МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра інформаційних систем
(повна назва кафедри)

Магістерська робота

на тему:

Розробка онлайн магазину з використанням MERN

Виконав: студент групи ПМІМ-22с
спеціальності
122-комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Легедза О. Р.
(прізвище та ініціали)

Керівник (підпис) Дреботій Р. Г.
(прізвище та ініціали)

Рецензент (підпис) Борачок І. В.
(прізвище та ініціали)

ДЕКАН
факультету прикладної
математики та інформатики
ЛНУ імені Івана Франка

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет *прикладної математики та інформатики*
Кафедра *інформаційних систем*
Спеціальність 122 – комп'ютерні науки
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри *проф. Шинкаренко Г.А.*

"*5*" *вересня* **20*22* року**

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Легедзі Олександру Романовичу
(прізвище, ім'я, по батькові)

1 .Тема роботи *Розробка онлайн магазину з використанням MERN*

керівник роботи *Дреботій Роман Григорович, доцент кафедри інформаційних систем*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від "13" *вересня* 20*22* року № 15

2. Строк подання студентом роботи *12.12.2022 р.*

3. Вихідні дані до роботи *Література та інтернет-ресурси за тематикою роботи*

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)
1. Опис та постановка задачі
1. Дослідження наявних застосунків
2. Ролі користувачів
3. Імплементація
4. Огляд вебсайту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Схеми, презентація дипломної роботи

6. Консультанти розділів роботи

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

7. Дата видачі завдання *05.09.2022 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Опис та постановка задачі	*Вересень*	*Виконано*
2.	Дослідження наявних застосунків	*Жовтень*	*Виконано*
3.	Ролі користувачів	*Листопад*	*Виконано*
4.	Імплементація	*Листопад*	*Виконано*
5.	Огляд вебсайту	*Грудень*	*Виконано*
6.	Оформлення роботи і подання до захисту	*Грудень*	*Виконано*

Студент _____ *Легедза О. Р.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Дреботій Р. Г.*
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

На сьогоднішній час важко уявити річ яку не можна було би замовити через інтернет. Люди замовляють продукти, одяг, техніку, все чого забажає душа. Особливо така тенденція зросла в зв’язку з пандемією.

Проте із цього можна зробити висновок, що навіть маючи надзвичайно професійних та позитивних працівників які б відгукались на кожне побажання покупця, найякісніші речі та оздоблення магазину, торгового центру чи іншого приміщення, всі ці речі не матимуть такий сильний вплив якщо потенційному покупцеві через нестачу часу або ж банальну лінь не захочеться виходити із власної домівки.

Отже привабливість та функціональність сайту магазину мають таке ж значення як і привітливий персонал та приємний інтер`єр.

Актуальність моєї магістерської роботи в тому що будь-який великий чи малий бізнес у якого є онлайн магазин має можливість розповісти про себе, показати асортимент доступних товарів користувачеві всього лише при введені домену в пошукову стрічку без потреби витрачати час щоб побачити все на власні очі в живу. Саме тому онлайн магазини займають сьогодні дуже важливу роль в процесі купівля-продаж.

РОЗДІЛ 1. ОПИС ТА ПОСТАНОВКА ЗАДАЧІ

Основна мета магістерської роботи полягає в тому, щоб розширити функціонал раніше розробленого сайту для створення блогів в онлайн магазин з різноманітними можливостями користувачів котрі присутні у популярних аналогів, розробити функціонал котрий не був реалізований у аналогів, але котрий би був корисним і дозволяв у подібного роду застосунках використовувати те що присутнє в магазинах в реальному житті.

1.1 Основні вимоги до сайту

Система повинна мати наступний функціонал:

- Створення/редагування/видалення товарів;
- Пошук товарів за різними властивостями (назва, теги);
- Продуктова корзина;
- Створення/редагування замовлень;
- Створення/редагування/видалення рецензій;
- Система знижок (знижка на товар, товари оптом, купони);
- Створення/редагування/видалення купонів;
- Уподобання товарів;
- Розділення ролей користувачів;

1.2 Архітектура системи

Система повинна бути побудована у форматі веб-застосунку який складається з трьох основних частин:

- Клієнтська частина зі зручним інтерфейсом що працює в браузері з адаптивною версткою котра буде приємною як на десктопних платформах так і на мобільних;
- Серверна частина для отримання даних з клієнтської частини, занесення змін у базу даних та повернення якихось даних зі сторони сервера на клієнт;
- База даних котра містить інформацію про дані різних колекцій котрі будуть повертатись на сторону клієнта при запитах;

Сайт повинен бути розміщений в інтернеті у відкритому доступі. Для цього буде використано сервіс для хостингу вебсайтів.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ НАЯВНИХ ЗАСТОСУНКІВ

Під час дослідження наявних застосунків особливу увагу було звернено на популярність застосунків та на наявний функціонал котрий був би доречним для нашого онлайн-магазину. Були обрані такі аналоги як Розетка та Udemy.

Розетка (англ. Rozetka) – український інтернет-магазин та маркетплейс, що з`явився у 2005 році. Найбільший ритейлер в Україні.

У розетки різноманітний функціонал продуктової корзини. Можна коригувати кількість товарів за допомогою кнопок або поля для вводу, доступна інформація про наявність знижки на товар.

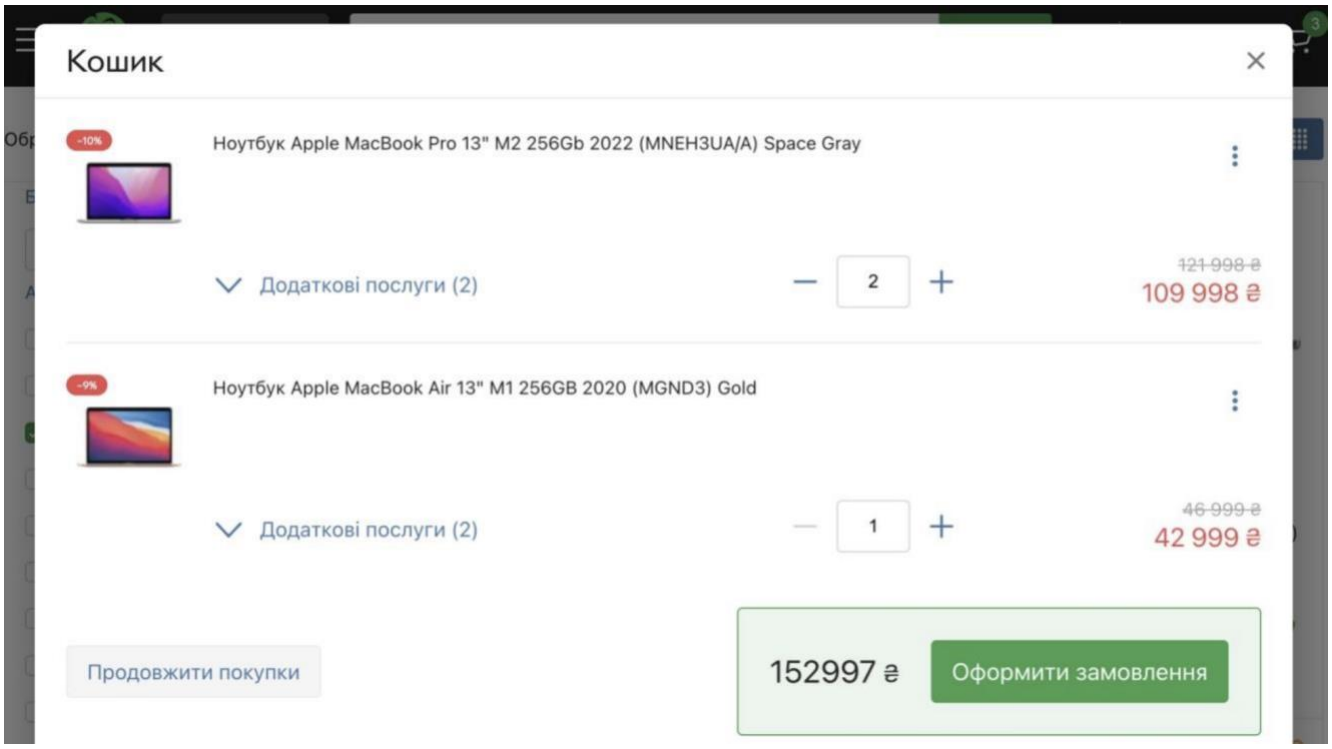

Рисунок 2.1 – Продуктова корзина Rozetka

Udemy – це сайт для онлайн-навчання та викладання з сотнею тисяч курсів по програмуванню, маркетингу, науці і величезною аудиторією котра начислює десятки мільйонів людей.

Користувачів приваблюють товари котрі мають серед детальної інформації такі речі як рейтинг та відгуки таких самих як вони покупців. Чим він вище тим ймовірніше він зацікавить потенційного покупця, така інформація буде доволі корисною для сайту в довгостроковій перспективі, адже користувачі будуть мати змогу бачити що саме думають про товар такі ж покупці як і вони.

Рисунок 2.2 – Рейтингова система Udemy

Також на сайті присутній такий фунціонал як продуктові купони. При введені правильного купону ціна продуктової корзини буде помножена на відсоток знижки який надає купон.

