МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інтернет-магазин з реалізації спортивного взуття»
здобувача групи ІН-93 Дрижова Ростислава Євгеновича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Ростислав ДРИЖОВ
(підпис)

Керівник,
асистентка кафедри комп'ютерних наук,
кандидат фізико-математичних наук Ольга ШУТИЛЄВА
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 – Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-93 Дрижова Ростислава Євгеновича

1. Тема роботи: «Інтернет-магазин з реалізації спортивного взуття»
затверджую наказом по СумДУ від «01» червня 2023 р. № 0475-VI
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для розробки. 3) Розробка інтернет-магазину. 4) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 2023 р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Аналіз проблеми предметної області, постановка й формування завдань дослідження.		
2	Огляд технологій, що використовуються для розробки.		
3	Розробка інтернет-магазину для реалізації спортивного взуття.		
4	Аналіз результатів.		
5	Оформлення пояснювальної записки до кваліфікаційної роботи		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 76 стор., 36 рис., 7 додатків, 11 використаних джерел.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки в сучасному світі електронна комерція набуває все більшої популярності, користувачі мають зростаючі очікування щодо зручності та якості веб-ресурсів, технологічний прогрес швидко розвивається, а інтерес до моди та взуття зростає.

Об’єкт дослідження – процес розробки та сучасні підходи до розробки веб-ресурсів

Мета роботи – розробка інтернет магазину з реалізації спортивного взуття.

Методи дослідження – методи спостереження, порівняння, аналізу.

Результати – розроблено веб-сайт для реалізації спортивного взуття, з дотримуванням сучасних тенденцій розробки веб-ресурсів. Сайт створений на базі мови Python з використання фреймворку Django.

ВЕБ-САЙТ, WEB, REACT, JAVASCRIPT, МОСКАРІ, КОМПОНЕНТИ, SCSS, HTML

ЗМІСТ

ВСТУП

Актуальність. Сучасний розвиток технологій та зростання популярності онлайн-торгівлі створюють актуальні можливості для розвитку бізнесу. Ринок спортивного взуття вже зацікавлений у використанні інтернет-магазинів для забезпечення широкого асортименту, зручності покупок та доступних цін.

Предметом дослідження є ринок спортивного взуття та можливості його реалізації через інтернет-магазин.

Об'єктом дослідження є аналіз ринкових тенденцій, вивчення сегменту спортивного взуття, аналіз існуючих інтернет-магазинів спортивного взуття. Дослідження також передбачає розробку ефективного інтернет-магазину, який задовольнятиме потреби клієнтів і сприятиме успішному функціонуванню бізнесу.

Гіпотезою дослідження є те, що розроблений інтернет-магазин спортивного взуття буде задовольняти потреби клієнтів, сприяти успішному веденню бізнесу та забезпечувати його прибутковість.

Новизна дослідження полягає у вивченні ринку спортивного взуття в онлайн-середовищі, розробці ефективного інтернет-магазину, а також в аналізі його ефективності та перспектив розвитку. Результати дослідження нададуть цінну інформацію та рекомендації для бізнесу у цьому конкурентному сегменті ринку.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибір методу розв'язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

Актуальність проблеми створення та розвитку інтернет-магазинів з реалізації спортивного взуття базується на кількох факторах [1].

По-перше, спортивне взуття є популярною категорією товарів, яка має великий попит серед спортсменів, фітнес-ентузіастів та широкої публіки, яка прагне активного та здорового способу життя. Зростаюча свідомість про користь фізичної активності спричиняє збільшення популярності спортивного взуття та його покупок.

По-друге, інтернет-магазини набувають все більшої популярності серед споживачів. У підтвердження цього, ми можемо розглянути офіційну статистику, щодо стану індустрії моди електронної комерції (рис. 1.1). Зручність онлайн-шопінгу, широкий вибір товарів, зручна доставка та можливість порівняти ціни товарів роблять їх привабливими для споживачів. Інтернет-магазини забезпечують доступ до спортивного взуття з будь-якого місця та у зручний для користувача час.

По-третє, розвиток інформаційних технологій та електронної комерції створює сприятливе середовище для створення та функціонування інтернет-магазинів. Застосування передових технологій в дизайні, веб-розробці дозволяють створити зручний та ефективний інтернет-магазин для реалізації спортивного взуття.

Сегмент взуття, до якого також входить спортивне взуття (рис. 1.2), є важливою складовою актуальної проблеми створення та розвитку інтернет-магазинів з реалізації спортивного взуття.

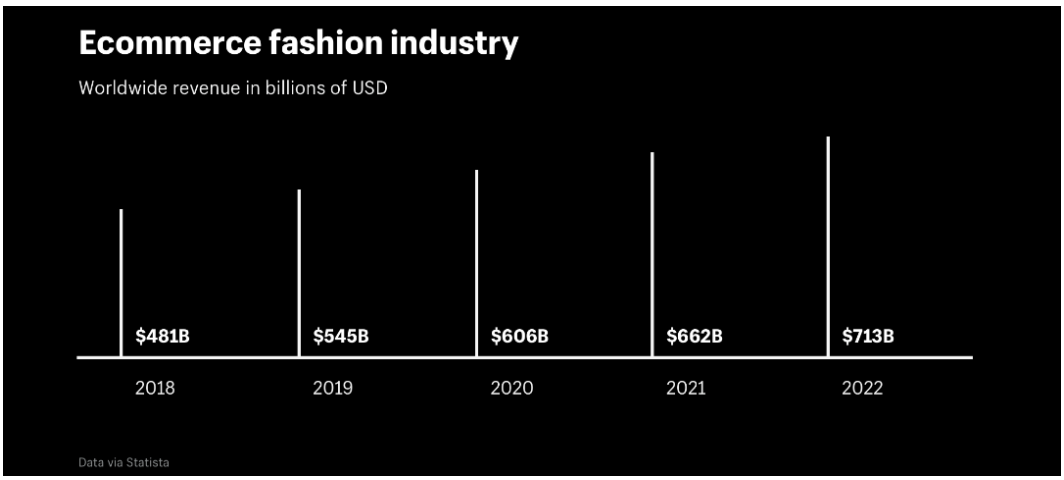

Рисунок 1.1 – Стан індустрії продажу електронної моди

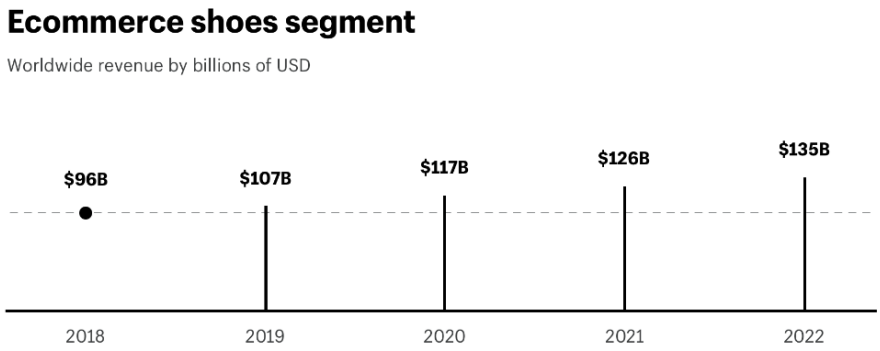

Рисунок 1.2 – Сегмент взуття

Таким чином, створення інтернет-магазину з реалізації спортивного взуття є актуальною темою, оскільки задовольняє зростаючий попит на спортивні товари, відповідає потребам споживачів у зручному та доступному шопінгу, а також використовує потужні можливості сучасних інформаційних технологій.

1.2 Аналіз аналогічних проєктів

При визначенні основних вимог до інтернет-магазину спортивного взуття, важливими факторами є інтуїтивно зрозумілий інтерфейс, безпечність, наявність основних функцій для зручного вибору продукту та оформлення замовлення. У наступному розділі будуть розглянуті та проаналізовані деякі

інтернет-магазини, які можуть бути використані для купівлі спортивного взуття.

1.1.1 Інтернет магазин Bosscross

Bosscross – це український онлайн-сервір, який дає змогу купувати спортивне взуття на ваш вибір. Давайте для початку розглянемо інтерфейс та дизайн сайту.

Головна сторінка має не дуже привабливий та сучасний вигляд. Підбір кольорів та їх поєднання не вдалий. Дизайн структури сайту хаотичний та не відповідає сучаним стандартам мінімалістичного оформлення (рис. 1.3).

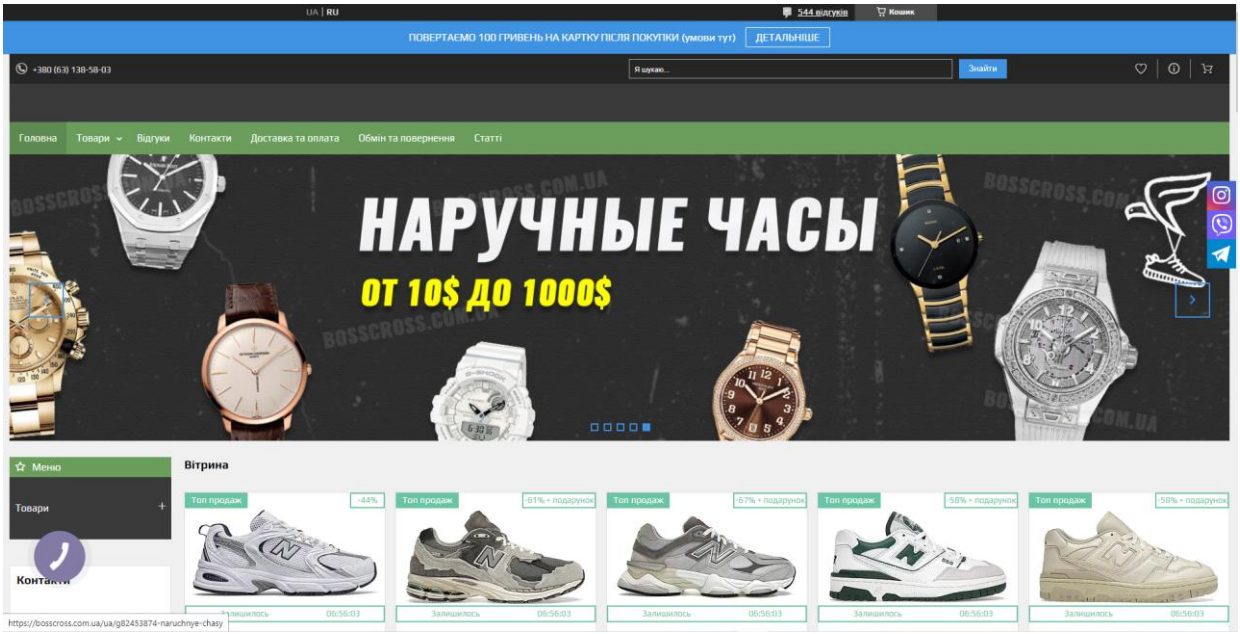

Рисунок 1.3 – Дизайн головної сторінки магазину Bosscross

При перегляді каталогу товарів можна відразу перейти до оформлення товару через кнопку “Купити” (рис. 1.4). У системі інтернет-магазину відсутня можливість додавання товарів до списку закладок або обраного, яка дозволяє користувачам зберігати товари, які їм сподобалися, для подальшого перегляду та розгляду, навіть якщо вони не планують їх купувати негайно.

Також, однією з недоліків є швидкодія сайту, оскільки спостерігається тривале завантаження певних візуальних елементів та затримки під час

переходу між сторінками, що не забезпечує миттєвої відгуку та позитивного враження від користування.

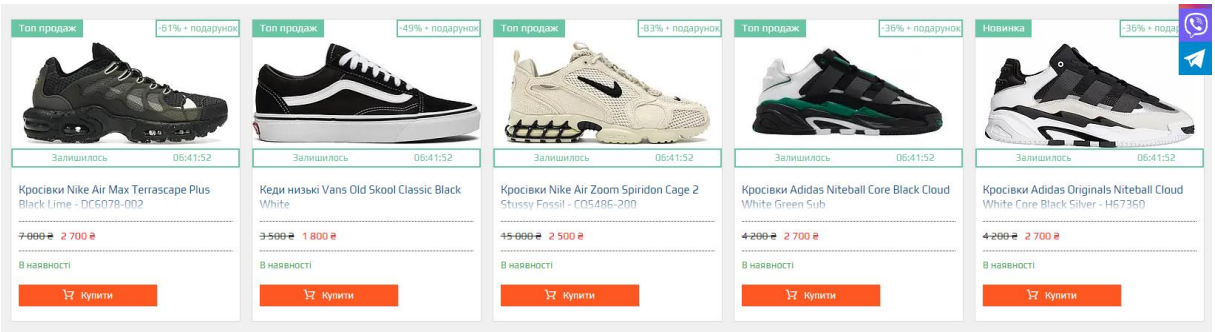

Рисунок 1.4 – Каталог товарів магазину Bosscross

Коши з товарами має наступний вигляд (рис. 1.5):

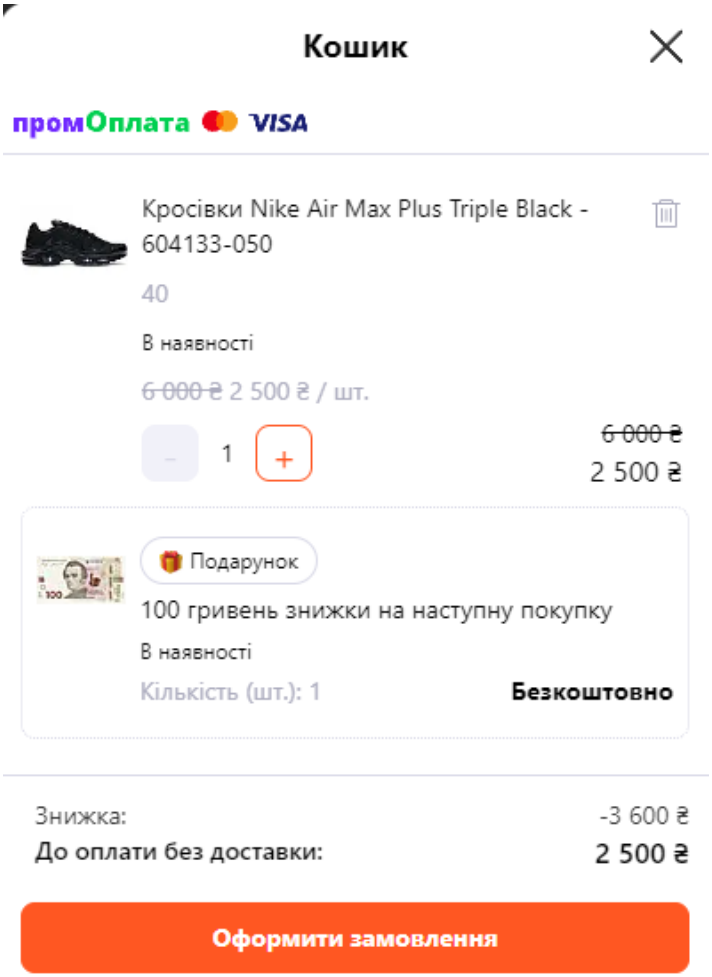

Рисунок 1.5 – Кошик магазину Bosscross

Можемо також зазначити, що інтерфейс відповідає принципам зрозумілості та чіткості, оскільки він відображає інформацію про знижку на товар. Проте, однією з недоліків є відсутність вказівки відсотка податку на товар, що може створювати незрозумілість щодо загальної ціни та фінансових аспектів покупки.

Слід відзначити, що наявність поле пошуку товарів є значним плюсом, оскільки сприяє швидкому знаходженню необхідної пари взуття. Крім того, на сайті відображається увесь асортимент товарів, але присутня лише текстова інформація, що не надає візуального контексту або ескізу взуття, що може бути недоліком (рис. 1.6).

