МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р. техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

«WEB-ЗАСТОСУНОК ДЛЯ ПРОДАЖУ КОМП'ЮТЕРНОЇ ТЕХНІКИ ТА ОБЛАДНАННЯ НА БАЗІ СТЕКІВ ТЕХНОЛОГІЙ MERN»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21910227

Виконав студент 4-го курсу, групи 402
_____*О. О. Шальнєв*
(підпис, ініціали та прізвище)
«___» _____ 2023 р.

Керівник: *___д-р. техн. наук, доцент___*
(наук. ступінь, вчене звання)
_____*О. В. Козлов*
(підпис, ініціали та прізвище)
«___» _____ 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р. техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Шальнєву Олександру Олександровичу.

1. Тема кваліфікаційної роботи «Web-застосунок для продажу комп'ютерної техніки та обладнання на базі стеків технологій MERN».

Керівник роботи Козлов Олексій Валерійович, д-р. техн. наук, доцент.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні дані до роботи: вимоги до функціональності, користувацького інтерфейсу та безпеки веб-застосунку для продажу комп'ютерів та комп'ютерного обладнання; оцінка потреб потенційних користувачів онлайн-магазинів комп'ютерного обладнання; обраний стек технологій для розробки веб-застосунку.

Очікуваний результат: розроблена система електронної комерції для продажу комп'ютерів та комп'ютерного обладнання, яка оптимізована для користувацького досвіду, забезпечує безпечну обробку даних та транзакцій, та легко масштабується для подальшого розвитку.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз сучасного стану ринку електронної комерції, з акцентом на продаж комп'ютерів та комп'ютерного обладнання;

– огляд існуючих технологій та методологій для розробки веб-застосунків;

– розробка веб-застосунку для продажу комп'ютерів та комп'ютерного обладнання на основі стеку технологій MERN, з використанням Google OAuth, AWS S3, Stripe та інші;

– проведення тестування розробленого веб-застосунку, включаючи його адміністративну панель, функціонал користувача, систему пошуку, кошик та оплату;

– оцінка роботи веб-застосунку та визначення потенційних напрямків для подальшого розвитку та оптимізації.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Охорона праці при роботі з екранними пристроями»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Боженко А. Л. викладач	

Керівник роботи __ д-р. техн. наук, доцент Козлов О. В. ___
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання _____Шальнєв О. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «__» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Web-застосунок для продажу комп'ютерної техніки та обладнання на базі стеків технологій MERN

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівника БКР	27.10.2022	30.10.2022	Виконано
2	Отримання завдання на виконання БКР	08.11.2022	10.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	01.12.2022	01.12.2022	Виконано
4	Отримання завдання на переддипломну практику	12.03.2023	12.03.202	Виконано
5	Аналіз ринку, визначення основних потреб користувачів та розробка вимог	13.03.2023	27.03.2023	Виконано
6	Вивчення та вибір технологій для розробки веб-застосунку	28.03.2023	11.04.2023	Виконано
7	Розробка архітектури, планування та дизайн веб-застосунку	12.04.2023	21.04.2023	Виконано
8	Розробка фронтенду та бекенду, налаштування бази даних і інтеграція її	22.04.2023	21.05.2023	Виконано
9	Фінальне злиття і налаштування коду веб-застосунку	22.05.2023	26.05.2023	Виконано
10	Розробка тестових сценаріїв і тестування веб-застосунку	27.05.2023	28.05.2023	Виконано
11	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
12	Доробка та остаточне оформлення БКР	02.06.2023	08.06.2023	Виконано
13	Подання БКР рецензенту	09.06.2023	13.06.2023	Виконано
14	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	14.06.2023	21.06.2023	
15	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	

Розробив студент _____Шальнєв О. О._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи _____ д-р. техн. наук, доцент Козлов О. В. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 2023 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 402 ЧНУ ім. Петра Могили

Шальнєва Олександра Олександровича

Тема: «Web-застосунок для продажу комп'ютерної техніки та обладнання на базі стеків технологій MERN»

Актуальність теми обумовлена стрімким розвитком електронної комерції та постійним зростанням потреби в комп'ютерній техніці та обладнанні. Потреба в створенні веб-застосунків для продажу комп'ютерної техніки стає все більш актуальною в сучасних умовах, коли традиційні магазини переходять в онлайн формат, що дозволяє значно розширити географію своїх покупців і покращити сервіс.

Об'єктом роботи є процеси створення веб-застосунків для продажу комп'ютерної техніки та обладнання на базі стеку технологій MERN.

Предметом роботи є технології та програмні засоби проектування веб-застосунків з продажу комп'ютерної техніки та обладнання.

Метою роботи є підвищення ефективності процесу продажу комп'ютерної техніки та обладнання за рахунок розробки веб-застосунку на основі стеку технологій MERN.

Завдання, які ставляться для досягнення поставленої мети, включають:

– аналіз ринку електронної комерції для продажу комп'ютерної техніки та обладнання;

– розробка архітектури та компонентів веб-застосунку, використовуючи стек технологій MERN;

– імплементація серверної та клієнтської частини веб-застосунку, інтеграція MongoDB для зберігання даних;

– тестування веб-застосунку на відповідність вимогам, вивчення можливостей щодо масштабування та оптимізації.

— Робота складається з теоретичного огляду, частини розробки, тестування, аналізу та спеціальної частини з охорони праці. Пояснювальна записка включає вступ, чотири розділи та висновки.

У першому розділі розглядається стан ринку електронної комерції для продажу комп'ютерної техніки та обладнання. У другому розділі представлений огляд технологій та програмних засобів, використаних у розробці веб-застосунку. Третій розділ описує процес розробки веб-застосунку на основі стеку технологій MERN, а у четвертому розділі наведено результати тестування та аналізу. Спеціальний розділ присвячено питанням охорони праці під час розробки та використання веб-застосунку.

Бакалаврська кваліфікаційна робота містить 81 сторінки, 88 рисунків, 35 використаних джерел.

Ключові слова: *MERN стек, електронна комерція, веб-застосунок, продаж комп'ютерної техніки, продаж комп'ютерного обладнання*.

ABSTRACT

for bachelor's qualification work of a student of 402 group at Petro Mohyla Black Sea National University

Shalniev Oleksandr Oleksandrovych

Theme: «Web-application for the sale of computer hardware and equipment based on MERN technology stacks»

The relevance of the topic is due to the rapid development of e-commerce and the ever-increasing demand for computer hardware and equipment. The need to create web applications for the sale of computer equipment is becoming increasingly relevant in modern conditions, when traditional stores are moving to an online format, which allows them to significantly expand the geography of their customers and improve service.

The object of the study is the processes of creating web applications for the sale of computer equipment and hardware based on the MERN technology stack.

The subject of the study is technologies and software tools for designing web applications for the sale of computer equipment and hardware.

The aim of the study is to increase the efficiency of the process of selling computer equipment and hardware by developing a web application based on the MERN technology stack.

The tasks set to achieve this goal include:

– analysis of the e-commerce market for the sale of computer equipment and hardware;

– development of the architecture and components of the web application using the MERN technology stack;

– implementation of the server and client parts of the web application, integration of MongoDB for data storage;

– testing the web application for compliance with the requirements, exploring opportunities for scaling and optimization.

The work consists of a theoretical overview, development, testing, analysis, and a special part on labor protection. The explanatory note includes an introduction, four chapters and conclusions.

The first section discusses the state of the e-commerce market for the sale of computer hardware and equipment. The second section provides an overview of the technologies and software used in the development of the web application. The third section describes the process of developing a web application based on the MERN technology stack, and the fourth section presents the results of testing and analysis. A special section is devoted to the issues of labor protection during the development and use of the web application.

The bachelor's thesis contains 81 pages, 88 images, and 35 references.

Key words: *MERN stack, e-commerce, web application, sale of computer equipment, sale of computer equipment*.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ПК – персональний комп'ютер

MERN – MongoDB Express React Node

OAuth – Open Authorization

API – application programming interface

CDN – content delivery network

VS – Visual Studio

CSS – cascading style sheets

AWS S3 – Amazon Web Simple Storage Service

CLI – command line interface

CRUD – create read update delete

ВСТУП

У сучасному світі інформаційних технологій електронна комерція відіграє важливу роль у національній економіці. Одним із напрямків електронної комерції є продаж комп’ютерної техніки через Інтернет. Це дозволяє користувачам швидко та легко вибирати, порівнювати та купувати товари, не відходячи від комп’ютера. Останнім часом вебдодатки, побудовані на стеку MERN, включаючи MongoDB, Express, React і Node.js, серед інших, стали дуже популярними. Ці технології дозволяють створювати надійні, швидкі та масштабовані вебдодатки.

Крім того, вони підтримують впровадження потужних і гнучких рішень електронної комерції в IT-обладнанні та роздрібній торгівлі обладнанням. Вони розроблені та впроваджені, щоб надати користувачам зручний і простий процес вибору продукту та покупки, тим самим збільшуючи продажі. Для бакалаврської кваліфікаційної роботи планується розробка вебдодатку, щоб задовольнити потреби ринку та спростити процес придбання готових комп’ютерів і комплектуючих.

Незважаючи на широку популярність сучасних вебдодатків для продажу комп’ютерної техніки та обладнання, все ще існують певні труднощі в окремих ситуаціях, які заважають користувачам швидко та легко здійснювати покупки. Серед них погана мобільна оптимізація, незрозумілий інтерфейс, повільне завантаження сторінок тощо.

Саме ці проблеми в веб-додатках для продажу комп’ютерної техніки та обладнання можна ефективно вирішити за допомогою MERN і стеку технологій Next.js. Використовуючи ці технології, можна розробляти надійні, швидкі та масштабовані вебдодатки, які надають користувачам високоякісні послуги та зручні інтерфейси.

Метою роботи є підвищення ефективності процесу продажу комп'ютерної техніки та обладнання за рахунок розробки вебзастосунку на основі стеку технологій MERN.

Завдання для досягнення поставленої мети:

– проаналізувати ринок електронної комерції для продажу комп'ютерної техніки та обладнання;

– розробити архітектуру та компоненти вебзастосунку, використовуючи стек технологій MERN;

– імплементувати серверну та клієнтську частини вебзастосунку, інтегрувати MongoDB для зберігання даних;

протестувати вебзастосунок на відповідність вимогам та вивчення можливостей щодо масштабування та оптимізації.

Об'єктом дослідження є процеси створення вебзастосунків для продажу комп'ютерної техніки та обладнання на базі стеку технологій MERN.

Предметом дослідження є технології та програмні засоби проектування вебзастосунків з продажу комп'ютерної техніки та обладнання.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ТА ПОСТАНОВКА ЗАДАЧІ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКУ З ПРОДАЖУ КОМП'ЮТЕРНОЇ ТЕХНІКИ ТА ОБЛАДНАННЯ НА ОСНОВІ СТЕКУ ТЕХНОЛОГІЙ MERN

1.1 Основні терміни, поняття та підходи в контексті розробки вебзастосунку для продажу комп'ютерної техніки та обладнання через Інтернет

У контексті обраної теми необхідно розуміти деякі ключові терміни та поняття, такі як:

– вебдодатки;

– комп'ютерна техніка та обладнання;

– стек MERN;

– електронна комерція;

– frontend;

– backend.

Зазвичай вебдодаток описують як програмне забезпечення, яке запускається через веб-браузер. Програми можуть виконувати різні функції, включаючи онлайн-продажі, управління бізнес-процесами, соціальні мережі тощо.

Стосовно комп’ютерного обладнання можна зазначити, що це широкий термін, який включає всі типи пов’язаних з комп’ютером товарів, включаючи основні системи, такі як персональні комп’ютери та ноутбуки, а також такі компоненти, як жорсткі диски, процесори, відеокарти тощо.

Електронна комерція – це ведення бізнесу в Інтернеті, що включає купівлю та продаж товарів або послуг через Інтернет, а також передачу даних і грошей для завершення транзакцій.

Кажучи про frontend, можна коротко виділити, що це частина вебпрограми, яку бачить користувач і з якою взаємодіє. Це включає інтерфейс користувача та візуальні частини програми.

Backend в свою чергу є частиною вебпрограми, яка відповідає за обробку даних, взаємодію з базами даних, виконання логіки програми та забезпечення зв’язку між інтерфейсом і сервером.

Усі ці концепції важливі для розуміння дослідницького завдання, і їх правильне використання є ключем до розробки ефективних вебдодатків для комп’ютерного обладнання та продажу обладнання на основі стеку технологій MERN.

Сфера продажу комп'ютерної техніки та техніки через Інтернет має свої унікальні особливості та компоненти. Сюди входять не лише фізичні товари, а й комп’ютерні послуги.

По-перше, асортимент пропонованої продукції. З одного боку, це можуть бути готові рішення: десктопи, ноутбуки, сервери, монітори, периферія. З іншого боку, він також включає комп’ютерні компоненти, включаючи процесори, материнські плати, оперативну пам’ять, відеокарти, жорсткі диски, SSD-накопичувачі, блоки живлення, корпуси та інші частини, які дозволяють створювати або оновлювати системні блоки комп’ютера.

Що стосується основних процесів у сфері електронної комерції, то серед них можна виділити: пошук і вибір товару, реєстрація замовлення, оплата, обробка замовлення, доставка товару клієнту, повернення товару, тощо.

Створення вебдодатків включає кілька підходів на основі різних технологій, мов програмування, фреймворків і бібліотек. Вибір методу залежить від цілей, вимог і ресурсів проекту. Нижче наведено найпопулярніші способи створення вебдодатків:

– традиційний або монолітний підхід [1]: вебпрограма побудована як єдине ціле, де інтерфейс і сервер тісно пов’язані між собою. Приклади мов програмування, які використовуються для цього підходу, включають PHP (з використанням фреймворків, таких як Laravel), Ruby (з використанням Ruby on Rails), Python (з використанням Django та Flask);

— підхід до мікросервісу: вебпрограма розділена на незалежні мікросервіси, і кожен мікросервіс відповідає за виконання певної функції. Мікросервіси можуть бути написані на різних мовах програмування і використовувати різні технології [2];

— підхід SPA: веб-програми, які запускаються у браузері та не потребують перезавантаження сторінок під час використання [3]. Прикладами фреймворків для створення SPA є React.js, Vue.js і Angular.js;

— підхід JAMstack: архітектура, заснована на клієнтському JavaScript, багаторазово використовуваних API та попередньо створеному вмісті. Цей підхід створює швидкі та надійні вебдодатки;

— підхід до стеку MERN: набір технологій JavaScript для створення сучасних вебдодатків. MongoDB використовується для бази даних, Express.js і Node.js створюють бекенд, а React.js використовується для інтерфейсу.

Кожен із цих методів має свої переваги та недоліки. Вибір підходу залежить від специфіки проекту, включаючи його розмір, вимоги до продуктивності, безпеки, швидкості розробки та інших факторів.

1.2 Огляд та аналіз наявних вебзастосунків для продажу комп'ютерної техніки та обладнання

Ринок електронної комерції має багато вебдодатків, які пропонують комп’ютерне обладнання та обладнання [4]. Деякі з них є звичайними магазинами, які пропонують широкий асортимент товарів у різних категоріях, тоді як інші спеціалізуються на конкретних продуктах або марках.

Специфічні функції відрізняються залежно від сайту, але багато з них включають фільтрування продуктів і пошук, системи перегляду та рейтингу. Безпека також є ключовим фактором, особливо коли йдеться про захист особистої інформації та платіжних даних.

У науковій літературі та Інтернет-ресурсах можна знайти численні дослідження та публікації, що аналізують тенденції та виклики у сфері комп’ютерної техніки та онлайн-торгівлі обладнанням. Ці ресурси сприяють глибшому розумінню потреб користувачів, поведінки покупців і технологічних інновацій у галузі.

У якості прикладу вже існуючих вебзастосунків з продажу комп’ютерної техніки та обладнання хотілося б виділити наступні:

- ArtLine;
- IT-Block;
- Telemart.UA.

Інтернет-магазин «ArtLine» [5]: Цей магазин існує з 2015 року і пропонує готові комп'ютери та комплектуючі. Він вже 8 років надає свої послуги та продовжує розвиватися. «ArtLine» є інтуїтивно зрозумілим та функціональним веб-застосунком з простим інтерфейсом та UI/UX дизайном (див. рис. 1.1).

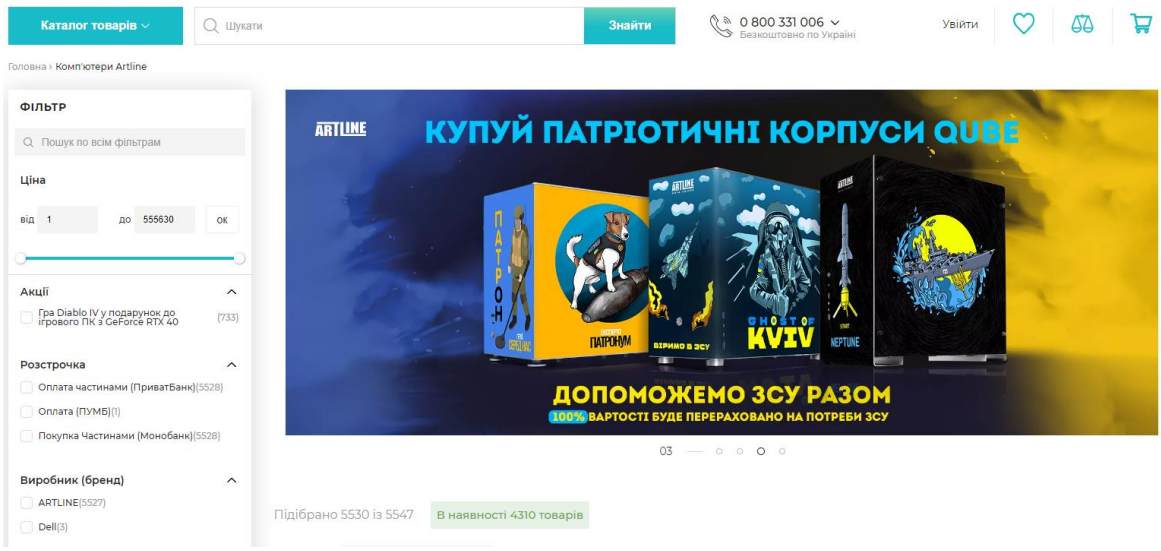

Рисунок 1.1 – Інтернет-магазин ArtLine

Інтернет-магазин «IT-Block» [6]: Цей магазин також спеціалізується на продажу готових комп'ютерів та комплектуючих. Окрім того, «IT-Block» надає можливість придбати товари у кредит та має цікаву функцію під назвою «Trade-In», яка дозволяє обміняти старий комп'ютер на новий зі знижкою. Дизайн та

користувацький інтерфейс «IT-Block» прості та зрозумілі, що сприяє зручному покупцеві та навігації по вебзастосунку (див. рис. 1.2).

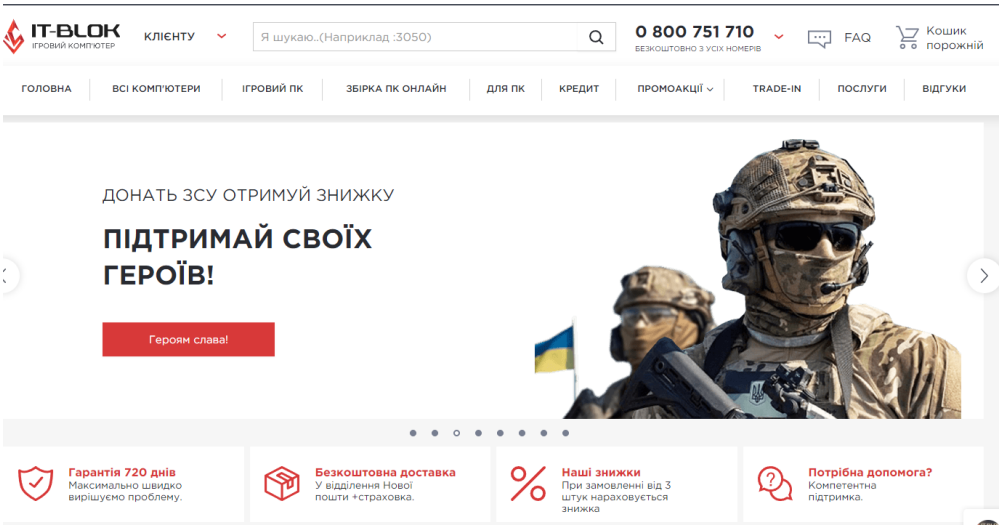

Рисунок 1.2 – Інтернет-магазин «IT-Block»

Інтернет-магазин «Telemart.UA» [7]: Як і його аналоги, спеціалізується на продажу, ремонті та збірці комп'ютерів та комплектуючих. «Telemart.UA» також пропонує функцію «Trade-In», яка дозволяє зекономити кошти при покупці нової техніки. Крім того, він відомий своїми низькими цінами на товари. Користувальницький інтерфейс «Telemart.UA» є простим та дружелюбним, а навігація по сайту зрозуміла (див. рис. 1.3).

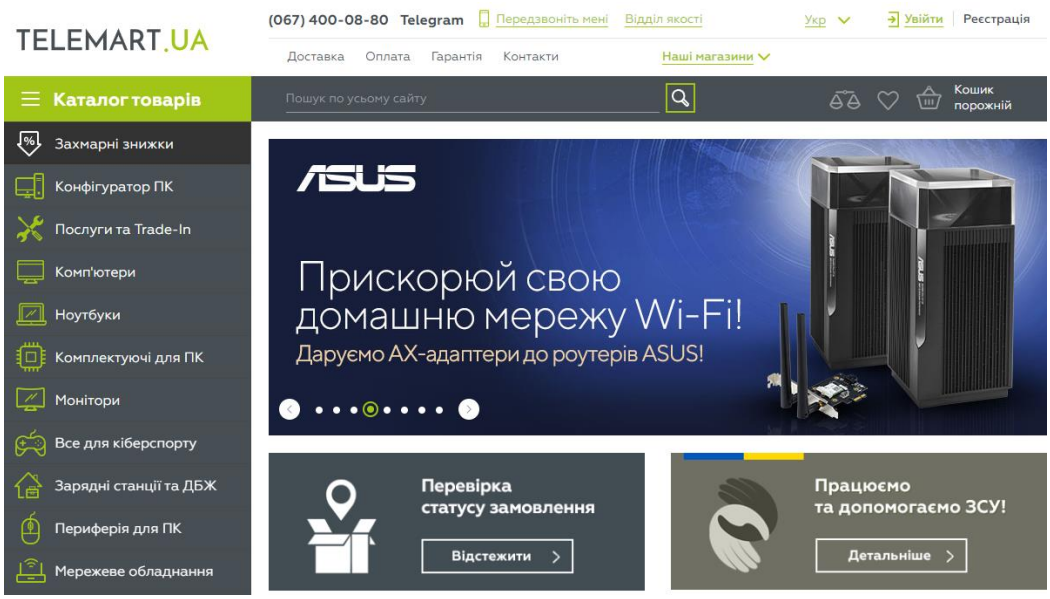

Рисунок 1.3 – Інтернет-магазин «Telemart.UA»

Всі ці магазини спеціалізуються на продажу комп'ютерної техніки та комплектуючих та надають зручні вебзастосунки для покупців. Взаємодія з цими успішними Інтернет-магазинами може надихнути на створення власного вебзастосунку, який надасть користувачам зручний та інтуїтивно зрозумілий інтерфейс, широкий асортимент товарів, привабливі акції та можливість обміну старого обладнання.

1.3 Ключові аспекти та постановка задачі

Створення вебзастосунку для продажу обчислювальної техніки та пристроїв на основі стеку технологій MERN передбачає виконання розробником ряду специфічних завдань, серед яких:

– проектування архітектури системи: важливим першим кроком є визначення загальної структури вашої веб-програми. Це включає визначення основних компонентів системи, їх взаємодії та розподілу відповідальності між ними;

– розробка інтерфейсу користувача: інтерфейс користувача повинен бути інтуїтивно зрозумілим і простим у використанні. Усі функції вебдодатків, мають бути легкодоступними;

– побудова бази даних: бази даних повинні бути оптимізовані для швидкого доступу до даних, надійності та масштабованості. Він повинен містити всю необхідну інформацію про товар, замовлення, користувача тощо;

– інтеграція з платіжними системами: щоб увімкнути онлайн-платежі, вебпрограма має бути інтегрована з однією чи кількома платіжними системами.

