Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Бакалавр
(назва освітнього ступеня)

на тему: Розробка інтернет-магазину комп'ютерної техніки «MOYO» з використанням технологій Node Js та React.

Виконав(ла): студент(ка) 4 курсу, групи СПз-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Свіченюк А.М. (прізвище та ініціали)

Керівник (підпис) Пастух О. А. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю. М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М. Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2025

АНОТАЦІЯ

Кваліфікаційна робота бакалавра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2025, Сторінок 57, таблиць 2, рисунків 18, джерел 13, додатки 3, презентація.

Ключові слова: комп’ютерна техніка, інтернет магазин, база даних, html, js, react, nodejs, mongodb, express замовлення, реєстрація.

Кваліфікаційна робота присвячена створенню інтернет-магазину комп’ютерної техніки “MOYO” із застосуванням сучасних вебтехнологій React на стороні клієнта та Node.js на стороні сервера. Основна мета – створити ефективну, швидку та зручну онлайн-платформу для торгівлі комп’ютерною технікою.

Метою роботи є розробка інтернет-магазину, що забезпечує інтуїтивно зрозумілий інтерфейс, швидкий доступ до каталогу товарів і можливість оформлення замовлень.

В роботі використано компонентний підхід, клієнт-серверну архітектуру, REST API, об'єктно-документне моделювання бази даних. Клієнтська частина реалізована на React з використанням Redux Toolkit, Серверна частина – на Node.js + Express, для бази даних було обрано MongoDB, також реалізовано авторизацію з JWT, валідацію форм, обробку кошика, сортування товарів та оформлення замовлень.

Розроблену систему можна розгорнути як у навчальному, так і у комерційному середовищі з мінімальними витратами на адаптацію.

Результатом роботи став успішно реалізовано онлайн-магазин з базовими функціональними модулями. Проведено тестування та налагодження основних процесів користувача.

Використання такого вебзастосунку сприяє цифровізації бізнесу, оптимізації торгових процесів, підвищенню доступності техніки для користувачів, а також розширенню ринку збуту.

У роботі розглянуто вимоги до організації безпечного робочого місця, регламентовані перерви для зменшення навантаження.

В подальшому можливе розширення функціоналу проєкту: інтеграція платіжних систем, логістичних служб, впровадження особистого кабінету, аналітики, відгуків, адаптації під мобільні платформи.

ABSTRACT

Bachelor’s Qualification Thesis. Ternopil Ivan Puluj National Technical University, Department of Software Engineering, specialty 121 “Software Engineering”. TNTU, 2025. 57 pages, 2 tables, 18 figures, 13 references, 3 appendices, presentation.

Key words: computer hardware, online store, database, html, javascript, react, node.js, mongodb, express, orders, registration.

The thesis is devoted to the development of the “MOYO” computer-hardware online store using modern web technologies: React on the client side and Node.js on the server side.

The main objective is to create an efficient, fast and user-friendly e-commerce platform for selling computer equipment.

The system provides an intuitive interface, quick access to the product catalogue and a convenient order-placement process. A component approach, client–server architecture, RESTful API and object-document data modelling were applied. The frontend is implemented in React with Redux Toolkit; the backend uses Node.js with Express; MongoDB is chosen as the database. The project includes JWT-based authentication, form validation, shopping-cart handling, product sorting and order processing.

The solution can be deployed both for educational purposes and in a commercial environment with minimal adaptation costs.

The result is a fully operating online store equipped with core functional modules. Testing confirmed correct operation of the main user scenarios.

The proposed web application supports business digitalisation, streamlines trading processes, increases product availability for customers and broadens market reach.

The paper examines the requirements for organizing a safe workplace, regulated breaks to reduce strain.

Future enhancements may include integration of payment gateways, logistics services, personal accounts, analytics, customer reviews and mobile-platform adaptation.

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням популярності електронної комерції. Розширення онлайн торгівлі витісняє традиційні форми продажу завдяки зручності, швидкості та широким можливостям для автоматизації процесів. Особливо актуальним стає створення інтернет-магазинів, які забезпечують користувачам зручний доступ до широкого асортименту товарів, а підприємцям – ефективний інструмент ведення бізнесу.

Інтернет-магазини комп'ютерної техніки займають живе місце на ринку, знижуючи особливий попит на цифрові пристрої, аксесуари та програмне забезпечення зростають постійно. Покупці все частіше обирають онлайн-платформи для придбання ноутбуків, моніторів, периферії, компонентів для ПК, орієнтуючись на доступність інформації, можливість порівняння характеристик і цін, а також швидке оформлення замовлення. Тому для компаній, які працюють у сфері продажу комп'ютерної техніки, особливо важливо мати якісний, надійний та функціональний веб-ресурс.

Метою даної кваліфікаційної роботи є розробка інтернет-магазину комп'ютерної техніки "MOYO" з використанням сучасних веб-технологій – React на стороні клієнта та Node.js на стороні сервера. Такий підхід дозволяє створити високопродуктивну, масштабовану та зручну у використанні веб-систему, що відповідає сучасним вимогам до онлайн-торгівлі.

React – це популярна JavaScript-бібліотека, призначена для створення інтерактивних інтерфейсів користувача. Вона забезпечує швидке оновлення сторінок без необхідності певного перезавантаження, що значно підвищує користувацький досвід. Node.js, у свій час, - це серверне середовище виконання JavaScript, яке дозволяє створювати масштабовані та швидкі сервери для рахунку

неблокуючої архітектури. Поєднання цих технологій створює потужну основу для розробки повнофункціонального веб-застосунку.

У кваліфікаційній роботи забезпечується реалізація таких основних функціональних можливостей:

- Перегляд каталогу товарів з можливістю сортування;
- Детальний опис кожного товару;
- Корзина з обліком обраних позицій;
- Оформлення замовлення;
- Система реєстрації та авторизації користувачів;

Крім того, особлива увага приділяється таким аспектам, як зручність використання інтерфейсу (UI/UX), адаптивність дизайну для різних пристроїв, захист персональних даних і надійність серверної частини.

Актуальність теми обумовлена потребою у впровадженні сучасних рішень у сфері онлайн-торгівлі, що забезпечує високу швидкість роботи сайту, масштабованість і зручність у використанні як для покупців, так і для адміністраторів, Застосування React і Node.js дозволяє ефективно використовувати ці завдання відповідно до сучасних тенденцій у розробці веб-застосунків.

Структура кваліфікаційної роботи охоплює аналіз предметної області, обґрунтування вибору інструментів і технологій, проектування архітектурних систем, реалізацію інтерфейсу та серверної частини, а також тестування та оцінку результатів. У результаті буде створено веб-застосунок, який може служити основою для реального інтернет-магазину комп’ютерної техніки.

1 ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНУ

1.1 Аналіз предметної області

Сфера електронної комерції стрімко розвивається, забезпечуючи користувачам зручний спосіб придбання товарів і послуг через мережу інтернет. Інтернет-магазини стали невід'ємною частиною сучасного бізнесу, оскільки дозволять суттєво зменшити витрати на оренду фізичних приміщень, автоматизувати процеси продажу та розширити аудиторію клієнтів

Інтернет-магазин – це веб-застосунок, який надає користувачам можливість переглядати асортимент товарів, отримувати інформацію про них, додавати їх до кошика, оформлювати замовлення та здійснювати оплату. З боку адміністратора – це інструмент для керування товарами, обробки замовлень, аналізу продажів та взаємодії з клієнтами.

У рамках дипломного проєкту розглядається реалізація простого інтернет-магазину комп'ютерної техніки, який має базовий набір функцій, характерних для більшості подібних систем. Для побудови застосунку було вирішено використовувати сучасні веб-технології: React для створення динамічного і зручного інтерфейсу користувача та Node.js для реалізації серверної логіки, обробки запитів і взаємодії з базою даних [1].

Система передбачає дві основні категорії користувачів:

- Звичайний користувач (покупець): має змогу переглядати товари, додавати їх до кошика, оформлювати замовлення, реєструватися та входити в свій акаунт.
- Адміністратор: має доступ до розширених можливостей управління магазином – додавання, редагування і видалення товарів, перегляд замовлень користувачів, зміна їх статусів.

Основні компоненти предметної області включають:

- Товари: мають характеристики, такі як назва, опис, ціна, зображення.

- Кошик: тимчасове сховище для обраних товарів користувача.
- Замовлення: оформлені покупки, які містять дані про товари, покупця та статус обробки.
- Користувачі: особи, які взаємодіють із системою, реєструються та здійснюють покупки.

Оскільки мета проєкту – створити простий, але функціональний інтернет-магазин, буде реалізовано лише базові можливості: перегляд товарів, робота з кошиком, реєстрація та авторизація, оформлення замовлень.

Проаналізувавши особливості предметної області, можна сказати, що розробка інтернет-магазину є актуальною задачею, яка охоплює реальні потреби користувачів і дає змогу реалізувати повноцінний веб-застосунок із практичним застосуванням.

1.2 Формування вимог розробки

Перед початком реалізації проєкту важливо чітко визначити, які саме функції має виконувати система, які вимоги ставляться до її роботи, зручності та безпеки. Це дозволяє на етапі проектування обґрунтовано обрати інструменти та архітектуру для побудови надійного та функціонального веб-застосунку [2].

Веб-сайт інтернет-магазину комп’ютерної техніки має реалізовувати наступні можливості:

- Реєстрація нових користувачів та авторизація зареєстрованих;
- Можливість переглядати товари, зокрема повний список та детальну інформацію про кожен товар;
- Реалізація функції кошика – додавання, редагування кількості та видалення товарів з кошика;
- Оформлення замовлення з введенням контактної інформації;

- Забезпечення швидкої відповіді системи при виконанні базових операцій;
- Кросплатформеність – доступність вебсайту на різних пристроях (ПК, планшет, смартфон).

В процесі реалізації проєкту мають бути враховані наступні структурні особливості:

- Компоненти, які повторюються, повинні бути винесені в окремі елементи;
- Структура всіх компонентів повинна бути уніфікованою, код – упорядкованим і читабельним (див. додаток А);
- Забезпечення зв’язності між клієнтською та серверною частинами проєкту;
- Єдність стилю написання коду як на клієнтській, так і на серверній стороні;
- Захист даних користувачів:
- Усі форми мають проходити валідацію (на клієнті та сервері);
- Паролі повинні зберігатись у зашифрованому вигляді.

Інтерфейс веб-застосунку повинен відповідати таким критеріям:

- Адаптивна верстка – всі елементи інтерфейсу мають коректно відображатись на екранах різного розміру;
- Простий і зрозумілий дизайн, орієнтований на користувачів різного рівня підготовки;
- Сумісність з різними браузерами.

1.3 Опис ключових варіантів використання

На цьому етапі важливо описати основні сценарії взаємодії користувача з розроблюваним веб-додатком. Кожен з описаних варіантів використання охоплює певну дію, які можуть виконуватись в рамках функціонування системи. Такі

сценарії допомагають краще зрозуміти логіку системи, забезпечити повноту реалізації функціоналу та уникнути пропусків у проектуванні. Стислий опис варіантів використання подано в таблиці 1.1.

