МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

(підпис)

01 червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерні науки,

освітньо-професійної програми «Інформатика»

на тему: «Вебсайт інтернет-магазину жіночого одягу»

здобувача групи ІН-02 Глеби Дениса Івановича

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

Денис ГЛЕБА

(підпис)

Асистент

кафедри комп'ютерних наук,

кандидат фізико-математичних наук Олександр ВЛАСЕНКО

(підпис)

Суми – 2024

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

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

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 – Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувача групи ІН-02 Глеби Дениса Івановича

1. Тема роботи: «Вебсайт інтернет-магазину жіночого одягу» затверджена наказом по СумДУ від *«22» квітня 2024 р. № 0414-VI*

2. Термін здачі здобувачем кваліфікаційної роботи *до 01 червня 2024 року*

3. Вхідні дані до кваліфікаційної роботи _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити): 1) *Інформаційний огляд, аналіз предметної області, постановка задачі, моделювання інформаційної системи. 2) Вибір методів рішення задачі. 3) Програмна реалізація. 4)Взаємодія користувача з системою.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____

6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «08» квітня 2024 р.

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

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Затвердження теми роботи	22.04.2024	
2	Вивчення та аналіз задачі	7.05.2024	
3	Поглиблене дослідження бібліотек, інструментів, технологій, що будуть використовуватися	10.05.2024	
4	Програмна реалізація системи	19.05.2024	
5	Аналіз отриманих результатів	22.05.2024	
6	Оформлення пояснювальної записки до кваліфікаційної роботи	25.05.2024	

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

АНОТАЦІЯ

Записка: 107 стр., 76 рис., 1 таблиця, 26 додатків, 17 використаних джерел.

Обґрунтування актуальності теми роботи – актуальність розробки вебсайту інтернет-магазину жіночого одягу обумовлена стрімким зростанням популярності онлайн-шопінгу як в Україні, так і у світі..

Об'єкт дослідження – процес розробки вебсайту та отримання знань щодо останніх технологій у сфері веброзробки..

Мета роботи – створення вебсайту інтернет-магазину жіночого одягу з максимально простим і зрозумілим інтерфейсом.

Методи дослідження – дослідження актуальних трендів у електронній торгівлі, оцінка найефективніших способів взаємодії з користувачами та порівняльний аналіз з іншими учасниками ринку.

Результати – було створено вебсайт інтернет-магазину жіночого одягу з максимально простим і зрозумілим інтерфейсом. Реалізовано функціонал пошуку, фільтрації по категоріям, сортування за різними критеріями. Опановано найновітніші методи розробки веб додатків.

ІНФОРМАЦІЙНА СИСТЕМА, WEB-РОЗРОБКА, REACT, TYPECRIPT, REDUX TOOLKIT, AXIOS, SASS, MOCKAPI, VERCEL, АДАПТИВНИЙ ВЕБСАЙТ

ЗМІСТ

ВСТУП

Актуальність. Актуальність розробки вебсайту інтернет-магазину жіночого одягу обумовлена стрімким зростанням популярності онлайн-шопінгу як в Україні, так і у світі. Згідно з останніми дослідженнями, обсяг інтернет-торгівлі продовжує збільшуватися з кожним роком, що підвищує попит на якісні та зручні платформи для покупок. В епоху цифрових технологій користувачі очікують на високий рівень обслуговування, зокрема інтуїтивно зрозумілий інтерфейс, швидке завантаження сторінок та зручні інструменти для пошуку і фільтрації товарів.

Сучасний вебсайт інтернет-магазину має забезпечувати не лише функціональність, але й відповідати вимогам адаптивності, щоб користувачі могли зручно користуватися ним з будь-якої роздільної здатності пристрою. Це особливо важливо, враховуючи, що значна частина покупок здійснюється саме з мобільних пристроїв. Окрім того, ефективний дизайн та зручна навігація сприяють підвищенню рівня задоволеності клієнтів та їхній лояльності.

В умовах зростаючої конкуренції серед інтернет-магазинів, здатність швидко адаптуватися до змін і впроваджувати новий функціонал є ключовою перевагою. Тому особливу увагу слід приділити вибору технологій, які забезпечать можливість легкого розширення функцій вебсайту в майбутньому.

Мета: Створити вебсайт інтернет-магазину жіночого одягу з максимально простим і зрозумілим інтерфейсом.

Об’єкт дослідження. Процес розробки вебсайту та отримання знань щодо останніх технологій у сфері веброзробки.

Предмет дослідження. Рішення та методи для створення вебсайту розташування компонентів та стилізація текстового контенту задля створення зрозумілого інтерфейсу інтернет-магазину, який можна буде легко підтримувати та розширювати, додавши новий функціонал.

Гіпотеза цього дослідження полягає в тому, що розробка сучасного вебсайту інтернет-магазину жіночого одягу з використанням передових

вебтехнологій та орієнтацією на зручність користувачів значно покращить якість користувацького досвіду, підвищить лояльність клієнтів та сприятиме успішному розвитку бізнесу в онлайн-середовищі.

Новизна фінального результату полягає у створенні вебсайту інтернет-магазину жіночого одягу з використанням сучасних технологій, таких як ReactJS, Typescript, SCSS-Modules, Redux Toolkit, MockAPI, Axios та Vercel. Цей підхід забезпечує високу продуктивність, адаптивність та легкість у розширенні функціональності вебсайту, що відповідає вимогам сучасних користувачів та ринкових тенденцій. Розробка такого вебсайту дозволить користувачам легко знайти та придбати необхідні товари, забезпечуючи при цьому високий рівень обслуговування та задоволення клієнтів.

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

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз існуючих рішень та предметної області

Вебсайти інтернет-магазинів жіночого одягу є дуже різноманітними та мають свої унікальні особливості, проте деякі загальні риси можна відзначити в більшості таких проектів.

- Дизайн інтерфейсу: важливо, щоб дизайн сайту був привабливим і легким у використанні. Використання яскравих кольорів, стильного шрифту та привабливих зображень одягу може привернути увагу відвідувачів.
- Інтернет-магазин жіночого одягу зазвичай має великий асортимент товарів, тому важливо мати добре організований каталог. Це може включати поділ товарів за категоріями (сукні, спідниці, блузи тощо), розмірами, кольорами та іншими параметрами.
- Зручний пошук і фільтрація: користувачам зручно мати можливість швидко знаходити потрібні товари за допомогою пошуку та фільтрів за різними параметрами, такими як ціна, розмір, колір тощо.
- Кожен товар повинен мати докладний опис, включаючи розміри, склад тканини, інструкції щодо догляду, а також якісні фотографії, що демонструють товар з різних кутів.
- Кошик і оформлення замовлення: зручний кошик покупок і простий процес оформлення замовлення допомагають користувачам легко купувати товари без зайвих труднощів.
- Опції оплати і доставки: інтернет-магазин повинен надавати різноманітні варіанти оплати (кредитні картки, платіжні системи, готівка при отриманні тощо) і доставки (кур'єрська доставка, поштові служби, самовивіз).
- Зворотний зв'язок і підтримка клієнтів: важливо мати можливість зв'язатися з покупцями для відповіді на їхні запитання, отримання відгуків і вирішення проблем, що виникають під час покупок.

Ці елементи сприяють створенню успішного вебсайту інтернет-магазину жіночого одягу, який привертає клієнтів і забезпечує їм приємний і зручний досвід покупок.

Давайте розглянемо декілька існуючих рішень – інтернет-магазинів, які працюють у сфері жіночого одягу та мають подібний функціонал, для проведення аналізу.

Розпочнемо з першого вебсайту, який ми проаналізуємо – https://leomirra.com.ua/ [1].

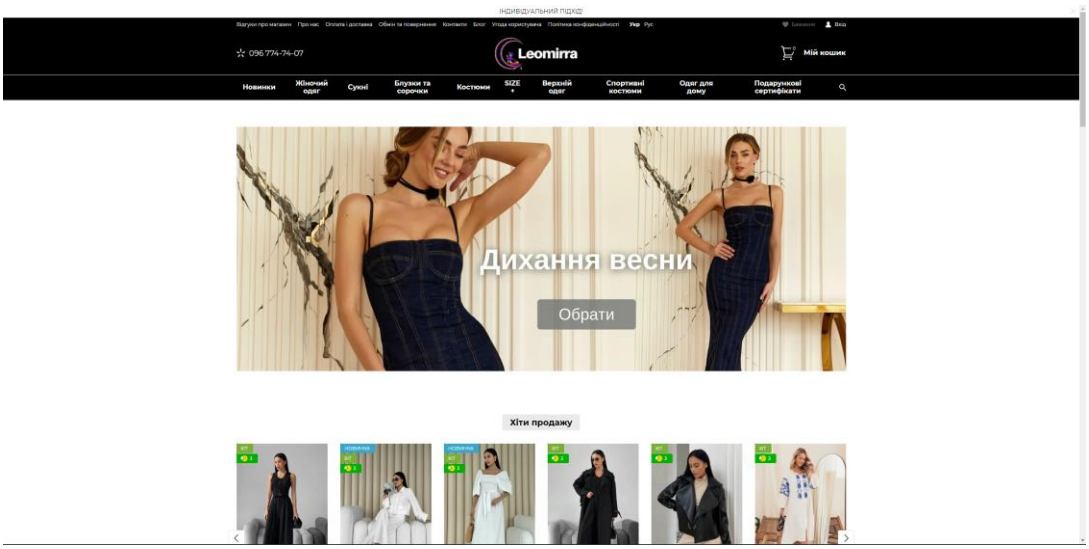

Рисунок 1.1 – Головна сторінка магазину Leomirra

Можемо побачити, що дизайн вигляду головної є доволі привабливим з гарними шрифтами, однак перейшовши в режим перегляду на смартфоні (див. рис.1.2), то в очі одразу кидається проблема з адаптивною верстkою. На роздільній здатності iPhone SE адаптивність повністю відсутня, що є ще однією суттєвою проблемою. Це означає, що вебсайт не оптимізований для перегляду на різних пристроях, таких як мобільні телефони або планшети, що може значно ускладнити навігацію і знизити зручність використання для користувачів, які використовують пристрої з іншими розмірами екрану.

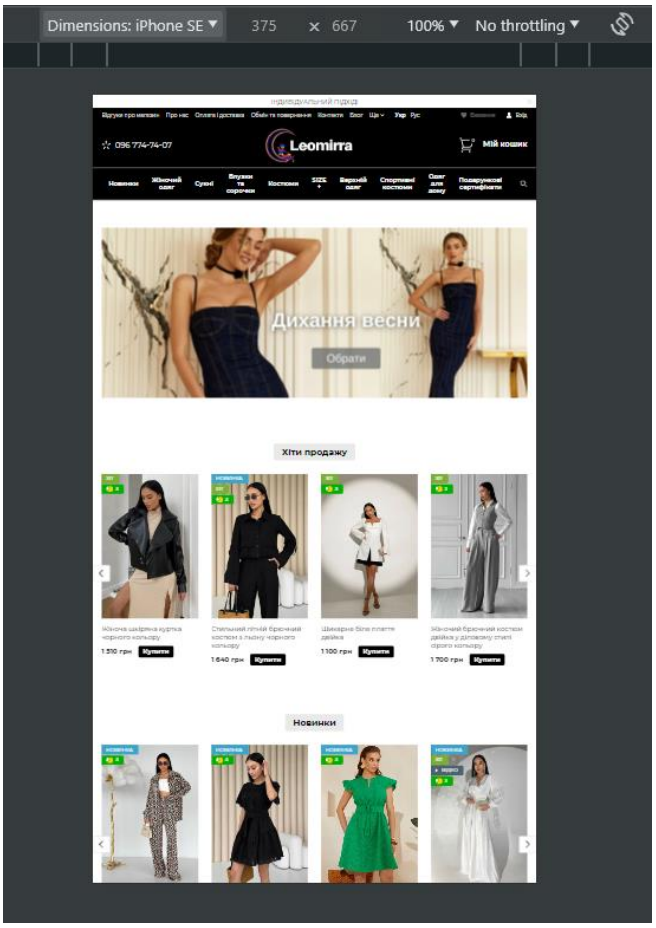

Рисунок 1.2 – Адаптивна верстка магазину Leomirra

Однак варто зауважити, що адаптивність – це єдина найслабша сторона цього сайту, так як каталог товарів, зручний пошук та фільтрація, інформація про товари і так далі присутні, що є великим плюсом для цього сайту, адже – це забезпечує кращий користувацький досвід.

Ще одним сайтом для аналізу розглянемо - https://maritel.com.ua/ [2]. Цей сайт також має достатньо привабливий дизайн, має зручний пошук та фільтрацію так само як і попередній його конкурент, але на відміну від нього цей сайт дуже гарно адаптований до різних роздільних здатностей екранів (див. рис. 1.4)

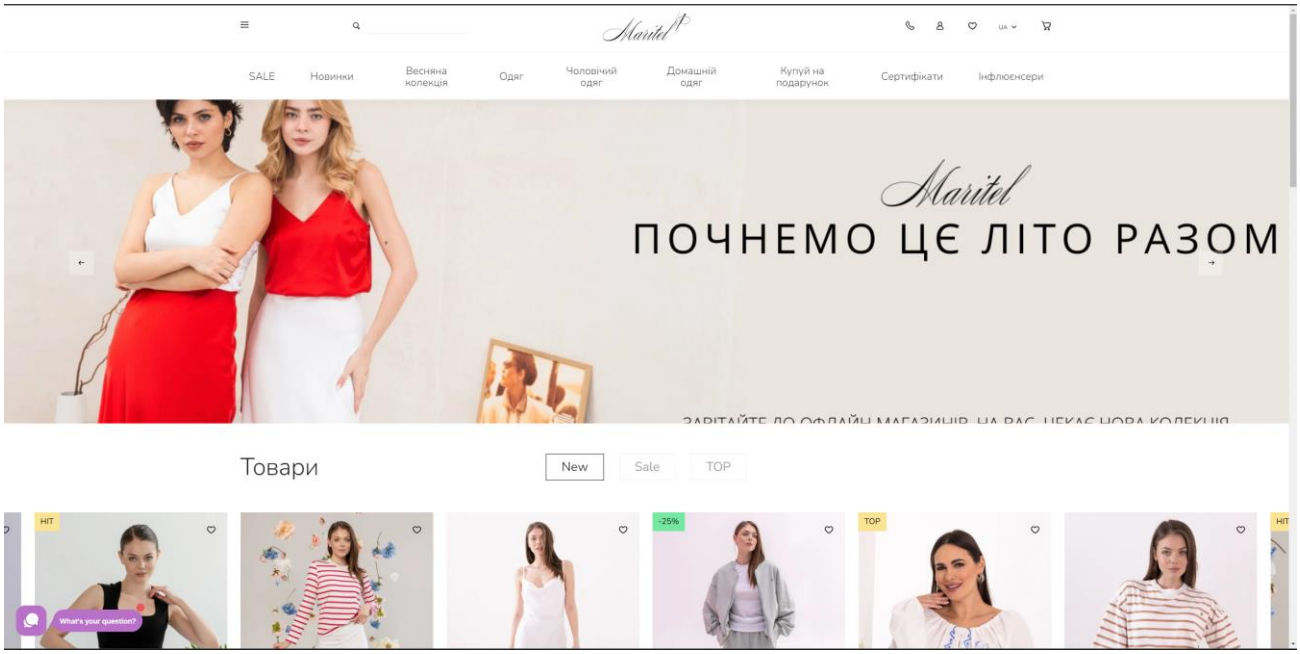

Рисунок 1.3 – Головна сторінка магазину Maritel

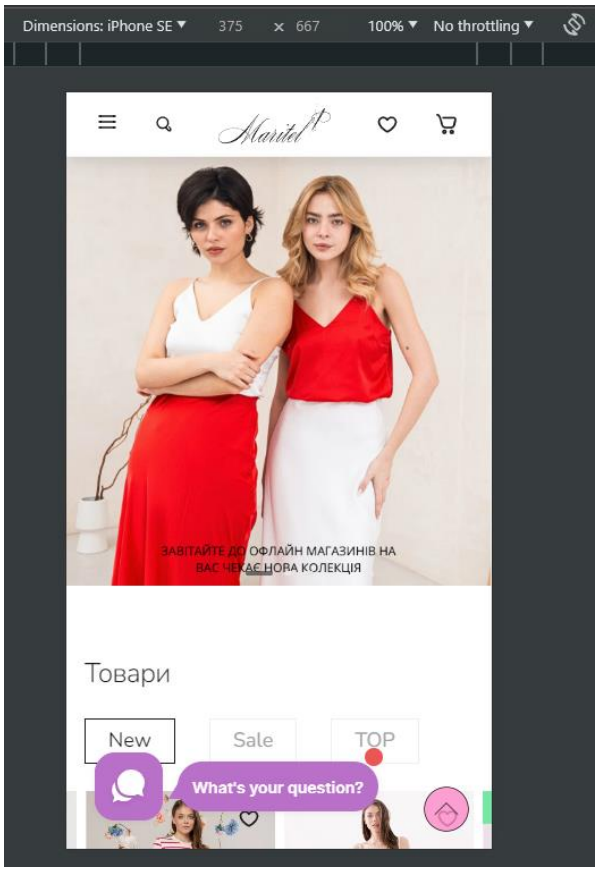

Рисунок 1.4 – Адаптивна верстка магазину Maritel

Однак, в деяких місцях, верстка цього сайту трішечки пливе (див. рис. 1.5), що сприяє не дуже гарному користувацькому досвіду, так як незаповнений простір розсіює увагу користувачів від основних товарів.

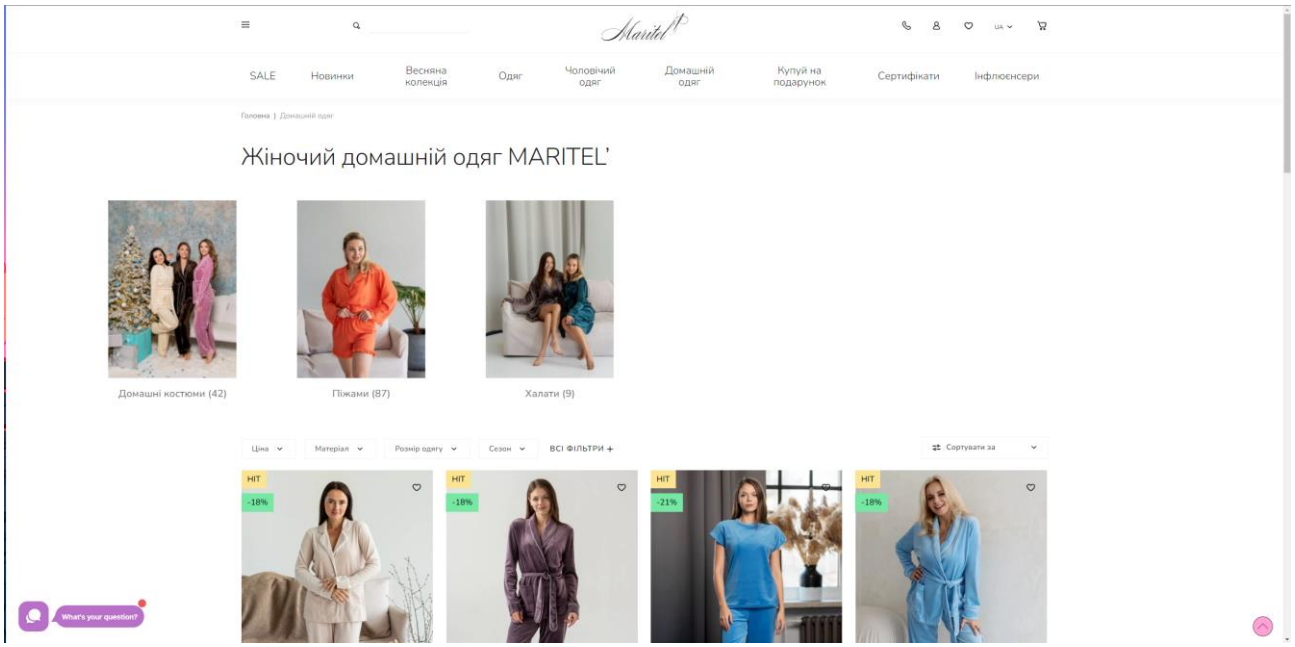

Рисунок 1.5 – Проблеми з верстkою магазину Maritel

Загальний аналіз обраних вебсайтів наведено в таблиці 1.

Таблиця 1 – Огляд та аналіз основних аспектів обраних вебсайтів

Аспект	Leomirra	Maritel
Зручний дизайн інтерфейсу	+	-
Каталог товарів	+	+
Зручний пошук і фільтрація	+	+
Інформація про товари	+	+
Кошик і оформлення замовлення	+	+
Присутність адаптивності	-	+

Після того як було проведено аналіз інтернет-магазинів, було прийняте рішення створити свій власний інтернет-магазин жіночого одягу, який уникне недоліки виявлені на проаналізованих сайтах.

1.2 Постановка задачі

Завданням кваліфікаційної роботи є створення вебсайту інтернет-магазину жіночого одягу, який буде забезпечувати користувачам зручний та приємний

досвід взаємодії. Основні аспекти, які необхідно врахувати при розробці сайту, включають:

1. Розробка дизайну головної сторінки та сторінки з асортиментом:

- Дизайн повинен бути привабливим, сучасним і відповідати очікуванням цільової аудиторії.
- Сторінки повинні мати чітку і логічну структуру, що дозволить користувачам легко орієнтуватися на сайті.

2. Створення простого та зрозумілого інтерфейсу:

- Інтерфейс має бути інтуїтивно зрозумілим для користувачів різних вікових категорій та технічного рівня.
- Навігація повинна бути максимально зручною, з мінімальною кількістю кліків для досягнення необхідного результату.

3. Імплементація функціоналу для пошуку, сортування та фільтрації товарів:

- Пошукова система повинна забезпечувати швидке та точне знаходження товарів за заданими параметрами.
- Можливість сортування товарів за різними критеріями, такими як ціна, популярність.
- Фільтрація товарів за категоріями та іншими характеристиками.

4. Забезпечення адаптивності вебсайту:

- Сайт повинен коректно відображатися на різних пристроях, включаючи мобільні телефони, планшети та комп’ютери.
- Інтерфейс і функціонал мають бути однаково зручними та доступними незалежно від розміру екрану.

5. Використання сучасних технологій веброзробки:

- Використання останніх технологій і стандартів для забезпечення високої продуктивності та безпеки сайту.
- Забезпечення можливості легкого розширення функціоналу в майбутньому.

У результаті виконання цих завдань буде створено вебсайт інтернет-магазину жіночого одягу, який забезпечить високий рівень зручності та задоволеності користувачів, сприятиме зростанню продажів і підвищенню конкурентоспроможності на ринку.

1.3 Моделювання інформаційної системи

Нижче подано приклад організації вебсторінок та їхні складові у вигляді діграми (див. рисунок 1.6).

Wrapper відповідає за вміст таких сторінок, як: Home, About, Catalog, Favorites, Cart, Contacts, ProductCard, Refund, і Delivery. Основними та незмінними для App є Header та Footer. MainHeader присутній лише на головній сторінці (Home) для більш гнучкого відображення елементів на різних сторінках. Вміст сторінок залежить від компонента App, який відображає зазначені сторінки залежно від URL-адреси.

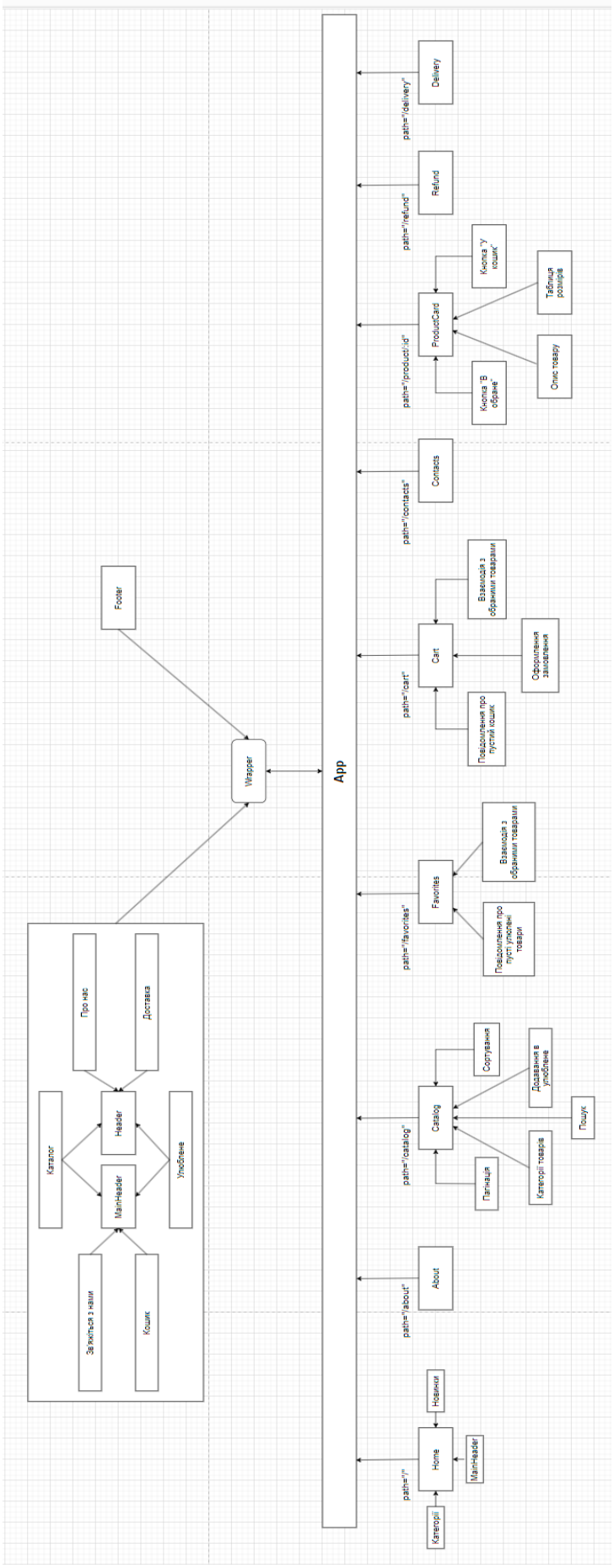

Рисунок1.6. Організація вебсторінок та їхні складові

2. МЕТОДИ ВИРІШЕННЯ ЗАДАЧІ

Розробка інтерактивного інтернет-магазину жіночого одягу передбачає використання наступних технологій:

2.1 ReactJS 18

ReactJS 18 – це оновлена версія популярної бібліотеки для створення інтерактивних вебінтерфейсів. Вона використовує концепцію компонентів для побудови користувацького інтерфейсу та надає можливість ефективного відслідковування та оновлення змін на сторінці без повного перезавантаження. React 18 включає в себе багато нових функцій і можливостей для розробників. [3]

2.2 Typescript

TypeScript - це мова програмування, яка розширює JavaScript додатковими можливостями типізації. Вона дозволяє визначати типи даних для змінних, функцій і об'єктів, що полегшує розробку великих та складних програм. TypeScript надає переваги, такі як покращена статична перевірка коду та підвищена читабельність. [6]

2.3 Інструменти для розробки

2.3.1 SCSS-Modules / SASS(SCSS syntax)

SCSS (Sassy CSS) – це розширення мови CSS з багатьма додатковими можливостями, які полегшують стилізацію вебсторінок. Модулі SCSS дозволяють створювати і використовувати локальні області видимості для стилів, що допомагає уникнути конфліктів стилів. SCSS також підтримує змінні, вкладені селектори, міксини і багато інших корисних можливостей для розробки стильового оформлення вашого вебсайту. [13]

2.3.2 Redux Toolkit

Redux Toolkit – це набір інструментів для спрощення розробки додатків на основі Redux, який використовується для управління станом додатків в React аплікаціях. Він містить готові шаблони для створення дій, редюсерів і магазину Redux, що робить процес розробки більш ефективним та зрозумілим. [9]

2.3.3 MockAPI

Mockapi.io – це простий інструмент, який дозволяє легко макетувати API, генерувати власні дані та виконувати над ними операції за допомогою інтерфейсу RESTful. Він призначений для використання як інструмент для створення прототипів/тестування/навчання. [8]

2.3.4 Axios

Axios – це бібліотека для здійснення HTTP-запитів з JavaScript-застосунків. Вона дозволяє легко взаємодіяти з серверами та отримувати або надсилати дані через HTTP. Axios підтримує різні методи запитів і дозволяє обробляти відповіді сервера зручним способом. [12]

2.3.5 Vercel

Vercel — це хмарна платформа для створення, розгортання та масштабування без серверних програм і статичних вебсайтів. У останні роки він став популярним серед розробників завдяки простоті використання, швидкості та здатності обробляти великі обсяги трафіку. [10]

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Створення проекту за допомогою «Create React App»

Create React App – це комфортний осередок для вивчення React, а також це найкращий шлях щоб почати будувати нові односторінкові додатки за допомогою React. Він встановлює осередок для розробки таким чином, щоб ми могли використовувати найновіші можливості JavaScript, робить розробку комфортнішою, а також оптимізує додаток для продакшну [7]. Ми будемо створювати додаток одразу з використанням TypeScript, замість JavaScript. Для цього потрібно ввести наступну команду в git bash (див. рисунок 3.1).

