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

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Купрієнко Вікторії Євгенівни*
(ПІБ)

академічної групи *122-17-2*
(шифр)

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

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

на тему: *Розробка та адміністрування front-end частини Інтернет магазину з продажу одягу*

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

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

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

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-17-2* *Купрієнко Вікторії Євгенівни*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка та адміністрування front-end частини Інтернет магазину з продажу одягу*

затверджена наказом ректора НТУ «ДП» від «07» червня 2021 р. № 317-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц. Гуліна І.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Купрієнко В.Є.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 85 с., 17 рис., 3 дод., 30 джерел.

Об'єкт розробки: система функціонування та адміністрування інтернет-магазину одягу .

Мета кваліфікаційої роботи: розробка та адміністрування front-end частини Інтернет магазину з продажу одягу.

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

У першому розділі проведено аналіз предметної галузі, 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 системи, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної підсистеми, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає у створенні системи, що забезпечує відвідувачам сайту доступ до каталогу товарів за категоріями, можливість редагування заказу та перегляду особистої інформації, що забезпечує комфортне користування інтернет-магазином . Адміністративна панель магазину дає можливість адміністрації магазину надавати актуальну інформацію щодо наявності та вартості товару.

Актуальність інтернет-магазину визначається великим попитом на онлайн послуги, що оптимізують та спрощують обслуговування клієнтів, сприяють створенню позитивного іміджу компанії.

Список ключових слів: ІНТЕРНЕТ-МАГАЗИН, АДМІНІСТРАТИВНА ПАНЕЛЬ, САЙТ, ВЕБ-ДОДАТОК, ФРОНТЕНД, БРАУЗЕР.

ABSTRACT

Explanatory note: 85 p., 17 figs, 3 apps, 30 sources.

Object of development: the system of functioning and administration of the online clothing store.

The purpose of the diploma project: development and administration of the front-end part of the online clothing store.

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

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

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the system, determines the input and output data, provides characteristics of the parameters of technical means, describes the call and application load, describes the program .

The economic section determines the complexity of the developed information subsystem, calculates the cost of work to create an application and calculates the time for its creation.

The practical significance is to create a system that provides site visitors with access to the catalog of products by category, the ability to edit the order and view personal information, which provides comfortable use of the online store. The administrative panel of the store allows the store administration to provide up-to-date information on the availability and cost of goods.

The relevance of the online store is determined by the high demand for online services that optimize and simplify customer service, contribute to the creation of a positive image of the company.

Keywords: ONLINE STORE, ADMINISTRATIVE PANEL, WEBSITE, WEB APP, FRONTEND, BROWSER.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

ПК - персональний комп’ютер;

СУБД - система управління базами даних;

API - Application Programming Interface;

CMS - система управління контентом;

CSS - Cascading Style Sheets;

DOM - Document Object Model;

IT - інформаційні технології;

FE - FrontEnd;

JS - Java Script;

LMS - Learning Management System;

MVC - Model – View – Controller;

HTML - Hypertext Markup Language.

ВСТУП

У наш час велика частина торгівлі приходиться на інтернет-магазини, бо кожен, хто має доступ в Інтернет, може замовити будь-що з будь-якої точки світу.

З кожним роком інтернет-магазини стають більш популярними, ринок IT росте і стає важко розробляти щось нове в інтерфейсі для користувачів. Вагомими факторами успішності веб-додатку є його візуальна частина та швидкість завантаження сторінок. Користувачів також приваблюють якісні фото товарів та легкість створення замовлення. Відвідавши сайт, потенційний покупець може подивитися фото товару, актуальні розміри і детальну інформацію. Сайти зберігають в собі масу корисної і важливої інформації, вони знаходяться у відкритому доступі для будь-якого користувача, а тому так широко користуються попитом на сьогоднішній день.

У всьому світі понад усе цінується якість, тому інформація на сайтах обов'язково повинна бути якісною, цікавою, розгорнутої і корисною. На яку б тему не був створений сайт, він повинен розкривати її максимально широко, від самого початку і до найменших дрібниць.

Дана робота дозволяє познайомитись з методами та технологіями розробки інтернет-магазинів та перевагами веб-додатків.

Отже задачею даної кваліфікаційної роботи є створення веб-додатку, що сприяє розвиненню продажу та бізнесу компанії з продажу одягу.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

У наш час є безліч сайтів для магазинів, державних служб, різних компаній і навіть для стеження на карті за поширенням захворювань у світі. Веб-додатки умовно можна поділити односторінкові та багатосторінкові сайти.

Односторінкові сайти використовують для простого відображення інформації, наприклад портфоліо, стаття, дані про захід, тощо. Багатосторінковими сайтами є інтернет-магазини, сервіси роботи з клієнтами, державні сервіси, банківські додатки - усі, що мають багато інформації, яку треба структурувати. Різниця роботи між одно- і багатосторінковими сайтами полягає в тому, що перші вантажать усю інформацію одразу і у більшості випадків не мають окремої бази даних. Другі у свою чергу довантажують дані відповідно до дій користувача, а саме працюють з базою данних, відправляють та обробляють запити.

Статичні сайти створюються, в основному, на «чистому» HTML з використанням різних стилів, фреймворків з шаблонами компонентів. Такі сайти мають свої переваги та недоліки.

Переваги систем на базі мови HTML:

- сайт буде працювати на будь-якому сервері хостингу, навіть на локальному;
- дані сайту завантажуються разом з усіма елементами і ми можемо бачити усе і одразу;
- для розробки додатку не потрібні спеціальні програми, писати код можна навіть у WordPad;
- легко змінити зовнішній вигляд будь-якої конкретної сторінки чи додати нову, невпливаючи на інші частини сайту;

- мале число використовуваних програмних компонентів утруднює злом такої системи.

Такий спосіб створення сайтів застарілий та використовується рідко, бо такісистеми мають свої недоліки:

- неможливо або досить важко забезпечити розділення прав доступу до вмісту сайту;
- складно внести зміни в структуру і зовнішній вигляд сайту, бо нема розподілення на «компоненти» і якщо ми змінюємо стиль шапки сайту у одному місці, треба змінювати це на всіх сторінках;
- неможливо використовувати такі компоненти, як коментарі та відгуки користувачів, голосування, форум, чат і т.д.;
- щоб змінити якісь дані контенту треба робити правки в коді, де в HTML тегах зберігаються абзаци.

Щоб компенсувати недоліки чистого HTML до сайту підключаються різні мови програмування, що дають можливість зробити сайт динамічним та допомагає у роботі з базами данних, створенні анімацій та комфортної навігації між частинами додатку.

Коли користувач запитує сторінку, відповідна інформація витягується з БД, вставляється в шаблон, утворюючи нову сторінку сайту, і пересилається сервером в браузер, який і відображає її належним чином. Крім інформаційного наповнення, динамічно можуть створюватися також і елементи навігації по сайту. Таким чином, якщо потрібно оновити вміст сайту, слід просто змінити текст для нової сторінки у базі даних чи відредагувати з адміністративної панелі, якщо така є. Якщо ви хочете розвивати свій інтернет магазин, то ці складні технічні процеси не повинні відволікати від ведення бізнесу.

З цієї властивості безпосередньо випливає одна з переваг - спрощення модифікації і оновлення сторінок на сайті. Інформація повинна бути свіжою, інакше відвідувачі швидко втратять інтерес до сайту.

Перше що потрібно зробити у розробці сайту – це вибрати спосіб зберігання інформаційного наповнення (контенту), що відображається на Web-сторінці. В даному проекті для цих цілей використовується база даних що містить усю інформацію про товари та замовлення магазину.

Після цього створюються сторінки на React JS, що складаються з компонентів. Усю інформацію з БД Frontend отримує через запити API.

Така концепція створення сторінок, коли усі данні та завантаження сторінок ми отримуємо динамічно спрощує проектування структури проекту та дає можливість коректно та швидко завантажувати дані.

1.2. Призначення розробки та галузь застосування

Як об'єкт розробки системи відображення і адміністрування інтернет-магазину одягу розглядається веб-додаток «Closes Shop», в якому користувач може знайти одяг закордонного виробництва та передивитися структурований каталог товарів з функціями пошуку товару, сортування за категоріями. Оскільки інтернет-магазин розрахований на роздрібну торгівлю, для підвищення продажів магазин повинен надавати повну інформацію про товар, легкість у користуванні, щоб користувач відчував себе як у справжньому магазині.

Для комфортного адміністрування магазину розробляється адміністративна панель, у якій адмінітратор може дивитися та редагувати замовлення, створювати нові позначки для категорій, редагувати контент магазину.

Розроблена Frontend частина додатку призначена для:

- надання відображення контенту інтернет-магазину за разними категоріями ;
- забезпечення відвідувачам сайту простоти і комфортності доступу до каталогу товарів та створення замовлення;

– підвищення продажу одягу магазину за рахунок швидкості роботи інтернет-магазину та швидкості обробки замовлень у адміністративній панелі.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки «виконання кваліфікаційної роботи» є:

– освітня програма 122 «Комп’ютерні науки»;
– навчальний план та графік навчального процесу;
– наказ ректора Національного технічного університету «Дніпровська політехніка» №317-с від 07.06.2021р.;
– завдання на кваліфікаційну роботу на тему «Розробка frontend та системи відображення і адміністрування інтернет магазину одягу».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є розробка та адміністрування front-end частини Інтернет магазину з продажу одягу.

Програмне забезпечення призначене для надання універсального інструменту для відображення контенту та адміністрування даного інтернет-магазину.

Програма повинна реалізувати наступні функції:

– легке та доступне адміністрування інтернет магазину;
– формування web-сторінок на основі шаблону з використанням інформації, отриманної з бази даних;

– відображення сторінок для різних девайсів користувача;
– контактування за адміністраторами данного магазину;
– отримання фідбеку від адміністраторів на пошту;
– оформлення замовлень.

Для досягнення поставленої мети необхідно:
– вивчити предметну галузь розв'язуваної задачі;
– створити алгоритм для реалізації поставленого завдання;
– створити базу даних і клієнтську програму та серверну программу, що працює з нею.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставлених цілей програмне забезпечення, що розробляється, повинно підтримувати виконання наступних дій:
– надання віддаленого доступу до застосунку через веб-браузер на комп'ютері користувача;
– зчитування вхідних даних з пристроїв, хмарних сервісів, за адресними посиланнями;
– зберігання даних підсистеми в реляційній базі даних.
– для виконання перерахованих вище функцій у застосунку повинні бути реалізовані:
– можливість інтеграції в платформу веб-сайту з продажу одягу;
– можливість отримати доступ до програми через веб-браузер;
– наявність типової конфігурації, що забезпечує можливість швидкого введення застосунку в експлуатацію;
– програмно-апаратна переносимість.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

- семантичний та синтаксичний контроль вхідних даних;
- обробку виняткових ситуацій;
- виведення повідомлень про помилки;
- можливість повторного введення даних;
- можливість безперервної роботи протягом не менше 120 годин (5 діб);
- платформну незалежність.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування данного застосунку необхідно, щоб персональний комп’ютер, на якому, буде функціонувати система інтренет магазину, відповідала наступним вимогам:

- процесор класу Intel Pantium з тактовою частотою не менш 2.4 ГГц та двума ядрами;
- доступ до мережі Internet;
- не менше 8 GB оперативної пам'яті;
- 500 GB вільного місця на жорсткому диску;
- клавіатура;
- маніпулятор "миша".

Наведені вище технічні характеристики ПК є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний застосунок буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

1.5.4 Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення персонального комп'ютера, на якому буде функціонувати веб-орієнтована система, відповідало наступним вимогам:

- операційна система Windows (7+), Linux, MacOs;
- веб-браузер Firefox / Google Chrome / Opera / Microsoft Edge / Safari.

Frontend частина додатку має бути реалізована на мові програмування JavaScript з використанням фреймворку React JS та бібліотеки Redux Saga. Робота має бути створена за допомогою компонентного підходу та solid архітектури.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

За завданням кваліфікаційної роботи було реалізовано розробку frontend частини системи відображення і адміністрування інтернет магазину одягу.

Призначення розробленої системи:

- систематизувати дані компанії;
- забезпечити просте адміністрування магазину;
- надати можливість обмінюватися даними та запитами між користувачем та адміністратором.

Розроблена програма реалізує наступні функції:

- легке та доступне адміністрування інтернет магазину;
- формування web-сторінок на основі шаблону з використанням контенту полученого з серверу;
- формування web-сторінок на основі шаблону з використанням контенту з бази даних;
- контакування за адміністратором данного магазину;
- оформлення замовлень з данного магазину.

Для досягнення поставленої задачі розроблене програмне забезпечення підтримує виконання таких операцій:

- надання віддаленого доступу до застосунку через веб-браузер на комп'ютері або іншому девайсі користувача;
- зчитування вхідних даних з хмарних сервісів та бази даних;
- формування замовлення та відправка на адміністративну частину;
- коригування контенту через адміністративну частину.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці системи відображення та управління контенту інтернет-магазину математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

FE частина web-додатку має бути реалізована на мові програмування JavaScript з використанням фреймворку React JS та бібліотеки Redux Saga. Робота створена за допомогою компонентного підходу та solid архітектури.

JavaScript - це мова програмування, яку зазвичай використовують для реалізації складної поведінки веб-сторінок. Звичайна веб-сторінка не лише просто відображає зміст, але і відповідає за синхронізацію з джерелом інформації, поведінку кожної кнопки та анімаційного елементу. Java Script також відповідає за роботу з відео, відображенням прелоадеру, навігацію між сторінками.

Саме найменування JavaScript з'явилося не відразу. Спочатку мова називалася Mocha, потім з'явився термін Livescript, що краще відображав суть. У той час в браузерах активно впроваджувалася підтримка нової, перспективної мови Java. Тоді і було вирішено перейменувати Livescript в JavaScript. Схожість між мовами є, перш за все, по синтаксису. Але по суті це дуже різні інструменти.

У 2015 році вийшла версія ES6 (ES2015). Ця подія вважається проривом у розвитку JavaScript. З'явилися нові стандарти та можливості. Наприклад - константи. Код став більш раціональним, була реалізована ідея «пиши менше - роби більше».

JavaScript застосовують також для створення мобільних додатків, в серверній (backend) розробці, в десктопних (наприклад, офісних) програмах. Та частіше все ж таки Java Script використовують для веб-додатків різної

складності: як односторінкові портфоліо так і інтернет-магазини чи LMS системи.

JavaScript популярний не випадково, а завдяки своїм безперечним перевагам:

- підтримка скриптів усіма популярними браузерами;
- повна інтеграція з верстkою сторінок (HTML + CSS) і серверної частиною (backend).
- швидкість роботи і продуктивність. Щоб зекономити час та навантаження на сервер JavaScript дозволяє частково обробляти веб-сторінки на комп'ютерах користувача без запитів до сервера;
- велика кількість бібліотек та фреймворків;
- прості завдання вирішуються швидко, а для складних є готові шаблони вирішення (патерни);
- зручність для користувача інтерфейсів
- легкість освоєння.

JavaScript також має деякі недоліки (обмеження):

- немає можливості читання та завантаження файлів
- нестрога типізація і вільне трактування. Мова ігнорує явні нестиковки, але до цього також можна звикнути.
- немає підтримки віддаленого доступу. Мову не можна використовувати для мережевих додатків, тому JavaScript іноді навіть не вважають повноцінною мовою програмування.

Саме в області Frontend є велика кількість компонентів та напрацювань, що створюють окремі фреймворки. Найбільш активно використовується приблизно 25-30 бібліотек і фреймворків серед сотні існуючих. Вони створені для економлі часу та вартості проектів, дають змогу сконцентруватися на логіці, а не витрачати час на створення блоків та компонентів дизайну.

У наш час найбільше використовують такі фреймворки та бібліотеки:

- Angular JS;

– Node.js;
– React.js;
– Vue.js.

Ми вже звикли бачити серед провідних JavaScript-фреймворків React, Angular і Vue.js. Звіт про стан фронтенда State of Frontend 2020 відображає результати опитування більш ніж 4500 професійних фронтенд-розробників (рис. 2.1.) [5].

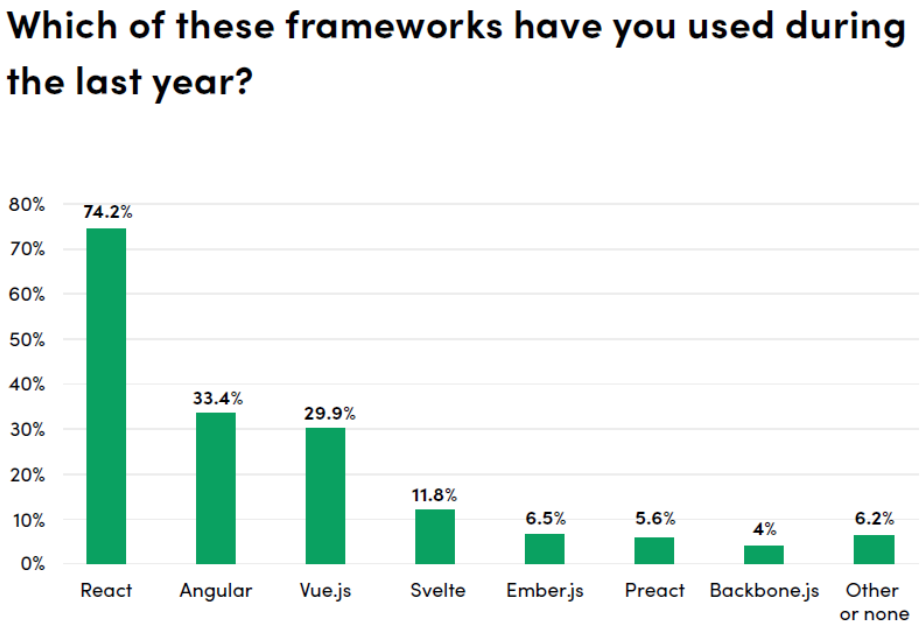

Рис. 2.1. Результати опитування

React - JavaScript-бібліотека для роботи з інтерфейсом від розробників Facebook. Бібліотеку почали використовувати на сайті цієї соціальної мережі в 2011 році. А в 2013 році Facebook відкрив вихідний код React.