Таблиця 1.1 – Варіантів використання інтернет-магазину “MOYO”

Актор	Найменування	Формулювання
Admin	Перегляд, додавання, редагування, видалення, товарів, обробка замовлень	Перегляд, додавання, редагування, видалення, товарів, обробка замовлень покупців
Authorized user	Перегляд, сортування, додавання товарів до кошика, оформлення замовлення	Перегляд, сортування, додавання, видалення товарів з кошику, оформлення замовлення
Unauthorized user	Реєстрація, авторизація	Дає можливість використання функціоналу інтернет-магазину без реєстрації

На основі описаних варіантів використання можна чітко уявити, як саме кожен з акторів взаємодіє з системою. Гості можуть переглядати товари, сортувати, реєструватися та авторизуватись. Авторизовані користувачі отримують доступ до розширеного. Адміністратор має змогу керувати товарним асортиментом і контролювати замовлення.

Усі ці взаємодії представимо за допомогою варіантів використання (use-case), що наочно демонструє зв'язки між акторами та функціональними модулями системи. Така діаграма дозволяє краще зрозуміти, які саме функції має реалізовувати сайт, та як між собою пов’язані окремі його компоненти [3].

Адміністратор має можливість переглядати, додавати, редагувати, видаляти, товари з кошика, обробляти замовлення покупців (див. рис. 1.1).

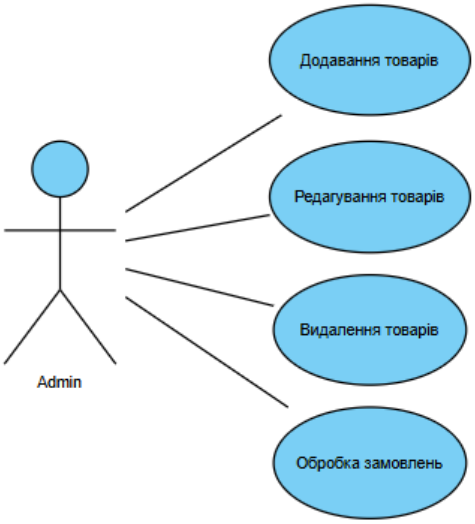


Рисунок 1.1 – діаграма варіантів використання адміністратора

Авторизований користувач має можливість переглядати, додавати, видаляти товари з кошика, оформляти замовлення (див. рис. 1.2).

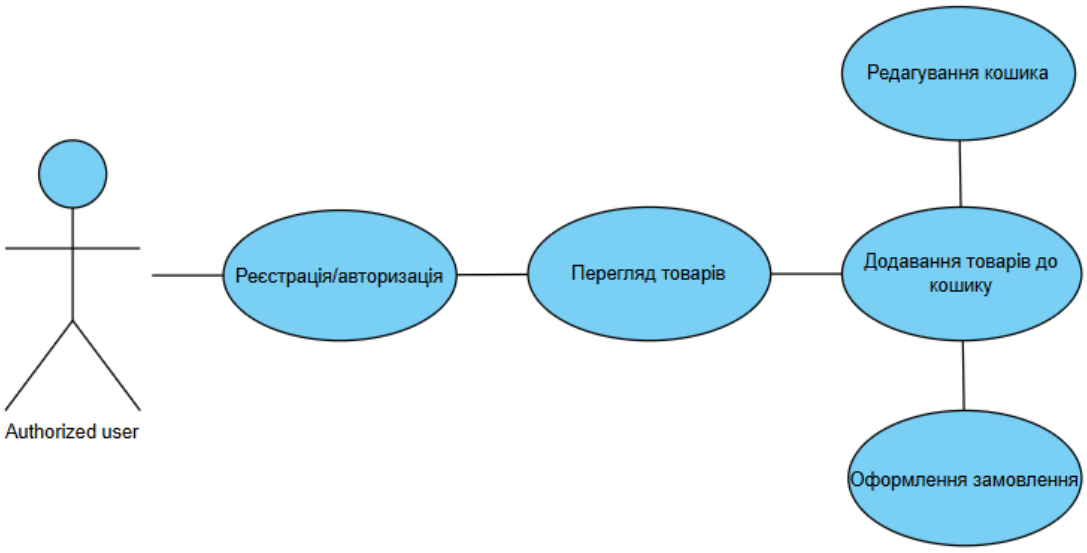


Рисунок 1.2 – діаграма варіантів використання авторизованого користувача

Не авторизований користувач має можливість реєстрації, авторизації та переглядати товари (див.рис. 1.3).

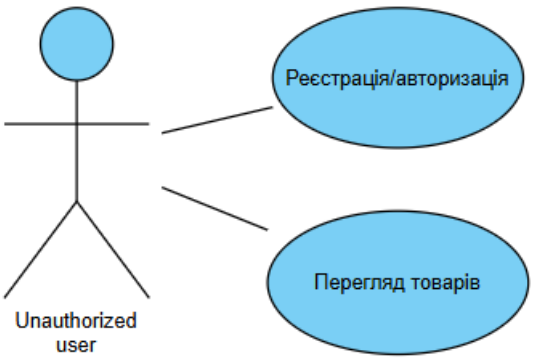


Рисунок 1.3 – діаграма варіантів використання не авторизованого користувача

Порівнюючи варіанти використання для різних ролей, можна зробити висновок, що неавторизовані користувачі мають обмежений доступ до функціоналу вебсайту – їм доступний лише перегляд товарів базова навігація, реєстрація та авторизація Натомість зареєстровані користувачі отримують розширені можливості, зокрема, роботу з кошиком та оформлення замовлень. Адміністратор, у свою чергу, має найвищий рівень доступу: він може керувати товарами, переглядати всі замовлення. Такий розподіл повноважень забезпечує логічну та безпечну модель взаємодії з системою.

1.4 Вибір технологічного стеку розробки

Під час реалізації веб-сайту інтернет-магазину “MOYO” було прийнято рішення використовувати сучасний, гнучкий та масштабований технологічний стек, що забезпечує високу продуктивність, зручність у розробці, підтримку та розширюваність функціоналу в майбутньому.

Загальна архітектура застосунку розділена на три головні рівні: клієнтська частина (frontend), серверна частина (backend) та база даних(database). Обмін даними між ними реалізований за допомогою REST API. Архітектура інтернет-магазину “MOYO” показана на рисунку 1.4 нижче.

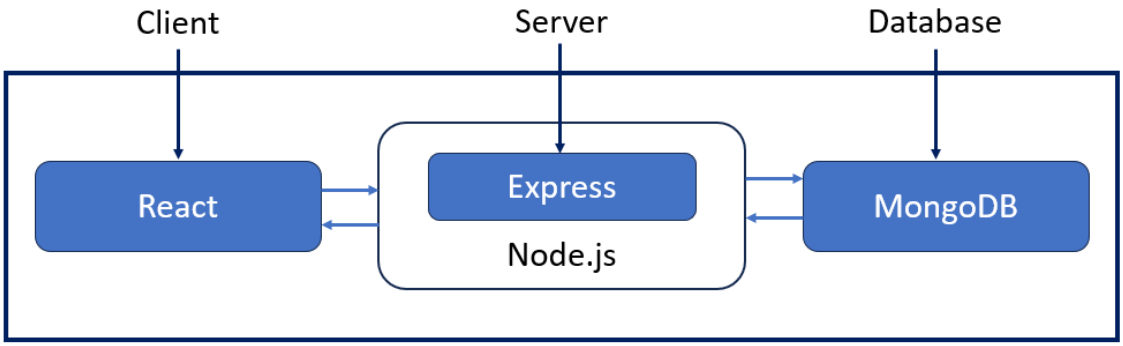


Рисунок 1.4 – Трирівнева архітектура інтернет-магазину “MOYO”

Для побудови інтерфейсу користувача було обрано бібліотеку React – одну з найпопулярніших JavaScript-бібліотек для створення односторінкових додатків. React дозволяє створювати багаторазові компоненти, ефективно оновлює DOM завдяки віртуальній моделі та має широку підтримку в спільноті.

У якості інструмента для збирання та запуску застосунку використано Vite – надшвидкий дев-сервер і збирач модулів, що значно пришвидшує процес розробки порівняно з традиційним Webpack. Стилізація інтерфейсу реалізована з використанням Tailwind CSS, що дозволяє швидко формувати адаптивний дизайн.

Основні інструменти фронтенду:

- React – побудова компонентів UI;
- Vite – середовище розробки та збірки;
- Tailwind CSS – адаптивна стилізація;
- React Router DOM – маршрутизація;
- Redux Toolkit – керування глобальним станом;
- React Hook Form – обробка та валідація форм;
- Framer Motion – анімації.

Для побудови серверної логіки обрано Node.js у поєднанні з фреймворком Express. Такий стек технологій дозволяє створювати легкі та продуктивні REST API, що обслуговують клієнтські запити.

У якості бази даних використовується MongoDB, що чудово підходить для зберігання документів з гнучкою структурою. Зв’язок між Node.js і MongoDB

реалізований через бібліотеку Mongoose, яка забезпечує зручну ORM-обгортку для взаємодії з базою.

Основні інструменти бекенду:

- Node.js – середовище виконання JavaScript на сервері;
- Express.js – побудова REST API;
- MongoDB – база даних NoSQL;
- Mongoose – ORM для MongoDB;
- dotenv – керування середовищем налаштувань;
- bcrypt – хешування паролів;
- jsonwebtoken (JWT) – авторизація користувачів.

Для розробки застосунку використовувалося середовище Visual Studio Code, що забезпечує розширену підтримку JavaScript/TypeScript, інтеграцію з Git, зручні розширення та дебагер. Для контролю версій використовувався Git.

1.5 Обґрунтування використовуваних технологій розробки

У процесі розробки інтернет-магазину комп'ютерної техніки "MOYO" було прийнято рішення використовувати стек технологій, що забезпечує високу продуктивність, масштабованість, зручність розробки та підтримку сучасних функціональних вимог [1]. Основними технологіями стали React, Node.js, Express, MongoDB. Далі наведено обґрунтування вибору кожної з них.

React – це популярна JavaScript-бібліотека, яка використовується для створення динамічних користувацьких інтерфейсів [4, 5]. Основними перевагами React є:

- Компонентний підхід: дозволяє розбивати інтерфейс на незалежні, повторно використовувані компоненти, що спрощує розробку і підтримку коду.

- Virtual DOM: забезпечує високу продуктивність за рахунок мінімізації операцій з реальним DOM.
- Підтримка великої спільноти: завдяки активному розвитку бібліотеки наявні численні рішення, приклади та документація.
- JSX: дає змогу писати розмітку прямо у коді JavaScript, що покращує читабельність та спрощує інтеграцію логіки.

Використання React дозволяє забезпечити швидке оновлення інтерфейсу без перезавантаження сторінки, покращуючи користувацький досвід (UX).

Node.js – це середовище виконання JavaScript на стороні сервера. Його застосування в даному проєкті дозволяє використовувати одну мову програмування як для клієнтської, так і для серверної частини, що спрощує розробку [6].