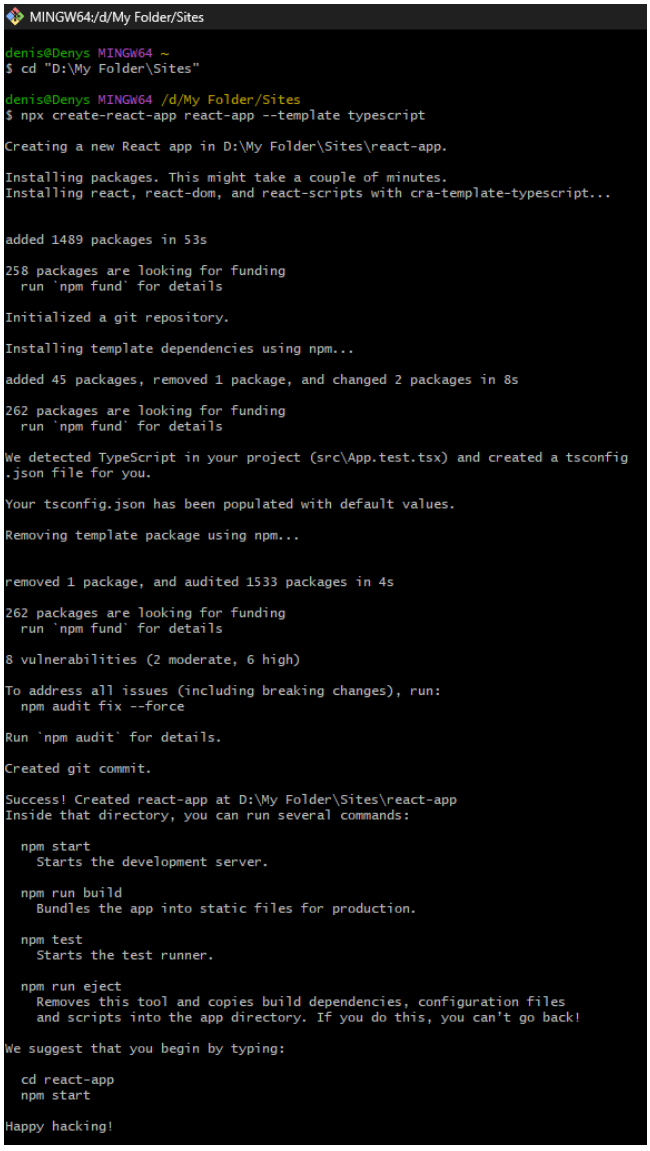

Рисунок 3.1. Команда Create React App

Таким чином було ініціалізовано проєкт і можна переходити до розробки.

3.2 Підготовка бази даних за допомогою сервісу «Mock API»

Перейшовши і авторизувавшись на сайті «Mock API» потрібно першим чином налаштувати сховище даних. Для цього створюємо нове сховище клікнувши на кнопку «New resource» (див. рисунок 3.2)

Рисунок 3.2. Кнопка «New resource»

Наша задача – це створити базу даних для жіночих товарів (див. рисунок 3.3), а також окрему базу даних для категорій (див. рисунок 3.4) по яким в подальшому буде відбуватися зручна фільтрація.

Рисунок 3.3. База даних жіночих товарів

Для збереження зображень жіночого одягу, я використовую безкоштовний хостинг зображень https://uk.imgbb.com/ [11]

RESOURCE DATA - CATEGORIES

Resource data

Edit/replace data for **categories** resource. Data must be an array and a valid JSON.


```
[
  {
    "id": "0",
    "categoryName": "Пальта",
    "categoryImage": "https://i.ibb.co/SnjX4f8/coat.png"
  },
  {
    "id": "1",
    "categoryName": "Плащі",
    "categoryImage": "https://i.ibb.co/D86CkfS/light-blue-cloak.jpg"
  },
  {
    "id": "2",
    "categoryName": "Костюми",
    "categoryImage": "https://i.ibb.co/Jzpj6Mr/green-costume.jpg"
  },
  {
    "id": "3",
    "categoryName": "Куртки",
    "categoryImage": "https://i.ibb.co/wcXMc2b/light-blue-jacket.jpg"
  },
  {
    "id": "4",
    "categoryName": "Шуби",
    "categoryImage": "https://i.ibb.co/g9m9VnF/pink-fur-coat.jpg"
  },
  {
    "id": "5",
    "categoryName": "Парки",
    "categoryImage": "https://i.ibb.co/VqPKPqc/black-parka.jpg"
  }
]
```


Рисунок 3.4. База даних категорій

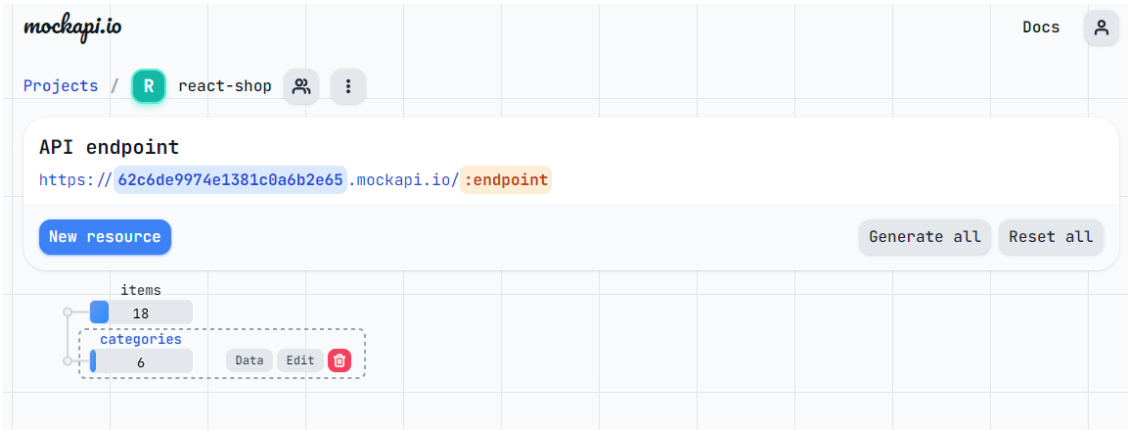

Рисунок3.5. Загальний вигляд структури Mock API

Таким чином, було підготовано базу даних для подальшої розробки нашого вебдодатку. Повний код файлу items.json наведено в **додатку**. Повний код файлу categories.json наведено в **додатку**.

3.3 Redux Toolkit та налаштування глобальних станів і сховищ

У даний час розробка левової частки вебзастосунків, заснованих на фреймворку React, ведеться з використанням бібліотеки Redux. Ця бібліотека є найпопулярнішою реалізацією FLUX-архітектури і, незважаючи на низку очевидних переваг, має дуже суттєві недоліки, такі як:

- складність і “багатослівність” рекомендованих патернів для написання та організації коду, що тягне за собою велику кількість бойлерплейту;
- відсутність вбудованих засобів управління асинхронною поведінкою та побічними ефектами, що призводить до необхідності вибору відповідного інструменту з безлічі аддонів, написаних сторонніми розробниками.

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

- стан каталогу:

Спочатку отримуємо елементи для каталогу з бази даних «Mock API» за допомогою бібліотеки Axios (див. рисунок 3.6):

Рисунок 3.6. Get запис до бази даних для отримання елементів каталогу

Далі створюємо так званий «slice» – функція, котра вирізає дані додатку, та визначає як ці дані будуть змінюватися (оновлюватися) у відповідь на якісь дії (actions) (див. рисунок 3.7)

Рисунок 3.7. Slice управління каталогу

Повний код файлу asyncActions.ts наведено в **додатку**. Повний код файлу catalogSlice.ts наведено в **додатку**

- стан картки товару: тут аналогічно робимо спочатку запит до бази даних для отримання даних за допомогою бібліотеки Axios:

```ts
import { createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";

import { Card, SearchCardParams } from "./types";

export const fetchCardItems = createAsyncThunk<Card, SearchCardParams>(
  'card/fetchCardStatus', async (params) => {

  const {id} = params;

  const { data } = await axios.get(
    `https://62c6de9974e1381c0a6b2e65.mockapi.io/items/${id}`,
  );

  return data;
});
```

Рисунок 3.8. Get запис до бази даних для отримання даних картки

І далі створюємо slice для управління даними (див. рисунок 3.9):

```ts
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
import { fetchCardItems } from './asyncActions';
import { Card, CardSliceState, CardStatus } from './types';

const initialState: CardSliceState = {
  items: {
    id: '',
    newId: 0,
    imageUrl: '',
    title: '',
    price: 0,
    size: [],
    sizes: []
  },
  status: CardStatus.PENDING,
};

const cardSlice = createSlice({
  name: 'card',
  initialState,
  reducers: {
    setItems(state, action: PayloadAction<Card>) {
      state.items = action.payload;
    }
  },
  extraReducers(builder) {
    builder.addCase(fetchCardItems.pending, (state) => {
      state.items = state.items;
      state.status = CardStatus.PENDING;
    });

    builder.addCase(fetchCardItems.fulfilled, (state, action) => {
      state.status = CardStatus.FULFILLED;
      state.items = action.payload;
    });

    builder.addCase(fetchCardItems.rejected, (state) => {
      state.status = CardStatus.REJECTED;
    });
  },
})

export const { setItems } = cardSlice.actions;

export default cardSlice.reducer;
```

Рисунок 3.9. Slice управління картки

Повний код файлу asyncActions.ts наведено в **додатку**. Повний код файлу cardSlice.ts наведено в **додатку**.

- стан кошика: для контролю стану кошика створюємо ще один slice (див. рисунок 3.10)

```
const cartSlice = createSlice({
  name: 'cart',
  initialState,
  reducers: {
    addItems(state, action: PayloadAction<CartItem>) {
      const duplicate = findDuplicates(state.items, action.payload)

      if(duplicate.length > 0) {
        state.items = deleteDuplicates(state.items, action.payload);

        const newItemChecked = {
            ...action.payload,
            newId: duplicate[0].newId,
            count: action.payload.count + duplicate[0].count
        }

        state.errorModal = true;
        state.items.push(newItemChecked);
      } else {
        const newItem = {
          ...action.payload, count: 1,  newId: state.items.length ? state.items[state.items.length - 1].newId + 1 : 1
        }

        state.successModal = true;
        state.items.push(newItem)
      }
      state.totalPrice = calcTotalPrice(state.items);
    },

    plusItems(state, action: PayloadAction<number>) {
      const findItem = state.items.find((obj) => obj.newId === action.payload);

      if (findItem) {
        findItem.count++;
        state.totalPrice = calcTotalPrice(state.items);
      }
    },

    minusItems(state, action: PayloadAction<number>) {
      const findItem = state.items.find((obj) => obj.newId === action.payload);

      if (findItem) {
        findItem.count--;
        state.totalPrice = calcTotalPrice(state.items);
     }
    },

    removeItems(state, action: PayloadAction<number>) {
      state.items = state.items.filter((obj) => obj.newId !== action.payload);
      state.totalPrice = calcTotalPrice(state.items);
    },

    setSuccessModal(state) {
      state.successModal = !state.successModal;
    },

    setErrorModal(state) {
      state.errorModal = !state.errorModal;
    },
  },
});
```

Рисунок 3.10. Slice управління кошиком

Повний код файлу cartSlice.ts наведено в **додатку**

- стан улюбленого: достворюємо slice для керування стану улюбленого (див. рисунок 3.11):

```
const cartSlice = createSlice({
  name: 'favorite',
  initialState,
  reducers: {
    addFavoriteItems(state, action: PayloadAction<FavoriteItem>) {

      const findItem = state.favorites.find((obj) => obj.id === action.payload.id);

      if (findItem) {
        state.errorFavModal = true;
      } else {
        state.favorites.push({
          ...action.payload,
          count: 1,
        });
        state.successFavModal = true;
      }
    },
    removeFavoriteItems(state, action: PayloadAction<string>) {
      state.favorites = state.favorites.filter((obj) => obj.id !== action.payload);
    },
    setSuccessFavModal(state) {
      state.successFavModal = !state.successFavModal;
    },
    setErrorFavModal(state) {
      state.errorFavModal = !state.errorFavModal;
    },
  },
});
```

Рисунок 3.11. Slice управління улюбленим

- стан категорій: додатково створюємо slice для керування стану категорій, попередньо перед цим отримавши дані для категорій з бази даних за допомогою бібліотеки Axios (див. рисунок 3.12, 3.13):

```
TS asyncActions.ts ×
src > redux > category > TS asyncActions.ts > ...
1   import { createAsyncThunk } from "@reduxjs/toolkit";
2   import axios from "axios";
3   import { Category } from "./types";
4
5   export const fetchCategoryItems = createAsyncThunk<Category[]>(
6     'category/fetchCategoryStatus', async () => {
7     const { data } = await axios.get<Category[]>(
8       `https://62c6de9974e1381c0a6b2e65.mockapi.io/categories`,
9     );
10    return data;
11  });
```

Рисунок 3.12. Get запис до бази даних для отримання даних категорії

```
const categorySlice = createSlice({
  name: 'category',
  initialState,
  reducers: {
    setCategories(state, action: PayloadAction<Category[]>) {
      state.categories = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder.addCase(fetchCategoryItems.pending, (state) => {
      state.status = Status.PENDING;
      state.categories = [];
    });

    builder.addCase(fetchCategoryItems.fulfilled, (state, action) => {
      state.status = Status.FULFILLED;
      state.categories = action.payload;
    });

    builder.addCase(fetchCategoryItems.rejected, (state) => {
      state.status = Status.REJECTED;
      state.categories = [];
    });
  },
});
```

Рисунок 3.13. Slice управління категоріями

Повний код файлу categorySlice.ts наведено в **додатку**

- стан фільтру для контролю стану фільтру (див. рисунок 3.14)

```
const filterSlice = createSlice({
  name: 'filters',
  initialState,
  reducers: {
    setCategoryId(state, action: PayloadAction<number>) {
      state.categoryId = action.payload;
    },
    setSearchValue(state, action: PayloadAction<string>) {
      state.searchValue = action.payload;
    },
    setSort(state, action: PayloadAction<SortType>) {
      state.sort = action.payload;
    },
    setCurrentPage(state, action: PayloadAction<number>) {
      state.currentPage = action.payload;
    },
  }
})
```

Рисунок 3.14. Slice управління фільтром

Повний код файлу filterSlice.ts наведено в **додатку**

Також важливим налаштуванням є налаштування сховища наших слайсів – це так званий store (див. рисунок 3.15):

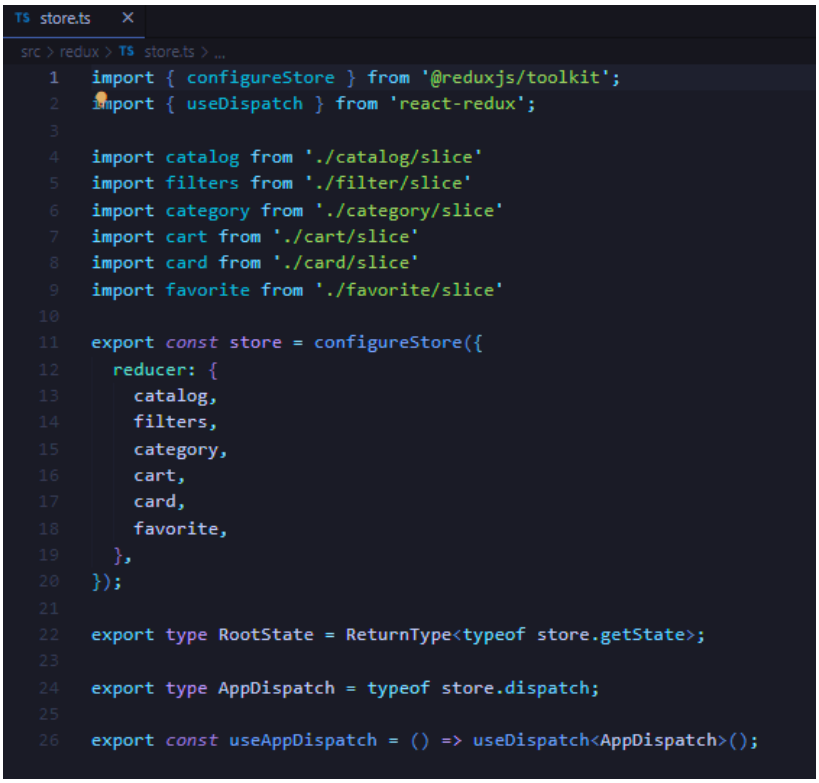

Рисунок 3.15. Redux Toolkit store

Можемо побачити, що в цьому store знаходяться усі наші попередньо налаштовані slice. Ці слайси сконфігуровані за допомогою функції «configureStore». І для того, щоб вебдодаток міг користуватися цим сховищем, нам потрібно заасайнити до кореневого компоненту index.tsx доступ до сховища, використовуючи спеціальний компонент Provider та його аргумент store (див. рисунок 3.16):

```tsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import { store } from './redux/store';

import './scss/app.scss';

const root = ReactDOM.createRoot(document.getElementById('root') as HTMLElement);

root.render(
  <Provider store={store}>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </Provider>,
);
```

Рисунок 3.16. Provider для забезпечення зв’язку додатку зі сховищем слайсів

Повний код файлу store.ts наведено в **додатку**. Повний код файлу index.tsx наведено в **додатку**.

3.4 Розробка компонентів та інтеграція їх на сторінках

3.4.1 Компоненти Header та Fullscreen

У нашому випадку компонент Header інтегровано в Fullscreen сайту. Таким чином вигляд зовнішній вигляд одразу приваблює користувача гарними зображенням одягу і також має гарну навігацію в шапці (див. рисунок 3.17):

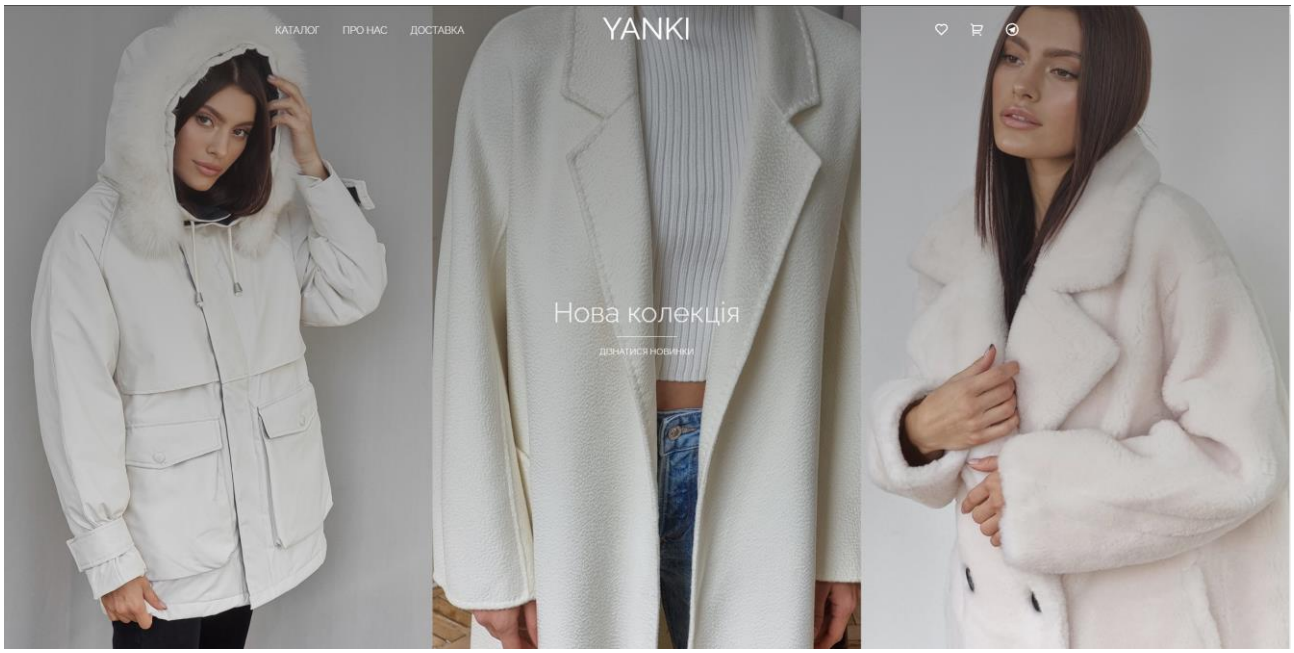

Рисунок 3.17. Вигляд Header та Fullscreen

```
  return (
    <header className={screenWidth > 992 || !isMobile ? 'header' : 'header_m'}>
      <div className="container">
        <button
          className="mobile-menu-icon mobile-menu-icon_new"
          aria-label="mobileMenu"
          onClick={() => setIsMobile(!isMobile)}>
          {isMobile ? <i className="fas fa-times"></i> : <i className="fas fa-bars"></i>}
        </button>
        <nav className="header__menu menu">
          <ul className={screenWidth > 992 || !isMobile ? 'menu__list' : 'menu__list_mobile'}>
            <li className="menu__link menu__link_color">
              <Link to="/catalog">КАТАЛОГ</Link>
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_color">
              <Link to="/about">ПРО НАС</Link>
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_color">
              <Link to="/delivery">ДОСТАВКА</Link>
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_m">
              <Link to="/favorites">···
              </Link>
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_m">
              <Link to="/cart">···
              </Link>
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_m">
              <Link to="/contacts">···
              </Link>
            </li>
            <div className="hr"></div>
          </ul>
          <ul className="menu__list">
            <li className="menu__link menu__link_color">
              <Link to="/favorites" aria-label="favorites">···
              </Link>
            </li>
            <li className="menu__link menu__link_color">
              <Link to="/cart" aria-label="cart">···
              </Link>
            </li>
            <li className="menu__link menu__link_color">
              <Link to="/contacts" aria-label="contacts">···
              </Link>
            </li>
          </ul>
        </nav>
        <div className="header__logo_yanki">
          <Link to={'/'}>
            <span>YANKI</span>
          </Link>
        </div>
      </div>
    </header>
  );
};
```

Рисунок 3.18. Код розмітки Header

```
const Fullscreen: React.FC = () => {
  return (
    <div className="fullscreen">
      <MainHeader />
      <div className="fullscreen__text text">
        <div className="text">
          <h1 className="text__title">Нова колекція</h1>
          <div className="text__hr"></div>
          <Link to="/catalog">
            <h2 className="text__subtitle">Дізнатися новинки</h2>
          </Link>
        </div>
      </div>
    </div>
  );
};
```

Рисунок 3.19. Код розмітки Fullscreen

Повний код файлу Header.tsx наведено в **додатку**

Повний код файлу Fullscreen.tsx наведено в **додатку**

3.4.2 Компоненти HomeCategories та Novelties

Ці компоненти відповідають за зручний перехід у відповідну категорії та повідомлення про новинки. Перший зроблений спеціально у вигляді каруселі, котру можна прогортувати, а другий – це чисто підписка на новинки:

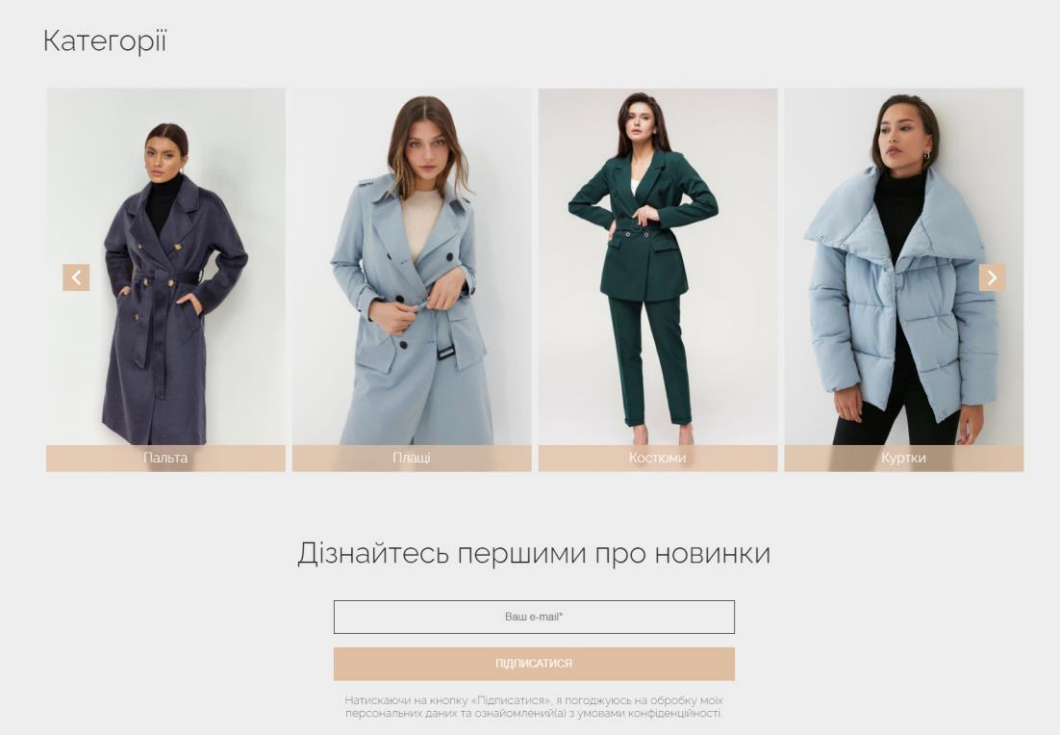

Рисунок 3.20 Вигляд HomeCategories та Novelties

```tsx
const Categories: React.FC = () => {
  const dispatch = useAppDispatch();

  const getCategories = async () => {
    dispatch(fetchCategoryItems());
  };

  React.useEffect(() => {
    getCategories();
  }, []);

  return (
    <div className="categories">
      <div className="container">
        <h3 className="categories__title">Категорії</h3>
        <Slider />
      </div>
    </div>
  );
};
```

