Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедр Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Данильченко Данїла Олексійовича*
(ПІБ)

академічної групи 122-18-3
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованого додатку з продажу смартфонів*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В*			
розділів:				
спеціальний	*доц. Спірінцев В.В*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-3* *Данильченко Д.О.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованого додатку з продажу смартфонів*

затверджена наказом ректора НТУ «ДП» від 18.05.2022 № 268-с

Розділ	**Зміст виконання**	**Термін виконання**
Спеціальний	***На основі матеріалів проектно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ доц. Спірінцев. В.В.
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ Данильченко Д.О.
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *16.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 77 с., 18 рис., 3 дод., 20 джерел.

Об'єкт розробки: веб-орієнтований додаток з продажу смартфонів.

Мета кваліфікаційної роботи: розробка інтернет-магазину з продажу смартфонів із застосуванням сучасної JavaScript-бібліотеки React.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточняється постановка завдання.

У першому розділі проведено аналіз предметної області, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування застосунку, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження сайту, описана робота інтернет-магазину.

В економічному розділі визначено трудомісткість розробленого програмного продукту, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає в розробці веб-орієнтованого додатку, що забезпечить відвідувачам сайту максимально зручної роботи із сайтом, простого та комфортного доступу до каталогу товарів за рахунок оптимальної візуалізації вмісту; підвищення продажу компанії, за рахунок надання зручного для сервісу як і для користувача так і для замовника сервісу.

Електронна комерція робить торгівлю більш гнучкою та стандартизованою, тому актуальність програмного продукту визначається великим попитом на подібні сервіси.

Список ключових слів: ІНТЕРНЕТ-МАГАЗИН, САЙТ, REACT, JAVASCRIPT, HTML, СМАРТФОН.

ABSTRACT

Explanatory note: 77 pp., 18 figs., 3 appendices, 20 sources.

Object of development: web-oriented application for the sale of smartphones.

The purpose of the qualification work: development of an online store for the sale of smartphones using a modern JavaScript-library React.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the task.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, chooses the choice of platform for development, design and development of the program, describes the algorithm and structure of the application, determines the input and output data, provides characteristics of the parameters of technical means, describes the call and download site, describes the Internet-shop.

The economic section determines the complexity of the developed software product, calculates the cost of work to create an application and calculates the time to create it.

The practical significance lies in the development of a web-oriented application that will provide site visitors with the most convenient work with the site, easy and comfortable access to the product catalog through optimal visualization of content; increase the company's sales by providing service-friendly for both the user and the customer service.

E-commerce makes trade more flexible and standardized, so the relevance of the software product is determined by the high demand for such services.

List of keywords: ONLINE STORE, SITE, REACT, JAVASCRIPT, HTML, SMARTPHONE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ГІК - графічний інтерфейс користувача;

ІС - інформаційна система;

ОС - операційна система;

ООП - об'єктно-орієнтоване програмування;

ПЗ - програмне забезпечення;

JS - мова програмування JavaScript;

ПК - персональний комп’ютер

ВСТУП

На цьому сучасному етапі розвитку суспільства комп’ютери стали засобом комунікації. Комп'ютери використовуються скрізь, від звичайного соціального життя до бізнесу та навчання. Наприклад, у бізнесі це необхідно, щоб більше людей дізналось про послуги які надає компанія, або про компанію в цілому.

У звичайному житті ми використовуємо інтернет, щоб залишатися на зв’язку з рідними та друзями, дізнаватись новини, навчатись, робити покупки та багато чого іншого. Інтернет створює нові принципи життя суспільства, в які входить зручність, швидкість та комфорт. Зараз, щоб придбати товар який забажаєш, достатньо мати доступ до інтернету. В сучасний час майже кожен магазин має або сторінку в інтернеті з описом того чим займається, або ж повноцінні інтернет-магазини в яких можливо придбати абсолютно всю продукцію компанії яка є в фізичних відділеннях, а зазвичай і навіть більше. Тому, такі інтернет-платформи створюють величезну конкуренцію звичайним торговельним точкам.

Один з найпопулярніших товарів, що продається в сучасному світі це – смартфони. Майже кожна сучасна людина має смартфон без якого навіть не уявляє своє життя, тому що майже всі люди використовують його щодня для спілкування, навчання, розваг и багато чого іншого. За даними аналітичної компанії Canalys, у першому кварталі 2022 року обсяг поставок смартфонів на глобальному ринку склав 311,2 млн. пристроїв. Це на 11% менше, ніж за аналогічний період минулого року. Найпопулярнішими брендами в світі смартфонів вважаються такі технічні гіганти як Apple, Samsung та Xiaomi. Насправді ж виробників набагато більше, але саме ці три виробника займають

більшу частину ринку сучасних смартфонів на даний час, наприклад тільки за перший квартал 2022 року Корейський гігант зміг реалізувати 73,7 млн пристроїв, а компанія Apple завдяки високому попиту на серію iPhone 13 і виходу нового iPhone SE змогла продати 56,5 млн. смартфонів.

Темою кваліфікаційної роботи є розробка веб-орієнтованого додатку з продажу смартфонів, оскільки на даний час інтернет-магазини з продажу смартфонів є одними з найпопулярніших в просторах інтернету та користаються попитом у розробників.

В результаті виконання кваліфікаційної роботи створено веб-орієнтований додаток з зручним інтерфейсом за допомогою якого замовник зможе продавати свій товар не тільки у фізичних точках, а і в інтернеті.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Веб-сайт є одним із найпопулярніших способів зв'язку клієнтів компанії, з якою вони хочуть мати справу. Різні функції веб-сайтів можуть зробити компанію унікальною. Зараз можна купувати товари в Інтернеті та навчатися, дивитися фільми, працювати та багато чого іншого, що дуже сильно полегшило життя багатьом людям.

Веб-сайт розробляються за допомогою мов веб-програмування. Мови програмування є технологією, яка стоїть за створенням кожної сторінки в Інтернеті, які використовуються в багатьох організаціях по всьому світу. Задача будь-якої мови програмування міститься в описі доступним способом вмісту сторінки. Всі вони орієнтовані під конкретні запити користувачів і розробників, які займаються створенням веб-сайту. Кожна із мов програмування відрізняється своїми плюсами і мінусами.

Потреби, складність і функціональність веб-сайтів розширюються разом із запитами користувачів Інтернету та розширенням бізнес-цілей. Мови веб-програмування еволюціонували від однієї технології до іншої. Ця еволюція була зумовлена зростаючою схильністю більшості людей до доступу до Інтернету. Інтернет – це глобальна платформа, де різні люди та організації збираються разом, щоб обмінюватися ідеями, інформацією та ресурсами.

Інтернет-магазини стали популярним способом покупок з тих пір, як Інтернетом почали користуватися люди. Є багато людей, яким потрібно щось придбати, але в них немає часу або можливості дійти до магазину, тому для і були створенні інтернет-магазини, щоб зробити покупки набагато

доступнішими. Шопінг завжди асоціювався з довгими черги, які тягнуться, проблеми з паркуванням і клопоти їздити від магазину до магазину поки нарешті знайдемо те, що нам потрібно.

Інтернет-магазини дають людям свободу робити покупки де завгодно навіть вдома на дивані. Інтернет-магазини зробили шопінг набагато легше та доступніше, як і для покупців так і для магазинів. Тому що, чим зручніше буде спосіб придбання товарів для покупців тим більше зможе магазин продати.
Види інтернет-магазинів за товарами, що представлені в каталозі можна класифікувати за такими видами:

- інтернет-вітрина;
- інтернет магазин;
- онлайн аукціон.

Інтернет-вітрина – це скоріше рекламна сторінка, що розповідає за товар. На вітрині викладають інформацію про товари, яку постійно оновлюють. Витрати її створення і адміністрування може бути досить низькими, а практична користь такої вітрини зазвичай велика. Потенційний покупець, завітавши до вітрини, може ознайомитись з товаром, зателефонувати на фірму для уточнень, оплатити товар, домовитися про доставку. Тому інтернет-вітрина використовують тоді, коли покупця треба ознайомити зі складною продукцією, на вивчення якої в звичайному торговому залі піде дуже багато часу.

Інтернет-магазин містить зазвичай каталог - багаторівневе логічне дерево розділів та підрозділів, усередині яких будуть розміщені списки з картками товарів. Він має бути добре структурованим і логічним, щоб покупці змогли легко знайти саме те, що їм справді потрібно. Про сам товар покупці можуть ознайомитись на картці товару, що містить його опис, ознаку наявності у продажу та ціну.

Онлайн-аукціон, або інтернет-аукціон - це аукціон, що проводиться за допомогою Інтернету. Він є видом інтернет-магазину, тому що має у своєму функціоналі прийом онлайн-платежів. Момент закінчення інтернет-аукціону наперед призначається самим продавцем при постановці товару на торги. Після завершення інтернет-аукціону покупець повинен переказати гроші продавцю за безготівковим розрахунком або готівкою при отриманні товару, якщо продавець надає таку можливість, а продавець зобов'язаний надіслати товар покупцю поштою, нерідко в будь-яку точку країни проведення або всього світу. Зазвичай, куди можуть надіслати товар, продавець має вказати наперед.

Кожен інтернет-магазин має виглядати привабливо щоб потенційному клієнту було приємно на ньому перебувати та робити покупки. Дизайн інтернет-магазину - це вигляд сайту, що впливає на перше враження і, звичайно, на продажі магазину. Якщо дизайн добре створений та інтуїтивно зрозумілий то користувачам буде простіше робити покупки, а також довго і із задоволенням розглядати сторінки інтернет-магазину, що цілком вигідно компанії.

Багато популярних компаній роками тестують розташування елементів на своїх сторінках, щоб зрозуміти, як користувачу буде зручніше працювати з їхнім сайтом. Тому більшість інтернет-платформ повторює приблизно однаковий дизайн і отримує гарні продажі. Є сервіси, які намагаються виділитися, але креативні ідеї іноді лякають користувачів - і компанії доводиться зрештою повертатися до основ та перевірених методів у дизайні.

Дизайн інтернет-магазину повинен чіпляти та зацікавлювати, тобто завдання розробника сайту або дизайнера - створити таку композицію із кольору, шрифту та розташування потрібних кнопок з привабливим

контекстом , щоб плавно привести відвідувача інтернет-магазину до покупки товару в інтернет-магазині.

Також важливо показувати на головній сторінці свої ключові товари. Цей прийом дуже ефективний, тому що не завжди відвідувачі приходять на сайт, точно знаючи, що саме їм потрібно. Часто, це абстрактний набір характеристик, якими вони підбирають те, що сподобається. А показавши на головній свої найкращі товари у всій красі, ви в такий спосіб можете значно спростити їхній вибір.

Потенційний покупець однозначно звертає увагу на опис товару, тому на сайті обов’язково має буди інформація про всі товари які подані в зручній для покупця формі.

1.2. Призначення розробки та галузь застосування

Як об'єкт розробки системи відображення і адміністрування інтернет-магазину смартфонів розглядається веб-додаток інтернет-магазин «Moby box», в якому користувач може знайти популярні смартфони та передивитися каталог товарів з функціями сортування за категоріями. Розроблений інтернет-магазин може бути використано при організації продажів смартфонів або інших товарів які мають схожі характеристики.

Розроблена інформаційна система призначена для:

- Підвищення продажів компанії;
- надання для ознайомлення клієнтів з товарами;
- забезпечення відвідувачам сайту простоти і комфортності доступу до каталогу товарів.

1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випусковою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 122 «Комп'ютерні науки»
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 268 від 18.05.2022 р;
- завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованого додатку з продажу смартфонів».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є розробка веб-орієнтованого додатку інтернет магазин з продажу смартфонів з зручним та інтуїтивно-зрозумілим інтерфейсом. Згідно з сучасних досліджень, більше половини користувачів інтернету проглядають сторінки самі через смартфони, тому сайт має буди адаптивним під різні пристрої такі як смартфон, планшет, так ПК.

Створення й розробка додатку повинно включати наступні етапи:

- визначення цілей створення веб-додатку та затвердження завдання на розробку;
- аналіз предметної області і постановка задачі;
- розробка програмного коду, компонентів та інших елементів необхідних у проекті;

- заповнення інформаційної системи контентом;

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблений додаток повинен відповідати наступним умовам:

- дизайн сайту має бути виконано в світлій кольоровій палітрі;
- інтерфейс має бути інтуїтивно зрозумілим для користувача;
- мати такі сторінки як: головна, каталог, сторінка товару, сторінка улюбленого та сторінка замовлення;
- мати можливість відображати тільки товари тих брендів, що обере користувач;
- мати фільтрування за ціною;
- відображення сторінок для різних девайсів;
- відправляти інформацію щодо замовлення на пошту для опрацювання;
- відправляти покупцю на електрону пошту лист про те що замовлення прийняте.

1.5.2. Вимоги до інформаційної безпеки

Безпека веб-додатків - це захисні заходи, за яких зловмисник не зможе отримати доступ до конфіденційних даних як ззовні при спробі злому, так і всередині компанії через нелегітимний доступ. В даному проекті данні користувачів не зберігаються, тому для захищеності сайту достатньо того, що сайт був розміщений на безпечному та надійному хостингу.

1.5.3. Вимоги до складу та параметрів технічних засобів

Веб-сайт інтернет-магазину для використання зі сторони користувача потребує лише неперервного з’єднання з інтернетом та наявність встановленого браузера.

Для комфортного використання веб інтерфейсу, повинно використовуватися наступні вимоги до технічного засобу на якому використовується веб інтерфейс.

- процесор з частотою не менше 2.4Ghz;
- жорсткий диск 200 Gb;
- операційна система: Microsoft Windows XP/7/8/10, Linux, MacOS;
- оперативна пам'ять 4 Gb або більше;
- веб браузер: Chrome 60+, Safari 10+ / iOS Safari 10+, Edge 12+, Firefox ESR+, Internet Explorer 11. , Opera;
- клавіатура;
- комп'ютерна миша;
- доступ до мережі Internet.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог висунутими замовником.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування веб додатку потрібно мати:

- операційна система Unix, Linux, Microsoft Windows XP/7/8/10, оперативна система Mac OS, від Mac OS X Cheetah до самої останньої macOS Monterey;
- браузер Інтернет (Google Chrome, MozillaFireFox, Microsoft Internet Explorer, Opera).

Frontend частина додатку має бути реалізована на мові програмування JavaScript з використанням фреймворку React JS та бібліотеки Redux.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення програми

За завданням кваліфікаційної роботи було реалізовано розробку front-end частини системи інтернет магазину смартфонів. Призначення розробленої системи:

- ознайомити потенційного клієнта з асортиментом компанії;
- легке та доступне керування інтернет магазину за рахунок простого коду;
- формування web-сторінок;
- оформлення замовлень з даного магазину;
- надання потенційним покупцям веб-сайту вичерпної інформації про надані товари;
- забезпечення користувачам сайту простого та комфортного доступу до каталогу товарів.