Основні переваги:

- Неблокуюча (асинхронна) архітектура: дозволяє обробляти велику кількість одночасних запитів без втрати продуктивності.
- Широкий вибір бібліотек (npm): забезпечує гнучкість при реалізації різних функцій — від підключення до бази даних до авторизації користувачів.
- Швидкість та ефективність: завдяки рушію V8 від Google.

Express – це легкий фреймворк для Node.js, який полегшує створення серверних застосунків та REST API [7]. Його використання обґрунтовується такими факторами:

- Простота створення маршрутів та обробників запитів;
- Гнучкість та масштабованість;
- Інтеграція з базами даних, middleware та JWT.

Express дозволив швидко реалізувати логіку взаємодії між клієнтом та сервером.

MongoDB – це документо-орієнтована база даних NoSQL, яка зберігає дані у форматі BSON (розширений JSON). Вибір MongoDB обґрунтовано наступними

перевагами:

Гнучкість структури даних: у порівнянні з реляційними базами даних, MongoDB не вимагає чіткого визначення схеми таблиць, що дозволяє зберігати документи з різною структурою.

- Швидкість розробки: можливість легко змінювати структуру даних під час розробки без складних міграцій.
- Хороша інтеграція з Node.js: за допомогою бібліотеки Mongoose, яка дозволяє визначати схеми, перевіряти типи даних та працювати з MongoDB як з об’єктами.
- Масштабованість: MongoDB легко масштабується горизонтально, що важливо при зростанні кількості даних та навантаження на систему.

MongoDB є оптимальним вибором для сучасних вебзастосунків, які потребують швидкої взаємодії з великими обсягами даних без складної транзакційної логіки.

2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Опис розробки інтернет-магазину

У рамках виконання кваліфікаційної роботи було розроблено сучасний інтернет-магазин “MOYO”, що спеціалізується на продажу комп’ютерної техніки. Головна мета ресурсу – забезпечити зручний доступ до широкого асортименту товарів, спростити процес покупки та підвищити лояльність клієнтів через якісний інтерфейс та функціональність. Сайт складається з таких сторінок: “Головна”, “Каталог”, “Контакти”.

Веб-сайт реалізовано з використанням сучасного стеку веб-технологій. Для розробки клієнтської частини (frontend) використано бібліотеку React, яка забезпечує динамічну побудову інтерфейсу, ефективне оновлення компонентів та високу інтерактивність.

Основна структура сайту організована з урахуванням принципів SPA (Single Page Application), що дозволяє завантажувати сторінки без повного перезавантаження браузера, забезпечуючи плавність та швидкість роботи [8]. Для маршрутизації застосовано бібліотеку React Router, яка дозволяє перемикатись між сторінками магазину (Головна, Каталог, Кошик, Контакти тощо). Фрагменти коду наведено в додатку Б.

У якості бекенда (серверної частини) використовувався Node.js разом з фреймворком Express.js. База даних реалізована на основі MongoDB, що дозволяє зберігати інформацію про товари, користувачів, замовлення та інше у гнучкому форматі.

Для зберігання та обробки користувацьких сесій, а також аутентифікації, використовується бібліотека JWT (JSON Web Token).

Зовнішній вигляд інтерфейсу розроблено з використанням Tailwind CSS – сучасної утилітно-класової системи стилізації, що дозволяє швидко створювати адаптивні та сучасні інтерфейси.

Сторінка “Головна” є першою, яку бачить користувач після переходу на сайт. Основна мета цієї сторінки – ознайомлення користувача з ключовими товарами, асортиментом та функціональністю магазину. Дизайн реалізований таким чином, щоб максимально швидко донести користувачу основну інформацію та заохотити до подальшого перегляду сайту.

У верхній частині розміщено навігаційне меню, що містить основні розділи сайту: “Головна”, “Каталог”, “Контакти”.. Таке меню є доступним на кожній сторінці для зручної навігації.

У центральній частині сторінки реалізований банер з акційною пропозицією – у цьому випадку, наприклад, монітор ASUS TUF Gaming 23.8". Поруч зображення товару розміщується короткий опис і кнопка «Купити зараз».

Нижче розміщені популярні позиції, серед яких корпус ПК ZALMAN N5 TF Black, IP-камера TP-LINK Tapo C510W. Під кожним товаром є кнопка «Переглянути більше», яка веде до каталогу.

Загалом сторінка виконана у простому, сучасному стилі з акцентом на візуальний контент та зручність користування. Вона є відправною точкою для подальшої навігації сайтом і сприяє залученню користувача до вивчення продукції магазину.

Вигляд головної сторінки показано на рисунку 2.1 нижче:

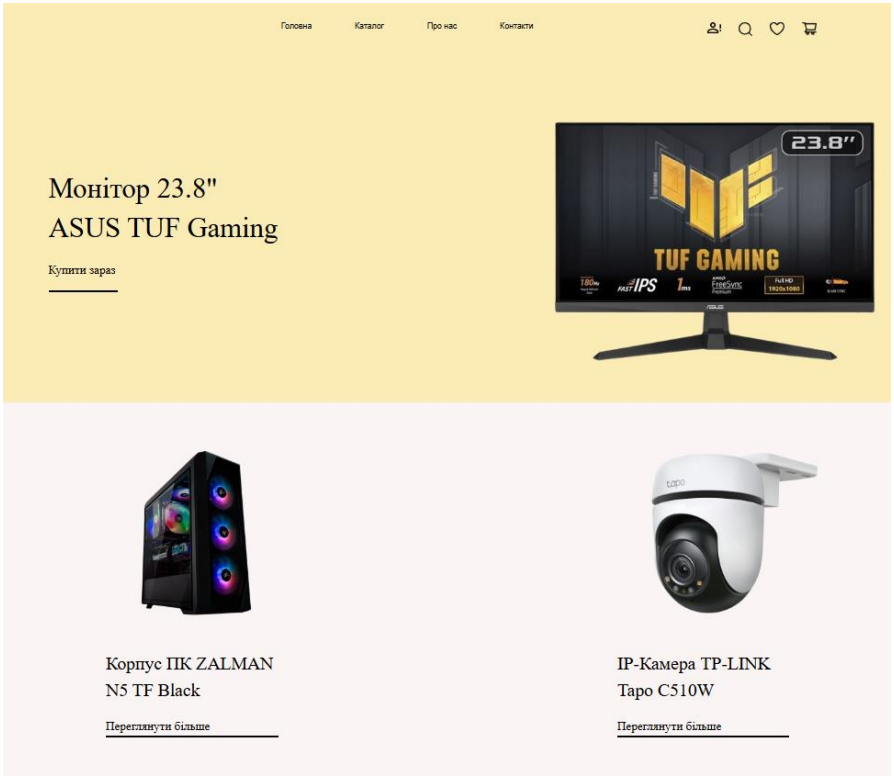


Рисунок 2.1 – Головна сторінка інтернет-магазину “MOYO”

Сторінка “Каталог” призначена для відображення всього асортименту товарів, наявних в інтернет-магазині “MOYO”. Вона є однією з найважливіших частин сайту, оскільки дозволяє користувачу швидко ознайомитися з доступними продуктами, їх цінами та характеристиками.

Нижче знаходяться фільтри, сортування та вибір кількості товарів на сторінці, що дозволяє користувачу зручно налаштувати відображення продукції згідно з його потребами.

Основна частина сторінки — це сітка товарів, де кожен продукт представлено у вигляді карточки з такими елементами:

- Фотографія товару;
- Назва;
- Ціна.

Серед представлених товарів: відеокарти, блоки живлення, корпуси, SSD-диски, кулери, роутери, монітори, планшети для графіки тощо. Наприклад, в каталозі можна знайти відеокарту ASUS GeForce RTX 4060, монітор ASUS TUF

Gaming 23.8”, IP-камеру TP-LINK Tapo C510W та багато інших популярних моделей.

Також під основним вмістом сторінки реалізовано інформаційний блок з перевагами магазину, а саме:

- Безкоштовна доставка для товарів на суму понад 5000 грн;
- 90 днів повернення;
- 100% безпечний платіж.

Вигляд сторінки “Каталог” показано на рисунку 2.2 нижче:

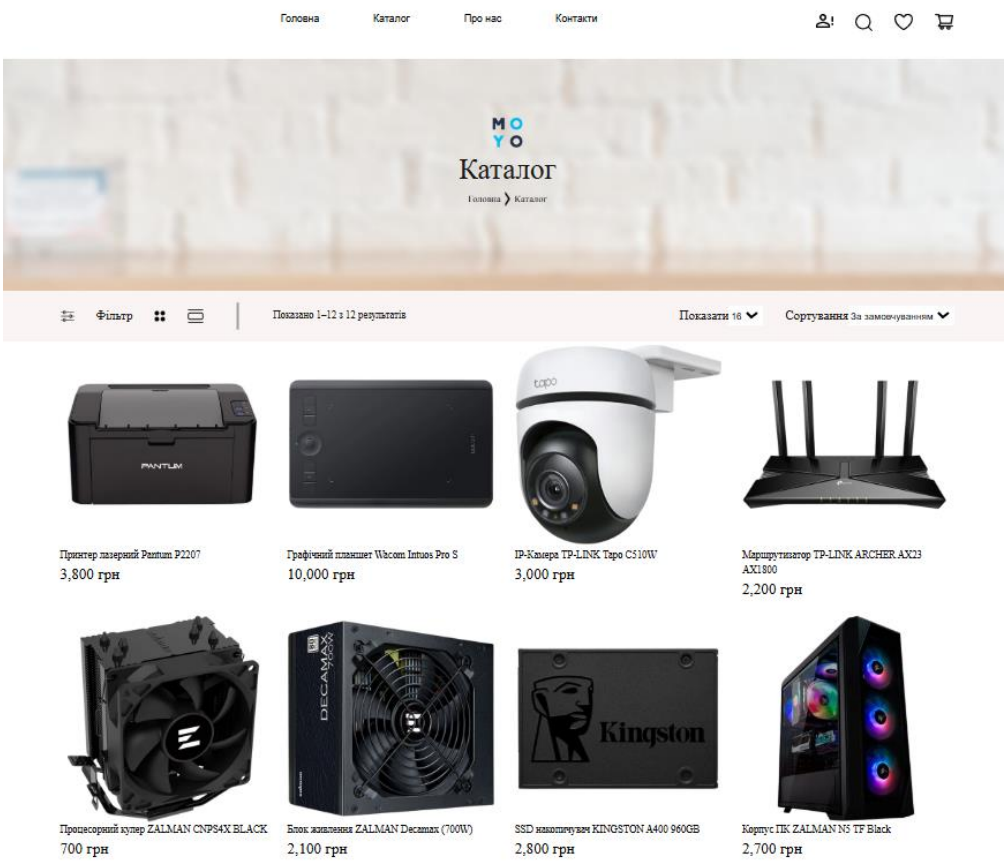


Рисунок 2.2 – Сторінка “Каталог” інтернет-магазину “MOYO”

Сторінка “Контакти” призначена для забезпечення зворотного зв’язку між відвідувачами сайту та адміністрацією магазину. Вона дозволяє клієнтам отримати необхідну контактну інформацію, а також швидко надіслати звернення чи запит.

