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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення інтернет-магазину електроніки з використанням фреймворків React та Express

Виконав: студент IV курсу, групи СН-43
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Роговський В.В.
(підпис) (прізвище та ініціали)

Керівник _____ Фриз М.Є.
(підпис) (прізвище та ініціали)

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

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

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» _січня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

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

Студенту Роговському Владиславу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення інтернет-магазину електроніки з використанням фреймворків React та Express

Керівник роботи Фриз Михайло Євгенович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*7*» *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 27 червня 2025р.

3. Вихідні дані до роботи Літературні та інтернет джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області. Аналіз наявних рішень. Вибір середовища розробки. Вибір технологій розробки. Вибір технологій розробки серверної частини. Вибір технологій розробки клієнтської частини. 2. Встановлення архітектури веб-застосунку. Опис структури Веб-застосунку. Опис структури клієнтського рівня. Опис структури серверного рівня. Проектування бази даних. 3. Розробка серверної частини веб-застосунку. Ініціалізація серверної частини проекту. Створення серверного файлу. Створення бази даних. Реалізація кінцевих точок веб-застосунку. Реалізація додаткових файлів. Розробка клієнтської частини веб-застосунку. Ініціалізація клієнтської частини веб-застосунку. Створення сторінок веб-застосунку. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік використаних джерел. Додатки.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., к.т.н., доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2025	*Виконано*
2.	Підбір джерел для розробки інтернет-магазину	31.01.2025-03.02.2025	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2025-06.02.2025	*Виконано*
4.	Виконання дослідження щодо існуючих інтернет магазинів.	07.02.2025-15.02.2025	*Виконано*
	Розроблення інтернет-магазину.		
5.	Оформлення розділу «Постановка завдання. Формування архітектури та технологій розробки»	16.02.2025-25.02.2025	*Виконано*
6.	Оформлення розділу «Планування структури веб-застосунку для онлайн продаж товарів електроніки»	26.02.2025-08.03.2025	*Виконано*
7.	Оформлення розділу «Розробка веб-застосунку для онлайн продаж товарів електроніки»	09.04.2025-21.04.2025	*Виконано*
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	10.06.2025-12.06.2025	*Виконано*
9.	Виконання завдання до підрозділу «Основи охорони	13.06.2025-15.06.2025	*Виконано*
	праці»		
10.	Оформлення кваліфікаційної роботи	16.06.2025-17.06.2025	*Виконано*
11.	Нормоконтроль	18.06.2025-19.06.2025	*Виконано*
12.	Перевірка на плагіат	20.06.2025	*Виконано*
13.	Попередній захист кваліфікаційної роботи	21.06.2025	*Виконано*
14.	Захист кваліфікаційної роботи	28.06.2025	

Студент _____ Роговський В.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Фриз М.Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення інтернет магазину електроніки з використанням фреймворків React та Express // Кваліфікаційна робота освітнього рівня «Бакалавр» // Роговський Владислав Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-43 // Тернопіль, 2025 // С. 63, рис. – 22, табл. – 1, додат. – 4, бібліогр. – 37.

Ключові слова: javascript, react, node.js, express.js, mongodb, rest api, інтернет-магазин, веб-застосунок..

Кваліфікаційна робота присвячена сучасного веб-застосунку для онлайн продажу товарів електроніки з використанням стеку технологій MERN та дослідженні практичних аспектів його впровадження з метою підвищення зручності для користувачів та ефективності бізнес-процесів.

У першому розділі виконано глибокий аналіз предметної області, проведено огляд існуючих рішень на ринку інтернет-торгівлі електронікою, сформульовано постановку завдання, обґрунтовано вибір архітектури застосунку та обрано технологічний стек для розробки.

Другий розділ присвячено детальному проектуванню структури веб-застосунку. Визначено трирівневу архітектуру системи, розроблено структуру клієнтської та серверної частин, спроектовано базу даних для ефективної взаємодії з даними та забезпечення масштабованості системи.

У третьому розділі детально описано процес реалізації серверної та клієнтської частин веб-застосунку. Зокрема, реалізовано REST API для взаємодії з клієнтом, розроблено сторінки користувацького інтерфейсу з використанням React, впроваджено механізми авторизації та безпеки, налаштовано роботу з базою даних MongoDB.

У четвертому розділі детально проаналізовано питання долікарської допомоги при ураженні електричним струмом та розглянуто вимоги ергономіки до організації робочого місця оператора ПК.

ANNOTATION

Creation of an Electronics Online Store Using the React and Express Frameworks // Qualification work of the educational level "Bachelor" // Rohovskyi Vladyslav Viktorovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-43 // Ternopil, 2025 // P. 63, fig. - 22, tabl. - 1, annexes. – 4, references - 37.

Keywords: javascript, react, node.js, express.js, mongodb, rest api, e-commerce, web application.

The qualification work is dedicated to creation a modern web application for online sales of electronic products using the MERN technology stack and to explore practical aspects of its implementation aimed at improving user experience and business efficiency.

The first chapter provides an in-depth analysis of the subject area, reviews existing e-commerce solutions on the market, defines the problem statement, and justifies the choice of application architecture and technology stack.

The second chapter focuses on designing the structure of the web application. A three-tier architecture is defined. The structure of both the client and server sides is developed, and a database schema is designed to ensure efficient data interaction and scalability.

The third chapter describes the implementation process of the server-side and client-side components of the web application. A REST API for client interaction is implemented, user interface pages are developed using React, authentication and security mechanisms are introduced, and integration with the MongoDB database.

The fourth chapter comprehensively analyzes issues related to medical assistance in cases of electric shock and the requirements of ergonomics for the organization of the workplace of the PC operator are considered.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – програмний інтерфейс додатка, що дозволяє взаємодіяти між клієнтською та серверною частинами застосунку.

REST API – архітектурний стиль створення веб-сервісів на основі протоколу HTTP з використанням CRUD-операцій.

DOM (Document Object Model) – об'єктна модель документа для взаємодії з HTML-структурою.

MERN – набір технологій для розробки веб-додатків, що включає MongoDB, Express.js, React, Node.js.

CRUD (Create, Read, Update, Delete) – базові операції управління даними.

IDE (Integrated Development Environment) – інтегроване середовище розробки програмного забезпечення.

JWT (JSON Web Token) – формат передачі безпечної інформації між сторонами у вигляді JSON-об’єкта.

Middleware – проміжне програмне забезпечення, що обробляє HTTP-запити між сервером і додатком.

UI (User Interface) – користувацький інтерфейс.

UX (User Experience) – досвід взаємодії користувача з інтерфейсом.

ЗМІСТ

ВСТУП

Актуальність теми. Інтернет-магазини електроніки є актуальними з кількох причин. По-перше, вони дозволяють значно зекономити час і зусилля покупців, надаючи можливість вибору та придбання товарів, не виходячи з дому. Це особливо важливо в умовах постійної зайнятості та швидкого темпу життя сучасної людини. По-друге, інтернет-магазини забезпечують широкий асортимент продукції, який часто перевищує можливості традиційних магазинів. Це дає можливість користувачам знайти необхідний товар за найвигіднішими умовами. По-третє, розвиток інтернет-магазинів сприяє зниженню цін на продукцію за рахунок зменшення витрат на оренду приміщень і персонал.

Мета і задачі дослідження. Основна мета створення інтернет-магазину електроніки – забезпечити зручний і доступний спосіб придбання електронних товарів для широкого кола споживачів. Для досягнення цієї мети необхідно виконати низку завдань:

– Здійснити аналіз предметної області, визначивши всі необхідні функції для веб-застосунку.

– Провести аналіз наявних рішень та зробити порівняльну оцінку кожного.

– Проектування основної архітектури застосунку. Визначення структури, компонентів та взаємодій між ними для забезпечення ефективного функціонування.

– Розробка зручного та інтуїтивно зрозумілого інтерфейсу. Забезпечити легкість навігації та пошуку товарів, щоб користувачі могли швидко знайти необхідну продукцію.

– Тестування створеного інтернет-магазину на готовність та перевірка виконання усіх вимог.

Практичне значення одержаних результатів. Готовий варіант інтернет-магазину електроніки має значне практичне значення для різних категорій користувачів і суспільства в цілому. Його впровадження і функціонування

сприяє підвищенню ефективності бізнес-процесів та розвитку економіки. Основні аспекти практичного значення включають:

– Зручність для споживачів: Інтернет-магазин забезпечує можливість купувати електроніку будь-якої миті і з будь-якого місця, де є доступ до інтернету. Це дозволяє заощаджувати час та зусилля, які інакше витрачалися б на поїздки до фізичних магазинів, черги та обмежений вибір.

– Ширший асортимент: Інтернет-магазин може запропонувати набагато ширший вибір товарів, ніж традиційний магазин, оскільки він не обмежений фізичним простором. Це дозволяє задовольнити потреби різних категорій споживачів, від початківців до професіоналів.

– Підтримка малого та середнього бізнесу: Інтернет-магазини надають можливість малим та середнім підприємствам виходити на глобальний ринок, не обмежуючись місцевими клієнтами. Це сприяє зростанню бізнесу і підвищенню його конкурентоспроможності.

Таким чином, розроблений варіант інтернет-магазину буде зручним та ефективним для користувачів способом купівлі товарів електроніки. Він матиме широкий асортимент товару, який задовольнить потреби великої категорії споживачів. Крім того, інтернет-магазин є чудовим рішенням для малих і середніх підприємств, щоб конкурувати в онлайн-світі.

РОЗДІЛ 1. ПОСТАНОВКА ЗАВДАННЯ. ФОРМУВАННЯ АРХІТЕКТУРИ ТА ТЕХНОЛОГІЙ РОЗРОБКИ

1.1 Аналіз предметної області

Цільове призначення інтернет-магазину полягає у створенні електронної торговельної платформи, яка дозволяє клієнтам здійснювати покупки електроніки онлайн з максимальним комфортом і безпекою. Голова мета полягає у створенні надійного і зручного середовища для покупок, де клієнти можуть отримати доступ до широкого вибору товарів за конкурентоспроможними цінами, а також отримати високоякісне обслуговування від команди професіоналів. Інтернет-магазин також має підтримувати розширені функціональні можливості, такі як інтерактивні відгуки клієнтів, персоналізовані рекомендації і зручні способи оплати та доставки [1].

Основними складовими та їх характеристиками для веб-додатку можна виділити:

– Каталог товарів: Ця складова містить повну інформацію про усі товари, що продаються, включаючи моделі, технічні характеристики, фотографії та ціни. Клієнти можуть швидко знаходити необхідні товари за допомогою фільтрів і категорій.

– Кошик покупок: Ця складова дозволяє клієнтам додавати товари до свого кошика, переглядати вміст, видаляти або змінювати кількість одиниць товарів перед оформленням замовлення.

– Список бажань: Ця складова має змогу обирати клієнтам вподобані товари у список. Це дозволить користувачам швидко знайти цікавий для них товар та додати його до кошика.

– Оформлення замовлення: Ця складова забезпечує процес оформлення замовлення, включаючи вибір способу оплати і доставки, введення адреси отримувача та підтвердження замовлення [2].

1.2 Аналіз наявних рішень

В сучасному світі існує безліч інтернет-магазинів, кожен з яких має свої особливості, переваги та недоліки. Аналіз веб-застосунків дозволяє нам краще розуміти їхню структуру, дизайн, контент, SEO-оптимізацію та інші важливі аспекти. Це допомагає визначити, які елементи працюють ефективно, а які можуть потребувати поліпшення. Крім того, це дає можливість вивчити найкращі практики та застосувати їх при створенні нового сайту або оптимізації існуючого. Такий аналіз є важливим кроком на шляху до успішного онлайн-бізнесу.

Для аналізу наявних рішень з багатьох доступних варіантів у просторі українського інтернету, були обрані такі варіанти: d.ua, comfy.ua та viatec.ua. Обрані варіанти представляють собою інтернет-магазини електроніки, які є популярними варіантами купівлі товарів для конкретних користувачів [3].

d.ua (див рис. 1.1) – інтернет-магазин побутової техніки та електроніки.

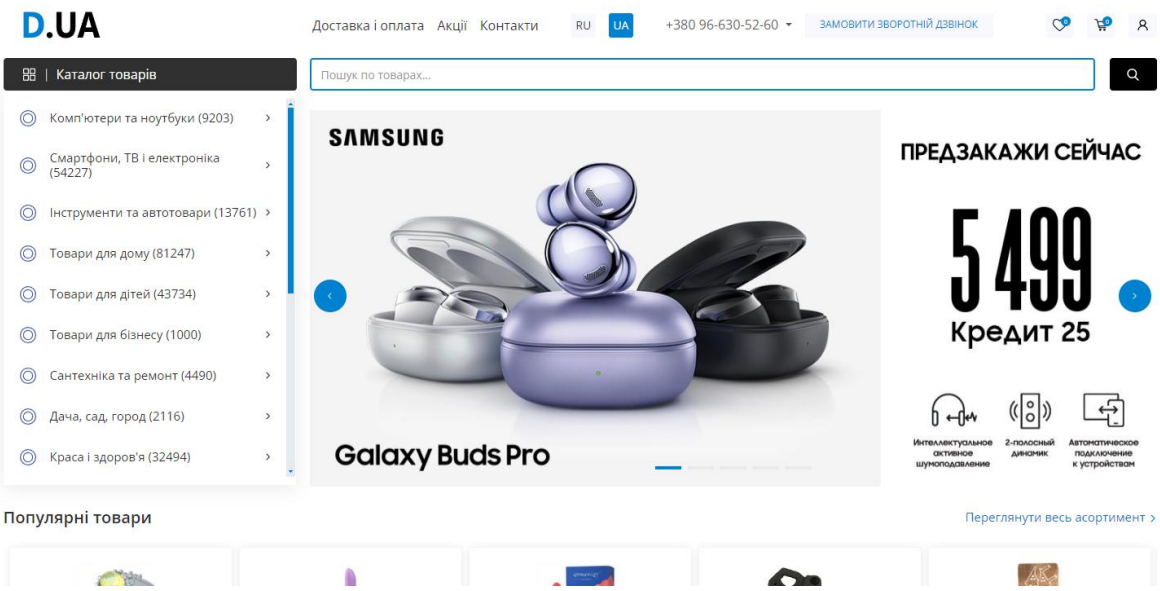

Рисунок 1.1 – Вигляд інтернет-магазину d.ua

Інтернет-магазин d.ua спеціалізується на продажу побутової техніки та електроніки. З першого погляду на веб-застосунок можна зробити висновок, що

інтерфейс сайту є інтуїтивно зрозумілим та не є нагромадженим великою кількістю інформації. Також відразу помітне різноманіття товару, яке включає в себе не тільки товари побутової техніки та електроніки. Проте, слід зазначити, що сайт є адаптованим для різних пристроїв, що дозволяє користувачам здійснювати покупки з різних пристроїв. Однак, як і будь-який інший веб-сайт, він має і свої недоліки. До одного з таких можна віднести те, що іноді інформація про товари може бути недостатньою або неактуальною. Також, можливо, сайт потребує покращення SEO-оптимізації для підвищення його видимості в пошукових системах.

Враховуючи ці фактори, можна зробити висновок, що d.ua є хорошим ресурсом для покупки електроніки та побутової техніки в Україні. Однак, як і будь-який інший веб-сайт, він має свої переваги та недоліки, які варто враховувати при виборі місця для онлайн-покупок.

Comfy.ua (див рис. 1.2) – популярний інтернет-магазин побутової техніки та електроніки [4].