Рисунок 3.21. Код розмітки HomeCategories

```tsx
const Novelties: React.FC = () => {
  return (
    <div className="novelties">
      <div className="container">
        <div className="novelties__title">Дізнайтесь першими про новинки</div>
        <form action="signin" autoComplete="off" method="post" className="novelties__form">
          <input
            type="text"
            name="email"
            placeholder="Ваш e-mail*"
            pattern='^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$'
            title="Електронна адреса має бути формату example@gmail.com"
            required
          />
          <button className="novelties__button">
            <span>ПІДПИСАТИСЯ</span>
          </button>
          <p className="novelties__subtitle">
            Натискаючи на кнопку «Підписатися», я погоджуюсь на обробку моїх персональних даних та
            ознайомлений(в) з умовами конфіденційності.
          </p>
        </form>
      </div>
    </div>
  );
};
```

Рисунок 3.22. Код розмітки Novelties

Повний код файлу HomeCategories.tsx наведено в **додатку**

Повний код файлу Novelties.tsx наведено в додатку

3.4.3 Компонент Footer

Футер використовується у нас для розміщення додаткової інформації (див. рисунок 3.23):

Рисунок 3.23. Вигляд Footer

```
return (
  <footer className="footer">
    <div className="container">
      <div className="footer__row">
        <div className="footer__column">
          <div className="footer__title" onClick={() => setFirstUl(!firstUl)}>
            <div className="footer__label">КОМПАНІЯ</div>
            <div className="footer__arrow">
              <svg…
              </svg>
            </div>
          </div>
          <ul
            className={screenWidth > 992 || !firstUl ? 'footer__text open-text' : 'closed-text'}>
            <li>
              <Link to="/about">Про нас</Link>
            </li>
            <li>
              <Link to="/contacts">Контакти</Link>
            </li>
          </ul>
          <div className="footer__hr"></div>
        </div>
        <div className="footer__column">
          <div className="footer__title" onClick={() => setSecUl(!secUl)}>
            <div className="footer__label">КОРИСНЕ</div>
            <div className="footer__arrow">
              <svg…
              </svg>
            </div>
          </div>
          <ul className={screenWidth > 992 || !secUl ? 'footer__text open-text' : 'closed-text'}>
            <li>
              <Link to="/delivery">Оплата та доставка</Link>
            </li>
            <li>
              <Link to="/refund">Умови повернення</Link>
            </li>
          </ul>
          <div className="footer__hr"></div>
        </div>
        <div className="footer__column">
          <div className="footer__title" onClick={() => setThirdUl(!thirdUl)}>
            <div className="footer__label">ПОКУПЦЮ</div>
            <div className="footer__arrow">
              <svg…
              </svg>
            </div>
          </div>
          <ul
            className={screenWidth > 992 || !thirdUl ? 'footer__text open-text' : 'closed-text'}>
            <li>
              <Link to="/cart">Кошик</Link>
            </li>
            <li>
              <Link to="/favorites">Обране</Link>
            </li>
          </ul>
          <div className="footer__hr"></div>
        </div>
        <div className="footer__column">
          <div className="footer__title">КОНТАКТИ</div>
          <ul className="footer__text">
            <li>
              <a target="_blank" aria-label="instagram" href="https://www.instagram.com">
                <svg…
                </svg>
              </a>
              <a target="_blank" aria-label="telegram" href="https://telegram.org/">
                <svg…
                </svg>
              </a>
            </li>
            <li>+38(073) 096 36 44</li>
            <li>
              <a target="_blank" href="mailto:info@yanki.com">
                info@yanki.com
              </a>
            </li>
          </ul>
        </div>
      </div>
      <div className="footer__bottom-text">© 2022 Yanki. All rights reserved</div>
    </div>
  </footer>
);
};
```

Рисунок 3.24. Код розмітки Footer

Повний код файлу Footer.tsx наведено в **додатку**

3.4.4 Сторінка каталогу

Перейшовши на сторінку каталогу ми одразу бачимо перед собою, фільтр по категоріям, зручний пошук, можливість сортування за критерієм та пагінацію сторінок. При цьому також можемо помітити, що Header та Footer зберігаються при переході на сторінку каталогу, що забезпечує найкращих User Experience. Варто зауважити, що цей весь функціонал розбито на маленькі компоненти для більш зручної розробки. У результаті маємо привабливу сторінку каталогу з усім потрібним функціоналом користувачу (див. рисунок 3.25):

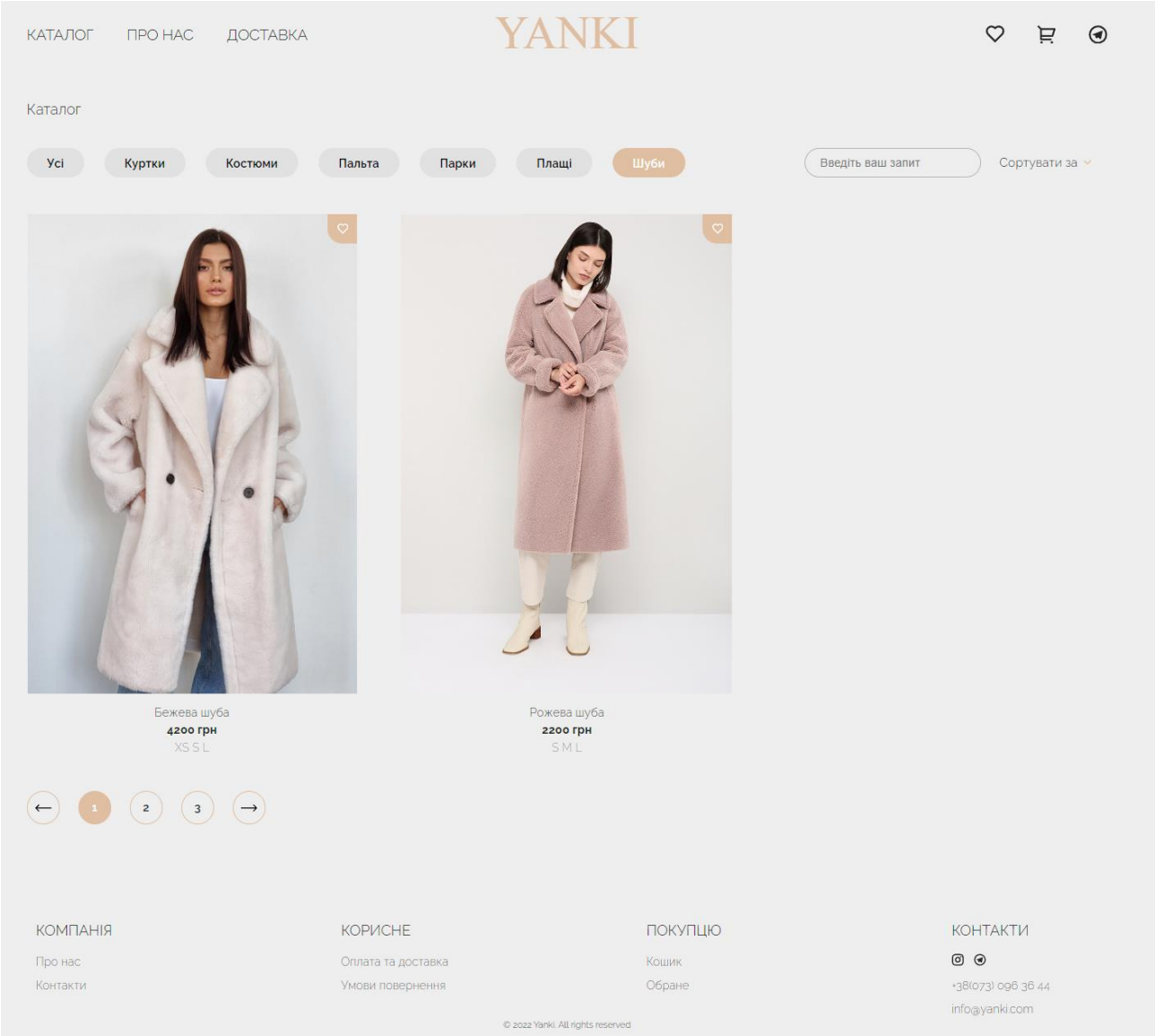

Рисунок 3.25. Вигляд сторінки каталогу

```
const categories = ['Усі', 'Куртки', 'Костюми', 'Пальта', 'Парки', 'Плащі', 'Шуби'];

const CatalogCategories: React.FC<CatalogCategoriesProps> = ({
  category,
  onClickChangeCategory,
}) => {
  return (
    <div className="catalog__categories">
      <ul className="catalog__category">
        {categories.map((categoryName, index) => (
          <li
            key={index}
            onClick={() => onClickChangeCategory(index)}
            className={category === index ? 'active' : ''}>
            {categoryName}
          </li>
        ))}
      </ul>
    </div>
  );
};
```

Рисунок 3.26. Код розмітки фільтру по категоріям

```
<div className="catalog__filter-item">
  <div className="catalog__search">
    <input
      value={value}
      onChange={onChangeInput}
      type="text"
      placeholder="Введіть ваш запит"
    />
  </div>
  <Sort value={sort} />
</div>
</div>
```

Рисунок 3.27. Код розмітки пошуку

```
return (
  <>
    <div ref={sortRef} onClick={() => setVisible(!visible)} className="catalog__filter">
      Сортувати за
      <svg
        className={visible ? 'svg-rotate' : ''}
        width="10"
        height="7"
        viewBox="0 0 10 7"
        fill="none"
        xmlns="http://www.w3.org/2000/svg">
        <path
          d="M5 3.88906L8.88906 -4.85606e-08L10 1.11094L5 6.11093L-4.85606e-08 1.11094L1.11094 -3.88553e-07L5 3.88906Z"
          fill="#E0BEA2"
        />
      </svg>
      {visible && (
        <div className="catalog__popup popup">
          <ul className="popup__items">
            {list.map((obj, i) => (
              <li
                className={value.sortProperty === obj.sortProperty ? 'active' : ''}
                onClick={() => clickOnPopUpItem(obj)}
                key={i}>
                {obj.name}
              </li>
            ))}
          </ul>
        </div>
      )}
    </div>
  </>
);
};
```

Рисунок 3.28. Код розмітки сортування за критеріями

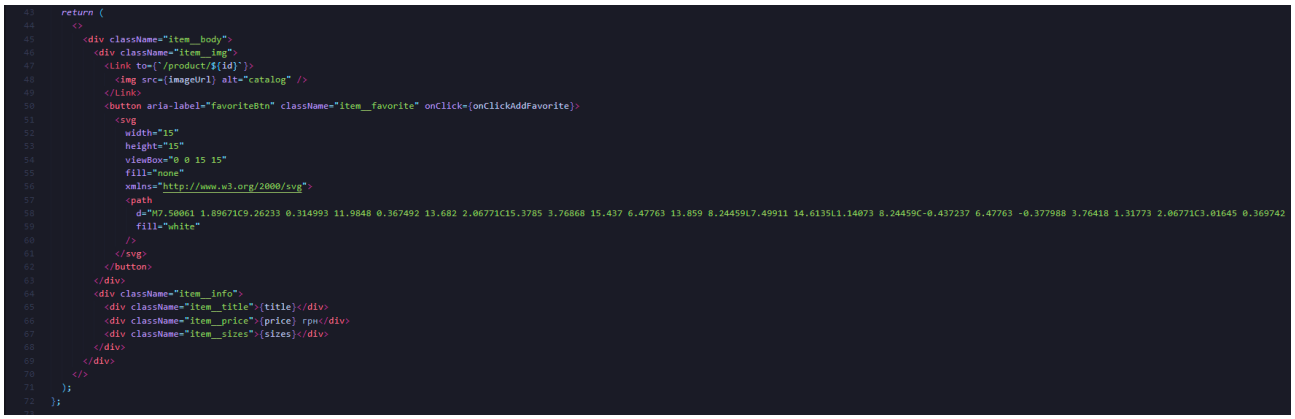

Рисунок 3.29. Код розмітки елементів каталогу

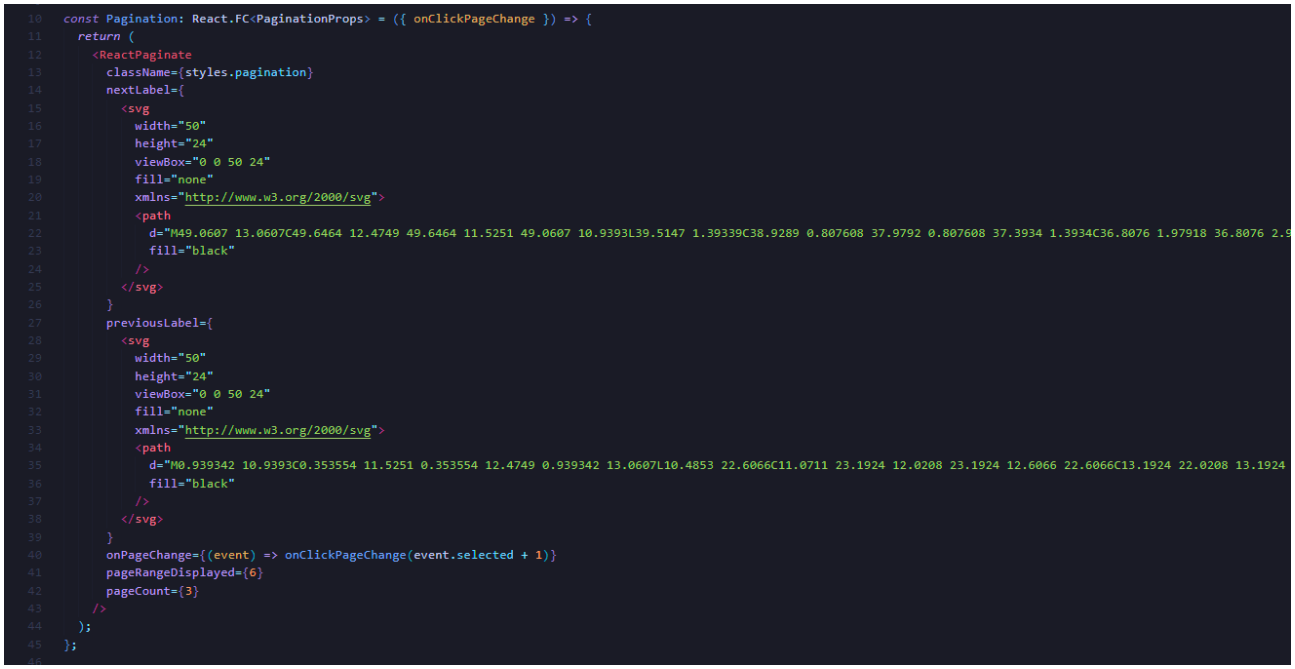

Рисунок 3.30. Код розмітки пагінації

Повний код файлу CatalogCategories.tsx наведено в **додатку**

Повний код файлу Sort.tsx наведено в **додатку**

Повний код файлу CatalogBlock.tsx наведено в **додатку**

Повний код файлу Pagination.tsx наведено в **додатку**

3.4.4 Сторінка товару

При кліку на картинку товару на сторінці каталогу, користувачу відкривається сторінка з конкретним обраним товаром, де він має можливість вибору потрібного розміру і додавання в кошик чи обране (див. рисунок 3.31). Також бачимо, що Header та Footer залишаються незмінними:

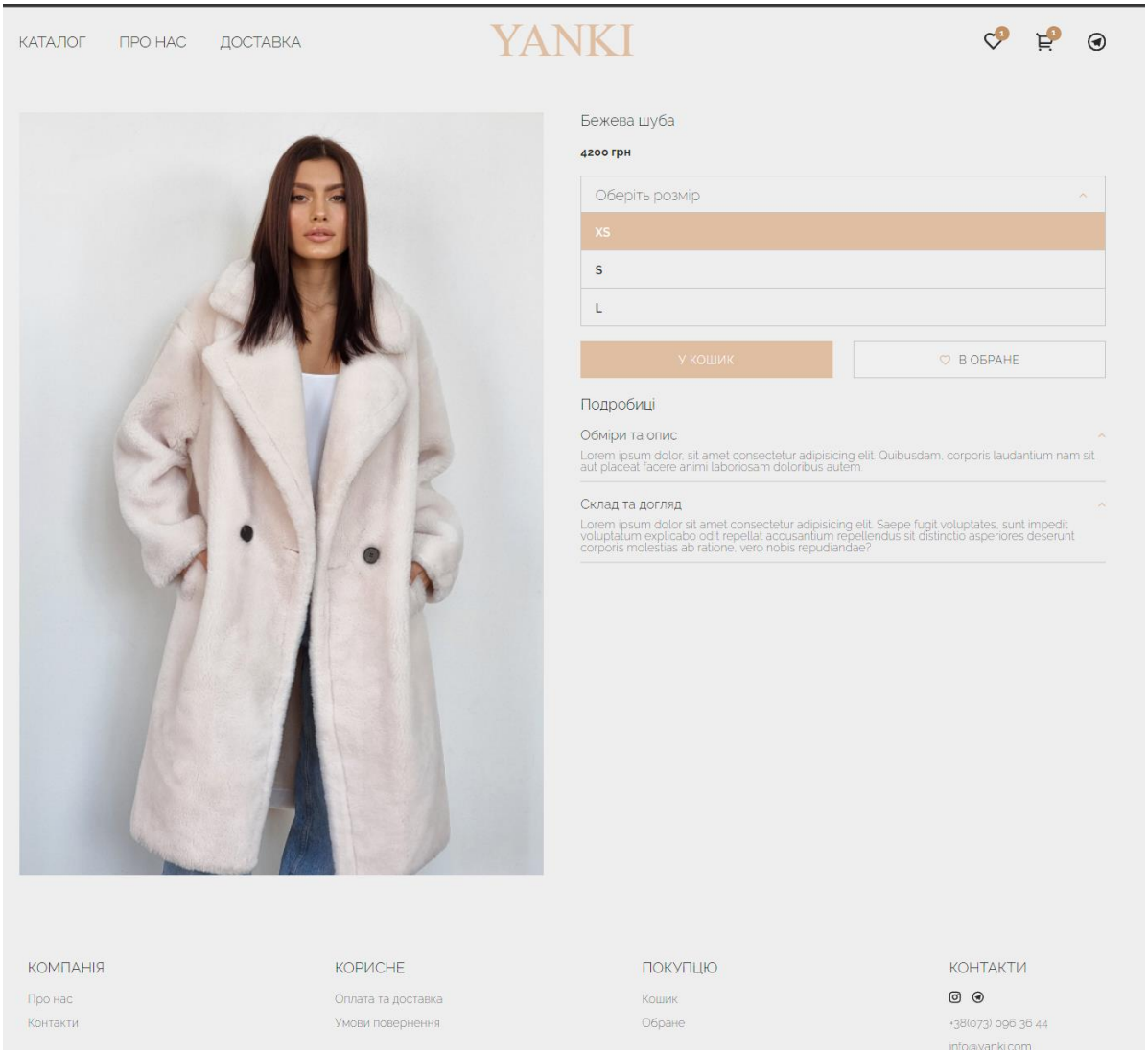

Рисунок 3.31. Вигляд сторінки товару

```
34    return (
35      <>
36        <Header />
37        <section className="card">
38          <div className="container">
39            {status === 'rejected' ? (
40              <div className="card__error-info">
41                <h2>Виникла помилка 😳</h2>
42                <p>На жаль, дана продукція відсутня на нашому сайті!</p>
43              </div>
44            ) : (
45              <>{status === 'pending' ? gif : cardInfo}</>
46            )}
47          </div>
48        </section>
49        <Footer />
50      </>
51    );
52  };
53
```

Рисунок 3.32. Код розмітки товару

Якщо, наприклад, товару не було знайдено, то користувач побачить

наступне повідомлення (див. рисунок 3.33):

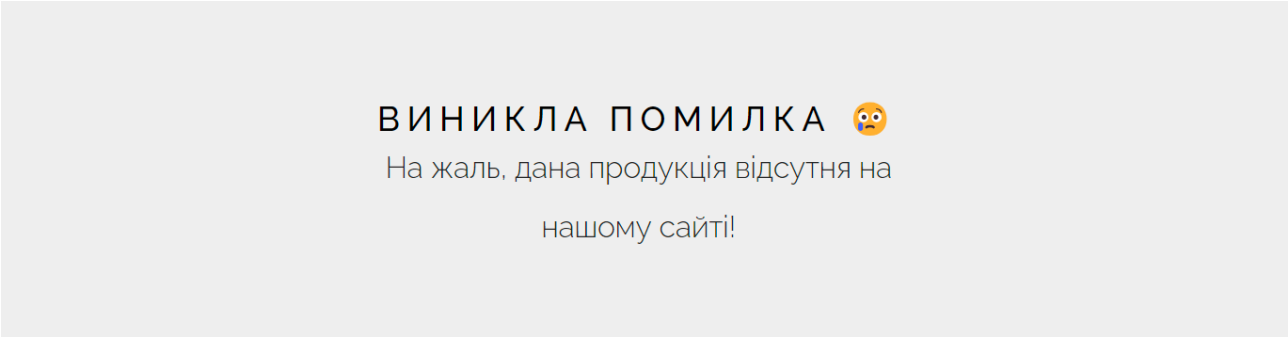

Рисунок 3.33. Вигляд помилки - відсутностні продукції на сайті

Повний код файлу ProductCard.tsx наведено в **додатку**

3.4.4 Сторінка обраного (favorites)

Якщо користувач, додав товар в обране зі сторінки товару, чи з самого каталогу клікнувши на серденько в правому верхньому куту (див. рисунок 3.34):

Рисунок 3.34. Вигляд кнопки «Обране» на каталозі товарів

То йому буде запропоновано перейти до обраних товарів чи продовжити покупки (див. рисунок 3.35):

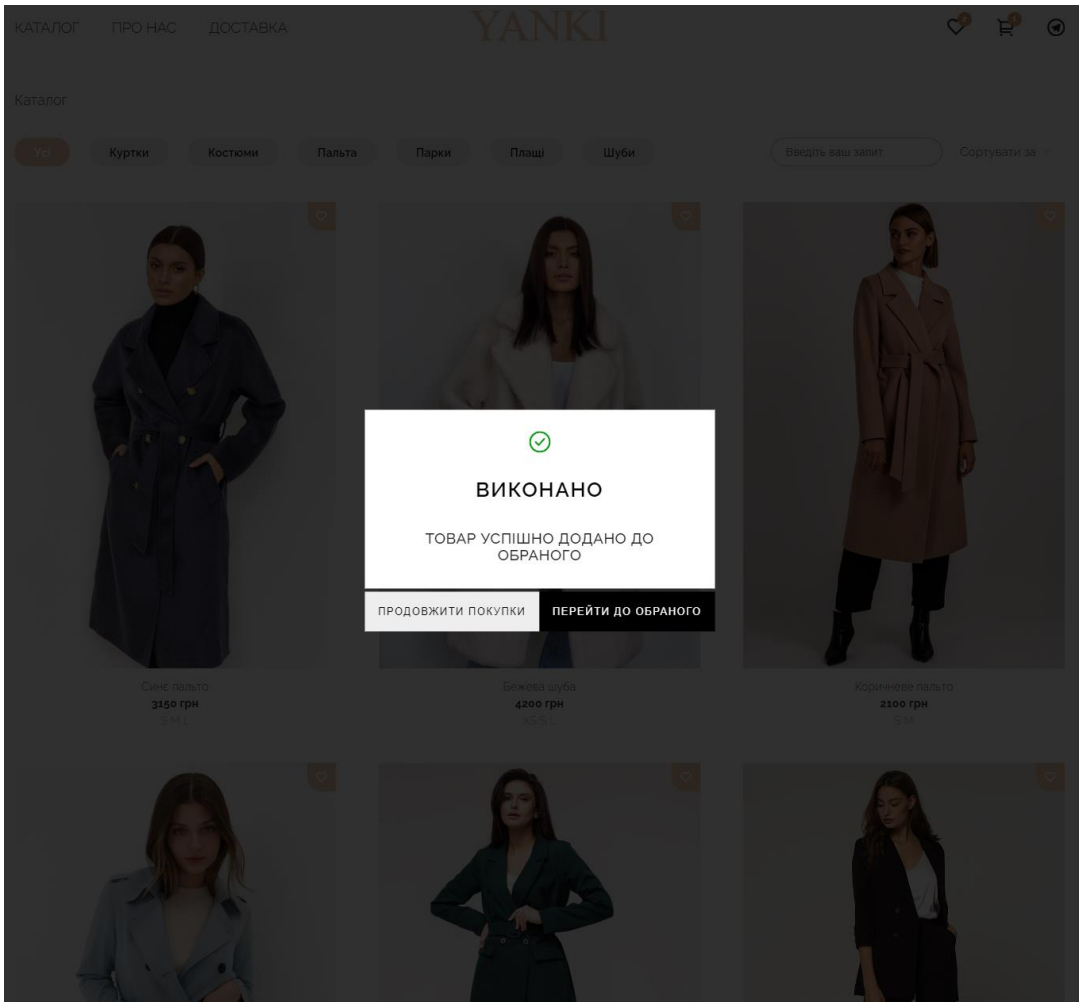

Рисунок 3.35. Вигляд модального вікна для обраного товару

Перейшовши до обраного товару, користувач може побачити перелік доданих товарів, видалити непотрібний йому товар та перейти на картку товару (див. рисунок 3.36)