У верхній частині сторінки знаходиться навігаційна панель та хлібні крихти,

які вказують розташування користувача на сайті (Головна > Контакти), що забезпечує зручну орієнтацію.

Основний блок сторінки містить заголовок “Зв’яжіться з нами”, після якого йде короткий опис з закликом звертатися з будь-яких питань щодо послуг, замовлень або співпраці.

Зліва розміщена контактна інформація:

- Адреса 01030 м. Київ, вул. Богдана Хмельницького, 44;
- Телефон 0 800 570 800;
- Робочий час: Пн–Нд 09:00 до 21:00.

Праворуч розташована форма зворотного зв’язку, яка містить такі поля:

- Ім’я користувача;
- Електронна пошта;
- Тема звернення;
- Повідомлення.

Після заповнення форми користувач може натиснути кнопку «Відправити», після чого запит буде передано до адміністрації магазину.

У нижній частині сторінки повторюється інформаційний блок з перевагами магазину

Вигляд сторінки “Контакти” показано на рисунку 2.3 нижче:

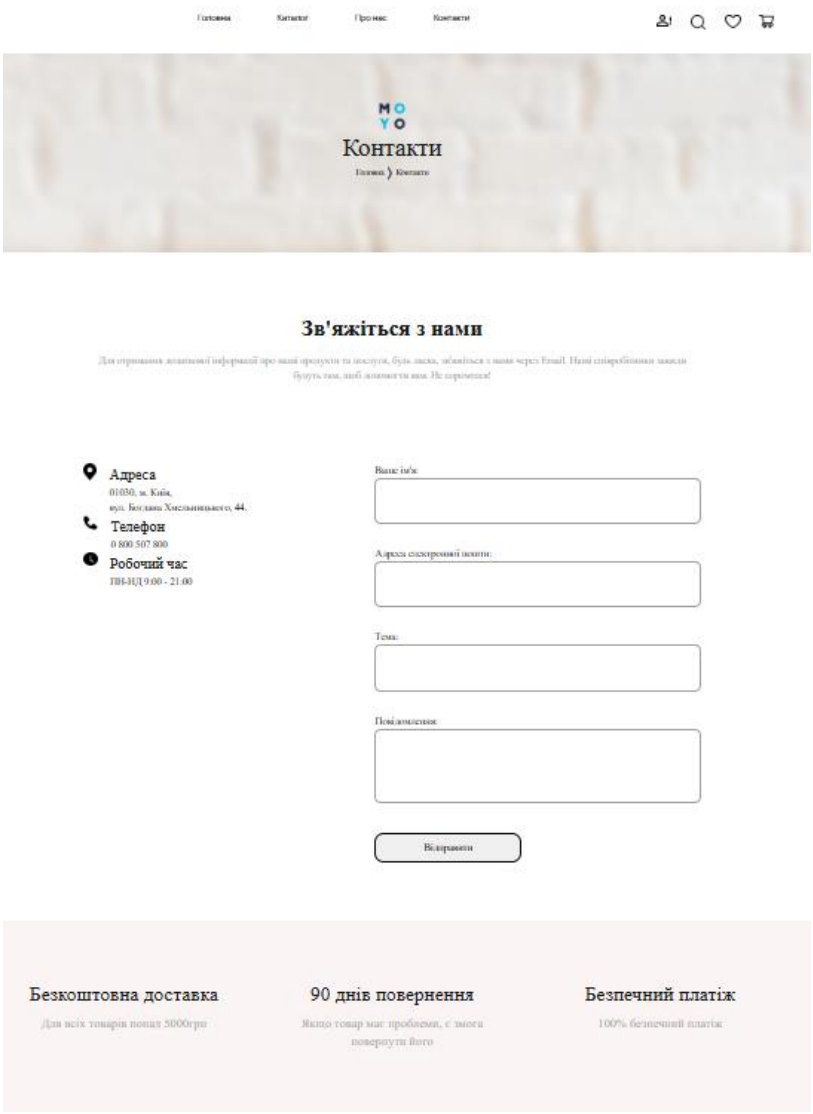


Рисунок 2.3 – Сторінка “Контакти” інтернет-магазину “MOYO”

Сторінка товару є однією з ключових складових інтернет-магазину, оскільки вона забезпечує користувача повною інформацією про конкретний продукт, його технічні характеристики, переваги, вартість та можливість здійснення покупки.

На прикладі сторінки товару “Принтер лазерний Pantum P2207” представлено класичну структуру типового товарного шаблону. У верхній частині розміщено навігаційний ланцюжок (breadcrumb), який вказує шлях до поточної сторінки: Головна > Каталог > Принтер лазерний Pantum P2207. Це покращує юзабіліті та дозволяє користувачеві легко повертатися до попередніх розділів.

Сторінка поділена на дві основні колонки. Ліворуч розміщується велике зображення товару, що дає змогу ознайомитися з його зовнішнім виглядом.

Праворуч подано назву товару, його вартість та короткий текстовий опис, у якому акцентується увага на основних перевагах пристрою.

Нижче розташовано інтерактивні елементи: лічильник для вибору кількості одиниць товару та кнопка “Додати в кошик”, яка ініціює процес покупки. Це дозволяє користувачу безпосередньо зі сторінки здійснити замовлення, не переходячи до інших розділів.

У нижній частині сторінки реалізовано вкладки, що містять додаткову інформацію. Зокрема, у вкладці “Опис” повторюється текст з короткої характеристики, а інші вкладки можуть містити додаткову інформацію та інше.

Вигляд сторінки товару показано на рисунку 2.4 нижче:

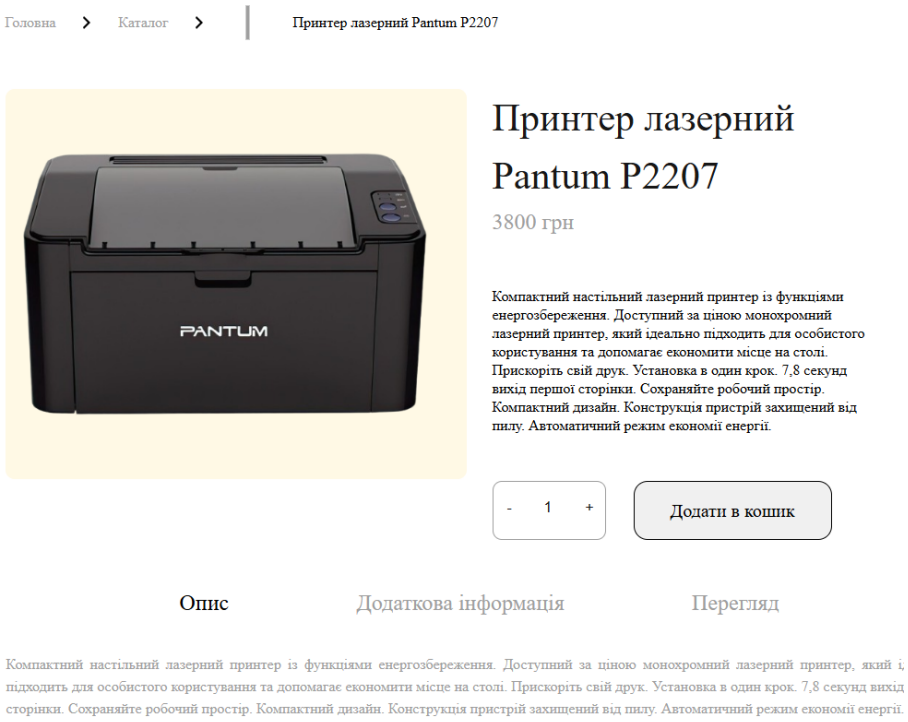


Рисунок 2.4 – Сторінка товару інтернет-магазину “MOYO”

Сторінка “Кошик” виконує функцію попереднього перегляду обраних користувачем товарів перед оформленням замовлення. Вона дозволяє зручно контролювати склад покупки, змінювати кількість позицій та бачити загальну вартість замовлення в режимі реального часу.

На прикладі представленої сторінки кошика відображено два товари:

- Принтер лазерний Pantum P2207 – вибрано 2 одиниці по ціні 3800 грн, що дає проміжну суму 7600 грн;
- Графічний планшет Wacom Intuos Pro S – 1 одиниця по ціні 10000 грн.

Праворуч від зображень кожного товару відображаються такі елементи:

- Назва товару;
- Одинична ціна;
- Поле для зміни кількості;
- Автоматично обчислена сума за кожну позицію.

У нижній частині сторінки розташовано підсумковий блок, який містить:

- Заголовок “Всього в кошику”;
- Розраховану загальну суму замовлення – 17600 грн;
- Кнопку “Оформлення”, що переводить користувача до наступного етапу покупки – введення контактних даних.

Вигляд сторінки “Кошик” показано на рисунку 2.5 нижче:

Рисунок 2.5 – Сторінка “Кошик” інтернет-магазину “MOYO”

Сторінка “Оформлення” інтернет-магазину “MOYO” є завершальним етапом у процесі купівлі товарів. Вона призначена для збору реквізитів покупця та вибору способу оплати.

Інтерфейс сторінки поділений на два основні блоки:

1. Форма введення реквізитів розташована зліва, містить такі обов’язкові поля для заповнення:

- Ім’я;
- Прізвище;
- Адреса;
- Місто;
- Поштовий індекс;
- Телефон;
- Адреса електронної пошти.

Ця інформація необхідна для формування рахунку та доставки товару.

2. Інформаційний блок замовлення розміщений праворуч, включає:

- Перелік вибраних товарів із зазначенням проміжної вартості;
- Загальна сума замовлення – 17600 грн;
- Вибір способу оплати.

У цьому прикладі користувачеві запропоновано варіант:

- Прямий банківський переказ – передбачає оплату за реквізитами, що будуть надіслані після оформлення;
- Кнопку “Оформити замовлення”, що запускає процес завершення покупки.

Під інформаційним блоком розміщено повідомлення щодо обробки персональних даних згідно з політикою конфіденційності, що гарантує безпеку і прозорість обробки інформації.