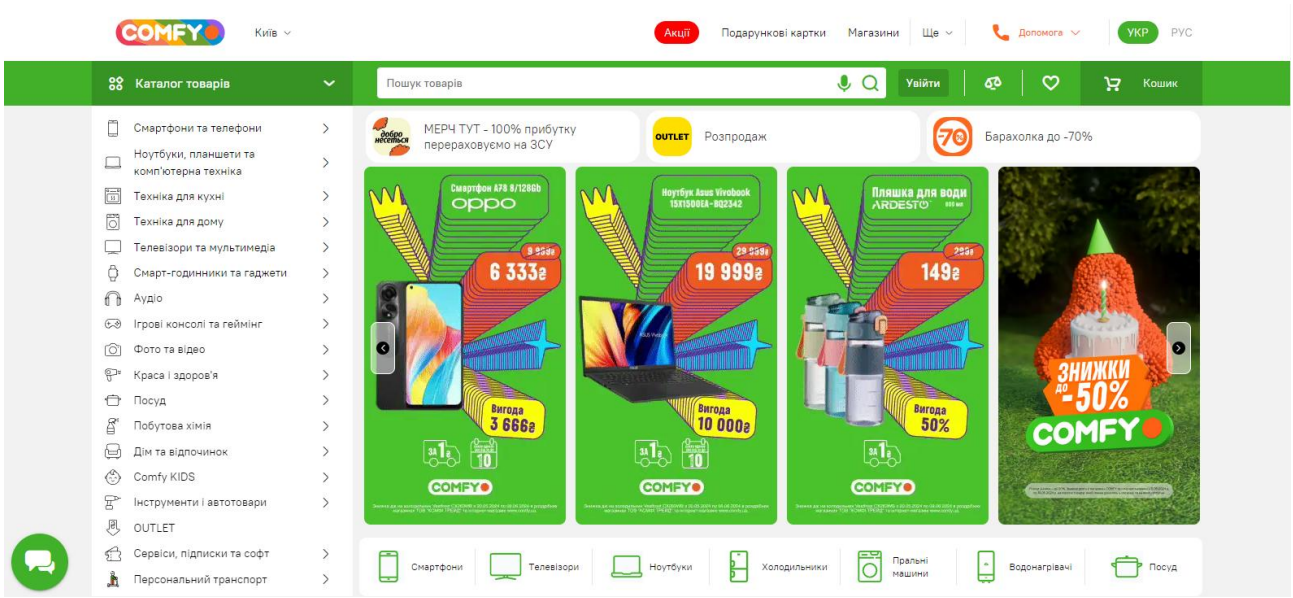

Рисунок 1.2 – Вигляд інтернет-магазину comfy.ua

Інтернет-магазин comfy.ua є одним із найпопулярніших рішень для купівлі товарів в інтернеті. Відразу після відкриття веб-сайту помітно нагромаджений

великою кількістю компонентів інтерфейс, але після деякого часу проведеному на сайті, він стає більш зрозумілим і не здається таким складним, як це здається на перший погляд. Comfy.ua пропонує обширний вибір продукції, які, в основному, включають в себе товари побутової техніки та електроніки, що і є основною концепцією інтернет-магазину. Також, веб-сайт comfy.ua має гарний зворотній зв'язок. У разі виникнення будь-яких проблем, менеджери швидко реагують та надають допомогу.

Загалом, comfy.ua є надійним ресурсом для покупки електроніки та побутової техніки в Україні. Веб-сайт має на своєму рахунку багато позитивних відгуків та є одним із найпопулярніших інтернет-магазинів.

Viatec.ua (див рис. 1.3) – дистриб'ютор технологій для відеоспостереження, сигналізації, контролю доступу [5].

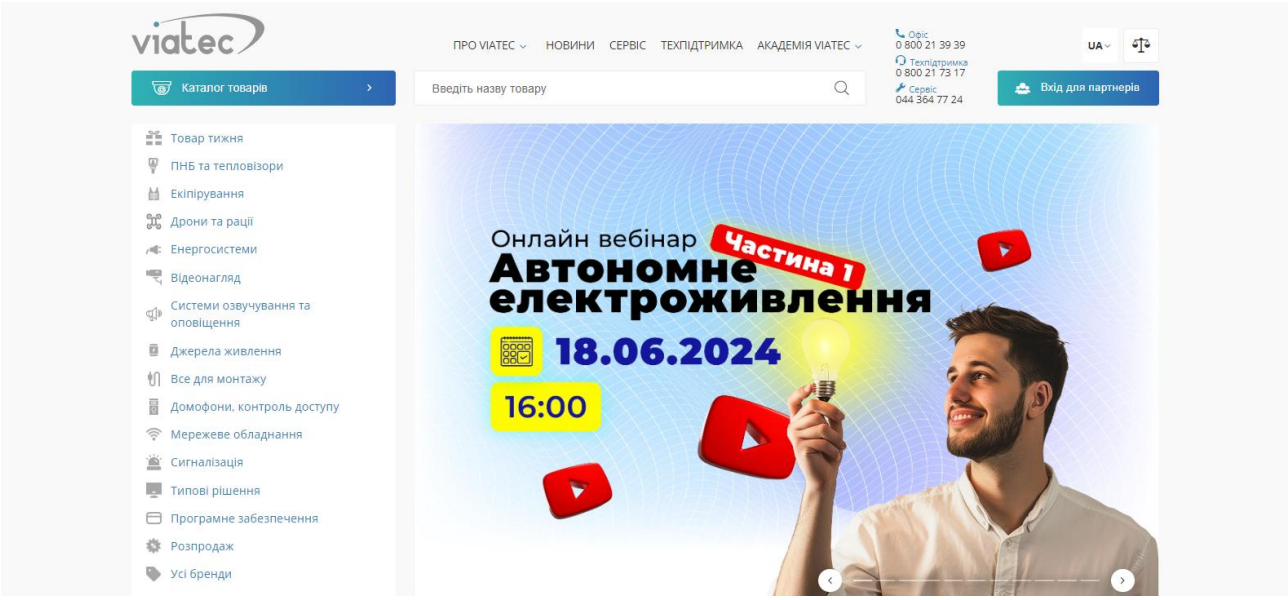

Рисунок 1.3 – Вигляд інтернет-магазину viatec.ua

Інтернет-магазин viatec.ua, за даними пошуку google не є одним з популярних сервісів, але є цікавим варіантом для розгляду. З перших вражень по веб-застосунку є інтуїтивно зрозумілий та простий інтерфейс у якому досить швидко розбираєшся та звикаєш. Веб-сайт пропонує великий вибір товарів

пов’язаних із відеоспостереженням, сигналізації та ін. З помітних недоліків можна зазначити те, що іноді інформації про товар може бути недостатньо.

В загальному, інтернет-магазин viatec.ua є хорошим ресурсом для купівлі потрібних користувачеві товарів онлайн.

Отже, провівши огляд усіх обраних варіантів було вирішено створити таблицю з усіма необхідними критеріями до веб-застосунку для врахування їх при розробці власного інтернет-магазину. Для оцінки веб-сайтів були обрані такі критерії: якість веб-сайту (дизайн, UX), асортимент, адаптивність до різних пристроїв, якість контенту.

Таблиця 1.1 – Аналіз наявних рішень

Характеристика	**d.ua**	**Comfy.ua**	**Viatec.ua**
Якість веб-сайту (дизайн, UX)	Зручний та сучасний	Нагромаджений, але зрозумілий	Простий та інтуїтивно зрозумілий
Асортимент	Широкий вибір товару, який виходить за межі електротоварів	Великий та різноманітний вибір товару	Великий вибір продукції для систем безпеки
Адаптивність до різних пристроїв	Так	Так	Так
Якість контенту	Місцями не достатньо інформативний	Якісний та різноманітний	Інформативний та професійний

Роблячи підсумок, можна сказати, що d.ua, comfy.ua та viatec.ua, є важливими гравцями на ринку інтернет-магазинів в Україні, кожен з яких має свої унікальні переваги. d.ua відрізняється широким асортиментом товарів та зручним дизайном сайту, але може мати проблеми з якістю контенту. Comfy.ua також пропонує широкий вибір товарів та високу якість обслуговування.

Viatec.ua є лідером в області систем безпеки, пропонуючи широкий асортимент товарів. Однак, через свою спеціалізацію, він може не відповідати потребам користувачів, які шукають інші види товарів.

1.3 Вибір середовища розробки

Програмісти мають у своєму розпорядженні низку середовищ розробки (IDE), які допомагають створювати веб-додатки. Ці IDE пропонують зручні інструменти програмування, що робить процес розробки ефективнішим. Коли дійшло до вибору оптимального середовища для цього проекту, варіанти були звужені до Atom, WebStorm і Visual Studio Code, які задовольняють різноманітні мови програмування та завдання.

Atom – це розширюване середовище розробки, створене командою GitHub і випущене у 2014 році. Основна його особливість полягає в тому, що воно є відкритим програмним забезпеченням, що означає доступність його вихідного коду для змін і розширень від спільноти розробників. Atom є одним з перших відомих текстових редакторів, побудованих на основі Electron, що дозволяє використовувати веб-технології (HTML, CSS, JavaScript) для розробки крос-платформних додатків. Основна мета Atom полягає у створенні відкритого, розширюваного середовища розробки, що може бути налаштоване для власних потреб. Ось деякі ключові аспекти Atom:

– Відкритий код і спільнота: Atom почався як відкритий проект, розроблений командою GitHub.

– Розширені можливості: Atom має широкий вибір плагінів і пакунків, які дозволяють розширювати його функціональність.

– Інтеграція з GitHub: Оскільки Atom розробляється тією ж командою, що й GitHub, він має потужну інтеграцію з цією платформою.

– Інтерфейс і користувацький досвід: Atom має досить простий інтерфейс, який легко налаштовувати. Він підтримує розширення для тем оформлення, робочих потоків і відображення різних елементів інтерфейсу.

Однак, не дивлячись на переваги, Atom може бути повільнішим за інші середовища розробки, особливо при роботі з великими проектами [6].

На рисунку 1.4 зображено інтерфейс Atom.

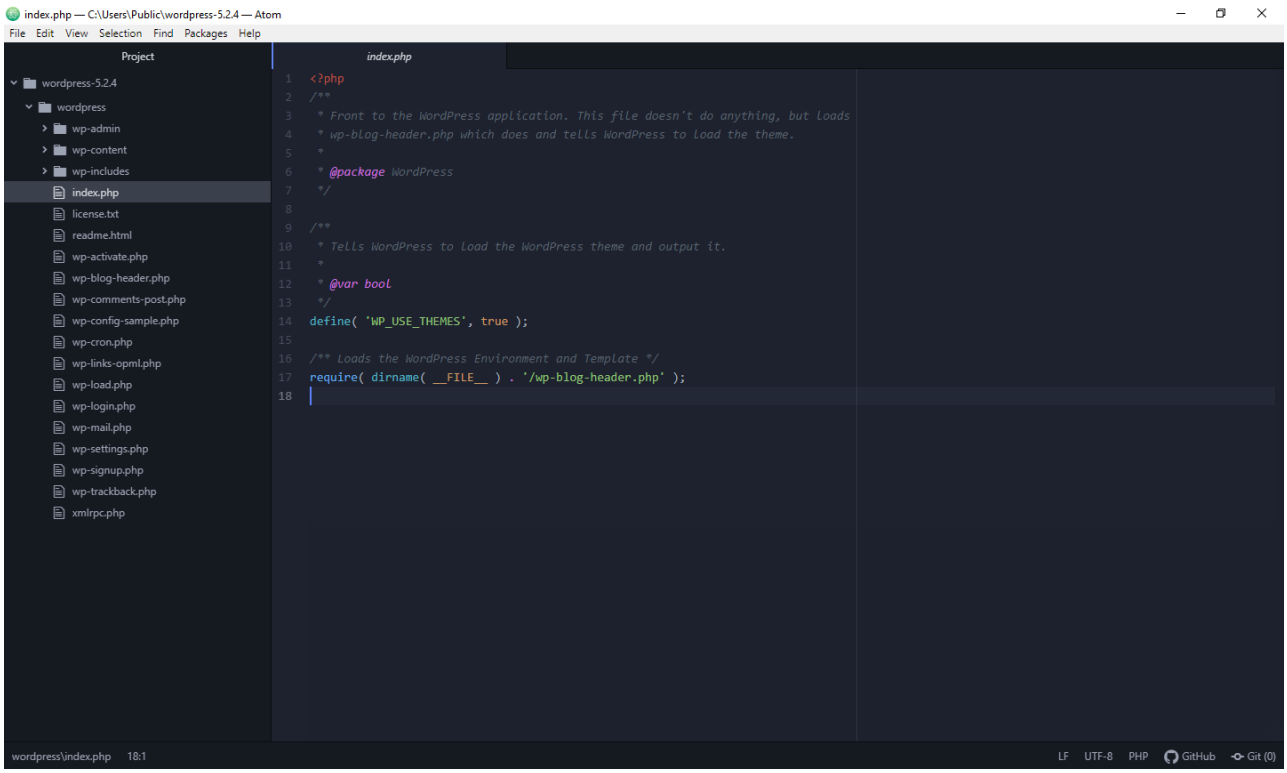

Рисунок 1.4 – Інтерфейс середовища розробки Atom

WebStorm – це потужне інтегроване середовище розробки (IDE), що зосереджується на веб-розробці і створене компанією JetBrains. Воно відоме своєю високою продуктивністю, широким спектром функцій і інтегрованими інструментами, які допомагають розробникам створювати якісний код і підтримувати процес розробки на високому рівні. Ось більш детальна інформація про особливості WebStorm:

– Підтримка мов програмування і технологій: WebStorm підтримує широкий спектр мов програмування, таких як JavaScript, TypeScript, HTML, CSS, і їхні сучасні фреймворки, такі як Angular, React, Vue.js, і Node.js. Це робить його ідеальним інструментом для розробки сучасних веб-додатків.

– Аналіз коду і рефакторинг: WebStorm пропонує розширені можливості аналізу коду, включаючи перевірку синтаксису, виявлення помилок і підказки

щодо покращення якості коду. Також підтримуються різні види рефакторингу, які дозволяють швидко і безпечно змінювати структуру коду.

– Live Editing і відладка: WebStorm має інтегровані інструменти для живого редагування (Live Editing) і відладки в реальному часі, що дозволяє відслідковувати зміни на сторінці без перезавантаження браузера і швидко виправляти помилки.

– Підтримка професійних стандартів: WebStorm орієнтований на професіоналів, що працюють у веб-розробці, і надає інструменти для створення надійних, масштабованих і високопродуктивних додатків.

З іншого боку, WebStorm може бути повільнішим та вимагати більше ресурсів, ніж інші середовища розробки. Крім того, для використання WebStorm потрібно придбати ліцензію [7].

На рисунку 1.5 зображено інтерфейс WebStorm.

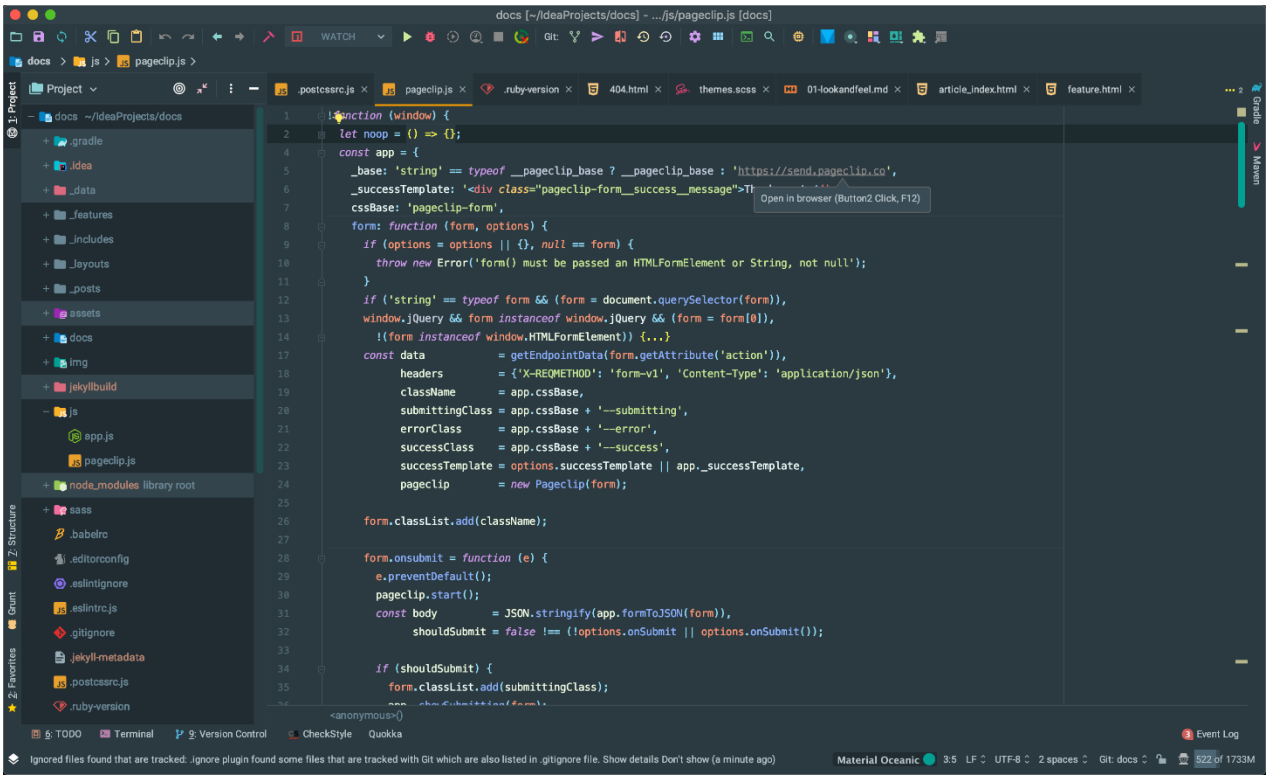

Рисунок 1.5 – Інтерфейс середовища розробки WebStorm

Visual Studio Code (VSCode) – це безкоштовне, легке, але потужне інтегроване середовище розробки (IDE) від Microsoft, яке стало надзвичайно

популярним серед розробників завдяки своїм розширеним можливостям, швидкодії і високій налаштованості [8].