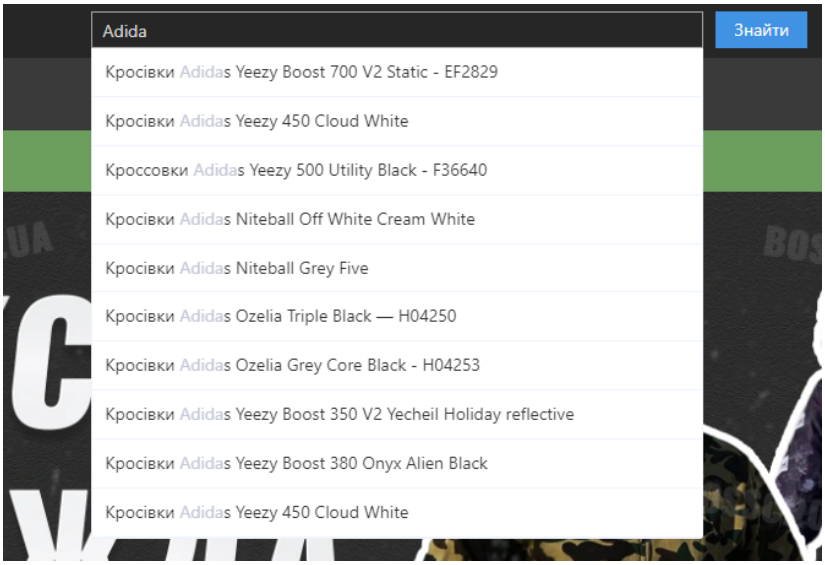

Рисунок 1.6 – Поле пошуку товару магазину Bosscross

Більшість взаємодій з товаром вимагають підвантаження, що призводить до затримок у часі, що також може впливати на користувацький досвід.

Узагальнюючи, варто відзначити, що цей веб-сайт має необхідний функціонал для здійснення покупок, але наявність застарілого дизайну, низької швидкодії та відсутньої можливості додавання товарів до особистих закладок негативно впливають на загальне враження користувачів.

1.1.2 Інтернет-магазин Topitop

Сторінка має привабливий чистий дизайн. Тут можна переглянути товари, виконати пошук за допомогою відповідного поля, залишити відгук та передивитися історію вже переглянутих товарів (рис.1.7).

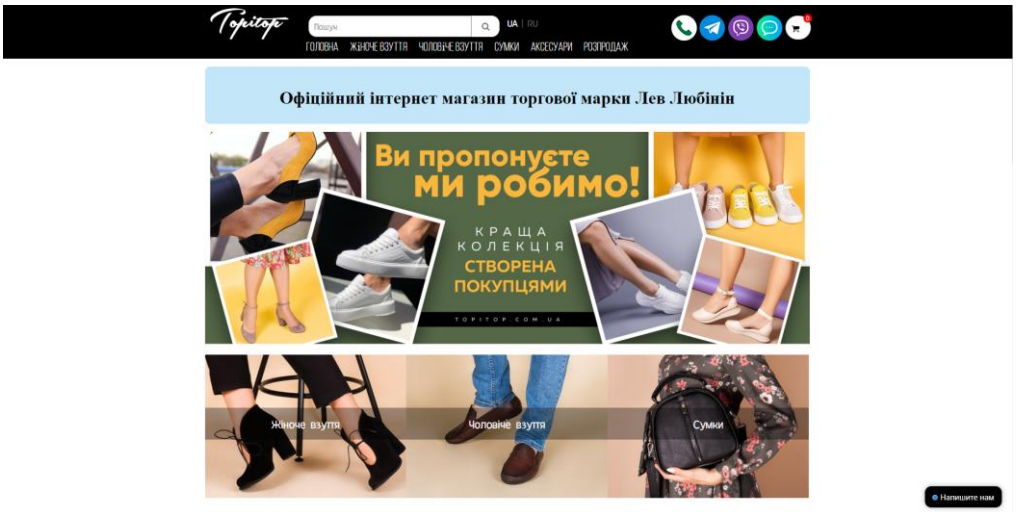

Рисунок 1.7 – Головна сторінка магазину Topitop

Під час перегляду товарів у каталозі, користувачам надається зручна можливість додавати товари до кошика та оформляти замовлення в один клік, що дозволяє спростити процес покупки та забезпечує зручність користувачів.

Рисунок 1.8 – Вигляд товару в каталозі магазину Topitop

При оформленні замовлення є можливість вписати свій номер та видно загальну суму товарів (Рис. 1.9).

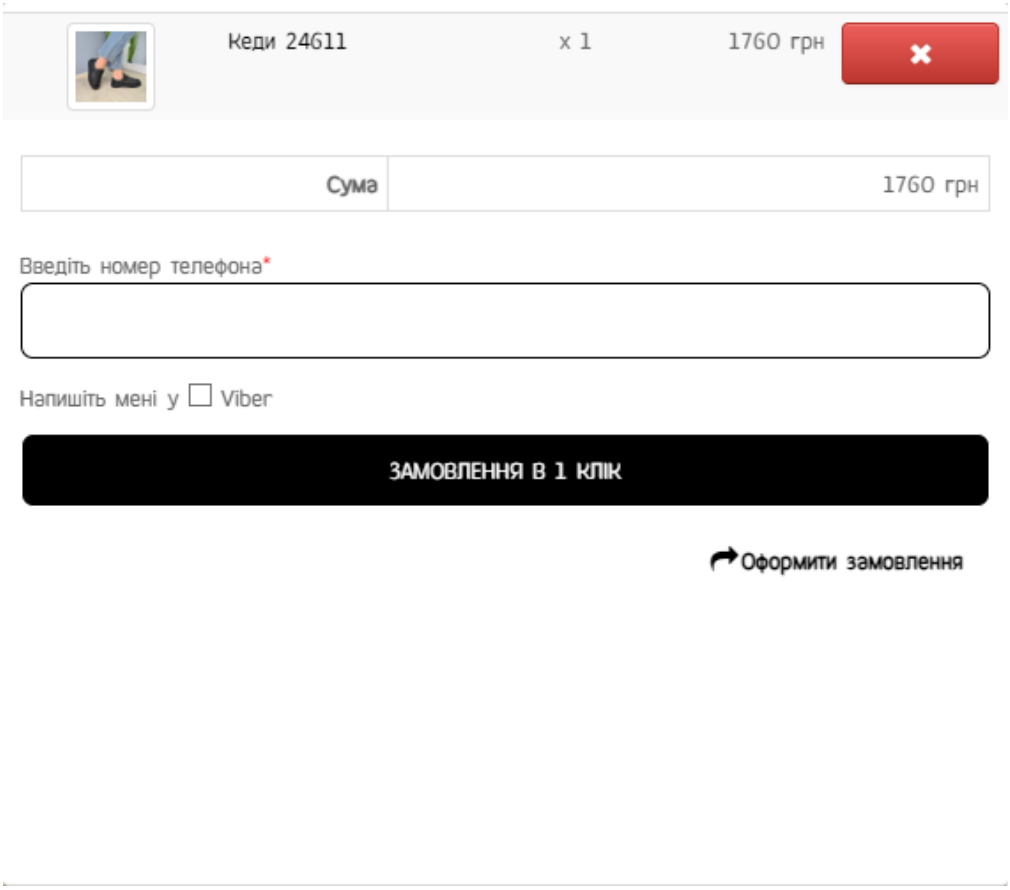

Рисунок 1.9 – Оформлення замовлення магазину Topitop

Підбиваючи підсумки, можна визначити, що даний веб-сайт відзначається привабливим дизайном та функціоналом, що робить його привабливим для покупців. Він має широкий спектр необхідних функцій, включаючи можливість перегляду історії переглянутих товарів, залишення відгуків та інше. Явних недоліків не виявлено, проте можна помітити відсутність можливості додавання товарів до закладок, це можна вдосконалити для поліпшення зручності користувачів.

1.1.3 Інтернет магазин Krosonavtik

Домашня сторінка даного магазину має не сильно привабливий інтерфейс але, за плюс можу відмітити правильну кольорову палітру та комбінацію кольорів. Присутня кнопка “Кошик” та поле пошуку (Рис. 1.10).

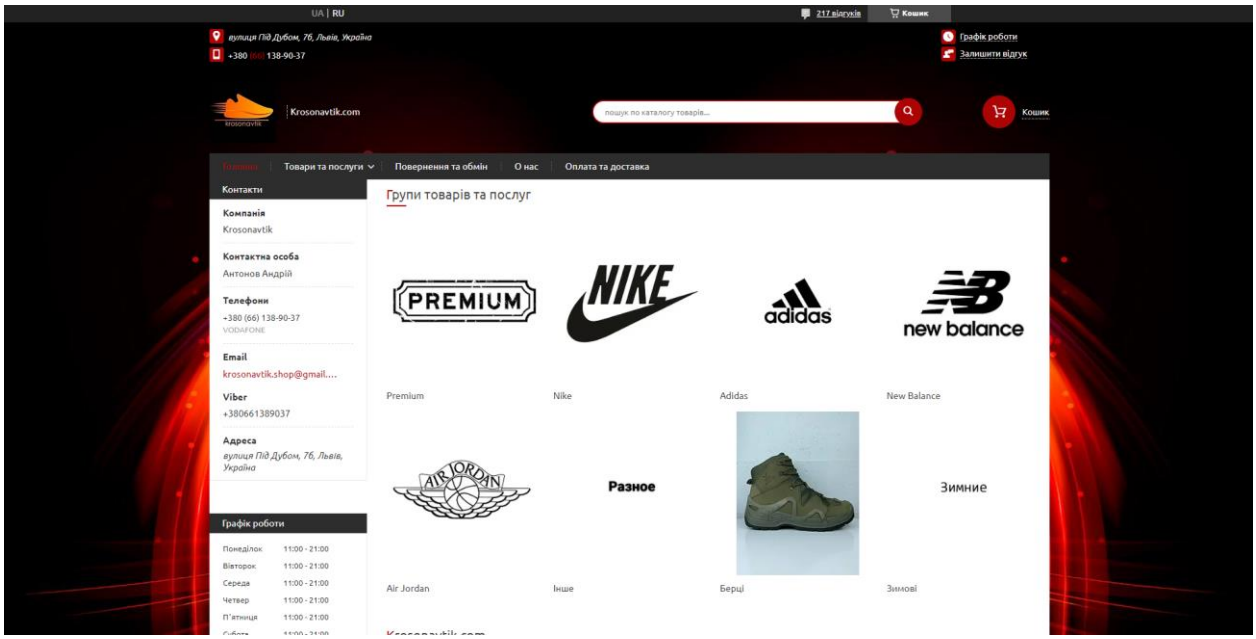

Рисунок 1.10 – Домашня сторінка Krosonavtik

Каталог з товарами, де можна побачити наявність взуття та швидко додати його до кошику.

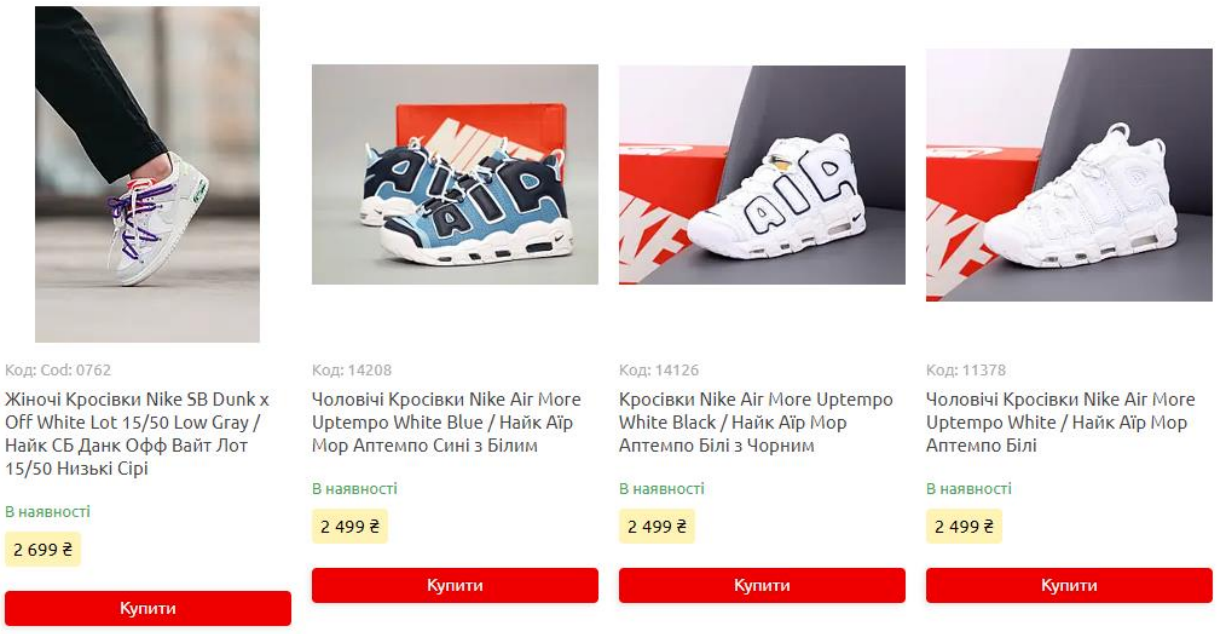

Рисунок 1.11 – Каталог товарів магазину Krosonavtik

Кошик даного магазину схожий на перший веб – сайт та дає змогу змінити кількість або навіть видалити товар. Окрім загальної суми товару, дані відсутні (Рис. 1.12).

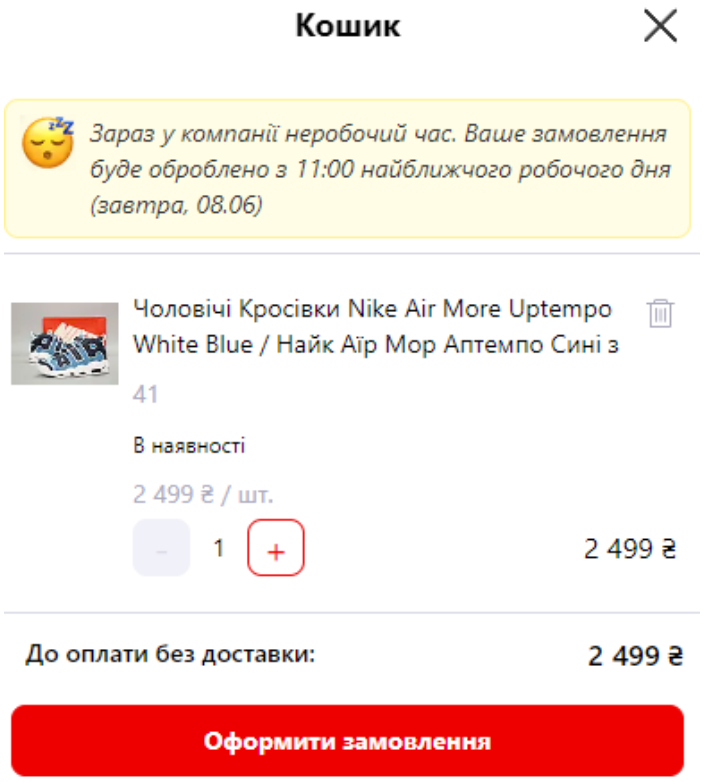

Рисунок 1.12 – Кошик магазину Krosonavtik

Даний інтернет-магазин спортивного взуття пропонує більшість необхідних функцій для зручного користування. Проте, варто зазначити, що він має певні недоліки, пов'язані з дизайном та відсутністю певного функціоналу. З точки зору дизайну, магазин може виглядати застарілим або не зовсім привабливим для покупців. Також важливо відмітити, що відсутність певних функцій, наприклад, можливості додавання товарів до особистих закладок, може обмежити зручність та персоналізацію користувацького досвіду. Для поліпшення магазину рекомендується враховувати ці недоліки та працювати над їх вирішенням для забезпечення максимального задоволення покупців та підвищення конкурентоспроможності.