Вигляд сторінки “Оформлення” показано на рисунку 2.6 нижче:

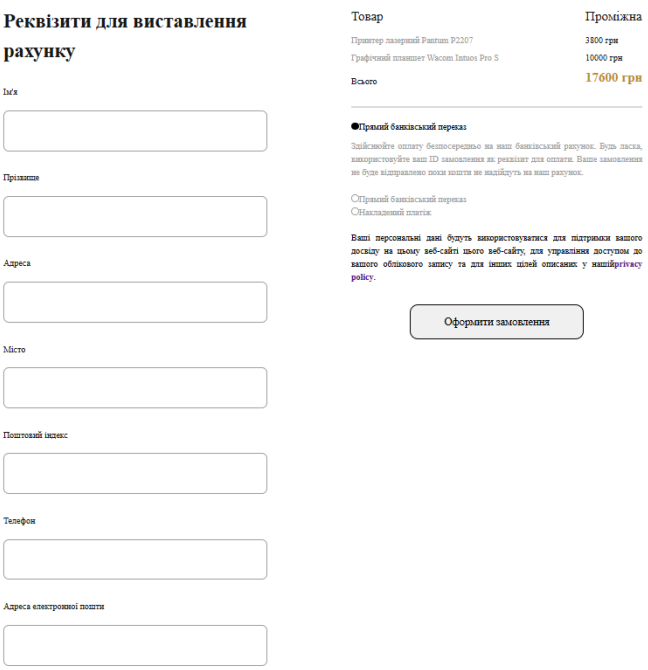


Рисунок 2.6 – Сторінка “Оформлення” інтернет-магазину “MOYO”

У такий спосіб було реалізовано основні сторінки вебзастосунку інтернет-магазину комп’ютерної техніки «MOYO».

2.2 База даних

У процесі розробки вебзастосунку інтернет-магазину комп’ютерної техніки “MOYO” ключовим етапом стало проектування та впровадження бази даних, яка забезпечує ефективне зберігання, пошук, обробку і управління інформацією про користувачів, товари, замовлення, категорії, а також інші сутності, що беруть участь у бізнес-логіці магазину.

Зважаючи на особливості предметної області, а також вимоги до масштабованості, швидкодії та гнучкості структури даних, було обрано сучасну документоорієнтовану NoSQL базу даних MongoDB. Це рішення стало результатом порівняльного аналізу кількох популярних систем керування базами даних, зокрема MySQL та PostgreSQL [9].

Хоча традиційні СКБД, такі як MySQL або PostgreSQL, досі широко використовуються, особливо в банківських та бухгалтерських системах, для інтернет-магазину MongoDB має низку переваг:

Таблиця 2.1 – Переваги MongoDB над MySQL/PostgreSQL

Критерій	MongoDB	MySQL/PostgreSQL
Тип бази	NoSQL (документна)	Реляційна
Гнучкість схеми	Динамічна, поля можна замінювати	Статична, вимагає міграцій
Підтримка JSON	Нативна (BSON)	Часткова
Масштабування	Горизонтальна (шардінг)	Вертикальне
Інтеграція з Node.js	Пряма (через mongoose/mongo драйвер)	Через ORM (наприклад Sequelize)
Продуктивність (CRUD)	Висока при роботі з великою кількістю читань	Висока, але більше під SQL запити

Система бази даних у рамках цього проєкту повинна задовольняти наступні вимоги:

- Гнучкість структури даних. Різні товари можуть мати різні характеристики, тому структура записів має бути легко адаптованою без складних міграцій.
- Масштабованість. Із ростом асортименту продукції, кількості користувачів і замовлень – база даних повинна ефективно обробляти великі об'єми даних.
- Продуктивність. Висока швидкодія при виконанні запитів читання/запису в режимі реального часу (особливо при відображенні списку товарів).
- Інтеграція з Node.js. Оскільки бекенд розроблено з використанням середовища Node.js, бажаною є нативна сумісність з JavaScript-форматом даних (JSON/BSON).

- Простота підтримки. Швидке впровадження змін і розширення без потреби кардинальної перебудови всієї схеми БД.

MongoDB повністю задовольняє зазначені вимоги, пропонуючи при цьому просту і зрозумілу модель збереження даних на основі документів.

У MongoDB дані зберігаються у вигляді документів, об'єднаних у колекції. Один документ є аналогом рядка в реляційній базі, але він має динамічну структуру, тобто кожен документ у колекції може містити інший набір полів.

У проєкті реалізовано такі основні колекції:

- Users – інформація про зареєстрованих користувачів (логін, e-mail, пароль зашифрований за допомогою bcrypt та мітки часу);
- Products – дані про товари (назва, опис, ціна, зображення);

Кожна з цих колекцій пов'язана з іншими за допомогою ідентифікаторів (ObjectId), що забезпечує логічну зв'язність між сутностями. Нижче на рисунку 2.7 продемонструю колекцію Product.

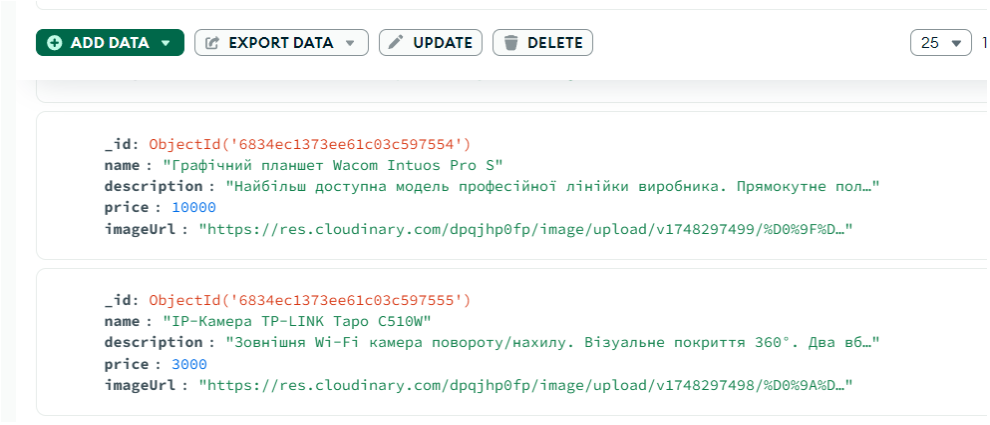


Рисунок 2.7 – Колекція Product

MongoDB особливо зручна, коли структура даних може змінюватися — наприклад, якщо магазин додасть нові категорії товарів, у кожної з яких свої властивості (розмір, колір, потужність, тип роз’єму тощо). У реляційній базі це потребувало б додавання нових стовпців або створення додаткових таблиць, у той час як у MongoDB достатньо просто додати відповідні поля до документа.

Для взаємодії з MongoDB у проєкті використано бібліотеку mongoose, яка забезпечує:

- Створення схем для документів (моделі);
- Автоматичну валідацію даних;
- Зручні методи роботи з колекціями (find, create, update, delete);
- Зв’язки між моделями через populate;
- Гнучке розширення функціональності (middleware, хелпери, віртуальні поля).

Отже, використання MongoDB як основної бази даних для інтернет-магазину “MOYO” є технічно обґрунтованим рішенням, що забезпечує високу продуктивність, масштабованість, простоту у використанні та інтеграцію з Node.js. На відміну від реляційних систем, MongoDB дозволяє ефективно працювати з даними, структура яких постійно змінюється, що є типовим для динамічного середовища електронної комерції. У поєднанні з бібліотекою mongoose ця СКБД створює зручну, надійну та гнучку платформу для реалізації сучасного застосунку.

2.3 Тестування інтернет-магазину

Кожен програмний продукт перед впровадженням повинен проходити обов’язкову перевірку на коректність роботи. У процесі розробки можуть застосовуватись юніт-тести для перевірки окремих функціональних компонентів. Однак, після завершення розробки доцільно використовувати метод тестування «чорної скриньки», який дозволяє виявити зовнішні помилки системи без заглиблення у внутрішню реалізацію. Такий підхід дає змогу швидко визначити потенційні збої, які необхідно усунути до запуску системи [10].

Перед запуском будь-якого веб-додатку в реальне користування необхідно провести повну перевірку його працездатності – від стартових сторінок до

ключових функціональних можливостей. У випадку з розробкою інтернет-магазину “MOYO”, одним із перших кроків перевірки є тестування функціоналу реєстрації та авторизації користувача.

Відповідно до логіки роботи застосунку, після натискання кнопки реєстрації новий користувач повинен бути успішно доданий до бази даних, після чого він має можливість увійти в систему, використовуючи введені дані. Щоб перевірити цей функціонал, переходимо на відповідну сторінку сайту, заповнюємо реєстраційну форму та відправляємо її. Приклад заповнення представлений на рисунку 2.8.

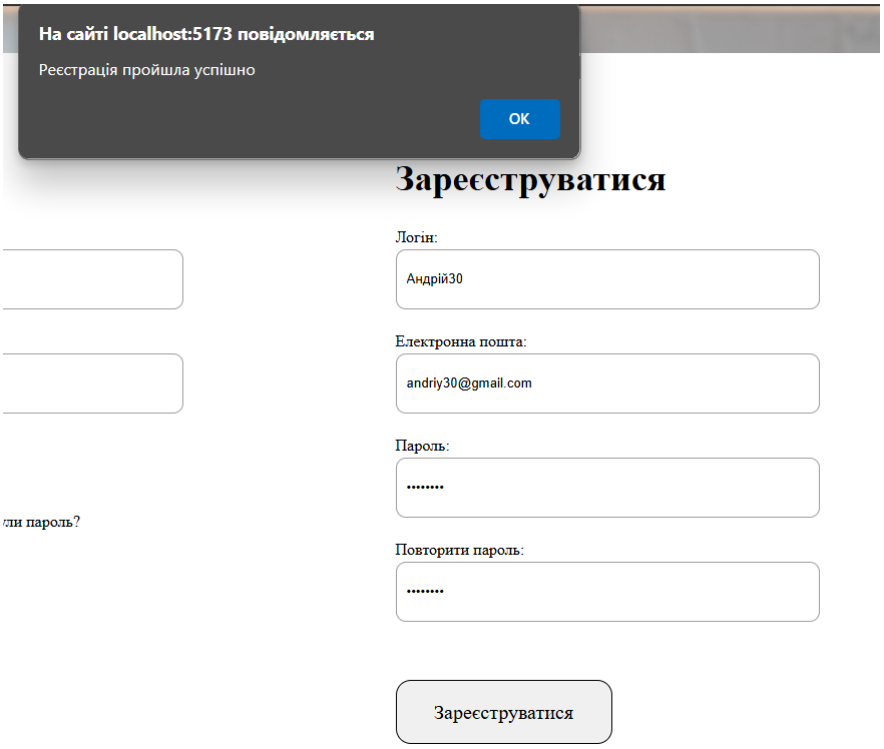


Рисунок 2.8 – Приклад заповнення реєстраційної форми

Після реєстрації користувач повинен зберігатися в базу даних, перевіряємо, а результат зображуємо на рисунку 2.9.

```
_id: ObjectId('6842c32354c0cd339a5c5725')
login : "Андрій30"
email : "andriy30@gmail.com"
password : "$2b$10$lQEl8TYEfGiA2YyfKKMl/uvPBbmZpHy.gHwHTjyBDMsd.y5LThAoG"
createdAt : 2025-06-06T10:29:55.094+00:00
updatedAt : 2025-06-06T10:29:55.094+00:00
__v : 0
```


Рисунок 2.9. – Результат реєстрації користувача

Під час тестування особлива увага приділяється перевірці надійності валідації введеної інформації. Зокрема, якщо користувач намагається зареєструватися, використовуючи електронну пошту, яка вже присутня у базі даних, система повинна відобразити відповідне повідомлення про помилку та заборонити повторне створення облікового запису. Це дозволяє уникнути дублювання акаунтів і підвищує безпеку. Після введення електронної пошти яка вже зареєстрована, користувачу повинно надійти повідомлення про те що користувач з таким email або логіном вже існує, результат тестування зображуємо на рисунку 2.10.

