МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Таврійський державний агротехнологічний університет
імені Дмитра Моторного
Факультет енергетики і комп'ютерних технологій

ЗАТВЕРДЖУЮ
Зав. каф. "Комп'ютерні науки"
доц._____ Сергій ШАРОВ
"17" _лютого____2025 р.

Пояснювальна записка

до кваліфікаційної роботи здобувача СВО Магістр
(ступінь вищої освіти)

на тему: «Інтернет-магазин для продажу мобільних телефонів з модулем рекомендацій»

52/4КНД.11260545.000000ПЗ

Виконав: здобувач вищої освіти 2 курсу, групи 21МБКН
спеціальності 122 Комп'ютерні науки
за ОПП Комп'ютерні науки
(шифр і назва спеціальності та ОПП)

_____ _____ Василь ДУДЛА
(підпис)

Керівник, проф. _____ Віра МАЛКІНА
(підпис)

Нормоконтроль, ст.викл. _____ Ольга ЗІНОВ'ЄВА
(підпис)

Рецензент_____ Альона ЧОРНА
(підпис)

Запоріжжя – 2025 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Таврійський державний агротехнологічний університет імені Дмитра Моторного

Інститут, факультет___ЕКТ Кафедра комп'ютерних наук_____
Ступінь вищої освіти _____
Спеціальність _____122 Комп'ютерні науки_____
(шифр і назва)
ОПП _____Комп'ютерні науки_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри_____
к.пед.н., доц. Сергій ШАРОВ
"____" _____2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) ЗДОБУВАЧУ ВО

_____Дудлі Василю Васильовичу_____
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Інтернет-магазин для продажу мобільних телефонів з модулем рекомендацій_____
керівник проєкту (роботи) Малкіна Віра Михайлівна, професор_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом університету від "23" вересня__2025__року №_442-С__
2. Строк подання здобувачем ВО проєкту (роботи) 17 лютого 2025 р._____
3. Вихідні дані до проєкту (роботи) вимоги до проектування інтернет-магазинів; технічне завдання на проектування інтернет-магазину з продажу мобільних телефонів, матеріали виробничих практик, нормативні документи, науково-технічна література, електронні ресурси та ін._____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області та існуючих рішень _____
2. Специфікація вимог до інтернет-магазину з продажу мобільних телефонів_____
3. Обґрунтування мови та середовища програмування._____
4. Дослідна експлуатація інтернет-магазину з продажу мобільних телефонів_____
5. Тестування інтернет-магазину з продажу мобільних телефонів_____
5. Перелік графічного матеріалу

Діаграма варіантів використання, діаграма послідовності, діаграма IDEF0, структура проекту, структура бази даних інтерфейс користувача_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	12.09.2024	виконано
2	Специфікація вимог до експертної системи	06.10.2024	виконано
3	Проектування експертної системи	16.10.2024	виконано
4	Обґрунтування інструментальних засобів	05.11.2024	виконано
5	Розробка експертної системи	10.12.2024	виконано
6	Тестування та налагодження експертної системи	20.01.2025	виконано
7	Підпис керівником проєкту	14.02.2025	виконано
8	Підпис завідувачем кафедри	17.02.2025	виконано

Здобувач ВО _____ _____Василь ДУДЛА_____
(підпис) (власне ім'я та ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ _____Віра МАЛКІНА_____
(підпис) (власне ім'я та ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота магістра: 76 сторінок, 54 рисунки, 8 таблиць, 32 посилання.

Актуальність: Сучасний розвиток електронної комерції потребує створення ефективних та зручних інтернет-магазинів, які забезпечують користувачам швидкий доступ до необхідних товарів. Важливим аспектом є персоналізація взаємодії із клієнтом, що дозволяє покращити досвід покупця та підвищити рівень продажів. Впровадження модуля рекомендацій у систему інтернет-магазину є ефективним способом автоматизації підбору товарів, що відповідають уподобанням користувача.

Об’єкт дослідження: програмне забезпечення для електронної комерції.

Предмет дослідження: інтернет-магазин для продажу мобільних телефонів із модулем рекомендацій.

Мета роботи: розробити інтернет-магазин для вибору та придбання мобільних телефонів.

Завдання дослідження:

а) Проаналізувати сучасні тенденції у розробці інтернет-магазинів, зокрема функціональні можливості модулів рекомендацій;
б) Розглянути існуючі програмні рішення, що забезпечують автоматизацію електронної комерції;
в) Вибрати інструментальні засоби для реалізації інтернет-магазину та модуля рекомендацій;
г) Розробити технічне завдання для створення інтернет-магазину;
д) Реалізувати інтернет-магазин із функціями каталогу товарів, кошика та модуля рекомендацій;
е) Провести тестування розробленого інтернет-магазину.

Практичне значення дослідження: розроблений інтернет-магазин може бути використаний як комерційне рішення для продажу мобільних телефонів.

Ключові слова: ІНТЕРНЕТ-МАГАЗИН, ЕЛЕКТРОННА КОМЕРЦІЯ, РЕКОМЕНДАЦІЙНА СИСТЕМА, MERN, АДАПТИВНИЙ ДИЗАЙН.

SUMMARY

Master’s Qualification Thesis: 76 pages, 54 figures, 8 tables, 32 references.

Relevance: The modern development of e-commerce requires the creation of efficient and user-friendly online stores that provide customers with quick access to the necessary products. A crucial aspect is the personalization of customer interactions, which enhances the shopping experience and increases sales. Implementing a recommendation module in an online store system is an effective way to automate product selection according to user preferences.

Object of research: Software for e-commerce.

Subject of research: An online store for selling mobile phones with a recommendation module.

Purpose of the study: develop an online store for selecting and purchasing mobile phones.

Research tasks:

a) Analyze current trends in online store development, particularly the functional capabilities of recommendation modules;
b) Examine existing software solutions that automate e-commerce;
c) Select the tools for implementing the online store and the recommendation module;
d) Develop a technical specification for creating the online store;
e) Implement an online store with product catalog, shopping cart, and recommendation module functionalities;
f) Conduct testing of the developed online store.

Practical significance of the study: The developed online store can be used as a commercial solution for selling mobile phones.

Keywords: ONLINE STORE, E-COMMERCE, RECOMMENDATION SYSTEM, MERN, ADAPTIVE DESIGN.

ЗМІСТ

ВСТУП

В сучасному світі мобільні телефони займають центральне місце у повсякденному житті людей, об’єднуючи комунікацію, роботу та розваги в одному пристрої. З огляду на стрімке зростання попиту на мобільні телефони, створення інтернет-магазинів є важливим кроком для бізнесу, що прагне відповідати очікуванням сучасного споживача. Інтернет-магазини не лише забезпечують зручність покупки для клієнтів, але й дозволяють автоматизувати процеси торгівлі, підвищуючи ефективність бізнесу.

Актуальність теми полягає у розробці інтернет-магазину, що спеціалізується на продажу мобільних телефонів та включає модуль рекомендацій. Такий функціонал дозволить не тільки спростити процес вибору товарів для клієнтів, але й підвищити продажі за рахунок використання персоналізованих рекомендацій. Інтеграція технологій машинного навчання у модуль рекомендацій є важливим трендом у розвитку електронної комерції та суттєво збільшує ефективність взаємодії з клієнтами.

Об’єктом дослідження є програмне забезпечення для електронної комерції, що використовується для продажу товарів через інтернет.

Предметом дослідження виступає інтернет-магазин для продажу мобільних телефонів з модулем рекомендацій.

Метою кваліфікаційної роботи є розробка інтернет-магазину, який забезпечить зручний механізм для вибору та покупки мобільних телефонів, а також впровадження модуля рекомендацій, що дозволить персоналізувати взаємодію з клієнтами та підвищити ефективність продажів.

Для досягнення поставленої мети необхідно виконати такі завдання:

а) Проаналізувати сучасні тенденції у розробці інтернет-магазинів, зокрема функціональні можливості модулів рекомендацій;
б) Розглянути існуючі програмні рішення, що забезпечують автоматизацію електронної комерції;
в) Вибрати інструментальні засоби для реалізації інтернет-магазину та модуля рекомендацій;

г) Розробити технічне завдання для створення інтернет-магазину;

д) Реалізувати інтернет-магазин із функціями каталогу товарів, кошика та модуля рекомендацій;

е) Провести тестування розробленого інтернет-магазину.

При виконанні поставлених завдань було використано теоретичні методи аналізу літератури та сучасних програмних підходів, емпіричні методи тестування функціональності системи, а також практичні підходи до створення програмного продукту з використанням сучасних технологій веб-розробки.

Практичне значення цієї роботи полягає в тому, що розроблений інтернет-магазин може бути використаний як готове рішення для торгівлі мобільними телефонами. Впроваджений модуль рекомендацій допоможе підвищити задоволеність клієнтів, збільшити обсяг продажів та забезпечити конкурентну перевагу в умовах сучасного ринку.

Структура роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків і списку використаних джерел.

У першому розділі представлено аналіз предметної області: розглянуто основні поняття електронної комерції, характеристики інтернет-магазинів, їхню класифікацію, а також визначено основні вимоги до створення інтернет-магазинів.

Другий розділ присвячено постановці задач і плануванню проекту. У ньому визначено мету та завдання роботи, розглянуто основні поняття стеку технологій для створення односторінкових веб-додатків, обґрунтовано вибір стеку MERN для реалізації проекту та описано механізми аутентифікації й авторизації користувачів за допомогою JWT.

Третій розділ присвячено проектуванню інтернет-магазину. У ньому описано розробку концепції веб-сайту, програмну реалізацію проекту та створення адаптивного інтерфейсу для забезпечення зручності використання платформи на різних пристроях.

Четвертий розділ охоплює дослідну експлуатацію та тестування інтернет-магазину. У ньому детально описано програмну реалізацію, розроблено інструкцію користувача та проведено аналіз функціональності системи на основі практичного тестування.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Опис предметної області

Предметною областю є онлайн-магазин для продажу мобільних телефонів, що має модуль рекомендацій для персоналізованих пропозицій користувачам. Основна мета предметної області – забезпечити ефективний процес продажу мобільних телефонів з урахуванням інтересів і вподобань користувачів, а також вдосконалити взаємодію з клієнтами через автоматизовані системи.

Електронна комерція, або «e-commerce», охоплює процес купівлі та продажу товарів чи послуг через Інтернет, а також обробку фінансових операцій і передачу даних, необхідних для цих транзакцій. Хоча цей термін зазвичай використовується для позначення онлайн-продажу фізичних товарів, він також охоплює будь-яку комерційну діяльність, що здійснюється в цифровому середовищі. Щороку електронна комерція продовжує зростати, займаючи дедалі більшу частку ринку, збільшуючи обсяги продажів і охоплюючи нові сфери бізнесу. Вона включає різноманітні процеси, такі як бронювання та виконання замовлень, а також фінансові операції через банківські системи або електронні платіжні сервіси.

Важливою складовою електронної комерції є розвиток технологій, які забезпечують безпечні та зручні методи здійснення транзакцій. Системи безпеки, такі як SSL (Secure Sockets Layer), захищають дані під час передачі, тоді як платіжні шлюзи спрощують процес оплати для споживачів. Крім того, аналіз даних та штучний інтелект відіграють ключову роль у персоналізації покупок, пропонуючи споживачам товари на основі їхніх попередніх покупок та переглядів.

Розрізняють чотири основні моделі електронної комерції, які охоплюють практично всі типи взаємодії між компаніями та споживачами [29]:

а) Business-to-Consumer (B2C) – продаж товарів або послуг компанією безпосередньо кінцевим споживачам (наприклад, придбання взуття в інтернет-магазині).

б) Business-to-Business (B2B) – комерційні угоди між компаніями, наприклад, продаж програмного забезпечення для корпоративного використання.

в) Consumer-to-Consumer (C2C) – транзакції між споживачами, коли один користувач продає товар іншому (наприклад, продаж уживаних меблів через eBay).

г) Consumer-to-Business (C2B) – ситуація, коли споживачі пропонують свої товари чи послуги компаніям, наприклад, блогер рекламує бренд за винагороду або фотограф продає свої знімки для комерційного використання [29].

Рисунок 1.1 — Основні типи моделей електронної комерції

Сучасна електронна комерція охоплює більшість ринків B2B і B2C, проте для її ефективного функціонування необхідні такі компоненти:

а) Інтернет-платформа (веб-сайт, обліковий запис, онлайн-магазин або цільова сторінка);
б) Канали залучення трафіку (SEO, SMM, контекстна та таргетована реклама);
в) Системи обробки замовлень і підтримки клієнтів (CRM, відділи продажів, служби підтримки).

Послуги в електронній комерції включають закупівлю, постачання, доставку та повернення товарів, а самі транзакції можуть відбуватися в різних форматах:

а) Роздрібна торгівля – продаж товарів безпосередньо покупцям без посередників;
б) Оптова торгівля – реалізація продукції великими партіями для подальшого перепродажу;
в) Дропшиппінг – продаж товарів, які виробляє та доставляє третя сторона;
г) Краудфандинг – збір коштів від споживачів ще до випуску товару для фінансування його виробництва;
д) Підписка – автоматичне продовження доступу до продукту чи послуги, поки користувач не скасує підписку.

Продукти поділяються на три основні категорії:

а) Фізичні товари, що потребують поповнення запасів і фізичної доставки [32];
б) Цифрові продукти, які доступні для завантаження (моделі, курси, медіафайли);
в) Послуги, що включають надання певних навичок або робочого часу за плату.

Здійснення онлайн-транзакцій потребує створення спеціалізованих ресурсів, що полегшують процес для всіх учасників. Різні моделі електронної комерції підтримують кілька типів веб-сайтів:

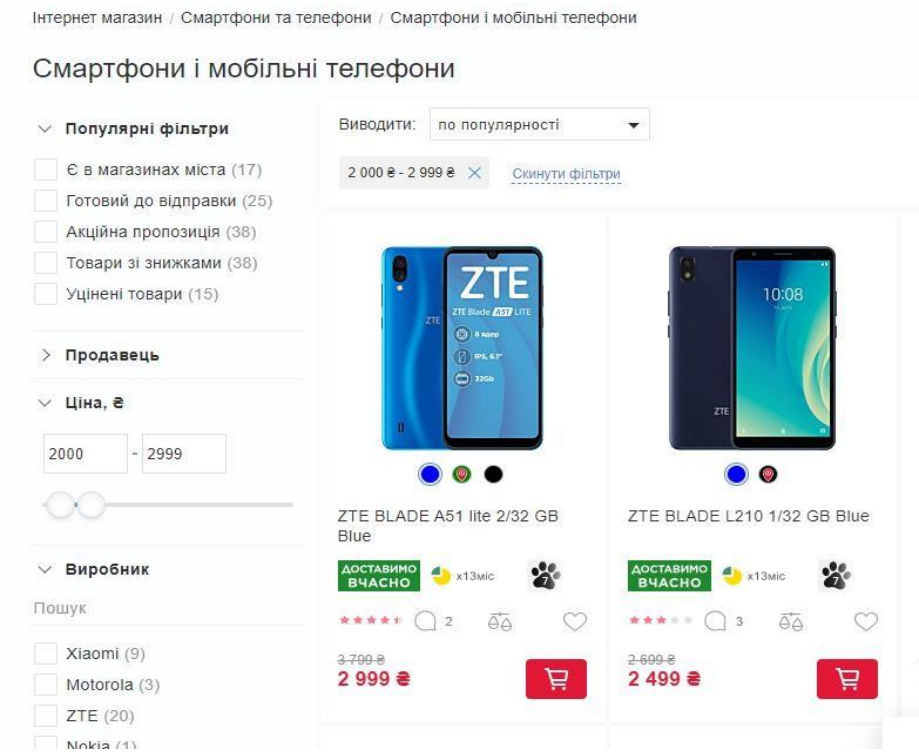

Рисунок 1.2 — Інтернет - магазин мережі «АЛЛО»

Сервіси оголошень орієнтовані на модель C2C і надають користувачам можливість рекламувати свої товари чи послуги. Вони дозволяють створювати оголошення з деталями покупки та категоризацією для зручного пошуку потенційними покупцями (як показано на рис. 1.2).

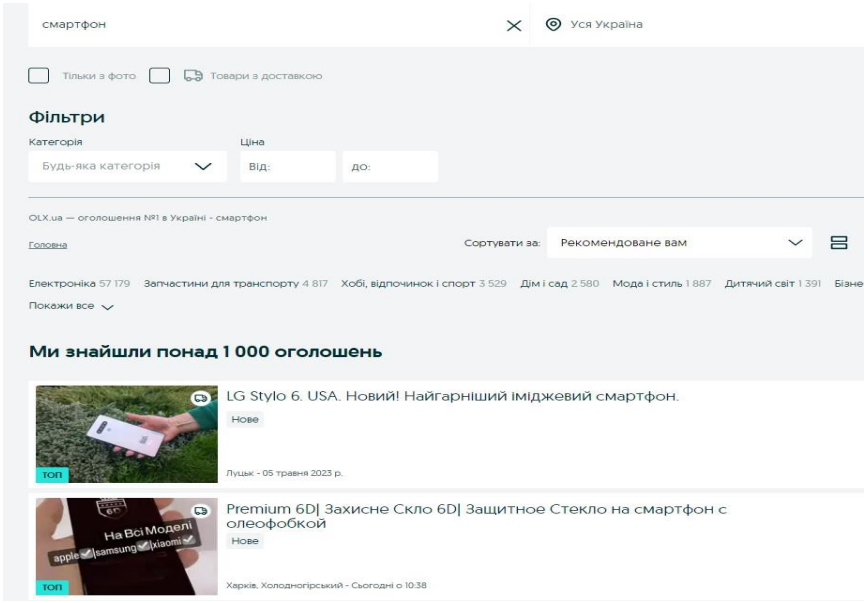

Рисунок 1.3 — Сервіс оголошень OLX

Сайти знижок агрегують акційні пропозиції від різних постачальників і продавців, об’єднуючи їх за спільною умовою – використання промокоду для отримання знижки (рис. 1.4).

Рисунок 1.4 — Сервісу знижок Superdeal

Класичним прикладом електронної комерції є інтернет-магазин. Далі розглянемо його поняття та ключові особливості.

1.2 Огляд і аналіз існуючих аналогів, що реалізують функції предметної області

Для порівняння інтернет-магазину з іншими вебресурсами продажу мобільних телефонів, слід звернути увагу на кілька основних аспектів: адаптивний дизайн, інтуїтивний інтерфейс та наявність модуля рекомендацій на основі вподобань користувачів.

