МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ СПОРТИВНИХ ТОВАРІВ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ REACT»**

Виконав: студент 5 курсу, групи 6.1269-з
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)
освітньої програми Інформаційні системи та технології
(назва освітньої програми)
Є.А. Мась
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, PhD, Чопорова О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н., Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 126 Інформаційні системи та технології
(шифр і назва)

Освітня програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Мась Євгену Андрійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет-магазину спортивних товарів з використанням фреймворку React

керівник роботи Чопорова Оксана Володимирівна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2181-с

2. Строк подання студентом роботи 17.05.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка інтернет-магазину спортивних товарів з використанням фреймворку React.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

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

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.01.2024	
2.	Збір вихідних даних.	30.01.2024	
3.	Обробка методичних та теоретичних	20.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	25.03.2024	
5.	Розробка третього розділу.	03.05.2024	
6.	Оформлення та нормоконтроль	10.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	31.05.2024	

Студент _____ Є.А. Мась
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Чопорова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет-магазину спортивних товарів з використанням фреймворку React»: 55 с., 42 рис., 2 табл., 17 джерел.

ВЕБЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, МАРШРУТИЗАЦІЯ, CSS, FRONT-END, HTML, REACT.JS, VISUAL STUDIO CODE.

Об’єкт дослідження – інтернет-магазин.

Мета роботи: розробити інтернет-магазин спортивних товарів з використанням фреймворку React.

Метод дослідження – порівняльний аналіз, розробка клієнтських компонентів за допомогою React, обробка літературних джерел.

В результаті виконання дипломної роботи було створено інтернет-магазин спортивних товарів з використанням сучасного фреймворку React, що забезпечує динамічну та ефективну розробку вебзастосунків. Реалізація магазину базується на використанні мов програмування HTML, CSS для структури та стилізації вебсторінок, відповідно.

Цей проєкт є прикладом використання сучасних технологій у веброзробці для створення функціонального та привабливого інтернет-магазину, що відповідає сучасним вимогам інтернет-торгівлі.

SUMMARY

Bachelor’s qualifying paper «Development of an Online Store of Sporting Goods Using the React Framework»: 55 pages, 42 figures, 2 tables, 17 references.

WEB APPLICATION, ROUTING, ONLINE STORE, CSS, FRONT-END, HTML, REACT.JS, VISUAL STUDIO CODE.

The object of the study is the online store.

The aim of the study is to develop an online store of sporting goods using the React framework.

The method of research are comparative analysis, development of client components using React, processing of literary sources.

As a result of the work: an online sporting goods store was created using the modern React framework, which provides dynamic and efficient development of web applications. The implementation of the store is based on the use of HTML and CSS programming languages for the structure and styling of web pages, respectively.

This project is an example of the use of modern technologies in web development to create a functional and attractive online store that meets the modern requirements of online commerce.

ЗМІСТ

ВСТУП

Сучасний світ переживає період надзвичайного розвитку технологій, який надзвичайними темпами змінив наше сприйняття та взаємодію з навколишнім середовищем. Від поширення інтернету та впровадження широкосмугових підключень до загальної цифровізації різних сфер життя, технології стають не лише необхідністю, але й двигуном прогресу.

Розробка сайтів та інтернет-магазинів є однією з головних сфер застосування сучасних технологій. Ці платформи не лише надають компаніям зручний інструмент для презентації своїх товарів та послуг, але й відкривають нові можливості для електронної комерції, забезпечуючи зручний та ефективний спосіб продажу для підприємств будь-якого масштабу.

Актуальність теми «Розробка інтернет-магазину спортивних товарів з використанням фреймворку React» визначається зростанням популярності спортивних товарів та послуг у сучасному суспільстві, а також швидким розвитком вебтехнологій. Створення вебсайту спортивних товарів на базі React відповідає потребам сучасного ринку та дозволить ефективно взаємодіяти з клієнтами, забезпечуючи зручний та привабливий користувацький досвід.

Предметом дослідження є процес розробки інтернет-магазину з використанням фреймворку React.

Метою кваліфікаційної роботи є розробка ефективного та функціонального вебдодатку, що задовольняє потреби клієнтів у придбанні товарів, а також вивчення та аналіз впливу візуального оформлення сайту на користувачів.

Об’єктом дослідження є сам інтернет-магазин спортивного екіпірування, який буде розроблено з використанням React. Завданнями дослідження визначено вивчення фреймворка React, аналіз вимог до проєкту, розробку архітектури та інтерфейсу магазину, а також тестування та оптимізацію продукту.

Наукова новизна роботи полягає в застосуванні фреймворку React для створення вебдодатку та вивченні впливу UX/UI дизайну на користувачів. Практичне значення отриманих результатів полягає в можливості їх використання розробниками та бізнес-власниками для підвищення ефективності та конкурентоспроможності вебмагазинів у сфері спортивних товарів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Методи розробки вебзастосунків

Вебзастосунок або вебдодаток – розподілений застосунок, в якому клієнтом виступає браузер, а сервером – вебсервер. Браузер може бути реалізацією так званих тонких клієнтів – логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у зображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому вебзастосунки є міжплатформовими сервісами [1].

Існує декілька методів розробки вебзастосунків, кожен з яких має свої певні переваги, недоліки або особливості. Ось декілька найпопулярніших із них.

Клієнт-серверний підхід: полягає у розподілені розробки вебзастосунку між клієнтською частиною (Front-end) та серверною (Back-end) (рис. 1.1).

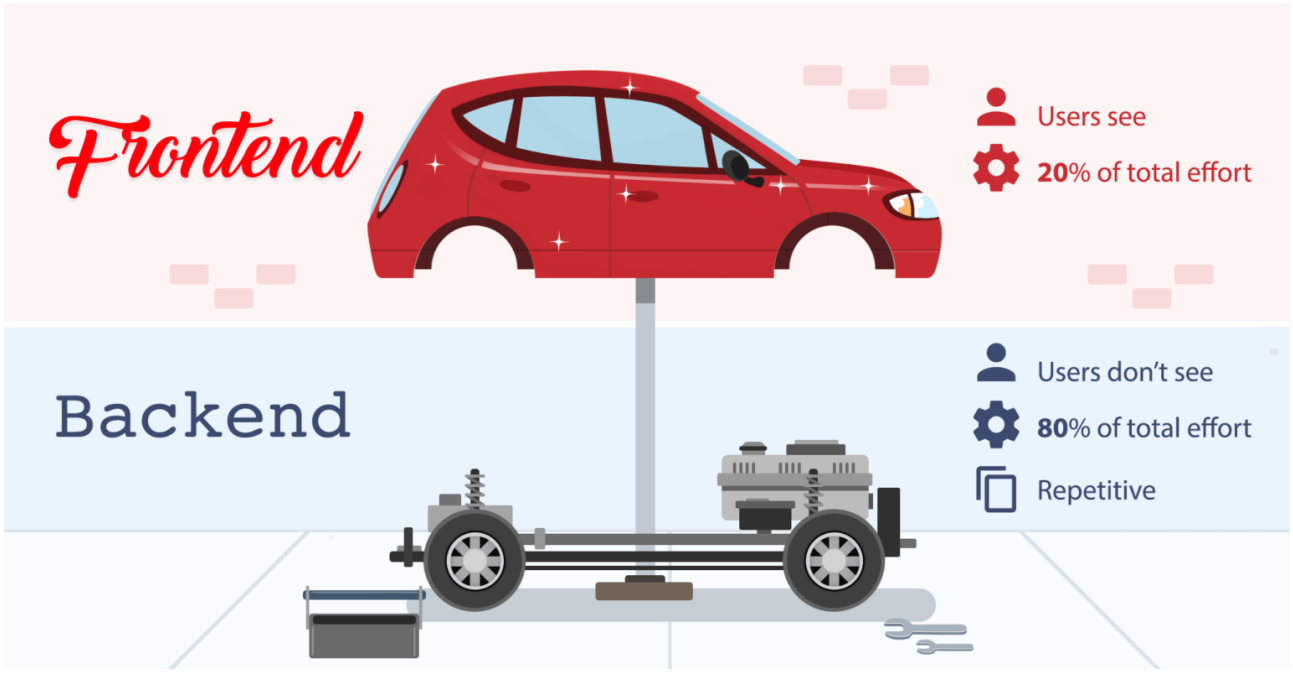

Рисунок 1.1 – Візуалізація клієнт-серверного підходу

Front-end, також відомий як клієнтська частина вебдодатку, – це та частина програмного забезпечення, яка подібна до кузова автомобіля. Подібно до кузова, який визначає зовнішній вигляд та структуру автомобіля, Front-end визначає те, як користувачі бачать та взаємодіють з вебсайтами або вебдодатками у своїх браузерах. Це все те, що ви бачите на екрані, коли відвідуєте вебсайт: текст, зображення, кнопки, форми, анімація та інші елементи інтерфейсу. Front-end включає в себе три основні технології: html, css, javascript.

HTML (HyperText Markup Language): він відповідає за структуру вебсторінок. HTML використовується для створення різних елементів на сторінці, таких як заголовки, абзаци, списки, посилання тощо.

CSS (Cascading Style Sheets): він відповідає за вигляд і оформлення вебсторінок. Він визначає кольори, шрифти, розміри, відступи, рамки та інші стилізаційні властивості для кожного елемента на сторінці.

JavaScript відповідає за інтерактивність вебсторінок. Він дозволяє реалізувати різноманітні функції, такі як анімація, перевірка форм, динамічне оновлення вмісту сторінки, взаємодія з користувачем та багато іншого.

Back-end, або серверна частина вебдодатку, – це та складова програмного забезпечення, яка подібна до технічної складової автомобіля. Подібно до того, як різноманітні системи і компоненти автомобіля працюють разом, щоб забезпечити його рух та функціонування, back-end відповідає за обробку запитів користувачів, доступ до бази даних, бізнес-логіку та інші логічні операції, які забезпечують роботу вебзастосунку.

Серверні мови програмування: це мови програмування, які використовуються для написання логіки back-end. Найпоширеніші мови включають JavaScript (з Node.js), Python, Ruby, PHP, Java та C#. Ці мови дозволяють розробникам створювати функціональність, яка відбувається на сервері, таку як обробка запитів, автентифікація користувачів та інші операції.

Бази даних: back-end розробники використовують бази даних для зберігання та управління даними, які використовуються в вебдодатку. Популярні

системи управління базами даних (СУБД) включають MySQL, PostgreSQL, MongoDB, SQLite та інші.

Фреймворки та бібліотеки: це інструменти, які спрощують розробку back-end додатків, надаючи готові рішення для таких завдань, як маршрутизація запитів, автентифікація, кешування даних та інші. Найпоширеніші фреймворки включають Express.js для JavaScript, Django для Python, Ruby on Rails для Ruby, Laravel для PHP та Spring для Java.

Односторінковий (Single Page Application або SPA) полягає у розробці вебзастосунку, де усі необхідні дані та функціональність завантажуються один раз при відкритті сторінки. Надалі, при взаємодії користувача з додатком, браузер динамічно оновлює вміст сторінки без повного перезавантаження. У звичайних вебсайтів, коли користувач переходить на нову сторінку, браузер завантажує заново HTML, CSS і JavaScript, що може призвести до затримок та менш зручного користування.

SPA відмінні тим, що вони завантажують лише один раз усі ресурси, після чого взаємодія з додатком відбувається без повторного завантаження сторінки. З цієї точки зору, перевагою є те, що ініціальне завантаження сторінки дає змогу користувачам швидко отримати доступ до основного контенту та функцій додатку. Однак, це також може бути недоліком, особливо при великому обсязі даних чи складності додатку, оскільки ініціалізація може займати багато часу, що впливає на загальний користувацький досвід та швидкість реакції додатку.

Розглянемо основні переваги SPA.

Швидкість: завантаження сторінок відбувається швидше, оскільки браузер завантажує лише необхідний код при першому відкритті сторінки, а не кожного разу при переході на нову сторінку.

Інтерактивність: SPA дозволяють створювати більш інтерактивні додатки, оскільки вони реагують на дії користувача швидше та без перезавантаження сторінок.

Зручність розробки: розробка SPA може бути зручнішою, оскільки

розробники працюють з однією сторінкою та переходять між різними компонентами за допомогою JavaScript, не потрібно підтримувати кілька окремих сторінок.

Проте SPA також мають свої недоліки, зокрема щодо SEO (оптимізації для пошукових систем) та управління пам’яттю. Також, вони можуть бути складнішими для розробки, оскільки вимагають більшу увагу до організації коду та управління станом додатку.

Компонентний метод розробки вебзастосунків – це підхід до створення вебінтерфейсів, в якому вебсторінка розбивається на невеликі незалежні компоненти, які можуть бути розвинуті та відлагоджені окремо. Кожен компонент відповідає за свою частину функціоналу та представлення, і може використовуватися множину разів на сторінці або навіть на різних сторінках сайту.

Розглянемо основні переваги методу компонентної розробки.

Підвищена перевикористовуваність коду: компоненти можна легко використовувати на різних сторінках та навіть у різних проєктах.

Зменшення повторюваного коду: ізольовані компоненти дозволяють уникнути дублювання коду та спрощують управління змінами.

Зручне управління структурою та оновленнями: кожен компонент може бути розроблений та тестований окремо, що полегшує управління розробкою та внесення змін.

Покращена спільна робота: розробники можуть працювати над різними компонентами паралельно, що підвищує продуктивність.

Масштабованість: за допомогою компонентів легше масштабувати проєкт із зростанням його складності.

Цей метод розробки вебзастосунків набув популярності у другій половині 2010-х років завдяки появі і поширенню фреймворків та бібліотек, таких як React, Vue.js та Angular, які дозволяють створювати та управляти компонентами ефективно.

1.2 Середовища розробки вебзастосунків

Редактори коду є важливими інструментами для будь-якого веброзробника, надаючи зручне середовище для написання, редагування та відлагодження коду. Нижче наведено декілька прикладів популярних редакторів коду, які використовуються для розробки вебзастосунків.

Visual Studio Code – найпопулярніший на сьогодні безкоштовний редактор коду від Microsoft. Він широко використовується завдяки своїм розширеним можливостям, великій кількості плагінів та підтримці різних мов програмування.