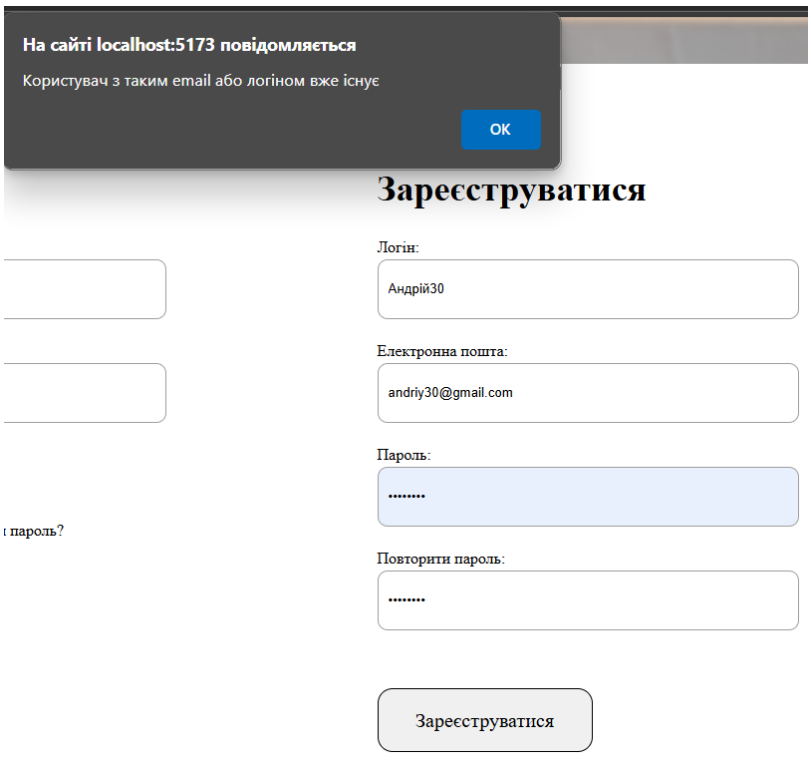


Рисунок 2.10. – Результат введення електронної пошти яка вже зареєстрована

Також тестується ситуація, коли користувач вводить некоректні або неповні дані (наприклад, неправильний формат email, занадто короткий пароль, пропущене поле тощо). У таких випадках система має не допускати завершення реєстрації, виводячи підказки для користувача. Це забезпечує зручність взаємодії та запобігає збереженню невалідної інформації в базі даних. Результат тестування зображуємо на рисунку 2.11.

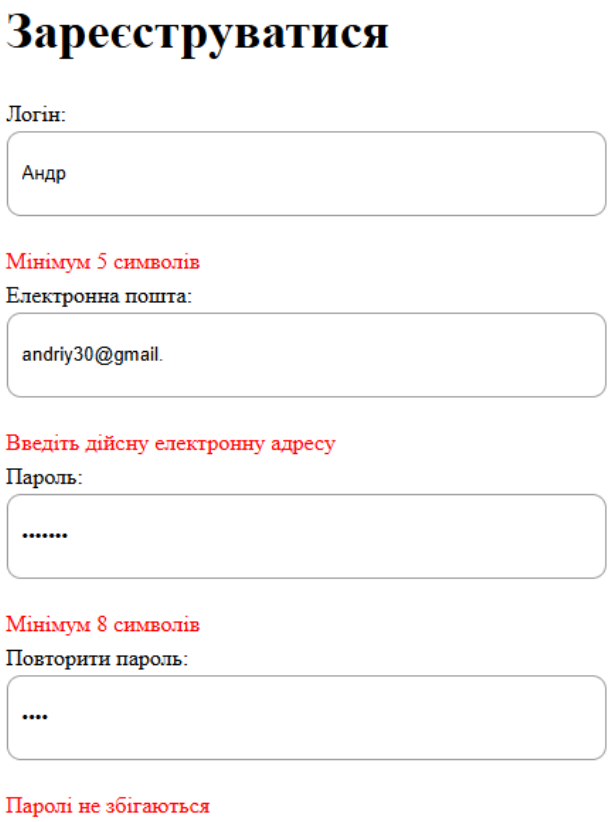


Рисунок 2.11. – Результат введення некоректних та неповних даних

У результаті тестування перевіряється:

- Правильність додавання нового користувача до бази даних;
- Обробка помилок у випадку спроби зареєструвати вже існуючий обліковий запис, введення некоректних та неповних даних.

Цей етап тестування є критично важливим, оскільки забезпечує базову функціональність взаємодії з користувачем та подальший доступ до можливостей системи.

Один із важливих функціоналів сучасного інтернет-магазину – це можливість зручного сортування товарів для користувача. У нашому вебзастосунку реалізовано сортування товарів за ціною – як за зростанням, так і за спаданням.

Щоб протестувати цей функціонал, відкриваємо сторінку каталогу товарів і активуємо опцію сортування. Спочатку обираємо сортування “від дешевших до дорожчих”. У результаті товари мають бути відсортовані в порядку зростання значення поля price, починаючи з найменшої вартості. Візуально перевіряємо правильність відображення: ціни товарів мають поступово зростати. Результат тестування зображуємо на рисунку 2.12.

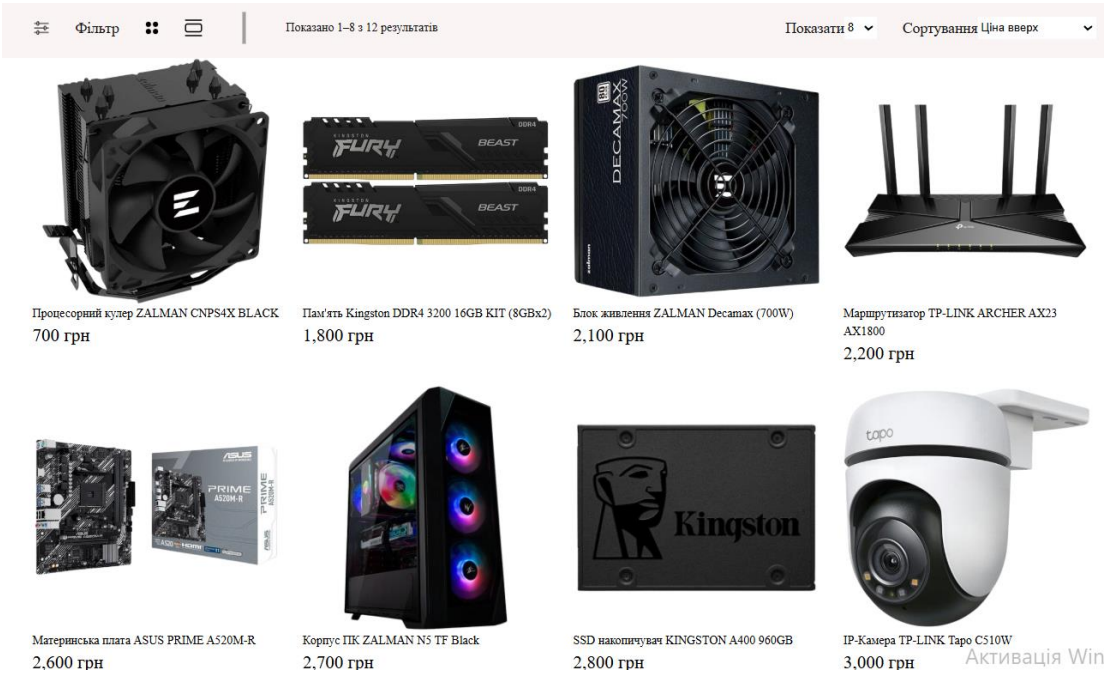


Рисунок 2.12. – Результат сортування за ціною в порядку зростання

Далі обираємо сортування “від дорожчих до дешевших”. Повторно перевіряємо порядок відображення – найдорожчі товари повинні бути першими у списку, а найдешевші – останніми. Якщо система коректно реагує на зміну опції сортування та відображає правильний порядок товарів – вважаємо тест пройденим успішно. Результат тестування зображуємо на рисунку 2.13.

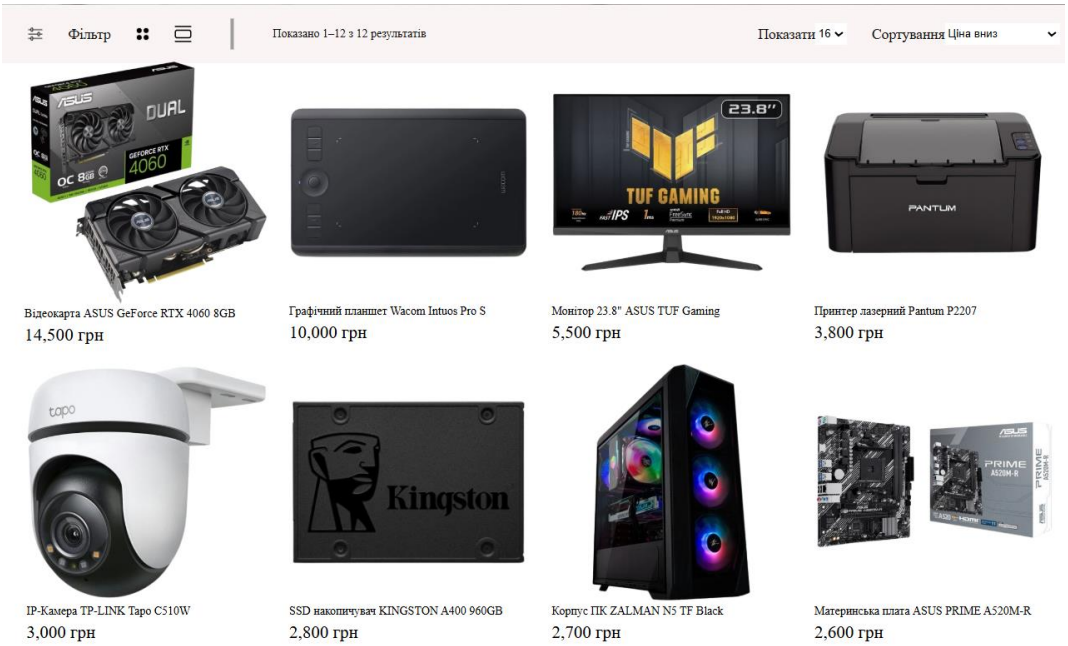


Рисунок 2.13. – Результат сортування за ціною в порядку спадання

Наступним кроком виконується тестування функціоналу додавання товарів до кошика. Для цього переходимо на сторінку будь-якого товару й натискаємо кнопку “Додати в кошик”. Очікувана поведінка системи – обраний товар має з’явитися в кошику користувача.

Одразу після натискання кнопки вискакує вікно “Кошик для покупок” Якщо все працює правильно, обрані товари і кількість потрапляють до кошика. Результат додавання товарів до кошика продемонстровано на рисунку 2.14.