За допомогою React розробники створюють веб-додатки, які змінюють відображення без перезавантаження сторінки. Завдяки цьому додаток швидко реагує на дії користувача, наприклад, заповнення форм, застосування фільтрів, додавання товарів в корзину і так далі. Також бібліотека може повністю управляти фронтендом. В цьому випадку React використовують з бібліотеками для управління станом і роутінгом, наприклад, Redux і React Router.

Одна з ключових особливостей React - універсальність. Цю бібліотеку можна використовувати на сервері і на мобільних платформах за допомогою React Native. Це принцип Learn Once, Write Anywhere або «Навчіться один раз, пишіть де завгодно».

React заснований на компонентах, це ще одна ключова особливість бібліотеки. Кожен компонент повертає частину призначеного для користувача інтерфейсу зі своїм станом. Об'єднуючи компоненти, програміст створює завершений інтерфейс веб-додатку.

Ще одна важлива особливість React - використання JSX. Це розширення синтаксису JavaScript, яке зручно використовувати для опису інтерфейсу. JSX схожий на HTML, проте це все-таки JavaScript. Приклад JSX (рис. 2.2.) можна побачити нижче :

```
const header = text ? <h1>{text}</h1> : null;

const vdom = (
  <div>
    {header}
    <Hello />
  </div>
);
```

Рис. 2.2. Приклад JSX

JSX дозволяє писати JavaScript-код за допомогою готових компонентів, які практично повністю повторюють HTML. Це спрощує розробку.

До важливих особливостей також відноситься використання віртуального DOM (Virtual DOM). Віртуальний DOM - об'єкт, в якому зберігається інформація про стан інтерфейсу. Використання віртуального DOM дозволяє бібліотеці ефективно оновлювати реальний DOM.

Redux-saga - це бібліотека, яка має на меті зробити побічні ефекти додатків (тобто асинхронні речі, такі як отримання даних та нечисті речі, такі як доступ до кешу браузера) простішими в управлінні, більш ефективними у виконанні, легкими для тестування та кращими при обробці помилок.

Логічна модель полягає в тому, що сага - це як окрема нитка у додатку, яка відповідає виключно за побічні ефекти. Redux-saga - це проміжне програмне забезпечення redux, що означає, що цей потік можна запускати, призупиняти та скасовувати з основної програми за допомогою звичайних дій відновлення, він має доступ до повного стану програми відновлення, а також може відправляти дії відновлення.

Бібліотека використовує концепцію ES6, під назвою генератори, для того, щоб зробити ці асинхронні потоки легкими для читання, написання та тестування. Тим самим, ці асинхронні потоки виглядають, як ваш стандартний синхронний JavaScript-код.

2.4. Опис структури системи та алгоритмів її функціонування

Під час проектування було проведно аналіз структури вхідних та вихідних даних, призначення Web-сайту, потоків інформації, складу користувачів та принципів їх роботи проектних рішень з урахуванням спрощення роботи користувачів.

Структура сайту – це описання маршруту користувача, за яким він мандрує серед інтерфейсу. Чим простіше розроблений цей шлях – тим більше буде зацікавленість клієнта. Також дуже зручно, коли з будь-якого місця ти можеш повернутися назад чи вийти з додатку.

Загальна структура веб-сайту магазину «Closes Shop» з точки зору користувача (рис. 2.3.) має такий вигляд:

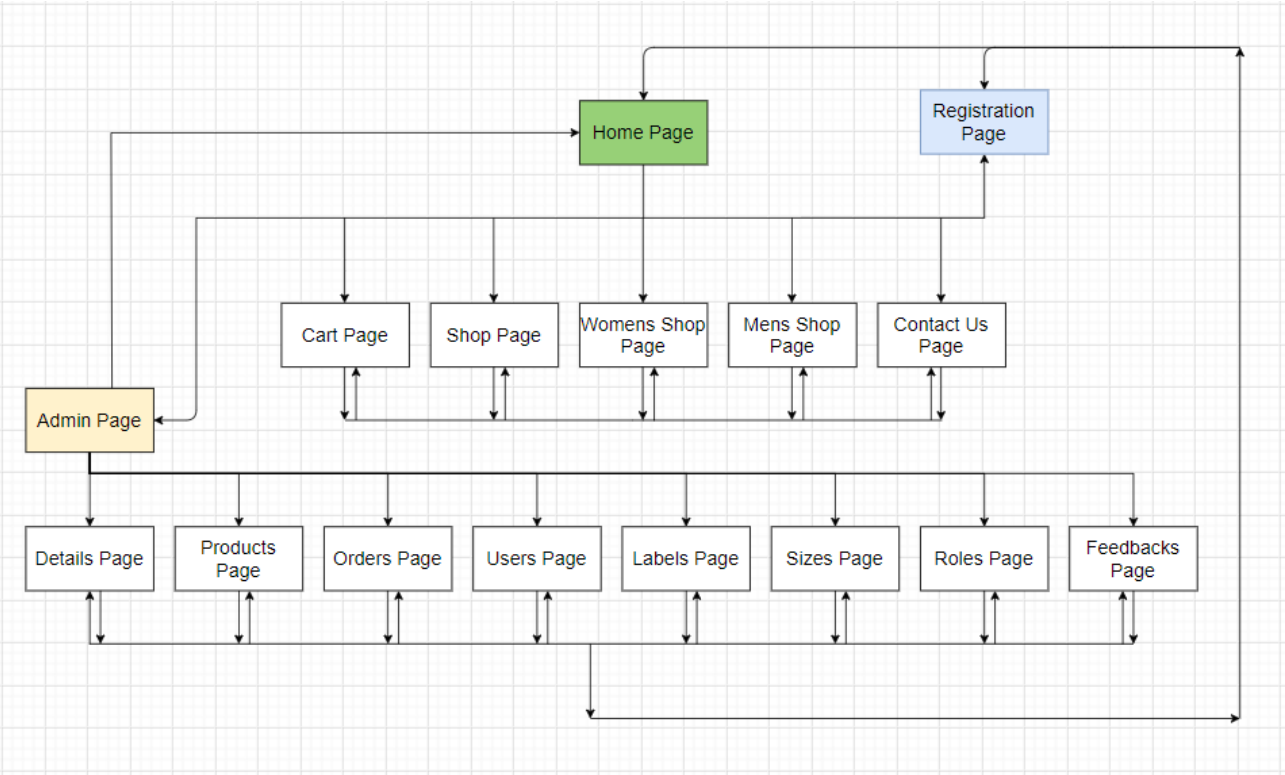

Рис. 2.3. Структура веб-сайту магазину «Closes Shop»

Структура інтернет-магазину складається з таких частин:

- сторінка реєстрації користувача;
- головна сторінка з рейтингом товарів;
- каталог товарів з категоріями одягу;
- деталі товару;
- сторінка профіля з історією заказу;
- кошик з товарами у замовленні;
- контакти з формою зворотнього зв'язку.

Адміністративна панель складається з таких вкладок:

- редагування товарів;
- інформація про користувачів;
- ролі користувачів;
- мітки для товарів;
- список замовлень;
- статистика;

– розміри товарів;
– відгуки від користувачів.

Існують універсальні правила, дотримання яких дозволяє створити комфортні умови для здійснення покупок клієнтами та збільшити конверсію магазину. Успішний інтернет-магазин включає в себе потрібний функціонал, привабливий зовнішній вигляд (дизайн) і маркетинг-мікс (взаємозв'язок маркетингових інструментів).,

Основна увага користувача має припадати на так званий каталог одягу, саме це меню має більший розмір і займає найбільше місце в інтерфейсі. На сторінці каталогу користувач може бачити список категорій, які можна обирати. Також зручності додають пошук по товару та особливі мітки, наприклад «топ продажу», «останні розміри» і інше.

Шапка (header, хедер) та підвал (footer, футер) сайту - це два елементи дизайну, що дублюються на всіх сторінках. У даному магазині шапка сайту подвійна та містить усю основну інформацію про навігацію по інтернет-магазину. У верхній правій частині верхньої панелі розташована кнопка, що визиває випадаюче меню. У цьому меню користувач може знайти кнопки для переходу на сторінку профіля, на адміністративну панель (якщо є такі права у користувача) і може вийти з сессії.

У правій нижні частині на будь-якій сторінці можна знайти кнопку кошика, яка перенесе користувача на окрему сторінку замовлення. Коли користувач додає товари до кошика, цифра кількості товарів змінюється на іконці кошика.

При створенні сучасних додатків останнім часом використовують найпопулярніші шаблони проектування для створення архітектури додатку, такі як, наприклад, MVC і MVP.

Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») - схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, представлення і контролер - таким чином, що модифікація кожного компонента

може здійснюватися незалежно. Він вирішує проблему поділу логіки обробки запиту користувача і логіки подання інформації (рис. 2.4.).

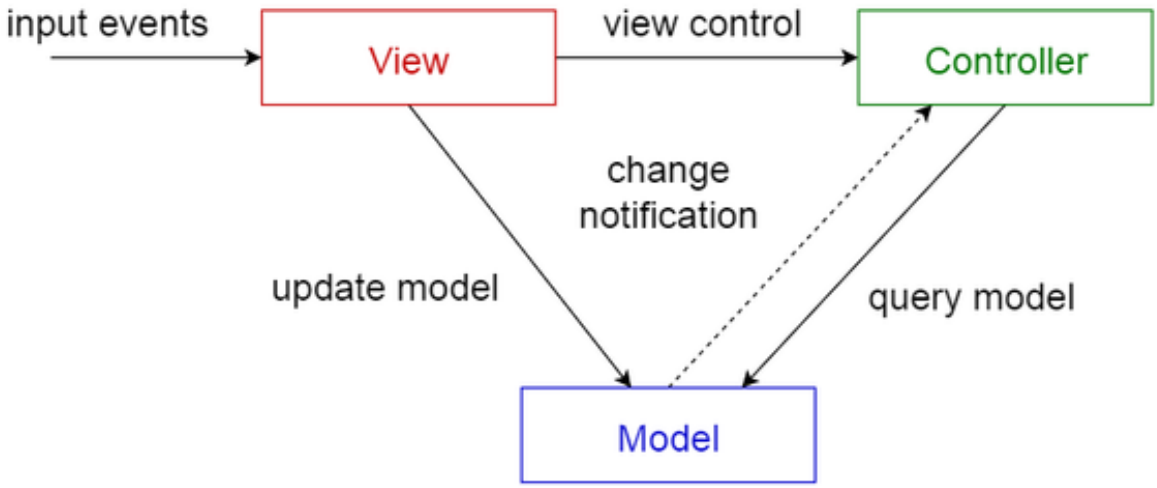

Рис. 2.4. Схема взаємодії компонентів веб-додатку в моделі MVC

Але у цій архітекрурі є свої недоліки, а саме важко розширювати функціонал, змінювати структуру додатку та перевикористовувати якісь частини сторінок, як це можно зробити з компонентним підходом по проектування.

Та саме у React використовується так званий компонентний підхід, який було обрано для цього проекту. У React немає контролерів, в'юшок, моделей, шаблонів і т.д. - все є компонент. Компоненти можна і потрібно перевикористати, успадковувати один від одного, компонувати. Компонент - це свого роду будівельна одиниця, з якої збирається інтерфейс.

React це не черговий MVC-фреймворк або будь-якої іншої фреймворк. Це всього лише бібліотека для рендеринга ваших відображень (views). Якщо ви прийшли зі світу MVC, то вам потрібно усвідомити, що React це буква 'V' в цій абревіатурі.

Можна скільки завгодно сперечатися, який фреймворк краще, але не можна не визнати очевидне - вони все базуються на компонентах. У React, в Vue, в Angular ви займаєтеся тим, що ділите своє додаток на невеликі частини і працюєте з ними як з самостійними одиницями.

Концепція компонентного підходу у фронтенді відкриває неймовірні можливості для повторного використання одного разу написаного коду. Тільки уявіть, ви створюєте компонент (наприклад, красивий прелоадер) для одного проекту, а потім використовуєте його у всіх інших, без всякого переписування або рефакторінга.

У React використовується компонентний підхід для створення додатків, згідно з яким додаток рекомендується складати з повторно використовуваних компонентів, код кожного з яких розташовується в окремих файлах. React поставляється з готовими вбудованими компонентами, функціональність яких можна розширити за допомогою власних користувальницьких компонентів

Коли ви створюєте додаток за допомогою реакту, ви збираєте безліч компонентів, щоб створити дерево компонентів. Ви починаєте з вашого <App /> і закінчуєте низькорівневими <input />, і <button />. Ви не керуєте усіма низькорівневими складовими компонентами, які ваше додаток рендерить, з якогось централізованого місця. Замість цього ви дозволяєте кожному окремому компоненту керувати ним. Як виявилося, це дійсно простий і ефективний спосіб створення UI.

Щоб отримати дані, компоненту необхідно виконати запит до бекенду. Для виконання цього запиту компоненту необхідно мати сервіс, який буде звертатится до бекенду, сагу, яка зробить асинхроний код синхроним та дію яка в свую чергу вкаже компоненту на те, що потрібно виконувати той чи інший запит. Також потрібно мати якесь локальне сховище даних. Робота з даними побудована за принципом flux архітектури.

Flux архітектура - це вид вид архітектури, яку Facebook використвує щоб процювати з React. Якщо розглядти flux в розрізі MVC, то React в свою чергу відповідає за V – View, а flux за M – Model.

Це архітектура (рис.2.5.), яка відповідає за створення слою данних у JavaScript додатках и розробку серверної частини в веб додатках, доповнює комплексні компоненти виду View у React та використовує однонаправлений потік даних.

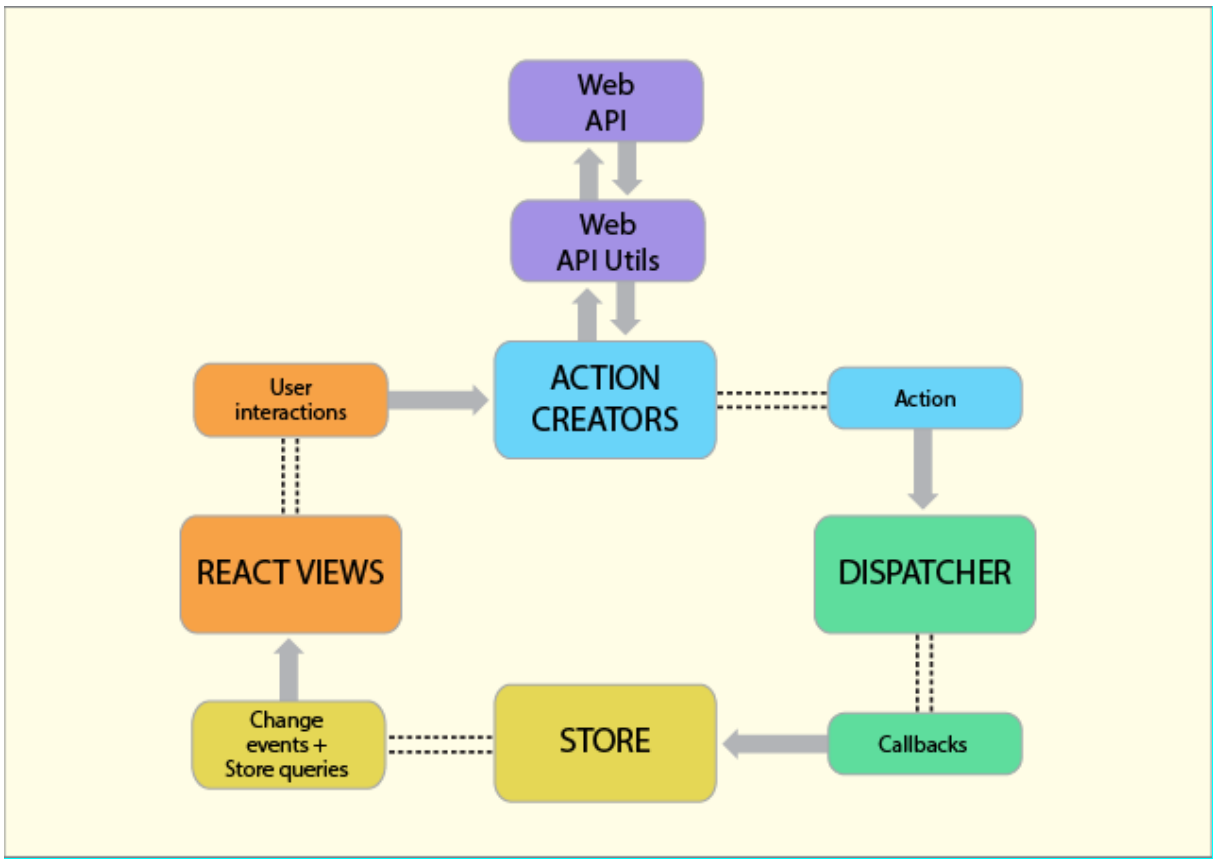

Рис. 2.5. Схема Flux архітектури

У стандартній архітектурі Flux є наступні компоненти:

– Actions – допоміжні елементи які передають дані у Dispatcher;
– Dispatcher – отримує ці дії и передає корисну нагрузку зареєстрованим callback-ом (функцією);
– Stores – діють як контейнери для стану додатку та логіки. Реальна робота додатку проходить в Stores. Stores, зареєстровані для прослухування Dispatcher, будуть оновлювати View;
– Controller View – React компоненти які захоплюють стан з Stores, а потім передають дочірнім компонентам.

