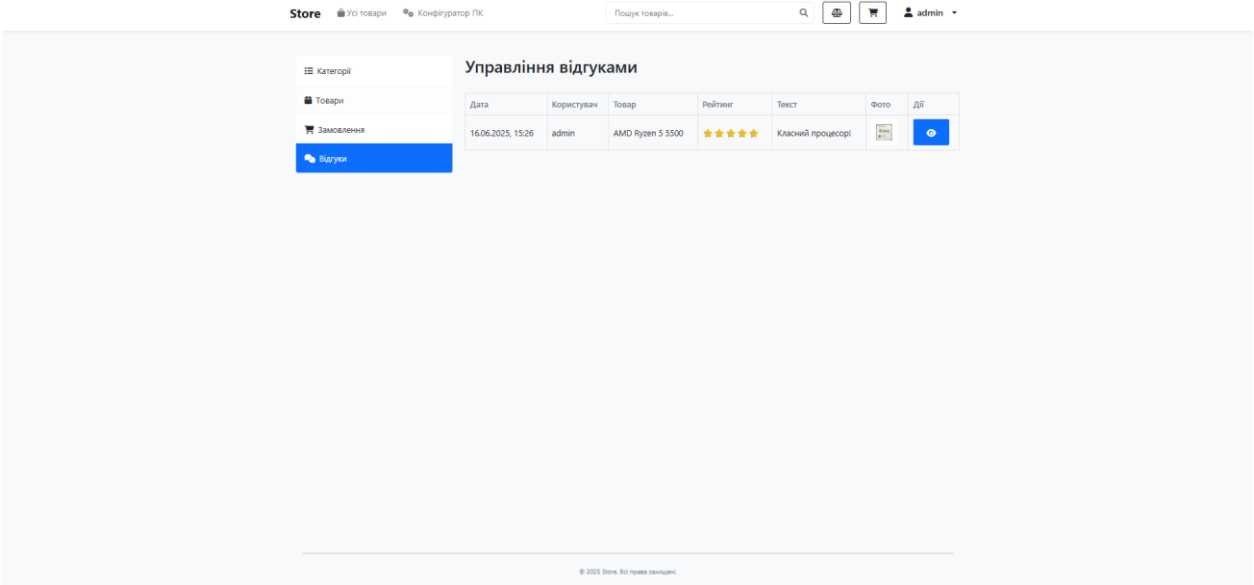


Рис 3.28 Сторінка Управління відгуками

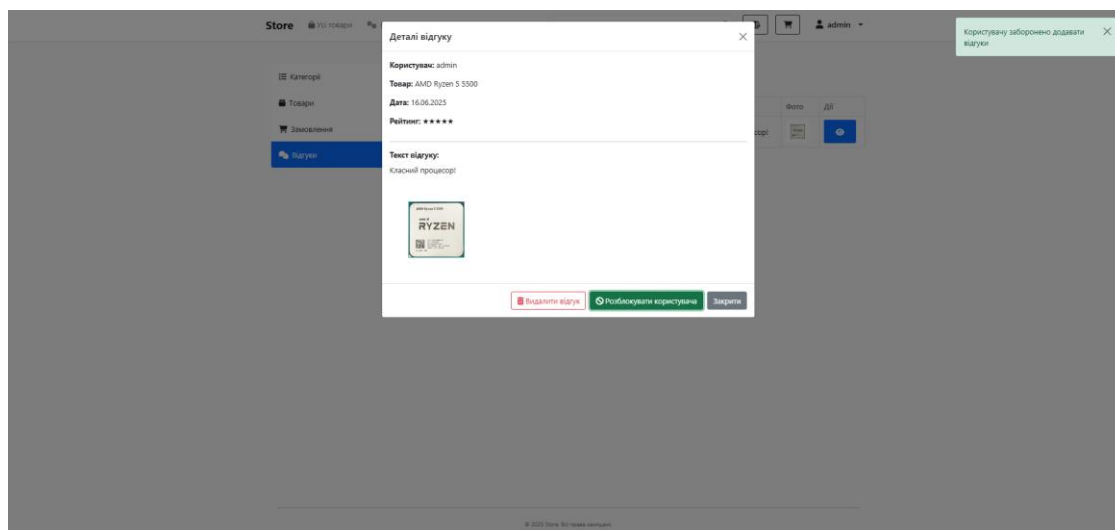


Рис 3.29 Модальне вікно деталей відгуку.