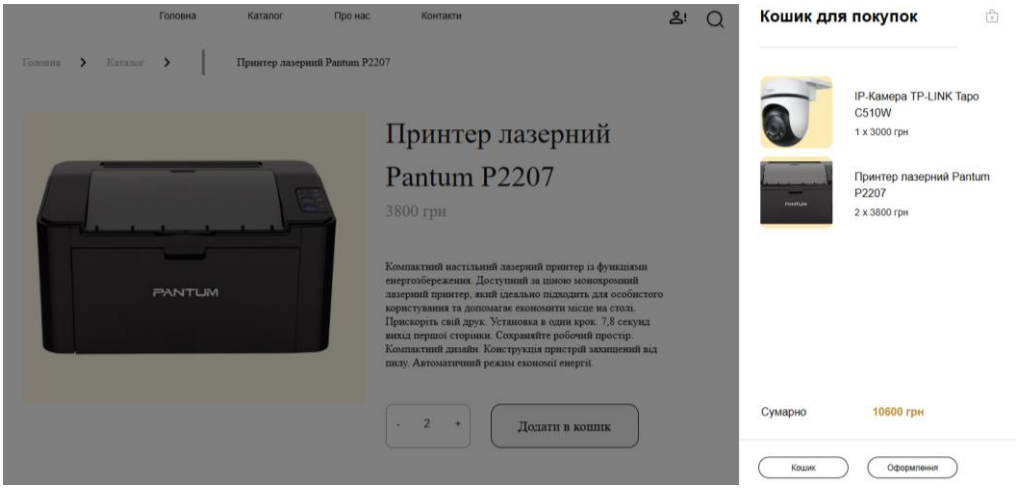


Рисунок 2.14. – Результат додавання товарів до кошика

На наступному етапі відкриваємо сам кошик, щоб переконатися, що доданий товар дійсно присутній у списку. Він повинен містити усю необхідну інформацію: назву, ціну, кількість та сумарну вартість. Результат перевірки присутності товарів у кошику продемонстровано на рисунку 2.15.

Рисунок 2.15. – Результат перевірки присутності товарів у кошику

У результаті тестування було підтверджено, що після додавання товару система коректно:

- Оновлює індикатор кількості товарів у кошику;
- Додає обраний товар до списку замовлення;
- Відображає правильну інформацію в кошику;
- Проводить точний розрахунок вартості замовлення.

Проведені перевірки підтверджують коректну роботу основного функціоналу веб-застосунку. Сайт стабільно виконує всі ключові операції: реєстрацію та авторизацію користувачів, сортування товарів, додавання їх до кошика, а також розрахунок вартості замовлення. Виявлені під час тестування сценарії відповідають очікуваним результатам, що свідчить про готовність системи до використання кінцевими користувачам

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОХОРОНА ПРАЦІ

3.1 Управління та нагляд за безпекою життєдіяльності в Україні.

Безпека життєдіяльності в Україні забезпечується системою цивільного захисту населення. Основою для її функціонування є Кодекс цивільного захисту України, який об’єднав попередні нормативно-правові акти [11]. Головна мета системи – захист населення і територій від наслідків надзвичайних ситуацій техногенного, природного і соціального характеру.

Управління у сфері цивільного захисту здійснюється через Єдину державну систему цивільного захисту, яка включає органи державної влади, місцевого самоврядування, підприємства, установи та добровільні формування. Основні її завдання: попередження надзвичайних ситуацій, інформування населення, ліквідація наслідків, організація евакуації, а також підготовка кадрів [12].

Організацію заходів забезпечують спеціально створені штаби, які відповідають за планування, оповіщення, мобілізацію ресурсів та координацію рятувальних робіт. Складовою частиною є моніторинг надзвичайних ситуацій, який реалізується на галузевому та територіальному рівнях – зокрема ДСНС, Мінекоенерго, місцевими адміністраціями та спеціалізованими лабораторіями.

Виконання політики у сфері захисту забезпечує Державна служба України з надзвичайних ситуацій (ДСНС). Вона контролює виконання вимог законодавства, організовує тренування, відповідає за реагування у кризових ситуаціях. Значну увагу приділено навчанню населення. Це включає об’єктові тренування, командно-штабні навчання та спеціальну підготовку керівного складу, яку здійснює Інститут державного управління у сфері цивільного захисту.

Функціонують також регіональні центри моніторингу і прогнозування, що забезпечують збір, аналіз та інтерпретацію даних для прийняття управлінських рішень. На державному рівні управління здійснює Кабінет Міністрів України, а

реалізацію політики – ДСНС. На місцевому рівні – обласні та районні адміністрації. На об'єктах відповідальність покладено на керівників підприємств.

Для моніторингу, реагування та координації дій у надзвичайних ситуаціях створено Єдину державну систему цивільного захисту (ЄДСЦЗ), яка складається з функціональних та територіальних підсистем. Основною метою функціонування ЄДСЦЗ є реалізація державної політики в сфері цивільного захисту як у мирний час, так і в особливий період. Система охоплює чотири рівні: загальнодержавний, регіональний, місцевий та об’єктовий. У її складі діють постійно функціонуючі комісії з питань техногенно-екологічної безпеки та надзвичайних ситуацій (ТЕБ та НС).

У системі цивільного захисту важливу роль відіграє система моніторингу надзвичайних ситуацій, яка включає спостереження, лабораторний контроль та прогнозування потенційних небезпек. Створена мережа спостереження та лабораторного контролю (МСЛК) на обласному рівні забезпечує збирання й аналіз інформації про стан довкілля, продукти харчування, воду, рівні радіаційного забруднення тощо. Координацію моніторингу здійснює міжвідомча комісія, створена Кабінетом Міністрів України.

Отже, система управління та нагляду за безпекою життєдіяльності в Україні є багаторівневою, інтегрованою в державну політику та побудованою відповідно до вимог Кодексу цивільного захисту України. Вона включає комплекс організаційних, правових, моніторингових і навчальних заходів, що забезпечують підвищення готовності населення до реагування на надзвичайні ситуації та зменшення їх наслідків для людей і довкілля.

3.2 Заходи, що забезпечують безпечні умови роботи.

З метою створення безпечних та комфортних умов праці на комп’ютеризованих робочих місцях необхідно впровадити комплекс заходів, що охоплюють організаційно-технічні, санітарно-гігієнічні, ергономічні та профілактичні складові. Основна мета цих заходів – мінімізувати негативний вплив таких факторів, як перенапруження зору, статичне навантаження, недостатній мікроклімат, шкідливе електромагнітне випромінювання, а також психофізіологічне навантаження, зумовлене тривалим перебуванням за комп’ютером.

Одним з найважливіших аспектів безпечної організації праці є ергономіка робочого місця. Робоче місце користувача ПК повинне мати площу не менше 6 м² та об’єм не менше 20 м³. Відстань між бічними поверхнями відеотерміналів має становити не менше 1,2 м, а між тильними сторонами – не менше 2,5 м. Висота сидіння має відповідати антропометричним параметрам працівника: висота очей над сидінням у чоловіків становить 774 мм, у жінок – 734 мм. Монітор має бути – розташований на відстані 500–600 мм від очей користувача, а його верхній край – на рівні очей або трохи нижче. Клавіатура повинна знаходитись під кутом у межах комфортного нахилу для кистей. Робочий стіл і крісло мають бути регульованими по висоті, а також забезпечувати правильну підтримку спини.

Освітлення також має важливе значення для створення безпечних умов. Робоча зона повинна бути освітлена природним та штучним світлом з коефіцієнтом відбиття від стелі 0,7–0,8, стін – 0,5–0,6, підлоги – 0,3–0,5, інших поверхонь 0,4 – 0,5. Світловий потік повинен падати зліва або ззаду зліва від користувача, уникати утворення відблисків на екрані. Джерела світла повинні забезпечувати рівномірне освітлення. Недостатнє або неправильне освітлення може призводити до зорової втоми, головного болю, а в подальшому – до розвитку астенопії та зниження гостроти зору.

Параметри мікроклімату в приміщенні з ВДТ регламентуються ДСанПіН 3.3.2-007-98. Температура повітря повинна становити 21–24°С взимку та 22–25°С влітку. Вологість має бути в межах 40–60 %, а швидкість руху повітря не повинна перевищувати 0,1–0,2 м/с. Приміщення повинні мати системи природної та примусової вентиляції, з можливістю підтримки оптимального мікроклімату. Поверхні підлоги мають бути антистатичними, наприклад, лінолеум із антистатичним просоченням.

Не менш важливою складовою є регламентація режиму праці. З метою профілактики фізичного та зорового перевантаження необхідно дотримуватись відповідного режиму праці й відпочинку. Згідно з нормативами, для користувачів ПК кожні 60 хвилин роботи повинна передбачатись перерва тривалістю не менше 15 хвилин. При 12-годинній зміні упродовж останніх чотирьох годин рекомендується перерва кожну годину. Для студентів та учнів – не більше 4 годин на день, з перервами кожні 40–45 хвилин.

Окрему увагу слід приділити зменшенню впливу електромагнітних випромінювань. Для цього застосовуються сучасні монітори з низьким рівнем випромінювання, а також захисні екрани та заземлення обладнання. Крім того, важливо уникати розміщення робочих місць біля джерел електромагнітного випромінювання, таких як трансформатори або магістральні кабелі.

Усі вищезазначені заходи відповідають чинним нормативним документам з охорони праці, зокрема ДНАОП 0.00-1.31-99 та ДСанПіН 3.3.2-007-98. Їхнє дотримання дозволяє знизити професійні ризики, запобігти розвитку професійних захворювань, підвищити продуктивність праці та забезпечити комфортне й безпечне робоче середовище для користувачів комп’ютерної техніки [13].

ВИСНОВКИ

У межах цієї дипломної роботи створено повноцінний онлайн-магазин комп’ютерної техніки “MOYO”, побудований на сучасному JavaScript-стеку: React для клієнтської частини, Node.js + Express для серверної логіки та MongoDB як гнучкого сховища даних

Попит на електронну комерцію невпинно зростає, а нові звички користувачів – зокрема прагнення замовляти техніку дистанційно та уникати великих торгових залів – роблять розроблений сервіс затребуваним і конкурентоспроможним

Система вже охоплює ключові сценарії роботи: реєстрацію й автентифікацію з JWT-токенами, каталог, кошик, оформлення замовлення, тож відповідає вимогам технічного завдання і може слугувати мінімально життєздатним продуктом

Під час розробки закріплено знання з дисциплін “Веб-технології” поглиблено навички роботи з мовою JavaScript, бібліотекою React, керуванням станом через Redux, серверним середовищем Node.js та об’єктно-документним моделюванням у MongoDB.

Основне середовище для розробки проекту використано Visual Studio Code з розширеннями для форматування.

