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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інтернет-магазину ”PixelPhone” із застосуванням MERN-стеку

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Назарук О.Ю.
(підпис) (прізвище та ініціали)

Керівник _____ Мацюк Г.Р.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2025

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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_20_» _червня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Назарук Олександр Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтернет-магазину "PixelPhone" із застосуванням MERN-стеку

Керівник роботи Мацюк Галина Ростиславівна, кандидат соціолог. наук, доцент кафедри ІМ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-446*

2. Термін подання студентом завершеної роботи 10 червня 2025 р.

3. Вихідні дані до роботи є результати аналізу сучасних e-commerce платформ та технологій. Проєкт повинен включати каталог товарів, кошик, а також комплексну систему безпеки з автентифікацією користувачів та розмежуванням доступу на основі ролей.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області та постановка завдання. 1.1. Огляд сучасних платформ електронної комерції. 1.2. Обґрунтування вибору стеку технологій MERN. 1.3. Постановка завдання на розробку інтернет-магазину "PixelPhone". 1.4. Висновок до першого розділу. Розділ 2. Проєктування інтернет-магазину "PixelPhone". 2.1. Проєктування архітектури вебзастосунку. 2.2. Проєктування бази даних у середовищі MongoDB. 2.3. Проєктування клієнтської частини на React. 2.4. Висновок до другого розділу. Розділ 3. Програмна реалізація та тестування інтернет-магазину "PixelPhone". 3.1. Налаштування середовища розробки та інструментарій. 3.2. Розробка серверної частини (Backend) на Node.js та Express.js. 3.3. Розробка клієнтської частини (Frontend) на React. 3.4. Реалізація системи безпеки та контролю доступу (RBAC). 3.5. Тестування розробленого застосунку. 3.6. Висновок до третього розділу. Розділ 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік використаних джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В. С., кандидат технічних наук, доцент кафедри МТ	02.06.2025	05.06.2025

7. Дата видачі завдання __ 07 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.05.2025	
2.	Підбір та опрацювання літературних джерел на тему	08.05.2025-15.05.2025	
	розробки інтернет магазину із застосуванням MERN-		
	стеку		
3.	Оформлення розділу «Аналіз предметної області та	16.05.2025-23.05.2025	
	постановка розробки інтернет-магазину з MERn-стек»		
4.	Оформлення розділу «Проєктування інтернет-	23.05.2025-01.06.2025	
	магазину "PixelPhone"»		
5.	Оформлення розділу «Програмна реалізація та	23.05.2025-01.06.2025	
	тестування інтернет-магазину "PixelPhone"»		
6.	Виконання завдання до підрозділу «Безпека	02.06.2025-05.06.2025	
	життєдіяльності»		
7.	Виконання завдання до підрозділу «Основи охорони	02.06.2025-05.06.2025	
	праці»		
8.	Оформлення кваліфікаційної роботи	06.06.2025-10.06.2025	
9.	Нормоконтроль	11.06.2025-13.06.2025	
10.	Перевірка на плагіат	16.06.2025	
11.	Попередній захист кваліфікаційної роботи	18.06.2025	
12.	Захист кваліфікаційної роботи	23.06.2025	

Студент _____ __ Назарук О.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ __ Мацюк Г.Р.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інтернет-магазину ”PixelPhone” із застосуванням MERN-стеку // Кваліфікаційна робота освітнього рівня «Бакалавр» // Назарук Олександр Юрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2025 // С. 61, рис. – 3, табл. – 6, кресл. – , додат. – 4, бібліогр. – 38.

Ключові слова: MERN, React, Node.js, MongoDB, JWT, автентифікація, контроль доступу на основі ролей (RBAC), REST API, інтернет-магазин.

Кваліфікаційна робота присвячена розробці повностекового веб-додатку – інтернет-магазину "PixelPhone" на стеку MERN.

В першому розділі кваліфікаційної роботи проведено аналіз предметної області, обґрунтовано вибір технологічного стеку та сформовано детальне технічне завдання на розробку.

В другому розділі спроєктовано архітектуру застосунку, включаючи структуру REST API, схеми бази даних MongoDB та компонентну архітектуру клієнтської частини на React.

В третьому розділі детально описано процес програмної реалізації серверної та клієнтської частин. Особливу увагу приділено імплементації багаторівневої системи безпеки, що включає хешування паролів, автентифікацію на основі JWT та контроль доступу на основі ролей (RBAC). Також представлено результати функціонального тестування.

У розділі «Безпека життєдіяльності, основи охорони праці» проаналізовано умови праці розробника, розглянуто ергономічні вимоги та проведено розрахунок штучного освітлення.

Об’єкт дослідження: Процес розробки веб-додатків для електронної комерції на стеку MERN.

Предмет дослідження: Методи та засоби проєктування й реалізації системи безпечної автентифікації та контролю доступу на основі ролей.

ANNOTATION

Development of the "PixelPhone" Online Store Using the MERN Stack // Qualification work of the educational level «Bachelor» // Nazaruk Oleksandr // Ternopil Ivan Pulyu National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, group SNs-42 // Ternopil, 2025 // P. 61, fig. – 3, tabl. – 6, chair. – , annexes. – 4, references – 38.

Keywords: MERN, React, Node.js, MongoDB, JWT, authentication, Role-Based Access Control (RBAC), REST API, e-commerce store.

The qualification thesis is dedicated to the development of a full-stack web application – the "PixelPhone" e-commerce store on the MERN stack.

The first chapter provides an analysis of the subject area, justifies the choice of the technology stack, and formulates a detailed technical specification for the development.

The second chapter outlines the application's architecture, including the REST API structure, MongoDB database schemas, and the component architecture of the React client-side.

The third chapter details the process of software implementation of both server and client sides. Special attention is paid to the implementation of a multi-layered security system, which includes password hashing, JWT-based authentication, and Role-Based Access Control. The results of functional testing are also presented.

The "Life Safety and Labor Protection" chapter analyzes the working conditions of a developer, reviews ergonomic requirements, and includes a calculation of artificial lighting.

Object of research: The process of developing e-commerce web applications on the MERN stack.

Subject of research: Methods and tools for designing and implementing a system for secure authentication and Role-Based Access Control.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – прикладний програмний інтерфейс, набір визначень та протоколів для створення та інтеграції програмного забезпечення.

BSON (англ. Binary JSON) – формат бінарної серіалізації, що використовується для зберігання документів у базі даних MongoDB.

CRUD (англ. Create, Read, Update, Delete) – акронім, що позначає чотири базові функції, які використовуються для роботи з даними: створення, читання, оновлення та видалення.

DOM (англ. Document Object Model) – об'єктна модель документа, незалежний від платформи та мови програмний інтерфейс, що дозволяє програмам та скриптам отримувати доступ до вмісту HTML, XHTML і XML-документів, а також змінювати вміст, структуру і оформлення таких документів.

JWT (англ. JSON Web Tokens) – відкритий стандарт для створення токенів доступу, що використовуються для безпечної передачі інформації між сторонами у вигляді JSON-об'єкта.

MERN (MongoDB, Express.js, React, Node.js) – набір (стек) технологій для розробки повностекових вебзастосунків, де кожен компонент базується на JavaScript.

ODM (англ. Object Document Mapper) – техніка програмування, що пов'язує об'єкти коду з документами в документо-орієнтованій базі даних (наприклад, Mongoose для MongoDB).

RBAC (англ. Role-Based Access Control) – контроль доступу на основі ролей; модель розмежування доступу, в якій права надаються не конкретним користувачам, а їх ролям.

REST (англ. Representational State Transfer) – архітектурний стиль взаємодії компонентів розподіленого застосунку в мережі, що використовується для побудови веб-сервісів.

SPA (англ. Single Page Application) – односторінковий застосунок; вебзастосунок, який завантажує єдину HTML-сторінку, а подальші оновлення контенту відбуваються динамічно за допомогою JavaScript.

БД – База даних.

ВДТ – Візуальний дисплейний термінал.

ТЗ – Технічне завдання.

ЗМІСТ

ВСТУП

Актуальність теми. В умовах стрімкого розвитку цифрової економіки та зростання ринку мобільних пристроїв, електронна комерція є однією з ключових галузей. Особливо гострою є конкуренція у ніші продажу споживчої електроніки, зокрема смартфонів, де для залучення та утримання клієнтів бізнесу необхідно пропонувати унікальний користувацький досвід та високий рівень сервісу.

Сучасні споживачі очікують від інтернет-магазинів не лише широкого асортименту, а й високої швидкодії, зручного інтерфейсу та, що найважливіше, гарантій безпеки персональних даних та платежів. Це вимагає створення динамічних односторінкових застосунків (SPA), які забезпечують миттєвий відгук та плавну навігацію без перезавантаження сторінок, що є стандартом для сучасних веб-платформ. Питання безпеки стає першочерговим, оскільки будь-який інцидент, пов'язаний з витоком платіжної інформації чи персональних даних, може завдати непоправної шкоди репутації бренду.

Хоча існує багато готових SaaS-платформ, вони часто обмежують гнучкість та можливості кастомізації, не дозволяючи повною мірою реалізувати унікальні бізнес-вимоги та складні механізми безпеки. Тому розробка індивідуальних веб-додатків з використанням сучасних технологічних стеків, таких як MERN (MongoDB, Express.js, React, Node.js), є надзвичайно актуальним завданням. Цей стек дозволяє побудувати єдину, гомогенну екосистему на базі JavaScript, що прискорює розробку та спрощує реалізацію складних клієнт-серверних архітектур з надійними REST API та гнучкими системами контролю доступу. Таким чином, створення кастомного, швидкого та безпечного інтернет-магазину на стеку MERN дозволяє не лише задовольнити високі вимоги ринку, а й отримати значну конкурентну перевагу.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка повнофункціонального інтернет-магазин "PixelPhone" для продажу смартфонів, що забезпечує зручний користувацький досвід та ефективне адміністрування.

Задачі:

– Проаналізувати сучасні підходи та технології для створення інтернет-магазинів.

– Обґрунтувати вибір стеку технологій MERN (MongoDB, Express.js, React, Node.js).

– Спроєктувати архітектуру клієнтської та серверної частин застосунку.

– Розробити структуру бази даних для зберігання інформації про товари, користувачів та замовлення.

– Реалізувати основний функціонал інтернет-магазину.

– Провести тестування розробленого застосунку.

Об'єкт дослідження: Процес розробки вебзастосунків для електронної комерції на стеку MERN.

Предмет дослідження: Методи, технології та засоби проєктування й реалізації клієнт-серверної архітектури, системи безпечної автентифікації та контролю доступу на основі ролей в рамках MERN-стеку.

Практичне значення одержаних результатів. Розроблений інтернет-магазин "PixelPhone" є готовим до розгортання програмним продуктом, що може бути використаний як основа для реального комерційного проєкту. Реалізована модель безпеки та контролю доступу може слугувати практичним прикладом для створення інших захищених веб-додатків.

Структура й обсяг роботи: Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, переліку використаних джерел та додатків. Загальний обсяг роботи становить 61 сторінок, вона містить 3 рисунків, 6 таблиць. Перелік джерел налічує 38 найменувань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА РОЗРОБКИ ІНТЕРНЕТ МАГАЗИНУ З MERN-СТЕК

1.1 Огляд сучасних платформ електронної комерції

Розвиток інформаційних технологій, зокрема глобальної мережі Інтернет, кардинально змінив підходи до ведення бізнесу, зробивши електронну комерцію однією з ключових галузей світової економіки. Для проєкту інтернет-магазину "PixelPhone", що орієнтований на нішевий ринок продажу смартфонів, правильний вибір технологічної основи є критичним фактором успіху, що впливає на все: від користувацького досвіду до майбутньої масштабованості.

Центральним елементом будь-якого бізнесу в цій сфері є вебзастосунок, що виступає як основний інструмент взаємодії з клієнтом. Вебзастосунок — це програма, в якій клієнтом є веб-браузер, а сервером — вебсервер. Логіка такого застосунку зосереджена на сервері, тоді як браузер відповідає за відображення інформації та передачу даних від користувача. Однією з головних переваг цього підходу є кросплатформеність, оскільки клієнти не залежать від конкретної операційної системи користувача. У контексті інтернет-магазину "PixelPhone", це означає, що вся логіка, пов'язана з каталогом товарів, обробкою замовлень та автентифікацією користувачів, буде реалізована на серверній частині, що забезпечить високий рівень безпеки та цілісність даних.

Сьогодні на ринку існує велика кількість готових рішень для створення інтернет-магазинів, які можна умовно поділити на дві великі категорії: SaaS-платформи та платформи з відкритим вихідним кодом.

SaaS-рішення (Software as a Service) – це хмарні сервіси, що працюють за моделлю підписки, надаючи готовий до використання інструментарій, включаючи хостинг та технічну підтримку.

- Shopify: Одна з найпопулярніших платформ, відома своєю простотою та інтуїтивно зрозумілим інтерфейсом. Вона ідеально підходить для швидкого

запуску магазину без залучення розробників, але можливості для глибокої кастомізації обмежені.

- BigCommerce: Конкурент Shopify, що пропонує потужний набір вбудованих функцій та гнучкі інструменти для SEO, орієнтований на бізнеси, що швидко зростають. Таким чином, SaaS-платформи пропонують бізнесу компроміс: вони значно знижують поріг входу та прискорюють запуск, але ціною цього є певна уніфікація та залежність від екосистеми постачальника послуг [1].