На рисунку 1.6 зображено інтерфейс Visual Studio Code.

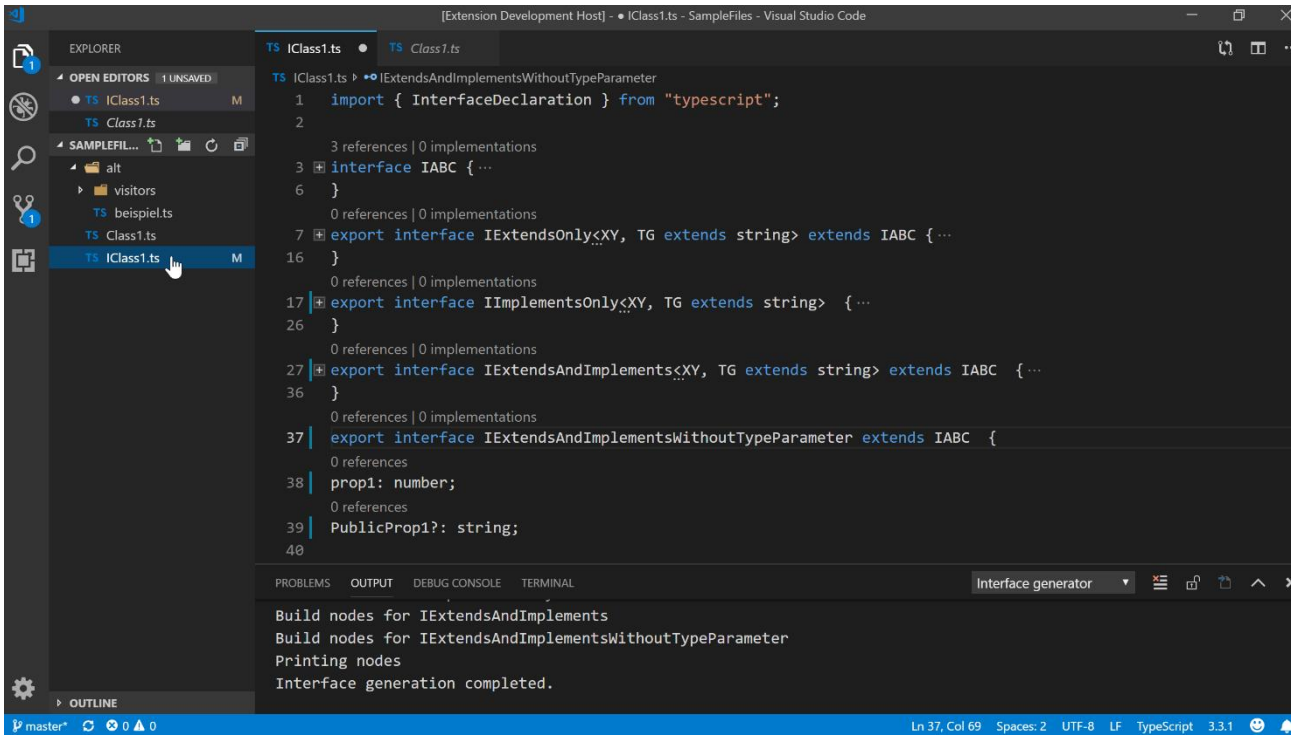

Рисунок 1.6 – Інтерфейс середовища розробки Visual Studio Code

Ось детальніший опис основних характеристик та особливостей VSCode:

– Підтримка мов програмування і технологій: VSCode підтримує велику кількість мов програмування, включаючи JavaScript, TypeScript, Python, Java, C++, PHP, HTML, CSS та багато інших. Вбудовані можливості для інтелектуального автодоповнення коду, підсвічування синтаксису і перевірки помилок забезпечують ефективну розробку незалежно від мови.

– Розширення: VSCode має потужну систему розширень, яка дозволяє додавати нові функції та інтеграції. Розширення можна легко встановити через вбудований менеджер розширень.

– Інтеграція з Git: VSCode має вбудовану підтримку для Git, що дозволяє легко виконувати основні операції з репозиторіями, такі як клонування, коміти, пуші, пулі та злиття.

– Інтелектуальні можливості: VSCode підтримує IntelliSense - функцію інтелектуального автодоповнення коду, яка надає пропозиції на основі типів змінних, функцій, імпортованих модулів і багато іншого.

Під час вибору середовища розробки для цього проекту було обрано Visual Studio Code як кращий варіант через кілька ключових факторів. Однією з головних причин є широкий спектр доступних розширень і плагінів, які дозволяють розробникам налаштовувати середовище відповідно до своїх потреб. Крім того, доступність VSCode є значною перевагою, оскільки це безкоштовне середовище [9].

1.4 Вибір технологій розробки

Визначення технологічного стеку є критичним етапом для успішного виконання програмного проекту. У контексті сучасного динамічного середовища, вибір відповідних технологій для веб-додатку стає ключовим фактором, що забезпечує ефективність розробки, високу продуктивність та можливість масштабування.

При створенні веб-додатку, було використано технологічний стек MERN. Цей стек було обрано замість стандартних мов програмування з огляду на його потужність та гнучкість. MongoDB використовується як база даних. Вона є нереляційною, що дозволяє зберігати дані в гнучкому, JSON-подібному форматі, що дозволяє швидко і легко модифікувати структуру даних. Express - це мінімалістичний та гнучкий серверний фреймворк для додатків Node.js, який надає набір потужних функцій для веб- та мобільних додатків. React використовується для клієнтської частини. Це JavaScript-бібліотека для побудови користувацьких інтерфейсів, яка дозволяє створювати швидкі та інтерактивні веб-сторінки. Node.js використовується як серверна платформа. Вона дозволяє виконувати JavaScript на сервері, що робить розробку більш єдиною та консистентною.

Використання технологічного стеку MERN дозволяє розробникам ефективно та ефективно створювати сучасні веб-додатки. Це надає можливість використовувати одну мову програмування (JavaScript) на всіх рівнях додатку, що спрощує процес розробки та підтримки [10].

1.4.1 Вибір технологій розробки серверної частини

Node.js є середовищем виконання JavaScript з відкритим вихідним кодом, яке працює на сервері та дозволяє розробникам виконувати код JavaScript поза межами веб-браузера. Він базується на рушії JavaScript V8, який було створено Google. Node.js використовує асинхронну, подієву, неблокуючу модель вводу-виводу, що робить його ефективним та високо масштабованим рішенням для розробки серверних програм. У основу розробки серверної частини покладено саме Node.js [11].

Місцем для зберігання даних веб-застосунку було обрано нереляційну базу даних MongoDB. Вона є документо-орієнтованою базою даних, що означає, що дані зберігаються у форматі JSON-подібних документів. Це дозволяє зберігати складні, вкладені структури даних без необхідності суворого визначення схем, як у реляційних базах даних. Це надає змогу легко додавати нові поля до документів без оновлення всієї схеми бази даних, що забезпечує високу гнучкість у розробці і швидку адаптацію до змін у вимогах. Також, одною з переваг MongoDB можна зазначити простий і зрозумілий API для розробників, що дозволяє швидко розпочати роботу з базою даних без необхідності вивчати складні реляційні моделі [12].

Express.js – це один із найпопулярніших фреймворків для Node.js, який надає потужні інструменти та функції для створення серверної частини веб-додатків і API. Він має простий та зрозумілий API, що робить його легким для вивчення і використання навіть для новачків. Це дозволяє розробникам швидко розпочати розробку веб-додатків без необхідності глибокого вивчення складних концепцій. Express.js хоч і є мінімалістичним фрейворком, який пропонує

базовий набір можливостей, проте він надає потужні можливості для маршрутизації запитів, що дозволяє легко створювати та керувати маршрутами для різних HTTP методів (GET, POST, PUT, DELETE тощо).

Ці технології взаємодіють, надаючи засоби створення та розгортання веб-додатків. Потужна та швидка нереляційна база даних MongoDB у співпраці з функціональним та зрозумілим фреймворком Express.js надають прекрасну можливість створення REST API архітектури.

1.4.2 Вибір технологій розробки клієнтської частини

JavaScript є однією з найпопулярніших і найпоширеніших мов програмування у світі. Вона використовується для створення інтерактивних веб-сторінок, серверних додатків, мобільних додатків і навіть для програмування апаратних засобів. JavaScript працює відмінно з HTML та CSS, дозволяючи створювати динамічні та інтерактивні веб-сторінки. Він виконується в браузері користувача, що дозволяє створювати швидкі та відгукові веб-додатки. Також, з переваг можна зазначити те, що він дозволяє змінювати вміст і структуру веб-сторінок в реальному часі. Це включає в себе динамічне оновлення тексту, зміни стилів, створення інтерактивних елементів (наприклад, випадаючих меню, слайдерів, модальних вікон) та обробку подій (кліки, введення даних, тощо). JavaScript має велику спільноту розробників, а також широкий вибір бібліотек і фреймворків, таких як React, Angular, Vue.js, які полегшують розробку складних додатків [13]. У нашому веб-застосунку було використано фреймворк React.

React – це популярна бібліотека JavaScript для створення користувацьких інтерфейсів, розроблена і підтримувана Facebook. Вона дозволяє розробникам будувати швидкі, динамічні і зручні у підтримці веб-додатки. React використовує архітектуру, засновану на компонентах, де елементи користувацького інтерфейсу формуються як конструктивні елементи. Ці елементи можна об’єднувати для створення більш складних користувацьких інтерфейсів. Окремої уваги варта реалізація Document Object Model (DOM), яку представили

розробники React [14]. При створенні компонентів у React, замість безпосередньої маніпуляції реальним DOM, він створює віртуальне під дерево (Virtual DOM), яке є абстракцією реального DOM. Замість того, щоб оновлювати кожен елемент окремо в реальному DOM, React здійснює мінімальну кількість змін, які потрібно зробити, для синхронізації віртуального та реального DOM. Це дозволяє розробникам писати код, який зручно і ефективно оновлює інтерфейс, не переживаючи про високу вартість операцій з DOM, що відбуваються позаду куліс [15].

1.5 Висновок до першого розділу

В першому розділі кваліфікаційної роботи була проведена детальна аналітика обраної тематики та аналіз наявних рішень, що створило перелік вимог, яким повинен відповідати веб-застосунок для створення конкуренції.

Також, у розділі було окреслено ключові елементи веб-застосунку, включаючи каталог товарів, кошик користувача, список вподобань, оформлення замовлення, пошук товарів та інше. Для отримання доступу до веб-сайту користувачу не потрібно виконувати авторизацію, доступ до покупки товарів може відбуватись і без наявного профілю користувача, хоча можливість його створення також представлена.

Вибір технологій розробки мав ключове значення у розробці веб-застосунку. Для реалізації серверної платформи було обрано Node.js із використанням фреймворку Express.js. Клієнтська частина ж використовує головний фреймворк React із допоміжними бібліотеками Axios, SWR та препроцесор SCSS. Зв’язок з базою даних MongoDB встановлюється за допомогою бібліотеки Mongoose. Середовищем розробки було обрано Visual Studio Code.

РОЗДІЛ 2. ПЛАНУВАННЯ СТРУКТУРИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ОНЛАЙН ПРОДАЖ ТОВАРІВ ЕЛЕКТРОНІКИ

2.1 Встановлення архітектури веб-застосунку

На основі аналізу предметної області та описаних сценаріїв використання, можемо прийти до висновку, що оптимальною структурою для розробки цього веб-застосунку буде трирівнева архітектура, схема якого представлена на рисунку 2.1. Він включає в себе три рівні: рівень клієнта (presentation layer), структурно-логічний рівень (logical layer) та рівень даних (data access layer) [16].

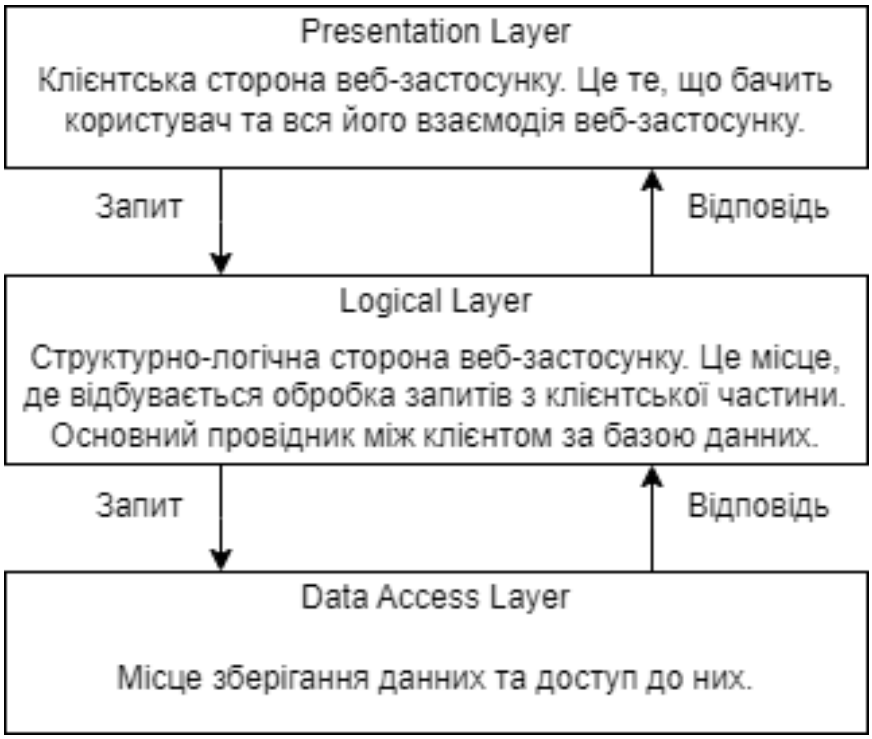

Рисунок 2.1 – Структура архітектури веб-застосунку

Рівень клієнта в веб-застосунку - це та частина, з якою користувач взаємодіє безпосередньо. Він включає в себе інтерфейс користувача, який створюється за допомогою фреймворку React. Цей фреймворк дозволяє створювати динамічні, інтерактивні компоненти та сторінки.

На рівні клієнта також відбуваються запити до серверної частини. Це досягається за допомогою бібліотек Axios та SWR. Axios - це відкрита бібліотека, яка дозволяє робити асинхронні HTTP-запити до REST-кінцевих точок у браузері та Node.js. Вона є обіцянкою на основі HTTP-клієнта, який може бути використаний як і у простому JavaScript так і у сучасних JavaScript-фреймворках. SWR від Vercel є потужним інструментом для управління станом даних у React-додатках, який забезпечує ефективне кешування, автоматичне оновлення даних і простоту інтеграції. Використання SWR дозволяє розробникам зосередитися на функціональності додатка, знижуючи складність роботи з асинхронними запитами і підвищуючи загальну продуктивність та зручність користувача [17].

Всі компоненти, створені на рівні клієнта, стилізуються за допомогою препроцесора SCSS. SCSS - це професійний рівень CSS, який включає в себе всі можливості CSS3 та додає власні нововведення. Він додає багато корисних функцій, таких як змінні, вкладеність, міксини, спадкування та інше, що робить CSS більш читабельним, гнучким та повторно використовуваним.

Серверна частина являє собою структурно-логічний рівень, що є основою веб-застосунку. На цьому рівні відбувається обробка запитів з клієнтської частини та запити до бази даних. Серверний рівень представлений головним фреймворком Express, який являється популярним рішенням для створення гнучкого та зрозумілого REST API.

Робота з базою даних MongoDB відбувається за допомогою офіційної бібліотеки Mongoose, яка дозволяє створювати схеми даних. На основі таких схем у подальшому створюються моделі, які в майбутньому будуть відповідати документам, що знаходяться в MongoDB.

Таким чином, вибір архітектури веб-застосунку дозволяє структуровано підходити до розробки, розподіляючи функціональність між окремими компонентами. Детальний розгляд і опис складових архітектури допомагає краще зрозуміти, як додаток буде функціонувати, і полегшує процес його розробки, тестування та підтримки.

2.2 Опис структури веб-застосунку

Створення структури веб-застосунку є складним і багатоступінчастим процесом, який вимагає ретельного планування і продуманості. Правильна структура забезпечує зручність розробки, підтримки і масштабування додатка. Використання сучасних технологій, фреймворків і інструментів значно полегшує цей процес і допомагає створювати високоякісні веб-додатки.