Подальші кроки розвитку передбачають інтеграцію платіжних систем, сервісів доставки, систему відгуків і рекомендацій, удосконалення адаптивного дизайну та впровадження CI/CD-процесів, що забезпечить масштабування й утримання високих стандартів якості

Отримані результати демонструють практичну цінність виконаної роботи та підтверджують готовність веб-застосунку до комерційного використання – як базової платформи, що може еволюціонувати разом із потребами ринку й очікуваннями користувачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кашалов Н. В. Сучасні тенденції при розробці інформаційних веб-сайтів з використанням сучасних технологій / Н. В. Кашалов // Збірник тез доповідей VII Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“, 28-29 листопада 2018 року. – Т. : ТНТУ, 2018. – Том 2. – С.68.

2. Як побудувати структуру сайту [Електронний ресурс] – режим доступу: https://sendpulse.ua/blog/website-structure-with-examples

3. Use cases. Як покращити взаємодію користувача з системою [Електронний ресурс] – режим доступу: https://cases.media/en/article/use-cases-yak-pokrashiti-vzayemodiyu-koristuvacha-z-sistemoyu?srsltid=AfmBOorvpJO4n6d01HyhBgPs2oVzZ9ltL5eHEw4xu6Sym9Ec80NJlIGh

4. React.js – Official Website [Електронний ресурс] – режим доступу: https://react.dev/

5. Що таке React JS і для чого він потрібен [Електронний ресурс] – режим доступу: https://dan-it.com.ua/uk/blog/chto-takoe-react-js-i-dlja-chego-on-nuzhen/

6. Node.js – Official Website [Електронний ресурс] – режим доступу: https://nodejs.org/en/

7. Express.js – Official website [Електронний ресурс] – режим доступу: https://expressjs.com/

8. Вербіцький І. Розробка SPA-застосунків на MERN-стек / Вербіцький І. // VI Міжнародна студентська науково-технічна конференція „Природничі та гуманітарні науки. Актуальні питання“, 27-28 квітня 2023. — Т. : ТНТУ, 2023. — С. 118–119. — (Інформаційні технології).

9. MongoDB – Official website [Електронний ресурс] – режим доступу: https://www.mongodb.com/

10. Як тестувати веб-сайт: основні етапи і поради [Електронний ресурс] – режим доступу: https://brainlab.com.ua/uk/blog-uk/yak-testuvati-veb-sayt-osnovn-etapi-poradi

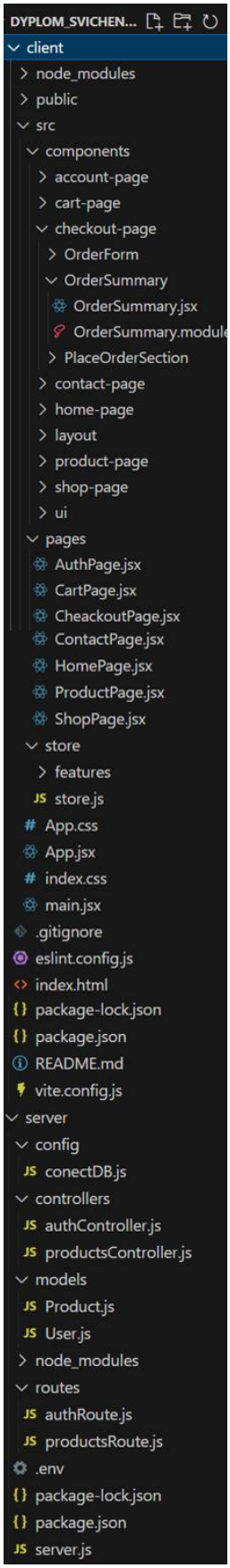
11. Гурик О. До питання безпеки життєдіяльності / О. Гурик, О. Король, В. Сенчишин // Матеріали XXI наукової конференції ТНТУ ім. І. Пулюя, 16-17 травня 2019 року. — Т. : ТНТУ, 2019. — С. 99. — (Матеріалознавство, міцність матеріалів і конструкцій, будівництво).

12. Кодекс цивільного захисту України: Закон України від 02.10.2012 № 5403-VI // Відомості Верховної Ради України. — 2013. — № 34-35. — Ст. 458.

13. Конспект лекцій з курсу «Охорона праці в галузі» / Укладачі: Яскілка В.Я., Олійник М.З. – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016. – 56 с.

ДОДАТКИ

ДОДАТОК А – Структура проєкту та її компоненти



ДОДАТОК Б – Лістинг коду веб-застосунку

Лістинг коду 1 – Файл App.jsx

```
import './App.css'
import { BrowserRouter, Routes, Route } from 'react-router-dom'
import ShopPage from './pages/ShopPage'
import HomePage from './pages/HomePage'
import ContactPage from './pages/ContactPage'
import ProductPage from './pages/ProductPage'
import CartPage from './pages/CartPage'
import CheackoutPage from './pages/CheackoutPage'
import AuthPage from './pages/AuthPage'

function App() {
    return (
        <BrowserRouter>
            <Routes>
                <Route path='/' element={<HomePage />} />
                <Route path='/shop' element={<ShopPage />} />
                <Route path='/shop/:id' element={<ProductPage
/>} />
                <Route path='/contact' element={<ContactPage
/>} />
                <Route path='/cart' element={<CartPage />} />
                <Route path='/checkout' element={<CheackoutPage
/>} />
                <Route path='/auth' element={<AuthPage />} />
            </Routes>
        </BrowserRouter>
    )
}

export default App
```

Лістинг коду 2 – Файл Product.js

```
import mongoose from 'mongoose'

const productsSchema = new mongoose.Schema(
    {
        name: {
            type: String,
            required: true,
        },
        description: {
            type: String,
            required: true,
        },
        price: {
            type: Number,
            required: true,
        },
        imageUrl: {
            type: String,
            required: true,
        },
    },
    {
        timestamps: true,
    }
)

const Product =
    mongoose.models.products || mongoose.model('products',
productsSchema)

export default Product
```

Лістинг коду 3 – Файл ShopPage.jsx

```
import Header from '../components/layout/Header/Header'
import InfoRow from '../components/ui/InfoRow/InfoRow'
import Footer from '../components/layout/Footer/Footer'
import FilterBar from '../components/shop-
page/FilterBar/FilterBar'
import HeroBanner from '../components/ui/HeroBanner/HeroBanner'
import { useState, useEffect } from 'react'
import Catalog from '../components/shop-page/Catalog/Catalog'

export default function ShopPage() {
    const [filters, setFilters] = useState({
        page: 1,
        limit: 16,
        sort: 'default',
    })
    const [products, setProducts] = useState([])
    const [totalPages, setTotalPages] = useState(0)
    const [isLoading, setIsLoading] = useState(true)
    const [viewType, setViewType] = useState('grid')
    const [totalProducts, setTotalProducts] = useState(0)
    const fetchProducts = async () => {
        try {
            setIsLoading(true)
            const params = new URLSearchParams({
                page: filters.page.toString(),
                limit: filters.limit.toString(),
                sort: filters.sort || 'default',
            })
            const response = await
fetch(`http://localhost:5000/products?${params}`)
            if (!response.ok) {
                throw new Error('Помилка при отриманні даних')
            }
            const data = await response.json()
```

```
            console.log('products:', data.products)
            setTotalPages(data.totalPages || 0)
            setProducts(data.products || [])
            setTotalProducts(data.totalProducts || 0)
        } catch (error) {
            console.error('Помилка при завантаженні
продуктів:', error)
            setProducts([])
        } finally {
            setIsLoading(false)
        }
    }
    useEffect(() => {
        fetchProducts()
    }, [filters])
    return (
        <>
            <Header />
            <HeroBanner title='Каталог' href='/shop' />
            <FilterBar
                filters={filters}
                setFilters={setFilters}
                products={products}
                setViewType={setViewType}
                totalProducts={totalProducts}
                isLoading={isLoading}
            />
            <Catalog
                products={products}
                totalPages={totalPages}
                filters={filters}
                setFilters={setFilters}
                isLoading={isLoading}
                viewType={viewType}
            />
```

```
            <InfoRow />
            <Footer />
        </>
    )
}
```

Лістинг коду 4 – Файл ShopPage.jsx

```
import styles from './Catalog.module.scss'
import { Link } from 'react-router-dom'
import { motion } from 'framer-motion'

const Catalog = ({
    products = [],
    totalPages,
    filters,
    setFilters,
    isLoading,
    viewType,
}) => {
    const handlePageChange = (page) => {
        setFilters((prevFilters) => ({
            ...prevFilters,
            page,
        }))
    }

    const MotionLink = ({ to, children, ...props }) => {
        return (
            <Link to={to}>
                <motion.div {...props}>{children}</motion.div>
            </Link>
        )
    }

    const formatPrice = (price) => {
```

```
        return new Intl.NumberFormat('en-US').format(price)
    }

    const visibleProducts = Array.isArray(products)
        ? products.filter(Boolean)
        : []

    return (
        <section className={`${styles.catalog} mb-24`}>
            <div className={styles.container}>
                <div className={viewType === 'grid' ? styles.grid
: styles.flex}>
                    {isLoading ? (
                        <p
                            style={{
                                fontSize: 20,
                                marginTop: 20,
                            }}
                        >
                            Завантажуються товари...
                        </p>
                    ) : visibleProducts.length > 0 ? (
                        visibleProducts.map((product) => (
                            <MotionLink
                                whileHover={{ scale: 1.1 }}
                                to={`/shop/${product._id}`}
                                key={product._id}
                                className='product'
                            >
                                <img
                                  className={styles.product_img}
                                    src={product.imageUrl}
                                    alt={product.name}
                                />
                                  <h3 className={`${styles.name}
```

```
mt-4 mb-3`}>{product.name}</h3>
                                                                <p
className={styles.price}>{`${formatPrice(
                                        product.price
                                    )} грн`}</p>
                                </MotionLink>
                            ))
                        ) : (
                            <p
                                style={{
                                    fontSize: 20,
                                    marginTop: 20,
                                    color: '#333',
                                    textAlign: 'center',
                                }}
                            >
                                Немає товарів у наявності
                            </p>
                        )}
                    </div>

                    {/* Пагинация */}
                    {!isLoading && totalPages > 1 && (
                      <div className='pagination flex gap-9 justify-
center mt-9'>
                            {Array.from({ length: totalPages }, (_,
i) => i + 1).map((page) => (
                                <motion.button
                                    whileHover={{ border: '2.5px
solid #000000' }}
                                    transition={{ duration: 0 }}
                                    className={`${styles.btn} ${
                                        page === filters.page ?
styles.active : ''
                                    }`}
```

```
                                key={page}
                                disabled={page === filters.page}
                                                onClick={()  =>
handlePageChange(page)}
                            >
                                {page}
                            </motion.button>
                        ))}
                        <motion.button
                               whileHover={{ border: '2px solid
#000000' }}
                            transition={{ duration: 0 }}
                            className={`${styles.btn_next}`}
                            onClick={() => {
                                if (filters.page < totalPages) {
                                        setFilters({ ...filters,
page: filters.page + 1 })
                                }
                            }}
                        >
                            Next
                        </motion.button>
                    </div>
                )}
            </div>
        </section>
    )
}

export default Catalog
```

ДОДАТОК В – Диск із кваліфікаційною роботою бакалавра