3.3 Тестування програмного продукту

Розробка будь-якої веб-системи, зокрема інтернет-магазину комп'ютерних комплектуючих, супроводжується виникненням певних похибок. Це пов'язано з неможливістю на етапі проєктування передбачити усі сценарії використання системи. Тому процес тестування є невіддільним етапом у створенні якісного програмного забезпечення. Тестування дозволяє оцінити коректність виконання бізнес-логіки та переконатися у стабільності роботи застосунку. Особливу увагу під час перевірки було приділено коректності роботи фільтрів та конфігуратора ПК, функціональності кошика, реєстрації, оформлення замовлення та управління адміністративною панеллю.

Основною метою тестування було виявлення дефектів, які можуть негативно вплинути на досвід користувача, а також забезпечення можливості їх оперативного усунення. Для цього було складено набір тест-кейсів — структурованих сценаріїв, які описують дії користувача та очікувану поведінку системи. Ці тест-кейси допомогли перевірити ключові функціональні модулі. Приклад тестових сценаріїв наведено у таблиці 3.1.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Перелік тест-кейсів для перевірки працездатності інтернет-магазину

Test Case Description	Test Steps	Test Data	Expected Results	Pass/Fail
Реєстрація користувача	Заповнити форму реєстрації, натиснути "Зареєструватися"	Ім'я: admin Email: admin@admin.com Пароль: 123456	Користувача успішно зареєстровано, відбувається редірект на сторінку входу.	Pass
Авторизація (локальна)	Ввести email та пароль на сторінці входу, натиснути "Увійти".	Email: admin@admin.com Пароль: 123456	Вхід успішний, користувач потрапляє на головну сторінку або в особистий кабінет.	Pass
Авторизація (Google)	На сторінці входу натиснути "Увійти через Google", обрати акаунт.	Акаунт Google	Вхід успішний, користувач авторизований в системі.	Pass
Фільтрація товарів	Обрати категорію "Процесори", у фільтрі "Від А до Я"	-	У списку товарів проводиться фільтрація "Від А до Я"	Pass
Пошук товару	Ввести в поле пошуку "RTX".	Текст: "RTX"	На сторінці результатів відображаються відеокарти, що відповідають запиту.	Pass
Робота конфігуратора ПК	1. Зайти в конфігуратор. 2. Обрати процесор Intel.	Процесор: Intel Core i5	У списку материнських плат показує що цей сумісний з цим сокетом.	Pass
Оформлення замовлення	Додати товар в кошик, перейти до оформлення, заповнити дані, обрати оплату "LiqPay".	Товар: будь-який, Адреса: тестова	Замовлення створено, користувача перенаправляється на сторінку оплати LiqPay.	Pass
Перевірка Callback від LiqPay	Після тестової оплати перевірити статус замовлення в адмін-панелі.	order_id	Статус замовлення автоматично змінився на "Оплачено".	Pass

Продовження таблиці 3.1

Сповіщення в Telegram	Створити нове замовлення.	-	Адміністратору в Telegram приходить повідомлення з деталями замовлення.	Pass
Додавання товару (адмін)	В адмін-панелі зайти в "Товари", натиснути "Додати", заповнити форму, зберегти.	Назва, ціна, фото та інше.	Новий товар з'являється у загальному списку товарів в адмін-панелі та на сайті.	Pass
Зміна статусу замовлення	В адмін-панелі знайти замовлення і змінити його статус на "Відправлено".	Статус: "Відправлено"	Статус успішно змінюється. Користувач бачить новий статус в своїх замовленнях.	Pass

Висновки до третього розділу

У третьому розділі було представлено діаграму розгортання інтернет-магазину, а також продемонстровано ключові сторінки інтерфейсу користувача та адміністратора. Було описано основні сценарії взаємодії із системою. Проведене функціональне тестування у вигляді набору тест-кейсів підтвердило стабільну роботу всіх основних компонентів системи та відповідність реалізованого функціоналу поставленим вимогам.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було спроектовано та реалізовано сучасний інтернет-магазин комп'ютерних комплектуючих, що відповідає актуальним вимогам ринку електронної комерції. На основі аналізу предметної області та порівняння з існуючими рішеннями було визначено ключові функціональні та технічні характеристики майбутньої системи.