Таблиця 1.1 – Порівняння моделей для реалізації аналізу

Характеристика \ Магазин	Bosscross	Topitop	Krosonavtik	Sneakers_Drzhv
Сучасний дизайн	-	+	-	+
Зручний інтерфейс	-	+	+	+
Функціональність	-	+	+	+
Навігація	+	+	+	+
Закладки	-	-	-	+
Історія покупки товарів	-	+	-	+
Зручний пошук	-	+	-	+
Замовлення товарів без обо’язкової реєстрації	+	+	+	+
Швидкодійність	-	+	+	+

1.3 Постановка задачі

Після аналізу аналогічних відомих рішень і визначення необхідних вимог до сайту, ми можемо приступити до складання списку задач, які потребують реалізації. Кінцевий результат нашого онлайн-магазину спортивного взуття повинен включати наступні ключові сторінки та функціонал:

- Головна сторінка – перша сторінка, яку бачать відвідувачі сайту. Вона повинна відображати доступний товар, а також надавати швидкий доступ до взаємодії з ним.
- Сторінка “Закладки” з взуттям повинна містити весь доданий товар та відображати його стан.
- Кошик – має відображати усі товари, додані до нього, можливість видаляти конкретний продукт, а також відображати кінцеву ціну, та процент від податку.

- Сторінка “Мої покупки” має містити та відображати замовлення, що були оформленні.
- Функція пошуку та коректного показу товару з візуальною частиною.
- Сучасний мінімалістичний дизайн з привабливою палітрою кольорів, яка буде забезпечувати максимальне просте використовування та візуальне задоволення для покупця.
- Красиві та плавні анімації для більш приємного враження від користування.

Слід не забути про те, що сайт повинен швидко та ефективно оброблювати усі дії користувачів, а також відповідати усім критеріям загальновідомих правил безпеки в інтернеті.

2. ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Вибір мов програмування

Для виконання поставлених задач було обрано наступні технології та мови програмування для візуальної частини: HTML, SCSS, JavaScript, з іншого боку серверна частина була підключена, за допомогою онлайн-сервісу MockAPI.

2.1.1 HTML – Hyper Text Markup Language

Представляє собою стандартну мову розмітки, яку використовують для створення веб-сторінок та організації їх візуальної структури. Ця мова використовується для визначення семантики, структури та вигляду веб-документів. За допомогою HTML розробники можуть створювати різноманітні елементи веб-сторінок, такі як заголовки, абзаци, зображення, таблиці, посилання та інші [6].

HTML використовується для створення інтерактивних веб-сторінок, додавання контенту, форм, мультимедійних елементів та стилізації їх за допомогою CSS. Вона є основою для розробки веб-додатків та дозволяє створювати зручний інтерфейс для користувачів.

Головна мета використання HTML полягає в створенні структурованих документів, які можуть бути читабельними для людей та зрозумілими для веб-переглядачів. Вона дозволяє розміщувати та організовувати вміст на сторінці, створювати зв'язки між різними сторінками, форматувати текст, вставляти зображення та відео, створювати таблиці, списки та інші елементи.

HTML є однією з основних мов програмування для веб-розробки і часто використовується разом з CSS та JavaScript для створення сучасних, динамічних та інтерактивних веб-сторінок та додатків. Ця мова дозволяє розробникам створювати ефективний та привабливий веб-контент для широкого кола користувачів.

2.1.2 SCSS (Sassy CSS)

SCSS є мовою розширення для CSS (каскадних таблиць стилів), яка дає додаткові можливості та функціонал для зручної та ефективної розробки стилів веб-сторінок. SCSS є синтаксичним варіантом CSS і використовується для покращення організації та керування стилями в проєктах [7][9].

Головна перевага SCSS полягає в його розширеному функціоналі, який включає змінні, вкладені селектори, міксіни, наслідування та багато іншого. Завдяки змінним, розробники можуть оголошувати значення, які можуть бути використані повторно в стилях, що полегшує їх зміну та підтримку. Вкладені селектори дозволяють групувати стилі для елементів, що знаходяться всередині інших елементів, спрощуючи їх структуру.

Міксіни є важливою функцією SCSS, яка дозволяє визначати та викликати набори стилів для багаторазового використання. Це полегшує створення стандартних стилів та забезпечує більшу гнучкість у розробці. Крім того, SCSS підтримує наслідування, що дозволяє одним стилем успадковувати властивості іншого стилю, що спрощує створення варіацій стилів.

SCSS зберігається у файлах з розширенням .scss і підтримується компіляторами до звичайного CSS, що дозволяє використовувати його безпосередньо на веб-сайті. Завдяки своїй зрозумілій синтаксису, SCSS зручно використовувати та підтримує багато інструментів розробки.

SCSS є потужним інструментом для розробки стилів веб-сторінок, який допомагає зберегти час та зусилля, забезпечуючи більшу гнучкість і швидкість в розробці CSS. Ця мова є популярним вибором серед веб-розробників і дозволяє створювати привабливі та доброорганізовані стилі для веб-додатків.

2.1.3 JavaScript (JS)

Це мова програмування, яка використовується для створення динамічних та інтерактивних веб-сторінок. Вона є високорівневою та інтерпретованою, дозволяючи розробникам впроваджувати різноманітні функціональні можливості на клієнтській стороні [8][11]. JavaScript взаємодіє

з користувачем, маніпулює вмістом веб-сторінок та реагує на події в реальному часі.

Мова JavaScript є однією з найбільш поширених у сфері веб-розробки та має високу підтримку у всіх сучасних веб-браузерах. Вона дозволяє змінювати елементи HTML та CSS, валідувати дані, створювати анімацію, взаємодіяти з сервером через AJAX-запити та багато іншого.

JavaScript має широкий екосистему, яка включає багато бібліотек і фреймворків, таких як React, Angular та Vue.js, які полегшують розробку складних веб-додатків. Вона також використовується для створення мобільних додатків, настільних програм та серверної розробки за допомогою платформи Node.js.

JavaScript має простий та доступний синтаксис, що дозволяє швидко навчитися мові навіть початківцям. Водночас вона надає розробникам потужні інструменти для створення веб-додатків різної складності. JavaScript є незамінним інструментом для веб-розробників, який допомагає створювати динамічні та інтерактивні веб-додатки.

2.1.4 Онлайн-сервіс MockAPI

MockAPI – це онлайн-сервіс, який надає можливість створення та емуляції неіснуючого API для розробки та тестування програмного забезпечення. Він дозволяє розробникам створювати та налаштовувати власні користувацькі ендпоінти, відповіді та дані для тестування різних сценаріїв без прив’язки до реального сервера або бази даних [5].

MockAPI дозволяє створювати імітацію реальних API з використанням різних типів запитів (GET, POST, PUT, DELETE) та задавати необхідні параметри для отримання певної відповіді. Ви можете визначати структуру даних, поля, типи даних та поведінку API за допомогою простого інтерфейсу.

Основні переваги використання MockAPI включають можливість швидко створювати прототипи або мокапи додатків, тестувати різні сценарії без реального сервера, спрощувати співпрацю між фронтенд- та бекенд-

розробниками, а також забезпечувати роботу над проєктом без доступу до реальних даних.

MockAPI є потужним інструментом для розробників, який допомагає забезпечити ефективний процес розробки та тестування програмного забезпечення, а також підвищує швидкість та гнучкість у роботі з API.

2.2 Вибір фреймворків

Фреймворк – це комплексний набір засобів, бібліотек та правил, які надають розробникам програмного забезпечення загальну структуру та шаблони для побудови програм або додатків. Він надає розробникам заздалегідь визначену архітектуру та компоненти, які спрощують розробку та розширення програмного забезпечення [3].

Особбливості фреймворків включають в себе:

- Інверсія керування – надають контроль над виконанням програми шляхом інверсії керування. Замість того, щоб розробникам визначати послідовність дій та потрібні компоненти, фреймворки дозволяють розробникам визначати свої компоненти та дозволяють фреймворку керувати потоком програми.
- Фреймворки надають стандартні шаблони проєктування та архітектуру, які полегшують розробку та підтримку програмного забезпечення. Це забезпечує однорідність між проєктами та дозволяє розробникам швидко орієнтуватися в коді.
- Готові компоненти та функціональність, що допомагають в розробці програмного забезпечення. Це можуть бути стандартні інтерфейси користувача, робота з базами даних, мережева взаємодія та інші загальноприйняті функції.
- Фреймворки надають механізми для управління життєвим циклом програмного забезпечення, такі як створення, ініціалізація, розгортання та

видалення. Це спрощує розробку та підтримку програм та забезпечує їхню стабільність.

Загалом, фреймворки надають розробникам засоби для швидкої та ефективної розробки програмного забезпечення, дозволяючи їм фокусуватися на бізнес-логіці та функціональності програми, а не на низькорівневих деталях реалізації.

2.2.1 React

React – це відкрита JavaScript бібліотека для розробки користувацького інтерфейсу (UI). Він розроблений компанією Facebook і використовується для побудови веб-додатків з високою реактивністю і швидкістю [4][11].

Основною метою React є полегшення розробки UI, розбиваючи його на компоненти, які можна повторно використовувати. Замість традиційного підходу до розробки веб-сторінок, де HTML-розмітка і JavaScript-логіка змішуються, React використовує спеціальний синтаксис, відомий як JSX, що дозволяє описати структуру UI у вигляді компонентів.

Одна з головних концепцій у React – це "унідирекційний потік даних". Це означає, що дані в компоненті можуть бути змінені лише через зміну їхнього стану. Компонент React може мати внутрішній стан, який може бути змінений за допомогою методу setState(). При зміні стану React автоматично оновлює відповідну частину UI, що дозволяє побудувати реактивний інтерфейс.

React також має вбудовану систему віртуального DOM (Document Object Model), яка ефективно оновлює лише необхідні частини UI під час зміни стану, замість повного оновлення всього дерева DOM. Це робить React швидким і ефективним при відображенні великих та складних UI.

Крім того, React має широку екосистему додаткових бібліотек і інструментів, які допомагають в розробці, такі як React Router для навігації, Redux для керування станом, Axios для роботи з API, і багато інших.

Узагальнюючи, React є потужним фреймворком для розробки користувацького інтерфейсу, який спрощує розробку, покращує продуктивність і забезпечує високу реактивність веб-додатків.

Знаком якості даного фреймворку є те, що його використовують найпопулярніші веб-додатки, такі як Twitter, Instagram, WhatsApp тощо. Ось приклад показового коду на React, що демонструє базові концепції та шаблони розробки:

```
  import React, { useState, useEffect } from 'react';
  const App = () => {
    const [count, setCount] = useState(0);
    const [data, setData] = useState([]);
    useEffect(() => {
      fetchData();
    }, []);
    const fetchData = async () => {
      // Симуляція отримання даних з API
      const response = await
fetch('https://api.example.com/data');
      const data = await response.json();
      setData(data);
    };
    const incrementCount = () => {
      setCount(count + 1);
    };
    return (
      <div>
        <h1>Мій перший React додаток</h1>
        <p>Лічильник: {count}</p>
        <button onClick={incrementCount}>Збільшити</button>
        <ul>
          {data.map(item => (
            <li key={item.id}>{item.name}</li>
          ))}
        </ul>
      </div>
    );
  };
  export default App;
```

У цьому коді ми створюємо компонент `App`, який є основним компонентом нашої програми. Використовується функціональний компонент, який використовує хук `useState` для збереження стану змінної `count` (лічильник) та `data` (дані). Ми також використовуємо хук `useEffect` для

отримання даних з API при завантаженні компонента (`componentDidMount` еквівалентності).

У функції `fetchData` ми використовуємо `fetch` API для отримання даних з віддаленого сервера, а потім оновлюємо стан `data` за допомогою функції `setData`.

У функції `incrementCount` ми збільшуємо значення `count` при кожному натисканні кнопки.

У JSX-розмітці ми відображаємо значення `count` у параграфі `<p>`, кнопці `<button>`, а також список `<ul>`, де кожен елемент масиву `data` відображається у `<li>` елементі.

Це лише простий приклад, який демонструє основні концепції React, такі як стан, ефект і розмітка. У реальних проєктах React використовується для створення більш складних та потужних додатків.

2.3 Вибір IDE

2.3.1 Visual Studio Code

Розроблюючи цей проєкт, ми використовували таке середовеще, як Visual Studio Code. Це відомий безкоштовний редактор коду, розроблений компанією Microsoft [2].

Він надає розширені можливості для редагування, налагодження та розробки програмного коду в різних мовах програмування.

Особливості Visual Studio Code включають:

1. Розширення та призначені для покращення продуктивності додатки: VS Code має велику кількість розширень, які дозволяють налаштувати редактор під конкретні потреби розробника.

2. Синтаксичне виділення: VS Code надає підсвічування синтаксису для багатьох мов програмування, що полегшує читання та редагування коду.

3. Вбудована система керування версіями: Ви можете інтегрувати VS Code з системами керування версіями, такими як Git, для управління кодовою базою та командною роботою.

4. Налагодження коду: VS Code надає можливості для налагодження програмного коду з використанням точок зупинки, крокування по коду та перегляду значень змінних.

5. Інтеграція з інструментами розробки: Ви можете інтегрувати VS Code з різноманітними інструментами розробки, такими як засоби перевірки правопису, системи автоматичного форматування та інші.

Загалом, Visual Studio Code є потужним та гнучким редактором коду, який підтримує багато мов програмування та надає широкий спектр функціональних можливостей для розробників.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Інформаційна модель

Враховуючи основні вимоги та принципи розробки веб-сайтів, пропонуємо створити діаграму, яка відображатиме структуру сайту (Рис. 3.1).

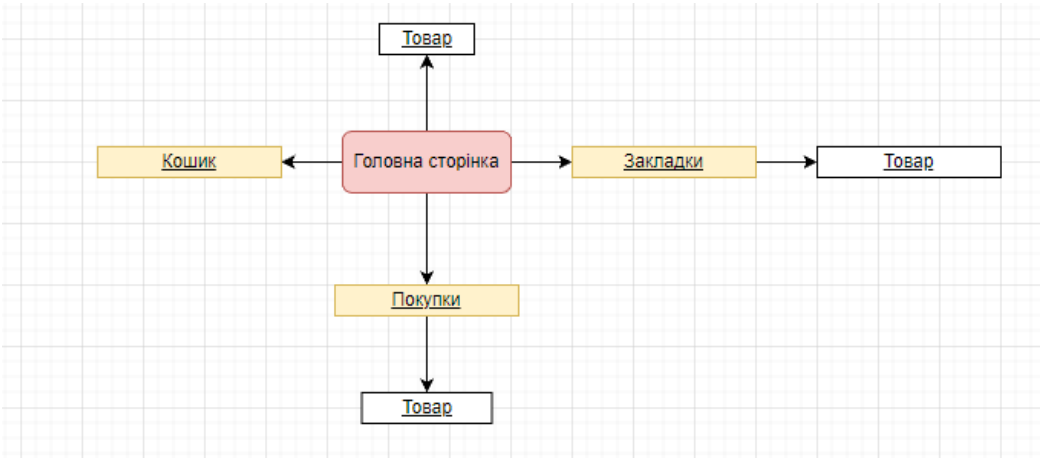

Рисунок 3.1 – Site Map діаграма

Наш веб-сайт включає два рівні доступу для користувачів: адміністратори та звичайні користувачі (Рис. 3.2).