2.2.1 Опис структури клієнтського рівня

Визначення структури клієнтського рівня є важливою складовою у створені веб-застосунку. Потрібно розробити просту, інтуїтивно зрозумілу структуру веб-сторінок для ефективного користування веб-додатком. Розроблений веб-застосунок повинен включати в себе такі види сторінок: головна сторінка, сторінка продуктів, сторінка продукту, сторінка кошику, сторінка списку бажань. Також, на кожній сторінці веб-застосунку використовуватимуться такі компоненти як заголовок (Header) та футер (Footer).

Заголовок матиме в собі такі компоненти:

– Логотип магазину.
– Навігаційне меню (кошик, список бажань, профіль користувача).
– Пошуковий рядок.

В свою чергу футер містить наступні компоненти:

– Логотип магазину.
– Навігаційне меню (категорії, кошик, список бажань).
– Контактна інформація.

Головна сторінка буде представлена такими компонентами:

– Банер: спеціальні пропозиції або знижки, промо зображення або відео.
– Розділ продуктів: популярні продукти, нещодавно додані, тощо.
– Про нас: інформація про компанію.

Сторінка продуктів використовує такі компоненти:

– Фільтри та сортування: фільтрація за категоріями продуктів, ціновим діапазоном, сортування за зниженням та зростанням ціни.

– Список продуктів: зображення продукту, назва продукту, короткий опис, ціна, кнопки «Додати в кошик» та «Додати до списку бажань».

– Пагінація: кнопки для переходу між сторінками списку продуктів.

Сторінка продукту має таку структуру:

– Основна інформація: зображення продукту, назва продукту, опис продукту, ціна, кнопки «Додати в кошик» та «Додати до списку бажань».

– Додаткова інформація: рекомендовані продукти, специфікації.

Основні компоненти сторінки кошику:

– Список продуктів у кошику: зображення продукту, назва продукту, вибір кількості, ціна вибраного товару, кнопка видалення з кошику.

– Загальна вартість: ціна за обрані товари, кнопка оформлення замовлення.

Представлення сторінки списку бажань:

– Список вподобаних продуктів: зображення продукту, назва продукту, ціна вибраного товару, кнопки додавання та видалення зі списку бажань.

На рисунку 2.2 зображено діаграму опису структури

Рисунок 2.2 – Діаграма опису структури клієнтської частини

Кожна сторінка інтернет-магазину має свою унікальну структуру і компоненти, що забезпечують зручність користування для відвідувачів. Головна сторінка привертає увагу користувачів та надає швидкий доступ до основних розділів, сторінка продуктів дозволяє легко знайти та відфільтрувати необхідні товари, сторінка продукту надає детальну інформацію про кожен товар, сторінка кошику допомагає швидко переглянути і оформити замовлення, а сторінка списку бажань дозволяє користувачам зберігати цікаві їм товари для майбутніх покупок.

2.2.2 Опис структури серверного рівня

Опис структури серверного рівня є важливим етапом у процесі розробки веб-застосунку, оскільки він визначає, як застосунок буде обробляти запити, виконувати бізнес-логіку та взаємодіяти з базою даних. Він допомагає краще розуміти, як застосунок буде функціонувати, і дозволяє планувати та оптимізувати процеси, що відбуваються на сервері. Крім того, він сприяє створенню ефективних, надійних та масштабованих веб-застосунків, що можуть впоратися з великим обсягом даних та високим навантаженням, що є особливо важливим для інтернет-магазинів. Нарешті, він допомагає забезпечити безпеку даних користувачів, що є критично важливим аспектом будь-якого веб-застосунку.

Обраним рішенням створення серверної частини стало використання технології REST API (Representational State Transfer Application Programming Interface) – це стандарт архітектури програмного забезпечення для створення веб-сервісів, які спілкуються між собою за допомогою протоколу HTTP. Основними принципами REST є використання стандартних HTTP методів (GET, POST, PUT, DELETE) для доступу до і керування даними, представлення ресурсів у вигляді URI (Uniform Resource Identifier), а також використання репрезентаційних форматів, таких як JSON або XML, для обміну даними між клієнтом і сервером. REST API використовується для забезпечення взаємодії між

різними програмними системами через інтернет. Він дозволяє зручний доступ до ресурсів (наприклад, дані про користувачів, продукти в інтернет-магазині) та їх керування з боку клієнтських додатків. REST API є популярним вибором для розробки мікросервісних архітектур і дозволяє створювати гнучкі та розширювані системи. Головна перевага REST API полягає в його простоті, масштабованості та стандартизації, що сприяє швидкому розвитку та інтеграції програмних рішень [18].

Реалізація REST API полягає у створені уніфікованих інтерфейсів (Uniform Interface), це означає, що кожен ресурс повинен мати унікальний ідентифікатор у вигляді URI, що спрощує отримання або зміну ресурсу. У створюваному веб-застосунку будуть використовуватись уніфіковані інтерфейси для отримання інформації про: категорії, товари, кошик, список бажань, пошук товарів та авторизацію користувача.

Першим розглянемо уніфікований інтерфейс який буде представлений у веб-застосунку – категорії. Отримати доступ до нього можна за допомогою URI параметру categories. Він включає в себе лише два GET методи, такі як:

– Отримання списку усіх доступних категорій на веб-сайті.

– Отримання списку усіх продуктів, доступних за обраною категорією.

Наступний інтерфейс буде зосереджений на обробці продуктів. URI параметр для доступу до нього – products. Хоча він також включає виключно GET методи, їх тут представлено вже чотири:

– Отримання групи рекомендованих продуктів.

– Отримання групи нещодавно доданих продуктів.

– Отримання усіх доступних товарів.

– Отримання конкретного товару.

Далі розглянемо інтерфейс, який спеціалізується на обробці кошика покупок. Для його використання ми використовуємо URI параметр cart. Він не обмежується лише GET методами, але також включає PUT та DELETE методи:

– GET метод: отримання кошика покупок користувача.

– PUT метод: оновлення товарів у кошику, та створення кошику за відсутністю товарів у ньому.

– DELETE метод: видалення товару з кошика.

Наступний на черзі є інтерфейс, який своєю функціональністю схожий на інтерфейс кошика покупок, але з додатковим методом. Для доступу до нього використовується URI параметр wishlist.

– GET метод: отримання списку бажань користувача.

– GET метод: отримання певного продукту зі списку бажань користувача.

– PUT метод: оновлення товарів у списку бажань, створення списку за відсутністю товарів у ньому.

– DELETE метод: видалення товару зі списку бажань.

Черговим уніфікованим інтерфейсом, який ми використовуємо, є пошук, з його URI параметром - search. Цей інтерфейс є одним з найпростіших у веб-застосунку, оскільки він включає в себе лише один GET метод:

– Отримання продукту за пошуковим параметром.

Останнім інтерфейсом у веб-застосунку, який ми розглянемо, є авторизація. Для доступу до нього ми використовуємо URI параметр auth. Цей інтерфейс є одним з найбільш складних серед описаних, оскільки в ньому реалізована логіка авторизації користувача. Це дозволяє користувачам отримувати доступ до покупки товарів без необхідності реєстрації. Детальніше розглянемо методи, які він включає:

– GET метод: аутентифікація користувача, визначення статусу користувача - чи він зареєстрований, чи є він гостем.

– POST метод: реєстрація нового користувача.

– POST метод: вхід користувача в систему.

– DELETE метод: вихід користувача з системи.

У висновку, структура REST API для веб-застосунку інтернет-магазину демонструє продуману організацію і стандартизацію інтерфейсів для різних аспектів функціональності. Інтерфейс категорій (categories) дозволяє отримувати списки категорій і продуктів за категоріями, забезпечуючи користувачам зручну

навігацію. Інтерфейс продуктів (products) розширює можливості отримання інформації про рекомендовані, нещодавно додані та всі доступні товари. Інтерфейс кошика покупок (cart) включає додаткові методи для керування товарами, дозволяючи користувачам оновлювати або видаляти товари з кошика. Аналогічно, інтерфейс списку бажань (wishlist) забезпечує управління списком бажаних товарів. Простий і ефективний інтерфейс пошуку (search) дозволяє швидко знаходити потрібні продукти за пошуковими параметрами. Найскладнішим є інтерфейс авторизації (auth), який реалізує логіку аутентифікації і управління користувачами, включаючи реєстрацію, вхід і вихід з системи. Всі ці інтерфейси разом забезпечують цілісну, гнучку і масштабовану платформу для інтернет-магазину, сприяючи зручності користування і ефективності взаємодії між клієнтами і сервером.

2.3 Проектування бази даних

Як було зазначено в першому розділі даної кваліфікаційної роботи, для проектування бази даних обрано одну з найпопулярніших нереляційних баз даних – MongoDB. Цей вибір обґрунтований її гнучкістю, високою продуктивністю та масштабованістю, що робить її ідеальною для сучасних веб-застосунків. MongoDB дозволяє зберігати дані у форматі JSON-подібних документів, що забезпечує легкість інтеграції з різними мовами програмування та простоту у використанні.

Після детального огляду структури веб-застосунку можна зробити висновок, що база даних буде складатися з наступних ключових колекцій:

Продукти (Products): Ця колекція міститиме інформацію про всі товари, доступні на веб-сайті. Кожен документ у цій колекції буде включати поля для зберігання детальної інформації про продукт, такі як унікальний ідентифікатор, назва, опис, ціна, бренд, категорія, зображення, специфікації, дата створення та дата оновлення.

На рисунку 2.3 зображено схему колекції продуктів.

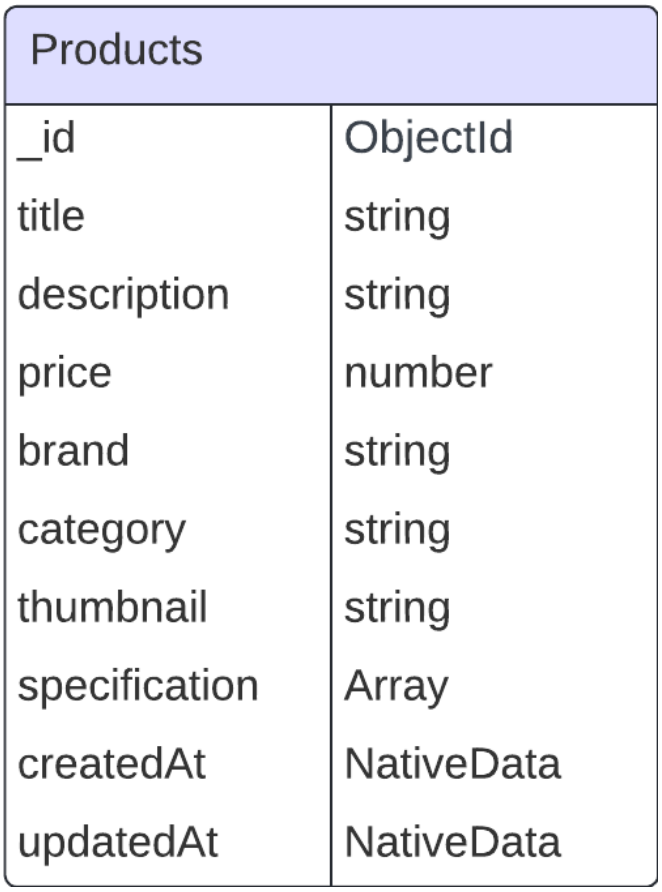

Рисунок 2.3 – Схема колекції продуктів в базі даних

Схема продуктів має такі поля:

– _id: унікальний ідентифікатор товару.
– title: назва товару.
– description: опис товару.
– price: ціна товару.
– brand: бренд виробника товару.
– category: категорія до якої відноситься товар.
– thumbnail: зображення товару.
– specification: характеристика товару.
– createdAt: дата створення товару в базі даних.
– updatedAt: дата останнього оновлення товару в базі даних.

Кошики (Carts): Колекція кошиків буде зберігати дані про товари, які користувачі додали до своїх кошиків. Це дозволить забезпечити користувачам можливість продовжувати покупки з того місця, де вони зупинилися, навіть після виходу з системи. Кожен документ у цій колекції буде включати унікальний ідентифікатор кошика, ідентифікатор користувача, список товарів у кошику, загальна вартість та кількість кожного товару.

На рисунку 2.4 зображено схему колекції кошику.

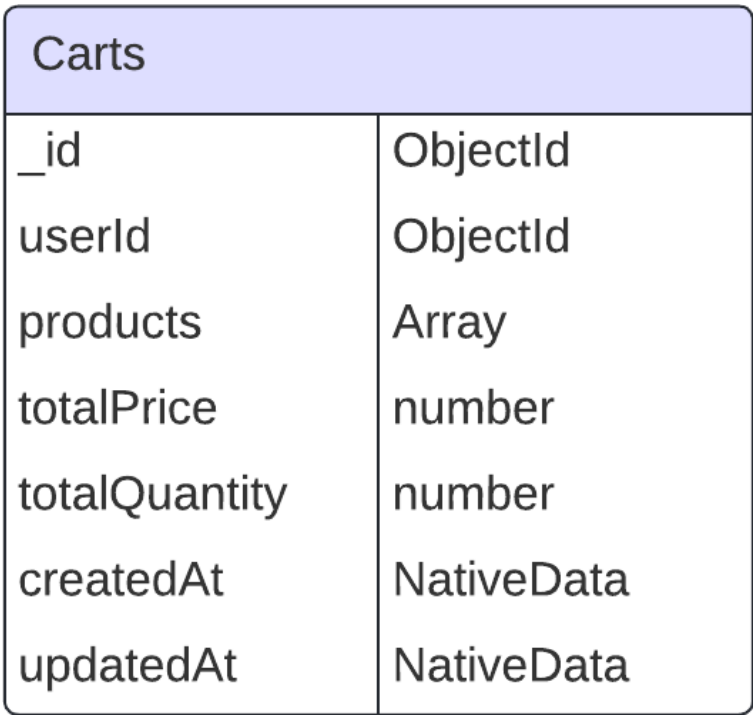

Рисунок 2.4 – Схема колекції кошику в базі даних

Схема кошику має такі поля:

– _id: унікальний ідентифікатор товару.

– userId: унікальний ідентифікатор користувача, який володіє цим кошиком.

– products: список продуктів, які знаходяться у кошику.

– totalPrice: загальна вартість усіх продуктів, що знаходяться у кошику.

– totalQuantity: загальна кількість продуктів у кошику.

– createdAt: дата створення кошику в базі даних.

– updatedAt: дата останнього оновлення кошику в базі даних.

Списки бажань (Wishlists): Ця колекція буде використовуватися для зберігання списків бажань користувачів. Користувачі зможуть додавати до списку бажань ті товари, які вони хочуть придбати у майбутньому. Документи у цій колекції включатимуть унікальний ідентифікатор списку бажань, ідентифікатор користувача і список товарів, доданих до списку.

На рисунку 2.5 зображено схему колекції списку бажань.

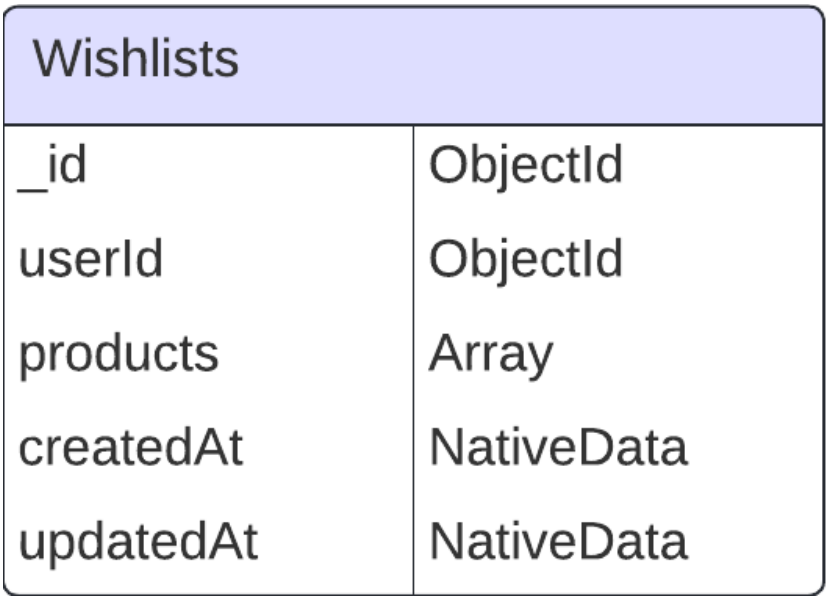

Рисунок 2.5 – Схема колекції списку бажань в базі даних

Схема списку бажань має такі поля:

– _id: унікальний ідентифікатор товару.

– userId: унікальний ідентифікатор користувача, який володіє цим кошиком.

