Форма № Н-9.02

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова
Херсонський навчально-науковий інститут

Кафедра інформаційних технологій та фізико-математичних дисциплін

«Допущений до захисту»
Завідувач кафедри
д.т.н., доцент Гучек П.Й.
«22» червня 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи на здобуття ступеня вищої освіти «бакалавр»

на тему: “ Розробка вебсайту інтернет-магазину одягу «Inside out» ”

Виконав: студентка IV курсу, групи 4157
спеціальності
121 - Інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми
Інженерія програмного забезпечення
(назва освітньої програми)
Козаченко Т.Ю.
(прізвище та ініціали)

Керівник Карпова С.О.
(прізвище та ініціали)

Рецензент Притула В.М.
(прізвище та ініціали)

м. Миколаїв-Херсон – 2024 р.

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова
ХЕРСОНСЬКИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

Кафедра інформаційних технологій та фізико-математичних дисциплін
Галузь знань 12 "Інформаційні технології"
(шифр і назва)
Спеціальність 121 "Інженерія програмного забезпечення"
(шифр і назва)
Освітня програма "Інженерія програмного забезпечення"
(назва)

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ Літвінова М.Б.
" 17 " березня 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Козаченко Тетяні Юріївні
(прізвище, ім'я, по батькові)
1. Тема кваліфікаційної роботи "Розробка вебсайту інтернет-магазину одягу «Inside out»"

Керівник роботи Карпова Світлана Олегівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені розпорядженням по інституту від "04" березня 2024 року № 01
2. Строк подання студентом роботи 03.06.2024 р.
3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1 Титульний аркуш, ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ (ДИПЛОМНУ) РОБОТУ, АНОТАЦІЯ (українською, російською, англійською), ЗМІСТ, ПЕРЕЛІК УМОВНИХ ОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ (при необхідності).
4.2 ВСТУП
4.3 ОПИС ПРЕДМЕТНОЇ ГАЛУЗІ, ПОСТАНОВКА ЗАДАЧІ
4.4 ПРОЄКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (ПЗ)(зовнішні специфікації ПЗ, стани програми при виконанні, структура програми, структура даних, потоки даних та управління, проєкт ПЗ на рівні програмних модулів, випробування ПЗ)
4.5 РЕЗУЛЬТАТИ РОЗРОБКИ
4.6 РОЗДІЛ З ОХОРОНИ ПРАЦІ (ОХОРОНИ ОТОЧУЮЧОГО СЕРЕДОВИЩА)
4.7 ВИСНОВКИ
4.8 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ
4.9 ДОДАТКИ (технічне завдання, опис програми, текст програми, програма та методика випробувань ПЗ, інструкція користувача)

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

5.1	Діаграма варіантів використання вебсайту "Inside out"
5.2	Концептуальна модель даних для системи веб-сайту "Inside out"
5.3	Діаграма класів серверної частини "Inside out"
5.4	Діаграма послідовності логіки дій вебсайту
5.5	Логічна модель даних вебсайту
5.6	Діаграма компонентів
5.7	Діаграма результатів тестів
5.8	Результати розробки вебсайту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4.1-4.3	Дудченко О.М., декан		
4.4-4.5	Латанська Л.О., доцент		
4.6	Щедролосєв О.В., професор		

7. Дата видачі завдання 25 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Опис та аналіз предметної галузі	28.03.2024 р.	
2.	Постановка та формалізація задачі	06.04.2024 р.	
3.	Ескізний проєкт програмного забезпечення	13.04.2024 р.	
4.	Технічний проєкт програмного забезпечення	20.04.2024 р.	
5.	Робочий проєкт програмного забезпечення	27.04.2024 р.	
6.	Випробування програмного забезпечення	04.05.2024 р.	
7.	Розробка робочої документації	11.05.2024 р.	
8.	Виконання графічних документів	13.05.2024 р.	
9.	Вирішення питань охорони праці	18.05.2024 р.	
10.	Оформлення пояснювальної записки	25.05.2024 р.	
11.	Подання кваліфікаційної роботи на попередній захист	03.06.2024 р.	
12.	Подання кваліфікаційної роботи рецензенту	06.06.2024 р	
13.	Подання на кафедру ІТ та ФМД тексту остаточного варіанту роботи, підписаного її керівником, разом з заявою щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи	08.06.2024 р.	
14.	Подання на кафедру ІТ та ФМД електронних версії наступних документів у форматі pdf: кваліфікаційної (дипломної) роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК від 19.05.2020 р. за №287-уч); презентації доповіді; письмового відгуку та рецензії на кваліфікаційну роботу	14.06.2024 р.	
15	Подання на кафедру ІТ ФМД письмового відгуку та рецензії на кваліфікаційну роботу	22.06.2024 р.	

Студент _____ Козаченко Т.Ю.
Підпис Прізвище та ініціали

Керівник роботи _____ Карпова С.О.
Підпис Прізвище та ініціали

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці вебсайту «Inside out» для електронної комерції.

Проаналізовані різні інтернет-магазини одягу та розроблений власний інтернет-магазин, з такими функціями:

- реєстрація та авторизація для клієнтів та адміністраторів;
- перегляд товарів на сайті клієнтами;
- додавання та видалення товарів з кошику для клієнтів;
- оформлення замовлень для клієнтів;
- управління товарами для адміністраторів;
- перегляд замовлень для адміністраторів.

Робота викладена українською мовою і виконана на 289 аркушах, містить 76 рисунків, 12 таблиць, 5 додатків. Список використаних джерел містить 13 найменувань.

Ключові слова: Інтернет-магазин, веб-сайт, програмне забезпечення, сучасні технології, одяг.

ABSTRACT

The qualification work is devoted to the development of the "Inside out" website for e-commerce. Analyzed various online clothing stores and developed own online store with the following features:

- Registration and authorization for clients and administrators;
- Viewing of goods on the site by customers;
- Adding and removing products from the cart for customers;
- Placing orders for clients;
- Product management for administrators;
- View orders for administrators.

The work is presented in Ukrainian and performed on 289 sheets, contains 76 figures, 12 tables, 5 appendices. The list of used sources contains 13 names.

Keywords: online store, website, software, modern technologies, clothing.

ЗМІСТ

ВСТУП

У сучасному цифровому світі електронна комерція відіграє ключову роль у поведінці споживачів. За останні пару десятиліть зростання популярності онлайн-покупок призвело до змін у способах купівлі товарів і послуг. Одним із найдинамічних сегментів цього ринку є онлайн-продаж одягу.

Розробка інтернет-магазину одягу має значний потенціал для інноваційного розвитку та задовольняє потреби сучасного споживача. Ринок онлайн-продажу одягу продовжує стрімко розвиватися, залучаючи все більше споживачів. Зручність, доступність і різноманітність асортименту роблять покупки одягу онлайн привабливим і зручним способом шопінгу. Як наслідок, розробка ефективного та зручного інтернет-магазину одягу має великий потенціал для успіху на ринку електронної комерції. Однак у світлі зростання конкуренції в цьому сегменті важливо розробляти та впроваджувати якісні та інноваційні рішення, щоб задовольнити потреби сучасних споживачів і створити позитивний досвід покупки.

Метою дипломної робота є дослідити процес розробки інтернет-магазину одягу з урахуванням сучасних технологій та вимог електронної комерції.

Основні аспекти які розглядаються, включають аналіз потреб споживачів, функціональних і дизайнерських вимог магазину, вибір технологічного стеку для реалізації проєкту, а також тестування та впровадження готового продукту.

Тема розвитку інтернет-магазину одягу є дуже актуальною через її значний потенціал в сучасних умовах електронної комерції.

Актуальність дипломної роботи:

- зростання електронної комерції – споживачі все частіше звертаються до онлайн-покупок, популярність торгівлі в інтернеті створює хороші умови для розвитку онлайн магазинів;

- зручність для споживачів - клієнтам не потрібно виходити з дому без зайвої необхідності, купуючи все в інтернет-магазинах, можна легко порівнювати

товари, наприклад за ціною, вибирати з великого асортименту, що змушує віддавати перевагу шопінгу онлайн;

- глобальний доступ - онлайн магазини це платформи, які мають змогу працювати, незважаючи на кордони, та доставляти продукцію клієнтам в кожний куток планети;

- пандемія COVID-19 - з настанням пандемії онлайн торгівля стала дуже популярною через обмеження та заборони пересування та контактування з іншими людьми.

Отже, швидке зростання електронної комерції призводить до необхідності створення різних платформ для продажу та надання послуг, як от продаж одягу чи доставці їжі. Розробка інтернет-магазину одягу актуальне питання у сучасних реаліях світу, що відкриває широкі перспективи для бізнесу та брендів привабити нових клієнтів, забезпечивши зручний інтерфейс для покупок, тим самим підвищити продажі. Для розробників важливо створювати ефективні, зручні та цікаві рішення, щоб покупці залишалися задоволені досвідом відвідування сервісу.

У дипломній роботі проведено дослідження та для інтернет-магазину одягу. Проєкт розроблений з урахуванням сучасних вимог та технологій. Очікуються результати, які вирішають завдання створення конкурентоспроможного та ефективного продукту.

1 ОПИС ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБСАЙТІВ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ САЙТУ ІНТЕРНЕТ-МАГАЗИНУ ОДЯГУ

1.1 Опис предметної галузі

Вебсайт - це сукупність взаємопов'язаних веб-сторінок, доступних через Інтернет, які об'єднані спільною темою, дизайном та структурою. Вебсайти відіграють важливу роль у сьогоденні, забезпечуючи доступ до інформації, товарів, послуг, а також полегшують комунікацію між людьми та організаціями. Вебресурси стали невід'ємною частиною повсякденного життя, що впливають на різні аспекти суспільства, економіки та культури.

Існує кілька типів вебсайтів, кожен з типів виконує свою функцію та має свої особливості:

1) Інформаційні вебсайти призначені для надання користувачам корисної та актуальної інформації. Вони можуть включати портали новин, блоги, енциклопедії або довідники. Такі вебсайти зазвичай мають чітку структуру, яка дозволяє легко знаходити потрібну інформацію.

2) Комерційні вебсайти призначені для продажу товарів і послуг. Це можуть бути інтернет-магазини, корпоративні сайти компаній та платформи електронної комерції. Основні цілі таких вебсайтів – це залучення клієнтів та підвищення рівня продажів.

3) Освітні вебсайти забезпечують доступ до навчальних матеріалів і курсів. Це і онлайн-курси, і освітні платформи, шкільні та університетські портали. Ці вебсайти сприяють розвитку та підвищенню рівня знань.

4) Соціальні мережі дають можливість користувачам спілкуватися та ділитися інформацією з друзями та колегами. Це такі платформи, як Facebook, Instagram, Twitter та LinkedIn. Соціальні мережі сприяють глобальній комунікації та обміну ідеями.

Ефективний вебсайт повинен мати декілька ключових елементів, які забезпечують його успішність та привабливість для користувачів, такі як:

1) Дизайн і юзабіліті. Дизайн вебсайту відіграє важливу роль у першому враженні користувачів. Він має бути привабливим, зручним і адаптивним до різних пристроїв. Юзабіліті забезпечує легкість навігації та використання вебсайту, що є критичним для залучення та утримання відвідувачів.

2) Контент. Вміст вебсайту має бути якісним, інформативним та регулярно оновлюваним. Важливо забезпечити високу якість текстів, зображень, відео та інших матеріалів. Контент має відповідати потребам цільової аудиторії та бути корисним для неї.

3) Функціональність. Вебсайт має бути технічно бездоганним. Це включає швидкість завантаження сторінок, інтерактивні елементи, безпеку даних користувачів та адаптивність до різних пристроїв. Важливо забезпечити, щоб вебсайт працював без збоїв та надавав усі необхідні функції для користувачів.

Створення вебсайту включає використання різних технологій та інструментів, які забезпечують його розробку та функціонування:

1) HTML/CSS. Основні мови розмітки та стилізації веб-сторінок, які забезпечують структуру та зовнішній вигляд вебсайту.

2) JavaScript. Мова програмування, яка додає інтерактивність та динамічність вебсайтам. З її допомогою реалізуються різні функціональні елементи, такі як анімації, форми зворотного зв'язку та інше.

3) Системи управління контентом (CMS). Інструменти, які дозволяють легко створювати, редагувати та керувати вмістом вебсайту. Найпопулярнішими CMS є WordPress, Joomla та Drupal.

4) Бекенд-технології. Серверні мови програмування, такі як PHP, Python, Ruby та інші, які забезпечують функціональність вебсайту з боку сервера.

Основні характеристики інтернет-магазинів одягу:

1) Зовнішній вигляд – дизайн інтернет-магазинів одягу має бути привабливим, сучасним і відповідати бренду. Чітке розташування елементів,

зручна навігація та адаптивний дизайн (пристосування до різних пристроїв) є критично важливими.

2) Юзабіліті – легкість у використанні є ключовим фактором успішності інтернет-магазину. Інтуїтивно зрозуміла навігація, простий процес оформлення замовлення та зручні фільтри для пошуку товарів роблять користувацький досвід приємним.

3) Кошик та оформлення замовлення – процес додавання товарів до кошика та оформлення замовлення має бути простим і зрозумілим. Важливо забезпечити можливість легко змінювати кількість товарів у кошику, видаляти або додавати нові позиції.

4) Методи оплати – підтримка різних методів оплати (кредитні картки, електронні гаманці, банківські перекази) є важливою для зручності покупців.

5) Доставка та повернення – чітка інформація про варіанти доставки та політику повернення товарів створює довіру у клієнтів.

6) Опис товарів – детальні описи товарів, які включають інформацію про матеріали, розміри, кольори та особливості догляду, допомагають покупцям приймати обґрунтовані рішення.

7) Фотографії та відео – високоякісні зображення та відео товарів, з можливістю перегляду з різних ракурсів, є важливими для візуального сприйняття продукту.

8) Відгуки клієнтів – розділ з відгуками покупців допомагає іншим користувачам робити вибір та створює довіру до магазину.

Інтернет-магазини одягу мають свої переваги та недоліки, але їхня популярність продовжує зростати завдяки зручності та широкому асортименту. Важливо забезпечувати високу якість обслуговування, зручний дизайн та функціональність вебсайту для залучення та утримання клієнтів.

Вебсайти стали невід'ємною частиною життя, впливаючи на різні сфери діяльності від бізнесу до освіти. Вони забезпечують зручний доступ до інформації, товарів та послуг, сприяють глобальній комунікації та розвитку суспільства. У майбутньому очікується подальший розвиток вебтехнологій, що

призведе до появи ще більш інтерактивних і персоналізованих вебсайтів. Це відкриє нові можливості для користувачів та розробників, забезпечуючи ще більш ефективне та зручне використання Інтернету.

1.2 Опис існуючих технологій розробки веб-сайтів

Вебсайти є невід'ємною частиною сучасного світу, забезпечуючи доступ до інформації, товарів і послуг, а також полегшуючи комунікацію. Створення та підтримка вебсайтів вимагає використання різних технологій та інструментів.

Фронтенд-технології відповідають за все, що бачить і з чим взаємодіє користувач на вебсайті.

1) HTML (HyperText Markup Language) – стандартизована мова розмітки документів для перегляду вебсторінок у браузері. Браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора [8].

2) CSS (Cascading Style Sheets) – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів [6].

3) JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки [9].

Основні технології бекенду.

Бекенд-технології забезпечують функціонування вебсайту з боку сервера, обробляючи запити користувачів та керуючи даними.

1) PHP (Hypertext Preprocessor) – серверна мова програмування, яка широко використовується для створення динамічних вебсайтів. PHP дозволяє взаємодіяти з базами даних, обробляти форми та керувати сесіями користувачів.

2) Python – універсальна мова програмування, яка все частіше використовується для веброзробки завдяки своїй простоті та потужності. Популярні фреймворки, такі як Django та Flask, дозволяють швидко створювати та розгортати веб-додатки.

3) Ruby – мова програмування, відома своєю простотою та читабельністю. Фреймворк Ruby on Rails дозволяє швидко розробляти веб-додатки, слідуючи принципам конвенцій понад конфігурацій.

4) Java – потужна мова програмування, яка використовується для створення масштабованих веб-додатків. Фреймворки, такі як Spring, забезпечують створення надійних та безпечних веб-додатків.

CMS дозволяють легко створювати, редагувати та керувати вмістом вебсайту без глибоких знань програмування.

1) WordPress – найпопулярніша CMS у світі, що використовується для створення блогів, корпоративних сайтів та інтернет-магазинів. WordPress пропонує велику кількість тем та плагінів, які дозволяють легко налаштовувати вебсайт.

2) Joomla – гнучка CMS, яка підходить для створення різноманітних типів вебсайтів. Joomla пропонує широкий вибір розширень та шаблонів для налаштування функціональності та дизайну.

3) Drupal – потужна CMS, яка використовується для створення складних та масштабованих вебсайтів. Drupal пропонує високу гнучкість та можливості налаштування для розробників.

Існують різноманітні інструменти та платформи, які спрощують процес розробки вебсайтів.

1) Git – система контролю версій, яка дозволяє розробникам відстежувати зміни в коді та співпрацювати над проєктами. GitHub та GitLab є популярними платформами для зберігання та керування репозиторіями.

2) Node.js – платформа на основі JavaScript, яка дозволяє запускати серверний код. Node.js забезпечує високу продуктивність та масштабованість для веб-додатків.

3) Webpack – інструмент для збирання модулів, який дозволяє оптимізувати ресурси вебсайту. Webpack забезпечує ефективне управління залежностями та оптимізацію продуктивності.

Сучасні технології розробки вебсайтів забезпечують широкий вибір інструментів та платформ для створення ефективних та привабливих веб-додатків. Вибір конкретних технологій залежить від вимог проєкту, навичок розробників та цільової аудиторії. У майбутньому очікується подальший розвиток вебтехнологій, що відкриє нові можливості для створення ще більш інтерактивних та персоналізованих вебсайтів.

1.3 Аналіз існуючих сайтів інтернет-магазинів одягу та обґрунтування необхідності розробки інтернет-магазину “Inside out”

Інтернет-магазини одягу стали невід'ємною частиною сучасного рітейлу.

Вони забезпечують зручний доступ до широкого асортименту одягу, взуття та аксесуарів для користувачів у всьому світі. Детально буде розглянуто три популярних інтернет-магазини одягу: ASOS, Zara та H&M, аналізуючи їхні основні характеристики, переваги та недоліки.

1) ASOS (As Seen On Screen) — один з найбільших онлайн-магазинів моди, орієнтований переважно на молодіжну аудиторію.

Дизайн сайту сучасний та мінімалістичний дизайн: ASOS має чистий і стильний інтерфейс, що сприяє приємному користувацькому досвіду.

Сайт адаптивний, добре оптимізований для різних пристроїв, включаючи смартфони та планшети, що дозволяє зручно здійснювати покупки з будь-якого гаджета.

Функціональність: різноманітність методів оплати, ASOS підтримує різні методи оплати, включаючи кредитні картки, PayPal та інші електронні платіжні системи.

Зручний процес оформлення замовлення: користувачі можуть легко додавати товари до кошика, переглядати його вміст і швидко оформлювати замовлення.

Інформація про доставку: магазин пропонує різні варіанти доставки, включаючи стандартну, експрес-доставку та міжнародну доставку.

Контент: детальні описи товарів, кожен товар має детальний опис, що включає інформацію про матеріали, розміри та рекомендації щодо догляду.

Високоякісні фотографії та відео: товари представлені через професійні зображення та відео, які дозволяють детально розглянути продукт з різних ракурсів.

Є відгуки від інших користувачів допомагають приймати рішення про покупку.

Розглянемо переваги та недоліки ASOS:

Переваги:

- Великий вибір товарів різних брендів.
- Швидка та зручна навігація.
- Часті знижки та розпродажі.

Недоліки:

- Висока вартість доставки для деяких регіонів.
- Можливі затримки у доставці під час розпродажів.

На рисунку 1.1 зображено інтерфейс ASOS.

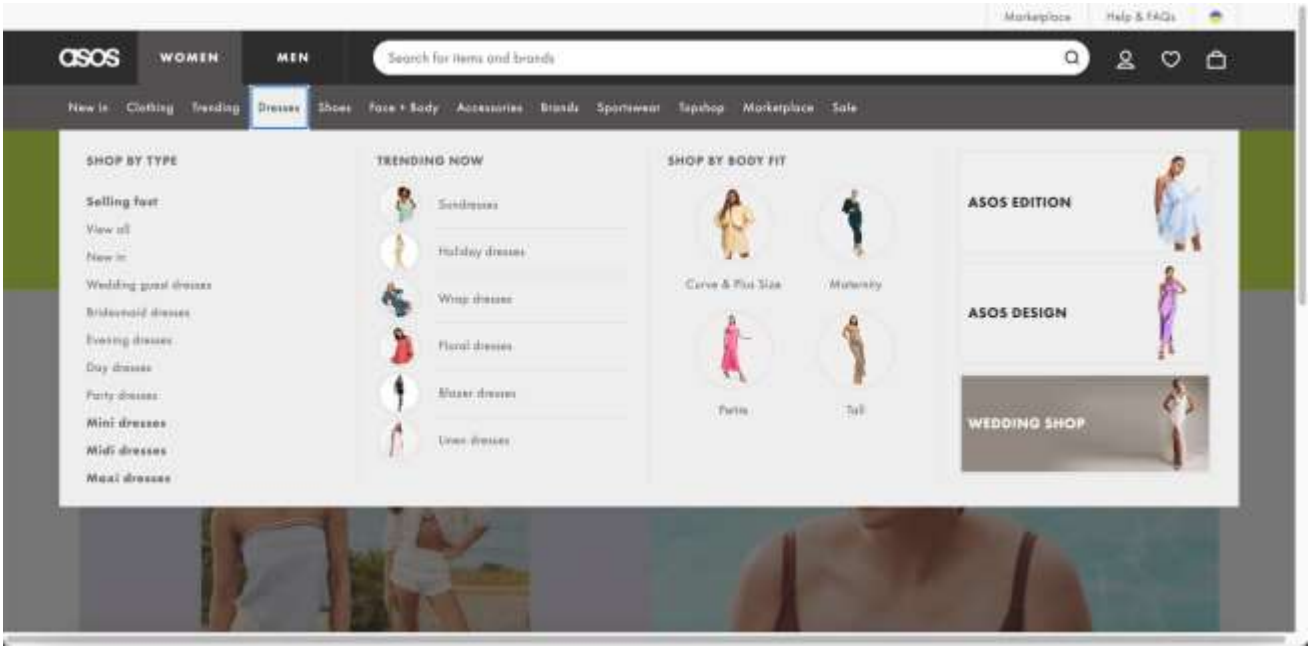

Рисунок 1.1 – Інтерфейс ASOS

2) Zara — популярний іспанський бренд одягу, відомий своїм стильним дизайном та швидким реагуванням на модні тренди.

Дизайн елегантний та стильний дизайн: Сайт Zara відображає преміум-якість бренду через свій вишуканий і елегантний дизайн.

Адаптивний сайт, добре оптимізований для мобільних пристроїв, забезпечуючи зручний доступ до товарів з будь-якого гаджета.

Функціональність: простий процес оформлення замовлення: Процес додавання товарів до кошика та оформлення замовлення є інтуїтивно зрозумілим.

Різноманітність методів оплати: Zara підтримує різні методи оплати, включаючи кредитні картки та PayPal.

Інформація про доставку та повернення: магазин надає детальну інформацію про варіанти доставки та політику повернення товарів.

Контент: детальні описи товарів, опис кожного товару включає інформацію про матеріали, розміри та інструкції щодо догляду.

Високоякісні зображення: товари представлені через професійні фотографії, що дозволяють детально розглянути продукт.

Відсутність відгуків клієнтів: Zara не надає можливості залишати відгуки, що може бути недоліком для деяких користувачів.

Переваги та недоліки магазину Zara:

Переваги:

- Стильний дизайн та якісний контент.
- Швидка реакція на модні тенденції.
- Часті оновлення асортименту.

Недоліки:

- Відсутність відгуків клієнтів.
- Висока вартість деяких товарів.

На рисунку 1.2 зображено інтерфейс Zara.

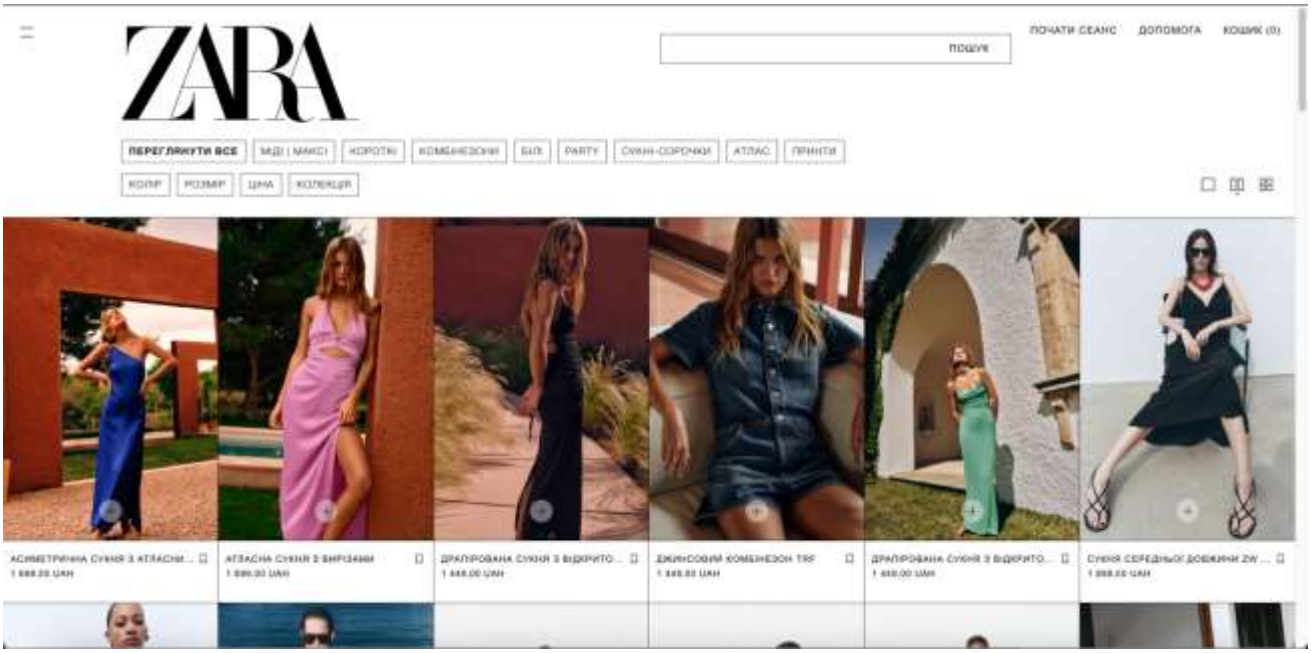

Рисунок 1.2 – Інтерфейс Zara

3) H&M (Hennes & Mauritz) — шведський бренд, відомий своїми доступними цінами та модним одягом для всієї родини.

Сайт H&M має яскравий інтерфейс, що приваблює користувачів.

Адаптивний сайт, оптимізований для мобільних пристроїв, забезпечуючи зручний доступ з різних гаджетів.

Функціональність: зручний процес оформлення замовлення, легкість у використанні кошика та оформленні замовлення.

Різноманітність методів оплати: H&M підтримує різні методи оплати, включаючи кредитні картки та PayPal.

Інформація про доставку та повернення: магазин надає чітку інформацію про варіанти доставки та політику повернення товарів.

Детальні описи товарів: опис кожного товару включає інформацію про матеріали, розміри та інструкції щодо догляду.

Високоякісні зображення: товари представлені через якісні фотографії.

Відгуки клієнтів: H&M надає можливість залишати відгуки, що допомагає іншим покупцям.

Переваги та недоліки магазину H&M.

Переваги:

- Доступні ціни.
- Широкий асортимент товарів для всієї родини.
- Можливість залишати відгуки.

Недоліки:

- Іноді обмежений асортимент у порівнянні з фізичними магазинами.
- Можливі проблеми з наявністю товарів під час розпродажів.

На рисунку 1.3 зображено інтерфейс H&M.

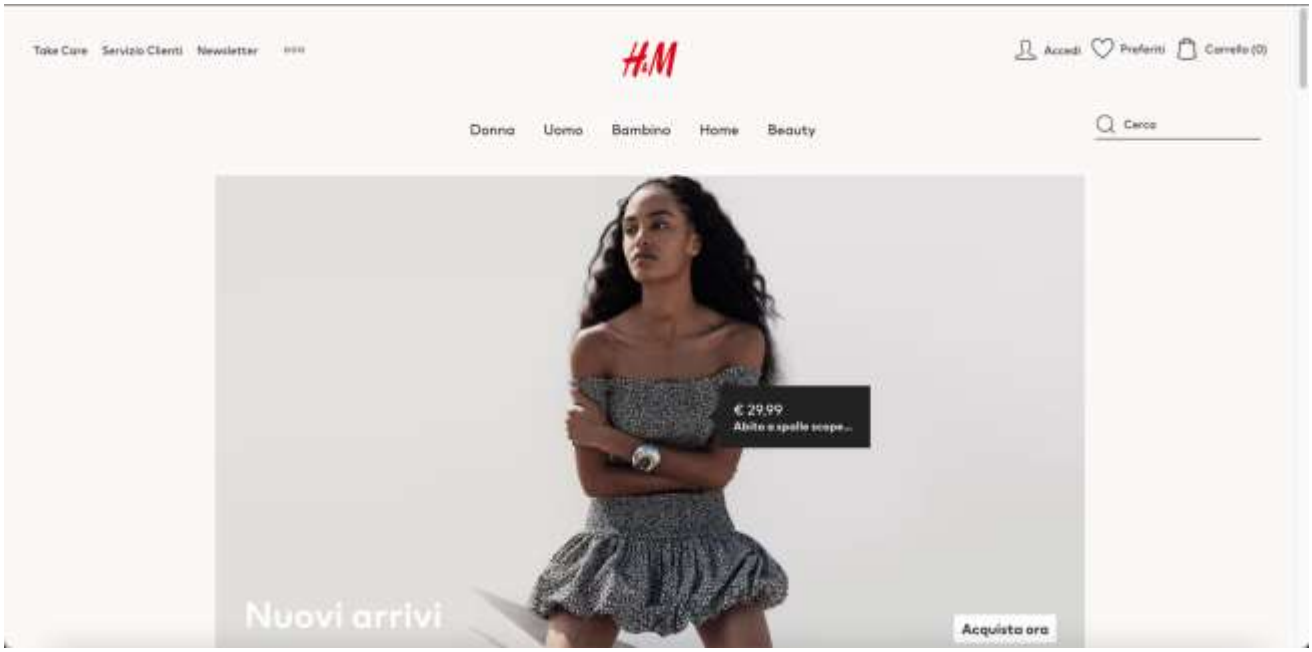

Рисунок 1.3 – Інтерфейс Zara

ASOS, Zara та H&M є трьома провідними інтернет-магазинами одягу, кожен з яких має свої особливості, переваги та недоліки. ASOS приваблює великим вибором товарів та зручністю користування, Zara виділяється стильним дизайном та швидким оновленням асортименту, а H&M пропонує доступні ціни та широкий асортимент для всієї родини. Кожен з цих магазинів продовжує вдосконалювати свої сервіси, щоб задовольнити потреби сучасних покупців та залишатися конкурентоспроможними на ринку електронної комерції.

Розробка «Inside out» є важливим етапом у розвитку програмістських навичок. Створення повноцінного веб-додатку дає можливість отримати глибокі знання та практичні вміння, які є незамінними у сучасному IT-секторі.

Процес розробки дозволяє закріпити знання з мов програмування, таких як HTML, CSS та JavaScript. Ці мови є основою фронтенд-розробки, відповідальні за створення інтерфейсу користувача. Створення структури веб-сторінок, стилізація та додавання інтерактивності включає роботу з бібліотеками та фреймворками, такими як React, що значно підвищує рівень професійної підготовки.

Крім того, здобуваються навички роботи з серверними мовами програмування, такими як Node.js, які є основою бекенд-розробки цього проєкту.

Створення серверної логіки, управління базами даних та забезпечення безпеки даних користувачів включає роботу з фреймворками, такими як Express.js, що дозволяє швидко та ефективно створювати веб-додатки.

Одним з ключових аспектів розробки інтернет-магазину є робота з базами даних. Набуття навичок створення та управління базами даних, використовуючи NoSQL бази даних, такі як MongoDB, включає проєктування структури баз даних, написання складних запитів та оптимізацію продуктивності.

Таким чином, розробка інтернет-магазину «Inside out» є необхідною з точки зору набуття та вдосконалення програмістських навичок. Вона забезпечує практичний досвід роботи з різними мовами програмування, фреймворками, базами даних. Це не тільки підвищує професійну кваліфікацію, але й робить програміста конкурентоспроможним на ринку праці, готовим до викликів сучасного IT-сектору.

1.4 Основні вимоги до проєкту

Основні функціональні вимоги до вебсайту.

1) Реєстрація та аутентифікація користувачів.

- Користувачі можуть створювати акаунт за допомогою електронної пошти та номера телефону.
- Користувачі можуть входити в систему за допомогою своїх облікових даних.

2) Головна сторінка

- На головній сторінці представлені нові колекції, акції, найпопулярніші та нові товари.
- Користувачі можуть швидко перейти до категорій або продуктів.

3) Каталог товарів.

- Відображення повного каталогу одягу з можливістю фільтрації (за назвою(в алфавітному порядку і у зворотному порядку) та за ціною(від найдешевшого до найдорожчого, і навпаки)).
- Окремі сторінки товарів для дітей, жінок, чоловіків.
- Кожен продукт повинен мати детальну інформацію, включаючи зображення, описи, доступні розміри, матеріали, ціну та відгуки.

4) Кошик.

- Користувачі можуть додавати товари до кошика.
- Можливість перегляду та видалення товарів з кошика.
- Відображення загальної суми та орієнтовних витрат на доставку.

5) Процес оформлення замовлення.

- Можливість оформлення замовлення з подальшим зв’язком з менеджером для уточнення деталей доставки і оплати.

6) Двофакторна аутентифікація (2FA).

- Можливість ввімкнення двофакторної аутентифікації для користувачів та адміністраторів.

7) Використання VPN.

- Можливість доступу до сайту з будь-якої країни.

Основні вимоги до адмін-панелі.

1) Реєстрація та аутентифікація адміністраторів.

- Створення акаунту для адміністратора, та авторизація для управління адмін-панеллю.

2) Управління товарами.

- Адміністратори можуть додавати(назва, ціна(стара і нова, та одна), категорія(чоловіки, жінки, діти), фотографія товару) або видаляти товари.

3) Перегляд замовленнями:

- Адміністратор може бачити інформацію про замовлення: ім’я, номер телефону, список замовлених товарів, вартість та дату замовлення.

Нефункціональні вимоги до системи.

1) Продуктивність.

- Сайт повинен завантажуватися швидко, і сторінки мають відкриватися без помітних затримок для користувачів. Наприклад, час завантаження головної сторінки не повинен перевищувати 2-3 секунди.

2) Масштабованість.

- Система повинна бути масштабованою для майбутнього збільшення кількості користувачів та товарів.

3) Безпека.

- Шифрування чутливих даних користувачів (паролі).
- Регулярна оцінка вразливостей.

4) Зручність.

- Зручний, інтуїтивний і зрозумілий інтерфейс як для інтернет-магазину, так і для адмін-панелі.
- Адаптивний дизайн для сумісності з мобільними пристроями.

5) Надійність.

- Впровадження процедур резервного копіювання даних(ручне копіювання та Google Drive) та відновлення після аварій.

6) Підтримка.

- Код повинен бути добре задокументований та відповідати галузевим стандартам.
- Забезпечити можливість швидкої розробки і впровадження оновлень.

7) Дизайн.

- Дизайн сайту має відзначатися своєю привабливістю та легкістю навігації.
- Логотип та кольорова палітра - сайт повинен мати унікальний логотип та відмінну кольорову схему, які підкреслюватимуть бренд та стиль.

8) Сумісність з браузерами.

- Сайт має працювати на різних веб-браузерах, таких як Chrome, Firefox, Safari, Edge і т.д., з однаковою якістю відображення.

Інтернет-магазин одягу розроблений з метою забезпечити зручний та цікавий спосіб покупок для клієнтів, доповнений корисними функціями та інформацією.

1.5 Постановка задачі на розробку вебсайту “Inside out”

З викладених вимог до програмного продукту була розроблена постанова задачі.

Головною метою даної роботи є розробка програмного забезпечення, яке відповідає наступним вимогам:

1. Вимоги до складу виконуваних функцій

Інтернет-магазин:

1) Реєстрація та аутентифікація користувачів:

Вхідна інформація: облікові дані користувача (електронна пошта, номер телефону, соціальні мережі).

Вихідна інформація: підтвердження реєстрації, успішний вхід у систему.

2) Каталог товарів:

Вхідна інформація: фільтри пошуку (за ціною, за назвою).

Вихідна інформація: список товарів, детальна інформація про товар.

3) Кошик:

Вхідна інформація: товари, які додаються до кошика, кількість товарів.

Вихідна інформація: зміст кошика, загальна сума, орієнтовні витрати на доставку.

4) Процес оформлення замовлення:

Вхідна інформація: список товарів в кошику зареєстрованого користувача.

Вихідна інформація: підтвердження замовлення.

5) Додаткові функції безпеки:

Вхідна інформація: дані для двофакторної аутентифікації (телефон, електронна пошта, додаток-аутентифікатор).

Вихідна інформація: підтвердження успішної аутентифікації.

Адмін-панель:

1) Реєстрація та аутентифікація адміністраторів:

Вхідна інформація: облікові дані користувача (електронна пошта, номер телефону).

Вихідна інформація: підтвердження реєстрації, успішний вхід у систему.

2) Управління товарами:

Вхідна інформація: інформація про товари (назва, ціна, зображення, категорії).

Вихідна інформація: оновлений каталог товарів, підтвердження додавання/видалення товарів.

2. Вимоги до контролю вхідної і вихідної інформації

Вхідною інформацією для даної системи можуть бути як події натискання клавіш клавіатури та кнопок миші, так і дані користувачів, які вводяться під час реєстрації та авторизації, а також замовлення товарів. Вихідною інформацією для

даної системи можуть бути як відповіді серверу на введені дані користувачів, так і оброблені дані, що передаються та зберігаються у вигляді замовлень.

Валідація вхідних даних включає події натискання клавіш клавіатури, події натискання кнопок миші, а також дані користувачів, такі як електронна пошта, пароль, особиста інформація та деталі замовлення. Результат валідації може бути як успішним, так і містити повідомлення про помилки, а також підтвердження реєстрації, авторизації або створення замовлення.

Обробка та зберігання даних здійснюється на основі вхідної інформації користувачів, включаючи особисту інформацію та деталі замовлення. Вихідною інформацією є оброблені дані, такі як акаунти користувачів/адміністраторів, замовлення, які зберігаються в базі даних.

Безпека даних забезпечується через обробку аутентифікаційних даних, таких як логіни, паролі. Вихідною інформацією є підтвердження аутентифікації, надання доступу до захищених ресурсів та шифровані дані.

3. Вимоги до інформаційної і програмної сумісності

Для запуску та коректної роботи інтернет-магазину та адмін-панелі необхідно виконати наступні вимоги:

- На комп'ютерах повинні бути встановлені операційні системи Windows 7 та вище, macOS 10.14 та вище, або будь-який сучасний Linux дистрибутив.
- Для коректної роботи на клієнтській частині, система повинна підтримувати основні веб-браузери, такі як Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. Це гарантує, що інтерфейс користувача буде правильно відображатися і функціонувати на різних пристроях і браузерах.
- Для забезпечення сумісності з базами даних система повинна підтримувати MongoDB версії 4.0 та вище. Також для серверної частини додатку необхідно встановити Node.js версії 14 та вище.

4. Вимоги до складу й параметрів технічних засобів

Апаратне забезпечення:

- Будь-який сучасний комп'ютер, ноутбук, планшет або смартфон з доступом до інтернету та підтримкою сучасних веб-технологій.
- Мінімальні вимоги включають процесор з частотою не менше 1.6 гігагерц, 2 гігабайт оперативної пам'яті та 512 мегабайт вільного місця на диску.

Веб-браузери:

- Підтримка основних веб-браузерів: Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. Це дозволить користувачам коректно відображати веб-сторінки на різних пристроях.

Детально постановка задачі описана в технічному завданні, що продемонстровано у Додатку А.

2 ПРОЄКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При розробці проєкту програмного забезпечення були виконані роботи з ескізного, технічного та робочого проєкту. Для цього використовувалися об'єктно-орієнтовані методології та засоби UML. Результатом роботи є розроблений вебсайт інтернет-магазин «Inside out».

2.1 Ескізний проєкт вебсайту для магазину одягу “Inside out”

Під час розробки ескізного проєкту будуть представлені такі елементи: діаграма варіантів використання, специфікація варіантів використання, діаграми діяльності, концептуальна модель, а також проєкт користувацького інтерфейсу.

2.1.1 Побудова моделі варіантів використання вебсайту для магазину одягу “Inside out”

Діаграма варіантів використання визначається як графічне представлення взаємодії між користувачами (акторами) і системою. Ілюструє різні способи взаємодії користувачів із системою для досягнення конкретних цілей або завдань. Це тип діаграми поведінки в уніфікованій мові моделювання (UML), яка широко використовується в розробці програмного забезпечення для візуалізації функціональних вимог системи. Діаграми варіантів використання забезпечують загальне уявлення про функціональність системи та залучених учасників без заглиблення у внутрішню роботу системи [12].

На рисунку 2.1 показана модель варіантів використання вебсайту «Inside out», що описує взаємодію між користувачами та системою. Актори: користувач та

адміністратор. Прецеденти користувача: реєстрація, авторизація, перегляд товарів, сортування товарів, додавання товарів у корзину, видалення товарів із корзини, оформлення замовлення, перегляд замовлень, підписка на розсилку. Прецеденти адміністратора: додавання нових товарів на сайт, видалення товарів із сайту, авторизація, реєстрація , перегляд замовлень.

Представлена діаграма демонструє основні відносини між актором та прецедентами, що дозволяє описати та продемонструвати систему на концептуальному рівні.

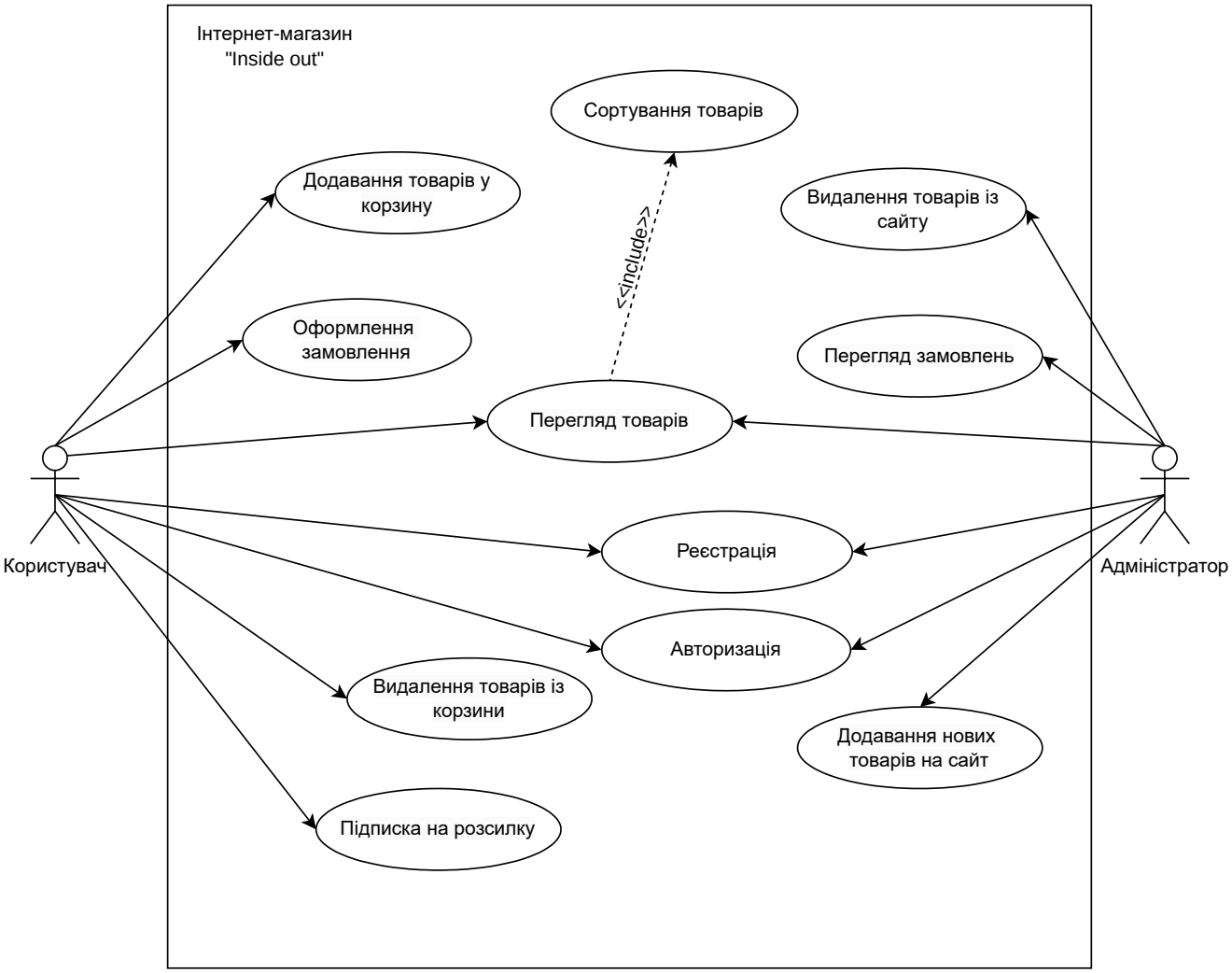

Рисунок 2.1 – Діаграма варіантів використання вебсайту «Inside out»

2.1.2 Специфікації варіантів використання вебсайту для магазину одягу «Inside out»

Для діаграми варіантів використання, що була продемонстрована в попередньому розділі, нижче будуть наведені специфікації варіантів використання.

Опис акторів

Користувач – фізична особа, яка взаємодіє з інтернет-магазином "Inside out" з метою перегляду товарів, додавання їх у корзину, оформлення замовлень, підписки на розсилки та інших дій, пов'язаних з покупками на сайті.

Адміністратор – особа, яка має адміністративні права для керування інтернет-магазином "Inside out". Адміністратор може додавати та видаляти товари, переглядати замовлення.

Опис системи

Інтернет-магазин «Inside out» – веб-платформа для продажу товарів, яка забезпечує користувачам можливість переглядати, вибирати та купувати товари онлайн. Система включає функціонал для реєстрації та авторизації користувачів, додавання товарів у корзину, оформлення замовлень, підписки на розсилки та інші функції, необхідні для зручного онлайн-шопінгу. Для адміністраторів передбачено інструменти для керування товарним асортиментом, обробки замовлень.

Специфікації:

1. Реєстрація

Основний актор: користувач.

Інші учасники: адміністратор.

Опис: користувач або адміністратор створює новий обліковий запис у системі.

Основний потік:

1. Актор вибирає опцію реєстрації.

2. Система відображає форму реєстрації.

3. Актор вводить обов'язкові дані (ім'я, електронну адресу, пароль, номер телефону).

4. Система перевіряє введені дані та реєструє актора.

Пост умова: актор зареєстрований та може використовувати свій обліковий запис для входу в систему.

Альтернативні потоки:

1. Якщо електронна адреса вже використовується, система повідомляє про це користувача та просить ввести іншу адресу.

2. Якщо користувач не заповнює всі обов'язкові поля, система відображає повідомлення про помилку.

2. Авторизація

Основний актор: користувач.

Інші учасники: адміністратор.

Опис: користувач або адміністратор вводить свої облікові дані для доступу до системи.

Основний потік:

1. Актор вводить логін та пароль.

2. Система перевіряє введені дані.

3. При успішній авторизації, актор отримує доступ до свого акаунту.

Пост умова: актор має доступ до системи під своїм обліковим записом.

Альтернативний потік: при невдалій авторизації, система відображає повідомлення про помилку.

3. Додавання товарів у корзину

Основний актор: користувач.

Інші учасники: відсутні.

Опис: користувач додає обрані товари до корзини для подальшого замовлення.

Основний потік:

1. Користувач вибирає товар.

2. Система відображає детальну інформацію про товар.

3. Користувач натискає кнопку "Додати у корзину".

4. Система додає товар у корзину користувача.

Пост умова: товар доданий у корзину користувача.

Альтернативні потоки: відсутні.

4. Оформлення замовлення

Основний актор: користувач.

Інші учасники: відсутні.

Опис: користувач оформляє замовлення на товари, що знаходяться у корзині.

Основний потік:

1. Користувач відкриває корзину.

2. Система відображає вміст корзини.

3. Користувач натискає кнопку "Оформити замовлення".

4. Система обробляє замовлення та відправляє підтвердження користувачу.

Пост умова: замовлення оформлено та підтверджено.

Альтернативні потоки: відсутні.

5. Видалення товарів із корзини

Основний актор: користувач .

Інші учасники: відсутні.

Опис: користувач видаляє товари із корзини.

Основний потік:

1. Користувач відкриває корзину.

2. Система відображає вміст корзини.

3. Користувач вибирає товар для видалення та натискає кнопку "Видалити".

4. Система видаляє обраний товар із корзини.

Пост умова: товар видалений із корзини користувача.

Альтернативні потоки: відсутні.

6. Підписка на розсилку

Основний актор: користувач.

Інші учасники: відсутні.

Опис: користувач підписується на розсилку новин та акцій.

Основний потік:

1. Користувач вписує свою електронну пошту в форму на сайті.

2. Система додає адресу користувача до списку розсилки.

3. З’являється повідомлення про успішну підписку.

Пост умова: користувач підписаний на розсилку новин та акцій.

Альтернативні потоки:

1. Якщо електронна адреса вже підписана, система повідомляє користувача про це.

2. Якщо користувач вводить некоректну адресу, система відображає повідомлення про помилку.

7. Сортування товарів

Основний актор: користувач.

Інші учасники: відсутні.

Опис: користувач сортує товари за різними критеріями.

Основний потік:

1. Користувач відкриває сторінку з товарами.

2. Система відображає варіанти сортування (за ціною, за назвою).

3. Користувач вибирає критерій сортування.

4. Система сортує товари відповідно до обраного критерію.

Пост умова: товари відсортовані за обраним критерієм.

Альтернативні потоки: відсутні.

8. Видалення товарів із сайту

Основний актор: адміністратор.

Інші учасники: відсутні.

Опис: адміністратор видаляє товари із сайту.

Основний потік:

1. Адміністратор вибирає товар для видалення.

2. Система видаляє товар із сайту.

Пост умова: товар видалений із сайту.

Альтернативні потоки: відсутні.

9. Перегляд замовлень

Основний актор: адміністратор.

Інші учасники: відсутні.

Опис: адміністратор переглядає список замовлень.

Основний потік:

1. Адміністратор вибирає опцію перегляду замовлень.

2. Система відображає список замовлень.

Пост умова: адміністратор переглянув список замовлень.

Альтернативні потоки:

1. Якщо немає замовлень, система повідомляє адміністратора про це.

10. Додавання нових товарів на сайт

Основний актор: адміністратор.

Інші учасники: відсутні.

Опис: адміністратор додає нові товари на сайт.

Основний потік:

1. Адміністратор вибирає опцію додавання нового товару.

2. Система відображає форму для введення даних товару.

3. Адміністратор заповнює форму необхідними даними (назва, ціна, фото, категорія).

4. Система перевіряє введені дані та додає товар на сайт.

Пост умова: новий товар доданий на сайт.

Альтернативні потоки:

1. Якщо введені дані некоректні або не повні, система відображає повідомлення про помилку.

2.1.3 Сценарії основних варіантів використання

Діаграми діяльності - UML-діаграма, на якій показується розкладання деякої діяльності на її складові частини. Під діяльністю розуміється специфікація виконуваного поведінки у вигляді координованого послідовного і паралельного виконання підлеглих елементів-вкладених видів діяльності і окремих дій з'єднаних між собою потоками, які йдуть від виходів одного вузла до входів іншого. Діаграми діяльності складаються з обмеженої кількості фігур, з'єднаних стрілками. Основні фігури: прямокутники з заокругленнями - дії; ромби - рішення; широкі смуги - початок (розгалуження) і закінчення (сходження) розгалуження дій; чорний коло - початок процесу (початковий стан); Чорний круг з обвідкою - закінчення процесу (кінцевий стан). Стрілки йдуть від початку до кінця процесу і показують послідовність переходів [2].

Для глибшого розуміння специфікацій в цьому розділі будуть продемонстровані діаграми діяльності варіантів використання на рисунках 2.2 – 2.11.