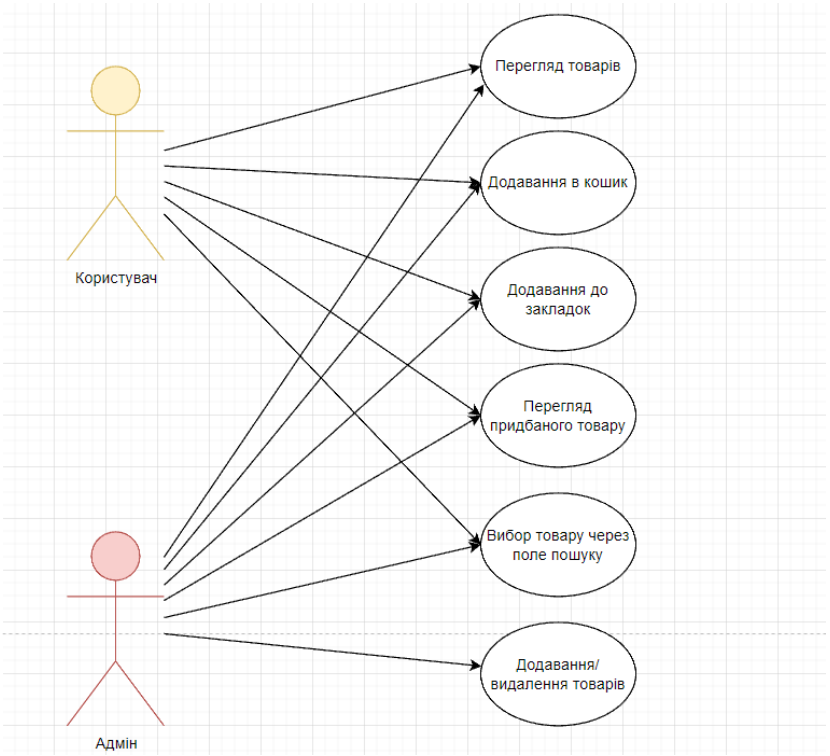

Рисунок 3.2 – Use Case діаграма

Головна відмінність полягає в тому, що адміністратор має привілеї для редагування, додавання та видалення товарів, а також виконання інших операцій у зв’язаній базі даних.

3.2 Розроблення бази даних

Для забезпечення повноцінної та ефективної роботи веб-сайту необхідно мати базу даних, в якій буде зберігатися інформація про товари, замовлення та інші важливі дані. Попередньо ми визначили, що будемо використовувати базу даних фейкового API для розробки та тестування програмного забезпечення під назвою MockAPI.

У нашому проєкті ми використовуємо декілька таблиць для зберігання даних:

1. Таблиця "Items" – містить інформацію про товари, такі як назва, опис, ціна та інші характеристики.
2. Таблиця "Cart" – зберігає дані про товари, які додані в кошик користувача. Вона містить інформацію про кількість товарів, ціни та інші деталі, пов'язані з покупками.
3. Таблиця "Favorites" – використовується для зберігання даних про закладки. Користувачі можуть додавати товари до списку обраного, і ця таблиця зберігає інформацію про ці обрані товари.
4. Таблиця "Orders" – містить дані про виконані замовлення. Вона включає інформацію про замовлені товари, адресу доставки, статус замовлення та інші важливі деталі.

Ці таблиці допомагають нам організувати та зберігати дані про товари, кошик, обрані товари та замовлення, що дозволяє користувачам зручно взаємодіяти з сайтом та здійснювати покупки.

Далі, розглянемо кожну таблицю більш детально. Наведемо структуру таблиці “Items” (Рис. 3.3)

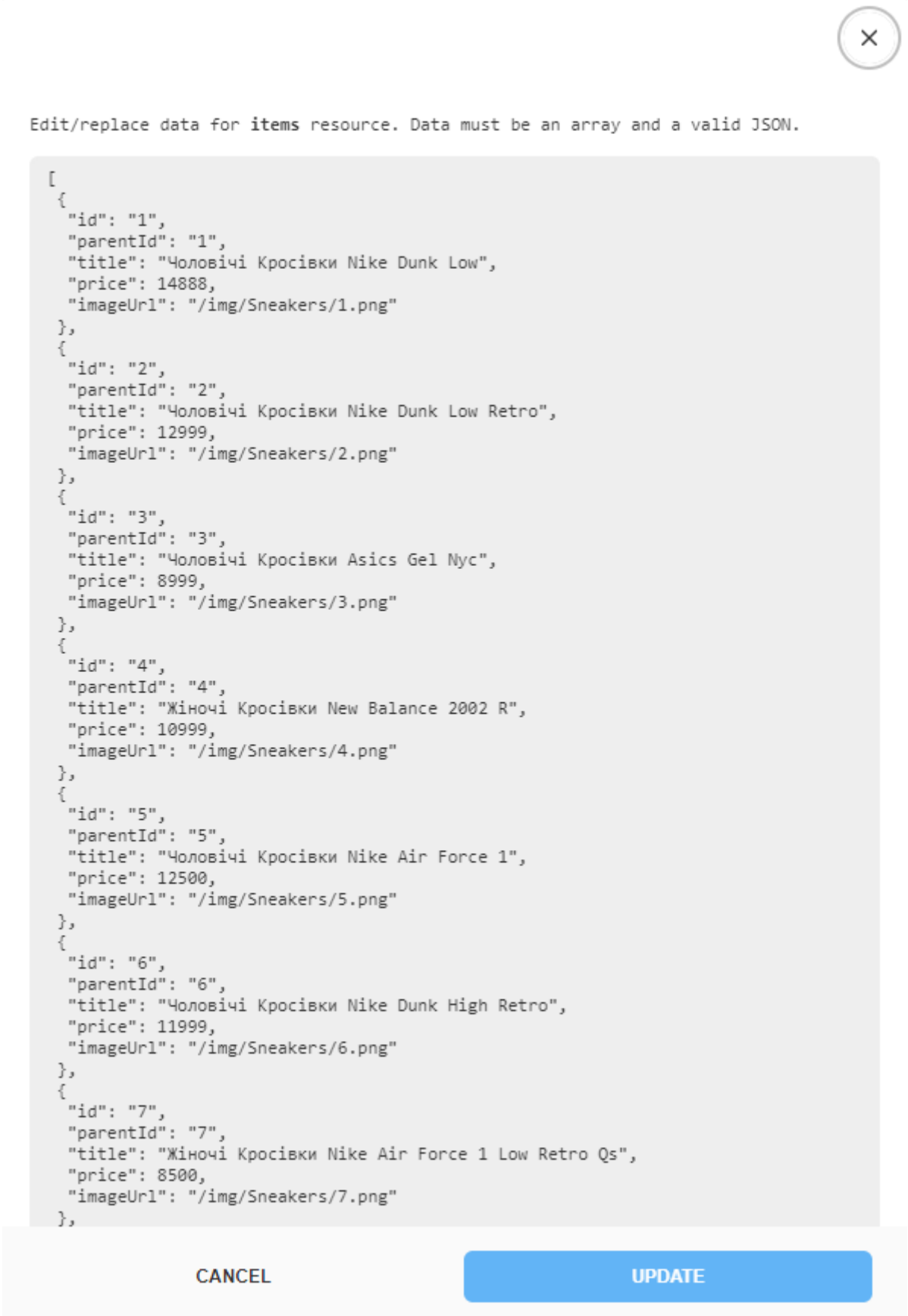

Рисунок 3.3 – Структура таблиці “Items”

Цей код містить масив об'єктів, кожен з яких представляє запис в таблиці "Items". Кожен об'єкт містить наступні поля:

- "id" – ідентифікатор товару, який є унікальним значенням для кожного запису.
- "parentId" – ідентифікатор батьківського запису, що вказує на родительську категорію товару. У данному коді всі значення "parentId" різні для кожного запису.
- "title" – назва товару, яка описує його.
- "price" – ціна товару, вказана у валюті.
- "imageUrl" – шлях до зображення товару.

Кожен об'єкт в масиві представляє один запис про товар в таблиці "Items". Ці дані використані для відображення товарів на веб-сторінці, включаючи їх назву, ціну та зображення.

Далі наведена таблиця “Cart”, для зберігання інформації про товари, які додані в кошик користувача (Рис. 3.4).

Цей код містить масив об'єктів, які представляють записи в таблиці "Items". Кожен об'єкт містить наступні поля:

- "id" – унікальний ідентифікатор товару.
- "parentId" – ідентифікатор батьківського запису.
- "title" – назва товару.
- "imageUrl" – шлях до зображення товару.
- "price" – ціна товару.

Кожен об'єкт в масиві представляє один запис про товар в таблиці "Cart". В даному випадку, таблиця містить записи про кросівк, такі як назва, зображення та ціна. "parentId" вказує на батьківську категорію, до якої відноситься кожен товар.

Таблиця “Favorites” є головною таблицею та використовується для зберігання даних про закладки.

Рисунок 3.4 – Структура таблиці “Cart”

Далі наведемо структуру таблиці (Рис. 3.5).

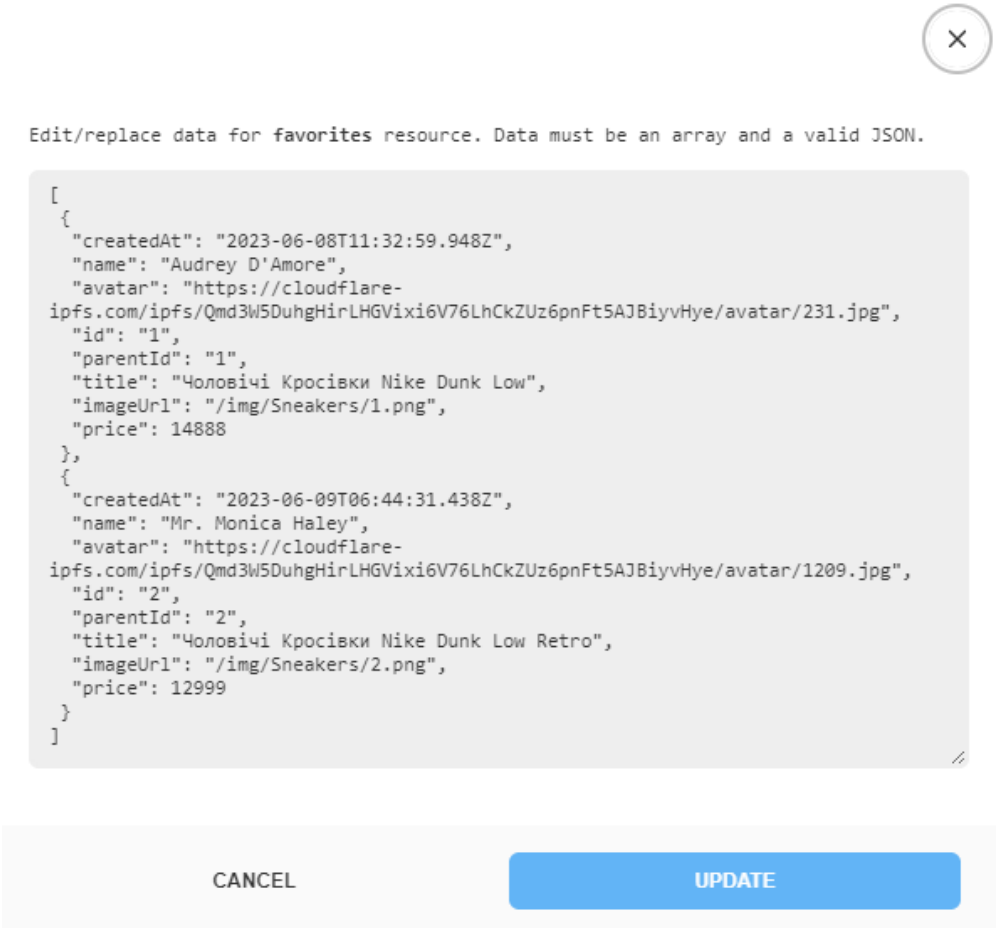

Рисунок 3.5 – Структура таблиці “Favorites”

Кожен об'єкт містить наступні поля:

- "createdAt" – дата та час створення запису.
- "name" – ім'я користувача, який створив запис.
- "avatar" – URL зображення аватарки користувача.
- "id" – унікальний ідентифікатор товару.
- "parentId" – ідентифікатор батьківського запису, який вказує на категорію або підкатегорію, до якої належить товар.
- "title" – назва товару.
- "imageUrl" – шлях до зображення товару.
- "price" – ціна товару.

Наступною таблицею розглянемо “Orders” (Рис. 3.6).

Рисунок 3.6 – Структура таблиці “Orders”

Після успішного оформлення замовлення, дані про отримувача, дату та час замовлення будуть збережені в таблиці "Orders". Ця таблиця взаємодіє з таблицями "Cart" та "Items" і містить інформацію про товари, які були додані до замовлення. Вона служить для збереження всієї необхідної інформації про замовлення, включаючи деталі товарів та інформацію про клієнта.

3.3 Програмна реалізація

На рисунку нижче зображено структуру файлів проєкту (Рисунок 3.7).

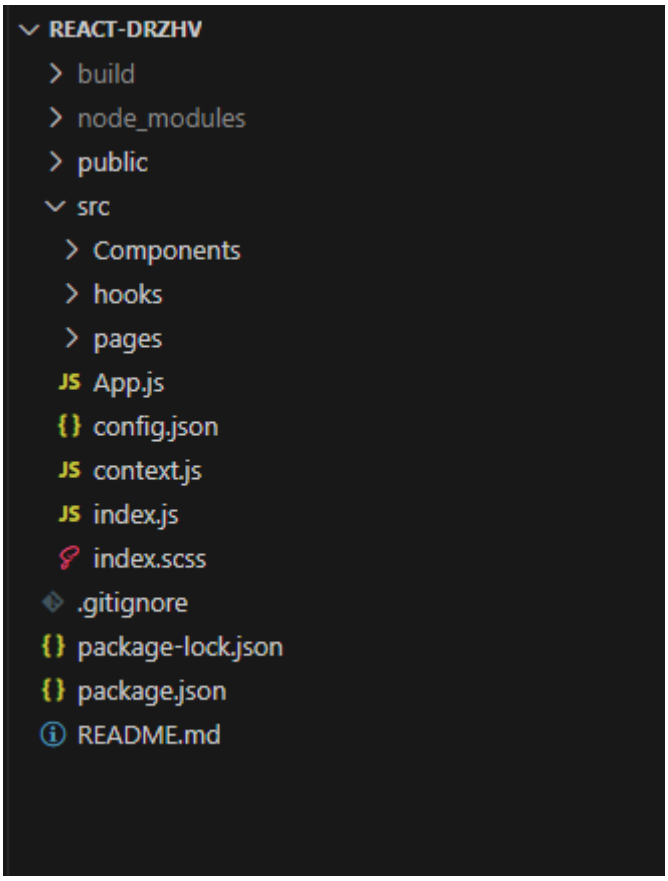

Рисунок 3.7 – Файлова структура проєкту

Правильною практикою є написанння компонетів, які можуть бути повторно використані, розширені та управляні незалежно одна від одної. Компоненти є основним концептом у React і дозволяють створювати складні інтерфейси шляхом комбінування простих компонентів.

Основним файлом додатку є App.js. В ньому визначений компонент App, який включає в себе інші компоненти, маршрутизацію та логіку взаємодії з сервером. Повний код App.js наведено в “Додаток А”.