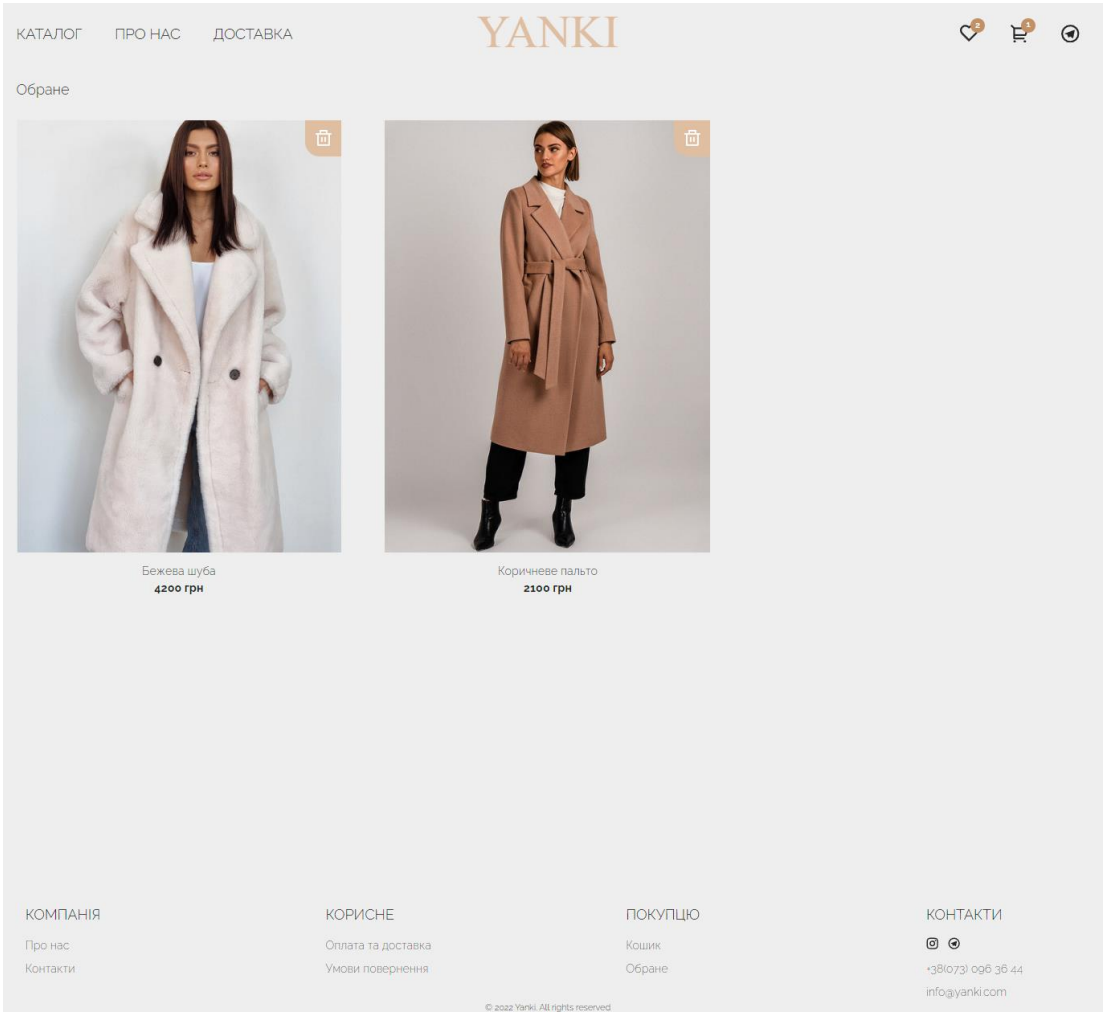

Рисунок 3.36. Вигляд сторінки обраного товару

```
return (
  <div className="favorites__column">
    <div className="favorites__img">
      <Link to={`/product/${id}`} aria-label="favoritesImg">
        <img src={imageUrl} alt="" />
      </Link>
      <button aria-label="favoritesIcon" className="favorites__icon" onClick={onClickRemove}>
        <svg
          width="22"
          height="22"
          viewBox="0 0 25 25"
          fill="none"
          xmlns="http://www.w3.org/2000/svg">
          <path
            d="M18.75 5H25V7.5H22.5V23.75C22.5 24.0815 22.3683 24.3995 22.1339 24.6339C21.8995
            fill="#fff"
          />
        </svg>
      </button>
    </div>
    <div className="favorites__info">
      <div className="favorites__label">{title}</div>
      <div className="favorites__price">{price} грн</div>
      <div className="favorites__sizes">{sizes}</div>
    </div>
  </div>
);
};
```

Рисунок 3.37. Код розмітки Favorites

Якщо ж, обраний список пустий, то користувач побачить наступне повідомлення (див. рисунок 3.38):

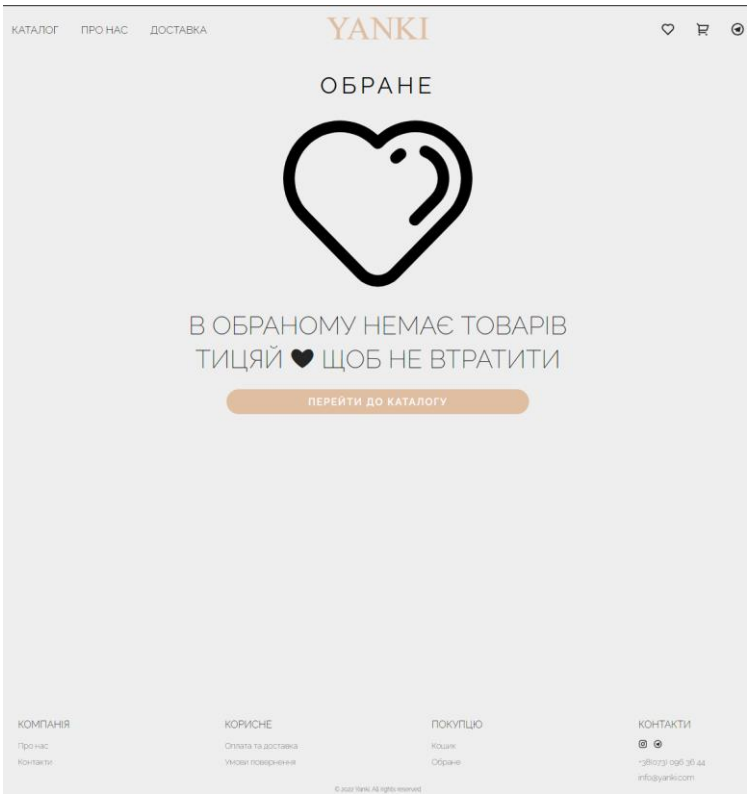

Рисунок 3.38. Вигляд пустого списку обраних товарів

Повний код файлу Favorites.tsx наведено в **додатку**

Повний код файлу FavoritesEmpty.tsx наведено в **додатку**

3.4.5 Сторінка кошику

Аналогічно з Favorites, користувач може додати будь який товар собі до кошику на картці товару, після чого його буде повідомлено про додавання обраного товару до кошику у модальному вікні (див. рисунок 3.39)

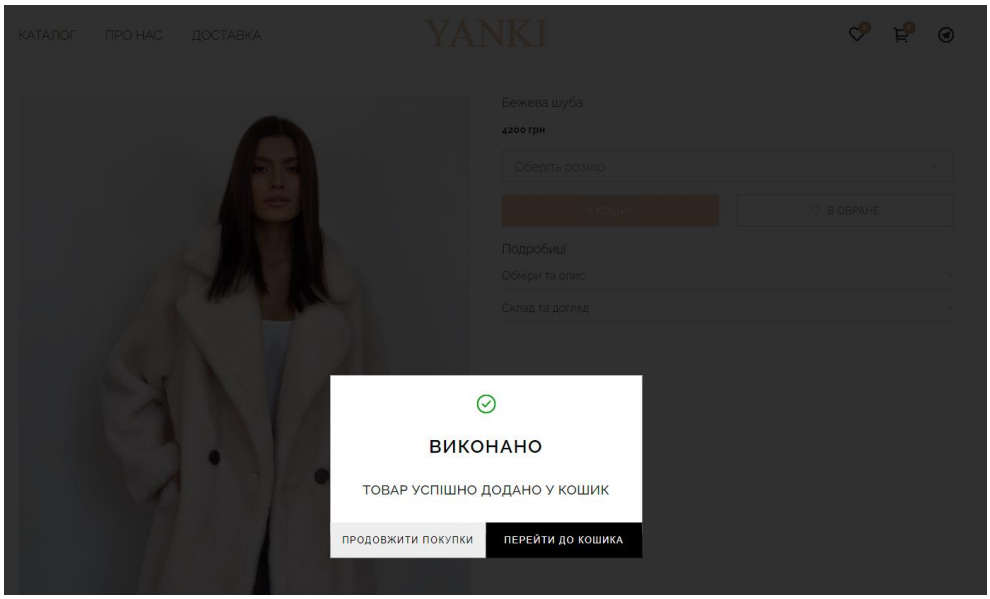

Рисунок 3.39. Вигляд модального вікна для доданого товару в кошик

З цього модального вікна, можна перейти до кошика (див. рисунок 3.40):

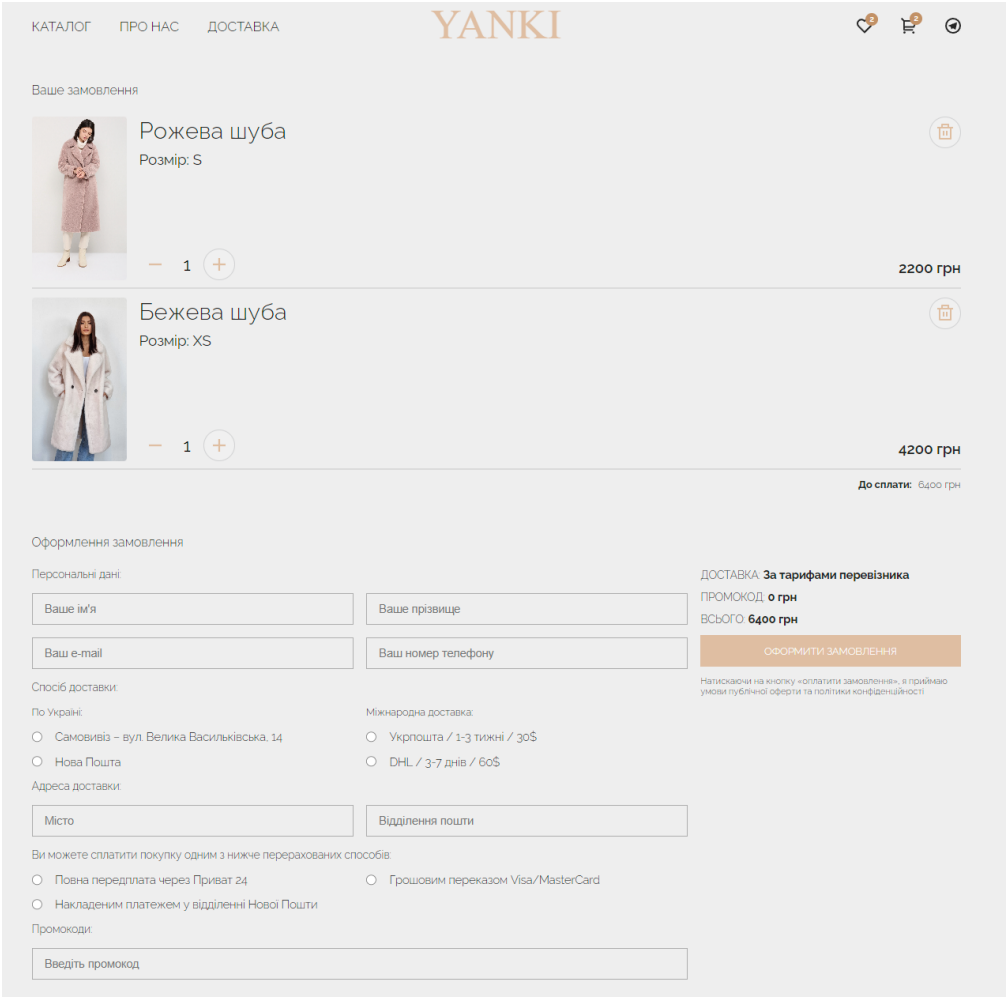

Рисунок 3.40. Вигляд сторінки кошика

Саме на цій сторінці, користувач може: додати кількість товару, видалити товар і також оформити замовлення.

```jsx
return (
  <>
    <div className="cart__order">
      <div className="cart__column">
        <div className="cart__item">
          <div className="cart__img">
            <img src={imageUrl} alt="" />
          </div>
          <div className="cart__info">
            <Link to={`/product/${id}`} className="cart__name">
              {title}
            </Link>
            <div className="cart__size">Розмір: {size}</div>
            <div className="cart__amount">
              <button
                aria-label="minusBtn"
                disabled={count === 1}
                className="cart__minus"
                onClick={onClickMinus}>
                <svg…
                </svg>
              </button>
              <div className="cart__number">{count}</div>
              <button aria-label="plusBtn" className="cart__plus" onClick={onClickPlus}>
                <svg
                  width="20"
                  height="20"
                  viewBox="0 0 10 10"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg">
                  <path
                    d="M4.28571 4.28571V0H5.71429V4.28571H10V5.71429H5.71429V10H4.28571V5.71429H0V4.28571H4.28571Z"
                    fill="#E0BEA2"></path>
                </svg>
              </button>
            </div>
          </div>
        </div>
      </div>
      <div className="cart__column">
        <div className="cart__block">
          <button aria-label="trashBtn" className="cart__trashbin" onClick={onClickRemove}>
            <svg…
            </svg>
          </button>
          <div className="cart__price">{price * count} грн</div>
        </div>
      </div>
    </div>
    <div className="cart__hr"></div>
  </>
);
```

Рисунок 3.41 Код розмітки CartItem

Якщо ж, кошик пустий, то користувач отримає наступне повідомлення (див. рисунок 3.42)

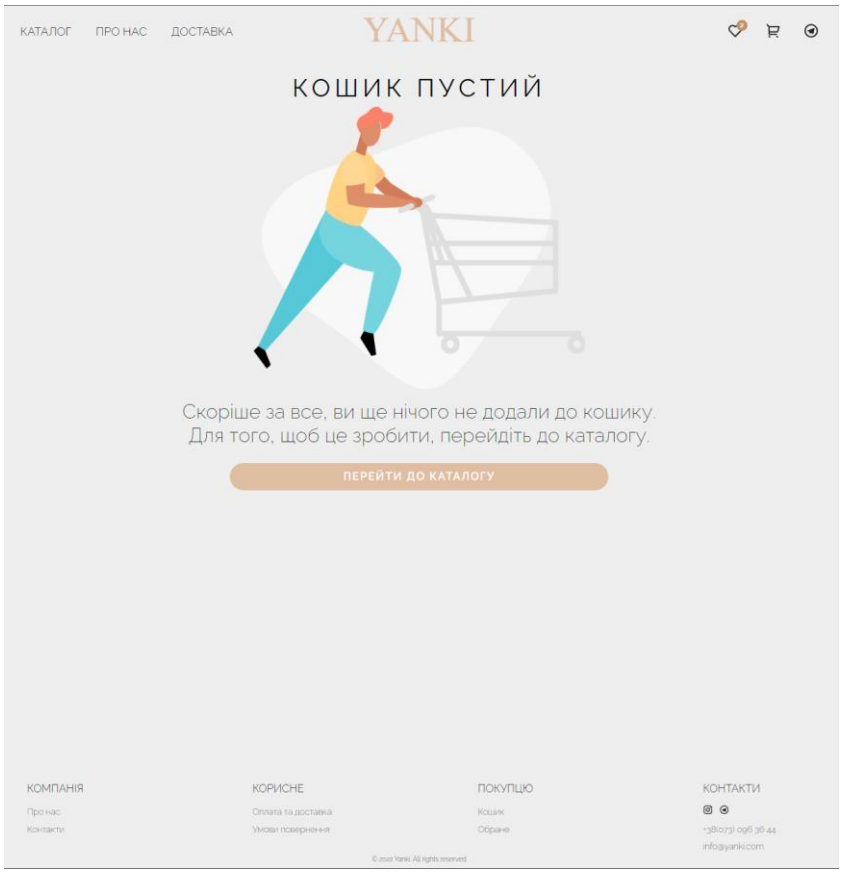

Рисунок 3.42. Вигляд пустого кошика

Повний код файлу CartItem.tsx наведено в **додатку**

Повний код файлу CartEmpty.tsx наведено в **додатку**

3.5 Реалізація адаптивної верстки

Спираючись на світові дані по найпопулярнішим роздільним здатностям серед користувачів смартфоні, планшетів та комп’ютерів [17], під час забезпечення адаптивної верстки було реалізовано підтримку для таких роздільних здатностей екрану:

1. 375 пікселів
2. 425 пікселів
3. 768 пікселів
4. 990 пікселів
5. 1130 пікселів

3.5.1 Домашня сторінка

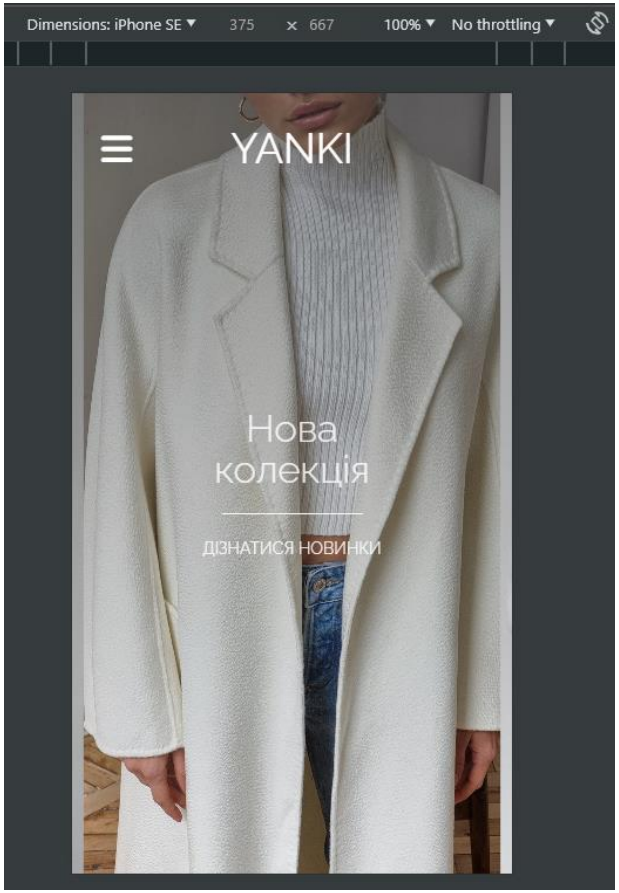

Рисунок 3.43. Домашня сторінка при роздільній здатності екрану 375 пікселів

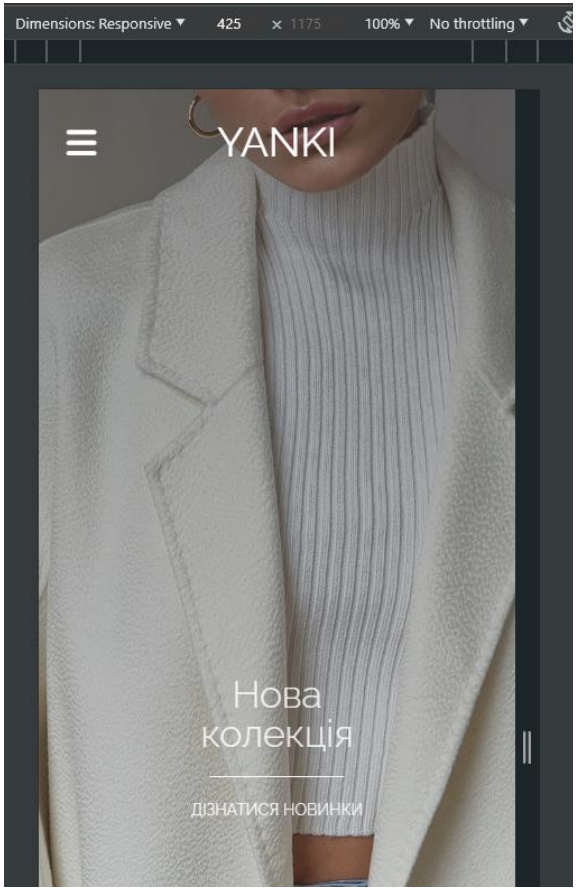

Рисунок 3.44. Домашня сторінка при роздільній здатності екрану 425 пікселів

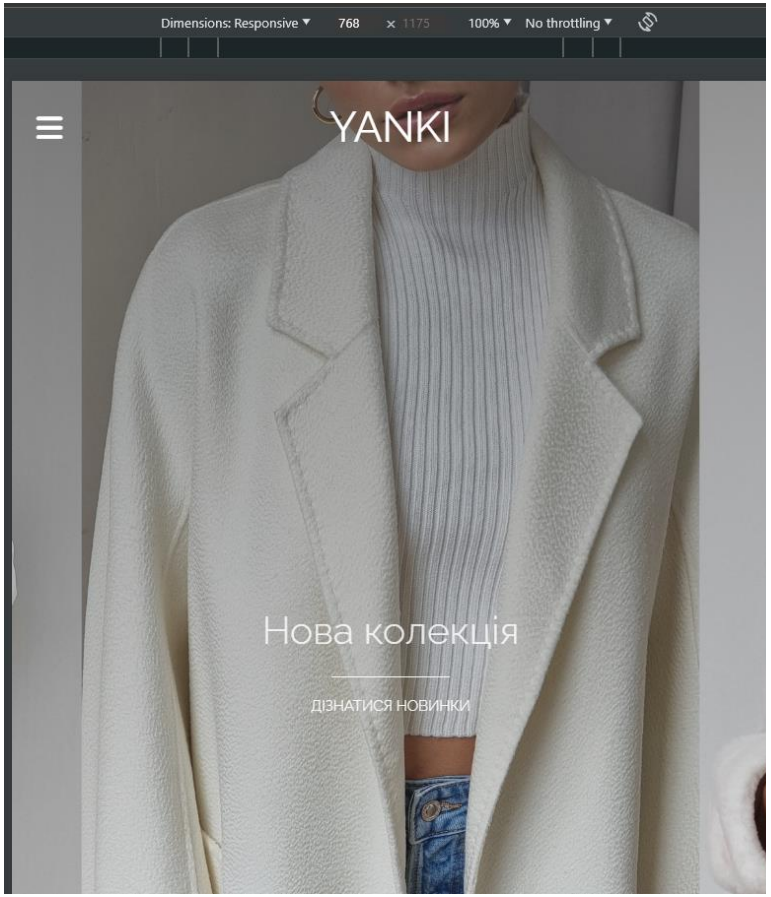

Рисунок 3.45. Домашня сторінка при роздільній здатності екрану 768 пікселів

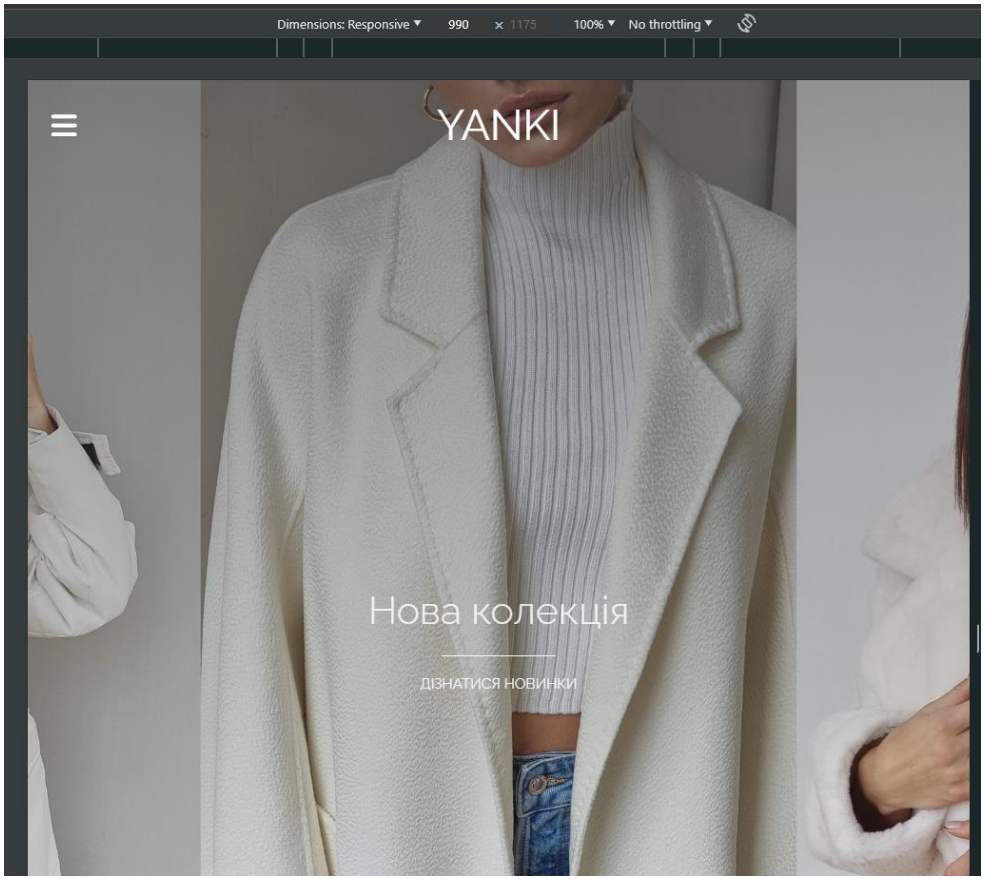

Рисунок 3.46. Домашня сторінка при роздільній здатності екрану 990 пікселів

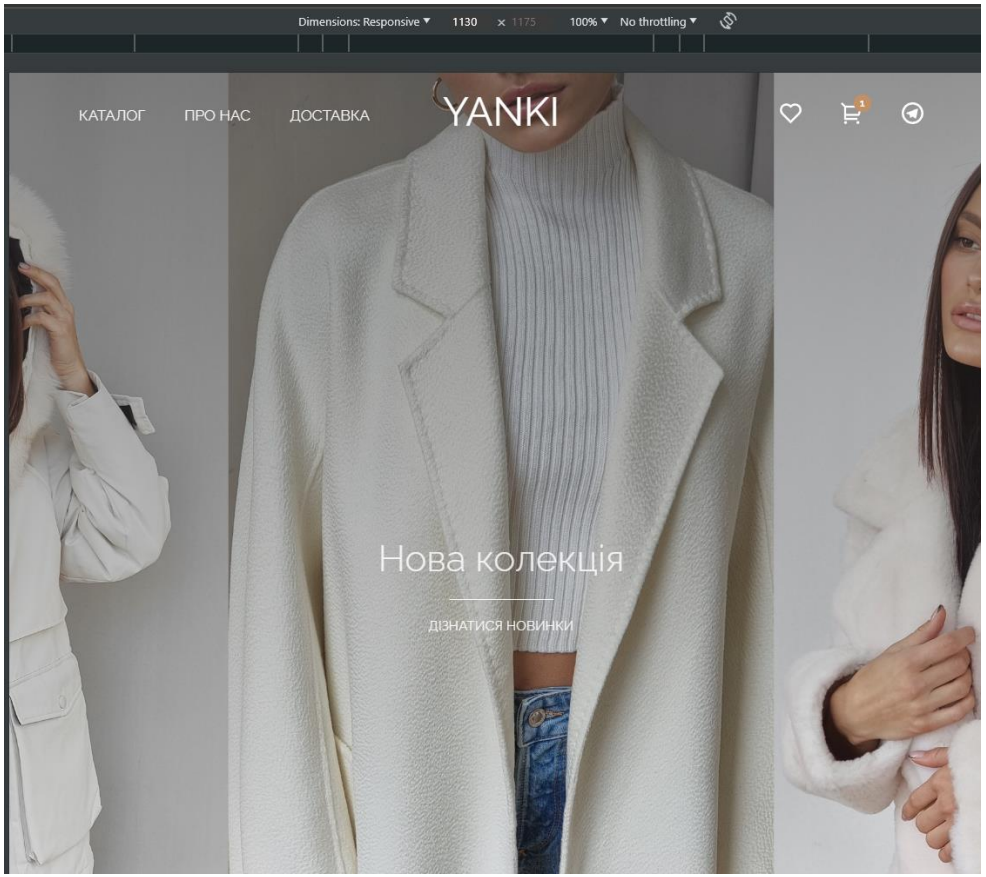

Рисунок 3.47. Домашня сторінка при роздільній здатності екрану 1130 пікселів

Повний код адаптивної стилізації наведено в git-репозиторії [14].

