ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Білецький Андрій Тарасович

УДК 004.378

Розробка веб-сайту з метою торгівлі побутовою технікою

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Білецький А.Т.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Білецький Андрій Тарасович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Розробка веб-сайту з метою торгівлі побутовою технікою

керівник роботи:
Ващишак Сергій Петрович, кандидат технічних наук, доцент

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: Python, Django, HTML, CSS, JavaScript

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
   1. Аналіз наявних аналогів
   2. Розробка прототипу сайту
   3. Реалізація функціоналу для особистого кабінету користувача

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд та аналіз існуючих аналогів	20.03.2024	Виконано
2.	Проектування прототипу сайту	27.03.2024	Виконано
3.	Розробка сайту	08.04.2024	Виконано
4.	Оформлення пояснювальної записки	13.05.2024	Виконано
5.	Оформлення графічного матеріалу та підготовка до захисту роботи	23.05.2024	Виконано

Студент _____ Білецький А.Т.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ващишак С.П,
(підпис) (прізвище та ініціали)

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

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
11	Головна сторінка веб сайту «Фокстрот»	26	Таблиця Користувачі
12	Контент веб сайту «Фокстрот»	27	Таблиця Продукти
13	Контент веб сайту «Фокстрот» на смартфоні	28	Таблиця Замовлення/Схема зв'язку таблиць БД
14	Головна сторінка веб сайту «Moyo.ua»	31	Сторінка реєстрації на веб-сайті
15	Опис товару на веб сайті «Moyo.ua»	32	Сторінка авторизації на веб-сайті
16	Головна сторінка веб сайту «Moyo.ua» на смартфоні	33	Сторінка параметрів акаунту
17	Головна сторінка веб сайту «Rozetka»	34	Каталоги товарів
18	Адаптивність сайту на різному розширенні екрану	35	Каталог холодильників
21	Структура веб-сайту	36	Додавання товару в кошик/Збережені товари
22	Вікно середовища Visual Studio Code		

АНОТАЦІЯ

Метою даної кваліфікаційної роботи є розробка веб сайту з метою продажу побутової техніки.

Об'єктом дослідження є розробка веб сайту для продажу побутової техніки та покращення деяких проблемних місць в сайтах аналогах.

В даній кваліфікаційній роботі було здійснено огляд та аналіз веб сайтів аналогів та проаналізовано сучасні вимоги до веб сайтів, було визначено основні переваги та недоліки аналогів.

Було обрано програмні засоби для реалізації системи, також було здійснено розроблення зручного інтерфейсу веб сайту, і зроблено адаптивність сайту під будь-який пристрій.

У першому розділі було проведено та представлено детальний аналіз що існує аналогами за методом аналізу ключового функціонала веб сайтів, а також складання таблиці переваг і недоліків. Також була здійснена постановка задачі.

У другому розділі було представлено вибір та обґрунтування програмних засобів реалізації веб сайт і розроблення структурної схеми сайту, також було обрано середовище розробки. Також було представлено вибір та обґрунтування програмних засобів для реалізації бази даних для веб сайту.

У третьому розділі було представлено розроблення інтерфейсу веб сайт, розробка гнучкої адаптивності веб сайт, було здійснено тестування виконаної роботи, а саме: функціоналу та адаптивності.

КЛЮЧОВІ СЛОВА: ВЕБ-САЙТ, ТЕСТУВАННЯ, РОЗРОБКА

SUMMARY

Objective of this qualification work is to develop a website for selling household appliances.

The object of research is the development of a website for selling household appliances and the improvement of certain problematic aspects in comparable websites.

In this qualification work, an overview and analysis of similar websites were conducted, and modern requirements for websites were analyzed. The main advantages and disadvantages of the comparable websites were identified.

Software tools for implementing the system were selected, and a user-friendly interface for the website was developed, including making the site adaptive for any device.

In the first section, a detailed analysis of existing comparable websites was conducted and presented using the method of key functionality analysis of the sites, as well as the compilation of a table of advantages and disadvantages. The task was also formulated.

In the second section, the selection and justification of the software tools for implementing the website were presented, along with the development of the site's structural scheme, and the development environment was selected. The selection and justification of the software tools for implementing the database for the website were also presented.

In the third section, the development of the website interface, the creation of a flexible adaptive website, and the testing of functionality and adaptability were presente.

KEY WORDS: WEBSITE, TESTING, DEVELOPMENT

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

QL - structured query language (мова структурованих запитів).

СУБД – Система управління базами даних.

БД – База даних.

JS – Java Script.

ПК – Персональний комп’ютер.

ВСТУП

Актуальність теми. Актуальність розробки веб-сайту для продажу побутової техніки зумовлена кількома важливими факторами, що відображають сучасні тенденції в економіці, технологіях та споживацькій поведінці. По-перше, цифровізація суспільства та поширення Інтернету значно змінили спосіб придбання товарів, включаючи побутову техніку. Сьогодні споживачі прагнуть зручності та швидкості, яку забезпечують онлайн-магазини, що дозволяє економити час і ресурси.

По-друге, зростання конкуренції на ринку побутової техніки вимагає від компаній інноваційних підходів до збуту продукції. Веб-сайт, як інтегрований маркетинговий інструмент, не лише розширює географію продажів, але й забезпечує постійний доступ до широкого асортименту товарів, інформації про них, а також відгуків інших покупців, що підвищує рівень довіри до бренду.