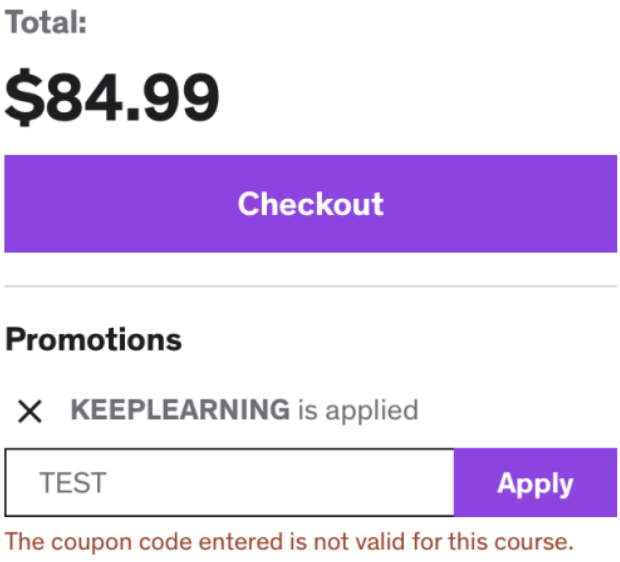

Рисунок 2.3 – Поле вводу для продуктового купону

РОЗДІЛ 3. ВИКОРИСТАНІ ТЕХНОЛОГІЇ

Існує дуже багато способів як досягти бажаного результату в розробці потрібного онлайн магазину: десятки мов програмування, велика кількість фреймворків та бібліотек. При виборі технологій для створення програми було зроблено акцент на тому що кожна з них може надати, як може спростити роботу. Про кожну технологію яка є фундаментальною для створеного онлайн-магазину буде розказано детальніше.

3.1 React.js

React – одна з найпопулярніших бібліотек для створення інтерфейсів користувача. Вона використовується не тільки для створення веб-інтерфейсів, але також для розробки мобільних (React Native) та настільних додатків (Electron).

Популярність React пов`язана з тим, що бібліотека дозволяє значно спростити розробку складних і динамічних інтерфейсів. При цьому код виходить зрозумілим і структурованим, його дуже просто підтримувати і оновлювати. Якщо такий інтерфейс програмувати на чистому JavaScript, то без достатньої практики код такого проекту буде дуже швидко ускладнюватись. В результаті довести до кінця такий проект буде проблематично.

React запроваджує так званий компонентний підхід, коли частини коду виносяться в окрему компоненту і можуть викликатись в будь-якому місці програми безліч разів без потреби повторювати один і той же код. Наглядний приклад компонентного підходу можна побачити на рисунку 3.1.

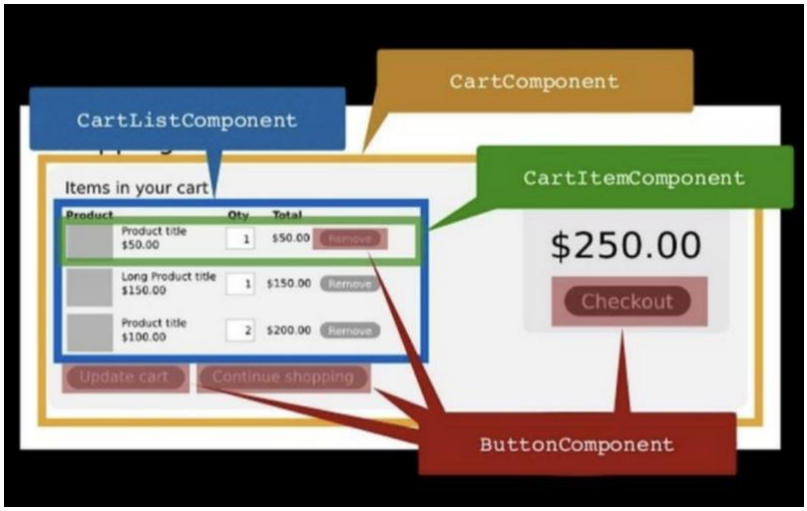

Рисунок 3.1 – Приклад компонентного підходу

3.2 Redux

В даний момент часу розробка більшої частини веб-застосунків, заснованих на React, ведеться з використанням бібліотек для контролю глобального стану (global state). React розробив не так давно свою власну версію global state менеджменту під назвою React Context. На практиці це доволі непогане рішення для середніх проектів, але не для громіздких де кількість даних котрі потрібно контролювати дуже велика, контролювати таке з допомогою React Context доволі непросто.

Одним із найкращих способів є використання Redux. Redux - бібліотека розібравшись в якій проблем зі зберіганням та маніпулюванням global state у розробника більше ніколи не буде. Ми виносимо весь state який потрібен у різних компонентах програми в global state та за допомогою синтаксису Redux можемо достукуватись та змінювати його у будь-якій компоненті. Візуально це все виглядає як на рисунку 3.2.

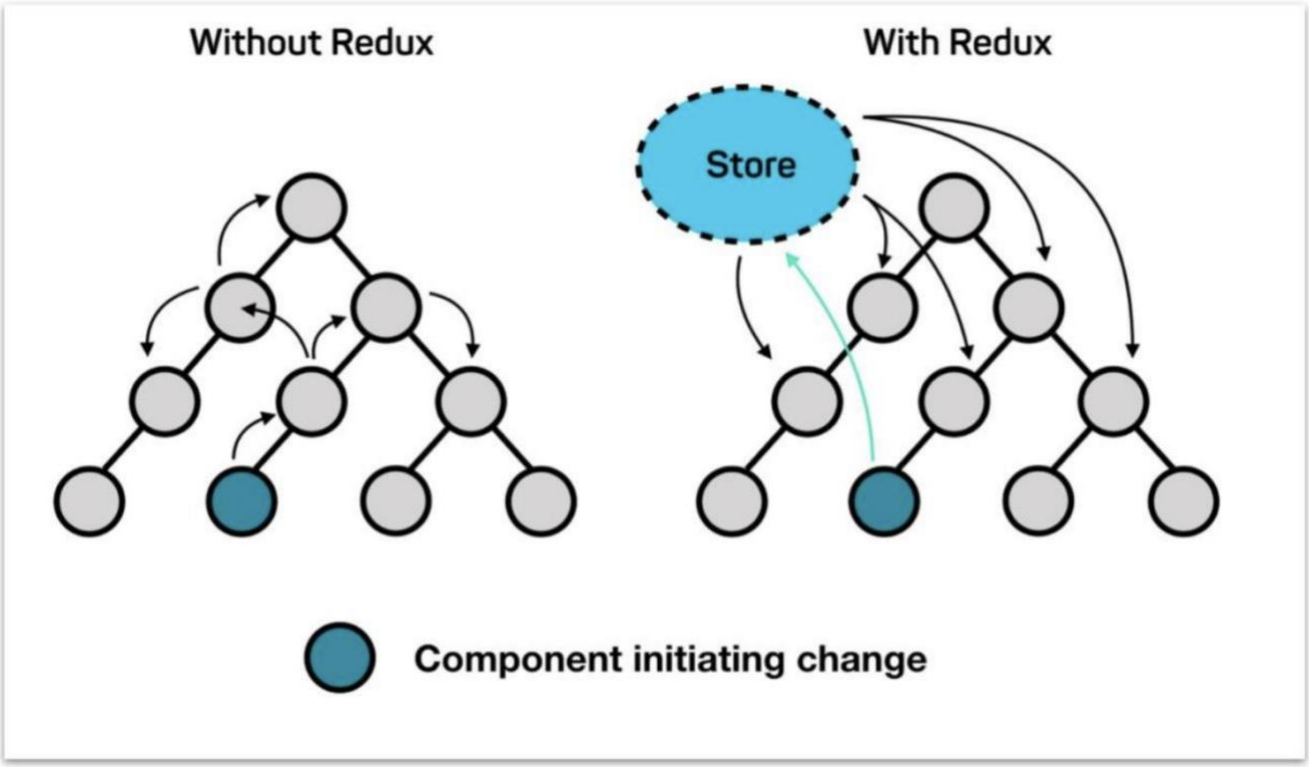

Рисунок 3.2 – Redux Global State

У Redux операції пов`язані зі зміною та використанням state відбуваються в декілька кроків.

• Виносимо всі дані котрі будуть нам потрібні неодноразово в різних компонентах програми в global state;

• Вся логіка роботи з global state описується в reducer (наприклад додати товар до карти чи змінити кількість товарів на вказану);

• Щоб застосувати reducer ми повинні з клієнтської сторони викликати action через dispatch;

• Щоб вибрати дані з global state в компонентах застосовуємо selector; Описані кроки візуально можна побачити на рисунку 3.3.