Таблиця 1.1 — Порівняльна характеристика програмних продуктів

Вебсайт	Адаптивний дизайн	Інтуїтивний інтерфейс	Модуль рекомендацій
Інтернет магазин	+	+	+
Hotline	+	+	-
Megamarket-shop	+	-	-
E-Katalog	+	+	-
Chaynikam	-	-	-

Інтернет-магазин має адаптивний дизайн, що автоматично підлаштовується під різні пристрої (мобільні телефони, планшети, десктопи), забезпечуючи комфортний перегляд. Інші сайти також мають адаптивний дизайн. Інтерфейс інтернет-магазину зручний і зрозумілий, що дозволяє легко знаходити необхідну інформацію і здійснювати покупки. Інші сайти також мають інтуїтивно зрозумілий інтерфейс. Також наявний модуль рекомендацій на основі вподобань користувачів, що персоналізує досвід покупок. Інші конкуренти не пропонують аналогічного функціоналу.

Екранні форми основних варіантів використання продукту:

а) Головна сторінка: Показує популярні продукти, категорії та персоналізовані рекомендації;

б) Сторінка продукту: Детальна інформація про товар, характеристики, відгуки та можливість додавання до кошика;

в) Кошик: Перегляд вибраних товарів, редагування кількості та оформлення замовлення;

г) Профіль користувача: Управління особистими даними, історією покупок та налаштуваннями рекомендацій.

Інтернет-магазин вирізняється на фоні конкурентів завдяки наявності модуля рекомендацій на основі вподобань користувачів, що підвищує персоналізацію і зручність використання. Це є важливою конкурентною перевагою, яка може бути використана для подальшого покращення проєктних рішень.

1.3 Технічне завдання на розробку інтернет магазину з модулем рекомендацій

Процес розробки веб-застосунку — це чітко визначена послідовність етапів, що ведуть до створення повноцінного програмного продукту. Він охоплює проєктування архітектури, розробку користувацького інтерфейсу, реалізацію серверної логіки, організацію взаємодії з базою даних та фінальне розгортання застосунку на сервері чи хостинговій платформі.

Таблиця 1.2 – Використані технології для створення інтернет магазину

Технологія	Зміст
HTML	Мова розмітки, що використовується для структурування вмісту веб-сторінок. Визначає заголовки, текст, зображення, форми та інші елементи інтерфейсу.
JAVASCRIPT	Основна мова програмування для веб-розробки, що використовується як на клієнтській, так і на серверній стороні
CSS (SCSS)	Каскадні таблиці стилів, що дозволяють створювати адаптивний та кросбраузерний дизайн. SCSS додає змінні, вкладеність та міксіни, що полегшує управління стилями та підвищує їх повторне використання.
React.js	Потужна бібліотека для створення динамічних інтерфейсів користувача за допомогою компонентного підходу. Використовує віртуальний DOM для оптимізації рендерингу
Express.js	Легкий серверний фреймворк для Node.js
MongoDB	NoSQL база даних, що використовується для зберігання структурованої та неструктурованої інформації
JWT (JSON Web Token)	Технологія токенів для автентифікації та авторизації користувачів
GitHub Pages	Платформа для хостингу статичних файлів, що використовується для розгортання фронтенд-частини веб-застосунку

При розробці онлайн-магазину мобільних телефонів на основі MERN-стеку було виконано наступні етапи:

а) визначення мети створення веб-застосунку та його конкурентних переваг: адаптивний дизайн, рекомендаційна система на базі AI, швидкий процес оформлення замовлення;
б) підготовка матеріалів для наповнення сайту: інформація про товари, характеристики, зображення та описи;
в) розробка архітектури застосунку, включаючи структуру бази даних та REST API;
г) розробка UI/UX дизайну, створення макетів та їх верстка;
д) Розробка клієнтської частини з використанням React.js;
е) Створення серверної частини на основі Node.js та Express.js;
ж) Реалізація взаємодії з базою даних MongoDB для зберігання інформації про товари, користувачів та замовлення;
з) Розгортання та тестування застосунку, включаючи налаштування хостингу та доменного імені;
и) Оптимізація продуктивності та налаштування безпеки (JWT-автентифікація, обмеження доступу тощо).

Розглянемо докладніше ключові технології, що використовуються для розробки веб-додатків.

JavaScript — це мова програмування, призначена для динамічного оновлення веб-сторінок, взаємодії з користувачем і обробки даних без необхідності перезавантаження сторінки. Вона широко застосовується як на клієнтській, так і на серверній стороні (завдяки середовищу Node.js).

Сучасний стандарт ECMAScript містить потужні можливості, такі як покращене керування винятками, більш жорсткий контроль помилок і підтримка багатопотокової обробки через Web Workers. JavaScript дозволяє створювати як прості інтерактивні елементи сторінки, так і складні веб-додатки, що взаємодіють із сервером у реальному часі.

Основні напрями використання JavaScript:

а) Фронтенд-розробка – створення інтерактивних інтерфейсів за допомогою React, Vue.js або Angular;

б) Бекенд-розробка – побудова серверної логіки через Node.js та Express.js;
в) Розробка мобільних додатків – використання технологій на базі React Native.

HTML (HyperText Markup Language) – це стандартна мова розмітки, що визначає структуру веб-сторінок. Вона використовується для створення заголовків, абзаців, списків, посилань, зображень, форм та інших елементів інтерфейсу. Завдяки HTML можна організувати контент, забезпечуючи логічну ієрархію сторінки.

CSS (Cascading Style Sheets) – це мова стилів, яка відповідає за оформлення HTML-елементів, дозволяючи змінювати кольори, розміри, шрифти, відступи та розташування елементів на сторінці. Використовуючи CSS, можна створювати адаптивний дизайн, щоб сторінки коректно відображалися на різних пристроях.

Переваги використання сучасних стандартів CSS:

а) Адаптивний дизайн – технології Flexbox та Grid Layout забезпечують зручне компонування елементів для різних екранів;
б) Оптимізація продуктивності – CSS-анімації виконуються на рівні GPU, що зменшує навантаження на процесор і покращує плавність інтерфейсу;
в) Використання CSS-змінних – дозволяє створювати теми оформлення та легко змінювати стилі без редагування кожного окремого елемента;
г) Підтримка темної та світлої теми – завдяки prefers-color-scheme можна автоматично змінювати кольорове оформлення залежно від налаштувань користувача;
д) Модульність стилів – використання BEM-методології або CSS-модулів у React допомагає уникнути конфліктів стилів.

Система управління базами даних (СУБД) – це програмне забезпечення, яке дозволяє створювати, керувати та взаємодіяти з базами даних. Вони бувають реляційними (SQL) та нереляційними (NoSQL), кожен тип має свої переваги та випадки використання. Однією з найпопулярніших NoSQL-баз є

MongoDB – документно-орієнтована база даних, що ідеально підходить для роботи з масштабованими веб-застосунками. Вона зберігає дані у форматі BSON (бінарний аналог JSON), що забезпечує швидке зчитування та запис інформації.

Основні можливості MongoDB:

а) гнучка схема даних – дозволяє легко змінювати структуру без складних міграцій, що особливо корисно для швидко змінюваних проєктів;

б) висока продуктивність – завдяки індексуванню, кешуванню та оптимізованій обробці запитів забезпечується швидке виконання операцій з даними;

в) масштабованість – підтримує горизонтальне масштабування через шардінг (розподіл даних між кількома серверами) та реплікацію (створення копій даних для підвищення надійності);

г) вбудовані механізми безпеки – підтримує аутентифікацію, шифрування даних і контроль доступу на рівні ролей;

д) вбудована підтримка геопросторових запитів – дозволяє ефективно зберігати та обробляти дані про розташування, що корисно для картографічних сервісів.MongoDB ідеально інтегрується з Node.js завдяки бібліотеці Mongoose, що дозволяє працювати з даними у зручному форматі об'єктних моделей.

Для побудови сучасних веб-додатків використовується стек MERN (MongoDB, Express.js, React, Node.js). Це повноцінний набір технологій, який охоплює всі аспекти розробки – від роботи з базою даних до створення інтерфейсу користувача.

MongoDB – нереляційна NoSQL база даних, що зберігає інформацію у вигляді JSON-подібних документів. Вона підтримує масштабування, має гнучку структуру даних і швидко обробляє запити.

Express.js – мінімалістичний бекенд-фреймворк для створення API та обробки HTTP-запитів. Він спрощує роботу з маршрутами, середовищем запитів і відповідає за зв’язок між клієнтською та серверною частинами.

React – популярна JavaScript-бібліотека для створення динамічних інтерфейсів на основі компонентного підходу. Вона забезпечує високу продуктивність і можливість розробки SPA (Single Page Applications).

Node.js – серверне середовище виконання JavaScript, що дозволяє працювати з бекенд-логікою, обробляти запити до бази даних і керувати серверними ресурсами.

Основна перевага використання MERN – можливість працювати з єдиною мовою програмування, JavaScript, як на фронтенді, так і на бекенді. Це спрощує процес розробки, дозволяє ефективно використовувати спільний код і зменшує необхідність вивчення додаткових мов. Завдяки цьому MERN є чудовим вибором для швидкої розробки масштабованих веб-додатків.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ З ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ

2.1 Концептуальна модель використання інформаційної системи

У цьому підрозділі представлено концептуальну модель використання інформаційної системи інтернет-магазину мобільних телефонів. Описано основні варіанти використання, наведено відповідну UML-діаграму, специфікацію сценаріїв роботи користувачів, розкадровку взаємодії із системою та базову структуру бази даних.

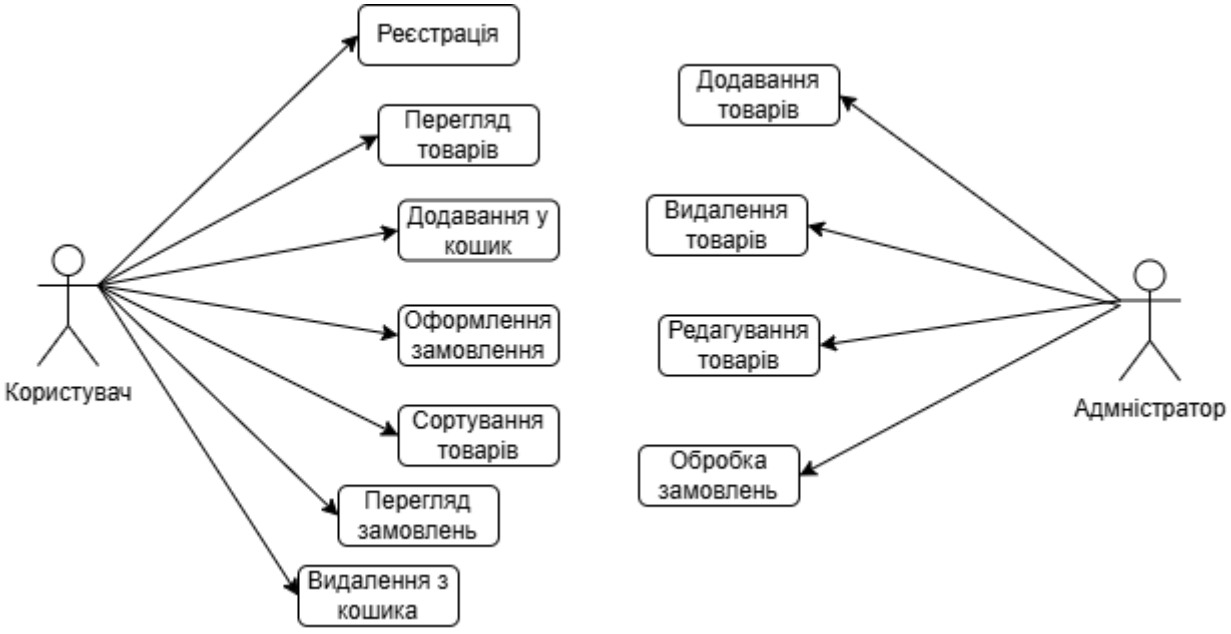

Рисунок 2.1 — Діаграма варіантів використання

Діаграма варіантів використання дозволяє наочно представити функціональність інформаційної системи, яка буде реалізована. Вона відображає, які дії може виконувати користувач, і які сценарії передбачено в системі.

На діаграмі зображено користувача системи, який має доступ до таких варіантів використання, як перегляд каталогу, пошук товару, перегляд детальної інформації про товар, оформлення замовлення та інші операції.

Для кожного варіанту використання необхідно визначити його основні параметри: контекст застосування, дійові особи, тригери, сценарій виконання та результат.

Таблиця 2.1 — Специфікація варіантів використання

Варіант	Контекст	Дійові особи	Тригер	Результат
Перегляд каталогу	Користувач відкриває каталог товарів	Користувач	Відкриття каталогу	Відображено список товарів
Пошук товару	Користувач вводить пошуковий запит	Користувач	Введення запиту	Відображено знайдені товари
Перегляд сторінки товару	Користувач відкриває сторінку товару	Користувач	Натискання на товар	Відображено характеристики та опис
Додавання товару в кошик	Користувач додає товар у кошик	Користувач	Натискання "Додати в кошик"	Товар додано в кошик
Оплата замовлення	Користувач обирає метод оплати	Користувач	Вибір способу оплати	Оплата виконана, підтвердження отримано
Перегляд рекомендацій	Користувач переглядає персоналізовані пропозиції	Користувач	Відкриття сторінки	Відображено список рекомендованих товарів

Розкадровка (storyboard) представляє візуальне відображення сценаріїв роботи користувача з інтернет-магазином. Вона допомагає візуалізувати інте-

рфейс системи та покроковий процес виконання завдань. Основні кадри розкадровки: Головна сторінка – містить банери, категорії товарів, персоналізовані рекомендації. Каталог товарів – містить перелік товарів із можливістю сортування та фільтрації. Сторінка товару – включає опис товару, галерею зображень, кнопку "Додати в кошик". Кошик – відображає список обраних товарів, підсумкову ціну та кнопку "Оформити замовлення". Оформлення замовлення – містить форми введення контактних даних та вибору способу оплати. Підтвердження замовлення – відображає повідомлення про успішне оформлення покупки.

2.2 Опис функціональних підсистем

Успішне функціонування онлайн-магазину забезпечується сукупністю взаємопов’язаних підсистем, кожна з яких виконує визначені бізнес-функції. Ці підсистеми відповідають за аналіз поведінки користувачів, формування персоналізованих рекомендацій, оформлення та обробку замовлень, а також надання клієнтської підтримки.

У цьому підрозділі розглянуто основні функціональні підсистеми, що складають бізнес-процес, зображений на рисунку 2.2. Визначено їхню роль у роботі магазину та взаємодію між собою для покращення користувацького досвіду та підвищення ефективності продажів.

Склад функцій, що входять до бізнес-процесу (рис. 2.2):

а) аналіз поведінки користувачів – збір даних про діяльність користувачів на сайті для формування рекомендацій;
б) формування рекомендацій – створення персоналізованих пропозицій на основі попередніх покупок та переглядів;
в) оформлення замовлення – процес додавання товару до кошика та завершення покупки;
г) обробка замовлень – підтвердження замовлення та організація доставки товарів;

д) підтримка клієнтів – надання консультацій і допомоги користувачам через онлайн-чат, електронну пошту чи телефон.

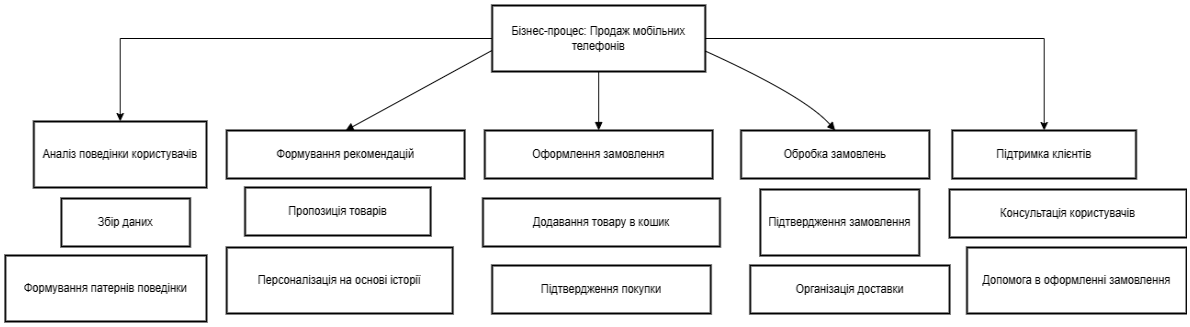

Рисунок 2.2 — Склад функцій бізнес процесу

Моделювання бізнес-процесів здійснюється з використанням CASE-інструментів. Це дає змогу детально описати як транзакційну, так і аналітичну складову бізнес-процесів. Транзакційна складова включає збори та обробку кількісних даних про діяльність користувачів, таких як кількість відвідувань, вибір товарів і завершення покупок. Ці дані використовуються для формування бази для подальших аналітичних досліджень. Аналітична складова аналізує ці дані для виявлення тенденцій, підвищення ефективності рекомендацій і оптимізації процесу продажу. На рисунку 2.3 представлена діаграма IDEF0, яка візуалізує основні процеси цього бізнес-процесу, включаючи вхідні дані, механізми та результати.

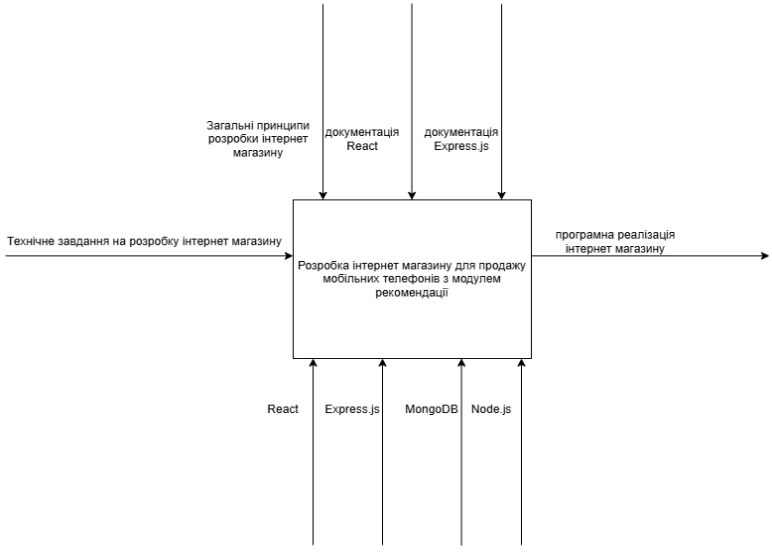

Рисунок 2.3 — Контекстна діаграма A-0

Таблиця нижче містить основні характеристики бізнес-процесу продажу мобільних телефонів з модулем рекомендацій, включаючи учасників, вхідні та вихідні дані, а також події, які ініціюють та завершують процес.