2.2. Опис застосованих математичних методів

Оскільки особливості вказаної у цій роботі предметної області не передбачають застосування математичних методів, при розробці веб-орієнтованого додатку для покращення продажів в сфері комерції математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Front-end частина web-додатку реалізована на мові програмування JavaScript з використанням фреймворку React JS та бібліотек Redux, Formik.

HTML (мова розмітки гіпертексту) і CSS (каскадні таблиці стилів) є двома основними технологіями для створення веб-сторінок. HTML надає структуру сторінки, CSS — (візуальний і звуковий) макет для різних пристроїв.

HTML - це мова, якою написані більшість веб-сайтів. HTML використовується для створення сторінок і надання їх функціональності.

HTML (Hyper Text Markup Language) вперше був створений Тімом Бернерсом-Лі, Робертом Кайо та іншими, починаючи з 1989 року [20].

Гіпертекст означає, що документ містить посилання, які дозволяють читачеві перейти до інших місць документа або взагалі до іншого документа. Остання версія відома як HTML5.

Мова розмітки – це спосіб, за допомогою якого комп’ютери спілкуються один з одним, щоб контролювати, як текст обробляється та представлений. Для цього HTML використовує дві речі: теги та атрибути.

CSS розшифровується як Cascading Style Sheets, і він використовується для додавання стилю до веб-сторінки, диктуючи спосіб відображення сайту в браузері. CSS унікальний тим, що не створює жодних нових елементів, як-от HTML або JavaScript. Натомість це мова, яка використовується для стилізації елементів HTML.

JavaScript — це мова програмування, яка дозволяє створювати інтерактивну веб-сторінку та відповідати на дії користувача.

Послідовність інструкцій (програма, сценарій або скрипт) виконується інтерпретатором, вбудованим у звичайний браузер. Це означає, що код можна

вставити в HTML-код програми і виконати на стороні клієнта. Вам не слід перезавантажувати веб-сторінку, щоб запустити програму, усі програми працюють у відповідь на невідому ситуацію. Наприклад, перед подачею форм даних ви можете переглянути самі значення і, якщо товар не відповідає вашим очікуванням, заборонити подачу інформації.

JavaScript зазвичай використовується як мова, створена для програмного забезпечення доступу до програмного забезпечення. Більшу частину часу в браузерах він використовується як мова письма для інтерактивності сторінки. Примітки щодо архітектури:

- динамічне введення тексту;
- автоматичне управління пам'яттю;
- прототип комп'ютерного програмування;
- функціонує як об'єкт першого класу.

JavaScript було переміщено на багато мов, і метою було зробити мову, схожу на Java, але не для простого у використанні програмного забезпечення. У JavaScript є кілька відомих скоординованих мов об’єктів, але прототип, створений цією мовою, спричиняє відмінності в операційних об’єктах порівняно з традиційними мовами операцій наборів. Крім того, JavaScript має кілька рядків функцій, властивих іншим мовам – функції, такі як об’єкти першого класу, такі об’єкти, як таблиці, дефіси, функції анонімного блокування – що забезпечує додаткову гнучкість мови.

React -це відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови вебзастосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux. В даний час React використовують Kha Academy, Netflix, Yahoo, Airbnb, Sony, Atlassian та інші.

Redux - це інструмент для керування станом даних і інтерфейсом користувача в додатках JavaScript з великою кількістю сутностей. Являє собою бібліотеку JavaScript.

Назва читається як "Редакс" і складається з двох слів: reduce та flux. Reduce – це функція, що наводить велику структуру даних одного значення. Flux — це архітектура програми, при якій дані передаються в одну сторону. Інструмент ґрунтується на цих двох поняттях, тому вони винесені у назву.

Зазвичай Redux використовується у зв'язці з фреймворками JavaScript: React, TypeScript, Vue, Angular та іншими. Рідше він потрібний для написання коду на чистому JS. Має відкритий вихідний код та доступний безкоштовно. З усіма залежностями важить лише близько 2 Кб.

Redux потрібен для:

- для керування станом програми, що працює з великою кількістю даних;
- зручної заміни вбудованих засобів роботи зі станом у React;

- легшого масштабування програми, його перетворення під різні завдання;
- позбавлення помилок, пов'язаних з безладом в об'єкті стану;
- передбачуваності та зрозумілості роботи програми;
- простіший налагодження та доопрацювання;
- підвищення продуктивності та працездатності програми.

Formik – бібліотека, яка допомагає працювати з формами. Вона спрощує отримання даних із форми, валідацію даних, виведення повідомлень про помилки та багато іншого.

Всі слайдери на сайті – React компоненти Swiper. Бібліотека постійно оновлюється та підтримується, мінімальна кількість багів. Є різні кейси та види використання слайдерів, що відрізняються лише передачею різних запитів у компонент. Нескладна у використанні, є окрема можливість робити адаптивний (передаючи в пропси брекпоінти і як його видозмінити: скільки слайдів за раз відображати, який відступ між ними робити і т.д.).

Для Емейл розсилки використаний сервіс Emailjs. Простий у використанні client-side сервіс для розсилок, який надає кілька точок для API під капотом своїх функцій. Все, що потрібно зробити, це правильно підключити сервіс і викликати потрібну функцію, якою передаси дані для листа. Шаблони листів створюються не в коді, а в самому сервісі. У функцію тільки передається id цього темплейту на ресурсі, щоб зрозуміти, який шаблон використовувати. У вкладці content - лист, який прийде магазину, у вкладці Auto-Replay - те, що прийде користувачеві.

Для форматування та аналізу коду були використані утіліти Eslint и Prettier.

Eslint - це статичний аналізатор для мови програмування. Він повідомляє про потенційно небезпечні вирази в коді, які можуть призвести до аварійного завершення програми. Також лінтер може повідомити про застарілі ділянки коду, синтаксичні помилки і невикористані змінні. Усього існує близько 300 правил, які можна увімкнути/вимкнути за бажанням.

На великому проекті зазвичай працює кілька програмістів. У кожного з них може бути своя думка щодо якості коду та розміру відступів у рядку. Eslint може форматувати текст, але вимагає додаткової копіткої установки. Полегшити завдання покликана утиліта Prettier. Вона має відкритий вихідний код і має гарну підтримку спільноти. Більшість правил оформлення вже задана за замовчуванням, але пакет має гнучку систему налаштування і зручний API, що дозволяє описувати свої правила в конфігураційному файлі формату .json/.yml. Це означає, що ми можемо написати файл із єдиними налаштуваннями для всієї команди.

2.4. Опис структури системи та алгоритмів її функціонування

Структура сайту – це те, як відвідувач може пересуватися між сторінками та користуватись інтерфейсом сайту. Щоб відвідувач був зацікавлений, структура сайту має буду простою та інтуїтивно-зрозумілою для покупця. Всі користувачі люблять коли на сайті на який вони зайшли одразу зрозуміло що де знаходиться і не потрібно тратити час для знаходження чого небудь.

В розробленому веб-додатку реалізована зручна структура яка дозволяє з кожної сторінки потрапити в будь-який розділ, що економить час користувачу.

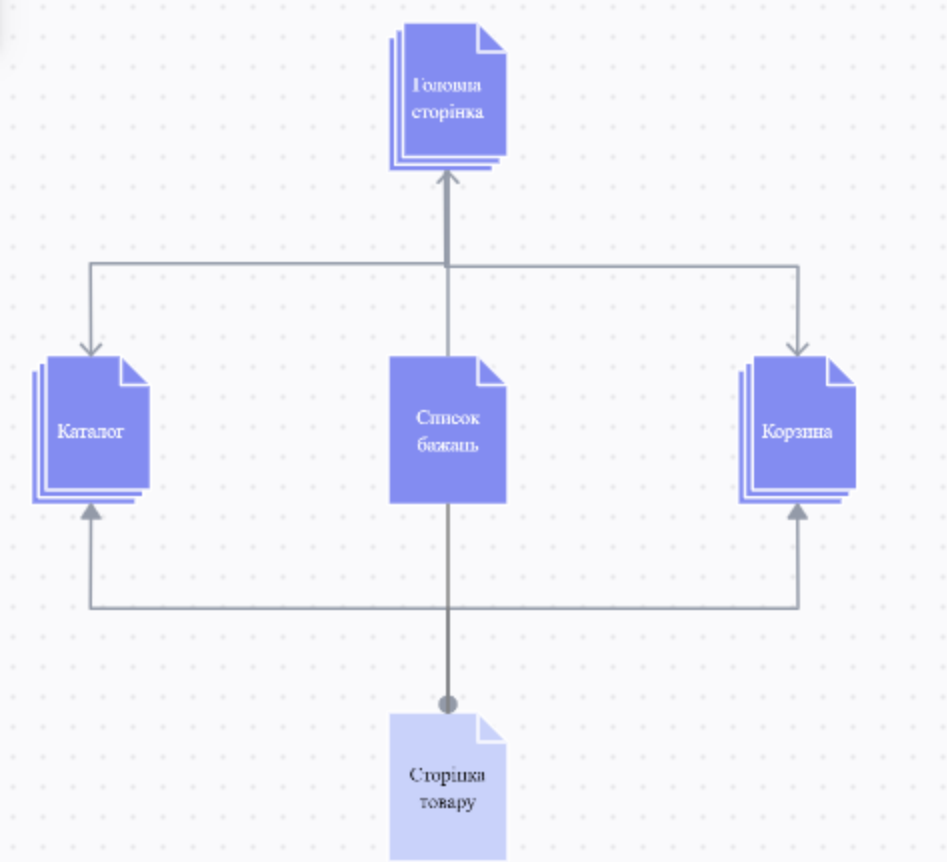

Рис. 2.1. Карта сайту для користувача

Структура інтернет-магазину складається з таких частин:

- головна сторінка з популярними товарами;
- катало товарів з можливістю сортуванням за ціною та брендом;
- сторінка товару з описом;
- сторінка улюбленого;
- сторінка с кошиком;
- сторінка замовлення.

Мінімальна структура додатку на React створюється автоматично після введення команди в консоль create-react-app.

Create-react-app генерує базовий проект для вас, у кореневій директорії якого містяться файли: .gitignore, package.json, README.md, yarn.lock.

Крім того, він створює папки: public та src. У src зберігається наш вихідний код.

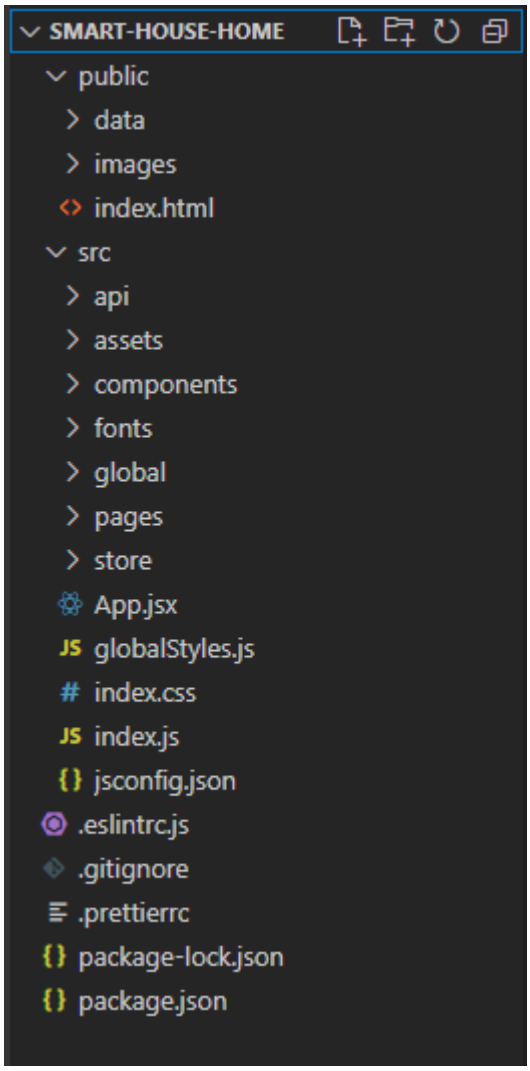

Рис. 2.2. Структура файлової системи

Папка public містить файл стандартний файл HTML, який можно налаштувати під свої потреби, наприклад, щоб встановити назву сторінки. Тег <script> зі скомпільованим кодом буде додано до нього автоматично під час процесу збірки.

В проектах які створюються за допомогою технологій React, в файлі index майже нічого не змінюють, тому він залишається стандартним.

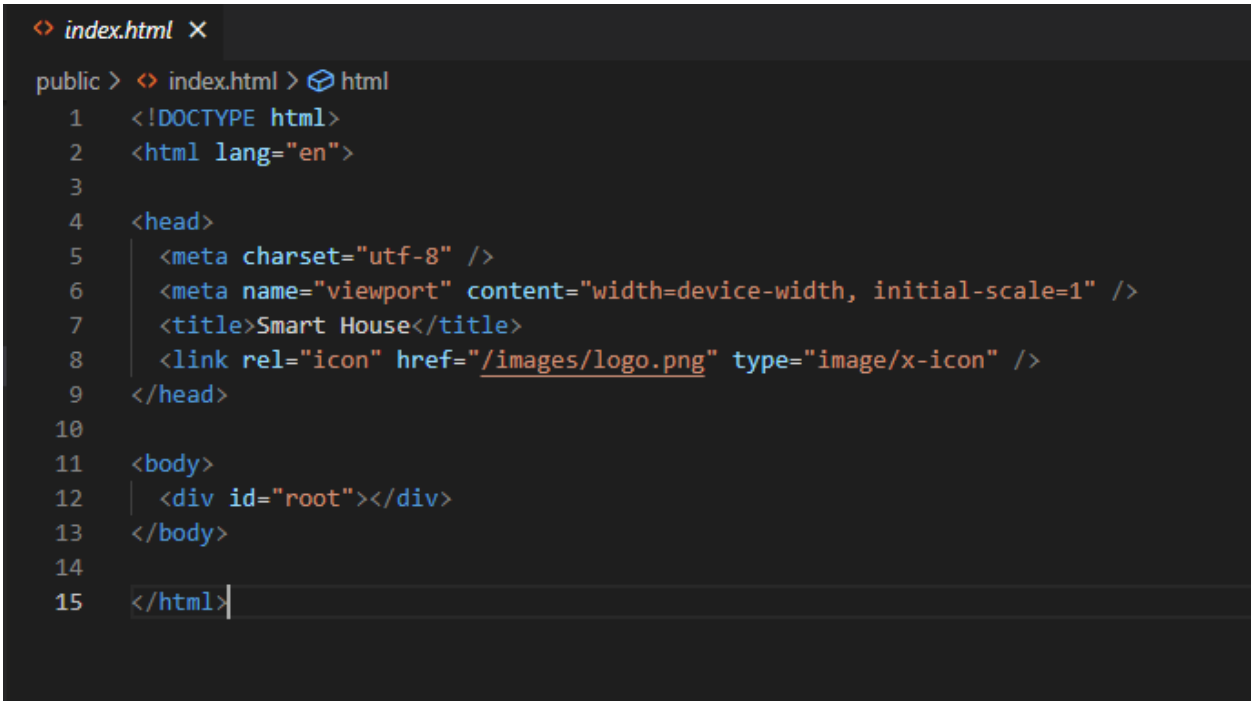

Рис. 2.3. Файл index.html

В папці src зберігається основний код додатку. Папка src має декілько розділів такі як:

- api;
- assets;
- components;
- fonts;
- global;
- pages;
- store.