У ході роботи створено повноцінну документо-орієнтовану базу даних на MongoDB, реалізовано серверну логіку на Node.js за патерном MVC та розроблено інтерфейс з адаптивним дизайном. Система включає механізми реєстрації, авторизації, управління кошиком, оформлення замовлень, а також спеціалізований функціонал, як-от конфігуратор ПК та інструмент для порівняння товарів. Для адміністратора розроблено панель для управління товарами, категоріями, замовленнями та відгуками. Забезпечено підтримку розширеної фільтрації, інтеграцію з платіжною системою LiqPay та сповіщення через Telegram-бот.

Платформа успішно пройшла функціональне тестування, під час якого перевірено її стабільність, працездатність основних модулів і зручність для користувачів. Усі виявлені недоліки були усунуті, що дозволило досягти високої якості реалізації.

Розроблений інтернет-магазин демонструє практичну реалізацію сучасного вебзастосунку для роздрібної торгівлі комп'ютерними комплектуючими, сприяє цифровізації торгівлі, розширенню бізнес-можливостей та підвищенню рівня клієнтського сервісу.

Отриманий результат може бути використаний як готове рішення для запуску бізнесу або як основа для подальшого вдосконалення з урахуванням нових вимог ринку.

		Слинок Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brain — офіційний сайт українського інтернет-магазину електроніки [Електронний ресурс]. - Режим доступу до ресурсу: <https://brain.com.ua>
2. ITbox — офіційний сайт інтернет-магазину IT-техніки [Електронний ресурс]. - Режим доступу до ресурсу: <https://www.itbox.ua>
3. CompX — офіційний сайт ритейлера та виробника ПК [Електронний ресурс]. - Режим доступу до ресурсу: <https://compx.ua>
4. Telemart — офіційний сайт спеціалізованого інтернет-магазину комп'ютерних комплектуючих [Електронний ресурс]. - Режим доступу до ресурсу: <https://telemart.ua>
5. LiqPay API — офіційна документація платіжного шлюзу [Електронний ресурс]. - Режим доступу до ресурсу: <https://www.liqpay.ua/documentation/api/>
6. Node.js — офіційна документація платформи Node.js [Електронний ресурс]. - Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>
7. PostgreSQL — офіційна документація СУБД PostgreSQL [Електронний ресурс]. - Режим доступу до ресурсу: <https://www.postgresql.org/docs/>
8. MySQL — офіційна документація СУБД MySQL [Електронний ресурс]. - Режим доступу до ресурсу: <https://dev.mysql.com/doc/>
9. MongoDB — офіційна документація СУБД MongoDB [Електронний ресурс]. - Режим доступу до ресурсу: <https://www.mongodb.com/docs/>
10. JavaScript | MDN — документація мови програмування JavaScript [Електронний ресурс]. - Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
11. Express — офіційна документація веб-фреймворку Express.js [Електронний ресурс]. - Режим доступу до ресурсу: <https://expressjs.com/>
12. Git — офіційна документація системи контролю версій Git [Електронний ресурс]. - Режим доступу до ресурсу: <https://git-scm.com/doc>

		Слинюк Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Mongoose — офіційна документація ODM-бібліотеки для MongoDB [Електронний ресурс]. - Режим доступу до ресурсу: <https://mongoosejs.com/docs/guide.html>

14. Handlebars — офіційний сайт шаблонізатора Handlebars.js [Електронний ресурс]. - Режим доступу до ресурсу: <https://handlebarsjs.com/>

15. Passport.js — офіційний сайт фреймворку для автентифікації в Node.js [Електронний ресурс]. - Режим доступу до ресурсу: <http://www.passportjs.org/>

16. bcrypt.js — документація бібліотеки для хешування паролів на npm [Електронний ресурс]. - Режим доступу до ресурсу: <https://www.npmjs.com/package/bcryptjs>

17. Bootstrap — офіційна документація CSS-фреймворку [Електронний ресурс]. - Режим доступу до ресурсу: <https://getbootstrap.com/docs/>