Кода даних компонентів:

Маршрутизація

```
import { Route, Routes } from 'react-router-dom';
// ...
function App() {
  // ...
  return (
    <AppContext.Provider
      value={{ items, cartItems, favorites, isItemAdded, onAddToFavorite,
onAddToCart, setCartOpened, setCartItems }}>
      <div className="Wrapper clear">
        <Cart
          items={cartItems}
          onClose={() => setCartOpened(false)}
          onRemove={onRemoveItem}
          opened={cartOpened}
        />
        <Header onClickCart={() => setCartOpened(true)} />
        <Routes>
          <Route
            exact path="/"
            element={
              <Home
                items={items}
                cartItems={cartItems}
                searchValue={searchValue}
                setSearchValue={searchValue}
                onChangeSearchInput={onChangeSearchInput}
                onAddToFavorite={onAddToFavorite}
                onAddToCart={onAddToCart}
                isLoading={isLoading}
              />
            }
          />
          <Route
            exact path="/favorites"
            element={<Favorites />}
          />
          <Route
            exact path="/orders"
            element={<Orders />}
          />
        </Routes>
      </div>
    </AppContext.Provider>
  );
}
export default App;
```

Папка Components містить компоненти React, які використовуються для створення різних частин або елементів веб – сайту "Sneakers_Drzhv". Кожен файл у цій папці представляє окремий компонент, який може бути

використаний в інших частинах додатку для відображення та управління певними функціями.

Наприклад:

- Header.js: Компонент Header відповідає за верхню частину сторінки з заголовком та іншими елементами.
- Cart: Компонент Cart представляє кошик для збереження товарів. Він відповідає за відображення та функціональність кошика.
- Card: Відображає карточку товару (Рис. 3.8).
- Компонент Info, відображає інформаційне повідомлення на сторінці, використовується для повідомлення про порожній кошик або успішне оформлення замовлення.

Користувач може використовувати ці компоненти, імпортуючи їх у відповідні файли та використовуючи їх для створення інтерфейсу та функціональності свого додатку.

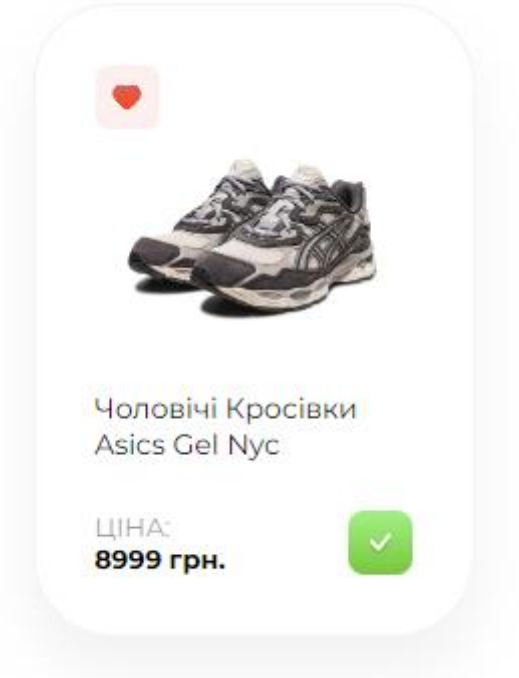

Рисунок 3.8 – Карточка товару

Давайте розглянемо його детальніше:

1. Імпорт бібліотек та файлів:

- React і ContentLoader імпортуються зовнішні бібліотеки React і react-content-loader.
- AppContext імпортується з внутрішнього файлу контексту додатку.
- Styles імпортується з файлу Card.module.scss для використання стилів компонента.

2. Оголошення змінних та стану:

- У компоненті Card оголошуються різні змінні та стани, такі як isFavorite, який використовується для відстеження стану вподобання товару, а також obj, який містить дані про товар.

3. Функції onClickPlus та onClickLiked:

- onClickPlus викликається при натисканні на кнопку "Додати в кошик". Вона викликає функцію onClickAdd та передає об'єкт obj.
- onClickLiked викликається при натисканні на кнопку "Вподобання". Вона викликає функцію onClickFavorite та змінює стан isFavorite.

4. Повернення розмітки компонента:

- Компонент Card повертає розмітку, яка відображає карточку товару.
- Якщо loading встановлено в true, відображається ContentLoader - анімаційний ефект завантаження. Інакше, відображається вміст карточки товару.
- Якщо функція onClickFavorite передана в компонент, відображається кнопка вподобання залежно від значення isFavorite.
- Відображаються зображення товару, назва, ціна та кнопка "Додати в кошик" залежно від наявності функції onClickAdd та стану isItemAdded.

У папці hooks розташований хук useCart, який використовується для отримання інформації про кошик (cart) з контексту додатку. Давайте розглянемо його детальніше.

Код навединий нижче:

```
import React from "react";
import AppContext from "../context";
export const useCart = () => {
    const { cartItems, setCartItems } = React.useContext(AppContext);
```

```
        const totalPrice = cartItems.reduce((sum, obj) => obj.price + sum,
0);

        return { cartItems, setCartItems, totalPrice };
    };
```

1. Імпорт бібліотек та файлів:

- **React** і **AppContext** імпортуються зовнішні бібліотеки React і внутрішній файл контексту додатку.

2. Визначення хука **useCart**:

- Хук **useCart** є функцією, яка повертає об'єкт зі змінними, пов'язаними з кошиком.
- За допомогою **React.useContext** отримується доступ до значень **cartItems** і **setCartItems** з контексту додатку. **cartItems** представляє масив елементів кошика, а **setCartItems** є функцією для оновлення значень кошика.
- За допомогою методу **reduce** обчислюється **totalPrice**, який представляє загальну вартість всіх елементів у кошику.
- Об'єкт зі змінними **cartItems**, **setCartItems** і **totalPrice** повертається з хука **useCart**.

Цей хук можна використовувати в інших компонентах, щоб отримати доступ до значень кошика та здійснювати необхідні операції з ним, такі як відображення елементів кошика, розрахунок загальної вартості тощо.

Папка pages зберігає в собі компоненти сторінок додатку.

Рисунок 3.9 – Папка pages

3.3.1 Favorites.jsx

"Мої закладки" (Favorites) в інтернет-магаизні "Sneakers_Drzhv". Основною функцією сторінки – відображення списку обраних товарів користувача.

У першому рядку імпортуються необхідні залежності, включаючи React, компонент Card та AppContext зі спільного контексту додатку.

Функція Favorites використовує хук React.useContext для отримання доступу до значень favorites (список обраних товарів) та onAddToFavorite (функція додавання до обраних) з AppContext.

В рендері компоненту Favorites, відображається контейнер з класом "Content" і оточений паддингом. В середині контейнера розташовані заголовок "Мої закладки" (h1), а під ним - блок для відображення списку обраних товарів.

У блоку, за допомогою методу map(), кожен елемент зі списку favorites рендериться в компоненті Card. Передаються необхідні властивості, такі як ключ (key), значення favorited (вказує, що товар є обраним), функція onClickFavorite для обробки події натискання на кнопку "обрано" та решта властивостей об'єкта item розпаковуються за допомогою оператора spread.

Отже, код відповідає за відображення списку обраних товарів та передачу необхідних властивостей до компонента Card для кожного елементу списку.

Повний код цього додатка дивись в “Додаток Ґ”.

Нижче наведемо як закладки виглядають візуально: (Рис. 3.10)

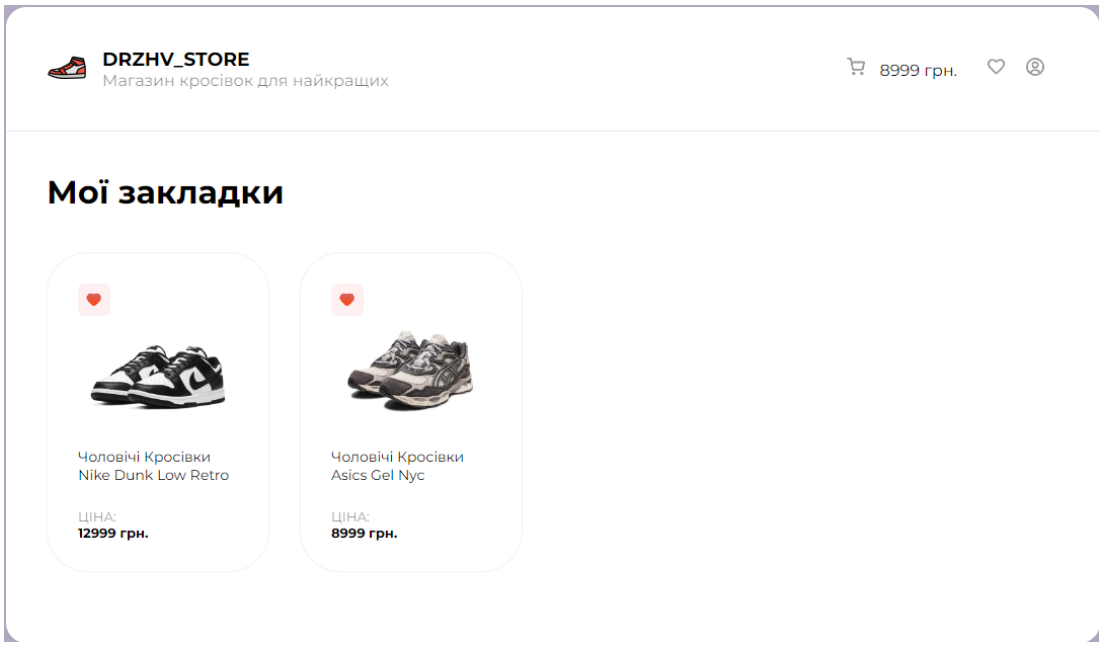

Рисунок 3.10 – Візуальний вигляд закладок

3.3.2 Home.jsx

Код представляє собою функціональний компонент "Home" у React, який відображає сторінку зі списком карток. Найголовніші елементи коду включають:

Компонент "Card": використовується для відображення окремої картки.

- Функція "renderItems": фільтрує та відображає відфільтровані елементи зі списку карток.
- Поле пошуку: дозволяє користувачеві вводити текст для пошуку конкретних карток.
- Інтерактивні функції: onAddToFavorite та onAddToCart - виконують певні дії, коли користувач натискає на кнопки "додати до обраного" та "додати до кошика".
- Завантаження: isLoading – відображається показник завантаження, якщо дані ще не отримані.

Компонент "Home" відображає блок з назвою, полем пошуку та списком карток. Він використовує передані параметри для відображення даних та обробки подій. Компонент забезпечує функціональність пошуку, додавання до обраного та додавання до кошика для кожної картки.

Для більш детального коду даного додатку дивись “Додаток Ґ”.

3.3.3 Orders.jsx

Код представляє собою функціональний компонент "Orders" у React, який відображає сторінку зі списком замовлень. Найголовніше в коді:

- Компонент "Card": використовується для відображення окремого замовлення.
- Використання Axios: використовується для здійснення HTTP-запиту до зовнішнього API та отримання списку замовлень.
- Використання React Hooks: використовуються useState та useEffect для збереження стану замовлень та відображення списку.

- Завантаження даних: встановлюється флаг isLoading, який показує, чи завантажуються дані з API.
- Відображення списку замовлень: відображається список замовлень, або показується індикатор завантаження, якщо дані ще не отримані.

Компонент "Orders" використовує ефект useEffect для виконання HTTP-запиту до MockAPI при завантаженні компонента. Результати запиту обробляються, а дані замовлень зберігаються в стані компонента. Замовлення відображаються за допомогою компонента "Card" (Рис. 3.11). Реалізацію цього коду можна знайти в “Додаток Ґ”.

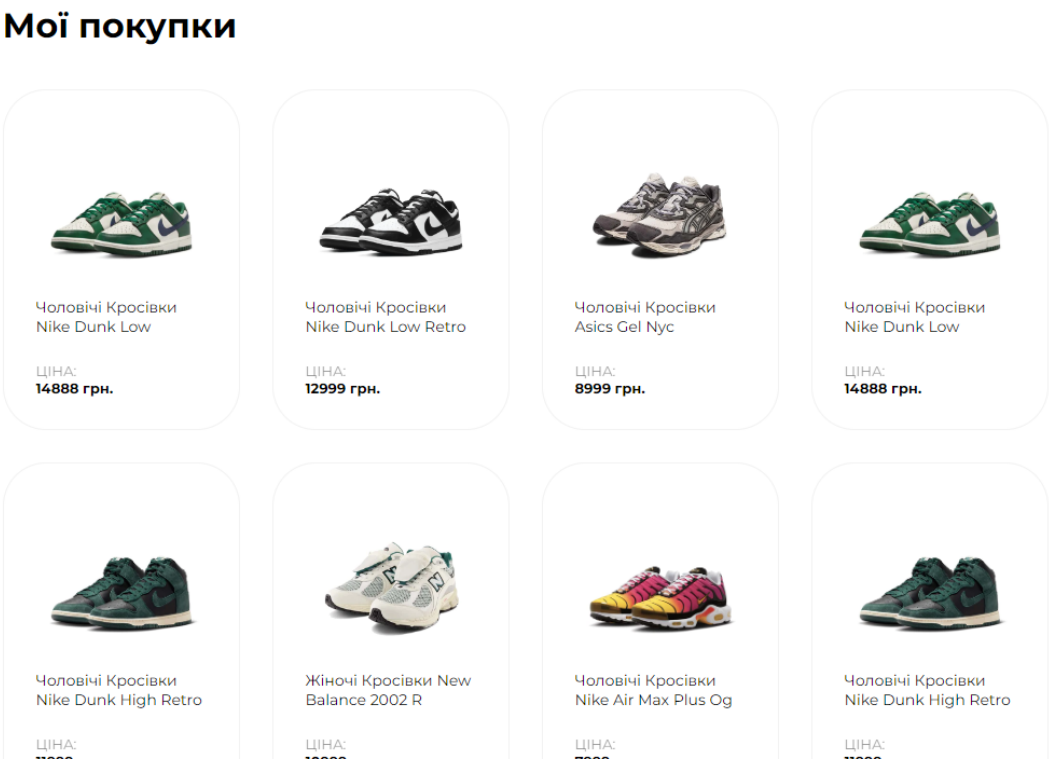

Рисунок 3.11 – Сторінка списку замовлень

Context.js створює контекст додатка з використанням React.createContext(). Код наведений нижче:

```
import React from 'react';
const AppContext = React.createContext({});
export default AppContext;
```

Контекст (AppContext) використовується в React для передачі даних вниз по дереву компонентів без необхідності передавати їх через пропси.

Ініціалізується порожнім об'єктом {} як значенням за замовчуванням контексту. Цей код не визначає значення контексту, а лише створює контекст, який використаний імпортом AppContext у інших компонентах для доступу до значень контексту. Значення контексту можна задати в компонентах вищого рівня, надаючи їм значення через компонент Provider, який обгортає дерево компонентів в файлі App.js. Повний код якого ви можете побачити в “Додаток А”.

Компонент **Cart**. Відповідає за відображення та функціонал компонента “Кошик”. Повний код цього компоненту можна подивитися в “Додаток Б”. Нижче наведемо, як кошик виглядає візуально: (Рис. 3.12).

Основні елементи коду:

- Компонент "Cart" приймає різні пропси, такі як onClose, onRemove, items, opened.
- Використовуються певні хуки з вбудованими функціями React, такими як useState та useEffect.
- Компонент здійснює HTTP-запити до фальшивого API за допомогою axios для створення замовлення та видалення елементів з кошика.
- Застосовується логіка відображення залежно від стану кошика та стану оформлення замовлення.
- Реалізовані обробники подій для кнопок та функціонал додавання/видалення елементів з кошика.

З найголовнішого можу виділити:

1. Компонент використовує хуки useState та useEffect для управління станами і здійснення певних дій при їх зміні.
2. Здійснюється взаємодія з фальшивим API за допомогою бібліотеки axios для створення замовлення та видалення елементів з кошика.
3. Відображення компонента залежить від стану кошика та стану оформлення замовлення. Якщо кошик порожній, відображається підказка, в іншому випадку відображаються елементи кошика та загальна сума.

4. Реалізовані обробники подій для кнопок, таких як додавання/видалення елементів з кошика та оформлення замовлення.