Sublime Text 3 – ще один популярний редактор коду, відомий своєю швидкістю та легкістю використання. Його широкий вибір плагінів та інтуїтивний інтерфейс роблять його популярним серед розробників усіх рівнів.

Atom – редактор із відкритим кодом від GitHub. Цей редактор відомий своєю високою ступеню налаштування та розширюється за допомогою різноманітних плагінів, що дозволяє кожному користувачеві налаштувати його під свої потреби.

WebStorm – Інтегроване середовище розробки від компанії JetBrains, спеціалізоване на веброзробці. WebStorm надає розширені можливості для роботи з JavaScript, HTML та CSS, включаючи автодоповнення та підтримку системи контролю версій.

Для зручного порівняння редакторів коду та їх основних характеристик, всі дані зведені у таблицю (табл. 1.1).

Кожен редактор коду має свої переваги та недоліки, і вибір між ними залежить від індивідуальних потреб та вподобань розробника. Наприклад, деякі редактори можуть мати велику кількість розширень та плагінів, що дозволяє розширити їх функціональність, тоді як інші можуть бути більш легкими та швидкими у використанні. При виборі важливо враховувати також мови програмування та фреймворки, які ви використовуєте у своєму проєкті, оскільки деякі редактори можуть підтримувати їх краще, ніж інші.

Таблиця 1.1 – Характеристики редакторів коду

Критерії	VS Code	Sublime Text	Atom	WebStorm
Виробник	Microsoft	Незалежний	GitHub	JetBrains
Операційна система/ платформа	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS, Linux
Крос-платформенність	Так	Так	Так	Так
Вартість	Безкоштовний	Пробний / Платний	Безкоштовний	Пробний / Платний
Розширення та плагіни	Так	Так	Так	Ні
Підтримка Git	Так	Так	Так	Так
Вбудований термінал	Так	Ні	Ні	Так
Швидкодія	Висока	Висока	Висока	Середня
Гнучкість налаштування	Висока	Висока	Висока	Висока
Дата створення	2015	2008	2014	2010
Автодоповнення	Так	Так	Так	Так

1.3 Структура та компоненти вебсторінки

Вебсторінка, як ключовий елемент інтернет-простору, складається з різноманітних компонентів, що разом створюють зручний та функціональний інтерфейс для користувачів (рис. 1.2). Ці компоненти не лише визначають вигляд

та структуру вебсторінки, але й впливають на її ефективність та взаємодію з відвідувачами.

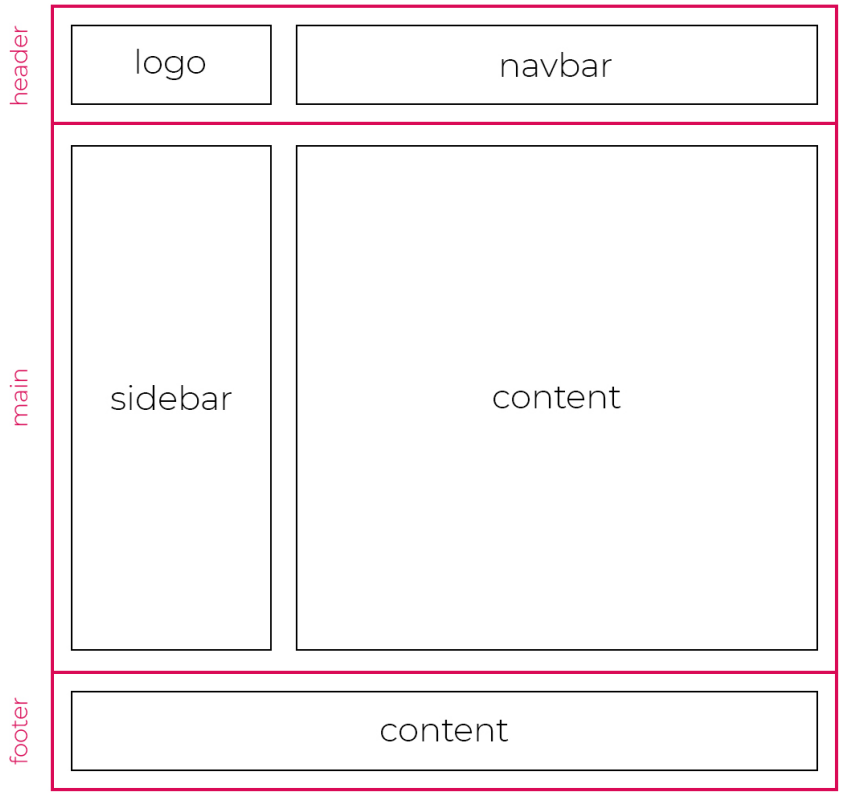

Рисунок 1.2 – Візуалізація структури вебсторінки

Давайте детальніше розглянемо основні компоненти вебсторінки та їх значення.

Шапка (Header): шапка вебсторінки – це перший елемент, який бачить користувач під час відкриття сторінки. Вона зазвичай містить логотип або назву сайту, навігаційне меню, елементи зворотного зв'язку, пошуку або кошику, якшо це інтернет-магазин. Шапка надає користувачам контекст про те, де вони знаходяться, та дозволяє швидко переходити до потрібних розділів сайту. Зазвичай у html-документі позначається тегом <header> (рис. 1.3) [2].

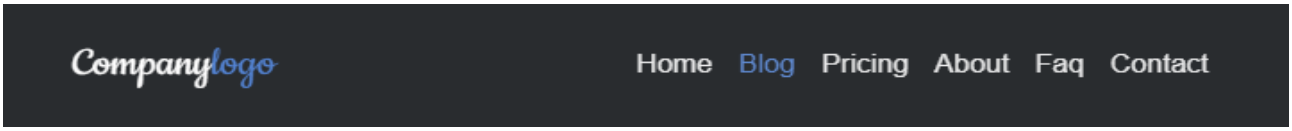

Рисунок 1.3 – Приклад компоненту Header [3]

Навігаційне Меню (Navigation Menu): навігаційне меню – це набір посилань, які допомагають користувачам рухатись по різним сторінках та розділах вебсайту. Воно забезпечує зручний доступ до ключового контенту та допомагає користувачам швидко знаходити потрібну інформацію. Зазвичай позначається тегом <navbar> (рис. 1.4) [2].

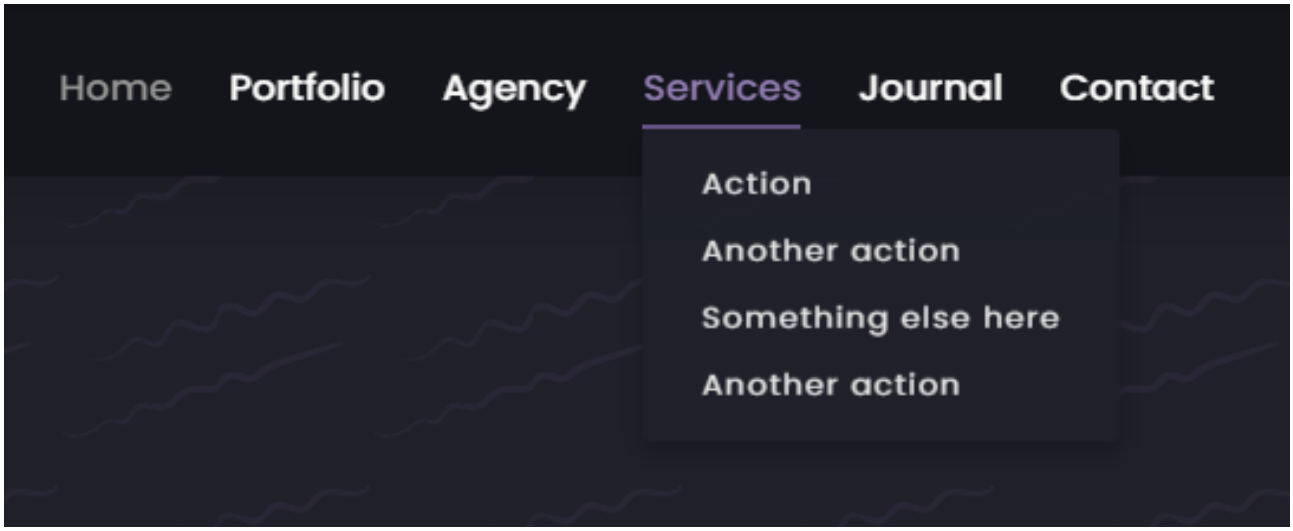

Рисунок 1.4 – Приклад компоненту Navigation Menu [4]

Контентні Блоки (Content Blocks): контентні блоки містять основний інформаційний та візуальний контент вебсторінки, такий як текст, зображення, відео або таблиці. Частіше за все, цей компонент реалізується завдяки тегам <main> та . Вони допомагають організувати інформацію та зробити її доступною для користувачів, що робить навігацію та сприйняття контенту більш зручними (див. рис. 1.5) [2].

Кнопки (Button): компонент вебсайту «Кнопка» є ще одним із найважливіших елементів веброзробки. Він позначається тегом <button> у HTML і використовуються для запуску різних дій, таких як відправлення форм, перехід на іншу сторінку або частину сайту, виклик функцій або обробників подій за допомогою JavaScript і т.д..

Бічна Панель (Sidebar): бічна панель – це додаткова область на сторінці, яка зазвичай містить додаткові посилання, віджети, навігаційні посилання або рекламні блоки. Вона допомагає організувати додаткову інформацію та зробити рух по вебсайту більш гнучкою та зручною для користувачів (див. рис. 1.6) [2].

THE CODEPEN TIMES

ISSUE #1 TUESDAY, 26 MAY, 2020 TWO MEMES EDITION

Woman spends too much time on Codepens despite advice; no plans to stop

Sometimes, Internet users land themselves in an uncommon situation where an app or website can be used for both work and personal situations. Well, a young front-end designer was spending up to 13 hours on the site, Codepen for work and her hobby, which unfortunately, is also coding.

The designer, whose work includes a *To-Do Terrarium*, clarified, "Well, I didn't *have* to use it for work but it was already part of my workflow. It just made sense as it was more efficient." The biggest issue, she said, is that she would hop on Codepen for work, but would get distracted by pens made by others in the community for unexpectedly long periods of time, which then causes her to spend even more time on the site.

"I mean, that's terrible right?" she asked. Codepen, which brands itself as the best place to build, test, and discover front-end code, has declined to comment at this time.

To Do List

Ta-da!

A to-do list that grows succulents when you've completed a task

50% OFF HOGWARTS EXPRESS

A pasta menu

Everybody loves pasta. It's

Hot this month

FOOD

Pies for everyone!

No man is left behind. There's infinite pie with this one div and a repeating background.

Frustrated designer runs amok with Captcha ideas

From Tic Tac Toe to solving meme-based questions, this is a different take on how web captchas should be. "I have not successfully picked out all photos of a truck on the first

Рисунок 1.5 – Приклад компоненту Content [5]

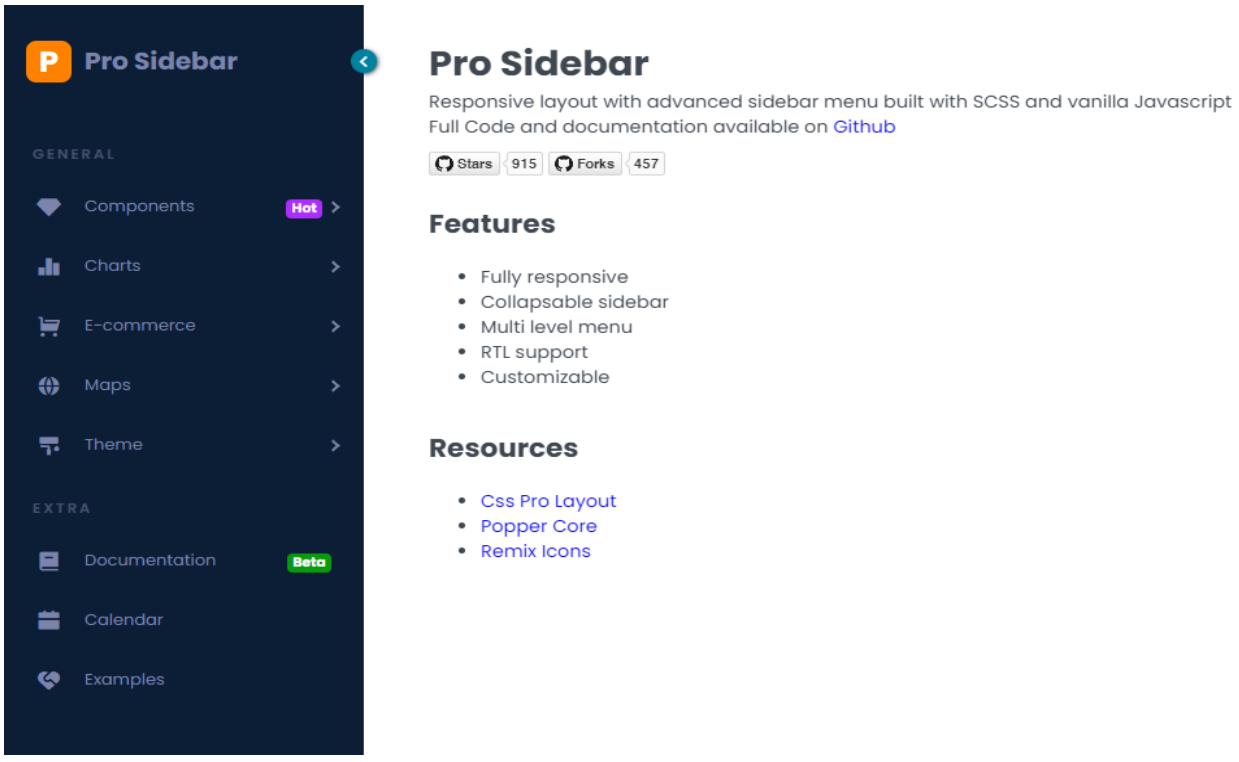

Рисунок 1.6 – Приклад компоненту Sidebar [6]

Підвал (Footer): підвал вебсторінки розташований у нижній частині сторінки і містить додаткові посилання, контактну інформацію, посилання на соціальні мережі, авторські права та інші елементи. Він завершує структуру

вебсторінки та надає додаткову інформацію користувачам. Зазвичай використовується тег <footer> (рис. 1.7) [2].