– products: список продуктів, які знаходяться у списку бажань.

– createdAt: дата створення списку бажань в базі даних.

– updatedAt: дата останнього оновлення списку бажань в базі даних.

Список зареєстрованих користувачів (AuthUsers): Колекція користувачів міститиме інформацію про зареєстрованих користувачів веб-застосунку. Кожен документ буде включати дані про користувача, такі як унікальний ідентифікатор, ім'я, електронна адреса, пароль (у зашифрованому вигляді), дата реєстрації.

На рисунку 2.6 зображено схему колекції зареєстрованих користувачів.

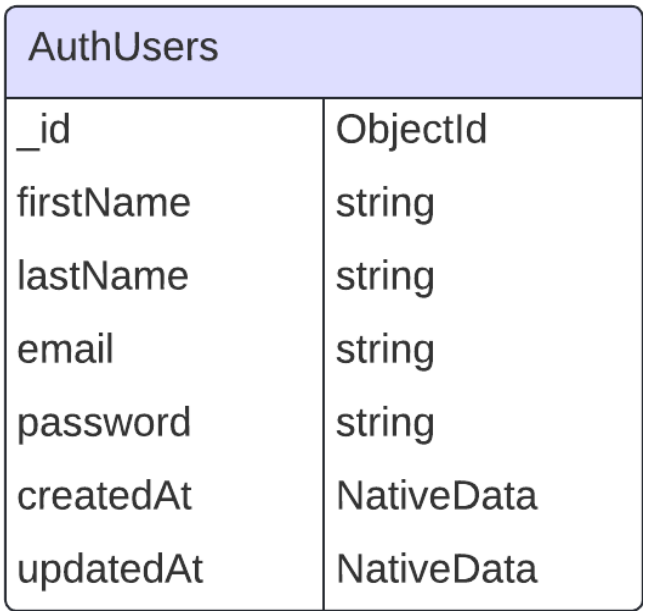

Рисунок 2.6 – Схема колекції зареєстрованих користувачів в базі даних

Схема зареєстрованих користувачів має такі поля:

– _id: унікальний ідентифікатор користувача.
– firstName: ім’я користувача.
– lastName: прізвище користувача.
– email: електрона адреса користувача.
– password: пароль користувача (у зашифрованому вигляді).
– createdAt: дата створення користувача в базі даних.
– updatedAt: дата останнього оновлення користувача в базі даних.

Для подальшої оптимізації та розуміння взаємозв'язків між цими колекціями планується створення схеми зв'язків. Вона дозволить нам візуально представити, які колекції пов'язані між собою і яким чином. Наприклад, з'єднання між колекціями Products і Carts вказуватиме на те, які продукти додані в кошики користувачів. Аналогічно, взаємозв'язки між Products і Wishlists покажуть, які товари додані до списків бажань. Схема буде корисним інструментом для вдосконалення структури бази даних, оптимізації запитів і покращення продуктивності системи в цілому [19].

На рисунку 2.7 представлено вигляд такої схеми.

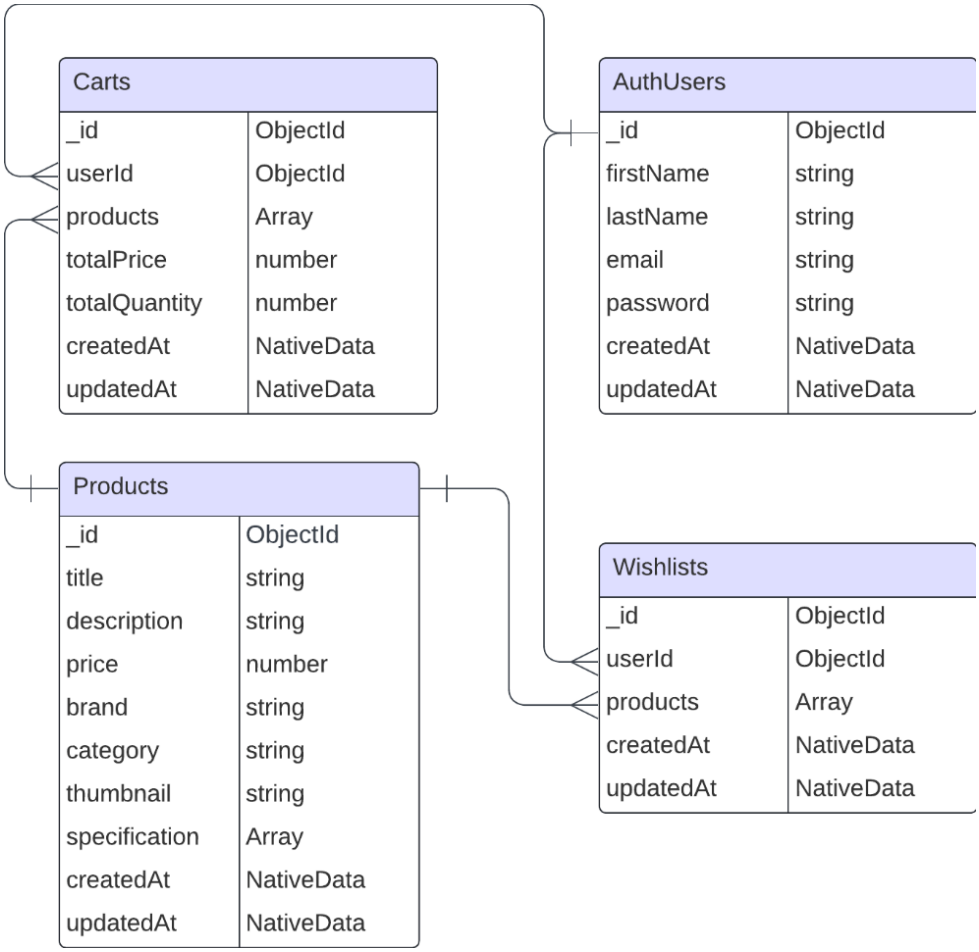

Рисунок 2.7 – Схема зв’язків між колекціями в базі даних

Наразі ми успішно спроектували колекції в нашій базі даних для різних об'єктів системи. У колекції Products ми зберігаємо інформацію про продукти, таку як назва, опис, ціна та інші характеристики. Для управління покупками і бажаннями користувачів ми створили колекції Carts і Wishlists, де зберігається інформація про товари, які користувачі додали до свого кошика або бажань. Крім того, є колекція AuthUsers, де зберігаються дані про авторизованих користувачів, такі як ім'я, електронна пошта, пароль (захешований) та інші дані профілю. Ці колекції сприяють організації та оптимізації доступу до даних і взаємодії в рамках програмної системи, підвищуючи її функціональність та ефективність.

2.4 Висновок до другого розділу

У другому розділі вашої кваліфікаційної роботи було встановлено архітектуру веб-застосунку, якою стала трирівнева архітектура через її численні переваги. Ця архітектура дозволяє розділити логіку вашого застосунку на три основні рівні: презентаційний, бізнес-логіки та даних, що сприяє більшій гнучкості та масштабованості.

Також було описано структуру клієнтської та серверної частини веб-додатку, включаючи основні компоненти та методи їх використання. Це включає в себе різні аспекти взаємодії користувача з веб-застосунком, а також логіку обробки даних на сервері.

Далі було спроектовано колекції бази даних, які будуть використовуватись у створенні веб-застосунку. Кожна колекція була детально спроектована та описана, включаючи її поля. Це включає в себе колекції для продуктів, кошиків, списків бажань та списку користувачів. Колекції дозволять зберігати всю необхідну інформацію для роботи веб-застосунку, включаючи деталі про продукти, інформацію про покупки користувачів, їхні списки бажань та особисті дані.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ОНЛАЙН ПРОДАЖ ТОВАРІВ ЕЛЕКТРОНІКИ

3.1 Розробка серверної частини веб-застосунку

Після визначення усіх необхідних технологій для розробки веб-застосунку, настав час переходити до їх реалізації. Першим етапом цього процесу буде створення серверної частини застосунку. Цей етап включатиме в себе реалізацію різних функцій та методів, які були описані в другому розділі даної кваліфікаційної роботи.

Серверна частина веб-застосунку відіграє важливу роль, оскільки вона відповідає за обробку запитів від клієнтів, управління даними та забезпечення взаємодії між різними частинами системи. Вона також відповідає за забезпечення безпеки даних користувачів та захисту від несанкціонованого доступу. Під час створення серверної частини веб-застосунку також буде створено базу даних, використовуючи колекції, які були описані в другому розділі роботи. Кожна колекція в базі даних представляє собою групу документів, які містять інформацію про певний аспект веб-застосунку, такий як продукти, кошики, списки бажань або користувачі. Ці колекції дозволять зберігати, оновлювати та отримувати дані ефективно та безпечно.

В процесі реалізації цих компонентів будуть використовуватись різні технології та методи, які були вибрані на початкових етапах проекту. Вони включають в себе використання таких технологій, як Node.js для створення серверної частини, Express.js для обробки HTTP-запитів, а також MongoDB для створення та управління вашою базою даних.

Цей етап розробки є критично важливим, оскільки від правильної реалізації серверної частини та бази даних залежить стабільність і продуктивність всього веб-застосунку. В результаті реалізації буде створена надійна основа, яка забезпечить коректну роботу клієнтської частини, високу продуктивність та безпеку системи в цілому.

3.1.1 Ініціалізація серверної частини проекту

Ініціалізація серверної частини у середовищі розробки VSCode з використанням Express.js на основі пакетного менеджера yarn включає кілька основних кроків. Нижче описано детальний процес:

– Відкриття проекту у VSCode: Першим кроком є відкриття робочої директорії у VSCode.

– Ініціалізація нового проекту Yarn: Далі, відкриваємо термінал у VSCode (View -> Terminal) та переходимо до директорії проекту (якщо ви вже не в ній). Виконуємо команду yarn init [20]. Ця команда створить новий файл package.json у директорії проекту, який буде використовуватись для керування залежностями проекту. Файл package.json є важливим елементом будь-якого проекту на Node.js, оскільки він зберігає інформацію про проект та його залежності.

– Встановлення залежностей: Наступним кроком є встановлення залежностей, які використовуються при створені веб-застосунку, такі як Express.js, mongoose та інші. Виконуємо команду yarn add express mongoose у терміналі. Ця команда встановить усі необхідні залежності у проект та додасть його до списку у файлі package.json.

Список усіх встановлених залежностей представлено на рисунку 3.1.

Рисунок 3.1 – Список встановлених залежностей серверної частини

Ініціалізація серверної частини у середовищі розробки VSCode з використанням Express.js та пакетного менеджера Yarn є відносно простою, але важливою процедурою, яка включає встановлення необхідних інструментів та налаштування середовища для подальшої розробки.

3.1.2 Створення серверного файлу

Після ініціалізації серверної частини проекту та встановлення усіх необхідних залежностей, необхідно створити новий файл у директорії проекту, у нашому випадку це буде index.ts. Розширення .ts позначає, що файл створений з використанням TypeScript – надбудовою JavaScript, що додає статичну типізацію та інші функції, які допомагають писати більш надійний та чистий код [21]. Створений файл буде “вхідною точкою” сервера, місцем, де будуть визначені всі маршрути та обробники запитів для веб-застосунку. Далі імпортуємо модулі Express.js та mongoose до файлу та створюємо новий екземпляр Express, який будете використовуватись для визначення маршрутів та обробки запитів. Також, встановлюємо зв’язок з базою даних та стартуємо сервер, попередньо обравши порт, на якому сервер буде прослуховувати вхідні запити, у нашому випадку було обрано порт 4000. Після успішного запуску стартового серверу у терміналі з’явиться повідомлення про це, яке зображено на рисунку 3.2.

```
yarn run v1.22.22
$ D:\JS\nodejs\mern-app\server\node_modules\.bin\nodemon .\dist\index.js
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node .\dist\index.js`
Server starts at port 4000
```

Рисунок 3.2 – Повідомлення про успішний старт серверу

У процесі ініціалізації серверної частини проекту було встановлено всі необхідні залежності та створено новий файл index.ts у директорії проекту, тому перейдемо до створення бази даних.

3.1.3 Створення бази даних

Після створення вхідного файлу для сервера, наступним важливим кроком є реалізація бази даних. Як було вказано у другому розділі даної кваліфікаційної роботи, база даних має включати в себе такі колекції: products, carts, wishlists та authusers.

Для відокремлення файлової структури проекту та забезпечення чистоти та організованості коду, створюємо окрему папку з назвою models. У цій папці створюємо наступні файли: ProductModel.ts, CartModel.ts, WishlistModel.ts, AuthUserModel.ts.

У цих файлах, використовуючи бібліотеку mongoose – потужний інструмент для роботи з MongoDB в середовищі Node.js. За допомогою методу new Schema, який надає ця бібліотека, створюємо схеми для ваших колекцій. Схема визначає структуру документів у колекції MongoDB і використовується для валідації даних перед їх збереженням у базу даних. Кожна схема визначає поля документа, їх типи даних, а також може включати додаткові параметри, такі як обов’язкові поля, значення за замовчуванням, валідатори тощо. Це допомагає забезпечити консистентність даних та полегшує роботу з базою даних [22].

В результаті, отримуємо добре структуровану базу даних з чітко визначеними схемами для кожної колекції, що сприяє ефективності та надійності веб-застосунку.

3.1.4 Реалізація кінцевих точок веб-застосунку

Кінцеві точки, або ендпоінти (endpoint) є важливими елементами будь-якого веб-застосунку. Вони представляють собою специфічні URL-адреси, до

яких клієнт може відправляти запити для отримання або відправлення певної інформації [23]. У контексті веб-застосунку, як було зазначено у другому розділі роботи, серверна частина повинна мати такі ендпоінти: categories, products, cart, wishlist та auth.

Для реалізації цих ендпоінтів створюємо окрему папку routes. У цій папці формуємо файли: categories.ts, products.ts, cart.ts, wishlist.ts та auth.ts. Кожен з цих файлів відповідає за реалізацію певного ендпоінта, що дозволяє веб-застосунку взаємодіяти з користувачами та обробляти їхні запити.

Основний функціонал цих файлів реалізується на аналізі URI параметр та на його основі надається потрібна інформація з бази даних. Для цього використовуються методи find(), findOne(), findById(), які надає mongoose для пошуку даних в базі. Для оновлення даних в базі використовується метод findOneAndUpdate().

Детальніше розглянемо створені ендпоінти у файлах. Розпочнемо з огляду файлу categories.ts:

– GET /categories – отримання усіх доступних категорій на веб-сайті.

– GET /categories/:category – отримання усіх продуктів, доступних за обраною категорією.

Наступний файл – products.ts:

– GET /products – отримання усіх доступних товарів.

– GET /products/:id – отримання конкретного товару за його унікальним ідентифікатором.

– GET /products/recommended/:id – отримання групи рекомендованих продуктів на основі обраного товару за його унікальним ідентифікатором.

– GET /products/newarrivals – отримання групи нещодавно доданих товарів.

Далі на черзі файл cart.ts:

– GET /cart/:userId – отримання кошика покупок користувача за його ідентифікатором.

– PUT /cart/:userId – оновлення товарів у кошику, та створення кошику за відсутністю товарів у ньому за ідентифікатором користувача.

– DELETE /cart/:userId/:productId – видалення товару з кошика за ідентифікаторами користувача та продукту.

Наступний розглянемо файл wishlist.ts:

– GET /wishlist/:userId – отримання списку бажань користувача за його ідентифікатором.

– GET /wishlist/:userId/:productId – отримання конкретного продукту зі списку бажань за ідентифікаторами користувача та продукту.

– PUT /wishlist/:userId – оновлення товарів у списку бажань, та створення списку за відсутністю товарів у ньому за ідентифікатором користувача.

– DELETE /wishlist/:userId/:productId – видалення товару зі списку бажань за ідентифікаторами користувача та продукту.

Останнім файлом у нашому проекті є auth.ts. Цей файл відіграє ключову роль, оскільки він включає в себе використання таких технологій, як jwt (JSON Web Token) та cookie.

JWT використовується для створення токенів, які містять інформацію про користувача. Ця інформація береться з нашої бази даних. Токен є цифровим підписом, який підтверджує автентичність користувача і дозволяє нам впевнено ідентифікувати його [24]. Щоб забезпечити довготривале зберігання токену, ми використовуємо cookie файли. Cookie - це невеликі файли, які зберігаються на комп’ютері користувача і дозволяють нам зберігати інформацію про сесію користувача.