Платформи з відкритим вихідним кодом (Open Source) надають повну свободу в модифікації, але вимагають самостійного налаштування хостингу та забезпечення безпеки.

WooCommerce – плагін для WordPress, що є надзвичайно гнучким рішенням. Його продуктивність та безпека повністю залежать від якості хостингу та компетенцій розробника.

Magento (Adobe Commerce) – потужна платформа, орієнтована на великий бізнес, що потребує високого рівня кастомізації. Проте вона є складною та вимагає значних ресурсів для розробки та підтримки. Це формує інший підхід до створення онлайн-бізнесу, де пріоритетом є не швидкість запуску, а максимальна гнучкість та можливість побудови унікального брендового досвіду. конкуренції у ніші продажу смартфонів

Проаналізувавши ці дві категорії готових рішень, стає очевидним, що для реалізації специфічних вимог інтернет-магазину "PixelPhone" вони є неоптимальними. SaaS-платформи не нададуть достатньої гнучкості для реалізації запланованого інтерактивного інтерфейсу та унікальних функцій порівняння товарів. З іншого боку, розгортання та підтримка Magento вимагатиме невиправдано великих ресурсів для проєкту такого масштабу. Тому, для досягнення максимального контролю над функціоналом та користувацьким досвідом, було прийнято рішення про індивідуальну розробку.

Нижче наведено порівняльну таблицю (Таблиця 1.1), що узагальнює ключові характеристики розглянутих підходів.

Таблиця 1.1 – Порівняльний аналіз підходів до створення інтернет-магазину

Критерій	SaaS (Shopify, BigCommerce)	Open Source (WooCommerce, Magento)	Індивідуальна розробка (MERN)
Гнучкість та Кастомізація	Низька/Середня. Обмежена доступними темами та додатками.	Висока. Можливість повної зміни коду та функціоналу.	Максимальна. Повний контроль над кожним аспектом застосунку.
Вартість	Щомісячна підписка + комісії з транзакцій. Вартість зростає з ростом бізнесу.	Умовно безкоштовно (плагін), але вимагає витрат на хостинг, розробку, теми та розширення.	Високі початкові витрати на розробку, але відсутність ліцензійних платежів та комісій.
Швидкість розробки	Дуже висока. Магазин можна запустити за кілька годин/днів.	Середня/Низька. Вимагає часу на налаштування, встановлення та кастомізацію.	Низька. Вимагає найбільше часу на проєктування, розробку та тестування.
Технічні навички	Мінімальні. Інтуїтивно зрозумілий інтерфейс для нетехнічних користувачів.	Середні (для WooCommerce) та високі (для Magento).	Високі. Необхідна кваліфікована команда розробників.
Підтримка та безпека	Включена у вартість підписки. Платформа відповідає за хостинг, оновлення та безпеку.	Відповідальність лежить на власнику магазину. Потрібно самостійно налаштовувати хостинг та оновлення.	Повністю лежить на команді розробників та власнику.

Отже, вибір технологічної основи є стратегічним рішенням. Для проєкту інтернет-магазину "PixelPhone" індивідуальна розробка із застосуванням MERN-стеку є найбільш обґрунтованим шляхом. Цей підхід, хоч і вимагає більших початкових інвестицій у розробку порівняно з готовими рішеннями, забезпечує безпрецедентну гнучкість, повний контроль над даними та можливість створення унікального користувацького досвіду, що є ключовим для конкуренції у ніші продажу смартфонів [2]. Подальший аналіз у роботі буде присвячений детальному обґрунтуванню, проєктуванню та реалізації саме такого рішення.

1.2 Обґрунтування вибору стеку технологій MERN

Для реалізації інтернет-магазину "PixelPhone" було обрано стек технологій MERN, що є акронімом чотирьох ключових компонентів: MongoDB, Express.js, React та Node.js. Цей набір інструментів дозволяє створювати високопродуктивні, масштабовані та гнучкі повностекові вебзастосунки, використовуючи JavaScript на всіх рівнях розробки — від клієнтської частини до бази даних [3] .

Проведемо огляд компонентів стеку MERN:

Node.js — це серверне середовище виконання JavaScript, побудоване на рушії V8 від Google Chrome. Його головна перевага полягає в асинхронній, неблокуючій моделі вводу-виводу, що робить його ідеальним для розробки швидких та масштабованих мережевих застосунків [5]. Для інтернет-магазину "PixelPhone" це означає здатність ефективно обробляти велику кількість одночасних запитів від користувачів без значних затримок.

Express.js – це мінімалістичний та гнучкий вебфреймворк для Node.js, який став де-факто стандартом для побудови серверної частини застосунків. Він надає надійний набір функцій для створення вебсерверів та розробки REST API (Representational State Transfer Application Programming Interface). У проєкті інтернет-магазину "PixelPhone" Express.js відповідає за маршрутизацію запитів, обробку HTTP-запитів (GET, POST, DELETE тощо) та організацію проміжного програмного забезпечення (middleware), що дозволяє структурувати логіку серверної частини та забезпечити її стабільну роботу [6].

React – це відкрита JavaScript-бібліотека, розроблена компанією Meta (раніше Facebook) для створення динамічних та інтерактивних інтерфейсів користувача. React дозволяє будувати інтерфейс із незалежних та повторно використовуваних компонентів, що значно спрощує розробку та підтримку складних застосунків. Його головна особливість — використання віртуального DOM, що дозволяє мінімізувати кількість реальних маніпуляцій з DOM-деревом і, як наслідок, досягти високої продуктивності [4]. Для інтернет-

магазину "PixelPhone" це забезпечить швидке оновлення вмісту сторінки, наприклад, при додаванні товару в корзину чи фільтрації каталогу, без необхідності перезавантаження всієї сторінки. У порівнянні з альтернативними стеками, такими як MEAN (з Angular), вибір React (літера "R" у MERN) для проєкту "PixelPhone" був зумовлений його великою екосистемою, нижчим порогом входження для розробників та компонентною архітектурою, що ідеально підходить для створення повторно використовуваних елементів інтерфейсу, таких як картка товару чи елемент корзини [8] [34].

MongoDB – це документо-орієнтована NoSQL система керування базами даних, яка зберігає дані у гнучкому, JSON-подібному форматі під назвою BSON [7]. На відміну від реляційних баз даних, MongoDB не вимагає жорстко визначеної схеми, що дозволяє легко змінювати структуру даних у процесі розробки. Для інтернет-магазину "PixelPhone" це особливо корисно, оскільки дозволяє зберігати інформацію про товари з різними наборами характеристик в одній колекції. Окрім того, гнучкість MongoDB дозволяє легко реалізувати складні системи фільтрації та пошуку товарів за динамічним набором характеристик, що є критично важливим функціоналом для будь-якого сучасного інтернет-магазину.

Ключовою перевагою MERN-стеку полягає в його синергії та гомогенності. Використання JavaScript на всіх етапах розробки – від інтерфейсу на React до сервера на Node.js/Express.js та запитів до MongoDB (через Mongoose ODM) – значно спрощує процес розробки. Команді розробників не потрібно перемикатися між різними мовами програмування та парадигмами, що зменшує ймовірність помилок і прискорює роботу.

Формат даних BSON, що використовується в MongoDB, легко трансформується в JSON, який є стандартним форматом для обміну даними в вебзастосунках та нативно підтримується в JavaScript. Це забезпечує безшовну передачу даних між клієнтом та сервером без необхідності складних перетворень. Саме ця комбінація гнучкості, продуктивності та єдиної

технологічної основи робить MERN-стек оптимальним вибором для розробки сучасного інтернет-магазину "PixelPhone".

1.3 Постановка завдання на розробку інтернет-магазину "PixelPhone"

Це технічне завдання (ТЗ) визначає цілі, вимоги та етапи розробки програмного продукту — інтернет-магазину "PixelPhone". Грамотно складене ТЗ є основою для створення сучасного та функціонального вебзастосунку, що відповідає очікуванням користувачів та цілям бізнесу.

1.3.1 Найменування та область застосування

Найменування: Інтернет-магазин "PixelPhone".

Область застосування: Комерційна діяльність у сфері електронної торгівлі (e-commerce). Продукт призначений для організації онлайн-продажу смартфонів та супутніх аксесуарів.

Створення спеціалізованого інтернет-магазину "PixelPhone" є відповіддю на високу конкуренцію на ринку електроніки. На відміну від універсальних маркетплейсів, даний проєкт фокусується на конкретній ніші, що дозволяє надати користувачам більш глибоку експертизу, зручні інструменти для порівняння моделей та якісний клієнтський сервіс.

1.3.2 Призначення розробки

Призначенням розробки є створення сучасного, швидкого та надійного вебзастосунку, що надає користувачам можливість:

- Ознайомитися з детальним каталогом смартфонів, переглядаючи їх технічні характеристики, фотографії та описи.

- Використовувати зручні інструменти для фільтрації та сортування товарів за ключовими параметрами (виробник, ціна, обсяг пам'яті, діагональ екрана).
- Додавати товари до віртуального кошика та керувати його вмістом.
- Проходити процедуру реєстрації та автентифікації для збереження історії замовлень.
- Безпечно оформлювати замовлення онлайн.

Для адміністратора системи інтернет-магазин "PixelPhone" повинен надавати інструменти для:

- Керування каталогом товарів (додавання, редагування, видалення).
- Перегляду всіх отриманих замовлень для їх подальшої обробки.

1.3.3 Вимоги до програмного забезпечення

Вимоги до програмного забезпечення визначають, яким чином система повинна функціонувати та які технічні характеристики мати. Вони поділяються на функціональні, що описують поведінку системи, та технічні/нефункціональні, що визначають її атрибути якості.

Функціональні вимоги описують конкретні дії, які користувач може виконувати в системі.

1. Клієнтська частина: Інтерфейс користувача повинен бути інтуїтивно зрозумілим та надавати повний доступ до основного функціоналу магазину [36]. Він має включати головну сторінку для привернення уваги до акцій та популярних товарів, а також сторінку каталогу з реалізацією динамічної фільтрації та сортування для зручного пошуку. Користувачі повинні мати можливість переглядати детальну інформацію про товар на окремій сторінці. Ключовим елементом є інтерактивний кошик, що дозволяє не лише додавати товари, а й змінювати їх кількість чи видаляти. Система має забезпечувати безпечну реєстрацію та авторизацію користувачів, а також зручну форму для фінального оформлення замовлення.

2. Адміністративна частина: Цей функціонал має бути доступним виключно авторизованим користувачам з відповідними правами. Захищена панель адміністратора повинна надавати повний набір інструментів для керування каталогом товарів, що включає операції створення, читання, оновлення та видалення (CRUD). Окрім того, адміністратор повинен мати доступ до інтерфейсу для перегляду всіх замовлень, зроблених клієнтами, для їх подальшої обробки та аналізу.

Технічні та нефункціональні вимоги:

1. Стек технологій: Розробка має бути виконана на стеку MERN (MongoDB, Express.js, React, Node.js), що забезпечує гнучкість та високу продуктивність.

2. Кросбраузерність: Веб-застосунок повинен коректно відображатися та функціонувати в останніх версіях популярних веб-браузерів (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge).

3. Адаптивність: Дизайн має бути адаптивним та забезпечувати зручне користування як на настільних комп'ютерах, так і на мобільних пристроях (смартфонах, планшетах) [10].

4. Супроводжуваність: Код проєкту має бути чистим, добре структурованим та прокоментованим, що спростить його подальшу підтримку, модифікацію та розширення функціоналу.

1.3.4 Вимоги до програмної документації

Проєктна документація має включати такі компоненти, що описують процес розробки та структуру застосунку:

- Загальний опис: Інформація про призначення інтернет-магазину, його цільову аудиторію та основні функції.
- Опис архітектури: Схеми, що пояснюють клієнт-серверну взаємодію, структуру REST API та компонентну архітектуру React.

- Структура бази даних: Опис колекцій та полів у MongoDB, ER-діаграма.
- Опис системи безпеки: Детальна документація, що пояснює механізми хешування паролів, JWT-автентифікації та реалізації контролю доступу на основі ролей (RBAC).
- Інструкція користувача: Пояснення щодо роботи з клієнтською частиною та панеллю адміністратора.
- Календарний план: Терміни виконання основних етапів розробки.

1.3.5 Вимоги до продуктивності та надійності

Основними вимогами до продуктивності та надійності є:

- Швидкодія: час завантаження основних сторінок (головна, каталог) не повинен перевищувати 3 секунди при стабільному інтернет-з'єднанні. Час відповіді сервера на API-запити не повинен перевищувати 500 мс.
- Надійність: система повинна забезпечувати стабільну роботу при одночасному доступі до 100 користувачів.
- Безпека:
  - Передача даних між клієнтом та сервером має відбуватися зашифрованим протоколом HTTPS.
  - Паролі користувачів повинні зберігатися у базі даних виключно у хешованому вигляді з використанням криптографічного алгоритму bcrypt.
  - Доступ до захищених ресурсів API та клієнтських маршрутів повинен контролюватися за допомогою JWT-токенів з обмеженим терміном дії.
  - Система повинна чітко розмежовувати права доступу між звичайними користувачами (user) та адміністраторами (admin).