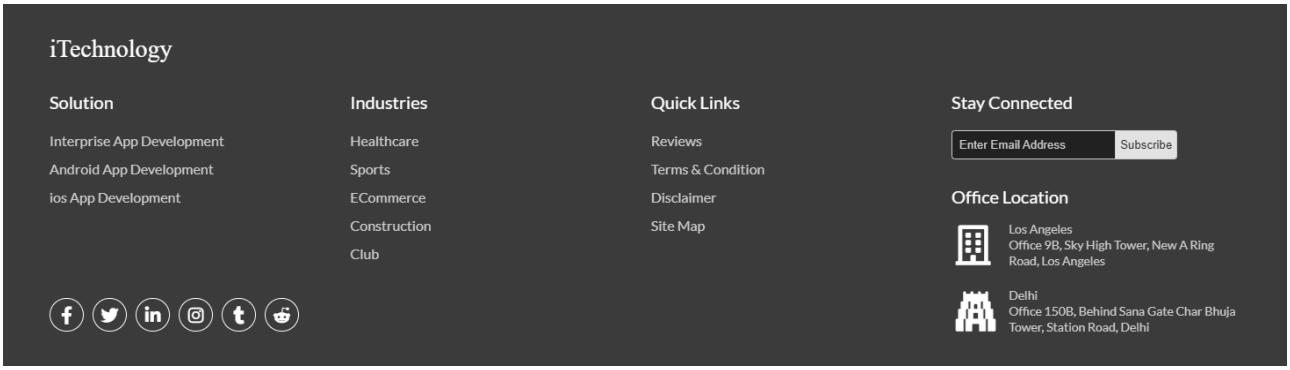

Рисунок 1.7 – Приклад компоненту Footer [7]

Форми (Form): форма – це елемент вебсторінки, який дає користувачам можливість вводити інформацію і відправляти її на сервер для подальшої обробки. За створення форми в html-документі відповідає тег <form>, в якому містяться усі інші необхідні теги, такі як: <form>, <input>, <select>, <optgroup>, <option>, <fieldset>, <legend>, <textarea>, <button>, <label> (рис. 1.8) [2].

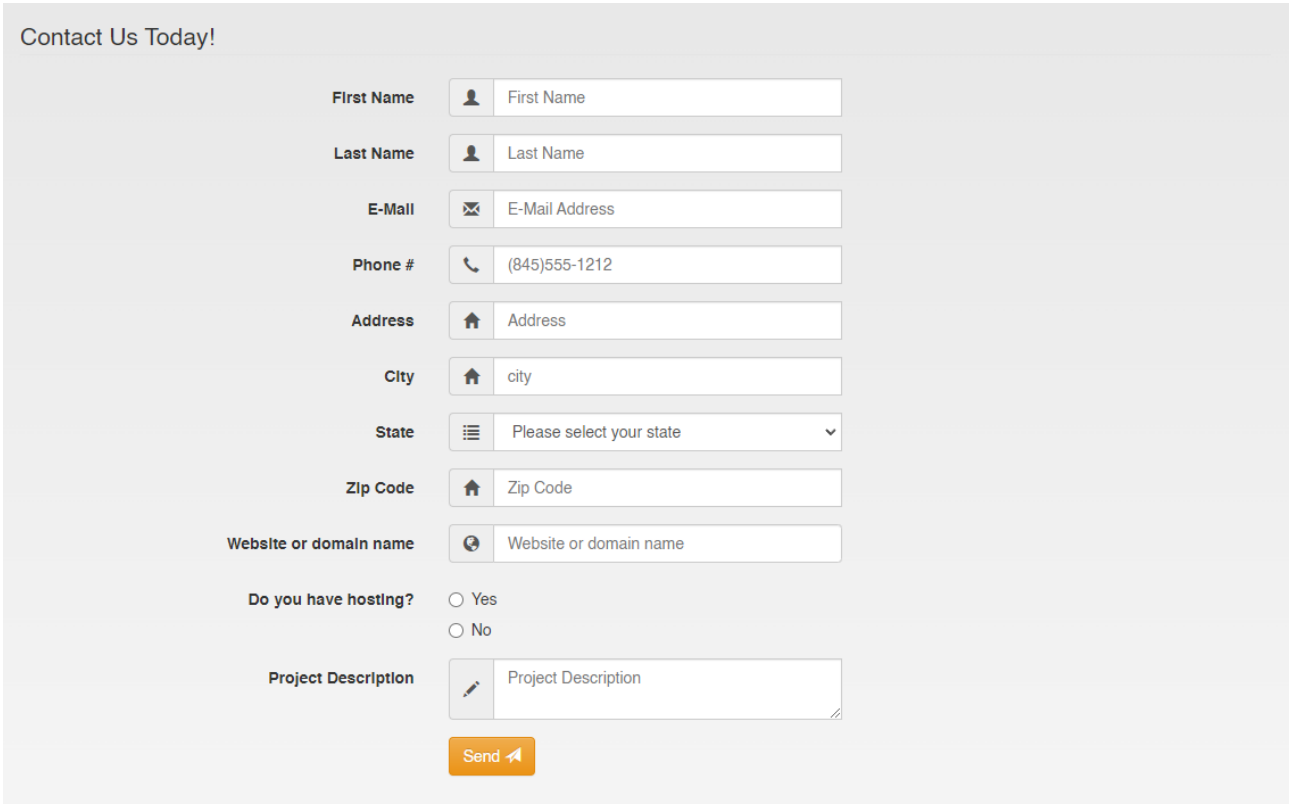

Рисунок 1.8 – Приклад компоненту Form [8]

Кожен з цих компонентів відіграє важливу роль у створенні зручного та привабливого інтерфейсу для користувачів, що робить їх невід'ємною частиною будь-якої успішної вебсторінки.

1.4 Загальне визначення React

React – це бібліотека JavaScript, яка знаходить широке застосування для розробки вебінтерфейсів. Її використання полягає в тому, щоб допомогти розробникам створювати динамічні та інтерактивні сторінки, які реагують на дії користувачів без необхідності повного перезавантаження вебсторінки.

Це досягається за допомогою компонентів, які є основними будівельними блоками у React. Компоненти – це невеликі, самодостатні шматки інтерфейсу, які можна повторно використовувати та комбінувати, щоб створювати складніші структури. Кожен компонент може мати свій внутрішній стан, який визначає, як він повинен виглядати та взаємодіяти з користувачем.

Одна з головних переваг React полягає в тому, що він використовує віртуальний DOM. DOM (Document Object Model) – це структура, що представляє вебсторінку або додаток у вигляді дерева об'єктів. Вона дозволяє JavaScript змінювати, додавати та видаляти елементи на сторінці. Повертаючись до React, це означає, що він зберігає в пам'яті віртуальну копію реального DOM і оновлює його лише в разі необхідності, використовуючи ефективні алгоритми оновлення. Це дозволяє підвищити продуктивність та ефективність додатка.

Крім того, React пропонує розширення, таке як React Native, яке дозволяє розробникам будувати мобільні додатки для платформ iOS та Android з використанням тієї ж синтаксичної структури та концепцій, що і для веброзробки. Такий підхід робить React дуже популярним інструментом для розробки як вебдодатків, так і мобільних додатків.

Хоча React вже більше десяти років, він все ще продовжує розвиватися і вдосконалюватися з часом. Нові версії React завжди приносять із собою важливі

оновлення та покращення. Наразі одним з ключових оновлень є перехід до React 18.

Ця версія має кілька значних змін.

Concurrent Mode (Режим конкурентності): це одна з найбільш очікуваних функцій React 18. Вона дозволяє додаткам React працювати більш ефективно, розподіляючи ресурси процесора таким чином, щоб забезпечити більш плавну відповідь на дії користувачів.

Automatic Batching (Автоматичне пакування): React 18 автоматично пакує оновлення стану компонентів, щоб уникнути непотрібних перерендерів та оптимізувати продуктивність.

Server Components (Серверні компоненти): ця функція дозволяє відокремлювати рендеринг на клієнтській та серверній сторонах. Вона спрощує розробку та покращує швидкість рендерингу за рахунок передачі частини обробки на сервер.

Improved Suspense (Покращений Suspense): Suspense – це механізм, який дозволяє відкладати рендеринг компонентів, поки вони не отримають необхідні дані. У React 18 Suspense стає більш потужним та простішим у використанні.

Оновлення, що вносить React 18, вдосконалили його як інструмент для розробки вебдодатків, підвищивши продуктивність та розширили можливості для розробників.

Розвиток React також значною мірою залежить від великої онлайн-спільноти розробників, яка робить свій внесок у бібліотеку. Ці віддані розробники створюють корисні сторонні інструменти та плагіни для використання рештою спільноти. Чудовим прикладом одного з таких інструментів є Redux, який дає розробникам можливість керувати станом своїх додатків, а потім вносити зміни до цього стану у більш передбачуваний та послідовний спосіб.

React здійняв хвилю після свого першого релізу і продовжує залишатися улюбленим інструментом веброзробників і донині. Попит на розробників, які спеціалізуються на цій технології, залишається високим з року в рік [9, 10].

1.5 Особливості React

JavaScript XML (JSX) – це потужний інструмент у світі веброзробки, який забезпечує зручну та ефективну способу створення інтерфейсів користувача для вебдодатків.

Розроблений для використання у бібліотеці React, JSX відкриває перед програмістами нові можливості в розробці високоякісних, динамічних та швидких вебдодатків. Давайте розглянемо, що це таке і як воно працює.

Основне поняття JSX полягає у поєднанні JavaScript та XML-подібного синтаксису для створення віртуальних DOM-елементів. Він дозволяє розробникам описувати структуру інтерфейсу користувача, використовуючи звичний синтаксис HTML разом з JavaScript виразами (рис. 1.9).

```
2
3 const element = <h1>Hello, world!</h1>;
```

Рисунок 1.9 – Приклад JSX-коду

Цей код легко читається та зрозумілий, що робить JSX ідеальним інструментом для швидкої та продуктивної розробки.

Однією з ключових переваг JSX є його декларативність. Замість того, щоб програмувати низькорівневі маніпуляції DOM, розробник описує, як повинен виглядати інтерфейс, а React бере на себе завдання створення та оновлення віртуального DOM відповідно до цих описів. Це спрощує розробку та підтримку великих і складних додатків, а також підвищує читабельність коду.

При виконанні, JSX-код транспілюється в виклики функцій React.createElement(), які створюють віртуальні DOM-елементи. Ці елементи потім рендеряться в реальному DOM. Такий підхід дозволяє досягти високої продуктивності та швидкодії вебдодатків, оскільки React ефективно оновлює тільки ті частини інтерфейсу, які змінилися.

Крім того, JSX дозволяє розробникам використовувати всі переваги мови

JavaScript, включаючи умови, цикли, функції та інші конструкції, для створення динамічного контенту.

Загалом, JSX відкриває перед розробниками нові горизонти в створенні інтерактивних інтерфейсів користувача для вебдодатків. З його допомогою, розробники можуть швидко та ефективно створювати високоякісні додатки, які надають користувачам захоплюючий досвід використання [11–13].

Компонентність – це методологія розбиття користувацького інтерфейсу на невеликі, автономні блоки, що називаються компонентами. Ці компоненти можуть бути простими, які представляють базові HTML-елементи або складніші, які включають в себе інші компоненти. Ця архітектурна парадигма спрощує розробку і підтримку вебдодатків, особливо тих, які мають складну ієрархію інтерфейсу.

Однією з ключових переваг компонентної архітектури є повторне використання коду. Компоненти можуть бути легко перенесені між проєктами або навіть між різними частинами одного проєкту, що дозволяє значно зменшити кількість коду та підтримувати його у стані, в якому зміни в одному місці автоматично відображаються в інших.

Крім того, компоненти в React мають власний стан (state), який визначає їх поведінку та вигляд. Це дозволяє створювати динамічні та відзивчиві інтерфейси, що змінюються відповідно до користувацького введення або змін в даних.

Узагальнюючи, компонентність є не лише ключовою особливістю React, а й фундаментальним принципом його дизайну, який визначає його вагомість та популярність серед розробників вебдодатків. Ця архітектурна концепція забезпечує модульність, повторне використання та легкість розробки, що робить React важливим інструментом у сучасній веброзробці [11–13].

Virtual DOM (VDOM) виступає в React як ефективний механізм оновлення реального Document Object Model (DOM), що використовується для відображення вебсторінок. Цей підхід використовується для зменшення витрат на оновлення елементів сторінки та покращення продуктивності додатків React.

Що таке DOM: Document Object Model (DOM) є програмним інтерфейсом для HTML і XML документів, що використовується для представлення структури документа як дерева вузлів. DOM надає можливість сценаріям (наприклад, JavaScript) доступу до вмісту і структури документа, а також можливість зміни цього вмісту.

Як працює VDOM: коли стан або властивості компонента React змінюються, Virtual DOM спочатку оновлюється згідно з цими змінами. Після цього він порівнюється з попереднім віртуальним DOM, щоб визначити мінімальну кількість змін, які необхідно внести до фактичного DOM для відображення нового стану. Це дозволяє здійснювати швидкі та ефективні оновлення інтерфейсу користувача.

Основні особливості VDOM.

Швидкі оновлення: завдяки використанню віртуального представлення DOM, React може розраховувати мінімальну кількість змін, потрібних для оновлення реального DOM, що призводить до прискорення оновлень інтерфейсу користувача.

Покращена продуктивність: алгоритм порівняння та оновлення віртуального DOM оптимізований для швидкості, що дозволяє покращити загальну продуктивність додатків React.

Простий у використанні API: React надає зручний та інтуїтивно зрозумілий інтерфейс для створення та оновлення віртуального DOM, спрощуючи розробку складних інтерфейсів користувача.

Підвищена модульність: використання віртуального DOM дозволяє відокремлювати логіку рендерингу від реального DOM, що полегшує управління та перевикористання компонентів.

Рендеринг на стороні сервера: віртуальний DOM робить можливим виконання рендерингу на стороні сервера, що покращує початкову продуктивність завантаження додатків і покращує доступність для пошукових систем.

Отже, використання віртуального DOM є ключовим фактором у

забезпеченні ефективності та продуктивності додатків React, спрощуючи процес оновлення інтерфейсу користувача та полегшуючи розробку складних додатків [11–13].