Крім того, сучасні веб-технології дозволяють забезпечити високу функціональність та інтерактивність веб-сайтів, що покращує користувацький досвід. Функції персоналізації, адаптивний дизайн, інтеграція з соціальними мережами та аналітичні інструменти сприяють більш точному розумінню потреб клієнтів та підвищенню ефективності продажів.

Також варто зазначити, що пандемія COVID-19 значно прискорила перехід до онлайн-торгівлі, що зробило наявність ефективного веб-сайту критично важливим для виживання та розвитку бізнесу в умовах нової реальності. Сучасний веб-сайт забезпечує можливість дистанційного обслуговування клієнтів, що відповідає вимогам безпеки та соціального дистанціювання.

Мета роботи. Основною метою кваліфікаційної роботи була розробка та реалізація веб-сайту для продажу побутової техніки, який відповідатиме сучасним вимогам користувачів і включатиме в себе передові технології та інноваційні підходи до управління контентом.

Об’єкт роботи. Сайту для продажу побутової техніки.

Предмет роботи. Розробка веб-сайту для продажу побутової техніки.

Завдання роботи. Відповідно до вибраної теми в роботі покладені такі з задачі як:

- пошук існуючих веб-сайтів з продажу техніки;
- вибір мови програмування та технологій розробки;
- розроблення зручного дизайну;
- проведення тестування продукту.

Методи роботи. Для вирішення поставленого завдання були використані мова програмування Python з фреймворком Django.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на І Всеукраїнській науково-практичній інтернет-конференції «ІТ ЕКОСИСТЕМА: Цифровізація бізнес-процесів в умовах війни», яка відбулась 23-24 листопада 2023 року в Університеті Короля Данила.

Результати роботи. Результатом роботи буде створений веб-сайт для продажу побутової техніки. Веб-сайт буде надавати інформацію користувачам про ціну техніки, її характеристики. Також буде можливість користувачу авторизуватись на веб-сайті та здійснити покупку.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ САЙТІВ АНАЛОГІВ

1.1 Сучасні вимоги для веб-сайту

Вимоги до структури веб-сайту змінюються в залежності від різних чинників, таких як цільова аудиторія, тип веб-сайту, його розмір і складність, а також цілі та завдання, які він має виконувати. Однак, є кілька загальних рекомендацій, які слід враховувати при створенні структури веб-сайту.

По-перше, навігація повинна бути чіткою та зрозумілою, щоб користувачі могли легко знаходити потрібну інформацію. Логічна та ієрархічна структура з чітко визначеними категоріями та під категоріями допоможе в цьому.

По-друге, вміст веб-сайту має бути якісним та цікавим, відповідати потребам користувачів і бути легко доступним та читабельним. Також важливо, щоб вміст містив ключові слова для покращення пошукової оптимізації (SEO).

По-третє, веб-сайт повинен бути адаптивним, тобто коректно відображатися на різних пристроях, таких як комп'ютери, планшети та смартфони. Інтерфейс користувача повинен бути інтуїтивно зрозумілим, щоб відвідувачі могли легко взаємодіяти з веб-сайтом.

Крім того, важливо забезпечити швидке завантаження веб-сайту, щоб уникнути втрати відвідувачів. Це можна досягти шляхом оптимізації вмісту, зменшення розміру зображень та інших елементів, а також використання технік кешування.

Нарешті, безпека веб-сайту є критично важливою. Веб-сайт повинен бути захищеним від шкідливих програм та хакерських атак. Це може бути забезпечено за допомогою захисних технологій, таких як SSL-шифрування, а також застосуванням надійних паролів та інших заходів безпеки. Регулярне оновлення програмного забезпечення та резервне копіювання даних також є важливими кроками для підтримки безпеки веб-сайту.

1.2 Веб Сайт «Фокстрот»

Аналіз функціональності веб-сайту «Фокстрот» [1], за певними критеріями, які були перераховані на початку розділу:

Сайт «Фокстроту» представлено на рисунку 1.1, пропонує широкий асортимент побутової техніки, зокрема телевізори, холодильники, пральні машини, смартфони та інші вироби відомих брендів. Функціонал надає користувачам можливість легко знаходити потрібні їм товари за допомогою інтуїтивно зрозумілого пошуку та фільтрації. Крім того, користувачі можуть у зручний спосіб додавати товари до кошику та оформлювати замовлення, використовуючи різні способи оплати та доставки.

Наявність розділу з акціями та знижками дозволяє клієнтам економити кошти при здійсненні покупок. Кожен користувач має можливість створити особистий кабінет, де він може переглядати історію своїх замовлень та виконувати інші дії щодо зручності здійснення покупок.