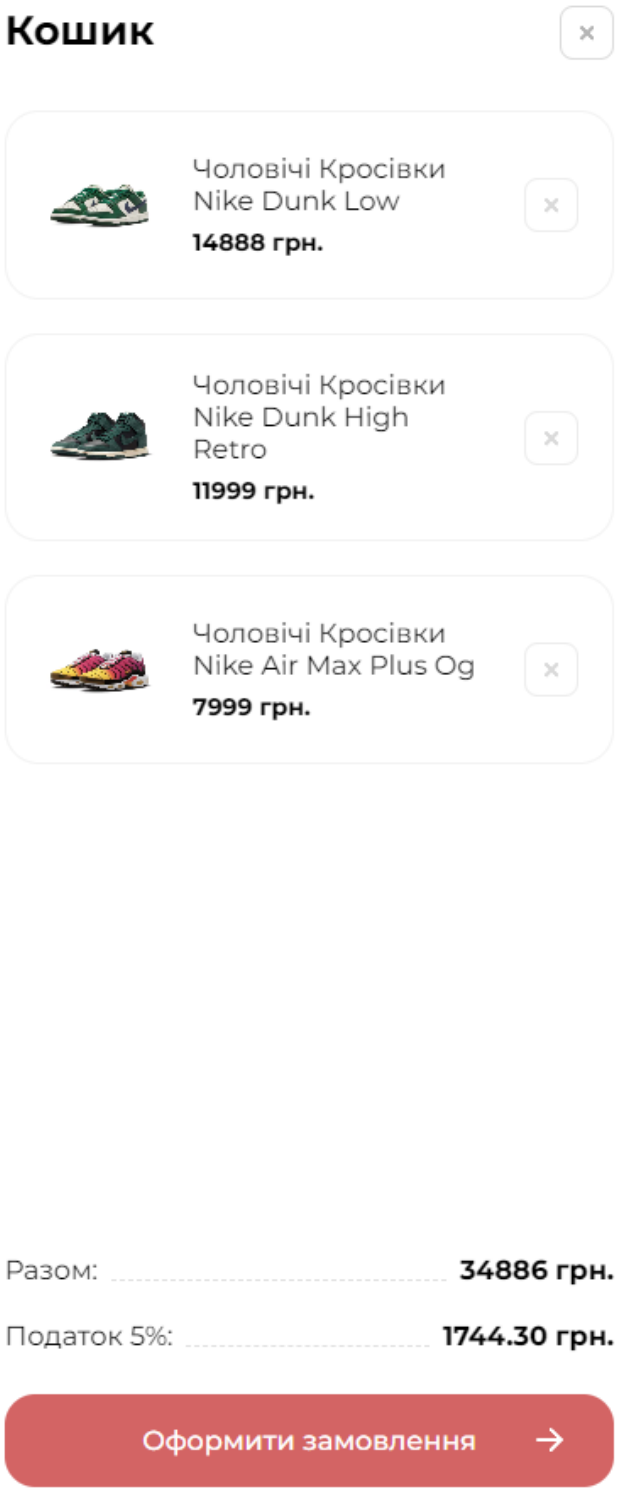

Рисунок 3.12 – Візуальний вигляд кошику

Зміст файлу index.js використовується для відображення кореневого компоненту App у веб – додатку за допомогою React та бібліотеки React Router:

```
import React from 'react';
import { BrowserRouter as Router } from 'react-router-dom';
import ReactDOM from 'react-dom/client';
import './index.scss';
import App from './App';
import 'macro-css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Router>
      <App />
    </Router>
  </React.StrictMode>
);
```

Основні елементи коду:

- Імпортується необхідні залежності, такі як React, BrowserRouter з бібліотеки react-router-dom та ReactDOM.
- Імпортується файл стилів './index.scss'.
- Використовується функція ReactDOM.createRoot() для створення кореневого вузла (root) у HTML-документі, який має ідентифікатор 'root'.
- За допомогою методу render() кореневого вузла виконується рендеринг компонента "App" в контексті строгого режиму (StrictMode) та обгортка BrowserRouter для включення маршрутизації у додаток.

Найголовніше:

1. Використовується бібліотека React Router для створення маршрутизації в додатку.

2. Файл стилів './index.scss' використовується для налаштування зовнішнього вигляду додатку.

3. ReactDOM.createRoot() використовується для створення кореневого вузла додатку.

4. Кореневий компонент "App" рендериться в контексті строгого режиму (StrictMode), що допомагає виявляти потенційні проблеми в додатку та покращує його продуктивність.

3.4 Тестування

Давайте використаємо ручний метод тестування, щоб подивитися чи всі функції виконуються вірно.

Розпочнемо з головної сторінки та виберемо бажаний нам продукт, додамо його до кошика та закладок. (Рис. 3.13):

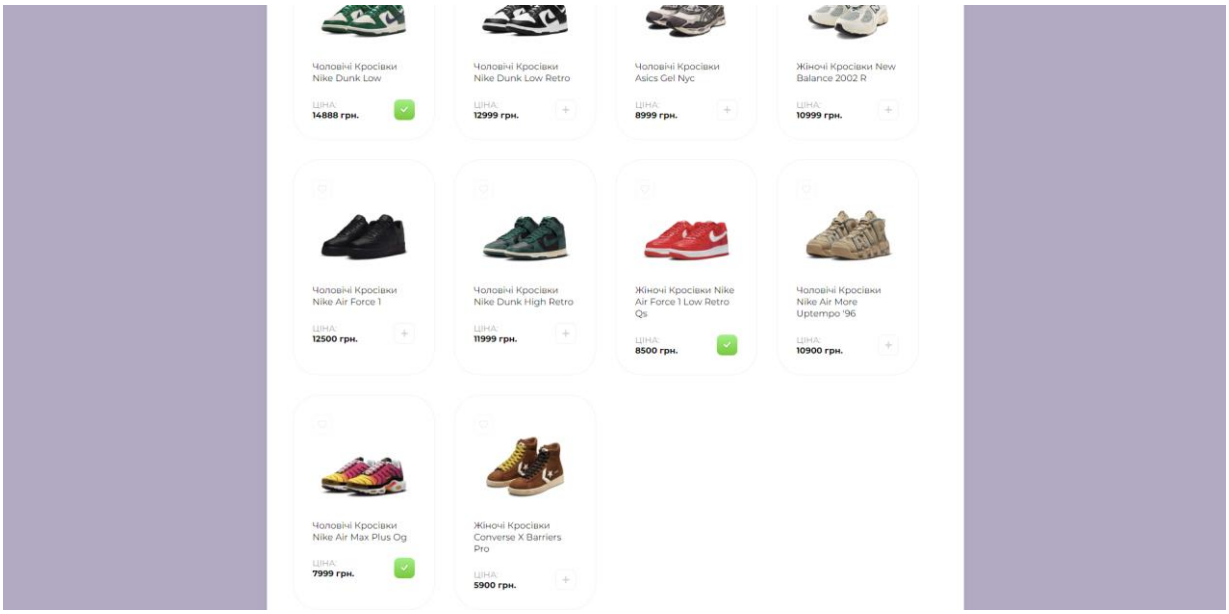

Рисунок 3.13 – Додавання до кошика

Далі додамо деякі товари до закладок. (Рис. 3.14)

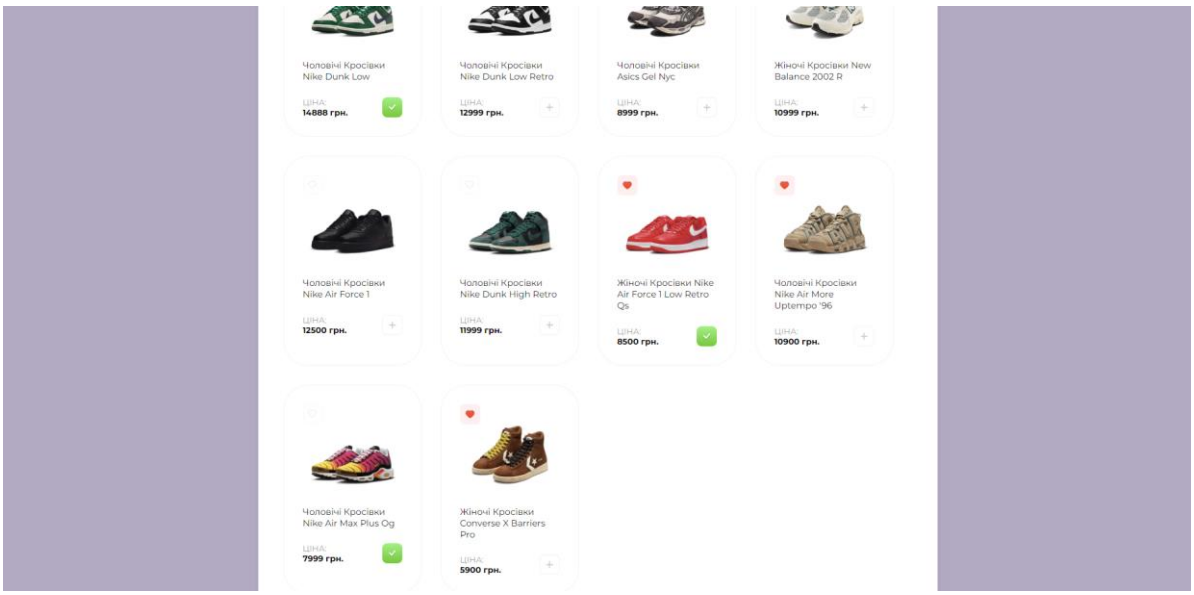

Рисунок 3.14 – Додавання до закладок

При натисканні кнопки додати до кошика запит з додаванням товару в корзину виконаний успішно, про що інформує відповідь 201. (Рис. 3.15)

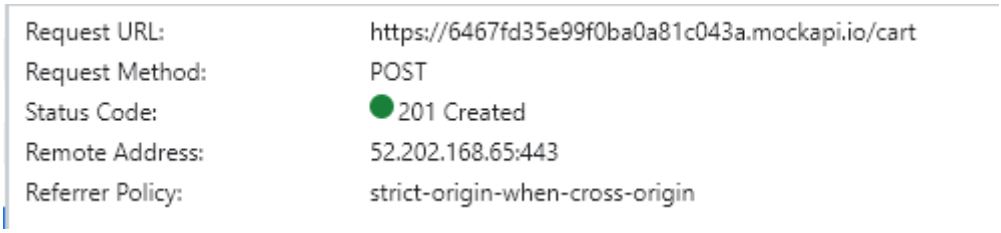

Рисунок 3.15 – Успішне додавання товару до кошику

Теж з саме з додаванням товару до кошику. (Рис. 3.16):

Рисунок 3.16 – Успішне додавання товару до закладок

Тепер давайте перейдемо до кошика. (Рис. 3.17)

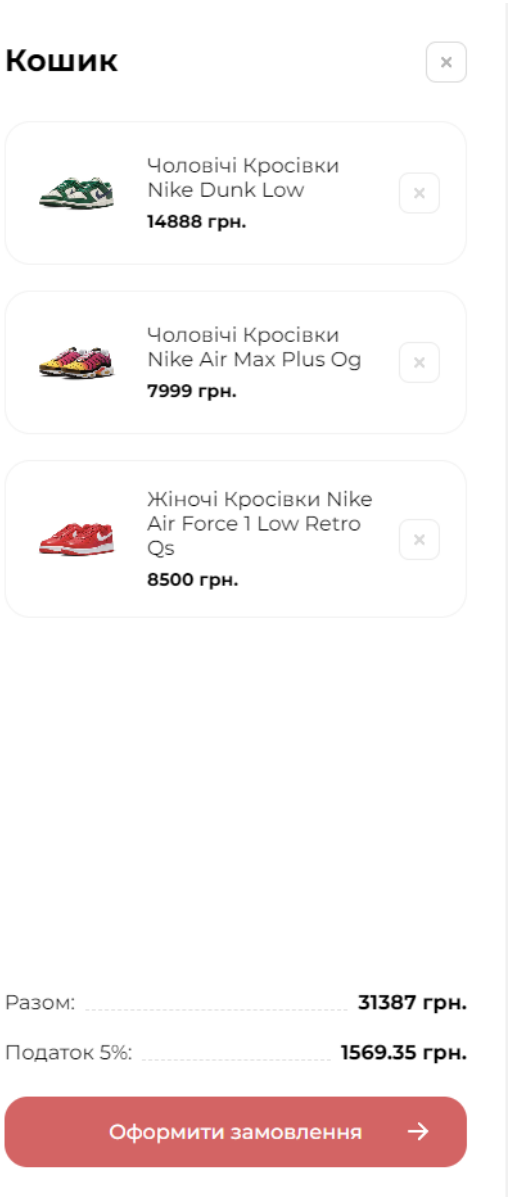

Рисунок 3.17 – Кошик нашого магазину

Як бачимо, загальну вартість товару та податок в п’ять відсотків рахується вірно, то ж можемо перейти до тестування функції видалення товару. Наприклад, видалимо товар з кошику. Повинно залишитися лише дві пари кросівок, а сума та податок повинні скласти 22887 грн. та 1144,35 грн. (Рис. 3.18)

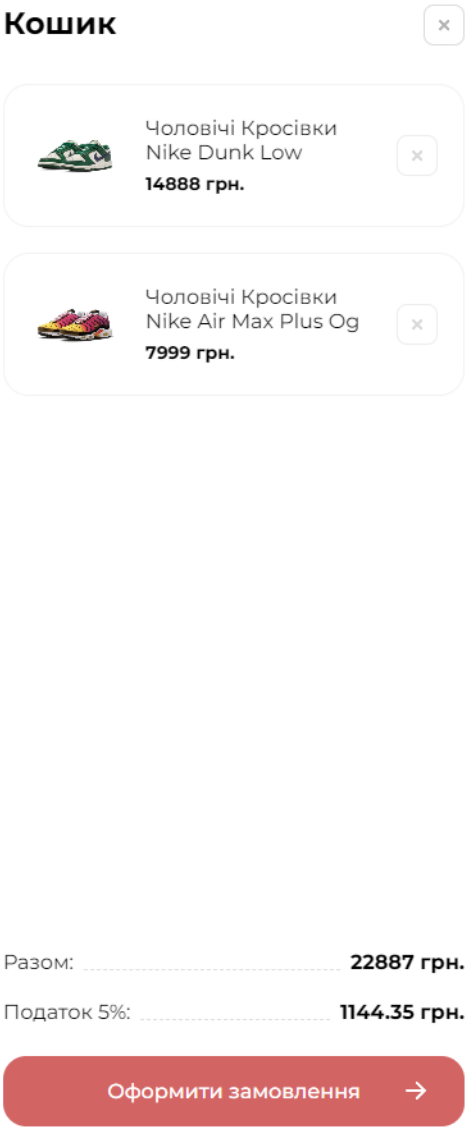

Рисунок 3.18 – Успішний результат видалення

Далі, при натисканні «Оформити замовлення» ми повинні побачити повідомлення про успішне замовлення та його номер. На рисунку нижче бачимо підтвердження. (Рис.3.19)