React дозволяє визначити компоненти як класи або функції. В даний час класові компоненти мають більше можливостей. Вони розібрані на сторінці.

Щоб визначити такий компонент, необхідно успадковуватись від React.Component. в розділі components розміщенні всі головні компоненти з їхніми стилями, які описані в файлі style.js та їх функціоналом в файлі index.jsx.

Файли front-end частини інтернет-магазину поділені на компоненти, що використовуються більше одного разу у додатку та сторінки, що містять окремі папки з власними компонентами.

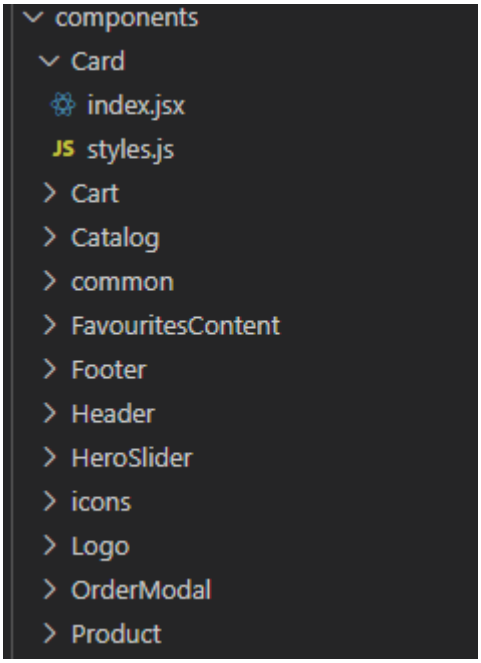

Рис. 2.4. Структура папки components

Компоненти дають змогу розбивати інтерфейс на незалежні частини, про які легко думати окремо. Їх можна складати разом та використати кілька разів, що робить розробку доволі зручною.

Для зручності буи створенні такі частини:

- Card;
- Cart;

- Common;
- FavouritesContent;
- Footer;
- Header;
- HeroSlider;
- Icons;
- Logo;
- OrderModal;
- Product.

Header (шапка сайту) – верхня частина сайту, призначена для полегшення навігації по сторінці, першим елементом, який привертає увагу користувача. За допомогою хедеру сайту відвідувач розуміє, де він знаходиться, як потрапити в наступний розділ і знайти потрібну інформацію. Тому потрібно щоб він був зручним для розуміння та мав привабливий дизайн. Хедер є одним з головних частин сайту, тому що використовується абсолютно на всіх сторінках як і футер.

Footer (підвал сайту) – нижня область на сайті, призначена для логічного завершення сторінки, полегшення навігації та розміщення інформації про фізичне знаходження магазину або контактної інформації з власником або сулжбою підтримки якщо вона є.

Приклади футеру та хедеру наведені нижче [17]:

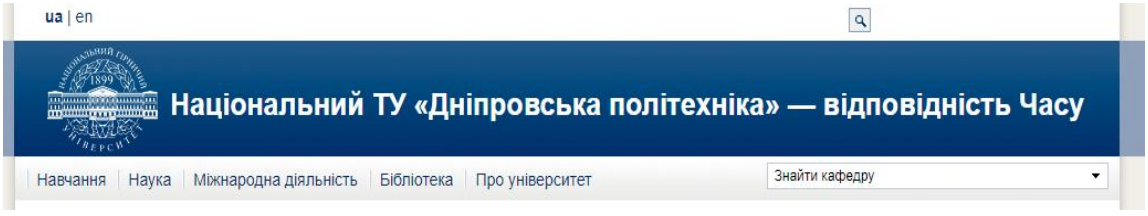

Рис. 2.5. Приклад хедеру

Рис. 2.6. Приклад футера

В папці pages є такі розділи:

- Cart;
- Catalog;
- Favourites;
- HomePage;
- Product.

В кожній з цих папок знаходяться файл index.jsx в якому зібрані всі компоненти разом, що утворюють кожну сторінку сайту.

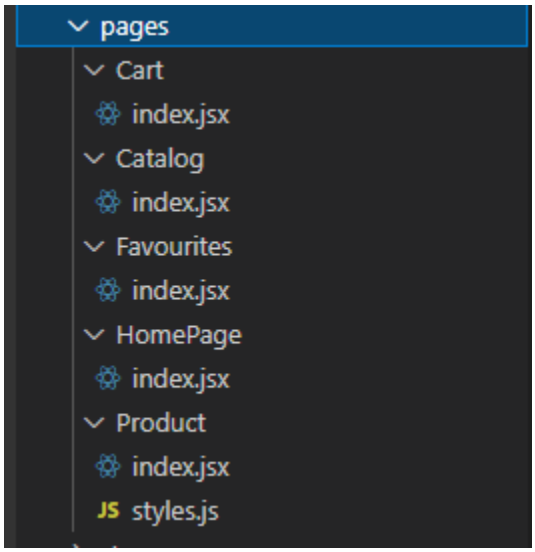

Рис. 2.7. Структура папки pages

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Із вхідних данних використовується графічні і текстові матеріали про товар та інформація, що вводиться користувачем із клавіатури на сайті так як:

- ПІБ;
- вік;
- e-mail;
- номер телефону;
- адреса.

З вихідних данних використовуються сторінки, що виводиться на екран.

2.6. Опис розробленої системи

2.6.1 Використані технічні засоби

Даний проект був створений за допомогою персонального комп’ютера на базі системи Windows 10 Pro з такими характеристиками:

- Процесор - Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz 2.30 GHz;
- Оперативна пам’ять - 6,00 ГБ;
- Тип системи - 64-розрядна операційна система, процесор x64.

2.6.2 Використані програмні засоби

Так як цей проект створений за допомогою мови програмування JavaScript з використанням бібліотеки React JS, то для створення потрібно використовувати IDE для розробки.

IDE (Integrated Development Environment) – це інтегроване середовище розробки (система програмних засобів, використовувана програмістами для розробки програмного забезпечення).

Для написання коду додатка був використан IDE Visual Studio Code. Visual Studio Code (VS Code) – редактор вихідного коду, розроблений Microsoft для Windows, Linux та macOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки веб- та хмарних програм. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису та засобами для рефакторингу. Має широкі можливості для кастомізації: теми користувача, поєднання клавіш і файли конфігурації.

Visual Studio Code був анонсований 29 квітня 2015 компанією Microsoft на конференції Build, і незабаром була випущена бета-версія.

18 листопада 2015 Visual Studio Code був випущений під ліцензією MIT, а вихідний код був опублікований на GitHub. Анонсовано підтримку розширень.

14 квітня 2016 року Visual Studio Code вийшов із стадії бета-тестування. В даному проекті використовується Visual Studio Code version 1.68.

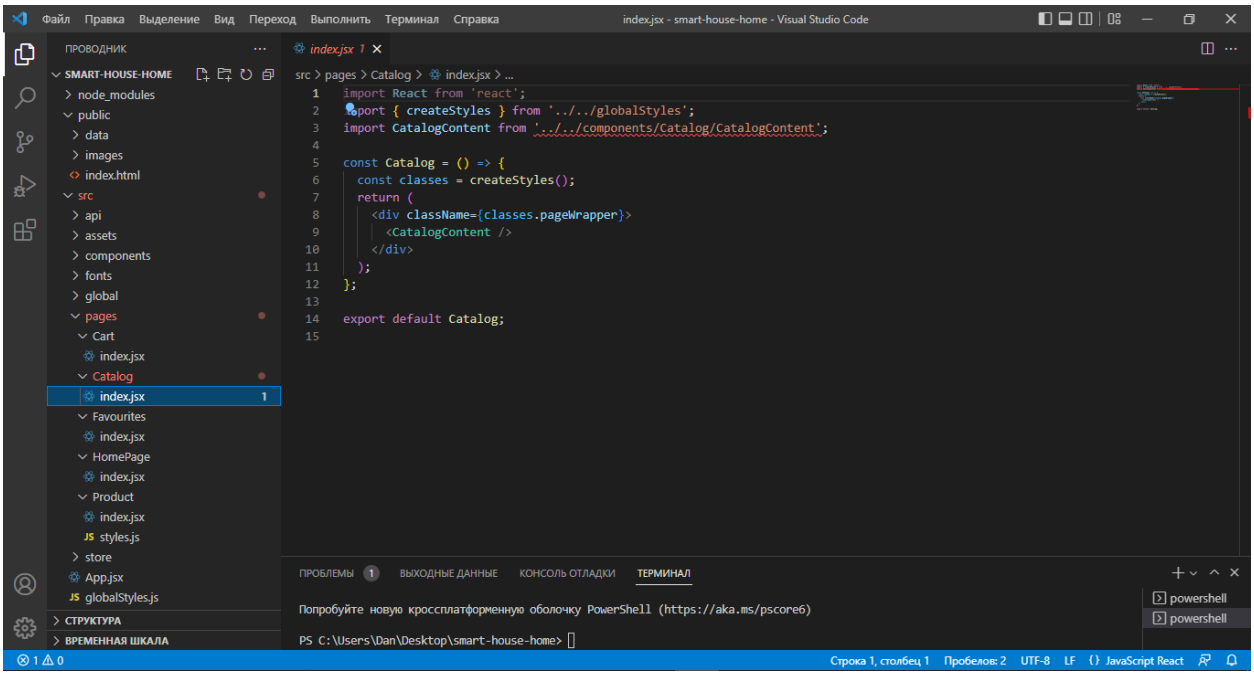

Рис. 2.8. Інтерфейс IDE VS Code

2.6.3. Опис інтерфейсу користувача

Інтерфейс (англ. interface - взаємодія, поєднання) - це місце дотику двох функціональних об'єктів. Якщо говорити вужче, то інтерфейс – це «міст», посередник між людиною, програмами та машинами, іншими системами. Це інструменти взаємодії, за допомогою яких одна система контактує з іншою. За допомогою інтерфейсів ми віддаємо команди програм та пристроїв. Вони їх аналізують, здійснюють необхідні дії та видають відповідь. За допомогою того ж чи іншого інтерфейсу. Ось які завдання вони допомагають нам вирішувати:

- введення команд, направлення запиту;
- отримання відповіді системи у зрозумілій формі (текст, зображення, звук) ;
- обмін інформацією між пристроями, програмами, системами ;
- взаємодія людини та операційної системи ;

- управління програмними засобами, апаратними комплексами ;
- отримання інформації про помилки (порушення алгоритму) та варіанти їх виправлення.

UI (англ. user interface) або інтерфейс користувача. Саме цей тип взаємодії мають на увазі найчастіше, коли згадують про інтерфейс. Він призначений для організації контакту між людиною та програмно-апаратними засобами, компонентами комп'ютерної системи. З його допомогою користувачі взаємодіють з операційними системами та програмами, що знаходяться під їх керуванням. Засобами реалізації інтерфейсу користувача найчастіше виступають такі інструменти, як: клавіатура комп'ютерна миша джойстик дисплей стилус При цьому UI-інтерфейс залежно від операційної системи може бути реалізований в декількох формах. Найпопулярніший – графічний інтерфейс.

Найчастіше для екранного відображення команд та результатів виконання використовуються вікна. Елементи управління програмами відображаються візуально всередині вікон – у формі:

- піктограм;
- кнопок;
- меню;
- списків;
- полів введення та інших.

Приклад графічного інтерфейсу, з яким ми стикаємось щоденно – web-інтерфейс. Це будь-який інтернет-браузер, через який ми читаємо новини, робимо покупки, дивимося погоду та замовляємо піцу. За його допомогою ми взаємодіємо з іншими сайтами – програмами в Інтернеті. Кожен сайт також

має свій інтерфейс, через який система взаємодіє з вами. І це також графічний web-інтерфейс.

Головна мета будь-якого інтерфейсу – забезпечення діалогу, продуктивного контакту функціональних систем, що взаємодіють. Якщо ми говоримо про UI-інтерфейс, з різновидами якого більшості з нас доводиться стикатися щодня, його головне завдання – полегшення для користувача процесу управління програмою чи пристроєм. Розглянемо на прикладі графічного web-інтерфейсу окремого сайту, з яким взаємодіє користувач. Він повинен:

- утримувати оптимальну кількість візуальних елементів управління для вирішення завдання користувача. Їх має бути рівно стільки, скільки потрібно відвідувачу;
- бути адаптивним до десктопних та мобільних пристроїв різних виробників та моделей;
- бути інтуїтивно зрозумілим. Користувач не повинен мати проблеми з керуванням, пошуком тих чи інших категорій меню, розділів. Він повинен користуватися сайтом на основі раніше здобутого досвіду контакту з іншими ресурсами. Елементи управління повинні бути «дохідливими» – якщо користувачі не зможуть здогадатися, як користуватися вашим сайтом, вони будуть спантеличені і розчаровані;
- бути лаконічним. Інакше користувач потрапить у пастку інформаційної перевантаженості. Описи та пояснення для кожної кнопки та елемента сайту – це добре, але від них «пухне» ваш інтерфейс. Користувач не повинен витрачати свій час на прочитання інструкцій ;

- Бути послідовним. Це про вміст сайту. У ньому має простежуватися певний рівень послідовності, який проходить через весь інтерфейс. За її допомогою користувач зможе сформувати шаблони роботи з ресурсом. При першому контакті вони познайомляться з кнопками, закладками, іконками та іншими графічними елементами і при наступній роботі використовуватимуть раніше отриманий досвід;
- Бути ефективним. UI – це транспорт, який має доставити користувача у потрібну точку. Хороший інтерфейс повинен допомагати виконувати функції швидше з мінімальним зусиллям. Щоб інтерфейс був справді ефективним, при його розробці важливо розуміти, що потрібно користувачеві.