Односпрямованість потоку даних (One-way Data Binding) – це концепція, яка використовується в бібліотеках та фреймворках, таких як React, для управління станом даних та оновленням користувацького інтерфейсу.

У React односпрямованість потоку даних означає, що дані можуть пересилатися тільки в одному напрямку: від батьківського компонента до дочірніх компонентів. Це означає, що зміни в стані батьківського компонента можуть впливати на дочірні компоненти, але не навпаки. Дочірні компоненти не можуть безпосередньо змінювати стан батьківського компонента або інших дочірніх компонентів (рис. 1.10).

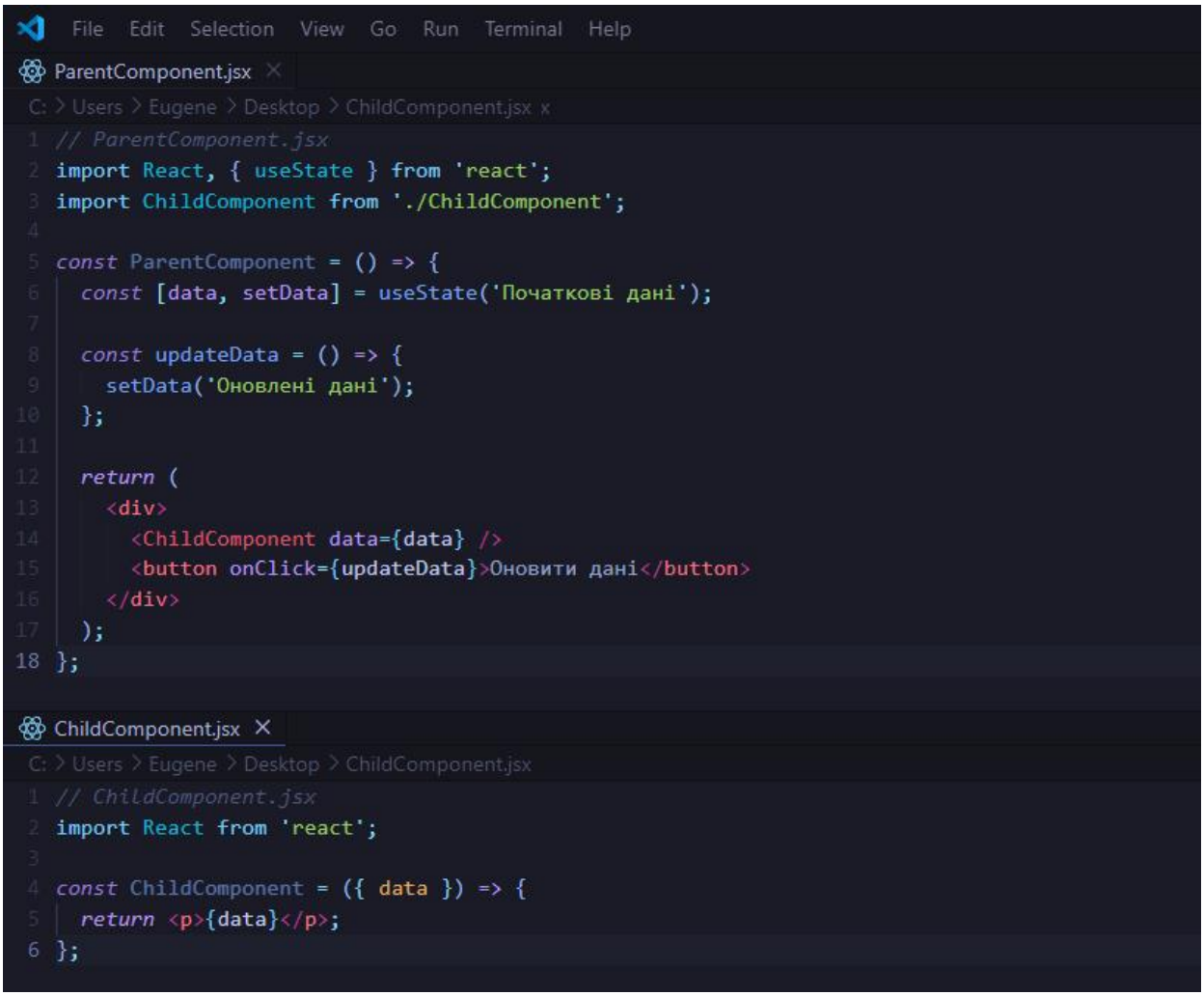

Рисунок 1.10 – Приклад One-way Data Binding

Розглянемо переваги односпрямованого потоку даних в React.

Простота управління станом: односпрямований потік даних робить управління станом простішим і передбачуваним.

Легкість налагодження: завдяки односпрямованому потоку даних легше налагоджувати програми, оскільки логіка оновлення даних більш прозора і передбачувана.

Підвищення продуктивності: React може ефективно керувати перерендерингом компонентів через використання віртуального DOM і розумні алгоритми обробки змін.

Отже, односпрямований потік даних – це ключова концепція у React, яка сприяє стабільності, простоті та продуктивності додатків. Вона сприяє легкому управлінню станом даних та створює певний порядок у взаємодії компонентів [11–13].

1.6 Поширеність React

Спільнота та підтримка відіграють важливу роль у розвитку будь-якого програмного забезпечення, і React не є винятком. З моменту свого випуску у 2013 році, React набув значної популярності завдяки активній спільноті розробників, що створюється навколо нього.

Спільнота React постійно зростає і розвивається. Розробники з усього світу стрімко приєднуються до неї, допомагаючи один одному у розв’язанні проблем, обмінюючись ідеями та створюючи нові рішення для спільного використання. Ця взаємодія формує середовище для розвитку як самого фреймворку, так і додатків, які на ньому базуються.

Завдяки залученості спільноти та великій кількості ресурсів, таких як документація, блоги, форуми та соціальні мережі, підтримка розробників React завжди на високому рівні. Це допомагає новачкам швидко вивчити фреймворк та знайти відповіді на свої запитання, а досвідченим розробникам – розв’язувати

складні проблеми та розширити свої знання.

В результаті поєднання постійно зростаючої спільноти та відмінної підтримки розробниками, React став одним із найпопулярнішим і найрозповсюдженішим фреймворком для розробки вебінтерфейсів сьогодні. Його широке використання в індустрії програмного забезпечення та популярність серед розробників свідчать про його значний вплив на сучасну веброзробку (рис. 1.11).

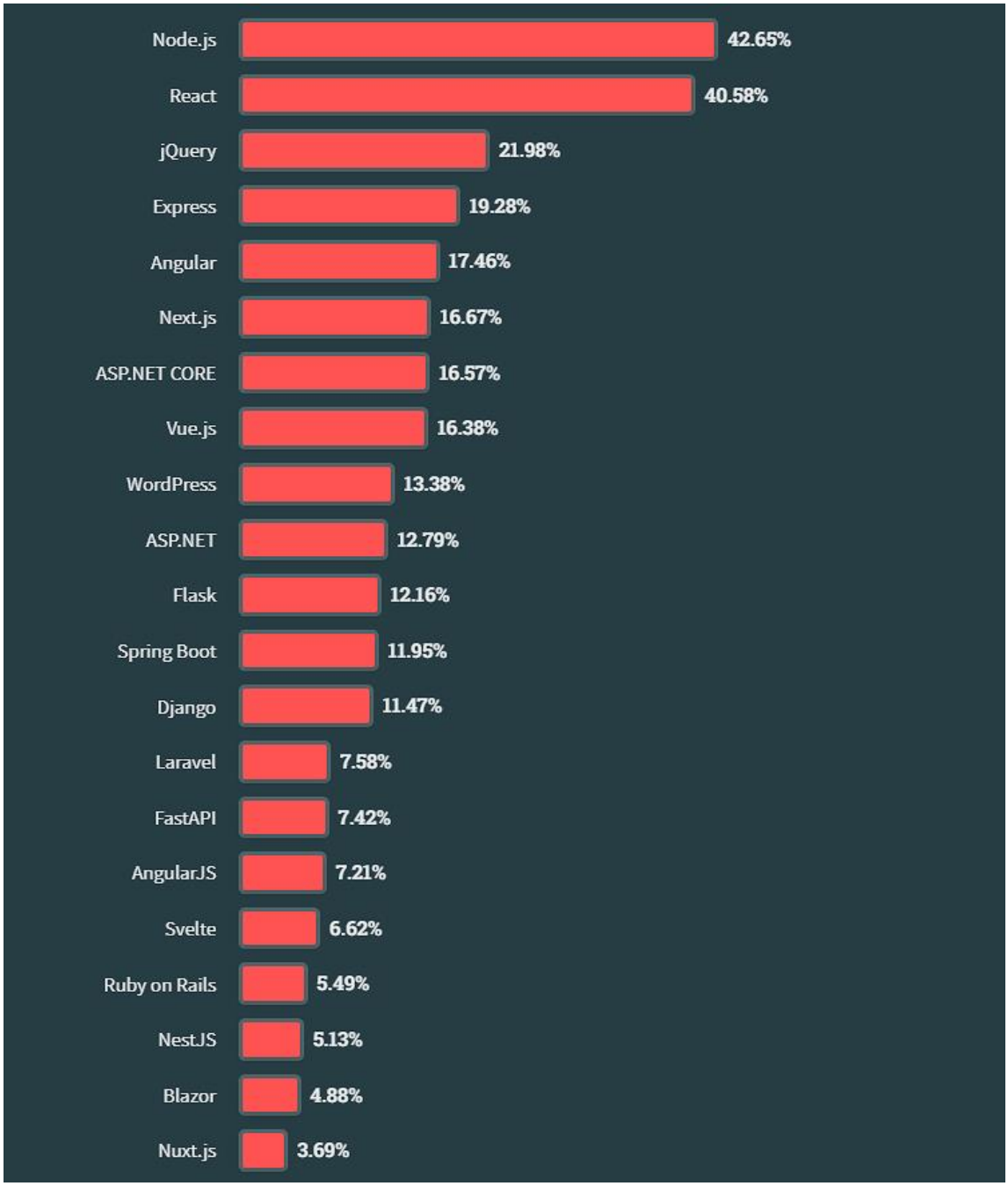

Рисунок 1.11 –Найпопулярніші JavaScript фреймворки 2023 [14]

1.7 Висновки до розділу 1

Розробка вебзастосунків є складним процесом, який вимагає глибокого розуміння різноманітних аспектів, включаючи методи розробки, структуру вебсторінок, мови розмітки та стилів, а також вибір середовища розробки. Цей процес вимагає постійного вдосконалення та адаптації до змін у технологічному середовищі та потребах користувачів.

React, завдяки своїм унікальним особливостям, значно полегшує розробку. Його компонентна архітектура дозволяє розбивати складні проєкти на більш компактні компоненти, що спрощує розуміння та підтримку коду. Віртуальний DOM робить маніпуляції з DOM більш ефективними, а використання JSX спрощує написання коду, роблячи його більш зрозумілим та ефективним. Односпрямованість потоку даних у React сприяє прозорій та передбачуваній роботі додатків.

Отже, розробка вебзастосунків – це не лише технічний процес, але й творчий виклик, що потребує комплексного підходу та поєднання різних інструментів та підходів. Хоча вона може бути складною та вимагати значних зусиль, але із розвитком технологій з’являються нові інструменти та методи, що роблять цей процес доступнішим та ефективнішим.

2 РОЗРОБКА ДІАГРАМИ ПРЕЦЕДЕНТІВ

2.1 Теоретичні відомості

Діаграма прецедентів – це графічне зображення, яке використовується для моделювання взаємодії між системою та її користувачами (акторами). Вона допомагає візуалізувати функціональність системи шляхом ідентифікації основних прецедентів (функціональних можливостей або те, що система може зробити) та акторів (користувачів або систем, які взаємодіють з цими прецедентами). Може описуватись текстом або у вигляді діаграми.

Прецеденти – це конкретні дії або функціональні можливості, які виконує система для задоволення потреб користувачів або виконання певних завдань. Наприклад, у вебзастосунку можуть бути прецеденти для реєстрації користувача, входу в систему, створення нового запису тощо.

Актори – це суб’єкти або ролі, які взаємодіють з системою, але не належать до неї. Вони можуть бути реальними людьми, іншими системами, зовнішніми компонентами. Наприклад, у вебзастосунку користувачі, адміністратори, зовнішні сервіси, можуть бути акторами. Навіть час, якщо від нього залежить запуск певних подій у системі.

Актори визначаються на основі їх ролей у системі та способів їх взаємодії з прецедентами [15, 16].

2.2 Побудова діаграми

Перш ніж приступити до побудови діаграми прецедентів, необхідно підготувати таблицю (табл. 2.1), в якій будуть визначені актори та прецеденти нашої системи.

Таблиця 2.1 – Таблиця акторів та прецедентів

Актори	Прецеденти
Клієнт	Пошук товару
	Перегляд товару
	Реєстрація на сайті
	Вхід до особистого кабінету
	Додавання товар до кошика
	Видалення товару із кошика
	Оплата замовлення
	Зміна паролю
Адміністратор	Пошук товару
	Перегляд товару
	Редагування товару
	Додавання товару
	Видалення товару
	Керування замовленнями
	Управління користувачами

Клієнт.

Пошук товару: клієнт може здійснювати пошук товарів за різними параметрами для зручного вибору.

Перегляд товару: клієнт може переглядати детальну інформацію про товар, включаючи опис, ціну та зображення.

Реєстрація на сайті: клієнт може створити обліковий запис на сайті для зручності покупок та отримання персоналізованих послуг.

Вхід до особистого кабінету: клієнт може увійти до свого особистого кабінету для перегляду історії замовлень та зміни особистої інформації.

Додавання товар до кошика: клієнт може додавати обрані товари до свого кошика для подальшого оформлення замовлення.

Видалення товару із кошика: клієнт може видаляти товари зі свого кошика, якщо вони більше не потрібні.

Оплата замовлення: клієнт може оплатити своє замовлення за допомогою різних методів оплати.

Зміна паролю: клієнт може змінювати свій пароль для забезпечення безпеки облікового запису.

Адміністратор.

Пошук товару: адміністратор може здійснювати пошук товарів у каталозі для адміністрування та редагування.

Перегляд товару: адміністратор може переглядати детальну інформацію про товари у каталозі.