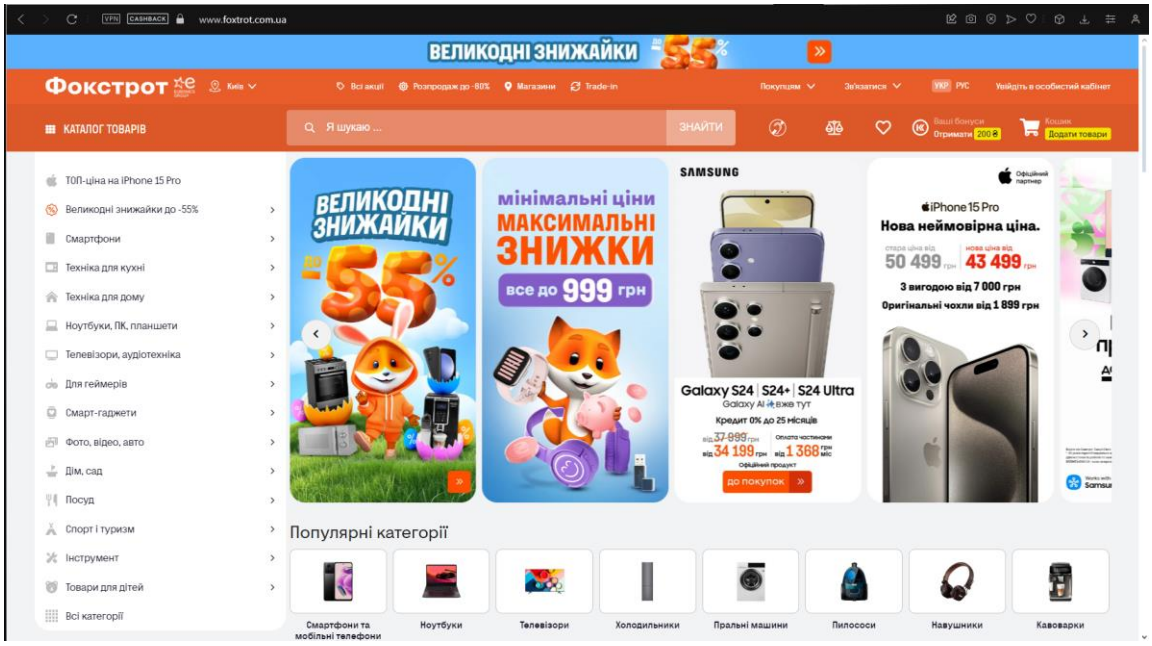

Рисунок 1.1 – Головна сторінка веб сайту «Фокстрот»

Дослідження дизайну: Дизайн інтерфейсу веб-сайту Фокстрот, вражає своєю сучасністю та привабливістю. Чистий і зрозумілий дизайн створює

комфортне середовище для користування, що сприяє легкій навігації та зручному пошуку товарів. Інформація розташована логічно, що дозволяє швидко орієнтуватися на веб-сайті. Колірна палітра та використання шрифтів гармонійно поєднуються, створюючи єдиний стиль, який вражає користувача з першого погляду.

Аналіз контенту: веб сайт Фокстрот містить різноманітний контент, який включає фотографії товарів, детальні описи, технічні характеристики, акції та знижки, новини та контактну інформацію, це все представлено на рисунку 1.2. Інформація про товари надається з достатньою повнотою, що дозволяє споживачам зробити усвідомлений вибір при здійсненні покупки. Крім того, на веб сайті регулярно розміщується акційна інформація, що привертає увагу покупців та сприяє здійсненню ними покупок.