3.5.2 Сторінка каталогу

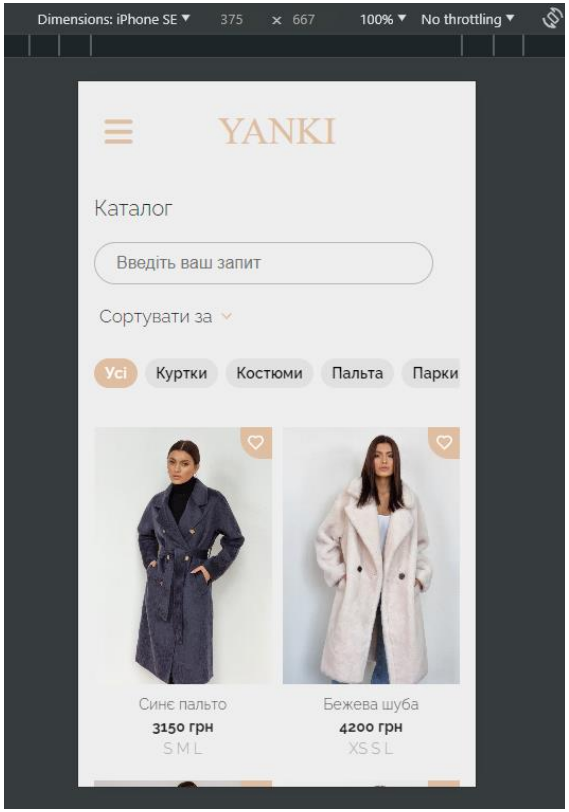

Рисунок 3.48. Сторінка каталогу при роздільній здатності екрану 375 пікселів

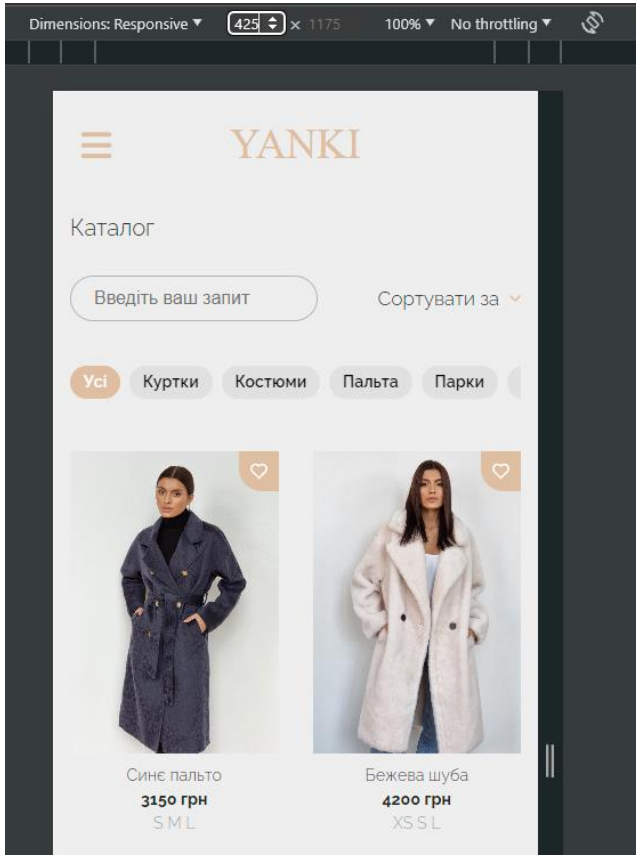

Рисунок 3.49. Сторінка каталогу при роздільній здатності екрану 425 пікселів

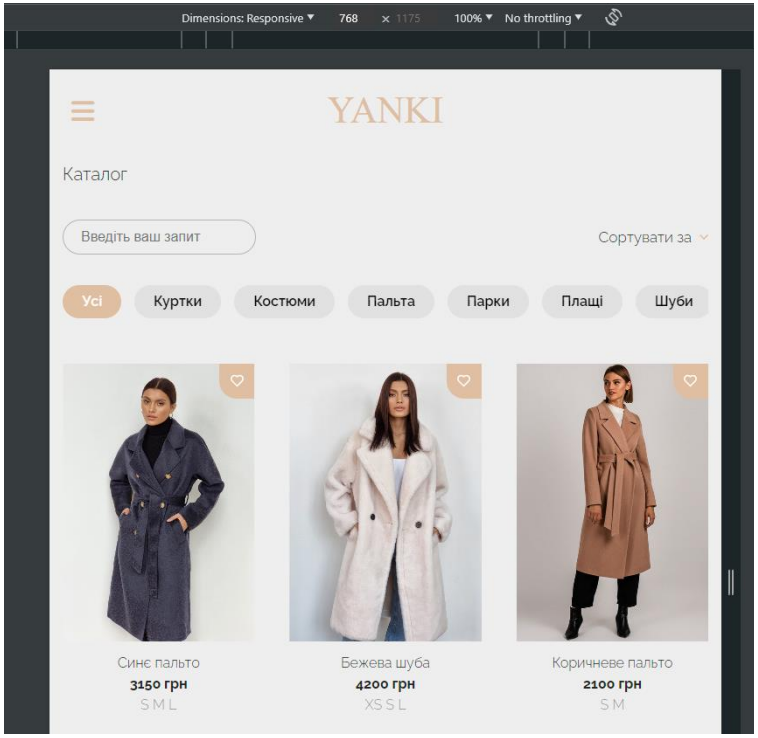

Рисунок 3.50. Сторінка каталогу при роздільній здатності екрану 768 пікселів

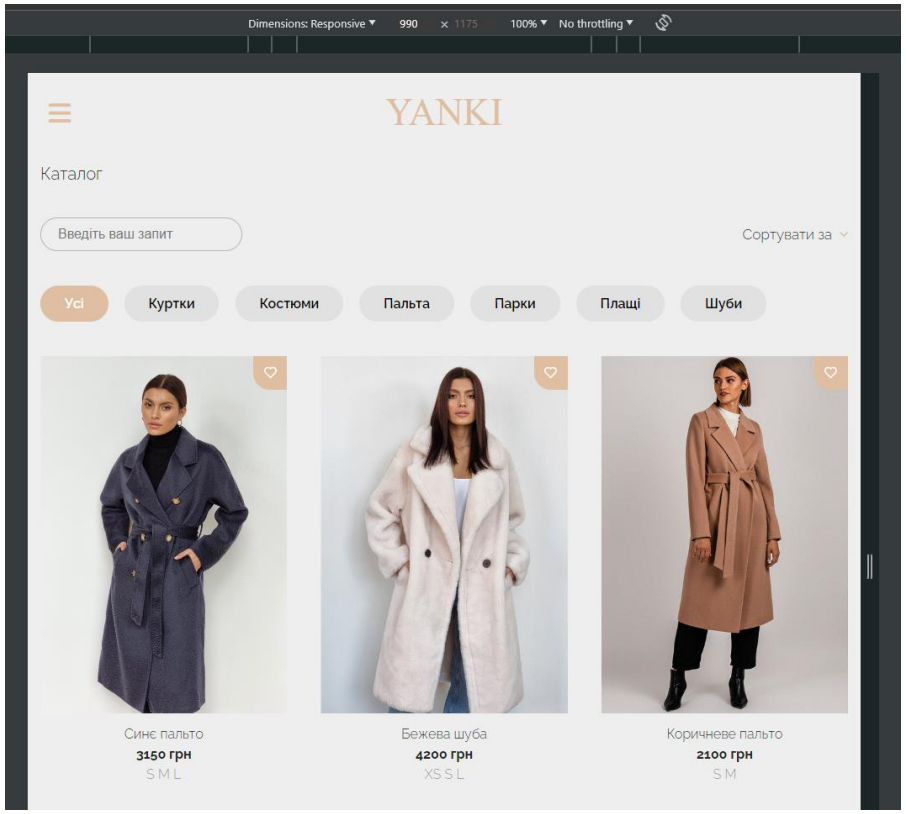

Рисунок 3.51 Сторінка каталогу при роздільній здатності екрану 990 пікселів

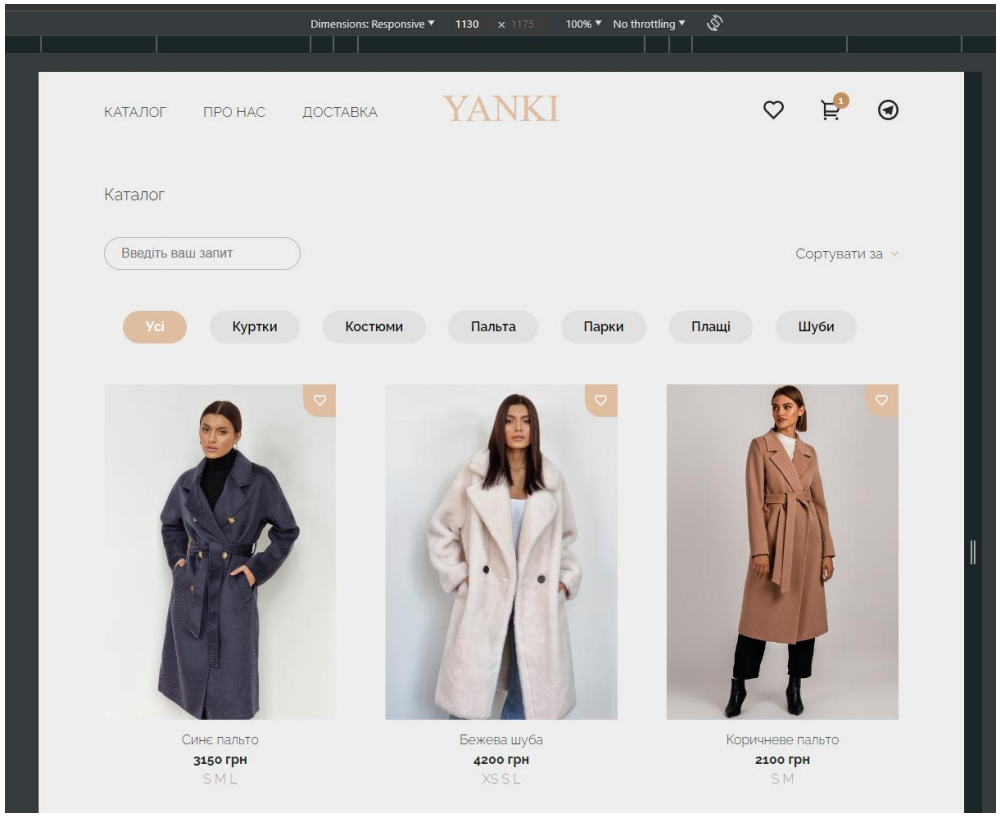

Рисунок 3.52. Сторінка каталогу при роздільній здатності екрану 1130 пікселів

Повний код адаптивної стилізації наведено в git-репозиторії [14].

3.5.3 Сторінка товару

Рисунок 3.53. Сторінка товару при роздільній здатності екрану 375 пікселів

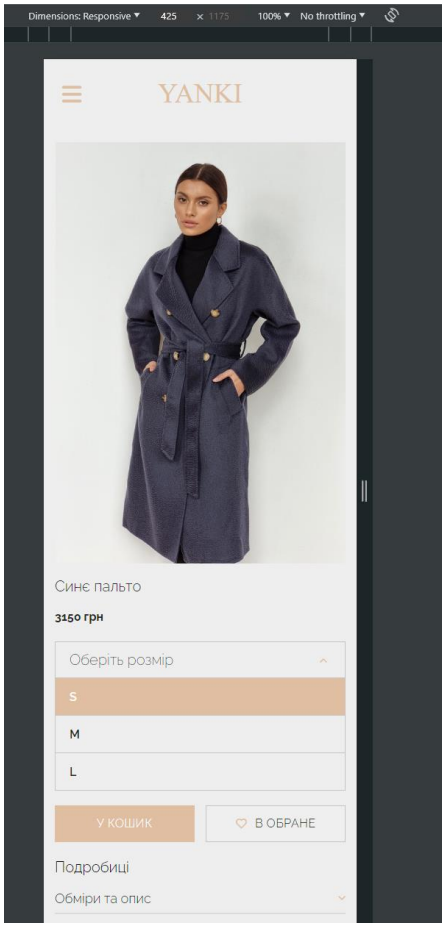

Рисунок 3.54. Сторінка товару при роздільній здатності екрану 425 пікселів

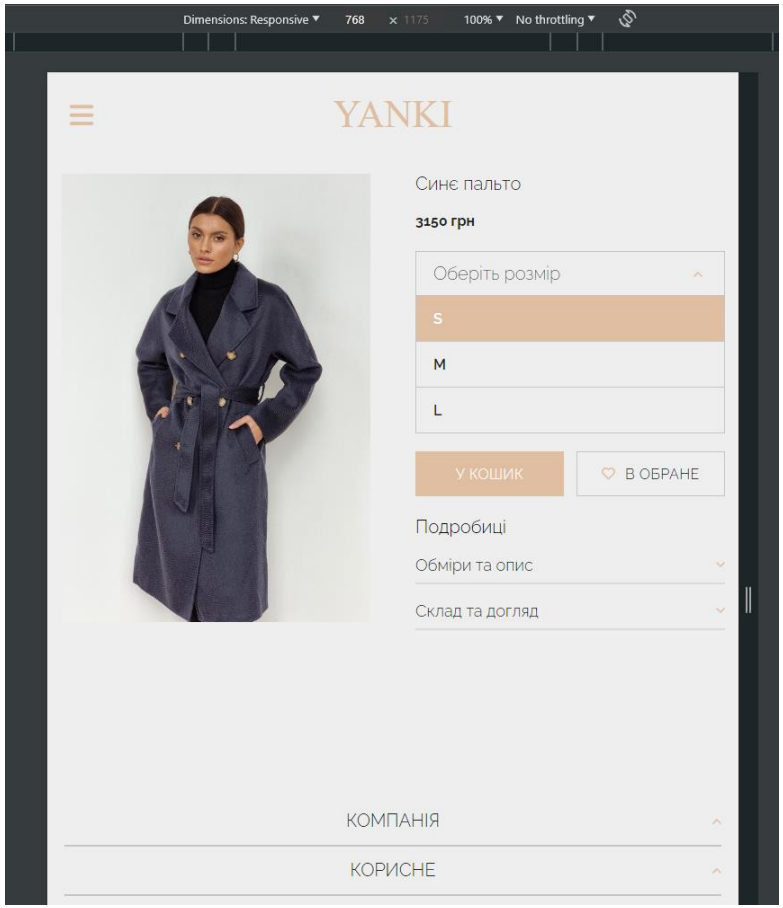

Рисунок 3.55. Сторінка товару при роздільній здатності екрану 768 пікселів

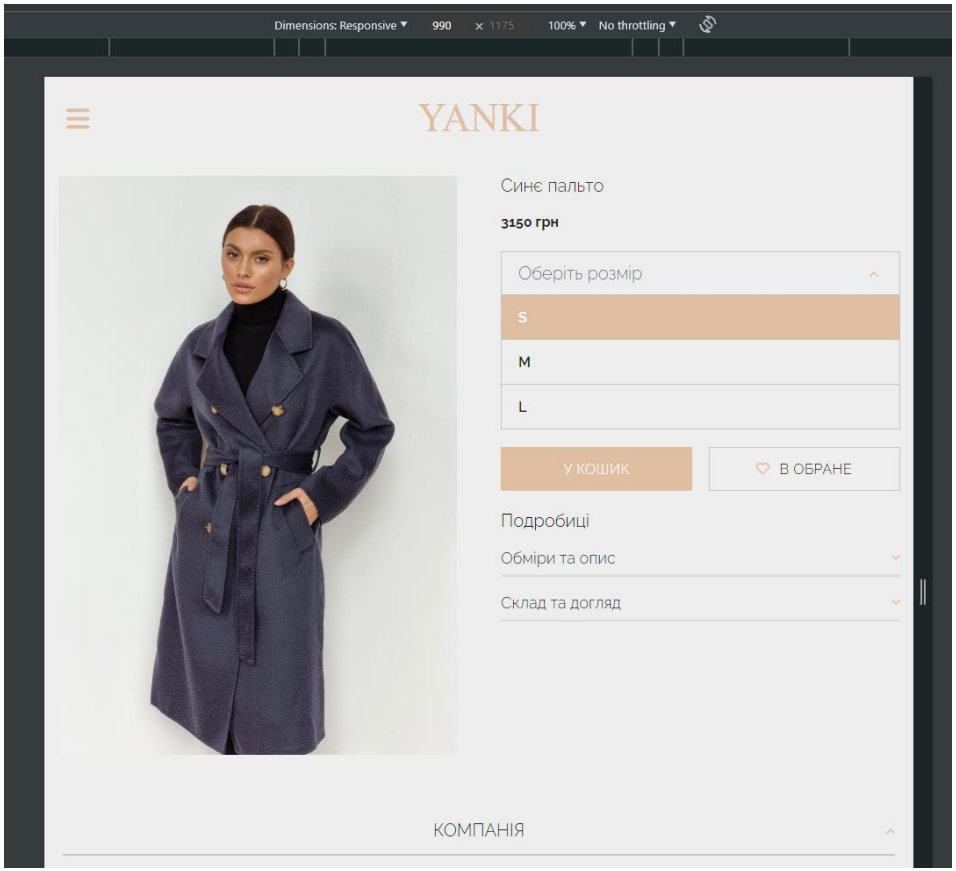

Рисунок 3.56. Сторінка товару при роздільній здатності екрану 990 пікселів

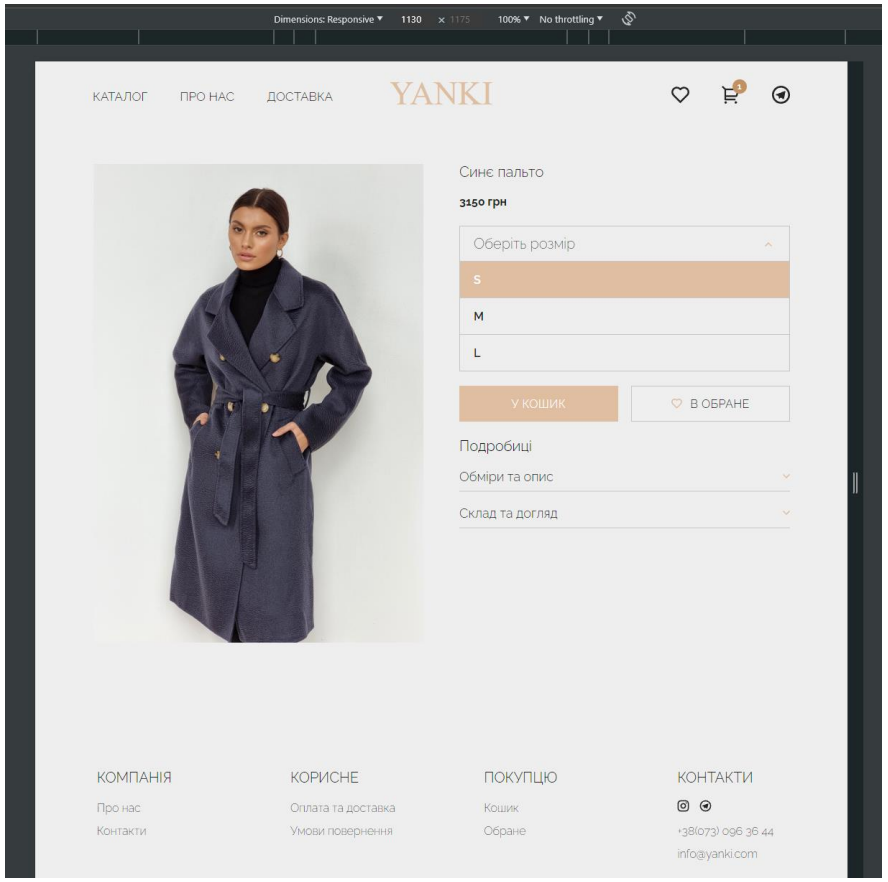

Рисунок 3.57. Сторінка товару при роздільній здатності екрану 1130 пікселів

Повний код адаптивної стилізації наведено в git-репозиторії [14].

3.5.4 Сторінка улюбленого

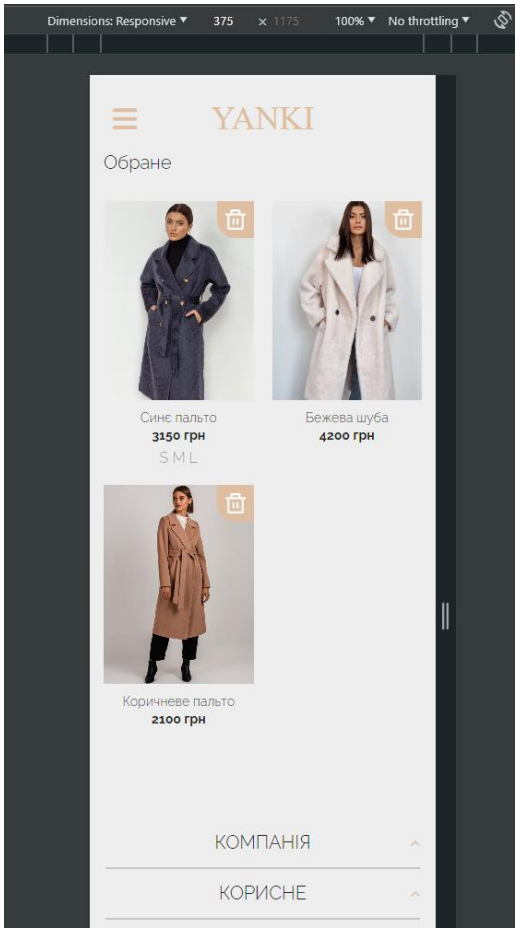

Рисунок 3.58. Сторінка улюбленого при роздільній здатності екрану 375 пікселів

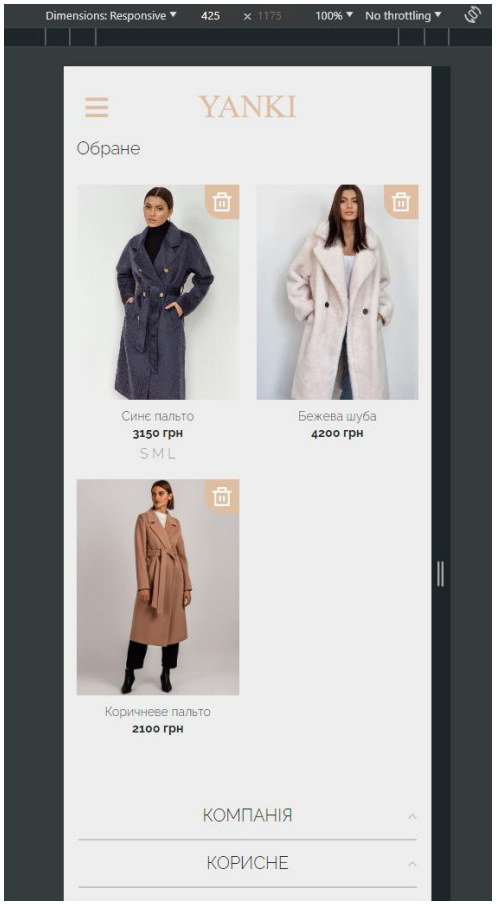

Рисунок 3.59. Сторінка улюбленого при роздільній здатності екрану 425 пікселів

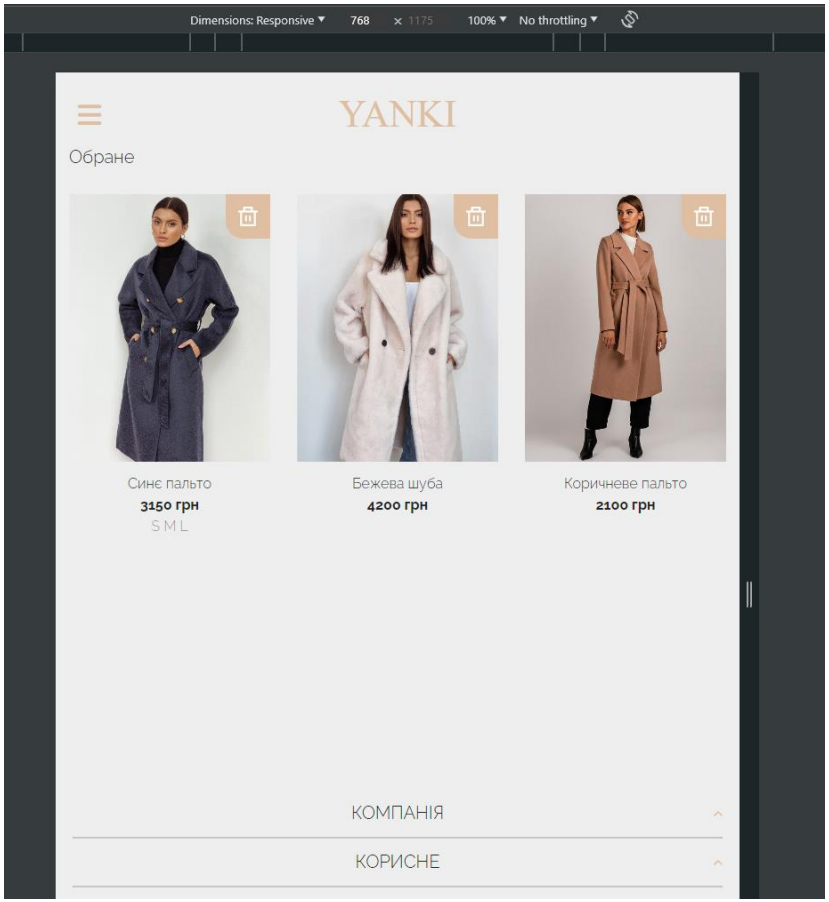

Рисунок 3.60. Сторінка улюбленого при роздільній здатності екрану 768 пікселів

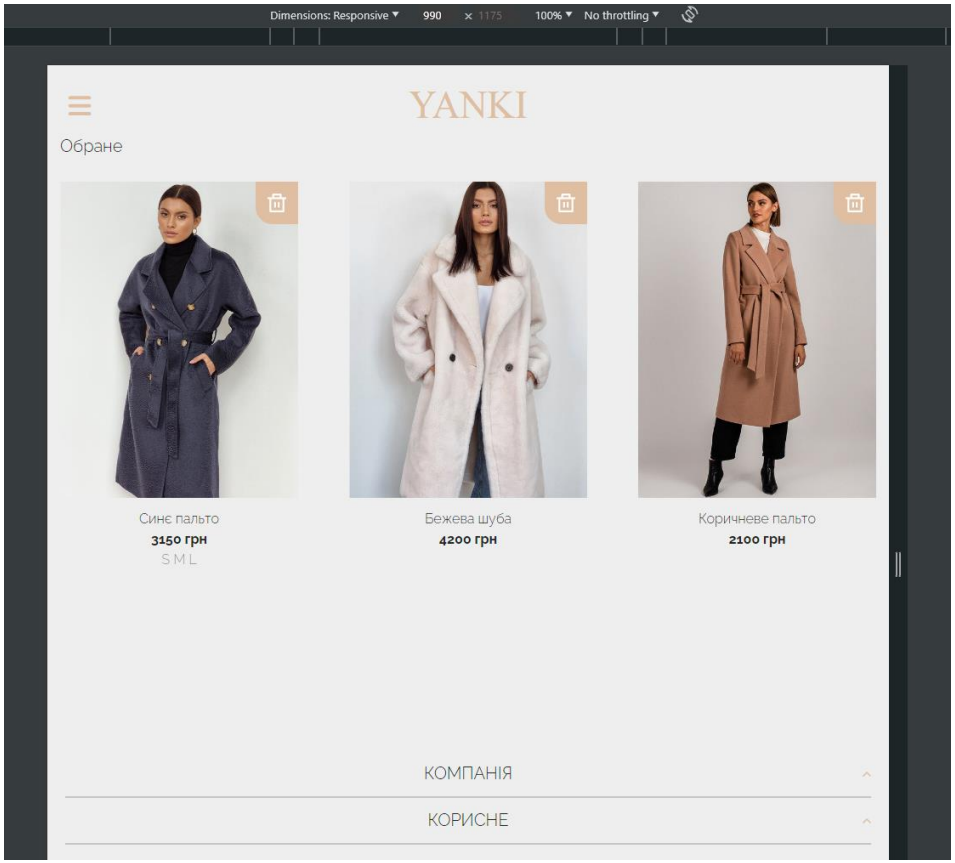

Рисунок 3.61. Сторінка улюбленого при роздільній здатності екрану 990 пікселів

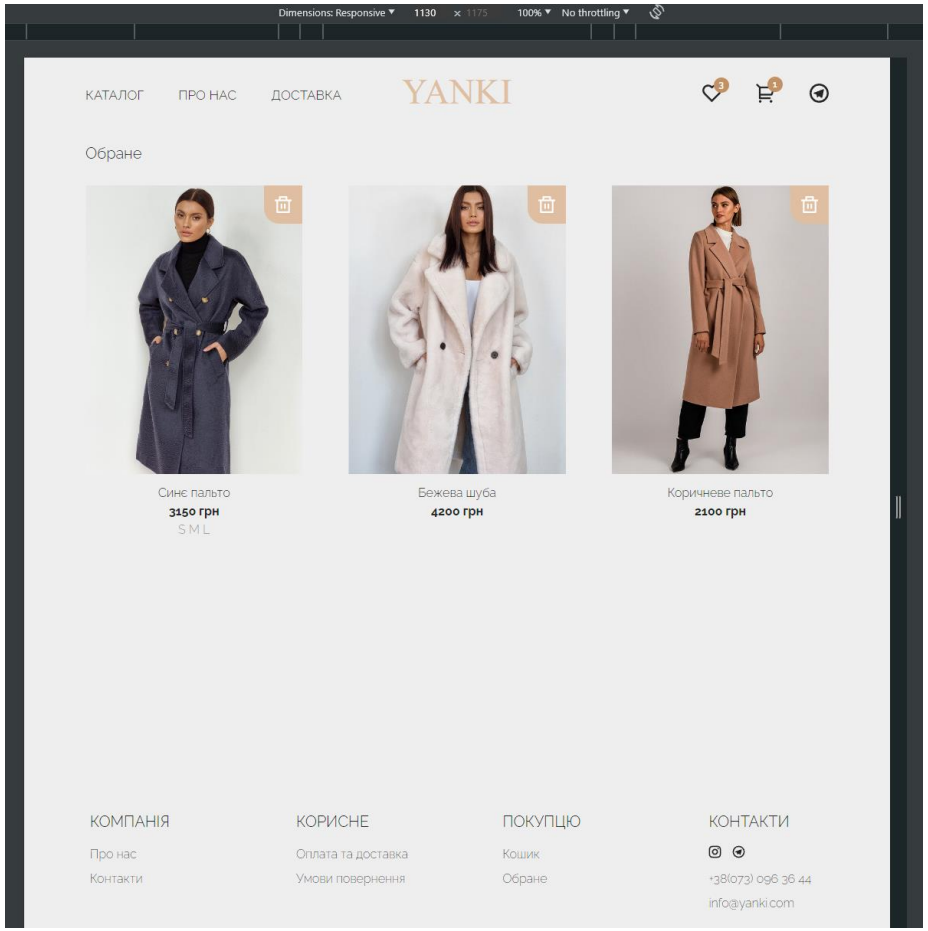

Рисунок 3.62. Сторінка улюбленого при роздільній здатності екрану 1130 пікселів

Повний код адаптивної стилізації наведено в git-репозиторії [14].