18. Twilio for Node.js — офіційна документація для розробників [Електронний ресурс]. - Режим доступу до ресурсу: <https://www.twilio.com/docs/node/>

19. Nodemailer — офіційний сайт бібліотеки для надсилання електронних листів [Електронний ресурс]. - Режим доступу до ресурсу: <https://nodemailer.com/>

20. node-telegram-bot-api — документація бібліотеки на GitHub [Електронний ресурс]. - Режим доступу до ресурсу: <https://github.com/yagop/node-telegram-bot-api>

21. Model-View-Controller (MVC) - MDN Web Docs [Електронний ресурс]. - Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>

22. REST - MDN Web Docs [Електронний ресурс]. - Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Glossary/REST>

23. Using middleware in Express.js [Електронний ресурс]. - Режим доступу до ресурсу: <https://expressjs.com/en/guide/using-middleware.html>

24. Helmet — документація бібліотеки для захисту Express-додатків [Електронний ресурс]. - Режим доступу до ресурсу: <https://helmetjs.github.io/>

		Слинюк Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						53

25. Asynchronous programming - JavaScript | MDN [Електронний ресурс]. - Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous>

26. express-session — документація бібліотеки для керування сесіями на npm [Електронний ресурс]. – Режим доступу: <https://www.npmjs.com/package/express-session>

27. Visual Studio Code — офіційний сайт редактора коду [Електронний ресурс]. – Режим доступу: <https://code.visualstudio.com/docs>

28. Brown E. Web Development with Node and Express, 2nd Edition. Ethan Brown – O'Reilly Media, 2019 – 444 p.

29. Haverbeke M. Eloquent JavaScript, 3rd Edition. Marijn Haverbeke – No Starch Press, 2018 – 472 p.

30. Нова пошта API — офіційна документація для розробників [Електронний ресурс]. – Режим доступу: <https://devcenter.novaposhta.ua/>

		Слинюк Б.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

ДОДАТКИ

Лістинг файлу Passport.js

```
const passport = require('passport');
const LocalStrategy = require('passport-local').Strategy;
const GoogleStrategy = require('passport-google-oauth20').Strategy;
const User = require('../models/userModel');
const bcrypt = require('bcryptjs');
const googleConfig = require('../google').google;
const googleService = require('../services/googleService');

passport.use('local', new LocalStrategy( {
  usernameField: 'email',
  passwordField: 'password'
}, async (email, password, done) => {
  try {
    const user = await User.findOne( { email } );
    if (!user) {
      return done(null, false, { message: 'Неправильний email або пароль' });
    }
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return done(null, false, { message: 'Неправильний email або пароль' });
    }
    return done(null, user);
  } catch (error) {
    return done(error);
  }
}));

passport.use('google', new GoogleStrategy( {
  clientID: googleConfig.clientID,
  clientSecret: googleConfig.clientSecret,
  callbackURL: googleConfig.callbackURL,
  authorizationURL: googleConfig.authorizationURL,
  tokenURL: googleConfig.tokenURL,
  userProfileURL: googleConfig.userProfileURL,
  passReqToCallback: true
}, async (req, accessToken, refreshToken, profile, done) => {
  try {
    const user = await googleService.findOrCreateUser(profile);
    return done(null, user);
  } catch (error) {
    return done(error, null);
  }
}));

passport.serializeUser((user, done) => {
  done(null, user.id);
});

passport.deserializeUser(async (id, done) => {
  try {
```

```

    const user = await User.findById(id);
    done(null, user);
  } catch (error) {
    done(error);
  }
});

```

```

module.exports = passport;