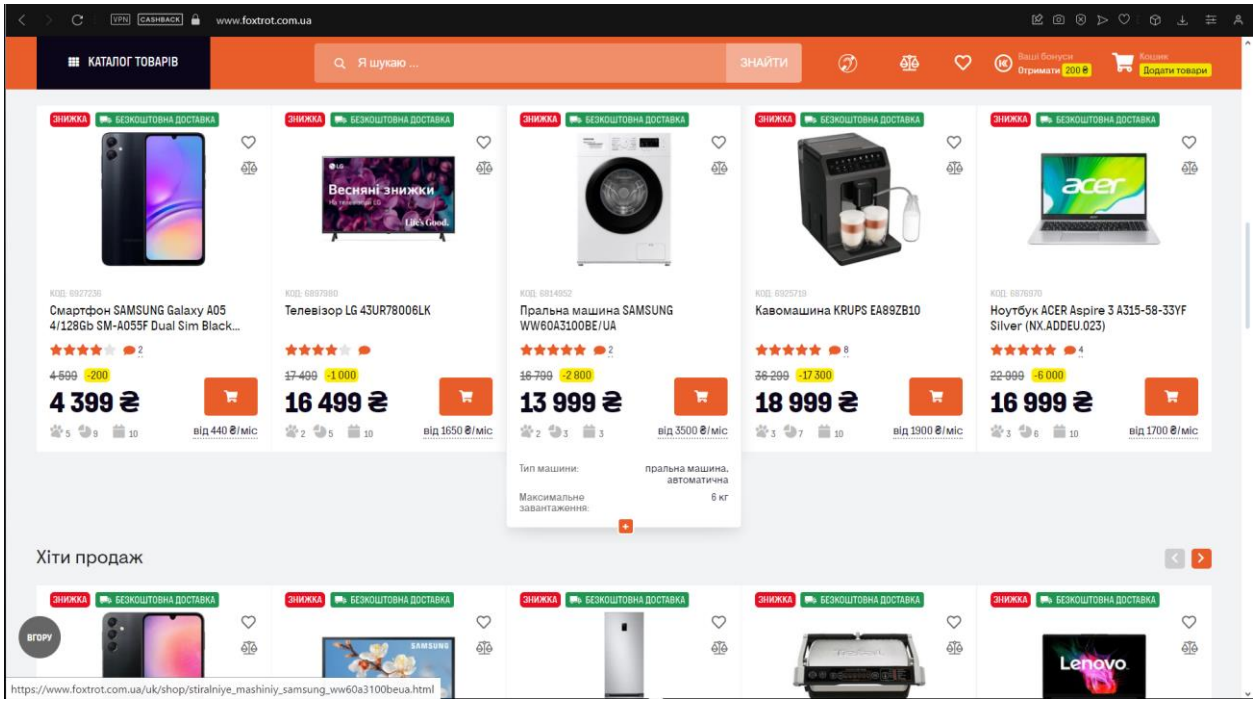

Рисунок 1.2 – Контент веб сайту «Фокстрот»

Оцінка технічних аспектів: з точки зору технічних аспектів, веб сайт Фокстрот працює швидко та ефективно, сторінки завантажуються миттєво, що дозволяє користувачам швидко переглядати та здійснювати покупки. Сайт

також оптимізований для використання на мобільних пристроях, що дозволяє користувачам зручно користуватися ним на різних пристроях, представлено на рисунку 1.3. Використання сучасних технологій і заходів безпеки гарантує надійність і безпеку покупок на сайті.

Рисунок 1.3 – Контент веб сайту «Фокстрот» на смартфоні

Основні переваги і недоліки веб сайту представлені в таблиці 1.1.

Таблиця 1.1

Переваги і недоліки веб сайту «Фокстрот»

Переваги	Недоліки
Широкий асортимент товарів	Відсутність можливості фільтрації товарів за певними ознаками

Продовження табл. 1.1

Інтуїтивний пошук та фільтрація товарів	Відсутність детальної інформації про деякі товари
Різні способи оплати та доставки	Відсутність вказаної кількості товарів, яка наявна на складі
Привабливий та зручний дизайн інтерфейсу	Відсутність онлайн-чату для консультації з покупцями
Швидкодія та оптимізованість для мобільних пристроїв	Відсутність розділу з корисними порадами для покупців

Основний недолік, веб сайту «Фокстрот», це не вказано кількість доступних одиниць товару, що не є зручну для користувачу, бо потрібно дзвонити в підтримку і уточнювати дану інформації, даний недолік ми виправимо на нашому веб сайті.

1.3 Веб сайт «Moyo.ua»

Головна сторінка представлена на рисунку 1.4, веб сайт пропонує широкий асортимент товарів побутової техніки, електроніки, аксесуарів та інших товарів [2]. Функціональність веб сайту дозволяє користувачам легко знаходити необхідні товари за допомогою різних категорій, фільтрів та пошуку.

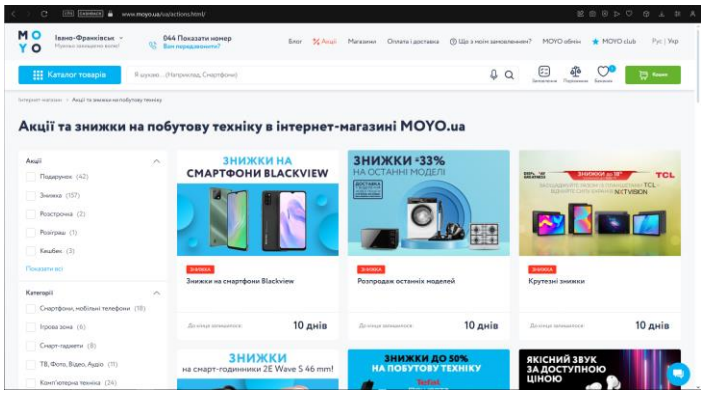

Рисунок 1.4 – Головна сторінка веб сайту «Moyo.ua»

Кожен товар має детальний опис, технічні характеристики та відгуки покупців, що допомагає користувачам зробити обдуманий вибір. Крім того, сайт надає можливість оформлення замовлення та обрання зручного способу доставки та оплати.

Дизайн веб сайту “Moyo.ua” вражає своєю модерністю та естетичністю. Структура веб сайту логічно організована, а навігація є інтуїтивно зрозумілою. Колірна палітра та використання шрифтів створюють приємне візуальне враження, а елементи дизайну гармонійно поєднуються між собою.

Сайт містить розгорнутий контент для кожного товару, представлено на рисунку 1.5, включаючи зображення, опис, характеристики та відгуки покупців. Крім того, на веб сайті регулярно оновлюється інформація про акції, знижки та спеціальні пропозиції, що привертає увагу покупців та стимулює їхні покупки.