Типова дія виглядає так:

```
export const getProductsRequest =
createAction(constants.GET_PRODUCTS_REQUEST);
```

Типова сага виглядає так:

```
export function* getProductsSaga({ payload }) {
  try {
    const response = yield call(ShopService.getProducts, payload);
    yield put(actions.getProductsSuccess(response.data));
  } catch (err) {
    yield put(actions.getProductsError(err));
    yield put(
      addApplicationNotification({
        type: NOTIFICATION_TYPES.error,
        content: err.message,
        canBeClose: true,
      })
    );
  }
}
```

Типове сховище має такий вигляд:

```
const initialState = {
  data: null,
  loading: false,
  error: null,
  filters: null,
};

const reducer = createReducer(initialState, {
  [actions.getProductsRequest]: (state) => {
    state.data = null;
    state.loading = true;
    state.error = null;
```

```
  },
  [actions.getProductsSuccess]: (state, action) => {
    state.loading = false;
    state.data = action.payload;
  },
  [actions.getProductsError]: (state, action) => {
    state.loading = false;
    state.error = action.payload;
  },
  [actions.getProductsFiltersSuccess]: (state, action) => {
    state.filters = action.payload;
  },
});
```

Типовий сервіс має такий вигляд:

```
class ShopService {
  static getProducts = async (url) => {
    try {
      const response = await axios.get(url, {
        headers: {
          "Content-Type": "application/json; charset=UTF-8",
        },
      });
      return response;
    } catch (err) {
      throw err;
    }
  };
```

```
  static getCategories = async () => {
    try {
      const response = await axios.get('/api-v1/categories/', {
        headers: {
          "Content-Type": "application/json; charset=UTF-8",
        },
      });
      return response;
    } catch (err) {
      throw err;
    }
  }

  static getTypes = async () => {
    try {
      const response = await axios.get('/api-v1/types/', {
        headers: {
          "Content-Type": "application/json; charset=UTF-8",
        },
      });
      return response;
    } catch (err) {
      throw err;
    }
  }
}
```

Грамотна файлова система також має велике значення, бо якщо усі файли упорядковані та правильно підключені один до одного – усе працюватиме.

Коли ви працюєте на веб-сайті локально на вашому комп'ютері, треба тримати все пов'язані файли в одній папці, яка відображає файлову структуру

опублікованого веб-сайту на сервері. Ця папка може розташовуватися де завгодно, але там, де можна легко її знайти, може бути, на ваш робочий стіл, в домашню папку або в корінь вашого жорсткого диска.

Також добре тримати файли у приватному репозиторії та кожну невелику частину нового коду комітити, щоб можна було легко повернутися до попередніх змін та завантажити код на комп’ютер при його втраті.

Файли frontend частини інтернет-магазину «Closes Shop» (рис. 2.6.) поділені на компоненти, що використовуються більше одного разу у додатку та сторінки, що містять окремі папки з власними компонентами. Також є файли сидів та міграцій для роботи з інформацією.

Стилі та дизайнерське оформлення, а також адаптивність дизайну інтернет-магазину описана у styles.js файлах, що містяться поряд з основним файлом JSX кожного компоненту та окремої сторінки.

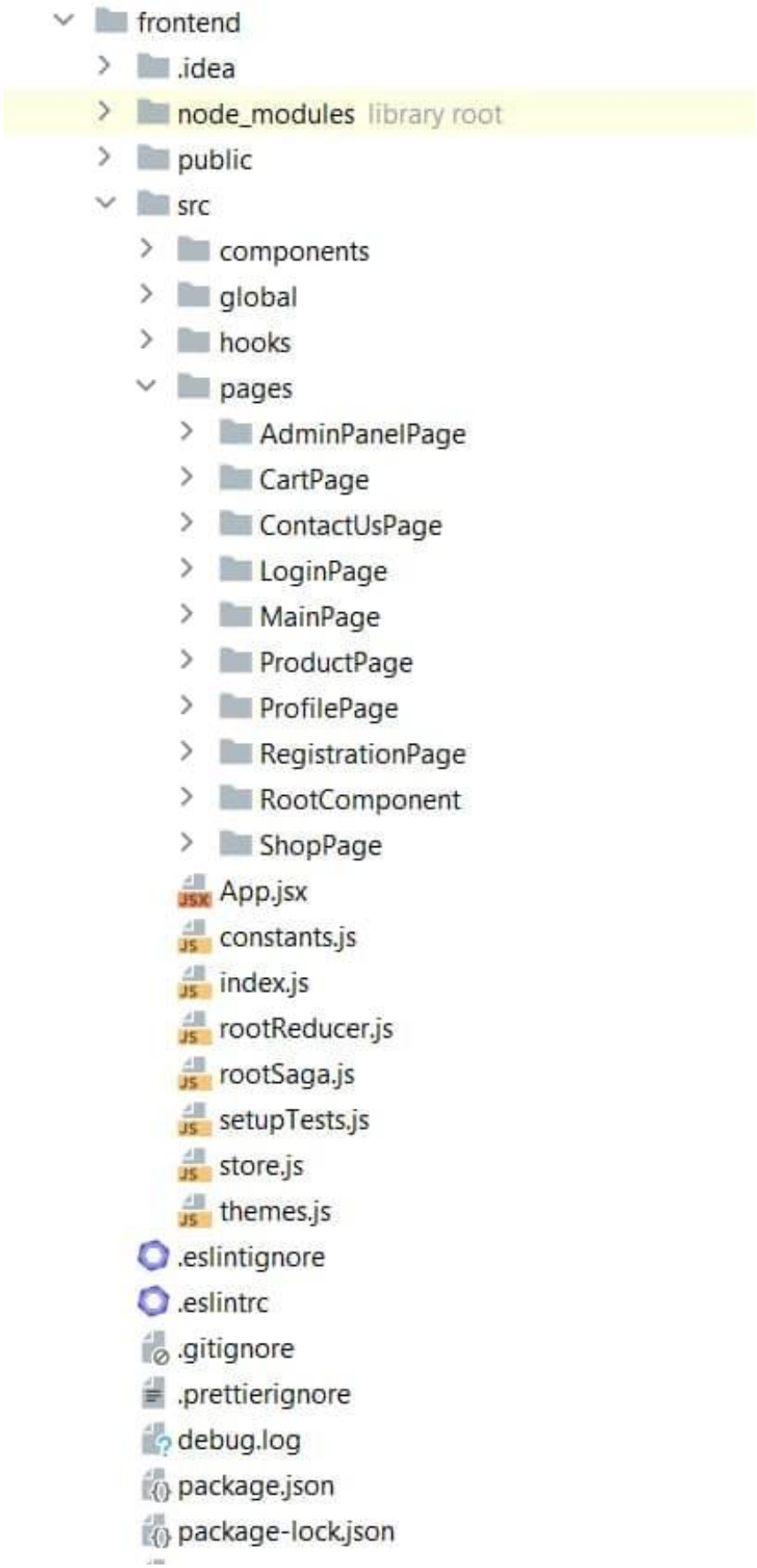

Рис. 2.6. Структура файлової системи

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані шляхом завантаження інформації із бази даних та сховищ даних, та через передачу значень від інших підсистем через глобальні змінні додатку інтернет-магазина.

Вхідні дані:

- інформація про товари магазину;
- таблиця розмірів та типів товару;
- списки користувачів;
- список міст і країн;
- список категорій і лейблів одягу;
- замовлення.

Вихідні дані:

- web-сторінки інтернет-магазину одягу;
- замовлення користувача;
- відгуки;
- запити на ВА.

Дані програми організовані в локальні сховища даних, для кожної сторінки – компонента, що виступає основним контейнером. Локальні сховища організують між собою одне велике сховище даних, до якого підключені усі основні компоненти програмного додатку. Компоненти отримують дані з глобального сховища, та реагують на кожну зміну даних. Дані зі сховищ потрапляють за допомогою спеціального методу, який називається mapStateToProps, далі вони потрапляють у компонент за допомогою отримання їх з props цього компонента.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для серверних технічних засобів рекомендована конфігурація, що забезпечує цілодобову роботу програми з резервуванням даних:

- процесор класу AMD Ryzen 5 2600 3.4(3.9)GHz 16MB sAM4 Box (YD2600BBAFBOX);
- материнська плата Gigabyte GA-A320M-H (sAM4, AMD A320);
- модулі пам’яті G.Skill DDR4 8GB 3000Mhz Aegis (F4-3000C16S-8GISB);
- система охолодження Deepcool GAMMA ARCHER;
- SSD диск Transcend MTS820S TLC 120GB M.2 (2280 SATA) (TS120GMTS820S);
- рідкокристалічний монітор з діагоналлю не менше 17 ";
- доступ до мережі Internet;
- маніпулятор "миша";
- клавіатура.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

2.6.2. Використані програмні засоби

Проект реалізований на мові програмування JavaScript з використанням бібліотеки React JS.

Для створення веб-сайтів існує безліч різних програм. Одні з них є редакторами html, інші - редакторами серверних або клієнтських скриптів, а треті - редакторами таблиць css. Всі вони найчастіше візуалізують процес редагування, і можна бачити як буде виглядати той або інший елемент на сайті.

В якості IDE для розробки був використан WebStorm від компанії Jet Brains з ліцензією від університету, що має вбудований вебсервер. Також для коректної роботи розробленого додатка необхідний довільний веб-браузер з підтримкою технології JavaScript (усі крім Lynx).

Необхідними програмними та технічними засобами для клієнта є мобільний телефон, ПК або інший девайс з підключенням до Інтернету та браузером, що підтримує JavaScript.

2.6.3. Виклик та завантаження програми

Для виклику та завантаженяя необхідно виконаті наступні дії:

– необхідно орендувати або придбати серверне обладнання та доменне ім’я;

– встановити на серверне обладнання одне з серверного забезпечення (Apache або Nginx);

– налаштувати серверне програмне забезпечення;

– за допомогою FTP клієнта приєднатися до сервера та перемістити у папку src усі скомпільовані файли програми;

– підключити доменне ім’я та сертифікати до серверу;

– ввімкнути сервер.

2.6.4. Опис інтерфейсу користувача

Користувач може зареєструвати власну сторінку на сайті, вказавши свої особисті дані. Це можливо зробити на сторінці реєстрації (рис.2.7.). Реєстрація буде неможливою без прийняття правил магазину.

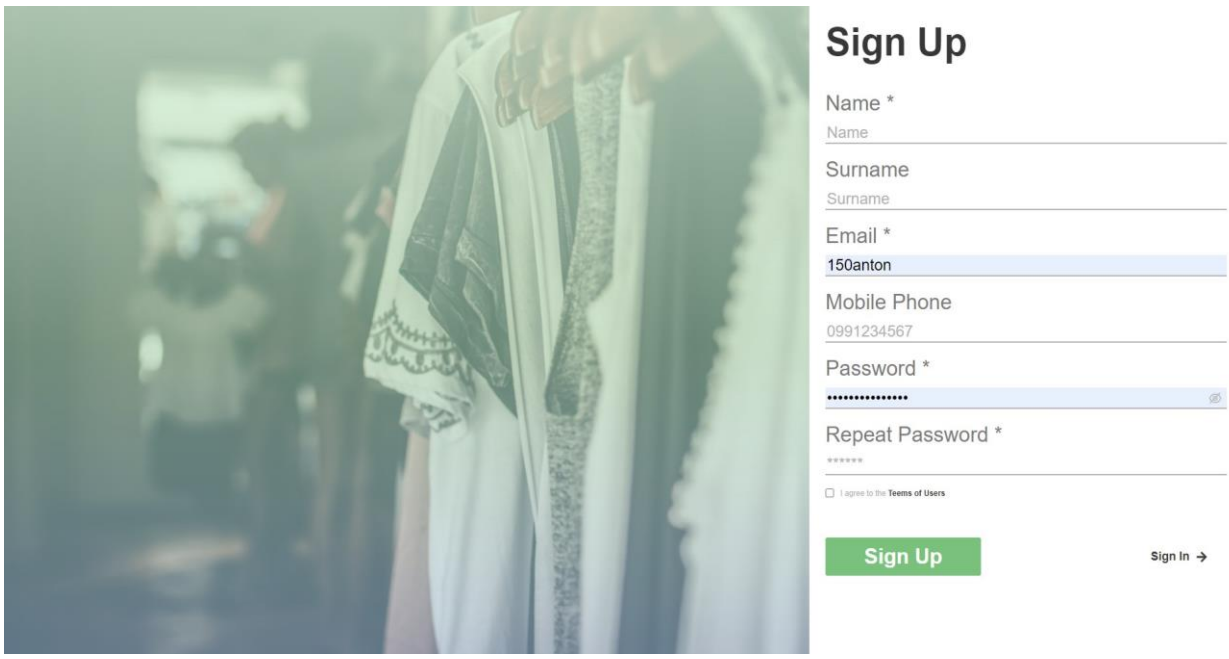

Рис. 2.7. Сторінка реєстрації

Після реєстрації клієнт може побачити основну сторінку (рис.2.8.), коли переходить на адресу сайту. На основній сторінці користувач бачить навігаційну панель, та рекламний слайдер. Слайдер з часом змінюється та відображає актуальні новини магазину.

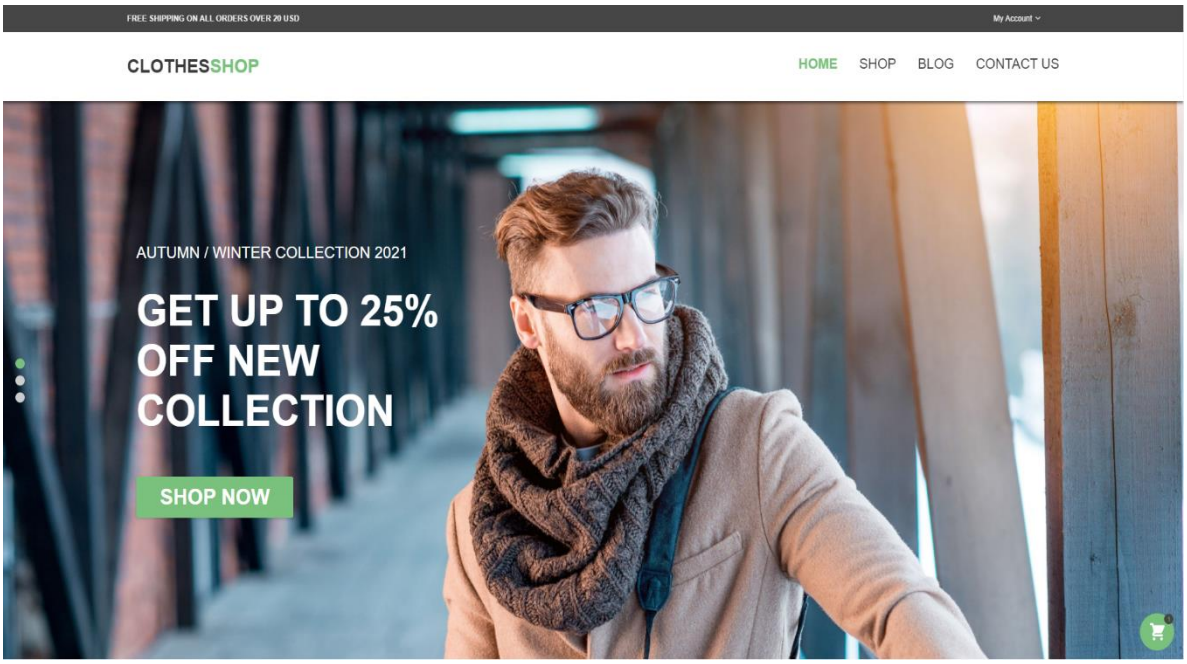

Рис. 2.8. Головна сторінка

Якщо користувач проскролить трохи униз, він побачить основні підрозділи товарів магазину (рис.2.9.), категорії. Якщо він натисне на один з них то потрапить на сторінку магазину з цими товарами. Тобто щоб подивитися дитячі товари я обираю блок з надписом «Kids».