Головна сторінка сайту має велике значення для відвідувача і виконує важливі завдання такі як допомога в тому щоб користувач зміг побачити структуру сайту (тобто елементи і інформацію, яку сайт в себе включає), а також зорієнтувати користувача. Головна сторінка, найчастіше, визначає загальне оформлення ресурсу в цілому.

На головній сторінці(Рис 2.10) розробленого веб-сайту користувач може побачити:

- Хедер сайту, на якому присутній логотип сайту, меню, кошик;
- Слайдер з акціями або новими товарами;
- Слайдер з популярними товарами;
- Футер, в якому вказана контактна інформація.

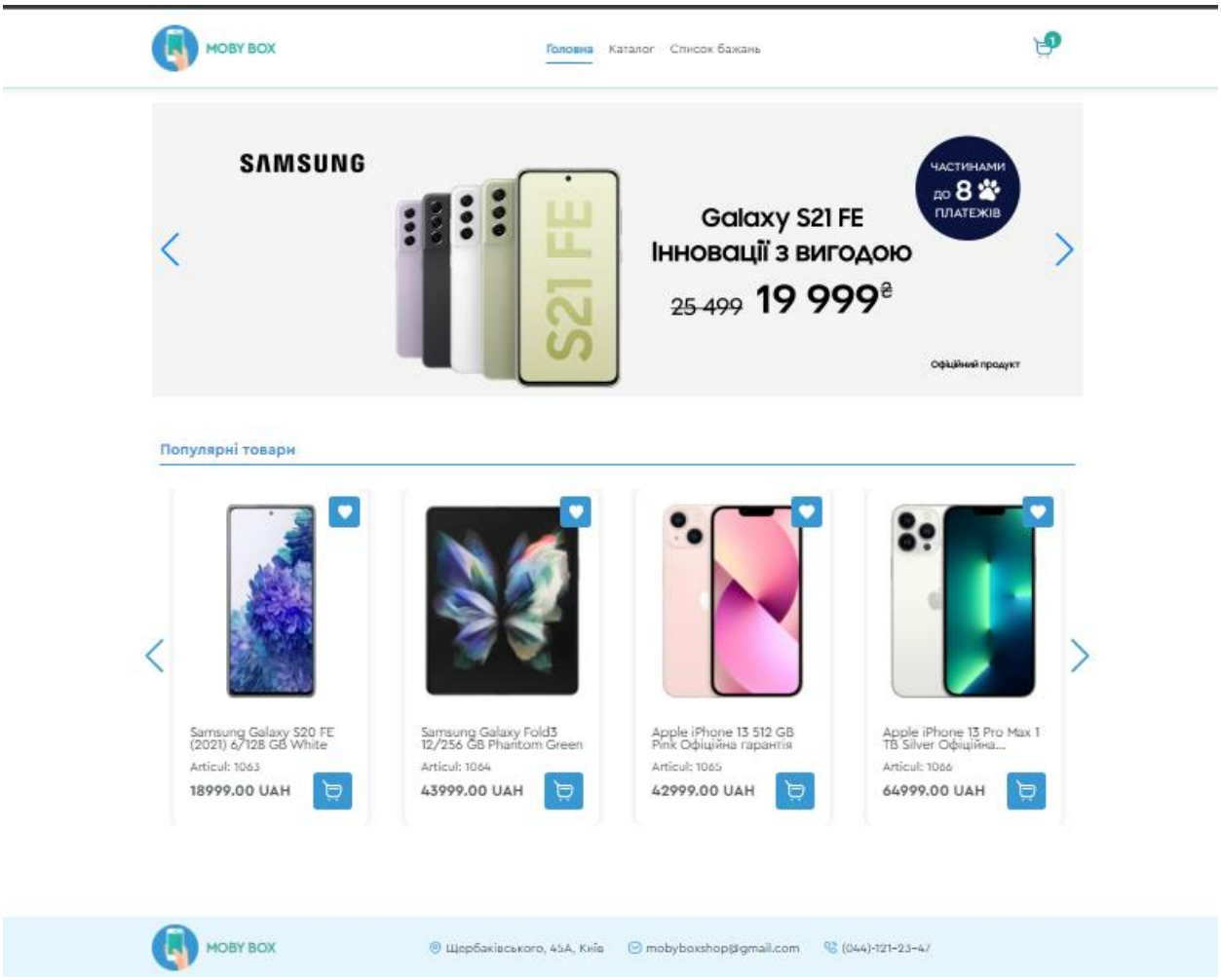

Рис. 2.10. Головна сторінка магазину

Каталог – це також одна з найважливіших сторінок в інтернет-магазині та точка тяжіння для покупців. Від структури каталогу залежить те, наскільки легко відвідувачі будуть знаходити потрібний товар і наскільки загалом зручно та приємно ним користуватиметься сайтом. Тому коли користувач натискає кнопку каталогу, то він бачить перелік смартфонів та форму для фільтрації товару по бренду або по вартості.

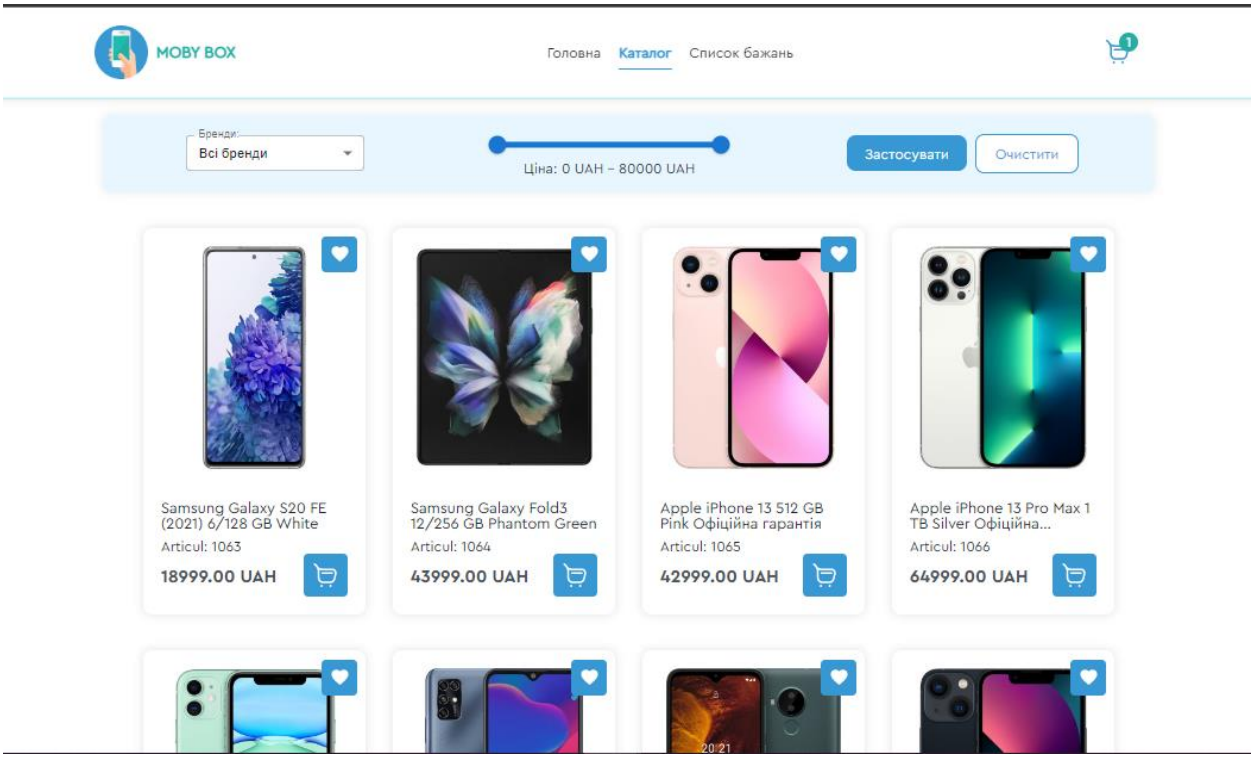

Рис. 2.11. Каталог товарів

Також на сторінці каталогу потенційні покупець може додати товар до улюбленого, або до кошику за допомогою інтуїтивно-зрозумілих значків.

Картка товару для інтернет-магазину — поєднання візуального та текстового контенту, сукупність опису характеристик товару та функціональних. Для того щоб карта товару була приваблива для потенційного покупця, то при розробці використовують такі рекомендації:

- розміщувати назву товару у вірному місці. Назву товару можна розмістити над картинкою або праворуч від неї. Відвідувач швидко буде розуміє, що знайшов те, що хотів;
- вказувати у назві всю важливу інформацію та все, що може бути зрозуміло неправильно. Назва може бути довгою, але так у покупця відразу будуть відповіді на його питання про товар;

- вказувати бренд у назві. Вказівка бренду в заголовку дозволить зіграти на його популярності та привернути увагу кліента.

При створенні сторінки товари, були виконані вишче наведенні умови, для того що потенційному покупцю було зручіше отримувати інформацію про товар який він може потенційно придбати.

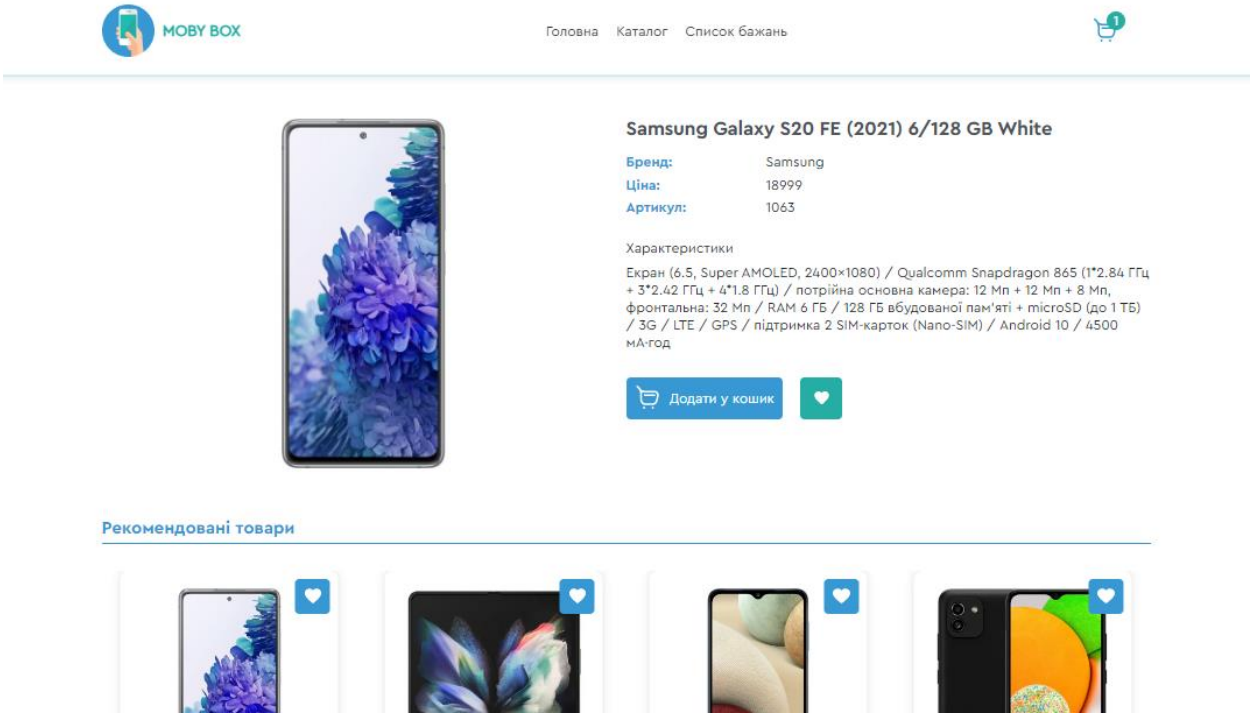

Рис. 2.12. Сторінка товару

Також на сторінці товари внизу присутній слайдер де відвідувач може одразу перейти на сторінки подібних смартфонів. Це зроблено для того щоб зекономити час та зробити більш комфортним перебування в інтернет-магзині.

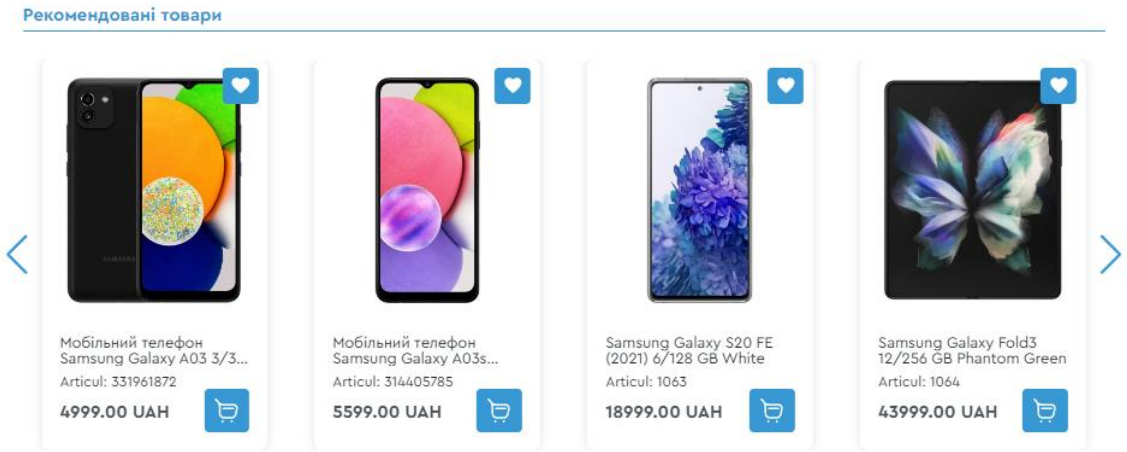

Рис. 2.13. Слайдер рекомендованих товарів

В розробленому проекті присутня сторінка з улюбленими товарами які обрав користувач самостійно за допомогою інтуїтивно зрозумілої позначки на товарі. Користувач може додати при перегляді каталогу в улюблені декілька товарів, що б потім їх було швидко побачити та знов не шукати.