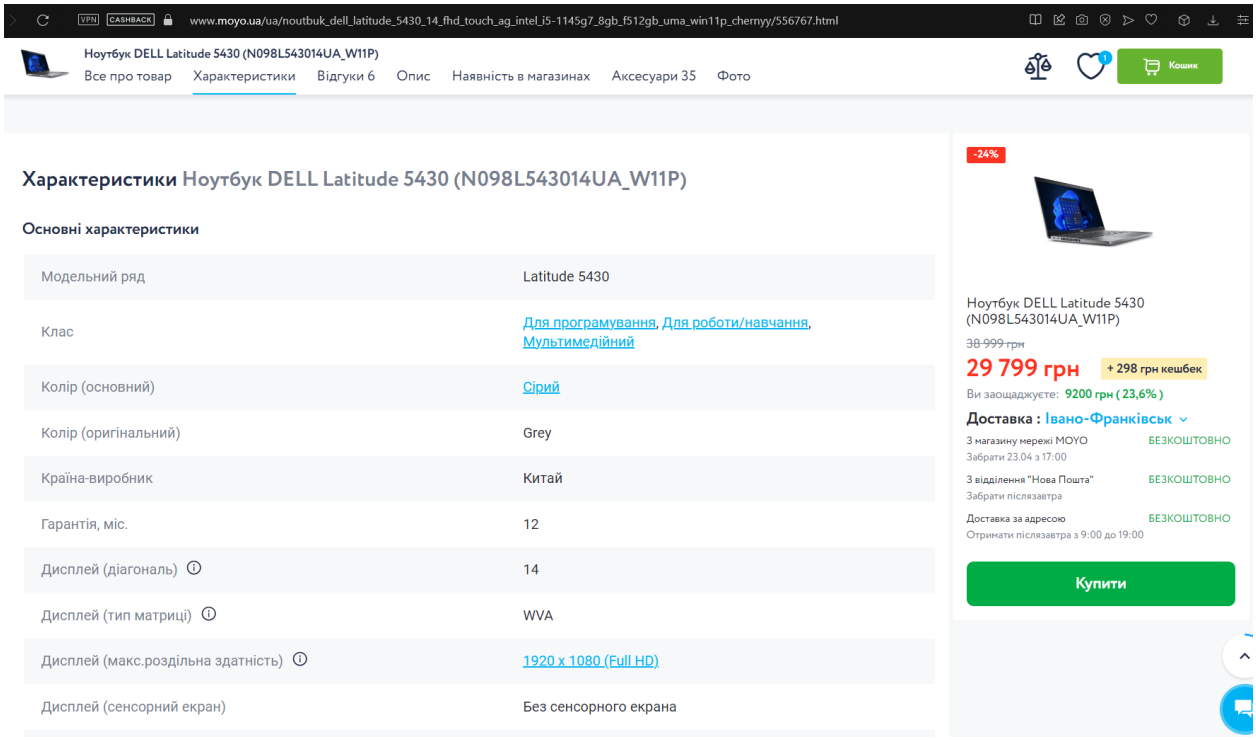

Рисунок 1.5 – Опис товару на веб сайті «Moyo.ua»

Сайт має швидку та ефективну роботу, сторінки завантажуються швидко, що забезпечує зручне переглядання товарів. Дизайн веб сайту також

адаптований для мобільних пристроїв, що дозволяє користувачам зручно переглядати його на різних пристроях, представлено на рисунку 1.6.

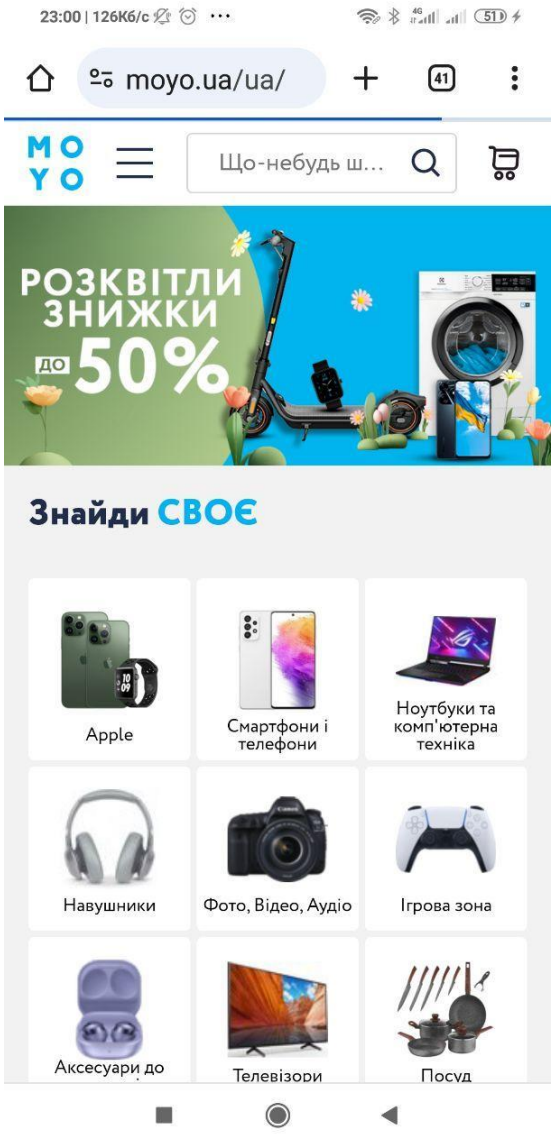

Рисунок 1.6 – Головна сторінка вебсайту «Moyo.ua» на смартфоні

Використання сучасних технологій забезпечує безпеку та надійність покупок на даному веб сайті. Основні переваги і недоліки веб сайту представлені в таблиці 1.2.