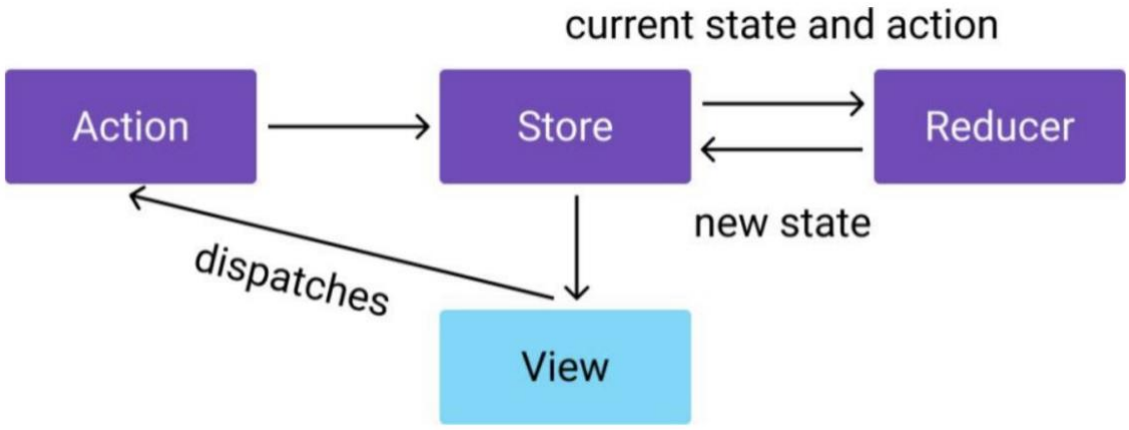

Рисунок 3.3 – Зміна state в Redux

3.3 Redux Toolkit

Проте Redux має хоч і невелику кількість недоліків, але 2 з них є доволі неприємними для розробників:

1. складність і “багатослівність” рекомендованих патернів для написання та організації коду, що тягне за собою велику кількість написання коду який потребує сама мова програмування, а не програміст чи програма (boilerplate code).

2. відсутність вбудованих засобів управління асинхронною поведінкою та побічними ефектами, що призводить до необхідності вибору відповідного інструменту з безліччю аддонів, написаних сторонніми розробниками.

Для усунення цих недоліків розробники Redux представили бібліотеку Redux Toolkit. Цей інструмент є набором практичних рішень та методів, призначених для спрощення розробки додатків з використанням Redux. Розробники цієї бібліотеки мали на меті спростити типові випадки використання Redux. Даний інструмент не є універсальним рішенням у кожному з можливих випадків використання Redux, але дозволяє спростити код, який потрібно написати розробнику.

3.4 Node.js

Node.js в основному використовується для створення швидких та масштабованих веб-застосунків. При цьому використовується керована подіями неблокуюча модель введення-виводу, що робить цю платформу простою та ефективною. Це відмінне рішення для розробки додатків реального часу, що обробляють великі обсяги даних та виконуються на розподілених пристроях.
Node.js люблять насамперед за універсальність та високу продуктивність. Крім того, серед найвідоміших переваг такі:

• Ефективність.

Ефективність одна з головних переваг Node.js. Можна виконувати кілька завдань одночасно завдяки неблокуючій моделі вводу-виводу. Менш значимі завдання виконуються паралельно, що дозволяє основному потоку працювати безперешкодно. Завдяки движку Google V8 код Javascript розбивається на більш низькорівневий машинний код без інтерпретатора.

• Масштабованість.

Node.js допомагає розробляти масштабовані програми, оскільки може справлятися з кількома запитами одночасно завдяки таким нативним інструментам як clusters, child process та worker threads. Навантаження на CPU не надто значне навіть при суттєвому зростанні запитів.

• Швидкість розробки.

Node.js може скоротити час розробки, оскільки дозволяє створювати компоненти, що перевикористовуються. До того ж є велике ком`юніті, де можна запозичити шаблони, створені іншими розробникам. Близько 836 000 компонентів, що повторно використовуються, доступні через npm – менеджер пакетів в екосистемі Node.js.

3.5 Express.js

Express – це мінімалістичний та гнучкий веб-фреймворк для програм Node.js, що надає широкий набір функцій для мобільних та веб-додатків.

Маючи у своєму розпорядженні безліч службових методів HTTP та проміжних обробників, створити надійний API можна швидко та легко.

Express надає тонкий шар фундаментальних функцій веб-застосунків, які не заважають працювати з давно знайомими та улюбленими функціями Node.js.

3.6 MongoDB

MongoDB – система управління базами даних, яка працює з документоорієнтованою моделлю даних. На відміну від реляційних СУБД, MongoDB не потребує таблиці, схеми або окремої мови запитів. Інформація зберігається як документ чи колекція.

Колекція – це група документів MongoDB. Є еквівалентом простої таблиці у реляційній базі даних. Колекція вміщена всередині однієї БД. Документ у колекції може мати різні поля. Найчастіше, всі документи в колекції можуть мати різні типи даних.

Розробники позиціонують продукт як проміжну ланку між класичними СУБД та NoSQL. MongoDB не використовує схеми, як це роблять реляційні бази даних, що підвищує продуктивність усієї системи.

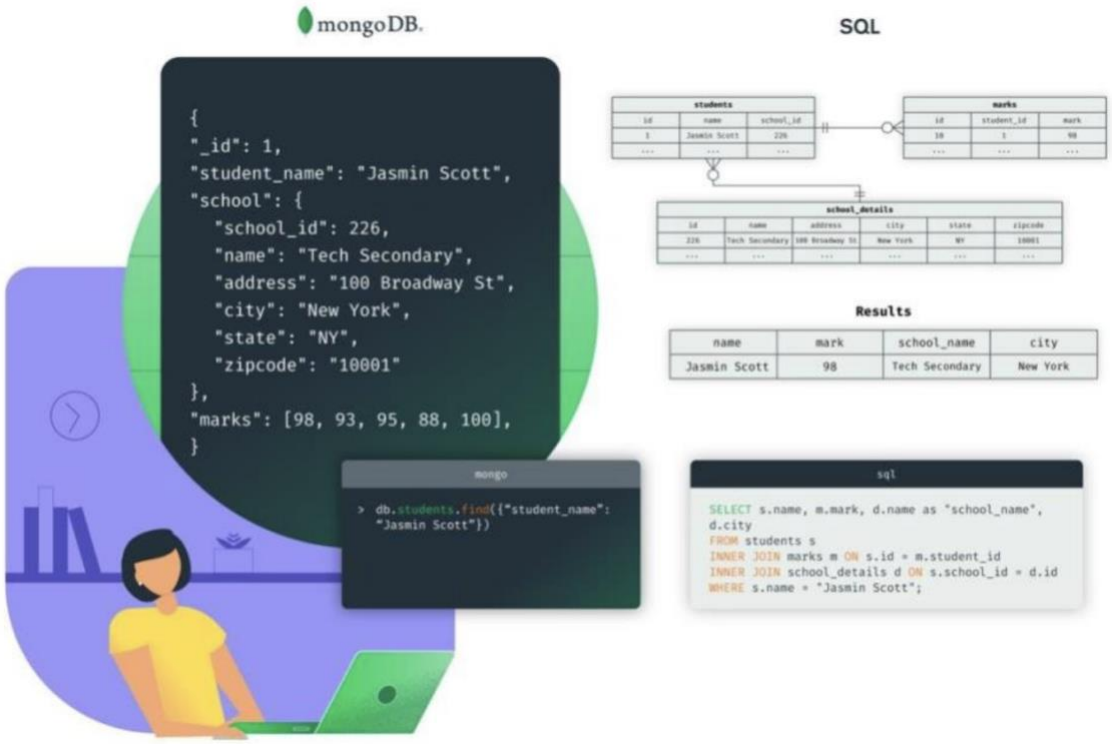

Рисунок 3.4 – MongoDB

Всі ці технології є складовими одного з найпопулярніших стеків для розробки повноцінних веб-застосунків MERN (MongoDB, Express.js, React, Node.js)

На рисунку 3.5 доволі чітко показано їх взаємодію. Сервер є посередником між базою даних та клієнтом, там ми визначаємо що саме та при яких умовах буде відправлено на сторону клієнта або записано у базу даних.

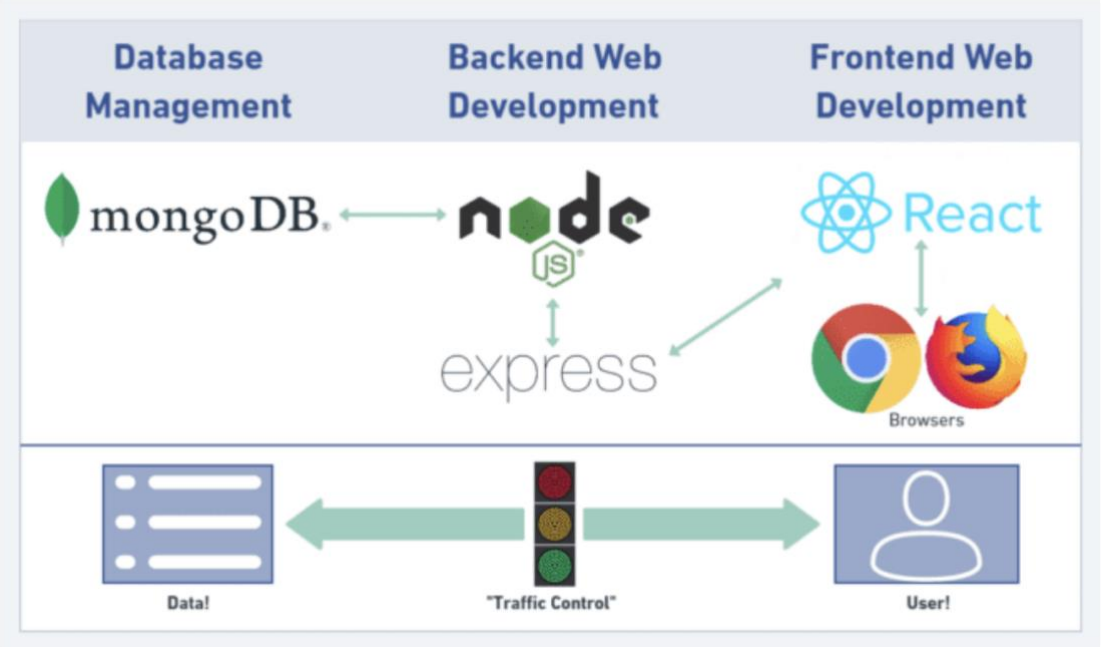

Рисунок 3.5 – Взаємодія між технологіями стеку MERN

3.7 Material UI

Material UI – бібліотека готових React компонентів. MUI пропонує повний набір інструментів інтерфейсу користувача, які допоможуть швидше почати працювати над логікою програми, а не витрачати велику кількість часу на стилі. Окрім цього MUI дозволяє займатись кастомізацією стилів, що в свою чергу дозволяє не обмежуватись лише стилями котрі доступні в бібліотеці, а й запроваджувати власні.