1.3.6 Стадії та етапи розробки

Процес розробки інтернет-магазину "PixelPhone" включає три основні стадії:

1. Розробка технічного завдання: аналіз вимог, погодження та затвердження даного ТЗ.
2. Робоче проєктування: проєктування архітектури, розробка програмного забезпечення, створення документації та тестування.
3. Впровадження: Підготовка до розгортання та передача готового програмного продукту.

1.3.7 Порядок тестування та прийому сайту

Тестування є заключним етапом розробки, що гарантує якість та відповідність продукту поставленим вимогам. Процес тестування включатиме:

- Функціональне тестування: перевірка коректності роботи всього основного функціоналу: реєстрації, додавання товару в кошик, роботи фільтрів у каталозі, процесу оформлення замовлення, функцій адміністративної панелі.
- Тестування верстки: перевірка відповідності відображення на різних пристроях та у різних браузерах (кросбраузерність), а також адаптивності дизайну.
- Тестування зручності використання (Usability): оцінка інтуїтивності навігації та загальної легкості взаємодії користувача з інтерфейсом.
- Тестування безпеки: перевірка захищеності адміністративної панелі та API-ендпоінтів від несанкціонованого доступу. Емуляція дій від імені трьох ролей (неавторизованого гостя, звичайного користувача та адміністратора) для підтвердження коректної роботи системи розмежування прав.
- Тестування продуктивності: оцінка швидкості завантаження сторінок та реакції системи під час пікових навантажень.

Приймання готового програмного продукту здійснюється після успішного завершення всіх етапів тестування та виправлення виявлених дефектів.

1.4 Висновок до першого розділу

У першому розділі кваліфікаційної роботи було проведено комплексний аналіз предметної області розробки веб-додатків для електронної комерції. Розглянуто та порівняно існуючі підходи до створення інтернет-магазинів, зокрема SaaS-платформи, рішення з відкритим вихідним кодом та індивідуальну розробку. На основі аналізу було зроблено висновок, що для досягнення максимальної гнучкості, повного контролю над функціоналом та створення унікального користувацького досвіду для проєкту "PixelPhone", оптимальним є шлях індивідуальної розробки.

Було обґрунтовано вибір технологічного стеку MERN (MongoDB, Express.js, React, Node.js) як єдиної, гомогенної екосистеми, що використовує JavaScript на всіх рівнях додатку, що значно спрощує та прискорює процес розробки [37].

На завершення розділу було сформовано детальне технічне завдання на розробку інтернет-магазину "PixelPhone". У завданні визначено ключові функціональні вимоги до клієнтської та адміністративної частин, а також нефункціональні вимоги до продуктивності, надійності та, що особливо важливо, до системи безпеки. Вимоги до безпеки включають реалізацію безпечної автентифікації користувачів на основі JWT-токенів, хешування паролів та розмежування прав доступу на основі ролей.

Таким чином, у першому розділі було закладено теоретичний та концептуальний фундамент, що дозволяє перейти до наступного етапу — детального проєктування архітектури, бази даних та компонентів майбутнього застосунку.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ "PIXELPHONE"

2.1 Проєктування архітектури вебзастосунку

Архітектура веб застосунку є фундаментальним етапом розробки, що визначає його структуру, спосіб взаємодії компонентів, масштабованість та надійність [35]. Для інтернет-магазину "PixelPhone" було обрано сучасну клієнт-серверну архітектуру з розділенням логіки на клієнтську та серверну частини, які комунікують між собою через REST API [11].

Основою такого підходу є чіткий розподіл завдань. Клієнтська частина, реалізована на бібліотеці React, функціонує у веб-браузері користувача і відповідає за відображення інтерфейсу та взаємодію з користувачем. Серверна частина, розроблена на Node.js та Express.js, виконує всю основну бізнес-логіку: обробку запитів, роботу з базою даних, автентифікацію та авторизацію.

Важливим аспектом проєктування є логічна структура сайту, яка визначає ієрархію сторінок та зручність навігації. Існує кілька підходів до структурування. Найпростішим є лінійний, де сторінки переглядаються у суворій послідовності, подібно до читання книги. Більш складним варіантом є гратчаста структура, що дозволяє переходити з будь-якої сторінки на будь-яку іншу, але є надлишковою для більшості проєктів.

Найпопулярнішим та універсальним на сьогодні є деревоподібний вид структури. Він дозволяє групувати інформацію на різних рівнях ієрархії, створюючи логічні розділи та підрозділи. Саме цей варіант є оптимальним для інтернет-магазину "PixelPhone", оскільки він дозволяє природно організувати шлях користувача: від головної сторінки до каталогу, потім до конкретного товару, і, нарешті, до оформлення замовлення.

Загальна структура вебзастосунку представлена у Додатку А. Вона відображає ієрархію сторінок та зв'язки між ними, включаючи головну сторінку, з якої можна перейти до ключових розділів: «Продукти», «Новини», «Про нас» та «Корзина».

Комунікація між клієнтською та серверною частинами відбувається за допомогою REST API (Representational State Transfer Application Programming Interface) [9]. Сервер надає набір кінцевих точок (endpoints), до яких клієнт може звертатися за допомогою стандартних методів HTTP для отримання, створення, оновлення чи видалення даних [12]. Архітектура API для інтернет-магазину "PixelPhone" визначає три основні ресурси:

1. Продукти (/api/products): ресурс для керування каталогом товарів.

- GET /api/products – Отримати список усіх товарів (публічний).
- GET /api/products/:id – Отримати детальну інформацію про один товар за його ідентифікатором (публічний).
- POST /api/products – Створити новий товар (лише для адміністратора).
- PUT /api/products/:id – Оновити інформацію про товар (лише для адміністратора).
- DELETE /api/products/:id – Видалити товар (лише для адміністратора).

2. Користувачі (/api/users): ресурс для реєстрації та автентифікації користувачів.

- POST /api/users/register – Зареєструвати нового користувача.
- POST /api/users/login – Автентифікувати користувача та отримати токен доступу.

3. Замовлення (/api/orders): ресурс для керування замовленнями.

- POST /api/orders – Створити нове замовлення на основі вмісту кошика (для авторизованих користувачів).
- GET /api/orders/my – Отримати історію замовлень поточного користувача (для авторизованих користувачів).
- GET /api/orders/all – Отримати список усіх замовлень у системі (лише для адміністратора).

Такий підхід забезпечує чітке розділення відповідальності, дозволяє незалежно розробляти та тестувати клієнтську і серверну частини, а також

створює гнучку основу для подальшого розвитку функціоналу інтернет-магазину "PixelPhone".

2.2 Проєктування бази даних у середовищі MongoDB

Для зберігання даних інтернет-магазину "PixelPhone" було обрано документо-орієнтовану систему керування базами даних (СКБД) MongoDB. Це система з відкритим вихідним кодом, яка не потребує опису жорсткої схеми таблиць і займає нішу між швидкими та масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними у формуванні запитів. Такий вибір обґрунтований гнучкістю MongoDB, що дозволяє легко зберігати об'єкти з різним набором характеристик, що є особливо актуальним для каталогу смартфонів.

У процесі проєктування було визначено, що база даних буде містити три основні колекції для зберігання даних про товари, користувачів та замовлення (Таблиця 2.1).

Таблиця 2.1 – Основні колекції бази даних

Ім’я колекції	**Призначення**
Products	Містить дані про товари (смартфони), доступні для продажу.
Users	Містить дані про зареєстрованих користувачів.
Orders	Містить дані про замовлення, зроблені користувачами.

2.2.1 Колекція Products

Ця колекція відповідає за зберігання повної інформації про кожен товар в інтернет-магазині. Структура документа для одного товару наведена у таблиці 2.2.

Таблиця 2.2 – Структура колекції Products

Ім’я поля	**Тип поля**	**Призначення**
_id	ObjectId	Унікальний ідентифікатор, що генерується MongoDB.
id	Number	Унікальний числовий ідентифікатор для зручності роботи на фронтенді.
title	String	Повна назва моделі смартфона.
img	String	Шлях до файлу зображення товару.
price	Number	Ціна товару.
company	String	Назва компанії-виробника.
info	String	Детальний технічний та маркетинговий опис товару.

2.2.2 Колекція Users

Колекція users призначена для зберігання облікових даних користувачів, що реєструються на сайті. З міркувань безпеки, пароль користувача зберігається не у відкритому вигляді, а у вигляді хешу, згенерованого за допомогою криптографічного алгоритму bcrypt. Для розмежування прав доступу використовується поле role. Структура документа для одного користувача наведена у таблиці 2.3.

Таблиця 2.3 – Структура колекції Users

Ім’я поля	**Тип поля**	**Призначення**
_id	ObjectId	Унікальний ідентифікатор, що генерується MongoDB.
login	String	Логін користувача для входу, має бути унікальним.
password	String	Хешований пароль користувача для безпечного зберігання.
email	String	Електронна пошта користувача, має бути унікальною.
role	String	Роль користувача в системі ('user' або 'admin'). За замовчуванням 'user'.

2.2.3 Колекція Orders

Дана колекція зберігає інформацію про всі замовлення, зроблені в інтернет-магазині. Кожен документ замовлення пов'язаний з конкретним користувачем через посилання на його _id з колекції users. Структура документа для одного замовлення наведена у таблиці 2.4.

Таблиця 2.4 – Структура колекції Orders

Ім’я поля	**Тип поля**	**Призначення**
_id	ObjectId	Унікальний ідентифікатор замовлення.
user	ObjectId	Посилання на _id користувача з колекції users.
products	Array	Масив об'єктів, що описують товари в замовленні та їх кількість.
totalAmount	Number	Загальна сума замовлення.
createdAt	Date	Дата та час створення замовлення.

Взаємодія між спроєктованими сутностями та їхня ієрархія візуалізована на UML-діаграмі класів у Додатку Б). Вона наочно демонструє, що одне замовлення (Order) завжди пов'язане з одним користувачем (User) і може містити масив продуктів (Products).

Сам процес розробки структури бази даних відповідав наступним ключовим етапам, що забезпечило логічність та цілісність моделі даних:

- Визначення призначення бази даних: чітке формулювання мети – ефективне зберігання даних для функціонування інтернет-магазину "PixelPhone".
- Пошук і впорядкування необхідної інформації: збір усіх типів даних, які необхідно зберігати (інформація про товари, дані користувачів, деталі замовлень).
- Розподілення інформації по колекціях: групування інформації за основними сутностями: products, users, orders.

- Перетворення елементів інформації на поля: визначення конкретних атрибутів (полів) та їх типів для кожної колекції.
- Визначення первинних ключів: використання унікальних ідентифікаторів _id, що автоматично генеруються MongoDB, для забезпечення унікальності кожного запису.
- Створення зв’язків між колекціями: визначення логічних зв'язків через посилання на ідентифікатори, наприклад, через поле user в колекції orders.
- Удосконалення структури: аналіз створеної структури на предмет надлишковості даних та її оптимізація [14].

2.3 Проєктування клієнтської частини на React

Клієнтська частина інтернет-магазину "PixelPhone" розроблена як сучасний односторінковий застосунок (Single Page Application, SPA) з використанням бібліотеки React. Такий підхід дозволяє створити швидкий та інтерактивний користувацький інтерфейс, де оновлення вмісту відбувається динамічно, без необхідності повного перезавантаження сторінки.

Основою архітектури React є компонентний підхід. Весь інтерфейс застосунку розбивається на незалежні, інкапсульовані та повторно використовувані компоненти, кожен з яких відповідає за свою частину логіки та відображення. Це значно спрощує розробку, тестування та подальшу підтримку проєкту. Використання віртуального DOM (Virtual DOM) в React мінімізує прямі маніпуляції з реальним DOM-деревом, що забезпечує високу продуктивність інтерфейсу.

2.3.1 Компонентна структура застосунку

Проєктом передбачено створення набору ключових компонентів, що формують основний функціонал та вигляд інтернет-магазину. Їх перелік та призначення наведено у таблиці 2.5.

Таблиця 2.5 – Основні компоненти клієнтської частини

Назва компонента	**Призначення**
App	Кореневий компонент застосунку, відповідає за налаштування маршрутизації за допомогою React Router DOM.
ProductProvider	Компонент-постачальник (Provider) глобального стану, реалізований за допомогою React Context API.
Navbar	Компонент навігаційної панелі. Динамічно відображає статус користувача та посилання "Адмін" залежно від ролі.
ProductList	Відповідає за відображення списку товарів (каталогу) на головній сторінці.
Cart	Компонент, що відображає вміст кошика, загальну вартість та надає інструменти для управління ним.
Modal	Модальне вікно для швидкого перегляду інформації про товар та додавання його до кошика.
PrivateRoute	Компонент-охоронець маршрутів, що дозволяє доступ лише авторизованим користувачам (незалежно від ролі).
AdminRoute	Спеціалізований компонент-охоронець, що надає доступ до маршруту лише користувачам з роллю 'admin'.
AdminPanel	Сторінка адміністративної панелі, що є контейнером для інструментів керування магазином [18].

2.3.2 Управління глобальним станом за допомогою Context API

Для управління глобальним станом, який є доступним для багатьох компонентів (наприклад, вміст кошика або статус авторизації користувача), було обрано вбудований в React інструмент — Context API [15]. Цей вибір дозволив уникнути підключення сторонніх бібліотек, таких як Redux, що є надлишковим для проєкту даного масштабу, та зберегти архітектуру легкою та зрозумілою [16].