Редагування товару: адміністратор може редагувати інформацію про існуючі товари, включаючи ціну, опис, зображення тощо.

Додавання товару: адміністратор може додавати нові товари до каталогу, вказуючи всю необхідну інформацію.

Видалення товару: адміністратор може видаляти товари з каталогу, якщо вони більше не продаються або стали недоступними.

Керування замовленнями: адміністратор може переглядати та керувати замовленнями клієнтів, включаючи підтвердження, відміну або відправку замовлень.

Управління користувачами: адміністратор може керувати обліковими записами користувачів, включаючи створення, редагування та видалення.

Після завершення цього етапу маємо достатньо даних для початку побудови діаграми (рис. 2.1). Ці дані виступатимуть основою для виявлення взаємодій між акторами та прецедентами, що дозволить нам краще зрозуміти систему та її функціонування. Діаграма буде важливим інструментом для візуалізації цих взаємодій і надасть чітку картину зв’язки між різними складовими системи.

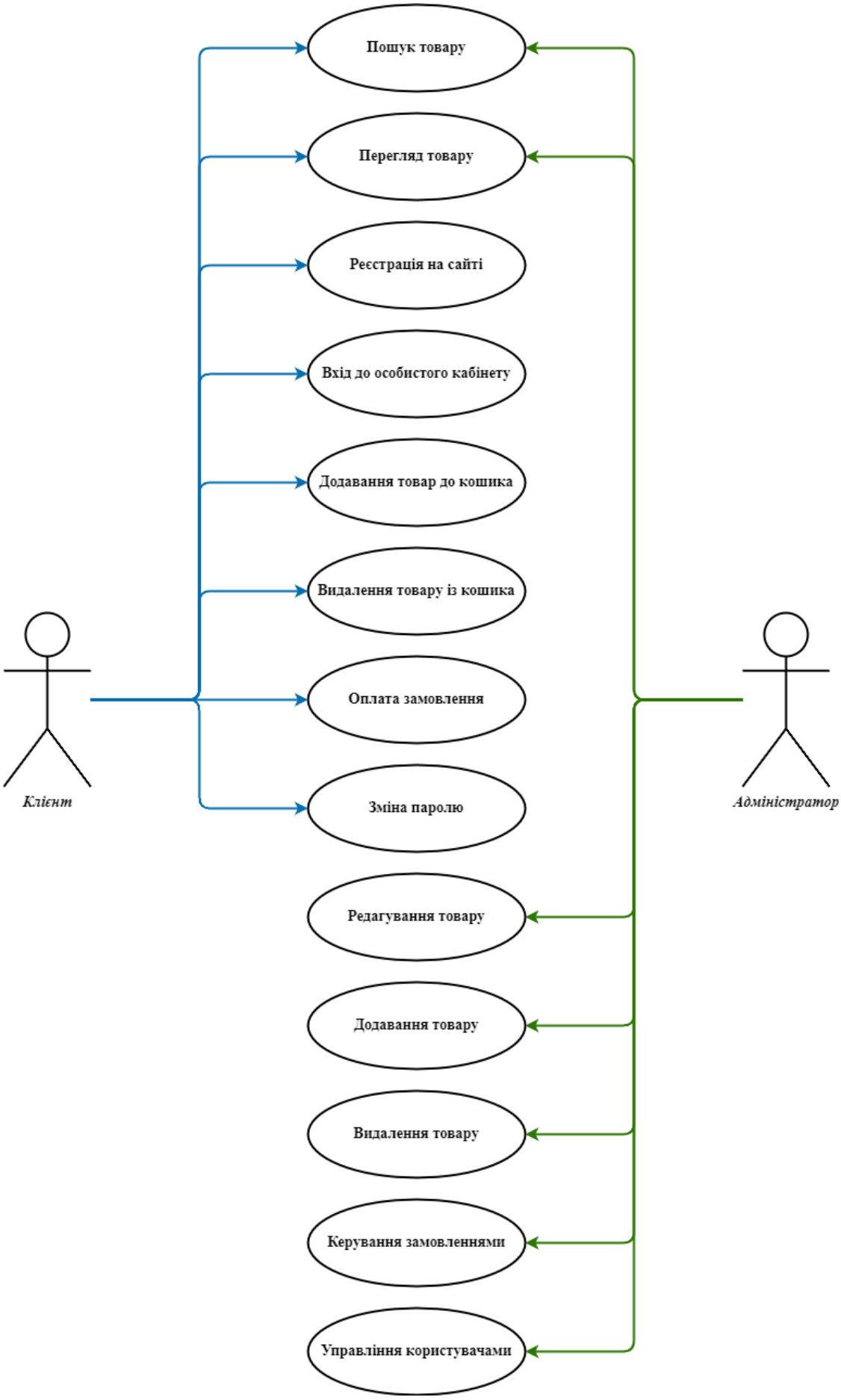

Рисунок 2.1 – Діаграма прецедентів

2.3 Висновки до розділу 2

У веброзробці діаграма прецедентів є критичним інструментом, який дозволяє ретельно проаналізувати функціональні можливості системи. Вона відображає взаємодію користувачів із вебзастосунком, визначає, як вони будуть використовувати систему та як вона буде реагувати на їхні дії. Це важливо не лише для забезпечення задоволення користувачів, а й для досягнення бізнес-цілей.

Процес створення діаграми прецедентів допомагає розкрити всі потенційні сценарії взаємодії з системою, включаючи як основні, так і альтернативні шляхи. Це дає можливість розробникам і бізнес-аналітикам зрозуміти потреби користувачів і врахувати їх при проєктуванні вебзастосунку.

Відповідно, ретельний аналіз прецедентів та авторів важливе для успішної реалізації вебзастосунку. Цей етап дозволяє не лише визначити функціональні вимоги, а й встановити фундамент для подальшого проєктування та розробки. Таким чином, діаграма прецедентів є необхідним інструментом для ефективної та успішної роботи над проєктом.

3 РОЗРОБКА ІНТЕРНЕТ МАГАЗИНУ

3.1 Структура вебзастосунку

Розробка в React допускає гнучкість у створенні файлової структури, оскільки вона не має жорстких правил щодо організації коду. Зазвичай проєктна структура розбивається на компоненти, контейнери, стилі, утиліти та інші папки залежно від потреб і розміру проєкту.

Для створення базової структури можна використати таку команду (рис. 3.1).

```
npx create-react-app online-shop
```

Рисунок 3.1 – Команда для базової структури

Вона автоматично створить каркас вашого React-застосунку із назвою “online-shop”, включаючи основні файли та каталоги, що забезпечать готову основу для подальшої розробки.

Розглянемо основні компоненти файлової структури.

Public: це папка, де розміщуються статичні файли, такі як HTML-файл, зображення, шрифти тощо. Файли, розміщені в цій папці, будуть доступні для клієнтів напряму через вебсервер.

Src: це основна папка вашого додатка React, де знаходяться всі вихідні файли вашого проєкту.

App.js: це головний компонент вашого додатка. Він містить основний код, який визначає структуру та поведінку вашого додатка.

Assets: тут зазвичай розміщуються різні ресурси, такі як зображення, CSS файли, SVG, анімації тощо. Ці ресурси можуть використовуватися в компонентах вашого додатка.

Components: ця папка зазвичай містить ваші власні компоненти React. Кожен компонент може бути відповідальним за конкретну частину UI або функціональність.

Pages: у цій папці зазвичай розміщуються компоненти, які відповідають за окремі сторінки вашого додатка. Наприклад, якщо ваш додаток має сторінки «Домашня», «Про нас», «Контакти» тощо, ви можете розмістити компоненти цих сторінок в цій папці.

Це загальна структура, але ви можете розширити або змінити її відповідно до ваших потреб. Ключове завдання – забезпечити легкість навігації та розуміння коду. Ось приклад типової структури вебзастосунку React (рис. 3.2).

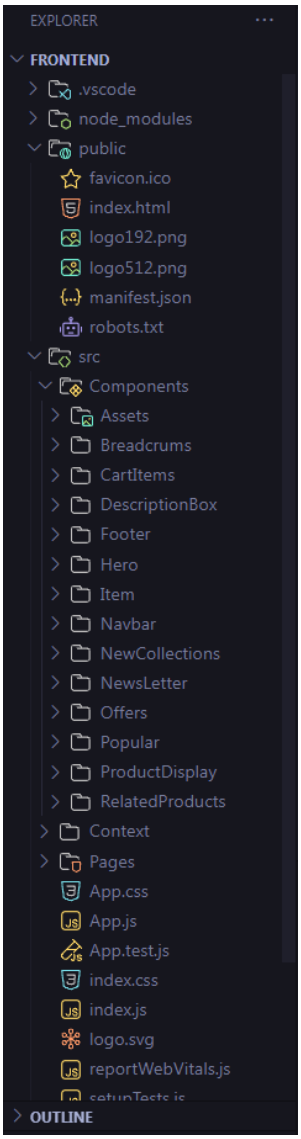

Рисунок 3.2 – Файлова структура React проєкту

3.2 Компоненти вебзастосунку

Меню (Navbar): навігаційне меню, яке включає логотип сайту, посилання на інші сторінки (Головна, Чоловікам, Жінкам, Дітям), кнопку для входу, а також іконку кошика, що відображає кількість товарів у ньому. Активний пункт меню підкреслюється, забезпечуючи зручну навігацію для користувачів (рис. 3.3, 3.4).

Рисунок 3.3 – Вигляд компоненту Navbar у вебзастосунку

```jsx
import React, { useContext, useState } from 'react'
import './Navbar.css'
import cart_icon from '../Assets/cart_icon.png'
import { Link } from 'react-router-dom'
import { ShopContext } from '../../Context/ShopContext'
import '../../Pages/CSS/Container.css'


export const Navbar = () => {

    const [menu,setMenu] = useState("shop");
    const {getTotalCartItems} = useContext(ShopContext);

  return (
    <div className='navbar container'>
        <div className="nav-logo">
            <p className='icon'>Ø</p>
            <p className='icontext'>SPORTSHOP</p>
        </div>
        <ul className="nav-menu">
            <li onClick={()=>{setMenu('shop')}}><Link style={{textDecoration:
            'none'}} to='/'>Головна</Link>{menu==="shop"?<hr/>:<></>}</li>
            <li onClick={()=>{setMenu('men')}}><Link style={{textDecoration:
            'none'}} to='/men'>Чоловікам</Link>{menu==="men"?<hr/>:<></>}</li>
            <li onClick={()=>{setMenu('women')}}><Link style={{textDecoration:
            'none'}} to='/women'>Жінкам</Link>                    li>
            <li onClick={()=>{setMenu('kids')}}                    :
            'none'}} to='/kids'>Дітям</Link>{menu==="kids"?<hr/>:<></>}</li>
        </ul>
        <div className="nav-login-cart">
            <Link to='/login'><button>Вхід</button></Link>
            <Link to='/cart'><img src={cart_icon} alt="" /></Link>
            <div className="nav-cart-count">{getTotalCartItems()}</div>
        </div>
    </div>
  )
}
```

(loading...) const menu: any

Рисунок 3.4 – Код компоненту Navbar

Футер (Footer): нижня частина сайту, яка містить логотип SPORTSHOP, навігаційні посилання на розділи сайту (Компанія, Продукти, Офіси, Про нас, Контакти), іконки соціальних мереж (Instagram, Pinterest, WhatsApp), а також інформацію про авторські права. Футер забезпечує зручний доступ до основної інформації про компанію та її контакти, а також дозволяє користувачам зв’язатися через соціальні мережі (рис. 3.5, 3.6).

Рисунок 3.5 – Вигляд компоненту Footer у вебзастосунку

```jsx
import React from 'react'
import './Footer.css'
import instagram_icon from '../Assets/instagram_icon.png'
import pintester_icon from '../Assets/pintester_icon.png'
import whatsapp_icon from '../Assets/whatsapp_icon.png'

export const Footer = () => {
  return (
    <div className='footer'>
        <div className="footer-logo">
            <p className='icon'>Ø</p>
            <p className='icontext'>SPORTSHOP</p>
        </div>
        <ul className="footer-links">
            <li>Компанія</li>
            <li>Продукти</li>
            <li>Офіси</li>
            <li>О нас</li>
            <li>Контакти</li>
        </ul>
        <div className="footer-social-icons">
            <img src={instagram_icon} alt="" />
            <img src={pintester_icon} alt="" />
            <img src={whatsapp_icon} alt="" />
        </div>
        <div className="footer-copyright">
            <hr />
            <p>Copyright @ 2024 - All Right Reserved</p>
        </div>
    </div>
  )
}

export default Footer;
```

Рисунок 3.6 – Код компоненту Footer

Далі розглянемо головну сторінку вебзастосунку (рис. 3.7, 3.8).