```

Лістинг файлу authMiddleware.js

```

exports.isAuthenticated = (req, res, next) => {
  console.log('Auth middleware check:', {
    hasSession: !!req.session,
    hasUser: req.session ? !!req.session.user : false,
    userId: req.session && req.session.user ? req.session.user._id : null,
    userIdOld: req.session && req.session.user ? req.session.user.id : null,
    userRole: req.session && req.session.user ? req.session.user.role : null
  });

  if (!req.session) {
    if (req.xhr || (req.headers.accept && req.headers.accept.includes('json'))) {
      return res.status(500).json({
        success: false,
        message: 'Помилка сесії. Спробуйте оновити сторінку'
      });
    }
    req.flash('error', 'Помилка сесії. Спробуйте оновити сторінку');
    return res.redirect('/login');
  }

  if (!req.session.user) {
    if (req.xhr || (req.headers.accept && req.headers.accept.includes('json'))) {
      return res.status(401).json({
        success: false,
        message: 'Необхідна авторизація'
      });
    }
    req.flash('error', 'Будь ласка, увійдіть у систему');
    return res.redirect('/login');
  }

  next();
};

exports.isAdmin = (req, res, next) => {
  if (!req.session) {
    if (req.xhr || (req.headers.accept && req.headers.accept.includes('json'))) {
      return res.status(500).json({
        success: false,
        message: 'Помилка сесії. Спробуйте оновити сторінку'
      });
    }
  }
}

```

```

    req.flash('error', 'Помилка сесії. Спробуйте оновити сторінку');
    return res.redirect('/login');
  }

  if (!req.session.user) {
    if (req.xhr || (req.headers.accept && req.headers.accept.includes('json'))) {
      return res.status(401).json({
        success: false,
        message: 'Необхідна авторизація'
      });
    }
    req.flash('error', 'Будь ласка, увійдіть у систему');
    return res.redirect('/login');
  }

  if (req.session.user.role !== 'admin') {
    if (req.xhr || (req.headers.accept && req.headers.accept.includes('json'))) {
      return res.status(403).json({
        success: false,
        message: 'Доступ заборонено. Необхідні права адміністратора'
      });
    }
    req.flash('error', 'Доступ заборонено. Необхідні права адміністратора');
    return res.redirect('/');
  }

  next();
};

```

Лістинг файлу categoryModel.js

```

const mongoose = require('mongoose');

const categorySchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Category name is required'],
    unique: true,
    trim: true
  },
  slug: {
    type: String,
    required: [true, 'Category slug is required'],
    unique: true,
    trim: true,
    lowercase: true
  },
  description: {
    type: String,
    trim: true
  },
  parent: {

```

```

    type: mongoose.Schema.Types.ObjectId,
    ref: 'Category',
    default: null
  },
  image: {
    type: String,
    default: 'default-category.jpg'
  },
  isActive: {
    type: Boolean,
    default: true
  }
}, {
  timestamps: true
});

categorySchema.index({ parent: 1 });

categorySchema.virtual('subcategories', {
  ref: 'Category',
  localField: '_id',
  foreignField: 'parent'
});

const Category = mongoose.model('Category', categorySchema);

module.exports = Category;

```

Лістинг файлу cartController.js

```

const Product = require('../models/productModel');

exports.getCart = async (req, res) => {
  try {
    if (!req.session.cart || !Array.isArray(req.session.cart.items)) {
      req.session.cart = {
        items: [],
        total: 0
      };
    }

    const updatedItems = await Promise.all(req.session.cart.items.map(async (item) => {
      if (item.type === 'pc_configuration') {
        return item;
      }
      const product = await Product.findById(item.id);
      if (product) {
        return {
          ...item,
          stock: product.stock
        };
      }
      return item;
    }));
  }

```

```

    }));

    req.session.cart.items = updatedItems.filter(item => {
      return item && item.price && item.quantity;
    });

    const total = req.session.cart.items.reduce((sum, item) => {
      return sum + (item.price * item.quantity);
    }, 0);

    req.session.cart.total = total;

    await new Promise((resolve, reject) => {
      req.session.save((err) => {
        if (err) {
          reject(err);
        } else {
          resolve();
        }
      });
    });

    const error = req.query.error;

    res.render('cart', {
      cart: req.session.cart,
      cartCount: req.session.cart.items.length,
      error: error || null
    });
  } catch (error) {
    res.render('cart', {
      cart: { items: [], total: 0 },
      cartCount: 0
    });
  }
};

exports.addToCart = async (req, res) => {
  try {
    if (!req.session) {
      return res.status(500).json({
        success: false,
        message: 'Помилка сесії. Спробуйте оновити сторінку'
      });
    }

    if (req.body && req.body.processor) {
      return await addConfigurationToCart(req, res);
    }

    let productId, quantity;

    if (req.is('application/json')) {