Було спроєктовано централізоване сховище стану в компоненті ProductProvider. Він відповідає за зберігання та управління такими даними:

- Список усіх товарів, завантажених з сервера.
- Масив товарів, що знаходяться у кошику (cart).
- Стан автентифікації користувача (isAuthenticated, token, об'єкт user з розкодованого JWT).

ProductProvider також надає дочірнім компонентам набір методів для взаємодії з цим станом (наприклад, addToCart, loginUser, logoutUser). Будь-який компонент може отримати доступ до даних та методів, обгорнувшись у компонент-споживач (ProductConsumer), що забезпечує централізований та передбачуваний потік даних у додатку.

Детальна візуалізація взаємозв'язків між ключовими компонентами клієнтської частини представлена на UML-діаграмі класів у Додатку Б.

2.4 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було виконано комплексне проєктування інтернет-магазину "PixelPhone", що заклало архітектурний та технологічний фундамент для подальшої розробки.

Було спроєктовано сучасну клієнт-серверну архітектуру на основі стеку MERN, яка чітко розділяє відповідальність між серверною частиною (Node.js/Express) та клієнтською (React). Було детально розроблено REST API, що визначає правила взаємодії між цими частинами, та, що особливо важливо, спроєктовано систему розмежування доступу до API-ресурсів для різних ролей користувачів.

Проведено детальне проєктування бази даних для середовища MongoDB. Визначено три ключові колекції (products, users, orders), а їхні схеми були розроблені для забезпечення гнучкості та цілісності даних. Зокрема, у схемі користувача було закладено поле role, що є основою для реалізації системи контролю доступу на основі ролей (RBAC).

Для клієнтської частини було спроєктовано компонентну архітектуру з використанням бібліотеки React. Визначено перелік основних компонентів та їх призначення, включаючи спеціалізовані компоненти-охоронці PrivateRoute та AdminRoute. Також було обґрунтовано вибір React Context API як оптимального інструменту для централізованого управління глобальним станом застосунку, включаючи дані про товари, кошик та статус автентифікації користувача.

Таким чином, у цьому розділі створено повний набір проєктних рішень, схем та діаграм, що визначають архітектуру, структуру бази даних та компонентну модель застосунку. Цей детальний проєкт є вичерпною основою для переходу до наступного етапу — безпосередньої програмної реалізації та подальшого тестування розробленого інтернет-магазину.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ "PIXELPHONE"

Цей розділ присвячений опису практичного етапу виконання кваліфікаційної роботи — безпосередній програмній реалізації інтернет-магазину "PixelPhone" відповідно до проєктних рішень, розроблених у попередньому розділі. У даному розділі буде детально розглянуто процес створення серверної частини на Node.js та Express.js, розробки клієнтського застосунку на React, а також імплементації ключових механізмів безпеки та контролю доступу. На завершення розділу буде представлено методику та результати функціонального тестування розробленого програмного продукту.

3.1 Налаштування середовища розробки та інструментарій

Для забезпечення ефективного процесу розробки, тестування та подальшої підтримки проєкту "PixelPhone" було налаштовано сучасне середовище розробки та обрано набір відповідних інструментів. Вибір технологій та програмного забезпечення ґрунтувався на їхній відповідності до стеку MERN, гнучкості та наявності широкої підтримки у спільноті розробників.

Основою серверної частини слугує програмна платформа Node.js — асинхронне середовище виконання JavaScript, що ідеально підходить для створення швидких та масштабованих мережевих застосунків. Для управління проєктними залежностями та підключення зовнішніх бібліотек, таких як серверний фреймворк Express.js, використовувався стандартний для екосистеми Node.js пакетний менеджер npm. Зберігання даних було організовано за допомогою документо-орієнтованої бази даних MongoDB, розгорнутої на хмарній платформі MongoDB Atlas, що забезпечує високу доступність та легкість масштабування.

Клієнтська частина, у свою чергу, була реалізована як односторінковий застосунок (SPA) на базі популярної бібліотеки React. Увесь процес написання коду відбувався в інтегрованому середовищі розробки Visual Studio Code, яке надає зручні інструменти для зневадження та роботи з проєктом. Для відстеження змін у вихідному коді та організації процесу розробки використовувалася система контролю версій Git. На етапі розробки та зневадження REST API застосовувався інструмент Postman, що дозволяв надсилати тестові HTTP-запити до кінцевих точок сервера та перевіряти їх коректність ще до інтеграції з клієнтською частиною.

3.2 Розробка серверної частини (Backend) на Node.js та Express.js

Серверна частина є ядром бізнес-логіки інтернет-магазину "PixelPhone". Вона відповідає за обробку запитів від клієнта, взаємодію з базою даних, автентифікацію та авторизацію користувачів. Розробка велася на платформі Node.js з використанням веб-фреймворку Express.js.

3.2.1 Ініціалізація сервера та проміжного програмного забезпечення

Вхідною точкою серверного додатку є файл server.js. У ньому виконуються ключові кроки для запуску та конфігурації сервера. Спочатку ініціалізується екземпляр додатку Express, який слугує основою для побудови API. Далі, за допомогою бібліотеки Mongoose, встановлюється з'єднання з базою даних, розгорнутою на хмарній платформі MongoDB Atlas, рядок підключення до якої безпечно зберігається у файлі змінних середовища .env.

Після успішного підключення до бази даних налаштовується набір проміжного програмного забезпечення (middleware), яке обробляє кожен вхідний запит. Для проєкту було підключено:

- cors – для дозволу крос-доменних запитів від клієнтського додатку React, що працює на іншому порту [20].

- express.json() – для коректного парсингу тіла запитів, що надходять у форматі JSON.
- passport.initialize() – для ініціалізації системи автентифікації Passport.js, яка є центральним елементом безпеки додатку.

Наостанок, у server.js реєструються основні файли маршрутизації для кожної сутності (/api/products, /api/users, /api/orders), після чого сервер запускається і починає прослуховувати вхідні з'єднання на вказаному порту.

3.2.2 Реалізація моделей даних за допомогою Mongoose

Для структурування даних та взаємодії з колекціями MongoDB було використано бібліотеку Mongoose, яка дозволяє працювати з даними в об'єктно-орієнтованому стилі [13]. На основі схем, спроєктованих у розділі 2.2, у директорії models були створені відповідні Mongoose-моделі: Product, User та Order.

Ці моделі не тільки визначають структуру документів, але й забезпечують валідацію даних перед їх збереженням у базу. Наприклад, модель User гарантує, що поля login та email будуть унікальними для кожного користувача, а поле role за замовчуванням отримає значення 'user'. Приклад реалізації схеми для моделі User наведено у лістингу 3.1.

Лістинг 3.1 – Схема моделі User у файлі models/user.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const userSchema = new Schema(
  {
    login: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    role: { type: String, default: 'user' }
  },  {timestamps: true
  }
);
module.exports = mongoose.model('User', userSchema);
```

3.2.3 Реалізація маршрутів та бізнес-логіки API

Вся бізнес-логіка, пов'язана з обробкою API-запитів, була інкапсульована у відповідних файлах у директорії routes. Такий підхід дозволяє підтримувати чистоту та структурованість коду. Наприклад, файл routes/users.js обробляє всю логіку, пов'язану з реєстрацією та автентифікацією користувачів.

Ключовим елементом системи є маршрут для авторизації POST /api/users/login. Його реалізація демонструє основні принципи роботи серверної частини: асинхронну взаємодію з базою даних, перевірку даних та генерацію відповіді. Логіка цього маршруту включає пошук користувача за логіном, безпечне порівняння наданого пароля з хешем у базі даних за допомогою bcrypt.compare, і, у випадку успіху, створення та підписання JWT-токена, який відправляється клієнту для подальших автентифікованих запитів. Фрагмент реалізації маршруту авторизації наведений у Додатку В.

Аналогічним чином були реалізовані й інші маршрути, що забезпечують повний функціонал для керування товарами та замовленнями, з урахуванням прав доступу, які перевіряються відповідними middleware.

3.2.4 Реалізація маршруту реєстрації та створення адміністратора

Ключовим елементом системи, що відповідає за створення нових облікових записів, є маршрут POST /api/users/register. Його логіка побудована з урахуванням вимог безпеки та гнучкості системи ролей. Першим кроком обробник запиту перевіряє, чи не існує в базі даних користувача з таким самим email, щоб уникнути дублікатів та забезпечити унікальність користувачів.

Далі, для забезпечення безпеки, пароль користувача не зберігається у відкритому вигляді. За допомогою бібліотеки bcrypt.js він проходить процедуру одностороннього хешування з використанням випадково згенерованої "солі", що унеможливлює його зворотне розшифрування [21]. Особливістю даного маршруту є реалізація механізму створення адміністратора. Якщо в тілі запиту

разом з основними даними надходить спеціальний adminCode, який збігається зі значенням, збереженим у змінних середовища на сервері, новому користувачеві присвоюється роль 'admin'. В іншому випадку роль за замовчуванням буде 'user'. Після цих перевірок новий об'єкт користувача з хешованим паролем та визначеною роллю зберігається в базі даних MongoDB.

Також ключовим елементом системи є маршрут для авторизації POST /api/users/login. Його реалізація демонструє основні принципи роботи серверної частини: асинхронну взаємодію з базою даних, перевірку даних та генерацію відповіді. Логіка цього маршруту (див. лістинг 3.2) включає пошук користувача за логіном, безпечне порівняння наданого пароля з хешем у базі даних за допомогою бібліотеки bcrypt.js, і, у випадку успіху, створення та підписання JWT-токена, який відправляється клієнту для подальших автентифікованих запитів

Лістинг 3.2 – Реалізація маршруту реєстрації користувача

```
// фрагмент з routes/users.js
router.post('/register', async (req, res) => {
    try {
        //    Отримуємо    дані,    включаючи    потенційний    код
адміністратора
        const { login, email, password, adminCode } = req.body;
        // Перевірка на існування користувача
        let user = await User.findOne({ email });
        if (user) {
            return res.status(400).json({ message: 'Користувач з
таким email вже існує' });
        }
        // Визначення ролі
        let role = 'user';
        if        (adminCode        &&        adminCode        ===
process.env.ADMIN_SECRET_CODE) {
            role = 'admin';
        }
        // Створення нового користувача з визначеною роллю
        user = new User({ login, email, password, role });
        const salt = await bcrypt.genSalt(10);
        user.password = await bcrypt.hash(password, salt);
        await user.save();
        res.status(201).json({    message:    `Користувач    успішно
зареєстрований з роллю: ${role}` });
    } catch (err) {
```