Рисунок 2.2 – Діаграма діяльності варіанту використання для реєстрації

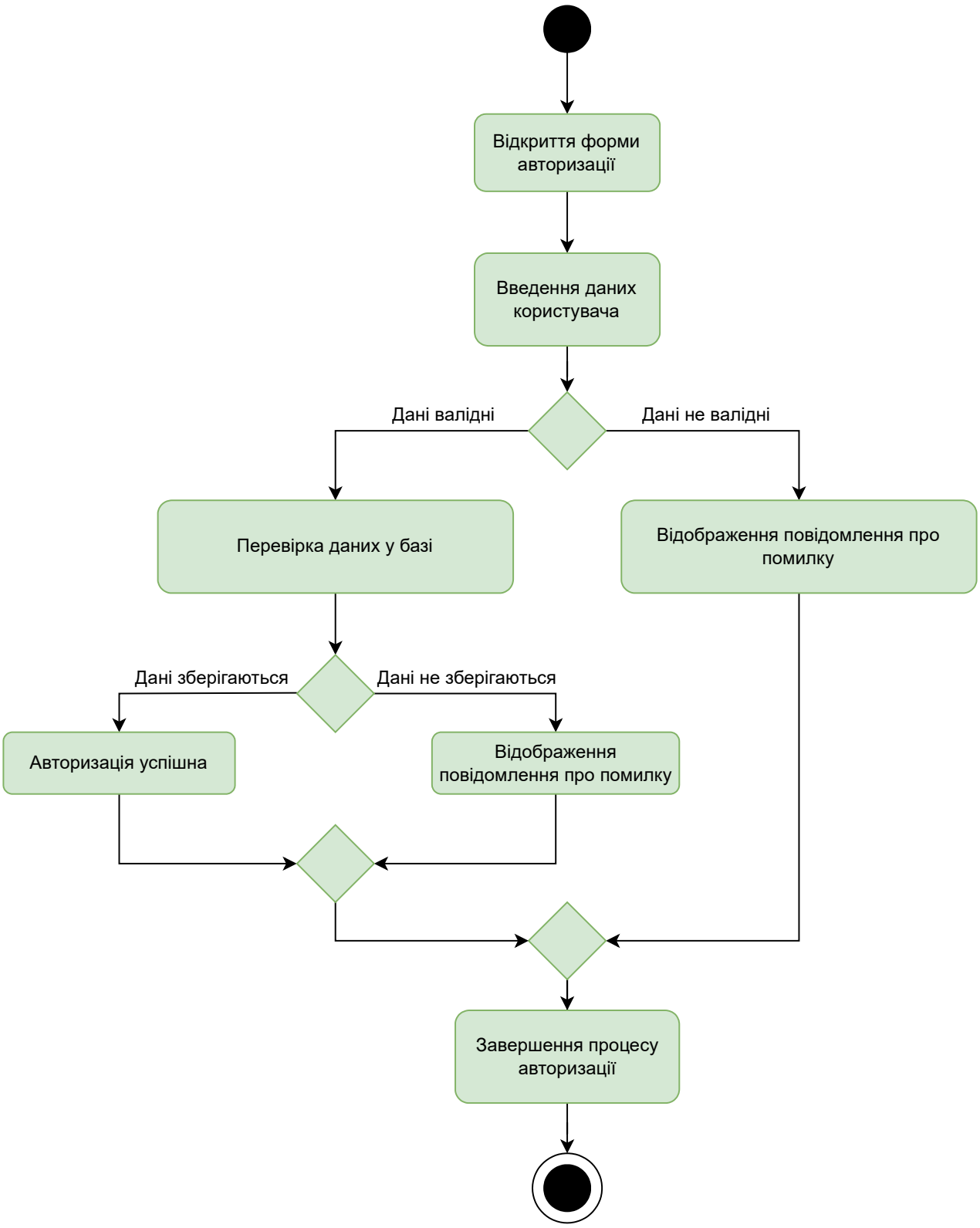

Рисунок 2.3 – Діаграма діяльності варіанту використання для авторизації

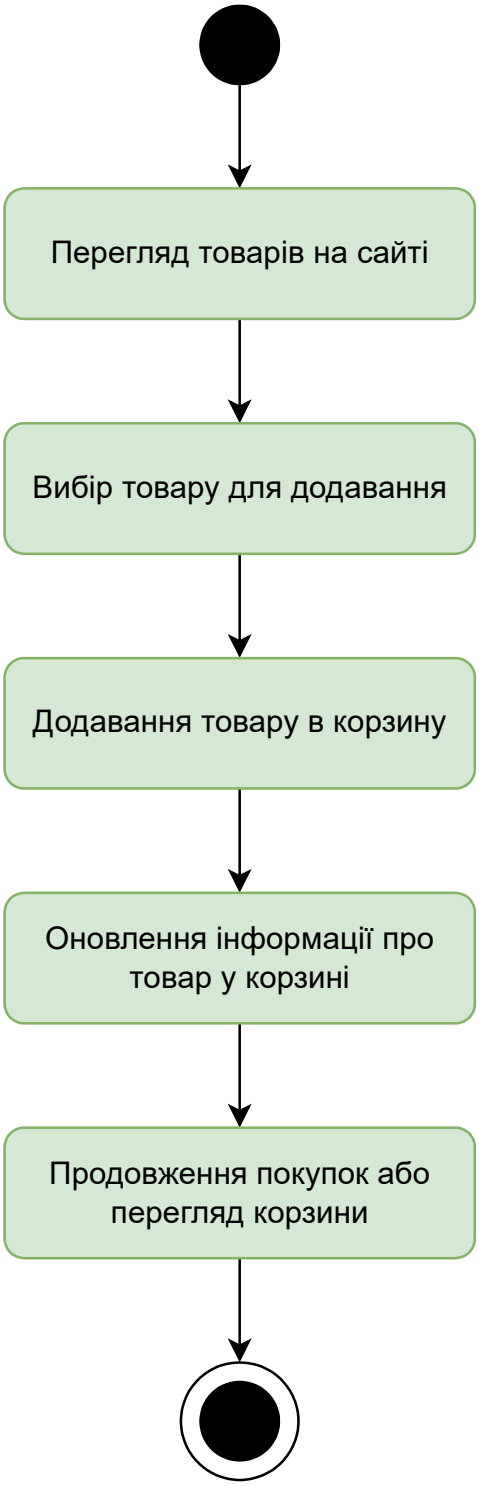

Рисунок 2.4 – Діаграма діяльності варіанту використання для додавання товару в корзину

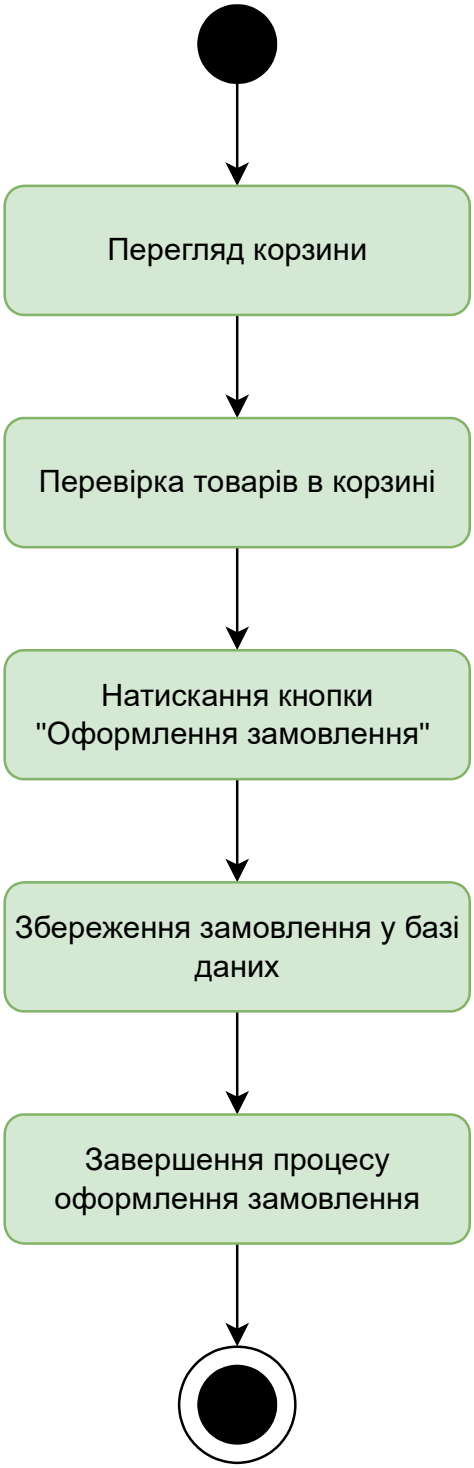

Рисунок 2.5 – Діаграма діяльності варіанту використання для оформлення замовлення

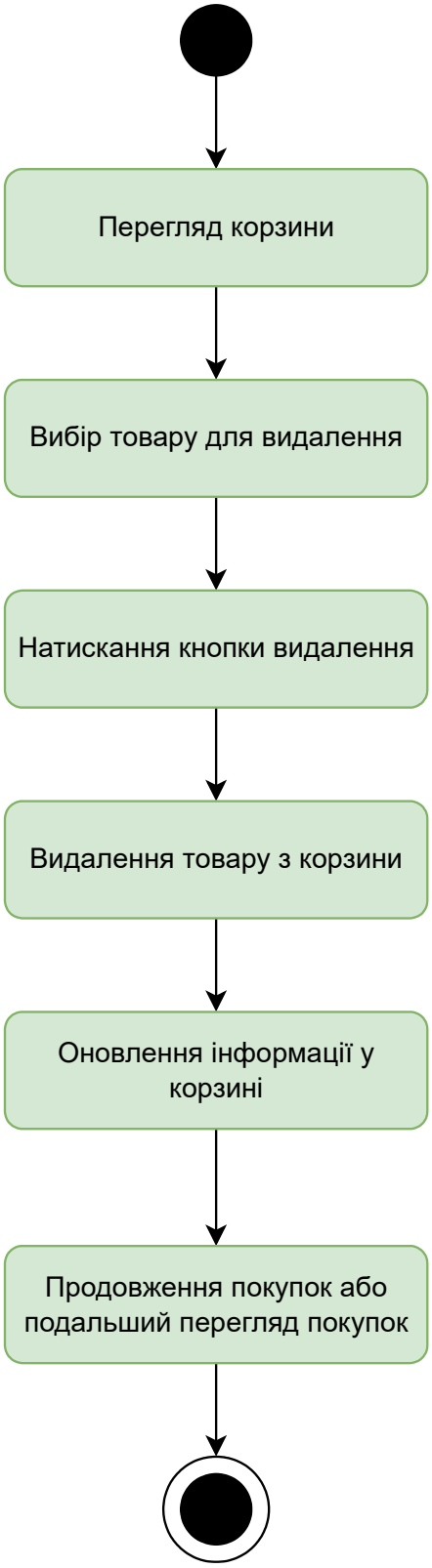

Рисунок 2.6 – Діаграма діяльності варіанту використання для видалення товару з корзини

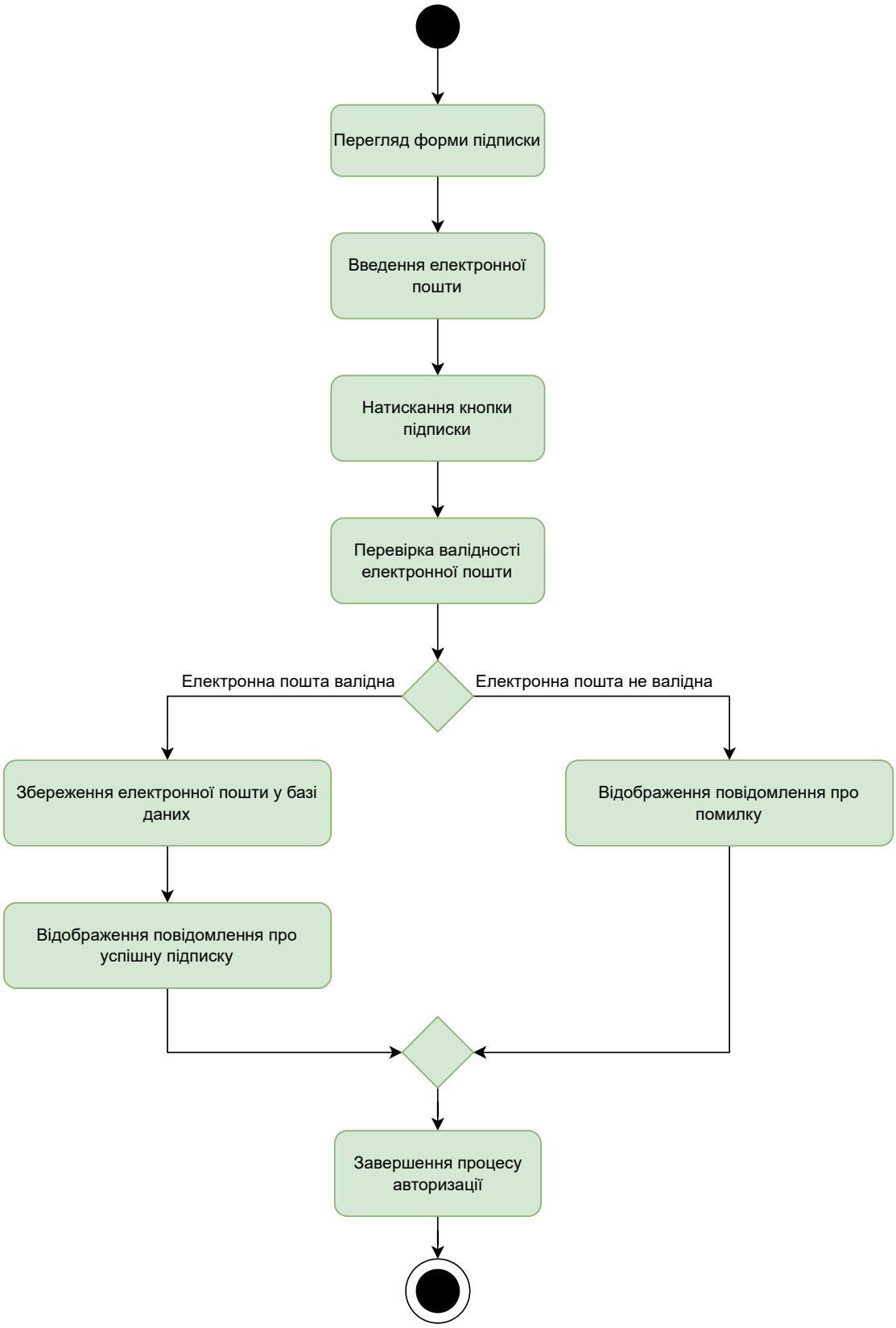

Рисунок 2.7 – Діаграма діяльності варіанту використання для підписки на розсилку

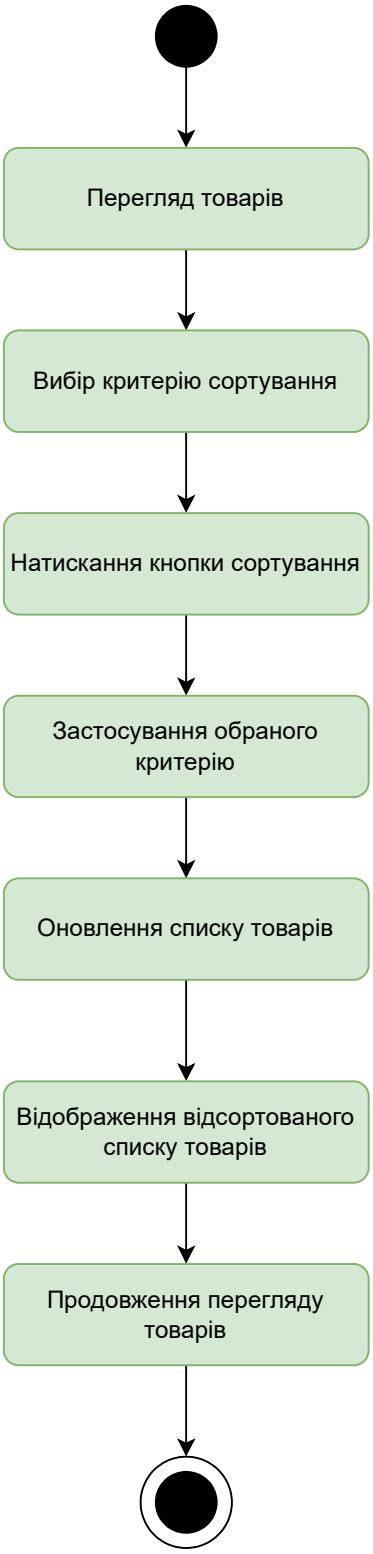

Рисунок 2.8 – Діаграма діяльності варіанту використання для сортування товарів

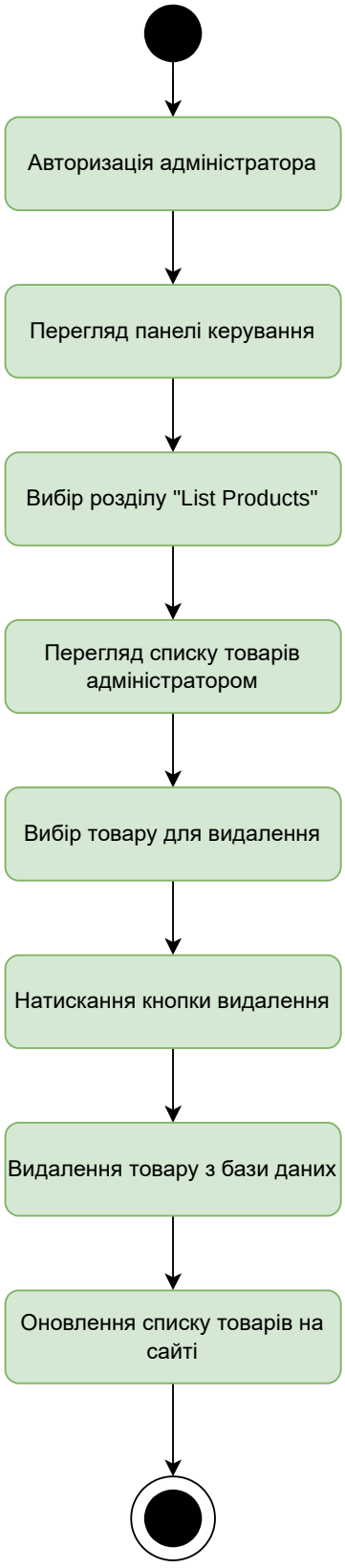

Рисунок 2.9 – Діаграма діяльності варіанту використання для видалення товарів з сайту

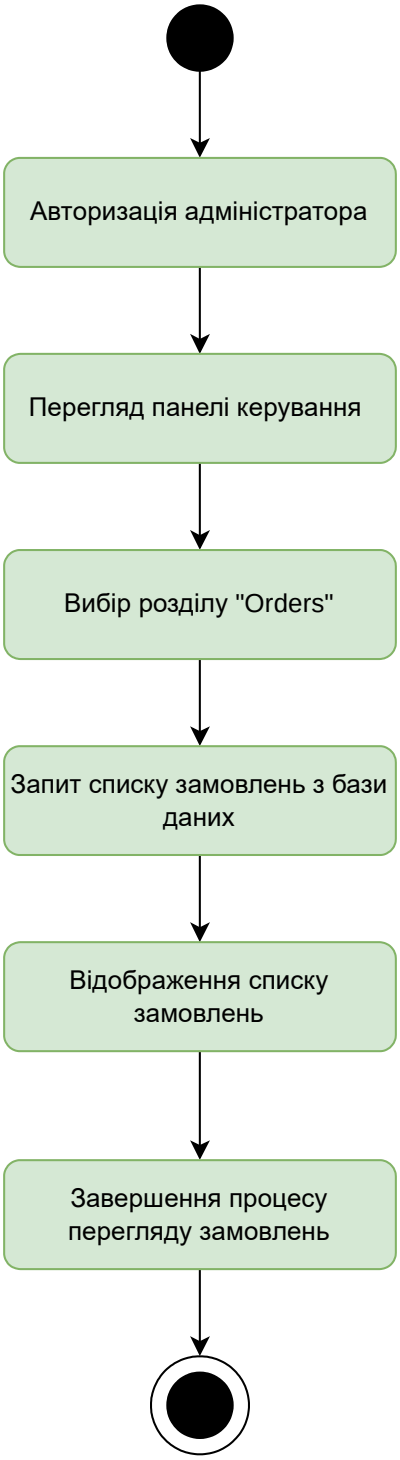

Рисунок 2.10 – Діаграма діяльності варіанту використання для перегляду замовлень адміністратором

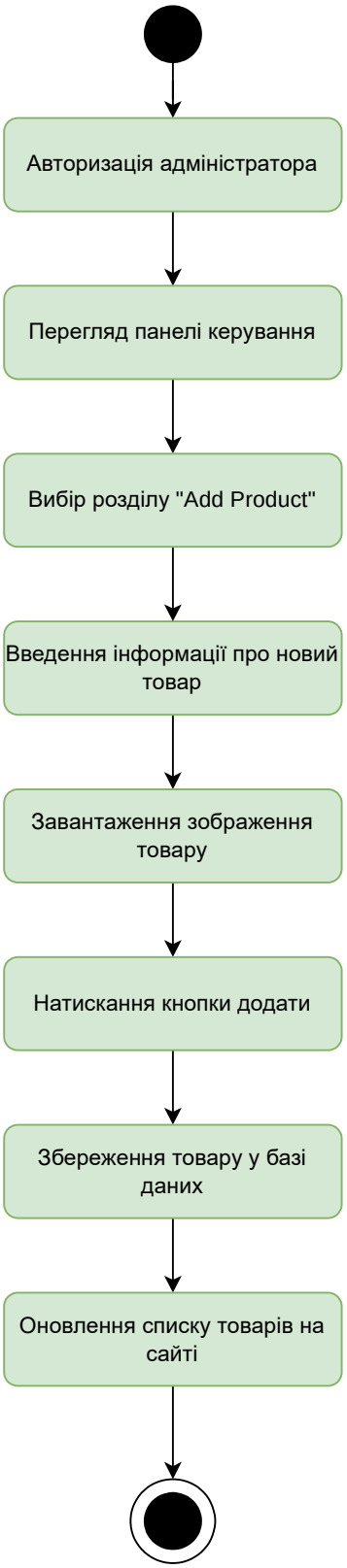

Рисунок 2.11 – Діаграма діяльності варіанту використання для додавання нових товарів на сайт

2.1.4 Інформаційна модель вебсайту «Inside out»

Модель «сутність – зв’язок» або ER-діаграма – дана модель дає візуальне представлення різних сутностей всередині системи і взаємозв’язків між ними [3].

Нижче показана концептуальна модель, що представлена у вигляді діаграми «сутність-зв’язок» на рисунку 2.12. Діаграма демонструє основну частину даних, щоб була зрозуміла структура проєкту та взаємодія елементів між собою.

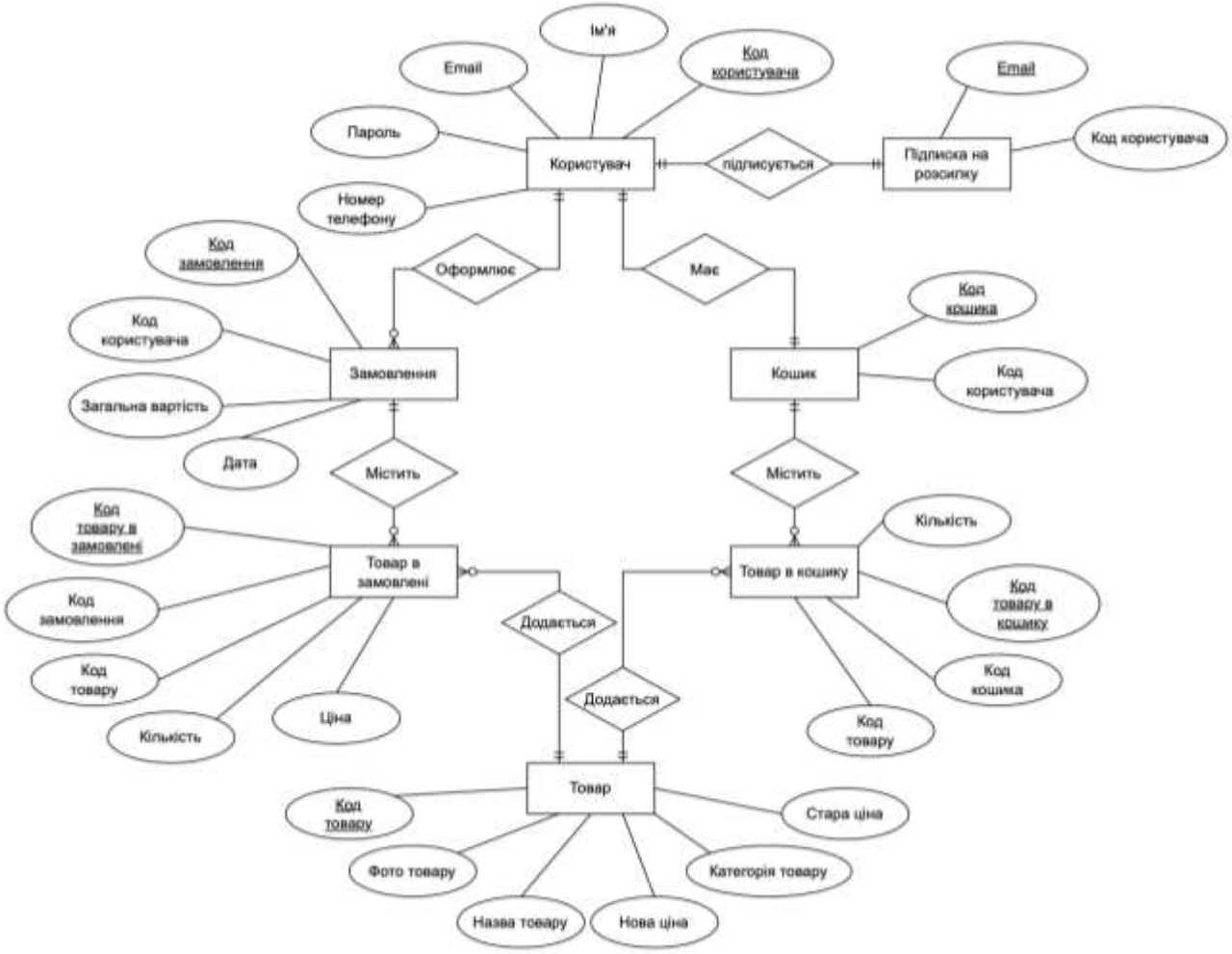

Рисунок 2.12 – Концептуальна модель даних для вебсайту «Inside out»

2.1.5 Проєкт користувальницького інтерфейсу

UI (User Interface) або користувацький інтерфейс — це простір, де відбувається взаємодія між користувачем і машиною, зазвичай комп'ютером. Інтерфейс включає в себе всі елементи, з якими користувач може взаємодіяти, такі як кнопки, іконки, меню, і графічні компоненти, що дозволяють користувачам ефективно керувати машиною і отримувати зворотний зв'язок [11, 13].

При розробці даного проєкту були використанні: бібліотека JavaScript React і CSS для стилізації. На рисунках 2.13-2.28 представлено інтерфейс сторінок вебсайту та адмін-панелі.

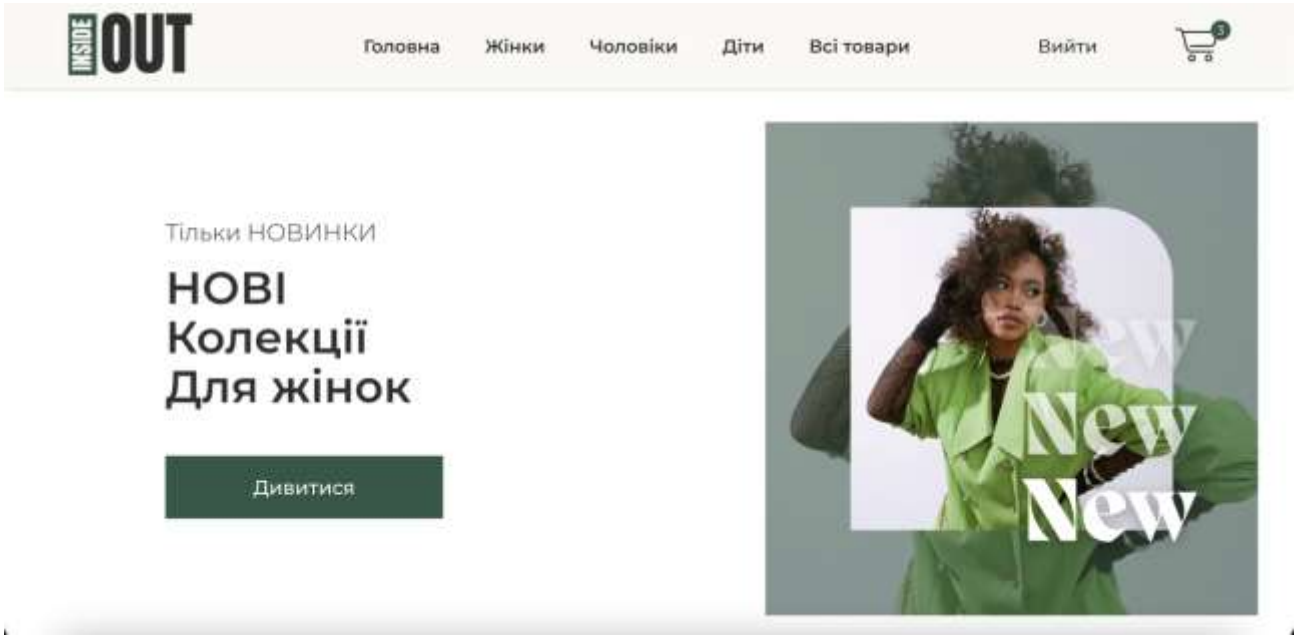

Рисунок 2.13– Інтерфейс головної сторінки

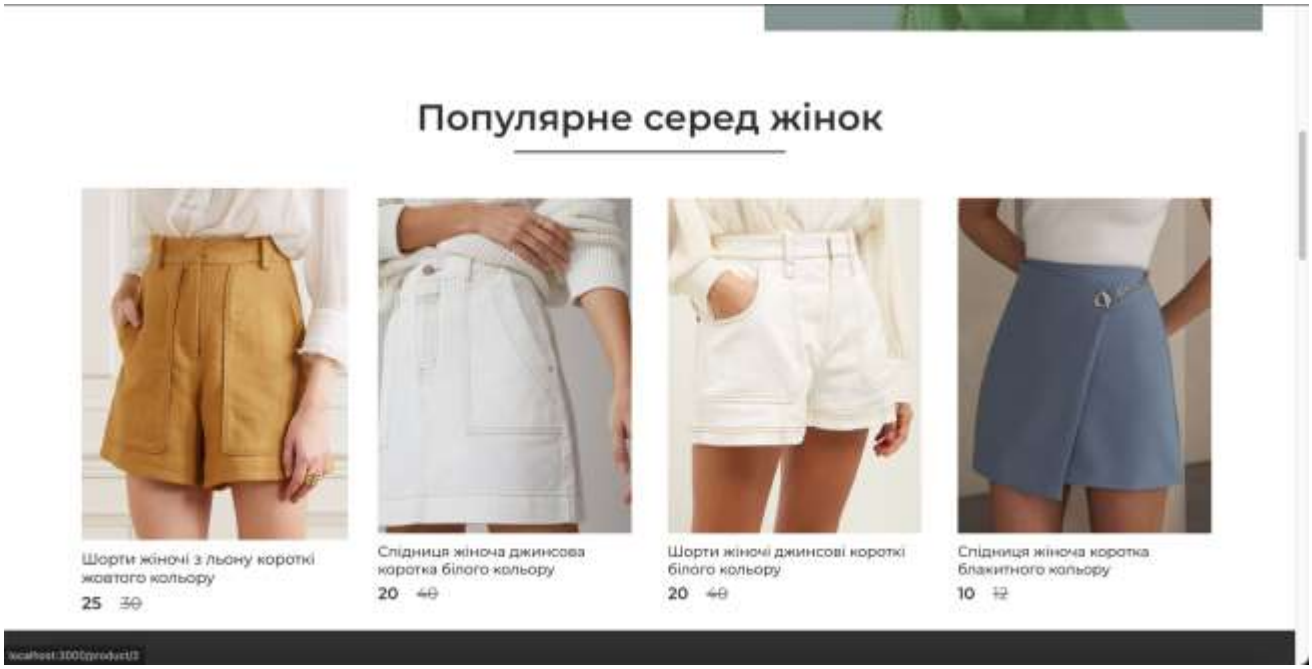

Рисунок 2.14– Інтерфейс головної сторінки «Популярне серед жінок

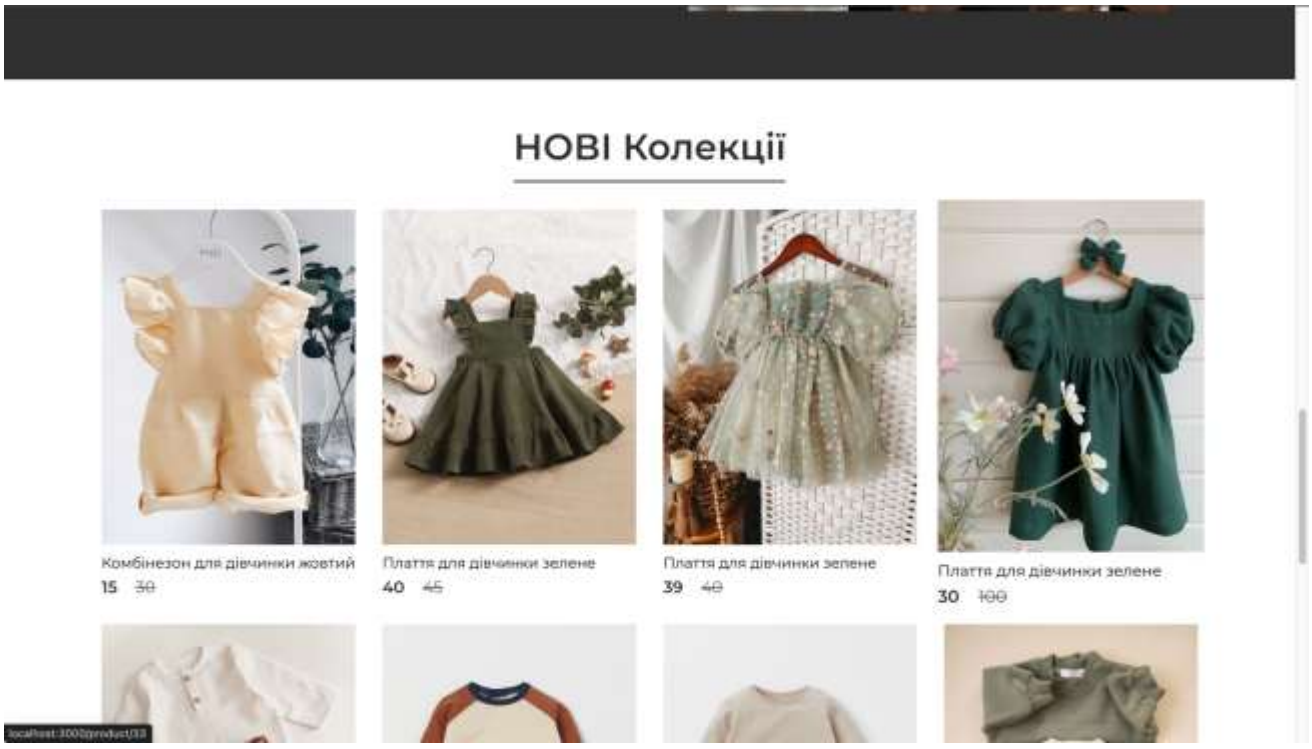

Рисунок 2.15– Інтерфейс головної сторінки «Нові колекції»

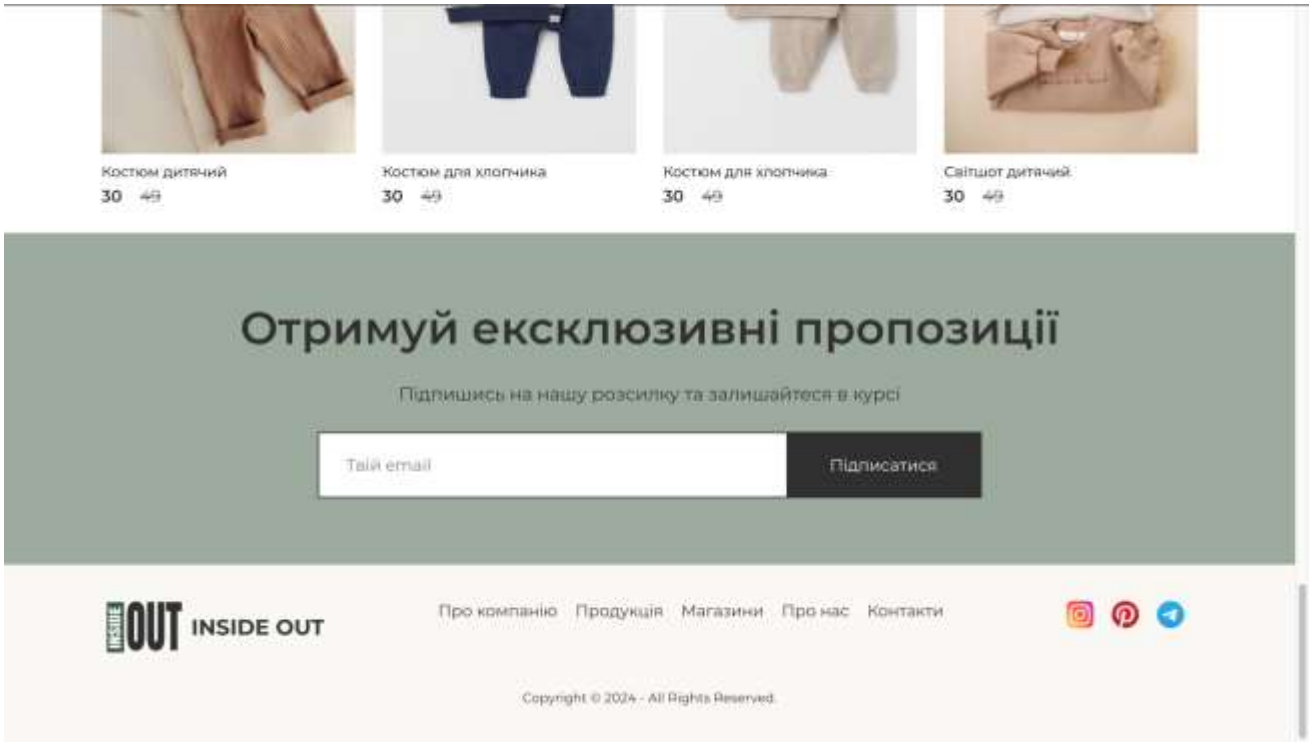

Рисунок 2.16– Інтерфейс головної сторінки «Підписка на розсилку»

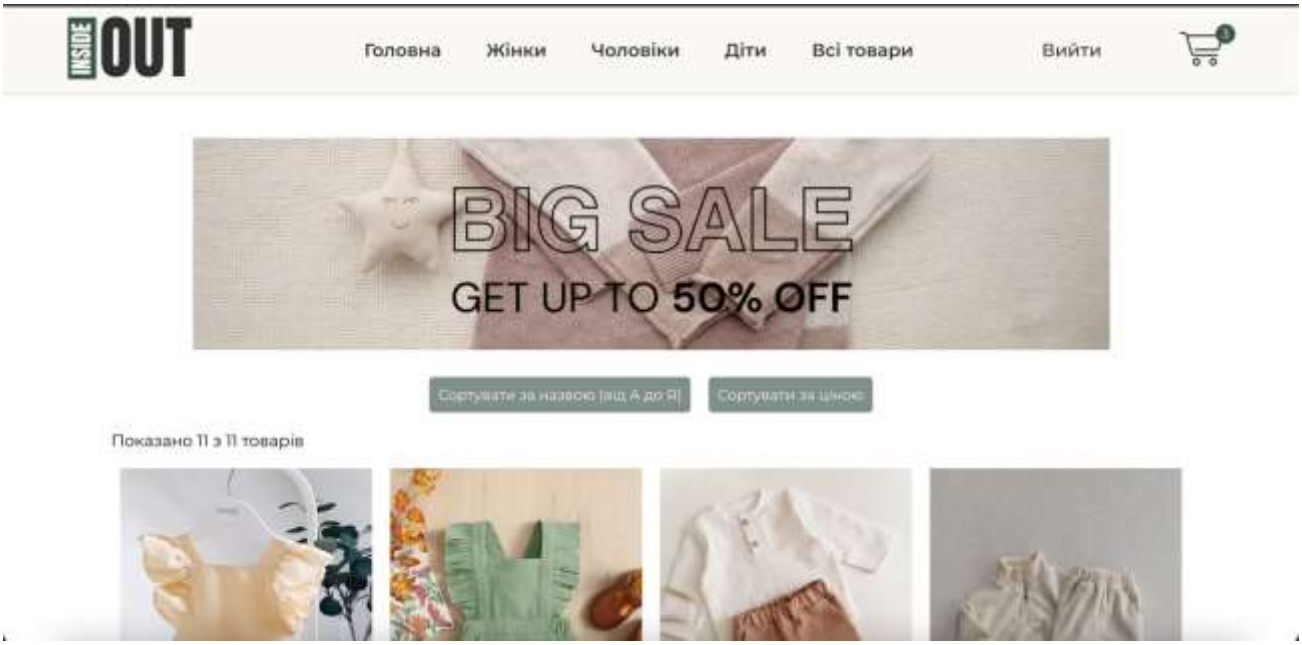

Рисунок 2.17 – Інтерфейс сторінки «Діти»

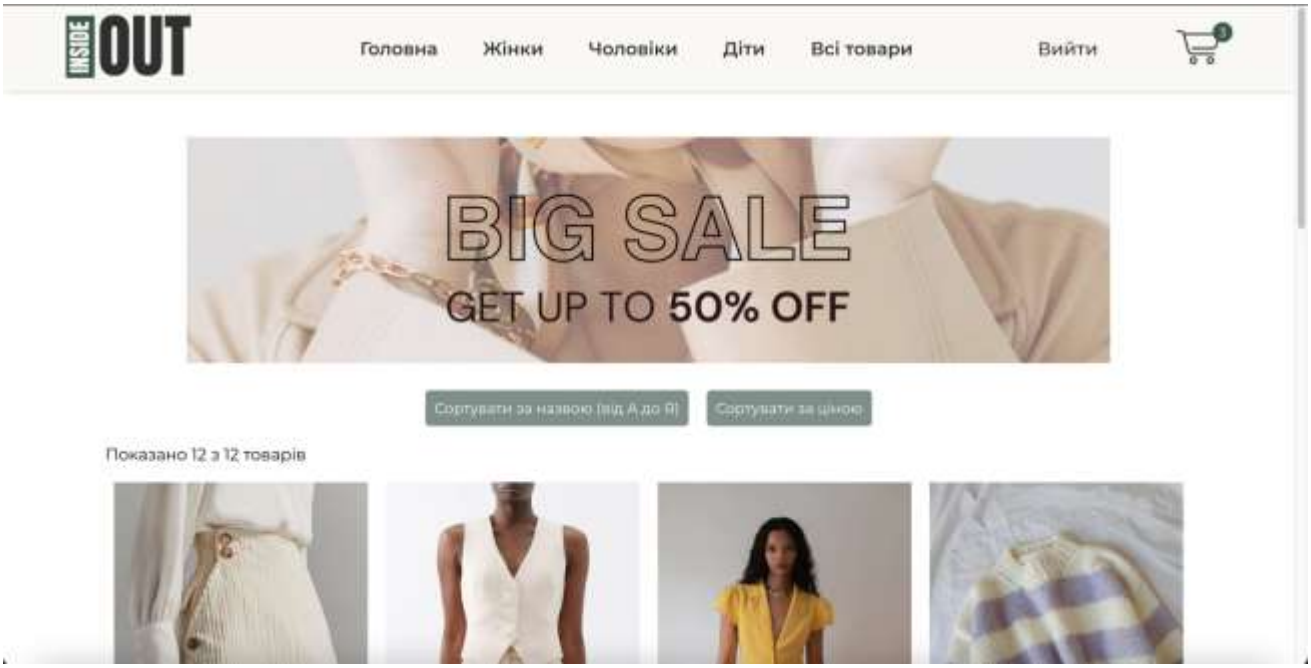

Рисунок 2.18 – Інтерфейс сторінки «Жінки»

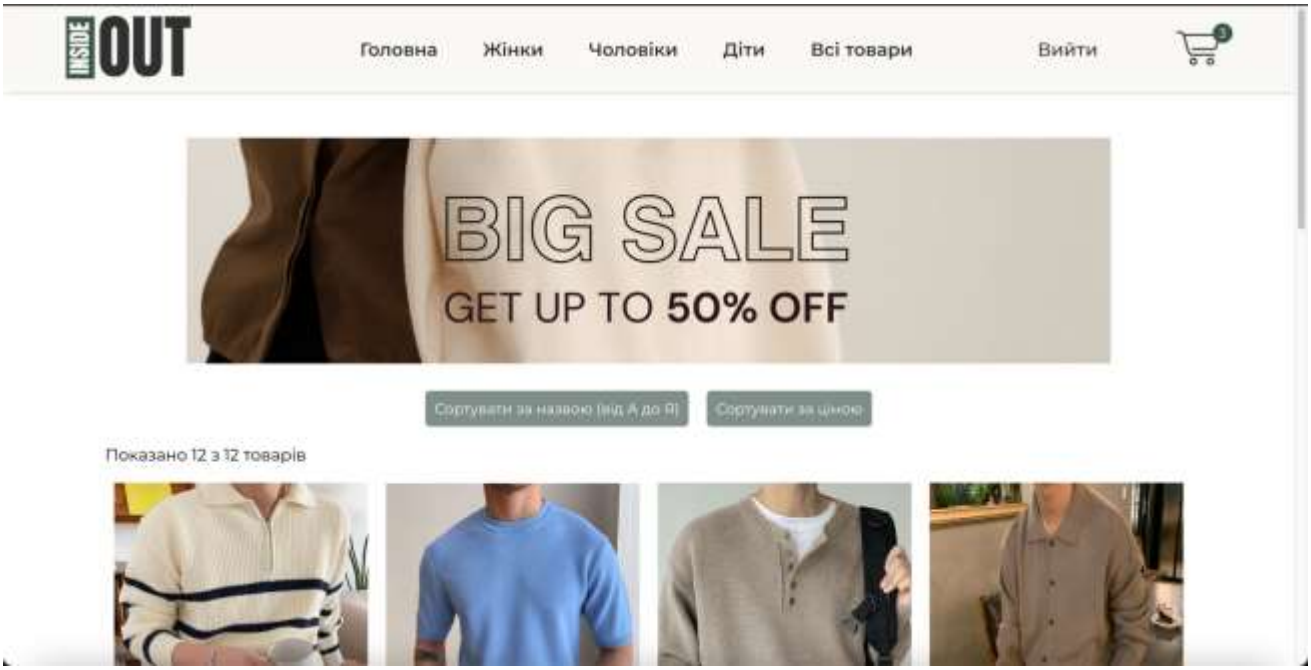

Рисунок 2.19 – Інтерфейс сторінки «Чоловіки»

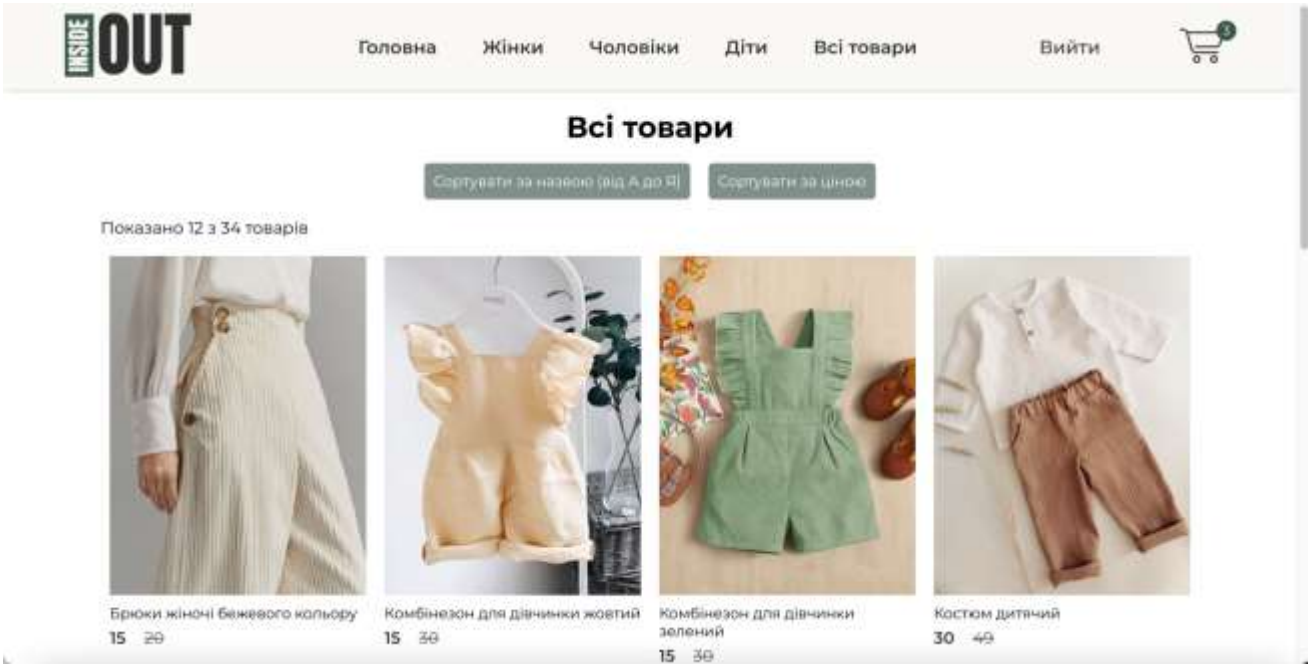

Рисунок 2.20 – Інтерфейс сторінки «Всі товари»

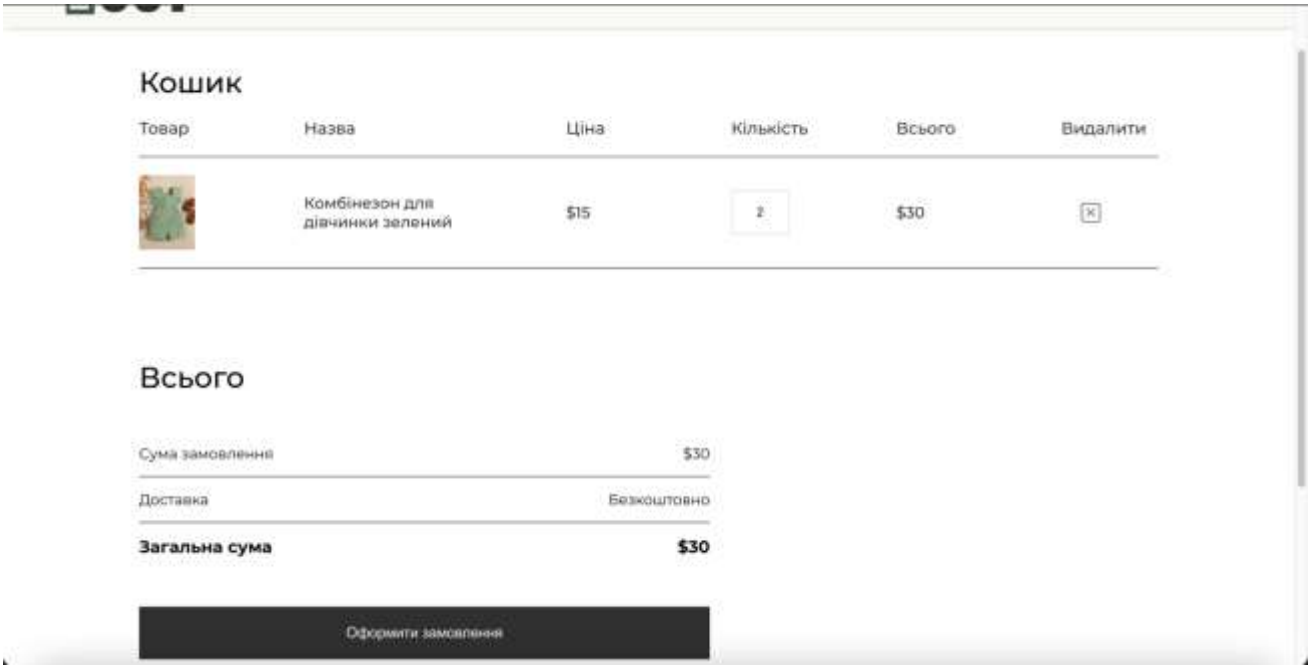

Рисунок 2.21 – Інтерфейс сторінки «Кошик»