3.5.5 Сторінка кошику

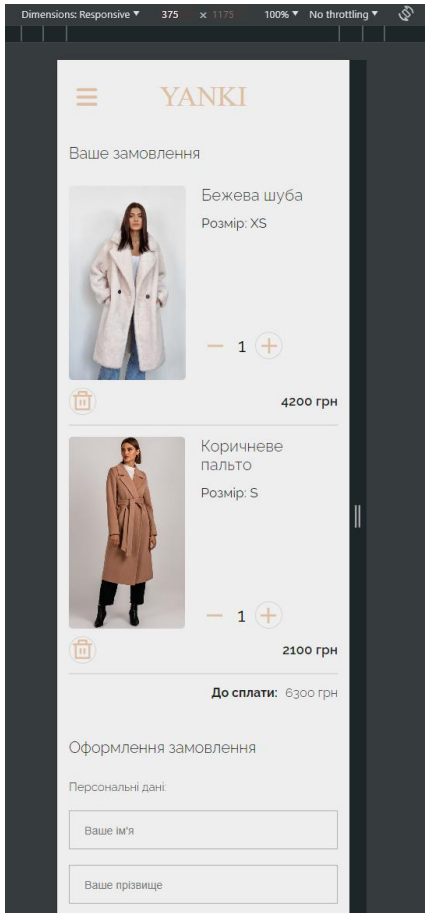

Рисунок 3.63. Сторінка кошику при роздільній здатності екрану 375 пікселів

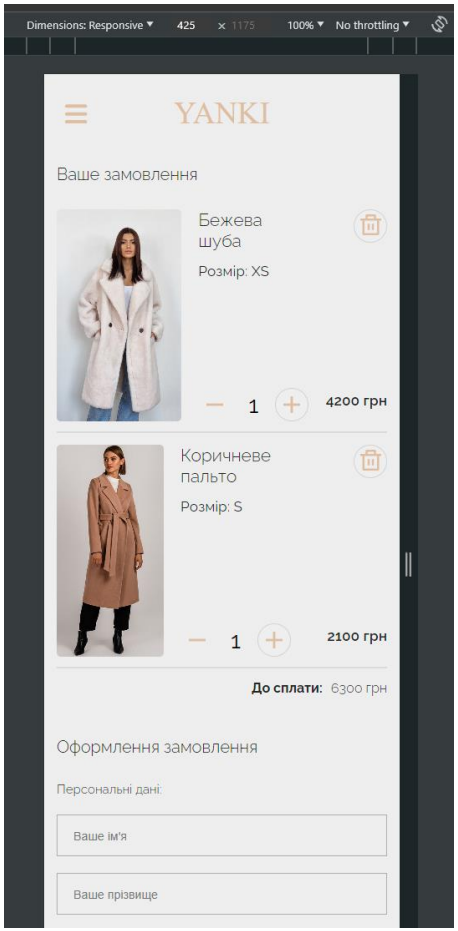

Рисунок 3.64. Сторінка кошику при роздільній здатності екрану 425 пікселів

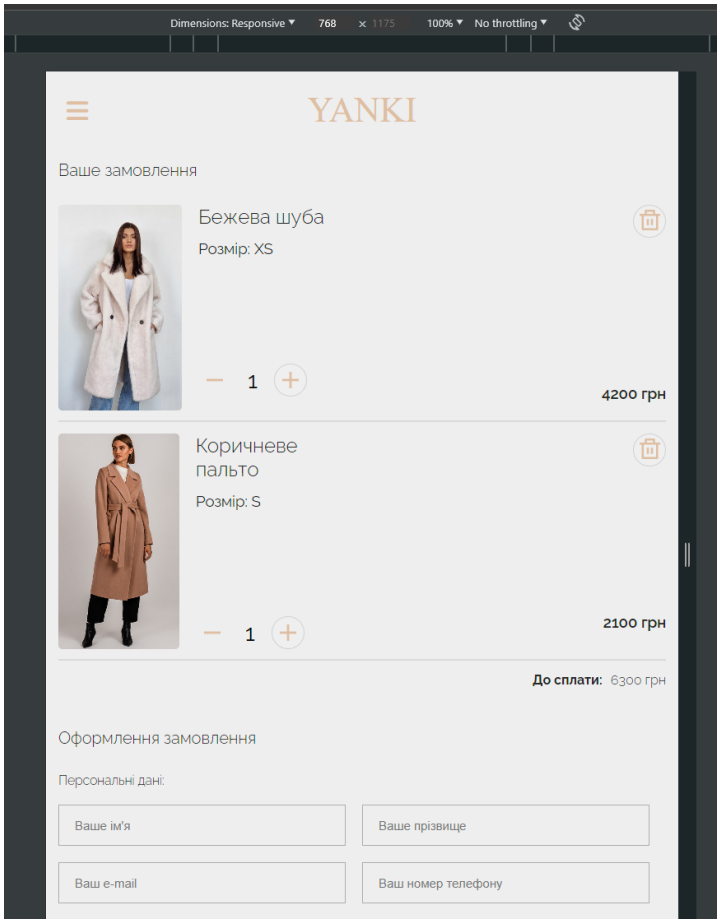

Рисунок 3.65. Сторінка кошику при роздільній здатності екрану 768 пікселів

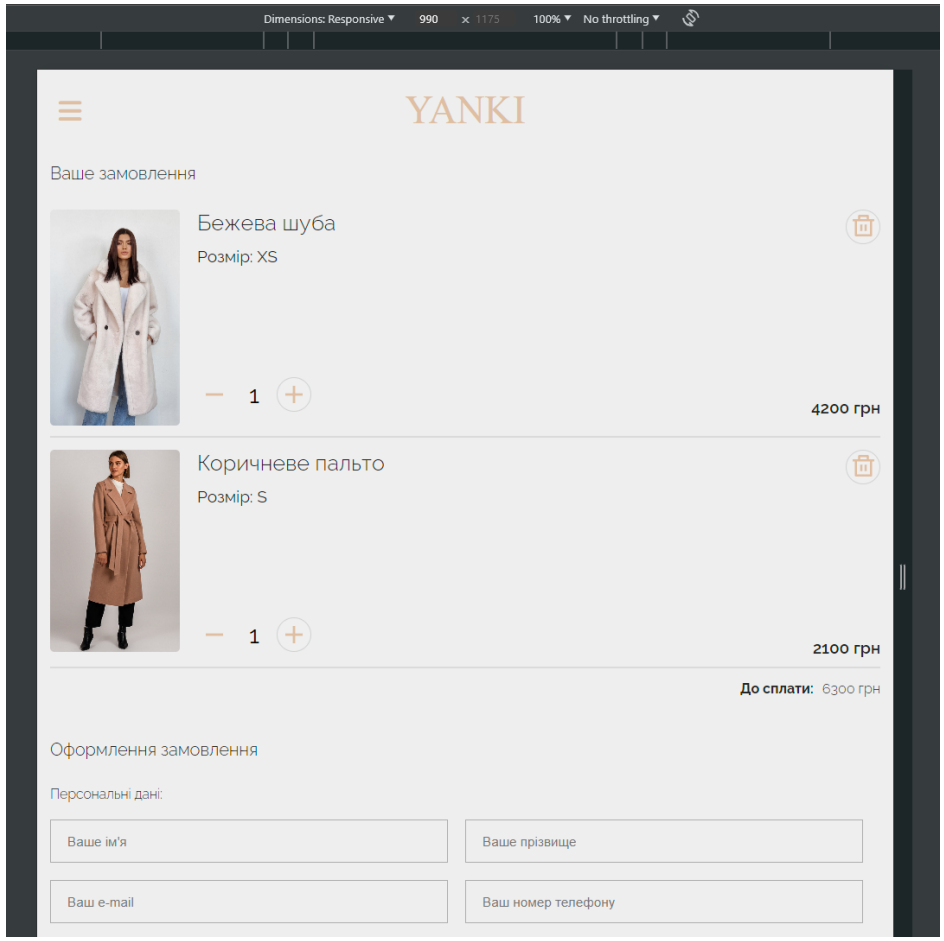

Рисунок 3.66. Сторінка кошику при роздільній здатності екрану 990 пікселів

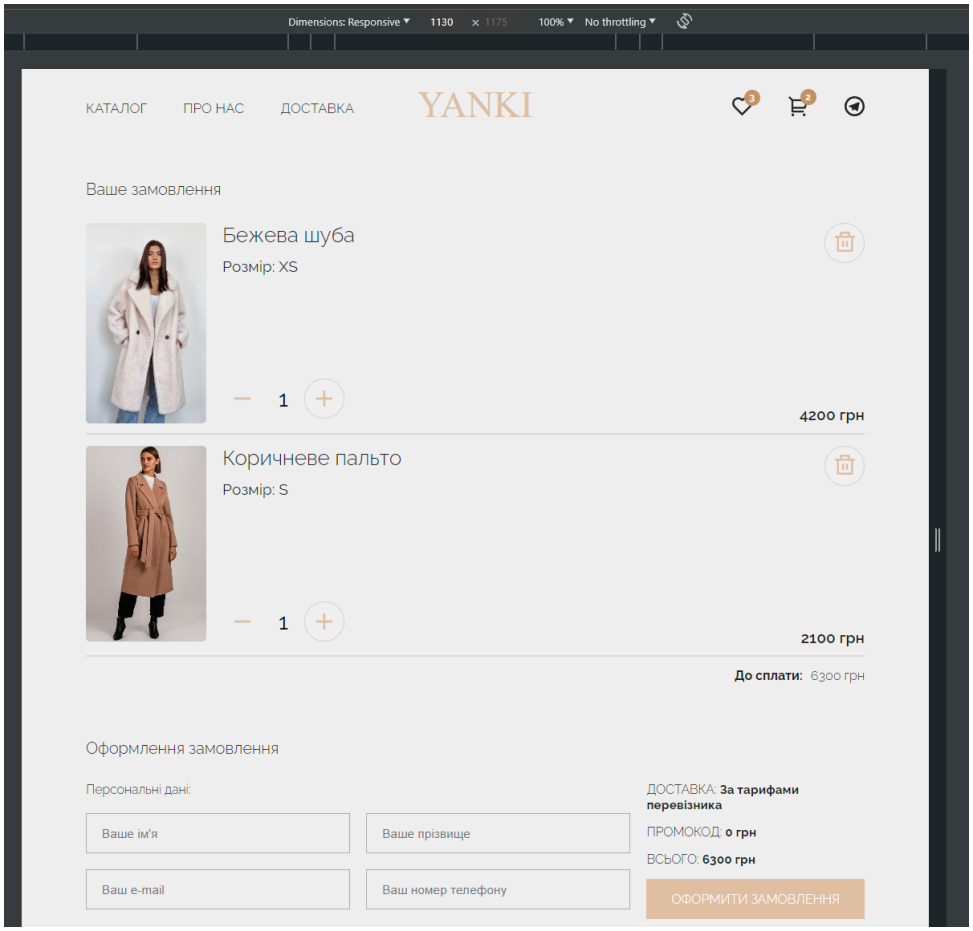

Рисунок 3.67. Сторінка кошику при роздільній здатності екрану 1130 пікселів

Повний код адаптивної стилізації наведено в git-репозиторії [14].

3.6 Розміщення вебсайту на хостингу

Для розміщення вебсайту в Інтернеті було використано безкоштовний сервіс Vercel https://vercel.com/ [10, 16]. Авторизувавшись на сайті, потрібно імпортувати GitHub репозиторій проекту, щоб розпочати деплоймент (див. рисунок 3.68):

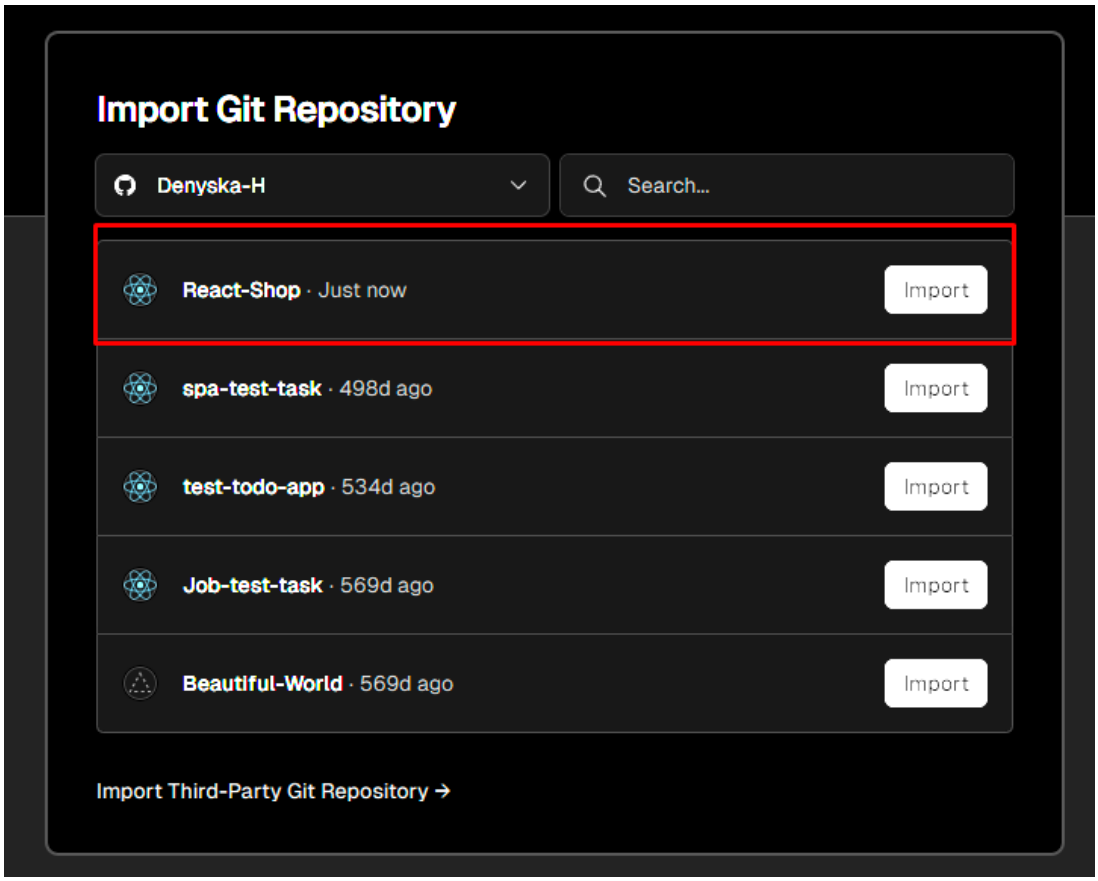

Рисунок 3.68. Імпорт GitHub репозиторію

Після успішного імпорту, потрібно лише натиснути Deploy і проект буде задеплоїно (див. рисунок 3.69)

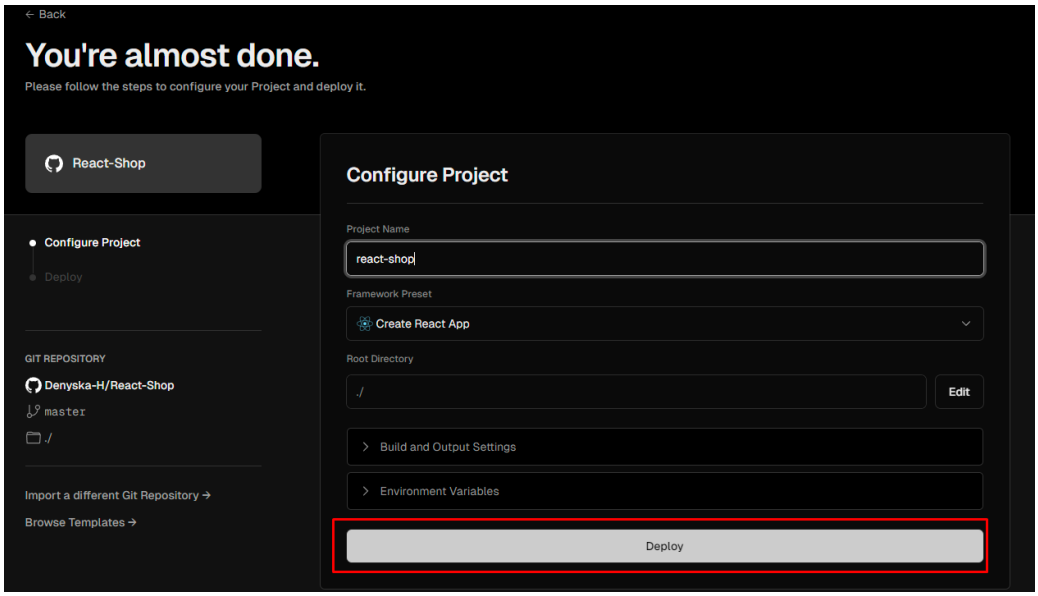

Рисунок 3.69. Деплоймент проєкту

У результаті отримуємо успішний деплоймент проєкту (див. рисунок 3.70) та посилання на сам вебсайт https://react-shop-swart.vercel.app/ [17]

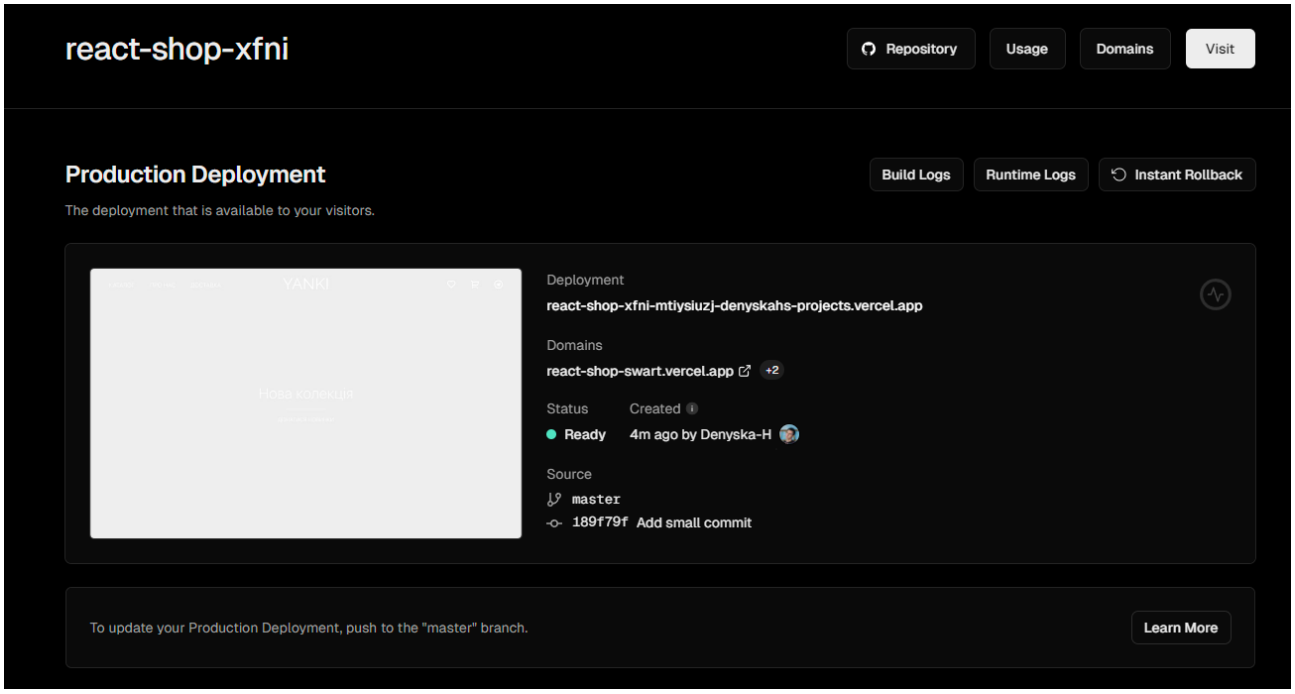

Рисунок 3.70. Успішний Production Deployment

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено вебсайт інтернет-магазину жіночого одягу з реалізацією сортування, пошуку, фільтрації товарів з максимально простим і зрозумілим інтерфейсом і адаптивною версткою.

Під час процесу розробки вебсайту були отримані знання щодо останніх технологій у сфері веброзробки, а також проаналізовані існуючі рішення в даній предметній області.

У роботі були використані найновіші рішення (бібліотеки) та методи для створення вебсайту, розташування компонентів та стилізації текстового контенту задля створення зрозумілого інтерфейсу інтернет-магазину, який можна буде легко підтримувати та розширювати, додавши новий функціонал.

Було створено організацію вебсторінок та їхні складові у вигляді діграми.

Також було додано адаптивну верстку для найкращого користувацького досвіду і можливості перегляду сайту на будь-якій доступній роздільній здатності екрану.

Реалізація цього проєкту є основою для подальшого розширення функціоналу, що може бути впроваджений в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Leomirra. Одяг від українських дизайнерів і брендів URL: https://leomirra.com.ua/ (дата звернення: 09.04.2024).
2. Maritel – Інтернет-магазин жіночого одягу від українського виробника. URL: https://maritel.com.ua/ (дата звернення: 09.04.2024).
3. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/ (дата звернення: 09.04.2024).
4. Introducing Hooks – React. *React – A JavaScript library for building user interfaces*. URL: https://legacy.reactjs.org/docs/hooks-intro.html (дата звернення: 09.04.2024).
5. React Skeleton - Create Content Loader. React Skeleton - Create Content Loader. URL: https://skeletonreact.com (дата звернення: 15.04.2024).
6. JavaScript With Syntax For Types. *TypeScript: JavaScript With Syntax For Types.* URL: https://www.typescriptlang.org/ (дата звернення: 12.04.2024).
7. Create React App. *Create React App*. URL: https://create-react-app.dev/ (дата звернення: 15.04.2024).
8. mockAPI. mockAPI. URL: https://mockapi.io (дата звернення: 10.04.2024).
9. Redux Toolkit | Redux Toolkit. *Redux Toolkit | Redux Toolkit*. URL: https://redux-toolkit.js.org/ (дата звернення: 10.04.2024).
10. Vercel Documentation. URL: https://vercel.com/docs (дата звернення: 20.04.2024).
11. Безкоштовний хостинг зображень / завантаження малюнків. *ImgBB*. URL: https://uk.imgbb.com/ (дата звернення: 20.05.2024).
12. Початок | Axios Docs. *Axios*. URL: https://axios-http.com/uk/docs/intro (дата звернення: 10.04.2024).
13. Sass: Syntax. *Sass: Syntactically Awesome Style Sheets*. URL: https://sass-lang.com/documentation/syntax/ (дата звернення: 10.04.2024).
14. GitHub - Denyska-H/React-Shop: Online e-commerce shop. *GitHub*. URL: https://github.com/Denyska-H/React-Shop (дата звернення: 01.05.2024).