– безпека системи: вебзастосунки повинні бути захищені від різного роду атак і витоку даних. Це включає захист від несанкціонованого доступу та інших загроз безпеці;

– якість тестів і коду: вебзастосунки повинні не тільки функціонувати, але й бути надійними та безпомилковими. Тому важливо запровадити стратегію тестування, щоб забезпечити якість коду та завчасно виявити потенційні проблеми;

– оптимізація продуктивності: веб-програми мають бути швидкими та ефективними. Це означає, що розробники повинні враховувати такі аспекти, як оптимізація використання ресурсів сервера, кешування та інші підходи для забезпечення високої продуктивності;

– сумісність і доступність: веб-програми мають бути доступними для якнайширшої аудиторії, включаючи підтримку кросбраузерності, сумісність із мобільними пристроями та доступність для людей з обмеженими можливостями;

– впровадження та підтримка: після розробки вам потрібно реалізувати вебдодаток. Це може включати налаштування сервера, розгортання програмного забезпечення, налаштування безпеки тощо. Крім того, після запуску програми важливо переконатися, що вона підтримується та оновлюється вчасно.

Ці завдання відображають ключові аспекти розробки вебдодатків і формують основу для подальшого проектування та впровадження системи.

Об'єктом дослідження є процеси створення вебзастосунків для продажу комп'ютерної техніки та обладнання на базі стеку технологій MERN.

Предметом дослідження є технології та програмні засоби проектування вебзастосунків з продажу комп'ютерної техніки та обладнання.

Метою роботи є підвищення ефективності процесу продажу комп'ютерної техніки та обладнання за рахунок розробки вебзастосунку на основі стеку технологій MERN.

Завдання для досягнення поставленої мети:

– проаналізувати ринок електронної комерції для продажу комп'ютерної техніки та обладнання;

– розробити архітектуру та компоненти вебзастосунку, використовуючи стек технологій MERN;

– імплементувати серверну та клієнтську частини вебзастосунку, інтегрувати MongoDB для зберігання даних;

– протестувати вебзастосунок на відповідність вимогам та вивчення можливостей щодо масштабування та оптимізації.

Висновки до першого розділу

В ході аналізу предметної сфери та постановки задачі для розробки вебзастосунку з продажу комп`ютерної техніки та обладнання на основі стеку технологій MERN було виявлено, що ця сфера включає широкий асортимент товарів та послуг, які потребують ефективної та гнучкої платформи для онлайн-продажу.

Огляд аналогів та публікацій показав, що існує велика кількість вебзастосунків, що спеціалізуються на продажу комп`ютерної техніки. Однак, їх функціональні можливості, інтерфейси користувача та рівень безпеки варіюються, що вказує на потребу в розробці добре продуманого та високоякісного вебзастосунку.

Основні підходи до створення вебзастосунку включають проектування архітектури системи, розробку інтерфейсу користувача, побудову бази даних, інтеграцію з платіжними системами, забезпечення безпеки системи, тестування та якість коду, оптимізацію продуктивності, сумісність і доступність, а також впровадження та підтримка вебзастосунку.

Усі ці елементи мають бути враховані при постановці задачі для розробки вебзастосунку, щоб забезпечити його успішну реалізацію та ефективне використання в предметній області.

2 ІНФОРМАЦІЙНІ МЕТОДИ, ТЕХНОЛОГІЇ ТА СЕРВІСИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ СТВОРЕННЯ ВЕБЗАСТОСУНКУ

2.1 Опис методу колаборативної фільтрації та обраного середовища розробки (IDE)

Колаборативна фільтрація є одним з основних методів рекомендаційних систем, використовуваних для прогнозування інтересів користувача. Цей метод базується на аналізі взаємодій користувача з різними елементами (наприклад, товари в онлайн-магазині) та пошуку подібності між користувачами або елементами.

Колаборативна фільтрація може бути реалізована через два основних підходи: користувач-користувач (user-user) та елемент-елемент (item-item). Підхід користувач-користувач використовує рейтинги, дані одним користувачем, для прогнозування рейтингів іншого користувача, що має подібні інтереси. Підхід елемент-елемент, з іншого боку, аналізує подібність між елементами на основі їхньої оцінки різними користувачами.

Цей метод є ефективним для персоналізації вебзастосунків і служить ключовою технологією в багатьох інтернет-платформах, таких як Amazon, Netflix, YouTube і багатьох інших, де він використовується для високої персоналізації контенту для кожного користувача.

Стосовно середовища розробки, вибір зупинився на VS Code – це відкритий і безкоштовний редактор коду, розроблений Microsoft, який займає провідні позиції серед розробників по всьому світу [8].

Visual Studio Code поєднує в собі переваги повноцінного середовища розробки з простотою текстового редактора, що робить його максимально зручним і ефективним для розробки різноманітних проектів. Він підтримує велику кількість мов програмування.

В подальшому описі більш детально будуть розглянуті основні можливості та переваги VS Code (див. рис. 2.1), його конфігурацію, плагіни та особливості використання в рамках даного проекту.

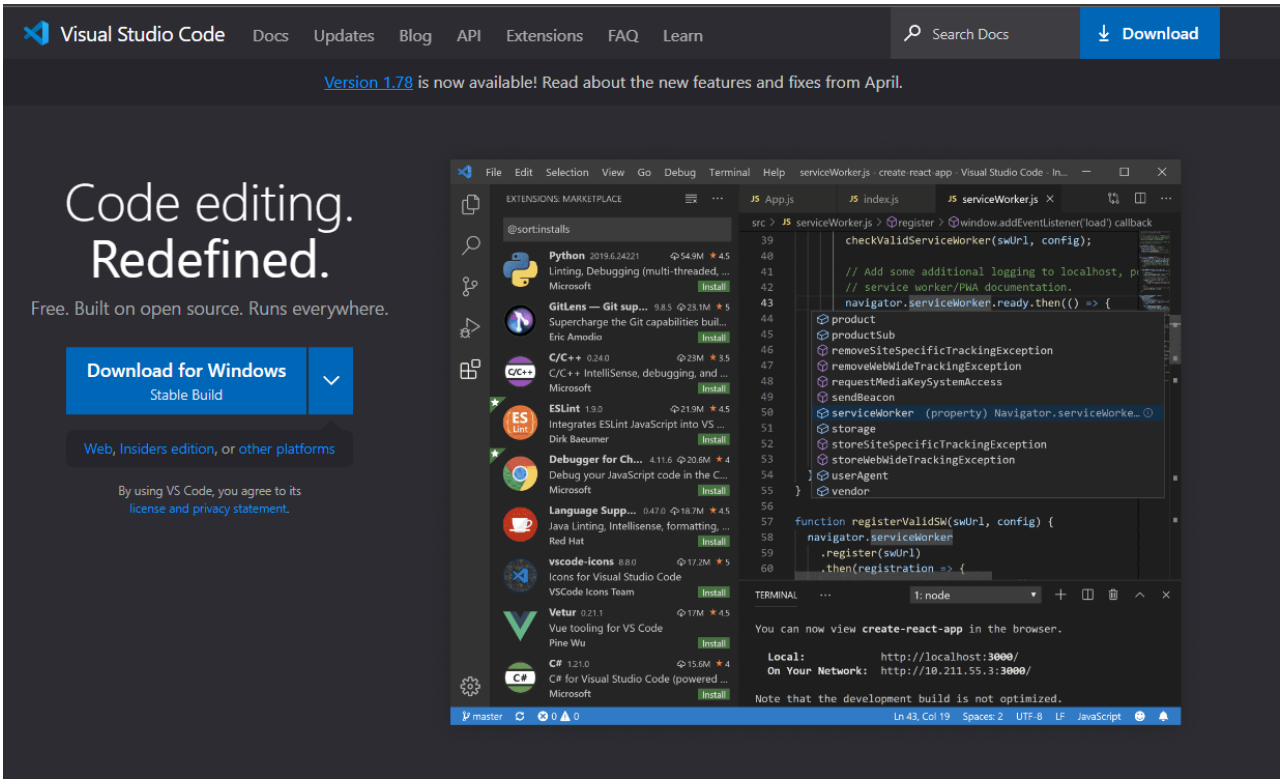

Рисунок 2.1 – Середовище розробки VS Code

Основні причини цього вибору включають:

– безкоштовність та відкритий код: VS Code є відкритим та безкоштовним середовищем розробки, що дозволяє зекономити кошти на ліцензіях та забезпечує доступність його використання для розробників незалежно від їх фінансових можливостей;

– підтримка мов програмування: VS Code пропонує підтримку синтаксису для багатьох мов програмування, включаючи JavaScript, TypeScript, HTML та CSS, які використовуються в стеках MERN та Next.js;

– розширення та налаштування: VS Code має велику кількість розширень, які дозволяють додати нові функції та підтримку технологій. Це дозволяє легко інтегрувати різні інструменти, необхідні для роботи з MERN стеком та Next.js;

– інтуїтивний інтерфейс користувача та навігація: VS Code має зручний інтерфейс, який спрощує навігацію та роботу з кодом. Функції, такі як IntelliSense, доповнення коду, швидкі фікси та перехід між файлами сприяють підвищенню продуктивності розробки;

– інтеграція з системами контролю версій: VS Code має вбудовану підтримку систем контролю версій, таких як Git, що дозволяє розробникам легко стежити за змінами коду, створювати гілки, злиття та розв'язувати конфлікти прямо в середовищі розробки;

– кросплатформеність: VS Code доступний для різних операційних систем, включаючи Windows, macOS та Linux. Це забезпечує гнучкість та зручність розробки для команди, що складається з розробників з різними операційними системами;

– вбудований термінал: VS Code має вбудований термінал, який дозволяє виконувати команди в командному рядку без потреби використання зовнішніх терміналів;

– спільнота та документація: оскільки VS Code є одним з найпопулярніших середовищ розробки, існує велика спільнота користувачів та розробників, яка створює додаткові матеріали, навчальні курси та підтримку.

Загалом, Visual Studio Code є відмінним вибором для розробки вебзастосунків на основі стеків технологій MERN та Next.js, завдяки його гнучкості, підтримці різних мов програмування, розширенням, зручному інтерфейсу.

2.2 Опис використання та особливостей MERN стеку в розробці вебзастосунків

При виконанні роботи було використано стек MERN (див. рис. 2.2), що складається з назв чотирьох ключових технологій, які використовуються для розробки повноцінних вебзастосунків.

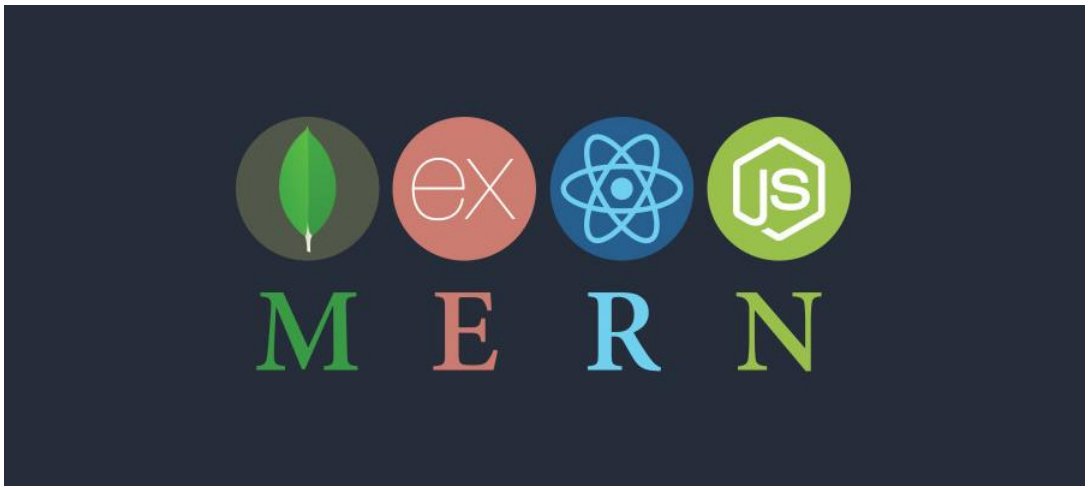

Рисунок 2.2 – MERN стек

База даних MongoDB [9] – це NoSQL база даних (див. рис. 2.3), що зберігає дані у вигляді гнучких документів JSON. MongoDB відрізняється високою продуктивністю, масштабованістю та легкістю використання. Ця база даних дозволяє розробникам працювати з даними, що мають різну структуру, без необхідності використання жорсткої схеми.

Рисунок 2.3 – База даних MongoDB

Гнучкий й легкий серверний фреймворк Express (див. рис. 2.4) для Node.js [10], який допомагає розробникам створювати вебдодатки та API швидко і зручно. Express надає набір інструментів для реалізації маршрутизації, обробки запитів та відповідей, а також роботи з middleware.

Рисунок 2.4 – Серверний фреймворк Express

Популярний фреймворк React для JavaScript від Facebook для створення користувацьких інтерфейсів. React використовує концепцію компонентів, що дозволяє розробникам створювати великі вебдодатки, розділяючи інтерфейс на незалежні, легко використовувані та повторно використовувані компоненти [11]. React також використовує віртуальний DOM для оптимізації рендерингу сторінки, що дозволяє підвищити продуктивність вебдодатку.

Кросплатформене середовище виконання для JavaScript на сервері Node.js (див. рис. 2.5), яке засноване на двигуні JavaScript V8 від Google. Node.js дозволяє розробникам використовувати одну мову програмування (JavaScript) на клієнтській та серверній сторонах, спрощуючи розробку та підтримку проекту. Node.js має потужний екосистему модулів та пакетів, що забезпечують розширені можливості та зручність у розробці. Одним з найпопулярніших менеджерів пакетів для Node.js є npm (Node Package Manager), який спрощує встановлення, оновлення та керування залежностями проекту [12].

Рисунок 2.5 – Кроспларформенне середовище Node.js

2.3 Опис використаних фреймворків

Сучасні вебзастосунки вимагають використання гнучких, потужних і швидких засобів розробки. У цьому випадку використовувані фреймворки Next.js і React.js відповідають усім цим вимогам, пропонуючи розробникам можливість створювати ефективні, надійні та високопродуктивні вебдодатки.

Як вже було зазначено, Next.js – це фреймворк JavaScript, спеціально розроблений для візуалізації та генерації статичних веб-сайтів на стороні сервера на основі React. Next.js надає розробникам інструменти для створення швидких вебдодатків, оптимізованих для пошукових систем [13].

Цей фреймворк автоматизує багато тонкощів розробки, наприклад: В. Візуалізація на стороні сервера, поділ коду, статична оптимізація, автоматичне додавання CSS тощо. Крім того, Next.js має вбудовану підтримку маршрутизації на основі файлової системи, що спрощує розробку багатосторінкових вебдодатків.

Next.js (див. рис. 2.6) також допомагає оптимізувати вебдодатки для пошукових систем за допомогою відтворення на стороні сервера та генерації статичної сторінки. Це означає, що весь вміст вашої веб-програми повністю доступний для індексування пошуковими системами, що покращує вашу видимість у результатах пошуку.

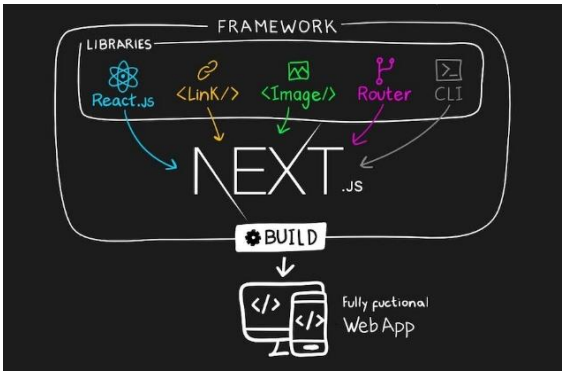

Рисунок 2.6 – Фреймворк Next.js

Також слід додати ще дещо про фреймворк React, який був описаний у попередньому підрозділі: основна ідея React полягає в тому, щоб розбити інтерфейс користувача на окремі компоненти, які можна багаторазово

використовувати [14]. Кожен компонент React може мати власний стан і життєвий цикл, що дозволяє створювати насичені та динамічні вебдодатки.

Однією з головних особливостей React (див. рис. 2.7) є використання віртуального DOM. Це спрощує процес відображення змін на веб-сторінці. Замість того, щоб оновлювати всю сторінку при кожній зміні, React оновлює лише елементи DOM, які змінилися. Це покращує продуктивність вебдодатків і забезпечує кращу взаємодію з користувачем.

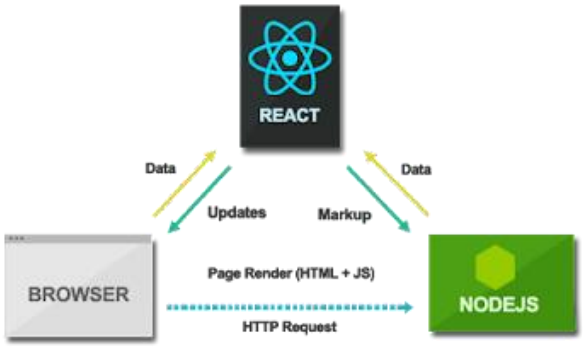

Рисунок 2.7 – Робота фреймворка React

Тому обидва фреймворки відіграють важливу роль у розробці сучасних вебдодатків. Вони спрощують процес розробки та надають розробникам потужні інструменти для створення потужних оптимізованих вебдодатків, які відповідають вимогам сучасного Інтернет-середовища.

2.4 Опис використаних бібліотек

У цьому підрозділі розглянуті деякі корисні бібліотеки та інструменти, які можуть бути використані при розробці вебзастосунків. Описані бібліотеки включають:

– бібліотека SweetModal: використовується для створення модальних вікон в веб-застосунках [15]. Ця бібліотека пропонує безліч можливостей для розробників, дозволяючи легко і швидко реалізувати бажаний вигляд модального вікна. Вона має вбудовані функції для анімації входу та виходу, підтримує вміст

AJAX, обробку подій та навіть має вбудовану підтримку обробки промісів. SweetModal є відмінним інструментом для створення інтуїтивно зрозумілих та красивих інтерфейсів взаємодії з користувачами;

– бібліотека SweetAlert: дозволяє створювати красиві, налаштовувані й анімовані спливаючі повідомлення [16]. SweetAlert надає велику гнучкість при створенні сповіщень та робить використання стандартних JavaScript-вікон попередження, підтвердження або повідомлення про помилку простим та елегантним. Це включає в себе можливість контролю часу відображення спливаючих вікон, кастомізацію тексту та кнопок, а також використання HTML вмісту і CSS стилізації;

– бібліотека Chart.js: дозволяє легко створювати вражаючі графіки і діаграми для вебзастосунків [17]. Вона пропонує 8 типів діаграм: лінійні, стовпчикові, бульбашкові, пирогові, об'ємні, радіальні, полярні та розсіювання. Chart.js має потужний API, який дозволяє налаштовувати діаграми за допомогою широкого спектра опцій, включаючи стилі, анімації, легенди, інструменти для наведення миші та багато іншого;

– бібліотека React-Chartjs-2: обгортка для бібліотеки Chart.js, що дозволяє використовувати всю потужність Chart.js в середовищі React. Вона включає у себе компоненти для всіх типів діаграм, що підтримуються Chart.js, і дозволяє реагувати на зміни стану та пропсів React, надаючи додатковий рівень інтеграції між двома бібліотеками [18]. Завдяки React-Chartjs-2, розробники можуть легко інтегрувати інтерактивні діаграми в свої вебзастосунки на React, забезпечуючи високий рівень гнучкості та контролю;

– мікросервер Micro: асинхронний HTTP мікросервер для Node.js, який забезпечує основну функціональність для розробки веб-сервісів з мінімальним набором залежностей [19]. Завдяки своїй простоті та ефективності, Micro є ідеальним вибором для створення невеликих, високопродуктивних HTTP-сервісів;

– бібліотека Styled-components: дозволяє використовувати стилі CSS безпосередньо в компонентах JavaScript [20]. Вона пропонує набір API для

створення налаштовуваних, масштабованих та автоматично оновлюваних стилів. Styled-components дозволяє використовувати всі можливості CSS, включаючи селектори, анімації, медіа-запити та багато іншого, прямо в коді компонентів React;

– бібліотека React-spinners: колекція компонентів завантаження для React, які використовують бібліотеку emotion для стилізації [21]. React-spinners надає велику кількість анімованих спінерів, які легко інтегруються в будь-який проект React;

– бібліотека Mongoose: об'єктно-орієнтована бібліотека, яка допомагає в роботі з MongoDB в Node.js. Mongoose надає простий API для створення, читання, оновлення та видалення документів в MongoDB (див. рис. 2.8), і включає в себе підтримку схем, валідації даних, пошуку та інших функцій, які спрощують роботу з базами даних MongoDB [22, 35];

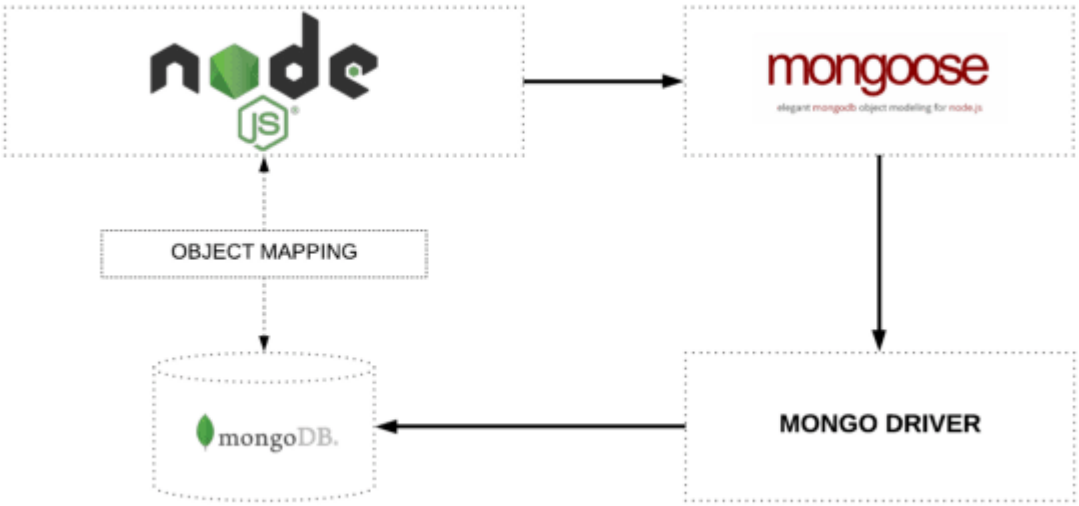

Рисунок 2.8 – Робота бібліотеки Mongoose

– бібліотека Multiparty: необхідна для обробки форм з множинними частинами, що включає обробку завантаження файлів. Бібліотека є незамінною для обробки даних форм, що надійшли через HTTP-запити, особливо коли є потреба в завантаженні файлів [23];

– бібліотека SortableJS: дозволяє створювати сортуємі списки з перетягуванням на основі HTML5. Сортуємі списки є важливою частиною багатьох

вебдодатків, і SortableJS надає простий спосіб реалізації цієї функціональності з мінімальними зусиллями [24];

– утилітарна CSS-бібліотека Tailwind CSS: надає набір низькорівневих утиліт для створення проектів веб-дизайну. Замість традиційного підходу з використанням готових компонентів, Tailwind дозволяє розробникам створювати власні дизайни з використанням атомарних CSS-класів (див. рис. 2.9). Цей підхід надає величезний рівень гнучкості та контролю, дозволяючи розробникам створювати унікальні та налаштовувані дизайни без необхідності переписувати готові стилі [25].

Рисунок 2.9 – Утилітарна CSS-бібліотека Tailwind CSS

Кожна з цих бібліотек має власні функції та переваги, які покращують якість і функціональність вебдодатків.

Ще одна потужна бібліотека називається Axios, яка дозволяє взаємодіяти із зовнішніми API та робити HTTP-запити з вебзастосунку. Простий у використанні та зрозумілий синтаксис. Axios надає зручні методи для роботи з різними видами запитів, підтримує обробку помилок і дозволяє встановлювати заголовки запитів. Це універсальний інструмент для розробки як браузерних, так і серверних

вебдодатків. Axios дозволяє веброзробникам легко взаємодіяти із сервером, виконувати запити та оновлювати статус веб-сторінок [33, 34].

2.5 Опис використаних сервісів

Аутентифікація та авторизація користувачів, а також надійне та масштабоване зберігання даних є ключовими складовими успішної веб-програми. У розробці цього проекту використовувалися дві основні служби: Google Cloud OAuth для автентифікації користувачів і AWS S3 для управління та зберігання даних. Обидві служби надають набір інструментів, які розробники можуть використовувати для створення потужних і безпечних вебдодатків.

Google Cloud OAuth є ключовим сервісом для розробників, які потребують безпечної автентифікації та авторизації користувачів. OAuth 2.0 є протоколом стандарту відкритої авторизації, що дозволяє стороннім застосункам отримувати обмежений доступ до користувацького облікового запису [26, 31, 32]. За допомогою Google Cloud OAuth, користувачі можуть надавати застосунку доступ до своїх даних, збережених в Google, без необхідності розкривати свій пароль.