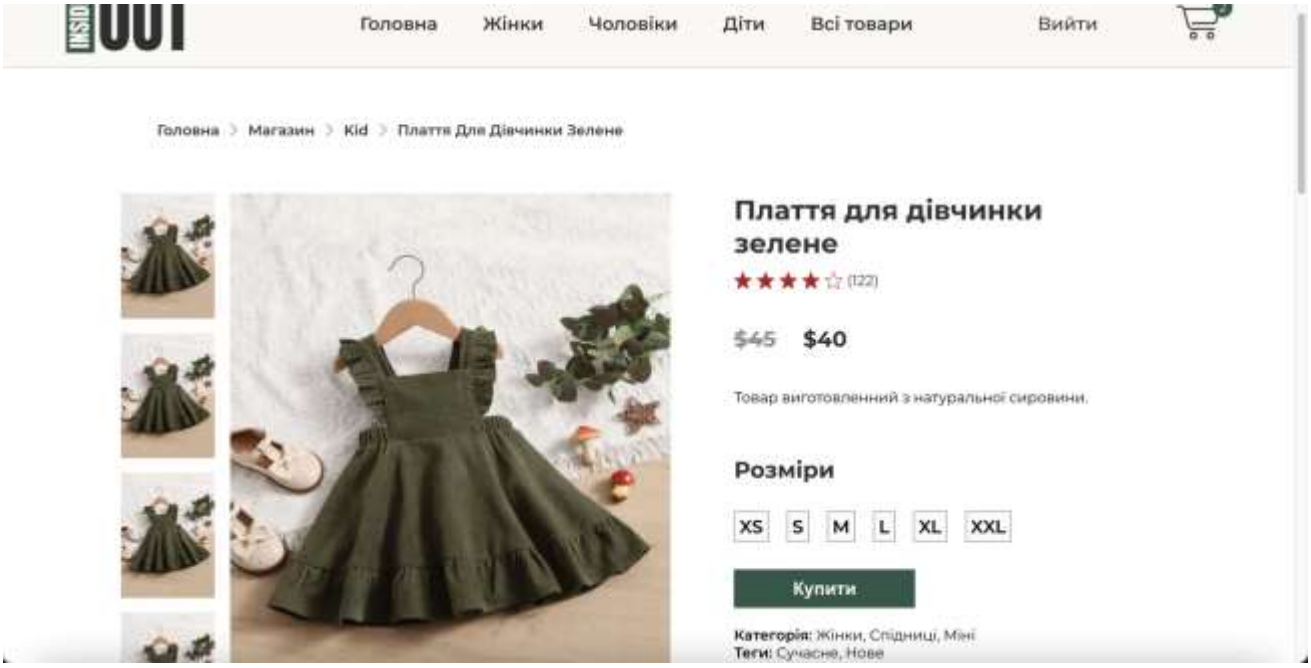

Рисунок 2.22 – Інтерфейс сторінки товару

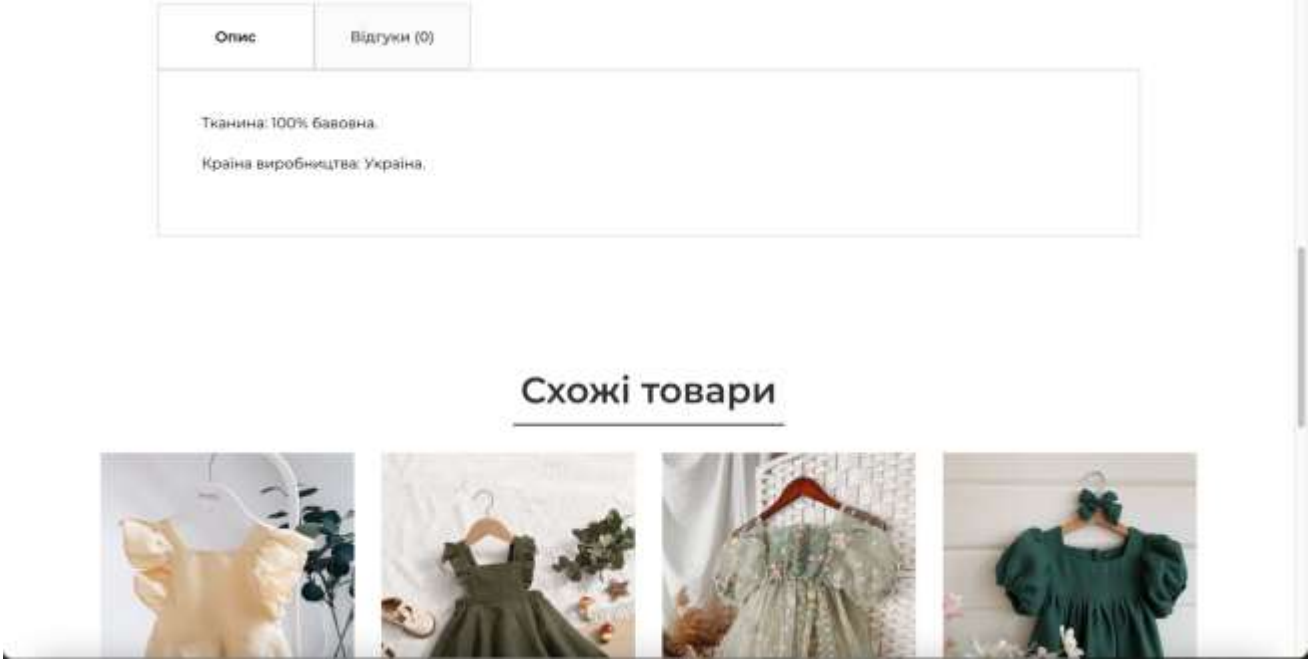

Рисунок 2.23 – Інтерфейс сторінки товару: «Опис» та «Схожі товари»

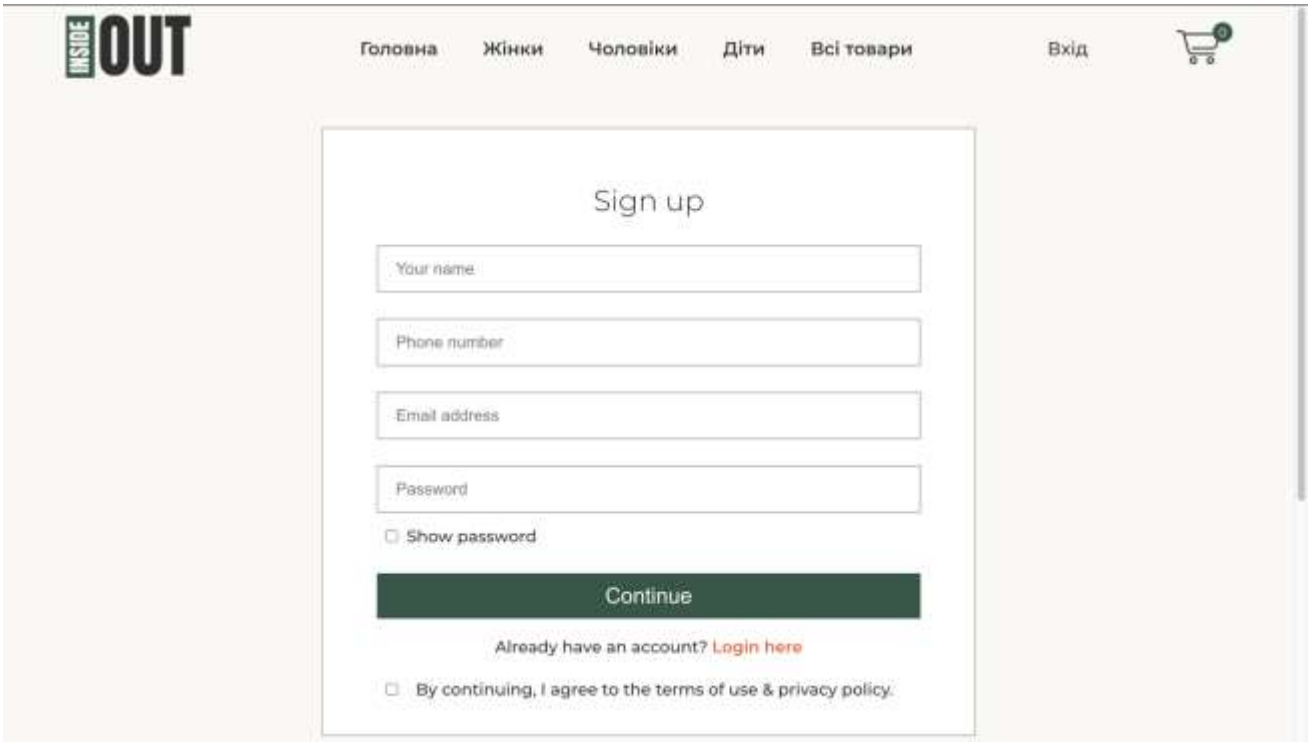

Рисунок 2.24 – Інтерфейс сторінки «Sign up»

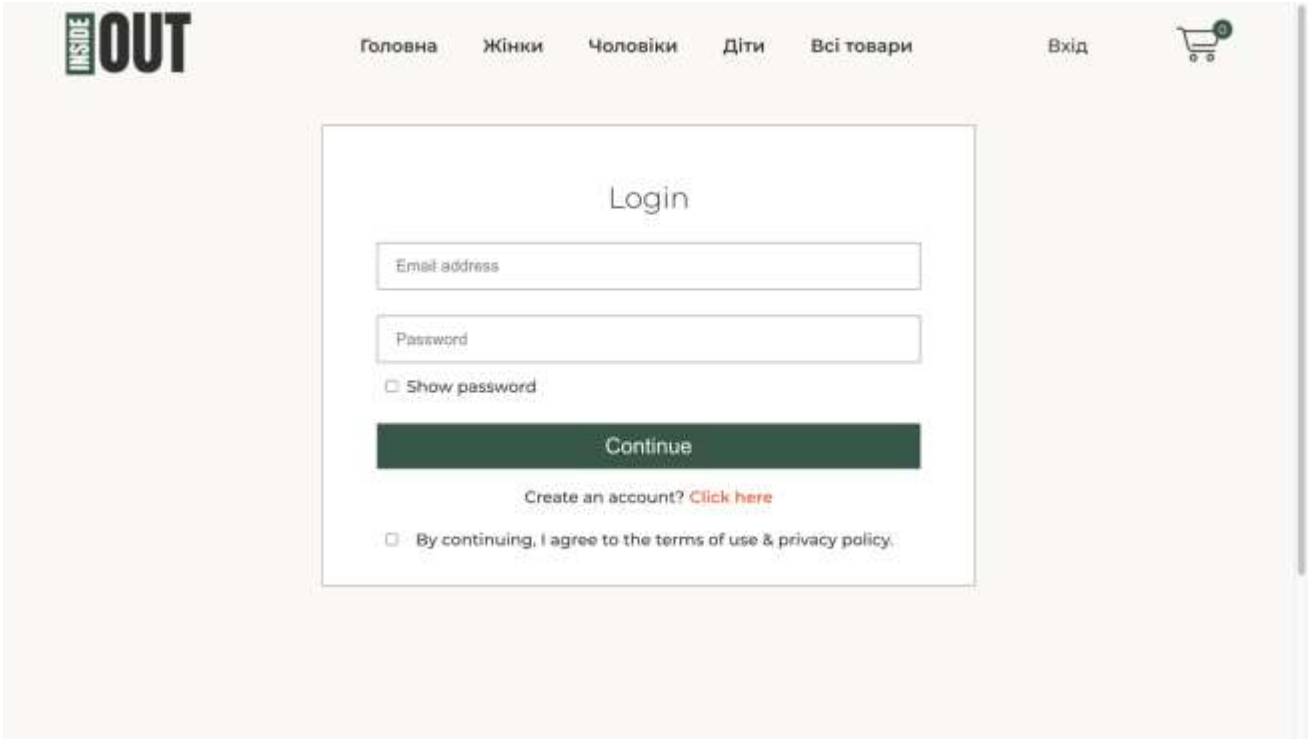

Рисунок 2.25 – Інтерфейс сторінки «Login»

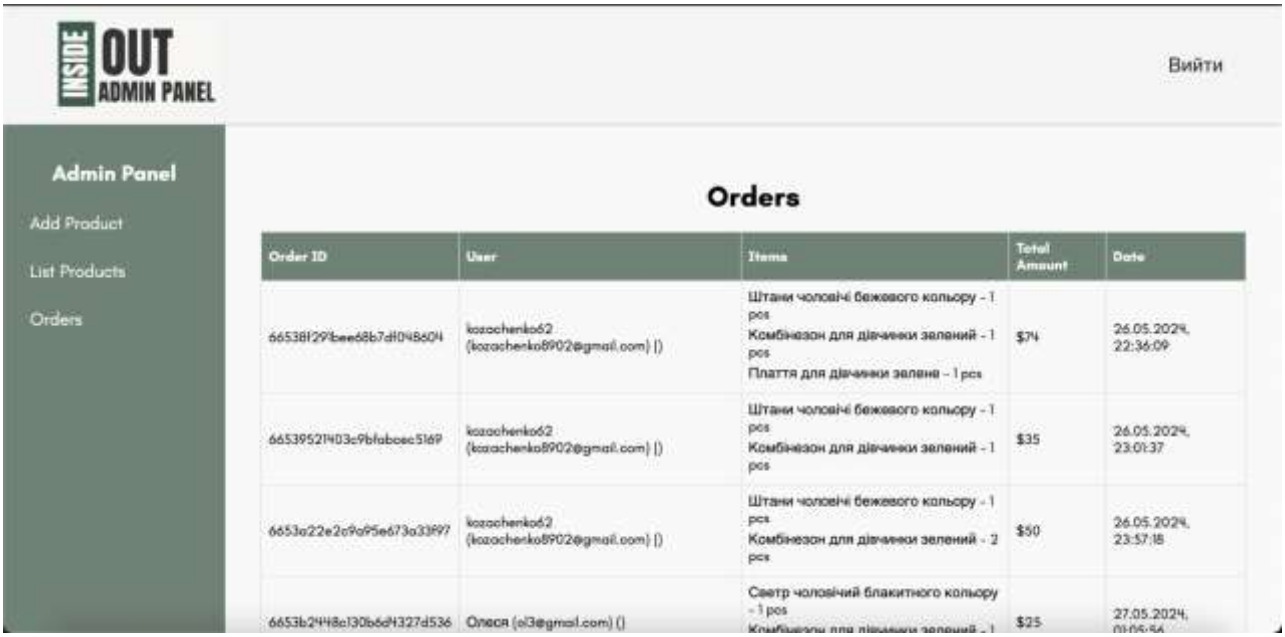

Рисунок 2.26 – Інтерфейс адмін панелі. Вкладка «Orders»

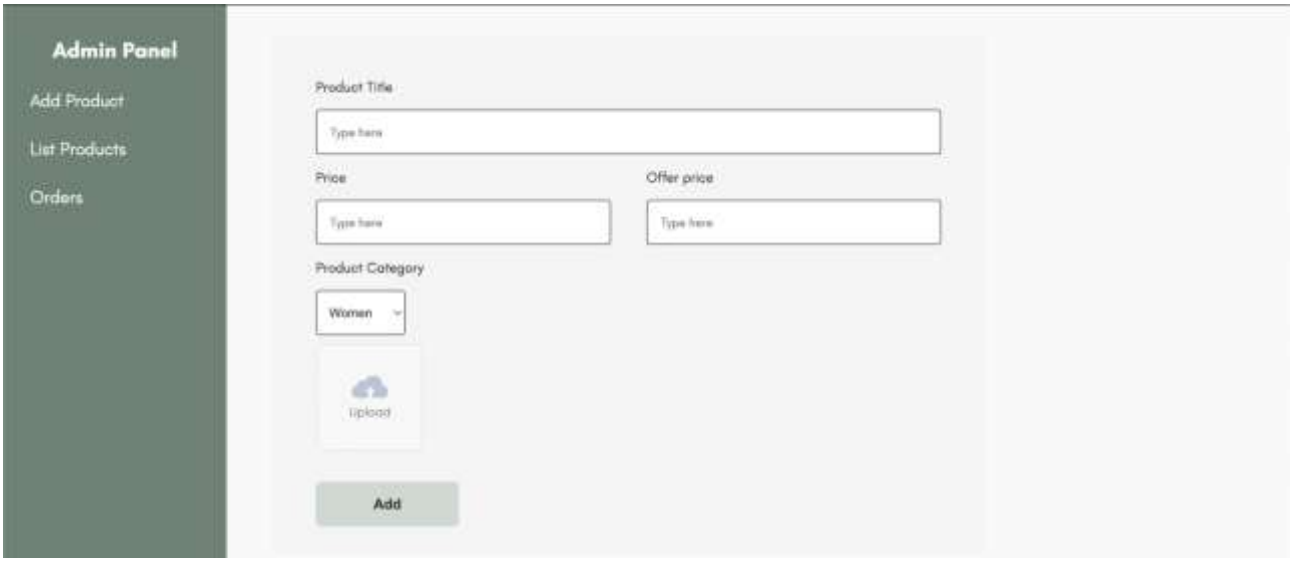

Рисунок 2.27 – Інтерфейс адмін панелі. Вкладка «Add Product»

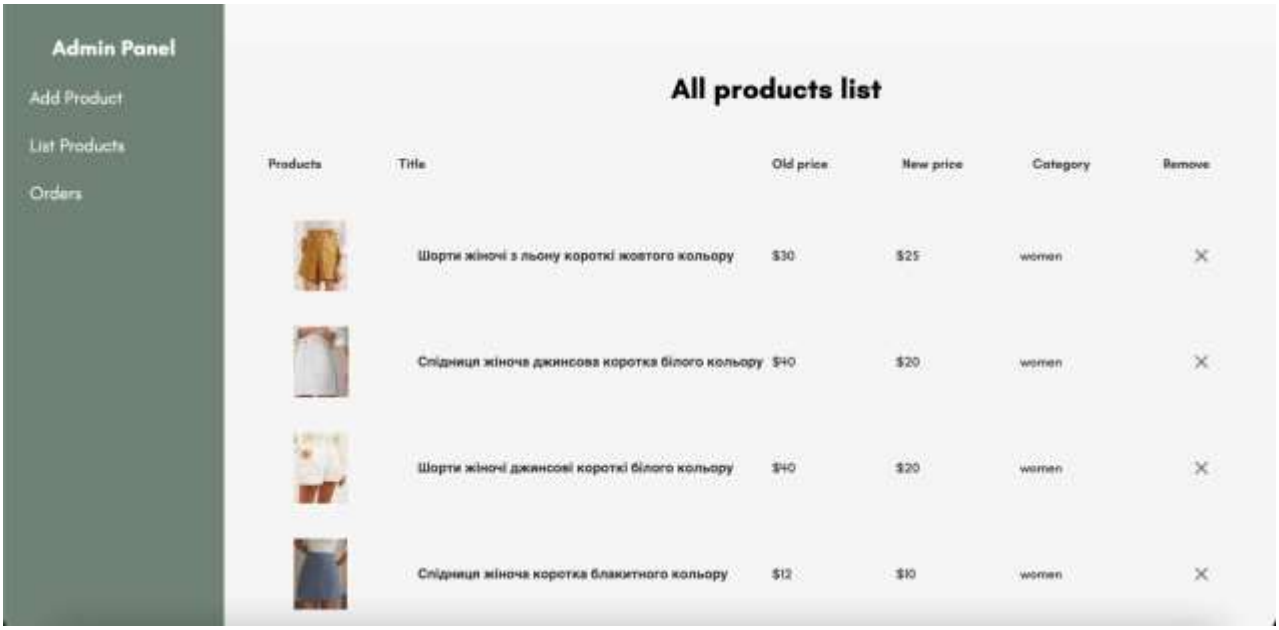

Рисунок 2.28 – Інтерфейс адмін панелі. Вкладка «All products list»

2.2 Технічний проєкт вебсайту для магазину одягу “Inside out”

Технічний проєкт являє собою етап розробки системи і документ. Основною метою цього документа є демонстрація кінцевих технічних рішень, які будуть реалізовані в процесі створення програмного продукту.

Було створено статичну модель програми (діаграма класів), описані специфікації класів, діаграма послідовності та логічна модель даних у рамках технічного проєкту.

2.2.1 Статична модель програми (діаграма класів) вебсайту «Inside out»

Статична модель дає змогу уявити компоненти та структуру програмного забезпечення, що вказує на важливість цього елементу проєктування системи. Модель часто використовують, щоб візуалізувати архітектуру програмного забезпечення для кращого розуміння зв’язків між компонентами системи.

Щоб представити статистичну модель буде використана діаграма класів на рисунку 2.29.

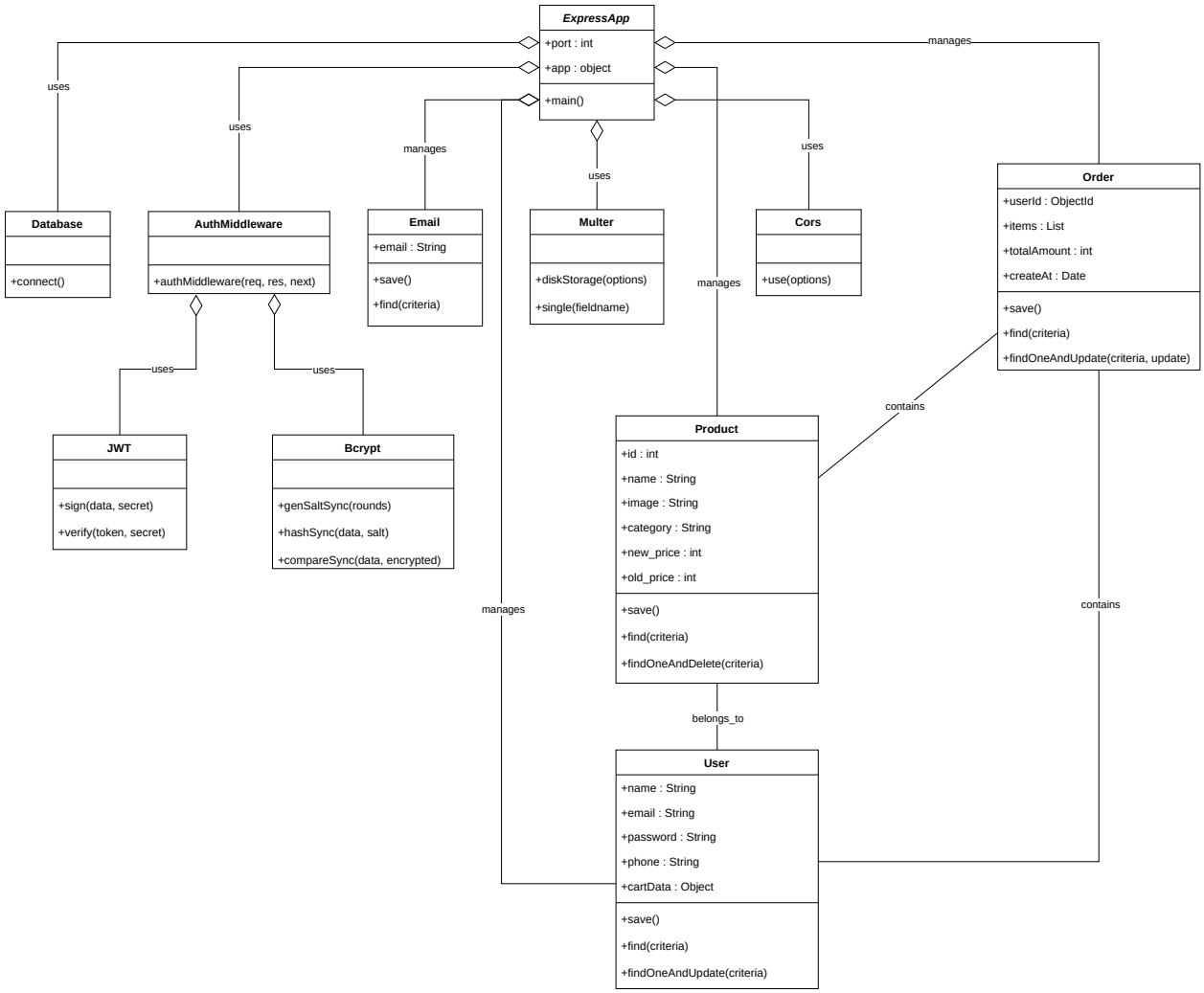

Рисунок 2.29 – Діаграма класів серверної частини «Inside out»

2.2.2 Специфікації класів

Кожен із наведених класів має своє призначення. Кожен атрибут застосовується в логіці системи, а кожен метод виконує певну частину цієї логіки. Нижче наведено описи класів.

Клас ExpressApp.

Опис: основний клас додатку Express, який налаштовує сервер та управляє додатком.

Атрибути:

- `port: int` — порт, на якому запущений сервер.
- `app: object` — об'єкт додатку Express.

Методи:

- `main()` — головний метод, який запускає сервер.

Клас Database.

Опис: клас для управління підключенням до бази даних.

Атрибути: відсутні.

Методи:

- `connect()` — метод для підключення до бази даних.

Клас AuthMiddleware.

Опис: проміжне програмне забезпечення для аутентифікації користувачів.

Атрибути: відсутні.

Методи:

- `authMiddleware(req, res, next)` — метод для обробки аутентифікації запитів.

Клас Email.

Опис: клас для управління електронною поштою користувачів.

Атрибути:

- `email: String` — адреса електронної пошти користувача.

Методи:

- `save()` — метод для збереження електронної пошти.
- `find(criteria)` — метод для пошуку електронної пошти за критеріями.

Клас Multer.

Опис: клас для обробки завантаження файлів.

Атрибути: відсутні.

Методи:

- `diskStorage(options)` — метод для налаштування сховища файлів.

- `single(fieldname)` — метод для завантаження одного файлу.

Клас Cors.

Опис: клас для налаштування політики CORS (Cross-Origin Resource Sharing).

Атрибути: відсутні.

Методи:

– `use(options)` — метод для застосування налаштувань CORS.

Клас JWT.

Опис: клас для роботи з JSON Web Tokens.

Атрибути: відсутні.

Методи:

- `sign(data, secret)` — метод для створення токена.

- `verify(token, secret)` — метод для верифікації токена.

Клас Bcrypt.

Опис: клас для хешування паролів.

Атрибути: відсутні.

Методи:

- `genSaltSync(rounds)` — метод для генерації «солі»(випадкова строка, яка додається до пароля перед хешуванням для підвищення безпеки).

- `hashSync(data, salt)` — метод для хешування даних.

- `compareSync(data, encrypted)` — метод для порівняння даних з хешем.

Клас Order.

Опис: клас для управління замовленнями.

Атрибути:

- `userId: ObjectId` — ідентифікатор користувача.

- `items: List` — список товарів у замовленні.

- `totalAmount: int` — загальна сума замовлення.

- `createdAt: Date` — дата створення замовлення.

Методи:

- `save()` — метод для збереження замовлення.

- `find(criteria)` — метод для пошуку замовлень за критеріями.

- `findOneAndUpdate(criteria, update)` — метод для пошуку та оновлення замовлення.

Клас Product.

Опис: клас для управління продуктами.

Атрибути:

- `id: int` — ідентифікатор продукту.

- `name: String` — назва продукту.

- `image: String` — зображення продукту.

- `category: String` — категорія продукту.

- `new_price: int` — нова ціна продукту.

- `old_price: int` — стара ціна продукту.

Методи:

- `save()` — метод для збереження продукту.

- `find(criteria)` — метод для пошуку продуктів за критеріями.

- `findOneAndDelete(criteria)` — метод для пошуку та видалення продукту.

Клас User.

Опис: клас для управління користувачами.

Атрибути:

- `name: String` — ім'я користувача.

- `email: String` — електронна пошта користувача.

- `password: String` — пароль користувача.

- `phone: String` — номер телефону користувача.

- `cartData: Object` — дані кошика користувача.

Методи:

- `save()` — метод для збереження користувача.

- `find(criteria)` — метод для пошуку користувачів за критеріями.

- `findOneAndUpdate(criteria, update)` — метод для пошуку та оновлення даних користувача.

2.2.3 Динамічна модель програми

Динамічна модель системи — сукупність співвідношень, що визначають вихід системи в залежності від входу та стану системи. Динамічна модель відтворює зміни об'єкта, які відбуваються з плином часу, або особливості функціювання об'єкта. Динамічні моделі називають також функціональними [1].

Дана модель буде представлена у вигляді діаграми послідовності.

На рисунку 2.30 зображено діаграму послідовності для цього проєкту.

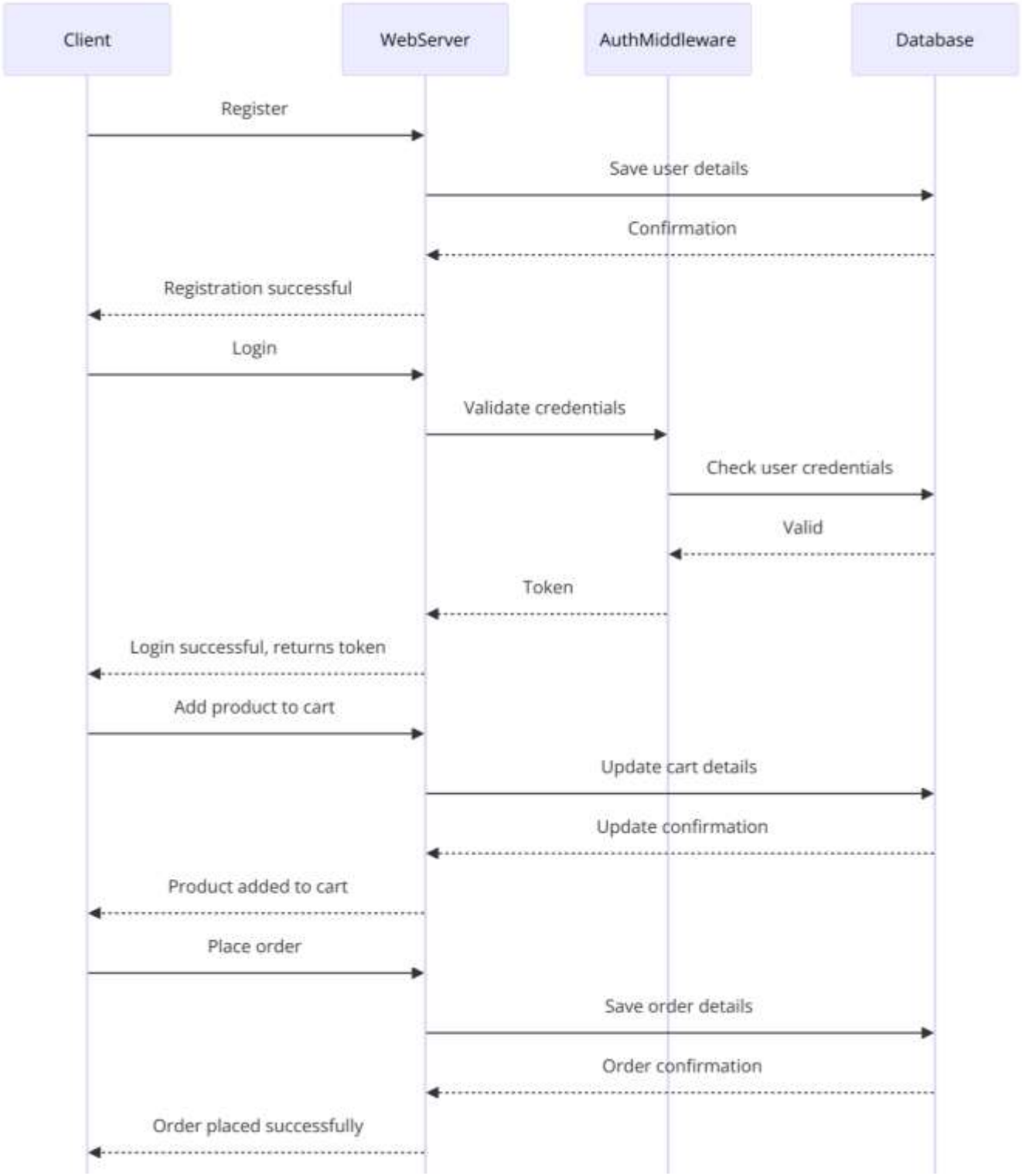

Рисунок 2.30 – Діаграма послідовності логіки роботи вебсайту

2.2.4 Логічна модель даних

Логічна модель даних або логічна схема — це модель даних конкретної проблемної області, виражена незалежно від конкретного продукту системи управління базами даних або технології зберігання (фізична модель даних), але в термінах структур даних, таких як реляційні таблиці та стовпці, об'єктно-орієнтовані класи, або теги XML. На відміну від концептуальної моделі даних, яка описує семантику організації без посилання на технологію [10].

Логічна модель даних наведена на рисунку 2.31. У табличному варіанті представлена логічна модель системи у таблицях 2.1-2.7.

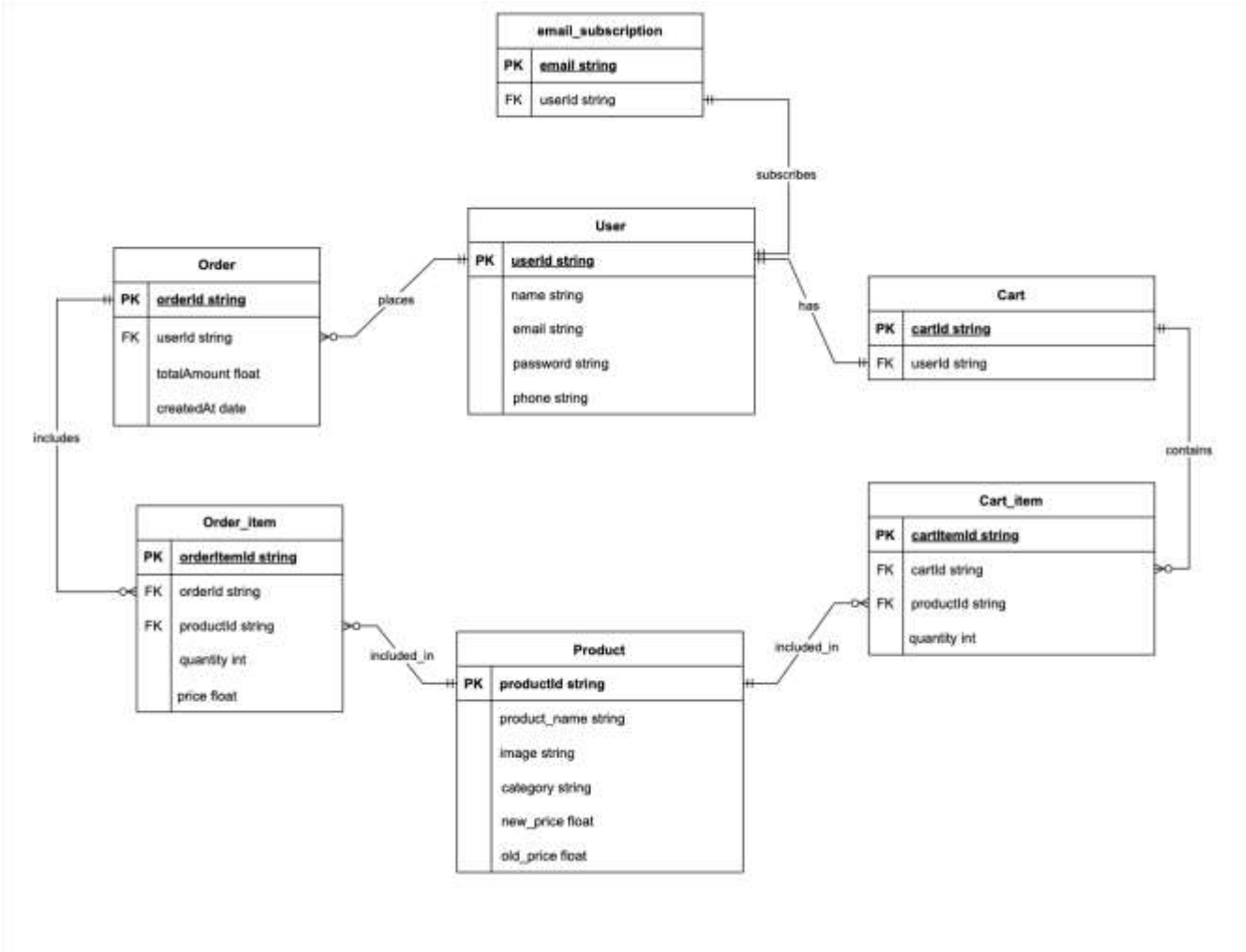

Рисунок 2.31 – Логічна модель даних вебсайту

Таблиця 2.1 – Таблиця User

Назва поля	Тип даних	Ключ
userId	string	PK
name	string	
email	string	
password	string	
phone	string	

Таблиця 2.2 – Таблиця Product

Назва поля	Тип даних	Ключ
productId	string	PK
product_name	string	
image	string	
category	string	
new_price	float	
old_price	float	

Таблиця 2.3 – Таблиця Cart

Назва поля	Тип даних	Ключ
cartId	string	PK
userId	string	FK

Таблиця 2.4 – Таблиця Cart_item

Назва поля	Тип даних	Ключ
cartItemtId	string	PK
cartId	string	FK
product_name	string	FK
quantity	int	

Таблиця 2.5 – Таблиця email_subscription

Назва поля	Тип даних	Ключ
email	string	PK
userId	string	FK

Таблиця 2.6 – Таблиця Order

Назва поля	Тип даних	Ключ
ordertId	string	PK
userId	string	FK
totalAmount	float	
createdAt	date	

Таблиця 2.7 – Таблиця Order_item

Назва поля	Тип даних	Ключ
orderItemId	string	PK
orderId	string	FK
productId	string	FK
quantity	int	
price	float	

2.3 Робочий проєкт вебсайту для магазину одягу “Inside out”

У процесі розробки проєкту була створена діаграма компонентів, обґрунтовано вибір мови і системи програмування, проведено тестування та випробування вебсайту.

2.3.1 Обґрунтування вибору мови й системи програмування

Для розробки вебсайту та адмін-панелі були використані такі інструменти:

- Node.js – платформа для виконання JavaScript на сервері.
- React.js – JavaScript-бібліотека для побудови користувацьких інтерфейсів.
- MongoDB – документно-орієнтована база даних NoSQL.
- Express – мінімалістичний веб-фреймворк для Node.js.
- CSS – мова стилізації.
- Visual Studio Code – кросплатформений редактор коду.

Для створення вебсайту та адмін-панелі були використані такі інструменти як Node.js, React.js, MongoDB, Express, CSS та Visual Studio Code. Вибір цих інструментів був зумовлений кількома важливими факторами.

Node.js було обрано завдяки своїй здатності виконувати JavaScript на сервері та забезпечувати асинхронну обробку запитів. Це дозволяє обробляти значну кількість одночасних запитів без блокування, що дає змогу підвищити продуктивність серверної частини. Крім того, використання JavaScript як на серверній, так і на клієнтській стороні спрощує розробку та обслуговування коду, оскільки дозволяє працювати з однією мовою програмування.

React.js було обрано для розробки інтерфейсів завдяки використанню віртуального DOM та своєму компонентному підходу. Це забезпечує ефективне оновлення інтерфейсу та підвищує продуктивність продукту. Компонентний підхід дозволяє створювати повторно використовувані UI-компоненти, що значно спрощує процес розробки та підтримки складних інтерфейсів.

MongoDB було обрано за свою гнучкість зберігання неструктурованих даних та здатність легко масштабуватися горизонтально. Документно-орієнтована модель даних, що використовується MongoDB, дозволяє зберігати дані у форматі JSON-подібних документів, що дає змогу спростити роботу з даними. Висока швидкість запису та читання даних забезпечує продуктивність вебсайту.

Express, мінімалістичний веб-фреймворк для Node.js, було обрано через його простоту та гнучкість. Спрощує створення серверних додатків та API завдяки використанню middleware. Express дозволяє швидко створювати RESTful API та легко їх інтегрувати з фронтенд-додатками, що робить його ідеальним вибором для розробки вебдодатків.

CSS був обраний для стилізації, оскільки забезпечує обширні можливості для створення привабливих та адаптивних інтерфейсів.

Visual Studio Code було обрано як основний редактор коду завдяки своїй популярності та широкому функціоналу. Він підтримує різноманітні розширення, які покращують редактор, включаючи автодоповнення коду, лінтери та інтеграцію з системами контролю версій, такими як Git. Також, Visual Studio Code є кросплатформеним редактором, що сумісний з Windows, macOS та Linux.

Отже, разом усі ці інструменти створюють потужний і гнучкий стек технологій, який дозволяє швидко та ефективно розробляти сучасні веб-додатки з високою продуктивністю та масштабованістю.

2.3.2 Розробка діаграми компонентів ПЗ

В уніфікованій мові моделювання (UML) діаграма компонентів показує, як компоненти з’єднані між собою, щоб утворити більші компоненти або програмні системи. Вони використовуються для ілюстрації структури складних систем [5].

На рисунку 2.32 продемонстрована діаграма компонентів, яка відображає архітектуру вебсайту. Показує, як різні компоненти взаємодіють між собою.

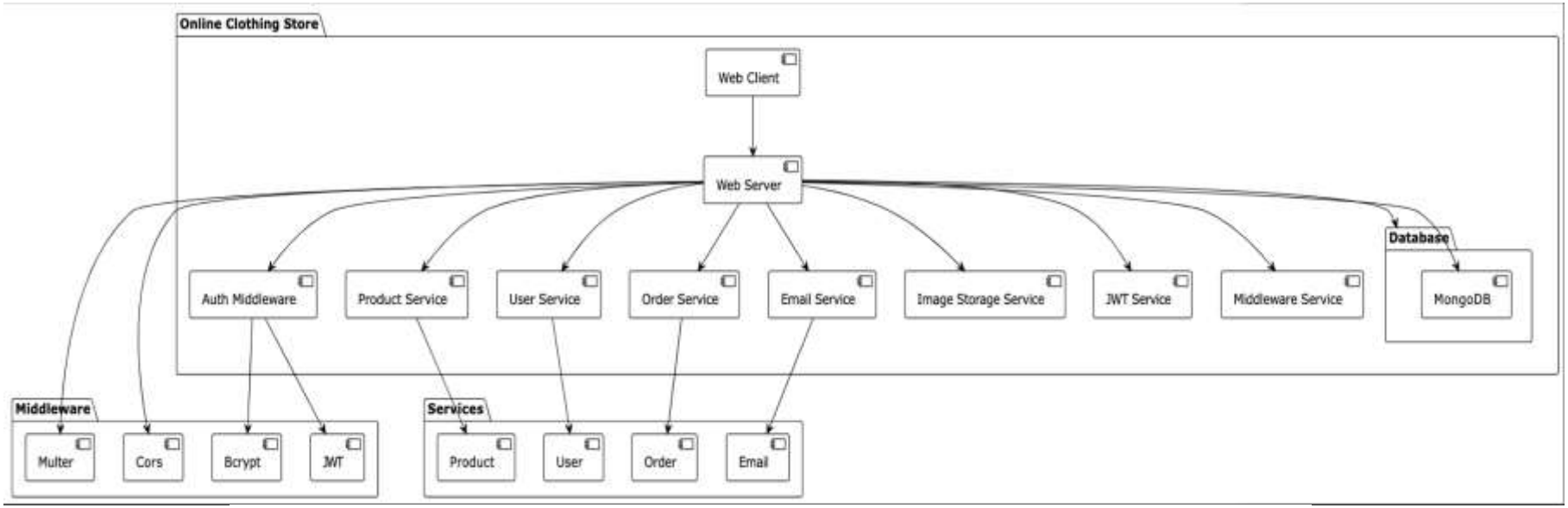

Рисунок 2.32 – Діаграма компонентів

2.3.3 Реалізація та тестування основних класів програмного забезпечення

У ході виконання кваліфікаційного проєкту було проведено тестування вебсайту та адмін-панелі методом функціонального тестування.

Функціональне тестування – один із видів тестування, спрямованого на перевірку відповідностей функціональних вимог ПЗ його реальним характеристикам. Основним завданням функціонального тестування є підтвердження того, що програмний продукт, який розробляється, володіє усім необхідним замовнику функціоналом [4].

Під час тестування перевірялися основні вимоги до проєкту, котрі описані у пункті 1.4. У цьому розділі продемонстровано лише ті протестовані вимоги, які можна підтвердити візуально.

Тест 1. Реєстрація користувача.

Вхідні данні: електронна пошта, номер телефону, пароль, ім’я.

Вихідні данні: після натискання кнопки «Continue» успішна реєстрація користувача.

Результат тестування: користувач створює акаунт успішно.

Рисунок з результатом тестування: рисунок 2.33.

Примітка: відсутня.

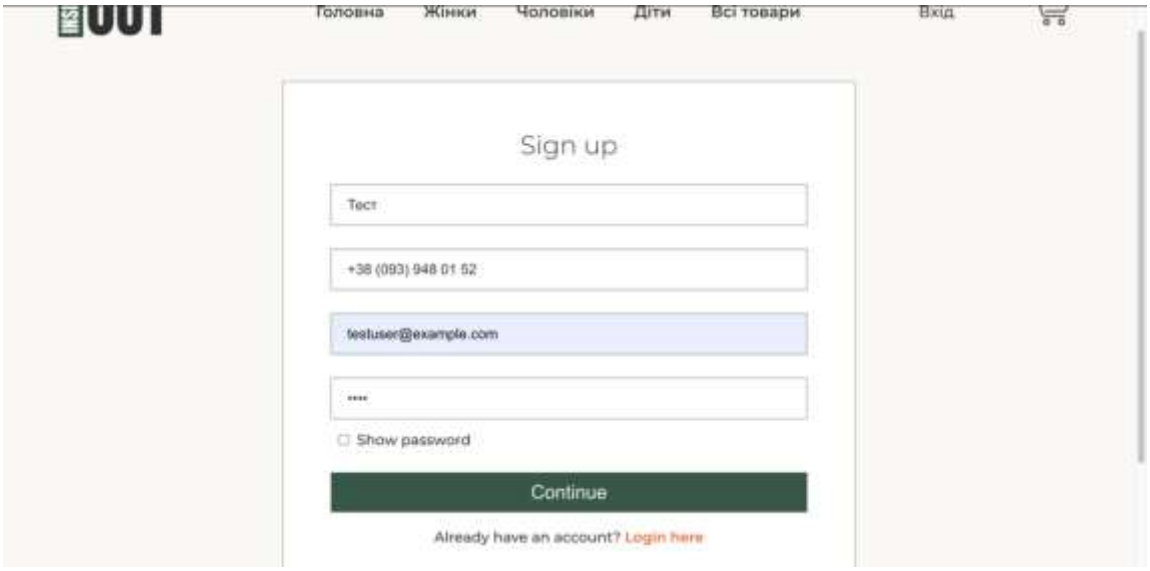

Рисунок 2.33 – Форма реєстрації з введеними даними

Тест 2. Аутентифікація користувача.

Вхідні данні: електронна пошта, пароль.

Вихідні данні: після натискання кнопки «Continue» успішний вхід у систему.

Результат тестування: користувач входить у систему.

Рисунок з результатом тестування: рисунок 2.34.

Примітка: відсутня.

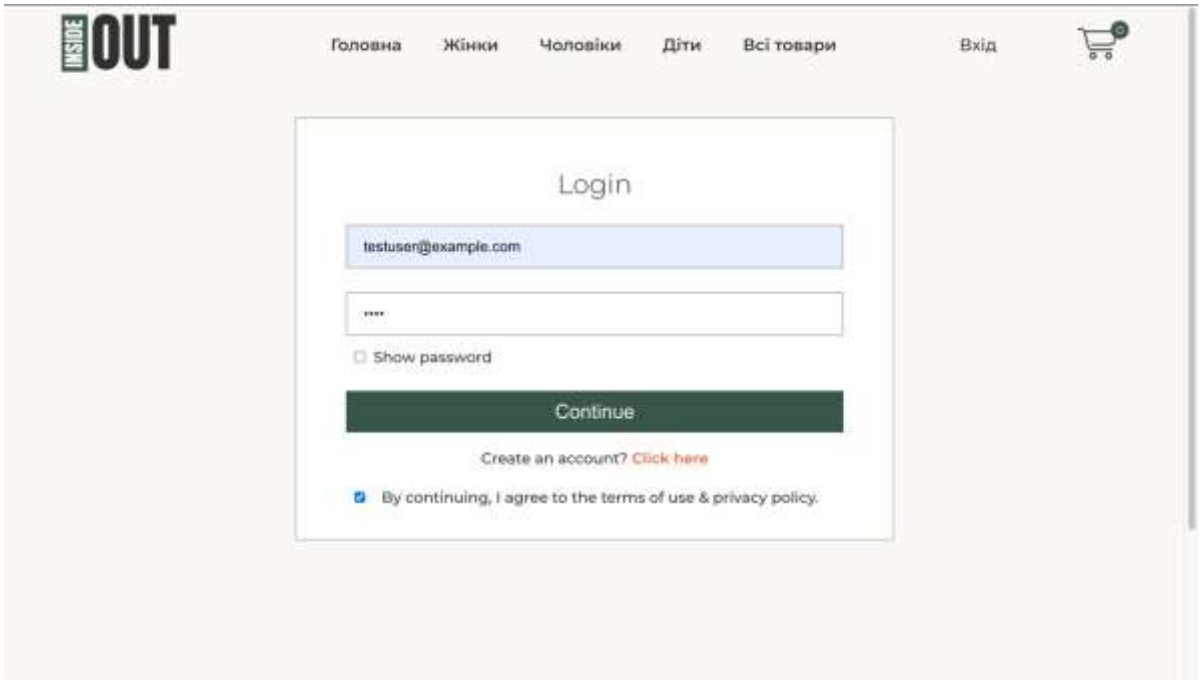

Рисунок 2.33 – Форма авторизації з введеними даними

Тест 3. Відображення головної сторінки

Вхідні дані: запуск вебсайту за посиланням.

Вихідні дані: відображення меню, кнопки «Вхід» та «Вихід», акцій, найпопулярніших, нових товарів, кнопки «Кошик», підписка на розсилку.

Результат тестування: головна сторінка відображає всі необхідні елементи.

Рисунок з результатом тестування: рисунок 2.35-2.38.

Примітка: відсутня.

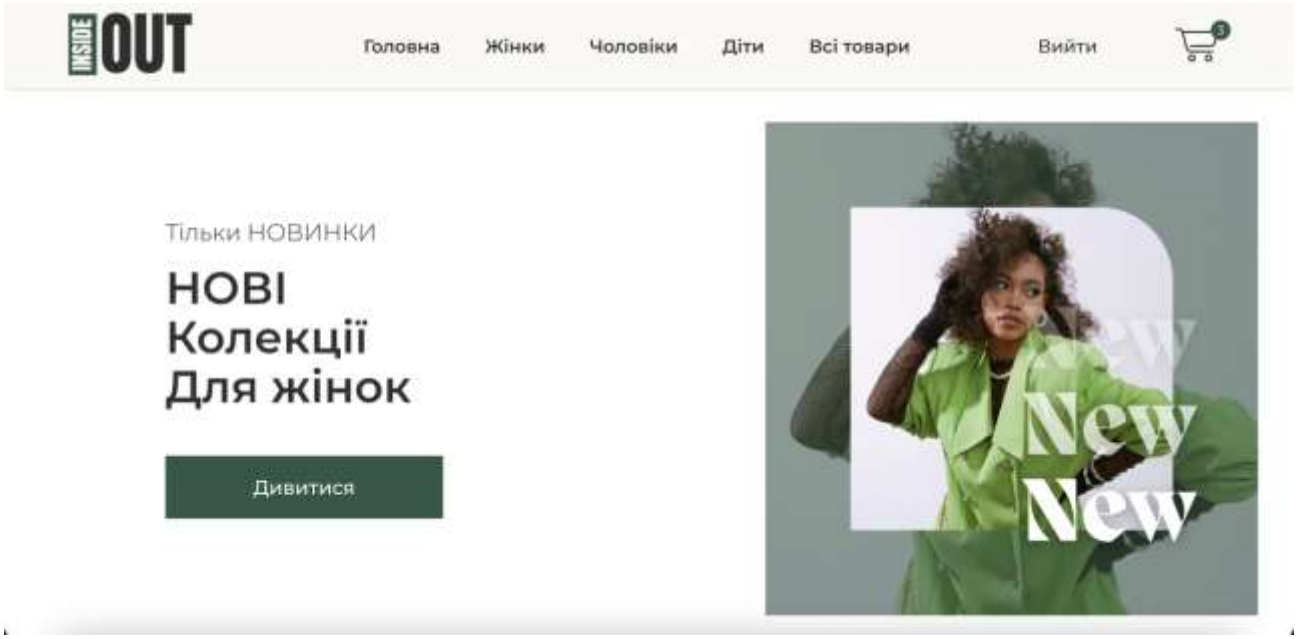

Рисунок 2.35 – Головна сторінка

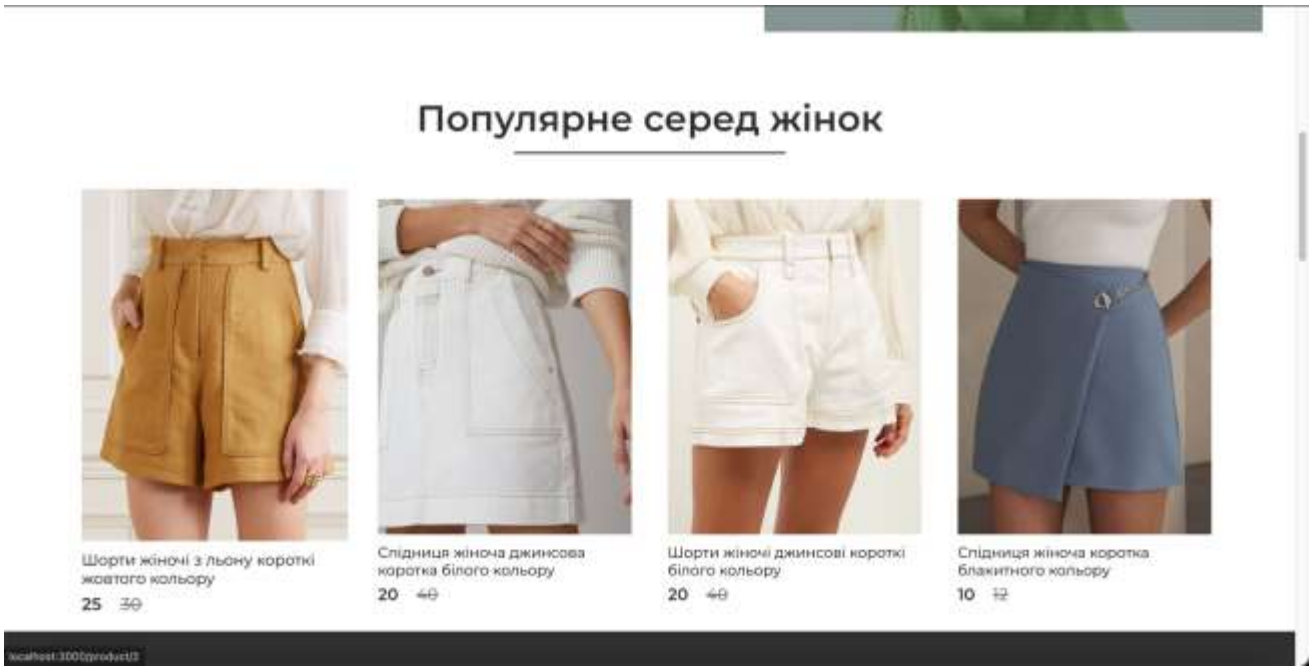

Рисунок 2.36 – Головна сторінка. «Популярне серед жінок»

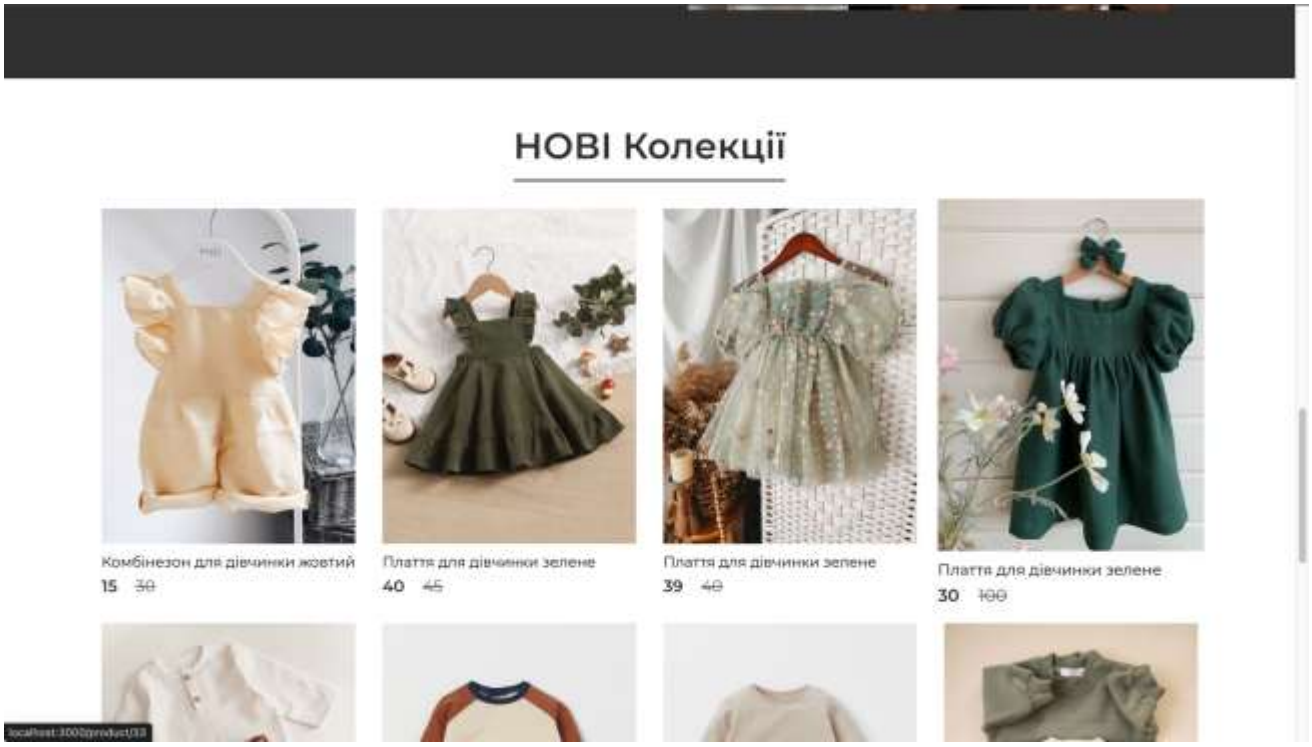

Рисунок 2.37 – Головна сторінка. «Нові колекції»

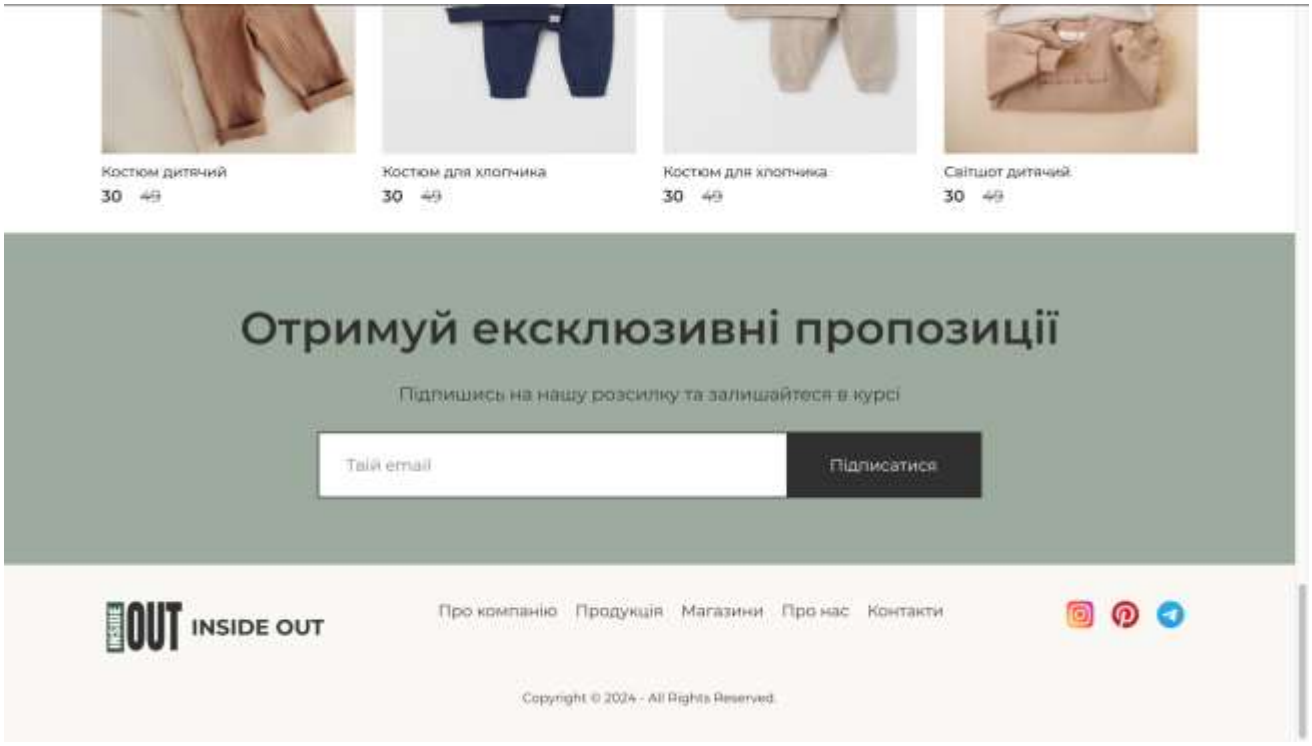

Рисунок 2.38 – Головна сторінка. «Підписка на розсилку»

Тест 4. Фільтрація каталогу товарів

Вхідні дані: вибір фільтру за назвою або ціною.