```
        console.error(err.message);
        res.status(500).send('Помилка сервера');
    }}
);
```

3.3 Розробка клієнтської частини (Frontend) на React

Клієнтська частина інтернет-магазину "PixelPhone" була реалізована як односторінковий застосунок (SPA) з використанням бібліотеки React. React дозволяє створювати великі веб-застосунки, що використовують дані, які змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим та масштабованим. Вся логіка відображення та взаємодії з користувачем інкапсульована на клієнті, який обмінюється даними з сервером через REST API.

3.3.1 Створення компонентної структури та маршрутизації

Клієнтська частина інтернет-магазину "PixelPhone" була реалізована як сучасний односторінковий застосунок (SPA) з використанням бібліотеки React. Розробка почалася з налаштування проєкту за допомогою інструменту create-react-app, який забезпечив готове, попередньо налаштоване середовище для розробки, включаючи інструменти для збірки (Webpack) та трансляції коду (Babel), що дозволило зосередитись безпосередньо на написанні логіки застосунку. Основою архітектури React є компонентний підхід, згідно з яким весь інтерфейс було розбито на ієрархію логічних та повторно використовуваних компонентів, що знаходяться в директорії src/components. Такий підхід не лише спрощує розробку, а й забезпечує візуальну консистентність елементів інтерфейсу та полегшує подальшу підтримку.

Ключові сторінки застосунку були реалізовані як набір взаємопов'язаних React-компонентів. Компонент ProductList.js відповідає за відображення головної сторінки з каталогом товарів. При його завантаженні викликається метод з глобального контексту для асинхронного отримання списку товарів з

API сервера. Отримані дані зберігаються в стані, після чого компонент ітерує по цьому масиву і динамічно рендерить набір дочірніх компонентів Product.js, передаючи в них дані кожного товару через пропси. Загальний вигляд головної сторінки з каталогом товарів представлено на рисунку 3.1. У свою чергу, компонент Cart.js є яскравим прикладом реактивного інтерфейсу. Він не виконує запитів до сервера, а "підписується" на зміни в масиві cart з глобального стану. Будь-яка дія користувача, наприклад, додавання товару, призводить до оновлення стану в ProductProvider, що автоматично викликає перерендер компонента Cart.js з актуальними даними. Сторінка кошика, що відображає обрані товари та загальну суму, показана на рисунку 3.2.

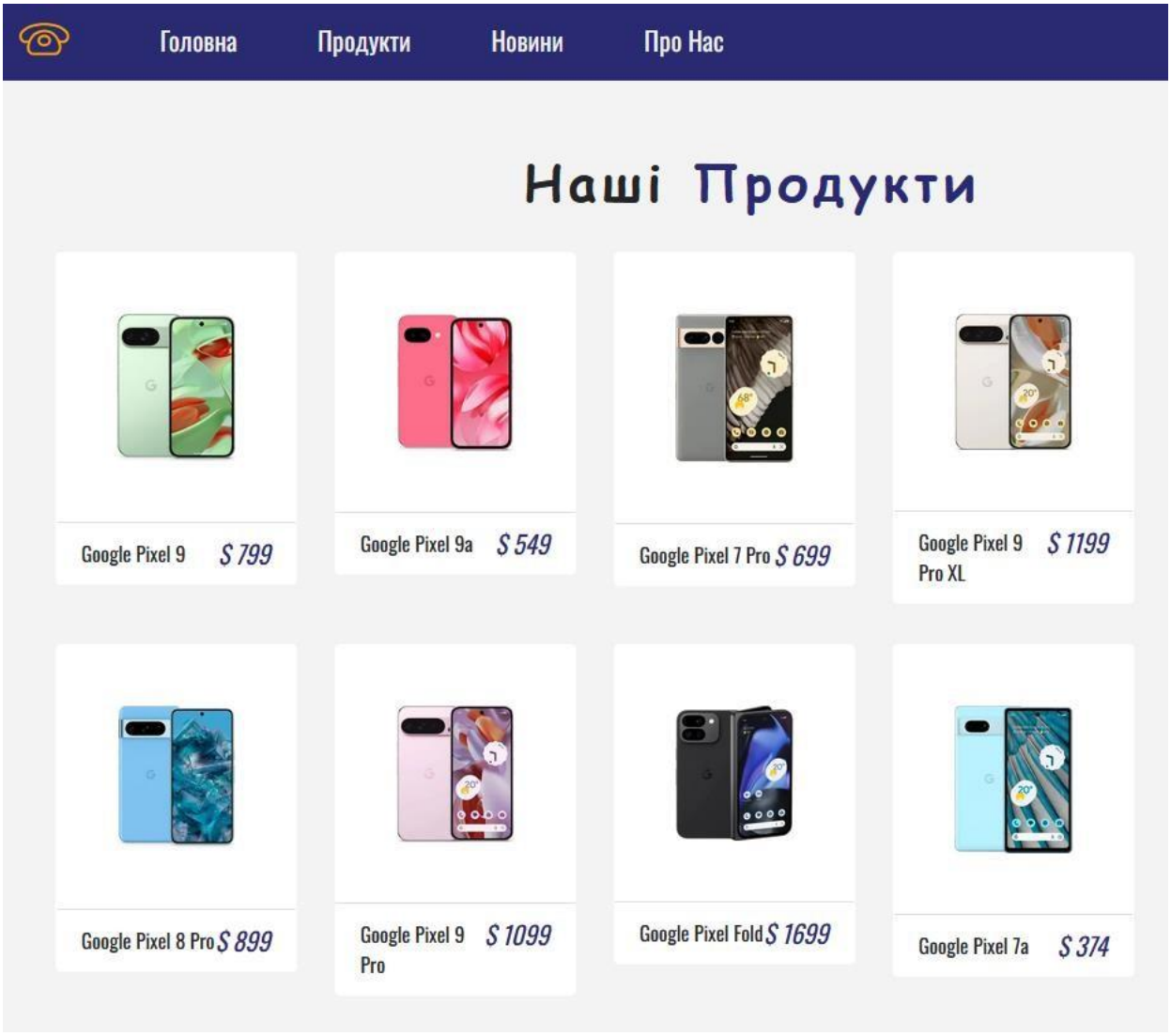


Рисунок 3.1 – Головна сторінка інтернет-магазину "PixelPhone" з каталогом товарів

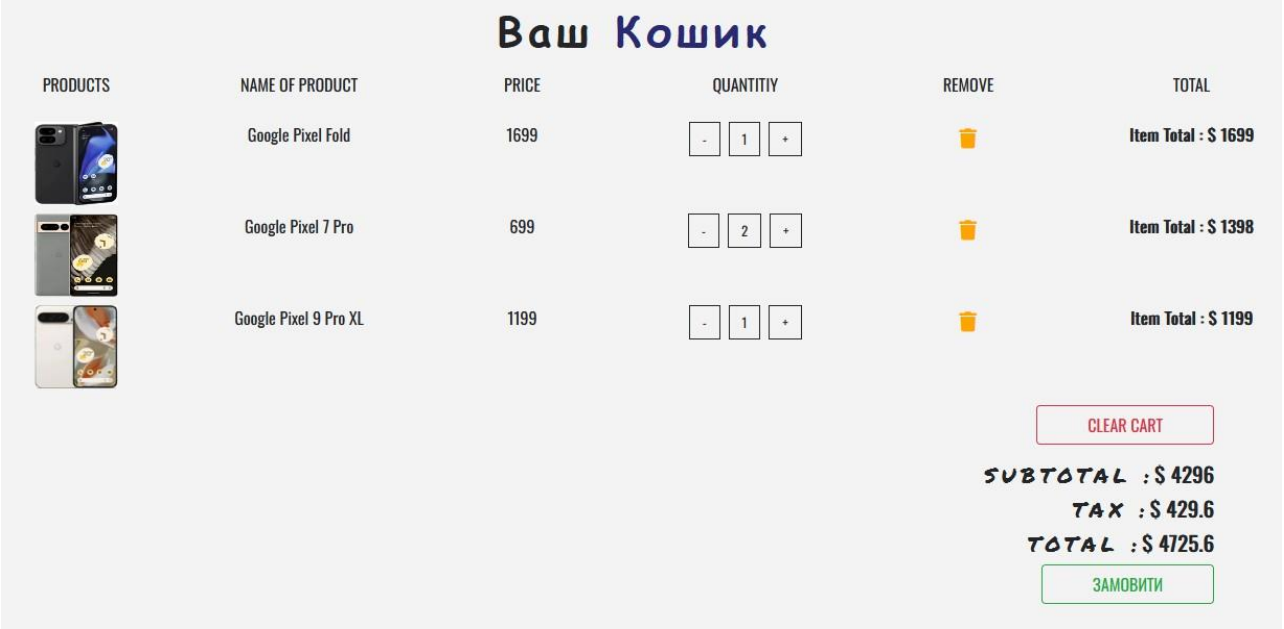


Рисунок 3.2 – Інтерфейс кошика з обраними товарами

Навігація в межах односторінкового застосунку та переходи між цими компонентами (сторінками) були реалізовані за допомогою бібліотеки react-router-dom [17]. Вона дозволяє пов'язувати певні URL-адреси з відповідними React-компонентами, створюючи ілюзію переходу між різними сторінками без фактичного перезавантаження. Основна логіка маршрутизації була налаштована в кореневому компоненті App.js. Як показано в додатку Г, для захисту адміністративної панелі використовується спеціально створений компонент-охоронець <AdminRoute>, який перевіряє не лише факт автентифікації, а й роль користувача.

3.3.2 Реалізація глобального управління станом

Для уникнення складної передачі даних через безліч рівнів компонентів (prop drilling) та для централізації бізнес-логіки на клієнті було використано вбудований в React механізм Context API. Було створено ProductContext, який слугує глобальним сховищем стану для всього застосунку.

Логіка роботи з контекстом інкапсульована в компоненті-провайдері ProductProvider. Ключовим аспектом його реалізації є конструктор, який при

першому завантаженні додатку перевіряє localStorage на наявність збереженого JWT-токена. Якщо токен знайдено і він є валідним, початковий стан додатку одразу встановлюється як "авторизований", що забезпечує стійкість сесії користувача між перезавантаженнями сторінки (див. лістинг 3.3) [25].

Лістинг 3.3 – Ініціалізація стану в конструкторі ProductProvider

```
constructor(props) {
    super(props);
    this.state = {
        products: [],
        cart: [],
        isAuthenticated: false,
        user: null,
        token: null,
        // ...інші поля стану
    };
    const token = localStorage.getItem('jwtToken');
    if (token) {
        try {
            const decodedUser = jwtDecode(token);
            const currentTime = Date.now() / 1000;

            if (decodedUser.exp < currentTime) {
                localStorage.removeItem('jwtToken');
            } else {
                this.state.isAuthenticated = true;
                this.state.user = decodedUser;
                this.state.token = token;
                this.state.loggedInUserName = decodedUser.login;
            }
        } catch (error) {
            localStorage.removeItem('jwtToken');
        }
    }
}
```

Окрім зберігання даних (список товарів, кошик, інформація про користувача), ProductProvider містить усі основні методи для взаємодії з ними, такі як addToCart, removeItem, а також асинхронні функції для комунікації з сервером: loginUser, registerUser, makeOrder. Інші компоненти отримують доступ до цього стану та методів через компонент-споживач ProductConsumer, що робить потік даних в додатку передбачуваним та керованим.

3.4 Реалізація системи безпеки та контролю доступу (RBAC)

Забезпечення безпеки даних та розмежування прав доступу є критично важливими вимогами для будь-якого комерційного веб-додатку [38]. В інтернет-магазині "PixelPhone" було реалізовано багаторівневу систему безпеки, що охоплює безпечне зберігання паролів, надійну автентифікацію користувачів та гнучкий контроль доступу на основі ролей (Role-Based Access Control, RBAC).

3.4.1 Безпечне зберігання паролів

Категорично неприпустимо зберігати паролі користувачів у базі даних у відкритому вигляді. Для захисту облікових даних було використано криптографічний підхід — одностороннє хешування з використанням "солі". Для реалізації цього механізму було обрано бібліотеку bcrypt.js.

При реєстрації нового користувача його пароль не зберігається напряму. Натомість, bcrypt.js генерує випадковий рядок ("сіль"), поєднує його з паролем і застосовує до них складний алгоритм хешування [21]. Результат (хеш) є унікальним для кожної пари пароль-сіль і не може бути розшифрований назад у початковий пароль. Цей хеш і зберігається в полі password документа користувача. Процес хешування показано у лістингу 3.4.

Лістинг 3.4 – Хешування пароля при реєстрації користувача

```
// фрагмент з routes/users.js
// ...
user = new User({ login, email, password, role });
// Хешуємо пароль перед збереженням
const salt = await bcrypt.genSalt(10);
user.password = await bcrypt.hash(password, salt);
await user.save();
```

При спробі входу в систему, наданий користувачем пароль хешується з тією ж "сіллю" і порівнюється з хешем у базі даних за допомогою функції bcrypt.compare.

3.4.2 Автентифікація на основі JWT

Процес автентифікації користувача в інтернет-магазині "PixelPhone" починається з інтерфейсу, що містить форми для входу та реєстрації (див. рисунок 3.3). Після того, як користувач вводить свої облікові дані та відправляє форму, для подальшого управління сесіями та захисту даних було використано стандарт JWT (JSON Web Tokens) [22]. Після успішної перевірки логіна та пароля сервер генерує JWT-токен — закодований об'єкт, що містить основну інформацію про користувача (його ID, логін та роль) і підписаний унікальним секретним ключем, який зберігається у змінних середовища на сервері.

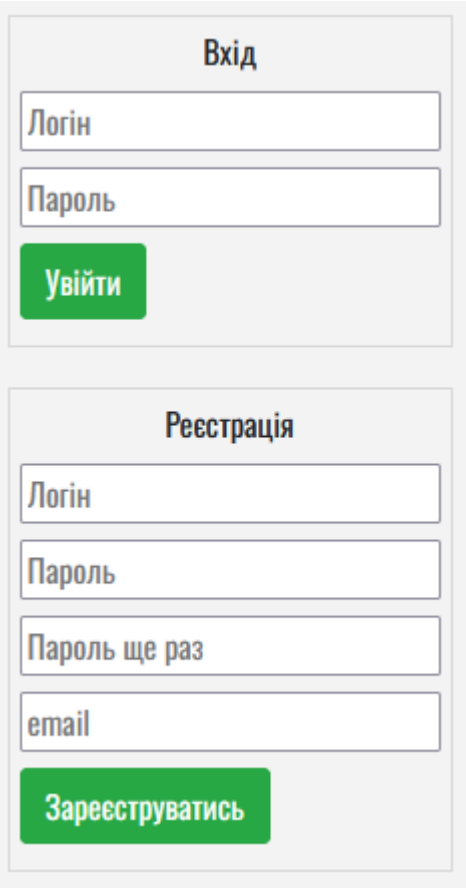


Рисунок 3.3 – Інтерфейс форм автентифікації та реєстрації

Цей токен відправляється на клієнт, де зберігається у localStorage [24]. При кожному наступному запиті до захищених ресурсів API клієнт автоматично додає цей токен у заголовок Authorization за схемою Bearer <токен>.

На сервері для перевірки токенів використовується бібліотека passport.js зі стратегією passport-jwt [23]. Було створено конфігураційний файл, що налаштовує passport на вилучення токена із заголовка, перевірку його підпису та терміну дії. У випадку успішної валідації, passport вилучає дані користувача з токена, знаходить цього користувача в базі даних і додає об'єкт користувача до об'єкта запиту (req.user), роблячи його доступним для подальшої обробки в маршруті. Реалізація стратегії показана у лістингу 3.5.

Лістинг 3.5 – Конфігурація стратегії Passport-JWT