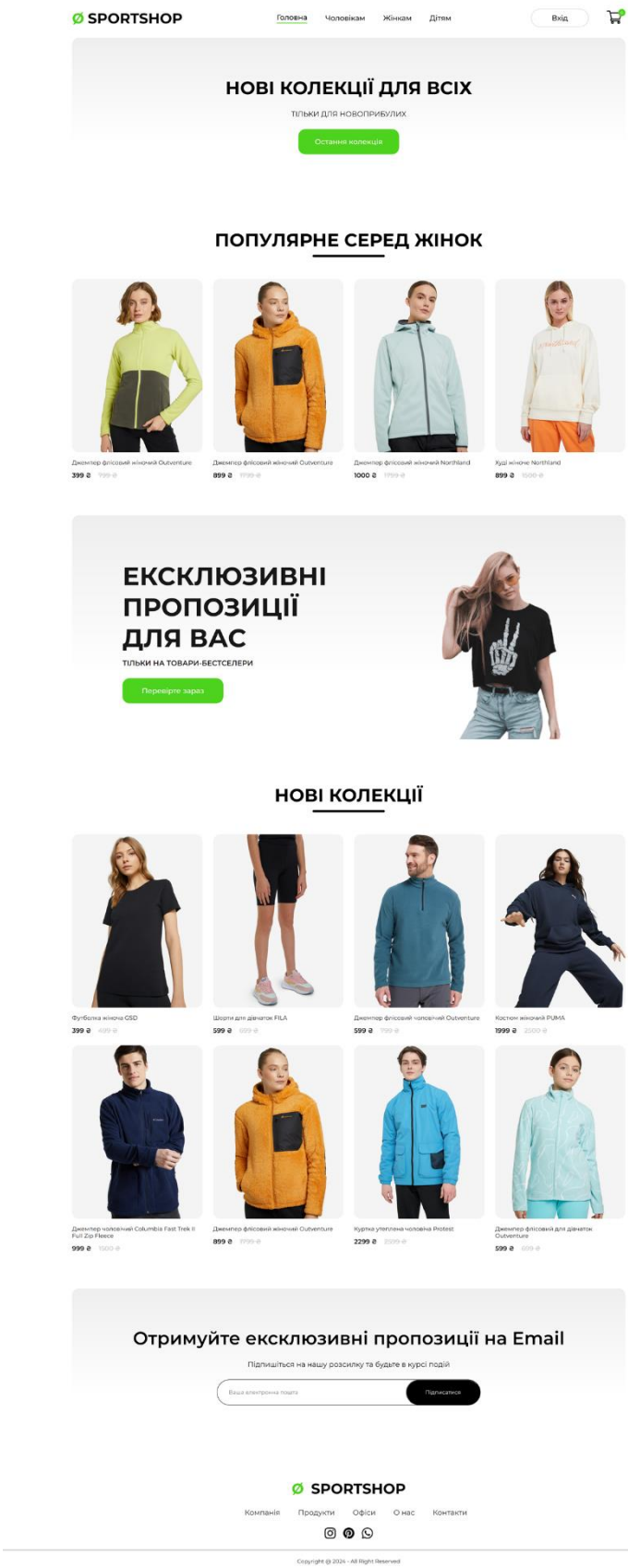

Рисунок 3.7 – Вигляд головної сторінки вебзастосунку

```jsx
import React from 'react'
import { Hero } from '../Components/Hero/Hero'
import { Popular } from '../Components/Popular/Popular'
import { Offers } from '../Components/Offers/Offers'
import { NewCollections } from '../Components/NewCollections/NewCollections'
import { NewsLetter } from '../Components/NewsLetter/NewsLetter'
import './CSS/Container.css'

export const Shop = () => {
  return (
    <div className='container'>
        <Hero/>
        <Popular/>
        <Offers/>
        <NewCollections/>
        <NewsLetter/>
    </div>
  )
}
```

Рисунок 3.8 – Код головної сторінки

Головна сторінка (Shop) складається з таких окремих елементів, як Hero, Popular, Offers, NewCollections та NewsLetter (рис. 3.9 – 3.13).

```jsx
import React from 'react'
import './Hero.css'

export const Hero = () => {
  return (
    <div className='hero'>
            <h1>НОВІ КОЛЕКЦІЇ ДЛЯ ВСІХ</h1>
            <p>ТІЛЬКИ ДЛЯ НОВОПРИБУЛИХ</p>
            <button>Остання колекція</button>
    </div>
  )
}
```

Рисунок 3.9 – Код компоненту Hero

```jsx
import React from 'react'
import './Popular.css'
import data_product from '../Assets/data'
import Item from '../Item/Item'

export const Popular = () => {
  return (
    <div className='popular'>
        <h1>ПОПУЛЯРНЕ СЕРЕД ЖІНОК</h1>
        <hr />
        <div className="popular-item">
            {data_product.map((item,i)=>{
                return <Item key={i} id={item.id} name={item.name} image={item.
                image} new_price={item.new_price} old_price={item.old_price}/>
            })}
        </div>
    </div>
  )
}
```

Рисунок 3.10 – Код компоненту Popular

```jsx
import React from 'react'
import './Offers.css'
import exclusive_image from '../Assets/exclusive_image.png'

export const Offers = () => {
  return (
    <div className='offers'>
        <div className="offers-left">
            <h1>ЕКСКЛЮЗИВНІ</h1>
            <h1>ПРОПОЗИЦІЇ</h1>
            <h1>ДЛЯ ВАС</h1>
            <p>ТІЛЬКИ НА ТОВАРИ-БЕСТСЕЛЕРИ</p>
            <button>Перевірте зараз</button>
        </div>
        <div className="offers-right">
            <img src={exclusive_image} alt="" />
        </div>
    </div>
  )
}

export default Offers;
```

Рисунок 3.11 – Код компоненту Offers

```jsx
import React from 'react'
import './NewCollections.css'
import new_collection from '../Assets/new_collections'
import Item from '../Item/Item'

export const NewCollections = () => {
  return (
    <div className='newcollections'>
        <h1>НОВІ КОЛЕКЦІЇ</h1>
        <hr />
        <div className="collections">
            {new_collection.map((item,i)=>{
                return <Item key={i} id={item.id} name={item.name} image={item.
                image} new_price={item.new_price} old_price={item.old_price}/>
            })}
        </div>
    </div>
  )
}

export default NewCollections;
```

Рисунок 3.12 – Код компоненту NewCollection

```jsx
import React from 'react'
import './NewsLetter.css'

export const NewsLetter = () => {
  return (
    <div className='newsletter'>
        <h1>Отримуйте ексклюзивні пропозиції на Email</h1>
        <p>Підпишіться на нашу розсилку та будьте в курсі подій</p>
        <div>
            <input type="email" placeholder='Ваша електронна пошта' />
            <button>Підписатися</button>
        </div>
    </div>
  )
}

export default NewsLetter;
```

Рисунок 3.13 – Код компоненту NewsLetter

Давайте більш детально розглянемо компоненти Popular та NewCollections. Кожен з них формується завдяки імпорту масивів динних товарів відповідного компоненту (рис. 3.14, 3.15), та передаючи їх властивості завдяки props, іншому компоненту Item (рис. 3.16), який представляє собою шаблон картки товару.

data.js | new_collections.js

```jsx
import p1_img from "./product_12.png";
import p2_img from "./product_35.png";
import p3_img from "./product_14.png";
import p4_img from "./product_8.png";
import p5_img from "./product_15.png";
import p6_img from "./product_2.png";
import p7_img from "./product_17.png";
import p8_img from "./product_28.png";

let new_collections = [
  {
    id: 12,
    name: "Футболка жіноча GSD",
    image: p1_img,
    new_price: 399,
    old_price: 499,
  },
  {
    id: 35,
    name: "Шорти для дівчаток FILA",
    image: p2_img,
    new_price: 599,
    old_price: 699,
  },
```

```
  {
    id: 14,
    name: "Джемпер флісовий чоловічий Outventure",
    image: p3_img,
    new_price: 599,
    old_price: 799,
  },
  {
    id: 8,
    name: "Костюм жіночий PUMA",
    image: p4_img,
    new_price: 1999,
    old_price: 2500,
  },
  {
    id: 15,
    name: "Джемпер чоловічий Columbia Fast Trek II Full Zip Fleece",
    image: p5_img,
    new_price: 999,
    old_price: 1500,
  },
  {
    id: 2,
    name: "Джемпер флісовий жіночий Outventure",
    image: p6_img,
    new_price: 899,
    old_price: 1799,
  },
  {
    id: 17,
    name: "Куртка утеплена чоловіча Protest",
    image: p7_img,
    new_price: 2299,
```

Рисунок 3.14 – Масив даних NewCollections

data.js | new_collections.js

```
import p1_img from './product_1.png'
import p2_img from './product_2.png'
import p3_img from './product_3.png'
import p4_img from './product_4.png'

let data_product = [
  {
    id: 1,
    name: "Джемпер флісовий жіночий Outventure",
    image: p1_img,
    new_price: 399,
    old_price: 799,
  },
  {id: 2,
    name: "Джемпер флісовий жіночий Outventure",
    image: p2_img,
    new_price: 899,
    old_price: 1799,
  },
  {id:3,
    name: "Джемпер флісовий жіночий Northland",
    image: p3_img,
    new_price: 1000,
    old_price: 1799,
  },
  {id:4,
    name: "Худі жіноче Northland",
    image: p4_img,
    new_price: 899,
    old_price: 1500,
  },
];

export default data_product;
```

Рисунок 3.15 – Масив даних Popular

```jsx
import React from 'react'
import './Item.css'
import { Link } from 'react-router-dom'

export const Item = (props) => {
  return (
    <div className='item'>
        <Link to={`/product/${props.id}`}><img onClick={window.scrollTo(0,0)}
        src={props.image} alt="" /></Link>
        <p>{props.name}</p>
        <div className="item-prices">
            <div className="item-price-new">
                {props.new_price} ₴
            </div>
            <div className="item-price-old">
                {props.old_price} ₴
            </div>
        </div>
    </div>
  )
}

export default Item;
```

Рисунок 3.16 – Код компоненту Item

Далі розглянемо формування категорій товарів (рис. 3.17).

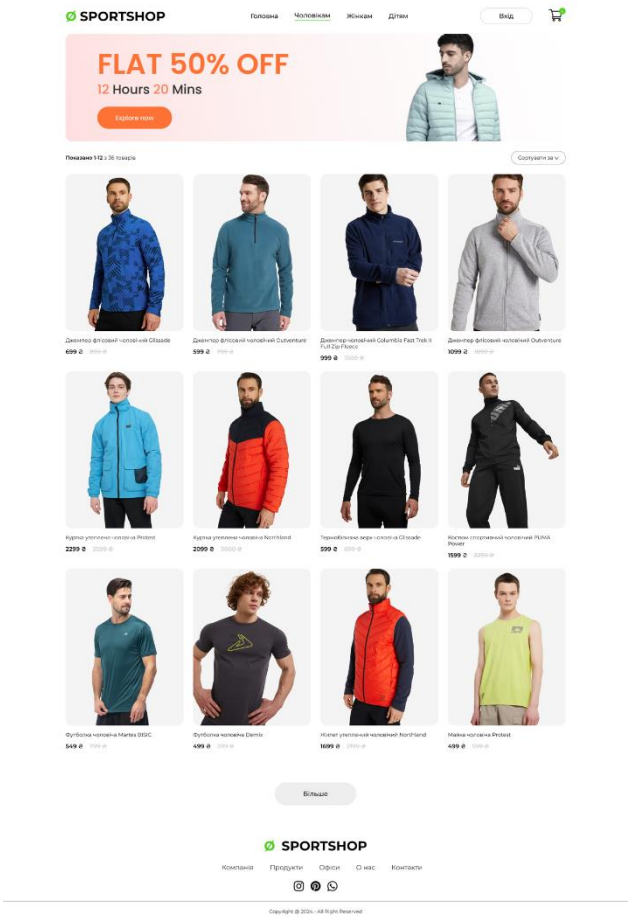

Рисунок 3.17 – Категорія товарів

Для фільтрування товарів по категоріям використовуємо React Router DOM.

Послідовність дій:

а) присвоєння посилання для кожного елементу компонента Navbar;

б) рендер компоненту ShopCategory із відповідним параметром категорії, відбувається за допомогою порівняння посилання Link та path у головному файлі вебзастосунку App.js (рис. 3.18);

в) формування картки товару в компоненті ShopCategory, завдяки порівнянню вхідного параметру із файлу App.js та параметру із масиву даних товарів (рис. 3.19).

```
import './App.css';
import { Navbar } from './Components/Navbar/Navbar';
import { BrowserRouter,Routes,Route} from 'react-router-dom';
import { Shop } from './Pages/Shop';
import { ShopCategory } from './Pages/ShopCategory';
import { Product } from './Pages/Product';
import { Cart } from './Pages/Cart';
import { LoginSign } from './Pages/LoginSign';
import { Footer } from './Components/Footer/Footer';
import men_banner from './Components/Assets/banner_mens.png'
import women_banner from './Components/Assets/banner_women.png'
import kids_banner from './Components/Assets/banner_kids.png'

function App() {
  return (
    <div>
      <BrowserRouter>
      <Navbar/>
      <Routes>
        <Route path='/' element={<Shop/>}/>
        <Route path='/men' element={<ShopCategory banner={men_banner}
        category="men"/>}/>
        <Route path='/women' element={<ShopCategory banner={women_banner}
        category="women"/>}/>
        <Route path='/kids' element={<ShopCategory banner={kids_banner}
        category="kid"/>}/>
        <Route path='/product' element={<Product/>}>
          <Route path=':productId' element={<Product/>}/>
        </Route>
        <Route path='/cart' element={<Cart/>}/>
        <Route path='/login' element={<LoginSign/>}/>
      </Routes>
      <Footer/>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

Рисунок 3.18 – Код головного файлу вебзастосунку App.js

```jsx
import React, { useContext } from 'react'
import './CSS/ShopCategory.css'
import { ShopContext } from '../Context/ShopContext'
import dropdown_icom from '../Components/Assets/dropdown_icon.png'
import Item from '../Components/Item/Item'
import './CSS/Container.css'

export const ShopCategory = (props) => {
  const {all_product} = useContext(ShopContext)
  return (
    <div className='shop-category container'>
      <img className='shopcategory-banner' src={props.banner} alt="" />
      <div className="shopcategory-indexSort">
        <p>
          <span>Показано 1-12</span> з 36 товарів
        </p>
        <div className="shopcategory-sort">
        Сортувати за <img src={dropdown_icom} alt="" />
        </div>
      </div>
      <div className="shopcategory-products">
        {all_product.map((item,i)=>{
          if (props.category===item.category) {
            return <Item key={i} id={item.id} name={item.name} image={item.
            image} new_price={item.new_price} old_price={item.old_price}/>
          }
          else {
            return null;
          }
        })}
      </div>
      <div className="shopcategory-loadmore">
        Більше
      </div>
    </div>
  )
}

export default ShopCategory;
```

Рисунок 3.19 – Код компоненту ShopCategory

Далі розглянемо сторінку окремого товару (рис. 3.20, 3.21).