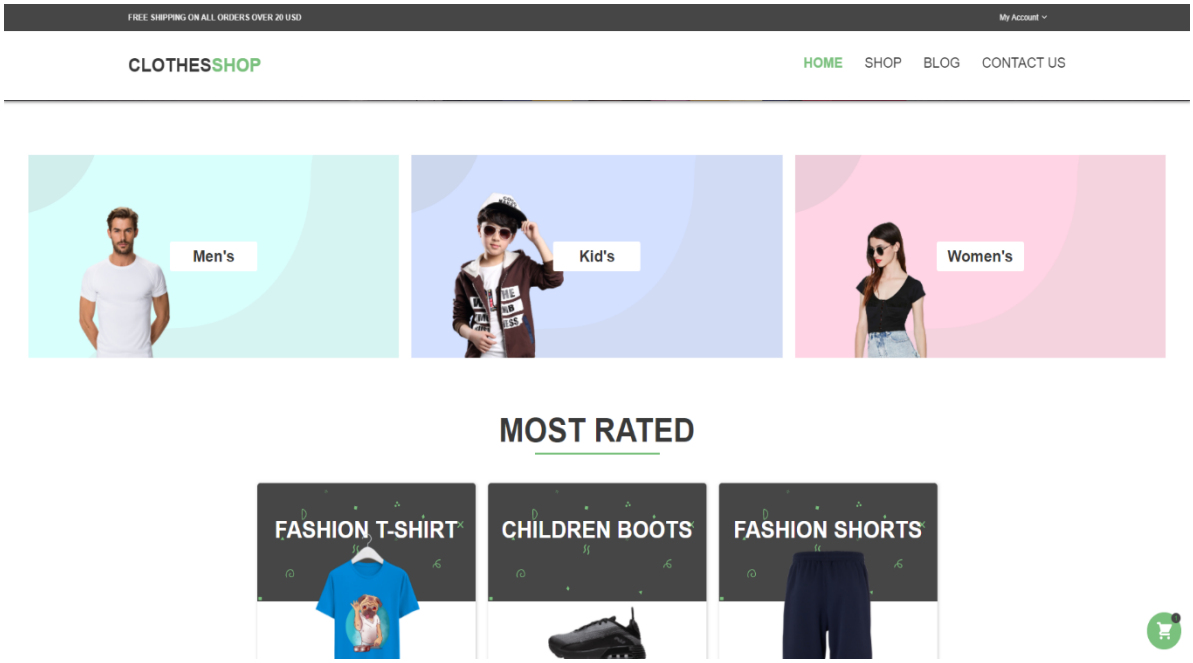

Рис. 2.9. Категорії товарів

Якщо опуститися ще трохи униз, то можна побачити найпопулярніші товари магазину (рис.2.10.). Вони формуються за рейтингом, що передає backend. Також можна клікнути на один з товарів та отримати більш детальну інформацію, подивившись сторінку з описом цього одягу.

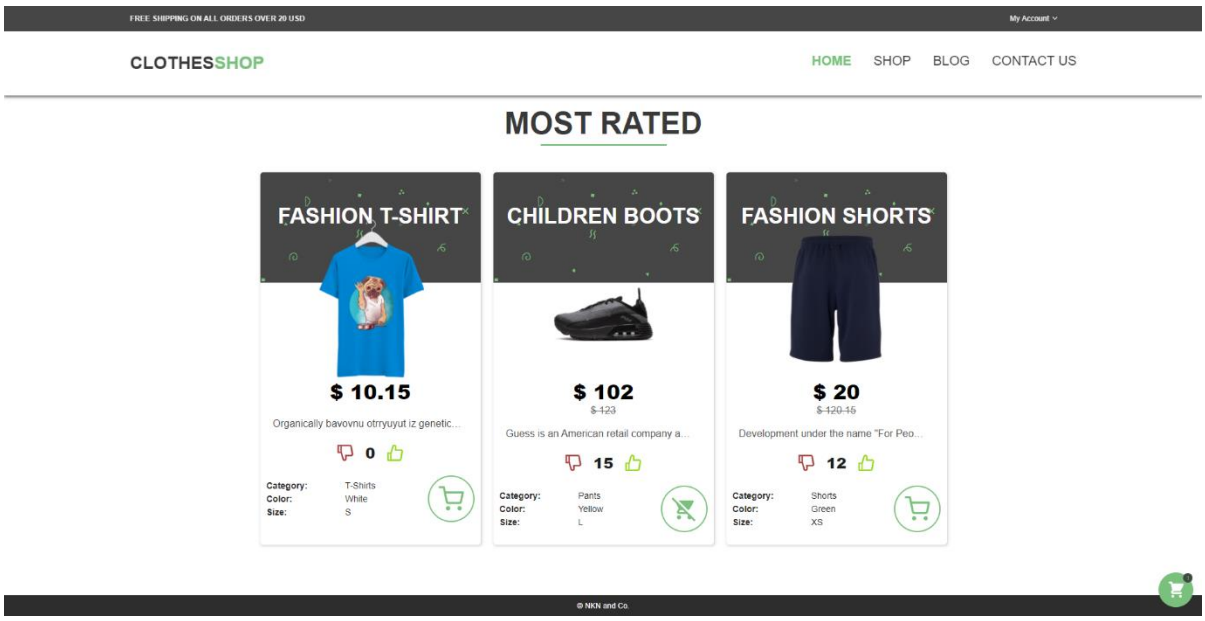

Рис. 2.10. Рейтинг товарів

Коли користувач переходить у вкладку Shop (рис.2.11.) на навігаційній панелі, він потрапляє на сторінку магазину з усіма товарами. На цій сторінці користувач може передивлятися товари, та фільтрувати їх за допомогою меню фільтрів. Також кожен товар може бути доданий до корзини. Зліва від категорій реалізовано пошук товарів за назвою, сортування та виставлення обмежень.

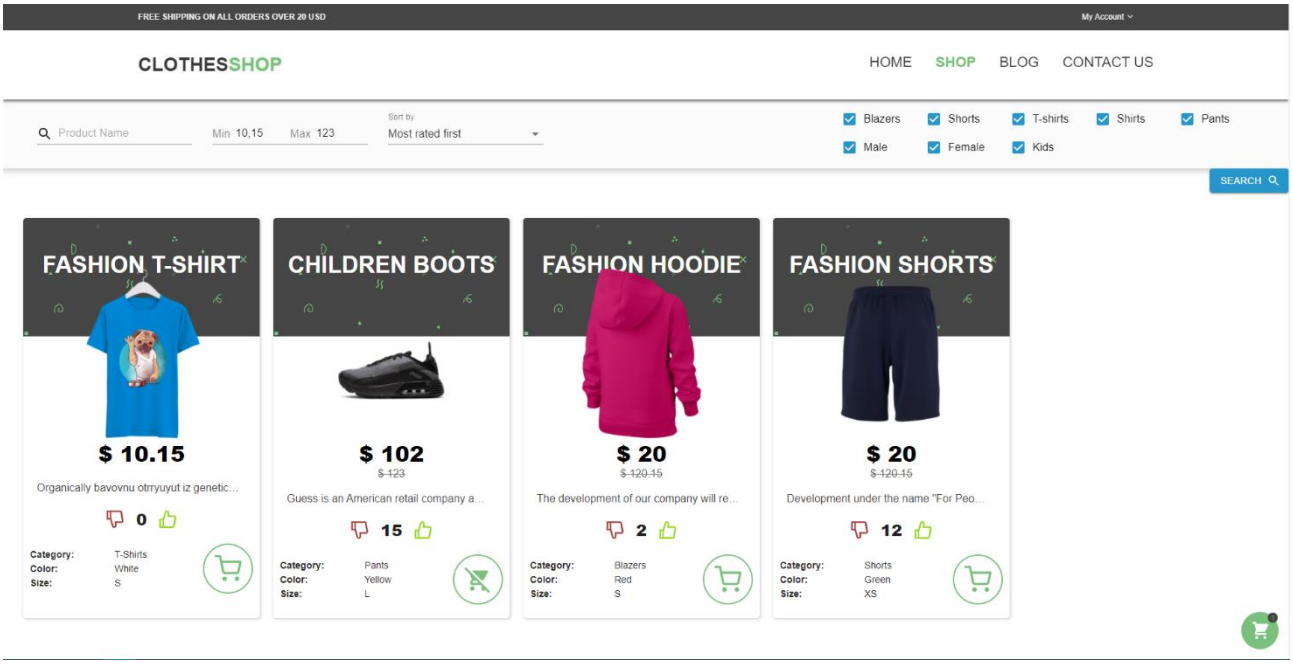

Рис. 2.11. Сторінка товарів

Якщо клієнт натисне на ім’я будь-якого товару то потрапить на сторінку детальної інформації про цей товар (рис.2.12.). На цій сторінці відображено назву та опис товару, розмір та його категорію, ціну та колір тощо. З цієї вкладки також можна або додати товар до кошика, або видалити.

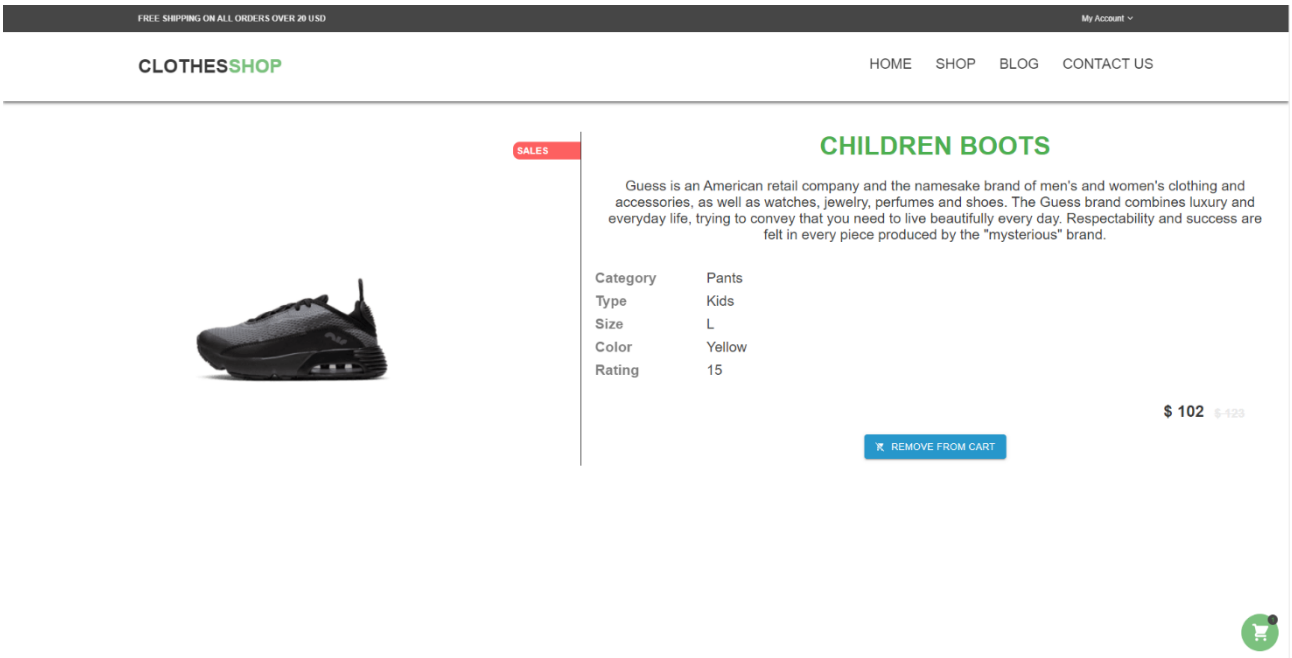

Рис. 2.12. Сторінка деталів товару

Якщо користувач перейде на вкладку Contact Us (рис.2.13.) у навігаційній панелі, він потрапить на сторінку на якій відображено контактні дані власників сайту, та форма зворотного зв’язку. Щоб звернутися з якимось питанням, треба вказати своє ім’я, електронну пошту та текст повідомлення. Тоді користувач може написати своє повідомлення та надіслати його до власників сайту. Незабаром, після відповіді адміністратора, користувач отримає відповідь на свою пошту.

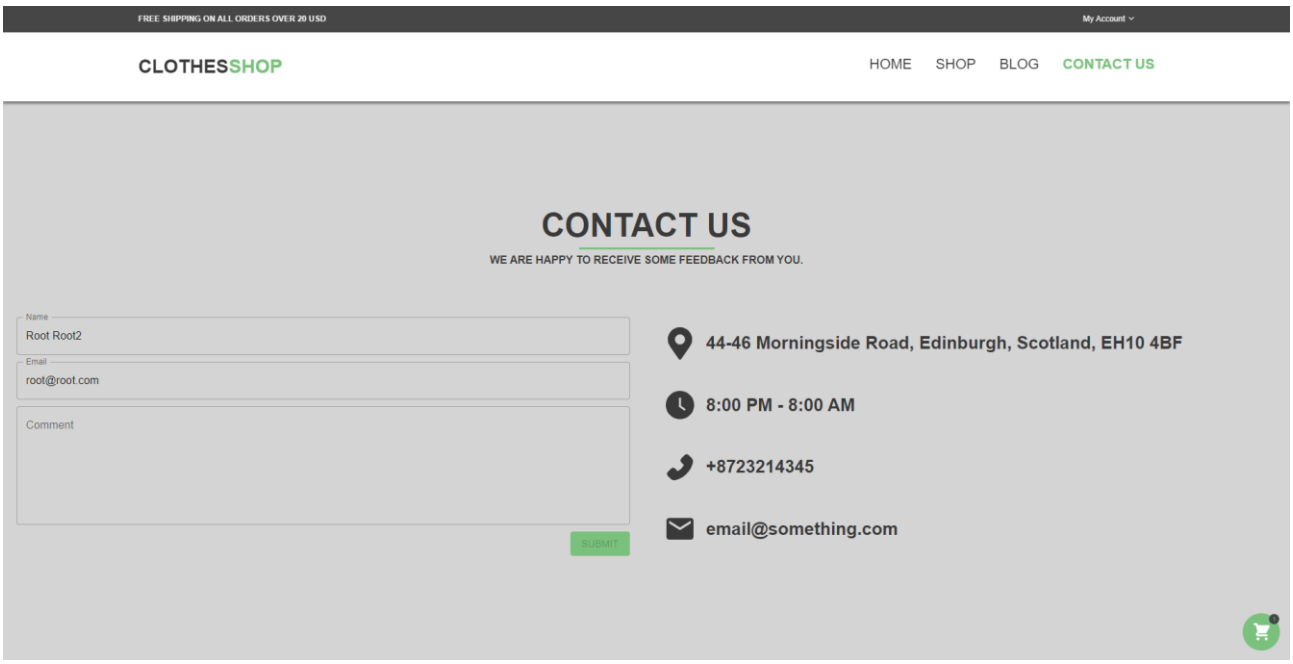

Рис. 2.13. Сторінка контактів

Якщо користувач натисне на кнопку My Account у верхньому навігаційному меню та потім натисне на My Profile він потрапить на сторінку власного профілю (рис.2.14.). На цій сторінці відображено інформацію про користувача, можливо редагувати дані користувача та переглядати усі замовлення що були виконані цим користувачем за час існування профілю.

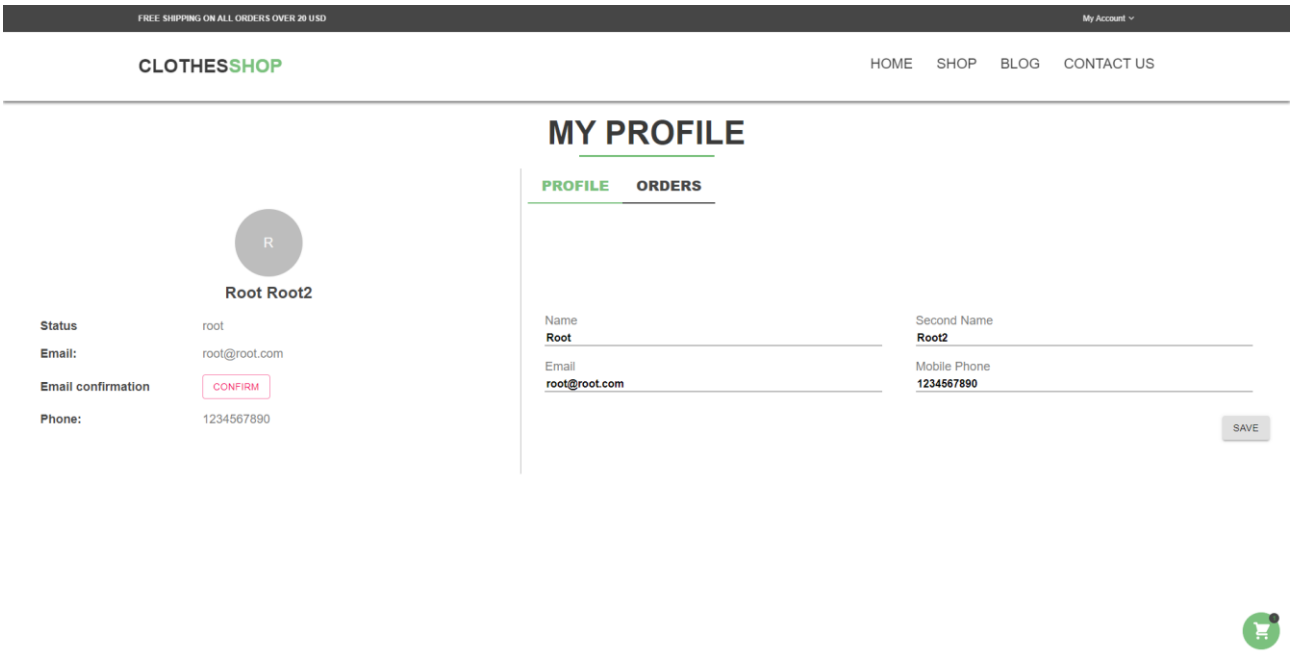

Рис. 2.14. Сторінка профілю користувача

На наступному зображені (рис.2.15.) показано як виглядає таблиця замовлень виконаних користувачем.