Вихідні дані: відфільтрований список товарів.

Результат тестування: товари просортовані за обраним критерієм.

Рисунок з результатом тестування: рисунок 2.39-2.42.

Примітка: сортування на даному етапі розробки не може бути одночасним по двом параметрам. На сторінках «Жінки», «Чоловіки» і «Діти» сортування таке саме.

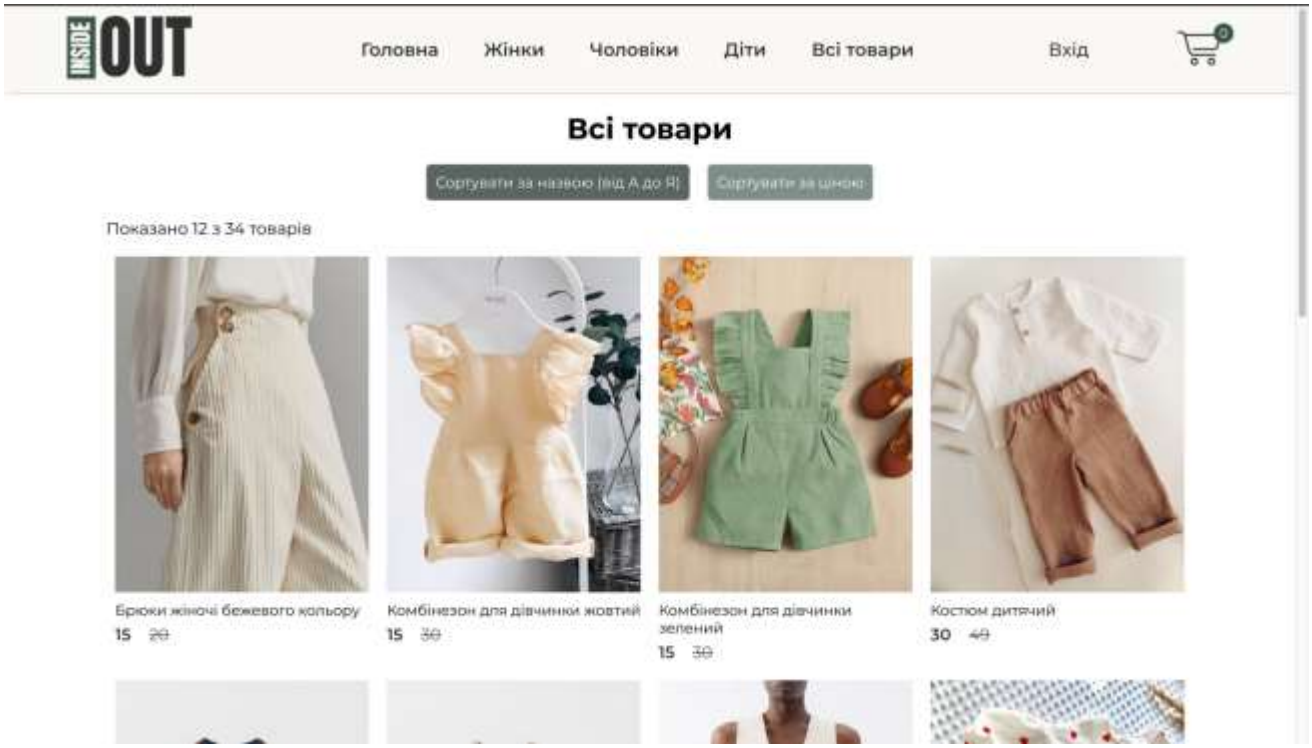

Рисунок 2.39 – Товари сортовані за алфавітом від А до Я

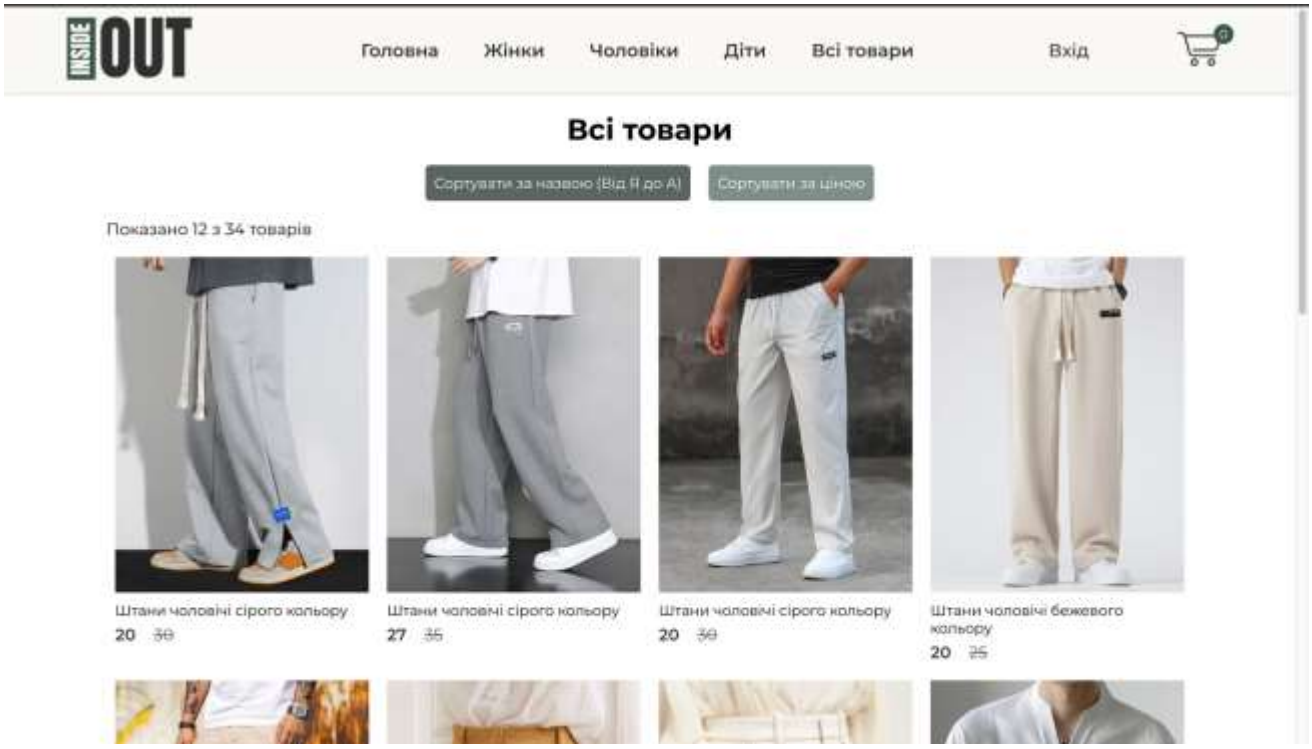

Рисунок 2.40 – Товари сортовані за алфавітом від Я до А

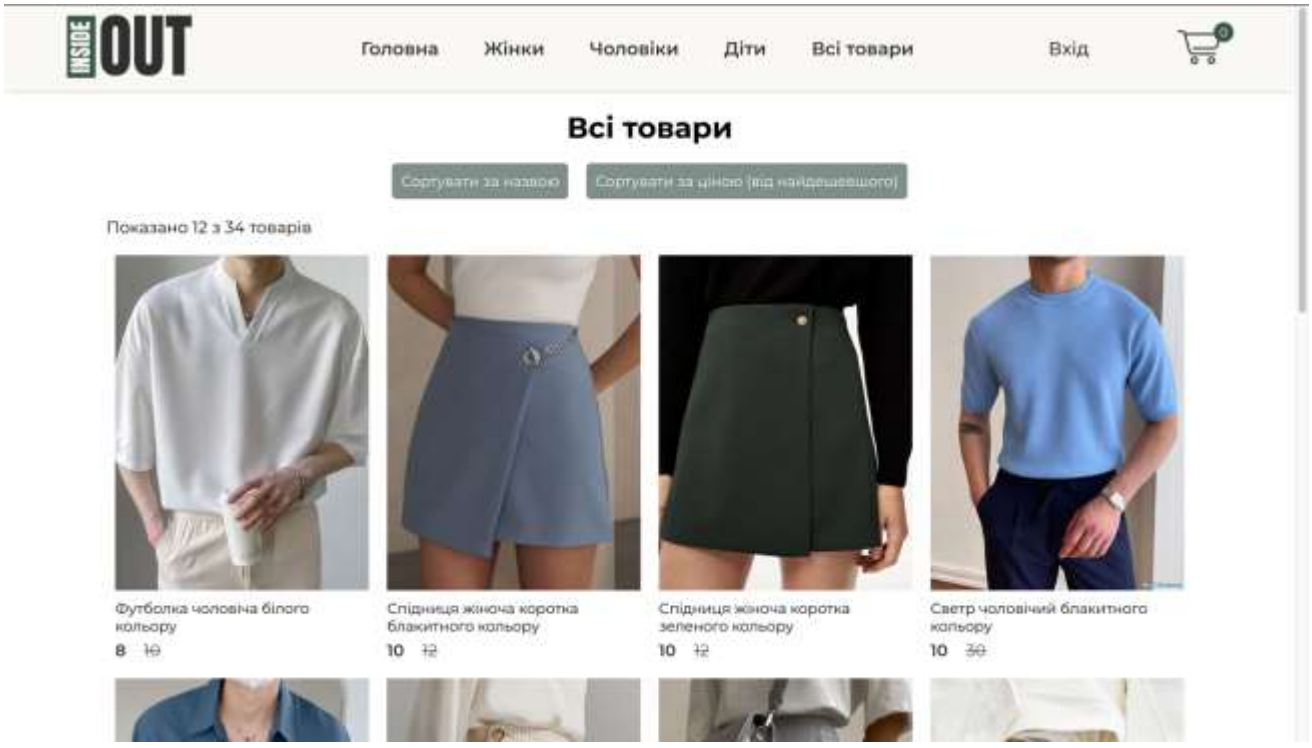

Рисунок 2.41 – Товари сортовані за ціною від найдешевшого

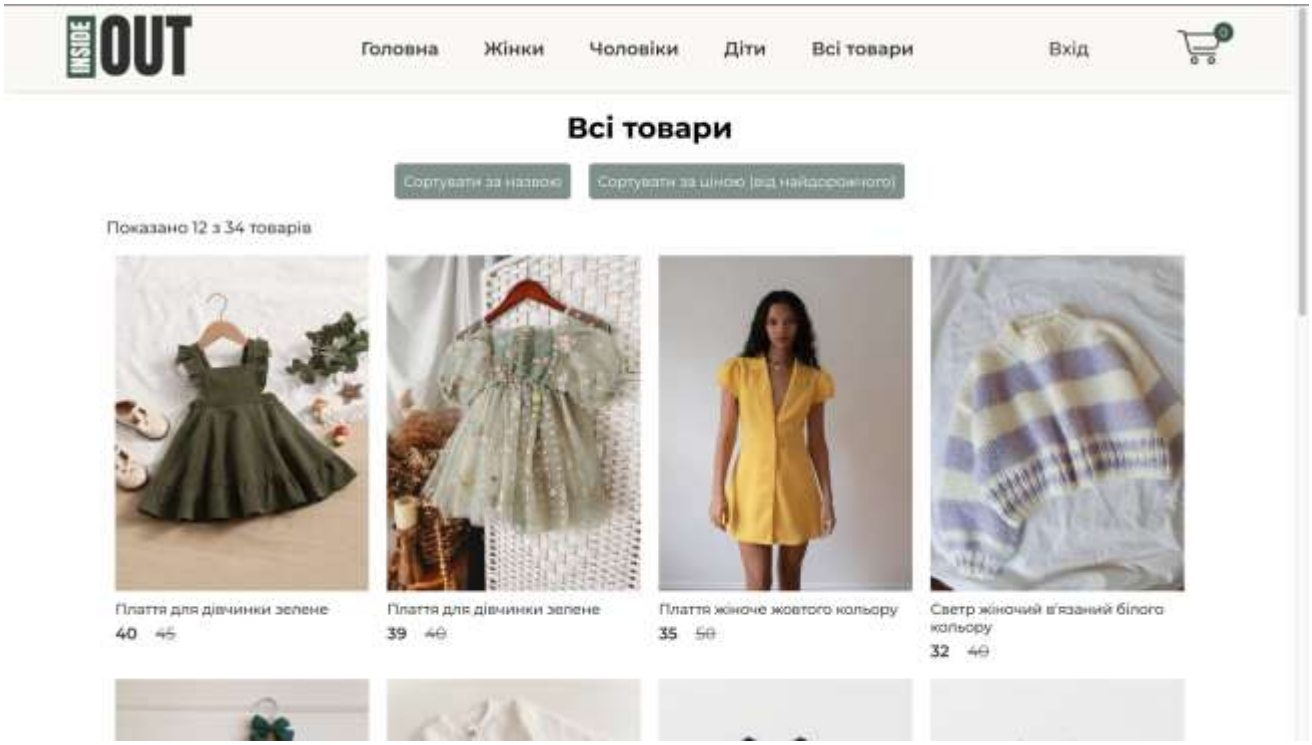

Рисунок 2.42– Товари сортовані за ціною від найдорожчого

Тест 5. Перегляд детальної інформації про товар

Вхідні дані: вибір товару з каталогу.

Вихідні дані: відображення детальної інформації про товар.

Результат тестування: користувач бачить детальну інформацію про товар (зображення, опис, розміри, матеріали, ціна).

Рисунок з результатом тестування: рисунок 2.43.

Примітка: відгуки нереалізовані.

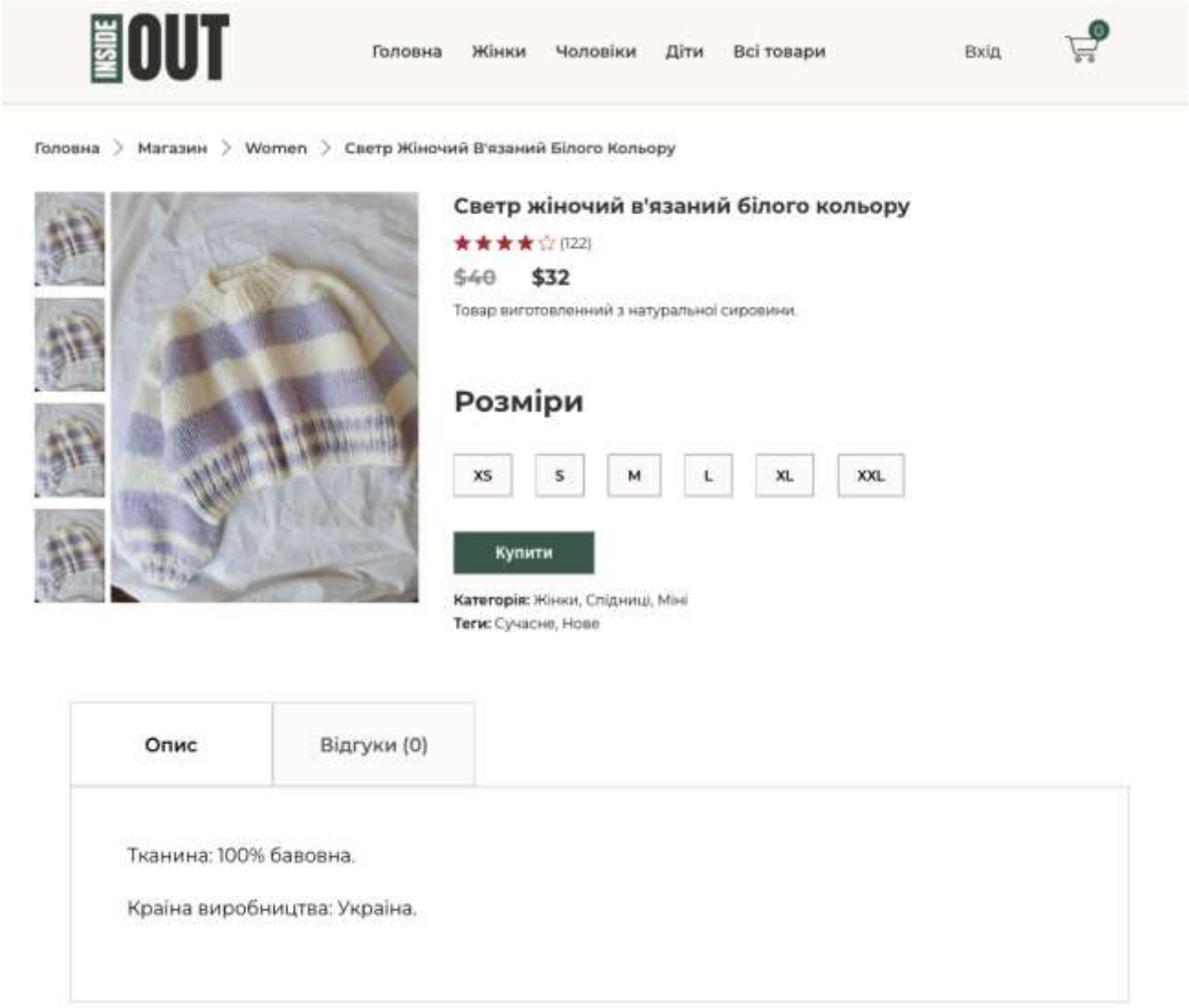

Рисунок 2.43 – Сторінка товару

Тест 6. Додавання товару до кошика

Вхідні дані: вибір товару та натискання кнопки "Купити".

Вихідні дані: товар додається до кошика.

Результат тестування: товар відображається в кошику.

Рисунок з результатом тестування: рисунок 2.44.

Примітка: вміст кошику не зберігається у неавторизованого користувача.

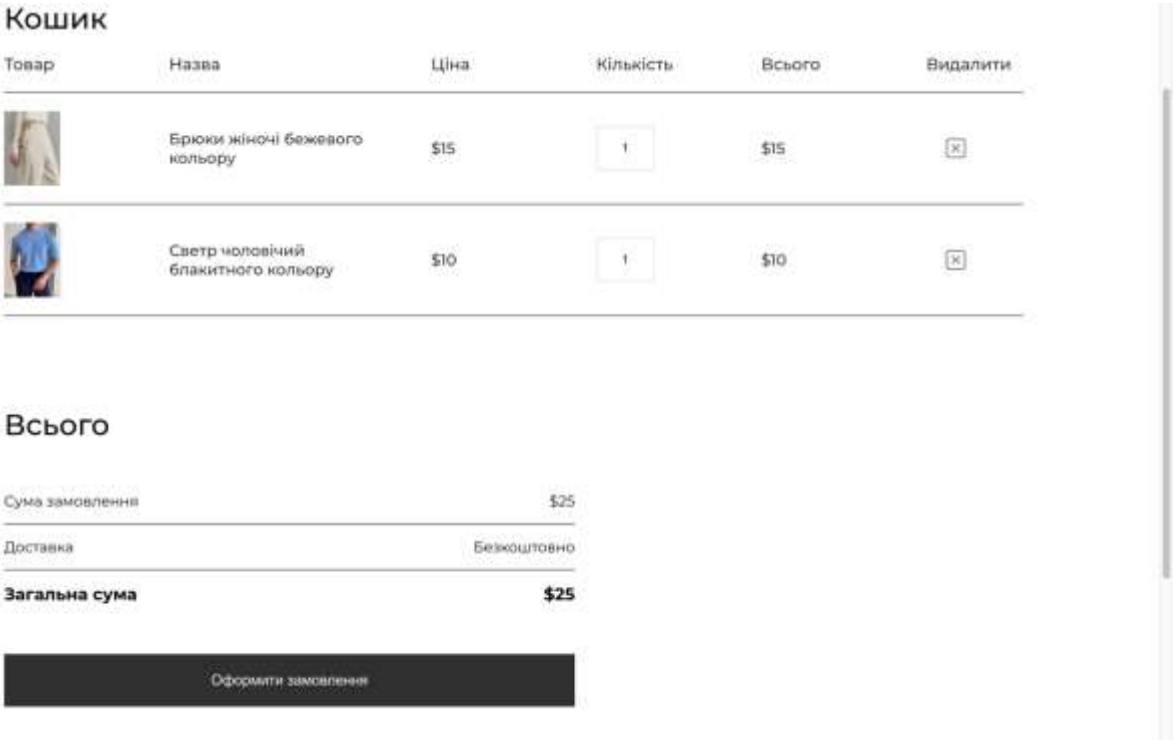

Рисунок 2.44 – Товари в кошику

Тест 7. Видалення товару з кошика

Вхідні дані: вибір товару в кошику та натискання іконки "Видалити".

Вихідні дані: товар видаляється з кошика.

Результат тестування: товар зникає з кошика.

Рисунок з результатом тестування: рисунок 2.45.

Примітка: відсутня.

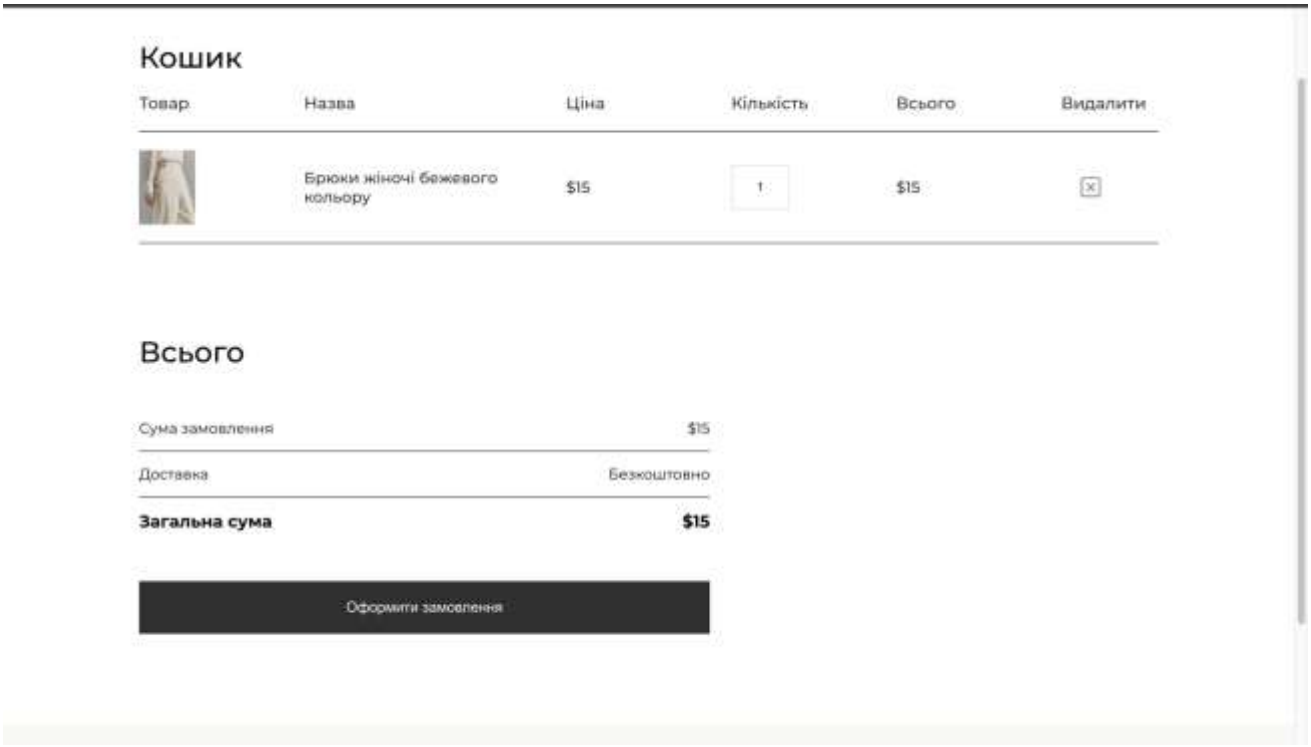

Рисунок 2.45 – Кошик після видалення товару «Светр чоловічий блакитного кольору»

Тест 8. Оформлення замовлення.

Вхідні дані: товари у кошику, натискання кнопки "Оформити замовлення".

Вихідні дані: оформлене замовлення, зв'язок з менеджером.

Результат тестування: замовлення успішно оформлене, менеджер зв'язується для уточнення деталей.

Рисунок з результатом тестування: рисунок 2.46.

Примітка: відсутня.

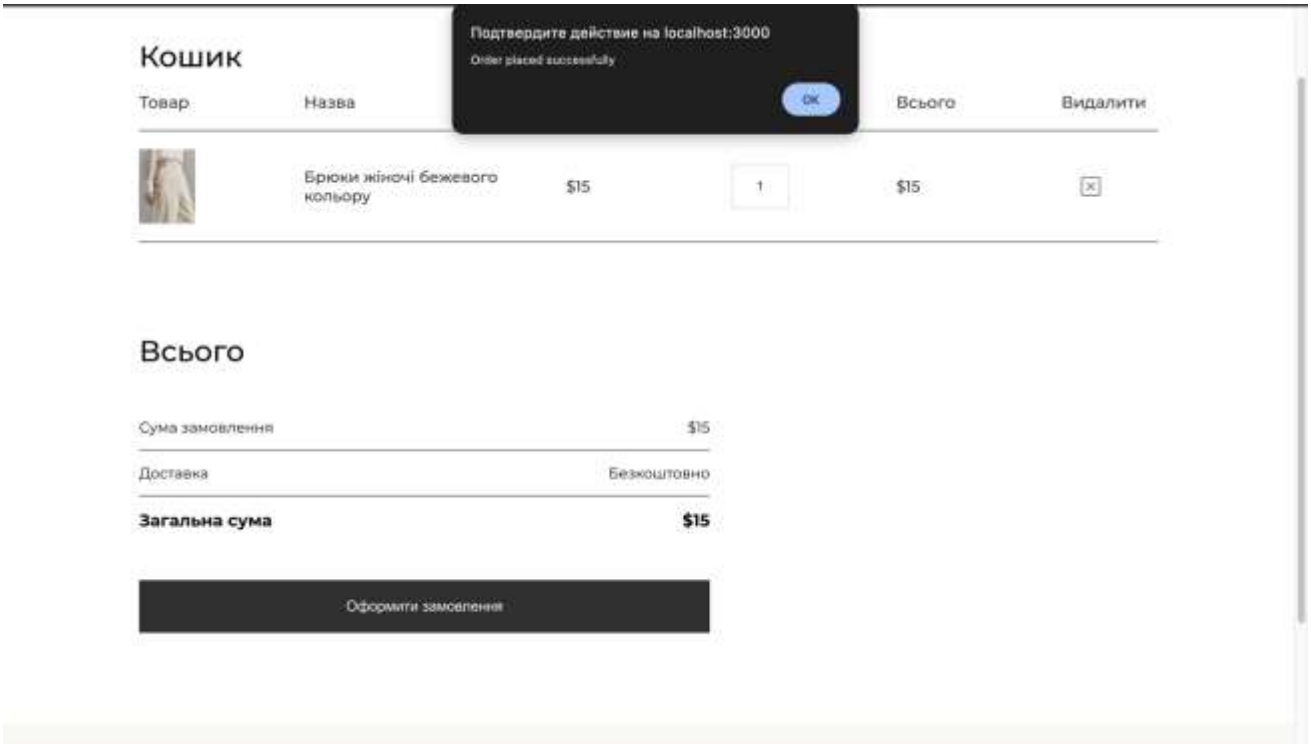

Рисунок 2.46 – Повідомлення про успішне розміщення замовлення

Тест 9. Двофакторна аутентифікація (2FA)

Вхідні дані: увімкнення 2FA для користувача.

Вихідні дані: запит на введення додаткового коду при вході.

Результат тестування: опція недоступна.

Рисунок з результатом тестування: відсутній.

Примітка: відсутня.

Тест 10. Реєстрація адміністратора

Вхідні данні: електронна пошта, номер телефону, пароль, ім’я.

Вихідні дані: успішна реєстрація адміністратора.

Результат тестування: адміністратор створює акаунт успішно.

Рисунок з результатом тестування: рисунок 2.47.

Примітка: відсутня.

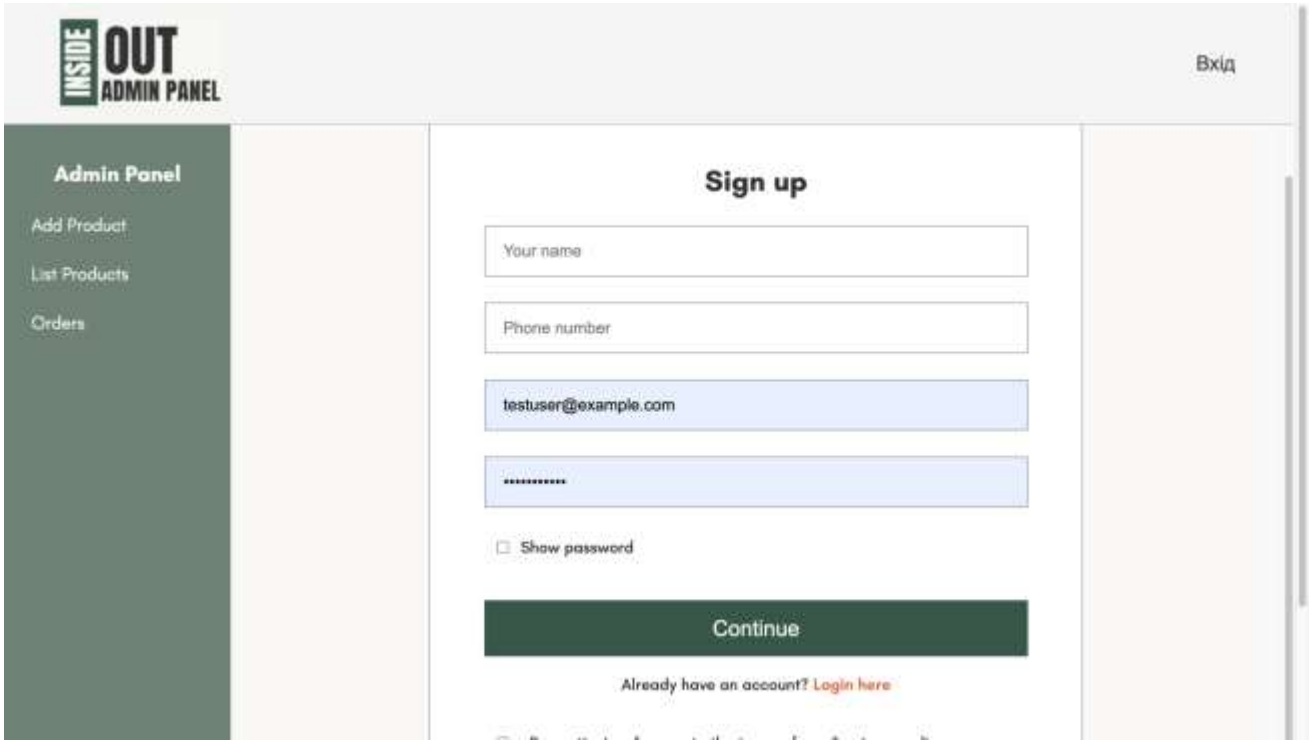

Рисунок 2.47 – Форма реєстрації адміністратора

Тест 11. Аутентифікація адміністратора.

Вхідні данні: електронна пошта, пароль.

Вихідні данні: після натискання кнопки «Continue» успішний вхід у систему.

Результат тестування: адміністратор входить у систему.

Рисунок з результатом тестування: рисунок 2.48.

Примітка: відсутня.

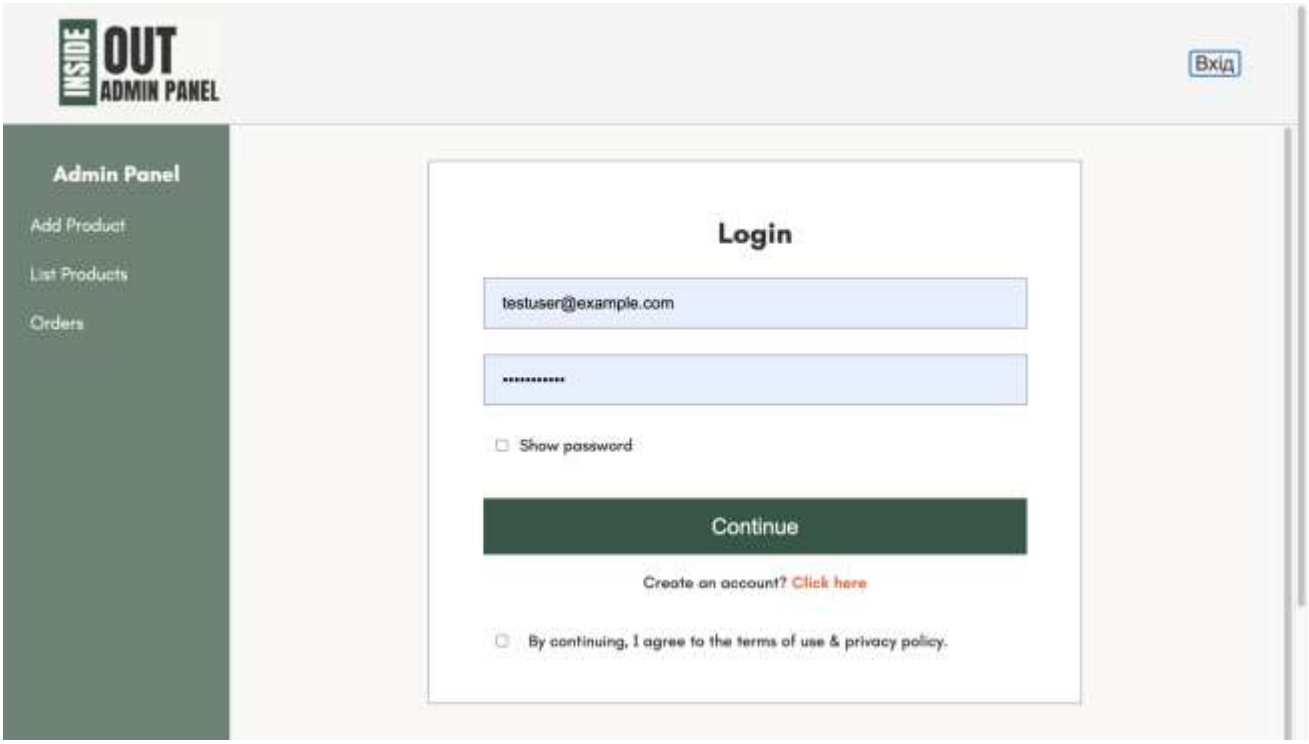

Рисунок 2.48 – Форма аутентифікації адміністратора

Тест 12. Додавання товару на сайт

Вхідні дані: інформація про товар (назва, ціна, категорія, фото).

Вихідні дані: товар додано на сайт.

Результат тестування: товар відображається на сайті.

Рисунок з результатом тестування: рисунок 2.49.

Примітка: відсутня.

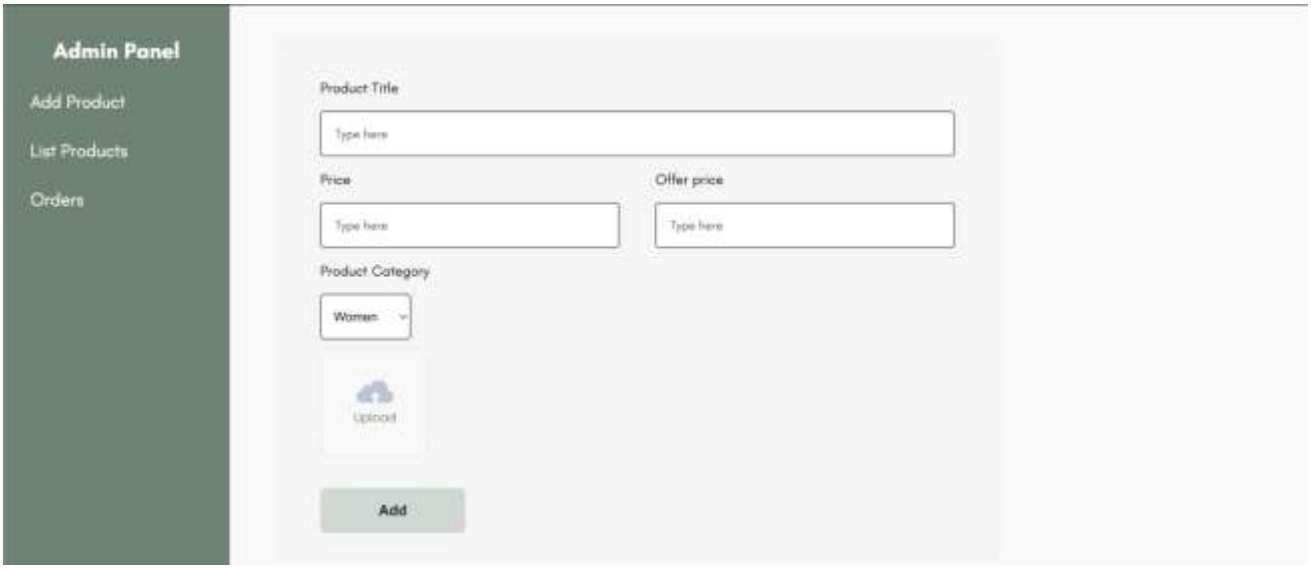

Рисунок 2.49 – Форма додавання товару на сайт

Тест 13. Видалення товару з сайту

Вхідні дані: вибір товару, натискання кнопки "Видалити".

Вихідні дані: товар видалено з сайту.

Результат тестування: товар зникає з сайту.

Рисунок з результатом тестування: рисунок 2.50-2.51.

Примітка: видалятися буде «Платтяя для дівчинки зелене».

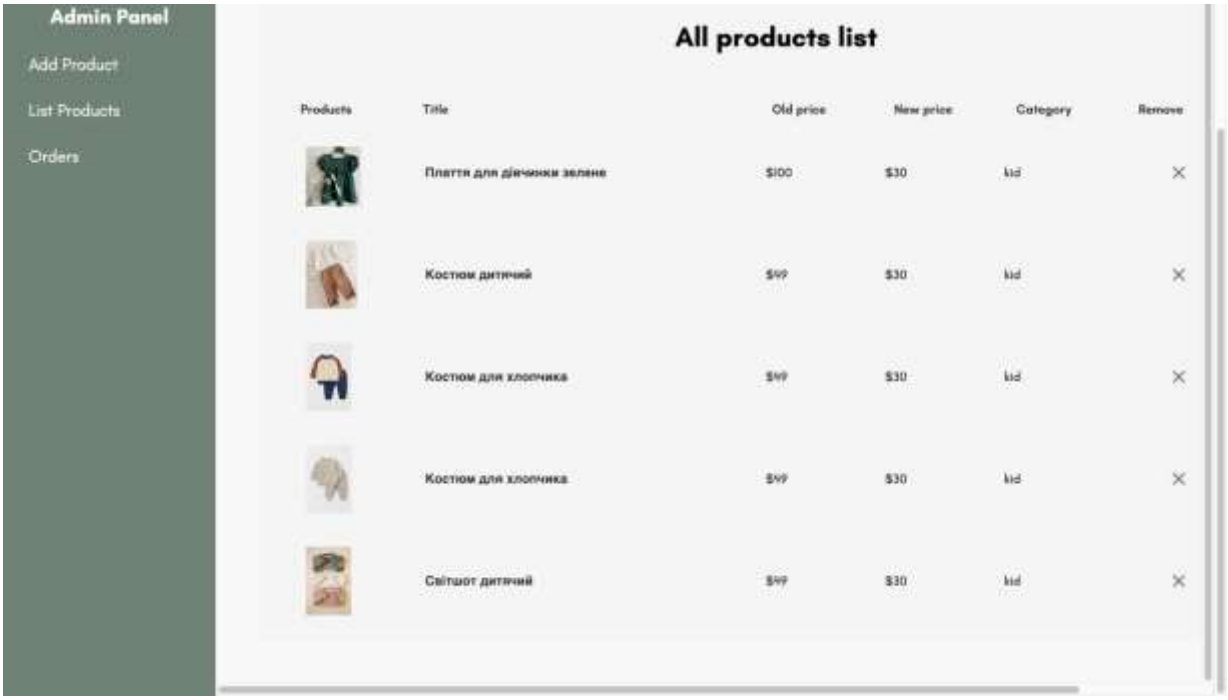

Рисунок 2.50 – Список товарів до видалення елементу

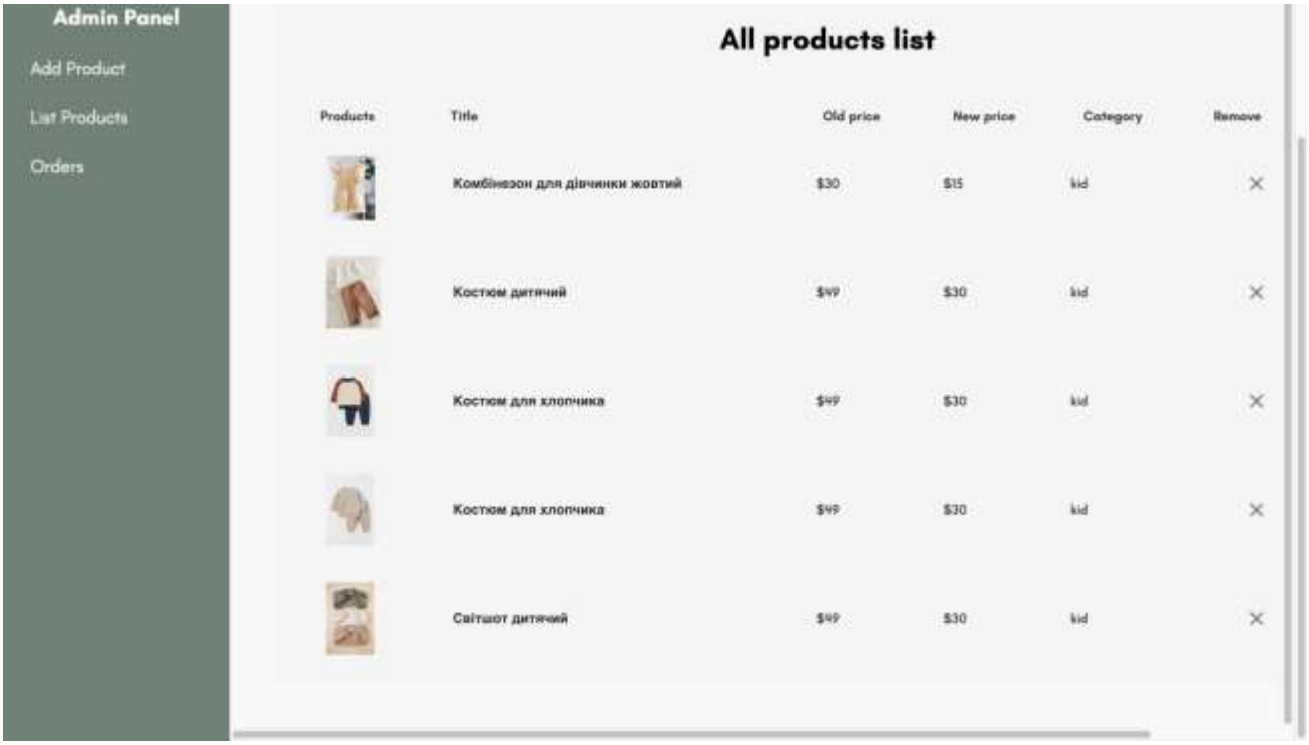

Рисунок 2.51 – Список товарів після видалення «Плаття для дівчинки зелене»

Тест 14. Перегляд замовлення

Вхідні дані: вибір вкладки «Orders» в адмін-панелі.

Вихідні дані: відображення інформації про замовлення.

Результат тестування: адміністратор бачить інформацію про замовлення.

Рисунок з результатом тестування: рисунок 2.52.

Примітка: відсутня.

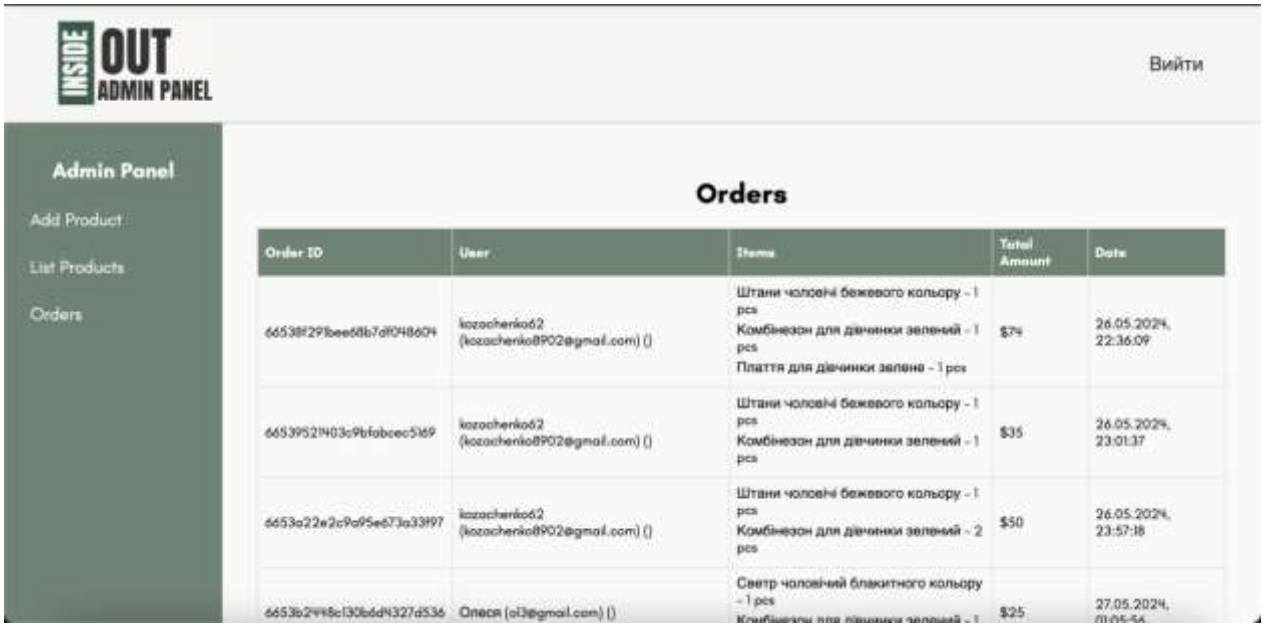

Рисунок 2.52 – Список замовлень

Тест 15. Безпека

Вхідні дані тестування шифрування даних.

Вихідні дані: захищеність паролів.

Результат тестування: паролі користувачів шифруються.

Рисунок з результатом тестування: рисунок 2.53.

Примітка: відсутня.

Рисунок 2.53 – Приклад в базі даних користувача з шифруванням пароля

Тест 16. Адаптивність

Вхідні дані: користування сайтом на IPhone 12.

Вихідні дані: відображення елементів сайту.

Результат тестування: інтерфейс зручний і зрозумілий, адаптивний дизайн.

Рисунок з результатом тестування: рисунок 2.54.

Примітка: відсутня.

Рисунок 2.54 – Відображення головної сторінки на IPhone 12

Тест 17. Дизайн та сумісність

Вхідні дані: перевірка відображення сайту на різних браузерах:Opera, Safari, Firefox.

Вихідні дані: якість відображення на різних браузерах:Opera, Safari, Firefox.

Результат тестування: сайт коректно відображається на всіх браузерах: Opera, Safari, Firefox.

Рисунок з результатом тестування: рисунок 2.55-2.57.

Примітка: відсутня.