```
// config/passport.js
const JwtStrategy = require('passport-jwt').Strategy;
const ExtractJwt = require('passport-jwt').ExtractJwt;
const User = require('../models/user');
const keys = process.env.JWT_SECRET;
const opts = {};
opts.jwtFromRequest = ExtractJwt.fromAuthHeaderAsBearerToken();
opts.secretOrKey = keys;
module.exports = passport => {
  passport.use(
    new JwtStrategy(opts, (jwt_payload, done) => {
      User.findById(jwt_payload.id)
        .then(user => {
          if (user) {
            return done(null, user);}
          return done(null, false);
        })
        .catch(err => console.log(err));
    })
  );
};
```

3.4.3 Контроль доступу на основі ролей (RBAC)

Для розмежування можливостей звичайних користувачів та адміністраторів було реалізовано систему контролю доступу на основі ролей.

Система підтримує дві ролі: 'user' та 'admin', які зберігаються у відповідному полі документа користувача та включаються до JWT-токена при авторизації.

Захист на рівні Backend.

Для захисту API-маршрутів, які мають бути доступні лише адміністраторам, було створено власне проміжне програмне забезпечення (middleware) checkAdmin (див. лістинг 3.6). Ця функція запускається в ланцюжку обробників маршруту одразу після успішної автентифікації passport.js. Вона перевіряє поле role в об'єкті req.user і, якщо роль не є 'admin', повертає клієнту помилку 403 Forbidden, блокуючи подальше виконання запиту.

Лістинг 3.6 – Middleware для перевірки ролі адміністратора

```
exports.checkAdmin = (req, res, next) => {
    if (req.user && req.user.role === 'admin') {
        next(); // Успіх: користувач є адміном, продовжуємо
    } else {
        res.status(403).json({
            message: 'Доступ заборонено. Потрібні права адміністратора.'
        });
}};
```

Захист на рівні Frontend.

Для захисту клієнтських маршрутів було створено спеціалізований компонент-охоронець AdminRoute.js (див. лістинг 3.7). Він діє як обгортка для маршрутів, що ведуть до адміністративних сторінок. Компонент отримує доступ до глобального стану через ProductConsumer і в своїй логіці рендерингу перевіряє, чи є користувач одночасно авторизованим (isAuthenticated) і чи має він роль 'admin'. Якщо обидві умови не виконуються, користувач автоматично перенаправляється на головну сторінку.

Лістинг 3.7 – Реалізація компонента AdminRoute.js

```
import React from 'react';
import { Route, Redirect } from 'react-router-dom';
import { ProductConsumer } from '../context';
```