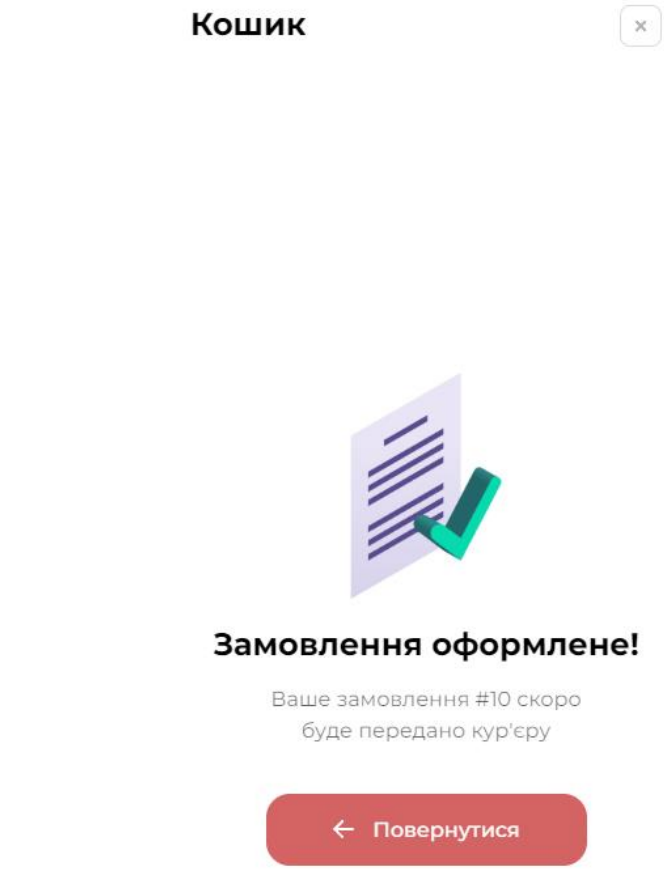

Рисунок 3.19 – Замовлення оформелене успішно

Перейдемо до закладок. На рисунку нижче, можемо побачити, що товар який ми вибрали раніше, був успішно доданий до сторінки “Мої закладки”. (Рис. 3.20)

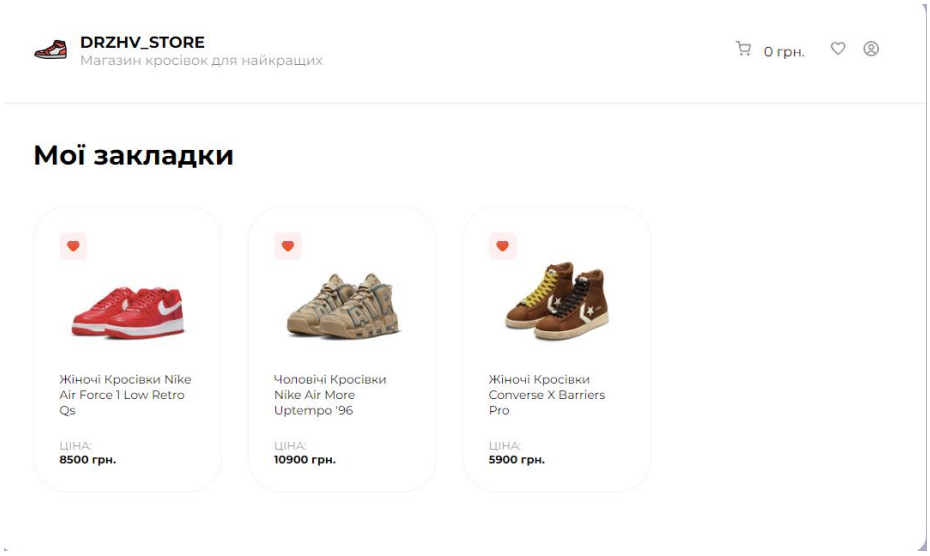

Рисунок 3.20 – Успішне додавання до закладок

Як бачимо товар було додано коректно. Тож можемо перейти до тесту видалення пар кросівок з вкладки “Мої закладки”. Видалимо другий товар з закладок. В нас повинно залишитися, лише перша та третя пара. (Рис. 3.21):

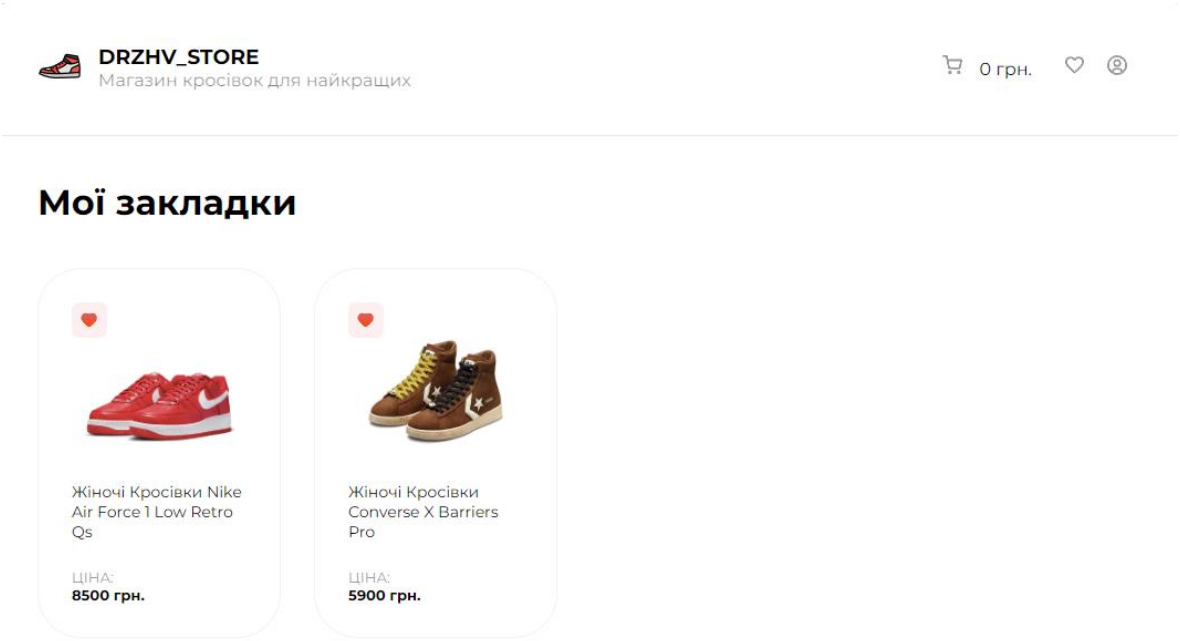

Рисунок 3.21 – Успішне видалення товару з закладок

Далі перейдемо до тестування пошуку товару. Введемо назву бренду кросівок “Asics”, в рядок “Пошук…” (Рис. 3.22)

Рисунок 3.22 – Успішний пошук за запитом

Роботу сайту можна буде назвати коректною, якщо інформацію про оформлений товар, що запишеться в тестову БД коректно виведе користувачу на сторінці “Мої покупки”. Нижче наведемо результат (Рис. 3.23):

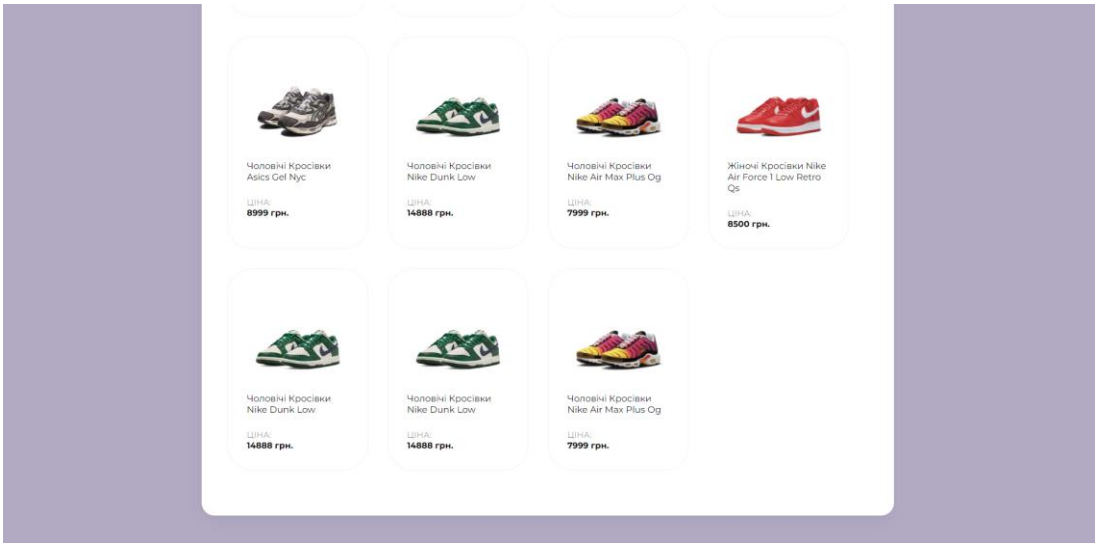

Рисунок 3.23 – Вкладка “Мої покупки”

Як ми можемо побачити, останні два товари відповідають нашому замовленню. То ж можемо сказати, що даний елемент сайту працює коректно.

На цьому дане тестування можна завершити. Отже, критичних помилок не було знайдено.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи був розроблений веб-сайт, що повністю відповідає всім основним вимогам та критеріям, зазначеним у завданні. Для зберігання відомостей про товари та замовлення була використана база даних на основі MockAPI. Були реалізовані всі основні функції, такі як перегляд товару, швидка навігація між сторінками, додавання товарів до кошика та закладок, а також перегляд раніше замовлених товарів.

Дизайн веб-сайту був розроблений з урахуванням сучасних стандартів, з використанням унікальних елементів та анімацій.

Після завершення розробки веб-сайту було проведено тестування, під час якого не було виявлено значних помилок.

З урахуванням всіх вищезазначених фактів можна зробити висновок, що поставлені завдання були успішно виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Статистика стану електронної моди - https://ecommercefastlane.com/the-state-of-the-ecommerce-fashion-industry-statistics-trends-strategy/
2. Офіційний веб-сайт Visual Studio Code - https://code.visualstudio.com/
3. Фреймворки у веб-розробці — що це, які існують і для чого потрібні - https://highload.today/uk/frejmvorki-u-veb-rozrobtsi-shho-tse-yaki-isnuyut-i-dlya-chogo-potribni/
4. React JavaScript-бібліотека для створення користувацьких інтрефейсів - https://uk.legacy.reactjs.org/
5. MockAPI docs - https://github.com/mockapi-io/docs/wiki
6. HTML: HyperText Markup Language - https://developer.mozilla.org/en-US/docs/Web/HTML
7. SCSS documentation - https://sass-lang.com/documentation/
8. Офіційна документація JavaScript на веб-сайті Mozilla Developer Network (MDN): https://developer.mozilla.org/uk/docs/Web/JavaScript/Guide
9. W3Schools документацію про SCSS: https://www.w3schools.com/sass/
10. The Modern JavaScript Tutorial - https://javascript.info/
11. What is React.js? (Uses, Examples, & More) - https://blog.hubspot.com/website/react-js

ДОДАТОК А

Код файлу src/App.js:

```
import React from 'react';
import axios from 'axios';
import { Route, Routes } from 'react-router-dom';
import Header from './Components/Header';
import Cart from './Components/Cart';
import AppContext from './context';

import Home from './pages/Home';
import Favorites from './pages/Favorites';
import Orders from './pages/Orders';



function App() {
  const [items, setItems] = React.useState([]);
  const [cartItems, setCartItems] = React.useState([]);
  const [favorites, setFavorites] = React.useState([]);
  const [searchValue, setSearchValue] = React.useState('');
  const [cartOpened, setCartOpened] = React.useState(false);
  const [isLoading, setIsLoading] = React.useState(true);




  React.useEffect(() => {
    async function fetchData() {
      try {

      const [cartResponse, favoritesResponse, itemsResponse] = await
Promise.all([ //вовращает все 3 промиса в массив, по порядку. С помощью
деструктуризации
        axios.get('https://6467fd35e99f0ba0a81c043a.mockapi.io/cart'),
//промис
        axios.get('https://646bd9eb7b42c06c3b2a8632.mockapi.io/favorites'),
//промис 2
        axios.get('https://6467fd35e99f0ba0a81c043a.mockapi.io/items'),
//промис 3
      ]);


      setIsLoading(false);
```

```
      setCartItems(cartResponse.data);
      setFavorites(favoritesResponse.data);
      setItems(itemsResponse.data);
      } catch (error) {
        alert('Помилка при запиті даних');
      }
    }

    fetchData();
  }, []);



  const onAddToCart = async (obj) => {
    try {
      const findItem = cartItems.find((item) => Number(item.parentId) ===
Number(obj.id));
      if (findItem) {
        setCartItems((prev) => prev.filter((item) => Number(item.parentId) !==
Number(obj.id)));
        await
axios.delete(`https://6467fd35e99f0ba0a81c043a.mockapi.io/cart/${findItem.id}
`);
      } else {
        setCartItems((prev) => [...prev, obj]); // передаем обьект
        const                  {data}                  =                  await
axios.post('https://6467fd35e99f0ba0a81c043a.mockapi.io/cart',    obj);    //
отправляем обьект на бекенд
        setCartItems((prev) =>
          prev.map(item => {
            if (item.parentId === data.parentId) {
              return {
                ...item,
                id: data.id
              };
            }
            return item;
          }),
        );
      }
    } catch (error) {
      alert('Помилка при додаванні до кошику');
      console.error(error);
```

```
    }
  };


  const onRemoveItem = (id) => {
    try {
      axios.delete(`https://6467fd35e99f0ba0a81c043a.mockapi.io/cart/${id}`);
      setCartItems((prev) => prev.filter((item) => Number(item.id)!==
Number(id)));
    } catch (error) {
      alert('Помилка при видаленні з кошика');
      console.error(error);
    }
  };

  const onAddToFavorite = async (obj) => {
    try {
      if (favorites.find((favObj) => Number(favObj.id) === Number(obj.id))) {

axios.delete(`https://646bd9eb7b42c06c3b2a8632.mockapi.io/favorites/${obj.id}
`);
        setFavorites((prev) => prev.filter((item) => Number(item.id) !==
Number(obj.id)));
    } else {
      const { data } = await
axios.post('https://646bd9eb7b42c06c3b2a8632.mockapi.io/favorites', obj);
      setFavorites((prev) => [ ...prev, data]);
    }
    } catch (error) {
      alert('Не вдалося додати до фаворитів');
      console.error(error);
    }
  };

  const onChangeSearchInput = (event) => {
    setSearchValue(event.target.value);
  };

  const isItemAdded = (id) => {
    return cartItems.some((obj) => Number(obj.parentId) === Number(id));
  };
```

```
  return (
    <AppContext.Provider
      value={{ items, cartItems, favorites, isItemAdded, onAddToFavorite,
onAddToCart, setCartOpened, setCartItems }}>
      <div className="Wrapper clear">
        <Cart
          items={cartItems}
          onClose={() => setCartOpened(false)}
          onRemove={onRemoveItem}
          opened={cartOpened}
        />



        <Header onClickCart={() => setCartOpened(true)} />

        <Routes>
          <Route
            exact path="/"
            element={
            <Home
              items={items}
              cartItems={cartItems}
              searchValue={searchValue}
              setSearchValue={searchValue}
              onChangeSearchInput={onChangeSearchInput}
              onAddToFavorite={onAddToFavorite}
              onAddToCart={onAddToCart}
              isLoading={isLoading}/>
            }
            />

          <Route
            exact path="/favorites"
            element={
            <Favorites />
            }
            />

          <Route
            exact path="/orders"
            element={
            <Orders />
```

```
            }
            />

        </Routes>
      </div>
    </AppContext.Provider>
  );
}

export default App;
```