РОЗДІЛ 4. РОЛІ КОРИСТУВАЧІВ

В кожному багатофункціональному сайті не обійтись без різного роду користувачів з різноманітними можливостями та правами доступу. Були розроблені наступні ролі та їх можливості:

4.1 Гість

Гість – відвідувач сайту з базовими можливостями. В нього недоступна більшість можливостей, але те що зазвичай і потрібно гостям доступно, а саме пошук потрібних товарів, додавання їх до карти для отримання остаточної ціни та можливість зареєструватись.

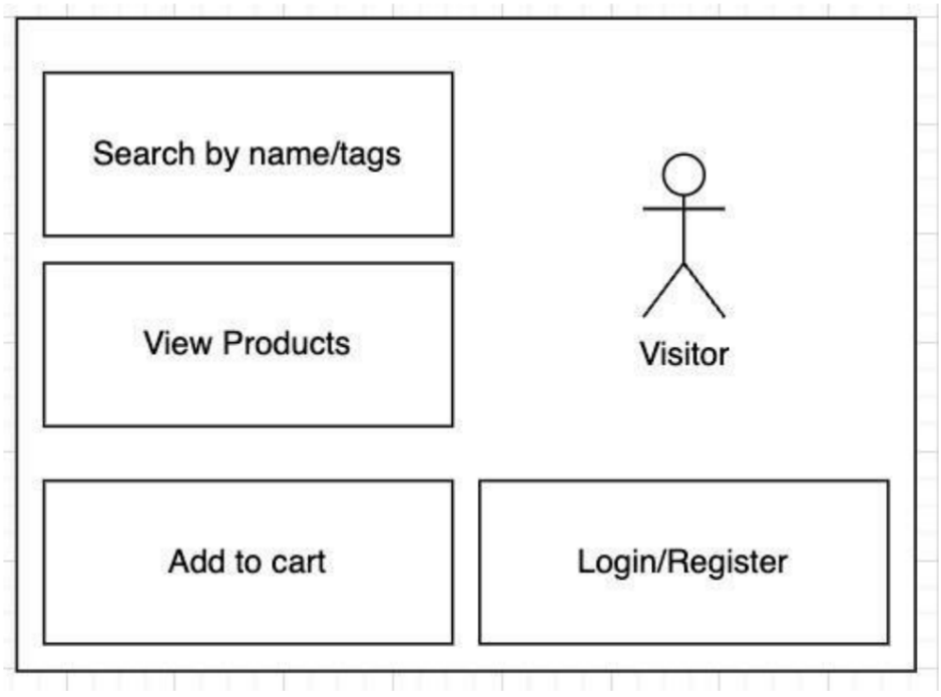

Рисунок 4.1 – Можливості гостя сайту

4.2 Зареєстрований користувач

Зареєстрований користувач – користувач який має в доступі всі базові можливості гостя, а також можливість лайкати, створювати замовлення та переглядати їх, створювати, видаляти та редагувати рецензії.

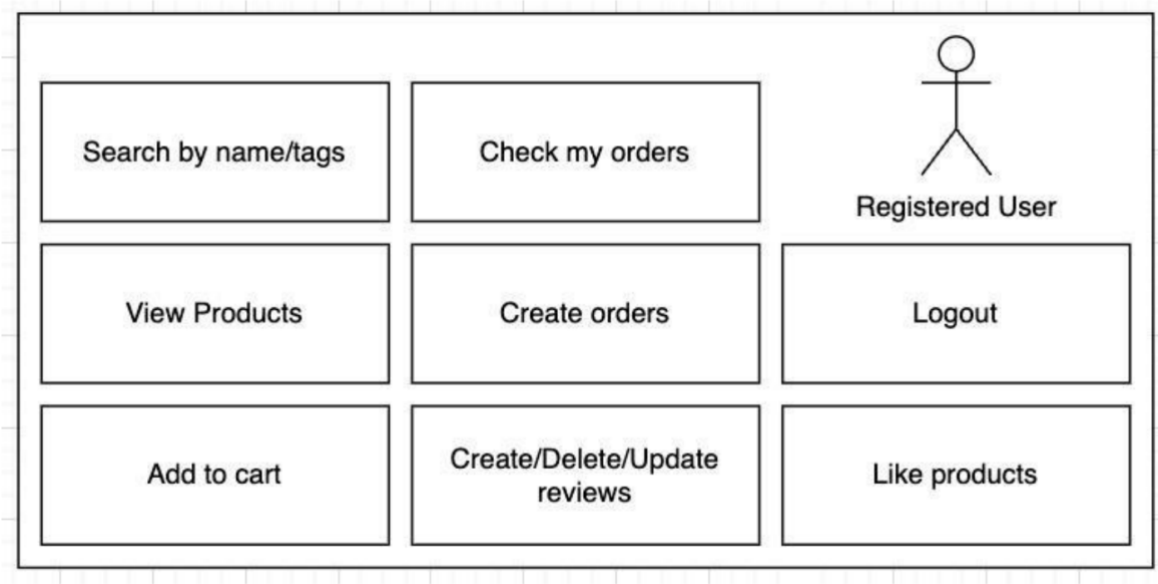

Рисунок 4.1 – Можливості зареєстрованого користувача сайту

4.3 Адміністратор

Адміністратор – користувач з можливостями доступними зареєстрованому користувачеві, а також створювати, редагувати та видаляти продукти, створювати, редагувати та видаляти купони, переглядати замовлення всіх користувачів, змінювати статус замовлень користувачів.

Рисунок 4.1 – Можливості адміністратора сайту

РОЗДІЛ 5. ОГЛЯД ВЕБСАЙТУ

Домашня сторінка Звідси (Рисунок 5.1) користувачі можуть бачити базову інформацію про продукти, шукати продукти за назвою, шукати за тегом, додавати до корзини та переходити до інших сторінок таких як реєстрація, логінізація та детальніших сторінок продуктів, лайкати (якщо користувач зареєстрований).