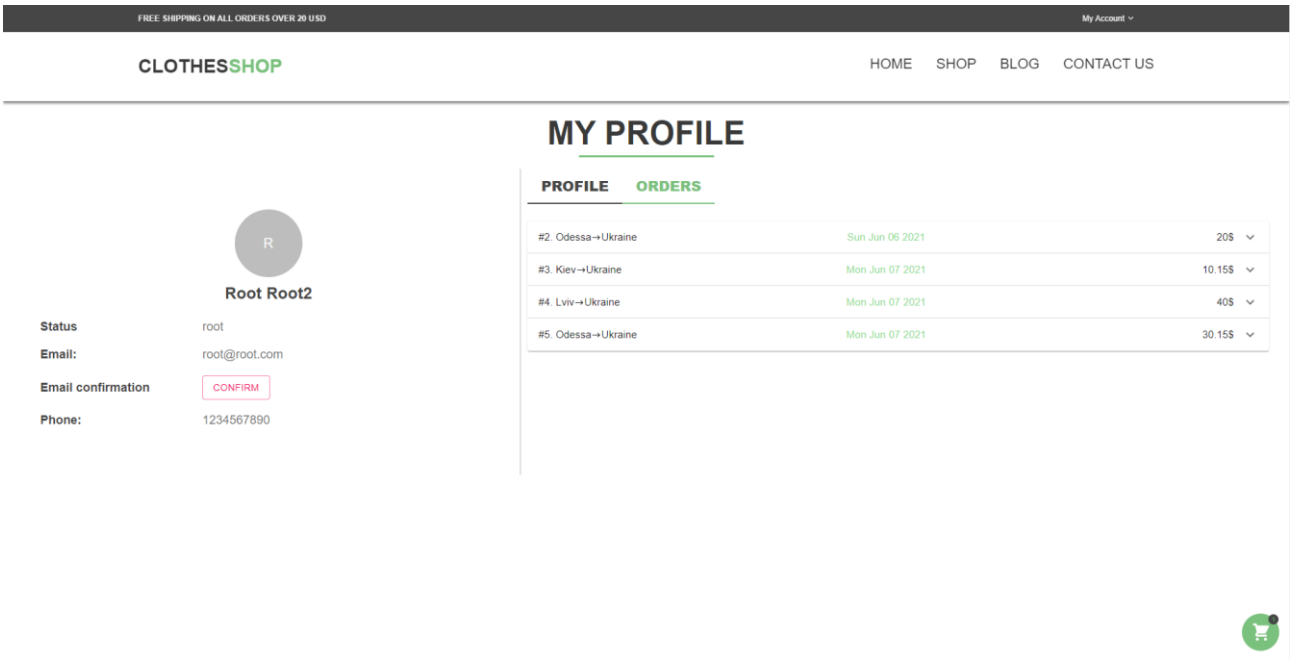

Рис. 2.15. Сторінка профілю користувача

Якщо натиснути на зелену кнопку кошика, яка є на кожній сторінці у правому нижньому кутку, то можна потрапити на її сторінку (рис.2.16.). На сторінці кошика відображено деталі про товари які знаходяться у корзині, та вартість усіх товарів зі знижкою та без неї.

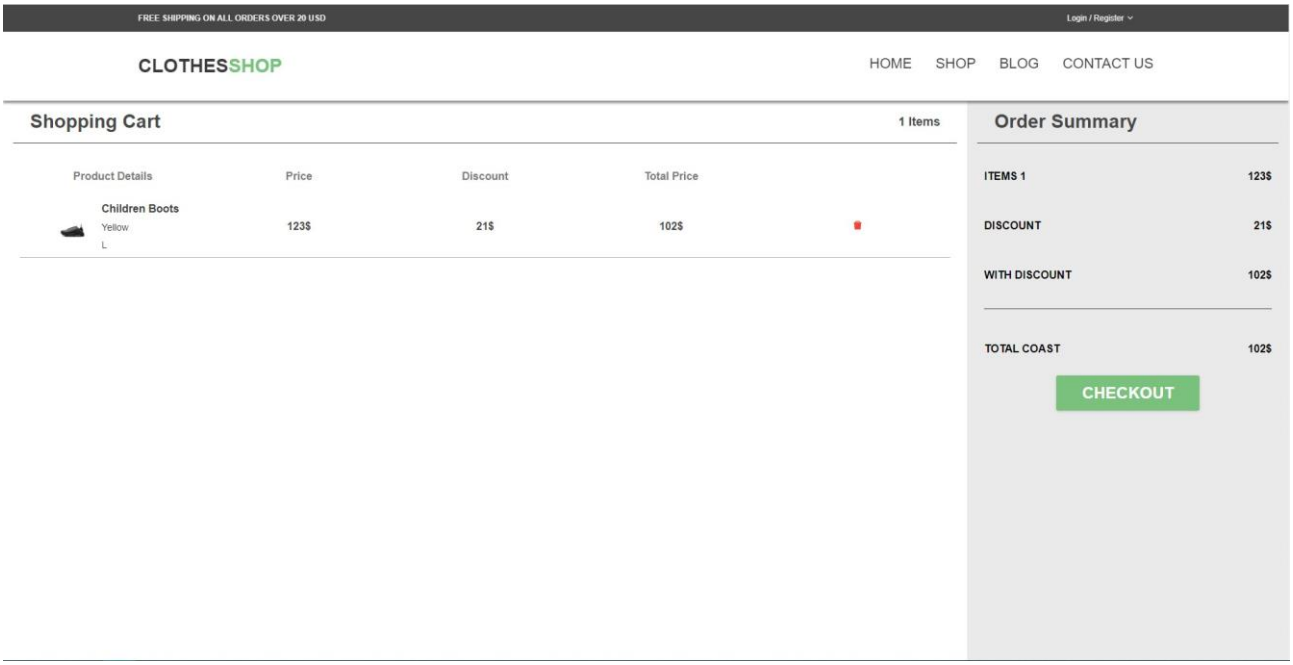

Рис. 2.16. Сторінка замовлення

Якщо користувач натисне кнопку «Checkout» у його корзині, то він побачить модальне вікно у якому він зможе надати потрібну інформацію та виконати замовлення (рис.2.17.).

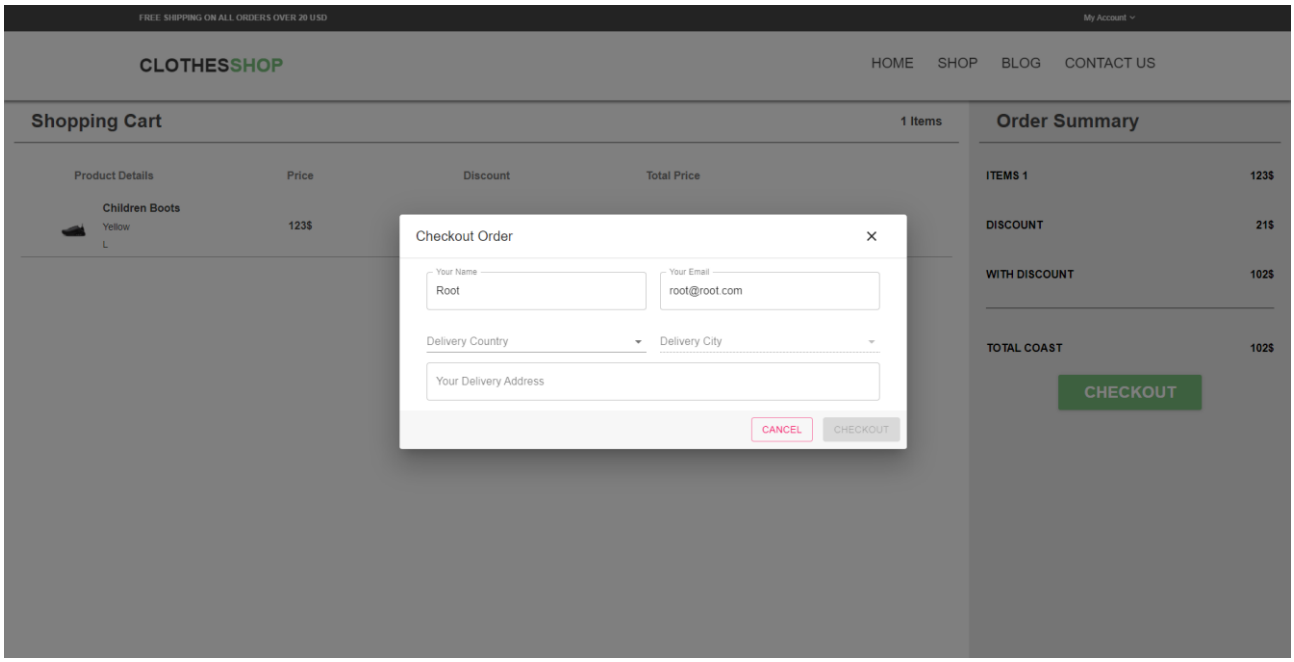

Рис. 2.17. Модальне вікно з деталями замовлення

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 3050;
2. коефіцієнт корекції програми в ході її розробки – 0,05;
3. коефіцієнт складності програми – 1,7;
4. годинна заробітна плата програміста– 125 грн/год;

Середня годинна зарплата Junior FE developer в Україні була вирахувати виходячи з даних «Української спільноти програмістів (DOU)» [10]. Станом на кінець 2020 року зарплата Junior FE розробника простягається від 500$ до 1100$. Вирахувавши середню заробітну плату програміста маємо плату 800$ у місяць. При курсі валют НБУ на початок червня 2021 року один американський долар дорівнює 27,34 грн, тому середня зарплата в гривнях дорівнює 21920 грн. При стандартному графіку (176 годин/місяць) зарплата за годину буде становити близько 125 грн.

5. коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,4;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,4;
7. вартість машино-години ЕОМ –21 грн/год.

Оскільки для цього проекту протрібна велика потужність ПК для бекенду та підняття великої кількості даних на локальному сервері, гарним рішенням буде аренда комп’ютера на час розробки додатку. Вартість аренди комп’ютера на місяць 1300 грн (монітор) та 2400 грн (системний блок). Загалом на місяць оренда коштуватиме 3700 грн. При стандартному графіку (176 годин/місяць) вартість машино-години ЕОМ за годину роботи буде становити 21 грн. В цю

вартість входить ремонт за гарантією та базовий комплект гарнітури (клавіатура та миша) [9].

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:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-витрати праці на програмування по готовій блок-схемі;

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (3050);

C - коефіцієнт складності програми (1,7);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}7 \cdot 3050 \cdot (1 + 0{,}05) = 5444{,}25$$

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

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

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 5 до 8 років він складає 1,4.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1,4, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (5444{,}25 \cdot 1{,}2) / (75 \cdot 1{,}4) = 62{,}22 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k} \text{ , людино-годин, (3.2)}$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 5444{,}25 / (20 \cdot 1{,}4) = 194{,}4 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}$$, людино-годин.

t_n = 5444,25 / (25 · 1,4) = 155,55 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}$$, людино-годин.

$t_{отл}$= 5444,25 / (5 · 1,4) = 777,75 чел.-ч.

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}$$, людино-годин.

$t^k_{отл}$= 1,5 · 777,75 = 1166,6 людино-годин.

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}$$, людино-годин,

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20)\cdot k}$$, людино-годин,

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

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

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 5444,25 / (18 · 1,4) = 245,2 людино-годин.

$t_{до}$ = 0,75 · 245,2 = 183,9 людино-годин.

$t_{д}$ = 245,2 + 183,9 = 429,1 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 62,22 + 194,4 + 155,55 + 777,75 + 429,1 = 1669 людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}$$, грн.

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}$$, грн,

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 125 грн / год, отримуємо:

$$З_{ЗП} = 1669 \cdot 125 = 208\ 625 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (21 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 777{,}75 \cdot 21 = 16\ 332{,}75 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 208\ 625 + 16\ 332{,}75 = 224\ 957{,}75 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k- число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 1669 / 1 \cdot 176 \approx 9{,}48 \text{ міс.}$$

Висновок: програмне забезпечення розроблено для забезпечення доступу користувачів до основних програм та пропозицій компанії продажу одягу, ефективної взаємодії між потенційними споживачами та компанією, створення позитивного іміджу компанії, підвищення продажу товарів. Вартість даного програмного забезпечення близько 224 957,75 тис. грн і не вимагає додаткових витрат як при розробці програми. Очікуваний час розробки становить 1669 годин, тобто 9,48 місяці. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було поставлено завдання розробити frontend частину системи відображення та адміністрування інтернет-магазину одягу.

Це програмне забезпечення призначене для надання універсального інструменту для відображення та управлінням контенту інтернет-магазину, управління магазину з адміністративної панелі. Практичне призначення даної системи полягає в забезпеченні відвідувачам сайту простого і комфортного доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту, що зробить процес покупки швидшим і зручнішим, і підвищить ефективність роботи інтернет-магазину.

Під час виконання даного проекту були виконані наступні задачі:

- вивчено предметну галузь розв'язуваної задачі;
- створено алгоритм для реалізації поставленого завдання;
- створено базу даних і клієнтську та серверну програму, що працює разом.

Розроблене програмне забезпечення дозволяє:

- легке та доступне адміністрування інтернет магазину;
- формування web-сторінок на основі шаблону з використанням контенту полученого з серверу;
- формування web-сторінок на основі шаблону з використанням контенту з бази даних;
- контакування за адміністратором данного магазину;
- оформлення замовлень з данного магазину.

Програма реалізована на базі фреймворка ReactJS та з використанням мови програмування JavaScript.

Для організації структури проекту був обраний компонентний підхід, що дозволяє перевикористовувати однакові частини коду у декількох місцях.

Також у кваліфікаційній роботі було визначено трудомісткість розробленої системи, на базі середьої зарплати розробника проведений підрахунок вартості роботи по створенню програми, який складає 224 957,75 грн та розраховано час на створення інтернет-магазину - 1669 людино-годин , тобто 9,48 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вебінар на тему веб-додатків та провідних веб-технологій [Електронний ресурс] - Режим доступу: https://dou.ua/calendar/37625/
2. Скотт Б. Проектирование веб-интерфейсов / Б. Скотт, Т. Нейл. —Спб.: Символ-Плюс, 2010. — 352 с.
3. Эрик Фримен, Элизабет Робсон. Head First JavaScript Programming – С.: O'Reilly,2010. – 267 с.
4. Марк Тиленс Томас. React в действии. Питер, 2018. -368 с. ISBN - 5446109996, 9785446109999
5. Скотт Б. Проектирование веб-интерфейсов / Б. Скотт, Т. Нейл. —Спб.: Символ-Плюс, 2018 — 352 с.
6. Стоян Стефанов. React.js. Быстрый старт, 2017. Бестселлеры O'Reilly. -304 с. ISBN - 978-5-496-03003-8
7. Інькова Н. А. Створення Web-сайтів: Навчально-методичний посібник [Електронний ресурс] / Інькова Н.А., Зайцева Е.А., Кузьміна Н.В., Толстих С.Г. / / Режим доступу: http://club-edu.tambov.ru/methodic/fio/p5.doc
8. М. Хавербеке – Выразительный JavaScript. Современное веб-программирование 2018. –С. 125-329.
9. Вартість аренди ноутбуку почасово [Електронний ресурс] - Режим доступу: https://notebooksbu.com/garantiya-3-goda/
10. Середня заробітна плата Junior FE developer в Україні станом на початок 2021 року. https://dou.ua/lenta/articles/salary-report-devs-june-2020/
11. Стоян Стефанов React. Быстрый старт (2016)
12. Fullstack React Native, Devin Abbott, Houssein Djirdeh, Anthony Accomazzo, and Sophia Shoemaker, 2017
13. Молер, Дж. Flash 8. Руководство Web-дизайнера; Эксмо - М., 2019. - 400 с.

14.Веб-приложение и его виды [Електронний ресурс] - Режим доступу: https://semantica.in/blog/veb-prilozhenie.html

15.Website и Web Application: в чем разница? [Електронний ресурс] - Режим доступу: https://dinarys.com/ru/blog/websitevs.-webapplication

16. Разработка мобильных и веб-приложений: что является лучшим решением [Електронний ресурс] - Режим доступу: https://smartum.pro/ru/blog-ru/razrabotka-mobilnykh-i-webprilozheniy/

17.Орлов Л. А. Як створити електронний магазин в Інтернет / Л. А. Орлов. – М.: БУК-ПРЕС, 2016. – 384 с

18.Стотлемайер Д. Тестирование Web-приложений / Д. Стотлемайер – М.: Кудиц-образ, 20017. – 240 с.

19.К. Дуглас – Как устроен JavaScript, 2019 - 394 с.

20.Васильев А.Н. JavaScript в примерах и задачах / А.Н. Васильев. – Москва: Издательство „Э”, 2017. – 720 с.