Таблиця 2.2 — Схема управління бізнес-процесом

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Продаж мобільних телефонів з модулем рекомендацій
Основні учасники	1. Менеджер каталогу товарів: оновлення інформації про товари
	2. Аналітик: аналіз поведінки користувачів
	3. Розробник: підтримка та оновлення системи рекомендацій
Вхідна подія	Перехід користувача на сайт та перегляд каталогу товарів
Вхідні документи	Каталог товарів, дані користувачів, історія покупок
Вихідна подія	Оформлення замовлення та підтвердження покупки
Вихідні документи	Замовлення, рахунок-фактура, підтвердження покупки
Клієнт бізнес-процесу	Користувач, який робить покупку мобільного телефону

Після визначення ключових елементів процесу та їх взаємодій, важливо також здійснити детальний аналіз кількісних показників, що дозволить підвищити ефективність бізнесу та краще зрозуміти тенденції продажів і поведінки користувачів. Основні напрямки для аналізу:

а) за періодами часу: Вивчення сезонних трендів у продажах дозволяє виявити пікові періоди та підготовитися до них, зокрема через акції або збільшення запасів товарів;

б) за товарами: Аналіз найбільш популярних моделей допомагає зрозуміти, які саме мобільні телефони користуються найбільшим попитом, і на основі цього адаптувати асортимент чи рекламні кампанії;
в) за клієнтами: Аналіз поведінки та попиту різних груп користувачів дозволяє персоналізувати пропозиції та вдосконалити рекомендаційну систему, що підвищує задоволеність клієнтів;
г) за підрозділами: Оцінка ефективності команд маркетингу, продажів та підтримки дозволяє виявити сильні та слабкі сторони у роботі цих команд, що сприяє подальшому вдосконаленню процесів.

Завдяки такому підходу можна приймати більш обґрунтовані рішення щодо оптимізації процесу продажу мобільних телефонів, що позитивно впливає на бізнес-показники та задоволеність клієнтів.

2.3 Специфікація функціональних та нефункціональних вимог

Важливим етапом у створенні інтернет-магазину є визначення ключових специфікацій, формування вимог до програмного продукту, написання технічного завдання, розподіл завдань, визначення етапів реалізації та розрахунок бюджету. Також на цьому етапі розробляється супровідна документація, яка описує всі аспекти проєкту.

Функціональні вимоги визначають поведінку інтернет-магазину та можуть змінюватися залежно від специфіки бізнесу. Вони є ядром системи та визначають основні можливості, доступні користувачам. Нефункціональні вимоги, своєю чергою, регламентують якісні характеристики продукту, такі як швидкість роботи, масштабованість, безпека, доступність і зручність використання.

Функціональні вимоги:

а) інтеграція зі сторонніми сервісами (платіжні системи, логістичні компанії);
б) мобільна адаптивність;

в) каталог товарів із фільтрами та сортуванням;

г) оформлення замовлення (кошик, сторінка оформлення покупки);

д) особистий кабінет користувача (перегляд замовлень, редагування профілю);

е) система відгуків і рейтингів;

ж) підписка на розсилку та push-сповіщення.

Нефункціональні вимоги:

а) Зручність використання (інтуїтивний інтерфейс, зрозуміла навігація);

б) Безпека (захист персональних даних, безпечні платежі);

в) Відмовостійкість (стабільна робота, запобігання втраті даних);

г) Висока продуктивність (швидке завантаження сторінок, оптимізація запитів);

д) Масштабованість (готовність до зростання навантаження).

У ході реалізації проєкту були визначені функціональні вимоги, які забезпечують основні можливості інтернет-магазину.

Основні функціональні можливості інтернет-магазину:

а) Користувачі можуть переглядати каталог товарів, використовувати фільтри та сортування для пошуку потрібної продукції;

б) Реєстрація та авторизація користувачів дозволяє зберігати історію замовлень, обрані товари та персональні налаштування;

в) Реалізована інтеграція з платіжними сервісами для зручної та безпечної оплати замовлень;

г) Є можливість залишати відгуки, оцінювати товари та спілкуватися через чат підтримки;

д) Оформлення замовлення відбувається через кошик із можливістю вибору способу доставки та оплати;

е) Адміністратори магазину можуть керувати каталогом товарів, змінювати ціни та додавати акції.

Таблиця 2.3 – Функціональні вимоги до інтернет-магазину

№	Назва вимоги	Пріоритет	Склад-ність	Контакт
1	Перегляд головної сторінки	Обов’язкове	Низька	Користувач, Адміністратор
2	Реєстрація користувачів	Обов’язкове	Середня	Користувач, Адміністратор
3	Авторизація користувачів	Обов’язкове	Середня	Користувач, Адміністратор
4	Навігація та пошук товарів	Обов’язкове	Середня	Користувач
5	Оформлення замовлення	Обов’язкове	Висока	Користувач, Адміністратор
6	Управління каталогом	Обов’язкове	Висока	Адміністратор
7	Відгуки та рейтинги	Необов’язкове	Середня	Користувач
8	Інтеграція з платіжними сервісами	Обов’язкове	Висока	Адміністратор

Окрім функціональності, важливо враховувати якісні характеристики веб-застосунку. Інтернет-магазин має бути не лише функціональним, а й швидким, безпечним і привабливим для користувачів.

Основні нефункціональні вимоги:

а) Дизайн сайту повинен бути адаптивним і відповідати сучасним стандартам UX/UI, щоб користувачі могли зручно користуватися ним на будь-якому пристрої;

б) Оптимізація продуктивності має забезпечувати швидке завантаження сторінок, навіть при великій кількості товарів у каталозі;

в) Захист персональних даних користувачів, шифрування паролів і використання HTTPS для безпечного з’єднання;

г) Інтернет-магазин повинен бути масштабованим, щоб підтримувати зростання кількості користувачів та замовлень без втрати продуктивності.

Таблиця 2.4 – Нефункціональні вимоги до інтернет-магазину

№	Назва вимоги	Характеристики
1	Зручність використання	Час навчання нових користувачів – 10-15 хв, досвідчених – 5 хв
2	Надійність	Час безвідмовної роботи – 14 днів
3	Продуктивність	Швидкість завантаження сторінок – не більше 3 секунд
4	Масштабованість	Підтримка великої кількості товарів та одночасних запитів
5	Безпека	Захист даних користувачів, інтеграція з платіжними шлюзами

Нефункціональні вимоги також регламентують взаємодію інтернет-магазину з зовнішніми сервісами, вимоги до бази даних, обмеження проєктування та розгортання системи.

Ретельно визначені функціональні та нефункціональні вимоги дозволяють створити інтернет-магазин, який відповідає очікуванням користувачів та бізнес-цілям. Функціональні вимоги забезпечують ключові можливості платформи, а нефункціональні – її зручність, продуктивність та безпеку.

2.4 Інформаційне забезпечення системи

Інформаційне наповнення та веб-дизайн є ключовими аспектами створення ефективного та привабливого онлайн-магазину. Інформаційне напов-

нення включає розміщення структурованої та актуальної інформації про товари, їхні характеристики, описи, відгуки клієнтів та рекомендації. Важливим елементом є також блоки з інформацією про способи оплати, умови доставки, гарантію та повернення товарів. Крім цього, на сайті повинна бути доступна контактна інформація, а також розділ з відповідями на поширені запитання.

Розробка веб-дизайну інтернет-магазину повинна враховувати сучасні тенденції та принципи зручності користування. Важливо створити привабливий і водночас інтуїтивно зрозумілий інтерфейс, що забезпечує швидку навігацію та зручний доступ до каталогу товарів. Дизайн повинен бути адаптивним, щоб коректно відображатися на різних пристроях, включаючи смартфони та планшети. Основні принципи UX/UI-дизайну, такі як зрозумілі кнопки, логічне розташування елементів та мінімізація зайвих дій для оформлення покупки, допомагають створити комфортний досвід для користувачів.

Для створення якісного веб-дизайну необхідно враховувати технічні обмеження та особливості верстки. Дизайн повинен бути сумісним із сучасними браузерами, коректно відображатися на різних роздільних здатностях екранів та забезпечувати швидке завантаження сторінок. Також необхідно врахувати особливості роботи анімаційних ефектів та інтерактивних елементів, оскільки вони можуть впливати на продуктивність сайту.

Щоб зробити сайт привабливим, розробники використовують сучасні технології, такі як CSS-препроцесори (Sass, LESS), бібліотеки компонентів (Bootstrap, Material UI) та інструменти для роботи з графікою (Figma, Adobe XD). Також важливим є використання ефективних рішень для шрифтів, таких як Google Fonts, які забезпечують гармонійний вигляд текстового контенту.

Таким чином, інформаційне наповнення та веб-дизайн інтернет-магазину повинні гармонійно поєднувати естетику, зручність користування та технічну ефективність, щоб створити комфортний досвід для покупців та підвищити конверсію.

РОЗДІЛ 3. ВИБІР ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЙ РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНУ З ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ

3.1 Загальний огляд технологічного стеку

Розробка інтернет-магазину мобільних телефонів здійснюється з використанням MERN-стеку (MongoDB, Express.js, React, Node.js). Ця технологія є одним із найпопулярніших рішень для створення веб-застосунків, оскільки забезпечує ефективну взаємодію між клієнтською та серверною частинами, а також підтримує використання JavaScript на всіх етапах розробки.

MERN є модифікованою версією стеку MEAN, який було представлено інженерами MongoDB у 2013 році. Назва MEAN є абревіатурою, що складається з технологій: "M" – MongoDB, "E" – Express, "A" – AngularJS, "N" – Node. У MERN-стеку фреймворк AngularJS замінено на React для створення користувацького інтерфейсу, що дозволяє ефективно поєднувати JavaScript і JSON у веб-розробці [15].

У MERN-стеку MongoDB використовується як документо-орієнтована база даних, Express – як веб-фреймворк і сервер, React – як бібліотека для клієнтської частини, а Node.js – як середовище виконання. Нижче наведено схему архітектури стеку MERN.

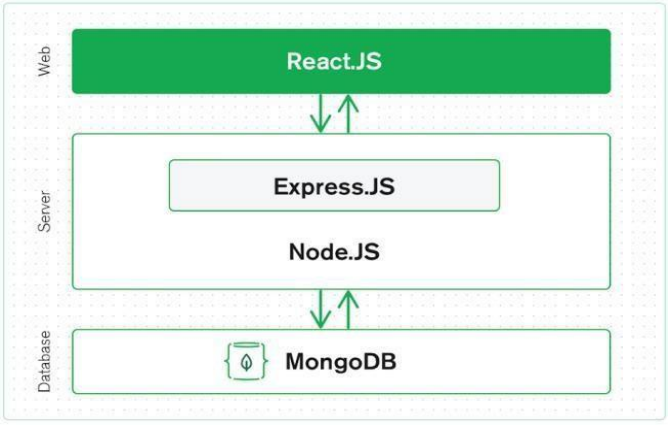

Рисунок 3.1 — Трирівнева архітектура (клієнт, сервер, база даних)

Як видно з рисунка 3.1, стек MERN є комплексним рішенням для створення додатків, яке базується на традиційній 3-рівневій архітектурі. Його складові включають рівень клієнтського відображення з React, рівень логіки додатка з Express і Node, а також рівень збереження даних у MongoDB [13].

MongoDB, вперше представлений у 2007 році, поступово став зручною технологією для розробників [13]. На відміну від SQL, який належить до структурованих мов запитів, MongoDB є представником NoSQL-баз, що використовують документоорієнтований підхід [13]. Термін NoSQL іноді інтерпретують як "не SQL" або "не тільки SQL". Основна відмінність полягає в тому, що NoSQL-бази зберігають дані у JSON-подібному форматі, на відміну від реляційних баз даних, що використовують таблиці [22].

Якщо SQL-структура організовує дані у вигляді рядків і стовпців, то NoSQL-структура впорядковує їх у вигляді документів, що містять об'єкти та масиви. У невеликих проєктах ця різниця може бути малопомітною, проте в програмах середнього рівня вибір типу бази даних стає важливим для команди розробників [22].

Згідно з офіційною документацією Express, цей фреймворк є мінімалістичним середовищем для Node, яке забезпечує гнучкий набір функцій для веб- і мобільних додатків [4]. Сам Express містить лише базові можливості, використовуючи проміжне програмне забезпечення для обробки запитів [4].

Проміжне програмне забезпечення виконує певні функції під час обробки запиту, наприклад, аутентифікацію або аналіз тіла запиту. Запит проходить через конвеєр таких модулів: перший може одразу сформувати відповідь або передати обробку наступному. Цей процес триває до останнього етапу, який завершує обробку та надсилає відповідь користувачу [18].

На наведеному рисунку показано послідовність роботи від моменту отримання запиту до його остаточної обробки та формування відповіді.

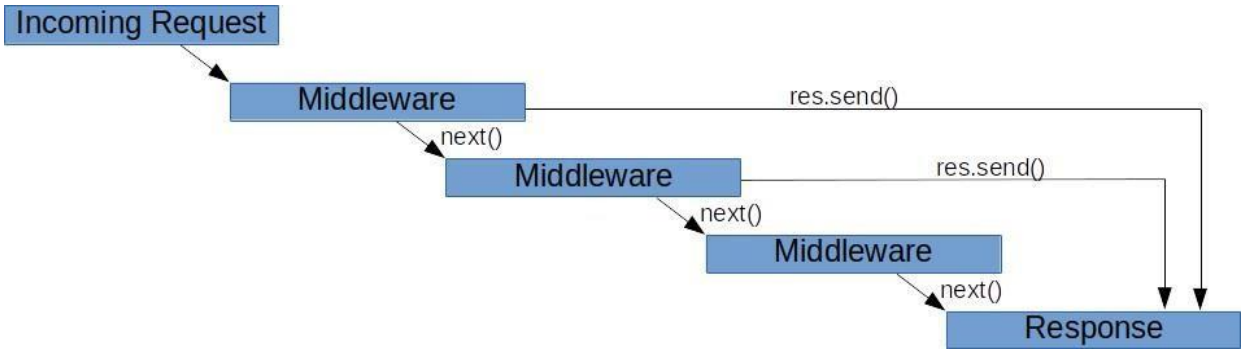

Рисунок 3.2 — Запит-відповідь через проміжне програмне забезпечення

Проміжне програмне забезпечення не лише обробляє HTTP-запити та відповіді, а й управляє послідовністю дій у циклі запит-відповідь [21]. Як зазначено вище, фреймворки проміжного програмного забезпечення, зокрема Express, можуть виконувати код, змінювати вхідні запити та результуючі об'єкти, припиняти обробку запиту або передавати управління наступному проміжному модулю у стеці. Якщо поточне проміжне програмне забезпечення не завершує обробку, слід скористатися методом next(), який передає керування наступному модулю, запобігаючи зависанню циклу запит-відповідь [18, 21]. Це дозволяє створювати гнучку та масштабовану систему обробки запитів, де кожен етап має чітко визначену функцію.

Існує кілька видів проміжного програмного забезпечення, серед яких особливу роль відіграє проміжне ПЗ рівня маршрутизатора. Воно використовується для управління маршрутами в Express і працює аналогічно до звичайного проміжного ПЗ, проте його застосування обмежується конкретним маршрутизатором Express [21]. Окрім маршрутизатора, існують також глобальні та спеціалізовані проміжні програмні модулі, які можуть виконувати завдання, такі як логування, обробка помилок, безпека або стиснення даних. Завдяки цим функціям Express є потужним інструментом для створення серверних додатків різної складності.

React — це бібліотека JavaScript, створена Facebook у 2013 році. Вона була розроблена для спрощення управління складними динамічними інтерфейсами, що оновлюються в реальному часі. React активно використовується

для розробки SPA-додатків і надає ефективні рішення для створення інтерфейсів будь-якого рівня складності. Однією з ключових функцій, що забезпечує високу продуктивність React, є віртуальний DOM (VDOM). Ця концепція дозволяє React працювати з віртуальним представленням інтерфейсу користувача, оновлюючи лише необхідні елементи в реальному DOM, що значно пришвидшує рендеринг [3]. Віртуальний DOM також мінімізує кількість безпосередніх змін у реальному DOM, що позитивно впливає на швидкість роботи застосунку та користувацький досвід.

Окрім основних принципів React, команда Facebook впровадила низку потужних можливостей, що дозволяють адаптивно масштабувати веб- та мобільні додатки. Ці функції отримали визнання серед розробників у всьому світі. У межах цієї курсової роботи розглянуто дві ключові концепції React: компоненти та хуки.

Компоненти є основою React, дозволяючи розбивати інтерфейс на незалежні та багаторазові елементи. Вони можуть бути двох типів: функціональні та класові. Найбільш оптимальним підходом до створення компонентів є функціональний підхід, у якому компоненти реалізуються як JavaScript-функції. Також можна використовувати класовий підхід на основі класів ES6. В обох випадках, з точки зору React, ці методи є еквівалентними [12].

Функціональні компоненти є легшими та зручнішими у використанні завдяки можливості застосування React-хуків, що дозволяють управляти станом і життєвим циклом компонента без необхідності в класах.

```
function Welcome(props) {
  return (
    <div>
        <p>Hello {props.name} </p>
    </div>
  );
}
```

Рисунок 3.3 — Функціональний компонент

```
class Welcome extends React.Component {
  render() {
    return (
      <div>
        <p>Hello {this.props.name}</p>
    </div>
    )
  }
}
```

Рисунок 3.4 — Класовий компонент

Як видно на малюнках 3.3 та 3.4, функціональні компоненти та компоненти класу можуть давати однакові результати. Відображається компонент "p" зі словом "Hello..." та пропсом "name", що передається в компонент. Компоненти можуть посилатися один на одного, оскільки один компонент може бути батьківським, містячи безліч інших дочірніх компонентів без обмежень на рівень деталізації. Незалежно від типу (клас чи функціональний компонент), вони обидва підпорядковуються єдиному правилу, яке встановлено React: всі компоненти React є чистими функціями, які не змінюють своїх властивостей. Пропси — це набір входів, які передаються як параметри компоненту, а чиста функція демонструє випадок, коли функція виконує логіку без зміни аргументів. Таким чином, компонент React функціонує як чиста функція, яка поважає свої вхідні дані та завжди повертає однакові результати для тих самих пропсів [19].