Коли неавторизований користувач вперше відвідує наш веб-сайт, ми автоматично створюємо для нього sessionToken. Цей токен містить інформацію, що користувач є гостем на нашому сайті. Для авторизованих користувачів ми створюємо authToken. Цей токен містить детальну інформацію про користувача, включаючи унікальний ідентифікатор користувача, статус (чи є користувач авторизованим або гостем), ім’я, прізвище та електронну адресу. Ця інформація

дозволяє системі ефективно і безпечно управляти доступом до ресурсів та персоналізувати взаємодію з користувачами на веб-сайті.

Процес реєстрації користувача відбувається з використанням бібліотеки bcrypt. Він починається з отримання даних від користувача, таких як ім’я користувача, електронна адреса та пароль. Перед збереженням пароля в базі даних, ми використовуємо bcrypt для його хешування. Це допомагає забезпечити безпеку даних користувача, оскільки навіть у випадку витоку даних, реальний пароль користувача залишається в безпеці. Після цього ми зберігаємо захищені дані користувача в базі даних. Коли користувач намагається увійти, ми порівнюємо введений ним пароль з хешем в базі даних, щоб перевірити його автентичність. Якщо вони співпадають, користувачу надається доступ до системи [25].

Розглянемо ендпоінти файлу auth.ts:

– GET /auth/me – перевіряє наявний в cookie файлах токен, за його відсутності створює його та повертає інформацію про користувача.

– POST /auth/register – створює нового користувача в базі даних.

– POST /auth/login – проводить авторизацію користувача на веб-сайті шляхом порівняння отриманих даних, з наявними в базі даних. Якщо результат успішний – створює файл cookie і вписує в нього створений authToken.

– DELETE /auth/login – вихід користувача з системи шляхом видалення його authToken з cookie файлу.

У результаті була розроблена імплементації кожного ендпоінту для забезпечення ефективної роботи веб-застосунку, а також створена безпека і конфіденційність даних користувачів під час взаємодії з системою.

3.1.5 Реалізація додаткових файлів

У додатковій категорії файлів нашого веб-застосунку ми можемо виділити middleware, який відіграє важливу роль у обробці запитів та відповідей. Middlewware – це важливий компонент будь-якого веб-застосунку, який

допомагає керувати потоком запитів та відповідей, а також забезпечує додатковий рівень безпеки та контролю. Використання middleware дозволяє нам створювати більш гнучкі та масштабовані веб-застосунки [26]. У нашому проекті було використано два таких проміжних файли: auth.ts (авторизаційний) та validation.ts (валідаційний). Для цієї категорії була створена окрема папку middleware, де розміщено відповідні файли.

Файл auth.ts використовується у початковому файлі проекту index.ts для визначення статусу користувача на самому початку користування веб-сайтом. Він містить в собі метод authTokenVerification, який перевіряє, який токен знаходиться у файлах cookie на даний момент. Це дозволяє нам визначити, чи є користувач авторизованим, або ж він є гостем нашого сайту.

Файл validation.ts використовується при валідації даних при зверненні до будь-якого з ендпоінтів. Це означає, що якщо користувач при користуванні веб-сайтом надає некоректні дані, то цей middleware відхиляє подальший запит. Це допомагає забезпечити, що всі дані, які надходять до нашого сервера, відповідають встановленим критеріям, що допомагає запобігти помилкам та забезпечити надійність нашого веб-застосунку.

3.2 Розробка клієнтської частини веб-застосунку

Після успішного створення серверної частини нашого веб-застосунку, наступним важливим кроком є розробка клієнтської частини. Цей процес, як і структура серверної частини, був детально описаний в другому розділі цієї роботи.

Під час розробки клієнтської частини зосередимося на створенні ряду компонентів. Ці компоненти відповідають за візуалізацію веб-застосунку, створюючи інтерфейс, який є інтуїтивно зрозумілим та привабливим для користувача. Окрім цього, розробимо HTTP-запити до ендпоінтів серверної частини. Ці запити дозволять взаємодіяти з сервером, отримувати необхідні дані та відправляти інформацію назад на сервер. Ще одним важливим аспектом

розробки клієнтської частини є втілення дизайну веб-сайту. Завдання полягає створити не тільки функціональний веб-сайт, але й візуально привабливий, що забезпечує приємний досвід користувача. Всі ці елементи будуть реалізовані з використанням різних технологій, головним з яких є React.

3.2.1 Ініціалізація клієнтської частини веб-застосунку

Ініціалізація проекту буде відбуватись у середовищі розробки VSCode на основі інструменту Vite з використанням фреймворку React [27]. Для цього процесу використовуватиметься пакетний менеджер yarn. Далі описано детальний процес налаштування:

– Відкриваємо термінал у VSCode і переходимо в директорію, де створимо проект. Для створення нового проекту за допомогою Vite виконуємо наступну команду: yarn create vite. Після запуску цієї команди Vite запропонує ввести ім'я проекту та вибрати шаблон. Обираємо шаблон React та мову програмування TypeScript.

– Наступним етапом є встановлення залежностей, які необхідні для роботи нашого веб-застосунку. Це включає бібліотеки, такі як axios для виконання HTTP-запитів, swr для кешування даних та sass для написання стилів. Для цього ми використовуємо команду yarn add в терміналі. Ця команда автоматично встановлює всі потрібні залежності в проекті та додає їх до файлу package.json, який відслідковує всі встановлені залежності в нашому проекті.

Список усіх встановлених залежностей представлено на рисунку 3.3.

Рисунок 3.3 – Список встановлених залежностей клієнтської частини

Після виконання цих кроків у вас створився налаштований базовий React-проект, ініціалізований за допомогою Vite. Далі продовжуємо розробку, додаючи нові компоненти, сторінки та функціональність відповідно до вимог вашого веб-застосунку.

3.2.2 Створення сторінок веб-застосунку

Після успішної ініціалізації проекту, наступним кроком є створення головного файлу, який ми назвемо main.tsx. Цей файл є важливим, оскільки він служить точкою входу в наш додаток. Розширення .tsx вказує на те, що цей файл створений на основі фреймворку React з використанням TypeScript.

Структура main.tsx є простою, але важливою. Вона контролюється з використанням роутера, який створений за допомогою бібліотеки react-router-dom. Ця бібліотека дозволяє нам легко створювати багатосторінкові додатки в одній сторінці (SPA), де кожна “сторінка” відповідає різному шляху URL.

Далі створюємо окрему папку pages, в якій будуть розміщені всі майбутні сторінки веб-застосунку. У цій папці створюємо перший файл router.tsx, який відіграє ключову роль у навігації по веб-сайту. Його основний функціонал

полягає в читанні URL-адреси браузера та відображенні відповідної сторінки на основі цієї адреси.

Наступний кроком створюємо ряд маршрутів, кожен з яких буде відповідати певному шляху URL і повертати відповідний React-елемент. Це будуть головна сторінка, сторінка продукту, сторінка корзини, сторінка списку бажань, сторінка авторизації та сторінка помилки. Кожен маршрут буде визначати, який компонент React відображати, коли користувач переходить до відповідного URL [28].

Таким чином, router.tsx стає центральною частиною нашого веб-застосунку, яка керує тим, як користувачі переходять між різними частинами веб-сайту і як вони взаємодіють з ним. Це дозволяє створити більш структурований та організований веб-застосунок, який може легко масштабуватися та адаптуватися до різних потреб користувачів.

Створимо такі шляхи, що будуть на основі адреси повертати React елемент:

– path: “/” – початкова адреса веб-застосунку, це та сторінка яку ми бачимо після входу на веб-сайт. Цей шлях повертає елемент Main.tsx.

– path: “/category/:category” – адреса для перегляду списку продуктів доступних за цією категорією. Повертає елемент Category.tsx.

– path: “/product/:productId” – адреса для перегляду обраного продукту. Елемент, що повертається Product.tsx.

– path: “/cart” – адреса для перегляду кошику замовлень. Тут повертається елемент Cart.tsx.

– path: “/wishlist” – адреса перегляду списку бажань. Вона повертає елемент Wishlist.tsx.

– path: “/login” – адреса для проходження входу в систему. Елемент, що вона повертає має назву Login.tsx.

– path: “/error” – адреса, що повертається кожного разу коли на якійсь із сторінок відбувається збій в роботі. Повертається елемент Error.tsx.

Почнемо з того, що ми створюємо елементи, які повертає наш роутер. Однак перед цим ми повинні розглянути два ключові компоненти: Header та

Footer. Ці компоненти відображаються на кожній сторінці нашого веб-застосунку, створюючи стабільний і зрозумілий інтерфейс для користувачів. Усі наші компоненти зберігаються в окремій папці components, що допомагає утримувати код організованим.

Компонент Header відіграє важливу роль в навігації користувачів. Він містить логотип нашого інтернет-магазину, навігаційне меню категорій, пошуковий рядок, список бажань, кошик та авторизацію. Меню категорій реалізовано через компонент Categories.tsx. Цей компонент містить функціонал запиту до сервера за усіма доступними категоріями та перехід до сторінки перегляду товарів, доступних за обраною категорією. Функціонал пошукового рядку представлений компонентом SearchBar.tsx. Він виконує запит до сервера для отримання товарів, що відповідають тексту, введеному в поле пошуку. Решта елементів в Header - це кнопки, які виконують перехід на відповідні ресурси веб-сайту: список бажань, кошик та авторизацію.

Компонент Footer включає в себе логотип, навігаційне меню та контактну інформацію. Цей компонент має функціональність переходу на обрані елементи навігації, допомагаючи користувачам легко знайти потрібну інформацію.

Почнемо з першого елемента роутеру, Main.tsx. Цей компонент відповідає за головну сторінку нашого веб-застосунку. Він виконує запит до сервера за даними про нещодавно додані товари, які потім відображаються на головній сторінці. Візуалізація головної сторінки відбувається за допомогою ряду компонентів, включаючи MainBanner, NewArrivals, AboutBlock. На рисунку 3.4 зображено результат відображення головної сторінки.

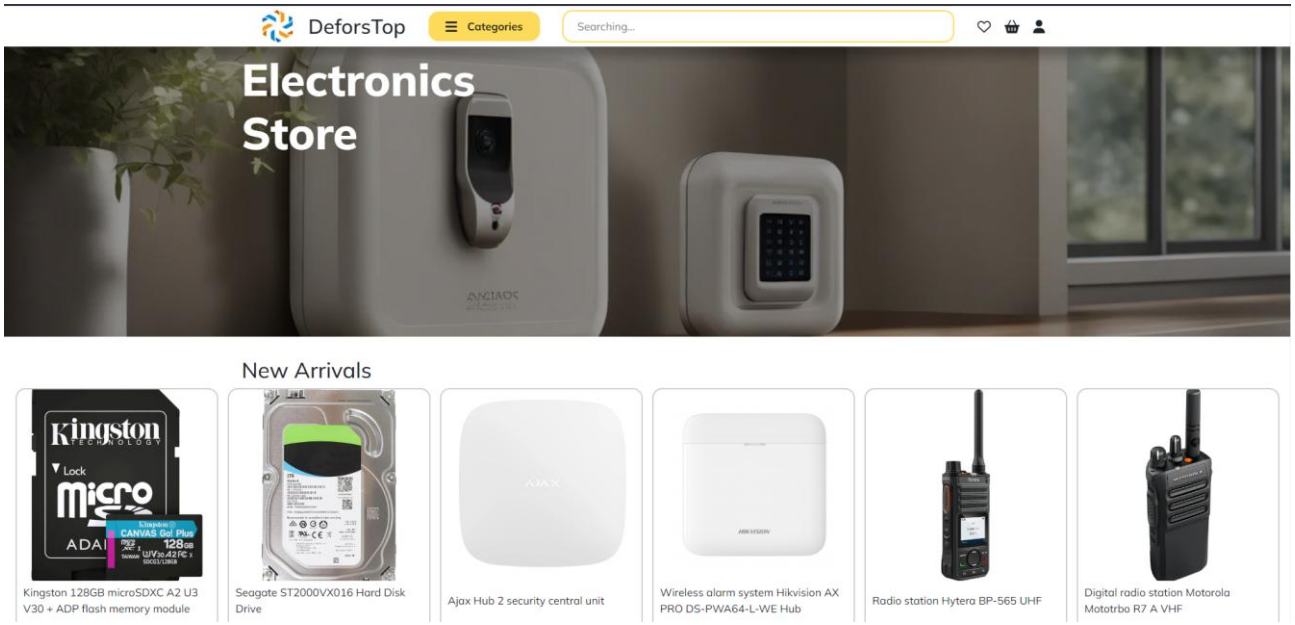

Рисунок 3.4 – Головна сторінка веб-застосунку

Наступний елемент, Category.tsx, відповідає за відображення товарів відповідної категорії. Він виконує запит до сервера для отримання даних про товари, які належать обраній категорії. Після отримання даних, вони передаються в компонент ProductsList, який відповідає за відображення товарів на веб-сторінці в привабливому вигляді.

На рисунку 3.5 зображено результат відображення товарів обраної категорії.

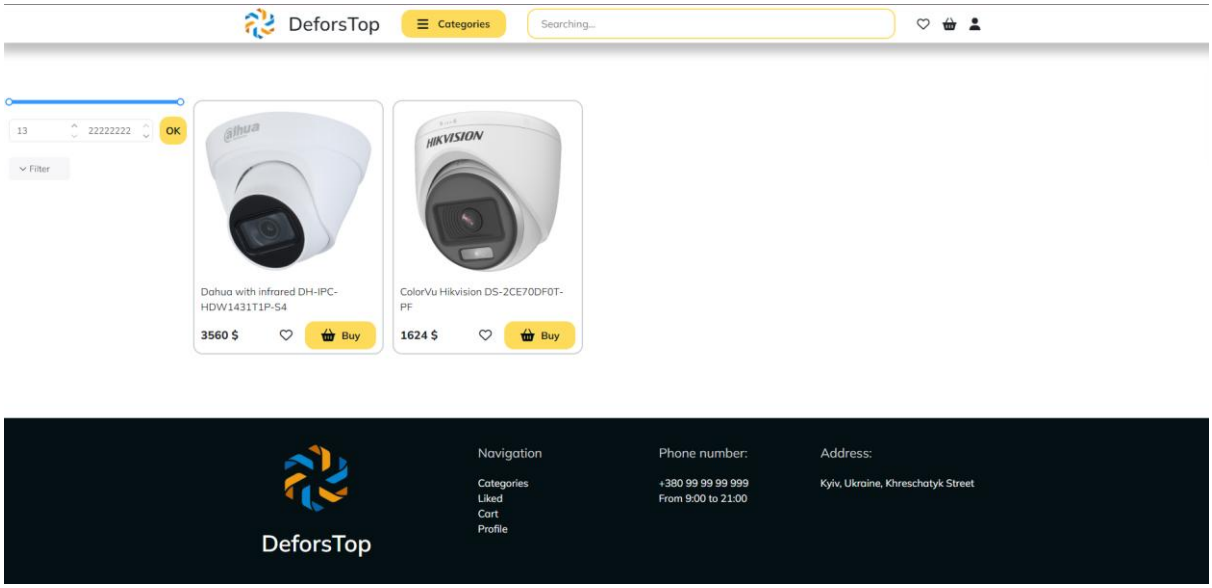

Рисунок 3.5 – Сторінка відображення товарів обраної категорії

Product.tsx - це сторінка відображення окремого товару. Він виконує запит до сервера для отримання детальної інформації про товар, а також даних про рекомендовані товари. Отримані дані передаються в компонент ProductInfo, який відповідає за візуалізацію інформації про товар на сторінці. На рисунку 3.6 можна побачити вигляд на веб-сайті.