15. Screen Resolution Stats Worldwide | Statcounter Global Stats. (n.d.). StatCounter Global Stats. URL: https://gs.statcounter.com/screen-resolution-stats (дата звернення 10.05.2024)
16. Vercel: Build and deploy the best Web experiences with The Frontend Cloud. (n.d.). Vercel. URL: https://vercel.com/ (дата звернення: 10.05.2024)
17. Women's Clothing Online Store Website. URL: https://react-shop-swart.vercel.app/ (дата звернення: 10.05.2024)

ДОДАТКИ

Увесь код вебсайту магазину жіночого одягу наведено в GitHub репозиторії [14]

items.json


```
[
  {
    "id": "0",
    "imageUrl": "https://i.ibb.co/SnjX4f8/coat.png",
    "title": "Синє пальто",
    "sizes": [
      "S ",
      "M ",
      "L "
    ],
    "price": 3150,
    "rating": 1,
    "category": 3
  },
  {
    "id": "1",
    "imageUrl": "https://i.ibb.co/1s6jQ2d/khaki-cloak.jpg",
    "title": "Хакі плащ",
    "sizes": [
      "M ",
      "L "
    ],
    "price": 1750,
    "rating": 4,
    "category": 5
  },
  {
    "id": "2",
    "imageUrl": "https://i.ibb.co/5n6zNm6/fur-coat.png",
    "title": "Бежева шуба",
    "sizes": [
      "XS ",
      "S ",
      "L "
    ],
    "price": 4200,
    "rating": 2,
    "category": 6
  },
  {
    "id": "3",
    "imageUrl": "https://i.ibb.co/88F9r2k/parka.png",
    "title": "Синя парка",
    "sizes": [
      "XS ",
```

```
      "S ",
      "M ",
      "L "
    ],
    "price": 2900,
    "rating": 5,
    "category": 4
  },
  {
    "id": "4",
    "imageUrl": "https://i.ibb.co/D86CkfS/light-blue-cloak.jpg",
    "title": "Блакитний плащ",
    "sizes": [
      "S ",
      "M ",
      "L "
    ],
    "price": 2000,
    "rating": 3,
    "category": 5
  },
  {
    "id": "5",
    "imageUrl": "https://i.ibb.co/D5x6T11/beigt-jacket.jpg",
    "title": "Бежева куртка",
    "sizes": [
      "S ",
      "M ",
      "L "
    ],
    "price": 1700,
    "rating": 4,
    "category": 1
  },
  {
    "id": "6",
    "imageUrl": "https://i.ibb.co/KNkvpZh/jacket.png",
    "title": "Біла куртка",
    "sizes": [
      "XS ",
      "S ",
      "M ",
      "L "
    ],
    "price": 2900,
    "rating": 4,
    "category": 1
  },
  {
    "id": "7",
    "imageUrl": "https://i.ibb.co/Jzpj6Mr/green-costume.jpg",
    "title": "Зелений костюм",
```

```
    "sizes": [
      "S ",
      "M ",
      "L "
    ],
    "price": 2600,
    "rating": 3,
    "category": 2
  },
  {
    "id": "8",
    "imageUrl": "https://i.ibb.co/wcXMc2b/light-blue-jacket.jpg",
    "title": "Блакитна куртка",
    "sizes": [
      "XS ",
      "S ",
      "M ",
      "L "
    ],
    "price": 2200,
    "rating": 5,
    "category": 1
  },
  {
    "id": "9",
    "imageUrl": "https://i.ibb.co/C7hYZNT/black-costume.jpg",
    "title": "Чорний костюм",
    "sizes": [
      "XS ",
      "S ",
      "M ",
      "L "
    ],
    "price": 2400,
    "rating": 3,
    "category": 2
  },
  {
    "id": "10",
    "imageUrl": "https://i.ibb.co/cyt70Z1/white-coat.jpg",
    "title": "Біле пальто",
    "sizes": [
      "M ",
      "L "
    ],
    "price": 2300,
    "rating": 3,
    "category": 3
  },
  {
    "id": "11",
    "imageUrl": "https://i.ibb.co/g9m9VnF/pink-fur-coat.jpg",
```

```
  "title": "Рожева шуба",
  "sizes": [
    "S ",
    "M ",
    "L "
  ],
  "price": 2200,
  "rating": 4,
  "category": 6
},
{
  "id": "12",
  "imageUrl": "https://i.ibb.co/NptC63P/pink-costume.jpg",
  "title": "Рожевий костюм",
  "sizes": [
    "S ",
    "M "
  ],
  "price": 2400,
  "rating": 3,
  "category": 2
},
{
  "id": "13",
  "imageUrl": "https://i.ibb.co/yRsB5Xb/brown-jacket.jpg",
  "title": "Коричнева куртка",
  "sizes": [
    "S ",
    "M ",
    "L "
  ],
  "price": 1850,
  "rating": 4,
  "category": 1
},
{
  "id": "14",
  "imageUrl": "https://i.ibb.co/dbLPvYs/light-blue-coat.jpg",
  "title": "Блакитне пальто",
  "sizes": [
    "XS ",
    "S ",
    "M ",
    "L "
  ],
  "price": 2700,
  "rating": 5,
  "category": 3
},
{
  "id": "15",
  "imageUrl": "https://i.ibb.co/1qGsWmk/brown-coat.jpg",
```

```
    "title": "Коричневе пальто",
    "sizes": [
      "S ",
      "M "
    ],
    "price": 2100,
    "rating": 2,
    "category": 3
  },
  {
    "id": "16",
    "imageUrl": "https://i.ibb.co/8cvrRgR/blue-jacket.jpg",
    "title": "Синя куртка",
    "sizes": [
      "M ",
      "L "
    ],
    "price": 1890,
    "rating": 3,
    "category": 1
  },
  {
    "id": "17",
    "imageUrl": "https://i.ibb.co/VqPKPqc/black-parka.jpg",
    "title": "Чорна парка",
    "sizes": [
      "XS ",
      "S ",
      "M ",
      "L "
    ],
    "price": 2150,
    "rating": 5,
    "category": 4
  }
]
```

categories.json


```
[
  {
    "id": "0",
    "categoryName": "Пальта",
    "categoryImage": "https://i.ibb.co/SnjX4f8/coat.png"
  },
  {
    "id": "1",
    "categoryName": "Плащі",
    "categoryImage": "https://i.ibb.co/D86CkfS/light-blue-cloak.jpg"
  },
  {
    "id": "2",
    "categoryName": "Костюми",
    "categoryImage": "https://i.ibb.co/Jzpj6Mr/green-costume.jpg"
  },
  {
    "id": "3",
    "categoryName": "Куртки",
    "categoryImage": "https://i.ibb.co/wcXMc2b/light-blue-jacket.jpg"
  },
  {
    "id": "4",
    "categoryName": "Шуби",
    "categoryImage": "https://i.ibb.co/g9m9VnF/pink-fur-coat.jpg"
  },
  {
    "id": "5",
    "categoryName": "Парки",
    "categoryImage": "https://i.ibb.co/VqPKPqc/black-parka.jpg"
  }
]
```

catalogSlice.ts

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';

import { fetchCatalogItems } from './asyncActions';
import { Catalog, CatalogSliceState, Status } from './types';

const initialState: CatalogSliceState = {
  items: [],
  status: Status.PENDING,
};

const catalogSlice = createSlice({
  name: 'catalog',
  initialState,
  reducers: {
    setItems(state, action: PayloadAction<Catalog[]>) {
      state.items = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder.addCase(fetchCatalogItems.pending, (state) => {
      state.status = Status.PENDING;
      state.items = [];
    });

    builder.addCase(fetchCatalogItems.fulfilled, (state, action) => {
      state.status = Status.FULFILLED;
      state.items = action.payload;
    });

    builder.addCase(fetchCatalogItems.rejected, (state) => {
      state.status = Status.REJECTED;
      state.items = [];
    });
  },
});

export const { setItems } = catalogSlice.actions;

export default catalogSlice.reducer;
```

asyncActions.ts

```
import { createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";
import { Catalog, SearchItemsParams } from "./types";

export const fetchCatalogItems = createAsyncThunk<Catalog[], SearchItemsParams>(
  'catalog/fetchCatalogStatus', async (params) => {
  const { category, search, sortBy, order, currentPage } = params;
  const { data } = await axios.get<Catalog[]>(
    `https://62c6de9974e1381c0a6b2e65.mockapi.io/items?&page=${currentPage}&limit=6&${category}&${search}&sor
tBy=${sortBy}&order=${order}`,
  );
  return data;
});
```

cardSlice.ts

```ts
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
import { fetchCardItems } from './asyncActions';
import { Card, CardSliceState, CardStatus } from './types';

const initialState: CardSliceState = {
  items: {
    id: '',
    newId: 0,
    imageUrl: '',
    title: '',
    price: 0,
    size: [],
    sizes: []
  },
  status: CardStatus.PENDING,
};

const cardSlice = createSlice({
  name: 'card',
  initialState,
  reducers: {
    setItems(state, action: PayloadAction<Card>) {
      state.items = action.payload;
    }
  },
  extraReducers(builder) {
    builder.addCase(fetchCardItems.pending, (state) => {
      state.items = state.items;
      state.status = CardStatus.PENDING;
    });

    builder.addCase(fetchCardItems.fulfilled, (state, action) => {
      state.status = CardStatus.FULFILLED;
      state.items = action.payload;
    });

    builder.addCase(fetchCardItems.rejected, (state) => {
      state.status = CardStatus.REJECTED;
    });
  },
})

export const { setItems } = cardSlice.actions;
export default cardSlice.reducer;
```

asyncActions.ts

```ts
import { createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";
```

```
import { Card, SearchCardParams } from "./types";

export const fetchCardItems = createAsyncThunk<Card, SearchCardParams>(
  'card/fetchCardStatus', async (params) => {

  const {id} = params;

  const { data } = await axios.get(
    `https://62c6de9974e1381c0a6b2e65.mockapi.io/items/${id}`,
  );

  return data;
});
```

cartSlice.ts

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
import { CartItem, CartSliceState } from './types';

import { calcTotalPrice } from '../../utils/calcTotalPrice';
import { getCartFromLS } from '../../utils/getCartFromLS';

const { totalPrice, items } = getCartFromLS();

const initialState: CartSliceState = {
  totalPrice,
  items,
  successModal: false,
  errorModal: false,
};


const cartSlice = createSlice({
  name: 'cart',
  initialState,
  reducers: {
    addItems(state, action: PayloadAction<CartItem>) {
      const duplicate = findDuplicates(state.items, action.payload)

      if(duplicate.length > 0) {
        state.items = deleteDuplicates(state.items, action.payload);

        const newItemChecked = {
            ...action.payload,
            newId: duplicate[0].newId,
            count: action.payload.count + duplicate[0].count
        }

        state.errorModal = true;
        state.items.push(newItemChecked);
      } else {
        const newItem = {
          ...action.payload, count: 1,  newId: state.items.length ? state.items[state.items.length - 1].newId
+ 1 : 1
        }

        state.successModal = true;
        state.items.push(newItem)
      }
      state.totalPrice = calcTotalPrice(state.items);
    },

    plusItems(state, action: PayloadAction<number>) {
      const findItem = state.items.find((obj) => obj.newId === action.payload);

      if (findItem) {
```

```
        findItem.count++;
        state.totalPrice = calcTotalPrice(state.items);
      }
    },

    minusItems(state, action: PayloadAction<number>) {
      const findItem = state.items.find((obj) => obj.newId === action.payload);

      if (findItem) {
        findItem.count--;
        state.totalPrice = calcTotalPrice(state.items);
     }
    },

    removeItems(state, action: PayloadAction<number>) {
      state.items = state.items.filter((obj) => obj.newId !== action.payload);
      state.totalPrice = calcTotalPrice(state.items);
    },

    setSuccessModal(state) {
      state.successModal = !state.successModal;
    },

    setErrorModal(state) {
      state.errorModal = !state.errorModal;
    },
  },
});

const findDuplicates = (arr: CartItem[], item: CartItem) => arr.filter(elem => elem.size === item.size &&
elem.title === item.title);
const deleteDuplicates = (arr: CartItem[], item: CartItem) => arr.filter(elem => elem.size !== item.size ||
elem.title !== item.title);

export const { addItems, removeItems, minusItems, plusItems, setSuccessModal, setErrorModal} =
cartSlice.actions;
export default cartSlice.reducer;
```

categorySlice.ts

```
import { createSlice, PayloadAction } from "@reduxjs/toolkit";

import { fetchCategoryItems } from "./asyncActions";
import { Category, CategorySliceState, Status } from "./types";

const initialState: CategorySliceState = {
  categories: [],
  status: Status.PENDING,
};

const categorySlice = createSlice({
  name: 'category',
  initialState,
  reducers: {
    setCategories(state, action: PayloadAction<Category[]>) {
      state.categories = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder.addCase(fetchCategoryItems.pending, (state) => {
      state.status = Status.PENDING;
      state.categories = [];
    });

    builder.addCase(fetchCategoryItems.fulfilled, (state, action) => {
      state.status = Status.FULFILLED;
      state.categories = action.payload;
    });

    builder.addCase(fetchCategoryItems.rejected, (state) => {
      state.status = Status.REJECTED;
      state.categories = [];
    });
  },
});

export const { setCategories } = categorySlice.actions;

export default categorySlice.reducer;
```

asyncActions.ts

```
import { createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";
import { Category } from "./types";

export const fetchCategoryItems = createAsyncThunk<Category[]>(
  'category/fetchCategoryStatus', async () => {
  const { data } = await axios.get<Category[]>(
    `https://62c6de9974e1381c0a6b2e65.mockapi.io/categories`,
  );
  return data;
});
```

filterSlice.ts

```ts
import { createSlice, PayloadAction } from "@reduxjs/toolkit";
import { FilterSliceState, SortPropertyEnum, SortType } from "./types";

const initialState: FilterSliceState = {
  categoryId: 0,
  searchValue: '',
  sort: {
    name: 'популярності',
    sortProperty: SortPropertyEnum.RATING,
  },
  currentPage: 1,
}

const filterSlice = createSlice({
  name: 'filters',
  initialState,
  reducers: {
    setCategoryId(state, action: PayloadAction<number>) {
      state.categoryId = action.payload;
    },
    setSearchValue(state, action: PayloadAction<string>) {
      state.searchValue = action.payload;
    },
    setSort(state, action: PayloadAction<SortType>) {
      state.sort = action.payload;
    },
    setCurrentPage(state, action: PayloadAction<number>) {
      state.currentPage = action.payload;
    },
  }
})

export const { setCategoryId, setSearchValue, setSort, setCurrentPage } = filterSlice.actions;

export default filterSlice.reducer;
```

store.ts

```ts
import { configureStore } from '@reduxjs/toolkit';
import { useDispatch } from 'react-redux';

import catalog from './catalog/slice'
import filters from './filter/slice'
import category from './category/slice'
import cart from './cart/slice'
import card from './card/slice'
import favorite from './favorite/slice'

export const store = configureStore({
  reducer: {
    catalog,
    filters,
    category,
    cart,
    card,
    favorite,
  },
});

export type RootState = ReturnType<typeof store.getState>;

export type AppDispatch = typeof store.dispatch;

export const useAppDispatch = () => useDispatch<AppDispatch>();
```

index.tsx

```tsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import { store } from './redux/store';

import './scss/app.scss';

const root = ReactDOM.createRoot(document.getElementById('root') as HTMLElement);

root.render(
  <Provider store={store}>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </Provider>,
);
```

Header.tsx

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

import { RootState } from '../../redux/store';

const Header: React.FC = () => {
  const { items } = useSelector((state: RootState) => state.cart);
  const { favorites } = useSelector((state: RootState) => state.favorite);

  const totalCartAmount = items.reduce((sum, item) => item.count + sum, 0);
  const totalFavoritesAmount = favorites.reduce((sum, item) => item.count + sum, 0);

  const [isMobile, setIsMobile] = React.useState(false);
  const [screenWidth, setScreenWidth] = React.useState(window.innerWidth);

  React.useEffect(() => {
    const changeWidth = () => {
      setScreenWidth(window.innerWidth);
    };

    window.addEventListener('resize', changeWidth);

    return () => {
      window.removeEventListener('resize', changeWidth);
    };
  }, []);

  return (
    <header className={screenWidth > 992 || !isMobile ? 'header' : 'header_m'}>
      <div className="container">
        <button
          className="mobile-menu-icon mobile-menu-icon_new"
          aria-label="mobileMenu"
          onClick={() => setIsMobile(!isMobile)}>
          {isMobile ? <i className="fas fa-times"></i> : <i className="fas fa-bars"></i>}
        </button>
        <nav className="header__menu menu">
          <ul className={screenWidth > 992 || !isMobile ? 'menu__list' : 'menu__list_mobile'}>
            <li className="menu__link menu__link_color">
              <Link to="/catalog">КАТАЛОГ</Link>
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_color">
              <Link to="/about">ПРО НАС</Link>
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_color">
              <Link to="/delivery">ДОСТАВКА</Link>
```

```
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_m">
              <Link to="/favorites">
                <svg
                  width="25"
                  height="24"
                  viewBox="0 0 25 24"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg">
                  <path
                    d="M12.501 2.16119C15.4372 -0.475012 19.9746 -0.387513 22.8033 2.44618C25.6308 5.28113
25.7283 9.79605 23.0983 12.741L12.4985 23.3558L1.90122 12.741C-0.728729 9.79605 -0.62998 5.27363 2.19622
2.44618C5.02741 -0.383763 9.55608 -0.478762 12.501 2.16119ZM21.0334 4.2124C19.1584 2.33494 16.1335 2.25869
14.171 4.02115L12.5023 5.51863L10.8323 4.0224C8.86359 2.25744 5.8449 2.33494 3.96493 4.2149C2.10247 6.07737
2.00872 9.05856 3.72494 11.0285L12.4998 19.8171L21.2746 11.0298C22.9921 9.05856 22.8983 6.08112 21.0334
4.2124Z"
                    fill="white"
                  />
                </svg>
                <p
                  className={totalFavoritesAmount === 0 ? 'menu__counter hidden' : 'menu__counter'}>
                  {totalFavoritesAmount}
                </p>
              </Link>
            </li>
            <div className="hr"></div>
            <li className="menu__link menu__link_m">
              <Link to="/cart">
                <svg
                  width="25"
                  height="25"
                  viewBox="0 0 25 25"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg">
                  <path
                    d="M3.81655 5.48063L0 1.66525L1.66525 0L5.48063 3.81655H23.4183C23.6017 3.81654 23.7826
3.85941 23.9466 3.94173C24.1105 4.02406 24.2529 4.14356 24.3624 4.2907C24.4719 4.43785 24.5456 4.60856
24.5774 4.78921C24.6092 4.96986 24.5984 5.15545 24.5457 5.33117L21.7213 14.746C21.6486 14.9886 21.4997
15.2012 21.2966 15.3524C21.0935 15.5035 20.847 15.5852 20.5938
15.5851H6.17027V17.9389H19.1157V20.2926H4.99341C4.68129 20.2926 4.38195 20.1686 4.16125 19.9479C3.94054
19.7272 3.81655 19.4278 3.81655 19.1157V5.48063ZM6.17027 6.17027V13.2314H19.7183L21.8366
6.17027H6.17027ZM5.58184 25C5.11366 25 4.66465 24.814 4.33359 24.483C4.00254 24.1519 3.81655 23.7029 3.81655
23.2347C3.81655 22.7665 4.00254 22.3175 4.33359 21.9865C4.66465 21.6554 5.11366 21.4694 5.58184
21.4694C6.05002 21.4694 6.49903 21.6554 6.83009 21.9865C7.16114 22.3175 7.34713 22.7665 7.34713
23.2347C7.34713 23.7029 7.16114 24.1519 6.83009 24.483C6.49903 24.814 6.05002 25 5.58184 25ZM19.7041
25C19.236 25 18.7869 24.814 18.4559 24.483C18.1248 24.1519 17.9388 23.7029 17.9388 23.2347C17.9388 22.7665
18.1248 22.3175 18.4559 21.9865C18.7869 21.6554 19.236 21.4694 19.7041 21.4694C20.1723 21.4694 20.6213
21.6554 20.9524 21.9865C21.2834 22.3175 21.4694 22.7665 21.4694 23.2347C21.4694 23.7029 21.2834 24.1519
20.9524 24.483C20.6213 24.814 20.1723 25 19.7041 25Z"
                    fill="white"
                  />
```

```
              </svg>
              <p className={totalCartAmount === 0 ? 'menu__counter hidden' : 'menu__counter'}>
                {totalCartAmount}
              </p>
            </Link>
          </li>
          <div className="hr"></div>
          <li className="menu__link menu__link_m">
            <Link to="/contacts">
              <svg
                width="25"
                height="25"
                viewBox="0 0 25 25"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M12.5 22.5C15.1522 22.5 17.6957 21.4464 19.5711 19.5711C21.4464 17.6957 22.5 15.1522
22.5 12.5C22.5 9.84783 21.4464 7.3043 19.5711 5.42893C17.6957 3.55357 15.1522 2.5 12.5 2.5C9.84783 2.5 7.3043
3.55357 5.42893 5.42893C3.55357 7.3043 2.5 9.84783 2.5 12.5C2.5 15.1522 3.55357 17.6957 5.42893
19.5711C7.3043 21.4464 9.84783 22.5 12.5 22.5ZM12.5 25C5.59625 25 0 19.4037 0 12.5C0 5.59625 5.59625 0 12.5
0C19.4037 0 25 5.59625 25 12.5C25 19.4037 19.4037 25 12.5 25ZM8.6125 13.9625L5.49 12.9887C4.815 12.7825
4.81125 12.3175 5.64125 11.9837L17.8075 7.28375C18.5137 6.99625 18.9137 7.36 18.685 8.27125L16.6137
18.0462C16.4688 18.7425 16.05 18.9088 15.4687 18.5875L12.28 16.2275L10.7937 17.6625C10.6412 17.81 10.5175
17.9363 10.2825 17.9675C10.0487 18 9.85625 17.93 9.715 17.5425L8.6275 13.9537L8.6125 13.9625Z"
                  fill="white"
                />
              </svg>
            </Link>
          </li>
          <div className="hr"></div>
        </ul>
        <ul className="menu__list">
          <li className="menu__link menu__link_color">
            <Link to="/favorites" aria-label="favorites">
              <svg
                width="25"
                height="24"
                viewBox="0 0 25 24"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M12.501 2.16119C15.4372 -0.475012 19.9746 -0.387513 22.8033 2.44618C25.6308 5.28113
25.7283 9.79605 23.0983 12.741L12.4985 23.3558L1.90122 12.741C-0.728729 9.79605 -0.62998 5.27363 2.19622
2.44618C5.02741 -0.383763 9.55608 -0.478762 12.501 2.16119ZM21.0334 4.2124C19.1584 2.33494 16.1335 2.25869
14.171 4.02115L12.5023 5.51863L10.8323 4.0224C8.86359 2.25744 5.8449 2.33494 3.96493 4.2149C2.10247 6.07737
2.00872 9.05856 3.72494 11.0285L12.4998 19.8171L21.2746 11.0298C22.9921 9.05856 22.8983 6.08112 21.0334
4.2124Z"
                  fill="white"
                />
              </svg>
              <p
                className={totalFavoritesAmount === 0 ? 'menu__counter hidden' : 'menu__counter'}>
```

```
                {totalFavoritesAmount}
              </p>
            </Link>
          </li>
          <li className="menu__link menu__link_color">
            <Link to="/cart" aria-label="cart">
              <svg
                width="25"
                height="25"
                viewBox="0 0 25 25"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M3.81655 5.48063L0 1.66525L1.66525 0L5.48063 3.81655H23.4183C23.6017 3.81654 23.7826
3.85941 23.9466 3.94173C24.1105 4.02406 24.2529 4.14356 24.3624 4.2907C24.4719 4.43785 24.5456 4.60856
24.5774 4.78921C24.6092 4.96986 24.5984 5.15545 24.5457 5.33117L21.7213 14.746C21.6486 14.9886 21.4997
15.2012 21.2966 15.3524C21.0935 15.5035 20.847 15.5852 20.5938
15.5851H6.17027V17.9389H19.1157V20.2926H4.99341C4.68129 20.2926 4.38195 20.1686 4.16125 19.9479C3.94054
19.7272 3.81655 19.4278 3.81655 19.1157V5.48063ZM6.17027 6.17027V13.2314H19.7183L21.8366
6.17027H6.17027ZM5.58184 25C5.11366 25 4.66465 24.814 4.33359 24.483C4.00254 24.1519 3.81655 23.7029 3.81655
23.2347C3.81655 22.7665 4.00254 22.3175 4.33359 21.9865C4.66465 21.6554 5.11366 21.4694 5.58184
21.4694C6.05002 21.4694 6.49903 21.6554 6.83009 21.9865C7.16114 22.3175 7.34713 22.7665 7.34713
23.2347C7.34713 23.7029 7.16114 24.1519 6.83009 24.483C6.49903 24.814 6.05002 25 5.58184 25ZM19.7041
25C19.236 25 18.7869 24.814 18.4559 24.483C18.1248 24.1519 17.9388 23.7029 17.9388 23.2347C17.9388 22.7665
18.1248 22.3175 18.4559 21.9865C18.7869 21.6554 19.236 21.4694 19.7041 21.4694C20.1723 21.4694 20.6213
21.6554 20.9524 21.9865C21.2834 22.3175 21.4694 22.7665 21.4694 23.2347C21.4694 23.7029 21.2834 24.1519
20.9524 24.483C20.6213 24.814 20.1723 25 19.7041 25Z"
                  fill="white"
                />
              </svg>
              <p className={totalCartAmount === 0 ? 'menu__counter hidden' : 'menu__counter'}>
                {totalCartAmount}
              </p>
            </Link>
          </li>
          <li className="menu__link menu__link_color">
            <Link to="/contacts" aria-label="contacts">
              <svg
                width="25"
                height="25"
                viewBox="0 0 25 25"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M12.5 22.5C15.1522 22.5 17.6957 21.4464 19.5711 19.5711C21.4464 17.6957 22.5 15.1522
22.5 12.5C22.5 9.84783 21.4464 7.3043 19.5711 5.42893C17.6957 3.55357 15.1522 2.5 12.5 2.5C9.84783 2.5 7.3043
3.55357 5.42893 5.42893C3.55357 7.3043 2.5 9.84783 2.5 12.5C2.5 15.1522 3.55357 17.6957 5.42893
19.5711C7.3043 21.4464 9.84783 22.5 12.5 22.5ZM12.5 25C5.59625 25 0 19.4037 0 12.5C0 5.59625 5.59625 0 12.5
0C19.4037 0 25 5.59625 25 12.5C25 19.4037 19.4037 25 12.5 25ZM8.6125 13.9625L5.49 12.9887C4.815 12.7825
4.81125 12.3175 5.64125 11.9837L17.8075 7.28375C18.5137 6.99625 18.9137 7.36 18.685 8.27125L16.6137
18.0462C16.4688 18.7425 16.05 18.9088 15.4687 18.5875L12.28 16.2275L10.7937 17.6625C10.6412 17.81 10.5175
17.9363 10.2825 17.9675C10.0487 18 9.85625 17.93 9.715 17.5425L8.6275 13.9537L8.6125 13.9625Z"
```

```
                  fill="white"
                />
              </svg>
            </Link>
          </li>
        </ul>
      </nav>
      <div className="header__logo_yanki">
        <Link to={'/'}>
          <span>YANKI</span>
        </Link>
      </div>
    </div>
  </header>
 );
};

export default Header;
```

Fullscreen.tsx

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

import { MainHeader } from '../MainHeader';

const Fullscreen: React.FC = () => {
 return (
  <div className="fullscreen">
    <MainHeader />
    <div className="fullscreen__text text">
      <div className="text">
        <h1 className="text__title">Нова колекція</h1>
        <div className="text__hr"></div>
        <Link to="/catalog">
          <h2 className="text__subtitle">Дізнатися новинки</h2>
        </Link>
      </div>
    </div>
  </div>
 );
};
export default Fullscreen;
```

HomeCategories.tsx

```tsx
import React from 'react';

import { fetchCategoryItems } from '../../redux/category/asyncActions';
import { useAppDispatch } from '../../redux/store';

import { Slider } from '../Slider';

const Categories: React.FC = () => {
  const dispatch = useAppDispatch();

  const getCategories = async () => {
    dispatch(fetchCategoryItems());
  };

  React.useEffect(() => {
    getCategories();
  }, []);

  return (
    <div className="categories">
      <div className="container">
        <h3 className="categories__title">Категорії</h3>
        <Slider />
      </div>
    </div>
  );
};

export default Categories;
```