Хоча передача глобального стану через Redux виглядає продуктивно, управління локальним станом в компоненті Redux вважається зайвим і непотрібним. Класи React з самого початку ефективно підтримують локальні стани з чіткою синтаксичною структурою, що залишається актуальним і сьогодні, незалежно від масштабу проекту. Крім того, альтернативою, яка виконує ту ж функцію, що й класи React, є React Хуки, які були представлені на React Conf 2018 [19]. Ця зміна є поступовим переходом від класичних класів React,

оскільки хуки не замінюють і не включають нові концепції, пов'язані з властивостями, станом і життєвими циклами компонентів.

Введення Хуків не означає відмову від класів React. Розробники та менеджери проектів можуть вільно вибирати, чи хочуть вони спробувати щось нове, чи залишатися на старому синтаксисі. Спочатку хуки можуть здаватися заплутаними, але зрештою, логіка та мета залишаються близькими до основних ідей класів. На практиці можна стверджувати, що React Hooks зменшили кількість рядків коду та усунули використання ключового слова "this". Насправді, є більше відмінностей, ніж просте зменшення загальної кількості рядків коду та зміна вигляду програми.

React Hooks представляють хуки стану, відомі як useState, які обробляють управління станом компонентів. Точніше, useState — це хук, який ініціалізує змінну стану, що зберігається у React. Цей хук отримує та повертає два значення: поточний стан та функцію для його зміни. Завдяки хуку useState стан компонента можна легко ініціалізувати, використовувати та оновлювати [19]. Ще один важливий хук — це useEffect, який допомагає програмістам обробляти життєві цикли компонентів. Проблема розподілу логіки та даних на кілька життєвих циклів класу, таких як componentDidMount, componentDidUpdate, componentWillUnmount, була чітко висвітлена в Effect Hook. Компонент React може включати кілька ефектів для розділення проблем, пов'язаних із маніпуляцією даними.

Те, чого не вистачає класам React, а хуки можуть забезпечити, — це спільна логіка функціональності. Раніше програмісти використовували компоненти вищого порядку (HOC) або шаблони пропсів для поділу логіки стану, що призводило до необхідності налаштування ієрархії компонентів і ускладнювало підтримку програми. Натомість, хуки дозволяють розробникам повторно використовувати логіку, не змінюючи структуру компонентів.

Хоча технологія була представлена 12 років тому, Node.js зарекомендував себе як важливе середовище виконання JavaScript, яке підтримує серверний JavaS-

cript [1]. Це не є ні мовою, ні фреймворком, але потужний інструмент, створений на базі двигуна V8 Chrome, який працює з JavaScript [21]. Використовуючи асинхронну подієву модель I/O, Node дозволяє розробникам створювати легкі додатки в реальному часі, окрім стандартного оброблення запитів.

Для розробників, які зосереджені на JavaScript, Node.js пропонує переваги в створенні програм, включаючи серверну та клієнтську частини. Зокрема, варто згадати менеджер пакетів Node або npm, який забезпечує доступ до тисяч пакетів, зареєстрованих у системі Node [1]. Реєстр npm вважається одним із найбільших у світі, у 2017 році було зафіксовано понад 350 тисяч пакетів, багато з яких є відкритим кодом, розроблені програмістами з усього світу [1].

Стек MERN не є єдиним у списку технологій для розробки веб-додатків. Іншими відомими стеками є MEAN (MongoDB, Express, Angular, Node.js), LAMP (Linux, Apache, MySQL або MongoDB, PHP), Django (Python, Django, Apache, MySQL) та багато інших. Кожен з цих стеків має свої унікальні особливості, що дозволяє вибрати найбільш підходящий варіант залежно від вимог проекту.

Є багато причин, чому стек MERN є популярним і схваленим вибором для створення веб-додатків. По-перше, він забезпечує можливість роботи з єдиною кодовою базою, використовуючи JavaScript та JSON, що дозволяє розробникам заглиблюватися у певну мову програмування та покращувати командну співпрацю. Це сприяє зменшенню часу на навчання нових членів команди, оскільки всі працюють з однією мовою.

По-друге, стек MERN забезпечує відмінну продуктивність завдяки асинхронним можливостям Node.js, що дозволяє обробляти численні запити одночасно. Це особливо корисно для веб-додатків, які потребують високої швидкості та ефективності.

Стек також сприяє скороченню часу на розробку, спрощенню прогресу, легкому розширенню та підтримці програми. Коли розробники можуть використовувати один стек для всіх компонентів, це спрощує процес інтеграції та

зменшує ймовірність виникнення помилок. Підтримка відкритого коду та активна спільнота забезпечують доступ до великої кількості ресурсів і інструментів, які можуть допомогти розробникам вирішувати проблеми та підвищувати якість коду.

Що не менш важливо, кожен компонент стека MERN підтримується великою спільнотою, яка активно вносить свій внесок у розвиток технологій та підтримку програмістів на будь-якому етапі. Існує безліч матеріалів з відкритим кодом, таких як документація, спеціальні пакети та додаткові бібліотеки, що робить процес розробки більш доступним і ефективним. З огляду на всі ці переваги, стек MERN залишається одним з найбільш популярних варіантів для створення сучасних веб-додатків [14].

3.2 Обгрунтування вибору мови програмування

JavaScript є універсальною мовою, яка використовується як у клієнтській частині (фронтенді), так і на сервері (бекенді) через Node.js. Використання JavaScript у проєкті інтернет-магазину забезпечує такі переваги:

а) Єдина мова для всієї архітектури: Використання JavaScript на фронтенді (React) та бекенді (Node.js) спрощує розробку та підтримку коду. Завдяки цьому можна використовувати одні й ті ж принципи та інструменти на всіх рівнях застосунку;

б) Швидкість та продуктивність: JavaScript є мовою з інтерпретованою природою, що дозволяє швидко виконувати код без потреби в компіляції. Використання асинхронного введення/виводу (через async/await, Promises, callbacks) значно підвищує продуктивність, особливо для серверних операцій;

в) Популярність та велика спільнота: JavaScript є однією з найпоширеніших мов програмування у світі (за даними GitHub та Stack Overflow). Велика кількість навчальних матеріалів, документації та активної спільноти спрощує навчання та вирішення проблем;

г) Широкий вибір бібліотек та фреймворків: React, Angular, Vue.js — для створення швидких та інтерактивних інтерфейсів. Express.js — для написання бекенд-логіки та обробки API-запитів. Mongoose — для взаємодії з MongoDB через JavaScript;

д) Гнучкість та кросплатформність: JavaScript використовується не тільки для веб-розробки, а й у мобільних застосунках (React Native), серверних додатках (Node.js), а також у роботі з базами даних;

е) Масштабованість: Завдяки використанню Node.js веб-застосунки на JavaScript можуть легко масштабуватися для обслуговування великої кількості запитів. Використання мікросервісної архітектури дозволяє розподіляти навантаження між кількома серверами;

ж) Підтримка сучасних стандартів: Завдяки ECMAScript 6+ (ES6) JavaScript отримав такі можливості, як стрілкові функції, let/const, деструктуризацію, async/await, модулі import/export, що робить код більш чистим і зручним.

Попри численні переваги, JavaScript має певні недоліки, які варто враховувати під час розробки:

а) Однопоточна природа: На відміну від багатопотокових мов (наприклад, Java, C++), JavaScript працює в одному потоці. Це може викликати проблеми при обробці складних розрахункових операцій. Хоча в Node.js є асинхронна модель обробки, вона не завжди підходить для обчислювально інтенсивних завдань;

б) Відсутність статичної типізації: JavaScript є динамічно типізованою мовою, що може призводити до непередбачуваних помилок у великих проєктах. Для вирішення цієї проблеми часто використовують TypeScript — надбудову над JavaScript із підтримкою статичної типізації;

в) Різна реалізація в браузерах: Незважаючи на стандартизацію ES6+, деякі браузери можуть мати власні особливості в реалізації JavaScript, що

може призводити до несумісності. Для виправлення цієї проблеми розробники використовують Babel — компілятор, який перетворює нові версії JavaScript у старі для сумісності;

г) Безпека: JavaScript-код виконується на клієнтському боці, що робить його вразливим до XSS-атак (Cross-Site Scripting). Щоб уникнути проблем, необхідно ретельно перевіряти вхідні дані, використовувати JWT (JSON Web Token) для авторизації та HTTPS для захисту переданих даних.

JavaScript є ідеальним вибором для розробки веб-застосунків завдяки своїй гнучкості, швидкості та широкому використанню. Його переваги, такі як можливість використання на всіх рівнях застосунку, велика екосистема бібліотек та популярність серед розробників, роблять його найкращим варіантом для створення інтернет-магазину.

Хоча у мови є певні недоліки (відсутність статичної типізації, ризики безпеки, однопоточність), вони компенсуються можливістю використання TypeScript, асинхронного програмування та належних заходів безпеки.

Обраний технологічний стек MERN (MongoDB, Express.js, React, Node.js) дозволяє ефективно використовувати всі переваги JavaScript та забезпечує швидку, продуктивну та масштабовану розробку інтернет-магазину мобільних телефонів.

3.3 Інструменти для розробки та CI/CD

Розробка веб-застосунків вимагає використання сучасних інструментів, які спрощують процес написання, тестування, розгортання та підтримки програмного забезпечення. Для забезпечення ефективного робочого процесу було обрано набір інструментів, які сприяють продуктивності, автоматизації та зручності командної роботи:

а) Visual Studio Code – основне середовище розробки з підтримкою розширень для роботи з React, Node.js та MongoDB. Має вбудований термінал, дебагер і можливість налаштування під конкретні завдання;
б) Git + GitHub – система контролю версій, яка дозволяє зберігати зміни у коді, працювати над проєктом у команді та автоматизувати процеси розгортання;
в) Swagger – використовується для тестування API-запитів, аналізу відповідей сервера та налагодження взаємодії між клієнтською і серверною частинами;
г) MongoDB Compass – графічний інструмент для взаємодії з базою даних MongoDB, що дозволяє переглядати, змінювати та аналізувати дані у зручному інтерфейсі;
д) ESLint + Prettier – допомагають підтримувати єдиний стиль коду, запобігати помилкам на ранніх етапах і забезпечують стандартизацію написання коду в команді;
е) Mongoose – бібліотека для взаємодії з MongoDB, яка спрощує роботу з базою даних.

Для автоматизації процесів розгортання, тестування та оновлення застосунку використовуються наступні інструменти:

а) GitHub Actions – автоматизує процес перевірки коду, тестування та розгортання. Допомагає запускати перевірки стилю коду, виконувати тести та автоматично оновлювати застосунок;
б) Render – використовується для хостингу та автоматичного розгортання застосунку (Node.js + Express.js);
в) React Testing Library – застосовується для тестування функціональності фронтенду та забезпечення стабільності роботи застосунку після внесення змін.

Застосування сучасних інструментів у розробці веб-застосунків значно підвищує продуктивність, забезпечує ефективну командну роботу та мінімізує

ризик технічних помилок. Використання Visual Studio Code як основного середовища розробки дозволяє швидко писати та налагоджувати код, а Git і GitHub гарантують зручне управління версіями, що є критично важливим у командній роботі. Завдяки Swagger та MongoDB Compass можна легко тестувати API-запити та взаємодіяти з базою даних, що суттєво спрощує процес розробки та налагодження бекенду.

Автоматизація є важливим аспектом ефективного робочого процесу. Інструменти, такі як GitHub Actions і Render, дозволяють швидко розгортати зміни та уникати помилок під час оновлення застосунку. Це значно зменшує час між внесенням змін у код і їхнім впровадженням у продакшн. ESLint і Prettier забезпечують стандартизацію коду, що робить його чистим, зрозумілим і легким у підтримці. Для перевірки стабільності застосунку використовується React Testing Library, що допомагає уникнути багів при впровадженні нових функцій або змін у наявному коді.

Використання Mongoose як бібліотеки для роботи з MongoDB дозволяє значно спростити взаємодію з базою даних, підвищуючи ефективність бекенду. Поєднання Express.js і MongoDB у розгортанні на Render забезпечує гнучкість та масштабованість інтернет-магазину, що є важливим фактором для його подальшого розвитку.

Усі ці інструменти та підходи не лише прискорюють розробку, а й підвищують надійність роботи застосунку. Вони дають змогу оперативно впроваджувати оновлення, тестувати код на ранніх етапах і підтримувати високу якість розробки. Завдяки автоматизації тестування та розгортання зменшується ризик збоїв у роботі сайту, що позитивно впливає на досвід користувачів і загальну конверсію.

Таким чином, використання сучасного технологічного стеку та інструментів для розробки, тестування й автоматизації дає змогу створити стабільний, зручний і масштабований інтернет-магазин. Це забезпечує високу продуктивність розробницького процесу та гарантує якість кінцевого продукту, що є ключовим фактором успіху онлайн-бізнесу.

РОЗДІЛ 4. ДОСЛІДНА ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРНЕТ МАГАЗИНУ

4.1 Опис програмної реалізації

Інтернет магазин було розроблено з використанням сучасних веб-технологій, які забезпечують високу продуктивність, інтерактивність і зручність для кінцевого користувача. У процесі створення було детально проаналізовано вимоги до системи, на основі яких сформовано архітектуру, підібрано відповідні технології та реалізовано всі функціональні модулі.

Архітектура сайту базується на клієнт-серверній моделі, яка є однією з найпопулярніших і найзручніших для сучасних веб-додатків. Серверна частина проєкту використовується для зберігання статичних ресурсів, таких як HTML, CSS, JavaScript, а також для зберігання основного контенту у форматі JSON. У якості серверного середовища було обрано платформу GitHub Pages, що забезпечує безкоштовне та стабільне розгортання проєкту. Це рішення є оптимальним для статичних сайтів, оскільки воно гарантує швидке завантаження ресурсу незалежно від кількості одночасних користувачів. Завдяки такому підходу, користувачі отримують доступ до сторінок сайту практично миттєво.

Клієнтська частина реалізована за допомогою бібліотеки React, яка стала основним інструментом для створення інтерфейсу користувача. Використання React дозволило розділити інтерфейс на компоненти, що значно спростило процес розробки, тестування та підтримки коду. Усі компоненти були створені так, щоб вони могли повторно використовуватись у різних частинах програми. Наприклад, список телефонів, пошуковий рядок та кошик реалізовані у вигляді окремих модулів, які інтегруються між собою через головний компонент програми. Такий підхід забезпечує зручну навігацію та високу гнучкість при зміні чи доповненні функціональності.

Сайт підтримує сучасний підхід до розробки веб-додатків — Single Page Application (SPA). Це означає, що вся сторінка завантажується один раз, а подальша навігація між різними частинами сайту виконується без перезавантаження. Це значно покращує користувацький досвід, оскільки сторінки відкриваються швидше, а взаємодія з сайтом стає плавною та безперервною. Для забезпечення SPA-функціональності було використано бібліотеку React Router, яка відповідає за маршрутизацію в межах додатка. Вона дозволяє користувачеві переміщуватися між головною сторінкою, списком товарів, детальною інформацією про телефони та кошиком без необхідності повторного завантаження сторінок.

Особливу увагу було приділено адаптивному дизайну сайту. Інтерфейс розроблений таким чином, щоб автоматично підлаштовуватися під розмір екрана користувача. Це забезпечує комфортне користування сайтом як на великих моніторах, так і на мобільних пристроях чи планшетах. Для реалізації адаптивності використовувалися CSS Media Queries, що дозволяють змінювати стилі елементів залежно від роздільної здатності екрана. Крім того, для кожного елемента інтерфейсу були підібрані оптимальні розміри, шрифти та пропорції, що робить сайт візуально привабливим і зручним для читання.

Ще однією важливою особливістю програмної реалізації є оптимізація продуктивності. Усі статичні файли, включаючи зображення та JavaScript-код, були попередньо мінімізовані, що дозволило зменшити розмір ресурсів і, відповідно, прискорити завантаження сторінок. Наприклад, зображення телефонів були стиснуті без втрати якості, а JavaScript-код зменшений за допомогою інструменту Webpack. Крім того, для зменшення навантаження на браузер користувача було застосовано методи lazy-loading, які дозволяють завантажувати контент лише тоді, коли він стає видимим у полі зору. Інтернет магазин побудований з акцентом на простоту використання, швидкість та інтуїтивно зрозумілий інтерфейс. Розроблений підхід до архітектури, використані технології та реалізовані функції дозволяють забезпечити стабільну та комфортну

роботу для всіх категорій користувачів. Клієнт може легко знаходити необхідну інформацію, переглядати деталі товарів, сортувати їх за ціною або назвою, а також додавати моделі до кошика для подальшого перегляду. Ці особливості роблять сайт конкурентоспроможним на ринку веб-додатків і забезпечують високу якість обслуговування користувачів.

4.1.1 Архітектура інтернет-магазину

Архітектура присутня у кожного веб-додатку, це певна структура яка визначає роль компонентів як вони взаємодіють між собою та як розподілені в мережі. Для віртуального навчального середовища я обрав клієнт-серверну архітектуру. Основними принципами цієї архітектури є розподілення обов'язків де клієнти відповідають за відображення інтерфейсу та взаємодію з користувачем, а сервер обробляє запити та зберігає дані в базі, комунікація між компонентами веб-додатку здійснюється за допомогою різних протоколів, найчастіше використовується HTTP та TCP/IP. Важливо також зазначити те що завдяки цій архітектурі можливо масштабовувати систему та розподіляти навантаження між декількома серверами.

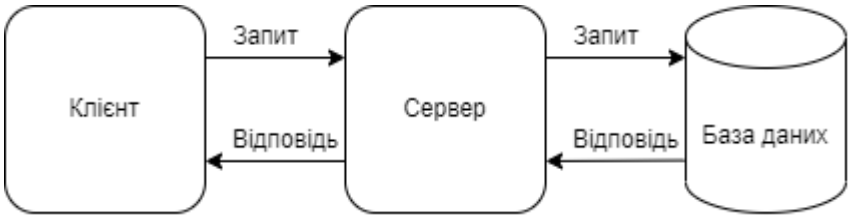

Рисунок 4.1 – Архітектура веб-додатку

Аналізуючи рисунок 4.1 видно що сервер відіграє головну роль, він отримує та обробляє різні запити від авторизації до змін інформації про користувачів. Дана архітектура широко використовується у веб-додатках та не потребує особливих технічних знань для її втілення в систему.