Google Cloud OAuth допомагає в реалізації безпечного процесу входу користувачів в застосунок. Він надає розробникам можливість використовувати Google як провайдера ідентифікації, що дозволяє користувачам входити в застосунок за допомогою свого Google облікового запису. Це полегшує процес реєстрації та входу для користувачів, а також підвищує рівень безпеки, оскільки користувацькі паролі не передаються безпосередньо застосунку.

Процес аутентифікації (див. рис. 2.10) з використанням Google OAuth, який включає в себе наступні покрокові етапи:

1) ініціалізація: користувач тисне на кнопку "Увійти з Google" на вебзастосунку. Вебзастосунок має вже зареєстрований Google Client ID, що є ідентифікатором цього додатку для Google;

2) запит на авторизацію: вебзастосунок перенаправляє користувача на сторінку авторизації Google з декількома параметрами в URL;

3) авторизація: користувач вводить свої облікові дані Google, щоб авторизуватися, і надає згоду на доступ до вказаних областей;

4) отримання коду авторизації: після успішної авторизації Google перенаправляє користувача назад на вказаний URL перенаправлення, додаючи до нього код авторизації в якості параметра;

5) обмін коду авторизації на токен: вебзастосунок тепер може обміняти цей код авторизації на токен доступу, відправивши ще один запит до сервера Google. В цей запит включаються такі деталі, як код авторизації, Client ID, Client Secret (секретний ключ клієнта) та URL перенаправлення;

6) отримання токену доступу: Google відповідає на цей запит, надсилаючи токен доступу;

7) використання токену доступу: тепер вебзастосунок може використовувати цей токен доступу для доступу до Google API.

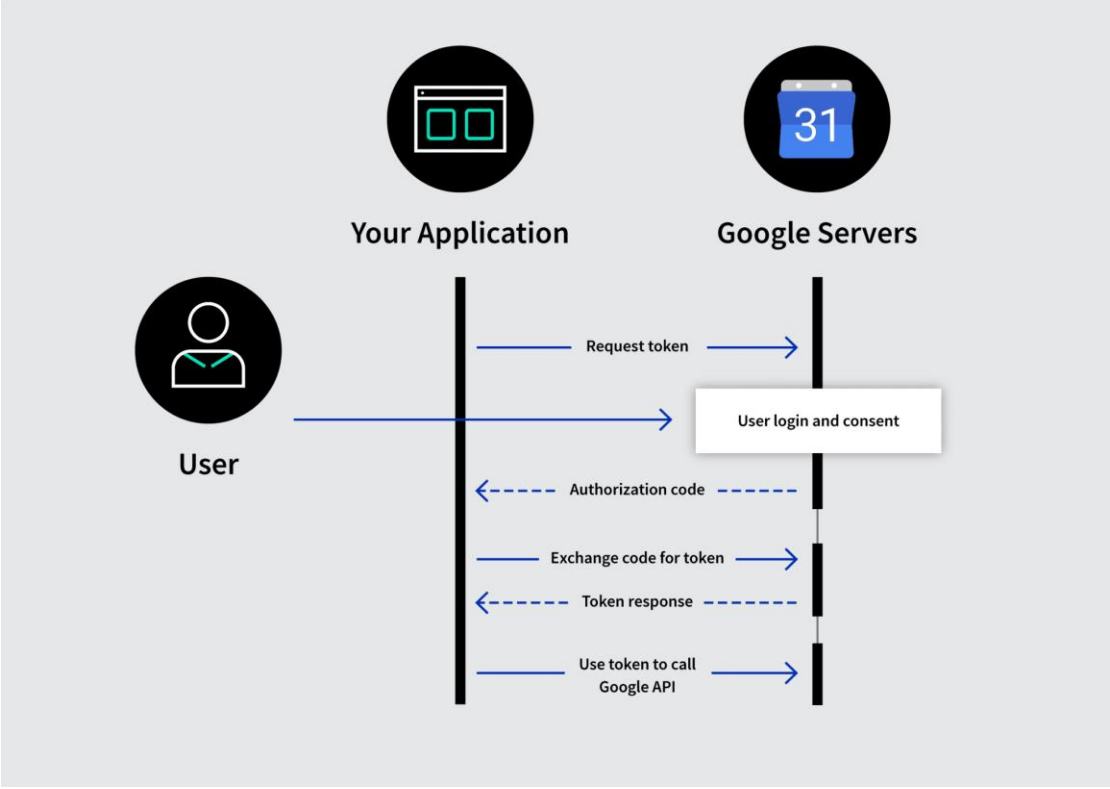

Рисунок 2.10 – Інтеграція користувача до застосунку з Google Cloud OAuth

AWS S3 є одним із найпопулярніших сервісів у хмарному середовищі. Це сховище об'єктних даних, призначене для зберігання та відтворення будь-якої

кількості даних будь-де та будь-коли в Інтернеті. Окрім цього, він зосереджений на забезпеченні надійності та доступності даних.

AWS S3 надає користувачам гнучкість у виборі моделі управління даними, що найкраще відповідає їх вимогам [27, 30]. Він підтримує три основні типи даних:

– стандартні, що використовуються для часто доступних даних;

– для даних, які менш часто запитуються, але потребують швидкого доступу, коли їх запитують;

– для довготривалого архівування даних.

AWS S3 забезпечує розподілене завантаження та відновлення, щоб оптимізувати передачу даних. Він також підтримує автомасштабування для обробки важких робочих навантажень, що робить його ідеальним для вебдодатків із інтенсивним об'ємом даних.

Використання цих послуг є основною частиною розробки сучасних, безпечних і масштабованих вебдодатків.

Висновки до другого розділу

У цьому розділі було зроблено глибокий аналіз інструментарію та технологій, що використовувались під час розробки вебзастосунку. В старті, було описано обране середовище розробки - Visual Studio Code, яке пропонує необхідну гнучкість, високу продуктивність і широкий вибір додаткових інструментів для ефективної роботи з JavaScript та супутніми технологіями.

Також було представлено заглиблений огляд MERN-стеку, який є ключовим технологічним рішенням для розробленого вебзастосунку. Стек, задіяний для побудови повноцінних вебдодатків, де кожен з компонентів виконує важливу роль у цьому процесі.

Наступним етапом було описання використаних фреймворків, їхніх особливостей та переваг при створенні сучасних вебзастосунків. Крім того, було детально описано додаткові бібліотеки, що використовуються для надання специфічної функціональності та забезпечення більш комфортного та ефективного

процесу розробки. Нарешті, було розглянуто використання служб Google Cloud OAuth і AWS S3, що надають великі можливості для авторизації користувачів і хмарного зберігання даних відповідно.

В результаті цього розділу було систематично і цілеспрямовано проаналізовано та описано основні технології та сервіси, що використовуються для вирішення задач створення вебдодатків. Вибір технології є обдуманим і раціональним, що гарантує ефективну розробку та впровадження вебзастосунку.

3 РЕАЛІЗАЦІЯ ОБРАНИХ ТЕХНОЛОГІЙ ТА СЕРВІСІВ ДЛЯ ПОСТАВЛЕНОЇ ЗАДАЧІ СТВОРЕННЯ ВЕБЗАСТОСУНКУ

3.1 Реалізація технічної частини адміністративної панелі

Адміністративна панель є ключовою частиною вебзастосунку, яка забезпечує ефективне управління всіма аспектами вебзастосунку, включаючи управління продуктами, замовленнями, категоріями, налаштуваннями та інше. В даному підрозділі детально описується процес реалізації адміністративної панелі з використанням вибраних технологій та сервісів.

3.1.1 Реалізація backend частини та підключення до БД

Серверна, вона ж backend частина вебдодатку, відповідає за обробку запитів від клієнтів, доступ до баз даних і повернення відповідей клієнтам. Спеціальні каталоги та файли використовуються для організації та структурування коду серверної частини проекту, включаючи такі головні директорії, як: компоненти, бібліотеки, моделі, API, сторінки та файли конфігурації 'yarn.json' та 'package.json' та інші (див. рис. 3.1). Кожен з них має певну роль у проекті.

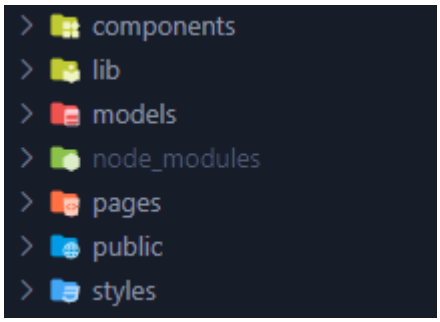

Рисунок 3.1 – Перелік директорій адміністративної панелі

– директорія components: містить реюзабельні компоненти, які використовуються в різних частинах серверної структури. Компоненти можуть включати логіку обробки запитів, обробки помилок, взаємодії з базою даних тощо;

– директорія lib: зазвичай використовується для зберігання корисних функцій та утиліт, які можуть використовуватися в різних місцях проекту. Це

можуть бути допоміжні функції для обробки даних, функції для роботи з датою та часом, функції для роботи з мережею та інше;

– директорія models: містить схеми бази даних, які використовуються для визначення структури даних, що зберігаються в базі даних. Схеми бази даних зазвичай включають визначення полів, типів даних, валідації та інші параметри;

– директорія API: містить код, який визначає API endpoints проекту. Кожен endpoint відповідає за конкретний тип запиту (GET, POST, DELETE тощо) та відповідає відповідно [28, 29];

– директорія pages: в цій директорії знаходяться шаблони веб-сторінок, що генеруються сервером. За допомогою цих шаблонів можна визначати вигляд веб-сторінки та динамічно вставляти дані на стороні сервера;

– package.json та yarn.json: і файли використовуються для управління залежностями проекту. Вони включають список пакетів, необхідних для роботи додатка, версій цих пакетів, команд для запуску проекту та інших конфігурацій.

Підключення до бази даних є критично важливим аспектом для будь-якого вебзастосунку, який обробляє дані. У цьому проекті використовується MongoDB як система управління базами даних. Щоб забезпечити ефективне та безпечне підключення до MongoDB, було створено два ключових файли: mongodb.js та mongoose.js.

Файл mongodb.js відповідає за підключення до MongoDB через драйвер MongoDB для Node.js. MongoClient є класом, який дозволяє програмам підключатися до MongoDB та виконувати операції з базою даних. Перш ніж підключитися до бази даних, перевіряється, чи існує змінна середовища MONGODB_URI, яка містить URL-адресу бази даних MongoDB. Якщо ця змінна не існує або є недійсною, генерується помилка. Інакше створюється новий екземпляр MongoClient, який підключається до бази даних.

Другий файл надає набір методів для роботи з даними, включаючи створення, оновлення, видалення та пошук. Функція mongooseConnect в цьому файлі перевіряє стан підключення до бази даних. Якщо підключення вже встановлено, воно

повертає існуюче підключення. Інакше, воно встановлює нове підключення до бази даних за допомогою mongoose.connect.

Обидва ці файли спільно забезпечують робоче підключення до MongoDB, що є важливим для реалізації всіх CRUD-операцій [29], що виконуються в адміністративній панелі.

3.1.2 Реалізація схем БД для визначення структури даних

Схеми бази даних є основою для організації та зберігання даних в будь-якому вебзастосунку. Вони визначають структуру даних, які будуть зберігатися в базі даних. У цьому підрозділі подається детальний огляд реалізації п'яти основних моделей бази даних , використаних в цьому проекті: Admin, Category, Order, Product, Setting.

Почнемо з моделі Admin.js (див. рис. 3.2), яка визначає схему для адміністраторів вебзастосунку. Вона використовується для того, щоб заносити до БД користувачів з правами «адміністратор» та лише вони можуть мати доступ до адміністративної панелі. Так як використовується авторизація за допомоги сервісів Google, дана моделі може містити лише поле з електронною поштою та моніторингом дати та часом створення і оновлення того чи іншого адміністратора.

Лістинг коду:

```
const { Schema, models, model } = require("mongoose");
const AdminSchema = new Schema
(
  {
    email: { type: String, unique: true, required: true },
  },
{ timestamps: true }
);
export const Admin = models?.Admin || model("Admin", AdminSchema);
```

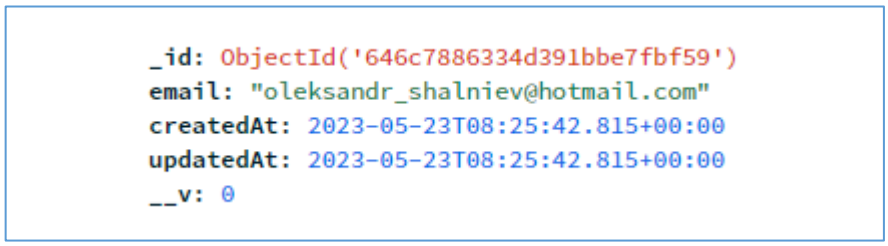

Рисунок 3.2 – Вигляд моделі Admin у базі даних

Модель Category (див. рис. 3.3) визначає схему для категорій продуктів вебзастосунку. Вона включає такі поля, як: назва категорії, батьківська категорія та властивості.

Лістинг коду:

```
import { Schema, model, models } from "mongoose";
const CategorySchema = new Schema({
  name: { type: String, required: true },
  parent: { type: Schema.Types.ObjectId, ref: "Category" },
  properties: [{ type: Object }],
});

export const Category = models?.Category || model("Category", CategorySchema);
```

Рисунок 3.3 – Вигляд моделі Category у базі даних

Наступна модель, яка називається Order (див. рис. 3.4) визначає схему для замовлень вебзастосунку. Вона включає такі поля, як: масив замовлених продуктів, ім'я замовника, пошта, місто, код, країна та адреса замовника, а також статус замовлення.

Лістинг коду:

```
const { Schema, model, models } = require("mongoose");
const OrderSchema = new Schema(
  {
    line_items: Object,
    name: String,
    email: String,
    city: String,
    postCode: String,
    country: String,
    address: String,
    paid: Boolean,
  },{timestamps: true,});
export const Order = models?.Order || model("Order", OrderSchema);
```

Рисунок 3.4 – Вигляд моделі Order у базі даних

Модель Product (див. рис. 3.5) визначає схему для продуктів вебзастосунку. Вона містить у собі такі поля, як: назва продукту, опис, ціна, зображення, категорія та властивості.

Лістинг коду:

```
import { model, Schema, models } from "mongoose";
const ProductSchema = new Schema(
  { title: { type: String, required: true },
    description: String,
    price: { type: Number, required: true },
    images: [{ type: String }],
    category: { type: Schema.Types.ObjectId, ref: "Category" },
properties: { type: Object },},
    {timestamps: true,}
);
export const Product = models.Product || model("Product",
ProductSchema);
```

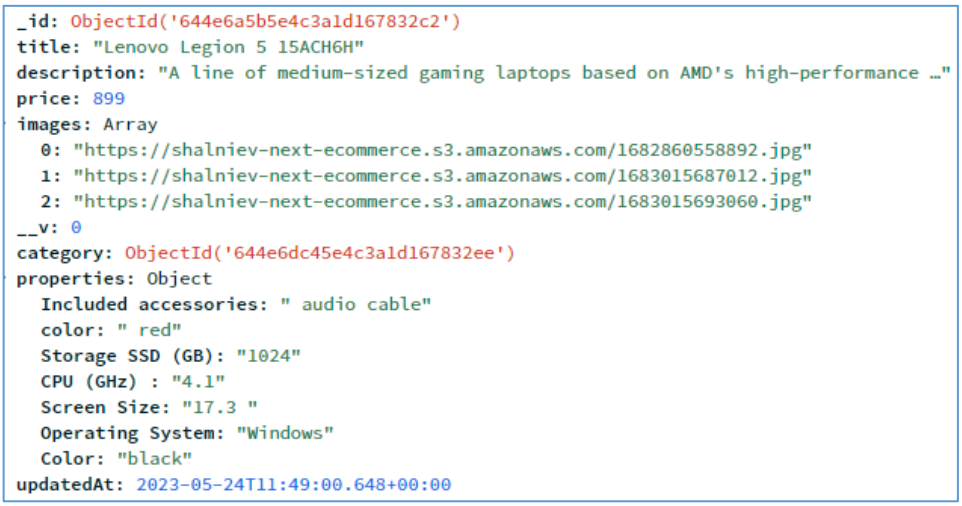

Рисунок 3.5 – Вигляд моделі Product у базі даних

Остання модель адміністративної панелі Setting (див. рис. 3.6) визначає схему для налаштувань вебзастосунку. В неї входять такі поля, як: ім'я та значення. Ця модель потрібна для того, щоб обирати найкращий продукт зі списку продуктів та встановлення плати за перевезення товару.

Рисунок 3.6 – Вигляд моделі Setting у базі даних

3.1.3 Реалізація кінцевих точок програми

API endpoints є основними точками взаємодії між адміністративною панеллю та серверною частиною вебзастосунку. Це ґрунтовні підсистеми, які забезпечують зв'язок між користувацьким інтерфейсом та базою даних, дозволяючи виконувати різні операції з даними, такі як отримання, оновлення, видалення та створення нових записів. В даному підрозділі розглядаються API endpoints (див. рис. 3.7), які використовуються для реалізації адміністративної панелі, а саме:

– admins використовується для управління даними адміністратора. Це включає можливості, такі як створення нового адміністратора, оновлення інформації про існуючого адміністратора, видалення адміністратора та отримання даних про адміністратора;

– categories дозволяє керувати категоріями продуктів. З його допомогою можна створювати нові категорії, оновлювати існуючі, видаляти категорії, а також отримувати список всіх доступних категорій;

– delete-images надає можливість видаляти зображення продуктів. Він використовується, коли продукт видаляється або коли зображення продукту оновлюється;

– orders забезпечує управління замовленнями. З його допомогою можна отримувати, оновлювати статус замовлення та видаляти замовлення;

– products використовується для управління даними про продукти. Це включає створення нових продуктів, оновлення даних про існуючі продукти, видалення продуктів та отримання даних про продукти;

– settings дозволяє керувати налаштуваннями вебзастосунку, популярними товарами, платою за доставку та перевезення, тощо;

– upload використовується для завантаження зображень продуктів. Він приймає файл зображення як вхідні дані та зберігає його на сервері для подальшого використання.

Отже, кожен API endpoint має свою конкретну роль і функції, які він виконує. Ці endpoints створені таким чином, щоб максимально спростити процес управління даними у вебзастосунку.

Лістинг коду (приклад оформлення кінцевої точки):

```
export default async function handle(req, res)
{
  await mongooseConnect();
  await isAdminRequest(req, res);
  if (req.method === "POST")
{
    const { email } = req.body;
    if (await Admin.findOne({ email }))
{
      res.status(400).json(
{
errorMessage: "This admin already exists." });
    } else {
      res.json(await Admin.create({ email }));
    }
  }
```

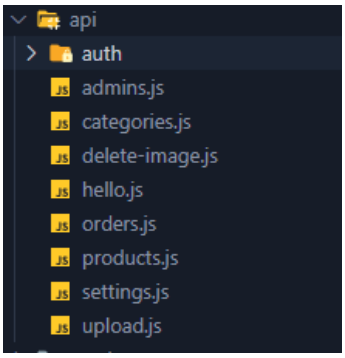

Рисунок 3.7 – Список кінцевих точок у проекті

Основні HTTP методи, що використовуються, включають GET, POST, PUT, PATCH та DELETE. Ось короткий огляд їх використання:

– GET метод використовується для отримання даних від сервера. Наприклад, можна використовувати GET для отримання інформації про конкретний продукт або список всіх продуктів;

– POST метод використовується для надсилання нових даних на сервер. Наприклад, при створенні нового продукту або нового замовлення використовується POST;

– PUT або PATCH методи використовуються для оновлення існуючих даних на сервері. PUT використовується, коли оновлюються всі дані ресурсу, тоді як PATCH використовується, коли оновлюється тільки частина даних;

– DELETE метод використовується для видалення існуючих даних з сервера. Наприклад, при видаленні продукту або замовлення використовується DELETE.

Всі ці методи використовуються разом для створення повного RESTful API, який дозволяє взаємодіяти з сервером в усіх необхідних аспектах керування даними.

3.1.4 Реалізація Google авторизації та ролі адміністратор

Застосування Google-аутентифікації не тільки полегшує процес входу для користувачів, але й забезпечує більший рівень безпеки для їх даних, оскільки вони користуються надійними механізмами захисту, розробленими Google. Це, в свою

чергу, допомагає покращити загальний досвід користувачів вебзастосунку та забезпечує підвищення їх довіри до сервісу.

Щоб це зробити, необхідно перейти до сервісу від компанії Google під назвою Google Cloud. У ньому створити зробити усі необхідні налаштування, починаючи від сторінки повноважень (див. рис. 3.8) та екрану згоди «Consent Screen» (див. рис. 3.9). Він відповідає за налаштування екрану згоди, який відображається користувачам під час запиту дозволу на доступ до їхніх даних. Цей екран згоди містить інформацію про те, які дані збирається зібрати та використовувати, хто буде мати до них доступ, і як довго ці дані будуть зберігатися.

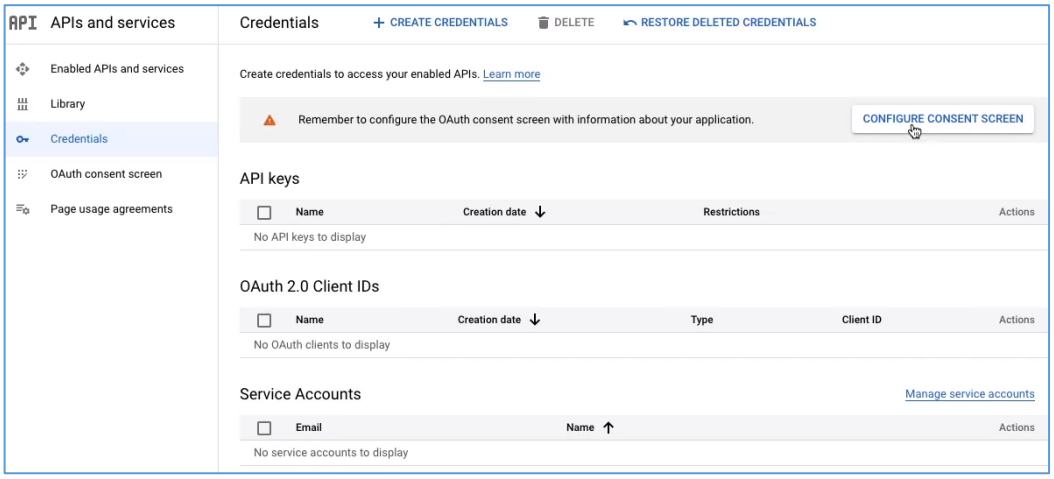

Рисунок 3.8 – Вкладка повноваження

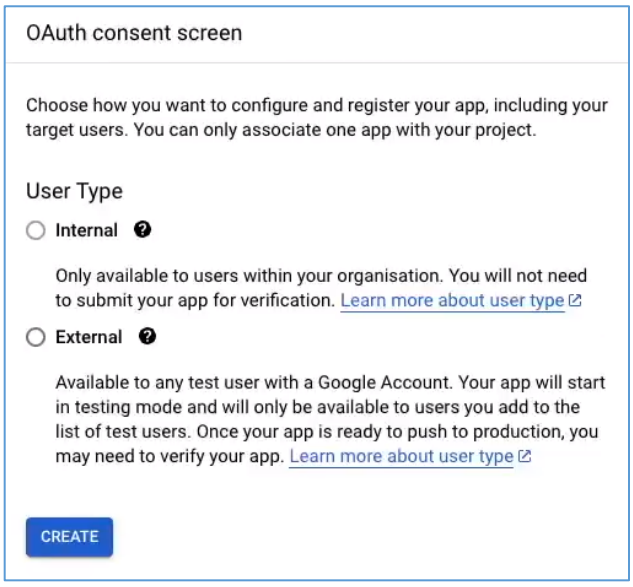

Рисунок 3.9 – Налаштування екрану згоди

Далі надаємо назву проекту та усі необхідні дані у вкладці створення проекту (див. рис. 3.10).

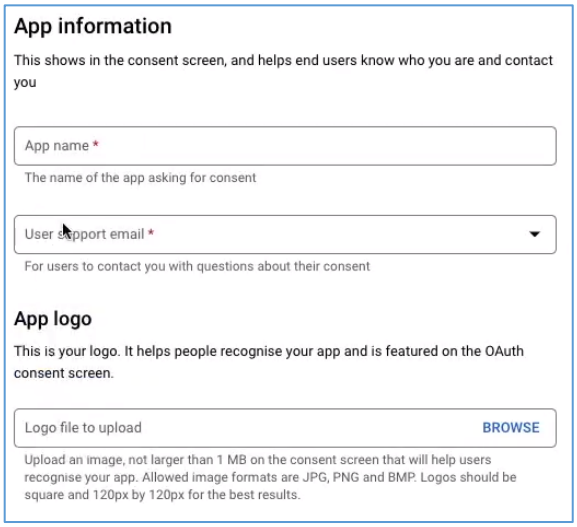

Рисунок 3.10 – Створення самого проекту

Після цього створимо клієнтський ідентифікатор, тобто обираємо тип застосунку та називаємо його. Сторінка Create OAuth client ID (див. рис. 3.11) використовується для створення OAuth-ідентифікаторів клієнта для взаємодії з API Google та іншими сервісами Google. OAuth - це протокол авторизації, який дозволяє користувачам контролювати доступ до своїх ресурсів в Інтернеті без використання пароля. Це означає, що користувачі можуть дозволити доступ до своїх облікових записів іншим додаткам без необхідності передавати свій пароль.