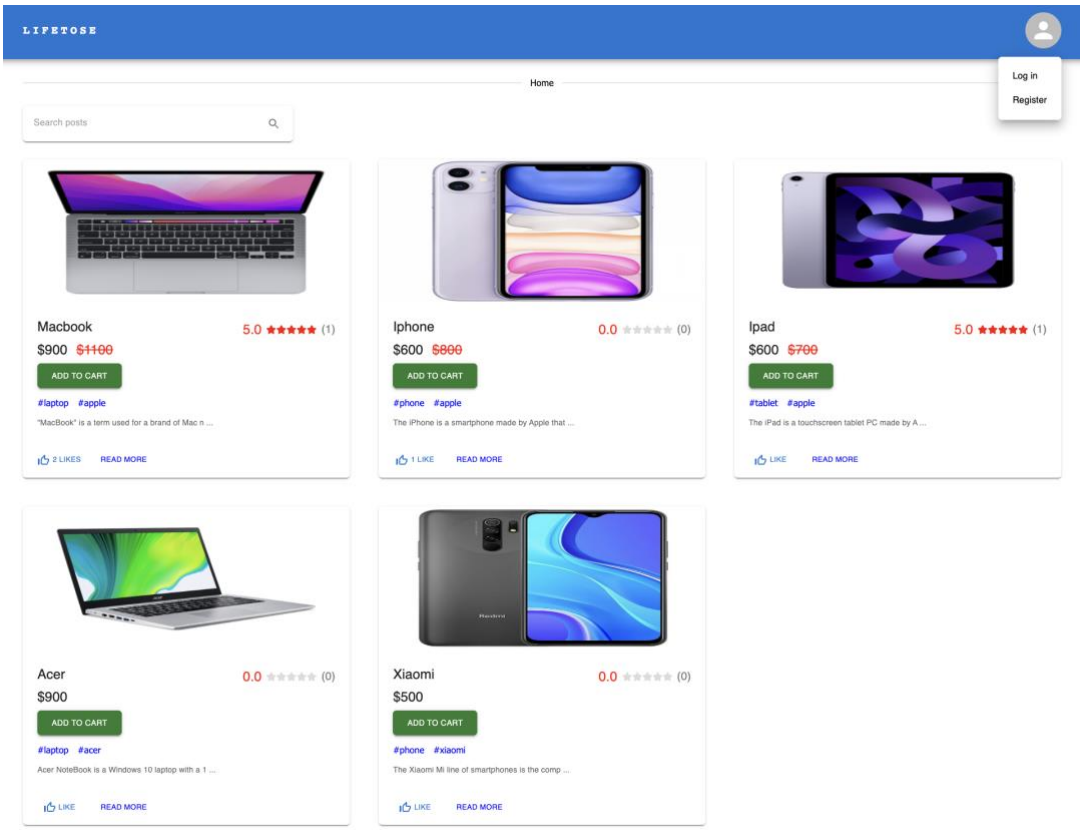

Рисунок 5.1 – Домашня сторінка

Компонент логіну

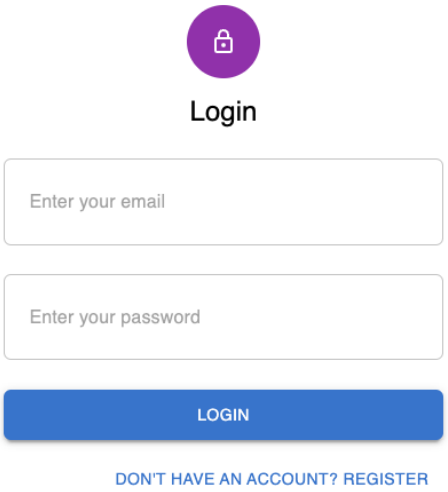

Рисунок 5.2 – Компонент логіну

Компонент реєстрації

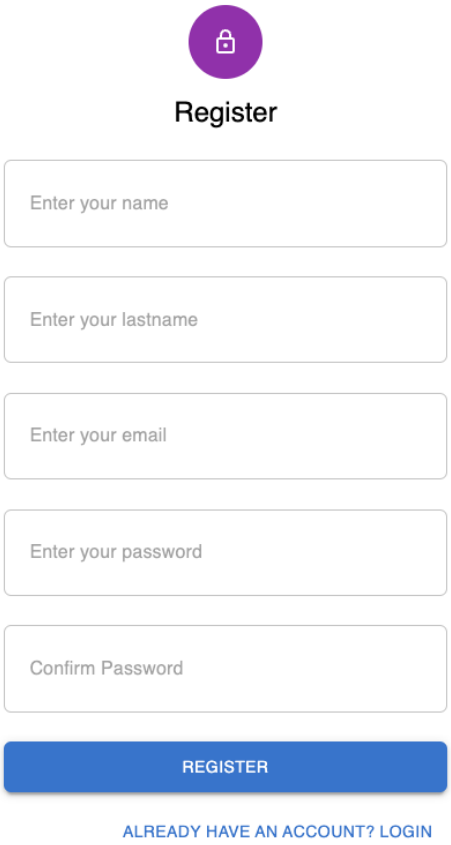

Рисунок 5.3 – Компонент реєстрації

Продукти з відповідним тегом

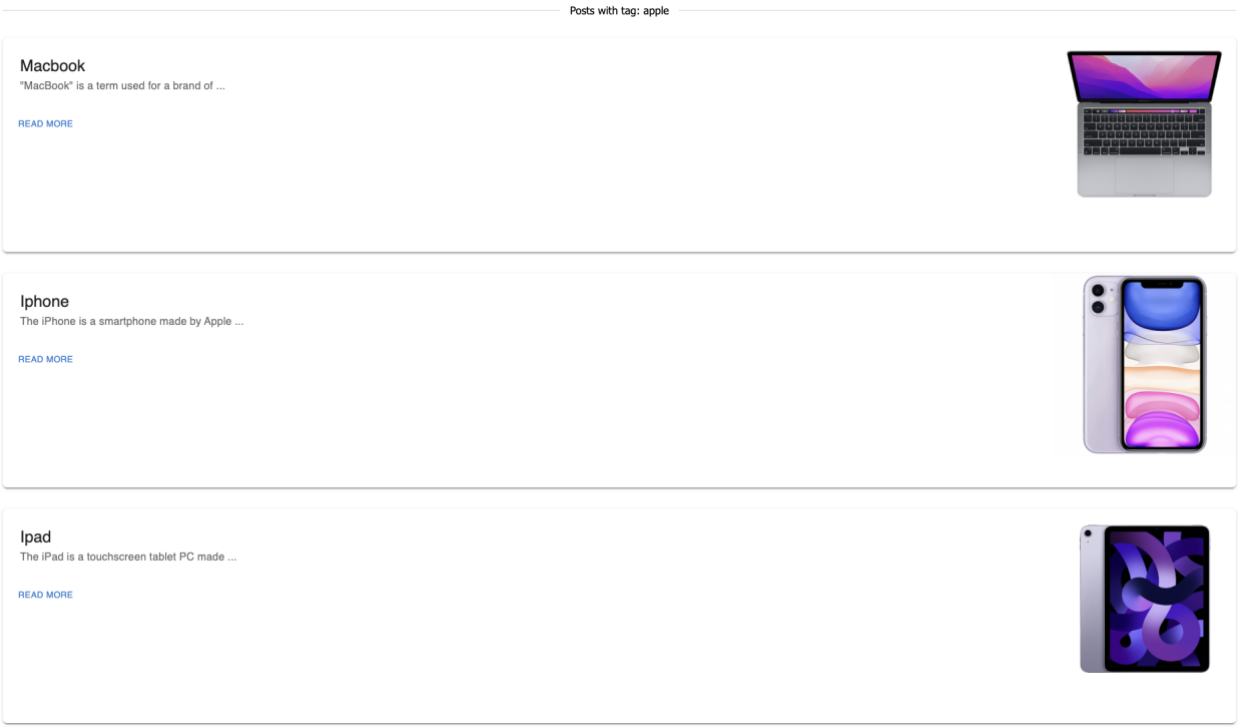

Рисунок 5.4 – Пошук продуктів за тегом

Сторінка продукту

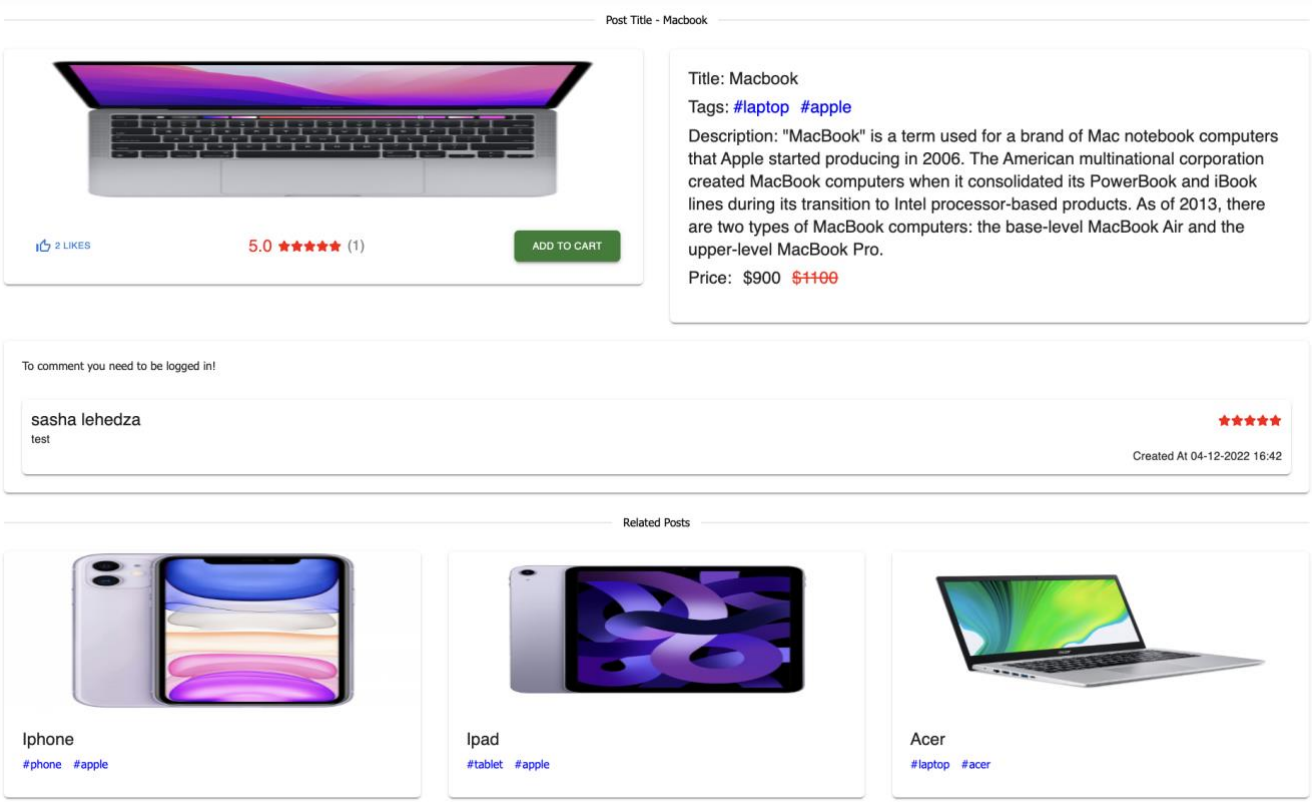

Рисунок 5.5 – Сторінка продукту

На цій сторінці (Рисунок 5.5) ми можемо побачити детальну інформацію відповідного продукту, залишити рецензію яка складається з тексту та оцінки (від 1 до 5), побачити обмежену кількість схожих за тегами продуктів (3 одиниці). Також користувач може редагувати або видаляти власну рецензію, лайкати і додавати продукт до корзини.

Редагування рецензії виглядає наступним чином

Рисунок 5.6 – Інтерфейс редагування рецензії

Сторінка корзини

Звідси (Рисунок 5.7) користувач може бачити детальну інформацію про товари які він додав до корзини, редагувати кількість товарів або видаляти їх, додавати купон та спосіб доставки. Також як можна побачити продукти які мають відповідні знижки, або ж оптові параметри наділені відповідними підказками для користувачів. Купон характеризується назвою та відсотком який буде застосований до корзини при введені правильної назви.