21.Пасічник Н.Р. Формалізм в постановці задачі створення якісного сайту / Н.Р. Пасічник // Наукові праці Донецького національного технічного університету. Інформатика, кібернетика та обчислювальна техніка. – Донецьк. – 2017. – Вип. 14 (188). – С. 325-329.

22.Горнаков, С.Г. Осваиваем популярные системы управления сайтом / Горнаков С.Г. – ДМК Пресс, 2019 – С. 336.

23.Савельева Н. Системы управления контентом / Савельева Н. // Открытые системы, 2020. — С. 41-47.

24.Комисаров Д.А., Станкевич А.Г. Персональный учитель по персональному компьютеру. – М, 2020.

25.Леонтьєв В.П. «Web-дизайн. Керівництво користувача »

26.Гукин Д. - FrontPage для "чайников". - К.,2016.- С. 102-123.

27.Э. Фримен, Э. Робсон – Изучаем программирование на JavaScript 2017. – 550 с

28.Э. Браун – Изучаем JavaScript. Руководство по созданию современных веб-сайтов 2019. — С. 35-47

29.React [Електронний ресурс] - Режим доступу: https://ru.react.js.org

30.Основы React.js [Електронний ресурс] - Режим доступу: https://learn.javascript.ru/screencast/react

ДОДАТОК А

КОД ПРОГРАМИ

Index.html // головний HTML файл

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Shop App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="shop-app"></div>
  </body>
</html>
```

Constants.js// файл з константами усього додатку

```
export const APP_HEADER = {
  shippingInfo: "Free shipping on all orders over",
  loginRegister: "Login / Register",
  accountDropDownTitle: "My Account",
  minShippingPrice: "20 usd",
  registration: "Create new account",
  login: "Login",
  navLinks: {
    "/": "Home",
    "/shop": "Shop",
    "/blog": "Blog",
    "/contacts": "Contact Us",
  },
};
export default APP_HEADER;
```

Index.js// головний JS файл

```
import React from "react";
import ReactDOM from "react-dom";
import { Provider } from "react-redux";
import App from "./App";
import store from "./store";
store()
  .then((_store) => {
    ReactDOM.render(
      // eslint-disable-next-line react/jsx-filename-extension
      <Provider store={_store}>
        <App />
      </Provider>,
      document.getElementById("shop-app")
    );
  })
```

```
  .catch((error) => {
    // eslint-disable-next-line no-console
    console.error(error.message);
  });
```

RootReduser.js// файл для опису основної навігації

```
import { combineReducers } from "redux";
import { rootComponentReducer } from "./pages/RootComponent";
import { loginReducer } from "./pages/LoginPage";
import { mainPageReducer } from "./pages/MainPage";
import { registrationReducer } from "./pages/RegistrationPage";
import { adminPanelReducer } from "./pages/AdminPanelPage";
import { contactUsReducer } from "./pages/ContactUsPage";
import { shopReducer } from "./pages/ShopPage";
import { cartReducer } from "./pages/CartPage";
import { profilePageReducer } from './pages/ProfilePage';
import { productReducer } from './pages/ProductPage';
export default combineReducers({
  rootComponentReducer,
  loginReducer,
  mainPageReducer,
  registrationReducer,
  adminPanelReducer,
  contactUsReducer,
  shopReducer,
  cartReducer,
  profilePageReducer,
  productReducer,
});
```

Cart.jsx// основний файл компоненту кошика

```
import React, { useEffect, useState } from "react";
import PropTypes from "prop-types";
import { HiShoppingCart } from "react-icons/hi";
import { Link, useLocation } from "react-router-dom";
import { connect } from "react-redux";
import useStyles from "./cart.styles";
const Cart = ({ cartData }) => {
  const classes = useStyles();
  const location = useLocation();
  const [isShowed, setIsShowed] = useState(location.pathname !== "/cart");
  useEffect(() => {
    if (location.pathname === "/cart") {
      setIsShowed(false);
    } else {
      setIsShowed(true);
    }
  }, [location.pathname]);
  if (!isShowed) {
    return null;
  }
  return (
```

```
    <Link to="/cart" className={classes.cartIconWrapper}>
      <div className={classes.cartCounter}>{cartData.length}</div>
      <HiShoppingCart />
    </Link>
  );
};
Cart.propTypes = {
  cartData: PropTypes.arrayOf(PropTypes.number).isRequired,
};
const mapStateToProps = (state) => ({
  cartData: state.cartReducer.data,
});
export default connect(mapStateToProps)(Cart);
```

Cart.styles.js// файл стилів для компоненту кошика

```
import { makeStyles } from "@material-ui/core";

import * as COLORS from "../../global/styles/palette";
const useStyles = makeStyles(() => ({
  cartIconWrapper: {
    position: "fixed",
    bottom: "15px",
    right: "15px",
    backgroundColor: COLORS.GREEN_LIGHT,
    width: "55px",
    height: "55px",
    borderRadius: "50%",
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    border: `1px solid ${COLORS.GREEN_LIGHT}`,
    transition: "0.3s",

    "& svg": {
      color: COLORS.WHITE,
      width: "30px",
      height: "30px",
    },

    "&:hover": {
      border: `1px solid ${COLORS.WHITE}`,
      backgroundColor: COLORS.WHITE,
      cursor: "pointer",

      "& svg": {
        color: COLORS.GREEN_LIGHT,
      },
    },
  },
  cartCounter: {
    position: "absolute",
    width: "15px",
```

```
    height: "15px",
    backgroundColor: COLORS.BLACK_LIGHT,
    borderRadius: "50%",
    color: COLORS.GREEN,
    fontSize: "10px",
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    top: 0,
    right: 0,
  },
}));
export default useStyles;
```

Button.jsx// основний файл компоненту кнопки

```
import React from "react";
import PropTypes from "prop-types";
import { StyledButton } from "./Button.style";
const Button = ({ type, htmlType, onClick, content, disabled }) => {
  return (
    <StyledButton type={htmlType} styleType={type} onClick={onClick} disabled={disabled}>
      {content}
    </StyledButton>
  );
};

Button.propTypes = {
  type: PropTypes.string,
  htmlType: PropTypes.string,
  content: PropTypes.oneOfType([PropTypes.element, PropTypes.string]),
  onClick: PropTypes.func,
  disabled: PropTypes.bool,
};

Button.defaultProps = {
  type: "primary",
  htmlType: "button",
  content: null,
  onClick: () => {},
  disabled: false,
};
export default Button;
```

Button.styles.js// файл стилів для компоненту кнопки

```
import style from "styled-components";
import { GREEN_LIGHT, GREEN_LIGHT_2, WHITE } from "../../global/styles/palette";
// eslint-disable-next-line import/prefer-default-export
export const StyledButton = style.button`
  ${(props) => {
    switch (props.styleType) {
      default:
        return `
```

```
      background-color: ${GREEN_LIGHT};
      color: ${WHITE};
    `;
  }
}}
 border: none;
 outline: none;
 border-radius: 3px;
 box-shadow: 0px 1px 4px 0px rgba(0, 0, 0, 0.19);
 font-size: 32px;
 text-transform: uppercase;
 font-weight: bold;
 padding: 12px 38px;
 cursor: pointer;
 transition: 0.3s;
 &:hover {
  background-color: ${GREEN_LIGHT_2};
 }
`;
```

Dropdown.jsx// основний файл випадаючого списка

```
import React, { useState, useEffect } from "react";
import PropTypes from "prop-types";
import { DropdownBody, DropdownTitle, DropdownWrapper } from "./Dropdown.style";
import { ChevronDown } from "../Select/Select.style";
const Dropdown = (props) => {
 // eslint-disable-next-line react/prop-types

 const { title, children, isCloseOutside } = props;
 const [isBodyShow, setIsBodyShow] = useState(false);
 const handleCloseByEsc = (event) => {
  if (event.key === "Escape") {
   setIsBodyShow(false);
  }
 };
 const handleCloseByClickOutside = (event) => {
  const isClickOutsideHere = !event.target.closest(".dropdown__wrapper");
  if (isClickOutsideHere && isCloseOutside) {
   setIsBodyShow(false);
  }
 };
 useEffect(() => {
  document.addEventListener("click", handleCloseByClickOutside);
  document.addEventListener("keydown", handleCloseByEsc);
  return () => {
   document.removeEventListener("click", handleCloseByClickOutside);
   document.removeEventListener("keydown", handleCloseByEsc);
  };
 }, []);
 return (
  <DropdownWrapper isBodyShow={isBodyShow} className="dropdown__wrapper">
   <DropdownTitle
    isBodyShow={isBodyShow}
```

```
      onClick={() => setIsBodyShow((prevState) => !prevState)}
    >
      {title}
      <ChevronDown />
    </DropdownTitle>
    <DropdownBody isBodyShow={isBodyShow}>{children}</DropdownBody>
  </DropdownWrapper>
 );
};

Dropdown.propTypes = {
 title: PropTypes.string.isRequired,
 isCloseOutside: PropTypes.bool,
};
Dropdown.defaultProps = {
 isCloseOutside: true,
};
export default Dropdown;
```

Header.jsx// основний файл компонента хедера

```
import React from "react";
import propTypes from "prop-types";
import { Formik, Field, ErrorMessage } from "formik";
import { connect } from "react-redux";
import {
 loginUserRequest as loginUserRequestAction,
 logoutUser as logoutUserAction,
} from "../../pages/RootComponent/actions";
import {
 ControlsInfo,
 ControlsRow,
 ControlsSelectorsWrapper,
 RegistrationLink,
 StyledFormikForm,
 HeaderWrapper,
} from "./Header.style";
import { APP_HEADER } from "../../constants";
import HeaderNavigation from "./components/HeaderNavigation";
import Dropdown from "../Dropdown";
import loginValidationSchema from "./validation";
import Spinner from "../Spinner";

import AccountControls from "./components/AccountControls";
import Cart from "../Cart";
const Header = ({ loginUserRequest, logoutUserRequest, user, loading }) => {
 const handleLogin = (values) => {
  loginUserRequest(values);
 };
 return (
  <HeaderWrapper>
   <ControlsRow>
    <ControlsInfo>
```

```
 {APP_HEADER.shippingInfo} {APP_HEADER.minShippingPrice}
</ControlsInfo>
<ControlsSelectorsWrapper>
 <Dropdown
  title={
   user ? APP_HEADER.accountDropDownTitle : APP_HEADER.loginRegister
  }
  isCloseOutside
 >
  {loading ? (
   <Spinner />
  ) : (
   <>
    {user ? (
     <AccountControls
      userRole={user.role.id}
      onExit={logoutUserRequest}
     />
    ) : (
     <>
      <Formik
       initialValues={{
        email: "",
        password: "",
       }}
       validationSchema={loginValidationSchema}
       onSubmit={handleLogin}
      >
       {({ values, handleChange, handleBlur }) => (
        <StyledFormikForm>
         <Field
          className="form__input"
          name="email"
          type="text"
          placeholder="email"
          value={values.email}
          onChange={handleChange}
          onBlur={handleBlur}
         />
         <ErrorMessage
          name="email"
          render={(msg) => (
           <div className="input-validation-message">
            {msg}
           </div>
          )}
         />
         <Field
          className="form__input"
          name="password"
          type="password"
          placeholder="password"
```

```
                  value={values.password}
                  onChange={handleChange}
                  onBlur={handleBlur}
                />
                <ErrorMessage
                  name="password"
                  render={(msg) => (
                    <div className="input-validation-message">
                      {msg}
                    </div>
                  )}
                />
                <button type="submit" className="btn__primary">
                  {APP_HEADER.login}
                </button>
              </StyledFormikForm>
            )}
          </Formik>
          <RegistrationLink to="/registration">
            {APP_HEADER.registration}
          </RegistrationLink>
        </>
      )}
    </>
  )}
  </Dropdown>
  <Cart />
 </ControlsSelectorsWrapper>
 </ControlsRow>
 <HeaderNavigation />
 </HeaderWrapper>
 );
};
const mapStateToProps = (state) => ({
 user: state.rootComponentReducer.loginUser.user,
 loading: state.rootComponentReducer.loginUser.loading,
});
const mapDispatchToPros = {
 loginUserRequest: loginUserRequestAction,
 logoutUserRequest: logoutUserAction,
};
Header.propTypes = {
 loginUserRequest: propTypes.func.isRequired,
 logoutUserRequest: propTypes.func.isRequired,
 user: propTypes.shape({
  name: propTypes.string,
  second_name: propTypes.string,
  role: propTypes.shape({
   id: propTypes.number,
  }),
 }),
 loading: propTypes.bool,
```

```
};

Header.defaultProps = {
  user: null,
  loading: false,
};
export default connect(mapStateToProps, mapDispatchToPros)(Header);
```

Header.styles.js// файл стилів для компонента хедера

```
import styled from "styled-components";
import { Form } from "formik";
import { Link } from "react-router-dom";
import {
  BLACK_LIGHT,
  GRAY_LIGHT,
  RED,
  WHITE_DARK,
} from "../../global/styles/palette";
export const HeaderWrapper = styled.nav`
  width: 100%;
  position: sticky;
  left: 0;
  top: 0;
  z-index: 100;
`;
export const ControlsRow = styled.div`
  display: flex;
  justify-content: space-between;
  background-color: ${BLACK_LIGHT};
  padding: 12px 200px 13px;

 @media screen and (max-width: 1280px) {
     padding: 12px 50px;
  }
    @media screen and (max-width: 640px) {
       justify-content: flex-end;
       padding: 12px 15px;
    }
`;

export const ControlsInfo = styled.div`
  font-size: 12px;
  font-weight: bold;
  color: ${WHITE_DARK};
  text-transform: uppercase;
  display: flex;
  align-items: center;
    @media screen and (max-width: 640px) {
       display: none;
    }
`;
export const ControlsSelectorsWrapper = styled.div`
```

```
  display: flex;
`
export const RegistrationLink = styled(Link)`
  font-size: 12px;
  color: ${GRAY_LIGHT};
  text-decoration: none;
  width: 100%;
  display: block;
  text-align: right;
  padding-right: 5px;
`;
export const StyledFormikForm = styled(Form)`
  & input {
    font-size: 14px;
    width: 150px;
    margin-bottom: 5px;
  }
 & .input-validation-message {
    color: ${RED};
    font-size: 12px;
    text-align: left;
    line-height: 1;
    display: block;
    margin-top: -4px;
    margin-bottom: 5px;
  }
 .btn__primary {
    font-size: 14px;
    width: 100%;
    padding: 5px 15px;
  }

 margin-bottom: 10px;
`;
```

ProductCard.jsx// основний файл компонента продукту

```
import React from "react";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import {

  FiShoppingCart,
  AiOutlineFileImage,
  AiOutlineLike,
  AiOutlineDislike,
  MdRemoveShoppingCart,
} from "react-icons/all";
import { Link } from 'react-router-dom';
import {
  addToCart as addToCartAction,
  removeFromCart as removeFromCartAction,
} from "../../pages/CartPage/actions";
```

```
import useStyles from "./ProductCard.styles";
import PRODUCT_CARD from "./constants";
const ProductCard = ({ product, addToCart, removeFromCart, cartData }) => {
  const classes = useStyles();
  const handleAddProductToCart = (id) => {
    const cart = JSON.parse(localStorage.getItem('nkn_cart') || '[]');
    localStorage.setItem('nkn_cart', JSON.stringify([...cart, id]));
    addToCart(id);
  };
  const handleRemoveProductFromCart = (id) => {
    const cart = JSON.parse(localStorage.getItem('nkn_cart') || '[]');
    localStorage.setItem('nkn_cart', JSON.stringify(cart.filter((p) => p !== id)));
    removeFromCart(id);
  };
  const isInCart = cartData.includes(product.id);
  return (
    <div className={classes.productCardWrapper}>
      <div className="product-card__header">
        <Link to={`/product/${product.id}`}><div className="product-
card__header__title">{product.name}</div></Link>
      </div>
      <div className="product-card__body">
        <div className="product-card__image">
          {product.image ? (
            <img src={product.image} alt={product.name} />
          ) : (
            <span>
              <AiOutlineFileImage />
              {PRODUCT_CARD.noImage}
            </span>
          )}

        </div>
        <div className="product-card__price">
          <div className="price__current">$ {product.price - product.discount}</div>
          {product.discount ? <div className="price__prev">$ {product.price}</div> : null}
        </div>
        <div className="product-card__description">{product.description}</div>
      </div>
      <div className="product-cad__rating">
        <AiOutlineDislike />
        <span>{product.rating}</span>
        <AiOutlineLike />
      </div>
      <div className="product-card__footer">
        <div className="product-characteristics">
          <div className="characteristics">
            <div className="characteristics__name">{PRODUCT_CARD.category}</div>
            <div className="characteristics__value category">
              {product.category.name}
            </div>
          </div>
```

```
        <div className="characteristics">
          <div className="characteristics__name">{PRODUCT_CARD.color}</div>
          <div className="characteristics__value color">{product.color}</div>
        </div>
        <div className="characteristics">
          <div className="characteristics__name">{PRODUCT_CARD.size}</div>
          <div className="characteristics__value size">
            {product.size.name}
          </div>
        </div>
      </div>