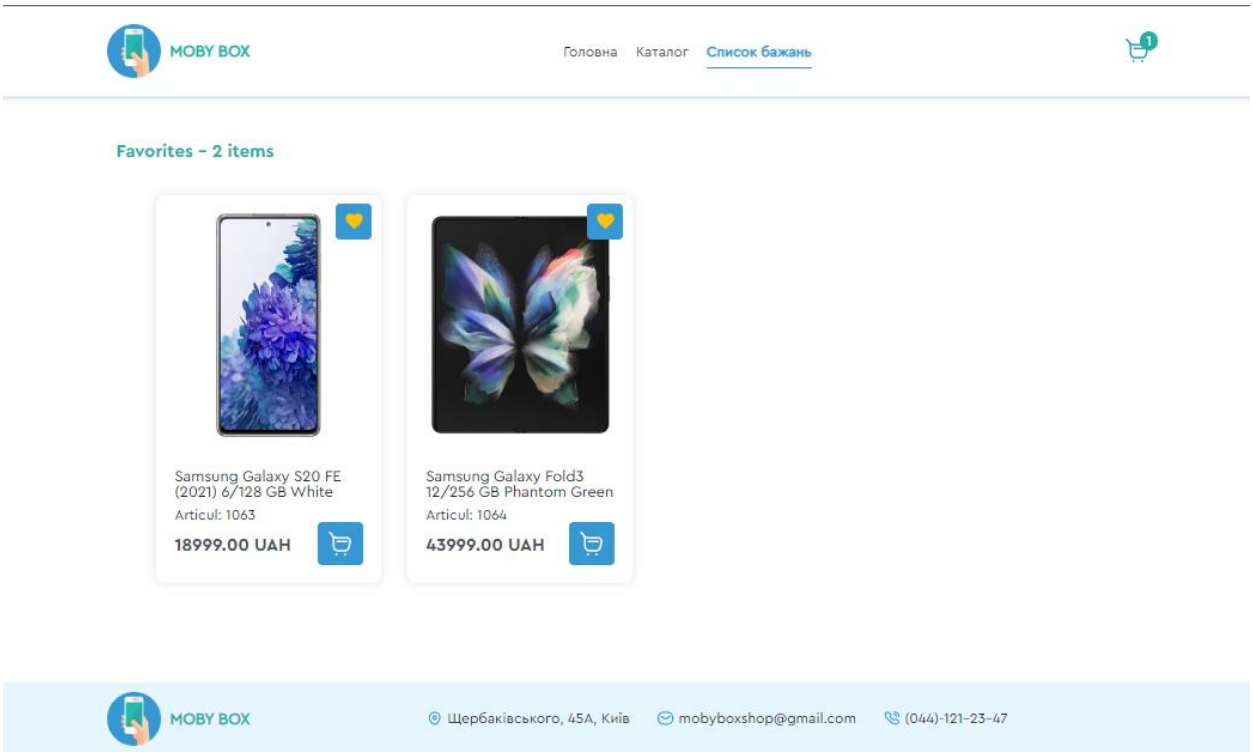

Рис. 2.14. Сторінка з обраними товарами

Кошик інтернет-магазину – функціональна частина веб-сайтів, пов'язаних із продажем товарів у мережі інтернет.

Кошик зазвичай є окремою сторінкою сайту, що виконує функцію інтернет-магазину. Інтернет-користувач попередньо вибирає цікаві для нього товари і додає кожен товар у кошик. Якщо в інтернет-кошик доданий товар, його значок може змінюватися на інший, що повідомляє, що у ньому є товар. Поруч із іконкою кошика може відображатися кількість товарів у кошику. Потім користувач має можливість перейти в кошик, подивитися і при необхідності змінити кількість товарів, а потім оформити замовлення. Кошик зазвичай показує вартість кожного товару окремо, а також загальну суму товару.

В розробленому проекті кошик поділяється на дві частини. В першій частині покупець може перевірити або змінити своє замовлення, ввести промокод, ознайомитись з загальною сумою.

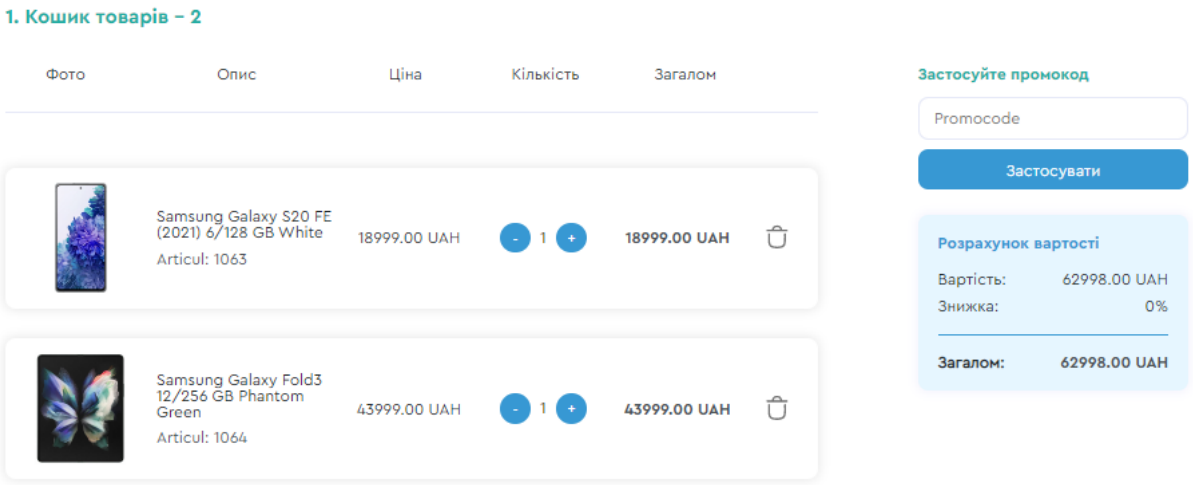

Рис. 2.15. Перша частина сторінки кошик

В другій частині, клієнт заповнює форму в якій вказує:

- ПІБ;
- E-mail;
- Вік;
- Номер телефону;
- Адресу.

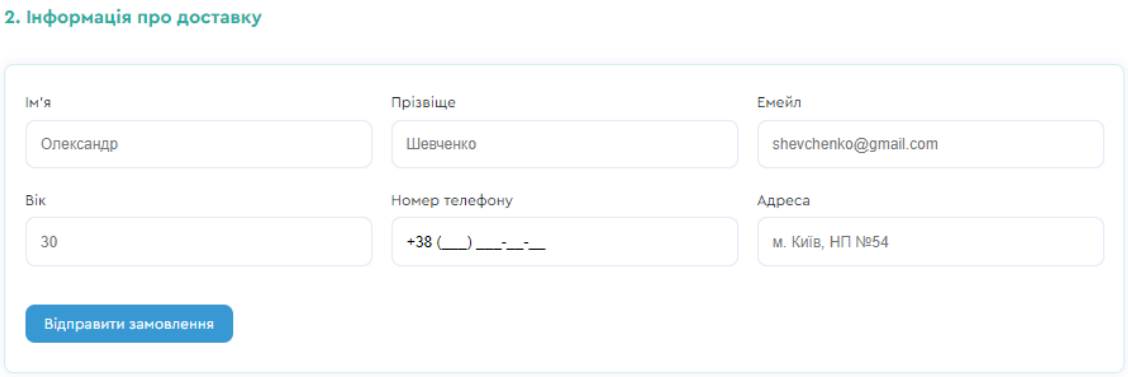

Рис. 2.16. Друга частина сторінки кошик

Після того як покупець перевірив своє замовлення та заповнив форму, то він натискає кнопку відправити замовлення і бачить вікно з інформацією про його замовлення та те що воно прийняте в обробку.

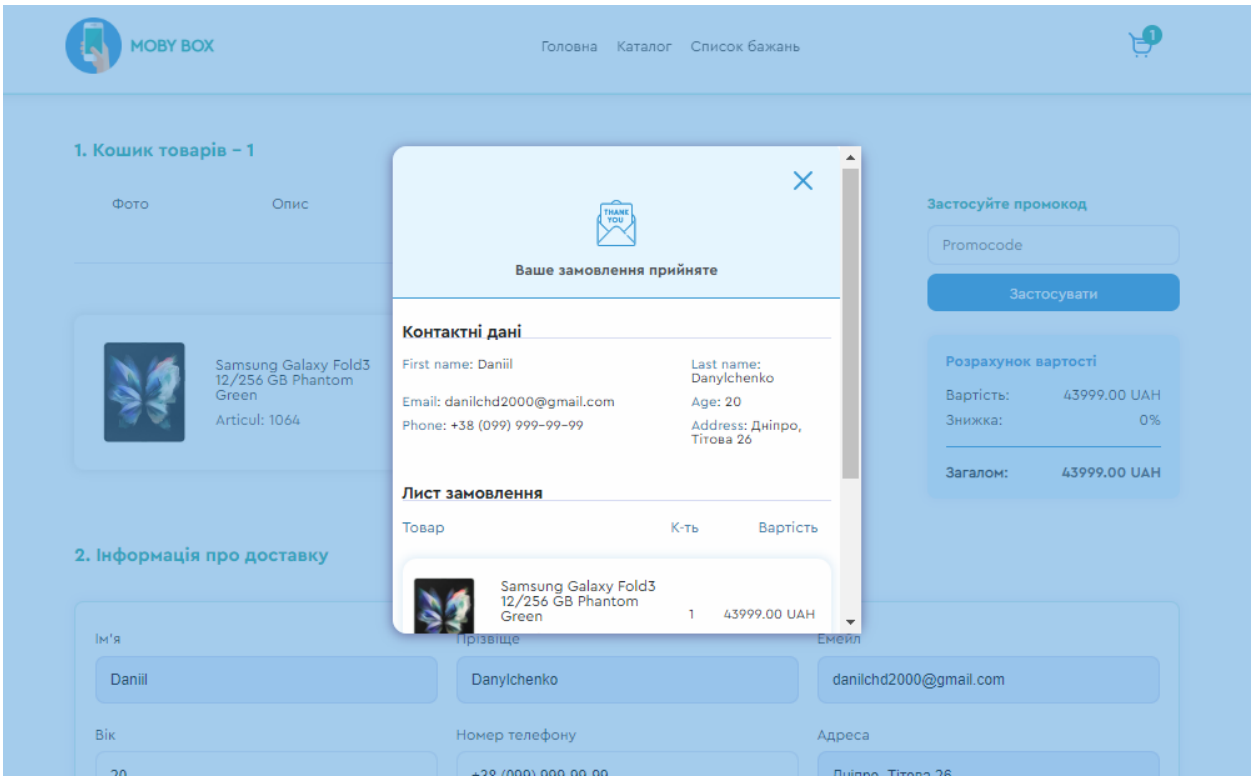

Рис. 2.17. Завершення замовлення

Після замовлення покупець отримує лист на емейл який був вказаний в замовлені про те що запит прийнято, очікуйте на дзвінок від оператора.

В цей час лист з інформацією про замовлення и контактною інформацією покупця находить на електронну адресу інтернет-магазину, для того що б менеджер магазину зміг зв’язатись з покупцем.

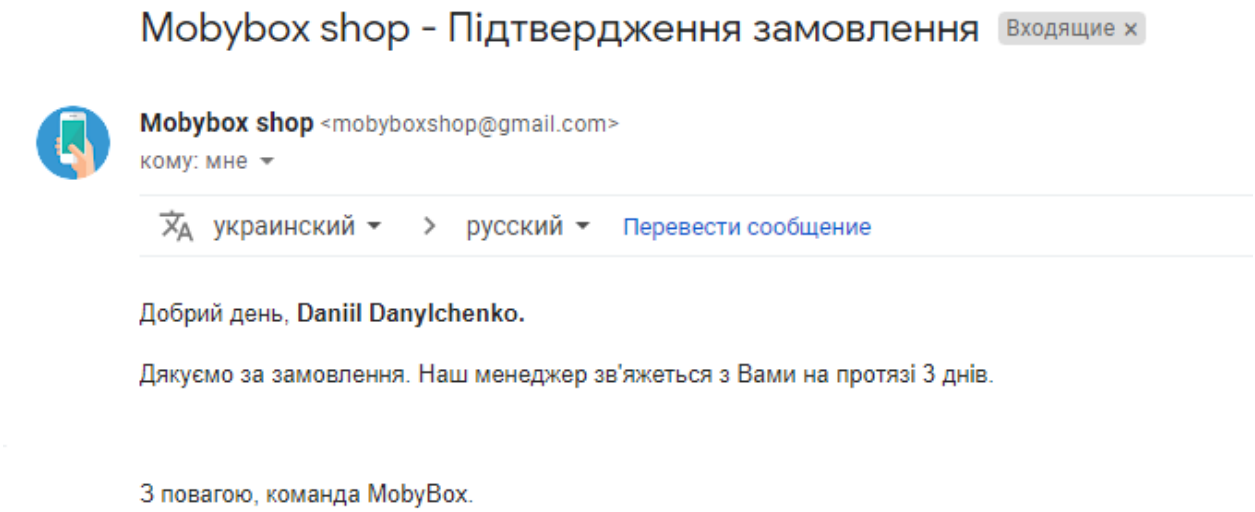

Рис. 2.18. Лист на e-mail покупця

Рис. 2.19. Лист на e-mail магазину

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

- вихідні дані розробки програмного забезпечення:
- передбачуване число операторів – 860;
- коефіцієнт складності програми – 1,2;
- коефіцієнт кореляції програми в ході її розробки - 0,1;
- середня годинна заробітна плата програміста, грн/год – 90;
- вартість машино-години ЕОМ, грн/год – 5.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою:

$t = to + ти + та + тп + тотл + тд$, людино-годин, (3.1)

де to – витрати праці на підготовку й опис поставленої задачі (приймається 50);

ти – витрати праці на дослідження алгоритму рішення задачі;

та– витрати праці на розробку блок-схеми алгоритму;

тп – витрати праці на програмування по готовій блок-схемі;

тотл – витрати праці на налагодження програми на ЕОМ;

тд – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C(1 + p), \quad (3.2)$$

де q – передбачуване число операторів;

С– коефіцієнт складності програми;

р – коефіцієнт кореляції програми в ході її розробки.

$Q = 860 * 1,2 * (1 + 0,1) = 1135$;

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин}, \quad (3.3)$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В = 1.2 ... 1.5;

К – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. до 2 – 0,8;

$$t_u = \frac{1135 \cdot 1,2}{85 \cdot 0,8} = 20, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K}; \quad (3.4)$$

$$t_a = \frac{1135}{25 \cdot 0,8} = 57, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{Q}{(20..25)K}; \tag{3.5}$$

$$t_a = \frac{1135}{25 \cdot 0{,}8} = 57, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K}; \tag{3.6}$$

$$t_{отл} = \frac{1135}{5 \cdot 0{,}8} = 284, \text{ людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл};$$

$$t^{к}_{отл} = 1{,}2 \cdot 284 = 340, \text{ людино-годин} \tag{3.7}$$

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до}; \tag{3.8}$$

де tдр – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20)K}; \tag{3.9}$$

$$t_{др} = \frac{1135}{20 \cdot 0{,}8} = 71, \text{ людино-годин.}$$

tдо – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}; \tag{3.10}$$

$t_{до} = 0{,}75 \cdot 71 = 53$, людино-годин.