Таблиця 1.2

Основні переваги і недоліки веб сайту «Moyo.ua»

Переваги	Недоліки

Широкий асортимент вибору товарів в каталозі	Не юзабіліті інтерфейс для звичайного користувача
Детальний опис кожного товару, який наявний в каталозі товарів	Занадто велика кількість рекламних банерів і спливаючих вікон
Є можливість здійснювати оформлення замовлення	Інколи повільне завантаження сторінок
Створений адаптивний дизайн для мобільних пристроїв та інших пристроїв з різними діагоналями і розширенням екрану	Обмежений вибір систем, через які можна здійснювати оплату вибраного товару

1.4 Вебсайт «Rozetka»

Сайт «Rozetka» [3], представлено на рисунку 1.7, веб сайт пропонує широкий асортимент товарів, включаючи побутову техніку, електроніку, одяг, взуття та багато іншого. Функціональність веб сайту дозволяє користувачам легко шукати та фільтрувати товари за категоріями, брендами, ціною та іншими параметрами. Кожен товар має детальний опис, відгуки покупців, а також можливість порівняння з аналогічними товарами.

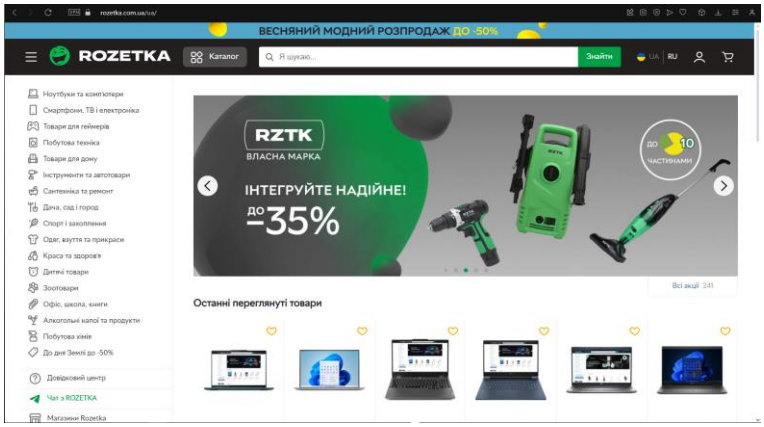

Рисунок 1.7 – Головна сторінка веб сайту «Rozetka»

Дизайн веб сайту, вражає своєю чіткістю та простотою. Інтерфейс веб сайту логічно організований, а навігація є зручною та інтуїтивно зрозумілою. Колірна схема та використання шрифтів створюють приємне візуальне враження, а анімаційні ефекти додають динаміки.

Аналіз контенту: веб сайт містить розгорнутий контент для кожного товару, включаючи зображення, опис, технічні характеристики та відгуки покупців. Крім того, на веб сайті регулярно оновлюється інформація про акції, знижки та спеціальні пропозиції.

Оцінка технічних аспектів: веб сайт має високу швидкість завантаження сторінок та відмінну оптимізацію і адаптивність сторінки на різних розширеннях екрану, представлено на рисунку 1.8. Використання сучасних технологій забезпечує безпеку та надійність покупок на веб сайті.