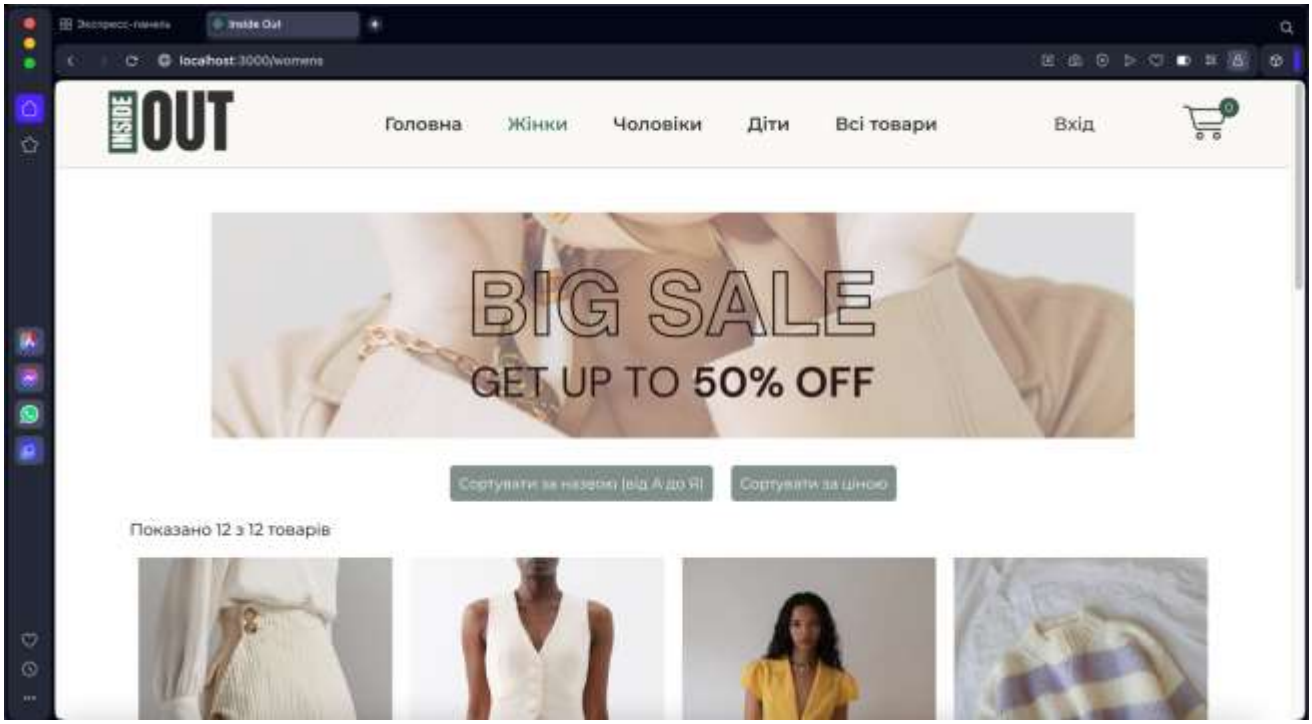

Рисунок 2.55 – Відображення вебсайту в Opera

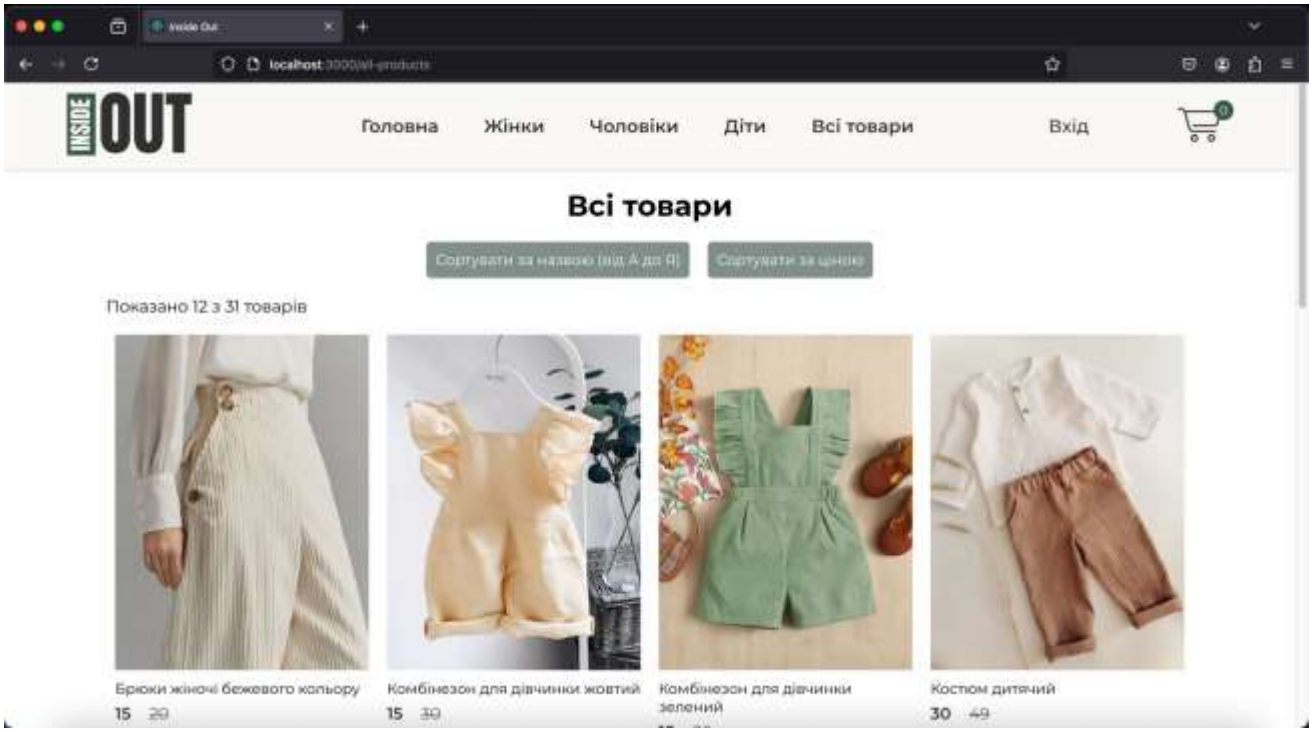

Рисунок 2.56 – Відображення вебсайту в Firefox

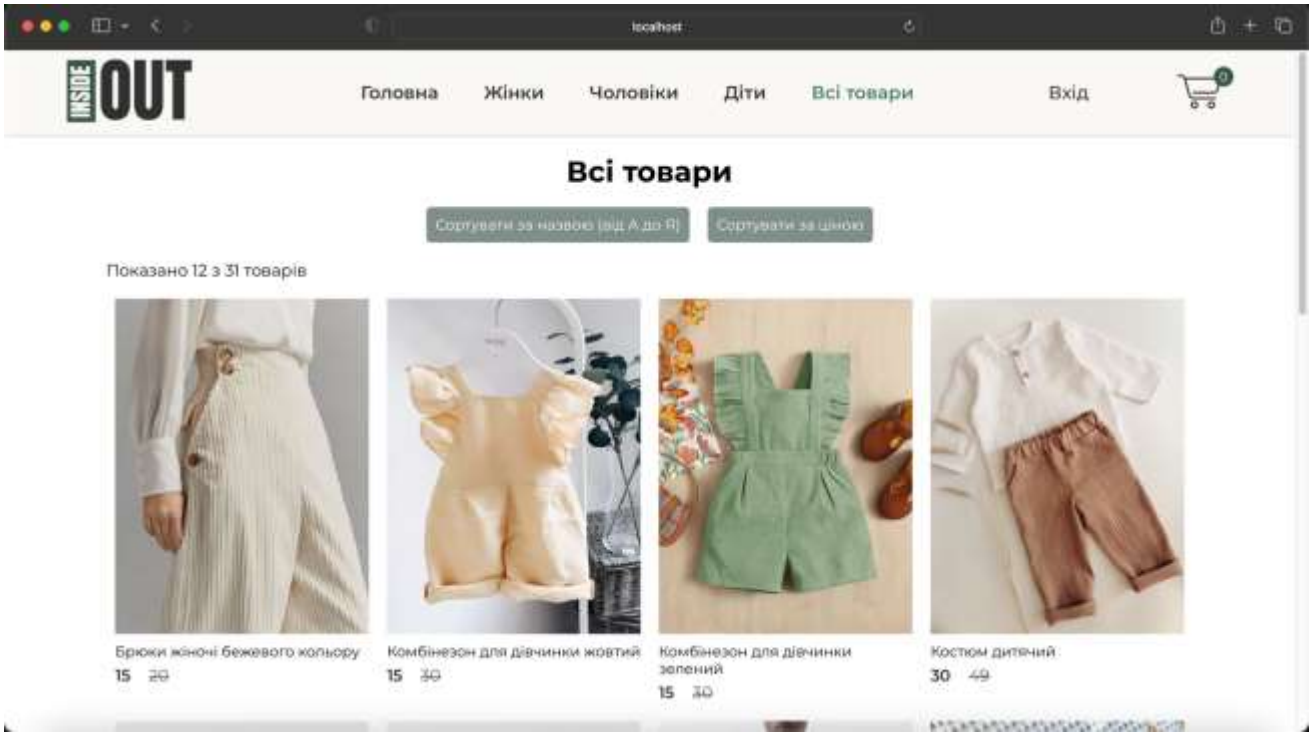

Рисунок 2.57 – Відображення вебсайту в Safari

Тест 18. VPN

Вхідні дані: перевірка відображення сайту з іншої країни.

Вихідні дані: якість відображення в іншій країні через VPN.

Результат тестування: сайт коректно відображається.

Рисунок з результатом тестування: рисунок 2.58.

Примітка: відсутня.

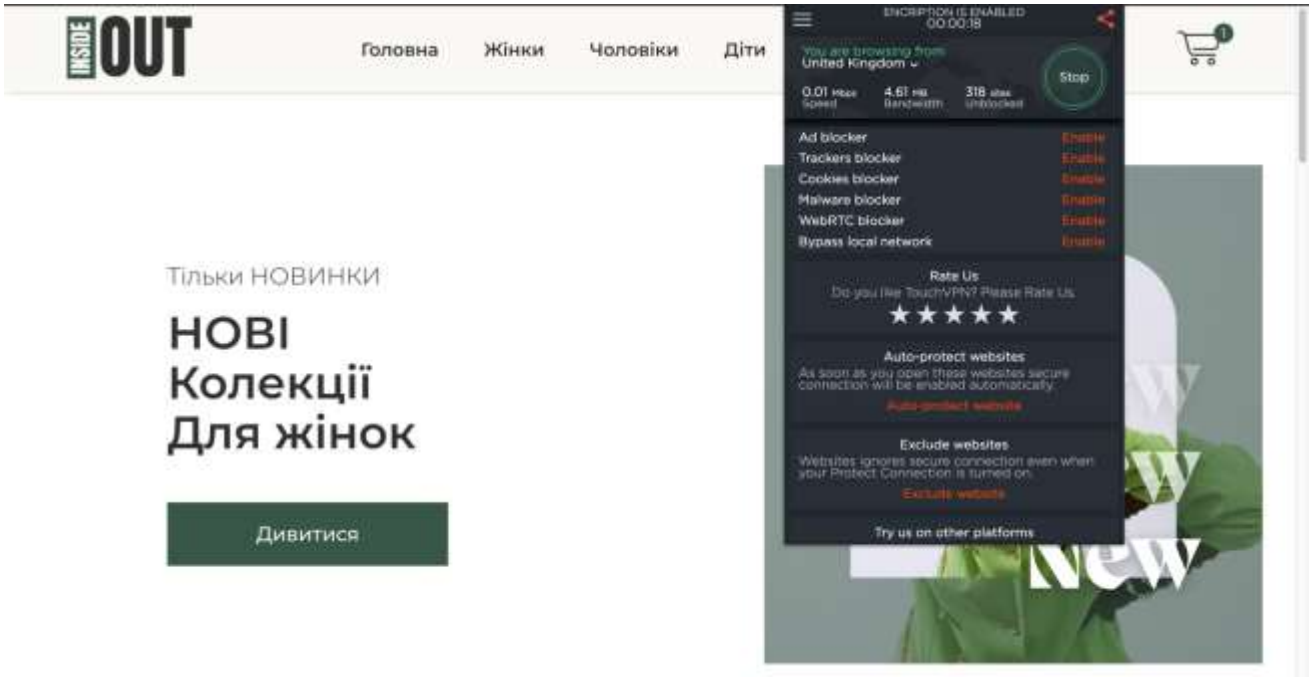

Рисунок 2.58 – Відображення вебсайту з ввімкненим VPN

Щоб підсумувати всі тести та наглядно відобразити результати тестування нижче буде наведено таблицю 2.8, де відображається Checklist.

Таблиця 2.8 – Checklist тестів

№	Назва тесту	Результат	Наявність
1	2	3	4
1	Реєстрація користувача	Реалізовано	Не виявлено
2	Аутентифікація користувача	Реалізовано	Не виявлено
3	Відображення головної сторінки	Реалізовано	Не виявлено
4	Фільтрація каталогу товарів	Реалізовано частково	Не виявлено
5	Перегляд детальної інформації про товар	Реалізовано частково	Не виявлено
6	Додавання товару в кошик	Реалізовано	Не виявлено
7	Видалення товару з кошика	Реалізовано	Не виявлено

Продовження таблиці 2.8

1	2	3	4
8	Оформлення замовлення	Реалізовано	не виявлено
9	Двохфакторна аутентифікація	Не реалізовано	не виявлено
10	Реєстрація адміністратора	Реалізовано	не виявлено
11	Аутентифікація адміністратора	Реалізовано	не виявлено
12	Додавання товару на сайт	Реалізовано	не виявлено
13	Видалення товару з сайту	Реалізовано	не виявлено
14	Перегляд замовлень	Реалізовано	не виявлено
15	Безпека	Реалізовано	не виявлено
16	Адаптивність	Реалізовано	не виявлено
17	Дизайн і сумісність	Реалізовано	не виявлено
18	VPN	Реалізовано	не виявлено

Результати тестів, що були проведені

Під час тестування вебсайту і адмін-панелі було проведено вісімнадцять тестів. На рисунку 2.59 зображена діаграма тестів, які були проведені.

Рисунок 2.59 – Діаграма результату тестів

2.3.4 Випробування програмного забезпечення

Метою випробувань було визначення функціональної робото-стійкості вебдодатку. Програма випробувань перевірила вебдодаток на відповідність вимогам, викладеним в технічному завданні (Додаток А). Згідно з програмою і методикою випробувань, що представлені в Додатку Б, були виконані випробування вебдодатку. Більша частина проєкту відповідає вимогам, відсутні помилки і грубі недоліки. Основні функції були реалізовані, тому вебдодаток готовий до експлуатації.

При випробуванні було розглянуто всі функції «Inside out» та адмін-панелі, але результати випробувань представлено тільки декількох функцій, що були реалізовані:

- Функція додавання нових товарів на сайт;
- Функція оформлення замовлення;
- Функція реєстрації для користувачів сайту.

Випробування функції додавання нових товарів.

Після відкриття адмін-панелі відображається три вкладки, після натискання на вкладку «Add Product» відкривається форма з наступними елементами:

- назва товару;
- категорія;
- стара ціна;
- нова ціна;
- фото;
- кнопка «Add».

Після вводу всієї інформації в поля та після натискання кнопки «Add» товар зберігається в базу даних та відображається на сайті.

Випробування функції оформлення замовлення.

Після того як користувач авторизувався/зареєструвався на сайті та додав товар в кошик, переходить в кошик та натискає кнопку «Оформити замовлення». Відображається повідомлення про те, що замовлення успішно оформлене. Після чого з користувачем зв’язується менеджер для уточнення деталей. Якщо користувач не авторизований, то система повідомляє про помилку.

Випробування функції реєстрації для користувачів сайту.

Після відкриття посилання на сайт користувач бачить на навігаційній панелі кнопку «Вхід». Якщо користувач не має акаунту, то на тикай випадок є підказка «Create an account? Click here», де «Click here» є кнопкою-посиланням на форму реєстрації. Форма реєстрації в себе включає такі елементи:

- ім’я нового користувача;
- номер телефону;
- електрона пошта;
- пароль;
- функція «показати пароль»;
- кнопка «Continue».

Новий користувач вводить всі необхідні дані у поля форми, натискає кнопку «Continue» та система його перекидає на головну сторінку сайту, після чого клієнт може надалі користуватися функціями сайту. У разі збігу електронною пошти у базі даних система повідомляє про помилку.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБСАЙТУ ІНТЕРНЕТ-МАГАЗИНУ ОДЯГУ “INSIDE OUT”

У результаті виконання кваліфікаційної роботи були розроблені інтернет-магазин «Inside out» та адмін-панель, що складаються з таких функцій:

- реєстрація користувачів/адміністраторів в системі;
- авторизація користувачів/адміністраторів в системі;
- перегляд товарів на сайті;
- сортування товарів;
- додавання товару в кошик;
- видалення товару;
- оформлення замовлення;
- додавання товару на сайт;
- видалення товару з сайту;
- перегляд замовлень;
- підписка на розсилку.

Був проведений аналіз вже існуючих інтернет-магазинів, обґрунтовано мотивацію створення вебсайту інтернет-магазину.

У процесі розробки були розроблені: ескізний, технічний та робочий проєкти, проведено налагодження та тестування, створені: наочний та інтуїтивно зрозумілий інтерфейс, завдяки якому користувачу приємно та легко робити онлайн-шопінг .

У рамках ескізного проєкту було побудовано діаграму варіантів використання, описані специфікації варіантів використання, розроблені діаграми діяльності та концептуальна модель даних у вигляді діаграми сутність-зв’язок, представлено проєкт користувацького інтерфейсу.

У рамках технічного проєкту програмного забезпечення було створено статичну модель програми (діаграма класів), описані специфікації класів, були

побудовані діаграми послідовності до варіантів використання та була розроблена логічна модель даних.

У рамках робочого проєкту програмного забезпечення було створено діаграму компонентів, описано вибір інструментів для реалізації, проведено функціональне тестування, створено checklist. Описане випробування вебсайту та адмін-панелі, що підтвердило працездатність системи.

Технічне завдання до вебсайту «Inside Out» та адмін-панелі представлено у додатку А. Програма та методика випробувань вебсайту «Inside Out» та адмін-панелі представлена у додатку Б. Була розроблена інструкція користувача, що розміщена у Додатку В. Опис вебсайту «Inside Out» та адмін-панелі представлено у Додатку Г. Текст програми представлений у Додатку Д.

Для створення вебсайту та адмін-панелі були використані такі інструменти як Node.js(для серверної частини), React.js(JavaScript-бібліотека), MongoDB(документно-орієнтована база даних NoSQL), Express(веб-фреймворк для Node.js), CSS(мова стилізації) та Visual Studio Code для редагування коду. Для роботи з вебсайтом та адмін-панеллю має бути встановлений будь-який браузер на комп’ютері або телефоні.

Результатом роботи є функціональний, адаптивний вебсайт інтернет-магазин одягу та адмін-панель до нього.

У майбутньому вебсайт та адмін-панель можна розширити та удосконалити за рахунок додавання таких нових функцій:

- доробити кабінет користувача, де буде інформація про акаунт, про замовлення, улюблені товари, особисті дані, можливість зміни паролю і так далі;
- доробити кабінет адміністратора з особистою інформацією;
- підключити системи оплати;
- зробити форму оформлення замовлення, де буде зазначатися детальна інформація про доставку;
- зробити більш розширене сортування товарів;
- доробити функціонал сторінки товару(наприклад, коментарі);

- розширити можливості адмін-панелі: додати можливість змінювати банери, робити розсилку акцій користувачам, які підписалися.

4 ОХОРОНА ПРАЦІ

Охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці.

Закон України визначає основні напрямки по реалізації конституційного права на охорону їхнього життя і здоров'я в процесі трудової діяльності, регулюють при участі відповідних державних органів відносини між власником підприємства, чи установи організації, чи органом і роботодавцем з питань зовнішнього середовища і встановлює єдиний порядок організації охорони праці в Україні.

Законодавство по охороні праці складається з дійсного Закону, Кодексу законів про працю України й інших нормативних актів. У випадку, коли міжнародними договорами чи угодами, у яких бере участь Україна, установлені більш високі вимоги до охорони праці, чим ті, котрі передбачені законодавством України, застосовуються правила міжнародного договору чи угоди.

4.1 Аналіз потенційно небезпечних і шкідливих факторів при роботі програміста

Співробітник, що експлуатує засоби обчислювальної техніки і периферійне устаткування може піддаватися небезпечним і шкідливим впливу, що по природі дії підрозділяються на наступні групи:

- ураження електричним струмом;
- механічні ушкодження;
- електромагнітне випромінювання;
- інфрачервоне випромінювання;

- небезпека пожежі
- підвищений рівень шуму і вібрації

Для зниження чи запобігання впливу небезпечних і шкідливих факторів необхідно дотримувати санітарні правила і норми, гігієнічні вимоги до відео дисплейним терміналам, персональним електронно-обчислювальним машинам і організації роботи.

4.2 Вимоги до організації й устаткування робочих місць

Робоче приміщення із відео дисплейним терміналом і ПК повинні мати природне і штучне освітлення.

Основне навантаження при роботі за комп'ютером приходиться на очі. Їхня стомлюваність багато в чому залежить не тільки від якості зображення на екрані, але і від загальної освітленості приміщення. У той час як для звичайних офісів рекомендується освітленість до 1600 люкс, для робочих місць, оснащених відеотерміналами, рекомендується освітленість 100-500 люкс. Відповідно до гігієнічних норм, освітленість на поверхні столу і клавіатурі повинна бути не менш 300 люкс, а вертикальна освітленість екрана - 100-250 люкс. Дослідження фізіологів і гігієністів переконливо довели, що і напівтемрява, і занадто висока освітленість екрана приводять до швидкого зорового стомлення.

Розміщати комп'ютер рекомендується так, щоб (природний чи штучний) падав збоку, краще ліворуч, це позбавить вас від тіней, що заважають, і допоможе знизити освітленість екрана. Як джерела висвітлення рекомендується застосовувати люмінесцентні лампи типу ЛБ зі світильниками серії ЛПО36 із зазеркалюванними ґратами. Лампи накалювання краще використовувати для місцевого висвітлення зони робочого документа (клавіатури, книги, зошита). Постарайтеся, щоб люстра у вашій робочій кімнаті мала закриті знизу світильники, так щоб на екран монітора падало неуважно-відбите світло. Це позбавить вас від відблисків і полегшить зорову роботу. А от настільна лампа,

навпаки, повинна мати щільний, непросвічуючий абажур, що направляє світло прямо в зону робочого документа.

Умови зовнішнього висвітлення часто впливають на оцінку якості передачі кольору й інших параметрів відображення. Багато виробників, такі як Mitsubishi і Panasonic, борються із зовнішніми факторами, зменшуючи кривизну екрана, аж до створення зовсім плоских екранів. За даними Panasonic, у моделі PanaFlat PF70, що випускається цією компанією, відблиски в порівнянні зі звичайними ЭЛТ зменшені на 87%. Також є ряд інших засобів, що дозволяють боротися з зовнішнім світлом, - спеціальні багатошарові покриття і каптури, такої компанії, що поставляються як з моделями серії Electron, LaCie.

Робочі місця з відео дисплейним терміналом і ПК стосовно світлових прорізів повинні розташовуватися так, щоб природне світло падало збоку, переважно ліворуч. Схеми розміщення робочих місць з відео дисплейним терміналом і ПК повинні враховувати відстані між робочими столами з відеомоніторами (у напрямку тилу поверхні одного відеомонітора й екрана іншого відеомонітора), що повинне бути не менш 2,0 м., а відстань між бічними поверхнями відеомоніторів - не менш 1,2 м. Віконні прорізи в приміщеннях використання відео дисплейним терміналом і ПК повинні бути обладнані регульованими пристроями типу жалюзі й ін.

При конструюванні устаткування й організації робочого місця співробітника з відео дисплейним терміналом і ПК варто забезпечити відповідність конструкції всіх елементів робочого місця і їхній взаємного роботи.

Робочий стілець (крісло) повинен бути під’ємно - поворотним і регулюватися по висоті і кутам нахилу сидіння і спинки, а також відстані спинки від переднього краю сидіння, при цьому регулювання кожного параметра повинно бути незалежним, легко здійснюватися і мати надійну фіксацію. Поверхня сидіння, спинки й інших елементів стільця (крісла) повинна бути напівм'якої, з нековзним, що не електризується і повітропроникним покриттям, що забезпечує легке очищення від забруднень. Екран відеомонітора повинен знаходитись від

очей користувача на оптимальній відстані 600 - 700 мм, але не ближче 500 мм з урахуванням розмірів алфавітно-цифрових знаків і символів.

Висота робочої поверхні столу для дорослих користувачів повинна регулюватися в межах 680 - 800 мм; при відсутності такої можливості висота робочої поверхні столу повинна складати 725 мм. Модульними розмірами робочої поверхні столу для відео дисплейного термінала і ПК, на підставі яких повинні розраховуватися конструктивні розміри, варто вважати: ширину 800, 1000, 1200 і 1400 мм, глибину 800 і 1000 мм при нерегульованій його висоті, рівної 725 мм. Робочий стіл повинен мати простір для ніг висотою не менш 600 мм, шириною - не менш 500 мм, глибиною на рівні колін - не менш 450 мм і на рівні витягнутих ніг - не менш 650 мм. Робочий стілець (крісло) повинен бути підъємно - поворотним і регульованої по висоті і кутам нахилу сидіння і спинки, а також - відстані спинки від переднього краю сидіння.

Конструкція його повинна забезпечувати:

- ширину і глибину поверхні сидіння не менш 400 мм;
- поверхня сидіння з закругленим переднім краєм;
- регулювання висоти поверхні сидіння в межах 400 - 550 мм і кутів нахилу вперед до 15 град. і назад до 5 град.;
- висоту опорної поверхні спинки 300 +- 20 мм, ширину - не менш 380 мм і радіус кривизни горизонтальної площини - 400 мм;
- кут нахилу спинки у вертикальній площині в межах 0 +- 30 градусів;
- регулювання відстані спинки від переднього краю сидіння в межах 260 - 400 мм;
- стаціонарні чи знімні підлокітники довжиною не менш 250 мм і шириною 50 - 70 мм;
- регулювання підлокітників по висоті над сидінням у межах 230 +- 30 мм і внутрішньої відстані між підлокітниками в межах 350 - 500 мм.

Робоче місце повинне бути обладнане підставкою для ніг, що має ширину не менш 300 мм, глибину не менш 400 мм, регулювання по висоті в межах до 150 мм

і по куті нахилу опорної поверхні підставки до 20 градусів. Поверхня підставки повинна бути рифленої і мати по передньому краї бортик висотою 10 мм:

- простір по глибині не менш 850 мм з урахуванням виступаючих частин устаткування для перебування людини - оператора;
- простір для стіп глибиною і висотою не менш 150 мм і шириною не менш 530 мм;
- розташування пристроїв уведення - висновку інформації, що забезпечує оптимальну видимість екрана;
- легку досяжність органів ручного керування в зоні моторного поля: по висоті - 900 - 1300 мм, по глибині - 400 - 500 мм;
- розташування екрана ВДТ чи ПК у місці робочої зони, що забезпечує зручність зорового спостереження у вертикальній площині під кутом +- 30 градусів від нормальної лінії погляду оператора, а також зручність використання ВДТ чи ПК (уведення - висновок інформації при коректуванні основних параметрів технологічного процесу, налагодження програм і ін.) одночасно з виконанням основних виробничих операцій (спостереження за зоною обробки на верстаті з програмним керуванням, при обслуговуванні роботизірованого технологічного комплексу й ін.);
- можливість повороту екрана ВДТ чи ПК навколо горизонтальної і вертикальної осей.

Клавіатуру варто розташовувати на поверхні столу на відстані 100 - 300 мм від краю, зверненого до користувача, чи на спеціальній регульованій по висоті робочої поверхні, відділеної від основної стільниці.

4.3 Вимоги до відео дисплейним терміналам і персональним електронно-обчислювальним машинам використовуваних на підприємстві

Візуальні ергономічні параметри відео дисплейних терміналів є параметрами безпеки, і їхній неправильний вибір приводить до погіршення здоров'я. Межі їхніх значень представлені в таблиці 4.9.

Таблиця 4.9

Ергономічні параметри відео дисплейного термінала і межі їхніх змін ДСанПиН 3.3.2 – 007-98

Назва параметра	Межі значень параметрів	
	мін. (не менш)	макс(не білш)
Яркість знака (яркість фона), кд/м2 (виміряна в темряві)	35	120
Зовнішня висветленість экрана, лк	100	250
Кутовий розмір знака, кут. хв.	16	60

Усі відео дисплейні термінали повинні мати гігієнічний сертифікат що включає, у тому числі оцінку візуальних параметрів.

Конструкція відео дисплейних терміналів повинна забезпечувати можливість фронтального спостереження екрана шляхом повороту корпуса в горизонтальній площині навколо вертикальної осі в межах +- 30 градусів і у вертикальній площині навколо горизонтальної осі в межах +- 30 градусів з фіксацією в заданому положенні. Дизайн відео дисплейних терміналів повинен передбачати фарбування корпуса в спокійні м'які тони з дифузійним розсіюванням світла. Корпус відео дисплейних терміналів і ПК, клавіатура й інші блоки і пристрої ПК повинні мати матову поверхню одного кольору з коефіцієнтом відображення 0,4 - 0,6 і не мати блискучих деталей, здатних створювати відблиски. На лицьовій стороні корпуса відео дисплейних терміналів не рекомендується розташовувати органи керування, маркірування, які-небудь

допоміжні написи і позначення. При необхідності розташування органів керування на лицьовій панелі вони повинні закриватися чи кришкою бути утоплені в корпусі.

Конструкція ВДТ повинна передбачати наявність ручок регулювання яскравості і контрасту, що забезпечують можливість регулювання цих параметрів від мінімальних до максимальних значень.

Конструкція відео дисплейного термінала і ПК повинна забезпечувати потужність експозиційної дози рентгенівського випромінювання в будь-якому місці на відстані 0,05 м від екрана і корпуса відео дисплейного термінала, при будь-яких положеннях регулювальних пристроїв не повинен перевищувати еквівалентної дози, рівної 0,1 мБэр/год (100 мкР/ч).

Конструкція клавіатури повинна передбачати:

- виконання у виді окремого пристрою з можливістю вільного переміщення;
- опорне пристосування, що дозволяє змінювати кут нахилу поверхні клавіатури в межах від 5 до 15 градусів;
- висоту середнього ряду клавіш не більш 30 мм;
- розташування часто використовуваних клавіш у центрі, внизу і праворуч, рідко використовуваних - вгорі і ліворуч;
- виділення кольором, розміром, формою і місцем розташування функціональних груп клавіш;
- мінімальний розмір клавіш - 13 мм, оптимальний - 15 мм;
- клавіші з поглибленням у центрі і кроком 19 +- 1 мм;
- відстань між клавішами не менш 3 мм;
- однаковий хід для всіх клавіш з мінімальним опором натисканню 0,25 Н и максимальним - не більш 1,5 Н;
- звуковий зворотний зв'язок від включення клавіш з регулюванням рівня звукового сигналу і можливості її відключення.

4.4 Вимоги до організації режиму праці і відпочинку

Режими праці і відпочинку при роботі з ПК і відео дисплейним терміналом повинні організовуватися в залежності від виду і категорії трудової діяльності. Види трудової діяльності розділяються на 3 групи:

- група А - робота зі зчитування інформації з екрана відео дисплейного чи термінала ПК із попереднім запитом;
- група Б - робота з введення інформації;
- група В - творча робота в режимі діалогу з ПК.

При виконанні протягом робочої зміни робіт, що відносяться до різних видів трудової діяльності, за основну роботу з ПК і відео дисплейним терміналом варто приймати таку, котра займає не менш 50% часу протягом робочої чи зміни робочого дня. Для видів трудової діяльності встановлюється 3 категорії ваги і напруженості роботи з ВДТ і ПК , що визначаються:

- для групи А - по сумарному числу знаків, що зчитуються, за робочу зміну, але не більш 60000 знаків за зміну;
- для групи Б - по сумарному числу що зчитуються чи знаків, що вводяться, за робочу зміну, але не більш 40000 знаків за зміну;
- для групи В - за сумарним часом безпосередньої роботи з відео дисплейним терміналом і ПК за робочу зміну, але не більш 6 годин за зміну.

Тривалість обідньої перерви визначається чинним законодавством про працю і Правилами внутрішнього трудового розпорядку підприємства (організації, установи). Для забезпечення оптимальної працездатності і збереження здоров'я співробітників кадрового агентства протягом робочої зміни повинні встановлюватися регламентовані перерви. Час регламентованих перерв

протягом робочої зміни варто встановлювати в залежності від її тривалості, виду і категорії трудової діяльності.

При 8-годинній робочій зміні і роботі на відео дисплейному терміналі і ПК регламентовані перерви варто встановлювати:

- для I категорії робіт - через 2 години від початку робочої зміни і через 2 години після обідньої перерви тривалістю 15 хвилин кожний;
- для II категорії робіт - через 2 години від початку робочої зміни і через 1.5 - 2 години після обідньої перерви тривалістю 15 хвилин кожний чи тривалістю 10 хвилин через щогодини роботи;
- для III категорії робіт - через 1.5 - 2 години від початку робочої зміни і через 1.5 - 2 години після обідньої перерви тривалістю 20 хвилин кожний чи тривалістю 15 хвилин через щогодини роботи.

Під час регламентованих перерв із метою зниження нервово - емоційної напруги, стомлення зорового аналізатора, усунення впливу гіподинамії і гипокинезіі, запобігання розвитку познотонического стомлення доцільно виконувати комплекси вправ.

4.5 Захист від випромінювання і поля

До числа шкідливих факторів, з якими зіштовхується людина, що працює за монітором, відносяться рентгенівське й електромагнітне випромінювання, а також електростатичне поле.

Припустимі значення параметрів випромінювань, згідно НРБУ – 97, генеруємих відеомоніторами представлені в таблиці 4.10

Таблиця 4.10

Припустимі значення параметрів випромінювань

Параметри	Припустимі значеня

Потужність експозиційної дози рентгенівського випромінювання на відстані 0,05 м навколо відеомонітора	100 мкР/година
Електромагнітне випромінювання на відстані 0,5 м вокруг видеомонитора по електрічній составляющей: в диапазоне 5 Гц-2 кГц	25 В/м
в диапазоні 2-400 кГц	2,5 В/м
по магнитній складовій: в диапазоні 5 Гц-2 кГц	250 нТл
в диапазоні 2-400 кГц	25 нТл
Поверховий електростатичний потенціал	Не более 500 В

Завдяки існуючим досить строгим стандартам дози рентгенівського випромінювання від сучасних відеомоніторів не небезпечні для більшості користувачів. Винятки складають люди з підвищеною чутливістю до нього (зокрема, рентгенівські випромінювання від монітора небезпечні для вагітних жінок, оскільки можуть зробити несприятливий вплив на плід на ранніх стадіях розвитку).Фахівці не прийшли до однозначного висновку щодо впливу електромагнітного випромінювання на організм людини, однак очевидно, що рівні випромінювання, фіксуємі поблизу монітора, небезпеки не представляють.

При роботі монітора виникає й електростатичне поле. Рівні його напруженості невеликі і не роблять істотного впливу на організм людини на відміну від більш високих рівнів електростатичного поля, характерних для промислових умов. Більше значення для користувачів є здатність заряджених мікрочастинок адсорбувати порошини, тим самим перешкоджаючи їхньому осіданню і підвищуючи додатковий ризик алергійних захворювань шкіри, очей, верхніх дихальних шляхів.

4.6 Регулювання параметрів мікроклімату в приміщенні програміста

У приміщеннях, у яких робота на відео дисплейному термінал і ПК є допоміжною, температура, відносна вологість і швидкість руху повітря на робочому місці співробітника повинні відповідати діючим санітарним нормам мікроклімату виробничих приміщень. Згідно НРБУ - 86 (нормами радіаційної безпеки) представлені в таблиці 3.

Підвищуючи вологість повітря в приміщеннях з відео дисплейним терміналом і ПК, варто застосовувати зволожувачі повітря, що заправляються щодня дистильованою чи прокип'яченою питною водою (згідно Державних санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99).

Зміст шкідливих хімічних речовин у повітрі приміщення, у яких робота на відео дисплейних терміналах і ПК є допоміжною, не повинне перевищувати "Гранично припустимих концентрацій шкідливих речовин у повітрі робочої зони"

Таблиця – Рівні іонізації повітря приміщень при роботі на відео дисплейному терміналі

Рівні	Число іонів в 1 см куб. повітря	
	п+	п-
1	2	2
1	2	3
Мінімально необхідні	400	600
Оптимальні	1500-3000	30000-50000
Максимально допустимі	50000	50000

4.7 Захист від шуму

У приміщеннях, у яких робота на відео дисплейних терміналах і ПК є допоміжною, рівні шуму на робочих місцях не повинні перевищувати значень, установлених для даних видів робіт "Санітарними нормами припустимих рівнів шуму на робочих місцях".

Згідно документу - Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 2.3.6.037-99 рівень шуму не повинен перевищувати 75 дБА на робочих місцях у приміщеннях для розміщення гучних агрегатів обчислювальних машин (АЦПУ, принтери і т.п.). Рівень шуму в приміщеннях з відео дисплейним терміналом і ПК можна знизити використанням звуковбирних матеріалів з максимальними коефіцієнтами звукопоглинання в області частот 63 - 8000 Гц для обробки приміщень (дозволених органами й установами Госсанепіднадзора), підтверджених спеціальними акустичними розрахунками.

Додатковим звукопоглинанням служать однотонні завіси з щільної тканини, що гармоніюють з фарбуванням стін і підвішені в складку на відстані 15 - 20 см від огородження. Ширина завіси повинна бути в 2 рази більше ширини вікна.

4.8 Пожежна безпека

Відповідно до ППБУ-2015 у приміщеннях забороняється:

- запалювати вогонь;
- включати електроустаткування, якщо в приміщенні пахне газом;
- курити;
- сушити що-небудь на опалювальних приладах;
- закривати вентиляційні отвори в електроапаратурі

Джерелами запалення є:

- іскра при розряді статичної електрики

- іскри від електроустаткування
- іскри від удару і тертя
- відкрите полум'я

При виникненні пожаронебезпечноі ситуації персонал повинен негайно вжити необхідних заходів для його ліквідації, одночасно сповістити про пожежу адміністрацію.

Приміщення з електроустаткуванням повинні бути оснащені вогнегасниками типу ОУ-3 чи ОУБ-5 у кількості 2 штук, відповідно до вимог ППБУ-2015.

4.9 Електробезпечність

Щоб уникнути поразки електричним струмом необхідно твердо знати і виконувати наступні правила безпечного користування електроенергією (ПУЕ-2017).

Щоб уникнути ушкодження ізоляції проводів і виникнення коротких замикань не дозволяється:

- зафарбовувати і білити шнури і проводи;
- висмикувати штепсельну вилку з розетки за шнур, зусилля повинне бути прикладене до корпуса вилки.

Для виключення поразки електричним струмом забороняється:

- часто включати і виключати комп'ютер без необхідності;
- працювати на засобах обчислювальної техніки і периферійному устаткуванні мокрими руками;
- працювати на засобах обчислювальної техніки і периферійному устаткуванні, що має порушення цілісності корпуса, порушення ізоляції проводів;
- під напругою очищати від пилу і забруднення електроустаткування.

- перевіряти працездатність електроустаткування в непристосованих для експлуатації приміщеннях зі струмопровідними підлогами, сирих, що не дозволяють заземлити доступні металеві частини.

Ремонт електроапаратури виробляється тільки фахівцями-техніками з дотриманням необхідних технічних вимог.

Щоб уникнути поразки електричним струмом, при користуванні електроприладами не можна стосуватися одночасно яких-небудь трубопроводів, батарей опалення, металевих конструкцій , з'єднаних із землею.

При користуванні електроенергією в сирих приміщеннях дотримувати особливої обережності. При виявленні проводу, що обірвався, необхідно негайно повідомити про це адміністрації, ужити заходів по виключенню контакту з ним людей. Дотик до проводу небезпечно для життя.

Порятунок потерпілого при поразці електричним струмом головним чином залежить від швидкості звільнення його від дії струму. В усіх випадках поразки людини електричним струмом негайно викликають лікаря. До прибуття лікаря потрібно, не втрачаючи часу, приступити до надання першої допомоги потерпілому.

4.10 Освітлення робочих місць і розрахунок необхідної кількості світильників для приміщення з комп'ютерами при загальному рівномірному висвітленні

Висвітлення приміщення на підприємстві впливає на працездатність співробітників. Безпечна і високопродуктивна робота може бути лише за умови гарного висвітлення робочого місця. Недостатність і нерівномірність висвітлення робочого місця кадрового агентства робить до перенавантаження зору, перевтоми організму, до нервозності і т.д.

Негативно впливає на зір не тільки недостатня освітленість, але і занадто яскраве висвітлення. Перевищення норм яскравості приводить до зниження функції зору, виникнення роздратувань і різі в очах, головних болів.

У приміщеннях бухгалтера експлуатація відео дисплейних терміналів і ПЭВМ штучне висвітлення повинне здійснюватися системою загального рівномірного висвітлення. У виробничих і адміністративно - суспільних приміщеннях, у випадках переважної роботи з документами, допускається застосування системи комбінованого висвітлення. До загального висвітлення додатково установлюються світильники місцевого висвітлення, призначені для висвітлення зони розташування документів.

Освітленість на поверхні столу в зоні розміщення робочого документа повинна бути 300 - 500 лк. Допускається установка світильників місцевого висвітлення для підсвічування документів. Місцеве висвітлення не повинне створювати відблисків на поверхні екрана і збільшувати освітленість екрана більш 300 лк відповідно до ДБН В.2.5-28-2018 "Природне і штучне освітлення".

Варто обмежувати пряму блесткість від джерел висвітлення, при цьому яскравість світних поверхонь (вікна, світильники й ін.), що знаходяться в полі зору, повинна бути не більш 200 кд/кв. м.

Варто обмежувати відбиту блесткість на робочих поверхнях (екран, стіл, клавіатура й ін.) за рахунок правильного вибору типів світильників і розташування робочих місць стосовно джерел природного і штучного висвітлення, при цьому яскравість відблисків на екрані ВДТ і ПЭВМ не повинна перевищувати 40 кд/кв. м і яскравість стелі при застосуванні системи відбитого висвітлення не повинна перевищувати 200 кд/кв. м.

Показник засліпленості для джерел загального штучного висвітлення в приміщеннях кадрового агентства повинен бути не більш 20, показник дискомфорту в адміністративно - суспільних приміщеннях - не більш 40, у дошкільних і навчальних приміщеннях - не більш 25. Як джерела світла при штучному висвітленні повинні застосовуватися переважно люмінесцентні лампи типу ЛБ. При пристрої відбитого висвітлення у виробничих і адміністративно -

суспільних приміщеннях допускається застосування металлогалогенних ламп потужністю до 250 вт. Допускається застосування ламп накалювання у світильниках місцевого висвітлення. Загальне висвітлення варто виконувати у виді суцільних чи переривчастих ліній світильників, розташованих збоку від робочих місць, паралельно лінії зору користувача при рядному розташуванні відео дисплейних терміналів і ПК. При периметральном розташуванні комп'ютерів лінії світильників повинні розташовуватися локалізовано над робочим столом ближче до його переднього краю, зверненому до оператора. Для висвітлення приміщень з відео дисплейним терміналом і ПК варто застосовувати світильники серії ЛПО36 із зазеркалюванними ґратами, укомплектовані високочастотними пускорегулючими апаратами.

Яскравість світильників загального висвітлення в зоні кутів випромінювання від 50 до 90 градусів з вертикаллю в подовжній і поперечній площинах повинна складати не більш 200 кд/кв. м, захисний кут світильників повинен бути не менш 40 градусів. Світильники місцевого висвітлення повинні мати непросвічуючий відбивач із захисним кутом не менш 40 градусів.

4.11 Розрахунок штучного освітлення

Розраховуємо необхідну кількість світильників для приміщення з комп'ютерами при загальному рівномірному висвітленні.

У кімнаті з розмірами А=5м, У=6м і висотою 3 м передбачені стельові світильники типу УСП 35. Коефіцієнти відображення світлового потоку стелі, стін і підлоги відповідно $\rho_п$=70% , $\rho_с$=50%, $\rho_{підлоги}$=10%. Затемнення робочих місць немає.

Для машинних залів рівень робочої поверхні над підлогою складає 0,8м. Тоді висота підвісок світильників над робочою поверхнею h=H-0.8=2.2м. У світильників УСП 35 відношення ξ=1,4. Звідси відстань між рядами світильників L=?*h=1.4*2.2=3 м. Розташовуємо світильник уздовж довжиною стіни

приміщення. Відстань між стінами і рядами світильників приймаємо рівним l=(0.3-0.5)L. При ширині машинного залу В=6 м маємо число рядів світильників n=У/L=2

Відповідно до норм висвітлення СНіП II-4-79 для даного приміщення і виду робіт установлена норма освітленості $E_н$=300 лк. З обліком заданої $\rho_п$, $\rho_с$, $\rho_{підлоги}$ при індексі приміщення

$$I = \frac{A * B}{[h * (A + B)]} = \frac{5 * 6}{2.2 * (5 + 6)} 1.24$$

З довідкових даних ŋ=0,45.

Номінальний світловий потік лампи ЛБ-40 $Ф_л$=3120 лм, тоді світловий потік, випромінюваний світильником, складе $Ф_{св}$=3120 лм. По формулі визначаємо необхідне число світильників.

$$N = \frac{E_н * К_з * S * z}{n * F * \eta} = \frac{300 * 1,5 * 30 * 1,1}{2 * 3120 * 0,45} = 5,3 \text{ штук}$$

N ≈ 5 шт.

Висновок

У даному розділі були проаналізовані шкідливі і небезпечні фактори при роботі з оргтехнікою в робочому приміщенні програміста, а також розглянуті заходи прийняті по їх усуненню. Також був проведений розрахунок штучного висвітлення. Результат розрахунку визначає кількість світильників для висвітлення робочого місця програміста відповідно до вимог ДБН В.2.5-28-2018 "Природне і штучне освітлення", у результаті якого визначена кількість необхідних світильників з використанням двох люмінесцентних ламп ЛБ-40.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи була виконана поставлена мета роботи та створено вебсайт інтернет-магазин та адмін-панель. Розроблений сайт дає змогу робити онлайн-покупки із задоволення завдяки функціоналу та приємному дизайну.

Було проведено аналіз предметної області та виділені основні концептуальні класи. Було проведено аналіз вимог, визначена функціональність системи і побудована діаграма варіантів використання, спроєктовано інтуїтивно зрозумілий інтерфейс для користувача. Була спроєктована структура класів системи завдяки проведеному аналізу предметної області та на основу діаграми використання.

У процесі було покращені та отриманні навички у роботі з такими інструментами: Node.js, React.js, MongoDB, Express, CSS.

Вивчення Visual Studio Code включало налаштування редактора, використання різноманітних розширень для покращення продуктивності, а також інтеграцію з Git для керування версіями коду.

Розробка відбувалася у Visual Studio Code, що є редактором коду, який підтримує розширення для різних мов програмування, дебаггінг та інтеграцію з системами контролю версій.

Була виконана переважна більшість поставлених вимог. Вебсайт та адмін-панель відповідають основним вимогам, що роблять систему функціональною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Динамічна модель. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Динамічна_модель (дата звернення: 04.06.2024).

2. Єфремов М., Єфремов Ю., Єфремов В. Проєктування програмного забезпечення з використанням UML. ЖДТУ. URL: https://eztuir.ztu.edu.ua/bitstream/handle/123456789/3296/12.pdf?sequence=1 (дата звернення: 01.06.2024).

3. Зелінська О., Потапова Н., Мельянова А. Інформаційна система ведення реєстру клієнтів банку. Вісник Хмельницького національного університету. Технічні науки. 2023. Т. 1. С. 96. URL: http://journals.khnu.km.ua/vestnik/pdf/technew/2023/VKNU-TS-2023-N1(317).pdf#page=94 (дата звернення: 16.05.2024).

4. Функціональне тестування. QALight. URL: https://qalight.ua/baza-znaniy/funktsionalne-testuvannya/ (дата звернення: 14.05.2024).

5. Component diagram. Wikipedia. URL: https://en.wikipedia.org/wiki/Component_diagram (date of access: 14.05.2024).

6. CSS. Вікіпедія. URL: https://uk.wikipedia.org/wiki/CSS#cite_note-1 (дата звертання: 10.06.2024).

7. Flanagan D. JavaScript: The Definitive Guide. 7th ed. O'Reilly Media, 2020. 706 p.

8. HTML. Вікіпедія. URL: https://uk.wikipedia.org/wiki/HTML (дата звертання: 13.05.2024).

9. JavaScript. Вікіпедія. URL: https://uk.wikipedia.org/wiki/JavaScript#cite_note-ECMA-262-5 (дата звертання: 10.06.2024).

10. Logical schema. Wikipedia. URL: https://en.wikipedia.org/wiki/Logical_schema (дата звернення: 02.06.2024).

11. User interface. Wikipedia. URL: https://en.wikipedia.org/wiki/User_interface (дата звернення: 12.06.2024).

12. What is the use case diagram. Ideascale. URL: https://ideascale.com/blog/what-is-the-use-case-diagram/ (дата звертання: 10.06.2024).

13. What is User Interface (UI) Design?. The Interaction Design Foundation. URL: https://www.interaction-design.org/literature/topics/ui-design (дата звернення: 04.06.2024).

ДОДАТОК А

Технічне завдання вебсайту «Inside out» та адмін-панелі

Вступ

Вебсайт, що був розроблений, призначений для зручного онлайн-шопінгу та має робочу назву «Inside out».

1 Підстави для розробки

Підставою для розробки програмного забезпечення є завдання для кваліфікаційної роботи бакалавра за напрямом «Інженерія програмного забезпечення».

Проєкт має бути розроблений відповідно до технічного завдання, а замовник повинен отримати:

- Вихідний код вебсайту та адмін-панелі.
- Опис вебсайту та адмін-панелі.
- Інструкцію користувача.

2 Призначення розробки

2.1 Функціональне призначення

Вебсайт дозволить спростити шопінг, робити покупки не виходячи з дому, також є інструмент для адміністраторів магазину, щоб оновлювати асортимент швидко і зручно.

2.2 Експлуатаційне призначення

Розроблювані вебсайт та адмін-панель слугують для полегшення процесів покупок та торгівлі.

3 Вимоги до вебсайту та адмін-панелі

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Вебсайт:

1. Реєстрація та аутентифікація користувачів:

Вхідна інформація: облікові дані користувача (електронна пошта, номер телефону, соціальні мережі).

Вихідна інформація: підтвердження реєстрації, успішний вхід у систему.

2. Каталог товарів:

Вхідна інформація: фільтри пошуку (за ціною, за назвою).

Вихідна інформація: список товарів, детальна інформація про товар.

3. Кошик:

Вхідна інформація: товари, які додаються до кошика, кількість товарів.

Вихідна інформація: зміст кошика, загальна сума, орієнтовні витрати на доставку.

4. Процес оформлення замовлення:

Вхідна інформація: список товарів в кошику зареєстрованого користувача.

Вихідна інформація: підтвердження замовлення.

5. Додаткові функції безпеки:

Вхідна інформація: дані для двофакторної аутентифікації (телефон, електронна пошта, додаток-аутентифікатор).

Вихідна інформація: підтвердження успішної аутентифікації.

Адмін-панель:

1. Реєстрація та аутентифікація адміністраторів:

Вхідна інформація: облікові дані користувача (електронна пошта, номер телефону).

Вихідна інформація: підтвердження реєстрації, успішний вхід у систему.

2. Управління товарами:

Вхідна інформація: інформація про товари (назва, ціна, зображення, категорії).

Вихідна інформація: оновлений каталог товарів, підтвердження додавання/видалення товарів.

3.1.2 Вимоги до часових характеристик і розміру пам’яті, необхідної для роботи програми

Головна сторінка повинна завантажуватися за 2-3 секунди. Сторінки продуктів – не більше 2 секунд. Сторінки кошика та оформлення замовлення – не більше 3 секунд. Час відгуку сервера повинен бути менше 200 мілісекунд.

Необхідний для роботи об’єм оперативної пам’яті вебсайту та адмін панелі не повинен перевищувати 256 Мбайт.

Необхідно мінімально 50 ГБ для зберігання медіафайлів. Рекомендується для швидкого доступу до даних використовувати SSD. Додаткового простору потребує регулярний бекап даних, бажано, щоб був окремий накопичувач для резервних копій.

3.1.3 Вимоги до організації вхідних та вихідних даних

Для відкриття сайту та адмін-панелі , які є основою вхідних і вихідних даних, потрібно перейти за визначеними посиланнями.

Вхідною інформацією для даної системи можуть бути як події натискання клавіш клавіатури та кнопок миші, так і дані користувачів, які вводяться під час реєстрації та авторизації, а також замовлення товарів. Вихідною інформацією для даної системи можуть бути як відповіді серверу на введені дані користувачів, так і оброблені дані, що передаються та зберігаються у вигляді замовлень.

3.2 Вимоги до надійності

3.2.1 Вимоги до надійного функціонування

Вебсайт та адмін-панель мають функціонувати при безперебійній роботі комп’ютера, телефона або планшета та інтернет з’єднання. Відновлення роботи

вебсайту та адмін-панелі має бути після перез’єднання інтернету, або в випадках проблем з сервером, то – перезапуску серверу.

3.2.2 Вимоги до контролю вхідної і вихідної інформації

Валідація вхідних даних включає події натискання клавіш клавіатури, події натискання кнопок миші, а також дані користувачів, такі як електронна пошта, пароль, особиста інформація та деталі замовлення. Результат валідації може бути як успішним, так і містити повідомлення про помилки, а також підтвердження реєстрації, авторизації або створення замовлення.

Обробка та зберігання даних здійснюється на основі вхідної інформації користувачів, включаючи особисту інформацію та деталі замовлення. Вихідною інформацією є оброблені дані, такі як акаунти користувачів/адміністраторів, замовлення, які зберігаються в базі даних.

Безпека даних забезпечується через обробку аутентифікаційних даних, таких як логіни, паролі. Вихідною інформацією є підтвердження аутентифікації, надання доступу до захищених ресурсів та шифровані дані.

3.2.3 Вимоги до часу відновлення після відмови

Після відмови оптимальний час відновлення роботи вебсайту та адмін-панелі, з боку бекенду, до 15 хвилин та повинен складатися з часу перепідключення серверу та часу перепідключення бази даних.

3.3 Вимоги до умов експлуатації та зберігання

Вебсайт та адмін-панель потребують базових навичок користування інтернет-ресурсом. Інтерфейс інтуїтивний, зрозумілий та зручний.

Для забезпечення надійної роботи обладнання та зберігання даних інтернет-магазину одягу важливо дотримуватися певних умов експлуатації та зберігання. Ось ключові вимоги:

- Температура в серверній кімнаті повинна бути в межах від 18°C до 24°C.