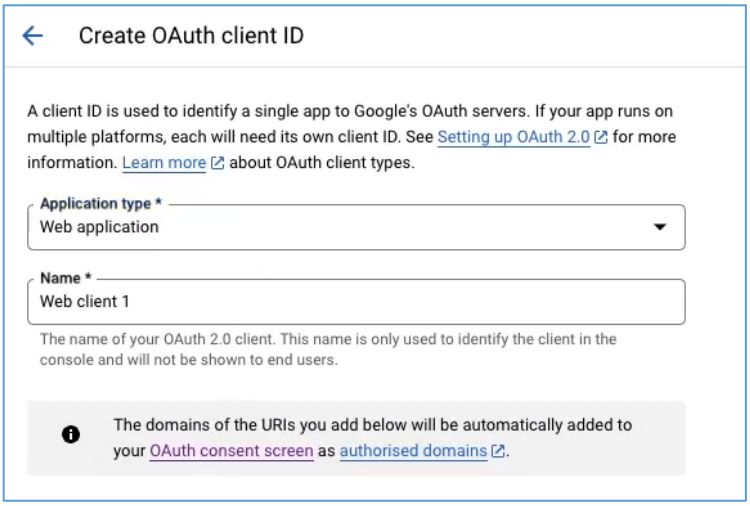

Рисунок 3.11 – Робота з OAuth-ідентифікатором клієнта

Після створення ідентифікатора, з'явиться вікно (див. рис. 3.12), в якому можна взяти необхідні для подальшої роботи «Client ID» та «Client secret».

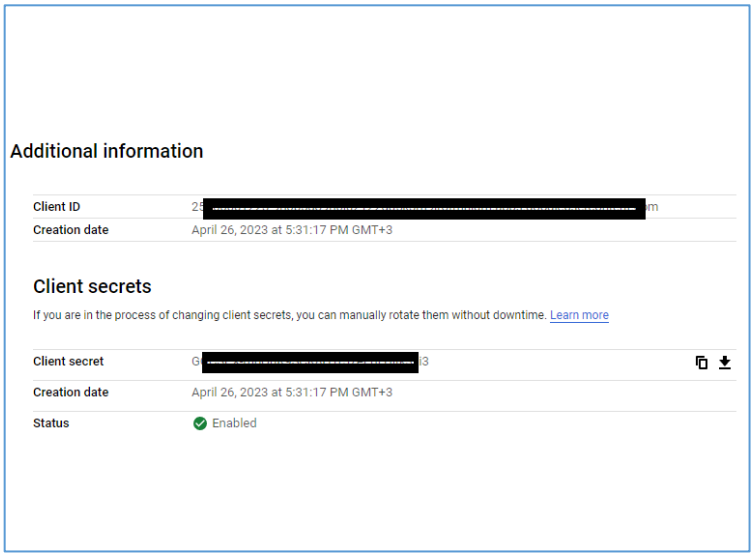

Рисунок 3.12 – Додаткова інформація (секретний ключ та ідентифікатор)

Далі створюємо відповідний файл у нашому проекті під назвою «.env», який буде зберігати усі секретні дані, ключі, тощо. А також відповідну сторінку-компонент, за допомогою якої буде здійснюватися вхід до нашої адміністративної панелі.

Лістинг коду ініціалізації провайдера Google:

```
providers: [
   GoogleProvider({
     clientId: process.env.GOOGLE_ID,
     clientSecret: process.env.GOOGLE_SECRET,
     authorizationUrl:

"https://accounts.google.com/o/oauth2/auth?prompt=consent",
     }),],
```

Властивості цього об'єкту включають:

– clientId: ідентифікатор клієнта, який був створений на платформі Google API Console;

– clientSecret: секретний ключ, який був згенерований разом з clientId;

– authorizationUrl: URL-адреса, на яку буде перенаправлено користувачів для авторизації через Google.

– process.env.GOOGLE_ID та process.env.GOOGLE_SECRET - це змінні середовища, які містять значення ідентифікатора клієнта та секретного ключа відповідно.

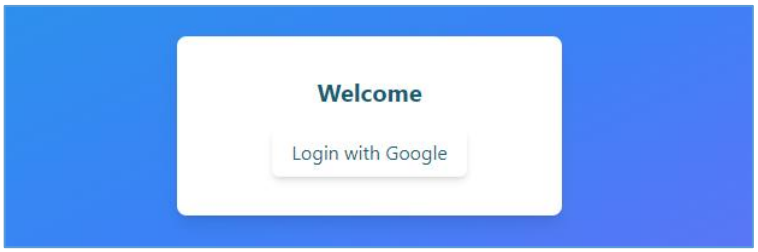

Рисунок 3.13 – Головне вікно логін-сторінки

При натисканні на кнопку «Login with Google» (див. рис. 3.13) користувачу буде запропоновано увійти за допомогою акаунту гугл (див. рис. 3.14).

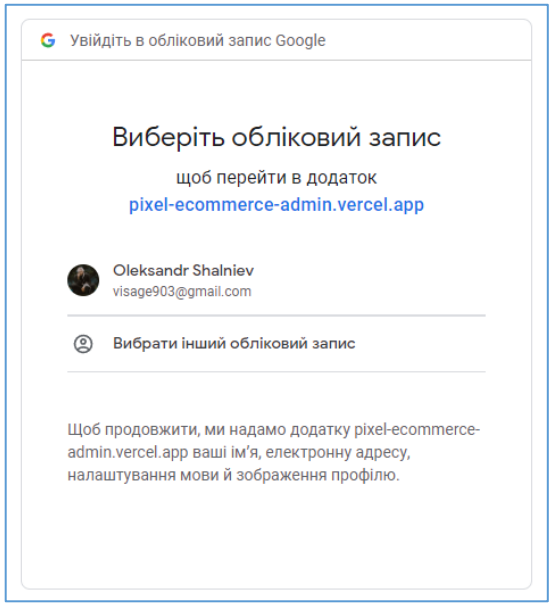

Рисунок 3.14 – Вибір акаунту гугл для входу

Для адміністративної панелі дозволити доступ будь-якому користувачеві недоцільно, оскільки це може призвести до несанкціонованої зміни, видалення або додавання даних на сайті. Тому було введено роль адміністратора.

Адміністратори мають ексклюзивний доступ до адміністративної панелі, що надає їм можливість керувати даними вебзастосунку відповідно до своїх потреб і

завдань. Це забезпечує безпеку даних та гарантує, що лише відповідальні особи матимуть можливість впливати на роботу сайту та його зміст.

Лістинг коду з перевіркою чи є користувач адміністратором:

```
  async function isAdminEmail(email) {
  try {
    mongooseConnect();
    return email === MAIN_ADMIN_EMAIL || !!(await
Admin.findOne({ email }));
  } catch (err)
{
    console.error(`Error occurred while checking admin email:
${err}`);
    return false;}
}
```

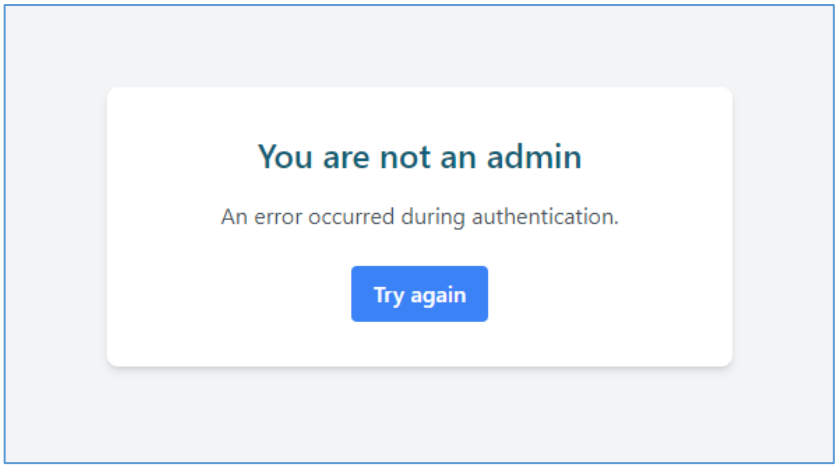

Рисунок 3.15 – Помилка аутентифікації

Якщо користувач є адміністратором сайту – він зможе увійти до системи, в іншому випадку побачить сторінку з повідомленням та забороною у доступі (див. рис. 3.15).

3.1.5 Використання AWS для збереження та доставки зображень

У процесі розробки вебзастосунку для продажу комп'ютерної техніки та комплектуючих, важливим аспектом є забезпечення швидкого та надійного збереження та доступу до картинок, що відображають товари. Для цієї мети було

вирішено використовувати послуги Amazon Web Services (AWS) (див. рис. 3.16), які надають надійні, масштабовані та гнучкі рішення для збереження даних.

Однією з ключових послуг AWS, яка використовується у даному проекті, є Amazon S3 (Simple Storage Service). Amazon S3 – це об'єктне сховище, яке дозволяє зберігати та отримувати будь-яку кількість даних у будь-який час з будь-якої точки Інтернету. Воно гарантує надійність, доступність та масштабованість на високому рівні.

Використання Amazon S3 для зберігання картинок товарів дає ряд переваг:

– надійність: AWS забезпечує високий рівень надійності та безпеки даних, зберігаючи копії файлів у декількох географічно розосереджених центрах обробки даних;

– швидкість завантаження: завдяки оптимізації роботи з мережею та географічному розміщенню серверів, AWS пропонує швидке завантаження файлів для користувачів з різних куточків світу.

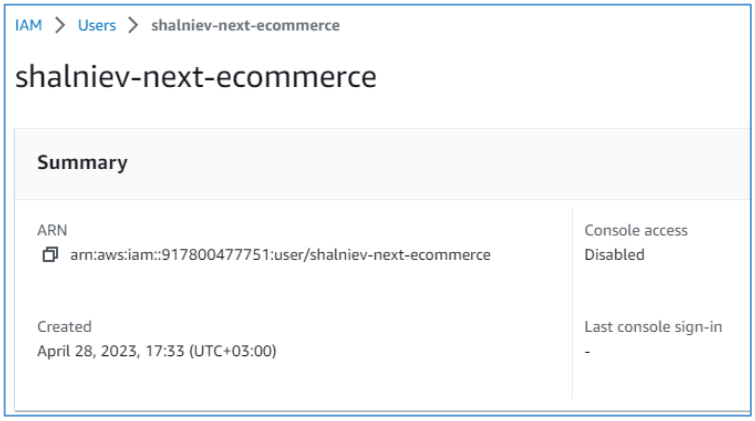

Рисунок 3.16 – Використання хмарного сервісу. Послуга S3

Також було зроблено налаштування Policy у AWS S3 (див. рис. 3.17). Вони відносяться до набору правил та дозволів, які контролюють доступ до ресурсів Amazon S3. Вони допомагають вам управляти, хто може отримати доступ до ваших S3 бакетів та об'єктів, а також визначають, які операції з ними можуть виконуватись.

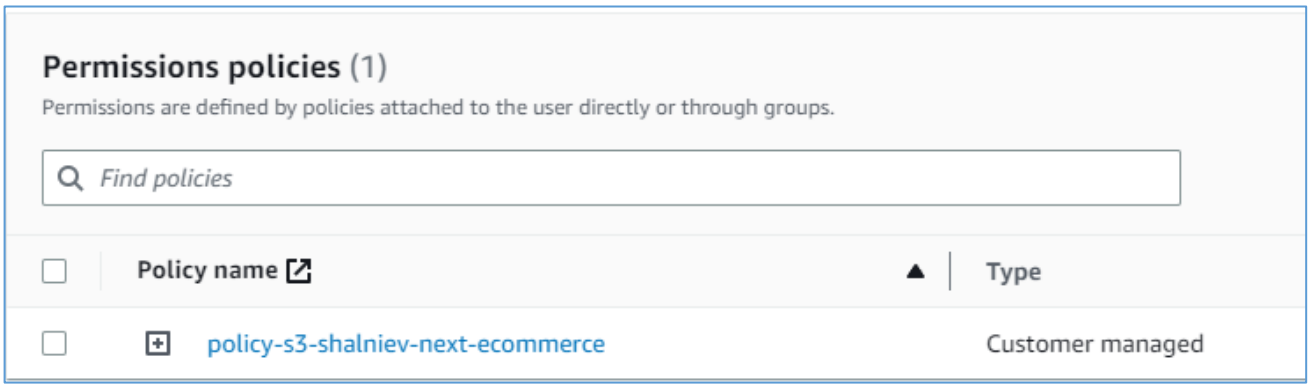

Рисунок 3.17 – Використання хмарного сервісу. Послуга S3

Лістинг коду імпорту модулів та створення клієнта S3:

```
function createS3Client() {
  return new S3Client({
    region: "eu-north-1",
    credentials: {
      accessKeyId: process.env.S3_ACCESS_KEY,
      secretAccessKey: process.env.S3_SECRET_ACCESS_KEY,
    },});}
```

Функція createS3Client створює новий екземпляр клієнта S3, вказуючи регіон eu-north-1 та передаючи об'єкт з обліковими даними (accessKeyId та secretAccessKey), які отримуються з змінних середовища.

Функція uploadFilesToS3 відповідає за завантаження файлів на S3. Вона використовує s3Client та метод send для відправки команди PutObjectCommand, яка вказує інформацію про завантаження файлу, таку як назва бакета, ключ об'єкта, тіло файлу, політику доступу та тип контенту.

3.1.6 Реалізація frontend частини

Frontend частина вебзастосунку відповідає за те, як користувач бачить та взаємодіє з ним. Ця частина зазвичай включає в себе візуалізацію даних, які надає сервер, та обробку користувацького введення. В нашому випадку, frontend частина була розроблена за допомогою React.js, сучасної JavaScript бібліотеки для побудови користувацьких інтерфейсів.

Основою React є компоненти, які можна вважати незалежними, повторно використовуваними блоками, з яких будується вебзастосунок. У проекті основні компоненти знаходяться в директорії components. Кожен компонент відповідає за конкретний аспект інтерфейсу, такий як кнопка, форма або навігаційне меню. Вони можуть бути використані в одному або декількох місцях на веб-сайті, що забезпечує консистентність візуального дизайну та спрощує розвиток та обслуговування коду.

Директорія pages містить компоненти (див. рис. 3.18), що представляють цілі веб-сторінки. Кожен файл в цій директорії відповідає за візуалізацію сторінки інтернет-магазину. Компоненти сторінок зазвичай включають в себе багато менш специфічних компонентів з директорії components, організуючи їх у вищий рівень структури для формування цілісного користувацького досвіду.

Крім того, додаткові ресурси та засоби, такі як стилі, зображення та утиліти, знаходяться в інших директоріях проекту. Вони допомагають підтримувати чистоту та організованість коду, дозволяючи розробникам легко знаходити та оновлювати ресурси.

Разом, ці ресурси і компоненти створюють користувацький інтерфейс вебзастосунку. Кожен компонент або ресурс відповідає за певний аспект візуалізації або функціональності, від навігації та відображення даних до обробки взаємодії користувача.

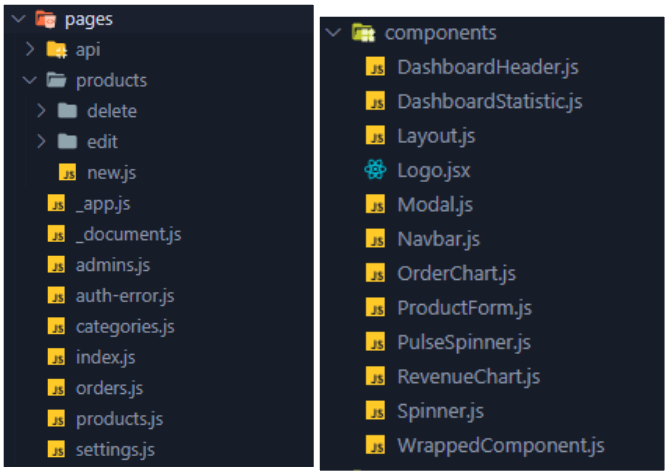

Рисунок 3.18 – Вміст директорій pages та components

3.2 Реалізація технічної частини Інтернет-магазину

Інтернет-магазин є фронтальною частиною цього вебзастосунку, який надає користувачам доступ до широкого спектру продуктів для покупки. В підрозділі подається детальний опис процесу реалізації Інтернет-магазину з використанням вибраних технологій та сервісів.

3.2.1 Реалізація підключення проекту, схем і кінцевих точко до БД

Оскільки інтернет-магазин та адміністративна панель використовують одну ту саму базу даних, підключення до неї здійснюється за допомогою однакових файлів mongodb.js і mongoose.js, які ми описали раніше. Це забезпечує єдність та консистентність роботи з базою даних.

В інтернет-магазині додатково було створено ще декілька моделей, що використовуються для обробки та зберігання даних специфічних для цієї частини проекту.

Модель Address (див. рис. 3.19) зберігає інформацію про адреси користувачів для доставки товарів. Модель містить поля, які описують електронну адресу користувача, ім'я, електронну пошту, країну, місто, поштовий індекс, та саму адресу.

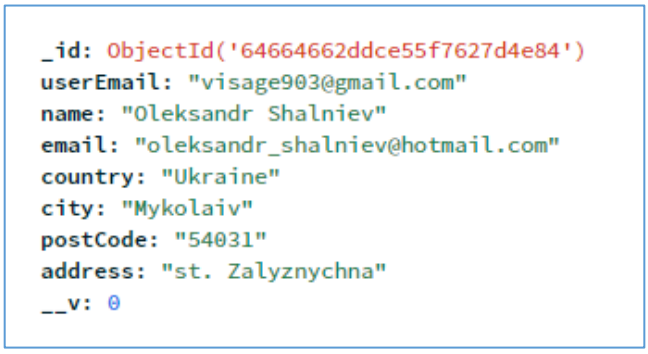

Рисунок 3.19 – Вигляд моделі Address у базі даних

Модель Review (див. рис. 3.20) зберігає відгуки користувачів на товари. Вона включає поля для імені користувача, опису відгуку, ідентифікатора товару, та кількості зірок (рейтингу).

Рисунок 3.20 – Вигляд моделі Review у базі даних

Модель WishlistedProduct (див. рис. 3.21) слугує для зберігання товарів, що були додані користувачами до їхнього списку бажань. Модель містить поля для електронної адреси користувача та ідентифікатора товару.

Рисунок 3.21 – Вигляд моделі WishlistedProduct у базі даних

Всі ці моделі, подібно до тих, що були описані раніше, використовують схеми Mongoose для визначення структури даних і забезпечення валідації на рівні бази даних.

Реалізація кінцевих точок для Інтернет-магазину слідує тому ж принципу, що й для адміністративної панелі. Тобто вони використовуються для обробки HTTP-запитів від клієнтської частини застосунку та взаємодії з базою даних.

В Інтернет-магазині були створені додаткові кінцеві точки, за допомогою яких можна взаємодіяти з базою даних та які відповідають за конкретні функції магазину:

– cart обробляє дії, пов'язані з кошиком користувача, включаючи додавання та видалення товарів.

– reviews забезпечує взаємодію з відгуками користувачів. Через цю кінцеву точку можна створювати нові відгуки, видаляти існуючі, а також відображати відгуки на сторінці товару.

– wishlist використовується для керування списком бажань користувача, включаючи додавання та видалення товарів.

– address дозволяє користувачам зберігати та оновлювати адресу для доставки товарів.

– checkout використовується для обробки процесу оформлення замовлення. Через цю кінцеву точку користувач може підтвердити замовлення та перейти до оплати.

– webhook використовується для обробки вебхуків від системи оплати. Коли оплата успішно здійснена, система оплати відправляє запит до цього вебхука, який потім оновлює стан замовлення.

Використання кінцевих точок для обробки різноманітних дій дозволяє ефективно організовувати логіку застосунку і спрощує розробку та підтримку коду.

3.2.2 Реалізація Google авторизації для Інтернет-магазину

Одним з ключових етапів розробки будь-якого вебзастосунку є імплементація системи авторизації. В контексті цього проекту, така ж сама система Google авторизації, що була впроваджена для адміністративної панелі, також використовується для Інтернет-магазину. Це дозволяє користувачам легко входити в систему, використовуючи свій обліковий запис Google, що сприяє зручності та безпеки.

При використанні Google авторизації в Інтернет-магазині, користувачам надається зручний та безпечний спосіб входу в систему, що дозволяє швидко та надійно авторизуватися. Вони можуть використовувати свої існуючі облікові записи Google, замість того, щоб створювати новий обліковий запис для магазину. Це не тільки зручно, але і забезпечує додатковий рівень безпеки, оскільки Google використовує надійні методи захисту для облікових записів своїх користувачів.

Для того, щоб підключити авторизацію через Google до проекту, необхідно зробити ті самі дії, які робилися для адміністративної панелі, починаючи з вкладки повноважень (див. рис. 3.22).

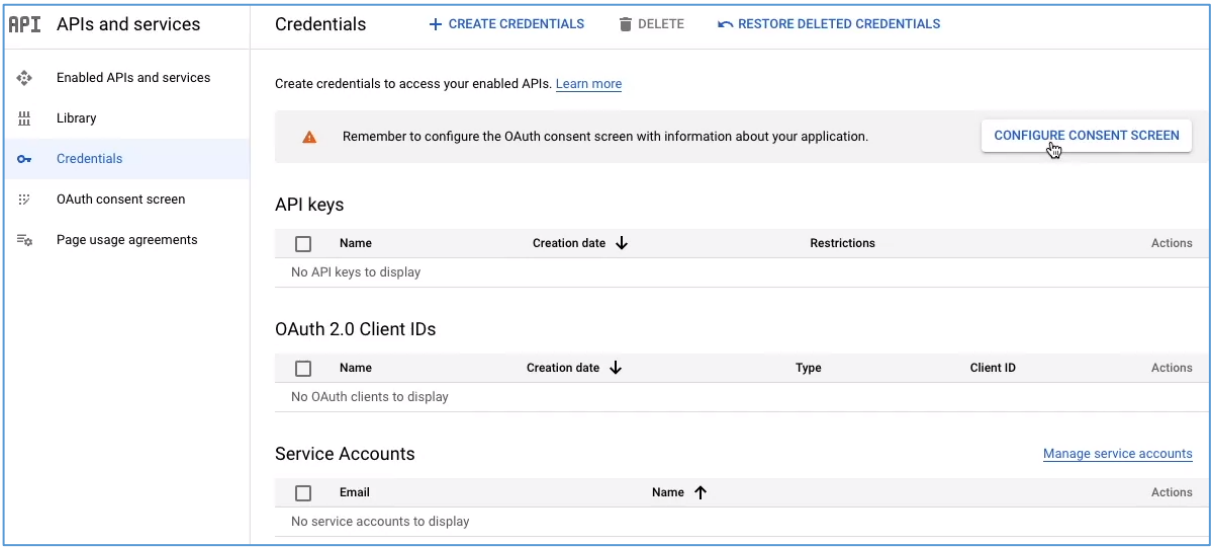

Рисунок 3.22 – Вкладка повноваження

Після чого переходимо до вікна екрани згоди (див. рис. 3.23) та обираємо необхідний користувацький тип.

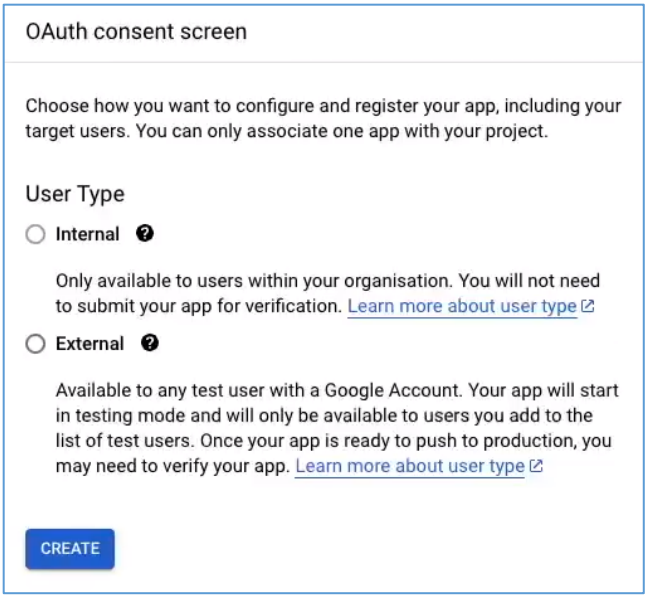

Рисунок 3.23 – Налаштування екрану згоди

Після цього знову створимо клієнтський ідентифікатор, але вже для Інтернет-магазину. Обираємо тип застосунку як вебзастосунок, та ім'я застосунку відповідно (див. рис. 3.24).