```
const AdminRoute = ({ component: Component, ...rest }) => {
    return (
        <ProductConsumer>
            {value => (
                <Route
                    {...rest}
                    render={props =>
                        (value.isAuthenticated && value.user &&
value.user.role === 'admin')
                        ? <Component {...props} />
                        : <Redirect to="/" />
                         }
                />
              )
             }
        </ProductConsumer>
    );
};
export default AdminRoute;
```

Такий комплексний підхід, що поєднує захист на рівні сервера та клієнта, забезпечує надійне розмежування прав та високий рівень безпеки для інтернет-магазину "PixelPhone".

3.5 Тестування розробленого застосунку

Тестування є заключним та одним з найважливіших етапів розробки, що гарантує якість програмного продукту та його відповідність вимогам, визначеним у технічному завданні. Метою тестування інтернет-магазину "PixelPhone" була перевірка коректності роботи всього реалізованого функціоналу, а також надійності системи безпеки та розмежування прав доступу. Особливу увагу було приділено тестуванню наскрізних сценаріїв, що охоплюють взаємодію між клієнтською частиною на React, серверною логікою на Node.js та базою даних, щоб переконатися у цілісності передачі даних на всіх етапах. Процес перевірки поєднував ручне тестування користувацьких інтерфейсів на предмет зручності та коректного відображення, а також тестування API для перевірки правильності відповідей сервера. Такий комплексний підхід дозволив не лише виявити та усунути потенційні дефекти,

а й підтвердити, що розроблений продукт готовий до реальної експлуатації та здатен забезпечити позитивний користувацький досвід..

3.5.1 Методика тестування

Враховуючи специфіку розробки веб-додатку, основним методом тестування було обрано функціональне тестування за принципом "чорного ящика". Цей підхід полягає у перевірці поведінки системи через її користувацький інтерфейс та API, базуючись на очікуваних результатах, без аналізу внутрішньої структури програмного коду [26].

Для комплексного покриття всіх можливих випадків використання, програмний продукт було логічно розділено на функціональні частини відповідно до ролей користувачів. На основі цього було розроблено набір тестових сценаріїв для трьох ключових ролей: неавторизований користувач ("Гість"), звичайний авторизований користувач (user) та адміністратор (admin).

3.5.2 Тестові сценарії та їх виконання

Було проведено тестування за наступними ключовими сценаріями, що охоплюють основний функціонал та систему безпеки додатку.

Сценарій 1: неавторизований користувач ("Гість").

Мета: Перевірити, що гість має доступ лише до публічної інформації та не може виконувати дії, що потребують авторизації.

Тестові випадки:

- Перегляд товарів: Користувач може зайти на сайт, переглядати каталог товарів та сторінки з детальним описом. Результат: Успішно.
- Доступ до адмін-панелі (UI): У навігаційній панелі відсутнє посилання "Адмін". Результат: Успішно.

- Доступ до адмін-панелі (URL): При спробі прямого переходу за адресою /AdminPanel користувача перенаправляє на головну сторінку. Результат: Успішно.
- Створення замовлення: Користувач може додати товари в кошик, але при спробі оформити замовлення (викликати API POST /api/orders) сервер повертає помилку 401 Unauthorized. Результат: Успішно.

Сценарій 2: авторизований користувач (роль 'user').

Мета: Перевірити функціонал, доступний звичайному зареєстрованому користувачеві, та переконатись у відсутності доступу до адміністративних функцій.

Тестові випадки:

- Авторизація та вихід: Користувач може успішно увійти в систему, після чого бачить персоналізоване привітання. Кнопка "Вийти" коректно завершує сесію. Результат: Успішно.
- Доступ до адмін-панелі (UI): У навігаційній панелі посилання "Адмін" відсутнє. Результат: Успішно.
- Доступ до адмін-панелі (URL): При спробі прямого переходу за адресою /AdminPanel користувача перенаправляє на головну сторінку, при цьому сесія користувача не переривається. Результат: Успішно.
- Створення замовлення: Користувач може успішно оформити замовлення. Запит на POST /api/orders проходить успішно. Результат: Успішно.

Сценарій 3: Авторизований користувач (роль 'admin')

Мета: Перевірити, що адміністратор має повний доступ до всіх захищених ресурсів, включаючи адміністративні.

Тестові випадки:

- Авторизація: Користувач з роллю admin успішно входить в систему. Результат: Успішно.
- Доступ до адмін-панелі (UI): У навігаційній панелі присутнє посилання "Адмін". Результат: Успішно.

- Доступ до адмін-панелі (URL): При прямому переході за адресою /AdminPanel користувач успішно потрапляє на сторінку адміністративної панелі. Результат: Успішно.
- Доступ до захищеного API: Запит на адміністративний ендпоінт (наприклад, GET /api/orders/all) з JWT-токеном адміністратора проходить успішно, повертаючи дані. Результат: Успішно.

3.5.3 Результати тестування

Проведення тестування за вищевказаними сценаріями показало, що розроблений інтернет-магазин "PixelPhone" функціонує коректно та у повній відповідності до вимог, сформульованих у технічному завданні. Усі основні функції, такі як перегляд товарів, робота кошика, реєстрація та авторизація, працюють без збоїв.

Особливо важливо, що тестування підтвердило надійність реалізованої системи безпеки та контролю доступу. Система розмежування прав на основі ролей коректно блокує несанкціонований доступ до адміністративних ресурсів як на клієнтській, так і на серверній стороні. Таким чином, можна зробити висновок, що програмний продукт є якісним, стабільним та готовим до подальшого розгортання та експлуатації.

3.6 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи було детально описано процес програмної реалізації інтернет-магазину "PixelPhone", що є практичним втіленням проєктних рішень, розроблених у попередньому розділі.

Було продемонстровано створення серверної частини на основі Node.js та Express.js, включаючи налаштування серверного середовища, реалізацію моделей даних за допомогою Mongoose для взаємодії з базою даних MongoDB, та розробку логіки REST API маршрутів.

Також було детально описано розробку клієнтської частини як односторінкового застосунку на бібліотеці React. Було висвітлено процес створення компонентної архітектури, налаштування маршрутизації та реалізацію централізованої системи управління станом за допомогою React Context API, що забезпечує стійкість сесії користувача при перезавантаженні сторінки.

Особливу увагу було приділено імплементації багаторівневої системи безпеки. Було описано механізми захисту паролів за допомогою хешування, автентифікації на основі JWT-токенів.

На завершення розділу було представлено методику та результати функціонального тестування. Проведені за ключовими сценаріями тести підтвердили, що розроблений додаток функціонує коректно, а всі механізми безпеки працюють у повній відповідності до поставлених вимог.

Таким чином, у третьому розділі було продемонстровано повний цикл перетворення проєктних вимог у готовий до експлуатації, функціональний та безпечний програмний продукт, якість якого підтверджена результатами тестування.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Аналіз умов праці розробника в рамках проєкту "PixelPhone"

Розробка кваліфікаційної роботи, зокрема інтернет-магазину "PixelPhone", є прикладом сучасної інтелектуальної праці, яка відбувається в складній системі "людина-машина-середовище". На мою думку, попри відсутність очевидних фізичних небезпек, притаманних промисловому виробництву, такий трудовий процес характеризується унікальним комплексом шкідливих та небезпечних чинників. Тому комплексний аналіз умов праці є невід'ємною частиною забезпечення безпеки та високої продуктивності протягом усього циклу розробки [27]. Аналізуючи власний досвід під час створення проєкту, можна виділити наступні ключові фактори, що класифікуються за показниками виробничого середовища, важкості та напруженості трудового процесу [33].

Робоче місце розробника оснащене персональним комп'ютером, який є джерелом переважно фізичних виробничих факторів. У контексті даної роботи, що виконувалась у типовому приміщенні, ці фактори вимагали особливої уваги. До них, насамперед, належать параметри мікроклімату, адже тривала робота комп'ютерної техніки призводить до підвищення температури повітря, що без належної вентиляції може викликати дискомфорт та зниження когнітивних функцій, як це регламентується ДСН 3.3.6.042–99. Не менш важливим є освітлення, оскільки розробка інтерфейсу "PixelPhone" вимагала роботи з дрібними графічними елементами та кодом, що підвищує чутливість до відблисків від екрана монітора та вимагає дотримання норм ДБН В.2.5-28:2018. Окрім того, на концентрацію може впливати підвищений рівень шуму; хоча він невисокий, аналіз власного досвіду показує, що навіть незначні фонові звуки можуть суттєво знижувати продуктивність при вирішенні складних алгоритмічних задач [32].

Важкість праці розробника програмного забезпечення визначається переважно статичними навантаженнями. Процес програмування та налагодження "PixelPhone" вимагав тривалих періодів безперервної роботи в сидячому положенні. Це, з власного досвіду, призводить до дискомфорту та напруження у м'язах спини, шиї та плечового поясу, а також створює ризики для опорно-рухового апарату при систематичному ігноруванні правил ергономіки. Це доповнюється монотонністю рухів, адже хоча розробка є творчим процесом, вона включає значну кількість одноманітних дій, таких як набір коду та маніпуляції з мишею, що створюють локальне навантаження на м'язи кистей та передпліч і є фактором ризику розвитку тунельного синдрому.

Проте, на мою думку, саме категорія напруженості трудового процесу є домінуючою та найбільш вираженою при розробці складного програмного продукту, яким є "PixelPhone". Найбільш очевидним фактором є значне зорове навантаження. Під час реалізації клієнтської частини та налагодження стилів доводилося працювати з великою кількістю дрібного тексту та графічних елементів, що вимагало постійної акомодації та напруження зорового аналізатора. Це неминуче призводило до симптомів "комп'ютерного зорового синдрому", що підкреслює важливість дотримання санітарних норм роботи з ВДТ, згідно з ДСанПіН 3.3.2.007-98 [30]. До цього додається високе інтелектуальне та емоційне навантаження. Реалізація таких складних модулів, як система автентифікації та контроль доступу на основі ролей (RBAC), вимагала максимальної концентрації та аналітичних здібностей. Процес зневадження (дебагінгу) помилок, особливо тих, що виникали на стику клієнтської та серверної частин, створював значне емоційне напруження, пов'язане з відповідальністю за кінцевий результат та дотриманням термінів. Парадоксально, але цей творчий процес включає і періоди монотонності, наприклад, при наповненні бази даних початковими даними про товари або при написанні однотипних тестів, що, як показує практика, знижує пильність і може стати причиною механічних помилок у коді.

Отже, робота розробника веб-додатку "PixelPhone" пов'язана з комплексним впливом психофізіологічних та фізичних факторів, що вимагає свідомого підходу до організації робочого місця та режиму праці для збереження здоров'я і забезпечення високої продуктивності.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

Проведений у попередньому підрозділі аналіз умов праці під час розробки інтернет-магазину "PixelPhone" чітко показав, що ключовими ризиками для розробника є статичні навантаження на опорно-руховий апарат та значне напруження зорового аналізатора. На мою думку, ключовим інструментом для мінімізації цих ризиків та створення безпечних і продуктивних умов праці є застосування принципів ергономіки. Ергономіка як наука покликана адаптувати робоче місце, інструменти та робочий процес до фізіологічних та психологічних можливостей людини. Вимоги ергономіки до організації робочого місця оператора ПК регламентуються низкою нормативних документів, зокрема ДСанПіН 3.3.2.007-98.

Правильна організація фізичних елементів робочого місця є фундаментом для профілактики захворювань опорно-рухового апарату. Насамперед, увага приділяється робочому кріслу, яке є найважливішим елементом для компенсації статичних навантажень. З власного досвіду можу стверджувати, що вибір правильного крісла є найбільш важливою інвестицією в довгострокову працездатність. Під час багатогодинних сесій кодування для "PixelPhone", відсутність підтримки попереку та можливості змінити кут нахилу спинки швидко призводила до дискомфорту та болю в спині. Тому робоче крісло повинно бути підйомно-поворотним, регульованим по висоті та кутах нахилу сидіння і спинки, а також мати регульовані по висоті підлокітники. Конструкція має забезпечувати підтримку раціональної робочої пози, що знижує статичне напруження м'язів шийно-плечової області та спини.

Не менш важливим є робочий стіл. Його поверхня повинна мати достатні розміри для раціонального розміщення всього обладнання та документів, а простір для ніг має бути достатнім для вільної зміни положення. При розробці проєкту "PixelPhone" стало очевидним, що розмір столу має критичне значення не лише для комфорту, а й для організації робочого процесу, оскільки недостатня глибина столу змушує розташовувати монітор занадто близько до очей.

Правильне розташування монітора є найдієвішим заходом профілактики зорової втоми. Відповідно до санітарних норм, екран має знаходитись на відстані 50-70 см від очей оператора, а його верхній край — на рівні очей або трохи нижче. Під час роботи з комплексними інтерфейсами, як-от адміністративна панель "PixelPhone", було важливо розмістити монітор так, щоб уникнути відблисків від вікна чи стельових ламп, які викликають додаткове напруження. Завершують організацію фізичного простору клавіатура та миша. Для запобігання розвитку травм від повторюваних навантажень, клавіатура повинна розташовуватися на столі на відстані 10-30 см від краю, забезпечуючи опору для передпліч. Обґрунтованим є використання ергономічних моделей, що підтримують більш природне положення руки.

Однак, навіть ідеально облаштоване робоче місце не може повністю компенсувати негативний вплив тривалої безперервної роботи. Тому невід'ємною частиною ергономіки є правильна організація режимів праці та відпочинку. ДСанПіН 3.3.2.007-98 встановлює регламентовані перерви для відпочинку, рекомендуючи встановлювати їх тривалістю 5-10 хвилин після кожних 45-60 хвилин роботи [28]. На практиці, особливо під час вирішення складних задач, як-от налагодження системи автентифікації в проєкті "PixelPhone", існує спокуса працювати безперервно. Проте, я переконався, що такий підхід є контрпродуктивним, оскільки призводить до накопичення втоми та збільшення кількості помилок. Впровадження коротких, але регулярних перерв, як це рекомендується в науковій літературі з психології безпеки,

дозволяє не тільки відпочити очам та м'язам, а й "перезавантажити" когнітивні процеси, що часто допомагає знайти рішення швидше та ефективніше [31].

Таким чином, комплексне застосування вимог ергономіки до організації робочого місця та дотримання раціональних режимів праці і відпочинку дозволяє значно знизити вплив шкідливих та небезпечних факторів, зберегти здоров'я та підтримувати високий рівень працездатності розробника протягом усього робочого дня.