4.1.2 Реалізація серверної частини

Для розробки фронтальної частини було обрано сучасний фреймворк ReactJS, оскільки він дозволяє створювати односторінкові застосунки (SPA), що підвищує швидкість роботи сайту та забезпечує зручний підхід до реалізації. Першим етапом став вибір дизайну проекту інтернет-магазину та його технічного завдання (ТЗ). Після цього почалась розробка проекту. Аналізуючи дизайн сайту, було виділено основні компоненти, на які можна розподілити сайт. Критерії для виділення фрагменту в якості компонента: а) Відносна простота, що передбачає невелику кількість елементів; б) Використання в кількох місцях.

Після такого поділу та вивчення дизайну і ТЗ стало зрозуміло, яку архітектуру доцільно закласти в сайт. У якості бази даних було обрано MongoDB як NoSQL базу, завдяки її простоті, можливості безкоштовного використання на початковому етапі (розробки) та чудовому допоміжному ПЗ Compass.

Після вибору бази даних і визначення шляху побудови архітектури було створено сам проект, для чого використано готовий шаблон React за допомогою команди npx create-react-app phone-catalog. В результаті виконання цієї команди було отримано шаблон для подальшої роботи (див. рисунок 4.2).

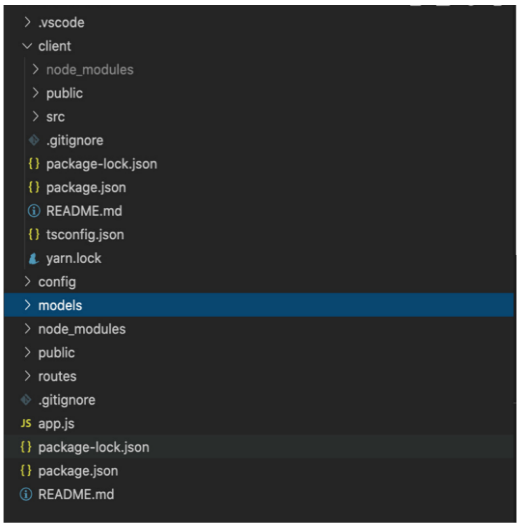

Рисунок 4.2 — Стуктура тек проекту

Після цього була створена база даних, яка була підключена до проекту. Для цього було зареєстровано користувача на сайті https://www.mongodb.com/, а потім створено нову базу даних. Після цього я скопіював адресу для підключення до неї і налаштував Compass (див. рис. 4.3).

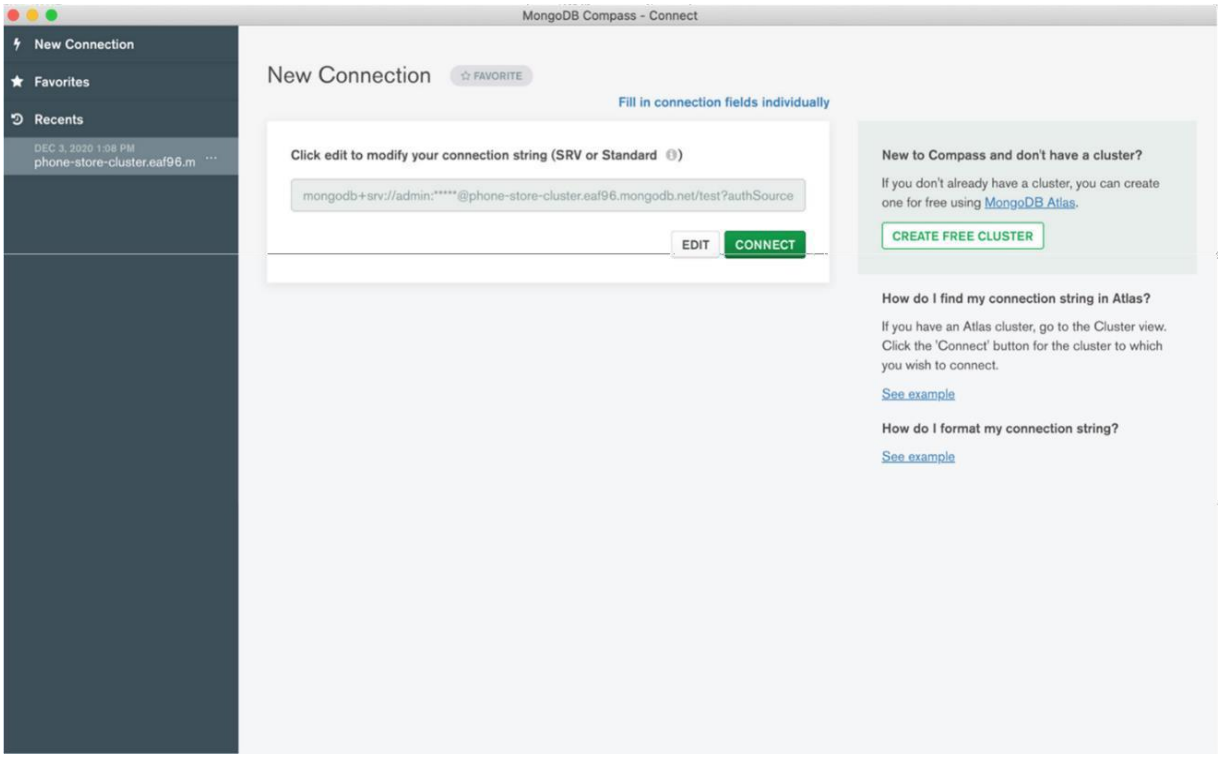

Рисунок 4.3 — Головний інтерфейс MongoDB Compass

Далі я створюю колекції для телефонів, планшетів, гарячих пропозицій і кращих товарів (див. рис. 4.4). Колекції — це логічно та змістовно відокремлені групи в межах однієї бази даних. Вони можуть бути пов'язані між собою за допомогою id або будь-якого іншого поля.

Рисунок 4.4 — Колекції для категорій товарів

Потім я під'єднав базу даних у коді, використавши залежність config для

зручного зберігання постійних даних. Я створив папку config і додав туди адресу для доступу до своєї бази даних, номер порту, на якому буде розміщуватися сайт під час розробки, а також секретний ключ на випадок, якщо знадобиться шифрувати особисті дані (див. рис. 4.8).

config > {} default.json > ...

```
{
    "port": 5000,
    "jwtSecret": "SenyaSecretKey",
    "mongoUri": "mongodb+srv://admin:admin@phone-store-cluster.eaf96.mongodb.net/phone-list?retryWrites=true&w=majority"
}
```

Рисунок 4.8 — Основні константи

Для ефективного використання MongoDB необхідно чітко визначити структуру бази даних, яка зображена на рисунку 4.9.

Рисунок 4.9 — Структура бази даних

Тепер ми переходимо до важливого етапу — створення самої схеми бази даних та її експорту, що ілюструється на рисунку 4.10.

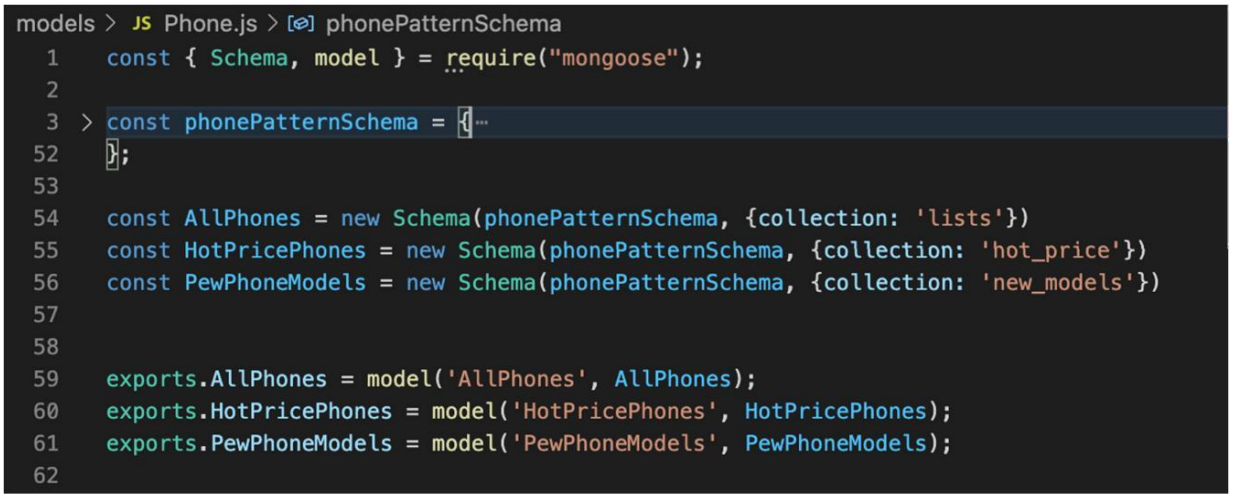

Рисунок 4.10 — Ініціалізація схеми бази даних

Тепер за допомогою бібліотеки Express налаштовуємо маршрути на стороні бекенду, що зображено на рисунку 4.11.

```
app.use("/public", express.static("public"));

app.use('/api/auth', require('./routes/auth.routes'))

app.use('/api/phone', require('./routes/phone.routes'))
app.use('/api/tablet', require('./routes/tablets.routes'))
```

Рисунок 4.11 — Налаштування основних маршрутів

Тепер база даних готова, а маршрути для запитів на сервер написані, тож залишилося реалізувати фронтову частину. Розробку фронтової частини починаємо з підключення Redux до проекту, оскільки він значно спрощує управління сховищем даних для сайту. Після цього необхідно додати екшени та редюсери, щоб ефективно керувати станом Redux (рис. 4.12 – рис. 4.13).

```
export const phoneLoading = (loading: boolean): AppStateActionTypes => ({
    type: PHONE_LIST_LOADING,
    loading
});
```

Рисунок 4.12 — Приклад дії для завантаження моделі телефону

```
export const phonesState = (state = initialState, action: AppStateActionTypes) => {
    switch (action.type) {
        case PHONE_LIST_LOADING:
            return { ...state, loading: action.loading }

        case PHONE_LIST_SUCCESS:
            return { ...state, phoneList: action.phoneList }

        case PHONE_LIST_ERROR:
            return { ...state, error: action.error }

        case PHONE_LIST_STATE:
            return { ...state, phoneListState: action.phoneListState }

        case PHONE_ITEM_SUCCESS:
            return { ...state, currentModel: action.currentModel }

        default:
            return state
    }
}
```

Рисунок 4.13 — Приклад редюсера для зберігання стану телефонів

Тепер можна перейти до реалізації основних компонентів, зокрема хедеру, футеру та карток товарів. Ці компоненти є ключовими елементами інтерфейсу, які забезпечують зручність навігації та відображення товарів для користувачів.

4.1.3 Реалізація клієнтської частини

Одразу створюємо два варіанти: один для версії з великим екраном, що перевищує 700 px, а інший — для мобільних пристроїв і планшетів (рис. 4.14). Це дозволяє забезпечити оптимальний вигляд і функціональність сайту на різних типах пристроїв.

Рисунок 4.14 — Структура десктопного і мобільного хедеру

Після цього була реалізована функція, яка відстежує ширину екрану в даний момент і передає цю інформацію в Redux. Завдяки цьому, весь додаток миттєво реагує на зміни ширини екрану, що дозволяє нам визначати, який хедер показати в залежності від розміру екрану (рис. 4.15 – рис. 4.16).

```
// check device screen width
const checkDeviceSize = (event?: any) => {
  const screenWidth = event ? event.currentTarget.innerWidth : window.innerWidth;

  store.dispatch(setDeviceScreen({
    value: screenWidth,
    name: screenWidth > 0 && screenWidth <= 500
      ? 'phone'
      : screenWidth > 500 && screenWidth <= 900
      ? 'tablet'
      : 'desktop'
  }))
}
```

Рисунок 4.15 — Визачення розміру екрану

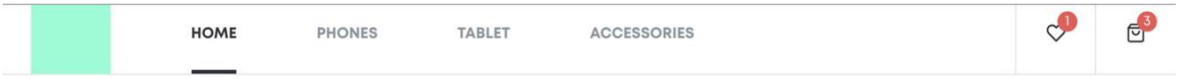

Рисунок 4.16 — Десктопний хедер

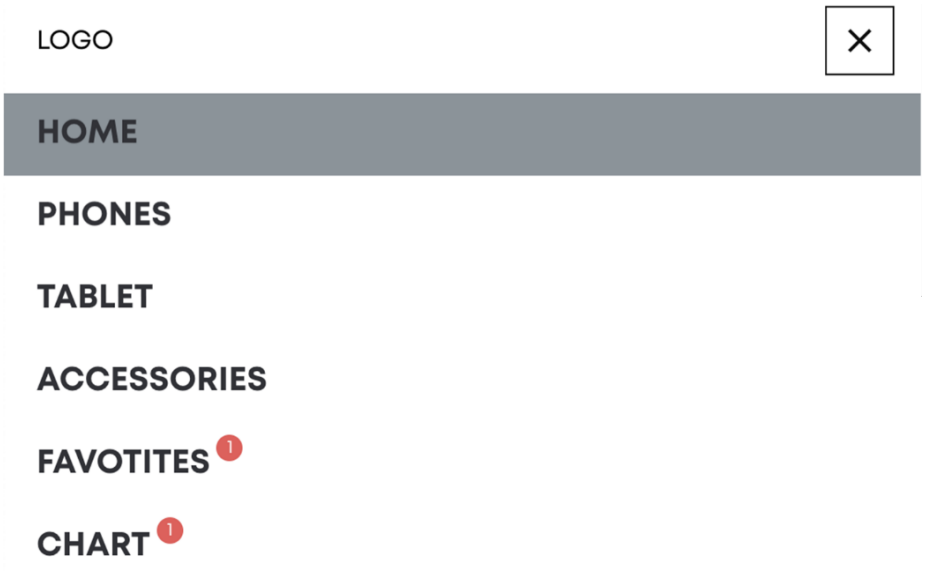

Рисунок 4.17 — Мобільний хедер

Важливим елементом є можливість пошуку потрібних товарів на сайті,

тому необхідно інтегрувати пошукове меню у хедер сайту. Однак, враховуючи адаптивність сайту та довжину пошукового меню, потрібно створити кілька варіантів розміщення поля пошуку в хедері для малих і великих розмірів екрану (рис. 4.17 – рис. 4.22).

Рисунок 4.18 — Компонент пошуку

Рисунок 4.19 — Хедер для великих екранів

Рисунок 4.20 — Хедер для середніх екранів

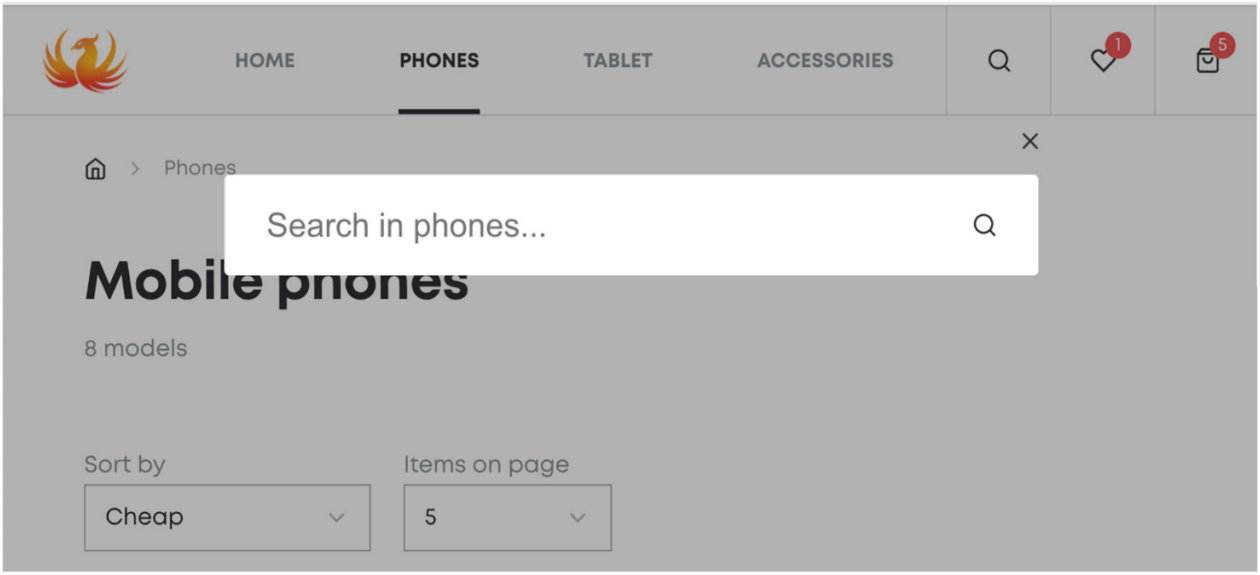

Рисунок 4.21 — Відкриття додатково поля для пошуку на середніх екранах

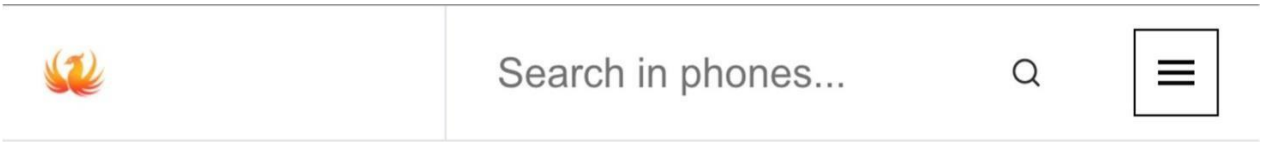

Рисунок 4.22 — Пошук на малих пристроях

Тепер настав час зайнятися футером. У цьому випадку немає потреби створювати два окремі варіанти для мобільних пристроїв і великих моніторів, оскільки тут не так багато елементів, і всі вони є еластичними. Тож можна зосередитися на покращенні адаптивності (рис. 4.23 – рис. 4.24).