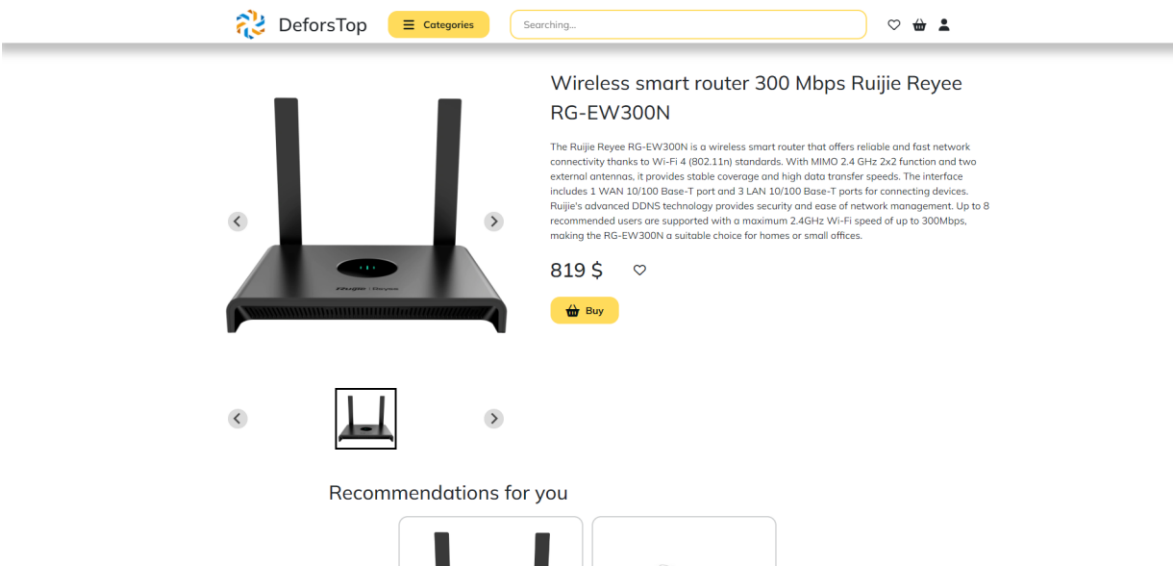

Рисунок 3.6 – Сторінка відображення товару

Cart.tsx - це компонент, який відповідає за відображення корзини покупок користувача. Він використовує унікальний ідентифікатор користувача для виконання запиту до сервера за даними про корзину покупок. Якщо дані про кошик відсутні, відображається компонент NoData, який сповіщає користувача про відсутність товарів в кошику. В іншому випадку, отримані дані передаються в компонент CartInfo і відображаються на веб-сторінці. Результат висвітлено на рисунку 3.7.

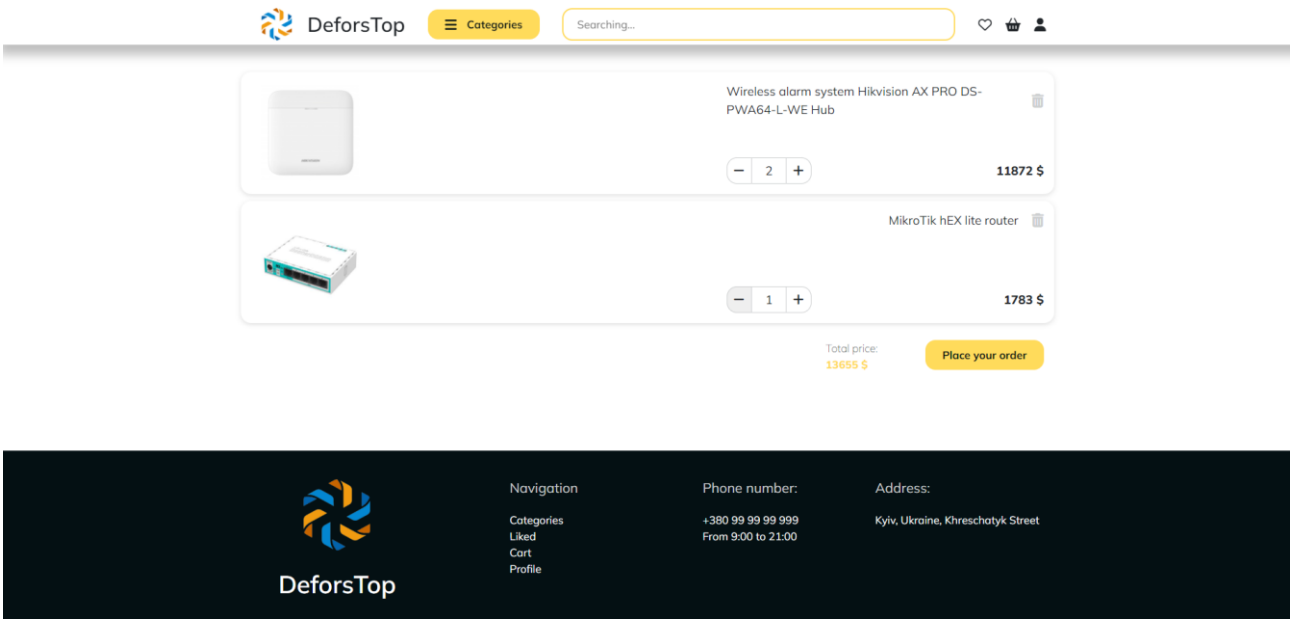

Рисунок 3.7 – Сторінка кошику

Функціональність елементу Wishlist.tsx ідентична до Cart.tsx, але він використовує інший ендпоінт на сервері для отримання даних про список бажань користувача. Отримані дані передаються в компонент WishlistInfo для відображення на веб-сторінці. Не дивлячись на ідентичність функціональності, візуалізація списку бажань відрізняється від кошику товарів. Відображення списку бажань на веб-сайті можна побачити на рисунку 3.8.

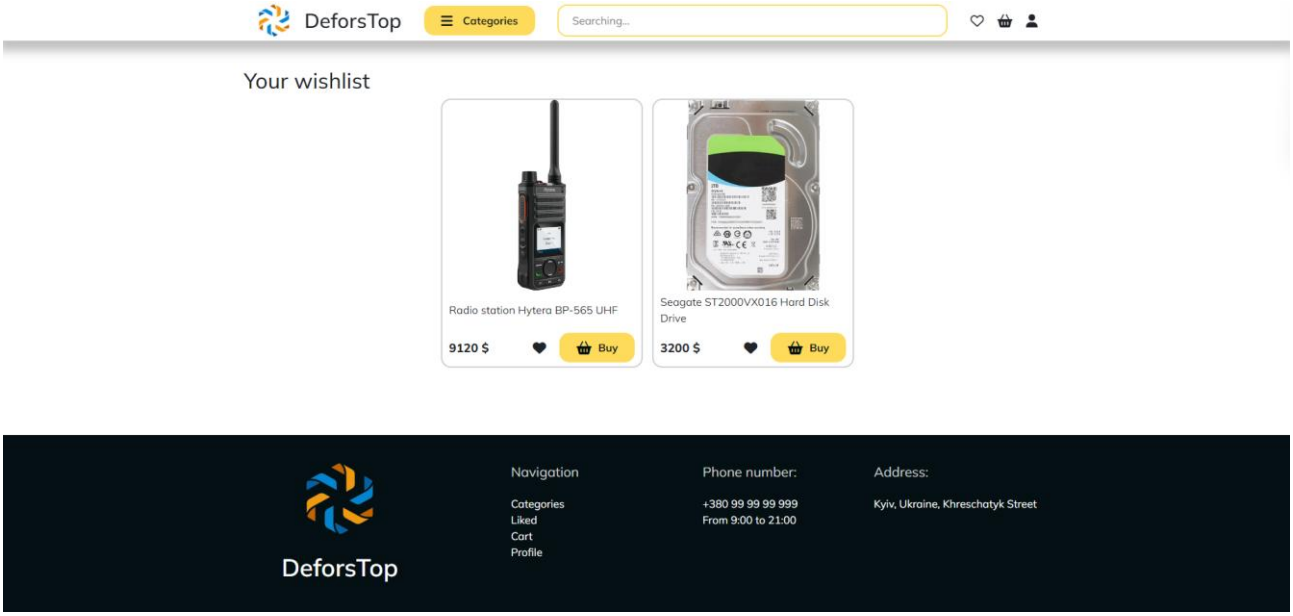

Рисунок 3.8 – Сторінка списку бажань

Login.tsx - це компонент, який відповідає за відображення форми авторизації. Він використовує компонент LoginForm для візуалізації форми авторизації на веб-сторінці. Відображення його на сайті показано на рисунку 3.9.

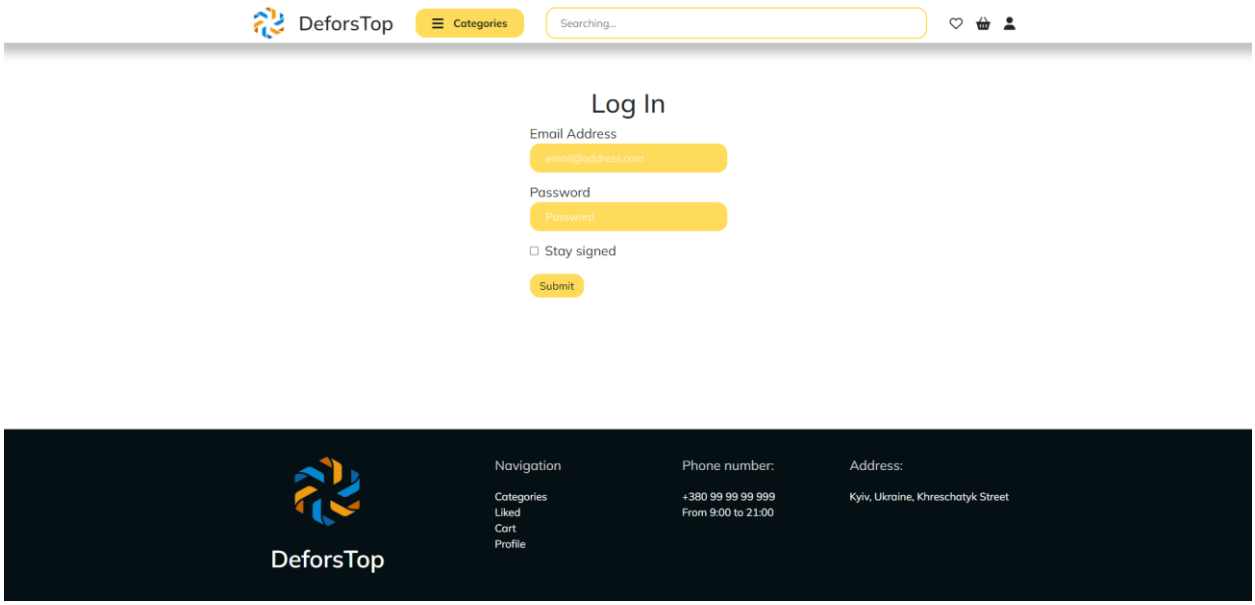

Рисунок 3.9 – Сторінка списку бажань

Останнім елементом є Error.tsx. Цей компонент відповідає за відображення інформації про помилку, якщо вона виникає під час використання веб-застосунку.

3.3 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи було розроблено функціонал веб-застосунку, який був описаний у другому розділі роботи. Розробка була поділена на два ключові етапи: розробка серверної частини та розробка клієнтської частини.

Розробка серверної частини починалася з ініціалізації проекту у середовищі розробки VSCode. Було встановлено ряд залежностей для розробки, включаючи такі пакети, як express, mongoose та інші. Після цього було

розроблено серверний файл, який є точкою входу в наш веб-застосунок. Цей файл включає в себе з’єднання з базою даних. Було створено базу даних з колекціями products, carts, wishlists та authusers. Для отримання доступу до даних зі сторони клієнта було створено REST API, яка включає в себе ендпоінти, такі як: categories, products, cart, wishlist та auth.

Розробка клієнтської частини починалася з ініціалізації проекту на основі інструменту Vite та використанням фреймворку React. Було встановлено ряд залежностей, таких як axios, swr, sass та інші. Наступним кроком стало створення сторінок веб-застосунку, які контролюються роутером, створеним за допомогою бібліотеки react-router-dom. Було представлено такі сторінки: Main, Category, Product, Cart, Wishlist, Login та Error.

В результаті було успішно розроблено функціонал веб-застосунку. Розробка була ефективно поділена на два етапи, що дозволило зосередитися на кожному аспекті проекту окремо. Результатом став повноцінний веб-застосунок, який включає в себе серверну та клієнтську частини, що взаємодіють між собою за допомогою REST API. Кожна сторінка веб-застосунку була ретельно розроблена та протестована, що забезпечило стабільність та надійність роботи веб-застосунку. Завдяки цьому веб-застосунок готовий до впровадження та використання.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ураженні електричним струмом

Долікарська допомога – це сукупність медичних заходів, спрямованих на надання допомоги при невідкладних станах, які виникають на роботі, у повсякденному житті, під час дорожньо-транспортних пригод, катастроф, техногенних аварій, а також при гострих неврологічних, терапевтичних, хірургічних та критичних станах. Відсутність долікарської допомоги у разі нещасних випадків або раптових гострих захворювань може призвести до тяжких наслідків, включаючи летальні. Своєчасне надання допомоги відіграє важливу роль у подальшому лікуванні постраждалих і хворих, сприяючи скороченню строків їх медичної та трудової реабілітації [33].

Менеджери та адміністратори онлайн-магазинів постійно працюють за персональними комп'ютерами або ноутбуками. Їхня діяльність включає взаємодію з різними електронними приладами та обладнанням, що необхідне для роботи інтернет-магазину. Незважаючи на сучасні засоби захисту, існує ймовірність ураження електричним струмом через несправність обладнання, неправильне використання приладів або технічні збої. Тому важливо знати, як надати долікарську допомогу при ураженні електричним струмом, щоб забезпечити безпеку працівників та мінімізувати наслідки таких випадків. Усвідомлення ризиків та вміння правильно реагувати на них є ключовими аспектами підтримання здоров'я та безпеки на робочому місці.

Ураження електричним струмом виникає при контакті людини з електричним ланцюгом, що викликає протікання струму через тіло. Електричний струм може бути різної сили та напруги, і навіть короткочасний контакт з високовольтним струмом може призвести до серйозних наслідків або смерті. Загалом електротравма становить 1-2 % усіх видів травм. Близько 80 % ураження електричним струмом виникає при контакті зі струмом низької напруги (до 1000 В), до 20 % - струмом високої напруги. У першому випадку летальність

становить 3 %, при контакті зі струмом високої напруги – до 30 %. Побутовий струм у 50 Гц особливо небезпечний для нормального функціонування міокарду [34].

Електротравми можна розділити на легкі, середні та тяжкі форми. При легких ураженнях постраждалий може втратити свідомість. У випадку середніх уражень спостерігаються загальні м'язові судоми, непритомність, розлади дихання та серцевої діяльності. Тяжкі ураження призводять до такого пригнічення дихання та серцевої діяльності, що звичайні методи реанімації не є ефективними, і постраждалий перебуває в стані клінічної смерті.

Ознаки ураження електричним струмом включають втрату свідомості у 70% випадків, яка може тривати від кількох хвилин до години, а іноді й більше доби. Ураження головного мозку проявляються клонічними та тонічними судомами, головним болем, сонливістю та ретроградною амнезією. Рідше виникають локальні ураження головного та спинного мозку. Ці симптоми часто з'являються, коли струм проходить через голову або верхні кінцівки. Якщо струм впливає на головний мозок або серце, наслідки можуть бути смертельними – виникає клінічна смерть, де у 80% випадків спостерігається фібриляція шлуночків, а у 20% – асистолія [35].

Порядок надання першої медичної допомоги при ураженні електричним струмом:

– Перш за все, необхідно забезпечити безпеку як рятувальника, так і потерпілого. Вимкнути джерело живлення або використати ізольовані інструменти, щоб усунути контакт постраждалого з електричним струмом.

– Після того, як потерпілий опиниться в безпечному середовищі, слід провести первинну оцінку його стану. Перевірити свідомість, дихання та наявність пульсу. Якщо потерпілий не реагує, не дихає або у нього відсутній пульс, слід негайно розпочати серцево-легеневу реанімацію (СЛР).

– При проведенні СЛР, необхідно переконатися, що дихальні шляхи відкриті: для цього нахилити голову потерпілого назад та підняти підборіддя. Виконувати штучне дихання та компресії грудної клітки.

– Під час проведення СЛР обов'язково викликати екстрену медичну допомогу або попросити когось із присутніх зробити це. Прибуття кваліфікованих медиків є критичним для подальшого обстеження та лікування постраждалого.

– Очікуючи на прибуття екстреної медичної допомоги, важливо постійно контролювати стан потерпілого та надавати йому емоційну підтримку. Якщо потерпілий починає приходити до тями, необхідно допомогти йому зайняти зручне положення та підтримувати його морально.

Надання долікарської допомоги при ураженні електричним струмом є важливим елементом забезпечення безпеки на робочому місці, особливо в умовах роботи з електронними приладами. Володіння навичками першої допомоги може суттєво знизити ризик серйозних ускладнень та врятувати життя.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

Проектування та розробка інтернет-магазину вимагає постійної роботи за персональним комп’ютером, тому важливо правильно організувати робоче місце оператора ПК. Ергономічно оптимізоване робоче місце сприяє здоров’ю, комфорту та продуктивності тих, хто проводить багато часу за комп’ютером.

Робочі місця з комп’ютерами повинні відповідати вимогам ДСТУ 8604:2015 [36] та ДСанПІН 3.3.2.007-98 [37].

Згідно вимог ДСТУ 8604:2015 конструкція робочого місця повинна забезпечувати такі умови праці:

– Конструкція робочого місця та розташування всіх його елементів повинні відповідати антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи.

– Конструкція робочого місця має забезпечувати виконання трудових операцій у межах досяжності моторного поля.