$t_{д} = 71 + 53 = 124$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 +20 + 57 +57 + 284 +124 = 592, людино-годин.

В результаті розраховано, що в загальній складності необхідно 592 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн}, \quad (3.11)$$

де ЗЗП – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн}, \quad (3.12)$$

де t – загальна трудомісткість, людино-годин; СПР – середня годинна заробітна плата програміста, грн/година

$$С_{ЗП} = 592 \cdot 90 = 53280, \text{грн.}$$

ЗМВ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}, \text{ грн}, \quad (3.13)$$

де тотл – трудомісткість налагодження програми на ЕОМ, год. СМЧ – вартість машино-години ЕОМ, грн/год

$$З_{МВ} = 283 \cdot 5 = 1419, \text{ грн.}$$

$$\hat{E}_{П} = 53280 + 1419 = 54429, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.} \quad (3.14)$$

де Bk - число виконавців; Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{592}{1 \cdot 176} = 3{,}4 \text{ міс.}$$

Висновок: програмне забезпечення призначене для роботи інтернет-магазину з продажів смартфонів.Таким чином, очікувана тривалість розробки складе 3,4 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення 54429 грн.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи був реалізований інтернет-магазин з продажу смартфонів на мові програмування JavaScript та JavaSript-бібліотеки React. В результаті виконання кваліфікаційної роботи було виконані такі задачі як:

- визначення цілей створення веб-додатку та затвердження завдання на розробку;
- аналіз предметної області і постановка задачі;
- розробка програмного коду, компонентів та інших елементів необхідних у проекті;
- заповнення інформаційної системи контентом;

Створений веб-додаток реалізовує основні можливості інтернет-магазину, а саме: корзина, каталог товарів, фільтрація, оформлення замовлення, улюблене.

Розроблений веб-додаток система призначений для:

- Підвищення продажів компанії;
- надання для ознайомлення клієнтів з товарами;
- забезпечення відвідувачам сайту простоти і комфортності доступу до каталогу товарів.

В економічній частині було підраховано трудомісткость розробки програмного забезпечення та розраховано витрати на створення програмного продукту. Таким чином, в ході виконання кваліфікаційної роботи було досягнуто всіх цілей та виконано всі вимоги, які висувалися при проектуванні веб-додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Васільев А.Н. JavaScript в прикладах та задачах / А.Н. Васільев.– 720 с.
2. Веб-приложение и его виды [Електронний ресурс]//URL: https://semantica.in/blog/veb-prilozhenie.html;
3. Веб-сайт ТУ «Дніпровська Політехніка» [Електронний ресурс] // URL: https://www.nmu.org.ua/ua/;
4. Вікіпедія [Електронний ресурс] // URL: https://ru.wikipedia.org/wiki/HTML;
5. Гото Г., Котлер Э. Web-дизайн. – СПб: Символ Плюс, 2009. – 340 с;
6. Горнаков, С.Г. Освоюємо популярні системи керування сайтом / Горнаков С.Г. – ДМК Пресс, 2019 – С. 336.
7. Документація React [Електронний ресурс] // URL: https://ru.reactjs.org/docs/getting-started.html;
8. Документація swiper [Електронний ресурс] // URL: https://swiperjs.com/get-started;
9. Эрик Фримен, Элизабет Робсон. Head First JavaScript Programming – С;
10. Леонтьев А.А. Web-дизайн. Керівництво користувача. - М.: Центр, 2000;
11. К. Дуглас – Як влаштований JavaScript, 2019 - 394 с;
12. Мардан Азат React швидко.Веб-додаток на React, JSX, Redux и GraphQL;
13. Молер, Дж. Flash 8. Руководство Web-дизайнера; Эксмо - М., 2019;
14. Основы React.js [Електронний ресурс] // URL: https://learn.javascript.ru/screencast/reac;
15. Орлов Л. Як створити електорний магазин в інтернеті, 2-е изд., М.: Бук. пресс, 2006. - 384 с;
16. Стоян Стефанов React. Швидкий старт (2016);
17. Тиленс Томас Марк React в дії Бэнкс Алекс React и Redux: функціональна веб-розробка;
18. Чіртік А.В. Популярний самовчитель HTML, 2012. 56 с;
19. React tutorial [Електронний ресурс] // URL: https://www.w3schools.com/REACT/DEFAULT.ASP;
20. Redax tutorial [Електронний ресурс] // URL: https://redux.js.org/tutorials/essentials/part-1-overview-concepts.

ДОДАТОК А

КОД ПРОГРАМИ

cardsList.json


```
[
  {
    "title": "Samsung Galaxy S20 FE (2021) 6/128 GB White",
    "price": 18999,
    "articul": 1063,
                                                    "url":                "https://res.cloudinary.com/moby-box/image/upload/v1655446312/slides/Products/176610125_zskpuc.jpg",
    "brand": "Samsung",
    "description": "Екран (6.5, Super AMOLED, 2400x1080) / Qualcomm Snapdragon 865 (1*2.84 ГГц + 3*2.42 ГГц + 4*1.8 ГГц) / потрійна основна камера: 12 Мп + 12 Мп + 8 Мп, фронтальна: 32 Мп / RAM 6 ГБ / 128 ГБ вбудованої пам'яті + microSD (до 1 ТБ) / 3G / LTE / GPS / підтримка 2 SIM-карток (Nano-SIM) / Android 10 / 4500 мА·год"
  },
  {
    "title": "Samsung Galaxy Fold3 12/256 GB Phantom Green",
    "price": 43999,
    "articul": 1064,
                                                    "url":                "https://res.cloudinary.com/moby-box/image/upload/v1655447048/slides/Products/201656236_ybyfdx.jpg",
    "brand": "Samsung",
     "description":"Екран (6.2, Dynamic AMOLED 2X, 2268x832) + (7.6, Dynamic AMOLED 2X, 2208x1768) / Qualcomm Snapdragon 888 (2.84 ГГц + 2.4 ГГц + 1.8 ГГц) / потрійна основна камера: 12 Мп + 12 Мп + 12 Мп, фронтальна камера 10 Мп / допоміжна фронтальна 4 Мп / RAM 12 ГБ / 256 ГБ вбудованої пам'яті / 3G / LTE / 5G / GPS / підтримка 2 SIM-карток (Nano-SIM) + eSIM / Android 11 / 4400 мА·год"
  },
  {
    "title": "Apple iPhone 13 512 GB Pink Офіційна гарантія",
    "price": 42999,
    "articul": 1065,
                                                    "url":                "https://res.cloudinary.com/moby-box/image/upload/v1655448237/slides/Products/221205342_lwj3on.jpg",
    "brand": "Apple",
     "description":"Екран (6.1, OLED (Super Retina XDR), 2532x1170) / Apple A15 Bionic / потрійна основна камера: 12 Мп + 12 Мп + 12 Мп, фронтальна камера: 12 Мп / 1 ТБ вбудованої пам'яті / 3G / LTE / 5G / GPS / Nano-SIM / iOS 15"
  },
  {
```

```
  "title": "Apple iPhone 13 Pro Max 1 TB Silver Офіційна гарантія",
  "price": 64999,
  "articul": 1066,
                                                    "url":                "https://res.cloudinary.com/moby-box/image/upload/v1655448312/slides/Products/221298339_jvcd23.jpg",
  "brand": "Apple",
   "description": "Екран (6.1, OLED (Super Retina XDR), 2532x1170) / Apple A15 Bionic / потрійна основна камера: 12 Мп + 12 Мп + 12 Мп, фронтальна камера: 12 Мп / 1 ТБ вбудованої пам'яті / 3G / LTE / 5G / GPS / Nano-SIM / iOS 15"
 },
 {
  "title": "Apple iPhone 11 128 GB Green Slim Box (MHDN3)",
  "price": 22999,
  "articul": 1067,
                                                    "url":                "https://res.cloudinary.com/moby-box/image/upload/v1655448417/slides/Products/37406402_gzcorf.jpg",
  "brand": "Apple",
  "description":"Екран (6.1, IPS (Liquid Retina HD), 1792x828) / Apple A13 Bionic / основна подвійна камера: 12 Мп + 12 Мп, фронтальна камера: 12 Мп / RAM 4 ГБ / 128 ГБ вбудованої пам'яті / 3G / LTE / GPS / ГЛОНАСС / Nano-SIM / iOS 13 / 3046 мА·год Зарядним блоком і навушниками не комплектується."
 }
]

promocodesList.json
[
  {
    "code": "promocode",
    "discountInPercentage": 5
  },
  {
    "code": "gtgtt90gt2",
    "discountInPercentage": 10
  },
  {
    "code": "half",
    "discountInPercentage": 50
  },
  {
    "code": "korgyland",
    "discountInPercentage": 50
  }
] [
 {
```

```
    "title": "Прямуй за своїм вибором",
                                                    "url":                    "https://res.cloudinary.com/moby-
box/image/upload/v1655402814/slides/270388484_zhyv25.jpg"
  },
  {
    "title": "Смартфони Samsung - інновації з вигодою",
                                                    "url":                    "https://res.cloudinary.com/moby-
box/image/upload/v1655402813/slides/266839514_pye7gs.jpg"
  },
  {
    "title": "Квадракоптери для ЗСУ",
   "url": "https://res.cloudinary.com/moby-box/image/upload/v1655402813/slides/1066400_frswkk.webp"
  },
  {
    "title": "Залишайся на зв'язку, купуй смартфон",
                                                    "url":                    "https://res.cloudinary.com/moby-
box/image/upload/v1655402814/slides/1366x469_jbgdkg.webp"
  }
]
index.html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Smart House</title>
  <link rel="icon" href="/images/logo.png" type="image/x-icon" />
</head>

<body>
  <div id="root"></div>
</body>

</html>
export const getCardsList =  async () => {
   const response = await fetch('../data/cardsList.json');
    if (!response.ok) {
        throw new Error(`Error - ${response.status}`);
    }
    return response.json();
}

export const getPromocodesList = async () => {
```

```
  const response = await fetch('../data/promocodesList.json');
  if (!response.ok) {
    throw new Error(`Error -  ${response.status}`);
  }
  return response.json();
};

export const getPromos = async () => {
  const response = await fetch('../data/slides.json');
  if (!response.ok) {
    throw new Error(`Error -  ${response.status}`);
  }
  return response.json();
};
```