4.3 Розрахунок штучного освітлення для робочого місця розробника

Проведений у попередньому підрозділі аналіз умов праці під час розробки інтернет-магазину "PixelPhone" чітко показав, що ключовими ризиками для розробника є статичні навантаження на опорно-руховий апарат та значне напруження зорового аналізатора. На мою думку, ключовим інструментом для мінімізації цих ризиків та створення безпечних і продуктивних умов праці є застосування принципів ергономіки. Розрахунок системи штучного освітлення є не лише формальною вимогою, а й практичним кроком для створення оптимальних умов праці, оскільки якість освітлення безпосередньо впливає на зорову втому, що було визначено як один з головних ризиків. Метою даного розрахунку є визначення необхідної кількості світильників та їх потужності для забезпечення нормативного рівня освітленості в умовному робочому приміщенні, де велася розробка.

Відповідно до ДБН В.2.5-28:2018 "Природне і штучне освітлення", для приміщень, де виконуються роботи з комп'ютерною технікою, нормований рівень освітленості на робочій поверхні має становити 300-500 лк [29]. Для розрахунку було обрано метод коефіцієнта використання світлового потоку, який є найбільш поширеним для розрахунку загального рівномірного освітлення. Для проведення розрахунку за цим методом було прийнято наступні вихідні дані для умовного офісного приміщення, де працює

розробник: розміри приміщення (довжина A = 6 м, ширина B = 4 м, висота H = 3 м); висота робочої поверхні (стіл) над підлогою $h_р$ = 0.8 м; висота підвісу світильників від стелі $h_{св}$ = 0.2 м. Нормована освітленість ($E_н$) приймається на рівні 400 лк. Коефіцієнти відбиття поверхонь прийнято як для типового офісного приміщення: стеля ($ρ_с$) = 70%, стіни ($ρ_{ст}$) = 50%, підлога ($ρ_п$) = 30%. Для забезпечення енергоефективності було обрано сучасний світлодіодний (LED) світильник панельного типу з однією лампою (n=1) та світловим потоком $Ф_л$ = 3600 лм. Також враховано коефіцієнт запасу $k_з$ = 1.5, що компенсує зниження світлового потоку в процесі експлуатації, та коефіцієнт нерівномірності Z = 1.1.

Розрахунок необхідної кількості світильників (N) виконується за формулою:

$$N = \frac{E_н \cdot S \cdot k_з \cdot Z}{n \cdot Ф_л \cdot \eta}$$

Спочатку визначається площа приміщення: $S = A \cdot B = 6 \cdot 4 = 24$ м². Наступним кроком є визначення розрахункової висоти підвісу світильників над робочою поверхнею: $h = H - h_р - h_{св} = 3 - 0.8 - 0.2 = 2$ м. Далі розраховується індекс приміщення, що характеризує його геометрію: $i = \frac{S}{h \cdot (A+B)} = \frac{24}{2 \cdot (6+4)} = 1.2$. Використовуючи отриманий індекс приміщення та коефіцієнти відбиття, за довідковими таблицями для обраного типу світильника визначається коефіцієнт використання світлового потоку η, який становить приблизно 0.55. Фінальна кількість світильників розраховується за основною формулою:

$$N = \frac{400 \cdot 24 \cdot 1.5 \cdot 1.1}{1 \cdot 3600 \cdot 0.55} = \frac{15840}{1980} \approx 8$$

Отже, проведений розрахунок показує, що для забезпечення нормативних умов освітлення у приміщенні площею 24 м² необхідно встановити 8 світлодіодних світильників обраного типу. На мою думку, їх варто розмістити

рівномірно по стелі у два ряди по чотири світильники, що забезпечить рівномірність освітлення по всій робочій зоні. Цей розрахунок є важливою демонстрацією того, як інженерні методи дозволяють вирішувати конкретні задачі з охорони праці. Правильно організоване освітлення напряму впливає на зниження зорової втоми, що було визначено як один з ключових ризиків у підрозділі 4.1, та, як наслідок, на підвищення продуктивності та якості роботи над проєктом “PixelPhone”.

4.4 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи було комплексно розглянуто питання безпеки життєдіяльності та охорони праці в контексті діяльності розробника програмного забезпечення на прикладі проєкту "PixelPhone". Було проведено детальний аналіз умов праці, в ході якого ідентифіковано ключові шкідливі та небезпечні чинники, зокрема психофізіологічні (зорове та інтелектуальне) та фізичні (статичне навантаження та освітлення).

На основі проведеного аналізу було сформульовано та обґрунтовано набір вимог до ергономічної організації робочого місця оператора ПК. Були розглянуті рекомендації щодо вибору та налаштування меблів, розміщення обладнання та важливості дотримання режимів праці та відпочинку для мінімізації виявлених ризиків.

Для практичного підтвердження теоретичних положень було виконано інженерний розрахунок системи штучного освітлення для умовного робочого приміщення. Розрахунок дозволив визначити необхідну кількість та тип світильників для забезпечення нормативного рівня освітленості, що є ключовим заходом для профілактики зорової втоми. Таким чином, у даному розділі було поєднано теоретичний аналіз ризиків з практичними рекомендаціями та інженерними розрахунками для створення безпечних та комфортних умов праці.

ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи було успішно досягнуто поставленої мети — розроблено повнофункціональний інтернет-магазин "PixelPhone" на стеку MERN. Було вирішено всі поставлені задачі, що дозволило створити сучасний, безпечний та готовий до розгортання програмний продукт.

В першому розділі кваліфікаційної роботи було подано комплексний аналіз сучасних підходів до створення платформ для електронної комерції, розглянуто та обґрунтовано вибір технологічного стеку MERN, а також висвітлено його ключові переваги. На завершення розділу було сформовано детальне технічне завдання, що визначає функціональні та нефункціональні вимоги до інтернет-магазину, з особливим акцентом на реалізацію системи безпеки та розмежування доступу користувачів.

У другому розділі було виконано проєктування застосунку. Було досліджено архітектурні підходи та обґрунтовано вибір клієнт-серверної моделі з REST API. Було сформовано гнучку структуру бази даних MongoDB з колекціями products, users та orders, а також спроєктовано компонентну архітектуру клієнтської частини на React, включаючи спеціалізовані компоненти-охоронці та модель управління станом через Context API.

В третьому розділі було розроблено повноцінний програмний продукт, що складається з серверної частини на Node.js та клієнтської на React. Було реалізовано багаторівневу систему безпеки, що включає хешування паролів, JWT-автентифікацію та контроль доступу на основі ролей. Також було запропоновано та проведено методику функціонального тестування за ключовими сценаріями. Успішне тестування підтвердило коректність роботи всього функціоналу та надійність реалізованих механізмів безпеки.

У розділі «Безпека життєдіяльності, основи охорони праці» було висвітлено основні шкідливі та небезпечні чинники, що впливають на розробника програмного забезпечення, розглянуто ергономічні вимоги до

організації робочого місця та проведено розрахунок штучного освітлення для забезпечення нормативних умов праці.

Отже, розроблений інтернет-магазин "PixelPhone" є завершеним програмним продуктом, що повністю відповідає поставленим вимогам, має гнучку архітектуру для подальшого розширення та демонструє застосування сучасних підходів до розробки безпечних веб-додатків.

ПЕРЕЛІК ДЖЕРЕЛ

1. Kinstler K. Shopify Vs. WooCommerce: Which Is Better For Your Business? Forbes Advisor. 2024. URL: https://www.forbes.com/advisor/business/software/woocommerce-vs-shopify/ (дата звернення: 14.05.2025).

2. Leimkuhler C. Why Choose Custom eCommerce Website Development? BigCommerce Blog. 2023. URL: https://www.bigcommerce.com/blog/custom-ecommerce-website-development/ (дата звернення: 22.05.2025).

3. What is the MERN Stack? MongoDB. 2025. URL: https://www.mongodb.com/mern-stack (дата звернення: 02.06.2025).

4. React Documentation. The React Team. 2025. URL: https://react.dev/ (дата звернення: 11.05.2025).

5. About Node.js. OpenJS Foundation. 2025. URL: https://nodejs.org/en/about (дата звернення: 25.05.2025).

6. Express – Node.js web application framework. The Express.js team. 2025. URL: https://expressjs.com/ (дата звернення: 04.06.2025).

7. MongoDB Documentation. MongoDB, Inc. 2025. URL: https://www.mongodb.com/docs/ (дата звернення: 18.05.2025).

8. What is MERN Stack? Learn All About MERN Stack. Simplilearn. 2024. URL: https://www.simplilearn.com/tutorials/mongodb-tutorial/what-is-mern-stack (дата звернення: 29.05.2025).

9. What is a REST API? Red Hat. 2023. URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api (дата звернення: 09.05.2025).

10. Responsive web design basics. Google Developers. 2024. URL: https://web.dev/learn/design/responsive-design (дата звернення: 16.05.2025).

11. What is a Client-Server Architecture? IBM Technology. 2023. URL: https://www.ibm.com/topics/client-server (дата звернення: 21.05.2025).

12. Hanes D. What is REST? Cloudflare Learning Center. 2024. URL: https://www.cloudflare.com/learning/ddos/what-is-a-rest-api/ (дата звернення: 03.06.2025).

13. Mongoose ODM v8.4.1. MongooseJS. 2025. URL: https://mongoosejs.com/ (дата звернення: 13.05.2025).

14. Data-Model-Design. MongoDB Documentation. 2024. URL: https://www.mongodb.com/docs/manual/core/data-model-design/ (дата звернення: 27.05.2025).

15. Passing Data Deeply with Context. The React Team. 2025. URL: https://react.dev/learn/passing-data-deeply-with-context (дата звернення: 08.05.2025).

16. Wieruch R. React Context API vs Redux. Robin Wieruch's Blog. 2024. URL: https://www.robinwieruch.de/react-context-redux/ (дата звернення: 30.05.2025).

17. React Router Documentation. The React Router Team. 2024. URL: https://reactrouter.com/en/main (дата звернення: 19.05.2025).

18. Petch S. Secure and Protected Routes with React Router V6. Medium. 2023. URL: https://medium.com/@s.petch.dev/secure-and-protected-routes-with-react-router-v6-4e173e673410 (дата звернення: 24.05.2025).

19. Create React App Documentation. The create-react-app team. 2023. URL: https://create-react-app.dev/ (дата звернення: 05.06.2025).

20. cors. The ExpressJS team on npm. 2022. URL: https://www.npmjs.com/package/cors (дата звернення: 12.05.2025).

21. bcrypt.js. The bcrypt.js team on npm. 2024. URL: https://www.npmjs.com/package/bcrypt.js (дата звернення: 28.05.2025).

22. JSON Web Tokens. Auth0. 2025. URL: https://jwt.io/ (дата звернення: 17.05.2025).

23. passport-jwt. The passport-jwt team on npm. 2023. URL: https://www.npmjs.com/package/passport-jwt (дата звернення: 20.05.2025).

24. Preet S. How To Handle JWT in React. Medium. 2023. URL: https://medium.com/@preetsinghmusic/how-to-handle-jwt-in-react-47a7c642646 (дата звернення: 01.06.2025).

25. jwt-decode. The jwt-decode team on npm. 2024. URL: https://www.npmjs.com/package/jwt-decode (дата звернення: 26.05.2025).

26. What is Black Box Testing? Example & Types. Guru99. 2024. URL: https://www.guru99.com/black-box-testing.html (дата звернення: 10.05.2025).

27. Про охорону праці : Закон України від 14.10.1992 р. № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 23.05.2025).

28. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин : ДСанПін 3.3.2.007-98. URL: https://dnaop.com/html/2297/doc-ДСанПіН_3.3.2.007-98 (дата звернення: 15.05.2025).

29. Природне і штучне освітлення : ДБН В.2.5-28:2018. URL: https://dbn.co.ua/load/normativy/dbn/1-1-0-1284 (дата звернення: 31.05.2025).

30. Computer Vision Syndrome. American Optometric Association. 2024. URL: https://www.aoa.org/healthy-eyes/eye-and-vision-conditions/computer-vision-syndrome (дата звернення: 09.05.2025).

31. Mental health at work. World Health Organization. 2022. URL: https://www.who.int/news-room/fact-sheets/detail/mental-health-at-work (дата звернення: 20.05.2025).

32. Гандзюк М. П., Желібо Є. П., Халімовський М. О. Основи охорони праці : підручник. Київ : Каравела, 2007. 392 с.

33. Гурик О. Я., Окіпний І. Б. (укладачі). Методичні вказівки для написання розділу „Безпека життєдіяльності, основи охорони праці” в кваліфікаційних роботах здобувачів освітнього ступеня «бакалавр». Тернопіль : ТНТУ, 2021. 36 с.

34. Duda, O., Martsenko, S., Matsiuk, O., Kunanets, N., & Pasichnyk, V. (2020). Building secure Urban information systems based on IoT technologies. CEUR Workshop Proceedings, 317-328.

35. Барабах, Р. Т., Дуда, О. М., Дуда, Х. О., Кунанець, Н. Е., Машіка, Г. В., & Пасічник, С. О. (2024). Побудова туристичних інтернет-порталів з інтуїтивно зрозумілими інтерфейсами. Scientific Bulletin of UNFU, 34(1), 67-77.

36. Kunanets, N., Zhovnir, Y., Burov, Y., Duda, O., & Pasichnyk, V. (2025). DESIGNING THE STRUCTURE AND ARCHITECTURE OF SITUATION-AWARE SECURITY INFORMATION SYSTEMS FOR RESIDENTIAL COMPLEXES. Eastern-European Journal of Enterprise Technologies, 133(9).

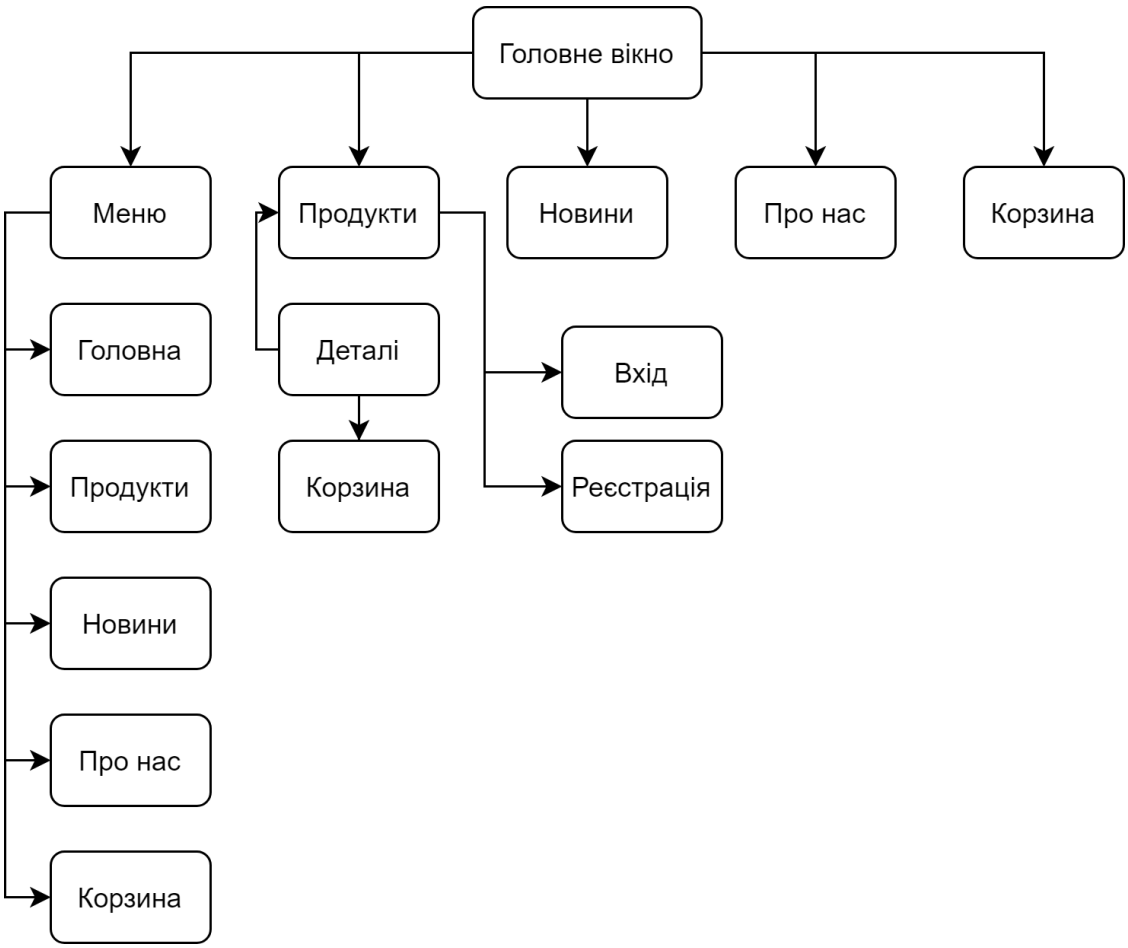
37. Пасічник, С., Мага, А., Кунанець, Н., Лозицький, О., Петрушина, Б., & Дуда, О. (2024). Проектування інтерфейсів інформаційної системи «розумне домогосподарство» з використанням методу персон. Вісник національного університету «Львівська політехніка» серія Інформаційні системи та мережі.

38. Лучкевич, М., Шаклеіна, І., & Дуда, О. М. (2025). Вплив сучасних хмарних технологій на ефективність DevOps-процесів. Вісник Тернопільського національного технічного університету, 117(1), 112-122.

ДОДАТКИ

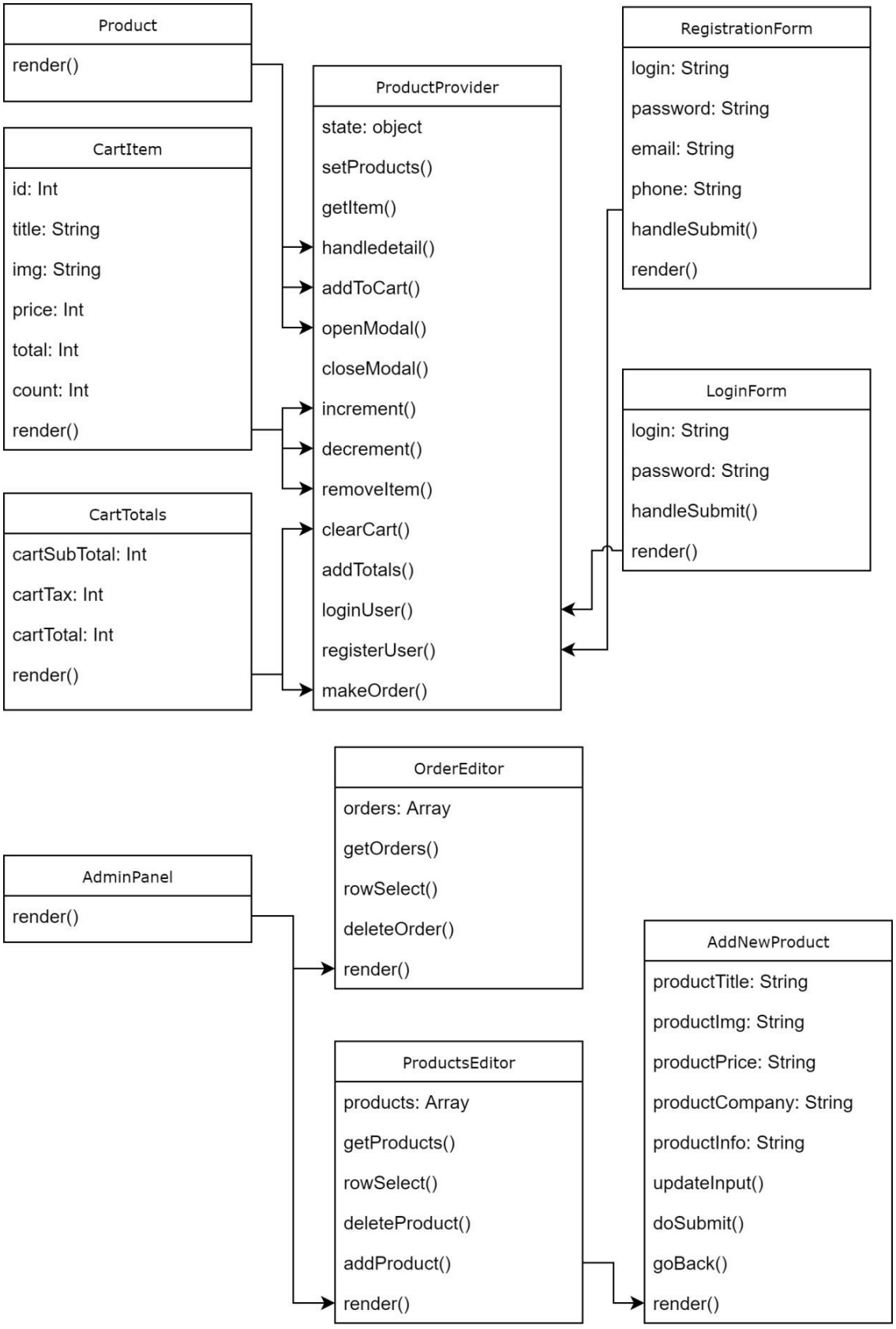
Додаток А

Структурна схема інтернет-магазину "PixelPhone"



Додаток Б

UML-діаграма класів клієнтської частини



Додаток В

Фрагмент реалізації маршруту авторизації

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/user');
router.post('/login', async (req, res) => {
    try {const { login, password } = req.body; const user = await
User.findOne({ login });
        if (!user) {
            return res.status(400).json({ message: 'Неправильний
логін або пароль' });}
        const    isMatch    =    await    bcrypt.compare(password,
user.password);
        if (!isMatch) {
            return res.status(400).json({ message: 'Неправильний
логін або пароль' }); }
        const payload = { id: user.id, login: user.login, role:
user.role };
        jwt.sign(
            payload,
            process.env.JWT_SECRET,
            { expiresIn: 3600 },
            (err, token) => {
                if (err) throw err;
                res.json({ success: true, token: 'Bearer ' + token
});
            }        );
    } catch (err) {
        res.status(500).send('Помилка сервера');
    }});
```

Додаток Г

Реалізація маршрутизації в компоненті App.js

```
import React, { Component } from 'react';
import { Switch, Route } from 'react-router-dom';
import Navbar from './components/Navbar';
import ProductList from './components/ProductList';
import Details from './components/Details';
import Cart from './components/Cart';
import AdminPanel from "./components/AdminPanel";
import Default from './components/Default';
import AdminRoute from './components/AdminRoute';
class App extends Component {
  render() {
    return (
      <React.Fragment>
        <Navbar />
        <Switch>
          <Route exact path="/" component={ProductList} />
          <Route path="/details" component={Details} />
          <Route path="/cart" component={Cart} />
          {/* Інші публічні маршрути */}
          <AdminRoute path="/AdminPanel" component={AdminPanel} />
          <Route component={Default} />
        </Switch>
      </React.Fragment>
    );  }}
export default App;
```