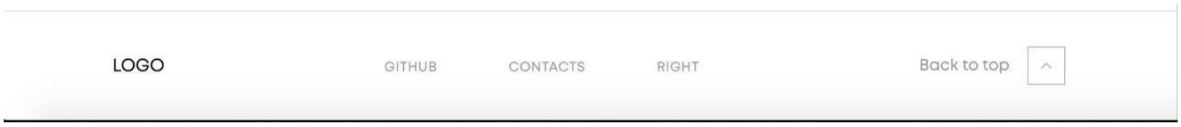

Рисунок 4.23 — Футер на великих екранах

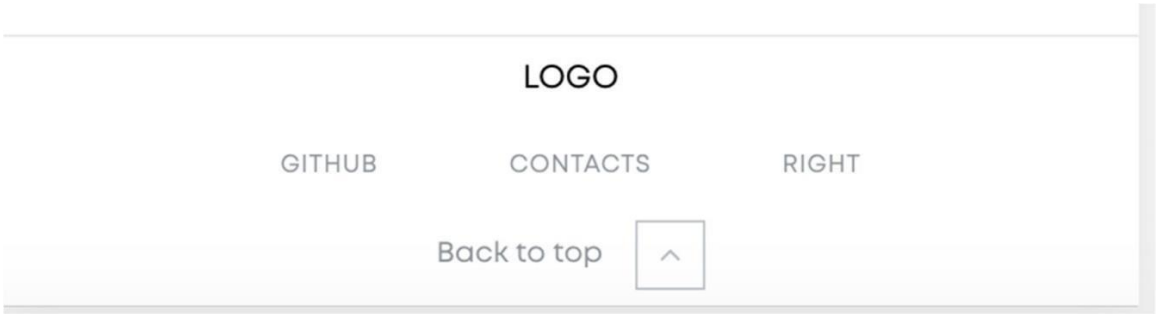

Рисунок 4.24 — Футер на малих екранах

На головному екрані повинен бути розміщений слайдер, але не такий, як для товарів, а простий слайдер для зображень. Тож потрібно внести деякі зміни у висоту кнопок "вперед" і "назад" (рис. 4.25).

Рисунок 4.25 — Слайдер для зображень

У дизайні головної сторінки передбачено використання слайдерів, тому доцільно спочатку створити шаблон для цього елемента. Однак, перед цим необхідно розробити компонент картки товару (рис. 4.26).

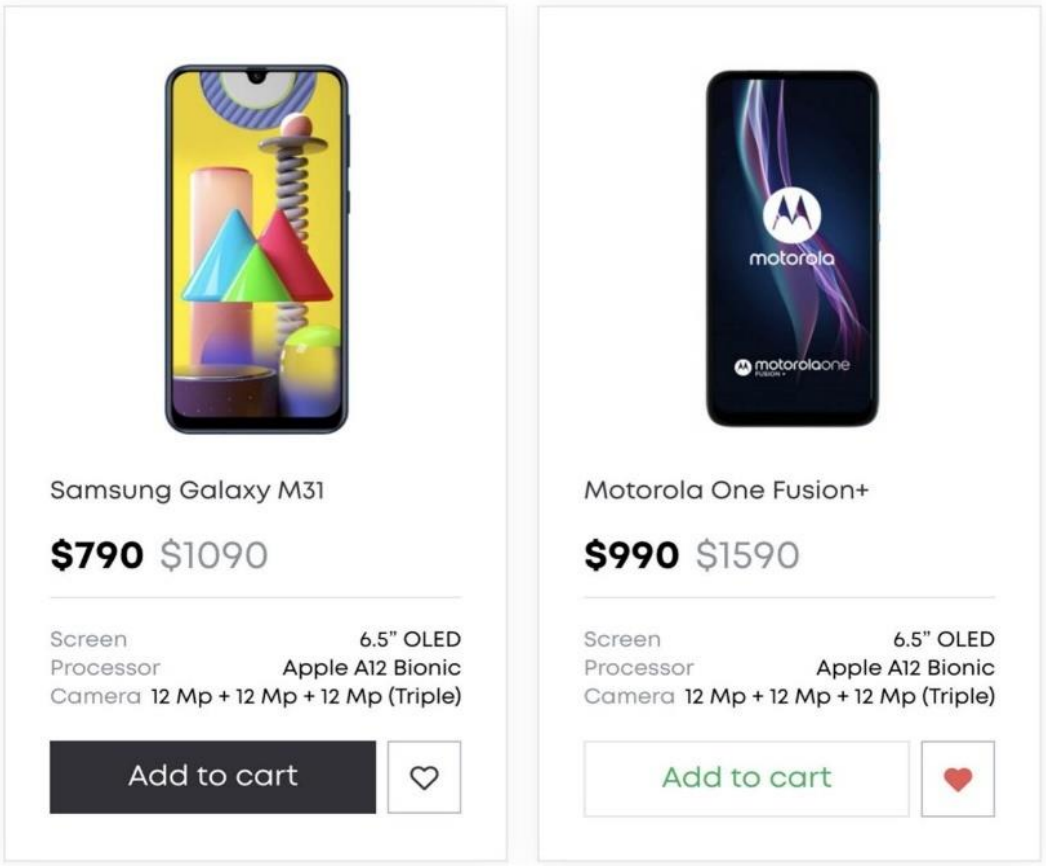

Рисунок 4.26 — Картки товару

Щоб реалізувати карусель (слайдер) на основі карток товарів, було використано бібліотеку Swiper. Основний принцип роботи полягає в тому, що Swiper отримує дані для рендерингу карток і автоматично створює інтерактивний слайдер. Проте, щоб зробити його більш зручним для користувачів, необхідно додати додаткові елементи керування, такі як кнопки для перемикання між товарами, а також можливість навігації за допомогою жестів на сенсорних пристроях. Після цих доробок слайдер стає повноцінним інструментом для зручного перегляду товарів (рис. 4.27).

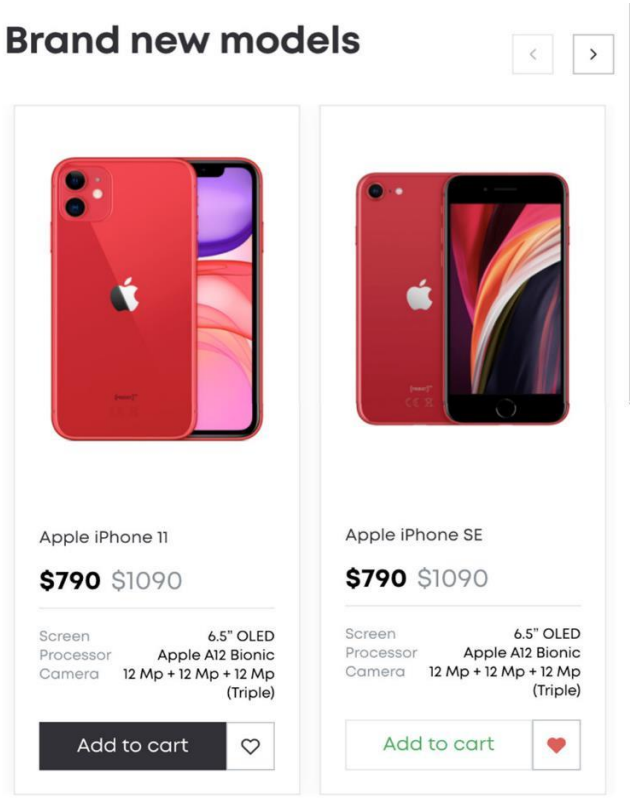

Рисунок 4.27 — Слайдер для телефонів

Окрім слайдера, згідно з дизайном головної сторінки сайту, необхідно додати список плиток із категоріями товарів (рис. 4.28).

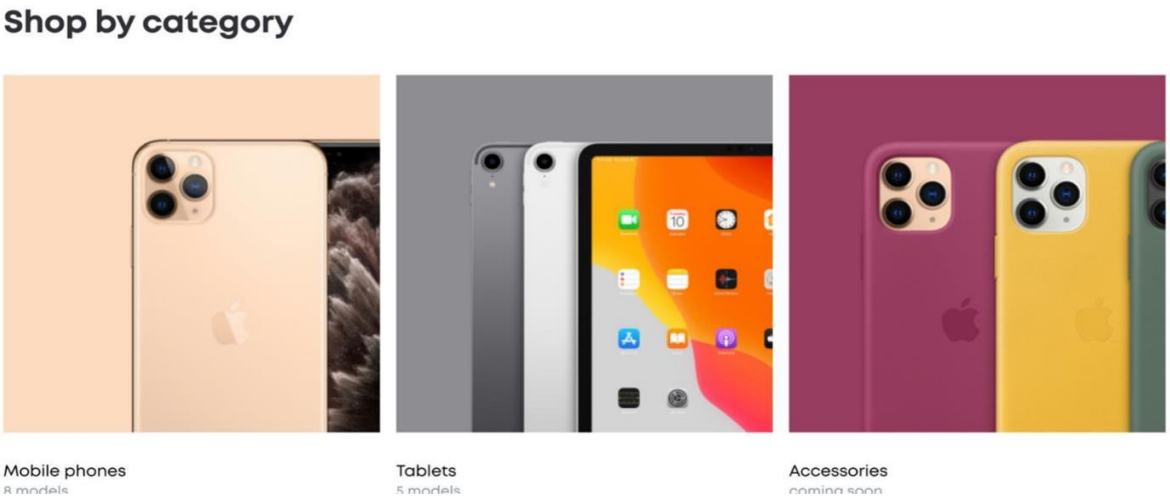

Рисунок 4.28 — Плитки на головній сторінці

Сторінка обраних товарів, які користувач відзначив під час перегляду сайту, міститиме хедер, футер і список товарів у вигляді плиток (рис. 4.29).

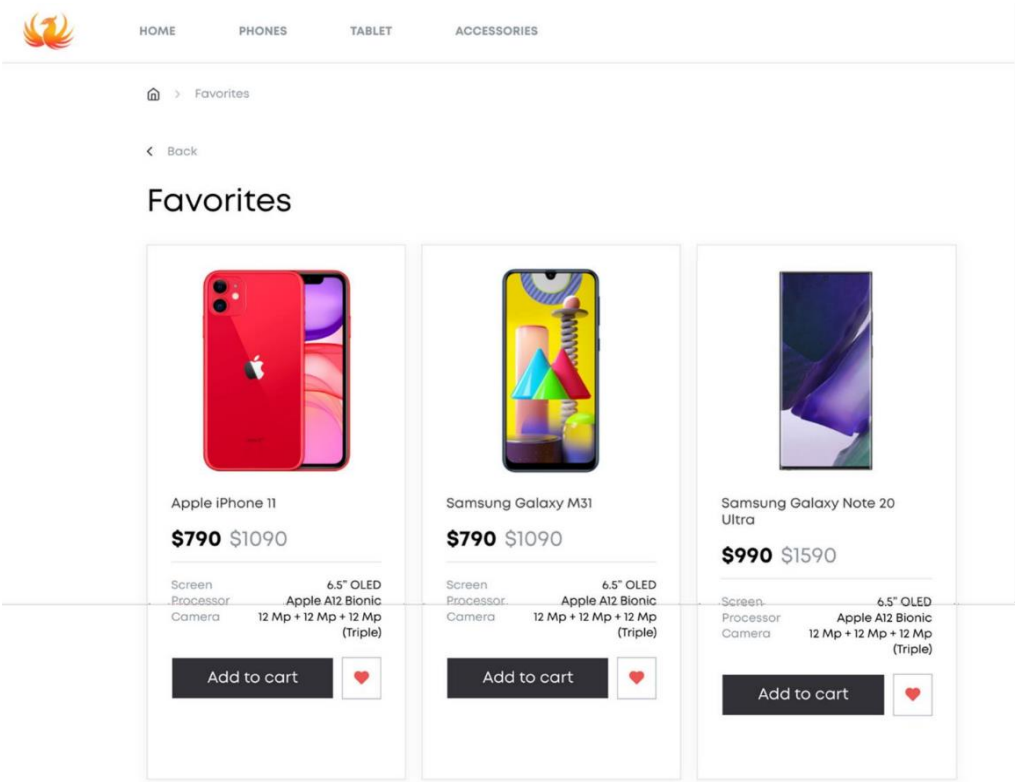

Рисунок 4.29 — Сторінка улюблених товарів

Крім того, для зручнішої навігації додамо міні-карту, яка допоможе користувачеві розуміти, де він знаходиться. Початковою точкою буде головна сторінка, а далі відображатиметься шлях, який було пройдено. На рисунку 4.30 показано приклад, коли користувач перейшов на сторінку з усіма телефонами, обрав модель Sansub Galaxy Note 20 і відкрив сторінку конкретного товару. Як видно з рисунка, остання точка шляху неактивна, оскільки користувач уже на ній знаходиться, тоді як решта залишаються активними, і, натиснувши на них, можна швидко повернутися до відповідної сторінки.

Рисунок 4.30 — Допоміжна навігація

Далі створимо сторінку для планшетів, яка матиме таку ж структуру, як і сторінка для телефонів, а також буде схожою на сторінку для аксесуарів (рис. 4.31).

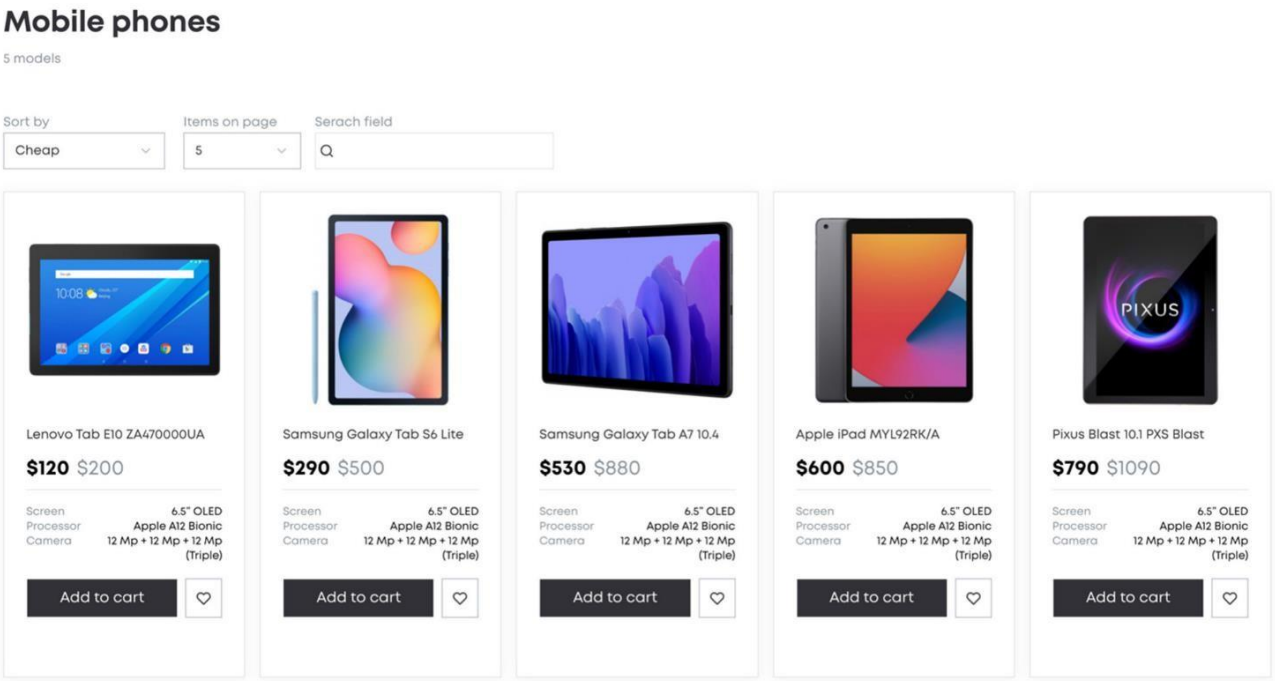

Рисунок 4.31 — Сторінка планшетів

Коли користувач заходить на сторінку, необхідно завантажити список товарів, але щоб повідомити його про цей процес, додаємо loader, який буде відображатися, поки сторінка не завантажиться (рис. 4.32). Це стандартний елемент будь-якого сайту, який може мати різний рівень складності реалізації.

Рисунок 4.32 — Loader

Тепер розробимо сторінку для конкретного товару (рис. 4.33). Вона складатиметься з двох основних блоків: першого – зображення товару разом із можливими варіаціями кольору та іншими опціональними характеристиками.

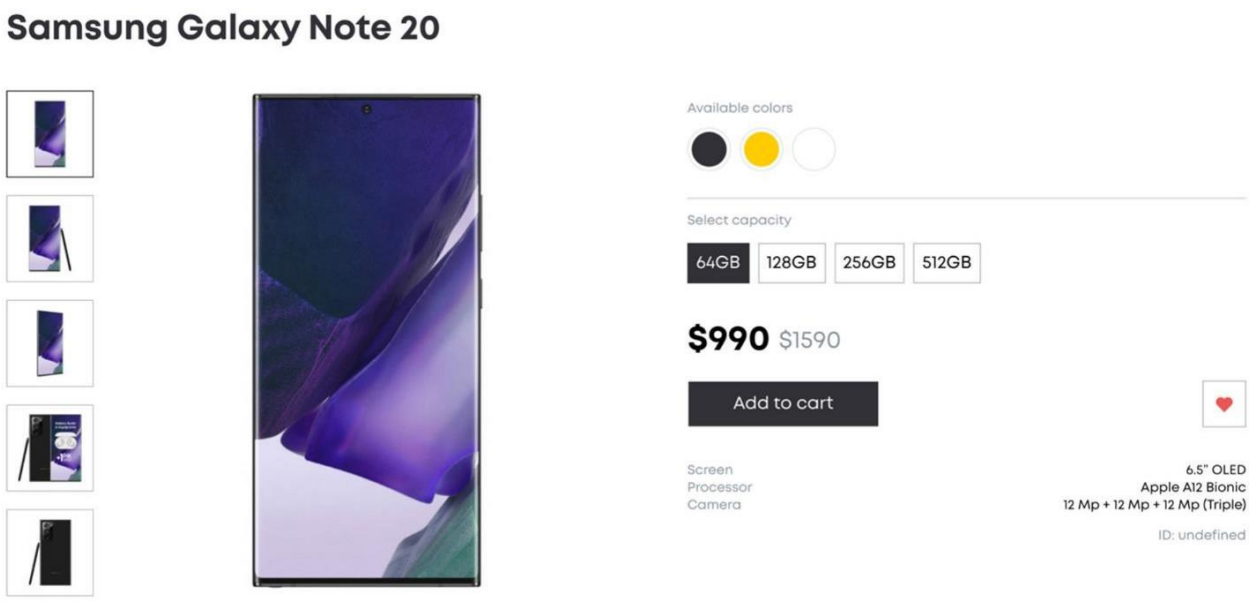

Рисунок 4.33 — Перший блок сторінки товару

Другий блок міститиме опис телефону або будь-якого іншого пристрою, а також перелік усіх характеристик, зазначених для цієї моделі товару (рис. 4.34).