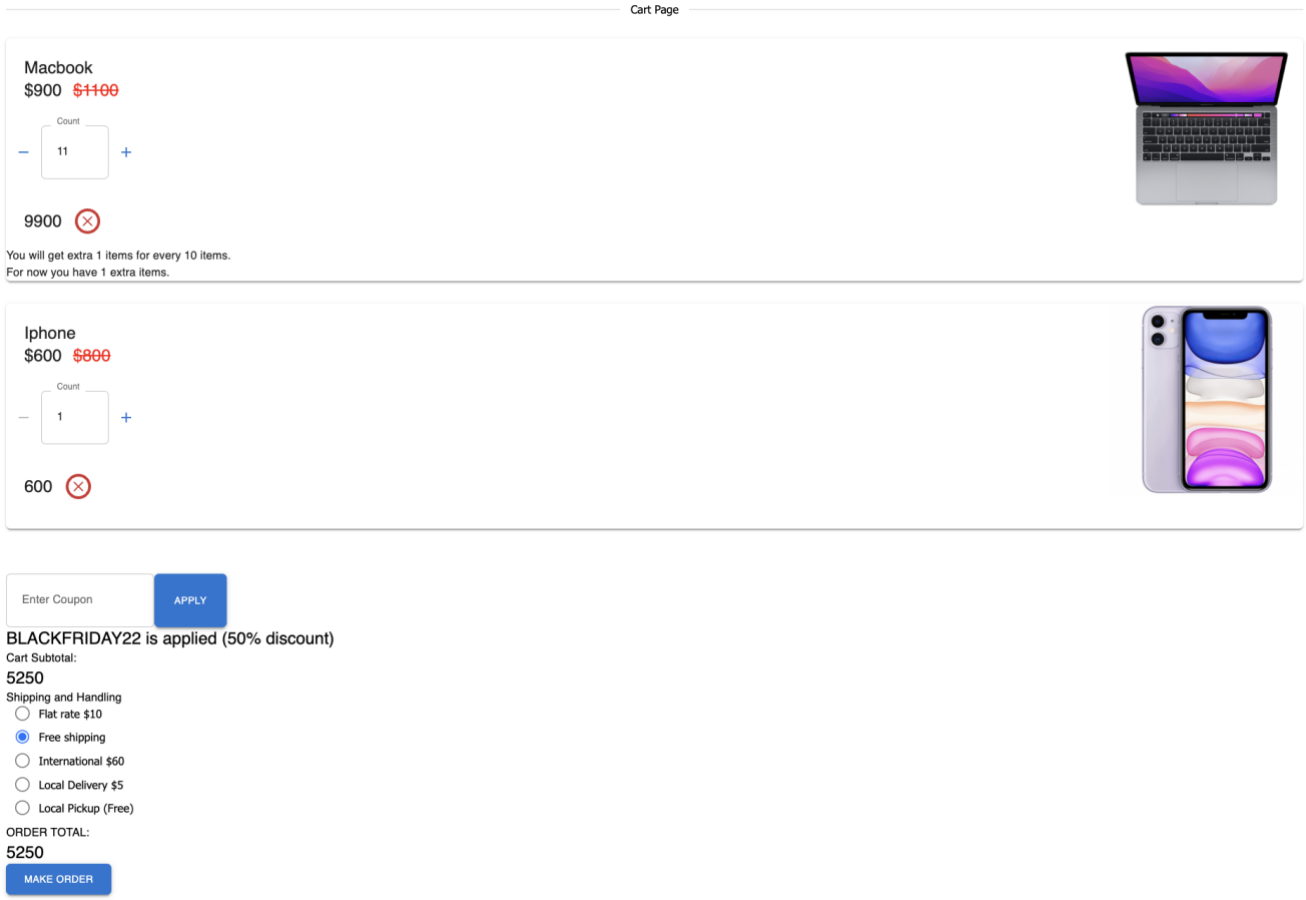

Рисунок 5.7 – Сторінка корзини

Повідомлення при введені неправильного купону

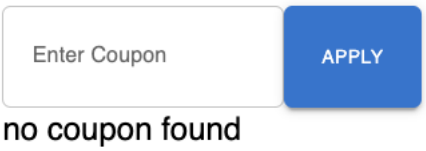

Рисунок 5.8 – Повідомлення при введені неправильного купону

Повідомлення при введені правильного купону

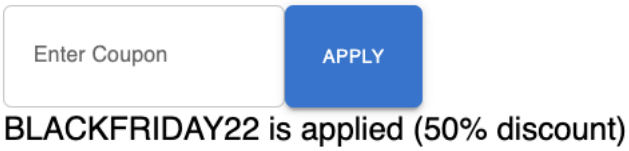

Рисунок 5.9 – Повідомлення при введені правильного купону

Повідомлення при спробі введення додаткового купону

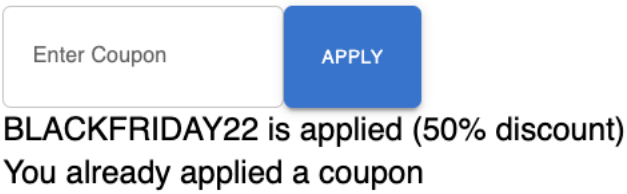

Рисунок 5.9 – Повідомлення при спробі введення додаткового купону

Сторінка замовлень користувача

Звідси (Рисунок 5.10) користувач може ознайомитись зі своїми замовленнями. Побачити їх статуси оплати та доставки.

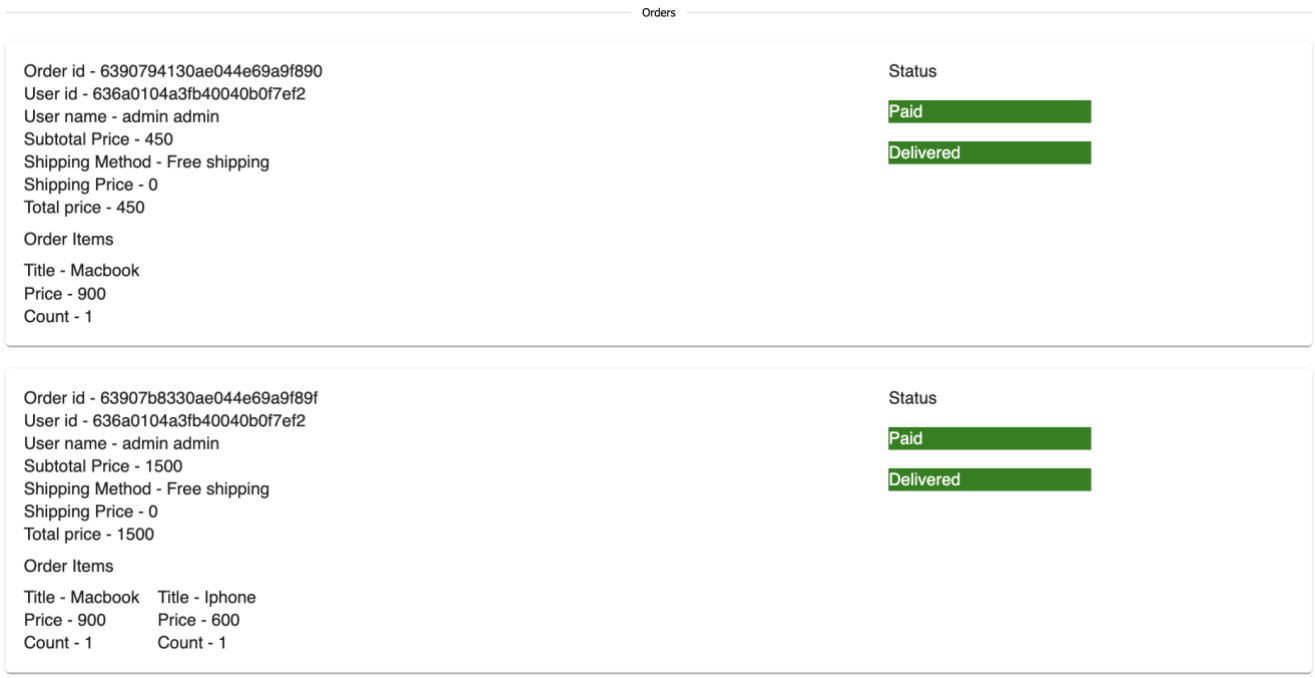

Рисунок 5.10 – Сторінка замовлень користувача

Сторінка маніпуляцій з продуктами

Звідси (Рисунок 5.11) адміністратор може додати, видалити або редагувати продукт.