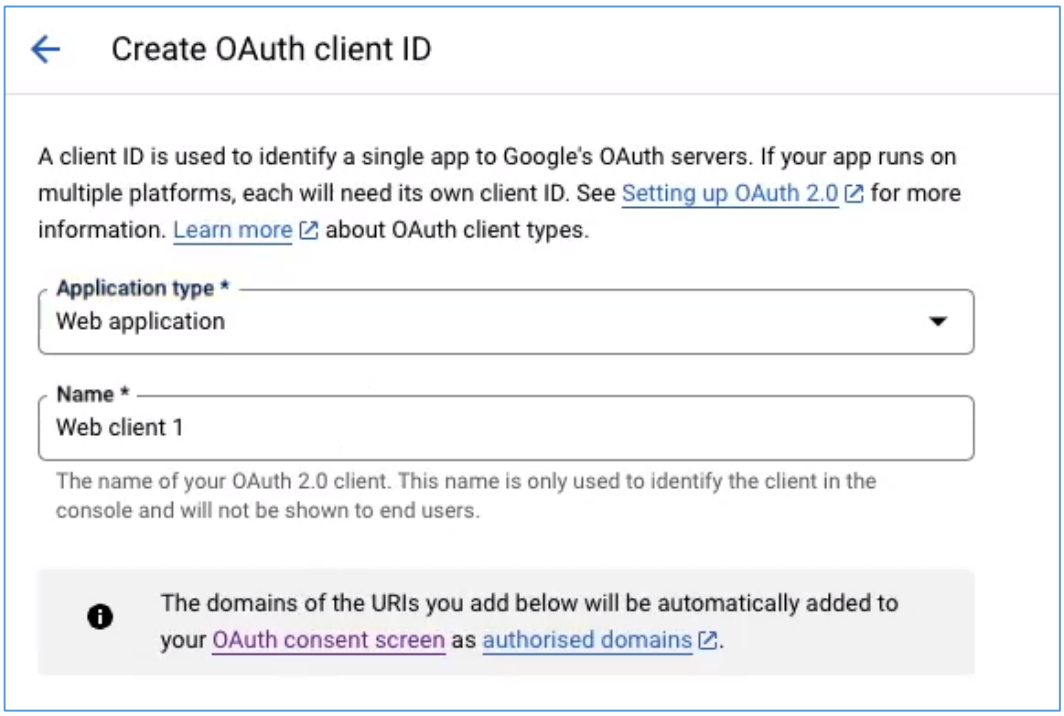

Рисунок 3.24 – Робота з OAuth-ідентифікатором клієнта

Як тільки ідентифікатор буде створений, з'явиться вікно (див. рис. 3.25), в якому можна взяти необхідні для подальшої роботи «Client ID» та «Client secret».

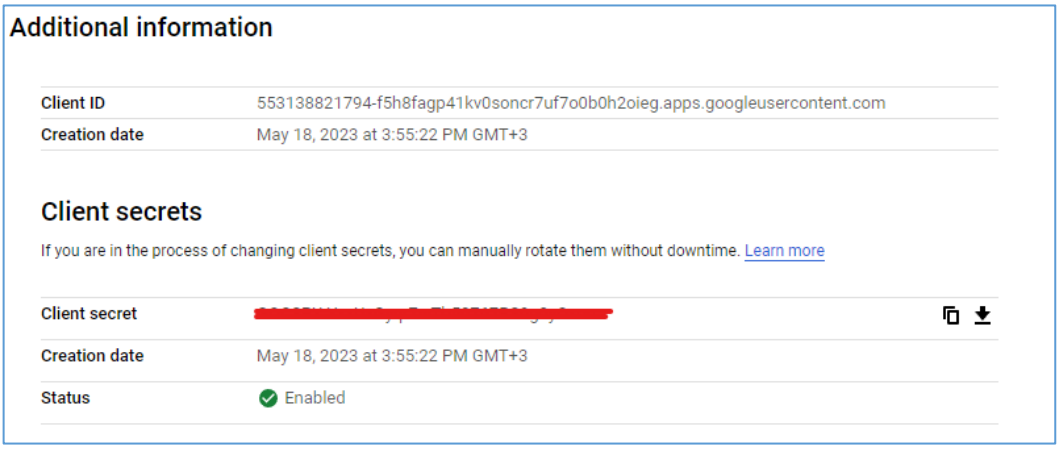

Рисунок 3.25 – Додаткова інформація (секретний ключ та ідентифікатор)

Ці дані потрібно зберегти в одному з файлів проекту, що відокремлює конфігураційних параметрів від вихідного коду програми.

3.2.3 Реалізація frontend частини та платіжного методу Stripe

Сучасний Інтернет-магазин – це не тільки продуктивний і функціональний інструмент для продажу товарів, але й важлива частина бренду, яка допомагає

залучити і утримати клієнтів. Отже, під час розробки frontend для Інтернет-магазину особливу увагу було приділено створенню естетичного і інтуїтивно зрозумілого дизайну, що спрощує процес покупок для кінцевих користувачів.

Так як цей проект зорієнтований на надання користувачам зручного та ефективного середовища для шопінгу, стильовий зовнішній вигляд, зручне розташування елементів на сторінці, гладкі анімації та високу продуктивність було вважено важливими факторами під час розробки.

Розробка фронтенду для Інтернет-магазину включає в себе створення структури сайту, дизайну інтерфейсу користувача, а також програмування клієнтської частини застосунку, що включає в себе обробку користувацького вводу, відображення даних від сервера та взаємодію з API (див. рис. 3.26).

Як і в адміністративній панелі, React.js було використано як основний фреймворк для розробки клієнтської частини Інтернет-магазину. Однак, порівняно з адміністративною панеллю, тут було приділено значно більше уваги деталям дизайну, анімацій та взаємодії з користувачем, що важливо для створення вражаючого досвіду користувача.

Останнім, але не менш важливим, є той факт, що при розробці фронтенду було приділено особливу увагу продуктивності та оптимізації. Це включає в себе використання ефективних методів рендерингу, мінімізацію завантаження ресурсів та використання різних технологій і практик для підвищення швидкості завантаження та покращення загальної продуктивності сайту.

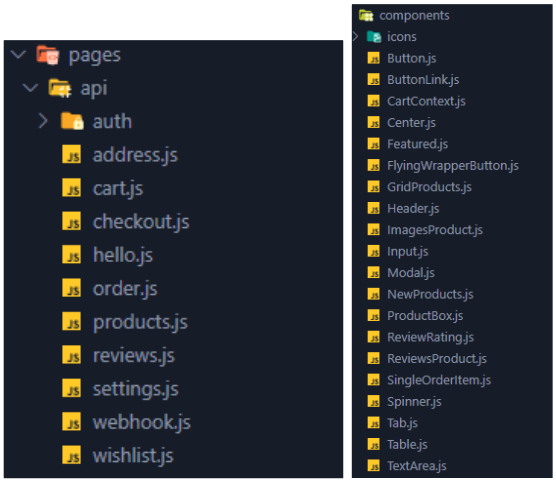

Рисунок 3.26 – Вміст директорій pages та components

Стосовно методу Stripe, то він є одним з провідних рішень для онлайн-платежів, яке надає веброзробникам потужний і гнучкий інструмент для обробки і управління платежем. Stripe надає API, що дозволяє інтегрувати платіжні сервіси безпосередньо в вебзастосунки, що підвищує безпеку і полегшує процес роботи з платежем.

Інтеграція Stripe в Інтернет-магазин включає в себе декілька основних кроків. По-перше, для використання Stripe потрібно створити аккаунт на сайті Stripe та отримати необхідні ключі API для взаємодії з сервісом. Далі, ці ключі API (див. рис. 3.27) використовуються для ініціалізації Stripe на стороні сервера та клієнта.

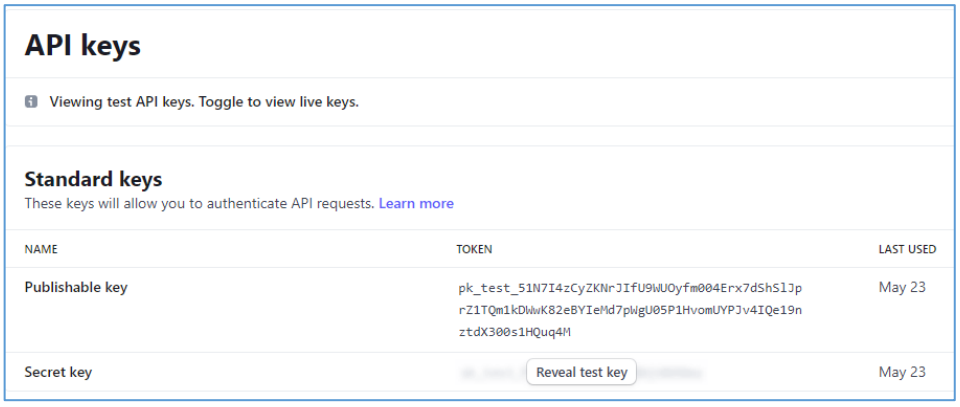

Рисунок 3.27 – Сторінка API ключів для розробників

На стороні сервера, Stripe використовується для створення інстанцій оплати, які потім передаються на клієнтську сторону для виконання платежу. На стороні клієнта, Stripe використовується для обробки даних платіжної картки та виконання платежу (див. рис. 3.28).

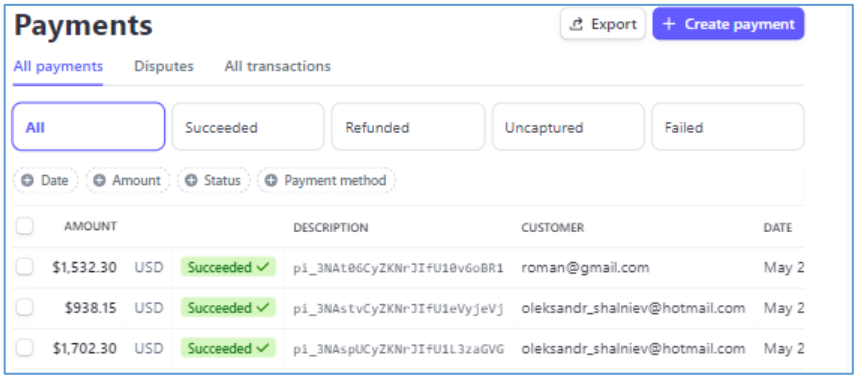

Рисунок 3.28 – Сторінка інформації про платежі

Після успішного завершення платежу, результат перевіряється на сервері через Stripe Webhook (див. рис. 3.29), який відправляє повідомлення про статус оплати. Якщо оплата успішна, інформація про покупку та платіж записується в базу даних і відправляється до адміністративної панелі.

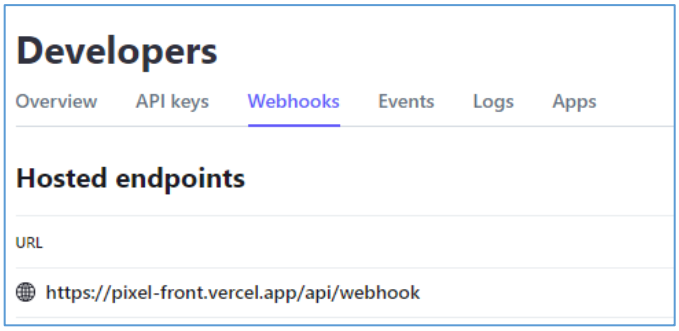

Рисунок 3.29 – Створення власної кінцевої точки

Важливо зауважити, що при використанні Stripe, всі платіжні дані обробляються та зберігаються на серверах Stripe, що допомагає забезпечити високий рівень безпеки платежів.

У процесі інтеграції Stripe, магазину також можна використовувати режим "test", який дозволяє перевірити процес платежу без реальних транзакцій.

Окрім цього було створено купони, з використанням яких користувачі можуть зекономити значну кількість грошей при покупці. Вони знаходяться у вкладці «Products» та розділ «Coupons» (див. рис. 3.30).

Інтеграція Stripe є важливим елементом реалізації Інтернет-магазину, оскільки вона дозволяє забезпечити безпеку і ефективність платежів, що є важливим аспектом успішного онлайн-бізнесу.

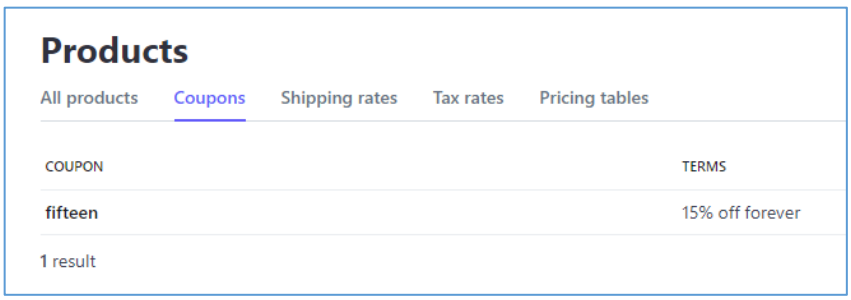

Рисунок 3.30 – Додавання купонів та знижок

Висновки до третього розділу

В даному розділі було виконано значну роботу з реалізації технічної сторони обох частин проекту: Інтернет-магазину та адміністративної панелі.

Було проведено детальну реалізацію backend частин, зокрема, було створено API кінцеві точки для взаємодії з базою даних та обробки запитів від користувачів. Важливою частиною реалізації backend було підключення до MongoDB та створення відповідних схем і моделей для взаємодії з БД.

З іншого боку, було здійснено реалізацію frontend частини, причому для Інтернет-магазину було приділено більше уваги дизайну, зручності користувачів і використанню анімації для покращення взаємодії користувачів з сайтом.

Велику увагу приділено взаємодії адміністративної панелі з Інтернет-магазином, контролювання та моніторинг тієї чи іншої частини магазину через адміністративну панель, а також реалізації авторизації через Google для обох проектів та впровадження платіжного методу Stripe

Загалом, реалізація технічної сторони обох проектів демонструє високий рівень використання сучасних технологій і практик розробки, що в свою чергу забезпечує надійність, продуктивність та зручність використання Інтернет-магазину і адміністративної панелі.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА КЕРІВНИЦТВО КОРИСТУВАЧА

Цей розділ детально описує програмну реалізацію та процес створення керівництва користувача для адміністративної панелі інтернет-магазину. Передбачено докладний огляд функціональних можливостей адміністративної панелі, а також інструкції для користувачів, щоб оптимізувати використання цього інструменту.

4.1 Керівництво користувача для адміністративної панелі

Адміністративна панель є важливим інструментом для управління Інтернет-магазином. Вона містить широкий спектр функцій, які допомагають керувати замовленнями, продуктами, категоріями продуктів, та адміністраторами магазину. Цей підрозділ надає детальне керівництво по користуванню адміністративною панеллю.

4.1.1 Авторизація і головна сторінка з аналітикою

Якщо користувач хоче скористатися адміністративною панеллю, йому потрібно пройти авторизацію. Для авторизації йому буде доступна можливість увійти з Google (див. рис. 4.1).

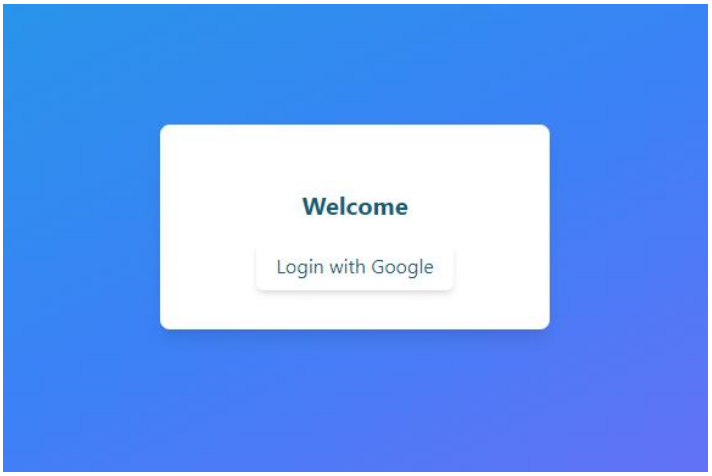

Рисунок 4.1 – Сторінка авторизації

Після натискання на кнопку, користувачеві буде запропоновано обрати один з його гугл аккаунтів (див. рис. 4.2).

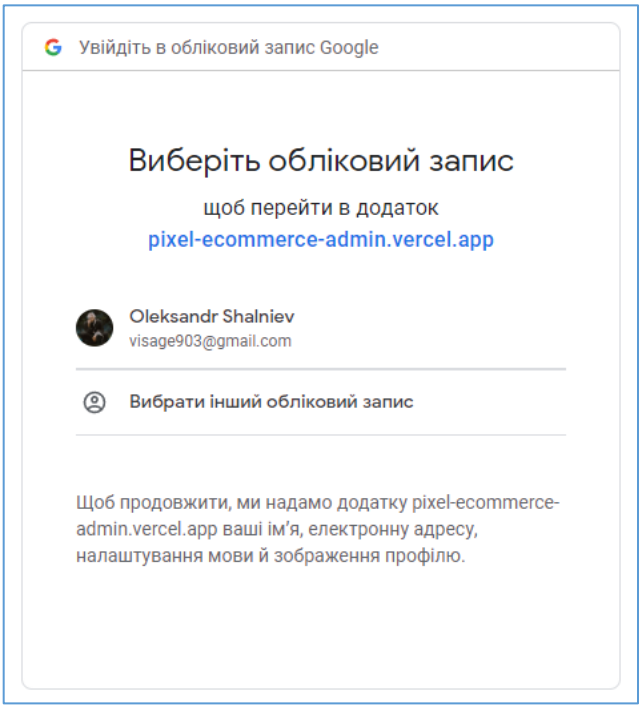

Рисунок 4.2 – Вибір аккаунту Google

Якщо цей аккаунт має права адміністратора, то він потрапить на головну сторінку з аналітикою, в іншому випадку отримає повідомлення про те, що вхід до сайту заборонено через те, що він не є адміністратором (див. рис. 4.3).

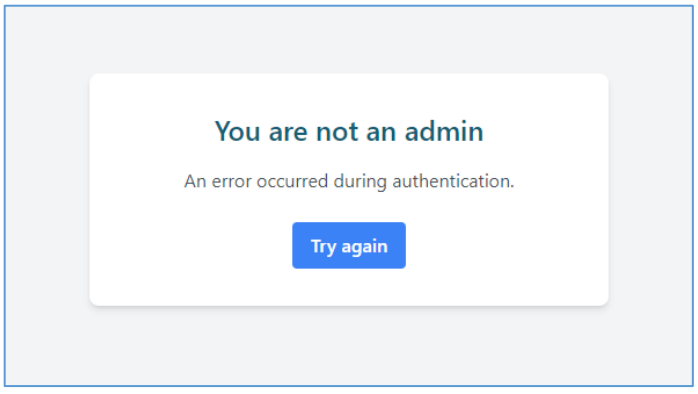

Рисунок 4.3 – Помилка аутентифікації (не адміністратор)

Перше, що побачить користувач, після того, як особистість було підтверджено та йому вдалось увійти до адміністративної панелі – це головна сторінка застосунку, вона ж секція аналітики. Секція аналітики є місцем, де адміністратор може одразу отримати ключові дані про продуктивність магазину.

Для зручності візуалізації дані представлені у вигляді графіків (див. рис. 4.4 та рис. 4.5).

Рисунок 4.4 – Головна секція аналітики (частина 1)

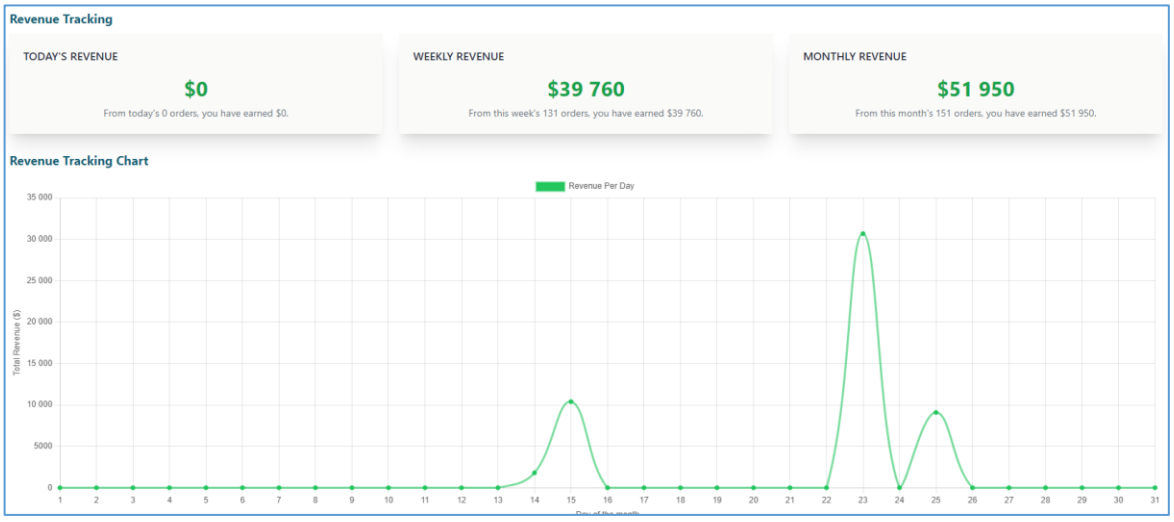

Рисунок 4.5 – Головна секція аналітики (частина 2)

На цій сторінці відображені такі дані: кількість товарів, придбаних протягом останнього дня/тижня/місяця, та доходи за цей період. Також, як було вже зазначено, побудовані графіки, що відображають певну кількість замовлених товарів у певний день протягом місяця, а також графік з доходами.

Крім того, у верхньому правому куті цієї вкладки можна побачити зображення та ім'я користувача, що беруться з його Google аккаунту. Ім'я також дублюється у лівій частині екрану.

4.1.2 Сторінки управління замовленнями та продуктами

Секція управління замовленнями становить основу роботи адміністративної панелі (див. рис. 4.6). Вона забезпечує адміністратору доступ до всіх замовлень, що надійшли в Інтернет-магазин. Завдяки деталізації, адміністратор має можливість бачити всі необхідні дані: список продуктів у замовленні, інформацію про користувача (ім'я, адресу, пошту та інші дані), статус оплати та дату і час замовлення. Це важливо для ефективного управління замовленнями, оцінки та прогнозування продажів, а також для моніторингу та вирішення будь-яких можливих проблем, пов'язаних зі замовленнями.

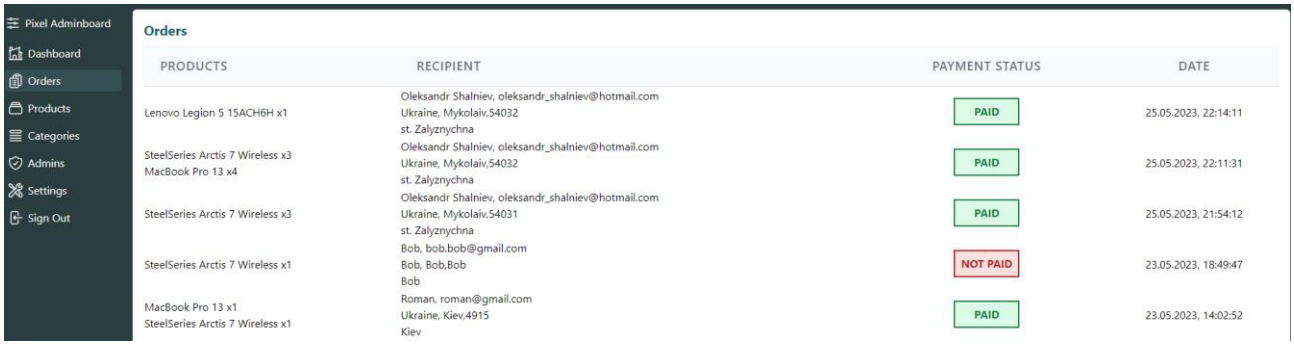

Рисунок 4.6 – Секція управління замовленнями

Також слід зазначити, що до аналітичних даних прибутку заносяться лише ті замовлення, що мають статус «PAID».

Секція управління продуктами надає адміністратору повний контроль над товарами, що продаються в Інтернет-магазині (див. рис. 4.7).