- Забезпечення постійного моніторингу температури та сигналізації при відхиленні від норми.

- Безперебійне живлення з використанням джерел безперебійного живлення (UPS).

- Резервні генератори для забезпечення живлення у випадку тривалих відключень електроенергії.

- Зберігання резервних копій даних у спеціально відведених місцях з контрольованим доступом.

- Зберігання даних в умовах, захищених від пилу, вологи та механічних пошкоджень.

- Зберігання резервних копій на віддалених локаціях або в хмарних сховищах для забезпечення додаткового рівня захисту.

- Регулярне оновлення та перевірка резервних копій.

- Використання шифрування для захисту чутливих даних(паролей).

- Регулярне тестування процесів відновлення даних для забезпечення їх працездатності у випадку необхідності.

- Документування процесів резервного копіювання та відновлення.

3.4 Вимоги до складу й параметрів технічних засобів

Апаратне забезпечення:

- Будь-який сучасний комп'ютер, ноутбук, планшет або смартфон з доступом до інтернету та підтримкою сучасних веб-технологій.
- Мінімальні вимоги включають процесор з частотою не менше 1.6 GHz, 2 GB оперативної пам'яті та 512 MB вільного місця на диску.

Веб-браузери:

- Підтримка основних веб-браузерів: Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. Це дозволить користувачам коректно відображати веб-сторінки на різних пристроях.

3.5 Вимоги до маркування та пакування

Для зберігання програмного продукту інтернет-магазину одягу можна використовувати наступні рішення:

1. Хмарні сервіси: Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure – це надійні та масштабовані платформи для зберігання даних і розгортання вебсайтів.

2. Віртуальні приватні сервери: VPS дозволяють отримати більше контролю над середовищем розгортання.

3. Спеціалізовані сервери: використання виділених серверів дозволяє мати повний контроль над апаратним та програмним забезпеченням.

4. Локальні сервери: для підприємств, котрі хочуть повністю контролювати свої дані – можна використовувати локальні сервери.

Вибір залежить від розміру бізнесу, технічних можливостей команди, бюджету та очікуваного трафіку.

3.6 Вимоги до транспортування та зберігання

Має бути забезпечене резервне копіювання даних перед транспортуванням, щоб уникнути втрати інформації у випадку непередбачених ситуацій.

Потрібно використовувати перевірені та надійні сервіси для передачі великих обсягів даних: AWS Snowball або Google Transfer Appliance.

Використання надійних серверів або хмарних платформ, щоб зберігати програмний продукт (наприклад, AWS, Google Cloud, Microsoft Azure).

Забезпечити резервне копіювання даних на постійній основі.

Зберігати сервери в умовах, що відповідають рекомендаціям виробника обладнання, що включає контроль вологості та температури.

Зберігати декілька версій резервних копій, що дасть змогу відновити дані за різні точки часу.

Регулярно потрібно тестувати відновлення даних з резервних копій, щоб переконатися чи ефективні вони.

4 Вимоги до програмної документації

Документація до програмного виробу має містити такі документи:

- технічне завдання;
- опис програми;
- текст програми;
- програма і методика випробувань;
- інструкція користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки вебсайту та адмін-панелі, терміни їх виконання мають відповідати затвердженому графіку проєктування кваліфікаційної роботи та зводитись до таких, що наведені в таблиці А.1.

Таблиця А.1– Стадії, етапи розробки програми, та терміни їх виконання

Стадії розробки	Етапи робіт	Зміст робіт	Терміни виконання
1	2	3	4
Технічне завдання	Розробка технічного завдання.	Обґрунтування необхідності розробки програми	05.03.24 – 26.03.24
	Затвердження технічного завдання.	Розробка пояснювальної записки. Узгодження і затвердження технічного завдання.	26.03.24 – 06.04.24
Ескізний проєкт	Розробка ескізного проєкту	Попередня розробка структури вхідних та вихідних даних, уточнення методів рішення завдань, загальний опис алгоритму.	06.04.24 – 09.04.24
	Затвердження ескізного проєкту	Розробка пояснювальної записки. Узгодження і затвердження ескізного проєкту.	09.04.24 – 13.04.24

Продовження таблиці А.1.

1	2	3	4
Технічний проєкт	Розробка технічного проєкту	Уточнення структури вхідних та вихідних даних, розробка алгоритму рішення задачі, розробка структури програми.	13.04.24 – 17.04.24
	Затвердження технічного проєкту	Розробка пояснювальної записки. Узгодження і затвердження технічного проєкту.	17.04.21 – 20.04.21
Робочий проєкт	Розробка програми	Програмування та налагодження програми	20.04.24 – 27.04.24
	Випробування програми	Розробити, узгодити та затвердити програму і методику випробувань, коригування програми за результатами випробувань.	27.04.24 – 05.05.24
	Розробка програмної документації	Розробка програмної документації	05.05.24 – 15.05.24

6. Порядок прийому та контролю

Для контролю й прийому необхідно надати інструкцію користувача та опис програми, а також програму та методику випробування.

Порядок контролю і прийому даної розробки здійснюється представником замовника у присутності представників розробника згідно з програмою та методикою випробувань.

За результатами прийому складається акт, який підписується представником замовника та представником розробника та утверджується керівниками організації-замовника і організації-розробника.

У випадку виявлення помилок під час прийому програмного виробу складається акт про виявлені помилки, який підписується представниками замовника та розробника й стверджується керівниками організації -замовника й організації-розробника. Розробник повинен в продовж не більш як 1-го місяця видалити вказані зауваження, й повідомити замовника про повторне проведення перевірки, не пізніше як за 2 тижні.

ДОДАТОК Б

Програма та методика випробувань вебсайту «Inside out» та адмін-панелі

1 Об’єкт випробування

Об’єктом випробування є вебсайт інтернет-магазин одягу «Inside out» та адмін-панель для управління сайтом.

Програмний продукт допомагає забезпечити зручний онлайн-шопінг та полегшити управління магазином.

2 Мета випробування

Метою випробування є перевірка вебсайту та його інструменту управління на наявність помилок і відповідність реалізованих функцій функціям, зазначеним у технічному завданні, яке наведене у додатку А. Якщо прийнятий рівень відповідності вебсайту вимогам досягнутий, а також відсутні помилки та грубі недоліки, то програмне забезпечення приймається в експлуатацію.

3 Вимоги до програми

При проведенні випробувань функціональні характеристики вебсайту та адмін-панелі підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

Для проведення випробування будемо розглядати такі функції:

- Функція додавання нових товарів на сайт;
- Функція оформлення замовлення;
- Функція реєстрації для користувачів сайту.

4 Вимоги до програмної документації

Документація, що пропонується до випробування програмного забезпечення для автоматизації обліку трудових ресурсів підприємства, повинна включати наступні документи:

– технічне завдання;
– опис програмного забезпечення;
– текст програмного забезпечення;
– програму та методику випробувань;
– інструкцію користувача.

5 Засоби та порядок випробувань

Засоби випробувань

Для випробувань даного інтернет-магазину необхідно, щоб на комп’ютері або на телефоні, де буде запущений сайт чи адмін-панель, стояла операційна система(наприклад: Windows, MacOS, IOS, Android, IPadOS), що підтримує браузер Chrome, Firefox, Safari або Opera.

Для роботи інтернет-магазину та адмін-панелі потрібен пристрій, що має такі мінімальні характеристики:

- процесор: 800 МГц(для ПК);
- оперативна пам’ять: 256 Мб(для ПК);
- HDD: 1.7 Гб на жорсткому диску(для ПК);
- клавіатура(для ПК);
- миша(для ПК);
- Android 5.0 чи новіший(для телефону);
- IOS 16.0 чи новіша(для телефону);
- IPadOS 16.0 чи новіша(для планшету).

Порядок проведення випробувань

Випробування вебсайту і адмін-панелі будуть проводитися протягом 3 (трьох) днів. Замовник, у присутності розробника, проводить тестування

самостійно усіх функцій. При наявності відповідної можливості у процесі проведення випробувань можуть додатково брати участь програмісти. Попередньо до початку випробувань визначається кінцевий склад групи, що буде проводити випробування. Кожен член групи одержує необхідну документацію до програми. При випробуванні проводиться:

- контроль складу функцій, які виконуються;
- контроль коректності виконання окремих функцій;
- випробування на правильність роботи окремих модулів;
- комплексне випробування.

Якщо наявне виявлення помилок у роботі програмного забезпечення, то складається список цих помилок і проговорюється термін виправлення їх розробником. Після цього замовник проводить повторне випробування у повному обсязі.

6 Методи випробування

Випробування проводитимуться за методом «чорного ящику».

Програма випробування:

1) Випробування функції додавання нових товарів.

Після відкриття адмін-панелі відображається три вкладки, після натискання на вкладку «Add Product» відкривається форма з наступними елементами:

- назва товару;
- категорія;
- стара ціна;
- нова ціна;
- фото;
- кнопка «Add».

Після вводу всієї інформації в поля та після натискання кнопки «Add» товар зберігається в базу даних та відображається на сайті.

2) Випробування функції оформлення замовлення.

Після того як користувач авторизувався/зареєструвався на сайті та додав товар в кошик, переходить в кошик та натискає кнопку «Оформити замовлення». Відображається повідомлення про те, що замовлення успішно оформлене. Після чого з користувачем зв’язується менеджер для уточнення деталей. Якщо користувач не авторизований, то система повідомляє про помилку.

3)Випробування функції реєстрації для користувачів сайту.

Після відкриття посилання на сайт користувач бачить на навігаційній панелі кнопку «Вхід». Якщо користувач не має акаунту, то на тикай випадок є підказка «Create an account? Click here», де «Click here» є кнопкою-посиланням на форму реєстрації. Форма реєстрації в себе включає такі елементи:

- ім’я нового користувача;
- номер телефону;
- електрона пошта;
- пароль;
- функція «показати пароль»;
- кнопка «Continue».

Новий користувач вводить всі необхідні дані у поля форми, натискає кнопку «Continue» та система його перекидає на головну сторінку сайту, після чого клієнт може надалі користуватися функціями сайту. У разі збігу електронною пошти у базі даних система повідомляє про помилку.

ДОДАТОК В

Інструкція користувача вебсайтом «Inside out» та адмін-панеллю

Для запуску вебсайту необхідно відкрити посилання на якому він розташований у веб-браузері, аналогічно і адмін-панель. Для цього треба ввести URL у рядок адреси браузера і натиснути Enter. Це дозволить побачити вебсайти та взаємодіяти з ними.

Після відкриття сайту з’явиться головна сторінка на якій є популярні жіночі товари, банер з інформацією про акції, нові товари та форма для підписки на розсилку та навігаційна панель.

На рисунках В.1-В.4 продемонстрована головна сторінка вебсайту, де клієнт може ознайомитися з акціями та асортиментом, підписатися на розсилку, а також, перейти на інші сторінки за допомогою навігаційної панелі в самому верху сайту.

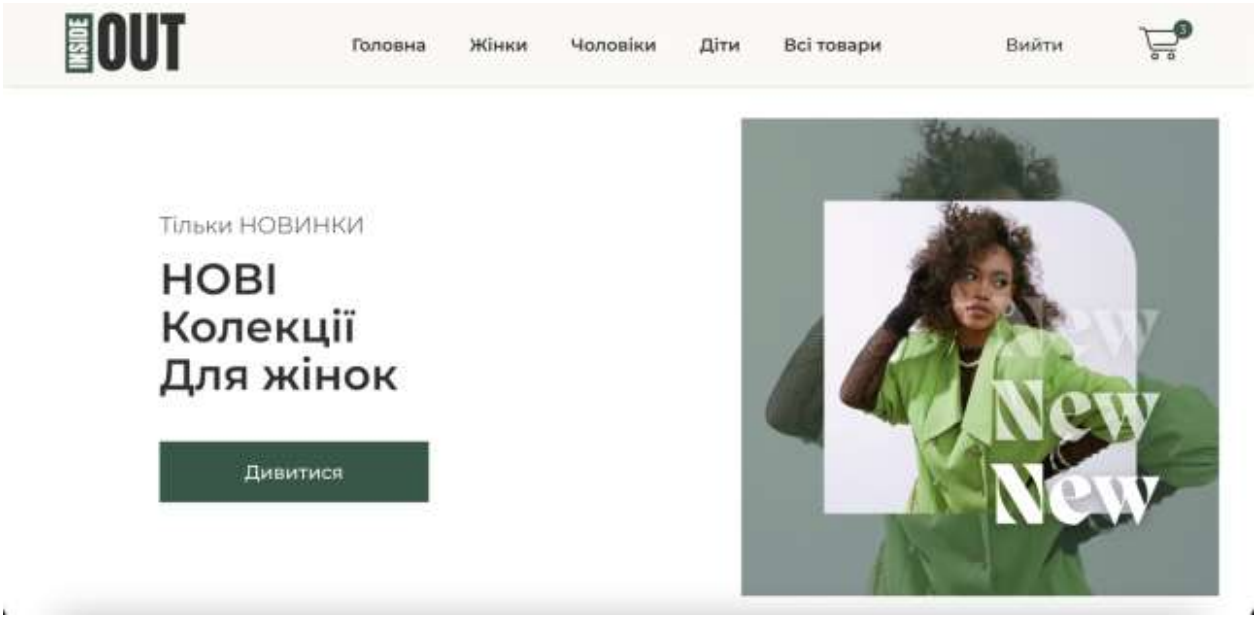

Рисунок В.1 – Інтерфейс головної сторінки

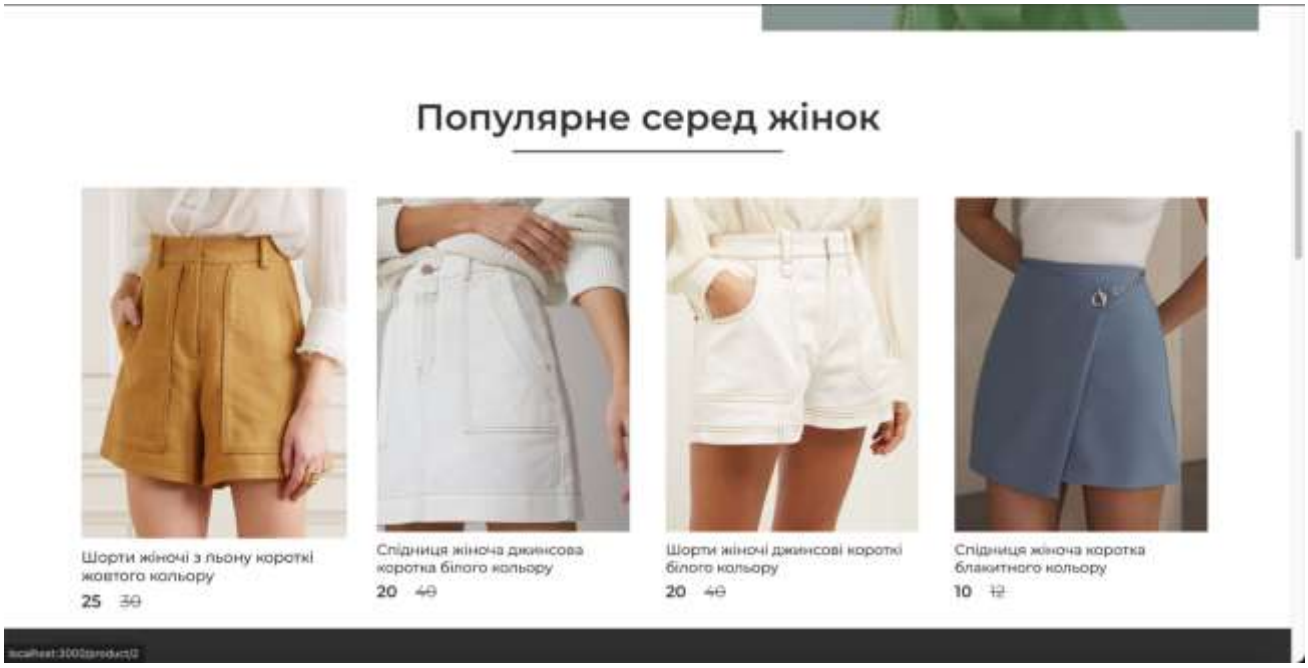

Рисунок В.2 – Головна сторінка. «Популярне серед жінок

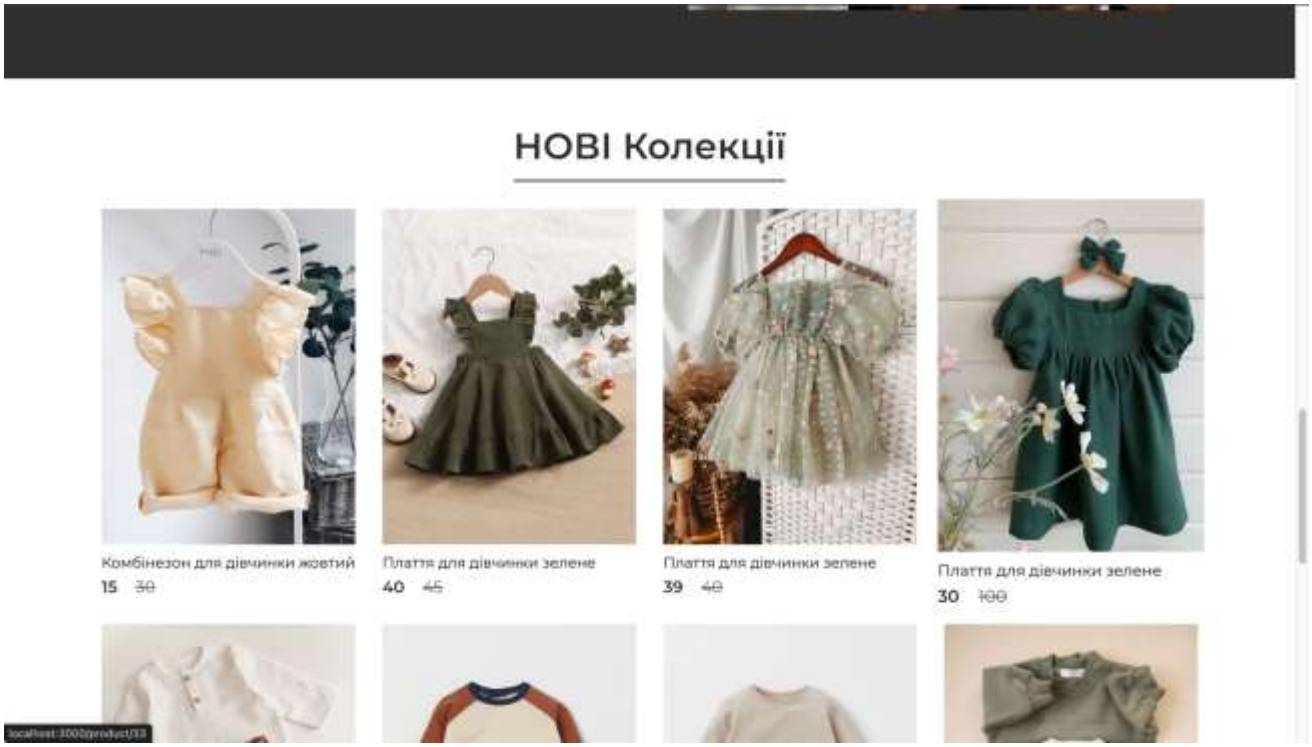

Рисунок В.3 – Головна сторінка. «Нові колекції»

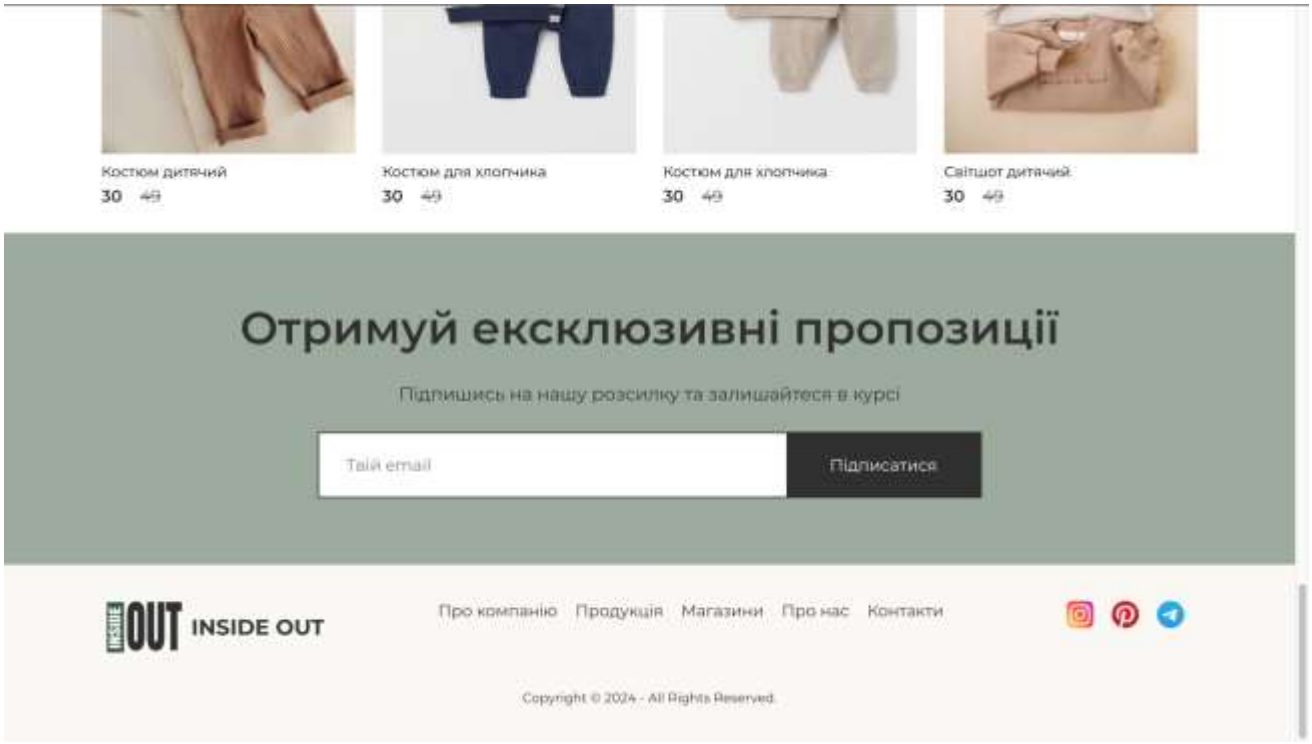

Рисунок В.4 – Форма «Підписка на розсилку»

Далі за допомогою навігаційного меню користувач переміщається по сайту, що показано на рисунках В.5-В.8, на сторінки «Жінки», «Чоловіки», «Діти», «Всі товари», де відповідно розміщені товари жіночі, чоловічі, дитячі, та каталог всіх товарів, також є функція сортування. «Головна» переводить користувача на головну сторінку.

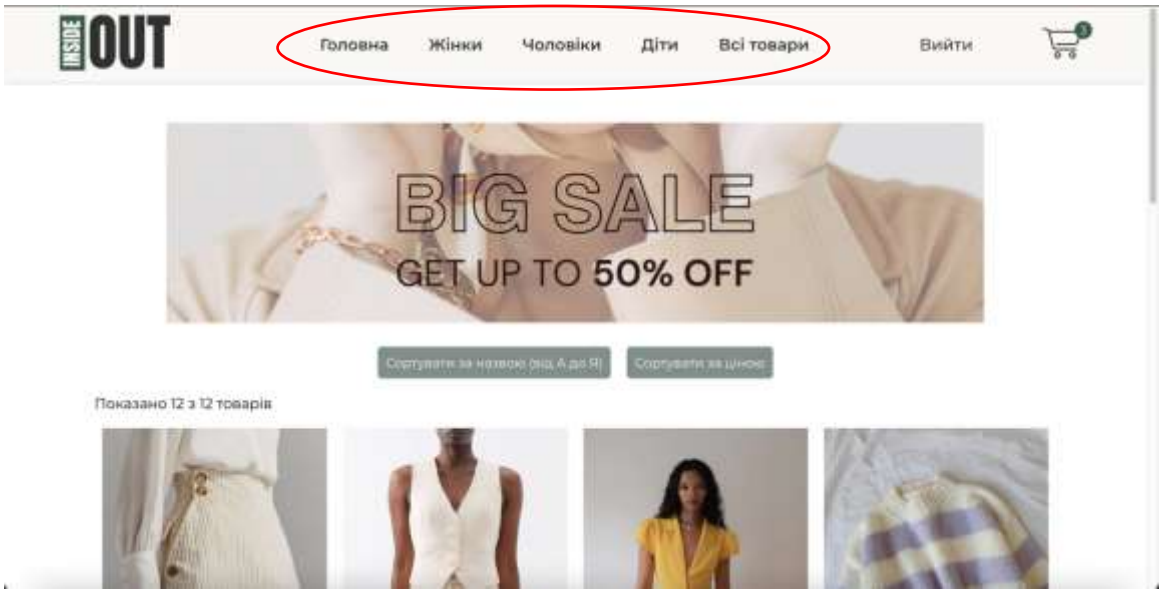

Рисунок В.5 – Сторінка «Жінки»

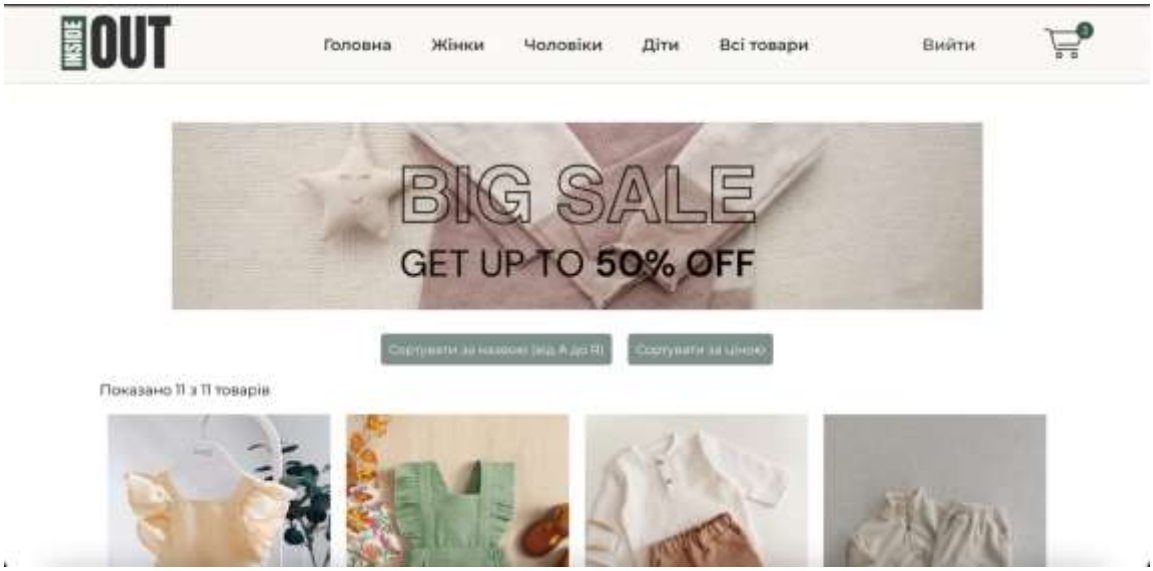

Рисунок В.6 – Сторінка «Діти»

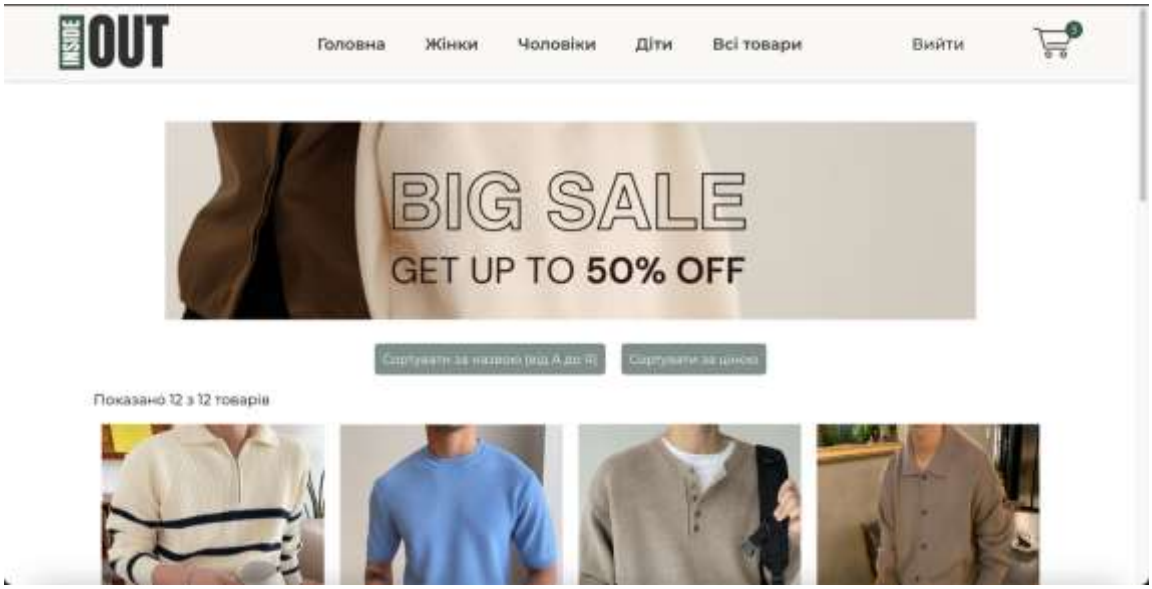

Рисунок В.7 – Сторінка «Чоловіки»

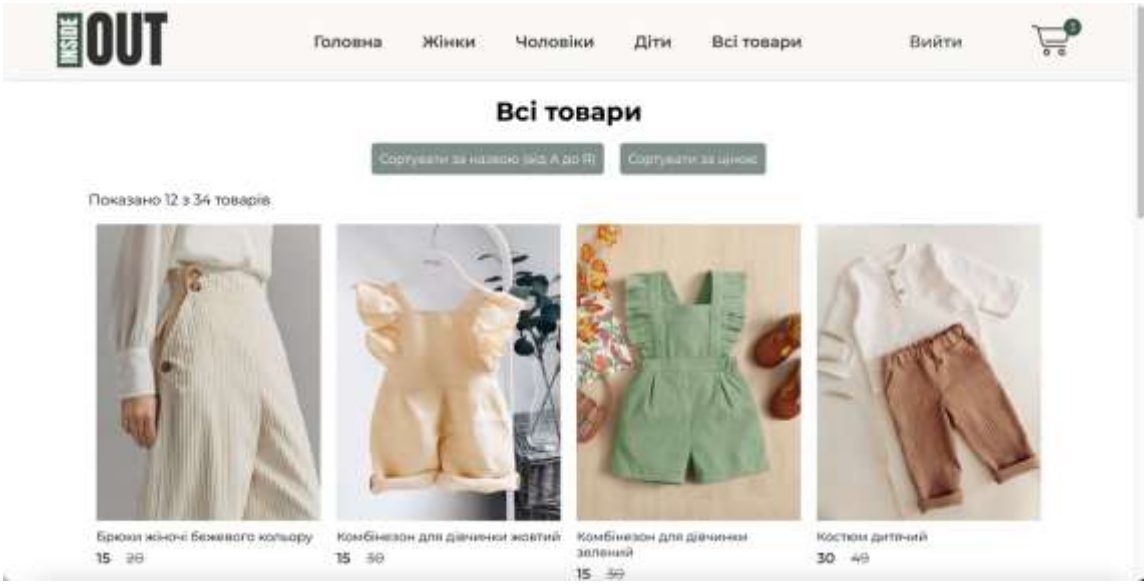

Рисунок В.8 – Сторінка «Всі товари»

Для замовлення товарів на сайті клієнт має бути зареєстрованим або авторизованим. Для цього треба заповнити всі поля у відповідних формах після натиснення кнопки «Вхід» на навігаційній панелі.

Форма авторизації – на рисунку В.9.

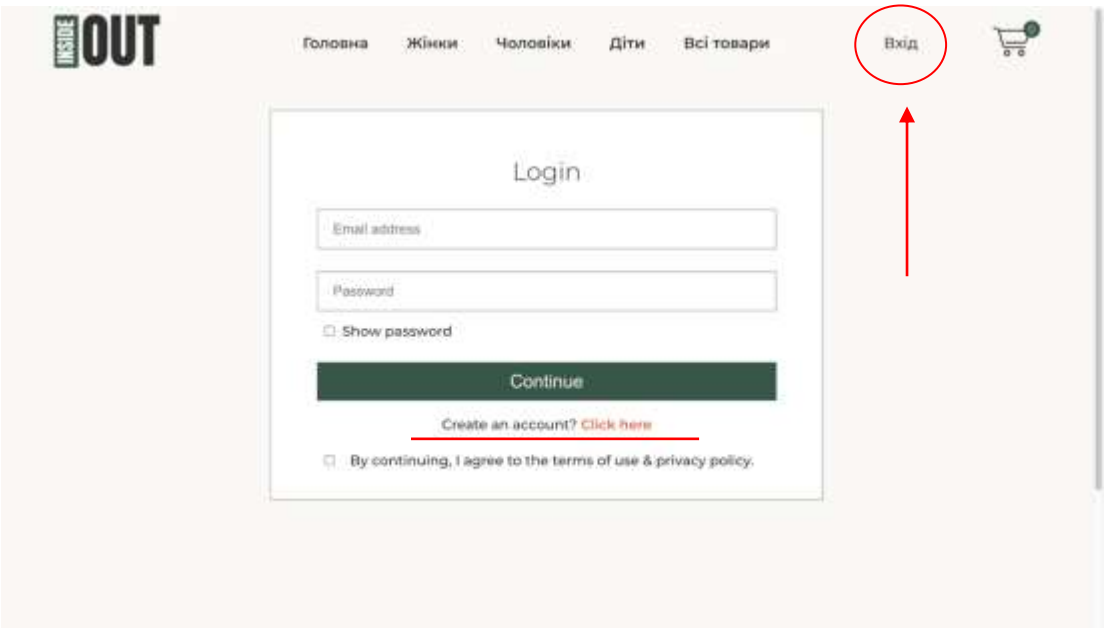

Рисунок В.9 – Форма авторизації

Форма авторизації відкривається по дефолту, але якщо користувач не має акаунту, то переходить по підказці внизу на форму реєстрації.

Форма реєстрації показана на рисунку В.10.

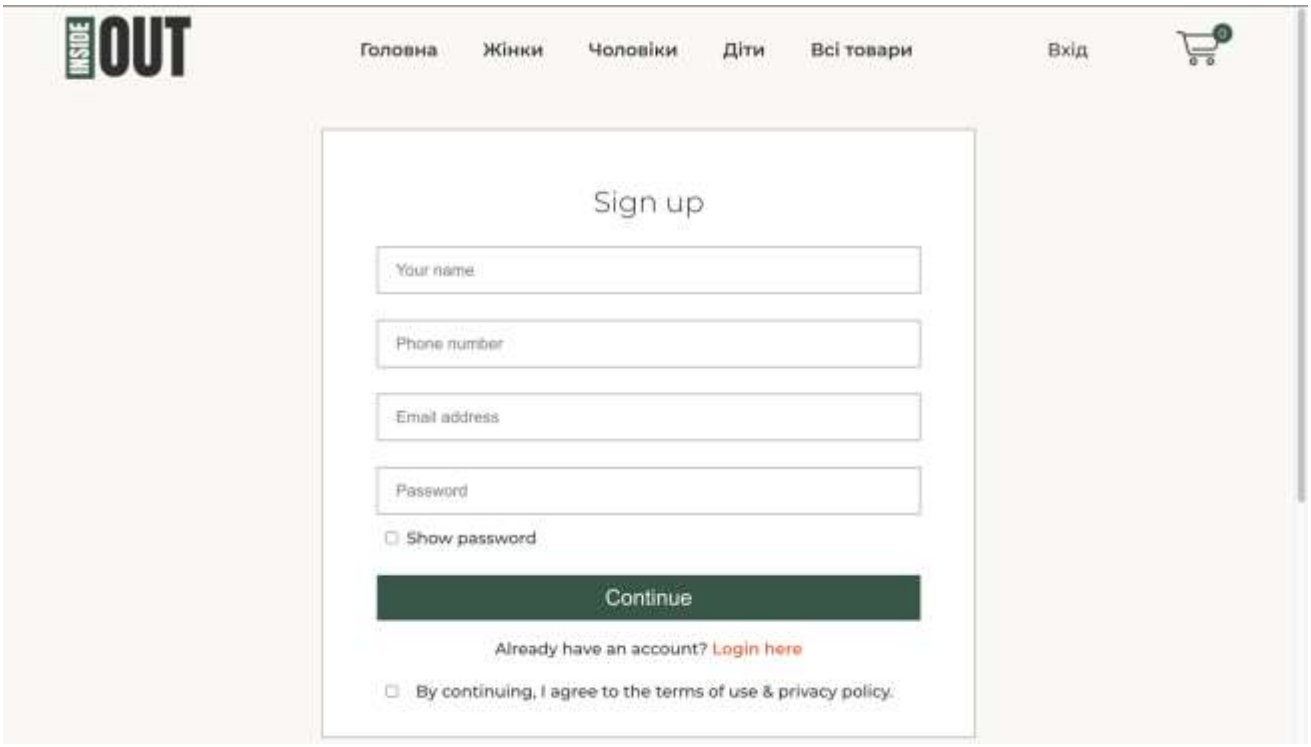

Рисунок В.10 – Форма реєстрації

Після авторизації користувача перекидає на головну сторінку для продовження покупок та перегляду сайту, а також з’являється кнопка «Вихід» для виходу з акаунту.

Щоб додати товар до кошика користувач натискає на будь-який товар та відкривається сторінка товару, де є інформація про елемент одягу та кнопка «Купити». Після натискання кнопки «Купити» товар вже в кошику.

На рисунку В.11 наведена сторінка одного з товарів.

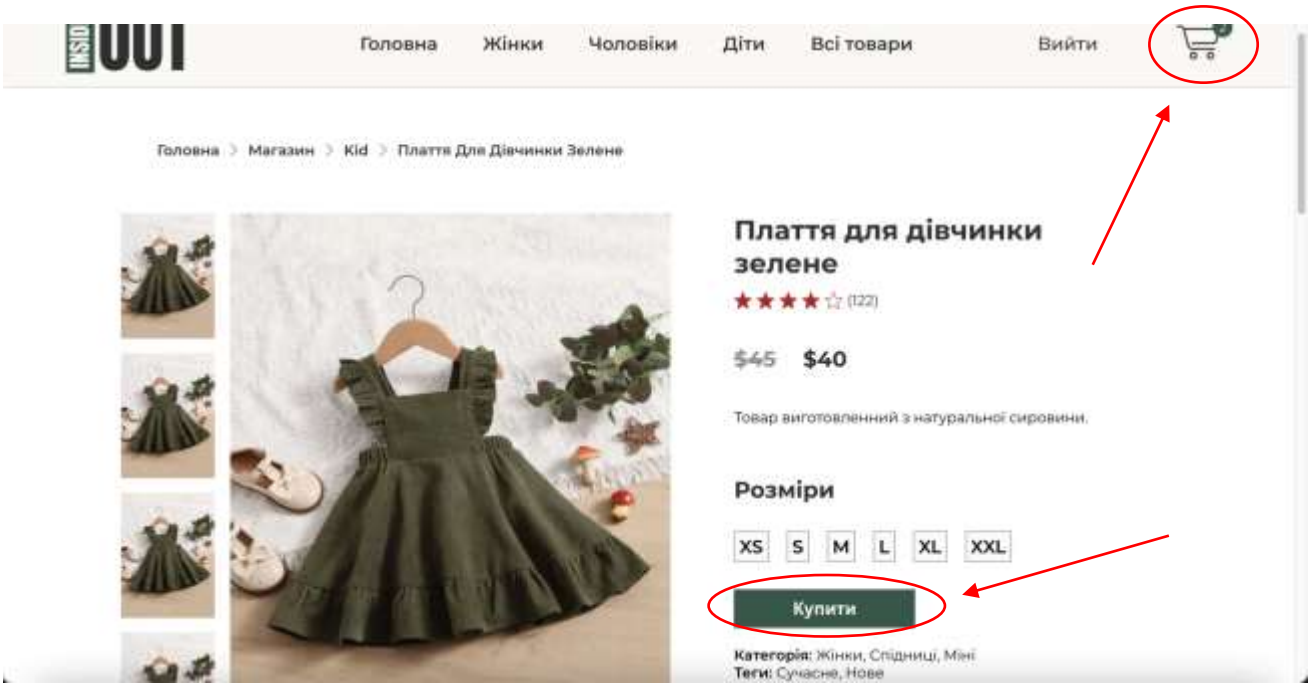

Рисунок В.11 – Сторінка товару

Після натиснення іконки кошика на навігаційній панелі користувач переходить на сторінку «Кошик», де після натиснення кнопки «Оформити замовлення» замовлення зберігається в базі даних та в адмін-панелі. Клієнту залишається чекати поки з ним зв’яжеться менеджер для уточнення деталей доставки.

На рисунку В.12 продемонстрована сторінка «Кошик».

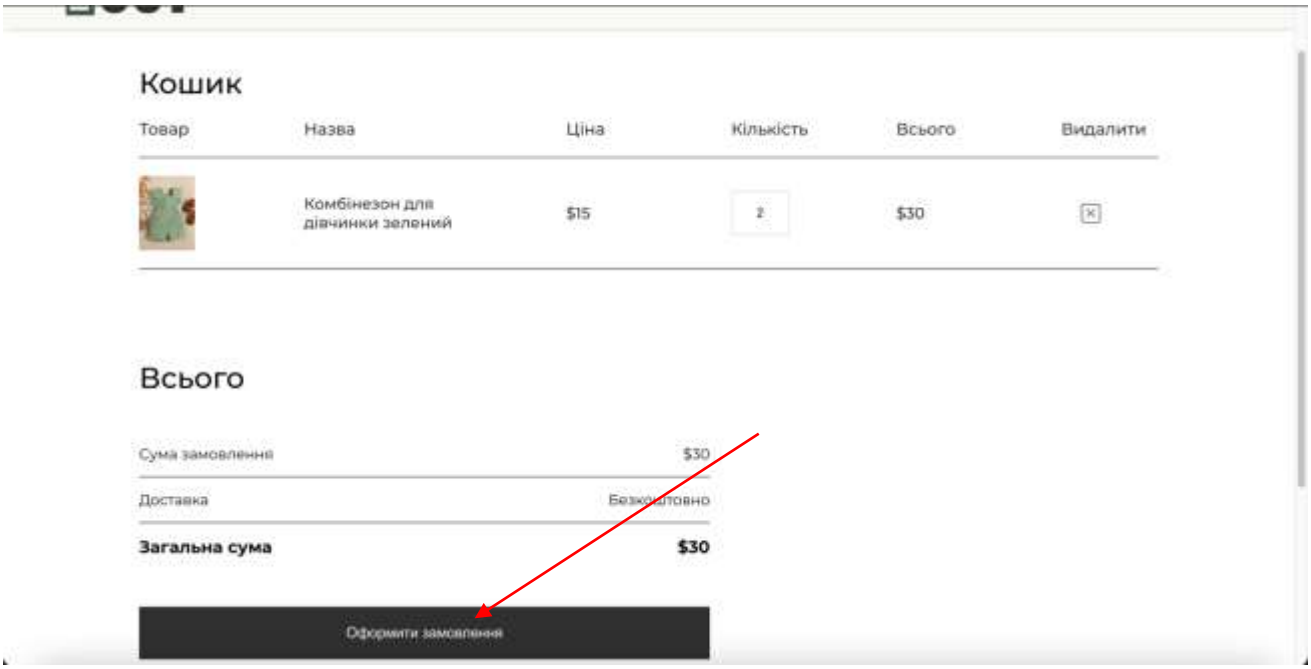

Рисунок В.12 – Сторінка «Кошик»

Для початку взаємодії з адмін-панеллю треба перейти за посиланням. Адміністратор сайту бачить навігаційну панель зліва та кнопку «Вхід». Щоб взаємодіяти з вкладками панелі треба зайти в акаунт. Після натиснення кнопки «Вхід» відобразиться форма авторизація.

Рисунок В.13 форма авторизації адміністратора.

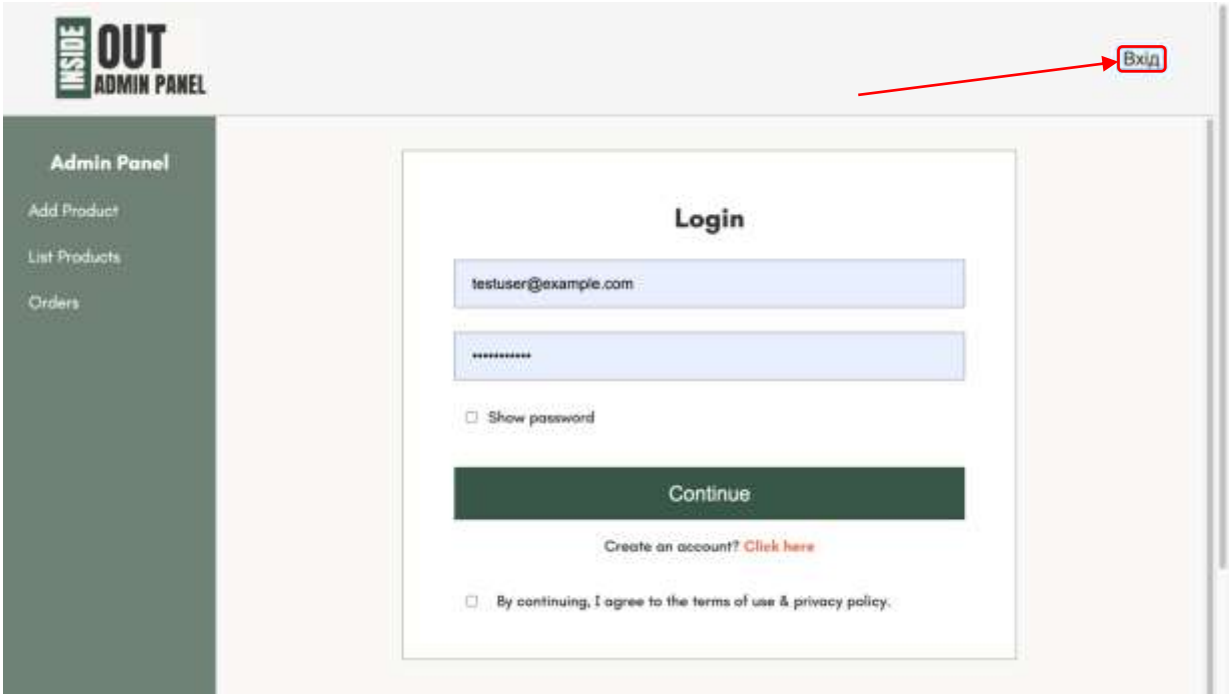

Рисунок В.13 – Форма авторизації адміністратора

Якщо адміністратор не зареєстрований, то є підказка «Click here», що переводить на форму реєстрації.

На рисунку В.14 форма реєстрації адміністратора.

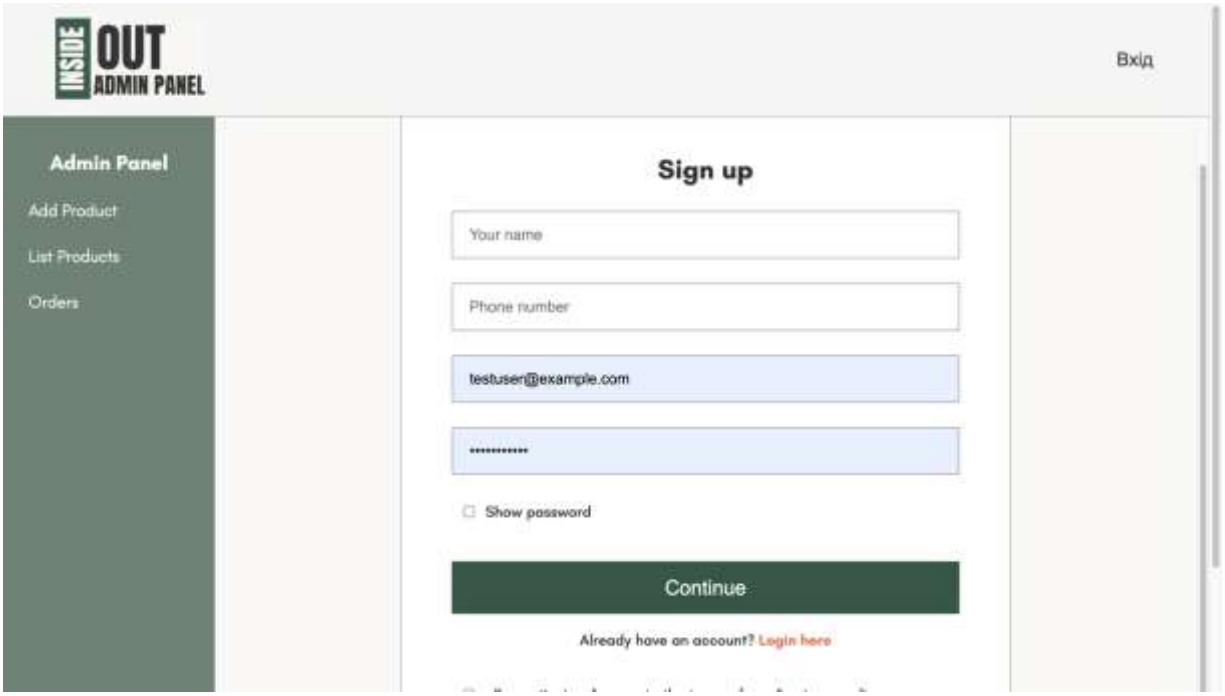

Рисунок В.14 – Форма реєстрації адміністратора

Після аутентифікації адміністратор може перейти на вкладки з додаванням нового продукту «Add Product», список всіх товарів «List Products» та список замовлень «Orders».

На вкладці «List Products» адміністратор може перевірити, які товари є на сайті їх ціну, категорію, фото, назву, а також видалити за потреби іконкою хрестика.

На рисунку В.15 показана вкладка списку всіх товарів.

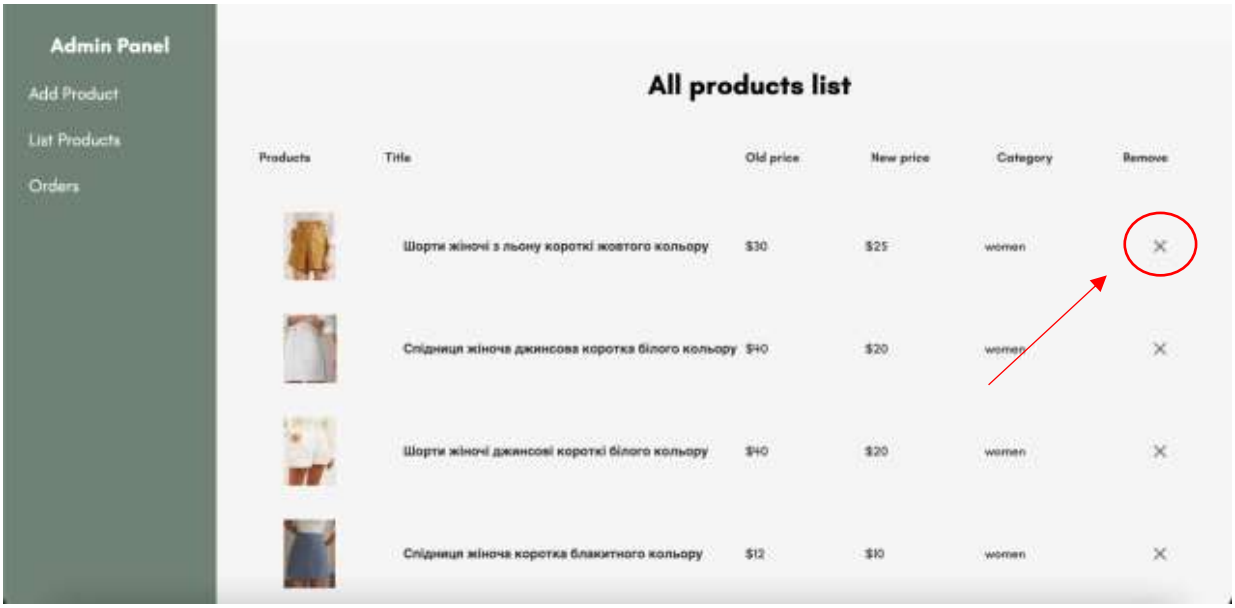

Рисунок В.15 – Вкладка «List Products»

На вкладці «Orders» адміністратор може перевірити, які замовлення були зроблені клієнтами та за допомогою зазначеного номеру телефону зв’язатися з ними для уточнення даних про замовлення та доставку.

На рисунку В.16 показана вкладка замовлень.