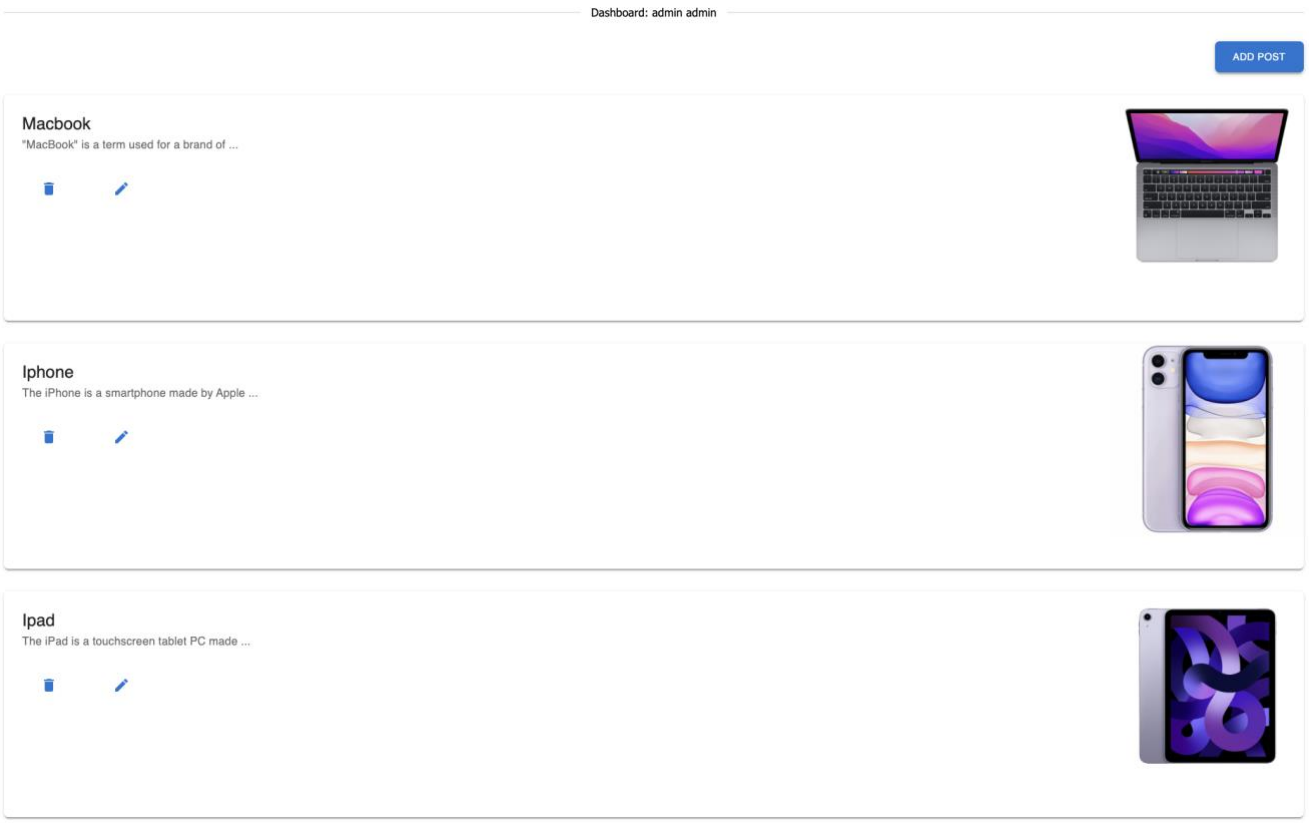

Рисунок 5.11 – Сторінка маніпуляцій з продуктами

Компонент створення купонів

Звідси (Рисунок 5.12) користувач може додавати купони (їхню назву та відсоток який буде застосований до корзини).

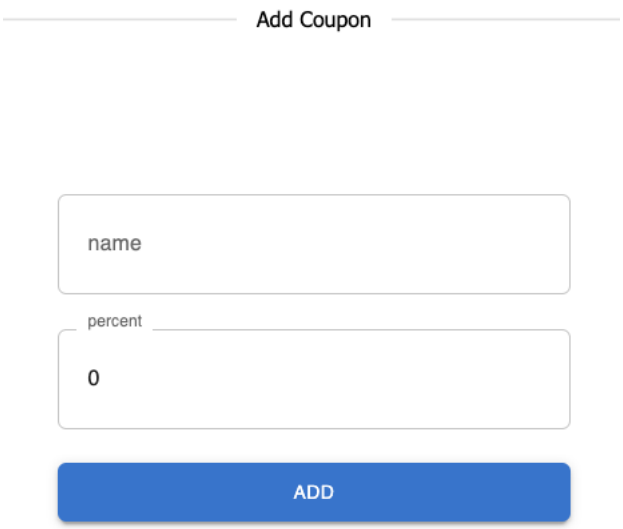

Рисунок 5.12 – Компонент створення купонів

Сторінка замовлень

Звідси (Рисунок 5.13) адміністратор може переглядати замовлення всіх користувачів та змінювати їх статуси оплати та доставки.

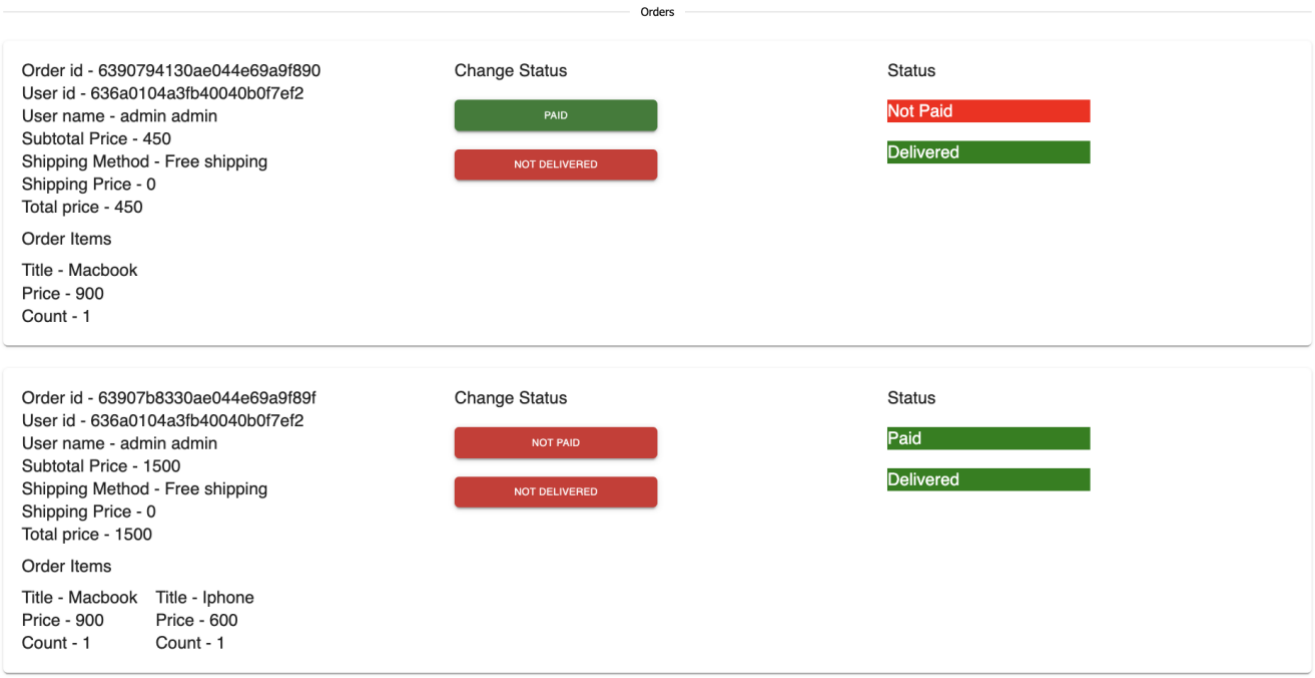

Рисунок 5.13 – Сторінка замовлень

Сторінка купонів

Звідси адміністратор може створювати, редагувати та видаляти існуючі купони.

Рисунок 5.14 – Сторінка купонів

Компонент створення продукту

Звідси (Рисунок 5.15) адміністратор може створювати продукт, додавати такі параметри як назва, опис, ціна, теги. Необов`язковими параметрами є наявність знижки та оптової кількості та вартості.

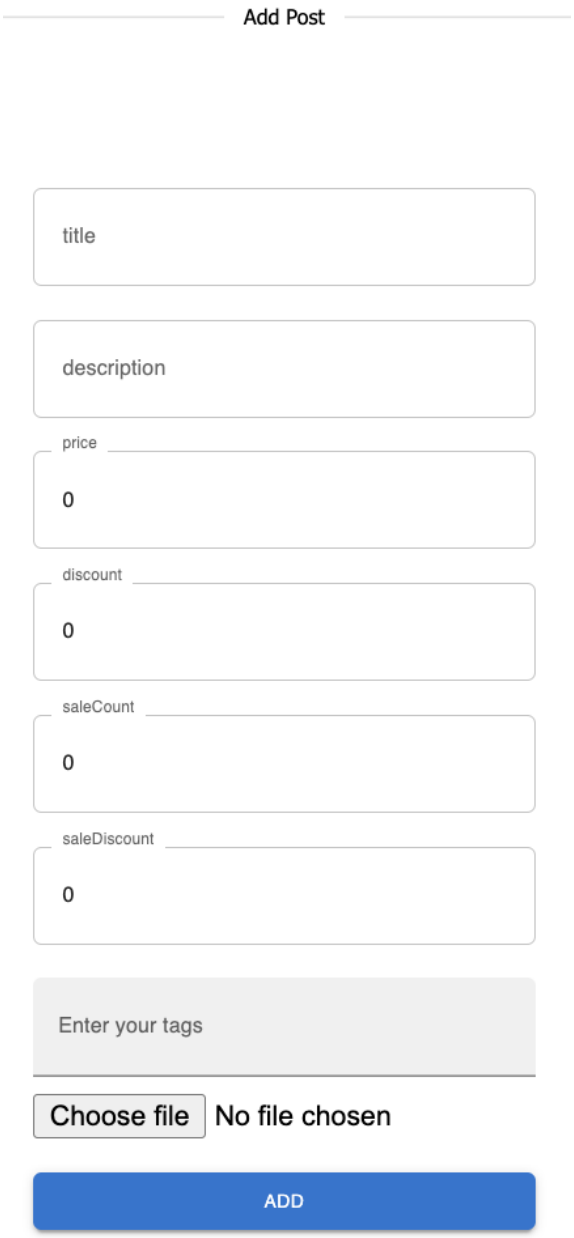

Рисунок 5.15 – Компонент створення продукту

РОЗДІЛ 6. ІМПЛЕМЕНТАЦІЯ

Весь функціонал є доволі різноманітним, але в більшості він розробляється по одному і тому ж патерну який відрізняється лише кількістю аргументів, формул

та розрахунків. Тому далі буде наведено приклад як розробляється рецензія до посту від бекенду до фронтенду для розуміння цього процесу.