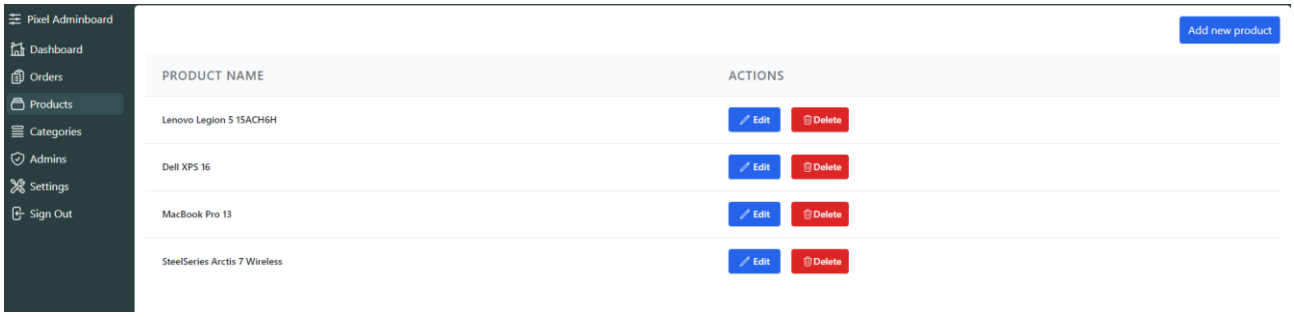

Рисунок 4.7 – Секція зі списком продуктів

За допомогою додавання нових продуктів, адміністратор може додати новий продукт, вказавши всю необхідну інформацію, включаючи назву, опис, ціну, зображення і категорію продукту (див. рис. 4.8).

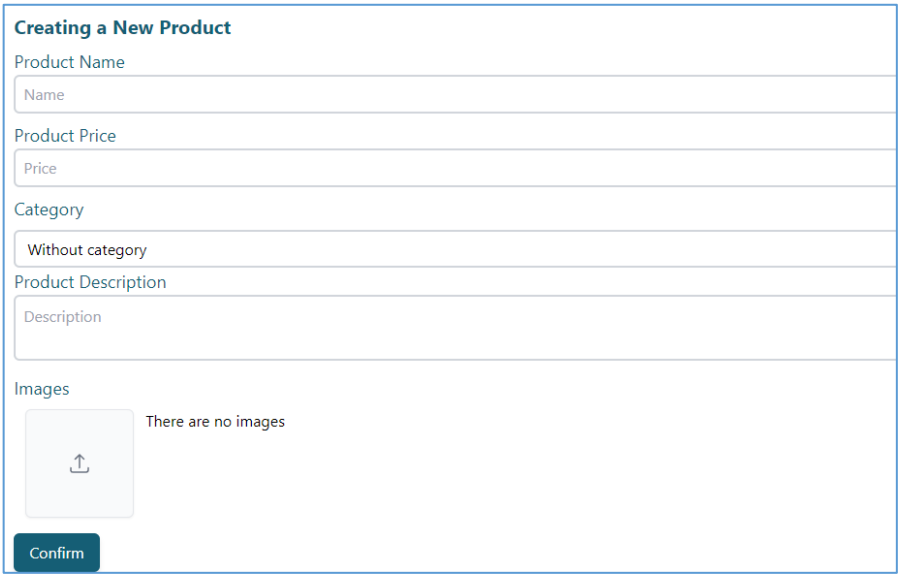

Рисунок 4.8 – Секція з додаванням нового продукту

Окрім цього, в нього є доступ переглядати і редагувати вже існуючі продукти (див. рис. 4.9). Всі продукти, які вже продаються, відображаються в цій секції. Адміністратор може вибрати будь-який продукт для перегляду або редагування деталей.

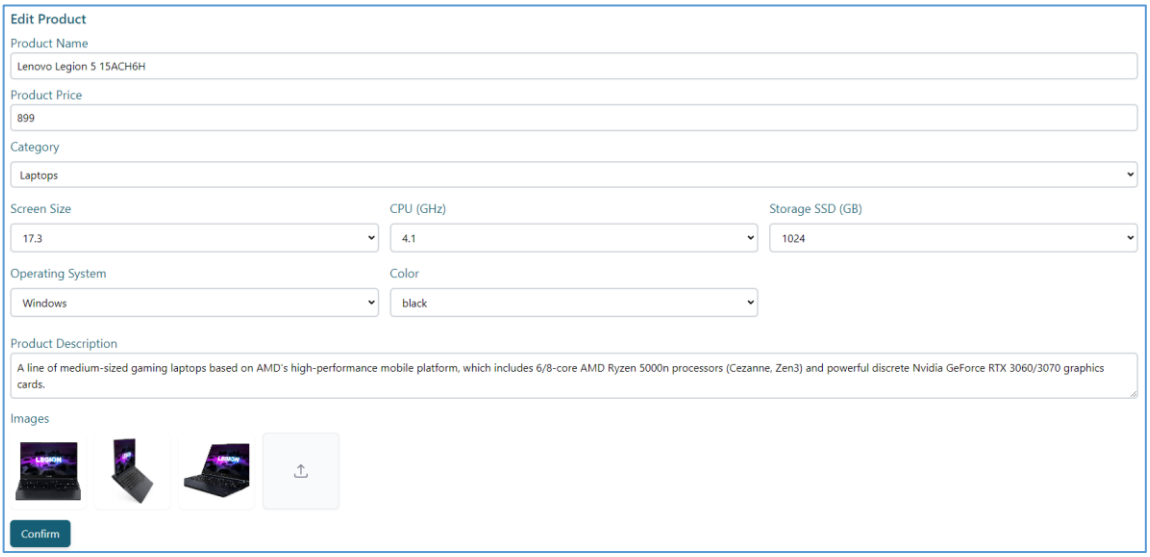

Рисунок 4.9 – Секція з редагуванням продукту

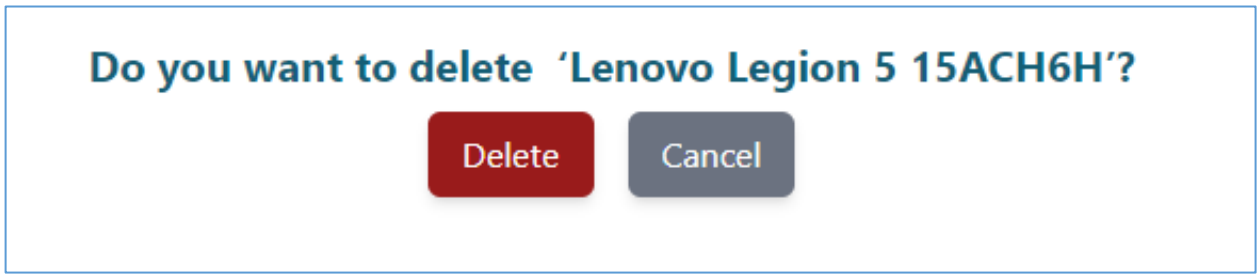

Рисунок 4.10 – Сторінка видалення продуктів

Також додана необхідна можливість видаленням продуктів. Адміністратор може видалити продукт, наприклад якщо цей товар більше не буде продавати або по якійсь іншій причині (див. рис. 4.10).

4.1.3 Сторінки управління категоріями та адміністраторами

Сторінка призначена для зберігання категорій товарів, що дозволяє в майбутньому впровадити фільтрацію веб-магазину за окремими категоріями. Сама сторінка включає таблицю з категоріями, які розділяються на основні та батьківські. Батьківські категорії – це ті, що можуть містити підкатегорії, наприклад, категорія телефонів, яка включає безліч брендів (див. рис. 4.11).

Сторінка також містить поле для назви категорії, можливість вибору належності до вже існуючої категорії або встановлення як батьківської, кнопки для додавання властивостей та кнопки збереження.

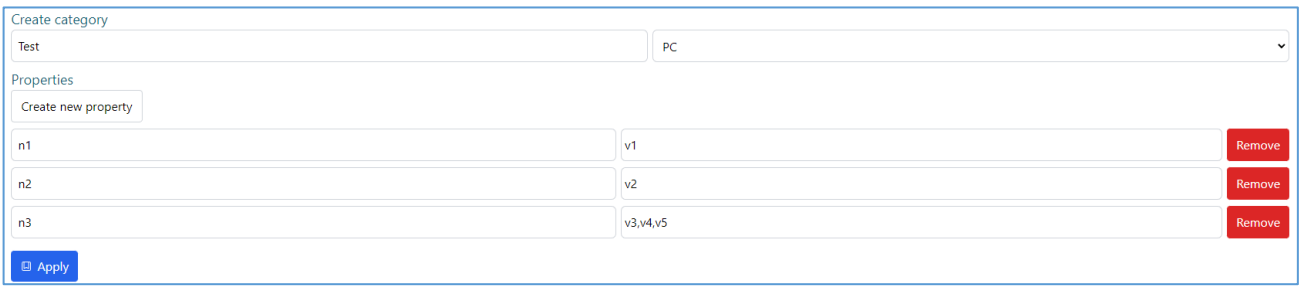

Рисунок 4.11 – Створення нової категорії та її властивостей

Як і на минулій сторінці з товарами, адміністратор має можливість редагувати (див. рис. 4.12) та видаляти категорії (див. рис. 4.13).

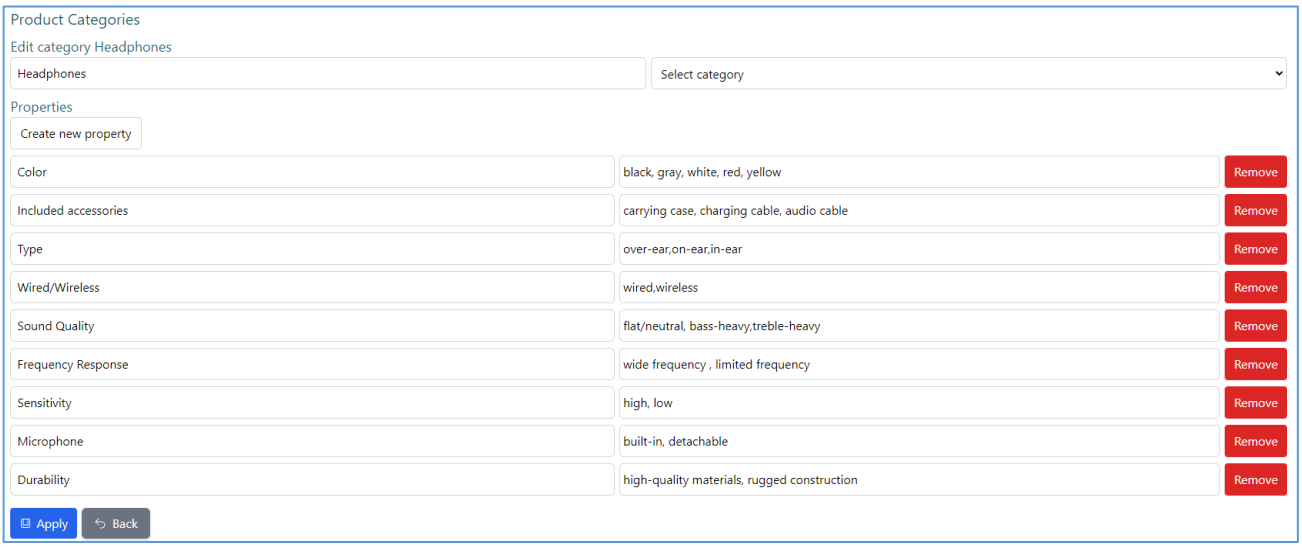

Рисунок 4.12 – Редагування обраної категорії

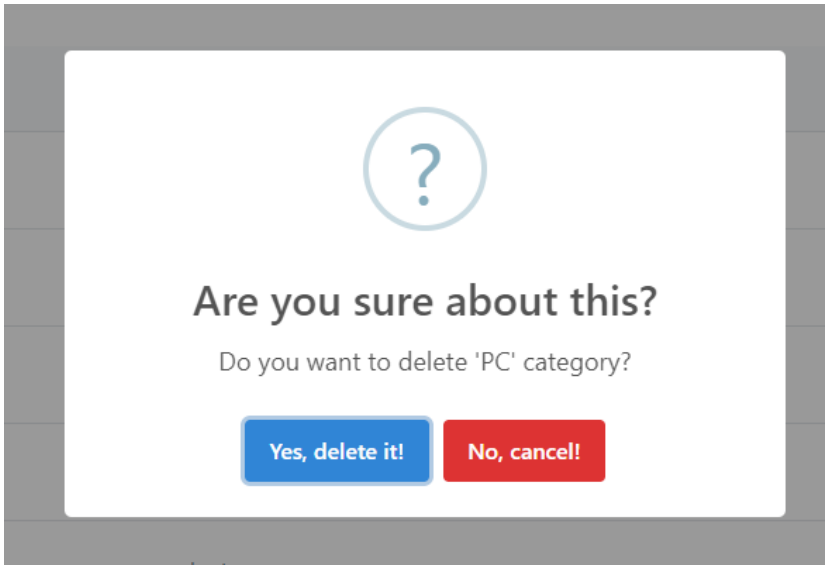

Рисунок 4.13 – Видалення обраної категорії

Управління адміністраторами є критично важливим для безперебійної роботи та захисту Інтернет-магазину. Ця секція адміністративної панелі надає можливість контролювати доступ до різних рівнів управління магазином (див. рис. 4.14).

Часто в одному Інтернет-магазині може бути декілька адміністраторів, кожен з яких відповідає за свою ділянку роботи - товари, замовлення, аналітику, технічну підтримку тощо. Управління адміністраторами дозволяє додавати нових користувачів до команди адміністраторів (див. рис. 4.15), а також, при необхідності, виділяти існуючих (див. рис. 4.16 та рис. 4.17).

Це надзвичайно важливо для забезпечення безпеки, ефективності та прозорості роботи магазину, адже кожний адміністратор може у майбутньому мати доступ лише до тих ресурсів та інструментів, які необхідні для виконання своїх робочих обов'язків.

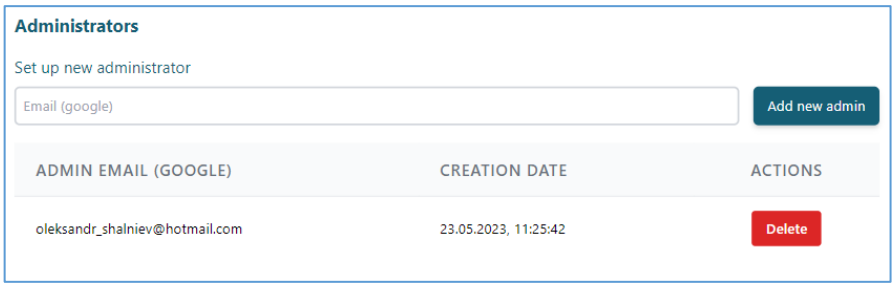

Рисунок 4.14 – Сторінка списку адміністраторів

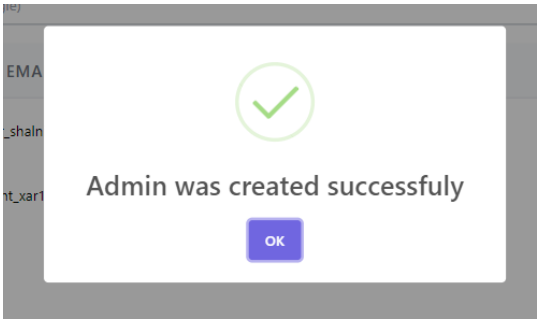

Рисунок 4.15 – Повідомлення при додаванні нового адміністратора

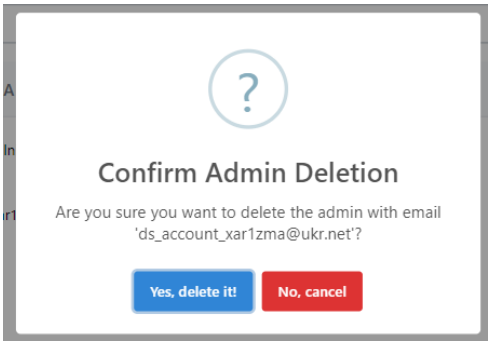

Рисунок 4.16 – Повідомлення підтвердження видалення адміністратора

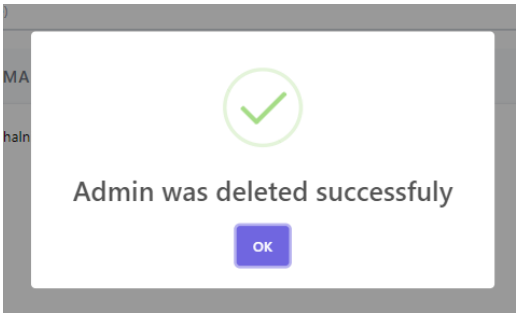

Рисунок 4.17 – Повідомлення після видалення адміністратора

4.1.4 Сторінка налаштувань та адаптивність застосунку

Секція налаштувань надає адміністратору можливість редагувати ключові параметри магазину (див. рис. 4.18).

Встановлення «featured» продукт дає можливість вибрати продукт, який буде відображатися на головній сторінці як рекомендований. Це може бути корисним для просування нових продуктів або продуктів, які є особливо популярними або важливими.

Встановлення «shipping fee» дозволяє встановити вартість доставки для замовлень. Можна встановити різні тарифи в залежності від місцезнаходження покупця, ваги замовлення або інших критеріїв.

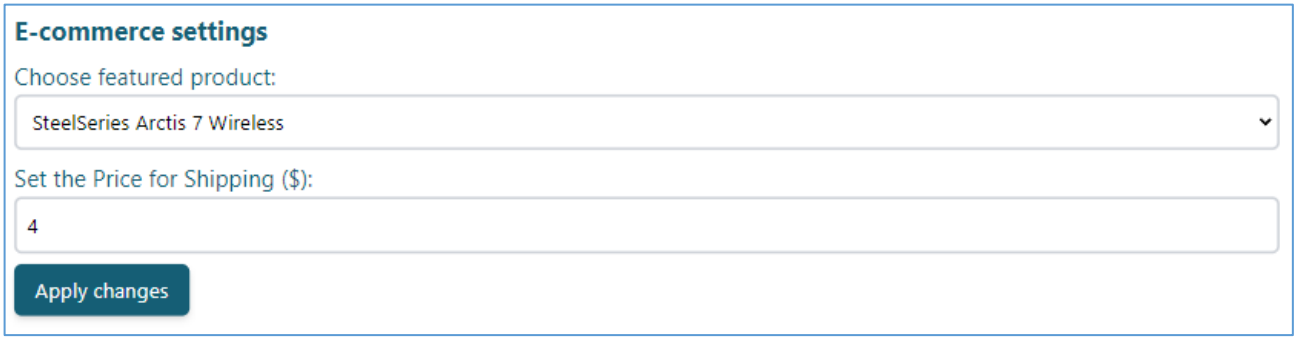

Рисунок 4.18 – Сторінка налаштувань

Адаптивність вебзастосунку є одним з ключових аспектів сучасного дизайну веб-сторінок та вебзастосунків. Адаптивний дизайн дозволяє користувачам легко і зручно взаємодіяти зі сторінкою, незалежно від пристрою, який вони використовують. Це означає, що вебзастосунок буде автоматично змінювати свій вигляд та розташування елементів на екрані в залежності від розміру вікна переглядача або типу пристрою (смартфон, планшет, настільний комп'ютер).

У рамках розробки даного вебзастосунку, особлива увага була приділена адаптивності. Використовуючи сучасні технології, такі як CSS Grid, Flexbox та медіа-запити, забезпечується гнучкість та адаптивність дизайну на різних пристроях та розширеннях екранів.

Це дозволяє користувачам зручно переглядати та взаємодіяти з веб-застосунком, не відчуваючи незручності або втрати функціональності. Таким чином, адаптивність застосунку полегшує користування та сприяє задоволеності користувачів, що, в свою чергу, підвищує шанси на успіх вебзастосунку на ринку.

У адміністративній панелі було враховано адаптивність дизайну, що дозволяє адміністраторам легко керувати контентом та налаштуваннями вебзастосунку з будь-якого пристрою та в будь-який час (див. рис. 4.19).

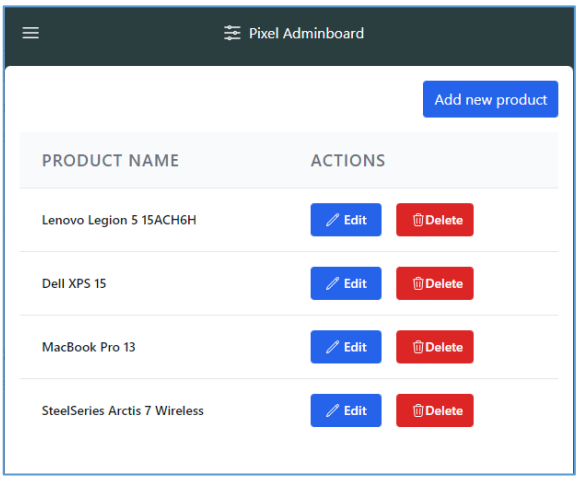

Рисунок 4.19 – Приклад однієї з сторінок на смартфоні

Як можна побачити, при зменшені розміру екрана, ліва бокова панель зникає, а замість неї у лівому верхньому куті з'являється так званий «гамбургер» (див. рис. 4.20). При натисканні на який, список усіх необхідних сторінок з'являється.

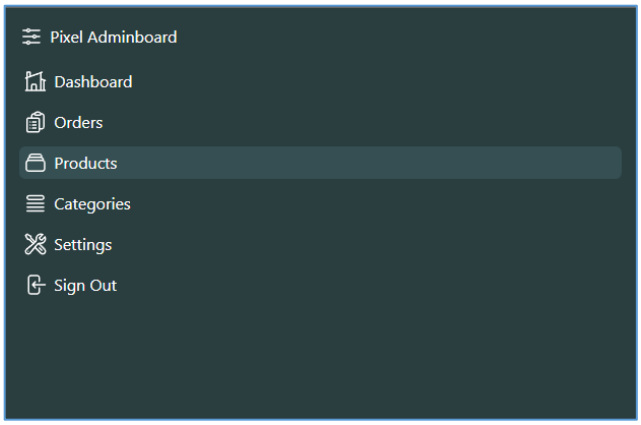

Рисунок 4.20 – Робота «гамбургеру»

Також внесені зміни до головної аналітичної сторінки, для мобільних пристроїв були прибрані графіки та залишені лише аналітичні дані у вигляді блоків (див. рис. 4.21).

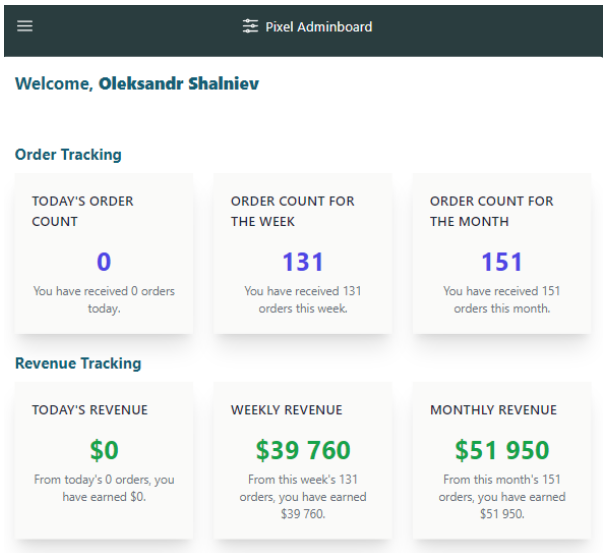

Рисунок 4.21 – Вигляд головної сторінки на телефоні

Це дозволить адміністраторам зручно користуватися застосунком з мобільних пристроїв та не матимуть зайвої навантаження на пристрій.

4.2 Керівництво користувача для Інтернет-магазина

Інтернет-магазин має багато особливостей та функцій, які важливо знати як для покупців, так і для адміністраторів сайту. Від перегляду та вибору товарів до процесу замовлення і оплати - кожен аспект важливий для забезпечення неперевершеного досвіду користувача. У цьому підрозділі надається детальне керівництво, що пояснює як найкраще використовувати Інтернет-магазин для максимальної вигоди та комфорту.

4.2.1 Головна сторінка Інтернет-магазину та сторінка продуктів

Головна сторінка Інтернет-магазину вітає відвідувачів оригінальним та інтуїтивно зрозумілим дизайном. Хедер містить навігаційне меню, яке допомагає користувачам легко перемикатися між різними сторінками сайту. Після хедера місце займає featured продукт, який адміністратор вибирає як головний. Для

кожного featured продукту наведено докладний опис, зображення, а також кнопки для додаткової інформації ("Про товар") або для покупки товару ("Придбати") (див. рис. 4.22).