```

```

    productId = req.body.productId;
    quantity = parseInt(req.body.quantity) || 1;
  } else {
    productId = req.body.productId;
    quantity = parseInt(req.body.quantity) || 1;
  }

  if (!productId) {
    return res.status(400).json({
      success: false,
      message: 'ID товару не вказано'
    });
  }

  const product = await Product.findById(productId);

  if (!product) {
    return res.status(404).json({
      success: false,
      message: 'Товар не знайдено'
    });
  }

  if (!product.isActive || product.stock <= 0) {
    return res.status(400).json({
      success: false,
      message: 'Товар не доступний для замовлення'
    });
  }

  if (product.stock < quantity) {
    return res.status(400).json({
      success: false,
      message: `На складі залишилось тільки ${product.stock} шт.`
    });
  }

  if (!req.session.cart) {
    req.session.cart = {
      items: [],
      total: 0
    };
  }

  const existingItemIndex = req.session.cart.items.findIndex(item =>
    item && item.id && item.id.toString() === productId.toString()
  );

  if (existingItemIndex !== -1) {
    const existingItem = req.session.cart.items[existingItemIndex];

    if (existingItem.quantity + quantity > product.stock) {
      return res.status(400).json({

```

```

        success: false,
        message: `На складі залишилось тільки ${product.stock} шт.`
    });
}

existingItem.quantity += quantity;
existingItem.total = existingItem.price * existingItem.quantity;
existingItem.stock = product.stock;

req.session.cart.items[existingItemIndex] = existingItem;
} else {
    const newItem = {
        id: product._id,
        name: product.name,
        price: product.price,
        quantity: quantity,
        image: product.image || 'default-product.jpg',
        stock: product.stock,
        total: product.price * quantity
    };
    req.session.cart.items.push(newItem);
}

req.session.cart.total = req.session.cart.items.reduce((sum, item) => {
    return sum + (item.price * item.quantity);
}, 0);

try {
    await new Promise((resolve, reject) => {
        req.session.save((err) => {
            if (err) reject(err);
            else resolve();
        });
    });
} catch (sessionError) {
    return res.status(500).json({
        success: false,
        message: 'Помилка збереження даних кошика'
    });
}

res.json({
    success: true,
    message: 'Товар додано до кошика',
    cart: req.session.cart
});
} catch (error) {
    res.status(500).json({
        success: false,
        message: 'Помилка при додаванні товару до кошика'
    });
}
};

```

```

async function addConfigurationToCart(req, res) {
  try {
    const configuration = req.body;

    const components = await Promise.all(
      Object.entries(configuration).map(async ([type, id]) => {
        const product = await Product.findById(id).lean();
        if (!product) {
          throw new Error(`Компонент не знайдено: ${type}`);
        }
        return {
          type,
          id: product._id,
          name: product.name,
          price: product.price,
          image: product.image
        };
      })
    );

    if (!req.session.cart) {
      req.session.cart = { items: [], total: 0 };
    }

    const totalPrice = components.reduce((sum, component) => sum + component.price, 0);

    const configurationItem = {
      id: Date.now().toString(),
      type: 'pc_configuration',
      name: 'Конфігурація ПК',
      components: components,
      price: totalPrice,
      quantity: 1,
      total: totalPrice,
      image: '/images/pc-configuration.png'
    };

    req.session.cart.items.push(configurationItem);

    req.session.cart.total = req.session.cart.items.reduce((sum, item) => {
      return sum + (item.price * item.quantity);
    }, 0);

    await new Promise((resolve, reject) => {
      req.session.save((err) => {
        if (err) {
          reject(err);
        } else {
          resolve();
        }
      });
    });
  });
}