Зберігати пов`язані між собою дані в MongoDB можна як в окремих колекціях так і вкладено в документі, на вибір розробника. Саме з рецензіями було вирішено спробувати другий варіант. Додаємо для нашого товару (Рисунок 6.1) 3 нові поля (reviews, rating, numReviews).

```
const postSchema = mongoose.Schema(
  {
    title: String,
    description: String,
    name: String,
    creator: String,
    price: Number,
    tags: [String],
    imageFile: String,
    createdAt: {
      type: Date,
      default: new Date(),
    },
    likes: {
      type: [String],
      default: [],
    },
    discount: Number,
    saleCount: Number,
    saleDiscount: Number,
    reviews: [],
    rating: {
      type: Number,
      required: true,
      default: 0,
    },
    numReviews: {
      type: Number,
      required: true,
      default: 0,
    },
    price: {
      type: Number,
      required: true,
      default: 0,
    },
  },
  { timestamps: true }
)
```

Рисунок 6.1 – Модель товару

В контролері (Рисунок 6.2) ми з допомогою запиту req.body приймаємо вхідні дані з запиту та id товару з допомогою req.params. Шукаємо відповідний пост і перевіряємо чи залишав даному товарові рецензію залогінений користувач. Якщо так то запит не пройде, якщо так то ми перевіряємо чи вхідні дані підходять рецензійній моделі і якщо вони підходять ми додаємо рецензію до нашого товару, змінюємо відповідно кількість рецензій та вираховуємо рейтинг.

```
export const createPostReview = async (req, res) => {
  const { rating, text } = req.body

  const post = await PostModal.findById(req.params.id)

  if (post) {
    const alreadyReviewed = post.reviews.find(
      (r) => r.user.toString() === req.userId.toString()
    )

    if (alreadyReviewed) {
      res.status(400).json({ message: 'Something went wrong' })
    }

    let review = CommentModal({
      text,
      rating: Number(rating),
      name: req.user.name,
      user: req.user._id,
      createdAt: new Date(),
      updatedAt: new Date(),
    })

    post.reviews.push(review)

    post.numReviews = post.reviews.length

    post.rating =
      post.reviews.reduce((acc, item) => item.rating + acc, 0) /
      post.reviews.length

    await post.save()
    res.json(post)
  } else {
    res.status(404).json({ message: 'Something went wrong' })
  }
}
```

Рисунок 6.2 – Контролер створення рецензії для товару

Створюємо route (Рисунок 6.3) з допомогою якого визначаємо шлях за яким ми зможемо до контролеру достукуватись.

```
router.post('/:id/reviews/create', auth, createPostReview)
```

Рисунок 6.3 – Створення route

Далі з Frontend нашого проекту ми створюємо запит (Рисунок 6.4) який буде приймати відповідні параметри і передавати їх контролерові.

```
export const createPostReview = (id, reviewData) =>
  API.post(`/post/${id}/reviews/create`, reviewData)
```

Рисунок 6.4 – Створення запиту

Обгортаємо запит в createAsyncThunk (Рисунок 6.5), адже дані будуть потрібні нам як глобально так і для оновлення продукту на стороні клієнта задля запобігання потреби перезавантажувати сторінку або виконувати повторно запит щоб побачити результат виконання запиту.

```
export const createPostReview = createAsyncThunk(
  'post/createPostReview',
  async ({ id, reviewData }, { rejectWithValue }) => {
    try {
      const response = await api.createPostReview(id, reviewData)
      return response.data
    } catch (err) {
      toast.error(extractErrorMessage(err))
      return rejectWithValue(extractErrorMessage(err))
    }
  }
)
```

Рисунок 6.5 – Обгортка для подальшого запису даних в global state

Результат який повертає нам Backend записуємо в global state (Рисунок 6.6).

```
[createPostReview.pending]: (state, action) => {},
[createPostReview.fulfilled]: (state, action) => {
  state.post = action.payload
},
[createPostReview.rejected]: (state, action) => {},
```

Рисунок 6.6 – Запис даних в global state

Виклик запиту зі сторони клієнта виглядає наступним чином (Рисунок 6.7).

```
const { id } = useParams()
const [text, setText] = useState('')
const [rating, setRating] = useState(5)

const addCommentHandler = async () => {
  try {
    let reviewData = { text, rating }
    dispatch(createPostReview({ id, reviewData }))
    setText('')
  } catch (err) {}
}
```

Рисунок 6.7 – Виклик запиту

І в кінцевому результаті дані зі state.post використовуємо для відображення на стороні клієнта (Рисунок 6.8)

```
<Box>
  <Comments comments={post.reviews} />
</Box>
```

Рисунок 6.8 – Код для відображення даних

ВИСНОВКИ

Під час виконання цієї магістерської роботи я розробив повноцінний онлайн магазин з доволі обширним функціоналом. Взяв до уваги популярні аналоги такі як Rozetka та Udemy, загострив увагу на цікавих можливостях та власноруч розробив присутній там функціонал для свого магазину.

В кінцевому результаті раніше розроблений мною сайт для створення блогів був розширений до онлайн магазину. Структура та стиль написання проекту підбирались таким чином, щоб при зацікавленості створювати подальший функціонал іншим розробникам було на рівні інтуїції зрозуміло куди саме в коді потрібно зайти, які технології для написання у разі потреби завчити або повторити та яким саме стилем спробувати опанувати, щоб від розробника до розробника проект не перетворювався у набір папок в яких дуже важко орієнтуватись і незрозуміло звідки починати.

У розробці використовував новітні технології такі як React 18 версії на функціональних компонентах, Redux Toolkit, Material UI.

Сайт вийшов інтуїтивно зрозумілим, а також масштабованим, тому підходить як для десктопних так і для мобільних платформ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] React documentation. Available from: https://reactjs.org/docs/getting-started.html

[2] Material UI documentation. Available from: https://mui.com/materialui/gettingstarted/overview/

[3] Redux Toolkit documentation. Available from: https://reduxtoolkit.js.org/introduction/getting-started

[4] Express.js documentation. Available from: https://expressjs.com/

[5] Mongoose documentation. Available from: https://mongoosejs.com/docs/queries.html

[6] MDN JS documentation. Available from: https://developer.mozilla.org/enUS/docs/Learn/JavaScript

[7] Formik documentation. Available from: https://formik.org/docs/overview

[8] Rozetka. Available from: https://rozetka.com.ua/

[9] Udemy. Available from: https://www.udemy.com/

[10] JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. Flanagan D. 7th edition. Available from: https://www.amazon.com/JavaScript-Definitive-Most-Used-ProgrammingLanguage/dp/1491952024

[11] React Redux Toolkit tutorials by Dave Gray. Available From: https://www.youtube.com/watch?v=u3KlatzB7GM&list=PL0Zuz27SZ-6M1J5I1w2-uZx36Qp6qhjKo

ДОДАТОК А. КОД ДЛЯ РОЗРАХУНКУ ЦІНИ ПРОДУКТОВОЇ КОРЗИНИ

```
export const discountCalc = (price, discount, count) => {
  return Number(discount) && Number(discount) !== 0
    ? (Number(price) - Number(discount)) * Number(count)
    : Number(price) * Number(count)
}

export const subtotalCalc = (couponname, couponpercent, carts) => {
  return couponname
    ? carts.reduce((accumulator, product) => {
        return (
          accumulator +
          discountCalc(product.price, product.discount, product.count)
        )
      }, 0) *
        ((100 - couponpercent) / 100)
    : carts.reduce((accumulator, product) => {
        return (
          accumulator +
          discountCalc(product.price, product.discount, product.count)
        )
      }, 0)
}
```

ДОДАТОК Б. КОД ДЛЯ РОЗРАХУНКУ ТА ВИВЕДЕННЯ У ПРАВИЛЬНОМУ ФОРМАТІ РЕЙТИНГУ ТОВАРУ

```
post.rating =
    post.reviews.reduce((acc, item) => item.rating + acc, 0) /
    post.reviews.length


const RateStatic = ({ count, rating, color }) => {
  const getColor = (index) => {
    return rating >= index
      ? color.filled
      : rating >= index - 0.5
      ? color.filled
      : color.unfilled
  }

  const starRating = useMemo(() => {
    return Array(count)
      .fill(0)
      .map((_, i) => i + 1)
      .map((idx) => (
        <FontAwesomeIcon
          key={idx}
          className='cursor-pointer'
          icon={
            rating >= idx ? faStar : rating >= idx - 0.5 ? faStarHalf : faStar
          }
          style={{ color: getColor(idx) }}
        />
      ))
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [count, rating])

  return <>{starRating}</>
}
```