– Висота сидіння та підставки для ніг повинні бути встановлені за номограмою для людей зі зростом 1800 мм. Для менших працівників висоту

сидіння та підставки для ніг потрібно збільшувати на різницю між висотою робочої поверхні для зросту 1800 мм і висотою, оптимальною для зросту конкретного працівника.

– Мінімальна товщина стільниці залежить від міцності матеріалу та технічних вимог і не повинна перевищувати 30 мм.

– Підставка для ніг має бути регульованою по висоті, шириною не менше 300 мм і довжиною не менше 400 мм, поверхня має бути рифленою. По передньому краю слід передбачити бортик висотою 10 мм.

Організація робочого місця повинна дотримуватись таких вимог ДСанПІН 3.3.2.007-98:

– Площа на одне робоче місце повинна бути не менше 6,0 м 2 , а об'єм - не менше 20,0 м3 .

– Приміщення повинні мати природне та штучне освітлення.

– Природне освітлення має здійснюватись через вікна, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природної освітленості (КПО) не нижче 1,5%.

– Для внутрішнього оздоблення приміщень слід використовувати дифузновідбивні матеріали з коефіцієнтами відбиття для стелі 0,7-0,8, для стін 0,5-0,6.

– Яскравість світильників загального освітлення в зоні кутів випромінювання від 50 до 90 градусів з вертикаллю має бути не більше 200 кд/м2 , а захисний кут світильників - не менше 40 градусів.

– Показник засліплення при використанні джерел загального штучного освітлення не повинен перевищувати 20.

– Необхідно обмежувати нерівномірність розподілу яскравості в полі зору працюючих з екранами. Співвідношення яскравості робочих поверхонь не повинно перевищувати 3:1, а співвідношення яскравості робочих поверхонь і поверхонь стін, обладнання – 5:1.

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи детально розглянуто питання безпеки життєдіяльності та основ охорони праці в контексті ураження електричним струмом та вимог ергономіки до організації робочого місця оператора ПК. Аналізуючи відповідність вимогам стандартів безпеки, виділено ключові аспекти, які включають надання долікарської допомоги при ураженні електричним струмом та вимоги ергономіки до організації робочого місця оператора ПК.

В контексті ураження електричним струмом було проаналізовано його можливі наслідки, зокрема різні форми ураження, ознаки та порядок надання першої медичної допомоги. Зазначено, що своєчасне та правильне реагування на такі ситуації є важливим для забезпечення безпеки працівників та мінімізації можливих наслідків.

Висвітлено важливість оптимізації робочого місця операторів ПК з точки зору ергономіки. Розглянуто ключові вимоги до організації робочих місць відповідно до стандартів ДСТУ 8604:2015 та ДСанПІН 3.3.2.007-98. Зокрема дані вимоги стосуються конструкції робочого місця, що включають в собі ергономічні вимоги до крісла, підставки для ніг, стільниці, освітленості та площі робочого простору. Вказані вимоги дозволять підвищити продуктивність праці працівників та забезпечити комфортні умови роботи.

Передбаченні в роботі ергономічні вимоги сприятимуть зниженню ризику травм та захворювань, пов'язаних з професійною діяльністю.

ВИСНОВКИ

У першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» проведено детальний аналіз предметної області ринку електронної комерції, що дозволило сформулювати ключові вимоги до створення веб-застосунку для онлайн продажу товарів електроніки. Виконано аналіз існуючих рішень конкурентів, виявлено їх переваги та недоліки, що стало основою для визначення функціональних та нефункціональних вимог до розроблюваного інтернет-магазину. Обґрунтовано вибір технологічного стеку MERN як найбільш придатного для ефективної реалізації поставлених завдань.

У другому розділі кваліфікаційної роботи досліджено можливі варіанти архітектури веб-застосунку та обґрунтовано вибір трирівневої архітектури системи. Визначено структуру клієнтської та серверної частин застосунку, спроектовано архітектуру бази даних, яка забезпечує ефективну взаємодію з інформаційними ресурсами системи. Розроблено моделі даних і REST API, які дозволяють реалізувати гнучку та масштабовану систему.

У третьому розділі кваліфікаційної роботи реалізовано серверну частину веб-застосунку на основі платформи Node.js з використанням фреймворку Express.js, що забезпечило стабільність та високу продуктивність сервера. Розроблено клієнтську частину застосунку з використанням React.js та сучасних бібліотек для управління станом і обробки запитів, що дозволило створити інтуїтивно зрозумілий та зручний інтерфейс для користувача. Реалізовано механізми авторизації, захисту даних та інтеграцію з базою даних MongoDB.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено важливі аспекти забезпечення безпеки під час експлуатації інтернет-магазину, включаючи процедури долікарської допомоги при ураженні електричного струму та вимог ергономіки до організації робочого місця оператора ПК.

Таким чином, досліджено, розроблено та реалізовано сучасний веб-застосунок для онлайн продажу товарів електроніки, що відповідає актуальним вимогам ефективності, зручності використання, масштабованості та безпеки.

ПЕРЕЛІК ДЖЕРЕЛ

1. Чому створення інтернет-магазину це один з кращих методів розширення бізнесу. [Електронний ресурс]. Режим доступу до ресурсу: https://lb.ua/tech/2023/10/23/579979_chomu_stvorennya_internetmagazinu_tse.html (дата звернення: 16.02.2025)

2. Які сторінки мають бути на сайті інтернет-магазину ? [Електронний ресурс]. Режим доступу до ресурсу: https://www.fishdigital.agency/blog-yaki-storinki-mayut-buti-na-sayti-internet-magazinu (дата звернення: 17.02.2025)

3. Інтернет-магазин D.UA. [Електронний ресурс]. Режим доступу до ресурсу: https://d.ua/ (дата звернення: 17.02.2025)

4. Інтернет-магазин Comfy. [Електронний ресурс]. Режим доступу до ресурсу: https://comfy.ua/ (дата звернення: 18.02.2025)

5. Інтернет-магазин Viatec. [Електронний ресурс]. Режим доступу до ресурсу: https://viatec.ua/ (дата звернення: 18.02.2025)

6. Текстовий редактор Atom. [Електронний ресурс]. Режим доступу до ресурсу: https://atom-editor.cc/ (дата звернення: 19.02.2025)

7. Середовище розробки WebStorm. [Електронний ресурс]. Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/ (дата звернення: 20.02.2025)

8. Середовище розробки Visual Studio Code. [Електронний ресурс]. Режим доступу до ресурсу: https://code.visualstudio.com/docs (дата звернення: 20.02.2025)

9. Найкращі IDE та текстові редактори. [Електронний ресурс]. Режим доступу до ресурсу: https://javarush.com/ua/groups/posts/uk.177.naykrajsh-ide-ta-tekstov-redaktori-dlja-frontendnika (дата звернення: 21.02.2025)

10. MERN Stack Explained. [Електронний ресурс]. Режим доступу до ресурсу: https://www.mongodb.com/resources/languages/mern-stack (дата звернення: 22.02.2025)

11. Introduction to Node.js. [Електронний ресурс]. Режим доступу до ресурсу: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (дата звернення: 22.02.2025)

12. What is MongoDB? [Електронний ресурс]. Режим доступу до ресурсу: https://www.mongodb.com/company/what-is-mongodb (дата звернення: 23.02.2025)

13. What is JavaScript. [Електронний ресурс]. Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (дата звернення: 23.02.2025)

14. DOM. [Електронний ресурс]. Режим доступу до ресурсу: https://uk.legacy.reactjs.org/docs/react-dom.html (дата звернення: 24.02.2025)

15. Які основні мови програмування використовуються у Front-end розробці? [Електронний ресурс]. Режим доступу до ресурсу: https://optima.study/blog/yaki-osnovni-movy-prohramuvannya-vykorystovuyut-sya-u-front-end-rozrobtsi (дата звернення: 25.02.2025)

16. Client Server Architecture – Detailed Explanation. [Електронний ресурс]. Режим доступу до ресурсу: https://www.interviewbit.com/blog/client-server-architecture/ (дата звернення: 26.02.2025)

17. SWR: React Hooks for Data Fetching. [Електронний ресурс]. Режим доступу до ресурсу: https://swr.vercel.app/ (дата звернення: 29.02.2025)

18. REST API як спосіб спілкування компонент веб-додатків. [Електронний ресурс]. Режим доступу до ресурсу: https://foxminded.ua/shcho-take-rest-api/ (дата звернення: 01.03.2025)

19. Schema Design for MongoDB. [Електронний ресурс]. Режим доступу до ресурсу: https://blog.det.life/the-inheritance-schema-design-pattern-for-mongodb-data-modelling-03540a484a93 (дата звернення: 05.03.2025)

20. Introduction to Yarn. [Електронний ресурс]. Режим доступу до ресурсу: https://yarnpkg.com/getting-started (дата звернення: 09.04.2025)

21. TypeScript – офіційний сайт. [Електронний ресурс]. Режим доступу до ресурсу: https://www.typescriptlang.org/ (дата звернення: 10.04.2025)

22. How works with Mongoose. [Електронний ресурс]. Режим доступу до ресурсу: https://docs.nestjs.com/recipes/mongodb (дата звернення: 11.04.2025)

23. Intro to Express.js: Endpoints, parameters, and routes. [Електронний ресурс]. Режим доступу до ресурсу: https://www.infoworld.com/article/3615615/intro-to-express-js-endpoints-parameters-and-routes.html (дата звернення: 12.04.2025)

24. JSON Web Token (JWT). [Електронний ресурс]. Режим доступу до ресурсу: https://jwt.io/ (дата звернення: 13.04.2025)

25. Хешування паролів: використання солі та bcrypt. [Електронний ресурс]. Режим доступу до ресурсу: https://drukarnia.com.ua/articles/kheshuvannya-paroliv-vikoristannya-soli-ta-bcrypt-fsme- (дата звернення: 13.04.2025)

26. Middleware in Express. [Електронний ресурс]. Режим доступу до ресурсу: https://www.geeksforgeeks.org/middleware-in-express-js/ (дата звернення: 16.04.2025)

27. Vite – офіційний сайт. [Електронний ресурс]. Режим доступу до ресурсу: https://vite.dev/ (дата звернення: 17.04.2025)

28. Роутинг у React. [Електронний ресурс]. Режим доступу до ресурсу: https://blog.ithillel.ua/articles/routing-in-react (дата звернення: 19.04.2025)

29. Fryz M., Scherbak L., Mlynko B., Mykhailovych T. Linear Random Process Model-Based EEG Classification Using Machine Learning Techniques. Proceedings of the 1st International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2023). Ternopil, Ukraine: CEUR Workshop Proceedings, 2023. Vol. 3468. P. 126–132.

30. Бабак В.П., Куц Ю.В., Мислович М.В., Фриз М.Є., Щербак Л.М. Об’єктно-орієнтована ідентифікація стохастичних шумових сигналів. Київ: Наукова думка, 2024. 240 с. https://doi.org/10.15407/978-966-00-1883-9

31. V. Babak, A. Zaporozhets, Y. Kuts, M. Fryz, L. Scherbak. Noise signals: Modelling and Analyses. Cham: Springer Nature Switzerland, 2025. 222 p. DOI: https://doi.org/10.1007/978-3-031-71093-

32. Бабак В. П., Марченко М. Є., Фриз. Б. Г. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с.

33. Перша долікарська допомога. [Електронний ресурс]. Режим доступу до ресурсу: https://www.pharmencyclopedia.com.ua/article/790/persha-dolikarskadopomoga (дата звернення: 10.06.2025)

34. Діагностика та лікування уражень електричним струмом. [Електронний ресурс]. Режим доступу до ресурсу: https://pidru4niki.com/76919/meditsina/diagnostika_likuvannya_urazhennya_elektrichnim_strumom (дата звернення: 11.06.2025)

35. Жидецький В. Ц. Основи охорони праці. / В. Ц. Жидецький, В. С. Джигирей, О. В. Мельников. 2- ге вид. Львів: Афіша, 2000. 348 с.

36. ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце під час виконання робіт сидячи» [Електронний ресурс]. Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 13.06.2025)

37. ДСанПІН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електроннообчислювальних машин» [Електронний ресурс]. Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення: 14.06.2025)

ДОДАТКИ

Додаток А

Програмний код JS для запуску сервера «index.ts»

```
import bodyParser from "body-parser";
import cookieParser from "cookie-parser";
import cors from "cors";
import dotenv from "dotenv";
import express from "express";
import helmet from "helmet";
import mongoose from "mongoose";
import morgan from "morgan";
import multer from "multer";
import fs from "fs";
import session from "express-session";
import { authTokenVerification, verifyToken } from
"./src/middleware/auth.js";
import { getAuthRouter } from "./src/routes/auth.js";
import { getCartRouter } from "./src/routes/cart.js";
import { getCategoriesRouter } from "./src/routes/categories.js";
import { getProductsRouter } from "./src/routes/products.js";
import { getLogoRouter } from "./src/routes/logo.js";
import { getWishlistRouter } from "./src/routes/wishlist.js";
import { getSearchRouter } from "./src/routes/search.js";

/* CONFIGURATION */
dotenv.config();
// const __filename = fileURLToPath(import.meta.url);
// const __dirname = path.dirname(__filename);

const app = express();
const port = process.env.PORT || 6001;

/* MIDDLEWARE */
app.use(express.json());
app.use(helmet());
app.use(helmet.crossOriginResourcePolicy({ policy: "cross-origin"
}));
app.use(morgan("common"));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
// app.use("/assets", express.static(path.join(__dirname,
"public/assets")));
app.use("/assets", express.static("public/assets"));

/* MONGOOSE SETUP */
const MONGO_URL = process.env.MONGO_URL;
if (!MONGO_URL) {
  console.log("MONGODB_URI is missing, please fill the value!");
  process.exit(1);
}
```

```
mongoose
  .connect(MONGO_URL)
  .then(() => {
    app.listen(port, () => console.log(`Server starts at port
${port}`));
  })
  .catch((err) => {
    console.log(`${err}. MongoDB did not connected`);
  });

/* ROUTES */
app.use(authTokenVerification);

app.post("/test", upload.array("avatar"), (req, res) => {
  // console.log(req.files);

  const thumbnailPath: string[] = [];
  const images = req.files as Express.Multer.File[];

  if (images) {
    images.forEach((i) => {
      const path = i.path.replace(`public\\assets\\`, "");
      thumbnailPath.push(path);
    });
    // else push path to placeholder image
  }
  console.log("🚀 ~ app.post ~ thumbnailPath:", thumbnailPath);
  res.json("Upload");
});

app.use("/auth", getAuthRouter());
app.use("/products", getProductsRouter());
app.use("/categories", getCategoriesRouter());
app.use("/cart", getCartRouter());
app.use("/wishlist", getWishlistRouter());
app.use("/logo", getLogoRouter());
app.use("/search", getSearchRouter());
```

Додаток Б

Програмний код JS для побудови сторінок та навігації «router.tsx»

```
import { createBrowserRouter, Navigate } from "react-router-dom";
import Cart from "./Cart";
import Category from "./Category";
import Error from "./Error";
import Login from "./Login";
import Main from "./Main";
import Product from "./Product";
import Wishlist from "./Wishlist";
import Profile from "./Profile";
import Admin from "./Admin";
import AdminDashboard from
"../components/AdminPage/AdminDashboard/AdminDashboard";
import AdminProducts from
"../components/AdminPage/AdminProducts/AdminProducts";
import AdminAddProduct from
"../components/AdminPage/AdminProducts/AdminAddProduct/AdminAddPro
duct";
import AdminProductsList from
"../components/AdminPage/AdminProducts/AdminProductsList/AdminProd
uctsList";
import AdminProductsCategories from
"../components/AdminPage/AdminProducts/AdminProductsCategories/Adm
inProductsCategories";
import AdminEditProduct from
"../components/AdminPage/AdminProducts/AdminEditProduct/AdminEditP
roduct";
import ParallaxPage from "./ParallaxPage";

export const router = createBrowserRouter([
  {
    path: "/",
    element: <Main />,
    errorElement: <Error />,
  },
  {
    path: "/category/",
    element: <Category />,
    children: [
      {
        path: ":category",
        element: <Category />,
      },
    ],
    errorElement: <Error />,
  },
  {
```

```
    path: "/product/:productId",
    element: <Product />,
    errorElement: <Error />,
  },
  {
    path: "/cart",
    element: <Cart />,
    errorElement: <Error />,
  },
  {
    path: "/wishlist",
    element: <Wishlist />,
    errorElement: <Error />,
  },
  {
    path: "/login",
    element: <Login />,
    errorElement: <Error />,
  },
  {
    path: "/profile",
    element: <Profile />,
    errorElement: <Error />,
  },

  {
    path: "/error",
    element: <Error />,
    errorElement: <Error />,
  },
]);
```