      <div className="add-to-cart-btn">
        {isInCart ? (
          <MdRemoveShoppingCart
            onClick={() => handleRemoveProductFromCart(product.id)}
          />
        ) : (
          <FiShoppingCart
            onClick={() => handleAddProductToCart(product.id)}
          />
        )}
      </div>
    </div>
  </div>
 );
};
ProductCard.propTypes = {
  product: PropTypes.shape({
    id: PropTypes.number,
    name: PropTypes.string,
    description: PropTypes.string,
    image: PropTypes.string,
    size: PropTypes.shape({
      name: PropTypes.string,
    }),
    price: PropTypes.number,
    discount: PropTypes.number,
    rating: PropTypes.number,
    color: PropTypes.string,
    category: PropTypes.shape({
      name: PropTypes.string,
    }),
  }).isRequired,
  addToCart: PropTypes.func.isRequired,
  removeFromCart: PropTypes.func.isRequired,
  cartData: PropTypes.arrayOf(PropTypes.number).isRequired,
};
const mapDispatchToProps = {
  addToCart: addToCartAction,
  removeFromCart: removeFromCartAction,
};
```

```
const mapStateToProps = (state) => ({
  cartData: state.cartReducer.data,
});
export default connect(mapStateToProps, mapDispatchToProps)(ProductCard);
```

Table.jsx// основний файл компонента таблиці

```
import React, { useState } from "react";
import PropTypes from "prop-types";
import {
  Table as MaterialTable,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TablePagination,
  TableRow,
} from "@material-ui/core";
import useStyles from "./Table.styles";
import TABLE from "./constants";
const Table = ({ columns, rows }) => {
  const classes = useStyles();
  const [page, setPage] = useState(0);
  const [rowsPerPage, setRowsPerPage] = useState(10);
  const handleChangePage = (event, newPage) => {
    setPage(newPage);
  };
  const handleChangeRowsPerPage = (event) => {
    setRowsPerPage(+event.target.value);
    setPage(0);
  };
  return (
    <div className={classes.tableWrapper}>
      <TableContainer className={classes.container}>
        <MaterialTable stickyHeader aria-label="sticky table">
          <TableHead>
            <TableRow>
              {columns.map((column) => (
                <TableCell
                  key={column.id}
                  align={column.align}
                  style={{ minWidth: column.minWidth }}
                >
                  {column.label}
                </TableCell>
              ))}
            </TableRow>
          </TableHead>
          <TableBody>
            {!rows.length && (
              <tr className="table__empty-data">
                <td colSpan={columns.length}>{TABLE.noData}</td>
              </tr>
```

```
          )}
          {rows
            .slice(page * rowsPerPage, page * rowsPerPage + rowsPerPage)
            .map((row) => {
              return (
                <TableRow hover role="checkbox" tabIndex={-1} key={row.id}>
                  {columns.map((column) => {
                    const value = row[column.id];
                    return (
                      <TableCell key={column.id} align={column.align}>
                        {column.format && typeof value === "number"
                          ? column.format(value)
                          : value}
                      </TableCell>
                    );
                  })}
                </TableRow>
              );
            })}
        </TableBody>
      </MaterialTable>
    </TableContainer>
    <TablePagination
      rowsPerPageOptions={[10, 25, 100]}
      component="div"
      count={rows.length}
      rowsPerPage={rowsPerPage}
      page={page}
      onChangePage={handleChangePage}
      onChangeRowsPerPage={handleChangeRowsPerPage}
    />
  </div>
 );
};
Table.propTypes = {
 columns: PropTypes.arrayOf(
  PropTypes.shape({
   id: PropTypes.string.isRequired,
   label: PropTypes.string.isRequired,
   align: PropTypes.string,
   minWidth: PropTypes.number,
  })
 ).isRequired,
 rows: PropTypes.arrayOf(PropTypes.shape({})).isRequired,
};
export default Table;
```

AdminPanel.jsx// основний файл сторінки адмінпанелі

```
import React, { useEffect, useState } from "react";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import { Switch, Route, Redirect, Link } from "react-router-dom";
```

```
import clsx from "clsx";
import {
  Drawer,
  AppBar,
  Toolbar,
  List,
  CssBaseline,
  Typography,
  Divider,
  IconButton,
  ListItemIcon,
  ListItemText,
  useTheme,
  ListItem,
} from "@material-ui/core";
import ChevronRightIcon from "@material-ui/icons/ChevronRight";
import ChevronLeftIcon from "@material-ui/icons/ChevronLeft";
import MenuIcon from "@material-ui/icons/Menu";
import { getUserByTokenRequest as getUserByTokenRequestAction } from
"../RootComponent/actions";
import Loader from "../../components/Loader";
import useStyles from "./AdminPanel.styles";
import ADMIN_PANEL, { ADMIN_PANEL_ROUTES } from "./constants";
import ProductsPage from "./components/ProductsPage/ProductsPage";
import LabelsPage from "./components/LabelesPage";
import OrdersPage from "./components/OrdersPage";
import OrderDetailsPage from "./components/OrdersPage/components/OrderDetailsPage";
import FeedbackPage from "./components/FeedbacksPage";
import UsersPage from './components/UsersPage/UsersPage';
import OverviewPage from './components/OverviewPage';
import SizesPage from './components/SizesPage/SizesPage';
import RolesPage from './components/RolesPage/RolesPage';
const AdminPanelPage = ({
  userData,
  getUserByTokenRequest,
  loadingRoot,
  token,
  history,
}) => {
  const classes = useStyles();
  const theme = useTheme();
  const [open, setOpen] = useState(false);
  const handleDrawerOpen = () => {
    setOpen(true);
  };

  const handleDrawerClose = () => {
    setOpen(false);
  };
  useEffect(() => {
    getUserByTokenRequest(token);
  }, [getUserByTokenRequest, token]);
```

```
useEffect(() => {
  if (userData && userData?.role?.id !== 4 && userData?.role?.id !== 3) {
    history.push("/");
  }
}, [userData, history]);
if (loadingRoot) {
  return <Loader />;
}
return (
  <div className={classes.root}>
    <CssBaseline />
    <AppBar
      position="fixed"
      className={clsx(classes.appBar, {
        [classes.appBarShift]: open,
      })}
    >
      <Toolbar style={{ display: "flex", justifyContent: "space-between" }}>
        <div className={classes.toolbarHeader}>
          <IconButton
            color="inherit"
            aria-label="open drawer"
            onClick={handleDrawerOpen}
            edge="start"
            className={clsx(classes.menuButton, {
              [classes.hide]: open,
            })}
          >
            <MenuIcon />
          </IconButton>
          <Typography variant="h6" noWrap>
            {ADMIN_PANEL.title}
          </Typography>
        </div>
        <Link to="/" className={classes.linkHeader}>
          {ADMIN_PANEL.link}
        </Link>
      </Toolbar>
    </AppBar>
    <Drawer
      variant="permanent"
      className={clsx(classes.drawer, {
        [classes.drawerOpen]: open,
        [classes.drawerClose]: !open,
      })}
      classes={{
        paper: clsx({
          [classes.drawerOpen]: open,
          [classes.drawerClose]: !open,
        }),
      }}
    >
```

```
      <div className={classes.toolbar}>
        <IconButton onClick={handleDrawerClose}>
          {theme.direction === "rtl" ? (
            <ChevronRightIcon />
          ) : (
            <ChevronLeftIcon />
          )}
        </IconButton>
      </div>
      <Divider />
      <List>
        {ADMIN_PANEL_ROUTES.map((route) => (
          // eslint-disable-next-line no-restricted-globals
          <Link
            to={route.to}
            key={route.id}
            className={clsx(
              classes.link,
              location.pathname === route.to && "active"
            )}
          >
            <ListItem button>
              <ListItemIcon className={classes.routeIcon}>
                {route.icon}
              </ListItemIcon>
              <ListItemText primary={route.name} />
            </ListItem>
          </Link>
        ))}
      </List>
      <Divider />
    </Drawer>
    <main className={classes.content}>
      <div className={classes.toolbar} />
      <Switch>
        <Route exact path="/adminpanel/" component={OverviewPage} />
        <Route exact path="/adminpanel/products" component={ProductsPage} />
        <Route exact path="/adminpanel/labels" component={LabelsPage} />
        <Route exact path="/adminpanel/orders" component={OrdersPage} />
        <Route exact path="/adminpanel/feedbacks" component={FeedbackPage} />
        <Route exact path="/adminpanel/users" component={UsersPage} />
        <Route exact path="/adminpanel/sizes" component={SizesPage} />
        <Route exact path="/adminpanel/roles" component={RolesPage} />
        <Route path="/adminpanel/orders/:id" component={OrderDetailsPage} />
        <Redirect to="/adminpanel" />
      </Switch>
    </main>
  </div>
 );
};
AdminPanelPage.propTypes = {
 userData: PropTypes.shape({
```

```
    role: PropTypes.shape({
      id: PropTypes.number,
    }),
  }),
  history: PropTypes.shape({
    push: PropTypes.func,

  }).isRequired,
  getUserByTokenRequest: PropTypes.func.isRequired,
  loadingRoot: PropTypes.bool,
  token: PropTypes.string,
};
AdminPanelPage.defaultProps = {
  userData: null,
  loadingRoot: false,
  token: null,
};
const mapStateToProps = (state) => ({
  userData: state.rootComponentReducer.loginUser.user,
  loadingRoot: state.rootComponentReducer.loading,
  token: state.rootComponentReducer.token,
});
const mapDispatchToProps = {
  getUserByTokenRequest: getUserByTokenRequestAction,
};

export default connect(mapStateToProps, mapDispatchToProps)(AdminPanelPage);
```

Constants.js // файл констант для адмінпанелі

```
/* eslint-disable */
import React from "react";
import { GiClothes } from "react-icons/gi";
import { MdLabel } from "react-icons/md";
import { ImStatsBars } from "react-icons/im";
import { RiShieldUserLine, RiFeedbackFill } from "react-icons/ri";
import { SiMatrix } from "react-icons/si";
import { FaClipboardList, FaUserFriends } from "react-icons/fa";
const ADMIN_PANEL = {
  title: "Admin panel",
  link: "Home page",
};
export const ADMIN_PANEL_ROUTES = [
  {
    id: "statistic",
    name: "Statistic",
    icon: <ImStatsBars />,
    to: "/adminpanel",
  },

  {
    id: "products",
    name: "Products",
```

```
    icon: <GiClothes />,
    to: "/adminpanel/products",
  },
  {
    id: "orders",
    name: "Orders",
    icon: <FaClipboardList />,
    to: "/adminpanel/orders",
  },

  {
    id: "users",
    name: "Users",
    icon: <FaUserFriends />,
    to: "/adminpanel/users",
  },

  { id: "labels", name: "Labels", icon: <MdLabel />, to: "/adminpanel/labels" },
  { id: "sizes", name: "Sizes", icon: <SiMatrix />, to: "/adminpanel/sizes" },
  {
    id: "roles",
    name: "Roles",
    icon: <RiShieldUserLine />,
    to: "/adminpanel/roles",
  },

  {
    id: "feedbacks",
    name: "Feedbacks",
    icon: <RiFeedbackFill />,
    to: "/adminpanel/feedbacks",
  },
];
export const GET_PRODUCTS_REQUEST =
"GET_PRODUCTS_ADMIN_PAGE_REQUEST";
export const GET_PRODUCTS_ERROR = "GET_PRODUCTS_ADMIN_PAGE_ERROR";
export const GET_PRODUCTS_SUCCESS =
"GET_PRODUCTS_ADMIN_PAGE_SUCCESS";
export const GET_LABELS_REQUEST = "GET_LABELS_REQUEST";
export const GET_LABELS_ERROR = "GET_LABELS_ERROR";
export const GET_LABELS_SUCCESS = "GET_LABELS_SUCCESS";
export const GET_PRODUCT_MODAL_DATA_REQUEST =
"GET_PRODUCT_MODAL_DATA_REQUEST";
export const GET_PRODUCT_MODAL_DATA_ERROR =
"GET_PRODUCT_MODAL_DATA_ERROR";
export const GET_PRODUCT_MODAL_DATA_SUCCESS =
"GET_PRODUCT_MODAL_DATA_SUCCESS";
export const GET_ORDERS_REQUEST = "GET_ORDERS_REQUEST";
export const GET_ORDERS_ERROR = "GET_ORDERS_ERROR";
export const GET_ORDERS_SUCCESS = "GET_ORDERS_SUCCESS";
export const GET_ORDER_BY_ID_REQUEST = "GET_ORDER_BY_ID_REQUEST";
export const GET_ORDER_BY_ID_ERROR = "GET_ORDER_BY_ID_ERROR";
```

```
export const GET_ORDER_BY_ID_SUCCESS = "GET_ORDER_BY_ID_SUCCESS";
export const GET_FEEDBACK_REQUEST = "GET_FEEDBACK_REQUEST";
export const GET_FEEDBACK_SUCCESS = "GET_FEEDBACK_SUCCESS";
export const GET_FEEDBACK_ERROR = "GET_FEEDBACK_ERROR";
export const GET_USERS_REQUEST = 'GET_USERS_REQUEST';
export const GET_USERS_SUCCESS = 'GET_USERS_SUCCESS';
export const GET_USERS_ERROR = 'GET_USERS_ERROR';
export const GET_OVERVIEW_REQUEST = 'GET_OVERVIEW_REQUEST';
export const GET_OVERVIEW_SUCCESS = 'GET_OVERVIEW_SUCCESS';
export const GET_OVERVIEW_ERROR = 'GET_OVERVIEW_ERROR';
export const GET_SIZES_REQUEST = 'GET_SIZES_REQUEST';
export const GET_SIZES_SUCCESS = 'GET_SIZES_SUCCESS';
export const GET_SIZES_ERROR = 'GET_SIZES_ERROR';
export const GET_ROLES_REQUEST = 'GET_ROLES_REQUEST';
export const GET_ROLES_SUCCESS = 'GET_ROLES_SUCCESS';
export const GET_ROLES_ERROR = 'GET_ROLES_ERROR';
export const POST_CREATE_NEW_PRODUCT_REQUEST =
  "POST_CREATE_NEW_PRODUCT_REQUEST";
export const POST_CREATE_NEW_PRODUCT_ERROR =
"POST_CREATE_NEW_PRODUCT_ERROR";
export const POST_CREATE_NEW_PRODUCT_SUCCESS =
  "POST_CREATE_NEW_PRODUCT_SUCCESS";
export const POST_CREATE_NEW_LABEL_REQUEST =
"POST_CREATE_NEW_LABEL_REQUEST";
export const POST_CREATE_NEW_LABEL_ERROR =
"POST_CREATE_NEW_LABEL_ERROR";
export const POST_CREATE_NEW_LABEL_SUCCESS =
"POST_CREATE_NEW_LABEL_SUCCESS";
export const PUT_UPDATE_PRODUCT_REQUEST =
"PUT_UPDATE_PRODUCT_REQUEST";
export const PUT_UPDATE_PRODUCT_ERROR = "PUT_UPDATE_PRODUCT_ERROR";
export const PUT_UPDATE_PRODUCT_SUCCESS =
"PUT_UPDATE_PRODUCT_SUCCESS";
export const PUT_UPDATE_LABEL_REQUEST = "PUT_UPDATE_LABEL_REQUEST";
export const PUT_UPDATE_LABEL_ERROR = "PUT_UPDATE_LABEL_ERROR";
export const PUT_UPDATE_LABEL_SUCCESS = "PUT_UPDATE_LABEL_SUCCESS";
export const SEND_FEEDBACK_ANSWER_REQUEST =
"SEND_FEEDBACK_ANSWER_REQUEST";
export const SEND_FEEDBACK_ANSWER_SUCCESS =
"SEND_FEEDBACK_ANSWER_SUCCESS";
export const SEND_FEEDBACK_ANSWER_ERROR =
"SEND_FEEDBACK_ANSWER_ERROR";
export const DELETE_PRODUCT_BY_ID_REQUEST =
"DELETE_PRODUCT_BY_ID_REQUEST";
export const DELETE_PRODUCT_BY_ID_ERROR =
"DELETE_PRODUCT_BY_ID_ERROR";
export const DELETE_PRODUCT_BY_ID_SUCCESS =
"DELETE_PRODUCT_BY_ID_SUCCESS";
export const DELETE_LABEL_BY_ID_REQUEST =
"DELETE_LABEL_BY_ID_REQUEST";
export const DELETE_LABEL_BY_ID_ERROR = "DELETE_LABEL_BY_ID_ERROR";
export const DELETE_LABEL_BY_ID_SUCCESS = "DELETE_LABEL_BY_ID_SUCCESS";
```

```
export const DELETE_ORDER_BY_ID_REQUEST =
"DELETE_ORDER_BY_ID_REQUEST";
export const DELETE_ORDER_BY_ID_ERROR = "DELETE_ORDER_BY_ID_ERROR";
export const DELETE_ORDER_BY_ID_SUCCESS =
"DELETE_ORDER_BY_ID_SUCCESS";
export const DELETE_PRODUCT_FROM_ORDER_REQUEST =
  "DELETE_PRODUCT_FROM_ORDER_REQUEST";
export const DELETE_PRODUCT_FROM_ORDER_ERROR =
  "DELETE_PRODUCT_FROM_ORDER_ERROR";
export const DELETE_PRODUCT_FROM_ORDER_SUCCESS =
  "DELETE_PRODUCT_FROM_ORDER_SUCCESS";
export const SWITCH_LABEL_MODAL_WINDOW =
"SWITCH_LABEL_MODAL_WINDOW";
export const LABELS_PAGE_CLEAR_SUCCESS_STATUS =
  "LABELS_PAGE_CLEAR_SUCCESS_STATUS";
export default ADMIN_PANEL;
```

Reducer.JS// файл редьюсер для адміністративної панелі