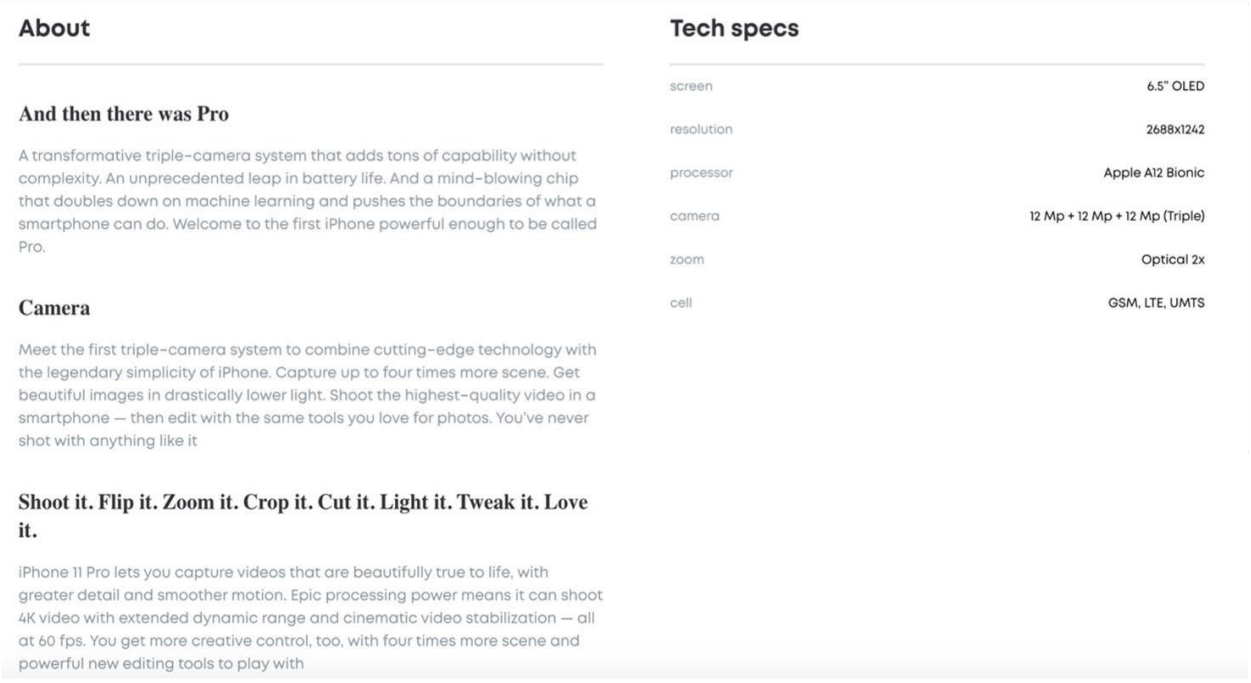

Рисунок 4.34 — Другий блок сторінки товару

В кінці було реалізовано функцію кошика (рис. 4.35), що дозволяє користувачеві просто клацати на іконку "додати в кошик" у будь-якому місці, де

вона розташована, і товар автоматично додається до кошика. Додавання до кошика здійснюється у два етапи: перший — це додавання у динамічну пам’ять, що означає, що під час перезавантаження дані зникнуть, але це забезпечує швидкий обмін даними. Другий етап — додавання цих даних у LocalStorage, яке не залежить від перезавантаження сторінки чи вимикання комп'ютера. Таким чином, якщо користувач додав товари, вони залишаться в кошику доти, поки він їх самостійно не видалить або не очистить кеш та куки. Перший елемент цієї сторінки — це картка доданого товару, яка повинна містити фото товару, його назву, блок для регулювання кількості та загальну суму.

Рисунок 4.35 — Товар в кошику

Після цього розташований блок для здійснення покупки, де вказується кінцева сума. Якщо в кошику кілька товарів, тут буде відображено загальну суму всіх цін, а також сумарну кількість одиниць товару (рис. 4.36).

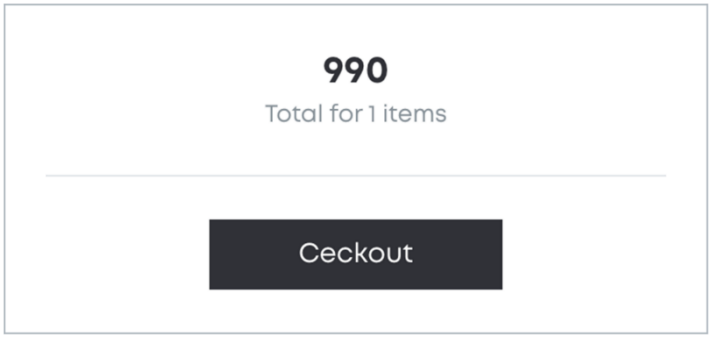

Рисунок 4.36 — Кнопка для завершення покупки

На рисунку 4.37 показано, як вся конструкція виглядатиме на середньому та великому екрані.

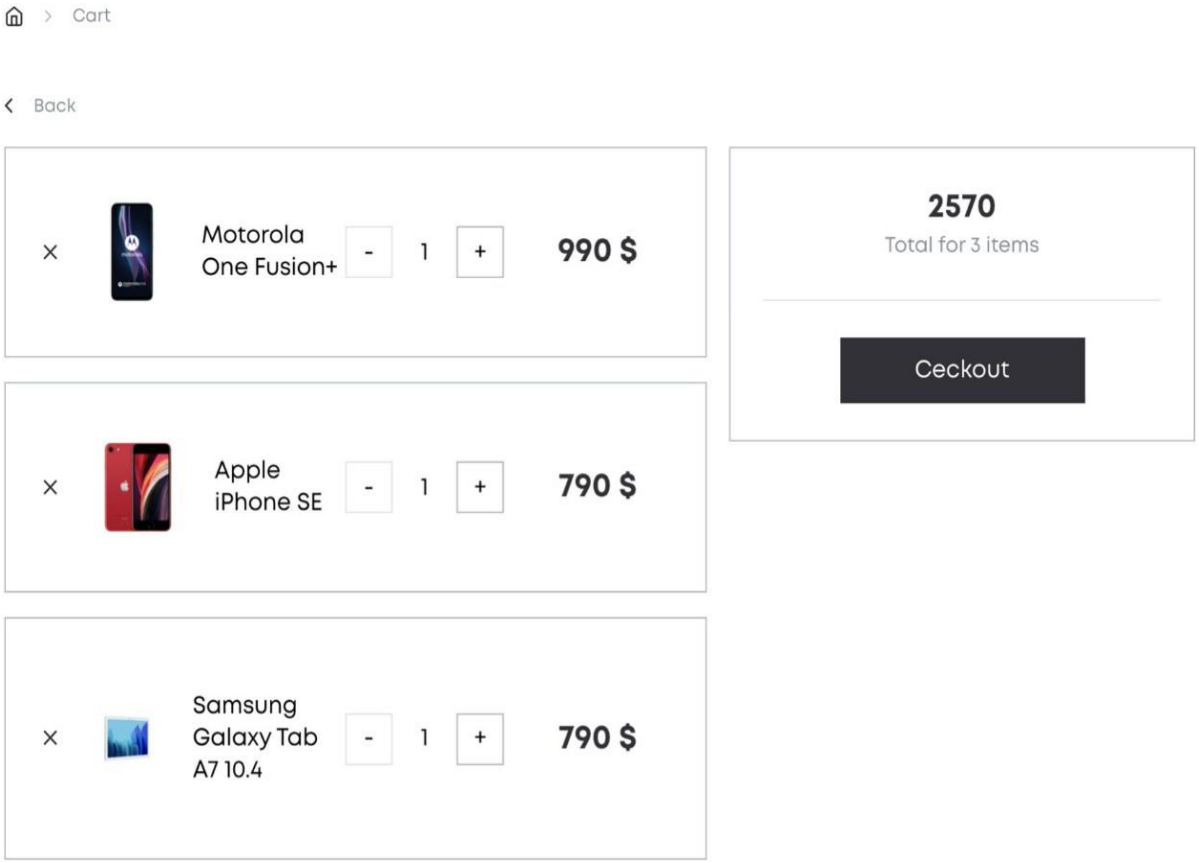

Рисунок 4.37 — Кошик на великих екранах

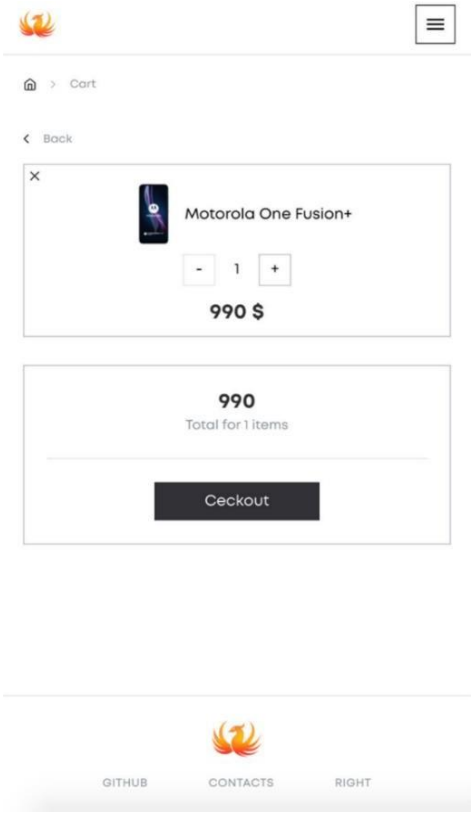

Рисунок 4.38 — Кошик для малих екранів

Оскільки таку конструкцію вкрай важко масштабувати для маленьких пристроїв, ми просто переверстали її під більш відповідну модель (рис. 4.38). Вся розробка вимагає багато часу та коду, тому очевидно, що під час процесу можуть виникати проблеми та непередбачувані ситуації, які потрібно буде переробляти. Для полегшення цього процесу використовувалася система контролю версій Git та сервіс GitHub.

Обрана модель SPA забезпечує швидкість і простоту використання. Під цю модель була вибрана база даних MongoDB, а також зберігання даних на серверах MongoDB. Створено роутери під необхідні групи товарів, які експортовані як модулі для спрощення написання коду. Розроблені інтерфейси під групи товарів і екшени в Redux. Написано модулі для хедера, адаптованість якого забезпечується наявністю мобільної та десктопної версій, які активуються залежно від ширини екрана. Створено модель картки товару та об’єднано їх у список товарів. Додано слайдер, щоб на головній сторінці картки можна було прокручувати як у слайдері. Реалізовано сторінки для мобільних пристроїв та планшетів, а також сторінки для купівлі товарів, улюблених товарів та кошика.

4.1.4 Реалізація модуля рекомендацій

Модуль рекомендацій реалізований у вигляді чату, який взаємодіє з користувачем за допомогою API OpenAI. Він працює як віртуальний асистент, який допомагає вибрати продукцію, надає інформацію про доступні товари та відповідає на запити користувачів.

Основні функціональні можливості модуля:

а) Динамічний чат: користувач може надсилати текстові повідомлення, а система відповідає, використовуючи OpenAI;

б) Обробка історії розмови: чат зберігає історію діалогу для більш природної взаємодії;

в) Обмежена база знань: асистент знає лише про певні моделі Apple (iPhone, iPad, Apple Watch), що робить його спеціалізованим помічником;
г) Автоматичні відповіді: якщо користувач цікавиться товарами, бот надає інформацію про їхню наявність і конфігурацію;
д) Реакція на нецільові запити: якщо запит не стосується доступних товарів, чат повідомляє, що може допомогти лише з вибором продукції.

Інтерфейс модуля містить кнопку відкриття/закриття чату, область повідомлень і поле для введення тексту. Даний підхід дозволяє створити інтерактивну систему рекомендацій, що підвищує зручність використання інтернет-магазину. Відповіді генеруються в режимі реального часу, що забезпечує миттєву реакцію на запити. Інтерфейс чату мінімалістичний і не заважає перегляду сторінки, залишаючись у вигляді компактного віджета, який можна легко згорнути або розгорнути. Передбачена можливість гнучкого налаштування – користувач може змінювати стиль чату, обирати мову спілкування та рівень деталізації відповідей. У майбутньому модуль може бути доповнений підтримкою голосових запитів або інтеграцією з іншими сервісами, такими як CRM чи системи управління замовленнями.

4.2 Інструкція користувача інтернет магазину

Інтернет-магазин створений для зручного перегляду каталогу телефонів, їх пошуку, сортування та додавання до кошика. Ця інструкція містить покроковий опис того, як користуватися інтернет-магазином, і допоможе навіть новачкові швидко розібратися з усіма функціями.

Для початку роботи користувачеві потрібно відкрити браузер, наприклад, Google Chrome, Mozilla Firefox або Safari, і перейти на головну сторінку інтернет-магазину. Після завантаження відкриється головна сторінка, на якій представлений каталог телефонів у вигляді списку. Кожен елемент списку включає назву моделі, зображення, ціну та рейтинг.

На самому початку роботи користувач має змогу ознайомитися з головними елементами інтерфейсу. У верхній частині сторінки розташоване меню, яке містить пошуковий рядок і елементи для сортування. Пошук дозволяє знайти конкретну модель телефону за назвою або ключовими словами. Для цього достатньо ввести текст у поле пошуку. Наприклад, якщо користувач шукає телефон "iPhone", потрібно ввести це слово, і список автоматично оновиться, відобразивши всі моделі iPhone, доступні в каталозі. Цей інструмент працює миттєво, забезпечуючи швидкий доступ до потрібної інформації.

На головній сторінці також є функція сортування, яка дозволяє впорядковувати телефони за ціною Рис. 4.39 (від найдешевшого до найдорожчого або навпаки) або за алфавітом. Це дуже зручно, якщо користувач хоче знайти найдоступніший телефон чи переглянути всі моделі певного бренду в алфавітному порядку.

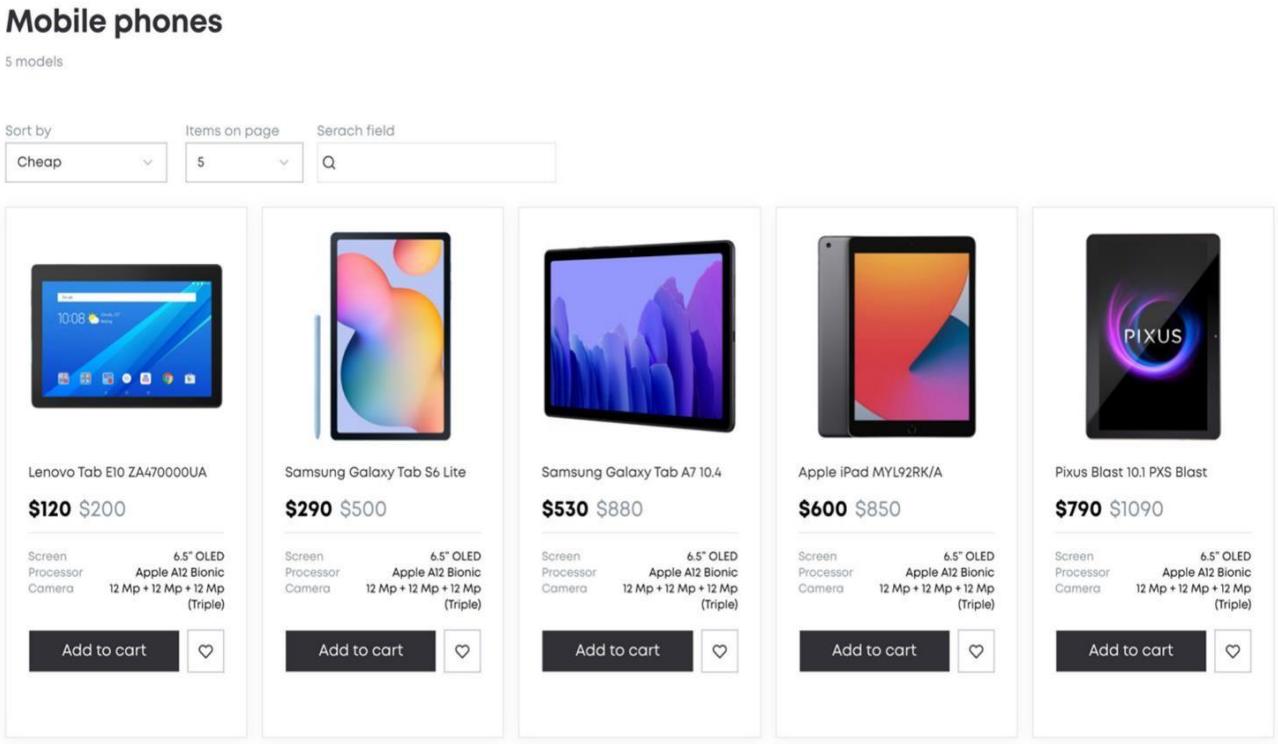

Рисунок 4.39 – Сортування спочатку найдешевші

Якщо користувач бажає отримати детальну інформацію про конкретну модель телефону, йому потрібно натиснути на відповідний елемент у списку. Наприклад, натискання на назву або зображення телефону відкриває окрему

сторінку, присвячену цій моделі. На цій сторінці представлено повний опис товару, включаючи основні технічні характеристики, такі як розмір екрана, обсяг пам'яті, якість камери тощо. Крім того, користувач може переглянути детальні зображення телефону, що дозволяють оцінити його дизайн. Для більш зручного вибору на сторінці відображається рейтинг товару, заснований на відгуках інших користувачів.

Одна з ключових функцій інтернет-магазину – це можливість додавання товарів до кошика. Якщо користувач хоче додати телефон у свій список покупок, йому потрібно натиснути кнопку "Add to Cart", розташовану поруч із відповідним товаром. Після цього телефон автоматично додається до кошика, і користувач може продовжувати перегляд каталогу або перейти до кошика для перегляду обраних товарів.