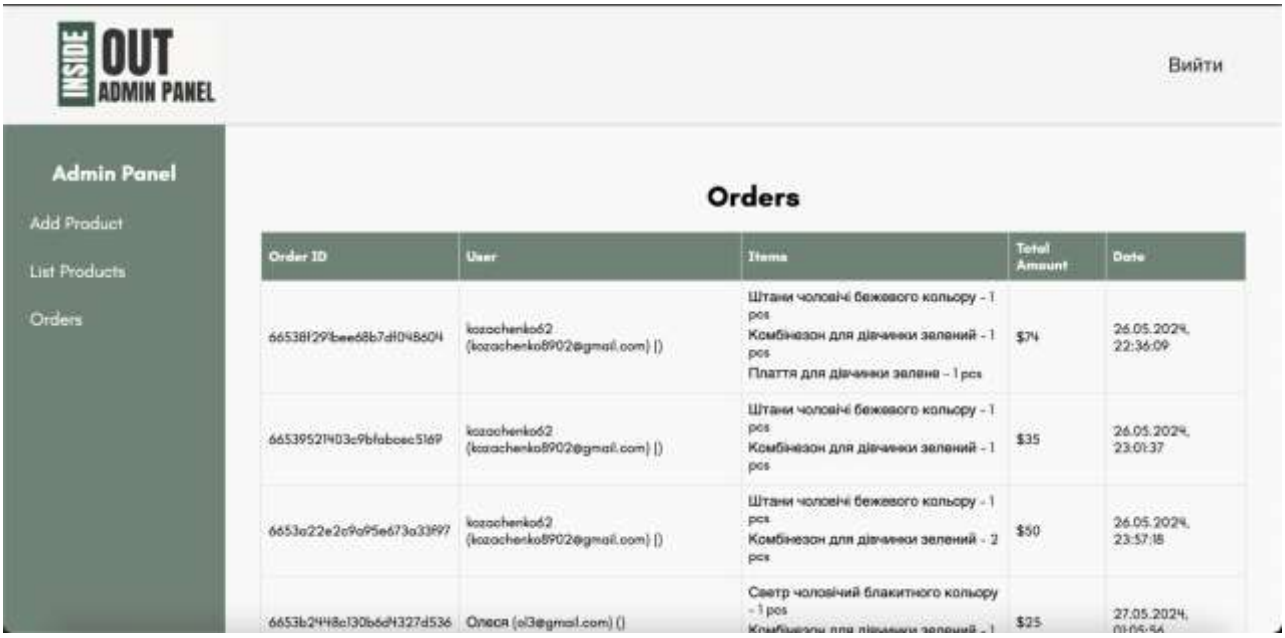

Рисунок В.16 – Вкладка «Orders»

На вкладці «Add Product» адміністратор може додати нові товари на сайт після зазначення їх назви, ціни, категорії та додавання фото. Нових товар буде додано після натискання кнопки «Add».

На рисунку В.17 показана вкладка додавання нового товару.

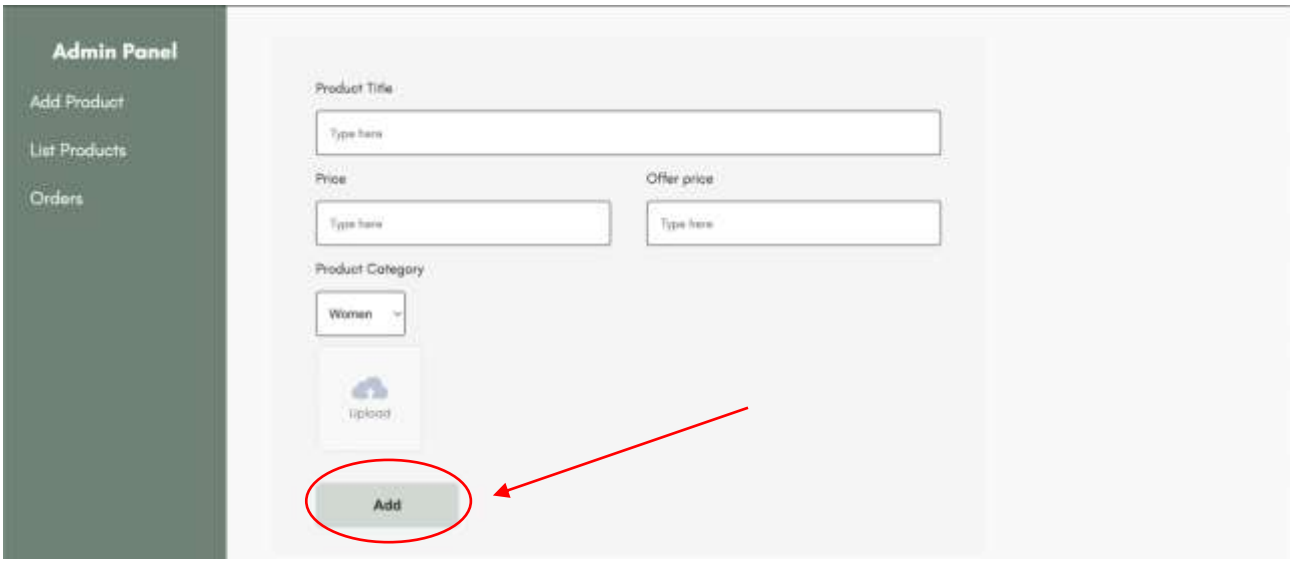

Рисунок В.17 – Вкладка «Add Product»

Після виконаних всіх дій, адміністратор може вийти з акаунту скориставшись кнопкою «Вихід».

ДОДАТОК Г

Опис вебсайту «Inside out» та адмін-панелі

1 Загальні відомості

Назва: вебсайт «Inside out».

Область застосування: електронна комерція .

Були використані для розробки: бібліотека JavaScript React, Node.js, що є основою бекенд-розробки цього проєкту, фреймворк Express.js. Для роботи з базами даних було використано MongoDB. Для стилізації сторінок було використано CSS.

Для роботи вебсайту та адмін-панелі потрібен пристрій, що підтримує будь-який браузер.

2 Функціональне призначення розробки

Функціональним призначенням проєкту є електронна комерція. Клієнти можуть купувати одяг зі своїх комп’ютерів та з інших пристроїв, таких як телефон або планшет, а адміністратори зручно керувати магазином.

Вимоги до складу виконуваних функцій:

1. Реєстрація та аутентифікація користувачів.
2. Перегляд товарів клієнтами.
3. Кошик.
4. Процес оформлення замовлення.
5. Реєстрація та аутентифікація адміністраторів.

6 Управління товарами.

7. Перегляд замовленнями.

4 Технічні засоби

Для коректної роботи вебсайту і адмін-панелі потрібно, щоб на комп’ютері або будь-якому іншому пристрої, стояла операційна система(наприклад: Windows, MacOS, IOS, Android, IPadOS), що підтримує браузер Chrome, Firefox, Safari або Opera.

Для роботи інтернет-магазину та адмін-панелі потрібен пристрій, що має такі мінімальні характеристики:

- процесор: 800 МГц(для ПК);
- оперативна пам’ять: 256 Мб(для ПК);
- HDD: 1.7 Гб на жорсткому диску(для ПК);
- клавіатура(для ПК);
- миша(для ПК);
- Android 5.0 чи новіший(для телефону);
- IOS 16.0 чи новіша(для телефону);
- IPadOS 16.0 чи новіша(для планшету).

5 Виклик і завантаження

Для запуску вебсайту необхідно відкрити посилання на якому він розташований у веб-браузері, аналогічно і адмін-панель. Для цього треба ввести URL у рядок адреси браузера і натиснути Enter. Це дозволить побачити вебсайти та взаємодіяти з ними.

6 Вхідні дані і вихідні дані

Вхідні дані.

- Облікові дані користувача (електронна пошта, номер телефону, соціальні мережі).
- Фільтри пошуку (за ціною, за назвою).
- Товари, які додаються до кошика, кількість товарів
- Список товарів в кошику зареєстрованого користувача.

- Дані для двофакторної аутентифікації (телефон, електронна пошта, додаток-аутентифікатор).
- Облікові дані адміністратора (електронна пошта, номер телефону).
- Інформація про товари (назва, ціна, зображення, категорії).

Вихідні дані.

- Підтвердження реєстрації, успішний вхід у систему.
- Список товарів, детальна інформація про товар.
- Вміст кошика, загальна сума, орієнтовні витрати на доставку.
- Підтвердження замовлення.
- Підтвердження успішної аутентифікації.
- Підтвердження реєстрації, успішний вхід у систему.
- Оновлений каталог товарів, підтвердження додавання/видалення товарів.

ДОДАТОК Д

Текст програмного забезпечення

1. Frontend

1) Navbar.jsx

```
import React, { useContext, useRef, useState } from 'react';
import './Navbar.css';
import logo from '../assets/logo.png';
import cart_icon from '../assets/cart_icon.png';
import { Link } from 'react-router-dom';
import { ShopContext } from '../../context/ShopContext';
import nav_dropdown from '../assets/nav_dropdown.png'
const Navbar = () => {
  const [menu, setMenu] = useState('shop');
  const {getTotalCartItems} = useContext(ShopContext);
  const menoRef = useRef();
  const dropdown_toggle = (e) =>{
    menoRef.current.classList.toggle('nav-menu-visible');
    e.target.classList.toggle('open');
  }
  return (
    <div className='navbar'>
      <div className='nav-logo'>
       <Link to='/'><img src={logo} alt='' /></Link>
      </div>
        <img          className='nav-drop-down'          onClick={dropdown_toggle}
src={nav_dropdown} alt='' />
```

```
        <ul ref={menoRef}  className='nav-menu'>
          <li onClick={() => { setMenu('shop') }}><Link className="menu-link" style={{textDecoration: 'none'}} to='/'>Головна</Link></li>
          <li onClick={() => { setMenu('womens') }}><Link className="menu-link" style={{textDecoration: 'none'}} to='/womens'>Жінки</Link></li>
          <li onClick={() => { setMenu('mens') }}><Link className="menu-link" style={{textDecoration: 'none'}} to='/mens'>Чоловіки</Link></li>
          <li onClick={() => { setMenu('kids') }}><Link className="menu-link" style={{textDecoration: 'none'}} to='/kids'>Діти</Link></li>
          <li onClick={() => { setMenu('kids, mens, womens') }}><Link className="menu-link" style={{textDecoration: 'none'}} to='/all-products'>Всі товари</Link></li>
        </ul>
        {menu === 'shop' ? null : <></>}
        <div className='nav-login-cart'>
          {localStorage.getItem('auth-token')
          ?<button onClick={()=>{localStorage.removeItem('auth-token');window.location.replace('/')}}>Вийти</button>:
          <Link to='/login'><button>Вхід</button></Link>}
         <Link to='/cart'><img src={cart_icon} alt=" /></Link>
          <div className='nav-cart-count'>
            {getTotalCartItems()}
          </div>
        </div>
      </div>
    )
}
export default Navbar;
```

2) Navbar.css

```
.navbar {
  display: flex;
  justify-content: space-around;
  box-shadow: 0 1px 3px -2px #000000;
background: #faf9f6;
padding: 10px 20px;
align-items: center;
}
.nav-logo {
  display: flex;
  align-items: center;
  gap: 10px;
  padding: 0px 90px 0px 0px;
}
.nav-logo img{
    width: 140px;
    height: auto;
}
.nav-login-cart img{
  width: 50px;
    height: 50px;
}
.nav-menu{
  display: flex;
  align-items: center;
  list-style: none;
  gap: 50px;
  color: #303030;
  font-size: 20px;
```

```
  font-weight: 600;
  padding: 0;
}
.nav-menu li{
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  gap: 3px;
  cursor: pointer;
}
.menu-link {
  color: rgb(48, 48, 48); /* Цвет текста ссылок */
  transition: color .2s ease-in-out;
}
.menu-link:hover{
  color: #457b60;
}
.nav-login-cart {
display: flex;
align-items: center;
gap: 45px;
}
.nav-login-cart button {
  width: 150px;
  height: 50px;
  outline: none;
 background: #faf9f6;
  color: rgb(48, 48, 48);
  font-size: 20px;
```

```
  font-weight: 500;
  cursor: pointer;
  font-family: 'Montserrat';
}
.nav-login-cart button:active{
  color: #395748;
}
.nav-cart-count {
  width: 22px;
  height: 22px;
  display: flex;
  justify-content: center;
  align-items: center;
  margin-top: -35px;
  margin-left: -55px;
  border-radius: 11px;
  font-size: 14px;
  background: #395748;
  color: white;
}
.nav-drop-down{
width: 32px;
height: 32px;
display: none;
}
@media(max-width:1280px){
  .navbar{
    padding: 12px 50px;
  }
  .nav-logo p{
```

```
    font-size: 25px;
  }
  .nav-menu {
    gap: 30px;
    font-size: 16px;
  }
  .nav-login-cart{
    gap: 30px;
  }
  .nav-logo img{
    width: 120px;
    height: auto;
}
.nav-logo {
  padding: 0px 27px 0px 0px;
}
.nav-login-cart button{
  width: 120px;
  height: 45px;
  font-size: 16px;
}
.nav-cart-count{
  margin-left: -40px;
  font-size: 12px;
}
}
@media (max-width:1024px) {
  .navbar{
    padding: 12px 30px;
  }
```

```
  .nav-menu{
    gap: 25px;
    font-size: 14px;
  }
  .nav-login-cart button{
    width: 80px;
    height: 35px;
    font-size: 14px;
  }
  .nav-login-cart img{
    width: 30px;
    height:auto;
  }
  .nav-cart-count{
    width: 18px;
    height: 18px;
  }
}
@media(max-width:800px){
  .navbar{
    padding: 10px 0px;
  }
  .nav-logo img{
    width: 90px;
    height: auto;
}
.nav-logo {
  padding: 0px 0px 0px 60px;
}
  .nav-drop-down{
```

```
    display: block;
    rotate: -90deg;
    transition: 0.5s;
  }
  .nav-menu{
      display: none;
      height: 80px;
      width: 100%;
      position: absolute;
      background-color: #faf9f6;
      justify-content: center;
      top: 72px;
  }
  .nav-menu-visible{
    display: flex;
  }
  .nav-drop-down.open{
    transform: rotate(90deg);
  }
}
@media(max-width:500px) {
  .navbar{
    padding: 8px 0px;
    gap: 0;
  }
  .nav-logo{
    transform: scale(0.8);
  }
  .nav-menu{
    height: 70px;
```

```
    top: 68px;
  }
  .nav-login-cart{
    transform: scale(0.8);
  }
}
@media(max-width:390px) {
  .navbar{
    padding: 20px 0px;
    gap: 0;
  }
  .nav-logo{
    transform: scale(0.8);
    padding: 0px;
  }
  .nav-menu{
    height: 70px;
    top: 68px;
  }
  .nav-login-cart{
    transform: scale(0.8);
  }
}
```

3) Hero.jsx

```
import React from 'react';
import { useNavigate } from 'react-router-dom';
import './Hero.css';
import hero_image from '../assets/hero_image.png';
```

```
const Hero = () => {
  const navigate = useNavigate();
  const handleViewClick = () => {
    navigate('/womens'); // Navigate to the "New Collections" page
  };
  return (
    <div className='hero'>
      <div className="hero-left">
        <h2>Тільки НОВИНКИ</h2>
        <div>
          <p>НОВІ</p>
          <p>Колекції</p>
          <p>Для жінок</p>
        </div>
        <div className="hero-latest-btn" onClick={handleViewClick}>
          <div>Дивитися</div>
        </div>
      </div>
      <div className="hero-right">
        <img src={hero_image} alt="Hero" />
      </div>
    </div>
  );
};
export default Hero;
```

4) Hero.css

```
.hero{
  height: 80vh;
```

```
  background: white;
  display: flex;
  margin: 30px auto;
  background: #ffffff;
}
.hero-left {
  flex: 1;
  display: flex;
  flex-direction: column;
  justify-content: center;
  gap: 20px;
  padding-left: 180px;
  line-height: 1.1;
}
.hero-left h2 {
  color: rgb(48, 48, 48);
  font-size: 26px;
  font-weight: 300;
}
.hero-left p{
  color: rgb(48, 48, 48);
  font-size: 50px;
  font-weight: 600;
}
.hero-latest-btn {
  display: flex;
  justify-content: center;
  align-items: center;
  gap: 15px;
  width: 310px;
```

```
  height: 70px;
 cursor: pointer;
  margin-top: 30px;
  background: #395748;
  color: #ffffff;
  font-size: 22px;
  font-weight: 500;
}
.hero-latest-btn img{
  width: 30px;
}
.hero-right img{
  width: 550px;
  height: 550px;
}
.hero-right{
  flex: 1;
  display: flex;
  align-items: center;
  justify-content: center;
}
@media(max-width:1280px) {
  .hero{
    height: auto;
  }
  .hero-left{
  padding-left: 100px;
  }
  .hero-left h2{
    font-size: 22px;
```

```
    font-weight: 300;
  }
  .hero-left p{
    font-size: 40px;
    font-weight: 600;
  }
.hero-latest-btn{
  gap: 10px;
  width: 250px;
  height: 60px;
  margin-top: 20px;
  font-size: 18px;
}
.hero-right img{
  width: 500px;
  height: auto;
}
}
@media(max-width:1024px) {
  .hero{
    height: auto;
  }
  .hero-left{
  padding-left: 80px;
  }
  .hero-left h2{
    font-size: 20px;
    font-weight: 600;
  }
  .hero-left p{
```

```
    font-size: 35px;
    font-weight: 300;
  }
.hero-latest-btn{
  width: 220px;
  height: 50px;
  font-size: 16px;
}
.hero-right img{
  width: 400px;
  height: auto;
}
}
@media(max-width:800px) {
  .hero{
    height: auto;
  }
  .hero-left{
  padding-left: 30px;
  }
  .hero-left h2{
    font-size: 16px;
    font-weight: 600;
  }
  .hero-left p{
    font-size: 30px;
    font-weight: 300;
  }
.hero-latest-btn{
  width: 175px;
```

```
  height: 40px;
  font-size: 13px;
}
.hero-right img{
  width: 330px;
  height: auto;
}
}
@media(max-width:500px) {
  .hero{
    flex-direction: column;
    height: auto;
  }
  .hero-right{
    display: none;
  }
  .hero-left h2{
    font-size: 18px;
    font-weight: 600;
  }
  .hero-left p{
    font-size: 50px;
    font-weight: 300;
  }
.hero-latest-btn{
  width: 200px;
  height: 45px;
  font-size: 14px;
}
.hero-right img{
```

```
  width: 330px;
  height: auto;
}
}
```

5) Popular.jsx

```
import React, { useEffect, useState } from 'react'
import './Popular.css'
import Item from '../Item/Item'
const Popular = () => {
  const [popularProducts, setPopularProducts] = useState([]);
  useEffect(()=>{
     fetch('http://localhost:4000/popularinwomen')
     .then((response)=>response.json())
     .then((data)=>setPopularProducts(data));
  },[])
  return (
    <div className='popular'>
      <h1>Популярне серед жінок</h1>
      <hr />
      <div className="popular-item">
        {popularProducts.map((item,i)=>{
          return <Item key={i} id={item.id} name={item.name} image={item.image} new_price={item.new_price} old_price={item.old_price}/>
        })}
      </div>
    </div>
  )
}
```

export default Popular

6) Popular.css

```
.popular{
  display: flex;
  flex-direction: column;
  align-items: center;
  padding-top: 25px;
padding-bottom: 30px;
height: auto;
}
.popular h1{
  color: rgb(48, 48, 48);
  font-size: 40px;
  font-weight: 600;
  padding-bottom: 13px;
}
.popular hr{
  width: 300px;
  height: 2px;
  border-radius: 10px;
  background-color: rgb(48, 48, 48);
}
.popular-item{
  margin-top: 50px;
  display: flex;
gap: 40px;
}
@media(max-width:1280px) {
```

```
  .popular{
    height: auto;
  }
  .popular h1{
    font-size: 40px;
  }
  .popular hr{
    width: 160px;
    height: 4px;
  }
.popular-item{
  gap: 20px;
  margin-top: 30px;
}
}
@media(max-width:1024px) {
  .popular{
    height: 45vh;
  }
  .popular h1{
    font-size: 30px;
  }
  .popular hr{
    width: 120px;
    height: 3px;
  }
.popular-item{
  gap: 15px;
  margin-top: 20px;
}
```

```
}
@media(max-width:800px) {
  .popular{
    height: 45vh;
    gap: 6px;
  }
  .popular h1{
    font-size: 20px;
  }
  .popular hr{
    width: 100px;
  }
.popular-item{
  gap: 5px;
}
}
@media(max-width:500px) {
  .popular{
    height: 100vh;
    gap: 6px;
  }
  .popular-item{
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
  }
}
@media(max-width:390px) {
  .popular{
    height: 90vh;
```

```
    gap: 6px;
  }
  .popular-item{
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
  }
}
```

7) Offers.jsx

```
import React from 'react'
import './Offers.css'
import exclusive_image from '../assets/exclusive_image.png'
const Offers = () => {
  return (
    <div className='offers'>
      <div className="offers-left">
        <h1>Ексклюзивні</h1>
        <h1>пропозиції для тебе</h1>
        <p>ТІЛЬКИ НА БЕСТСЕЛЛЕРИ</p>
      </div>
      <div className="offers-right">
        <img src={exclusive_image} alt="" />
      </div>
    </div>
  )
}
export default Offers
```

8) Offers.css

```
.offers{
  height: 85vh;
  display: flex;
  margin: auto;
  padding: 0px 140px;
  background: #303030;
  margin-bottom: 25px;
}
.offers-left{
  flex: 1;
  display: flex;
  flex-direction: column;
  justify-content: center;
}
.offers-left h1{
  color: #fffaf2;
  font-size: 60px;
  font-weight: 600;
}
.offers-left p{
  color: #ff8000;
  font-size: 22px;
  font-weight: 600;
  margin-top: 30px;
}
.offers-left button{
  width: 282px;
  height: 70px;
```

```
  background: #fffaf2;
  color: #303030;
  cursor: pointer;
  font-family: 'Montserrat';
  border: none;
  font-size: 22px;
  font-weight: 500;
  margin-top: 30px;
}
.offers-right{
  flex: 1;
  display: flex;
  align-items: center;
  justify-content: flex-end;
  padding-top: 10px;
}
.offers-right img{
    width: 530px;
    height: 530px;
  }
@media(max-width:1280px) {
  .offers{
    padding: 0px 80px;
    height: 57vh;
  }
  .offers-left h1{
    font-size: 45px;
  }
  .offers-left p{
    font-size: 20px;
```

```
  }
  .offers-right img{
    width: 500px;
    height: auto;
  }
  .offers-left button{
    width: 250px;
    height: 70px;
    font-size: 20px;
  }
}
@media(max-width:1024px) {
  .offers{
    padding: 0px 80px;
    height: 40vh;
  }
  .offers-left h1{
    font-size: 40px;
  }
  .offers-left p{
    font-size: 18px;
  }
  .offers-right img{
    width: 340px;
    height: auto;
  }
  .offers-left button{
    width: 200px;
    height: 60px;
    font-size: 18px;
```

```
  }

}
@media(max-width:800px) {
  .offers{
    margin-bottom: 60px;
  }
  .offers-left h1{
    font-size: 22px;
  }
  .offers-left p{
    font-size: 13px;
  }
  .offers-right img{
    width: 240px;
    height: auto;
  }
  .offers-left button{
    width: 130px;
    height: 40px;
    font-size: 12px;
  }
}
@media(max-width:500px) {
  .offers{
    height: 35vh;
    margin-bottom: 30px;
  }
  .offers-left h1{
    font-size: 16px;
```

```
  }
  .offers-left p{
    font-size: 10px;
  }
  .offers-right img{
    width: 140px;
    height: auto;
  }
  .offers-left button{
    width: 100px;
    height: 30px;
    font-size: 10px;
    margin-top: 12px;
  }
}
@media(max-width:390px) {
  .offers{
    margin-top: 20px;
    height: 30vh;
    margin-bottom: 20px;
    gap: 30px;
    padding-left: 40px;
    padding-right: 40px;
  }
  .offers-left h1{
    font-size: 14px;

  }
  .offers-left p{
    font-size: 10px;
```

```
  }
  .offers-right img{
    width: 160px;
    height: auto;
  }
  .offers-left button{
    width: 100px;
    height: 30px;
    font-size: 10px;
    margin-top: 12px;
  }
}
```

9) Newcollection.jsx

```
 import React, { useEffect, useState } from 'react'
import './Newcollection.css'
import Item from '../Item/Item'
const Newcollection = () => {
  const[new_collection,setNew_collection] = useState([]);
  useEffect(()=>{
    fetch('http://localhost:4000/newcollections')
    .then((response)=>response.json())
    .then((data)=>setNew_collection(data));
  },[])
  return (
    <div className='new-collections'>
      <h1>НОВІ Колекції</h1>
      <hr />
      <div className="collections">
```

```
        {new_collection.map((item,i)=>{
          return <Item key={i} id={item.id} name={item.name} image={item.image}
new_price={item.new_price} old_price={item.old_price}/>
        })}
        </div>
      </div>
    )
}
export default Newcollection
```

10) Newcollection.css

```
.new-collections{
  display: flex;
  flex-direction: column;
  align-items: center;
  padding-top: 25px;
padding-bottom: 30px;
height: auto;
}
.new-collections h1{
  color: rgb(48, 48, 48);
  font-size: 40px;
  font-weight: 600;
  padding-bottom: 13px;
}
.new-collections hr{
  width: 300px;
  height: 2px;
  border-radius: 10px;
```

```
  background-color: rgb(48, 48, 48);
}
.collections{
  display: grid;
  grid-template-columns: 1fr 1fr 1fr 1fr;
  margin-top: 30px;
  gap: 30px;
}
@media(max-width:1280px) {
  .new-collections{
    height: auto;
  }
  .new-collections h1{
    font-size: 40px;
  }
  .new-collections hr{
    width: 160px;
    height: 4px;
  }
.collections{
  gap: 20px;
  margin-top: 30px;
}
}
@media(max-width:1024px) {
  .new-collections{
    height: 83vh;
  }
  .new-collections h1{
    font-size: 30px;
```

```
  }
  .new-collections hr{
    width: 120px;
    height: 3px;
  }
.collections{
  gap: 15px;
  margin-top: 20px;
}
}
@media(max-width:800px) {
  .new-collections{
    height: 85vh;
    gap: 6px;
  }
  .new-collections h1{
    font-size: 20px;
  }
  .new-collections hr{
    width: 100px;
  }
.collections{
  gap: 5px;
}
}
@media(max-width:500px) {
  .new-collections{
    height: 180vh;
    gap: 6px;
  }
```

```
  .collections{
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
  }
}
@media (max-width:390px) {
  .new-collections{
  height: 170vh;
  gap: 6px;
}
.collections{
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 20px;
}
}
```

11) Newsletter.jsx

```
 import React, { useState } from 'react';
import axios from 'axios';
import '../Newsletter/Newsletter.css';
const Newsletter = () => {
  const [email, setEmail] = useState('');
  const [message, setMessage] = useState('');
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post('http://localhost:4000/subscribe', { email });
```

```
      setMessage(response.data);
      setEmail('');
    } catch (error) {
      console.error('Subscription error:', error); // Log the error for debugging
      setMessage('Error subscribing');
    }
  };
  return (
    <div className='news-letter'>
      <h1>Отримуй ексклюзивні пропозиції</h1>
      <p>Підпишись на нашу розсилку та залишайтеся в курсі</p>
      <form onSubmit={handleSubmit}>
        <div>
          <input
            type="email"
            placeholder='Твій email'
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            required
          />
          <button type="submit">Підписатися</button>
        </div>
      </form>
      {message && <p>{message}</p>}
    </div>
  );
};
export default Newsletter;
```

12) Newsletter.css

```
.news-letter{
  height: 45vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  margin: auto;
  padding: 0px 140px;
  background: #9cab9d;
  gap: 30px;
}
.news-letter h1{
  color: #303030;
  font-size: 50px;
  font-weight: 600;
}
.news-letter p{
  color: #303030;
  font-size: 20px;
}
.news-letter div{
  display: flex;
  align-items: center;
  justify-content: space-between;
  background: white;
  width: 730px;
  height: 70px;
  border: 1.5px solid rgb(48, 48, 48);
}
```

```
.news-letter input {
  width: 500px;
  margin-left: 30px;
  border: none;
  outline: none;
  color: rgb(48, 48, 48);
  cursor: text;
  font-family: 'Montserrat';
  font-size: 18px;
}
.news-letter div button{
  width: 220px;
  height: 71px;
  background: #303030;
  color: white;
cursor: pointer;
font-size: 18px;
font-family: 'Montserrat';
}
@media(max-width:1280px) {
 .news-letter{
  height: 25vh;
  padding: 0px 80px;
  gap: 20px;
 }
 .news-letter h1{
  font-size: 36px;
 }
 .news-letter p{
  font-size: 16px;
```

```
 }
 .news-letter div{
  width: 600px;
  height: 60px;
 }
 .news-letter div input{
  width: 400px;
 }
 .news-letter div  button{
width: 200px;
height: 60px;
 }
}
@media(max-width:1024px) {
  .news-letter{
    height: 25vh;
   }
   .news-letter h1{
    font-size: 30px;
   }
   .news-letter p{
    font-size: 14px;
   }
   .news-letter div{
    width: 500px;
    height: 50px;
   }
   .news-letter div input{
    width: 300px;
    font-size: 14px;
```

```
  }
  .news-letter div  button{
 width: 140px;
 height: 50px;
 font-size: 14px;
  }
}
@media(max-width:800px) {
 .news-letter{
  height: 25vh;
  }
  .news-letter h1{
   font-size: 26px;
  }
  .news-letter p{
   font-size: 13px;
  }
  .news-letter div{
   width: 400px;
   height: 40px;
  }
  .news-letter div input{
   width: 200px;
   font-size: 13px;
  }
  .news-letter div  button{
 width: 120px;
 height: 40px;
 font-size: 13px;
  }
```

```
}
@media(max-width:500px) {
  .news-letter{
    height: 25vh;
    width: 100%;
    padding: 0;
    gap: 15px;
   }
   .news-letter h1{
    font-size: 16px;
   }
   .news-letter p{
    font-size: 13px;
    max-width: 200px;
    text-align: center;
   }
   .news-letter div{
    width:290px;
   }
   .news-letter div input{
    width: 130px;
   }
}
```

13) footer.jsx

```
import React from 'react';
import './footer.css';
import footer_logo from '../assets/logo.png';
import instagram_icon from '../assets/instagram_icon.png';
```

```
import pinterest_icon from '../assets/pinterest_icon.png';
import telegram_icon from '../assets/telegram_icon.png';
const Footer = () => {
 return (
  <div className='footer'>
   <div className="footer-container">
    <div className="footer-logo">
     <img src={footer_logo} alt="Logo" />
     <p>INSIDE OUT</p>
    </div>
    <ul className="footer-links">
     <li><a href="#about">Про компанію</a></li>
     <li><a href="#products">Продукція</a></li>
     <li><a href="#stores">Магазини</a></li>
     <li><a href="#about-us">Про нас</a></li>
     <li><a href="#contact">Контакти</a></li>
    </ul>
    <div className="footer-social-icons">
     <a href="https://instagram.com" target="_blank" rel="noopener
noreferrer">
      <img src={instagram_icon} alt="Instagram" />
     </a>
     <a href="https://pinterest.com" target="_blank" rel="noopener noreferrer">
      <img src={pinterest_icon} alt="Pinterest" />
     </a>
     <a href="https://telegram.org" target="_blank" rel="noopener noreferrer">
      <img src={telegram_icon} alt="Telegram" />
     </a>
    </div>
   </div>
```

```
      <div className="footer-copyright">
        <hr />
        <p>Copyright © 2024 - All Rights Reserved.</p>
      </div>
    </div>
  );
};
export default Footer;
```

14) footer.css

```
.footer {
  background-color: #faf9f6;
  color: #333;
  padding: 40px 20px;
  text-align: center;
}
.footer-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  max-width: 1200px;
  margin: 0 auto;
}
.footer-logo {
  display: flex;
  align-items: center;
  margin-bottom: 20px;
}
.footer-logo img {
```

```
  width: 84px;
  height: 58px;
  margin-right: 10px;
}
.footer-logo p {
  font-size: 1.5rem;
  font-weight: bold;
}
.footer-links {
  list-style-type: none;
  padding: 0;
  margin-bottom: 20px;
}
.footer-links li {
  margin: 5px 0;
}
.footer-links a {
  color: #333;
  text-decoration: none;
  font-size: 1rem;
  transition: color 0.3s ease;
}
.footer-links a:hover {
  color: #457b60;
}
.footer-social-icons {
  display: flex;
  justify-content: center;
  margin-bottom: 20px;
}
```

```
.footer-social-icons a {
  margin: 0 10px;
  transition: transform 0.3s ease;
}
.footer-social-icons img {
  width: 30px;
  height: 30px;
}
.footer-social-icons a:hover {
  transform: scale(1.2);
}
.footer-copyright {
  width: 100%;
}
.footer-copyright hr {
  border-color: #444;
  margin: 20px 0;
}
.footer-copyright p {
  margin: 0;
  font-size: 0.9rem;
}
@media (min-width: 768px) {
  .footer-container {
    flex-direction: row;
    justify-content: space-between;
    align-items: flex-start;
  }
  .footer-logo {
    margin-bottom: 0;
```

```
  }
  .footer-links {
    display: flex;
    justify-content: center;
    gap: 20px;
  }
  .footer-links li {
    margin: 0;
  }
  .footer-links a {
    font-size: 1.1rem;
  }
  .footer-social-icons {
    justify-content: flex-end;
  }
}
@media (max-width:1280px) {
  .footer{
    gap: 20px;
  }
  .footer-logo{
    gap: 10px;
    align-items: end;
  }
  .footer-logo img{
      width: 64px;
      height: auto;
  }
  .footer-logo p{
    font-size: 28px;
```

```
  }
  .footer-links{
    font-size: 18px;
    gap: 15px;
  }
  .footer-social-icons img{
width: 32px;
height: auto;
  }
  .footer-copyright{
    font-size: 16px;
  }
}
@media (max-width:500px) {
  .footer{
    gap: 20px;
  }
  .footer-logo{
    gap: 10px;
    align-items: end;
  }
  .footer-logo img{
      width: 64px;
      height: auto;
  }
  .footer-logo p{
    font-size: 22px;
  }
  .footer-links{
    font-size: 14px;
```

```
    gap: 15px;
  }
  .footer-social-icons img{
width: 32px;
height: auto;
  }
  .footer-copyright{
    font-size: 12px;
  }
}
@media (max-width:390px) {
  .footer{
    gap: 20px;
  }
  .footer-logo{
    gap: 10px;
    align-items: end;
  }
  .footer-logo img{
      width: 64px;
      height: auto;
  }
  .footer-logo p{
    font-size: 20px;
  }
  .footer-links{
    font-size: 11px;
    gap: 15px;
  }
  .footer-social-icons img{
```

```
width: 32px;
height: auto;
  }
  .footer-copyright{
    font-size: 12px;
  }
}
```

15) AllProducts.jsx

```
import React, { useContext, useState, useEffect } from 'react';
import { ShopContext } from '../../context/ShopContext';
import Item from '../Item/Item';
import './AllProducts.css';
const AllProducts = () => {
  const { all_product } = useContext(ShopContext);
  const [sortField, setSortField] = useState('name');
  const [sortOrder, setSortOrder] = useState('asc');
  const [selectedCategory, setSelectedCategory] = useState('all');
  const [currentPage, setCurrentPage] = useState(1);
  const productsPerPage = 12;
  useEffect(() => {
    setCurrentPage(1); // Reset page when sorting or category changes
  }, [sortField, sortOrder, selectedCategory]);
  const handleSortChange = (field) => {
    const order = sortField === field && sortOrder === 'asc' ? 'desc' : 'asc';
    setSortField(field);
    setSortOrder(order);
  };
```

```
const handleCategoryChange = (category) => {
  setSelectedCategory(category);
  console.log("Selected Category:", category); // Debug log
};
const sortedProducts = [...all_product].sort((a, b) => {
  if (sortField === 'name') {
    return sortOrder === 'asc' ? a.name.localeCompare(b.name) : b.name.localeCompare(a.name);
  } else if (sortField === 'new_price') {
    return sortOrder === 'asc' ? a.new_price - b.new_price : b.new_price - a.new_price;
  } else {
    return 0;
  }
});
const filteredProducts = selectedCategory === 'all' ? sortedProducts : sortedProducts.filter(item => {
  console.log("Item Category:", item.category.toLowerCase());
  return item.category.toLowerCase() === selectedCategory;
});
const displayedProducts = filteredProducts.slice(0, currentPage * productsPerPage);
const loadMoreProducts = () => {
  setCurrentPage(prevPage => prevPage + 1);
  // Control scroll position after loading more products
  const scrollPosition = window.scrollY;
  window.scrollTo(0, scrollPosition);
};
return (
  <div className='all-products'>
```

```
<h1>Всі товари</h1>
<div className="all-products-sort">
 <span onClick={() => handleSortChange('name')}>
  Сортувати за назвою {sortField === 'name' && (sortOrder === 'asc' ? '(від А до Я)' : '(Від Я до А)')}
 </span>
 <span onClick={() => handleSortChange('new_price')}>
  Сортувати за ціною {sortField === 'new_price' && (sortOrder === 'asc' ? '(від найдешевшого)' : '(від найдорожчого)')}
 </span>
</div>
<div className="shopcategory-container">
 <div className="shopcategory-content">
  <div className="shopcategory-indexsort">
   <p>
    <span>Показано {displayedProducts.length}</span> з {filteredProducts.length} товарів
   </p>
  </div>
  <div className="shopcategory-products">
   {displayedProducts.map((item, i) => (
    item && (
     <Item key={i} id={item.id} name={item.name} image={item.image} new_price={item.new_price} old_price={item.old_price} />
    )
   ))}
  </div>
  {displayedProducts.length < filteredProducts.length && (
   <button className="all-products-loadmore" onClick={loadMoreProducts}>
```

```
            Більше товарів
          </button>
        )}
      </div>
    </div>
  </div>
 );
};
export default AllProducts;
```

16) AllProducts.css

```
.all-products {
 padding: 20px;
 max-width: 1200px;
 margin: 0 auto;
}
.all-products h1 {
 text-align: center;
 margin-bottom: 20px;
}
.all-products-sort {
 display: flex;
 justify-content: center;
 margin-bottom: 20px;
 gap: 20px;
}
.all-products-sort span {
 cursor: pointer;
 padding: 10px;
```

```
  background-color: #808f8a;
  color: white;
  border-radius: 5px;
  transition: background-color 0.3s ease;
}
.all-products-sort span:hover {
  background-color: #5a6562;
}
.all-products-grid {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  justify-content: center;
}
.all-products-grid .item {
  flex: 1 1 200px; /* Adjust this value based on your design */
  max-width: 300px;
}
.all-products-loadmore {
  cursor: pointer;
  text-align: center;
  padding: 10px;
  background-color: #808f8a;
  color: white;
  border-radius: 5px;
  margin-top: 20px;
  transition: background-color 0.3s ease;
  border: none; /* Make sure there are no default button borders */
}
.all-products-loadmore:hover {
```

```
  background-color: #5a6562;
}
/* Media queries for responsive design */
/* Screens smaller than 1280px */
@media (max-width: 1280px) {
  .shopcategory-products .item {
    flex: 1 1 calc(33.333% - 20px); /* 3 items per row */
    max-width: calc(33.333% - 20px);
  }
}
/* Screens smaller than 1024px */
@media (max-width: 1024px) {
  .shopcategory-products .item {
    flex: 1 1 calc(50% - 20px); /* 2 items per row */
    max-width: calc(50% - 20px);
  }
}
/* Screens smaller than 800px */
@media (max-width: 800px) {
  .shopcategory-container {
    flex-direction: column;
  }
  .shopcategory-content {
    padding-left: 120px;
  }
  .shopcategory-products .item {
    flex: 1 1 calc(50% - 10px); /* 2 items per row */
    max-width: calc(50% - 10px);
  }
}
```

```
/* Screens smaller than 500px */
@media (max-width: 500px) {
  .all-products-sort {
    flex-direction: column; /* Stack sorting options vertically */
  }
  .all-products-sort span {
    width: 100%;
    text-align: center;
  }
  .shopcategory-content {
    padding-left: 110px;
  }
  .shopcategory-products .item {
    flex: 1 1 calc(100% - 10px); /* 1 item per row */
    max-width: calc(100% - 10px);
  }
  .all-products-loadmore {
    padding: 8px;
    font-size: 0.9rem;
  }
}
/* Screens smaller than 390px */
@media (max-width: 390px) {
  .all-products-sort span {
    font-size: 0.8rem;
  }
  .shopcategory-content {
    padding: 5px;
  }
  .all-products-loadmore {
```

```
    padding: 6px;
    font-size: 0.8rem;
  }
}
```

17. Breadcrum.jsx

```
import React from 'react'
import './Breadcrum.css'
import arrow_icon from '../assets/side_arrow.png'
const Breadcrum = (props) => {
  const {product} = props;
  return (
    <div className='breadcrum'>
      Головна <img src={arrow_icon} alt="" /> Магазин <img src={arrow_icon} alt="" /> {product.category} <img src={arrow_icon} alt="" /> {product.name}
    </div>
  )
}
export default Breadcrum
```

18. Breadcrum.css

```
.breadcrum{
  display: flex;
  align-items: center;
  gap: 8px;
  color:rgb(48, 48, 48);
  font-size: 16px;
  font-weight: 600;
```

```
  margin: 60px 170px;
  text-transform: capitalize;
}
@media (max-width:1280px) {
  .breadcrum{
    margin: 30px 50px;
    font-size: 14px;
  }
}
@media (max-width:1024px) {
  .breadcrum{
    margin: 30px 30px;
    font-size: 13px;
  }
}
@media (max-width:800px) {
  .breadcrum{
    margin: 30px 10px;
    font-size: 12px;
  }
}
@media (max-width:500px) {
  .breadcrum{
    font-size: 10px;
  }
}
```

19. Cartitems.jsx

```
import React, { useContext } from 'react';
```

```
import axios from 'axios';
import './Cartitems.css';
import { ShopContext } from '../../context/ShopContext';
import remove_icon from '../assets/cart_cross_iconn.png';

const Cartitems = () => {
  const { getTotalCartAmount, all_product, cartItems, removeFromCart } = useContext(ShopContext);

  const cartItemsInCart = all_product.filter(product => cartItems[product.id] > 0);

  const handleOrder = async () => {
    const orderItems = cartItemsInCart.map(product => ({
      productId: product._id, // Ensure you are using MongoDB ObjectId
      quantity: cartItems[product.id]
    }));
    const orderData = {
      items: orderItems,
      totalAmount: getTotalCartAmount()
    };
    try {
      const token = localStorage.getItem('auth-token'); // Assuming you store the auth token in localStorage
      const response = await axios.post('http://localhost:4000/createOrder', orderData, {
        headers: { 'auth-token': token }
      });
      if (response.status === 201) {
        alert('Order placed successfully');
        // Optionally clear the cart or redirect the user
```

```
      }
    } catch (error) {
      console.error('Error placing order', error);
      alert('Failed to place order');
    }
  };
  return (
    <div className='cart-items'>
      <h1>Кошик</h1>
      <div className="cart-items-formatmain">
        <p>Товар</p>
        <p>Назва</p>
        <p>Ціна</p>
        <p>Кількість</p>
        <p>Всього</p>
        <p>Видалити</p>
      </div>
      <hr />
      {cartItemsInCart.map(product => (
        <div key={product.id}>
          <div className="cart-items-format cart-items-formatmain">
            <img src={product.image} alt={product.name} className='cart-icon-product-icon' />
            <p>{product.name}</p>
            <p>${product.new_price ? product.new_price : 0}</p>
            <button className='cart-items-quantity'>{cartItems[product.id]}</button>
            <p>${product.new_price ? product.new_price * cartItems[product.id] : 0}</p>
```

```
            <img  className='cart-items-remove-icon'  src={remove_icon}
onClick={() => removeFromCart(product.id)} alt={`Remove ${product.name} from
cart`} />
          </div>
          <hr />
        </div>
      ))}
      <div className="cart-items-down">
        <div className="cart-items-total">
          <h1>Всього</h1>
          <div>
            <div className="cart-items-totalitem">
              <p>Сума замовлення</p>
              <p>${getTotalCartAmount()}</p>
            </div>
            <hr />
            <div className="cart-items-totalitem">
              <p>Доставка</p>
              <p>Безкоштовно</p>
            </div>
            <hr />
            <div className="cart-items-totalitem">
              <h3>Загальна сума</h3>
              <h3>${getTotalCartAmount()}</h3>
            </div>
          </div>
          <button onClick={handleOrder}>Оформити замовлення</button>
        </div>
        {/* <div className="cart-items-promocode">
          <p>Якщо Ви маєте промокод, то введіть його у поле нижче</p>
```

```
          <div className="cart-items-promo-box">
            <input type="text" placeholder='Промокод' />
            <button>Підтвердити</button>
          </div>
        </div> */}
      </div>
    </div>
  );
}

export default Cartitems;
```

20. Cartitems.css

```
.cart-items{
  margin: 40px 150px;
}
.cart-items h1{
font-weight: 500;
}
.cart-items hr{
  height: 1px;
  background: rgb(48, 48, 48);
  border: 0;
}
.cart-items-formatmain{
  display: grid;
  grid-template-columns: 1fr 2fr 1fr 1fr 1fr 1fr;
  align-items: center;
  gap: 75px;
```

```
  padding: 20px 0px;
  color: rgb(48, 48, 48);
  font-size: 18px;
  font-weight: 500;
}
.cart-items-format{
  font-size: 17px;
  font-weight: 500;
}
.cart-icon-product-icon{
  height: 82px;
}
.cart-items-remove-icon{
  width: 25px;
  margin: 0px 20px;
  cursor: pointer;
}
.cart-items-quantity{
  width: 64px;
  height: 50px;
  border: 2px solid #ebebeb;
  background: #fff;
}
.cart-items-down{
    display: flex;
    margin: 100px 0px;
}
.cart-items-total{
  flex: 1;
  display: flex;
```

```
  flex-direction: column;
  margin-right: 500px;
  gap: 40px;

}
.cart-items-totalitem{
  display: flex;
  justify-content: space-between;
  padding: 15px 0px;
}
.cart-items-total button{
  width: 630px;
  height: 58px;
  outline: none;
  border: none;
  background: rgb(48, 48, 48);
  color: rgb(255, 255, 255);
  font-size: 16px;
  font-weight: 500;
  cursor: pointer;
}
/*
.cart-items-promocode{
  flex: 1;
  font-size: 16px;
  font-weight: 500;
}
.cart-items-promocode p{
  color: rgb(48, 48, 48);
}
```

```
.cart-items-promo-box{
  width: 504px;
  margin-top: 15px;
  padding-left: 20px;
  height: 58px;
  background: rgb(239, 239, 239);
}
.cart-items-promo-box input{
  border: none;
  outline: none;
  background: transparent;
  font-size: 16px;
  width: 330px;
  height: 50px;
}
.cart-items-promo-box button{
    width: 170px;
    height: 58px;
    font-size: 16px;
    background: rgb(48, 48, 48);
    color: #ffffff;
    cursor: pointer;
} */
@media (max-width: 1280px) {
  .cart-items{
    margin: 60px 50px;
  }
  .cart-items-formatmain{
    grid-template-columns: 0.5fr 3fr 0.5fr 0.5fr 0.5fr 0.5fr;
  gap: 20px;
```

```
padding: 15px 0px;
font-size: 15px;
}
.cart-icon-product-icon{
  height: 50px;
}
.cart-items-remove-icon{
  margin: auto;
}
.cart-items-quantity{
  width: 40px;
  height: 30px;
}
.cart-items-down{
  margin: 80px 0px;
  flex-direction: column;
  gap: 80px;
}
.cart-items-total{
  margin: 0;
}
.cart-items-promo-box{
  width: auto;
  max-width: 500px;
}
.cart-items-promo-box input{
  width: 100%;
}
.cart-items-promo-box button{
  width: 200px;
```

```
    margin-left: -200px;
  }
  .cart-items-total button{
    width: auto;
    height: 58px;
    outline: none;
    border: none;
    background: rgb(48, 48, 48);
    color: rgb(255, 255, 255);
    font-size: 16px;
    font-weight: 500;
    cursor: pointer;
  }
}
@media (max-width: 1024px) {
  .cart-items-total button{
    width: auto;
    height: 58px;
    outline: none;
    border: none;
    background: rgb(48, 48, 48);
    color: rgb(255, 255, 255);
    font-size: 16px;
    font-weight: 500;
    cursor: pointer;
  }
}
@media (max-width: 800px) {
  .cart-items-total button{
    width: auto;
```

```
    height: 58px;
    outline: none;
    border: none;
    background: rgb(48, 48, 48);
    color: rgb(255, 255, 255);
    font-size: 16px;
    font-weight: 500;
    cursor: pointer;
  }
}

@media (max-width: 500px) {
  .cart-items-formatmain{
    display: none;
    grid-template-columns: 0.5fr 3fr 0.5fr;
    gap: 10px;
  }
  .cart-items-format{
    display: grid;
  }

}
@media (max-width: 390px) {
  .cart-items-total button{
    width: 280px;
    height: 58px;
    outline: none;
    border: none;
    background: rgb(48, 48, 48);
    color: rgb(255, 255, 255);
```

```
    font-size: 16px;
    font-weight: 500;
    cursor: pointer;
  }
}
```

21. Descriptionbox.jsx

```
import React from 'react'
import './Descriptionbox.css'
const Descriptionbox = () => {
  return (
    <div className='description-box'>
      <div className="descroption-box-navigator">
        <div className="description-box-navbox">
          Опис
        </div>
        <div className="description-box-navbox fade">
          Відгуки (0)
        </div>
      </div>
      <div className="description-box-description">
        <p>Тканина: 100% бавовна.</p>
        <p>Країна виробництва: Україна.</p>
      </div>
    </div>
  )
}
export default Descriptionbox
```

22. Descriptionbox.css

```
.description-box{
  margin: 120px 170px;
}
.descroption-box-navigator{
  display: flex;
}
.description-box-navbox{
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 16px;
  font-weight: 600;
  width: 171px;
  height: 70px;
  border: 1px solid #d0d0d0;
}
.description-box-navbox.fade{
background: #fbfbfb;
color: #555;
}
.description-box-description{
  display: flex;
  flex-direction: column;
  gap: 25px;
  border: 1px solid #d0d0d0;
  padding: 48px;
  padding-bottom: 70px;
  text-align: justify;
```

```
}
@media (max-width:1280px) {
  .description-box{
    margin: 60px;
  }
}
@media (max-width:800px) {
  .description-box{
    margin: 60px 20px;
  }
  .description-box-description{
    font-size: 14px;
  }
}
```

23. Item.jsx

```
import React from 'react'
import './Item.css'
import { Link } from 'react-router-dom'
const Item = (props) => {
  return (
    <div className='item'>
      <Link  to={`/product/${props.id}`}><img  onClick={window.scrollTo(0,0)}
src={props.image} alt="" /></Link>
      <p>{props.name}</p>
      <div className="item-prices">
        <div className="item-price-new">
        {props.new_price}
        </div>
```

```
        <div className="item-price-old">
        {props.old_price}
        </div>
      </div>
    </div>
  )
}
export default Item
```

24. Item.css

```
.item{
  width: 280px;
}
.item img{
  width: 280px;
  height: 370px;
}
.item p{
  margin: 6px 0px;
}
.item-prices{
  display: flex;
  gap: 20px;
}
.item-price-new{
  color: rgb(48, 48, 48);
  font-size: 18px;
  font-weight: 600;
}
```

```
.item-price-old{
  color: rgb(90, 90, 90);
  font-size: 18px;
  font-size: 500;
  text-decoration: line-through;
}
.item:hover{
  transform: scale(1.05);
  transition: 0.6s;
}
@media(max-width:1280px) {
  .item{
    width: 285px;
    font-size: 14px;
  }
  .item img{
    width: 280px;
  }
  .item-price-old{
    font-size: 14px;
  }
}
@media(max-width:1024px) {
  .item{
    width: 230px;
    font-size: 13px;
  }
  .item img{
    width: 230px;
    height: 280px;
```

```
  }
  .item-price-old{
    font-size: 13px;
  }
}
@media(max-width:800px) {
  .item{
    width: 180px;
    font-size: 12px;
  }
  .item img{width: 180px;
    height: 230px;}
  .item-price-old{font-size: 12px;}

}
@media(max-width:500px) {
  .item{
    width: 180px;
  }
  .item img{width: 180px;
  height: 200px;}

}
@media(max-width:390px) {
  .item{
    width: 160px;
  }
  .item img{width: 160px;
    height: 180px;}
}
```

25. ProductDisplay.jsx

```
import React, { useContext } from 'react'
import './productDisplay.css'
import star_icon from '../assets/star_icon.png';
import star_dull_icon from '../assets/star_dull_icon.png';
import { ShopContext } from '../../context/ShopContext';
const ProductDisplay = (props) => {
  const {product} = props;
  const {addToCart} = useContext(ShopContext);
  return (
    <div className='productdisplay'>
      <div className="productdisplay-left">
      <div className="productdisplay-imglist">
        <img src={product.image} alt="" />
        <img src={product.image} alt="" />
        <img src={product.image} alt="" />
        <img src={product.image} alt="" />
      </div>
      <div className="productlist-img">
        <img className='productdisplay-main-img' src={product.image} alt="" />
      </div>
      </div>
      <div className="productdisplay-right">
      <h1>{product.name}</h1>
      <div className="productdisplay-right-stars">
        <img src={star_icon} alt="" />
        <img src={star_icon} alt="" />
        <img src={star_icon} alt="" />
        <img src={star_icon} alt="" />
```

```
          <img src={star_dull_icon} alt="" />
          <p>(122)</p>
        </div>
        <div className="productdisplay-right-prices">
          <div                                    className="productdisplay-right-
oldprice">${product.old_price}</div>
          <div                                    className="productdisplay-right-
newprice">${product.new_price}</div>
        </div>
        <div className="productdisplay-right-description">
         Товар виготовленний з натуральної сировини.
        </div>
        <div className="productdisplay-right-size">
          <h2>Розміри</h2>
          <div className="productdisplay-right-sizes">
            <div>XS</div>
            <div>S</div>
            <div>M</div>
            <div>L</div>
            <div>XL</div>
            <div>XXL</div>
          </div>
        </div>
        <button onClick={()=>{addToCart(product.id)}}>Купити</button>
        <p      className='productdisplay-right-category'><span>Категорія:</span>
Жінки, Спідниці, Міні</p>
        <p              className='productdisplay-right-category'><span>Теги:</span>
Сучасне, Нове</p>
        </div>
      </div>
```

```
  )
}
export default ProductDisplay
```