```jsx
import React, { useContext } from 'react'
import { ShopContext } from '../Context/ShopContext'
import { useParams } from 'react-router-dom';
import Breadcrum from '../Components/Breadcrums/Breadcrum';
import ProductDisplay from '../Components/ProductDisplay/ProductDisplay';
import DescriptionBox from '../Components/DescriptionBox/DescriptionBox';
import RelatedProducts from '../Components/RelatedProducts/RelatedProducts';
import './CSS/Container.css'

export const Product = () => {
  const {all_product} = useContext(ShopContext);
  const {productId} = useParams();
  const product = all_product.find((e)=> e.id === Number(productId))
  return (
    <div className='container'>
      <Breadcrum product={product}/>
      <ProductDisplay product={product}/>
      <DescriptionBox />
      <RelatedProducts />
    </div>
  )
}
```

Рисунок 3.20 – Код сторінки Product

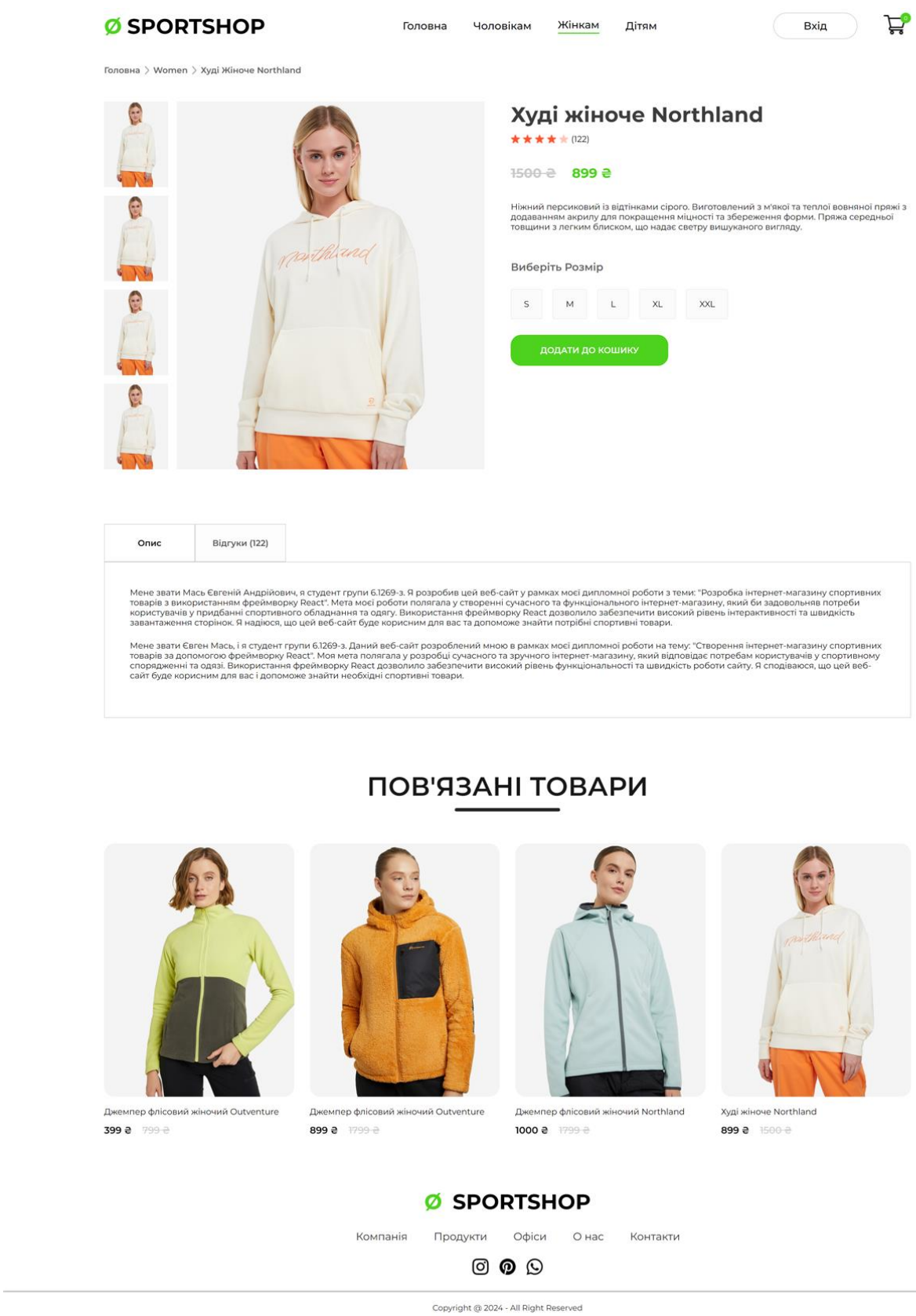

Рисунок 3.21 – Вигляд окремого товару

Сторінка товару (Product) складається з таких окремих компонентів, як Breadcrum, ProductDisplay, DescriptionBox, та RelatedProducts (рис. 3.22 – 3.25).

```jsx
import React, { useContext } from 'react'
import './ProductDisplay.css'
import star_icon from '../Assets/star_icon.png'
import star_dull_icon from '../Assets/star_dull_icon.png'
import { ShopContext } from '../../Context/ShopContext'

export const ProductDisplay = (props) => {
    const {product} = props;
    const {addToCart} = useContext(ShopContext);
  return (
    <div className='productdisplay'>
        <div className="productdisplay-left">
            <div className="productdisplay-img-list">
                <img src={product.image} alt="" />
                <img src={product.image} alt="" />
                <img src={product.image} alt="" />
                <img src={product.image} alt="" />
            </div>
            <div className="productdisplay-img">
                <img className='prodactdisplay-main-img' src={product.image}
                alt="" />
            </div>
        </div>
        <div className="productdisplay-right">
            <h1>{product.name}</h1>
            <div className="productdisplay-right-stars">
                <img src={star_icon} alt="" />
                <img src={star_icon} alt="" />
                <img src={star_icon} alt="" />
                <img src={star_icon} alt="" />
                <img src={star_dull_icon} alt="" />
                <p>(122)</p>
            </div>
            <div className="productdisplay-right-prices">
                <div className="productdisplay-right-price-old">{product.
                old_price} ₴</div>
                <div className="productdisplay-right-price-new">{product.
                new_price} ₴</div>
            </div>
            <div className="productdisplay-right-description">
                Ніжний персиковий із відтінками сірого. Виготовлений з м'якої
                та теплої вовняної пряжі з додаванням акрилу для покращення
                міцності та збереження форми. Пряжа середньої товщини з легким
                блиском, що надає светру вишуканого вигляду.
            </div>
            <div className="productdisplay-right-size">
                <h1>Виберіть Розмір</h1>
                <div className="productdisplay-right-sizes">
                    <div>S</div>
                    <div>M</div>
                    <div>L</div>
                    <div>XL</div>
                    <div>XXL</div>
                </div>
            </div>
            <button onClick={()=>{addToCart(product.id)}}>ДОДАТИ ДО КОШИКУ</
            button>
        </div>
    </div>
  )
```

Рисунок 3.22 – Код компоненту DisplayProduct

```jsx
import React from 'react'
import './Breadcrum.css'
import arrow_icon from '../Assets/breadcrum_arrow.png'

export const Breadcrum = (props) => {
    const {product} = props;
  return (
    <div className='breadcrum'>
        Головна <img src={arrow_icon} alt=""/> {product.category} <img src=
        {arrow_icon} alt=""/> {product.name}
    </div>
  )
}

export default Breadcrum;
```

Рисунок 3.23 – Код компоненту Breadcrum

```jsx
import React from 'react'
import './DescriptionBox.css'

export const DescriptionBox = () => {
  return (
    <div className='descriptionbox'>
        <div className="discription-navigator">
            <div className="discription-nav-box">Опис</div>
            <div className="discription-nav-box fade">Відгуки (122)</div>
        </div>
        <div className="discriptionbox-description">
            <p>Мене звати Мась Євгеній Андрійович, я студент групи 6.1269-з. Я
            розробив цей веб-сайт у рамках моєї дипломної роботи з теми:
            "Розробка інтернет-магазину спортивних товарів з використанням
            фреймворку React". Мета моєї роботи полягала у створенні сучасного
            та функціонального інтернет-магазину, який би задовольняв потреби
            користувачів у придбанні спортивного обладнання та одягу.
            Використання фреймворку React дозволило забезпечити високий рівень
            інтерактивності та швидкість завантаження сторінок. Я надіюся, що
            цей веб-сайт буде корисним для вас та допоможе знайти потрібні
            спортивні товари.</p>
            <p>Мене звати Євген Мась, і я студент групи 6.1269-з. Даний
            веб-сайт розроблений мною в рамках моєї дипломної роботи на тему:
            "Створення інтернет-магазину спортивних товарів за допомогою
            фреймворку React". Моя мета полягала у розробці сучасного та
            зручного інтернет-магазину, який відповідає потребам користувачів у
            спортивному спорядженні та одязі. Використання фреймворку React
            дозволило забезпечити високий рівень функціональності та швидкість
            роботи сайту. Я сподіваюся, що цей веб-сайт буде корисним для вас і
            допоможе знайти необхідні спортивні товари.</p>
        </div>
    </div>
  )
}

export default DescriptionBox;
```

Рисунок 3.24 – Код компоненту DescriptionBox

```jsx
import React from 'react'
import './RelatedProducts.css'
import data_product from '../Assets/data'
import Item from '../Item/Item'

export const RelatedProducts = () => {
  return (
    <div className='relatedproducts'>
        <h1>ПОВ'ЯЗАНІ ТОВАРИ</h1>
        <hr />
        <div className="relatedproducts-item">
            {data_product.map((item,i)=> {
                return <Item key={i} id={item.id} name={item.name} image={item.
                image} new_price={item.new_price} old_price={item.old_price}/>
            })}
        </div>
    </div>
  )
}

export default RelatedProducts;
```

Рисунок 3.25 – Код компоненту RelatedProducts

Далі розглянемо сторінку реєстрації особистого кабінету (LoginSign) (рис. 3.26, 3.27). Сторінка кошику товарів (див. рис. 3.28).

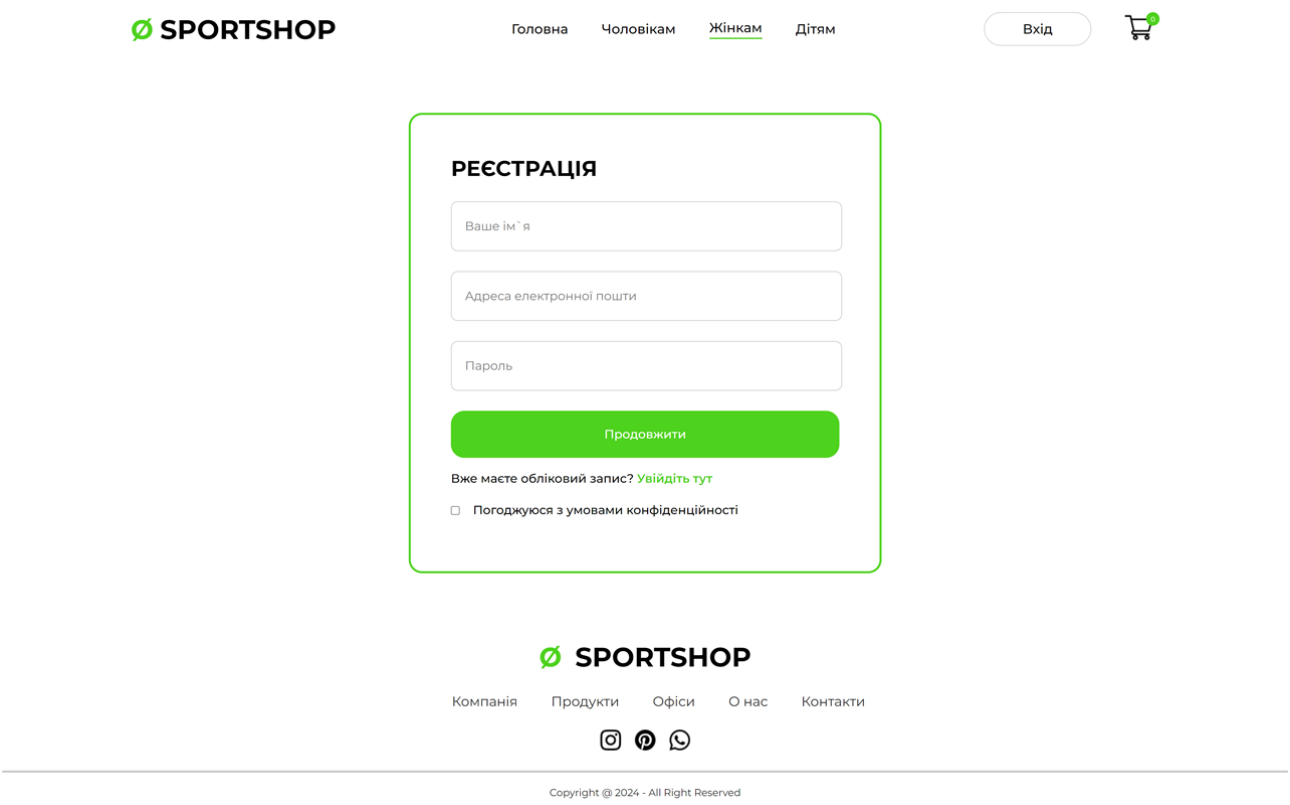

Рисунок 3.26 – Вигляд сторінки входу та реєстрації