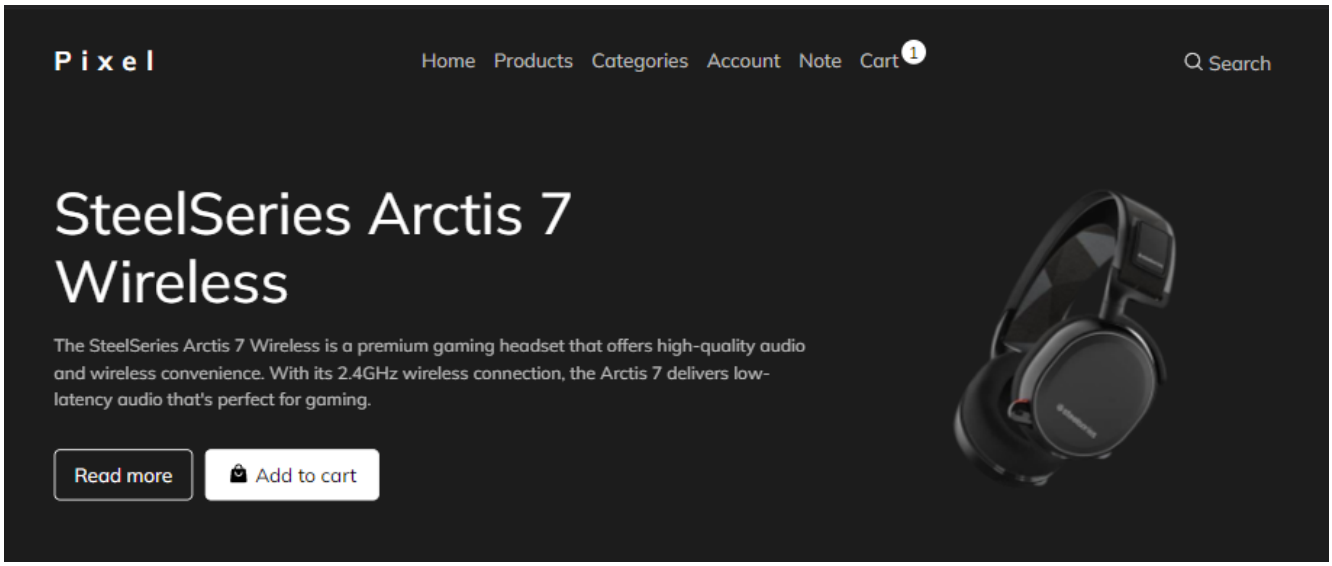

Рисунок 4.22 – Шапка вебзастосунку та featured продукт

Ця сторінка також представляє новітні товари (див. рис. 4.23), що були додані до магазину, надаючи користувачам найсвіжішу інформацію про продукцію.

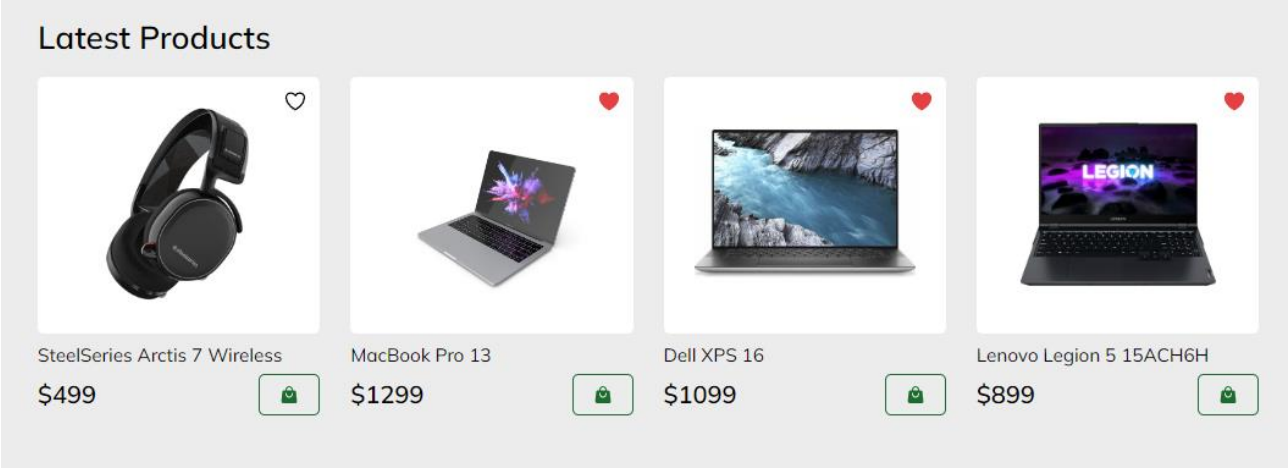

Рисунок 4.23 – Список нещодавно доданих товарів

Сторінка продуктів відображає портфоліо товарів, яке доступне для придбання в Інтернет-магазині (див. рис. 4.24). Кожний товар відображається зі своїм зображенням, коротким описом і вказаною ціною. Ця сторінка дозволяє користувачам відкрити для себе нові товари або швидко знайти ті, які їх цікавлять.

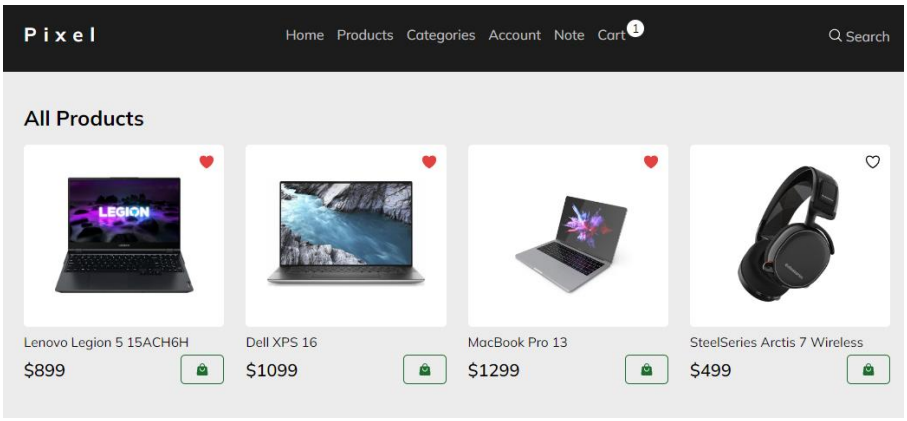

Рисунок 4.24 – Список усіх товарів

Використовуючи кнопки на кожному товарі, користувач може вибрати, що він хоче зробити далі: він може додати товар до свого списку бажань, відразу його придбати, або ж натиснути на зображення товару, щоб переглянути більше деталей про цей продукт.

4.2.2 Сторінка обраного товару

Сторінка обраного товару створена для детального представлення товару (див. рис. 4.25). Вона включає зображення товару, які можна переглядати з різних ракурсів, натискаючи на них для збільшення. Тут також вказано назву, опис і ціну товару. Якщо користувач вирішить придбати товар, він може натиснути на кнопку "Додати до кошику" для додавання товару до кошику.

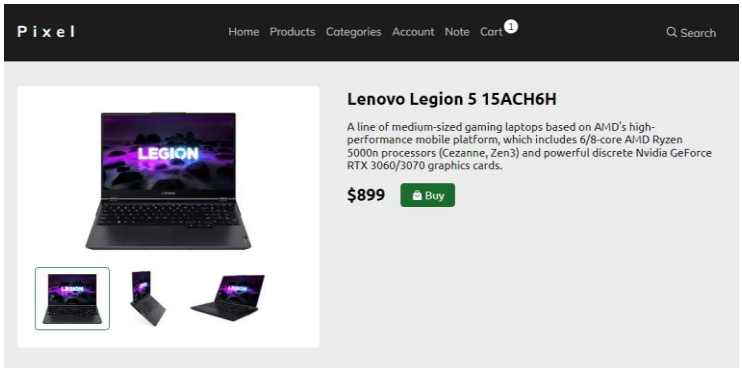

Рисунок 4.25 – Деталі обраного товару

Також на цій сторінці реалізовано алгоритм колаборативної фільтрації, який рекомендує покупцю додаткові продукти, що можуть бути йому цікаві (див. рис. 4.26).

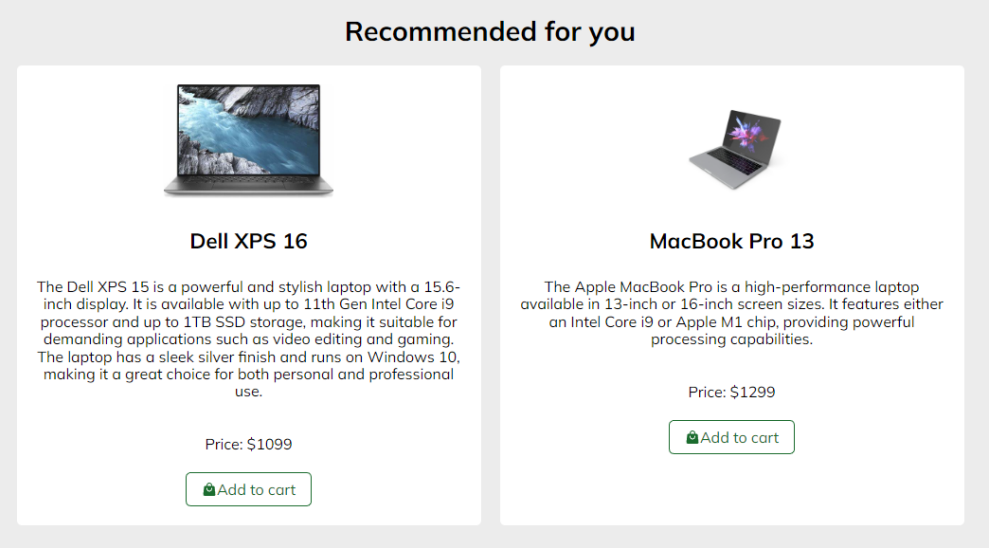

Рисунок 4.26 – Деталі обраного товару

Особливістю цієї сторінки є розділ з відгуками користувачів. Відгуки відображають рейтинг товару, а також текстовий відгук від користувачів, які вже купували товар. Крім того, користувачі, які переглядають сторінку, можуть залишити свій власний відгук, оцінити товар зірками та поділитися своїм досвідом користування товаром (див. рис. 4.27).

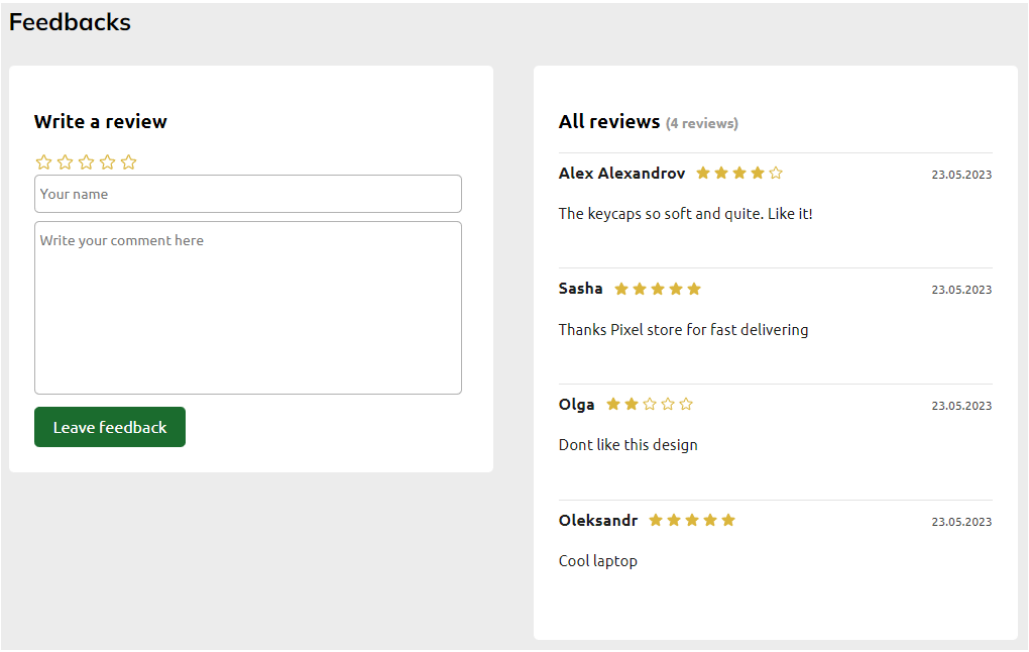

Рисунок 4.27 – Секція відгуків про товар

Якщо товар не буде містити жодного відгуку, користувач побачить текст, який повідомляє йому про це та відповідне зображення (див. рис. 4.28).

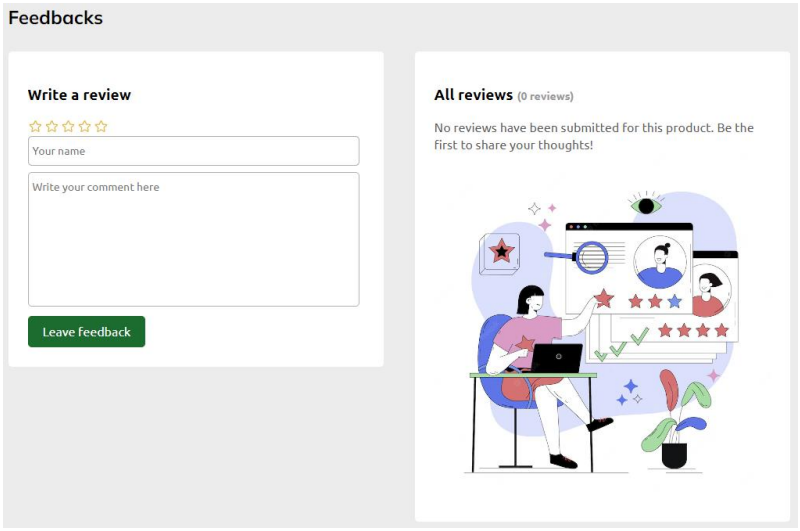

Рисунок 4.28 – Секція відгуків при відсутності жодного відгуку

Текст навмисно написано так, щоб спонукати користувача захотіти бути першим, хто залишить свій відгук.

4.2.3 Сторінка категорій

Сторінка категорії має інтуїтивний інтерфейс, що дозволяє користувачам взаємодіяти між різними групами товарів, базуючись на їхніх спільних характеристиках (див. рис. 4.29). Категорії товарів можуть бути організовані за різними принципами, такими як функціональність, використання, ціновий діапазон або будь-які інші характеристики. Це дозволяє створювати логічну структуру для широкого спектру товарів і допомагає покупцям швидко зорієнтуватися у великому асортименті.

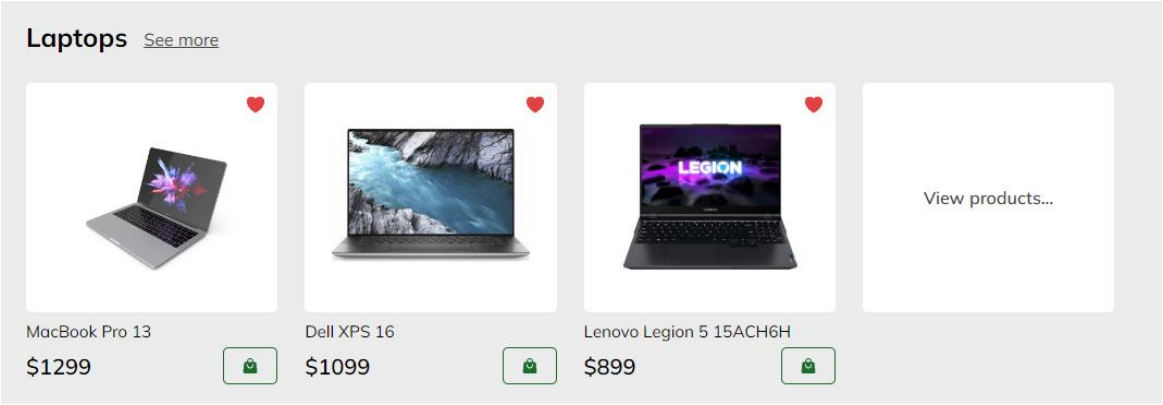

Рисунок 4.29 – Сторінка категорій товарів

Користувач може обрати конкретну категорію, натиснувши на "View products..." або "See more" і відразу ж перейти до списку товарів цієї категорії. На цій сторінці доступна функція сортування товарів (див. рис. 4.30). Користувач може вибрати сортувати товари за датою додавання (нові/старі), ціною (низькою/високою), а також іншими критеріями (див. рис. 4.31). Це полегшує відбір і пошук необхідних товарів.

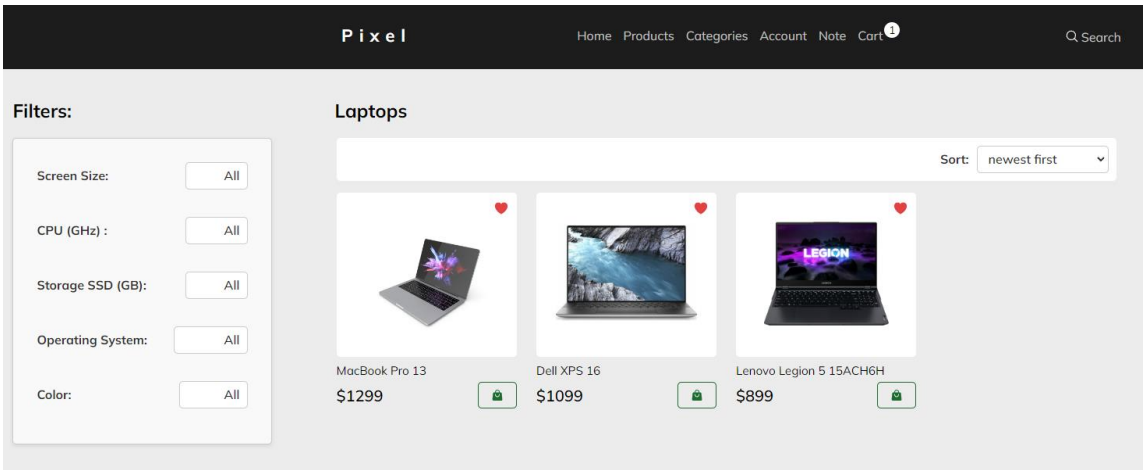

Рисунок 4.30 – Сторінка фільтрації за категорією

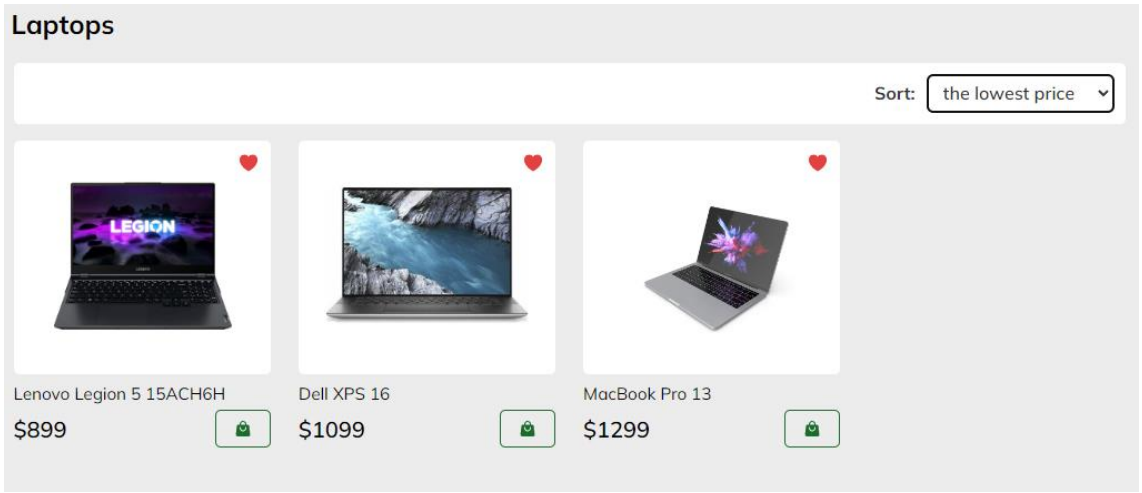

Рисунок 4.31 – Відсортовані товари за ціною (спочатку низька)

Додатково, сторінка категорії пропонує корисну функцію фільтрації товарів (див. рис. 4.32). Користувач може використовувати цей фільтр для відбору товарів за певними властивостями, такими як колір, розмір та інші, в залежності від категорії.

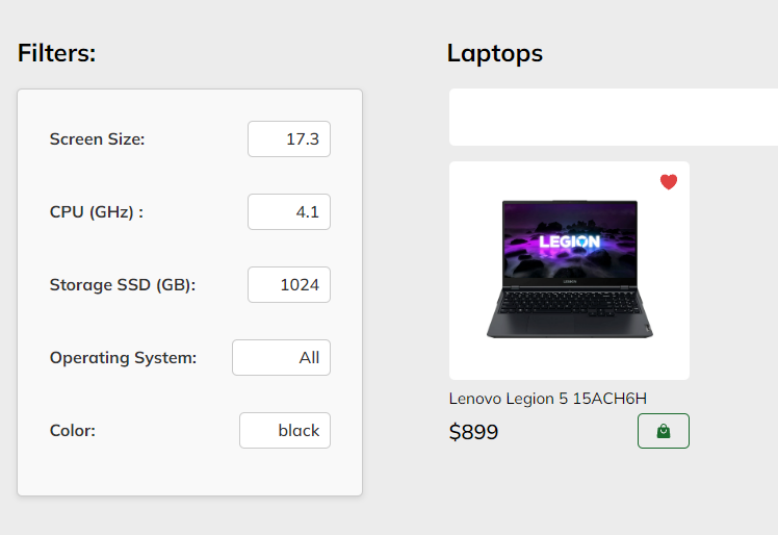

Рисунок 4.32 – Відфільтрований товар

4.2.4 Сторінка аккаунту користувача

Сторінка аккаунту служить для особистої інформації користувача. Користувач може увійти в свій аккаунт за допомогою Google, що надає йому повний доступ до всіх його особистих функцій, включаючи список бажаних товарів (див. рис. 4.33) та особисті дані.

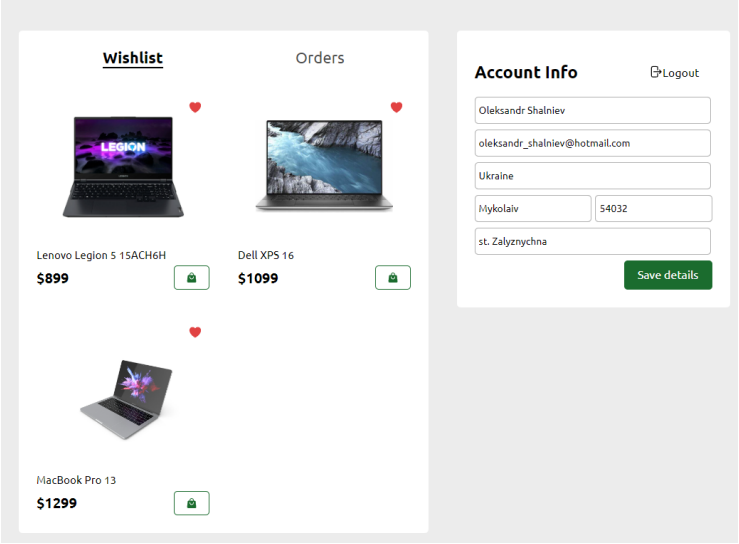

Рисунок 4.33 – Сторінка аккаунту користувача та список бажаних товарів

Крім того, на цій сторінці він може переглядати свою історію покупок, включаючи деталі кожного замовлення - які продукти були замовлені, коли та ким (див. рис. 4.34).

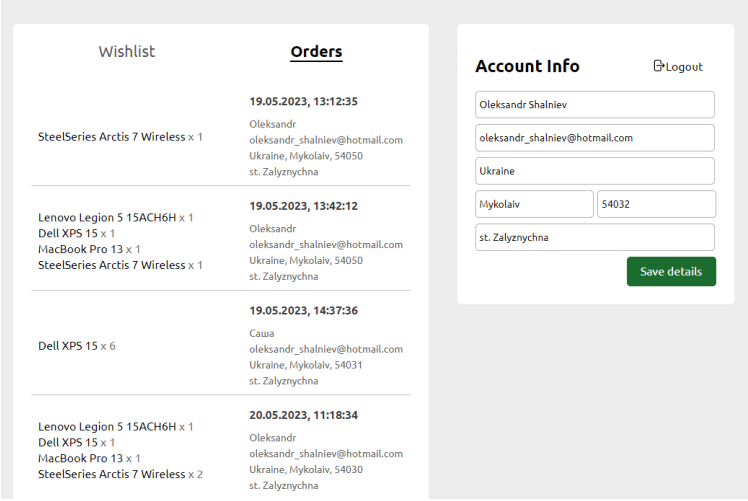

Рисунок 4.34 – Сторінка аккаунту користувача та історія замовлень

Якщо користувач не увійшов, йому не будуть доступні ці функції. Замість цих даних, він побачить текст з повідомленням про те, що для цього потрібно увійти в систему (див. рис. 4.35).