Кошик є важливою складовою інтернет-магазину і доступний у будь-який момент, натиснувши відповідну іконку у верхньому правому куті сторінки. У кошику зберігається список усіх обраних товарів. Рис. 4.40

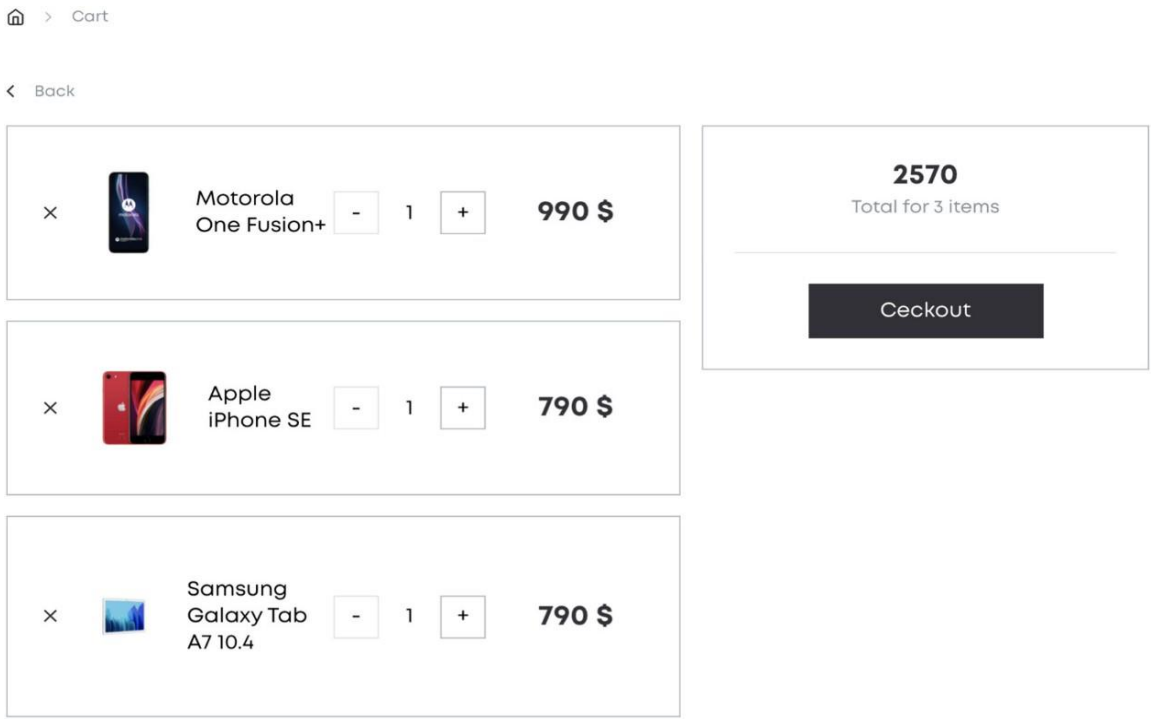

Рисунок 4.40 – Кошик інтернет магазину

Користувач може переглянути ці товари, видалити непотрібні або підготувати їх для подальшого замовлення. Наприклад, якщо користувач випадково додав неправильну модель, він може легко її видалити, натиснувши відповідну

кнопку в кошику. Усі дії виконуються інтуїтивно зрозуміло, а інтерфейс інтернет-магазину допомагає швидко зорієнтуватися.

Інтернет-магазин підтримує адаптивний дизайн, тому його зручно використовувати як на комп'ютері, так і на смартфоні чи планшеті. Усі елементи інтерфейсу автоматично підлаштовуються під розмір екрана, забезпечуючи зручний доступ до функцій незалежно від типу пристрою. Наприклад, на мобільному телефоні меню компактно складається, а кнопки збільшуються для зручності натискання.

Ще одна особливість інтернет-магазину – збереження даних у локальній пам'яті пристрою (LocalStorage). Це означає, що навіть якщо користувач закриє браузер, товари у кошику залишаться збереженими, і при наступному відвідуванні інтернет-магазину він побачить свій вибір. Це зручно, якщо користувач переглядає каталог у кілька етапів і не хоче втрачати вже обрані моделі.

Після завершення роботи з інтернет-магазином користувач може просто закрити вкладку браузера. Жодних додаткових дій для збереження інформації виконувати не потрібно. Завдяки зрозумілому інтерфейсу, гнучким функціям та адаптивному дизайну, інтернет-магазин "Phone Catalog" є простим і зручним у використанні для всіх категорій користувачів.

4.3 Тестування програмного продукту

Тестування програмного забезпечення (ПЗ) — це процес, який передбачає оцінку програмного продукту на предмет відповідності вимогам, виявлення дефектів та перевірки правильності роботи всіх функцій [35]. Основною метою тестування є забезпечення якості програмного забезпечення, його здатності виконувати задані функції у відповідності до специфікацій та вимог замовника. Основні етапи тестування:

а) Планування тестування — визначення критеріїв якості, тестових ситуацій і методів;
б) Розробка тестових сценаріїв — створення тестів на основі специфікацій;

в) Виконання тестів — реальне проведення тестування;

г) Збір та аналіз результатів тестування — фіксація знайдених помилок та оцінка відповідності ПЗ вимогам.

Існують різні види тестування, зокрема:

а) Модульне тестування — перевірка окремих частин системи;

б) Інтеграційне тестування — перевірка взаємодії між компонентами;

в) Системне тестування — перевірка всієї системи як єдиного цілого;

г) Приймальне тестування — перевірка ПЗ за критеріями користувача.

У таблиці 4.1 наведено основні тестові ситуації для перевірки функціональності онлайн-магазину мобільних телефонів:

Таблиця 4.1 Тестові ситуації

№	Тестова ситуація	Очікуваний результат	Статус
1	Перевірка реєстрації нового користувача	Користувач успішно зареєстрований, отримує підтвердження	Пройдено
2	Перевірка авторизації користувача	Користувач успішно входить у систему з правильними даними	Пройдено
3	Перевірка перегляду списку товарів	Товари відображаються коректно з усіма необхідними даними	Пройдено
4	Перевірка додавання товару в кошик	Товар додається в кошик, кількість товару оновлюється	Пройдено
5	Перевірка оформлення замовлення	Користувач успішно оформлює замовлення та отримує підтвердження	Пройдено
6	Перевірка роботи рекомендаційного модуля	Система рекомендує товари на основі попередніх покупок користувача	Пройдено
7	Перевірка відображення інформації про товар	Вся інформація про товар (ціна, опис, характеристики) коректно відображається	Пройдено

Модульне тестування Проводиться для окремих компонентів онлайн-магазину. Наприклад, тестування функціональності кошика: додавання товарів, оновлення кількості товарів, видалення товарів з кошика.

Інтеграційне тестування Перевірка взаємодії між компонентами. Напри-

клад, тестування взаємодії між фронтендом і бекендом при оформленні замовлення. Перевірка, чи коректно відображаються дані з бази даних на сайті, чи відправляються дані про замовлення до серверу.

Тестова ситуація: Перевірка додавання товару в кошик.

Кроки:

а) Користувач заходить на сторінку товару;

б) Натискає кнопку "Додати в кошик";

в) Перевіряється, чи змінилась кількість товарів у кошику;

г) Перевіряється, чи відображається товар у кошику.

Очікуваний результат: Товар додається в кошик, кількість товарів збільшується на 1.

Результат: Успішно виконано, товар відображається в кошику.

Для автоматизованого тестування фронтенд-компонентів використовувалася React Testing Library у поєднанні з Jest. Це дозволяє перевірити, чи правильно працюють інтерфейсні компоненти без необхідності запуску браузера.

```
import { render, screen } from "@testing-library/react";
import ProductList from "../components/ProductList";

test("відображення списку товарів", () => {
  render(<ProductList />);
  const productItems = screen.getAllByTestId("product-item");
  expect(productItems.length).toBeGreaterThan(0);
});
```

Рисунок 4.41 — Тестування рендерингу списку товарів

Метою тесту на рисунку 4.41 є перевірка правильності відображення списку товарів на сторінці інтернет-магазину. Тест виконується за допомогою React Testing Library і перевіряє, чи коректно рендеряться компоненти товарів у списку.

Під час тесту виконується рендеринг компонента ProductList, після чого перевіряється наявність товарів у DOM-дереві. Якщо список товарів містить

хоча б один елемент, тест вважається успішним.

```
import { render, screen, fireEvent } from "@testing-library/react";
import Cart from "../components/Cart";

test("додавання товару в кошик", () => {
  render(<Cart />);
  const addButton = screen.getByText("Додати в кошик");
  fireEvent.click(addButton);
  const cartItems = screen.getByTestId("cart-items");
  expect(cartItems.children.length).toBe(1);
});
```

Рисунок 4.42 — Тестування додавання товару в кошик

Тест на рисунку 4.42 перевіряє, чи правильно працює механізм додавання товарів у кошик. Основним сценарієм тестування є натискання кнопки «Додати в кошик» і перевірка, чи відображається товар у списку доданих товарів.

```
import { render, screen, fireEvent } from "@testing-library/react";
import SearchBar from "../components/SearchBar";

test("пошук товарів за ключовими словами", () => {
  render(<SearchBar />);
  const searchInput = screen.getByPlaceholderText("Пошук...");
  fireEvent.change(searchInput, { target: { value: "iPhone" } });
  expect(searchInput.value).toBe("iPhone");
});
```

Рисунок 4.43 — Тестування пошуку товарів

Тест на рисунку 4.43 має на меті перевірити працездатність пошуку товарів за ключовими словами. Під час тестування здійснюється введення запиту в поле пошуку, після чого перевіряється правильність його оновлення. Тестовий сценарій включає рендеринг компонента SearchBar, симуляцію введення тексту користувачем та перевірку того, чи було оновлено поле пошуку відповідно до введеного значення.

Цей програмний продукт можна використовувати не лише в онлайн-торгівлі мобільними телефонами, а й в інших сферах електронної комерції, таких як продаж побутової техніки, книг, одягу та аксесуарів. Завдяки адаптивному дизайну та розширеній системі рекомендацій, система може бути корисною для інших типів товарів, забезпечуючи зручний користувацький інтерфейс та інтуїтивно зрозумілу навігацію.

Крім того, модульна архітектура програми дозволяє легко адаптувати та налаштовувати функціональність під конкретні вимоги різних бізнесів, що робить його універсальним рішенням для електронної комерції. Інтеграція таких функцій, як пошук, фільтрація та категоризація товарів, також сприяє підвищенню ефективності продажів. Отже, ця платформа може стати основою для розвитку різноманітних проектів у сфері електронної комерції.

ВИСНОВКИ

Проаналізовано предметну область дослідження, визначено поняття та основні характеристики електронної комерції, особливості концепції інтернет-магазинів як одного з основних типів веб-сайтів, проаналізовано основні переваги та недоліки інтернет-магазинів, визначено класифікацію та характеристики методів розробки.

Встановлено, що інтернет-магазини мають складну класифікаційну структуру і час створення інтернету-магазину незрівнянно менше, ніж звичайного магазин. З’являється свобода переміщення продавця, з інтернет- магазином з'являється можливість розширити географію бізнесу на світові ринки і професійно створений інтернет-магазин може функціонувати повністю автономно.

Проаналізовані та детально пояснені різні технології, які поєднуються для формування стеку MERN. Крім того здійснено аналіз, інших інструментів, які допомагають керувати даними додатків, обробкою помилок та аутентифікацією користувача. Обговорено та вивчено шляхи реалізації кінцевої програми – інтернет магазину з продажу мобільної техніки.

Обрано модель SPA, оскільки вона дає швидкість і простоту використання. Під цю модель обрано базу даних MongoDb і зберігання цієї бази даних на серверах MongoDb. Створено роути під необхідні групи товарів і експортовано їх як модулі для обробки простоти написання. Розроблено інтернет-магазин з модулем рекомендацій. Проведено тестування та аналіз дослідної експлуатації.

СПИСОК ЛІТЕРАТУРИ

1. A brief history of Node.js. NodeJS : веб-сайт. URL: https://nodejs.dev/learn/a-brief-history-of-nodejs (дата звернення: 10.12.2024).
2. Bachuk A. Redux · An Introduction. Articles. 2016. URL: https://www.smashingmagazine.com/2016/06/an-introduction-to-redux/ (дата звернення: 10.12.2024).
3. Components and Props. React : веб-сайт. URL: https://reactjs.org/docs/components-and-props.html (дата звернення: 10.12.2024).
4. Express. Express : веб-сайт. URL: https://expressjs.com (дата звернення: 10.12.2024).
5. Garcia R. JWT tokens and security – working principles and use cases. VAADATA. 2016. URL: https://www.vaadata.com/blog/jwt-tokens-and-security-working-principles-and-use-cases/ (дата звернення: 10.12.2024).
6. Git Handbook. GitHub : веб-сайт. URL: https://guides.github.com/introduction/git-handbook/ (дата звернення: 10.12.2024).
7. Goldwater M. An abbreviated history of JavaScript package managers. JavaScript. 2019. URL: https://javascript.plainenglish.io/an-abbreviated-history-of-javascript-package-managers-f9797be7cf0e (дата звернення: 10.12.2024).
8. Historical trends in the usage statistics of server-side programming languages for websites. URL: https://w3techs.com/technologies/history_overview/programming_language (дата звернення: 10.12.2024).
9. Hoque S. Full-stack React Projects Second Edition. Birmingham, UK. Packt Publishing. 2020. URL: https://www.packtpub.com (дата звернення: 10.12.2024).

10. Horowitz E. A Founder's Reflections on 10 Years of MongoDB. MongoDB. 2017. URL: https://www.mongodb.com/mern-stack (дата звернення: 10.12.2024).
11. Introducing Hooks. React : веб-сайт. URL: https://reactjs.org/docs/hooks-intro.html (дата звернення: 10.12.2024).
12. Karnik N. Introduction to Mongoose for MongoDB. freeCodeCamp. 2018. URL: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/ (дата звернення: 10.12.2024).
13. Karpov V. The MEAN Stack: MongoDB, ExpressJS, AngularJS and Node.js. MongoDB. 2017. URL: https://www.mongodb.com/blog/post/the-mean-stack-mongodb-expressjs-angularjs-and (дата звернення: 10.12.2024).
14. MERN Stack. MongoDB : веб-сайт. URL: https://www.mongodb.com/mern-stack (дата звернення: 10.12.2024).
15. Models. Mongoose : веб-сайт. URL: https://mongoosejs.com/docs/models.html (дата звернення: 10.12.2024).
16. Motivation. Redux : веб-сайт. URL: https://redux.js.org/understanding/thinking-in-redux/motivation (дата звернення: 10.12.2024).
17. Using middleware. Express : веб-сайт. URL: https://expressjs.com/en/guide/using-middleware.html (дата звернення: 10.12.2024).
18. Using the State Hook. React : веб-сайт. URL: https://reactjs.org/docs/hooks-state.html (дата звернення: 10.12.2024).
19. Virtual DOM and Internals. React : веб-сайт. URL: https://reactjs.org/docs/faq-internals.html (дата звернення: 10.12.2024).
20. What is middleware? How it works in nodeJS? A Simple implementation. Medium. 2019. URL: https://medium.com/dataseries/what-is-middleware-how-it-works-in-nodejs-a-simple-implementation-485bcb9c3d53 (дата звернення: 10.12.2024).
21. Wodehouse C. 5 SQL vs. NoSQL Databases: What’s the Difference?.

Upwork. 2019. URL: https://www.upwork.com/resources/sql-vs-nosql-databases-whats-the-difference (дата звернення: 10.12.2024).

22. Zuraida A. Authorization and Authentication. Medium. 2018. URL: https://medium.com/@audira98/authorization-and-authentication-cd0a7c994278 (дата звернення: 10.12.2024).
23. Документація бази даних MongoDB. URL: https://www.mongodb.com/cloud (дата звернення: 10.12.2024).
24. Документація мови програмування JavaScript. URL: https://javascript.info/ (дата звернення: 10.12.2024).
25. Документація мови програмування TypeScript. URL: https://www.typescriptlang.org/ (дата звернення: 10.12.2024).
26. Документація фреймворку ReactJS. URL: https://reactjs.org/ (дата звернення: 10.12.2024).
27. Іванкевич О. В., Кременецький Г. М., Мазур В. І. Інформаційні системи та структури даних : навч. посіб. Київ : НАУ, 2006. 232 с.
28. Класифікація сучасних інтернет-магазинів в електронній комерції. URL: https://sebweo.com/klasifikatsiya-suchasnih-internet-magaziniv-v-elektronnij-komertsiyi/ (дата звернення: 10.12.2024).
29. Проникнення Інтернету в Україні. Жовтень, 2019. Автор: статистичне агентство Factum Group. URL: https://inau.ua/sites/default/files/file/1910/dani_ustanovchyh_doslidzhen_iii_kvartal_2019_roku.pdf (дата звернення: 10.12.2024).
30. Райчев І. Е., Харченко О. Г., Замковий В. В. Принципи проектування відкритих розподілених систем : навч. посіб. Київ : Вид-во Нац. авіац. ун-ту "НАУ-друк", 2015. 240 с.
31. Шалева О. І. Електронна комерція : навч. посіб. Київ : Центр учбової літератури, 2011. 216 с.
32. 9 Types of Software Testing in Software Engineering. URL: https://thectoclub.com/test-management/types-of-software-testing/ (дата звернення: 10.12.2024).

ДОДАТКИ

ДОДАТОК А. Код компонентів ProductList.js та ProductItem.js

```
import React, { useEffect, useState } from
'react'; import axios from 'axios';
const ProductList = () => {
const [products, setProducts] = useState([]);
useEffect(() => {
const fetchProducts = async () =>
{ try {
const response = await axios.get('/api/products');
setProducts(response.data);
} catch (error) {
console.error(error);
}
};
fetchProducts();
}, []);
return (
<div>
<h1>Магазин мобільних телефонів</h1>
<ul>
{products.map((product) => (
<li key={product._id}>{product.name}</li>
))}
</ul>
</div>
);
};
export default ProductList;

// src/components/ProductItem.js
import React from 'react';
const ProductItem = ({ product })
=> { return (

<div>
<h3>{product.name}</h3>
<p>{product.description}</p>
<img src={product.image} alt={product.name} />
<p>Ціна: {product.price}</p>
</div>
);
};
export default ProductItem;
```

ДОДАТОК Б. Вміст файла server.js

```
const express =
require('express'); const cors =
require('cors');
const mongoose = require('mongoose');
const app = express();
const port = process.env.PORT || 5000;
app.use(cors());
app.use(express.json());
const uri = 'mongodb://localhost:27017/mern_shop'; // При необ-
хідності, замініть URL MongoDB
mongoose.connect(uri, { useNewUrlParser:true,useCreateIndex:
                        true, useUnifiedTopology: true });
const connection = mongoose.connection;
connection.once('open', () => {
console.log('Connected to MongoDB database');
})
productSchema = new mongoose.Schema({ name:
String,
description: String,
image: String, price:
Number
});
const Product = mongoose.model('Product', productSchema);
app.get('/api/products', async (req, res)
=> { try {
const products = await Product.find();
res.json(products);
} catch (error) {
console.error(error);
res.status(500).json({ error: 'Internal server error' });
}
});
// Start the server
app.listen(port, () => {
console.log(`Server is running on port: ${port}`);
});
```