ДОДАТОК Б

Код файлу src/Components/Card/index.js:

```
import React from 'react';
import ContentLoader from 'react-content-loader';

import AppContext from '../../context';

import Styles from './Card.module.scss';


function Card({
  id,
  title,
  imageUrl,
  price,
  onClickFavorite,
  onClickAdd,
  favorited = false,
  loading = false,
}) {
    const { isItemAdded } = React.useContext(AppContext);
    const [isFavorite, setIsFavorite] = React.useState(favorited);
    const obj = { id, parentId: id, title, imageUrl, price };



    const onClickPlus = () => {
      onClickAdd(obj);
    };

    const onClickLiked = () => {
      onClickFavorite (obj);
      setIsFavorite (!isFavorite);
    };

    return (
        <div className={Styles.Card}>
          {loading ? (
            <ContentLoader
              speed={2}
              width={155}
              height={256}
```

```
              viewBox="0 0 155 265"
              backgroundColor="#f3f3f3"
              foregroundColor="#ecebeb">
              <rect x="1" y="0" rx="10" ry="10" width="155" height="155" />
              <rect x="0" y="167" rx="5" ry="5" width="155" height="15" />
              <rect x="0" y="187" rx="5" ry="5" width="100" height="15" />
              <rect x="1" y="234" rx="5" ry="5" width="80" height="25" />
              <rect x="124" y="230" rx="10" ry="10" width="32" height="32" />
            </ContentLoader>
          ) : (
            <>
            {onClickFavorite && (
              <div className={Styles.Favorite} onClick={onClickLiked}>
                <img src={isFavorite ? '/img/Liked.svg' : '/img/Unliked.svg'}
alt="Unliked" />
              </div>
            )}
            <img width='100%' height={135} src={imageUrl} alt="Sneakers" />
            <h5>{title}</h5>
            <div className="d-flex justify-between align-center">
              <div className="d-flex flex-column">
                <span>Ціна:</span>
                <b>{price} грн.</b>
              </div>
              {onClickAdd && (
                <img
                  className={Styles.Plus}
                  onClick={onClickPlus}
                  src={isItemAdded(id) ? "/img/Btn_checked.svg" :
"/img/Btn_add.svg"}
                  alt="Plus"
                />
              )}
              </div>
            </>
        )}
      </div>
    );
}

export default Card;
```

Код файлу src/Components/Card/Card.module.scss:

```
.Card {
    border: 1px solid #F3F3F3;
    padding: 30px;
    width: 220px;
    border-radius: 40px;
    margin-right: 30px;
    margin-bottom: 30px;
    transition: box-shadow 0.1s ease-in-out, transform 0.1s ease-in-out;

    .Plus {
      cursor: pointer;
    }

    &:hover{
      box-shadow: 0px 18px 30px rgba(0, 0, 0, 0.05);
      transform: translateY(-5px);
    }

    .Favorite {
      position: absolute;
      cursor: pointer;
    }

    span{
      font-size: 13px;
      opacity: 0.5;
      text-transform: uppercase;
    }

    b{
      font-size: 13px;
    }

    h5 {
      font-weight: 400;
      font-size: 14px;
    }
  }
```

Код файлу src/Components/Cart/index.js

```
//библиотеки (чужое)
import React from 'react';
```

```
import axios from 'axios';

//javascripts файлы, компонентов
import Info from '../info';
import { useCart } from '../../hooks/useCart';

//стили, прочее
import styles from './Cart.module.scss';

const delay = (ms) => new Promise((resolve) => setTimeout(resolve, ms));

function Cart({ onClose, onRemove, items = [], opened }) {
  const { cartItems, setCartItems, totalPrice } = useCart();
  const [orderId, setOrderId] = React.useState(null);
  const [isOrderComplete, setIsOrderComplete] = React.useState(false);
  const [isLoading, setIsLoading] = React.useState(false);


  const onClickOrder = async () => {
    try {
      setIsLoading(true);
      const { data } = await
axios.post('https://646bd9eb7b42c06c3b2a8632.mockapi.io/orders', {
        items: cartItems,
      });
      setOrderId(data.id);
      setIsOrderComplete(true);
      setCartItems([]);

      //"костыль" для мокапи
      for (let i = 0; i < cartItems.length; i++) {
        const item = cartItems[i];
        await
axios.delete('https://6467fd35e99f0ba0a81c043a.mockapi.io/cart/' + item.id);
        await delay(1000);
      }
    } catch (error) {
      alert('Помилка під час створення замовлення');
    }
    setIsLoading(false);
  };

  return (
```

```
    <div className={`${styles.shadow} ${opened ? styles.shadowVisible :
''}`}>
      <div className={styles.cart}>
        <h2 className="d-flex justify-between mb-30">
          Кошик
          <img onClick={onClose} className="cu-p" src="/img/Btn-remove.svg"
alt="Close" />
        </h2>

        {items.length > 0 ? (
          <div className="d-flex flex-column flex">
            <div className="Items flex">
              {items.map((obj) => (
                <div key={obj.id} className="CartItem d-flex align-center mb-
20">
                  <div style={{ backgroundImage: `url(${obj.imageUrl})` }}
className="CartItemImg"></div>

                  <div className="mr-20 flex">
                    <p className="mb-5">{obj.title}</p>
                    <b>{obj.price} грн.</b>
                  </div>
                  <img
                    onClick={() => onRemove(obj.id)}
                    className="RemoveBtn"
                    src="/img/Btn-remove.svg"
                    alt="Remove"
                  />
                </div>
              ))}
            </div>
            <div className="CartTotal">
              <ul>
                <li>
                  <span>Разом:</span>
                  <div></div>
                  <b>{totalPrice} грн.</b>
                </li>
                <li>
                  <span>Податок 5%:</span>
                  <div></div>
                  <b>{parseFloat((totalPrice / 100) * 5).toFixed(2)} грн.</b>
                </li>
```

```
              </ul>
              <button disabled={isLoading} onClick={onClickOrder}
className="redButton">
                Оформити замовлення <img src="/img/Arrow.svg" alt="Arrow" />
              </button>
            </div>
          </div>
        ) : (
          <Info
            title={isOrderComplete ? 'Замовлення оформлене!' : 'Кошик
порожній'}
            description={
              isOrderComplete
                ? `Ваше замовлення #${orderId} скоро буде передано кур'єру`
                : 'Додайте хоча б одну пару кросівок, щоб зробити
замовлення.'
            }
            image={isOrderComplete ? '/img/complete-order.jpg' : '/img/empty-
cart.jpg'}
            items={items}
          />
        )}
      </div>
    </div>
  );
}

export default Cart;
```

Код файлу src/Components/Cart/Cart.module.scss

```
.shadow {
    position: absolute;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.6);
    z-index: 1;
    visibility: hidden;
    opacity: 0;
    position: fixed;
    transition: opacity 0.1s ease-out, visibility 0.1s ease-in-out;
    overflow: hidden;
```

```
}

.shadowVisible {
    visibility: visible;
    opacity: 1;

    .cart {
        transform: translateX(0);
    }
}

.cart {
    display: flex;
    flex-direction: column;
    position: absolute;
    width: 420px;
    height: 100%;
    right: 0;
    background: #FFFFFF;
    box-shadow: -10px 4px 24px rgba(0, 0, 0, 0.1);
    padding: 30px;
    transform: translateX(100%);
    transition: transform 0.3s ease-in-out;

    .Items {
      flex: 1;
      overflow: auto;
      margin-bottom: 40px;
    }

    h2 {
      margin: 0;
      font-size: 24px;
    }
}
```

ДОДАТОК В

Код файлу src/Components/Header.js:

```
import React from 'react';
import { Link } from 'react-router-dom';

import { useCart } from '../hooks/useCart';

function Header(props) {
  const { totalPrice } = useCart();

    return (
        <header className="d-flex justify-between align-center p-40">
          <Link to="/">
            <div className="d-flex align-center">
              <img width={40} height={40} src="/img/Logo.png" alt="Logo" />
                <div>
                  <h3 className="text-uppercase">Drzhv_Store</h3>
                  <p className="opacity-5">Магазин кросівок для найкращих</p>
                </div>
            </div>
        </Link>
        <ul className="d-flex">
          <li onClick={props.onClickCart} className="mr-30 cu-p">
            <img width={18} height={18} src="/img/Cart.svg" alt="Кошик" />
            <span>{totalPrice} грн.</span>
          </li>
          <li className="mr-5 cu-p">
            <Link to="/favorites">
              <img width={18} height={18} src="/img/Heart.svg" alt="Закладки"
/>
            </Link>
          </li>
          <li>
          <Link to="/orders">
              <img width={18} height={18} src="/img/User.svg" alt="Юзер" />
            </Link>
          </li>
        </ul>
      </header>
    );

}
```

```
export default Header;
```

Код файлу src/Components/info.jsx:

```
import React from 'react'
import AppContext from '../context';

const Info = ({ title, image, description }) => {
    const { setCartOpened } = React.useContext (AppContext);


  return (
    <div className="cartEmpty d-flex align-center justify-center flex-column
flex">
            <img
                className="mb-20"
                width="120px"
                src={image}
                alt="Empty"
            />
            <h2>{title}</h2>
            <p className="opacity-6">{description}</p>
            <button onClick={() => setCartOpened(false)}
className="redButton">
                <img src="/img/Arrow.svg" alt="Arrow" />
                Повернутися
            </button>
          </div>
  )
}

export default Info;
```

ДОДАТОК Г

Код файлу src/hooks/useCart.jsx:

```
import React from "react";
import AppContext from "../context";

export const useCart = () => {
    const { cartItems, setCartItems } = React.useContext(AppContext);
    const totalPrice = cartItems.reduce((sum, obj) => obj.price + sum, 0);

    return { cartItems, setCartItems, totalPrice };
};
```

ДОДАТОК Ґ

Код файлу src/pages/Favorites.jsx:

```
import React from 'react';
import Card from '../Components/Card';
import AppContext from '../context';


    function Favorites() {
      const { favorites, onAddToFavorite } = React.useContext(AppContext);

      return (
        <div className="Content p-40">
          <div className="d-flex align-center justify-between mb-40">
            <h1>Мої закладки</h1>
          </div>

          <div className="d-flex flex-wrap">
            {favorites.map((item, index) => (
              <Card
                key={index}
                favorited={true}
                onClickFavorite={onAddToFavorite}
                { ... item}
              />
            ))}
          </div>
        </div>
      );
}

export default Favorites;
```

Код файлу src/pages/Home.jsx:

```
import React from 'react';

import Card from '../Components/Card';

function Home({
    items,
    searchValue,
    setSearchValue,
```

```
    onChangeSearchInput,
    onAddToFavorite,
    onAddToCart,
    isLoading
}){
  const renderItems = () => {
      const filtredItems = items.filter((item) =>
        item.title.toLowerCase().includes (searchValue.toLowerCase()),
      );
      return ( isLoading ? [...Array(12)] : filtredItems).map((item, index)
=> (
        <Card
          key={index}
          onClickFavorite={(obj) => onAddToFavorite(obj)}
          onClickAdd={(obj) => onAddToCart(obj)}
          loading={isLoading}
          {...item}
        />
      ));
  };




    return (
        <div className="Content p-40">
        <div className="d-flex align-center justify-between mb-40">
          <h1>{searchValue ? `Пошук за запитом: "${searchValue}"` : 'Усі
кросівки'}</h1>
          <div className="Search-block d-flex">
            <img src="/img/search.svg" alt="Search" />
            {searchValue && (
            <img
              onClick={() => setSearchValue('')}
              className="clear cu-p"
              src="/img/Btn-remove_noBorder.svg"
              alt="Clear"
              />
            )}
            <input onChange={onChangeSearchInput} value={searchValue}
placeholder="Пошук..."  />
          </div>
        </div>
```

```
        <div className="d-flex flex-wrap">
          {renderItems()}
        </div>
      </div>
    );
}

export default Home;
```

Код файлу src/pages/Orders.jsx:

```
import React from 'react';
import axios from 'axios';

import Card from '../Components/Card';
// import AppContext from '../context';

    function Orders() {
      // const {onAddToFavorite, onAddToCart} = React.useContext(AppContext);
      const [orders, setOrders] = React.useState([]);
      const [isLoading, setIsLoading] = React.useState(true);


      React.useEffect(() => {
        (async() => {
          try {
            const { data } = await
axios.get('https://646bd9eb7b42c06c3b2a8632.mockapi.io/orders');
            setOrders(data.reduce((prev, obj) => [...prev, ...obj.items],
[]));
            setIsLoading(false);
          } catch (error) {
            alert('Помилка при запиті замовлень');
            console.error(error);
          }
        })();
      }, []);

      return (
        <div className="Content p-40">
          <div className="d-flex align-center justify-between mb-40">
```

```
            <h1>Мої покупки</h1>
          </div>

          <div className="d-flex flex-wrap">
            {( isLoading ? [...Array(12)] : orders).map((item, index) => (
              <Card key={index}
              loading={isLoading}
              {...item} />
            ))}
          </div>
        </div>
      );
}

export default Orders;
```

ДОДАТОК Д

Код файлу src/context.js:

```
import React from 'react';

const AppContext = React.createContext({});

export default AppContext;
```

Код файлу src/index.js:

```
import React from 'react';
import { BrowserRouter as Router } from 'react-router-dom';
import ReactDOM from 'react-dom/client';
import './index.scss';
import App from './App';
import 'macro-css';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Router>
      <App />
    </Router>
  </React.StrictMode>
);
```

ДОДАТОК Е

Код файлу src/index.scss:

```
body {
  margin: 0;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  background-color: #B2AAC3;
}

* {
  font-family: 'Montserrat', system-ui;
}

.Wrapper {
  background: #FFFFFF;
  box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.04);
  border-radius: 20px;
  max-width: 1080px;
  margin: 50px auto;
}

header {
  border-bottom: 1px solid #EAEAEA;
  img {
    margin-right: 15px;
  }

  h3,
  p {
    margin: 0;
  }
}

.Content{
  h1 {
    margin: 0;
  }
}

.cartEmpty {
  text-align: center;
```

```
  p {
    width: 280px;
    line-height: 24px;
  }

    .redButton {
      width: 245px;
      margin-top: 20px;

      &:hover {
        img {
          transform: rotate(180deg) translate(3px);
        }
      }

      img {
        position: relative;
        top: 1px;
        transform: rotate(180deg);
        margin-right: 15px;
        transition: transform 0.15s ease-in-out;
      }
    }
  }

.Search-block {
  border: 1px solid #F3F3F3;
  border-radius: 10px;
  padding: 0 15px;
  position: relative;

  .clear{
    position: absolute;
    right: 0;
    width: 10px;
    top: 15px;
    right: 15px;
  }

  input {
    border: 0;
    padding: 10px;
    font-size: 16px;
```

```
    width: 250px;
  }
}


.CartTotal {
  ul {
    display: block;
    margin-bottom: 25px !important;

    li{
      display: flex;
      align-items: flex-end;
      margin-bottom: 20px;

      div {
        flex: 1;
        height: 1px;
        border-bottom: 1px dashed #dfdfdf;
        position: relative;
        top: -3px;
        margin: 0 8px;
      }
    }
  }

  .redButton {
    position: relative;

    &:disabled {
      animation: button-loading 0.8s ease-in-out infinite;
    }

    &:hover {
      img{
        transform: translateX(5px);
    }
  }


    img {
      position: absolute;
      right: 30px;
```

```
      top: 20px;
      transition: transform 0.9s ease-in-out;
    }
  }
}

.redButton {
    width: 100%;
    height: 55px;
    background: #D36464;
    border-radius: 18px;
    border: 0;
    color: #FFFFFF;
    font-size: 16px;
    font-weight: 450;
    cursor: pointer;
    transition: background 0.1s ease-in-out;

    &:disabled {
      background-color: #d4d4d4 !important;
      cursor: default;
    }

    svg {
      margin-left: 20px;
    }

    &:hover {
      background: lighten(#D36464, 5%);
    }

    &:active {
      background: darken(#D36464, 5%);
    }
}

.CartItem {
  border: 1px solid #F3F3F3;
  border-radius: 20px;
  overflow: hidden;
  padding: 20px;

  .CartItemImg {
```

```
    height: 70px;
    width: 70px;
    background-size: contain;
    background-position: 0 -4px;
    background-repeat: no-repeat;
    margin-right: 20px;
  }


  p {
    font-size: 16px;
    margin: 0;
  }

  b {
    font-size: 14px;
  }

  .RemoveBtn {
    opacity: 0.5;
    cursor: pointer;
    transition: opacity 0.20s ease-in-out;

    &:hover {
      opacity: 1;
    }

  }
}

@keyframes button-loading {
  0% {
    opacity: 1;
  }
  50% {
    opacity: 0.7;
  }
  100% {
    opacity: 1;
  }
}
```