```

```

    res.json({
      success: true,
      message: 'Конфігурацію додано до кошика',
      redirect: '/cart',
      cart: req.session.cart
    });
  } catch (error) {
    res.status(500).json({
      success: false,
      message: error.message || 'Помилка при додаванні конфігурації до кошика'
    });
  }
}

exports.updateQuantity = async (req, res) => {
  try {
    const { productId, quantity } = req.body;

    if (!req.session.cart) {
      return res.status(400).json({ success: false, message: 'Кошик порожній' });
    }

    const item = req.session.cart.items.find(item => item.id === productId);

    if (!item) {
      return res.status(404).json({ success: false, message: 'Товар не знайдено в кошику' });
    }

    const product = await Product.findById(productId);
    if (!product) {
      return res.status(404).json({ success: false, message: 'Товар не знайдено' });
    }

    if (quantity > product.stock) {
      return res.status(400).json({
        success: false,
        message: `На складі залишилось тільки ${product.stock} шт.`
      });
    }

    item.quantity = parseInt(quantity);
    item.total = item.quantity * item.price;
    item.stock = product.stock;

    req.session.cart.total = req.session.cart.items.reduce((sum, item) => sum + item.total, 0);

    await new Promise((resolve, reject) => {
      req.session.save((err) => {
        if (err) {
          reject(err);
        } else {
          resolve();
        }
      })
    })
  }
}

```

```

    });
  });

  res.json({
    success: true,
    message: 'Кількість оновлено',
    cart: req.session.cart
  });
} catch (error) {
  res.status(500).json({
    success: false,
    message: 'Помилка при оновленні кількості'
  });
}
};

exports.removeItem = async (req, res) => {
  try {
    const { productId } = req.params;

    if (!req.session.cart) {
      return res.status(400).json({
        success: false,
        message: 'Кошик порожній'
      });
    }

    const itemIndex = req.session.cart.items.findIndex(item => item.id.toString() ===
productId.toString());

    if (itemIndex === -1) {
      return res.status(404).json({
        success: false,
        message: 'Товар не знайдено в кошику'
      });
    }

    req.session.cart.items.splice(itemIndex, 1);
    req.session.cart.total = req.session.cart.items.reduce((sum, item) => sum + item.total, 0);

    res.json({
      success: true,
      message: 'Товар видалено з кошика',
      cart: req.session.cart
    });
  } catch (error) {
    res.status(500).json({
      success: false,
      message: 'Помилка при видаленні товару'
    });
  }
};

```

```

exports.clearCart = async (req, res) => {
  try {
    req.session.cart = { items: [], total: 0 };
    res.json({ success: true, message: 'Кошик очищено' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Помилка при очищенні кошика' });
  }
};

```

Лістинг коду CategoryController.js

```

const Category = require('../models/categoryModel');
const Product = require('../models/productModel');
const slugify = require('slugify');

exports.createCategory = async (req) => {
  try {
    const { name, description, parent, image, order } = req.body;

    const slug = slugify(name, { lower: true });

    const category = await Category.create({
      name,
      slug,
      description,
      parent: parent || null,
      image,
      order
    });

    return {
      success: true,
      data: category
    };
  } catch (error) {
    throw error;
  }
};

exports.getAllCategories = async (req) => {
  try {
    const categories = await Category.find({ isActive: true })
      .sort({ name: 1 })
      .populate('parent', 'name slug');

    return {
      success: true,
      data: categories
    };
  } catch (error) {
    throw error;
  }
};

```

```

exports.getCategoriesForHomePage = async () => {
  try {
    const categories = await Category.find({ isActive: true })
      .sort({ order: 1, name: 1 })
      .populate('parent', 'name slug');

    return categories;
  } catch (error) {
    return [];
  }
};

exports.getCategoryById = async (req) => {
  try {
    const category = await Category.findById(req.params.id)
      .populate('parent', 'name slug');

    if (!category) {
      throw new Error('Category not found');
    }

    return {
      success: true,
      data: category
    };
  } catch (error) {
    throw error;
  }
};

exports.getCategoryBySlug = async (req) => {
  try {
    const category = await Category.findOne({ slug: req.params.slug })
      .populate('parent', 'name slug');

    if (!category) {
      throw new Error('Category not found');
    }

    return {
      success: true,
      data: category
    };
  } catch (error) {
    throw error;
  }
};

exports.updateCategory = async (req) => {
  try {
    const { name, description, parent, image, order, isActive } = req.body;