index.jsx

```
import React from 'react';
import { useDispatch } from 'react-redux';
import { Link } from 'react-router-dom';
import PropTypes from 'prop-types';
import FavouriteIcon from '../icons/FavouriteIcon';
import Button from '../common/Button';
import { addToCart } from '../../store/cart/actions';
import { createStyles } from './styles';
import { colors } from '../../global/theme/colors';
import { CartIcon } from '../icons';

const NOIMGSRC = 'img/notfound.png';

const Card = ({ title, price, articul, url, changeFavouriteHandler, isFavourite }) => {
  const classes = createStyles();
  const dispatch = useDispatch();

  const addToCartHandler = () => {
    dispatch(addToCart(articul));
  };

  return (
    <li className={classes.item}>
      <Link to={`/product/${articul}`}>
        <img className={classes.itemImg} src={url || NOIMGSRC} alt={title} />
      </Link>
      <div className={classes.cardInfo}>
        <h2 className={classes.title}>{title}</h2>
```

```
      <p>Articul: {articul}</p>
      <Button
        className={classes.favouriteBox}
        onClick={() => {
          changeFavouriteHandler(articul);
        }}
      >
        <FavouriteIcon width="20px" height="20px" stroke="" fill={isFavourite ? '#ffc107' : '#fff'} />
      </Button>
      <div className={classes.cardFooter}>
        <p className={classes.cardPrice}>{price.toFixed(2)} UAH</p>
        <Button onClick={addToCartHandler} className={classes.cartBtn}>
          <CartIcon fill={colors.white} width="30" height="28" />
        </Button>
      </div>
    </div>
  </li>
 );
};

Card.propTypes = {
 title: PropTypes.string.isRequired,
 price: PropTypes.number.isRequired,
 articul: PropTypes.number.isRequired,
 url: PropTypes.string,
 changeFavouriteHandler: PropTypes.func.isRequired,
 isFavourite: PropTypes.bool,
};

Card.defaultProps = {
 url: '',
 isFavourite: false,
};
export default Card;

import { createUseStyles } from 'react-jss';
import { colors } from '../../global/theme/colors';

const styles = {
 item: {
   borderRadius: '10px',
   boxShadow: '0px 0px 10px 1px hsl(0deg, 0%, 82%, 49%)',
   position: 'relative',
   boxSizing: 'border-box',
```

```
  width: '250px',
},
itemImg: {
  display: 'block',
  width: '250px',
  height: '250px',
  objectFit: 'contain',
  margin: '0 auto',
  paddingTop: '20px',
},
cardInfo: {
  padding: '20px',
  marginTop: '15px',
},
title: {
  fontSize: '17px',
  color: colors.neutral,
  marginBottom: '10px',
},

cardPrice: {
  color: colors.grey,
  fontFamily: 'CeraPro Bold',
  fontWeight: '700',
  fontSize: '20px',
},

favouriteBox: {
  borderRadius: '5px',
  zIndex: 2,
  position: 'absolute',
  width: '40px',
  height: '40px',
  right: '10px',
  top: '10px',
  backgroundColor: colors.blue,
  '& svg': {
    width: '23px',
    height: '23px',
    position: 'absolute',
    top: '8px',
    right: '8px',
  },
},
```

```
  isFavourite: {
    '& path': {
      fill: '#ffc107',
    },
  },
  cartBtn: {
    padding: '10px',
    borderRadius: '5px',
  },
  cardFooter: {
    display: 'flex',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
};

const createStyles = createUseStyles(styles);
export { createStyles };

import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import PropTypes from 'prop-types';
import { decreaseProductQuantity, increaseProductQuantity, removeFromCart } from
'../../../store/cart/actions';
import { createStyles } from './styles';
import { TrashIcon } from '../../icons';

const NOIMGSRC = 'img/notfound.png';
const CardInCart = ({ articul, url, title, price, isPriceShow, hasQuantityBtns, hasTrashIcon,
orderModalStyles }) => {
  const classes = createStyles();
  const cardsInCart = useSelector((state) => state.cardsInCart);
  const currentCard = cardsInCart.find(({ id }) => id === articul);
  const dispatch = useDispatch();

  const decrementHandler = () => {
    dispatch(decreaseProductQuantity(articul));
  };

  const incrementHandler = () => {
    dispatch(increaseProductQuantity(articul));
  };
```

```
  const removeCard = () => {
    dispatch(removeFromCart(articul));
  };

  return (
    <li className={orderModalStyles ? classes.checkoutItem : classes.item}>
      <img src={url || NOIMGSRC} alt={title} className={orderModalStyles ? classes.checkoutImg : classes.img} />
      <div className={orderModalStyles ? classes.checkoutItemInfo : classes.itemInfo}>
        <h3 className={classes.itemTitle}>{title}</h3>
        <p className={classes.articul}>Articul: {articul}</p>
      </div>
      {isPriceShow ? <p className={classes.price}>{price.toFixed(2)} UAH</p> : ''}
      <p className={orderModalStyles ? classes.checkoutQuantity : classes.quantity}>
        {hasQuantityBtns ? (
          <button type="button" className={`${classes.quantityBtn} ${classes.decrement}`} onClick={decrementHandler}>
            -
          </button>
        ) : (
          ''
        )}
        <span>{currentCard.count}</span>
        {hasQuantityBtns ? (
          <button type="button" className={`${classes.quantityBtn} ${classes.increment}`} onClick={incrementHandler}>
            +
          </button>
        ) : (
          ''
        )}
      </p>
      <p className={orderModalStyles ? classes.checkoutTotal : classes.total}>
        {(currentCard.count * price).toFixed(2)} UAH
      </p>
      {hasTrashIcon ? (
        <button type="button" className={classes.deleteItemBtn} onClick={removeCard}>
          <TrashIcon width="25" height="25" fill="none" stroke="#727272" />
        </button>
      ) : (
        ''
      )}
    </li>
  );
```

```
};

CardInCart.propTypes = {
  articul: PropTypes.number.isRequired,
  url: PropTypes.string,
  title: PropTypes.string.isRequired,
  price: PropTypes.number.isRequired,
  isPriceShow: PropTypes.bool,
  hasQuantityBtns: PropTypes.bool,
  hasTrashIcon: PropTypes.bool,
  orderModalStyles: PropTypes.bool,
};

CardInCart.defaultProps = {
  url: '',
  isPriceShow: true,
  hasQuantityBtns: true,
  hasTrashIcon: true,
  orderModalStyles: false,
};

export default CardInCart;

import { createUseStyles } from 'react-jss';
import { colors } from '../../../global/theme/colors';

const styles = {
  item: {
    display: 'grid',
    gridTemplateColumns: '120px 1.5fr repeat(3, 1fr) 50px',
    columnGap: '2%',
    borderRadius: '10px',
    boxShadow: '0px 0px 10px 1px hsl(0deg, 0%, 82%, 49%)',
    minHeight: '80px',
    padding: '15px',
    alignItems: 'center',
    textAlign: 'center',
    '@media (max-width: 725px)': {
      gridTemplateColumns: '120px 2fr 50px',
      gridTemplateAreas: '"img info deleteBtn" "count totalPrice totalPrice"',
    },
    '@media (max-width: 450px)': {
      gridTemplateColumns: '80px 1fr 1fr 0.1fr',
      columnGap: '3%',
```

```
    gridTemplateAreas: '"img info info deleteBtn" "count count totalPrice totalPrice"',
  },
},
img: {
  maxWidth: '100%',
  height: '110px',
  objectFit: 'cover',
  alignSelf: 'center',
  justifySelf: 'center',
  display: 'block',
  '@media (max-width: 725px)': { gridArea: 'img', marginBottom: '20px' },
  '@media (max-width: 450px)': { width: '80px', height: '80px' },
},
itemTitle: {
  marginTop: '0',
  marginBottom: '10px',
  fontWeight: 500,
  '@media (max-width: 450px)': { fontSize: '14px' },
},
priceOne: {
  color: colors.grey,
},
articul: {
  color: colors.grey,
},
price: {
  fontWeight: 500,
  '@media (max-width: 725px)': { marginTop: '0', display: 'none' },
},

total: {
  color: colors.grey,
  fontFamily: 'CeraPro Bold',
  '@media (max-width: 725px)': {
    gridArea: 'totalPrice',
    justifySelf: 'right',
  },
},
deleteItemBtn: {
  width: '30px',
  height: '30px',
  padding: '0',
  justifySelf: 'center',
  cursor: 'pointer',
```

```
  backgroundColor: 'transparent',
  '@media (max-width: 725px)': { gridArea: 'deleteBtn' },
  '&:hover path': { transition: '0.5s', stroke: 'rgb(245, 66, 66)' },
},
trashIcon: {
  width: '25px',
  height: '25px',
  objectFit: 'contain',
  transition: '0.5s',
},
quantity: {
  '@media (max-width: 725px)': { gridArea: 'count' },
  '@media (max-width: 450px)': { justifySelf: 'start' },
},
quantityBtn: {
  width: '30px',
  height: '30px',
  backgroundColor: colors.blue,
  borderRadius: '50%',
  cursor: 'pointer',
  color: colors.white,
  '&:hover': { backgroundColor: colors.lightblue },
},
decrement: { marginRight: '10px' },
increment: { marginLeft: '10px' },
itemInfo: {
  textAlign: 'left',
  '@media (max-width: 725px)': { gridArea: 'info' },
  '@media (max-width: 450px)': { fontSize: '14px' },
},
checkoutItem: {
  display: 'grid',
  gridTemplateColumns: '80px 1.5fr min-content 1fr',
  columnGap: '20px',
  borderRadius: '15px',
  padding: ['20px', '5px 10px'],
  justifyItems: 'center',
  alignItems: 'center',
  boxShadow: '0px 0px 10px 1px #3798d32e',
  '@media (max-width: 478px)': {
    gridTemplateColumns: '1.5fr 0.5fr 1fr',
    gridTemplateAreas: "'img count total' 'info count total'",
    rowGap: '10px',
    columnGap: '10px',
```

```
    },
  },
  checkoutImg: {
    // width: '80px',
    height: '80px',
    // borderRadius: '15px',
    '@media (max-width: 478px)': { gridArea: 'img', justifySelf: 'left' },
  },
  checkoutItemInfo: {
    justifySelf: 'start',
    '& p': {
      margin: '5px 0',
      fontSize: '14px',
    },
    '& h3': {
      marginBottom: '5px 0',
      fontSize: '16px',
      '@media (max-width: 478px)': { fontSize: '14px' },
    },
  },
  checkoutQuantity: {
    '@media (max-width: 478px)': { gridArea: 'count', fontSize: '14px' },
  },
  checkoutTotal: {
    '@media (max-width: 478px)': { gridArea: 'total', fontSize: '14px' },
  },
};

const createStyles = createUseStyles(styles);
export { createStyles };

import React, { useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { fetchCardsList } from '../../../store/cards/actions';
import Loader from '../../common/Loader';
import { CartForm } from '../CartForm';
import CartList from '../CartList/CartList';
import OrderTotals from '../OrderTotals';
import { createStyles } from './styles';

const CartContent = () => {
  const isLoading = useSelector(({ cards }) => cards.isLoading);
  const cardsList = useSelector(({ cards }) => cards.cards);
  const cardsInCart = useSelector((state) => state.cardsInCart);
```

```
const hasError = useSelector((state) => state.hasError);
const dispatch = useDispatch();

const classes = createStyles();

useEffect(() => {
  dispatch(fetchCardsList());
}, []);

let content;

if (hasError) {
  content = <div>Вибачте,щось пішло не так</div>;
} else {
  const filteredCards = cardsList.filter(({ articul }) => cardsInCart.find(({ id }) => articul === id));
  content = (
    <>
      <CartList cards={filteredCards} />
      <div className={classes.totalBlock}>
        <OrderTotals />
      </div>
      <CartForm />
    </>
  );
}

/* eslint no-return-assign: "error" */

return (
  <div className={classes.cartSection}>
    <div className={classes.container}>
      <h2 className={classes.cartTitle}>1. Кошик товарів - {cardsInCart.length}</h2>
      {isLoading ? <Loader /> : ''}
      {cardsInCart.length >= 1 ? (
        <div className={classes.cartInner}>
          <ul className={classes.listTitles}>
            <li>Фото</li>
            <li>Опис</li>
            <li>Ціна</li>
            <li>Кількість</li>
            <li>Загалом</li>
          </ul>
          {content}
        </div>
```

```
      ) : (
        (content = <p className={classes.noItemsTitle}>У кошику поки немає товарів</p>)
      )}
    </div>
  </div>
 );
};

export default CartContent;

import { createUseStyles } from 'react-jss';
import { colors } from '../../../global/theme/colors';

const styles = {
 container: {
  maxWidth: '1200px',
  margin: '0 auto',
  padding: '0 10px',
 },
 cartInner: {
  display: 'grid',
  gridTemplateColumns: '3fr 1fr',
  gridTemplateAreas: '"listTitles totalBlock" "list totalBlock" "orderForm orderForm"',
  columnGap: '100px',
  rowGap: '20px',
  '@media (max-width: 1180px)': {
   columnGap: '55px',
   gridTemplateColumns: '2.6fr 1fr',
  },
  '@media (max-width: 1080px)': {
   gridTemplateAreas: '"listTitles" "list" "totalBlock" "orderForm"',
   gridTemplateColumns: 'minmax(300px, 1fr)',
   rowGap: '40px',
  },
  '@media (max-width: 725px)': {
   gridTemplateAreas: '"list" "totalBlock" "orderForm"',
  },
 },

 totalBlock: {
  gridArea: 'totalBlock',
  ' @media (max-width: 1080px)': {
   display: 'grid',
   gridTemplateColumns: '1fr 1fr',
```

```
    columnGap: '12%',
  },
  ' @media (max-width: 840px)': {
    gridTemplateColumns: '1.2fr 1fr',
    columnGap: '5%',
  },
  '@media (max-width: 725px)': {
    gridTemplateColumns: 'minmax(min-content, 450px)',
    rowGap: '40px',
    columnGap: 0,
    justifyContent: 'center',
  },
},

cartTitle: {
  marginBottom: '40px',
  color: colors.green,
  fontWeight: 700,
  fontSize: '20px',
  fontFamily: 'CeraPro Bold',
  '@media (max-width: 725px)': {
    textAlign: 'center',
    marginBottom: '30px',
  },
},

listTitles: {
  gridArea: 'listTitles',
  display: 'grid',
  gridTemplateColumns: '120px 1.5fr repeat(3, 1fr) 50px',
  textAlign: 'center',
  columnGap: '3%',
  justifyContent: 'space-between',
  marginBottom: '34px',
  borderBottom: '1px solid #586ece26',
  paddingBottom: '30px',

  '@media (max-width: 1080px)': {
    marginBottom: 0,
  },
  '@media (max-width: 725px)': {
    display: 'none',
  },
},
```

```
  cartSection: { padding: '30px 0' },
  noItemsTitle: {
    fontWeight: 500,
    fontSize: '18px',
    color: colors.neutral,
  },
};

const createStyles = createUseStyles(styles);
export { createStyles };

import React from 'react';
import { Field, ErrorMessage } from 'formik';
import PropTypes from 'prop-types';
import { createStyles } from '../styles';

export const FormikInputBlock = ({ id, name, placeholder, type, label }) => {
  const classes = createStyles();
  return (
    <div key={id} className={classes.fieldContainer}>
      <label className={classes.orderLabel} htmlFor={name}>
        {label}
      </label>
          <Field  className={classes.orderInput}  key={id}  name={name}  placeholder={placeholder}
type={type} />
      <ErrorMessage component="p" className={classes.fieldError} name={name} />
    </div>
  );
};

FormikInputBlock.propTypes = {
  id: PropTypes.string,
  name: PropTypes.string.isRequired,
  placeholder: PropTypes.string,
  type: PropTypes.string,
  label: PropTypes.string.isRequired,
};

FormikInputBlock.defaultProps = {
  placeholder: '',
  type: 'text',
  id: '',
};
```

```
import React from 'react';
import { ErrorMessage, useField } from 'formik';
import NumberFormat from 'react-number-format';
import PropTypes from 'prop-types';
import { createStyles } from '../styles';

export const NumberFormatInputBlock = ({ name, label, id, placeholder, type }) => {
  const classes = createStyles();
  const [field] = useField(name);

  return (
    <div key={id} className={classes.fieldContainer}>
      <label className={classes.orderLabel} htmlFor={name}>
        {label}
      </label>
      <NumberFormat
        {...field}
        className={classes.orderInput}
        format="+38 (###) ###-##-##"
        allowEmptyFormatting
        mask="_"
        name={name}
        placeholder={placeholder}
        type={type}
      />
      <ErrorMessage component="p" className={classes.fieldError} name={name} />
    </div>
  );
};

NumberFormatInputBlock.propTypes = {
  id: PropTypes.string,
  name: PropTypes.string.isRequired,
  placeholder: PropTypes.string,
  type: PropTypes.string,
  label: PropTypes.string.isRequired,
};

NumberFormatInputBlock.defaultProps = {
  placeholder: '',
  type: 'text',
  id: '',
};
```

```
/* eslint react/prop-types: 0 */
/* eslint react/no-unused-prop-types: 0 */
import React, { useState } from 'react';
import { useDispatch } from 'react-redux';
import { Formik, Form, FieldArray } from 'formik';
import { BasicFormSchema } from './BasicFormSchema';
import { formDataFields } from './formDataFields';
import { FormikInputBlock } from './formFields/FormikInputBlock';
import { NumberFormatInputBlock } from './formFields/NumberFormatInputBlock';
import { createStyles } from './styles';
import Button from '../../common/Button';
import { OrderModal } from '../../OrderModal';