```
/* eslint-disable */
import { createReducer } from "@reduxjs/toolkit";
import * as actions from "./actions";
import { mapCategories, mapLabels, mapSizes, mapTypes } from "./helpes";
const initialState = {
  productsPage: {
    loading: false,
    success: false,
    error: null,
    data: null,
    productModal: {
      loading: false,
      success: false,
      error: null,
      categories: null,
      labels: null,
      sizes: null,
      types: null,
    },

  },
  labelsPage: {
    loading: false,
    success: false,
    error: null,
    data: null,
    labelModal: {
      loading: false,
      isOpen: false,
      error: null,
      success: false,
    },
```

```
  },
  ordersPage: {
    loading: false,
    success: false,
    error: null,
    data: null,
  },
  orderDetailsPage: {
    loading: false,
    success: false,
    error: null,
    data: null,
  },

  feedbacksPage: {
    loading: false,
    success: false,
    error: null,
    data: null,
  },

  usersPage: {
    data: null,
    error: null,
    loading: false,
  },

  overviewPage: {
    data: null,
    error: null,
    loading: false,
  },

  sizesPage: {
    data: null,
    error: null,
    loading: false,
  },

  rolesPage: {
    data: null,
    error: null,
    loading: false,
  }

};
const reducer = createReducer(initialState, {
  [actions.getProductsRequest]: (state) => {
    state.productsPage.loading = true;
    state.productsPage.success = false;
    state.productsPage.error = null;
```

```
  state.productsPage.data = null;
},
[actions.getProductsSuccess]: (state, action) => {
  state.productsPage.loading = false;
  state.productsPage.data = action.payload;
},

[actions.getProductsError]: (state, action) => {
  state.productsPage.loading = false;
  state.productsPage.error = action.payload;
},

[actions.getLabelsRequest]: (state) => {
  state.labelsPage.loading = true;
  state.labelsPage.error = null;
  state.labelsPage.data = null;
},

[actions.getLabelsSuccess]: (state, action) => {
  state.labelsPage.loading = false;
  state.labelsPage.data = action.payload;
},

[actions.getLabelsError]: (state, action) => {
  state.labelsPage.loading = false;
  state.labelsPage.error = action.payload;
},

[actions.getProductModalDataRequest]: (state) => {
  state.productsPage.productModal.error = null;
  state.productsPage.productModal.loading = true;
  state.productsPage.productModal.success = false;
  state.productsPage.productModal.categories = null;
  state.productsPage.productModal.sizes = null;
  state.productsPage.productModal.labels = null;
},

[actions.getProductModalDataSuccess]: (state, action) => {
  state.productsPage.productModal.loading = false;
  state.productsPage.productModal.categories = mapCategories(
    action.payload.categories
  );

  state.productsPage.productModal.sizes = mapSizes(action.payload.sizes);
  state.productsPage.productModal.labels = mapLabels(action.payload.labels);
  state.productsPage.productModal.types = mapTypes(action.payload.types);
},

[actions.getProductModalDataError]: (state, action) => {
  state.productsPage.productModal.loading = false;
  state.productsPage.productModal.error = action.payload;
},
```

```
[actions.getOrdersRequest]: (state) => {
  state.ordersPage.data = null;
  state.ordersPage.loading = true;
  state.ordersPage.error = null;
},
[actions.getOrdersError]: (state, action) => {
  state.ordersPage.loading = false;
  state.ordersPage.error = action.payload;
},
[actions.getOrdersSuccess]: (state, action) => {
  state.ordersPage.loading = false;
  state.ordersPage.data = action.payload;
},
[actions.getOrderByIdRequest]: (state) => {
  state.orderDetailsPage.loading = true;
  state.orderDetailsPage.success = false;
  state.orderDetailsPage.error = null;
  state.orderDetailsPage.data = null;
},
[actions.getOrderByIdError]: (state, action) => {
  state.orderDetailsPage.loading = false;
  state.orderDetailsPage.error = action.payload;
},
[actions.getOrderByIdSuccess]: (state, action) => {
  state.orderDetailsPage.loading = false;
  state.orderDetailsPage.data = action.payload;
},
[actions.getFeedbacksRequest]: (state) => {
  state.feedbacksPage.loading = true;
  state.feedbacksPage.data = null;
  state.feedbacksPage.error = null;
  state.feedbacksPage.success = false;
},

[actions.getFeedbacksSuccess]: (state, action) => {
  state.feedbacksPage.loading = false;
  state.feedbacksPage.data = action.payload;
},

[actions.getFeedbacksError]: (state, action) => {
  state.feedbacksPage.loading = false;
  state.feedbacksPage.error = action.payload;
},

[actions.getUsersRequest]: (state) => {
  state.usersPage.loading = true;
  state.usersPage.data = null;
  state.usersPage.error = null;
},

[actions.getUsersSuccess]: (state, action) => {
```

```
  state.usersPage.loading = false;
  state.usersPage.data = action.payload;
 },

 [actions.getUsersError]: (state, action) => {
  state.usersPage.loading = false;
  state.usersPage.error = action.payload;
 },
 [actions.getOverviewRequest]: (state) => {
  state.overviewPage.loading = true;
  state.overviewPage.data = null;
  state.overviewPage.error = null;
 },
 [actions.getOverviewSuccess]: (state, action) => {
  state.overviewPage.loading = false;
  state.overviewPage.data = action.payload;
 },
 [actions.getOverviewError]: (state, action) => {
  state.overviewPage.loading = false;
  state.overviewPage.error = action.payload;
 },
 [actions.getSizesRequest]: (state) => {
  state.sizesPage.loading = true;
  state.sizesPage.data = null;
  state.sizesPage.error = null;
 },
 [actions.getSizesSuccess]: (state, action) => {
  state.sizesPage.loading = false;
  state.sizesPage.data = action.payload;
 },
 [actions.getSizesError]: (state, action) => {
  state.sizesPage.loading = false;
  state.sizesPage.error = action.payload;
 },
 [actions.getRolesRequest]: (state) => {
  state.rolesPage.loading = true;
  state.rolesPage.data = null;
  state.rolesPage.error = null;
 },
 [actions.getRolesSuccess]: (state, action) => {
  state.rolesPage.loading = false;
  state.rolesPage.data = action.payload;
 },
 [actions.getRolesError]: (state, action) => {
  state.rolesPage.loading = false;
  state.rolesPage.error = action.payload;
 },
 [actions.postCreateNewProductRequest]: (state) => {
  state.productsPage.productModal.loading = true;
  state.productsPage.productModal.error = null;
  state.productsPage.productModal.success = false;
 },
```

```
[actions.postCreateNewProductError]: (state, action) => {
  state.productsPage.productModal.loading = false;
  state.productsPage.productModal.error = action.payload;
},
[actions.postCreateNewProductSuccess]: (state, action) => {
  state.productsPage.productModal.loading = false;
  state.productsPage.productModal.success = true;
  state.productsPage.data = [...state.productsPage.data, action.payload];
},

[actions.postCreateNewLabelRequest]: (state) => {
  state.labelsPage.labelModal.loading = true;
  state.labelsPage.labelModal.success = false;
  state.labelsPage.labelModal.error = null;
},

[actions.postCreateNewLabelError]: (state, action) => {
  state.labelsPage.labelModal.loading = false;
  state.labelsPage.labelModal.error = action.payload;
},

[actions.postCreateNewLabelSuccess]: (state, action) => {
  state.labelsPage.labelModal.loading = false;
  state.labelsPage.labelModal.success = true;
  state.labelsPage.data = [...state.labelsPage.data, action.payload];
},

[actions.sendFeedbackAnswerRequest]: (state) => {
  state.feedbacksPage.loading = true;
},

[actions.sendFeedbackAnswerSuccess]: (state) => {
  state.feedbacksPage.loading = false;
},

[actions.sendFeedbackAnswerError]: (state, action) => {
  state.feedbacksPage.loading = false;
  state.feedbacksPage.error = action.payload;
},

[actions.putUpdateProductByIdRequest]: (state) => {
  state.productsPage.productModal.loading = true;
  state.productsPage.productModal.error = null;
  state.productsPage.productModal.success = false;
},

[actions.putUpdateProductByIdError]: (state, action) => {
  state.productsPage.productModal.loading = false;
  state.productsPage.productModal.error = action.payload;
},

[actions.putUpdateProductByIdSuccess]: (state) => {
```

```
  state.productsPage.productModal.loading = false;
  state.productsPage.productModal.success = true;
 },
 [actions.putUpdateLabelByIdRequest]: (state) => {
  state.labelsPage.labelModal.loading = true;
  state.labelsPage.labelModal.success = false;
  state.labelsPage.labelModal.error = null;
 },
 [actions.putUpdateLabelByIdError]: (state, action) => {
  state.labelsPage.labelModal.loading = false;
  state.labelsPage.labelModal.error = action.payload;
 },
 [actions.putUpdateLabelByIdSuccess]: (state) => {
  state.labelsPage.labelModal.loading = false;
  state.labelsPage.labelModal.success = true;
 },

 [actions.deleteProductByIdRequest]: (state) => {
  state.productsPage.loading = true;
  state.productsPage.success = false;
  state.productsPage.error = null;
 },

 [actions.deleteProductByIdError]: (state, action) => {
  state.productsPage.loading = false;
  state.productsPage.error = action.payload;
 },

 [actions.deleteProductByIdSuccess]: (state, action) => {
  state.productsPage.loading = false;
  state.productsPage.success = true;
  state.productsPage.data = state.productsPage.data.filter(
    (product) => product.id !== action.payload
  );
 },

 [actions.deleteLabelByIdRequest]: (state) => {
  state.labelsPage.loading = true;
  state.labelsPage.error = null;
  state.labelsPage.success = false;
 },

 [actions.deleteLabelByIdError]: (state, action) => {
  state.labelsPage.loading = false;
  state.labelsPage.error = action.payload;
 },
 [actions.deleteLabelByIdSuccess]: (state, action) => {
  state.labelsPage.loading = false;
  state.labelsPage.success = true;
  state.labelsPage.data = state.labelsPage.data.filter(
    (label) => label.id !== action.payload
  );
```

```
  },
  [actions.switchLabelModalWindow]: (state, action) => {
    state.labelsPage.labelModal.isOpen = action.payload;
    state.labelsPage.labelModal.error = null;
  },
  [actions.labelsPageClearSuccessStatus]: (state) => {
    state.labelsPage.success = false;
    state.labelsPage.labelModal.success = false;
  },
  [actions.deleteOrderByIdRequest]: (state) => {
    state.orderDetailsPage.loading = true;
    state.orderDetailsPage.success = false;
    state.orderDetailsPage.error = null;
  },
  [actions.deleteOrderByIdError]: (state, action) => {
    state.orderDetailsPage.loading = false;
    state.orderDetailsPage.error = action.payload;
  },
  [actions.deleteOrderByIdSuccess]: (state) => {
    state.orderDetailsPage.loading = false;
    state.orderDetailsPage.success = true;
  },
});
export default reducer;
```

MainPage.js // основний файл головної сторінки

```
import React, { useEffect } from "react";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import { getInitMainPageDataRequest as getInitMainPageDataRequestAction } from "./actions";
import Slider from "./components/Slider";
import ClothesTypes from "./components/ClothesTypes";
import MostRated from "./components/MostRated";
import Spinner from "../../components/Spinner";
import useStyles from "./MainPage.styles";
const MainPage = ({ getInitMainPageDataRequest, loading, data }) => {
  const classes = useStyles();
  useEffect(() => {
    getInitMainPageDataRequest();
  }, [getInitMainPageDataRequest]);
  return (
    <div className={classes.mainPageWrapper}>
      {loading ? (
        <Spinner />
      ) : (
        <div className="main-page__content">
          <Slider />
          <ClothesTypes />
          {data?.mostRated && <MostRated data={data.mostRated} />}
        </div>
      )}
    </div>
```

```
  );
};
MainPage.propTypes = {
  loading: PropTypes.bool,
  data: PropTypes.shape({
    mostRated: PropTypes.arrayOf(PropTypes.shape({})),
  }),
  getInitMainPageDataRequest: PropTypes.func.isRequired,
};
MainPage.defaultProps = {
  loading: false,
  data: null,
};
const mapDispatchToProps = {
  getInitMainPageDataRequest: getInitMainPageDataRequestAction,
};
const mapStateToProps = (state) => ({
  data: state.mainPageReducer.mainPage.data,
  loading: state.mainPageReducer.mainPage.loading,
});
export default connect(mapStateToProps, mapDispatchToProps)(MainPage);
```

ProfilePage.js //основний файл сторінки профілю

```
import React, { useEffect } from 'react';
import PropTypes from 'prop-types';
import { connect } from 'react-redux';
import { Avatar, Button } from '@material-ui/core';
import { Link, Redirect, Route, Switch } from 'react-router-dom';
import { getUserInfoRequest as getUserInfoRequestAction } from './actions';
import useStyles from './ProfilePage.styles';
import PROFILE_PAGE from './constants';
import PageSubTitle from '../../global/styles/global.style';
import Loader from '../../components/Loader';
import ProfileEditTab from './components/ProfileEditTab/ProfileEditTab';
import OrdersTab from './components/OrdersTab/OrdersTab';
const ProfilePage = ({ getUserInfoRequest, token, error, loading, data }) => {
  const classes = useStyles();
  useEffect(() => {
    getUserInfoRequest(token);
  }, [token, getUserInfoRequest]);
  if (loading) {
    return <Loader />;
  }
  return (
    <div className={classes.profilePageWrapper}>
      <PageSubTitle>{PROFILE_PAGE.title}</PageSubTitle>
      <div className="profile__controls_wrapper">
        <div className="profile__total">
          {error || !data ? (<div className="loading__error">{PROFILE_PAGE.error}</div>) : (
            <>
              <div className="main__info">
                <Avatar className="avatar" src={data?.image}>{data.name ? data.name[0] :
```

```
null}</Avatar>
            <div className="user_name">{data.name} {data.second_name}</div>
          </div>
          <div className="other__info">
            <div className="info__row">
              <div className="row__title">{PROFILE_PAGE.values.role}</div>
              <div className="row__value">{data.role.name.toLowerCase() || '-'}</div>
            </div>
            <div className="info__row">
              <div className="row__title">{PROFILE_PAGE.values.email}</div>
              <div className="row__value">{data.email || '-'}</div>
            </div>
            <div className="info__row">
              <div className="row__title">{PROFILE_PAGE.values.emailStatus}</div>
              <div className="row__value">{data.is_email_authorized ? <span
className="status_good">{PROFILE_PAGE.status.confirmed}</span> : <Button
variant="outlined" color="secondary">{PROFILE_PAGE.status.confirm}</Button>}</div>
            </div>
            <div className="info__row">
              <div className="row__title">{PROFILE_PAGE.values.phone}</div>
              <div className="row__value">{data.mobile_phone || '-'}</div>
            </div>
          </div>
        </>
      )}
    </div>
    <div className="profile__tabs">
      <div className="tabs__navigation">
        {PROFILE_PAGE.navLinks.map((l) => (
          // eslint-disable-next-line no-restricted-globals
          <div className={`tab__nav ${location.pathname === l.url ? 'active' : ''}`}
key={l.id}>
            <Link to={l.url}>{l.title}</Link>
          </div>
        ))}
      </div>
      <Switch>
        <Route exact path="/profile/main/" component={ProfileEditTab} />
        <Route exact path="/profile/orders/" component={OrdersTab} />
        <Redirect to="/profile/main/" />
      </Switch>
    </div>
  </div>
</div>
); };
ProfilePage.propTypes = {
  getUserInfoRequest: PropTypes.func.isRequired,
  token: PropTypes.string,
  loading: PropTypes.bool,
  error: PropTypes.shape({}),
  data: PropTypes.shape({
    name: PropTypes.string,
```

```
    second_name: PropTypes.string,
    image: PropTypes.string,
    email: PropTypes.string,
    mobile_phone: PropTypes.string,
    is_email_authorized: PropTypes.bool,
    role: PropTypes.shape({
      name: PropTypes.string,
    }),  }),
};
ProfilePage.defaultProps = {
  token: null,
  loading: false,
  error: null,
  data: null,
};
const mapDispatchToProps = {
  getUserInfoRequest: getUserInfoRequestAction,
};
const mapStateToProps = (state) => ({
  token: state.rootComponentReducer.token,
  data: state.profilePageReducer.data,
  loading: state.profilePageReducer.loading,
  error: state.profilePageReducer.error,
})
export default connect(mapStateToProps, mapDispatchToProps)(ProfilePage);
```

У цьому додатку були вказали файли з кожного розділу файлової системи. Повний перелік файлів зі змістом знаходяться на електронному носієві.

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка та адміністрування front-end частини Інтернет магазину з продажу одягу»
студентки групи 122-17-2 Купрієнко Вікторії Євгенівни

Керівник економічного розділу
доцент каф. ПЕП та ПУ, к.е.н **Л. В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом Купрієнко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом Купрієнко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
diplom.zip	Архів. Містить коди програми.
Презентація	
Презентація Купрієнко.ppt	Презентація кваліфікаційної роботи.