Novelties.tsx

```tsx
import React from 'react';

const Novelties: React.FC = () => {
  return (
    <div className="novelties">
      <div className="container">
        <div className="novelties__title">Дізнайтесь першими про новинки</div>
        <form action="signin" autoComplete="off" method="post" className="novelties__form">
          <input
            type="text"
            name="email"
            placeholder="Ваш e-mail*"
            pattern='^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$'
            title="Електронна адреса має бути формату example@gmail.com"
            required
          />
          <button className="novelties__button">
            <span>ПІДПИСАТИСЯ</span>
          </button>
          <p className="novelties__subtitle">
            Натискаючи на кнопку «Підписатися», я погоджуюсь на обробку моїх персональних даних та
            ознайомлений(а) з умовами конфіденційності.
          </p>
        </form>
      </div>
    </div>
  );
};

export default Novelties;
```

Footer.tsx

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

const Footer: React.FC = () => {
  const [firstUl, setFirstUl] = React.useState(true);
  const [secUl, setSecUl] = React.useState(true);
  const [thirdUl, setThirdUl] = React.useState(true);

  const [screenWidth, setScreenWidth] = React.useState(window.innerWidth);

  React.useEffect(() => {
    const changeWidth = () => {
      setScreenWidth(window.innerWidth);
    };

    window.addEventListener('resize', changeWidth);

    return () => {
      window.removeEventListener('resize', changeWidth);
    };
  }, []);

  return (
    <footer className="footer">
      <div className="container">
        <div className="footer__row">
          <div className="footer__column">
            <div className="footer__title" onClick={() => setFirstUl(!firstUl)}>
              <div className="footer__label">КОМПАНИЯ</div>
              <div className="footer__arrow">
                <svg
                  className={firstUl ? 'active-svg' : 'default-svg'}
                  width="10"
                  height="7"
                  viewBox="0 0 10 7"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg">
                  <path
                    d="M5 3.88903L8.88906 -3.05043e-05L10 1.11091L5 6.1109L1.32478e-08 1.11091L1.11094 -
3.04116e-05L5 3.88903Z"
                    fill="#E0BEA2"
                  />
                </svg>
              </div>
            </div>
            <ul
              className={screenWidth > 992 || !firstUl ? 'footer__text open-text' : 'closed-text'}>
              <li>
                <Link to="/about">Про нас</Link>
```

```
            </li>
            <li>
              <Link to="/contacts">Контакти</Link>
            </li>
          </ul>
          <div className="footer__hr"></div>
        </div>
        <div className="footer__column">
          <div className="footer__title" onClick={() => setSecUl(!secUl)}>
            <div className="footer__label">КОРИСНЕ</div>
            <div className="footer__arrow">
              <svg
                className={secUl ? 'active-svg' : 'default-svg'}
                width="10"
                height="7"
                viewBox="0 0 10 7"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M5 3.88903L8.88906 -3.05043e-05L10 1.11091L5 6.1109L1.32478e-08 1.11091L1.11094 -
3.04116e-05L5 3.88903Z"
                  fill="#E0BEA2"
                />
              </svg>
            </div>
          </div>
          <ul className={screenWidth > 992 || !secUl ? 'footer__text open-text' : 'closed-text'}>
            <li>
              <Link to="/delivery">Оплата та доставка</Link>
            </li>
            <li>
              <Link to="/refund">Умови повернення</Link>
            </li>
          </ul>
          <div className="footer__hr"></div>
        </div>
        <div className="footer__column">
          <div className="footer__title" onClick={() => setThirdUl(!thirdUl)}>
            <div className="footer__label">ПОКУПЦЮ</div>
            <div className="footer__arrow">
              <svg
                className={thirdUl ? 'active-svg' : 'default-svg'}
                width="10"
                height="7"
                viewBox="0 0 10 7"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M5 3.88903L8.88906 -3.05043e-05L10 1.11091L5 6.1109L1.32478e-08 1.11091L1.11094 -
3.04116e-05L5 3.88903Z"
                  fill="#E0BEA2"
                />
```

```
              </svg>
            </div>
          </div>
          <ul
            className={screenWidth > 992 || !thirdUl ? 'footer__text open-text' : 'closed-text'}>
            <li>
              <Link to="/cart">Кошик</Link>
            </li>
            <li>
              <Link to="/favorites">Обране</Link>
            </li>
          </ul>
          <div className="footer__hr"></div>
        </div>
        <div className="footer__column">
          <div className="footer__title">КОНТАКТИ</div>
          <ul className="footer__text">
            <li>
              <a target="_blank" aria-label="instagram" href="https://www.instagram.com">
                <svg
                  width="17"
                  height="17"
                  viewBox="0 0 17 17"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg">
                  <path
                    d="M8.49992 6.12504C7.87003 6.12504 7.26594 6.37526 6.82054 6.82066C6.37514 7.26606
6.12492 7.87015 6.12492 8.50004C6.12492 9.12993 6.37514 9.73402 6.82054 10.1794C7.26594 10.6248 7.87003
10.875 8.49992 10.875C9.12981 10.875 9.7339 10.6248 10.1793 10.1794C10.6247 9.73402 10.8749 9.12993 10.8749
8.50004C10.8749 7.87015 10.6247 7.26606 10.1793 6.82066C9.7339 6.37526 9.12981 6.12504 8.49992
6.12504ZM8.49992 4.54171C9.54974 4.54171 10.5566 4.95875 11.2989 5.70108C12.0412 6.44341 12.4583 7.45023
12.4583 8.50004C12.4583 9.54986 12.0412 10.5567 11.2989 11.299C10.5566 12.0413 9.54974 12.4584 8.49992
12.4584C7.4501 12.4584 6.44329 12.0413 5.70095 11.299C4.95862 10.5567 4.54159 9.54986 4.54159 8.50004C4.54159
7.45023 4.95862 6.44341 5.70095 5.70108C6.44329 4.95875 7.4501 4.54171 8.49992 4.54171ZM13.6458
4.34379C13.6458 4.60625 13.5415 4.85795 13.3559 5.04353C13.1703 5.22912 12.9186 5.33337 12.6562
5.33337C12.3937 5.33337 12.142 5.22912 11.9564 5.04353C11.7708 4.85795 11.6666 4.60625 11.6666
4.34379C11.6666 4.08134 11.7708 3.82963 11.9564 3.64405C12.142 3.45847 12.3937 3.35421 12.6562
3.35421C12.9186 3.35421 13.1703 3.45847 13.3559 3.64405C13.5415 3.82963 13.6458 4.08134 13.6458
4.34379ZM8.49992 2.16671C6.54134 2.16671 6.2215 2.17225 5.31029 2.21262C4.68963 2.24192 4.27321 2.32504
3.88688 2.47546C3.54329 2.60846 3.2955 2.76758 3.03188 3.032C2.78409 3.2714 2.59357 3.56367 2.47454
3.887C2.32413 4.27492 2.241 4.69054 2.2125 5.31042C2.17134 6.18442 2.16659 6.49 2.16659 8.50004C2.16659
10.4586 2.17213 10.7785 2.2125 11.6897C2.24179 12.3095 2.32492 12.7268 2.47454 13.1123C2.60913 13.4567
2.76746 13.7045 3.03029 13.9673C3.29709 14.2333 3.54488 14.3924 3.88529 14.5238C4.27638 14.675 4.69279 14.759
5.31029 14.7875C6.18429 14.8286 6.48988 14.8334 8.49992 14.8334C10.4585 14.8334 10.7783 14.8278 11.6895
14.7875C12.3086 14.7582 12.7258 14.675 13.1122 14.5254C13.455 14.3916 13.7043 14.2325 13.9672 13.9697C14.234
13.7029 14.3931 13.4551 14.5245 13.1147C14.6749 12.7244 14.7588 12.3072 14.7873 11.6897C14.8285 10.8157
14.8333 10.5101 14.8333 8.50004C14.8333 6.54146 14.8277 6.22162 14.7873 5.31042C14.758 4.69133 14.6749
4.27333 14.5245 3.887C14.4052 3.564 14.215 3.27184 13.968 3.032C13.7287 2.78409 13.4364 2.59355 13.113
2.47467C12.725 2.32425 12.3086 2.24112 11.6895 2.21262C10.8155 2.17146 10.51 2.16671 8.49992 2.16671ZM8.49992
0.583374C10.6509 0.583374 10.9193 0.591291 11.7632 0.630874C12.6063 0.670457 13.1803 0.802666 13.6853
0.998999C14.2078 1.20008 14.648 1.47242 15.0882 1.91179C15.4907 2.30754 15.8022 2.78626 16.001
3.31462C16.1965 3.81892 16.3295 4.39367 16.3691 5.23679C16.4063 6.08071 16.4166 6.34908 16.4166
```

```
8.50004C16.4166 10.651 16.4087 10.9194 16.3691 11.7633C16.3295 12.6064 16.1965 13.1804 16.001 13.6855C15.8028
14.2141 15.4912 14.6929 15.0882 15.0883C14.6923 15.4907 14.2136 15.8022 13.6853 16.0011C13.181 16.1966
12.6063 16.3296 11.7632 16.3692C10.9193 16.4064 10.6509 16.4167 8.49992 16.4167C6.34896 16.4167 6.08059
16.4088 5.23667 16.3692C4.39354 16.3296 3.81959 16.1966 3.3145 16.0011C2.78593 15.8028 2.30713 15.4912
1.91167 15.0883C1.50903 14.6926 1.19753 14.2139 0.998877 13.6855C0.802544 13.1812 0.670335 12.6064 0.630752
11.7633C0.593544 10.9194 0.583252 10.651 0.583252 8.50004C0.583252 6.34908 0.591169 6.08071 0.630752
5.23679C0.670335 4.39287 0.802544 3.81971 0.998877 3.31462C1.19698 2.78593 1.50855 2.30709 1.91167
1.91179C2.30724 1.50902 2.78601 1.19749 3.3145 0.998999C3.81959 0.802666 4.39275 0.670457 5.23667
0.630874C6.08059 0.593666 6.34896 0.583374 8.49992 0.583374Z"
                    fill="#252525"
                  />
                </svg>
              </a>
              <a target="_blank" aria-label="telegram" href="https://telegram.org/">
                <svg
                  width="17"
                  height="17"
                  viewBox="0 0 17 17"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg">
                  <path
                    d="M8.49992 14.8334C10.1796 14.8334 11.7905 14.1661 12.9783 12.9784C14.166 11.7907
14.8333 10.1797 14.8333 8.50004C14.8333 6.82034 14.166 5.20943 12.9783 4.0217C11.7905 2.83397 10.1796 2.16671
8.49992 2.16671C6.82021 2.16671 5.20931 2.83397 4.02158 4.0217C2.83385 5.20943 2.16659 6.82034 2.16659
8.50004C2.16659 10.1797 2.83385 11.7907 4.02158 12.9784C5.20931 14.1661 6.82021 14.8334 8.49992
14.8334ZM8.49992 16.4167C4.12754 16.4167 0.583252 12.8724 0.583252 8.50004C0.583252 4.12767 4.12754 0.583374
8.49992 0.583374C12.8723 0.583374 16.4166 4.12767 16.4166 8.50004C16.4166 12.8724 12.8723 16.4167 8.49992
16.4167ZM6.03784 9.42629L4.06025 8.80958C3.63275 8.67896 3.63038 8.38446 4.15604 8.17308L11.8613
5.19642C12.3086 5.01433 12.562 5.24471 12.4171 5.82183L11.1053 12.0127C11.0135 12.4536 10.7483 12.5589
10.3801 12.3555L8.36059 10.8608L7.41929 11.7696C7.32271 11.863 7.24434 11.943 7.0955 11.9628C6.94746 11.9834
6.82554 11.939 6.73609 11.6936L6.04734 9.42075L6.03784 9.42629Z"
                    fill="#252525"
                  />
                </svg>
              </a>
            </li>
            <li>+38(073) 096 36 44</li>
            <li>
              <a target="_blank" href="mailto:info@yanki.com">
                info@yanki.com
              </a>
            </li>
          </ul>
        </div>
      </div>
      <div className="footer__bottom-text">© 2022 Yanki. All rights reserved</div>
    </div>
  </footer>
 );
};

export default Footer;
```

CatalogCategories.tsx

```tsx
import React from 'react';

type CatalogCategoriesProps = {
  category: number;
  onClickChangeCategory: (index: number) => void;
};

const categories = ['Усі', 'Куртки', 'Костюми', 'Пальта', 'Парки', 'Плащі', 'Шуби'];

const CatalogCategories: React.FC<CatalogCategoriesProps> = ({
  category,
  onClickChangeCategory,
}) => {
  return (
    <div className="catalog__categories">
      <ul className="catalog__category">
        {categories.map((categoryName, index) => (
          <li
            key={index}
            onClick={() => onClickChangeCategory(index)}
            className={category === index ? 'active' : ''}>
            {categoryName}
          </li>
        ))}
      </ul>
    </div>
  );
};

export default CatalogCategories;
```

Sort.tsx

```tsx
import React from 'react';
import { useDispatch } from 'react-redux';

import { setSort } from '../../redux/filter/slice';

import { SortPropertyEnum } from '../../redux/filter/types';
import { List, SortProps } from './types';

const list: List[] = [
  {
    name: 'популярністю',
    sortProperty: SortPropertyEnum.RATING,
  },
  {
    name: 'ціною (зростання)',
    sortProperty: SortPropertyEnum.PRICE_ASC,
  },
  {
    name: 'ціною (падіння)',
    sortProperty: SortPropertyEnum.PRICE_DESC,
  },
  {
    name: 'алфавітом',
    sortProperty: SortPropertyEnum.TITLE,
  },
];

const Sort: React.FC<SortProps> = ({ value }) => {
  const dispatch = useDispatch();

  const [visible, setVisible] = React.useState(false);
  const sortRef = React.useRef<HTMLDivElement>(null);

  const clickOnPopUpItem = (obj: List) => {
    dispatch(setSort(obj));
    setVisible(false);
  };

  React.useEffect(() => {
    const handleClickOutside = (event: MouseEvent) => {
      if (sortRef.current && !event.composedPath().includes(sortRef.current)) {
        setVisible(false);
      }
    };

    document.body.addEventListener('click', handleClickOutside);

    return () => document.body.removeEventListener('click', handleClickOutside);
  }, []);
```

```
  return (
    <>
      <div ref={sortRef} onClick={() => setVisible(!visible)} className="catalog__filter">
        Сортувати за
        <svg
          className={visible ? 'svg-rotate' : ''}
          width="10"
          height="7"
          viewBox="0 0 10 7"
          fill="none"
          xmlns="http://www.w3.org/2000/svg">
          <path
            d="M5 3.88906L8.88906 -4.85606e-08L10 1.11094L5 6.11093L-4.85606e-08 1.11094L1.11094 -3.88553e-
07L5 3.88906Z"
            fill="#E0BEA2"
          />
        </svg>
        {visible && (
          <div className="catalog__popup popup">
            <ul className="popup__items">
              {list.map((obj, i) => (
                <li
                  className={value.sortProperty === obj.sortProperty ? 'active' : ''}
                  onClick={() => clickOnPopUpItem(obj)}
                  key={i}>
                  {obj.name}
                </li>
              ))}
            </ul>
          </div>
        )}
      </div>
    </>
  );
};

export default Sort;
```

CatalogBlock.tsx

```tsx
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { Link } from 'react-router-dom';

import { addFavoriteItems } from '../../redux/favorite/slice';
import { FavoriteItem } from '../../redux/favorite/types';
import { RootState } from '../../redux/store';

type CatalogBlockProps = {
  id: string;
  imageUrl: string;
  title: string;
  sizes: string[];
  price: number;
};

const CatalogBlock: React.FC<CatalogBlockProps> = ({ id, imageUrl, title, sizes, price }) => {
  const dispatch = useDispatch();

  const onClickAddFavorite = () => {
    const favoriteItem: FavoriteItem = {
      id,
      title,
      price,
      imageUrl,
      count: 0,
      sizes: [],
    };
    dispatch(addFavoriteItems(favoriteItem));
  };

  const { favorites } = useSelector((state: RootState) => state.favorite);
  const isMountedFavorites = React.useRef(false);

  React.useEffect(() => {
    if (isMountedFavorites.current) {
      const dataFavorites = JSON.stringify(favorites);
      localStorage.setItem('favorites', dataFavorites);
    }
    isMountedFavorites.current = true;
  }, [favorites]);

  return (
    <>
      <div className="item__body">
        <div className="item__img">
          <Link to={`/product/${id}`}>
            <img src={imageUrl} alt="catalog" />
          </Link>
```

```
          <button aria-label="favoriteBtn" className="item__favorite" onClick={onClickAddFavorite}>
            <svg
              width="15"
              height="15"
              viewBox="0 0 15 15"
              fill="none"
              xmlns="http://www.w3.org/2000/svg">
              <path
                d="M7.50061 1.89671C9.26233 0.314993 11.9848 0.367492 13.682 2.06771C15.3785 3.76868 15.437
6.47763 13.859 8.24459L7.49911 14.6135L1.14073 8.24459C-0.437237 6.47763 -0.377988 3.76418 1.31773
2.06771C3.01645 0.369742 5.73365 0.312743 7.50061 1.89671ZM12.62 3.12744C11.495 2.00096 9.68007 1.95521
8.50259 3.01269L7.50136 3.91118L6.49938 3.01344C5.31816 1.95446 3.50694 2.00096 2.37896 3.12894C1.26148
4.24642 1.20523 6.03514 2.23496 7.21711L7.49986 12.4903L12.7648 7.21786C13.7952 6.03514 13.739 4.24867 12.62
3.12744Z"
                fill="white"
              />
            </svg>
          </button>
        </div>
        <div className="item__info">
          <div className="item__title">{title}</div>
          <div className="item__price">{price} грн</div>
          <div className="item__sizes">{sizes}</div>
        </div>
      </div>
    </>
  );
};

export default CatalogBlock;
```

Pagination.tsx

```tsx
import React from 'react';
import ReactPaginate from 'react-paginate';

import styles from './Pagination.module.scss';

type PaginationProps = {
  onClickPageChange: (page: number) => void;
};

const Pagination: React.FC<PaginationProps> = ({ onClickPageChange }) => {
  return (
    <ReactPaginate
      className={styles.pagination}
      nextLabel={
        <svg
          width="50"
          height="24"
          viewBox="0 0 50 24"
          fill="none"
          xmlns="http://www.w3.org/2000/svg">
          <path
            d="M49.0607 13.0607C49.6464 12.4749 49.6464 11.5251 49.0607 10.9393L39.5147 1.39339C38.9289
0.807608 37.9792 0.807608 37.3934 1.3934C36.8076 1.97918 36.8076 2.92893 37.3934 3.51472L45.8787 12L37.3934
20.4853C36.8076 21.0711 36.8076 22.0208 37.3934 22.6066C37.9792 23.1924 38.9289 23.1924 39.5147
22.6066L49.0607 13.0607ZM1.31134e-07 13.5L48 13.5L48 10.5L-1.31134e-07 10.5L1.31134e-07 13.5Z"
            fill="black"
          />
        </svg>
      }
      previousLabel={
        <svg
          width="50"
          height="24"
          viewBox="0 0 50 24"
          fill="none"
          xmlns="http://www.w3.org/2000/svg">
          <path
            d="M0.939342 10.9393C0.353554 11.5251 0.353554 12.4749 0.939342 13.0607L10.4853 22.6066C11.0711
23.1924 12.0208 23.1924 12.6066 22.6066C13.1924 22.0208 13.1924 21.0711 12.6066 20.4853L4.12132 12L12.6066
3.51472C13.1924 2.92893 13.1924 1.97919 12.6066 1.3934C12.0208 0.807611 11.0711 0.807611 10.4853
1.3934L0.939342 10.9393ZM50 10.5L2 10.5V13.5L50 13.5V10.5Z"
            fill="black"
          />
        </svg>
      }
      onPageChange={(event) => onClickPageChange(event.selected + 1)}
      pageRangeDisplayed={6}
      pageCount={3}
    />
```

```
  );
};

export default Pagination;
```

ProductCard.tsx

```tsx
import React from 'react';
import { useParams } from 'react-router-dom';
import { useSelector } from 'react-redux';

import { Footer, Header, ProductBlock } from '../components';
import { RootState, useAppDispatch } from '../redux/store';
import { fetchCardItems } from '../redux/card/asyncActions';

const ProductCard: React.FC = () => {
  const { id } = useParams<{ id?: string }>();

  const { items, status } = useSelector((state: RootState) => state.card);

  const dispatch = useAppDispatch();

  React.useEffect(() => {
    const getCardInfo = async () => {
      const cardId = id ? id : '';

      dispatch(
        fetchCardItems({
          id: cardId,
        }),
      );
    };

    getCardInfo();
    window.scrollTo(0, 0);
  }, [dispatch, id]);

  const cardInfo = <ProductBlock {...items} />;
  const gif = <img className="card__gif" alt="loader" src={require('../assets/img/loader.gif')} />;

  return (
    <>
      <Header />
      <section className="card">
        <div className="container">
          {status === 'rejected' ? (
            <div className="card__error-info">
              <h2>Виникла помилка 😢</h2>
              <p>На жаль, дана продукція відсутня на нашому сайті!</p>
            </div>
          ) : (
            <>{status === 'pending' ? gif : cardInfo}</>
          )}
        </div>
      </section>
      <Footer />
```

```
    </>
  );
};

export default ProductCard;
```

Favorites.tsx

```tsx
import React from 'react';
import { useSelector } from 'react-redux';

import { FavoritesBlock, FavoritesEmpty, Footer, Header } from '../components';
import { RootState } from '../redux/store';

const Favorites: React.FC = () => {
  window.scrollTo(0, 0);

  const { favorites } = useSelector((state: RootState) => state.favorite);
  const isMountedFavorites = React.useRef(false);

  React.useEffect(() => {
    if (isMountedFavorites.current) {
      const dataFavorites = JSON.stringify(favorites);
      localStorage.setItem('favorites', dataFavorites);
    }
    isMountedFavorites.current = true;
  }, [favorites]);

  return (
    <>
      <Header />
      {favorites.length === 0 ? (
        <FavoritesEmpty />
      ) : (
        <section className="favorites">
          <div className="container">
            <div className="favorites__title">Обране</div>
            <div className="favorites__row">
              {favorites.map((item) => (
                <FavoritesBlock key={item.id} {...item} />
              ))}
            </div>
          </div>
        </section>
      )}
      <Footer />
    </>
  );
};

export default Favorites;
```

FavoritesEmpty.tsx

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

import favorites from '../../assets/img/favorites.svg';

const FavoritesEmpty: React.FC = () => {
  return (
    <>
      <div className="favorites__empty">
        <div className="container">
          <h2>Обране</h2>
          <img src={favorites} alt="Empty favorites" />
          <p>
            В ОБРАНОМУ НЕМАЄ ТОВАРІВ
            <br />
            ТИЦЯЙ ♥ ЩОБ НЕ ВТРАТИТИ
          </p>
          <Link to="/catalog" className="favorites__button-back">
            <span>Перейти до каталогу</span>
          </Link>
        </div>
      </div>
    </>
  );
};
export default FavoritesEmpty;
```

CartItem.tsx

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

import { minusItems, plusItems, removeItems } from '../../redux/cart/slice';

type CartItemProps = {
  id: string;
  newId: number;
  title: string;
  price: number;
  imageUrl: string;
  size: [];
  sizes: [];
  count: number;
};

const CartItem: React.FC<CartItemProps> = ({ id, newId, title, imageUrl, price, size, count }) => {
  const dispatch = useDispatch();

  const onClickPlus = () => {
    dispatch(plusItems(newId));
  };

  const onClickMinus = () => {
    dispatch(minusItems(newId));
  };

  const onClickRemove = () => {
    dispatch(removeItems(newId));
  };

  return (
    <>
      <div className="cart__order">
        <div className="cart__column">
          <div className="cart__item">
            <div className="cart__img">
              <img src={imageUrl} alt="" />
            </div>
            <div className="cart__info">
              <Link to={`/product/${id}`} className="cart__name">
                {title}
              </Link>
              <div className="cart__size">Розмір: {size}</div>
              <div className="cart__amount">
                <button
                  aria-label="minusBtn"
                  disabled={count === 1}
```

```
                  className="cart__minus"
                  onClick={onClickMinus}>
                  <svg
                    width="20"
                    height="20"
                    viewBox="0 0 10 2"
                    fill="none"
                    xmlns="http://www.w3.org/2000/svg">
                    <path
                      d="M10 0.285706H5.71429H4.28571H0V1.71428H4.28571H5.71429H10V0.285706Z"
                      fill="#E0BEA2"></path>
                  </svg>
                </button>
                <div className="cart__number">{count}</div>
                <button aria-label="plusBtn" className="cart__plus" onClick={onClickPlus}>
                  <svg
                    width="20"
                    height="20"
                    viewBox="0 0 10 10"
                    fill="none"
                    xmlns="http://www.w3.org/2000/svg">
                    <path
                      d="M4.28571
4.28571V0H5.71429V4.28571H10V5.71429H5.71429V10H4.28571V5.71429H0V4.28571H4.28571Z"
                      fill="#E0BEA2"></path>
                  </svg>
                </button>
              </div>
            </div>
          </div>
        </div>
        <div className="cart__column">
          <div className="cart__block">
            <button aria-label="trashBtn" className="cart__trashbin" onClick={onClickRemove}>
              <svg
                width="25"
                height="25"
                viewBox="0 0 25 25"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M18.75 5H25V7.5H22.5V23.75C22.5 24.0815 22.3683 24.3995 22.1339 24.6339C21.8995 24.8683
21.5815 25 21.25 25H3.75C3.41848 25 3.10054 24.8683 2.86612 24.6339C2.6317 24.3995 2.5 24.0815 2.5
23.75V7.5H0V5H6.25V1.25C6.25 0.918479 6.3817 0.600537 6.61612 0.366116C6.85054 0.131696 7.16848 0 7.5
0H17.5C17.8315 0 18.1495 0.131696 18.3839 0.366116C18.6183 0.600537 18.75 0.918479 18.75 1.25V5ZM20
7.5H5V22.5H20V7.5ZM8.75 11.25H11.25V18.75H8.75V11.25ZM13.75 11.25H16.25V18.75H13.75V11.25ZM8.75
2.5V5H16.25V2.5H8.75Z"
                  fill="#E0BEA2"
                />
              </svg>
            </button>
            <div className="cart__price">{price * count} грн</div>
```

```
          </div>
        </div>
      </div>
      <div className="cart__hr"></div>
    </>
  );
};

export default CartItem;
```

CartEmpty.tsx

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

import emptyCart from '../../assets/img/empty-cart.png';

const CartEmpty: React.FC = () => {
  return (
    <>
      <div className="cart__empty">
        <div className="container">
          <h2>Кошик пустий</h2>
          <img src={emptyCart} alt="Empty cart" />
          <p>
            Скоріше за все, ви ще нічого не додали до кошику.
            <br />
            Для того, щоб це зробити, перейдіть до каталогу.
          </p>
          <Link to="/catalog" className="cart__button-back">
            <span>Перейти до каталогу</span>
          </Link>
        </div>
      </div>
    </>
  );
};

export default CartEmpty;
```