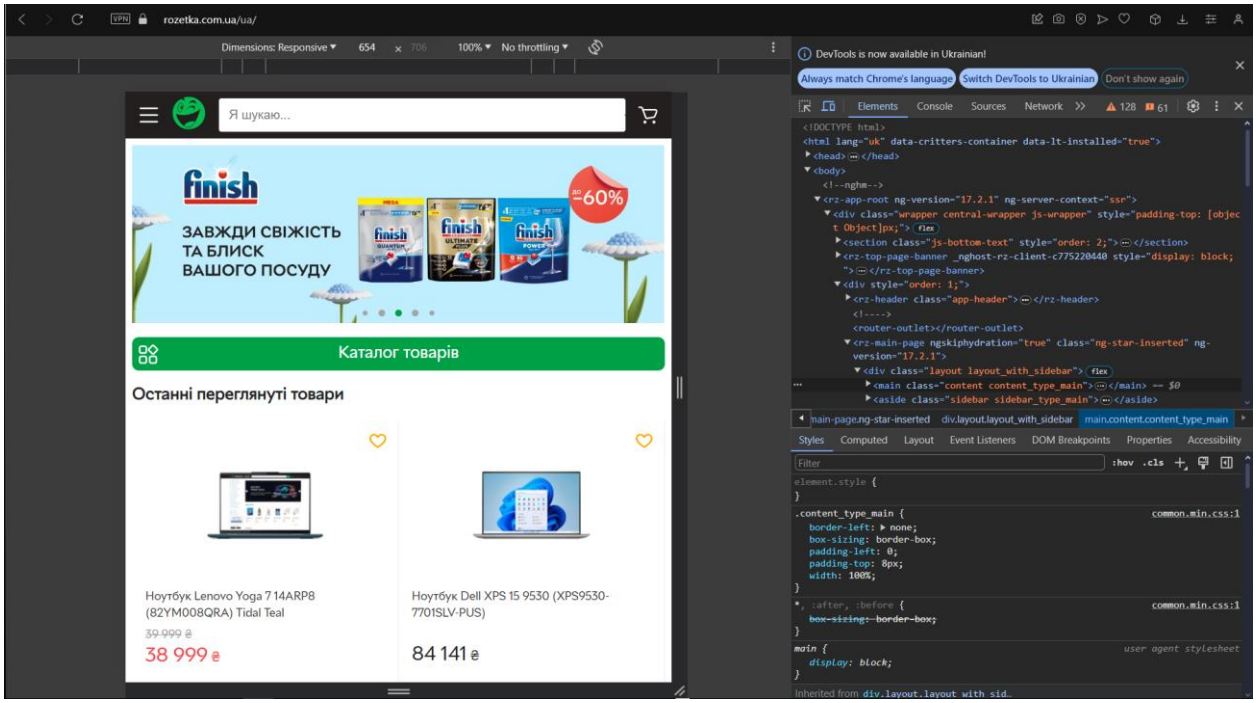

Рисунок 1.8 – Адаптивність веб сайту на різному розширенні екрану

Основні переваги і недоліки веб сайту представлено в таблиці 1.3.

Таблиця 1.3

Основні переваги і недоліки веб сайту «Rozetka»

Переваги	Недоліки
Широкий вибір товарів в каталозі	Велика кількість рекламних пропозицій і акцій
Детальний опис кожного товару, який знаходиться в каталозі товарів	Обмежена вибір платіжних систем
Наявність відгуків покупців	Мала кількість вибору фільтрів
Наявність маркетплейсу	———
Свій додаток на телефон, а також адаптивний дизайн для різних екранів	———

1.5 Постановка задачі

Об’єкт проєкту: розробка веб сайту для продажу побутової техніки.

Предмет проєкту: розробка веб сайту для продажу побутової техніки.

Мета проєкту полягає в створенні та впровадженні веб сайту для ефективної та зручної покупки побутової техніки онлайн.

Завдання проєкту:

Аналіз ринку та аналогів: Дослідити роботу існуючих сайтів для продажу побутової техніки. Проаналізувати їх функціонал, дизайн, контент, технічні аспекти, а також ключові переваги та недоліки.

Визначення цілей та завдань сайту: Спроектувати функціонал веб сайту з урахуванням потреб цільової аудиторії. Визначити основні завдання, які веб сайт повинен вирішувати, такі як зручний пошук товарів, замовлення, відміна замовлення, оплата, доставка тощо.

Розробка дизайну та інтерфейсу: Розробити зручний та привабливий дизайн інтерфейсу веб сайту, що відповідає сучасним тенденціям та сприяє зручності користування. Врахувати адаптивність для різних типів пристроїв.

Розробка функціоналу: Створити необхідний функціонал для користувачів, такий як пошук товарів, фільтрація за параметрами, оформлення замовлення, оплата, відстеження доставки тощо.

Технічна реалізація: Використати мови програмування та технології, такі як JavaScript, React, Django, для створення веб сайту. Забезпечити швидку та надійну роботу веб сайту, оптимізувати його для пошукових систем та мобільних пристроїв та пристроїв з великими екранами.

Тестування та впровадження: Провести тестування роботи веб сайту на різних етапах розробки. В першу чергу за допомогою такого метода тестування, як тестування чорного ящика. Впровадити розроблений веб сайт в онлайн середовище та забезпечити його стабільну, безперебійну і найголовніше – безпечну роботу для комфортного користування.

Висновки до розділу 1

У розділі було проведено дослідження веб сайтів головних конкурентів на ринку продажу побутової техніки. Проведене дослідження сайтів конкурентів дає нам чітке уявлення про те, які сильні та слабкі сторони існують у цій галузі онлайн-продажу побутової техніки.

Також був проведений детальний опис плану задачі. Чітко обґрунтовано вибрані технології розробки та тестування. Був описаний функціонал, який повинен містити розроблений програмний продукт.

Не менш важливою складовою стало дослідження сучасних вимог до веб сайтів, що дозволило створити уявлення про той користувацький досвід, яким веб сайт повинен бути наділений для його комфортного використання та розширення клієнтської бази.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-САЙТУ

2.1 Розробка структури веб-сайту

Структура веб сайту представлена на рисунку 2.1.