```jsx
import React from 'react'
import './CSS/LoginSign.css'
import './CSS/Container.css'

export const LoginSign = () => {
  return (
    <div className='loginsign container'>
      <div className="loginsign-container">
        <h1>РЕЄСТРАЦІЯ</h1>
        <div className="loginsign-fields">
          <input type="text" placeholder='Ваше ім`я' />
          <input type="email" placeholder='Адреса електронної пошти' />
          <input type="password" placeholder='Пароль' />
        </div>
        <button>Продовжити</button>
        <p className="loginsign-login">Вже маєте обліковий запис?
        <span>Увійдіть тут</span></p>
        <div className="loginsign-agree">
          <input type="checkbox" name='' id='' />
          <p>Погоджуюся з умовами конфіденційності</p>
        </div>
      </div>
    </div>
  )
}

export default LoginSign;
```

Рисунок 3.27 – Код компоненту LoginSign

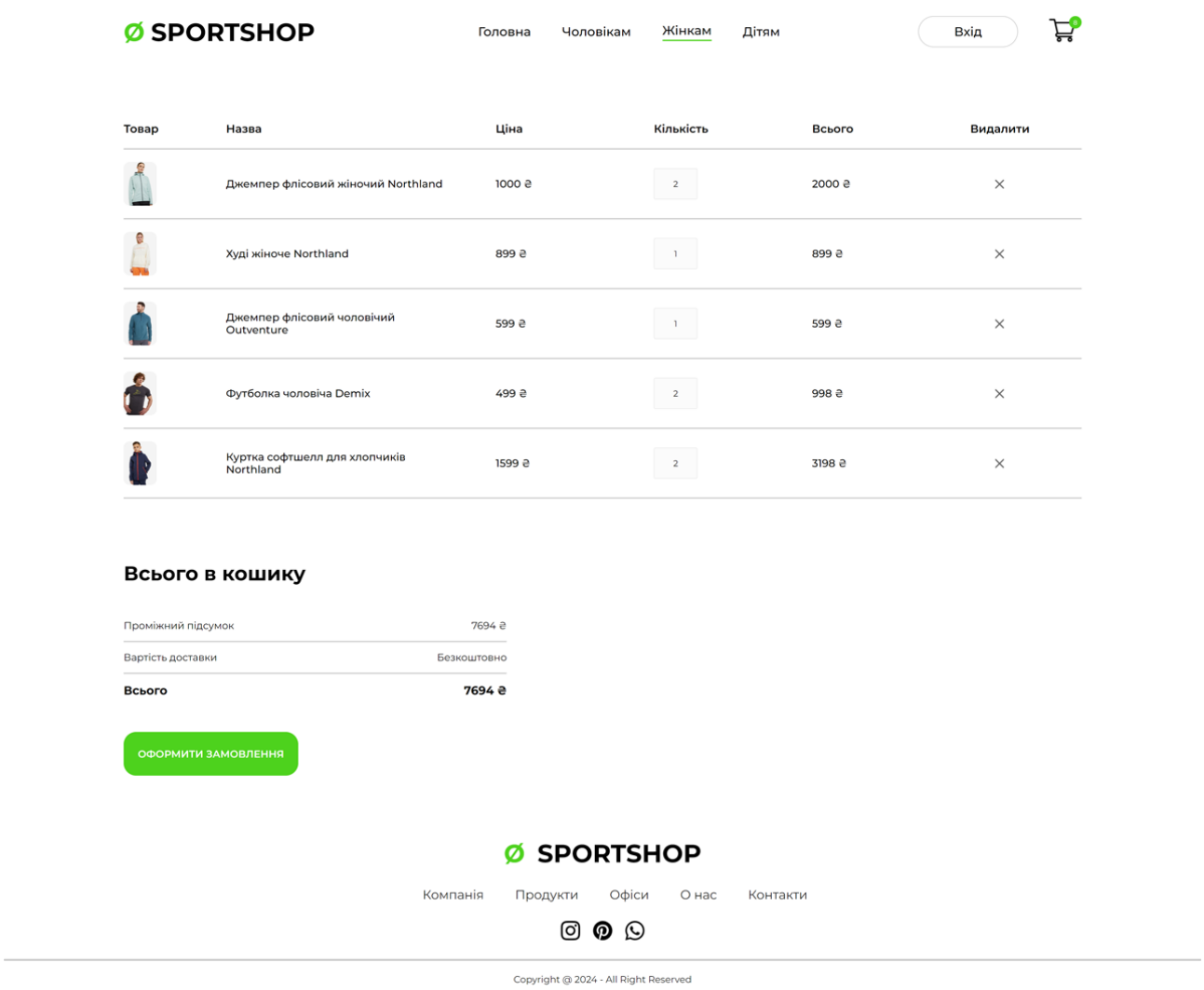

Рисунок 3.28 – Вигляд сторінки кошику товарів

Управління кошиком та товарами в ньому відбувається завдяки компоненту ShopContext. Контекст (Context) у React є інструментом, який дозволяє ділитися даними між компонентами без необхідності передавати ці дані через props на кожному рівні компонента. Це особливо корисно для управління глобальним станом, наприклад, станом кошика покупок, що забезпечує зручний доступ до даних та функцій для всіх компонентів у застосунку [17].

ShopContext створює та надає цей контекст, дозволяючи іншим компонентам отримувати доступ до стану кошика та функцій для управління ним.

Компонент ShopContextProvider ініціалізує стан кошика за допомогою функції, яка встановлює початкову кількість кожного товару на 0.Він також

містить функції для додавання товарів до кошика, видалення товарів з кошика, підрахунку загальної кількості товарів та їхньої загальної вартості. Завдяки цьому всі компоненти, які споживають цей контекст, мають доступ до необхідних даних і методів для управління кошиком (рис. 3.29).

```jsx
import React, { createContext, useState } from 'react';
import all_product from '../Components/Assets/all_product';

export const ShopContext = createContext(null);

const getDefaultCart = () => {
    let cart = {};
    for (let index = 0; index < all_product.length+1; index++) {
        cart[index] = 0;
    }
    return cart;
};

const ShopContextProvider = (props) => {
    const [cartItems, setCartItems] = useState(getDefaultCart());

    const addToCart = (itemId) => {
        setCartItems((prev)=>({...prev,[itemId]:prev[itemId]+1}))
        console.log(cartItems);
    }
    const removeFromCart = (itemId) => {
        setCartItems((prev)=>({...prev,[itemId]:prev[itemId]-1}))
    }

    const getTotalCartAmount = () => {
        let totalAmount = 0;
        for(const item in cartItems)
        {
            if (cartItems[item]>0) {
                let itemInfo = all_product.find((product)=>product.id===Number
                (item))
                totalAmount += itemInfo.new_price * cartItems[item];
            }
        }
        return totalAmount
    }
    const getTotalCartItems = () => {
        let totalItem = 0;
        for(const item in cartItems)
        {
            if (cartItems[item]>0) {
                totalItem += cartItems[item];
            }
        }
        return totalItem
    }

    const contextValue = {getTotalCartItems, getTotalCartAmount, all_product,
    cartItems, addToCart, removeFromCart};
    return (
        <ShopContext.Provider value={contextValue}>
            {props.children}
        </ShopContext.Provider>
    );
};

export default ShopContextProvider;
```

Рисунок 3.29 – Код компоненту управління кошиком ShopContext

CartItems використовує дані та функції з ShopContext для відображення товарів у кошику та взаємодії з ними. Спочатку він отримує дані з контексту, такі як усі доступні товари, стан кошика та функції для видалення товарів з кошика. Потім він перебирає всі товари і відображає лише ті, що є у кошику (кількість більше нуля). Для кожного товару відображається його зображення, назва, ціна, кількість у кошику та загальна вартість для цього товару. Також відображається іконка для видалення товару з кошика (рис. 3.30).

```jsx
import React, { useContext } from 'react'
import './CartItems.css'
import { ShopContext } from '../../Context/ShopContext'
import remove_icon from '../Assets/cart_cross_icon.png'

export const CartItems = () => {
    const {getTotalCartAmount, all_product, cartItems, removeFromCart} =
    useContext(ShopContext);
  return (
    <div className='cartitems'>
        <div className="cartitem-format-main">
            <p>Товар</p>
            <p>Назва</p>
            <p>Ціна</p>
            <p>Кількість</p>
            <p>Всього</p>
            <p>Видалити</p>
        </div>
        <hr />
        {all_product.map((e)=> {
            if (cartItems[e.id]>0){
                return <div>
                <div className="cartitems-format cartitem-format-main">
                    <img src={e.image} alt="" className='carticon-product-icon'/
                    >
                    <p>{e.name}</p>
                    <p>{e.new_price} ₴</p>
                    <button className='cartitems-quantity'>{cartItems[e.id]}</
                    button>
                    <p>{e.new_price*cartItems[e.id]} ₴</p>
                    <img className='cartitems-remove-icon' src={remove_icon}
                    onClick={()=>{removeFromCart(e.id)}} alt="" />
                </div>
                <hr />
                </div>
            }
            return null;
        })}
        <div className="cartitems-down">
            <div className="cartitems-total">
                <h1>Всього в кошику</h1>
                <div>
                    <div className='cartitems-total-item'>
                        <p>Проміжний підсумок</p>
                        <p>{getTotalCartAmount()} ₴</p>
                    </div>
                    <hr />
                    <div className="cartitems-total-item">
                        <p>Вартість доставки</p>
                        <p>Безкоштовно</p>
                    </div>
                    <hr />
                    <div className="cartitems-total-item">
                        <h3>Всього</h3>
                        <h3>{getTotalCartAmount()} ₴</h3>
                    </div>
                </div>
                <button>ОФОРМИТИ ЗАМОВЛЕННЯ</button>
            </div>
        </div>
    </div>
  )
}

export default CartItems;
```

Рисунок 3.30 – Код компоненту CartItems

Таким чином, ці два компоненти забезпечують повний функціонал для роботи з кошиком у інтернет-магазині, дозволяючи користувачам переглядати, додавати та видаляти товари, а також бачити загальну вартість їхнього замовлення і оформлювати його. Управління кошиком та товарами відбувається завдяки компоненту ShopContext, який надає всі необхідні дані та функції для цих дій.

3.3 Висновки до розділу 3

Під час розробки інтернет-магазину з використанням фреймворку React, активно використовувалися HTML та CSS для створення візуальної частини додатку.

Проте, основним інструментом, що визначив архітектуру та функціональність проєкту, став саме React. Використання цього фреймворку надало можливість ефективно розбити структуру вебзастосунку на окремі самостійні компоненти. Це спростило як процес розробки, так і розуміння коду.

Крім того, використання React Router DOM надало можливість реалізувати навігацію в інтернет-магазині. Завдяки цьому інструменту вдалося забезпечити зручний та інтуїтивно зрозумілий спосіб переміщення між різними сторінками та розділами магазину, без їх перезавантаження.

У підсумку, використання фреймворку React значно полегшило процес розробки вебзастосунку, дозволяючи розбити його на компоненти та забезпечивши зручну навігацію. Це сприяло створенню ефективного та зрозумілого вебзастосунку як для користувачів, так і для розробників.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи, був розроблений інтернет-магазин спортивних товарів з використанням фреймворку React. Проэкт влючав в себе реалізацію функціоналу таких елементів, як навігаційне меню (NavBar), підвал (Footer), головна сторінка (Shop), категорії товару (ShopCategory), огляд окремого товару (Product/DisplayProdact), реєстрація (LoginSign), кошик (Cart/CartItems) товарів та окремих компонентів.

У даній роботі використовується популярний фреймворк React, та його основні особливості.

Компонентний підхід до розробки вебзастосунків, який дозволив структурувати проєкт та ефективніше орієнтуватися та редагувати код.

Віртуальний DOM (Document Object Model), дозволив оптимізувати процес оновлення вебсторінки, замість безпосередньої маніпуляції реальним DOM.

Інструмент для маршрутизації React Router, бібліотека яка дозволила ефективно керувати маршрутизацією між компонентами вебзастосунку.

У порівнянні з методами розробки, що використовувались раніше, React значно підвищив продуктивність роботи над проєктом. Раніше розробка вебзастосунків без використання таких фреймворків вимагала більше часу та зусиль на підтримку та оновлення коду, особливо при роботі з великими і складними інтерфейсами. Завдяки React, процес розробки став більш структурованим та зрозумілим, а також дозволив швидше реагувати на зміни вимог та виправляти помилки.

В підсумку виконаної робити, React зарекомендував себе як одна з най зручніших та ефективних технологій у розробці вебзастосунків, завдяки своїм особливостям, підтримці розробниками та великої спільноти.

ПЕРЕЛІК ПОСИЛАНЬ

1. Levinson D., Belton T. Build Your First Web App: Learn to Build Your First Web Applications From Scratch. Gmc Group, 2018. 256 p.

2. Weiner A. Essential parts of a website. *Wix*. URL : https://www.wix.com/blog/parts-of-a-website (дата звернення: 01.04.2024).

3. Tedesco C. Responsive Header Code. *CodePen*. URL : https://codepen.io/ctedesco02/pen/Mjxwzw (дата звернення: 01.04.2024).

4. Grozdic I. Home Responsive Bootstrap. *CodePen*. URL : https://codepen.io/ig_design/details/omQXoQ (дата звернення: 01.04.2024).

5. Ng O. CSS Grid: Newspaper Layout. *CodePen*. URL : https://codepen.io/oliviale/pen/BaoXOOP (дата звернення: 05.04.2024).

6. Azouaoui M. Pro Sidebar template. *CodePen*. URL : https://codepen.io/azouaoui-med/pen/wpBadb (дата звернення: 05.04.2024).

7. CodingTuting. Responsive Footer Design. *CodePen*. URL : https://codepen.io/codingtuting/pen/NWKXeKL (дата звернення: 05.04.2024).

8. Jay. Bootstrap 3 Contact form with Validation. *CodePen*. URL : https://codepen.io/jaycbrf/pen/NWYjYr (дата звернення: 08.04.2024).

9. Taylor C. The History of ReactJS. *HRTechcube*. URL : https://hrtechcube.com/the-history-of-reactjs (дата звернення: 08.04.2024).

10. Anderson L. The History of React.js: A Story of Innovation and Community. *Linkedin*. URL : https://www.linkedin.com/pulse/history-reactjs-story-innovation-community-l-anderson (дата звернення: 10.04.2024).

11. Ushna I. What are the features of React? *Educative*. URL : https://www.educative.io/answers/what-are-the-features-of-react (дата звернення: 10.04.2024).

12. Harikatalari08. What are the features of ReactJS? *GeeksForGeeks*. URL : https://www.geeksforgeeks.org/what-are-the-features-of-reactjs (дата звернення: 15.04.2024).

13. ReactJS – Features. *Tutorialspoint*. URL : https://www.tutorialspoint.com/reactjs/reactjs_features.htm (дата звернення: 15.04.2024).

14. Most popular technologies (Web frameworks and technologies). *Stackoverflow*. URL : https://survey.stackoverflow.co/2023/#section-most-popular-technologies-web-frameworks-and-technologies (дата звернення: 15.04.2024).

15. Каграманова Ю. Діаграма варіантів використання. *DOU*. URL : https://dou.ua/forums/topic/40575 (дата звернення: 19.04.2024).

16. Мильцев О. М. Лабораторна робота №3. Визначення переліку акторів та прецедентів. *Moodle ZNU*. URL : https://moodle.znu.edu.ua/mod/assign/view.php?id=273857 (дата звернення: 19.04.2024).

17. Boateng D. How to Use the React Context API in Your Projects. *freeCodeCamp*. URL : https://www.freecodecamp.org/news/context-api-in-react (дата звернення: 19.04.2024).