26. ProductDisplay.css

```
.productdisplay{
  display: flex;
  margin: 0px 130px;
  margin-bottom: 40px;
}
.productdisplay-left{
  display: flex;
  gap: 17px;
}
.productdisplay-imglist{
  display: flex;
  flex-direction: column;
  gap: 16px;
}
.productdisplay-imglist img{
  height: 138px;
}
.productdisplay-main-img{
  width: 486px;
  height: 600px;
}
.productdisplay-right{
  margin: 0px 70px;
  display: flex;
```

```
  flex-direction:column;
}
.productdisplay-right h1{
  color: rgb(48, 48, 48);
  font-size: 30px;
  font-weight: 700;
}
.productdisplay-right h2{
  color: rgb(48, 48, 48);
  font-size: 25px;
  font-weight: 700;
}
.productdisplay-right-stars{
  display: flex;
  align-items: center;
  margin-top: 13px;
  gap: 5px;
  color: rgb(48, 48, 48);
  font-size: 16px;
}
.productdisplay-right-prices{
  display: flex;
  margin: 40px 0px;
  gap: 30px;
  font-size: 24px;
  font-weight: 700;
}
.productdisplay-right-oldprice{
  color: rgb(143, 143, 143);
  text-decoration: line-through;
```

```
}
.productdisplay-right-newprice{
  color: #303030;
}
.productdisplay-right-size{
  margin-top: 55px;
  color: rgb(48, 48, 48);
  font-size: 20px;
  font-weight: 600;
}
.productdisplay-right-sizes{
  display: flex;
  margin: 30px 0px;
  gap: 20px;
  color: rgb(48, 48, 48);
}
.productdisplay-right-sizes div{
  padding: 4px 5px;
  color: rgb(0, 0, 0);
  background: #fcfbfb;
  border: 1px solid rgb(143, 143, 143);
  border-radius: 1px;
  cursor: pointer;
}
.productdisplay-right button{
  padding: 10px 20px;
  width: 200px;
  font-size: 20px;
  font-weight: 600;
  color: rgb(255, 255, 255);
```

```
  background: #395748;
  margin-bottom: 20px;
  border: none;
  outline: none;
  cursor: pointer;
}
.productdisplay-right-category span{
  font-weight: 600;
}
.productdisplay-right-stars img{
  width: 20px;
  height: 20px;
}
@media (max-width:1280px) {
    .productdisplay{
      margin: 0px 60px;
    }
    .productdisplay-left{
      gap: 10px;
    }
    .productdisplay-imglist{
      gap: 10px;
    }
    .productdisplay-imglist img{
      height: 120px;
    }
    .productdisplay-main-img{
      width: auto;
      height: 510px;
    }
```

```
.productdisplay-right{
  margin: 0px 30px;
}
.productdisplay-right h1{
  font-size: 22px;
}
.productdisplay-right-stars{
  gap: 3px;
  font-size: 13px;
}
.productdisplay-right-prices{
  margin: 10px 0px;
  font-size: 18px;
}
.productdisplay-right-description{
  font-size: 13px;
}
.productdisplay-right-size h1{
  margin-top: 20px;
  font-size: 20px;
}
.productdisplay-right-sizes div{
  padding: 14px 20px;
}
.productdisplay-right button{
  width: 150px;
  padding: 15px 0px;
  margin-bottom: 20px;
}
.productdisplay-right-category{
```

```
    margin-top: 5px;
    font-size: 14px;
   }
}
@media (max-width:1024px) {
 .productdisplay{
   margin: 0px 30px;
 }
 .productdisplay-left{
   gap: 5px;
 }
 .productdisplay-imglist img{
   height: 80px;
 }
 .productdisplay-main-img{
   height: 350px;
 }
 .productdisplay-right h1{
   font-size: 18px;
 }
 .productdisplay-right-stars img{
   width: 15px;
   height: auto;
 }
 .productdisplay-right-description{
   font-size: 12px;
 }
 .productdisplay-right-size h1{
   margin-top: 10px 0px;
 }
```

```
  .productdisplay-right-sizes div{
    padding: 10px 16px;
    font-size: 12px;
  }
  .productdisplay-right button{
    width: 120px;
    padding: 10px 0px;
    margin-bottom: 10px;
    font-size: 14px;
  }
  .productdisplay-right-category{
    font-size: 12px;
  }
}
@media (max-width:800px) {
  .productdisplay{
    margin: 0px 10px;
  }
  .productdisplay-left{
    gap: 0px;
    flex-direction: column-reverse;
    align-items: center;
  }
  .productdisplay-imglist{
    flex-direction: row;
    gap: 6px;
  }
  .productdisplay-imglist img{
    height: 72px;
  }
```

```
.productdisplay-main-img{
  height: 310px;
}
.productdisplay-right h1{
  font-size: 14px;
}
.productdisplay-right-stars img{
  width: 13px;
  height: auto;
}
.productdisplay-right-description{
  font-size: 10px;
}
.productdisplay-right-size h1{
  margin-top: 10px 0px;
}
.productdisplay-right-sizes{
  margin-top: 10px;
  gap: 5px;
}
.productdisplay-right-sizes div{
  padding: 4px 11px;
}
.productdisplay-right button{
  width: 100px;
  font-size: 12px;
}
.productdisplay-right-category{
  font-size: 10px;
}
```

```
}
@media (max-width:500px) {
    .productdisplay{
      flex-direction: column;
    }
    .productdisplay-left{
      gap: 10px;
      flex-direction: row;
      margin: auto;
    }
    .productdisplay-imglist{
      flex-direction: column;
      gap: 8px;
    }
    .productdisplay-imglist img{
      height: 75px;
    }
    .productdisplay-main-img{
      height: 330px;
    }
    .productdisplay-right{
      margin: 5px;
    }
    .productdisplay-right h1{
      margin-top: 15px;
      font-size: 20px;
      font-weight: 500;
    }
    .productdisplay-right-stars img{
      width: 15px;
```

```
    height: auto;
  }
  .productdisplay-right-sizes{
    gap: 10px;
    margin: 20px 0px;
  }
  .productdisplay-right-sizes div{
    padding: 10px 16px;
    font-size: 14px;
  }
  .productdisplay-right button{
width: 130px;
font-size: 15px;
padding: 12px 0px;
  }
  .productdisplay-right-category{
    font-size: 16px;
  }
  .productdisplay-right-description{
    font-size: 14px;
  }
}
```

27. Relatedproducts.jsx

```
import React, { useEffect, useState } from 'react'
import './Relatedproducts.css'
import Item from '../Item/Item'
const Relatedproducts = () => {
const[new_collection,setNew_collection] = useState([]);
```

```
  useEffect(()=>{
    fetch('http://localhost:4000/newcollections')
    .then((response)=>response.json())
    .then((data)=>setNew_collection(data));
  },[])
  return (
    <div className='new-collections'>
      <h1>Схожі товари</h1>
      <hr />
      <div className="collections">
      {new_collection.map((item,i)=>{
        return <Item key={i} id={item.id} name={item.name} image={item.image}
new_price={item.new_price} old_price={item.old_price}/>
      })}
      </div>
    </div>
  )
}
export default Relatedproducts
```

28. Relatedproducts.css

```
.related-products{
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 10px;
  height: 90vh;
}
.related-products h1{
```

```
  color: #171717;
  font-size: 50px;
  font-weight: 600;
}
.related-products hr{
  width: 200px;
  height: 6px;
  border-radius: 10px;
  background: #252525;
}
.related-products-item{
  margin-top: 50px;
  display: flex;
  gap: 30px;
}
@media(max-width:1280px) {
  .related-products{
    height: 60vh;
  }
  .related-products h1{
    font-size: 40px;
  }
  .related-products hr{
    width: 160px;
    height: 4px;
  }
.related-products-item{
  gap: 20px;
  margin-top: 30px;
}}
```

```
@media(max-width:1024px) {
  .related-products{
    height: 45vh;
  }
  .related-products h1{
    font-size: 30px;
  }
  .related-products hr{
    width: 120px;
    height: 3px;
  }
.related-products-item{
  gap: 15px;
  margin-top: 20px;
}}
@media(max-width:800px) {
  .related-products{
    height: 45vh;
    gap: 6px;
  }
  .related-products h1{
    font-size: 20px;
  }
  .related-products hr{
    width: 100px;
  }
.related-products-item{
  gap: 5px;
}}
@media(max-width:500px) {
```

```
  .related-products{
    height: 90vh;
    gap: 6px;
  }
  .related-products-item{
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
  }}
@media(max-width:390px) {
  .related-products{
    height: 90vh;
    gap: 6px;
  }
  .related-products-item{
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
  }}
```

29. ShopCategory.jsx

```
import React, { useContext, useState, useEffect } from 'react';
import './CSS/Shopcategory.css';
import { ShopContext } from '../context/ShopContext';
import Item from '../Components/Item/Item';
const ShopCategory = (props) => {
  const { all_product } = useContext(ShopContext);
  const [sortField, setSortField] = useState('name');
  const [sortOrder, setSortOrder] = useState('asc');
```

```
const [currentPage, setCurrentPage] = useState(1);
const productsPerPage = 12;
useEffect(() => {
  setCurrentPage(1); // Reset page when category or sorting changes
}, [props.category, sortField, sortOrder]);
const handleSortChange = (field) => {
  const order = sortField === field && sortOrder === 'asc' ? 'desc' : 'asc';
  setSortField(field);
  setSortOrder(order);
};
const sortedProducts = [...all_product].sort((a, b) => {
  if (sortField === 'name') {
    return sortOrder === 'asc' ? a.name.localeCompare(b.name) : b.name.localeCompare(a.name);
  } else if (sortField === 'new_price') {
    return sortOrder === 'asc' ? a.new_price - b.new_price : b.new_price - a.new_price;
  } else {
    return 0;
  }
});
const filteredProducts = sortedProducts.filter(item => item.category === props.category);
const displayedProducts = filteredProducts.slice(0, currentPage * productsPerPage);
const loadMoreProducts = () => {
  setCurrentPage(prevPage => prevPage + 1);
};
return (
  <div className='shop-category'>
```

```
        <img className='shopcategory-banner' src={props.banner} alt="" />
        <div className="all-products-sort">
          <span onClick={() => handleSortChange('name')}>
            Сортувати за назвою {sortField === 'name' && (sortOrder === 'asc' ? '(від А до Я)' : '(Від Я до А)')}
          </span>
          <span onClick={() => handleSortChange('new_price')}>
            Сортувати за ціною {sortField === 'new_price' && (sortOrder === 'asc' ? '(від найдешевшого)' : '(від найдорожчого)')}
          </span>
        </div>
        <div className="shopcategory-container">
          <div className="shopcategory-content">
            <div className="shopcategory-indexsort">
              <p>
                <span>Показано {displayedProducts.length}</span> з {filteredProducts.length} товарів
              </p>
            </div>
            <div className="shopcategory-products">
              {displayedProducts.map((item, i) => (
                <Item key={i} id={item.id} name={item.name} image={item.image} new_price={item.new_price} old_price={item.old_price} />
              ))}
            </div>
            {displayedProducts.length < filteredProducts.length && (
              <button className="all-products-loadmore" onClick={loadMoreProducts}>
                Більше товарів
              </button>
```

```
        )}
      </div>
    </div>
  </div>
 );
};
export default ShopCategory;
```

30. Shop.jsx

```
import React from 'react'
import Hero from '../Components/Hero/Hero'
import Popular from '../Components/Popular/Popular'
import Offers from '../Components/Offers/Offers'
import Newcollection from '../Components/Newcollection/Newcollection'
import Newsletter from '../Components/Newsletter/Newsletter'
const Shop = () => {
 return (
  <div>
   <Hero/>
   <Popular/>
   <Offers/>
   <Newcollection/>
   <Newsletter/>
  </div>
 )
}
export default Shop
```

31. Product.jsx

```
import React, { useContext } from 'react';
import { useParams } from 'react-router-dom';
import Breadcrum from '../Components/Breadcrums/Breadcrum';
import { ShopContext } from '../context/ShopContext';
import ProductDisplay from '../Components/ProductDisplay/ProductDisplay';
import Descriptionbox from '../Components/Descriptionbox/Descriptionbox';
import Relatedproducts from '../Components/Relatedproducts/Relatedproducts';
const Product = () => {
  const {all_product} = useContext(ShopContext);
  const {productId} = useParams();
  const product = all_product.find((e) => e.id === Number(productId));
  if (!product) {
    return <div>Продукт не найден</div>;
  }
  return (
    <div>
      <Breadcrum product={product} />
      <ProductDisplay product={product}/>
      <Descriptionbox/>
      <Relatedproducts/>
    </div>
  );
};
export default Product;
```

32. LoginSignup.jsx

```
import React, { useState } from 'react';
```

```
import './CSS/LoginSignup.css';
const LoginSignup = () => {
  const [showPassword, setShowPassword] = useState(false);
  const [agreeTerms, setAgreeTerms] = useState(false);
  const [state, setState] = useState("Login");
  const [formData, setFormData] = useState({
    username: "",
    password: "",
    email: "",
    phone: "" // Add this line
  });
  const changeHandler = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };
  const handlePasswordCheckboxChange = () => {
    setShowPassword(!showPassword);
  };
  const handleTermsCheckboxChange = () => {
    setAgreeTerms(!agreeTerms);
  };
  const login = async () => {
    try {
      console.log("Login function executed", formData);
      const response = await fetch('http://localhost:4000/login', { // Use the correct
IP address or localhost
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(formData)
```

```
    });
    if (response.ok) {
      const responseData = await response.json();
      if (responseData.success) {
        localStorage.setItem('auth-token', responseData.token);
        window.location.replace("/");
      } else {
        throw new Error(responseData.error || "Login failed");
      }
    } else {
      const errorData = await response.json();
      throw new Error(errorData.error || 'Network response was not ok');
    }
  } catch (error) {
    console.error("Login failed", error.message);
    alert(`Login failed: ${error.message}`); // Display error to the user
  }};
const signup = async () => {
  try {
    console.log("Sign up function executed", formData);
    const response = await fetch('http://localhost:4000/signup', {
      method: 'POST',
      headers: {
        'Content-Type':'application/json'
      },
      body: JSON.stringify(formData)
    });
    if (response.ok) {
      const responseData = await response.json();
      if (responseData.success) {
```

```
        localStorage.setItem('auth-token', responseData.token);
        window.location.replace("/");
      } else {
        throw new Error(responseData.error || "Signup failed");
      }
    } else {
      throw new Error('Network response was not ok');
    }
  } catch (error) {
    console.error("Signup failed", error.message);
    alert("Signup failed. Please try again."); // Display error to the user
  }
};
return (
  <div className='loginsignup'>
    <div className="loginsignup-container">
      <h1>{state}</h1>
      <div className="loginsignup-fields">
        {state === "Sign up" &&
          <input
            name='username'
            value={formData.username}
            onChange={changeHandler}
            type="text"
            placeholder='Your name'
          />
        }
        {state === "Sign up" &&
          <input
            name='phone'
```

```
        value={formData.phone}
        onChange={changeHandler}
        type="tel"
        placeholder='Phone number'
      />
    }
    <input
      name='email'
      value={formData.email}
      onChange={changeHandler}
      type="email"
      placeholder='Email address'
    />
    <input
      name='password'
      value={formData.password}
      onChange={changeHandler}
      type={showPassword ? 'text' : 'password'}
      placeholder='Password'
    />
  </div>
  <div className="loginsignup-show-password">
    <input
      className='loginsignup-show-password-checkbox'
      type="checkbox"
      onChange={handlePasswordCheckboxChange}
    />
    <p>Show password</p>
  </div>
  <button onClick={() => state === "Login" ? login() : signup()}>
```

```
          Continue
        </button>
        <p className="loginsignup-login">
          {state === "Sign up"
            ? "Already have an account? "
            : "Create an account? "
          }
          <span onClick={() => setState(state === "Login" ? "Sign up" : "Login")}>
            {state === "Sign up" ? "Login here" : "Click here"}
          </span>
        </p>
        <div className="loginsignup-agree">
          <input
            className='loginsignup-agree-checkbox'
            type="checkbox"
            onChange={handleTermsCheckboxChange}
          />
          <p>By continuing, I agree to the terms of use & privacy policy.</p>
        </div>
      </div>
    </div>
  );
};
export default LoginSignup;
```

33. Cart.jsx

```
import React from 'react'
import Cartitems from '../Components/Cartitems/Cartitems'
const Cart = () => {
```

```
  return (
    <div>
      <Cartitems/>
    </div>
  )
}
export default Cart
```

34. shopcategory.css

```
.shopcategory-banner {
  display: block;
  margin: 30px auto;
  width: 85%;
}
.shop-category {
  padding: 20px;
  max-width: 1200px;
  margin: 0 auto;
}
.all-products-sort {
  display: flex;
  justify-content: center;
  margin-bottom: 20px;
  gap: 20px;
}
.all-products-sort span {
  cursor: pointer;
  padding: 10px;
  background-color: #708174;
```

```
  color: white;
  border-radius: 5px;
  transition: background-color 0.3s ease;
}
.all-products-sort span:hover {
  background-color: #7d8d82;
}
.shopcategory-container {
  display: flex;
  flex-direction: row;
}
.shopcategory-sidebar {
  flex: 0 0 200px; /* Sidebar width */
  padding: 20px;
  background-color: #faf9f6;
  box-shadow: 2px 0 5px rgba(0, 0, 0, 0.1);
}
.shopcategory-sidebar h3 {
  margin-bottom: 10px;
  font-size: 22px;
}
.shopcategory-sort {
  margin-bottom: 10px;
  cursor: pointer;
  padding: 10px;
  background-color: #708174;
  color: white;
  border-radius: 5px;
  transition: background-color 0.3s ease;
}
```

```
.shopcategory-sort:hover {
  background-color: #7d8d82;
}
.shopcategory-content {
  flex: 1;}
.shopcategory-products {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  justify-content: center;
}
.shopcategory-products .item {
  flex: 1 1 calc(25% - 20px); /* Adjusted to ensure 4 items per row */
  max-width: calc(25% - 20px); /* Adjusted to ensure 4 items per row */
}
.shopcategory-indexsort {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
  font-size: 18px;
}
.all-products-loadmore {
  cursor: pointer;
  text-align: center;
  padding: 10px;
  background-color: #5d6357;
  color: white;
  border-radius: 5px;
  margin-top: 20px;
```

```
  width: 500px;
  align-items: center;
  display: flex;
  justify-content: center;
  margin-left: auto;
  margin-right: auto;
  font-size: 16px;
}
.all-products-loadmore:hover {
  background-color: #80867b;
}
```

35. loginsignup.css

```
.loginsignup{
  width: 100%;
  height: auto;
  padding-top: 40px;
  padding-bottom: 200px;
background: #faf9f6;
}
.loginsignup-container{
  width: 600px;
  height: auto;
background: #ffffff;
  border: 0.5px solid rgb(48, 48, 48);
  margin: auto;
  padding: 40px 60px;
}
.loginsignup h1{
```

```
    margin: 20px 0px;
    justify-content: center;
    display: flex;
    color: rgb(48, 48, 48);
    font-size: 32px;
    font-weight: 300;
}
.loginsignup-fields{
    display: flex;
    flex-direction: column;
    gap: 29px;
    margin-top: 30px;
}
.loginsignup-show-password{
  display: flex;
  align-items: center;
  margin-left: 10px;
  margin-top: 15px;
  gap: 10px;
  color: rgb(48, 48, 48);
  font-size: 18px;
  font-weight: 500;
}
.loginsignup-fields input{
  height: 50px;
  width: auto;
  padding-left: 20px;
  border: 1px solid #888;
  outline: none;
  color: rgb(48, 48, 48);
```

```
  font-size: 18px;
}
.loginsignup-container button{
  width: 600px;
  height: 50px;
  color: rgb(255, 255, 255);
  background: #395748;
  margin-top: 30px;
  border: none;
  font-size: 24px;
  font-weight: 300;
  cursor: pointer;
}
.loginsignup-login {
  margin-top: 20px;
  color: rgb(48, 48, 48);
  font-size: 18px;
  font-weight: 500;
  display: flex;
justify-content: center;
gap: 5px;
}
.loginsignup-login span{
  color: rgb(223, 102, 64);
  font-weight: 600;
  cursor: pointer;
}
.loginsignup-agree{
  display: flex;
  align-items: center;
```

```
  margin-top: 25px;
  gap: 20px;
  color: rgb(48, 48, 48);
  font-size: 18px;
  font-weight: 500;
  margin-left: 10px;
}
@media (max-width:1280px) {
  .loginsignup{
    padding-top: 50px;
  }
  .loginsignup-container{
    width: auto;
    max-width: 500px;
    max-height: 550px;
  }
  .loginsignup-container h1{
    margin: 10px 0px;
  }
  .loginsignup-fields{
    gap: 20px;
    margin-top: 20px;
  }
  .loginsignup-fields input{
    height: 65px;
    width: 93%;
  }
  .loginsignup-container button{
    width: 97%;
  }
```

```
  .loginsignup-login{
    font-size: 16px;
  }
  .loginsignup-show-password{
    font-size: 16px;
  }
  .loginsignup-agree{
    font-size: 16px;
    gap: 10px;
  }
}
@media (max-width:800px) {
  .loginsignup-container{
   padding: 20px 30px;
   max-height: 500px;
  }
  .loginsignup-container h1{
   font-size: 22px;
  }
  .loginsignup-fields{
    margin-top: 20px;
  }
  .loginsignup-fields input{
    height: 50px;
    width: 93%;
    font-size: 14px;
  }
  .loginsignup-container button{
    width: 97%;
    height: 50px;
```

```
    font-size: 14px;
  }
  .loginsignup-login{
    font-size: 14px;
  }
  .loginsignup-show-password{
    font-size: 14px;
  }
  .loginsignup-agree{
    font-size: 14px;
    gap: 10px;
  }}
@media (max-width:390px) {
  .loginsignup{
    height: 50vh;}
  .loginsignup{
    padding-top: 0%;
  }}
```

36. App.js

```
import './App.css';
import Navbar from './Components/Navbar/Navbar';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Shop from './pages/Shop';
import ShopCategory from './pages/ShopCategory';
import Product from './pages/Product';
import AllProducts from './Components/AllProducts/AllProducts';
import Cart from './pages/Cart';
import LoginSignup from './pages/LoginSignup';
```

```
import Footer from './Components/footer/footer';
import men_banner from './Components/assets/banner_mens.png';
import women_banner from './Components/assets/banner_women.png';
import kid_banner from './Components/assets/banner_kids.png';
function App() {
 return (
  <div>
   <BrowserRouter>
   <Navbar/>
   <Routes>
   <Route path='/' element={<Shop/>}/>
   <Route path='/womens' element={<ShopCategory banner={women_banner} category="women"/>}/>
   <Route path='/mens' element={<ShopCategory banner={men_banner} category="men"/>}/>
   <Route path='/kids' element={<ShopCategory banner={kid_banner} category="kid"/>}/>
   <Route path="/all-products" element={<AllProducts />} />
   <Route path="/product" element={<Product/>}>
   <Route path=':productId' element={<Product/>}/>
   </Route>
   <Route path='/cart' element={<Cart/>}/>
   <Route path='/login' element={<LoginSignup/>}/>

   </Routes>
   <Footer/>
   </BrowserRouter>
  </div>
 );
}
```

```
export default App;
```

37. App.css

```
.App {
  text-align: center;
}
.App-logo {
  height: 40vmin;
  pointer-events: none;
}
@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}
.App-link {
  color: #61dafb;
}
@keyframes App-logo-spin {
```

```
 from {
   transform: rotate(0deg);
 }
 to {
   transform: rotate(360deg);
 }}
```

38. index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import ShopContextProvider from './context/ShopContext';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
  <ShopContextProvider>
    <App />
  </ShopContextProvider>
</React.StrictMode>
);
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

39. index.css

```
*{
  margin: 0;
    padding: 0;
    border: none;
}
body {
  margin: 0;
    padding: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
     'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
     sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  font-family: 'Montserrat';
}
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

2. Admin panel

40. main.jsx

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

```
import './index.css';
import { BrowserRouter } from "react-router-dom";
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>,
);
```

41. index.css

```
@import
url(https://fonts.googleapis.com/css?family=Teachers:regular,500,600,700,800,italic,500italic,600italic,700italic,800italic);
body{
  margin: 0;
  font-family: "Teachers";
  background-color: rgb(208, 216, 212);
  height: 100vh;}
```

42. App.jsx

```
import React from 'react'
import Navbar from './Components/Navbar/Navbar'
import Admin from './Pages/Admin/Admin'
import LoginSignup from './Pages/LoginSignup'
const App = () => {
  return (
    <div>
```

```
      <Navbar/>
      <Admin/>
    </div>
)}
export default App
```

43. LoginSignup.jsx

```
import React, { useState } from 'react';
import './LoginSignup.css';
const LoginSignup = () => {
  const [showPassword, setShowPassword] = useState(false);
  const [agreeTerms, setAgreeTerms] = useState(false);
  const [state, setState] = useState("Login");
  const [formData, setFormData] = useState({
    username: "",
    password: "",
    email: "",
    phone: "" // Add this line
  });
  const changeHandler = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };
  const handlePasswordCheckboxChange = () => {
    setShowPassword(!showPassword);
  };

  const handleTermsCheckboxChange = () => {
    setAgreeTerms(!agreeTerms);
  };
```

```
const login = async () => {
  try {
    console.log("Login function executed", formData);
    const response = await fetch('http://localhost:4000/login', { // Use the correct
IP address or localhost
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(formData)
    });
    if (response.ok) {
      const responseData = await response.json();
      if (responseData.success) {
        localStorage.setItem('auth-token', responseData.token);
        window.location.replace("/");
      } else {
        throw new Error(responseData.error || "Login failed");
      }
    } else {
      const errorData = await response.json();
      throw new Error(errorData.error || 'Network response was not ok');
    }
  } catch (error) {
    console.error("Login failed", error.message);
    alert(`Login failed: ${error.message}`); // Display error to the user
  }
};
const signup = async () => {
  try {
```

```
    console.log("Sign up function executed", formData);
    const response = await fetch('http://localhost:4000/signup', {
      method: 'POST',
      headers: {
        'Content-Type':'application/json'
      },
      body: JSON.stringify(formData)
    });
    if (response.ok) {
      const responseData = await response.json();
      if (responseData.success) {
        localStorage.setItem('auth-token', responseData.token);
        window.location.replace("/");
      } else {
        throw new Error(responseData.error || "Signup failed");
      }
    } else {
      throw new Error('Network response was not ok');
    }
  } catch (error) {
    console.error("Signup failed", error.message);
    alert("Signup failed. Please try again."); // Display error to the user
  }
};
return (
  <div className='loginsignup'>
    <div className="loginsignup-container">
      <h1>{state}</h1>
      <div className="loginsignup-fields">
        {state === "Sign up" &&
```

```
  <input
    name='username'
    value={formData.username}
    onChange={changeHandler}
    type="text"
    placeholder='Your name'
  />
}
{state === "Sign up" &&
  <input
    name='phone'
    value={formData.phone}
    onChange={changeHandler}
    type="tel"
    placeholder='Phone number'
  />
}
<input
  name='email'
  value={formData.email}
  onChange={changeHandler}
  type="email"
  placeholder='Email address'
/>
<input
  name='password'
  value={formData.password}
  onChange={changeHandler}
  type={showPassword ? 'text' : 'password'}
  placeholder='Password'
```

```
  />
</div>
<div className="loginsignup-show-password">
  <input
    className='loginsignup-show-password-checkbox'
    type="checkbox"
    onChange={handlePasswordCheckboxChange}
  />
  <p>Show password</p>
</div>
<button onClick={() => state === "Login" ? login() : signup()}>
  Continue
</button>
<p className="loginsignup-login">
  {state === "Sign up"
    ? "Already have an account? "
    : "Create an account? "
  }
  <span onClick={() => setState(state === "Login" ? "Sign up" : "Login")}>
    {state === "Sign up" ? "Login here" : "Click here"}
  </span>
</p>
<div className="loginsignup-agree">
  <input
    className='loginsignup-agree-checkbox'
    type="checkbox"
    onChange={handleTermsCheckboxChange}
  />
  <p>By continuing, I agree to the terms of use & privacy policy.</p>
</div>
```

```
      </div>
    </div>
  );
};
export default LoginSignup;
```

44. LoginSignup.css

```
.loginsignup{
  width: 100%;
  height: 80vh;
  padding-top: 20px;
  padding-bottom: 200px;
background: #faf9f6;
  /*background: url('/src/Components/login_back3.png') center center/cover;*/
}
.loginsignup-container{
  width: 600px;
  height: auto;
background: #ffffff;
  border: 0.5px solid rgb(48, 48, 48);
  margin: auto;
  padding: 40px 60px;
}
.loginsignup h1{
    margin: 20px 0px;
    justify-content: center;
    display: flex;
    color: rgb(48, 48, 48);
}
```

```
.loginsignup-fields{
    display: flex;
    flex-direction: column;
    gap: 29px;
    margin-top: 30px;
}
.loginsignup-show-password{
  display: flex;
  align-items: center;
  margin-left: 10px;
  margin-top: 15px;
  gap: 10px;
  color: rgb(48, 48, 48);
  font-size: 18px;
  font-weight: 500;
}
.loginsignup-fields input{
  height: 52px;
  width: auto;
  padding-left: 20px;
  border: 1px solid #888;
  outline: none;
  color: rgb(48, 48, 48);
  font-size: 18px;
}
.loginsignup-container button{
  width: 600px;
  height: 62px;
  color: rgb(255, 255, 255);
```

```
  background: #395748;
  margin-top: 30px;
  border: none;
  font-size: 24px;
  font-weight: 500;
  cursor: pointer;
}
.loginsignup-login {
  margin-top: 20px;
  color: rgb(48, 48, 48);
  font-size: 18px;
  font-weight: 500;
  display: flex;
justify-content: center;
gap: 5px;
}
.loginsignup-login span{
  color: rgb(223, 102, 64);
  font-weight: 600;
  cursor: pointer;
}
.loginsignup-agree{
  display: flex;
  align-items: center;
  margin-top: 25px;
  gap: 20px;
  color: rgb(48, 48, 48);
  font-size: 18px;
  font-weight: 500;
  margin-left: 10px;
```

```
}
@media (max-width:1280px) {
  .loginsignup{
    padding-top: 50px;
  }
  .loginsignup-container{
    width: auto;
    max-width: 500px;
    max-height: 550px;
  }
  .loginsignup-container h1{
    margin: 10px 0px;
  }
  .loginsignup-fields{
    gap: 20px;
    margin-top: 20px;
  }
  .loginsignup-fields input{
    height: 65px;
    width: 93%;
  }
  .loginsignup-container button{
    width: 97%;
  }
  .loginsignup-login{
    font-size: 16px;
  }
  .loginsignup-show-password{
    font-size: 16px;
  }
```

```
  .loginsignup-agree{
    font-size: 16px;
    gap: 10px;
  }
}
@media (max-width:800px) {
  .loginsignup-container{
   padding: 20px 30px;
   max-height: 500px;
  }
  .loginsignup-container h1{
   font-size: 22px;
  }
  .loginsignup-fields{
    margin-top: 20px;
  }
  .loginsignup-fields input{
    height: 50px;
    width: 93%;
    font-size: 14px;
  }
  .loginsignup-container button{
    width: 97%;
    height: 50px;
    font-size: 14px;
  }
  .loginsignup-login{
    font-size: 14px;
  }
  .loginsignup-show-password{
```

```
    font-size: 14px;
  }
  .loginsignup-agree{
    font-size: 14px;
    gap: 10px;
  }
}
@media (max-width:390px) {
  .loginsignup{
    height: 50vh;
  }
  .loginsignup{
    padding-top: 0%;
  }}
```

45. Admin.jsx

```
import React from 'react';
import './Admin.css';
import Sidebar from '../../Components/Sidebar/Sidebar';
import { Routes, Route, Navigate } from 'react-router-dom';
import AddProduct from '../../Components/AddProduct/AddProduct';
import ListProduct from '../../Components/ListProduct/ListProduct';
import Orders from '../../Components/Orders/Orders';
import LoginSignup from '../LoginSignup';
import Newsletter from '../../../../frontend/src/Components/Newsletter/Newsletter';
const Admin = () => {
  return (
    <div className='admin'>
      <Sidebar />
```

```
      <div className='admin-content'>
        <Routes>
          <Route path="/addproduct" element={<AddProduct />} />
          <Route path="/listproduct" element={<ListProduct />} />
          <Route path="/orders" element={<Orders />} />
          <Route path='/login' element={<LoginSignup/>}/>
          <Route path="*" element={<Navigate to="/admin/addproduct" />} />
        </Routes>
      </div>
    </div>
  );
};
export default Admin;
```

46. Admin.css

```
.admin {
  display: flex;
}
.admin-content {
  flex: 1;
  padding: 20px;
  background-color: #f9f9f9;
  height: 100vh;
  overflow-y: auto;
}
@media (max-width: 800px) {
  .admin{
    flex-direction:column;
  }
```

```
}
```

47. AddProduct.css

```
.add-product{
  box-sizing: border-box;
  width: 100%;
  max-width: 800px;
  padding: 30px 50px;
  margin: 20px 30px;
  border-radius: 6px;
  background: whitesmoke;
}
.add-product-item-field p{
  font-family: "Teachers";
}
.add-product-item-field{
  width: 100%;
  color: rgb(48, 48, 48);
  font-size: 16px;
}
.add-product-item-field input{
  box-sizing: border-box;
  width: 100%;
  height: 50px;
  border-radius: 4px;
  padding-left: 15px;
  border: 1px solid #333;
  outline: none;
  color: #303030;
  font-family: 'Teachers';
```

```
  font-size: 14px;
}
.add-product-price{
  display: flex;
  gap: 40px;
}
.add-product-selector{
  padding: 10px;
  width: 100px;
  height: 50px;
  font-size: 14px;
  color: #303030;
  border: 1px solid #333;
  border-radius: 4px;
}
.add-product-thumnail-img{
  height: 120px;
  width: 120px;
  border-radius: 10px;
  object-fit: contain;
  margin: 10px 0px;
}
.add-poduct-btn{
  margin-top: 20px;
  width: 160px;
  height: 50px;
  border-radius: 6px;
  background: rgb(208, 216, 212);
  border: none;
  cursor: pointer;
```

```
  color: #303030;
  font-weight: 600;
  font-size: 16px;
}
@media (max-width: 800px) {
  .add-product{
    width: auto;
    padding: 30px;
    margin: 20px;
  }}
```

48. AddProduct.jsx

```
import React, { useState } from 'react'
import './AddProduct.css'
import upload_area from '../../assets/upload_area.svg'
const AddProduct = () => {
const [image, setImage] = useState(false);
const [productDetails, setProductDetails] = useState({
  name:"",
  image:"",
  category:"women",
  new_price:"",
  old_price:""
})
      const imageHandler = (e) =>{
            setImage(e.target.files[0]);
      }
      const changeHandler = (e) =>{
        setProductDetails({...productDetails,[e.target.name]:e.target.value})
```

```
}
const Add_Product = async ()=>{
  console.log(productDetails);
  let responseData;
  let product = productDetails;
  let formData = new FormData();
  formData.append('product',image);
  await fetch('http://localhost:4000/upload',{
    method: 'POST',
    headers:{
      Accept:'application/json',
    },
    body:formData,
  }).then((resp) => resp.json()).then((data)=>{responseData=data});
  if(responseData.success)
  {
    product.image = responseData.image_url;
    console.log(product);
    await fetch('http://localhost:4000/addproduct',{
      method:'POST',
      headers:{
        Accept: 'application/json',
        'Content-Type': 'application/json',
      },
      body:JSON.stringify(product),
    }).then((resp)=>resp.json()).then((data)=>{
      data.success?alert("Product Added"):alert("Failed")
    })
  }
}
```

```
        return (
          <div className='add-product'>
            <div className="add-product-item-field">
              <p>Product Title</p>
              <input value={productDetails.name} onChange={changeHandler}
type="text" name='name' placeholder='Type here' />
            </div>
            <div className="add-product-price">
              <div className="add-product-item-field">
                <p>Price</p>
                <input value={productDetails.old_price} onChange={changeHandler}
type="text" name='old_price' placeholder='Type here'/>
              </div>

              <div className="add-product-item-field">
                <p>Offer price</p>
                <input value={productDetails.new_price} onChange={changeHandler}
type="text" name='new_price' placeholder='Type here'/>
              </div>
            </div>
            <div className="add-product-item-field">
              <p>Product Category</p>
            <select value={productDetails.category} onChange={changeHandler}
name="category" className='add-product-selector'>
              <option value="women">Women</option>
              <option value="men">Men</option>
              <option value="kid">Kid</option>
            </select>
            </div>
            <div className="add-product-item-field">
```

```
            <label htmlFor="file-input">
              <img                src={image?URL.createObjectURL(image):upload_area}
className='add-product-thumnail-img' alt="" />
            </label>
            <input onChange={imageHandler} type="file" name='image' id='file-input'
hidden />
          </div>
          <button        onClick={()=>{Add_Product()}}        className='add-poduct-
btn'>Add</button>
        </div>
      )
    }
    export default AddProduct
```

49. ListProduct.css

```
.listproduct{
  display: flex;
  flex-direction: column;
  align-items: center;
  width: 100%;
  height: 740px;
  padding: 10px 50px;
  margin: 30px;
  border-radius: 6px;
  background: whitesmoke;
}
.list-product-format-main{
  display: grid;
  grid-template-columns: 1fr 3fr 1fr 1fr 1fr 1fr;
```

```
  gap: 10px;
  width: 100%;
  padding: 20px 0px;
  color: #303030;
  font-size: 15px;
  font-weight: 600;
}
.list-product-format{
  align-items: center;
  font-weight: 500;
}
.list-product-product-icon{
  height: 80px;
}
.list-product-remove-icon{
  cursor: pointer;
  margin: auto;
}
.list-product-all-products{
  overflow-y: auto;
}
@media (max-width: 800px) {
  .listproduct{
    box-sizing: border-box;
    width: 90%;
    height: 100%;
    padding: 10px 30px;
    margin: 20px auto;
  }
  .list-product-format-main{
```

```
  padding: 15px 0px;
  color: #303030;
  font-size: 12px;
 }
 .list-product-product-icon{
  height: 60px;
 }}
```

50. ListProduct.jsx

```
import React, { useState, useEffect } from 'react';
import './ListProduct.css';
import cross_icon from '../../assets/cross_icon.png';
const ListProduct = () => {
  const [allproducts, setAllProducts] = useState([]);
  const fetchInfo = async () => {
  await fetch('http://localhost:4000/allproducts')
  .then((res)=>res.json())
  .then((data)=>{setAllProducts(data)});
  }
  useEffect(() => {
   fetchInfo();
  }, []); // Empty dependency array to run the effect only once on component mount
  // Function to remove a product
  const remove_product = async (id) => {
     await fetch('http://localhost:4000/removeproduct', {
       method: 'POST',
       headers: {
         Accepte:'application/json',
```

```
            'Content-Type':'application/json'
           },
           body:JSON.stringify({id:id})
          })
          await  fetchInfo();
         }
        return (
         <div className='listproduct'>
          <h1>All products list</h1>
          <div className="list-product-format-main">
           <p>Products</p>
           <p>Title</p>
           <p>Old price</p>
           <p>New price</p>
           <p>Category</p>
           <p>Remove</p>
          </div>
          <div className="list-product-all-products">
           {allproducts.map((product,index) => {
            return <div key={index} className="list-product-format-main list-product-format">
             <img src={product.image} alt="" className="list-product-product-icon" />
             <p>{product.name}</p>
             <p>${product.old_price}</p>
             <p>${product.new_price}</p>
             <p>{product.category}</p>
             <img onClick={()=>{remove_product(product.id)}} className='list-product-remove-icon' src={cross_icon} alt="" />
            </div>
```

```
      })}
    </div>
  </div>
 );
};
export default ListProduct;
```

51. Navbar.jsx

```
import React from 'react';
import { Link, useNavigate } from 'react-router-dom';
import './Navbar.css';
import navlogo from '../../assets/nav-logo.svg';
import navProfile from '../../assets/nav-profile.svg';
const Navbar = () => {
 const isAuthenticated = localStorage.getItem('auth-token');
 const navigate = useNavigate();
 const handleLogout = () => {
  localStorage.removeItem('auth-token');
  navigate('/'); // Navigate to home or login page after logout
 };
 return (
  <div className='navbar'>
   <Link to="/">
    <img src={navlogo} className="nav-logo" alt="" />
   </Link>
   <div className='nav-links'>
   {localStorage.getItem('auth-token')
```

```
            ?<button                    onClick={()=>{localStorage.removeItem('auth-
token');window.location.replace('/')}}>Вийти</button>:
              <Link to='/login'><button>Вхід</button></Link>}
            </div>
          </div>
        );
      };
      export default Navbar;
```

52. Navbar.css

```
.navbar{
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 15px 60px;
  box-shadow: 0 1px 3px -2px #000;
  margin-bottom: 1px;
  background: whitesmoke;
}
.nav-logo {
  width: 180px;
  height: auto;
}
.nav-links {
  display: flex;
  align-items: center;
}
.nav-links a, .nav-links button {
  margin-left: 15px;
```

```
  text-decoration: none;
  color: #333;
  font-weight: bold;
  cursor: pointer;
}
.nav-links button {
  background: none;
  border: none;
  color: #333;
  font-weight: bold;
  cursor: pointer;
  font-size: 22px;
  font-weight: 500;
}
.nav-profile {
  height: 40px;
  margin-left: 20px;
  cursor: pointer;
}
@media (max-width: 800px) {
  .navbar{
    padding: 15px 30px;
  }
  .nav-logo{
    width: 150px;
  }
  .nav-profile{
    width: 75px;
  }
}
```

```
@media (max-width: 390px) {
  .navbar{
    padding: 10px 20px;
  }
  .nav-logo{
    width: 130px;
  }
  .nav-profile{
    width: 75px;
  }
}
```

53. Orders.jsx

```
import React, { useState, useEffect } from 'react';
import './Orders.css';
const Orders = () => {
  const [orders, setOrders] = useState([]);
  useEffect(() => {
    const fetchOrders = async () => {
      try {
        const token = localStorage.getItem('auth-token');
        if (!token) {
          console.error('No token found in local storage');
          return;
        }
        const response = await fetch('http://localhost:4000/orders', {
          method: 'GET',
          headers: {
            'Content-Type': 'application/json',
```

```
          'auth-token': token
        }
      });
      if (response.status === 401) {
        console.error('Authentication error');
        return;
      }
      const data = await response.json();
      console.log('Fetched orders:', data); // Log the fetched orders
      setOrders(data);
    } catch (error) {
      console.error('Error fetching orders', error);
    }
  };
  fetchOrders();
}, []);
const deleteOrder = async (orderId) => {
  try {
    const response = await fetch(`http://localhost:4000/deleteOrder/${orderId}`, {
      method: 'DELETE',
      headers: {
        'auth-token': localStorage.getItem('auth-token')
      }
    });
    if (response.ok) {
      setOrders(orders.filter(order => order._id !== orderId));
      alert('Order deleted successfully');
    } else {
      const errorData = await response.json();
      throw new Error(errorData.error || 'Failed to delete order');
```

```
      }
    } catch (error) {
      console.error('Delete order failed', error.message);
      alert(`Delete order failed: ${error.message}`);
    }
  };
  return (
    <div className='orders'>
      <h1>Orders</h1>
      {orders.length > 0 ? (
        <table className="orders-table">
          <thead>
            <tr>
              <th>Order ID</th>
              <th>User</th>
              <th>Items</th>
              <th>Total Amount</th>
              <th>Date</th>
            </tr>
          </thead>
          <tbody>
            {orders.map(order => (
              <tr key={order._id}>
                <td>{order._id}</td>
                <td>{order.userId?.name}                ({order.userId?.email})
({order.userId.phone})</td>
                <td>
                  {order.items.map(item => (
                    <div key={item.productId._id}>
                      {item.productId.name} - {item.quantity} pcs
```

```
                </div>
              ))}
            </td>
            <td>${order.totalAmount}</td>
            <td>{new Date(order.createdAt).toLocaleString()}</td>

          </tr>
        ))}
      </tbody>
    </table>
  ) : (
    <p>No orders found</p>
  )}
 </div>
 );
};
export default Orders;
```

54. Orders.css

```
.orders {
  padding: 20px;
  max-width: 1200px;
  margin: 0 auto;
}
.orders h1 {
  text-align: center;
  margin-bottom: 20px;
}
.orders-table {
```

```
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
}
.orders-table th, .orders-table td {
  border: 1px solid #ddd;
  padding: 8px;
  text-align: left;
}
.orders-table th {
  background-color: #708174;
  color: white;
}
.orders-table td div {
  margin-bottom: 5px;
}
```

55. Sidebar.jsx

```
import React from 'react';
import { Link } from 'react-router-dom';
import './Sidebar.css';
const Sidebar = () => {
  return (
    <div className='sidebar'>
      <h2>Admin Panel</h2>
      <ul>
        <li>
          <Link to="../addproduct">Add Product</Link>
        </li>
```

```
        <li>
          <Link to="../listproduct">List Products</Link>
        </li>
        <li>
          <Link to="../orders">Orders</Link>
        </li>
      </ul>
    </div>
  );
};
export default Sidebar;
```

56. Sidebar.css

```
.sidebar {
  width: 250px;
  height: auto;
  background-color: #708174;
  color: white;
  padding: 20px;
  box-sizing: border-box;
}
.sidebar h2 {
  margin-bottom: 20px;
  text-align: center;

}
.sidebar ul {
  list-style: none;
  padding: 0;
```

```
}
.sidebar ul li {
  margin-bottom: 10px;
  font-size: 20px;
}
.sidebar ul li a {
  color: white;
  text-decoration: none;
  padding: 10px;
  display: block;
  border-radius: 5px;
  transition: background-color 0.3s ease;
}
.sidebar ul li a:hover {
  background-color: #7d8d82;
}
```

3. Backend

57. index.js

```
const express = require('express');
const mongoose = require('mongoose');
const multer = require('multer');
const path = require('path');
const cors = require('cors');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');
const { log, assert } = require('console');
const port = 4000;
```

```
const app = express();
app.use(express.json());
app.use(cors());
// Database connection with MongoDB
mongoose.connect("mongodb+srv://ttnkchnk:lY2drqAzSTWNTg1S@cluster0.v7toi1y.mongodb.net/inside-out", { useNewUrlParser: true, useUnifiedTopology: true });
// Middleware to fetch user
const authMiddleware = (req, res, next) => {
  const token = req.header('auth-token');
  if (!token) {
    return res.status(401).send({ errors: "Please authenticate using a valid token" });
  }
  try {
    const data = jwt.verify(token, 'secret_ecom');
    req.user = data.user;
    next();
  } catch (error) {
    res.status(401).send({ errors: "Please authenticate using a valid token" });
  }
};
// API creation
app.get("/", (req, res) => {
  res.send("Express App is Running");
});
// Image storage engine
const storage = multer.diskStorage({
  destination: './upload/image',
  filename: (req, file, cb) => {
```

```
    return cb(null,
`${file.fieldname}_${Date.now()}${path.extname(file.originalname)}`);
    }
  });
  const upload = multer({ storage: storage });
  // Creating upload endpoint for images
  app.use('/image', express.static(path.join(__dirname, 'upload/image')));
  app.post("/upload", upload.single('product'), (req, res) => {
    res.json({
      success: 1,
      image_url: `http://localhost:${port}/image/${req.file.filename}`
    });
  });
  // Schema for creating products
  const productSchema = new mongoose.Schema({
    id: {
      type: Number,
      required: true,
    },
    name: {
      type: String,
      required: true
    },
    image: {
      type: String,
      required: true,
    },
    category: {
      type: String,
      required: true,
```

```
  },
  new_price: {
    type: Number,
    required: true,
  },
  old_price: {
    type: Number,
    required: true,
  },
  date: {
    type: Date,
    default: Date.now,
  },
  available: {
    type: Boolean,
    default: true,
  },
});
const Product = mongoose.model("Product", productSchema);
// Schema for creating users
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, unique: true, required: true },
  password: { type: String, required: true },
  phone: { type: String, required: true },
  cartData: { type: Object },
  date: { type: Date, default: Date.now }
});
const Users = mongoose.model('Users', userSchema);
// API for adding products
```

```
app.post('/addproduct', async (req, res) => {
  let products = await Product.find({});
  let id;
  if (products.length > 0) {
    let last_product_array = products.slice(-1);
    let last_product = last_product_array[0];
    id = last_product.id + 1;
  } else {
    id = 1;
  }
  const product = new Product({
    id: id,
    name: req.body.name,
    image: req.body.image,
    category: req.body.category,
    new_price: req.body.new_price,
    old_price: req.body.old_price,
  });
  console.log(product);
  await product.save();
  console.log("Saved:");
  res.json({
    success: true,
    name: req.body.name,
  });
});
// API for deleting products
app.post('/removeproduct', async (req, res) => {
  await Product.findOneAndDelete({ id: req.body.id });
  console.log("Removed");
```

```
    res.json({
      success: true,
      name: req.body.name
    });
  });
  // API for getting all products
  app.get('/allproducts', async (req, res) => {
    let products = await Product.find({});
    console.log("All Products Fetched");
    res.send(products);
  });
  // API for registering user
  app.post('/signup', async (req, res) => {
    const { username, email, password, phone } = req.body;
    console.log('Received signup request:', req.body);
    try {
      let check = await Users.findOne({ email });
      if (check) {
        console.log('User with this email already exists:', email);
        return res.status(400).json({ success: false, error: "User with this email already exists" });
      }
      const salt = await bcrypt.genSalt(10);
      const hashedPassword = await bcrypt.hash(password, salt);
      let cart = {};
      for (let i = 0; i < 300; i++) {
        cart[i] = 0;
      }
      const user = new Users({
        name: username,
```

```
            email,
            password: hashedPassword,
            phone,
            cartData: cart,
          });
          await user.save();
          const data = {
            user: {
              id: user.id
            }
          };
          const token = jwt.sign(data, 'secret_ecom');
          console.log('User registered successfully:', user);
          res.json({ success: true, token });
        } catch (error) {
          console.error('Signup error:', error);
          res.status(500).json({ success: false, error: "Signup failed due to server error"
});
        }
      });
      // API for user login
      app.post('/login', async (req, res) => {
        const { email, password } = req.body;
        try {
          let user = await Users.findOne({ email });
          if (!user) {
            return res.status(400).json({ success: false, error: "Invalid email or password"
});
          }
          const passCompare = await bcrypt.compare(password, user.password);
```

```
        if (!passCompare) {
          return res.status(400).json({ success: false, error: "Invalid email or password"
});
        }
        const data = {
          user: {
            id: user.id
          }
        };
        const token = jwt.sign(data, 'secret_ecom');
        res.json({ success: true, token });
      } catch (error) {
        console.error('Login error:', error);
        res.status(500).json({ success: false, error: "Login failed due to server error"
});
      }
    });
    // API for fetching new collection
    app.get('/newcollections', async (req, res) => {
      let products = await Product.find({});
      let newcollection = products.slice(1).slice(-8);
      console.log("New collection fetched");
      res.send(newcollection);
    });
    // API for fetching popular items in women section
    app.get('/popularinwomen', async (req, res) => {
      let products = await Product.find({ category: "women" });
      let popular_in_women = products.slice(0, 4);
      console.log("Popular in women fetched");
      res.send(popular_in_women);
```

```
});
// API for adding products to cart
app.post('/addtocart', authMiddleware, async (req, res) => {
  console.log("added", req.body.itemId);
  let userData = await Users.findOne({ _id: req.user.id });
  userData.cartData[req.body.itemId] += 1;
  await Users.findOneAndUpdate({ _id: req.user.id }, { cartData: userData.cartData });
  res.send("Added");
});
// API for removing products from cart
app.post('/removefromcart', authMiddleware, async (req, res) => {
  console.log("removed", req.body.itemId);
  let userData = await Users.findOne({ _id: req.user.id });
  if (userData.cartData[req.body.itemId] > 0)
    userData.cartData[req.body.itemId] -= 1;
  await Users.findOneAndUpdate({ _id: req.user.id }, { cartData: userData.cartData });
  res.send("Removed");
});
// API for getting cart data
app.post('/getcart', authMiddleware, async (req, res) => {
  console.log("GetCart");
  let userData = await Users.findOne({ _id: req.user.id });
  res.json(userData.cartData);
});
// Newsletter
const emailSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true }
});
```

```
const Email = mongoose.model('Email', emailSchema);
app.post('/subscribe', async (req, res) => {
  const { email } = req.body;
  try {
    const newEmail = new Email({ email });
    await newEmail.save();
    res.status(200).send('Subscription successful');
  } catch (error) {
    console.error('Subscription error:', error);
    res.status(400).send('Error subscribing');
  }
});
// API for sorting products
app.get('/products', async (req, res) => {
  const sortField = req.query.sortField || 'name'; // Default sort by name
  const sortOrder = req.query.sortOrder === 'desc' ? -1 : 1; // Default sort order is
ascending
  try {
    const products = await Product.find({}).sort({ [sortField]: sortOrder });
    res.json(products);
  } catch (error) {
    console.error('Sorting error:', error);
    res.status(500).send('Error fetching products');
  }
});
// Schema for creating orders
const OrderSchema = new mongoose.Schema({
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Users',
```

```
    required: true
  },
  items: [
    {
      productId: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Product',
        required: true
      },
      quantity: {
        type: Number,
        required: true
      }
    }
  ],
  totalAmount: {
    type: Number,
    required: true
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});
const Order = mongoose.model('Order', OrderSchema);
// API for creating an order
app.post('/createOrder', authMiddleware, async (req, res) => {
  const { items, totalAmount } = req.body;
  const userId = req.user.id;
  try {
```

```
    const newOrder = new Order({
      userId,
      items,
      totalAmount
    });
    await newOrder.save();
    res.status(201).json({ message: 'Order created successfully' });
  } catch (error) {
    console.error('Order creation error:', error);
    res.status(500).json({ message: 'Failed to create order', error });
  }
});
// API for fetching all orders
app.get('/orders', authMiddleware, async (req, res) => {
  try {
    const orders = await Order.find({})
      .populate('userId', 'name email phone') // Include phone number
      .populate('items.productId', 'name image new_price');
    console.log('Fetched orders from DB:', orders); // Log the fetched orders
    res.json(orders);
  } catch (error) {
    console.error('Fetching orders error:', error);
    res.status(500).send('Error fetching orders');
  }
});
app.listen(port, (error) => {
  if (!error) {
    console.log("Server Running on Port " + port);
  } else {
    console.log("Error : " + error);
```

```
  }
});
```