```

```

const updateData = {
  description,
  parent: parent || null,
  image,
  order,
  isActive
};

if (name) {
  updateData.name = name;
  updateData.slug = slugify(name, { lower: true });
}

const category = await Category.findByIdAndUpdate(
  req.params.id,
  updateData,
  { new: true, runValidators: true }
).populate('parent', 'name slug');

if (!category) {
  throw new Error('Category not found');
}

return {
  success: true,
  data: category
};
} catch (error) {
  throw error;
}
};

exports.deleteCategory = async (req, res) => {
  try {
    const category = await Category.findById(req.params.id);

    if (!category) {
      req.flash('error', 'Категорію не знайдено');
      return res.redirect('/admin/categories');
    }

    const subcategories = await Category.find({ parent: category._id });

    for (const subcategory of subcategories) {
      await Category.findByIdAndDelete(subcategory._id);
    }

    await Product.updateMany(
      { category: category._id },
      { $set: { category: null } }
    );
    await Category.findByIdAndDelete(req.params.id);
  }
};

```

```

    req.flash('success', 'Категорію та її підкатегорії успішно видалено');
    res.redirect('/admin/categories');
  } catch (error) {
    req.flash('error', 'Помилка при видаленні категорії');
    res.redirect('/admin/categories');
  }
};

```

```

exports.getSubcategories = async (req) => {
  try {
    const subcategories = await Category.find({
      parent: req.params.id,
      isActive: true
    }).sort({ name: 1 });

    return {
      success: true,
      data: subcategories
    };
  } catch (error) {
    throw error;
  }
};

```

```

exports.getCategoryDetails = async (req, res) => {
  try {
    const categoryId = req.params.id;

    const currentCategory = await Category.findById(categoryId)
      .populate('parent', 'name slug');

    if (!currentCategory) {
      req.flash('error', 'Категорію не знайдено');
      return res.redirect('/');
    }

    const subcategories = await Category.find({
      parent: categoryId,
      isActive: true
    });

    let products = await Product.find({
      category: categoryId,
      isActive: true
    })
      .populate('category', 'name slug')
      .lean();

    products = products.map(product => {
      if (product.specs) {
        let formattedSpecs = { ...product.specs };

        if (formattedSpecs.capacity) {

```

```

        formattedSpecs.capacity = parseInt(formattedSpecs.capacity);
    }
    if (formattedSpecs.frequency) {
        formattedSpecs.frequency = parseFloat(formattedSpecs.frequency);
    }
    if (formattedSpecs.power) {
        formattedSpecs.power = parseInt(formattedSpecs.power);
    }
    if (formattedSpecs.memory) {
        formattedSpecs.memory = parseInt(formattedSpecs.memory);
    }

    if (formattedSpecs.socket) {
        formattedSpecs.socket = formattedSpecs.socket.toString().replace(/[{}]/g, "");
    }
    if (formattedSpecs.chipset) {
        formattedSpecs.chipset = formattedSpecs.chipset.toString().replace(/[{}]/g, "");
    }
    if (formattedSpecs.formFactor) {
        formattedSpecs.formFactor = formattedSpecs.formFactor.toString().replace(/[{}]/g,
");
    }
    if (formattedSpecs.type) {
        formattedSpecs.type = formattedSpecs.type.toString().replace(/[{}]/g, "");
    }
    if (formattedSpecs.certification) {
        formattedSpecs.certification =
formattedSpecs.certification.toString().replace(/[{}]/g, "");
    }
    if (formattedSpecs.chip) {
        formattedSpecs.chip = formattedSpecs.chip.toString().replace(/[{}]/g, "");
    }

    product.specs = formattedSpecs;
}
return product;
});

res.render('category-detail', {
    title: currentCategory.name,
    currentCategory,
    subcategories,
    products
});
} catch (error) {
    req.flash('error', 'Помилка при завантаженні категорії');
    res.redirect('/');
}
};

exports.getCategoryByName = async (req, res) => {
    try {
        const categoryName = req.params.name;

```

```
const category = await Category.findOne({
  name: { $regex: new RegExp(categoryName, 'i') }
}).populate('parent', 'name');

if (!category) {
  req.flash('error', 'Категорію не знайдено');
  return res.redirect('/');
}

const categories = await Category.find({ parent: null })
  .sort({ order: 1 });

const products = await Product.find({ category: category._id })
  .populate('category', 'name')
  .sort({ createdAt: -1 });

res.render('category-detail', {
  title: category.name,
  currentCategory: category,
  categories,
  products
});
} catch (error) {
  req.flash('error', 'Помилка при отриманні даних категорії');
  res.redirect('/');
}
};
```