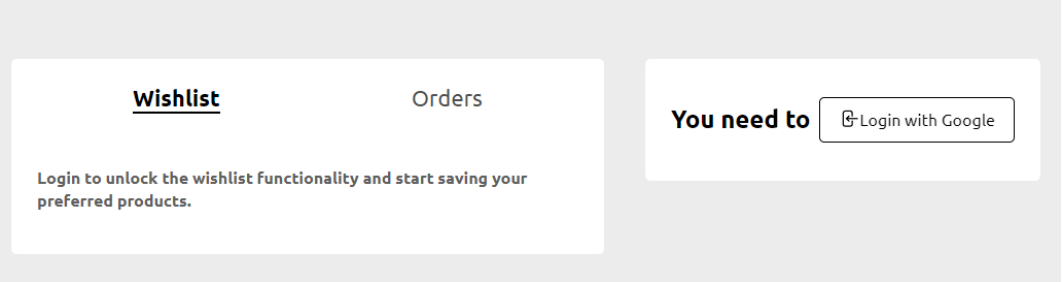

Рисунок 4.35 – Сторінка аккаунту неавторизованого користувача

Якщо список бажаних товарів пустий, користувач побачить заміть товарів красиву картинку (див. рис. 4.36).

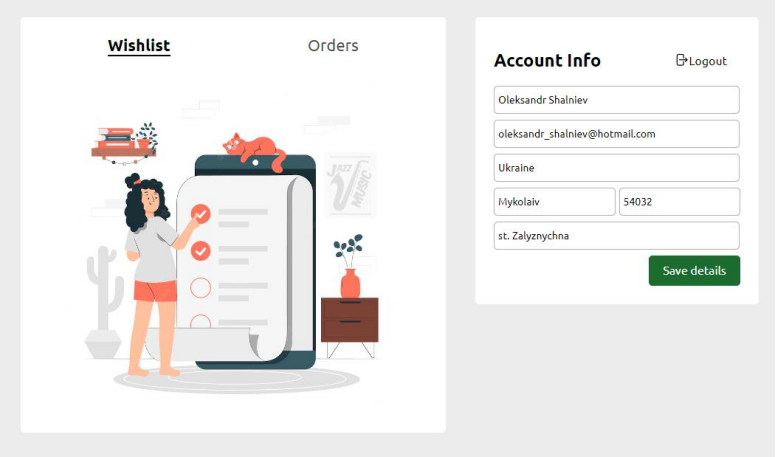

Рисунок 4.36 – Пустий список бажаних товарів

4.2.5 Сторінки нотаток та кошику

Сторінка нотаток являє особистий органайзер користувача всередині онлайн-магазину (див. рис. 4.37). Вона дозволяє користувачам створювати власні списки товарів для майбутнього розгляду або покупки. Користувач може додати посилання на конкретний товар, вказати ціну та кількість, що має велике значення при плануванні покупок. Ця функція особливо корисна, якщо користувач планує скласти комп'ютер і потребує багато різних компонентів.

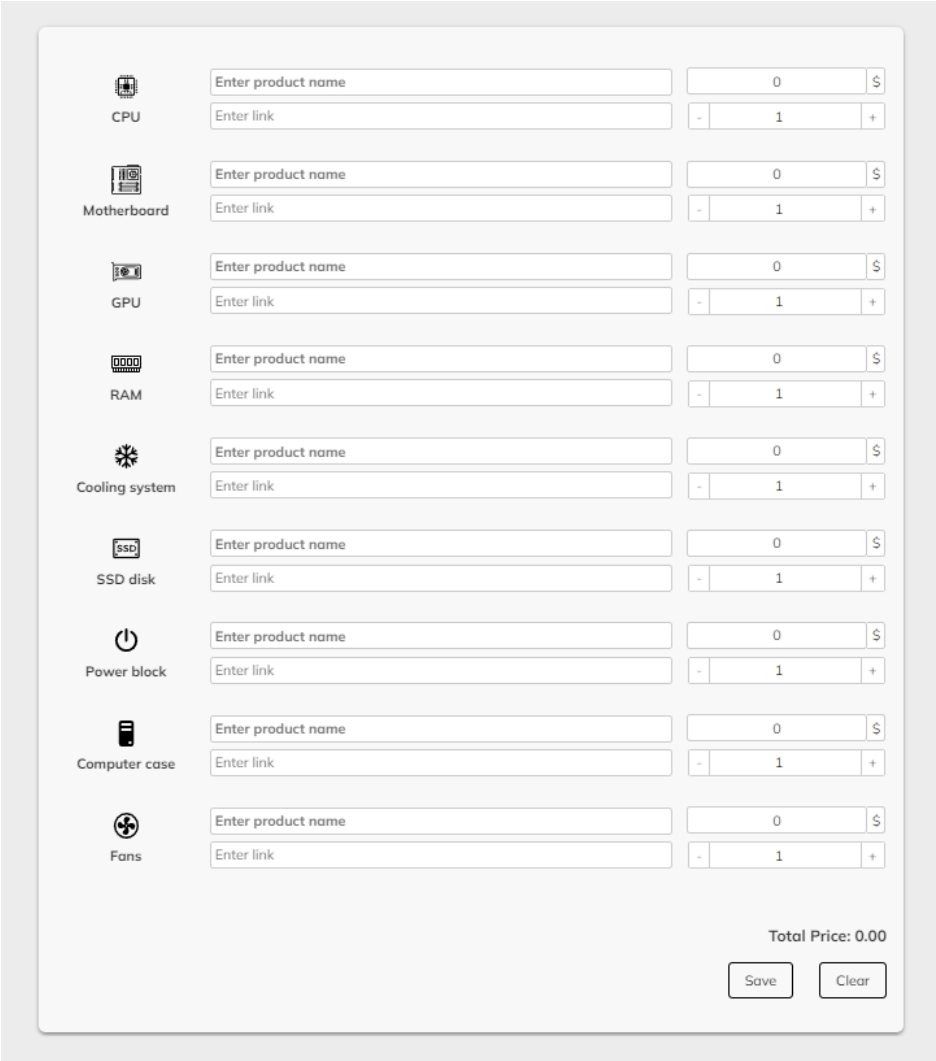

Рисунок 4.37 – Сторінка нотаток

На сторінці кошику користувач може переглянути та модифікувати всі товари, які він наміряється придбати (див. рис. 4.38). Користувач може змінювати кількість товару, видаляти товари з кошика або продовжувати покупки. Крім того,

на цій сторінці можна вказати адресу доставки та вибрати спосіб доставки. Всі ці деталі можуть бути змінені перед остаточним підтвердженням замовлення.

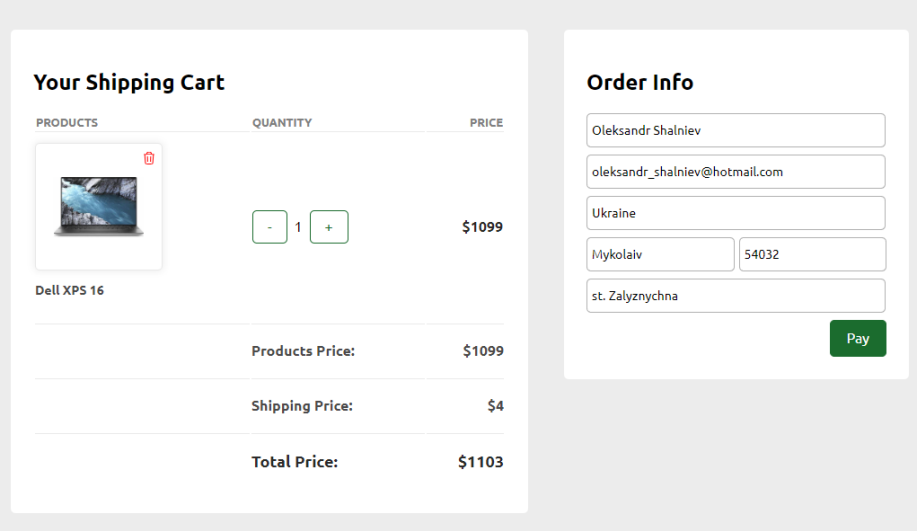

Рисунок 4.38 – Сторінка кошику з товаром

У випадку коли в кошику немає жодного товару, користувач побачить відповідне повідомлення (див. рис. 4.39).

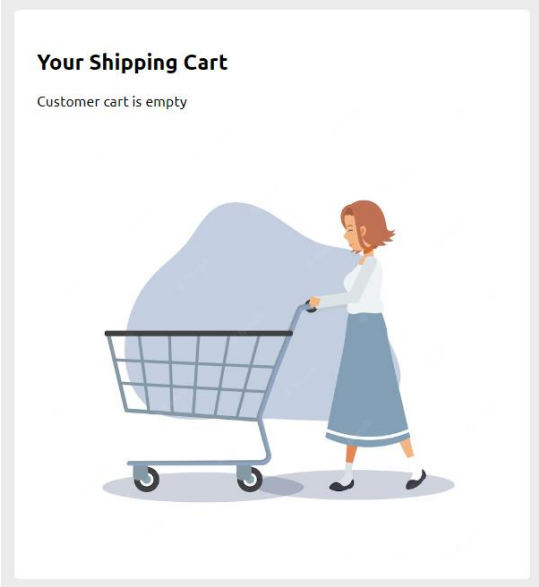

Рисунок 4.39 – Пуста сторінка кошику

Окрім цього, відбувається аналіз категорії продуктів, що знаходяться у кошику та рекомендується випадковий товар з цієї категорії, що додатково може збільшити прибуток магазину, за рахунок зацікавлення користувача придбати більше товарів (див. рис. 4.40).

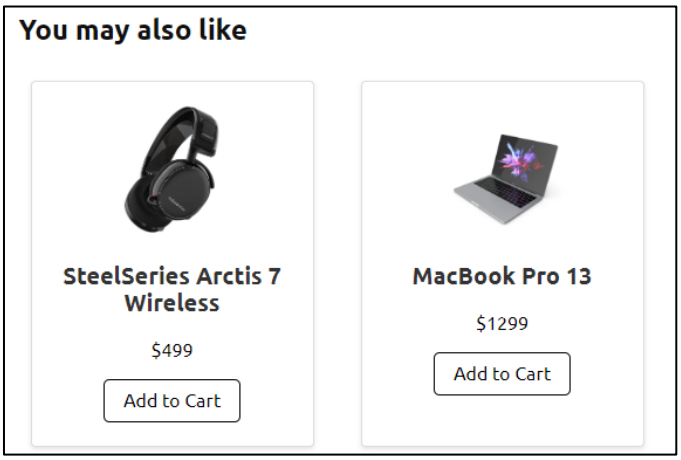

Рисунок 4.40 – Секція рекомендованих продуктів

Цю секцію було вирішено розмістити саме у кошику, тому що переглядаючи вже обрані товари, користувач може побачити ще товари, які можуть бути для нього корисні та придбати їх також.

4.2.6 Сторінка оплати та функція пошуку

На цій сторінці користувач може оформити замовлення, переглянути фінальну ціну з урахуванням всіх податків та доставки (див. рис. 4.41). Також користувач має можливість ввести промокоди для отримання знижок. Після перевірки всіх деталей замовлення, користувач може обрати спосіб оплати, який йому найбільш підходить. Це може бути оплата кредитною або дебетовою карткою (Visa, Mastercard), через платіжні системи, або інші доступні способи оплати.

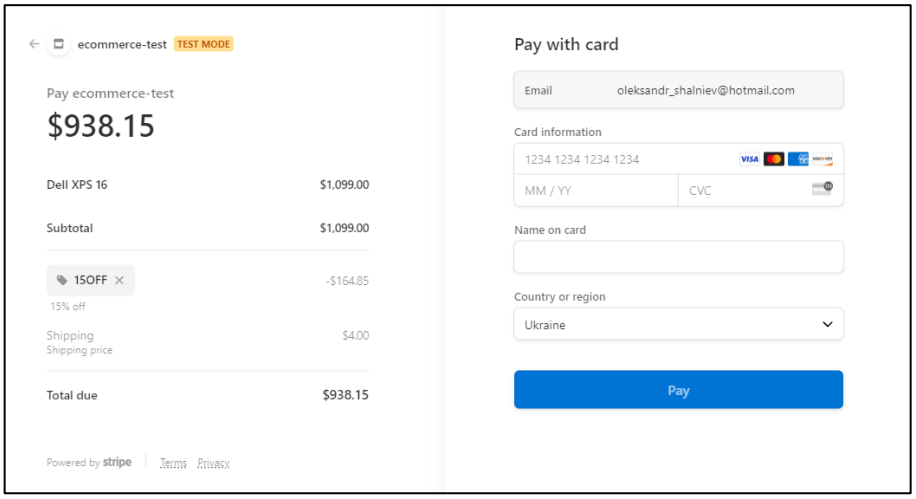

Рисунок 4.41 – Сторінка оплати

Функція пошуку є ключовою на будь-якому сайті електронної комерції. Вона дозволяє користувачам ефективно навігувати по магазину та швидко знаходити потрібні товари.

На сторінці пошуку користувачі можуть ввести ключові слова, що відповідають назві товару, його опису, характеристикам або іншим категоріям. Результати пошуку відображаються у відповідному розділі сайту, який включає зображення товару, короткий опис та ціну (див. рис. 4.42).

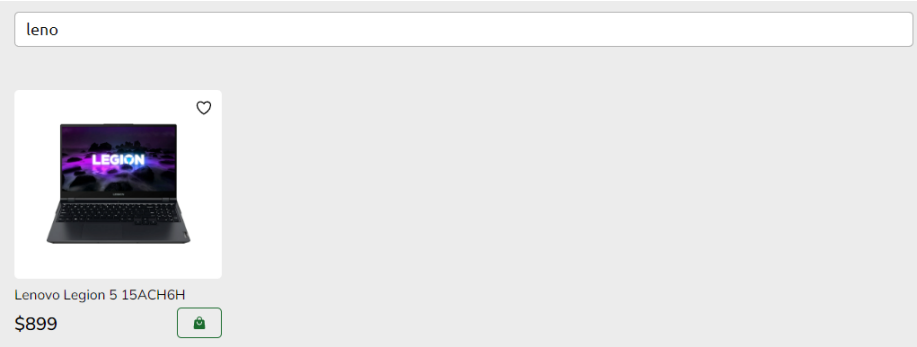

Рисунок 4.42 – Вдалий пошук товарів

При невдалому пошуку товару, якщо його немає – користувач побачить відповідне повідомлення (див. рис. 4.43).

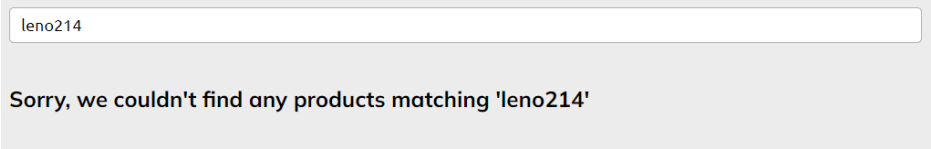

Рисунок 4.43 – Невдалий пошук товарів

4.2.7 Тестування швидкодії застосунку

Швидкодія вебзастосунку відіграє важливу роль у задоволенні потреб користувачів та забезпеченні їхньої високої взаємодії з платформою. Вона може бути визначена як швидкість, з якою застосунок відповідає на запити користувача. Швидкодія застосунку залежить від багатьох факторів, включаючи алгоритмічну складність, оптимізацію коду, кешування даних, використання пам'яті, мережеві затримки та інші фактори. Для оцінки ефективності вебзастосунку було проведено аналіз та побудовано відповідну діаграму (див. рис. 4.44).

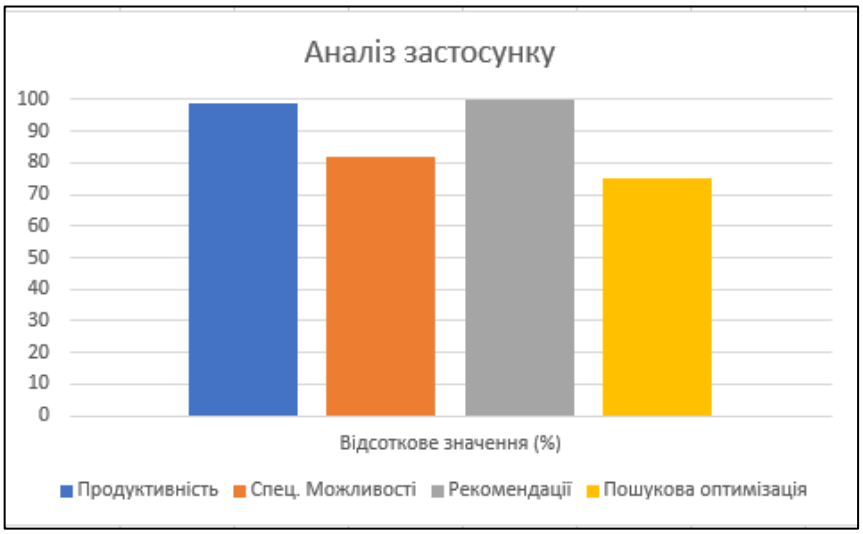

Рисунок 4.44 – Аналіз застосунку

За результатами аналізу, продуктивність застосунку та його спеціальні можливості виявилися на досить високому рівні.

Окрім діаграми з аналізом було побудовано кругову діаграму, яка демонструє фактори що впливають на швидкодію вебзастосунку (див. рис. 4.45).

Рисунок 4.45 – Аналіз застосунку

За діаграмою, оптимізація коду відіграє найважливішу роль у покращенні швидкодії, становлячи 40% від загального впливу. Кешування даних та використання пам'яті також є значущими факторами. Мережеві затримки та інші фактори впливають менше, але все одно важливі для загального розуміння швидкодії.

Висновки до четвертого розділу

Згортуючи четвертий розділ, можна прийти до висновку про необхідність та корисність такого вебзастосунку з продажу комп'ютерних компонентів. Було детально описано основний функціонал сайту, включаючи роботу з головною сторінкою, сторінкою продуктів, сторінкою конкретного товару, сторінкою категорії, сторінкою аккаунту, сторінкою нотаток, сторінкою кошику, сторінкою оплати, а також функцією пошуку на сайті. Кожен з цих підрозділів було аналізовано, й було вказано, в якому випадку кожен з них використовується.

Система демонструє високу функціональність та ефективність, надаючи користувачам можливість швидко та легко знаходити потрібні компоненти, відслідковувати їх доступність та ціни, зберігати обрані товари та керувати своїми покупками та замовленнями. Крім того, простота використання, доступний інтерфейс, а також зручність керування зробили цей застосунок дуже привабливим для користувачів, надаючи йому переваги над багатьма іншими системами електронної комерції.

ВИСНОВКИ

У ході проведеного дослідження розроблено вебзастосунок на основі стеку технологій MERN для підвищення ефективності процесу продажу комп'ютерної техніки та обладнання. Було здійснено детальний аналіз предметної області продажу комп'ютерної техніки та обладнання через Інтернет, розглянуто основні підходи до створення вебзастосунку і аналізовано аналоги у цій сфері.

Під час роботи також було використано різні інформаційні технології, сервіси і бібліотеки для вирішення поставлених завдань. Описано використання обраного середовища розробки (IDE), особливості використання MERN стеку, методів, фреймворків, бібліотек і сервісів.

У процесі роботи було реалізовано обрані технології і сервіси, описано технічну частину адміністративної панелі та Інтернет-магазину, включаючи реалізацію backend і frontend частин, підключення до БД, використання Google авторизації та AWS S3 сервіс. Було також підготовлено керівництво користувача для адміністративної панелі та Інтернет-магазину, детально описано всі сторінки та основні функції вебзастосунку.

Вебзастосунок було протестовано на відповідність вимогам та були вивчені можливості щодо масштабування та оптимізації. Всі вказані завдання були успішно виконані, що підтверджує досягнення мети роботи.

Результати цього дослідження можуть слугувати основою для подальшого розвитку та масштабування проекту, а також як основа для проведення подібних досліджень в інших сферах електронної комерції. Досвід розробки та імплементації цього вебзастосунку може бути використаний при розробці подібних проектів на основі стеку технологій MERN.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. D. Flanagan, JavaScript: The Definitive Guide, 7th ed. Sebastopol, CA, USA: O'Reilly Media, 2020. pp. 704.

2. Мікросервісна архітектура: плюси та мінуси. URL: https://blog.iteducenter.ua/articles/microservices-architecture-advantages-and-disadventages/ (дата звернення: 29.04.2023).

3. Fink G, Flatow I. Pro Single Page Application Development. Apress, 2014. pp. 287.

4. 9 трендів eCommerce, які вам потрібно знати у 2023 році. URL: https://blog.payoneer.com/ua/e-sellers-ua/industry-tips-sellers-ua/9-ecommerce-trends-you-need-to-know-for-2023/ (дата звернення: 01.05.2023).

5. Магазин комп’ютерної техніки та електроніки ArtLine. URL: https://artline.ua/uk/catalog/kompyutery-artline (дата звернення: 01.05.2023).

6. Магазин комп’ютерної техніки та електроніки IT-Block. URL: https://www.it-blok.com.ua/ (дата звернення: 01.05.2023).

7. Магазин комп’ютернох техніки та електроніки Telemart.UA. URL: https://telemart.ua/ua/ (дата звернення: 01.05.2023).

8. Середовище розробки VS Code. URL: https://code.visualstudio.com/ (дата звернення: 02.05.2023).

9. S. Bradshaw, E. Brazil, and K. Chodorow, MongoDB: The Definitive Guide, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2019. pp. 514.

10. E. Brown, Web Development with Node and Express: Leveraging the JavaScript Stack, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2019. pp. 343.

11. A. Banks and E. Porcello, Learning React: Functional Web Development with React and Redux, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2017. pp. 350.

12. D. Herron, Node.js Web Development: Server-side web development made easy with Node 14 using practical examples, 5th ed. Birmingham, UK: Packt Publishing, Date Missing. pp. 760.

13. M. Riva, Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production. Birmingham, UK: Packt Publishing, 2022. pp. 366.

14. 8 Benefits of ReactJS: Is It Worth Using in Your Project. URL: https://procoders.tech/blog/advantages-of-using-reactjs/ (дата звернення: 05.05.2023).

15. Використання бібліотеки sweet-modal. URL: https://classic.yarnpkg.com/en/package/sweet-modal (дата звернення: 05.05.2023).

16. Інструція підключення бібліотеки SweetAlert. URL: https://sweetalert.js.org/guides/ (дата звернення: 05.05.2023).

17. Chart.js Samples. URL: https://www.chartjs.org/docs/latest/samples/information.html (дата звернення: 05.05.2023).

18. How to use chart.js to create charts in React. URL: https://www.educative.io/answers/how-to-use-chartjs-to-create-charts-in-react (дата звернення: 05.05.2023).

19. Why Micro JavaScript Library Should Be Used in Your Next Application. URL: http://surl.li/ibere (дата звернення: 06.05.2023).

20. Using styled-components in React. URL: https://blog.logrocket.com/using-styled-components-in-react/ (дата звернення: 06.05.2023).

21. How to Create a Loading Animation in React with react-spinners. URL: https://stackabuse.com/how-to-create-a-loading-animation-in-react-with-react-spinners/ (дата звернення: 06.05.2023).

22. R. Copeland, MongoDB Applied Design Patterns: Practical Use Cases with the Leading NoSQL Database, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2013. pp. 176.

23. How to use multiparty. URL: https://www.tabnine.com/code/javascript/modules/multiparty (дата звернення: 08.05.2023).

24. Understanding the interface to sortable.js. URL: https://cran.r-project.org/web/packages/sortable/vignettes/understanding_sortable_js.html (дата звернення: 09.05.2023).

25. Setting up Tailwind CSS in a Create React App project. URL: https://tailwindcss.com/docs/guides/create-react-app (дата звернення: 09.05.2023).

26. Google Cloud documentation. URL: https://cloud.google.com/docs/ (дата звернення: 10.05.2023).

27. What is Amazon S3? URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html (дата звернення: 11.05.2023).

28. How To Use an API (The Complete Guide). URL: https://rapidapi.com/blog/how-to-use-an-api/ (дата звернення: 12.05.2023).

29. What Is an API Endpoint? URL: https://www.baeldung.com/cs/api-endpoints (дата звернення: 11.05.2023).

30. Amazon Simple Storage Service Documentation. URL: https://docs.aws.amazon.com/s3/index.html (дата звернення: 11.05.2023).

31. Authentication at Google. URL: https://cloud.google.com/docs/authentication/ (дата звернення: 12.05.2023).

32. Authentication token types. URL: https://cloud.google.com/docs/authentication/token-types (дата звернення: 13.05.2023).

33. What is Axios? URL: https://axios-http.com/docs/intro (дата звернення: 14.05.2023).

34. D. Flanagan, JavaScript: The Definitive Guide, 7th ed. Sebastopol, CA, USA: O'Reilly Media, 2020. pp. 656.

35. A. Giamas, Mastering MongoDB 4.x: Expert techniques to run high-volume and fault-tolerant database solutions using MongoDB 4.x, 2nd ed. Birmingham, UK: Packt Publishing, 2019. pp. 394.