import { setCheckoutModalShow, setModalClose } from '../../../store/modal/actions';
import { checkoutOrder, removeDiscount } from '../../../store/cart/actions';

export const CartForm = () => {
  const classes = createStyles();
  const dispatch = useDispatch();
  const [values, setvalues] = useState(null);

  const handleFormSubmit = (info) => {
    setvalues(info);
    dispatch(setCheckoutModalShow(true));
  };

  const closeModalHandler = () => {
    dispatch(setModalClose(false));
    dispatch(checkoutOrder());
    dispatch(removeDiscount());
  };
  return (
    <>
      <div className={classes.formContainer}>
        <h1 className={classes.cartTitle}>2. Інформація про доставку</h1>

        <Formik
          initialValues={{
            firstName: '',
            lastName: '',
            email: '',
            age: '',
            phone: '',
            address: '',
```

```
        }}
        validationSchema={BasicFormSchema}
        onSubmit={handleFormSubmit}
      >
        {({ isSubmitting }) => (
          <Form className={classes.form}>
            <FieldArray
              name="fields"
              render={() => (
                <>
                  <div className={classes.formInner}>
                    {formDataFields.map(({ id, name, label, placeholder, type }) =>
                      name === 'phone' ? (
                        <NumberFormatInputBlock key={id} name={name} type={type} label={label} />
                      ) : (
                                  <FormikInputBlock key={id} name={name} type={type} label={label}
placeholder={placeholder} />
                      ),
                    )}
                  </div>

                  <Button htmlType="submit" className={classes.submitBtn} disabled={isSubmitting}>
                    Відправити замовлення
                  </Button>
                </>
              )}
            />
          </Form>
        )}
      </Formik>
    </div>
    <OrderModal
      closeModalHandler={closeModalHandler}
      header="Ваше замовлення прийняте"
      closeButton
      formValues={values || null}
    />
  </>
 );
};

import React from 'react';
import PropTypes from 'prop-types';
import CardInCart from '../CardInCart/CardInCart';
```

```
import { createStyles } from './styles';

const CartList = ({ cards }) => {
  const classes = createStyles();
  const cardsComponents = cards.map(({ title, price, articul, url }) => (
    <CardInCart key={articul} title={title} price={price} articul={articul} url={url} />
  ));

  return <ul className={classes.list}>{cardsComponents}</ul>;
};

CartList.propTypes = {
  cards: PropTypes.arrayOf(
    PropTypes.shape({
      title: PropTypes.string.isRequired,
      price: PropTypes.number.isRequired,
      articul: PropTypes.number.isRequired,
      url: PropTypes.string,
      brand: PropTypes.string,
      description: PropTypes.string,
    }),
  ),
};

CartList.defaultProps = {
  cards: [],
};

export default CartList;

import React from 'react';
import PropTypes from 'prop-types';
import CardInCart from '../CardInCart/CardInCart';
import { createStyles } from './styles';

const CheckoutList = ({ cards }) => {
  const classes = createStyles();
  const cardsComponents = cards.map(({ title, price, articul, url }) => (
    <CardInCart
      key={articul}
      title={title}
      price={price}
      articul={articul}
      url={url}
```

```
      hasQuantityBtns={false}
      hasTrashIcon={false}
      isPriceShow={false}
      orderModalStyles
    />
  ));

  return <ul className={classes.products}>{cardsComponents}</ul>;
};

CheckoutList.propTypes = {
  cards: PropTypes.arrayOf(
    PropTypes.shape({
      title: PropTypes.string.isRequired,
      price: PropTypes.number.isRequired,
      articul: PropTypes.number.isRequired,
      url: PropTypes.string,
      brand: PropTypes.string,
      description: PropTypes.string,
    }),
  ),
};

CheckoutList.defaultProps = {
  cards: [],
};

export default CheckoutList;

import { createUseStyles } from 'react-jss';

const styles = {
  products: {
    display: 'grid',
    gridTemplateColumns: '1fr',
    alignItems: 'center',
    rowGap: '15px',
    fontSize: '15px',
  },
  item: {
    display: 'grid',
    gridTemplateColumns: '80px 1.5fr min-content 1fr',
    columnGap: '20px',
    borderRadius: '15px',
```

```
    padding: ['20px', '5px 10px'],
    backgroundColor: '#d3cdf94d',
    justifyItems: 'center',
    alignItems: 'center',
    boxShadow: '0px 0px 10px 1px hsl(245deg, 76%, 97%)',
    '@media (max-width: 478px)': {
      gridTemplateColumns: '1.5fr 0.5fr 1fr',
      gridTemplateAreas: "'img count total' 'info count total'",
      rowGap: '10px',
      columnGap: '10px',
    },
  },
  itemInfo: {
    justifySelf: 'start',
    '& p': {
      margin: '5px 0',
      fontSize: '14px',
    },
    '& h3': {
      marginBottom: '5px 0',
      fontSize: '16px',
      '@media (max-width: 478px)': { fontSize: '14px' },
    },
  },
  itemName: {
    margin: '0',
    '@media (max-width: 478px)': { gridArea: 'name' },
  },
  itemArticul: {
    margin: '0',
    gridArea: 'articul',
    color: 'rgb(171, 171, 171)',
  },
  img: {
    width: '80px',
    height: '80px',
    borderRadius: '15px',
    '@media (max-width: 478px)': { gridArea: 'img', justifySelf: 'left' },
  },
  quantity: { '@media (max-width: 478px)': { gridArea: 'count', fontSize: '14px' } },
  total: { '@media (max-width: 478px)': { gridArea: 'total', fontSize: '14px' } },
};

const createStyles = createUseStyles(styles);
```

```
export { createStyles };

import React, { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchCardsList } from '../../store/cards/actions';
import { removeFavourites, addToFavourites } from '../../store/favourites/actions';
import CardsList from '../Catalog/CardsList';
import Loader from '../common/Loader';
import { createStyles } from './styles';

const FavouritesContent = () => {
  const classes = createStyles();

  const isLoading = useSelector(({ cards }) => cards.isLoading);
  const cardsList = useSelector(({ cards }) => cards.cards);
  const cardsInFavorites = useSelector(({ favourites }) => favourites);
  const hasErrorMessage = useSelector(({ hasError }) => hasError);
  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(fetchCardsList());
  }, []);

  // Favourites
  const changeFavouriteHandler = (articul) => {
    if (cardsInFavorites.includes(articul)) {
      dispatch(removeFavourites(articul));
    } else {
      dispatch(addToFavourites(articul));
    }
  };

  const filteredCards = cardsList.filter(({ articul }) => cardsInFavorites.includes(articul));

  return (
    <div className={classes.favoritesSection}>
      <div className={classes.container}>
        {isLoading && <Loader />}
        {hasErrorMessage && <h3>Sorry, error</h3>}
        {!isLoading && !hasErrorMessage && cardsInFavorites.length < 1 ? (
          <>
            <h2 className={classes.favoritesTitle}>Список бажань - {cardsInFavorites.length}</h2>
            <p className={classes.noItemsTitle}>Немає товарів у списку бажань</p>
          </>
```

```
      ) : (
        <>
          <h2 className={classes.favoritesTitle}>Favorites - {cardsInFavorites.length} items</h2>
          <CardsList
            cards={filteredCards}
            changeFavouriteHandler={changeFavouriteHandler}
            favouritesCardsArr={cardsInFavorites}
          />
        </>
      )}
    </div>
  </div>
 );
};

export default FavouritesContent;

import React, { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchCardsList } from '../../store/cards/actions';
import { removeFavourites, addToFavourites } from '../../store/favourites/actions';
import CardsList from '../Catalog/CardsList';
import Loader from '../common/Loader';
import { createStyles } from './styles';

const FavouritesContent = () => {
 const classes = createStyles();

 const isLoading = useSelector(({ cards }) => cards.isLoading);
 const cardsList = useSelector(({ cards }) => cards.cards);
 const cardsInFavorites = useSelector(({ favourites }) => favourites);
 const hasErrorMessage = useSelector(({ hasError }) => hasError);
 const dispatch = useDispatch();

 useEffect(() => {
  dispatch(fetchCardsList());
 }, []);

 // Favourites
 const changeFavouriteHandler = (articul) => {
  if (cardsInFavorites.includes(articul)) {
   dispatch(removeFavourites(articul));
  } else {
   dispatch(addToFavourites(articul));
```

```
  }
 };

 const filteredCards = cardsList.filter(({ articul }) => cardsInFavorites.includes(articul));

 return (
  <div className={classes.favoritesSection}>
   <div className={classes.container}>
    {isLoading && <Loader />}
    {hasErrorMessage && <h3>Sorry, error</h3>}
    {!isLoading && !hasErrorMessage && cardsInFavorites.length < 1 ? (
     <>
      <h2 className={classes.favoritesTitle}>Список бажань - {cardsInFavorites.length}</h2>
      <p className={classes.noItemsTitle}>Немає товарів у списку бажань</p>
     </>
    ) : (
     <>
      <h2 className={classes.favoritesTitle}>Favorites - {cardsInFavorites.length} items</h2>
      <CardsList
       cards={filteredCards}
       changeFavouriteHandler={changeFavouriteHandler}
       favouritesCardsArr={cardsInFavorites}
      />
     </>
    )}
   </div>
  </div>
 );
};

export default FavouritesContent;

import React from 'react';
import PropTypes from 'prop-types';
import { useSelector } from 'react-redux';
import clsx from 'clsx';
import { countSubtotal } from '../../global/helpers/countSubtotal';
import { countTotalWithDiscount } from '../../global/helpers/countTotalWithDiscount';
import CheckoutList from '../Cart/CheckoutList/CheckoutList';
import { CheckoutIcon, CloseIcon } from '../icons';
import { createStyles } from './styles';
import Button from '../common/Button/index';
import { colors } from '../../global/theme/colors';
```

```
export const OrderModal = ({ header, closeButton, formValues, closeModalHandler }) => {
  const classes = createStyles();
  const modalIsShown = useSelector(({ modal }) => modal.checkoutModalIsOpen);
  const classHide = !modalIsShown ? classes.hide : '';
  const cardsList = useSelector(({ cards }) => cards.cards);
  const cardsInCartList = useSelector(({ cardsInCart }) => cardsInCart);
  const discountValue = useSelector(({ discount }) => discount.discount);

  const filteredCards = cardsList.filter(({ articul }) => cardsInCartList.find(({ id }) => articul === id));

  const normalizeInputKeys = (key) => {
    if (key === 'firstName') return 'First name';
    if (key === 'lastName') return 'Last name';
    return key[0].toUpperCase() + key.slice(1);
  };

  const dataArr = [];
  if (formValues) {
    // eslint-disable-next-line no-restricted-syntax
    for (const [key, value] of Object.entries(formValues)) {
      dataArr.push(
        <li key={key}>
          <span className={classes.fieldName}>{normalizeInputKeys(key)}</span>: {value}
        </li>,
      );
    }
  }

  const keyPressHandler = (e) => {
    if (e.keyCode === 13) {
      closeModalHandler();
    }
  };

  const subTotal = countSubtotal(cardsInCartList, cardsList);
  const total = countTotalWithDiscount(subTotal, discountValue);

  return (
    <>
      <div className={`${classes.modalBox} ${modalIsShown ? '' : classes.hide}`}>
        <div className={`${classes.header} ${classes.header}`}>
          <Button
            type="icon"
            aria-label="Close modal"
```

```
    onClick={closeModalHandler}
    className={closeButton ? classes.closeBtn : ''}
   >
    <CloseIcon width="15" height="15" fill={colors.blue} stroke="" />
   </Button>
   <div className={classes.iconWrapper}>
    <CheckoutIcon width="66" height="80" fill={colors.blue} stroke="" />
   </div>
   <h2 className={`${classes.headerTitle} ${classes.headerTitle}`}>{header}</h2>
  </div>
  <div className={classes.orderInfoBlock}>
   <p className={classes.orderDetailsTitle}>Контактні дані</p>

   <ul className={classes.customerDataList}>{[...dataArr]}</ul>
   <p className={classes.orderDetailsTitle}>Лист замовлення</p>
   <ul className={classes.productsTitles}>
    <li>Товар</li>
    <li>К-ть</li>
    <li>Вартість</li>
   </ul>
   <div>
    <CheckoutList cards={filteredCards} />
   </div>

   <p className={clsx(classes.discount, classes.priceInfo)}>
    <span>Знижка:</span>
    <span>{discountValue || '0'}%</span>
   </p>
   <p className={classes.priceInfo}>
    <span>Загалом:</span>
    <span className={classes.totalPrice}>{total.toFixed(2)} UAH</span>
   </p>
  </div>
  <Button type="button" className={classes.btn} onClick={closeModalHandler}>
   Ок
  </Button>
 </div>
 <div
  role="button"
  aria-label="Close modal"
  tabIndex={0}
  onKeyDown={keyPressHandler}
  onClick={closeModalHandler}
  className={`${classes.overlay} ${classHide}`}
```

```
      />
    </>
  );
};

OrderModal.propTypes = {
  header: PropTypes.string.isRequired,
  closeButton: PropTypes.bool,
  formValues: PropTypes.shape({
    firstName: PropTypes.string,
    lastName: PropTypes.string,
    email: PropTypes.string,
    age: PropTypes.string,
    phone: PropTypes.string,
    address: PropTypes.string,
  }),
  closeModalHandler: PropTypes.func.isRequired,
};

OrderModal.defaultProps = {
  formValues: {},
  closeButton: true,
};

module.exports = {
  env: {
    browser: true,
    es2021: true,
    commonjs: true,
  },
  extends: ['plugin:react/recommended', 'airbnb', 'prettier'],
  parser: 'babel-eslint',
  parserOptions: {
    ecmaFeatures: {
      jsx: true,
    },
    ecmaVersion: 12,
    sourceType: 'module',
  },
  plugins: ['react', 'prettier'],
  rules: {
    'import/prefer-default-export': 0,
    'import/extensions': 0,
    'import/no-unresolved': 0,
```

```
    'react/jsx-filename-extension': [1, { extensions: ['.js', '.jsx'] }],
    'react/prop-types': 1,
    'react/jsx-props-no-spreading': 0,
    'react/destructuring-assignment': 0,
    'react/jsx-one-expression-per-line': 0,
    'react/require-default-props': "off",
    'import/no-extraneous-dependencies': ['error', { devDependencies: true }],
    'no-use-before-define': [0], // for this weird "React was used before it was defined" rule.
    'no-console': ['warn', { allow: ['warn', 'error'] }],
    'react/jsx-wrap-multilines': ['error', { declaration: false, assignment: false }], // Disable inconsistent eslint
and prettier multiline error.
    'prefer-destructuring': 0,
  },
};

import React from 'react';
import { Reset } from 'styled-reset';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import { createStyles } from './globalStyles';
import Catalog from './pages/Catalog';
import Homepage from './pages/HomePage';
import ScrollToTop from './global/hooks/ScrollToTop';
import Header from './components/Header';
import Footer from './components/Footer';
import Cart from './pages/Cart';
import Favourites from './pages/Favourites';
import ProductPage from './pages/Product';

const App = () => {
  const classes = createStyles();
  return (
    <div className={classes.appWrapper}>
      <Router>
        <Reset />
        <ScrollToTop>
          <Header />
          <div>
            <Switch>
              <Route exact path="/" component={Homepage} />
              <Route exact path="/catalog" component={Catalog} />
              <Route exact path="/cart" component={Cart} />
              <Route exact path="/favourites" component={Favourites} />
              <Route exact path="/product/:id" component={ProductPage} />
            </Switch>
```

```
          </div>
          <Footer />
        </ScrollToTop>
      </Router>
    </div>
  );
};

export default App;
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

на кваліфікаційну роботу бакалавру

на тему: Розробка веб-орієнтованого додатку з продажу смартфонів

Керівник кваліфікаційної роботи
доцент, каф. ПЕП та ПУ, к.е.и **Л.В. Касьяненко**

ДОДАТОК В.

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	**Опис**
Пояснювальні документи	
Кваліфікаційна робота_Данильченко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_Данильченко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Project.rar	Архів. Містить код програми і откомпільовану програму
Презентація	
Презентація_Данильченко.pptx	Презентація кваліфікаційної роботи