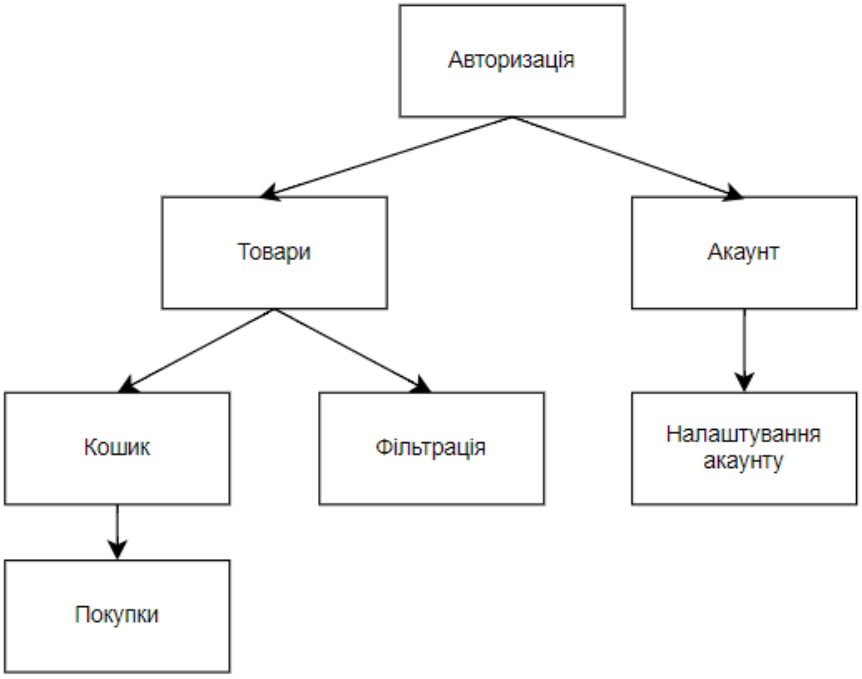

Рисунок 2.1 – Структура веб сайту

Зображення демонструє структуру веб сайту, що орієнтована на користувача із зосередженням на взаємодії та доступності різних функцій. Центральним елементом є розділ "Товари", який є основою для навігації по сайту та взаємодії з іншими розділами. Від цього блоку відходять стрілки до таких ключових елементів, як "Авторизація", що дозволяє користувачам входити на сайт або зареєструвати новий аккаунт, та "Акаунт", де можна переглядати та редагувати особисті дані.

"Налаштування акаунту" пропонують користувачам змінювати пароль та налаштовувати параметри приватності, що забезпечує додатковий рівень безпеки та персоналізації. "Кошик" відіграє важливу роль у процесі покупки,

дозволяючи користувачам переглядати вибрані товари перед оформленням замовлення, а "Покупки" є логічним завершенням процесу покупки, де можна оформити та оплатити товари. Опція "Фільтрація" надає можливість сортувати товари за різними параметрами, спрощуючи пошук необхідних товарів. Загалом, структура веб-сайту здається інтуїтивно зрозумілою та зручною для користувачів, оскільки вона спрямована на забезпечення ефективного та приємного досвіду покупок онлайн.

2.2 Середовище розробки Visual Studio Code

Вікно середовища Visual Studio Code представлене на рисунку 2.2.

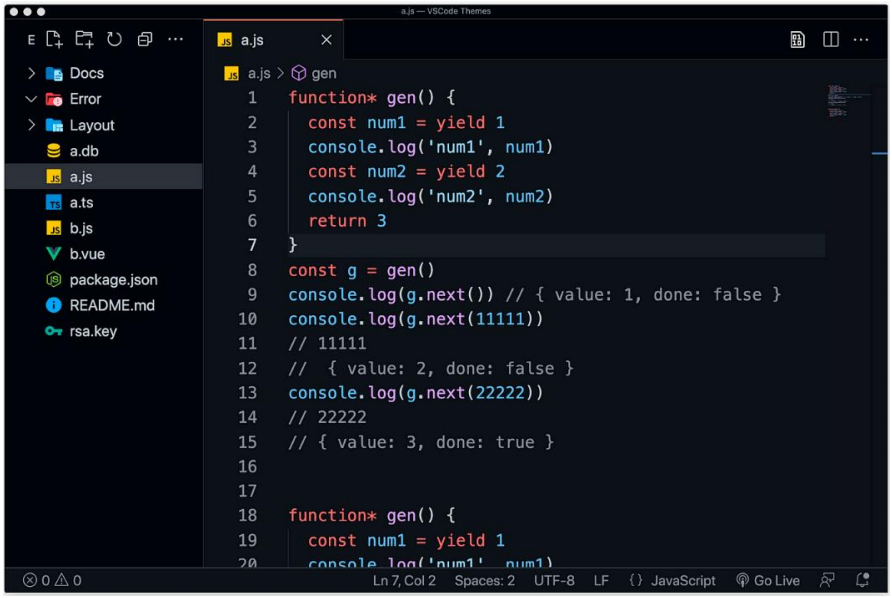

Рисунок 2.2 – Вікно середовища Visual Studio Code

В середовищі розробки Visual Studio Code відзначається високий рівень функціональності, що сприяє зручній та продуктивній роботі програмістів у процесі написання коду. Зокрема, інтегрована система контролю версій забезпечує ефективне ведення робочих проектів та сприяє колективному взаємодії у команді. Механізм автоматичного доповнення коду, базований на алгоритмах машинного навчання, дозволяє швидко і точно завершувати

написання команд та ідентифікаторів. Крім того, наявність інтегрованих інструментів аналізу коду дозволяє вчасно виявляти та виправляти помилки, що сприяє підвищенню якості розроблюваного програмного забезпечення [5].

Продуктивність в Visual Studio Code підтримується за рахунок оптимізованого інтерфейсу користувача, що забезпечує зручний доступ до усіх необхідних функцій та інструментів. Інтуїтивно зрозумілі команди та швидкі клавіатурні скорочення сприяють ефективній навігації та редагуванню коду. Вбудована система розширень дозволяє кратно розширювати функціональність середовища розробки відповідно до індивідуальних потреб користувача, надаючи йому можливість інтеграції з різноманітними зовнішніми інструментами та широким асортиментом сервісів.

Ключовою особливістю Visual Studio Code є його крос-платформенність, що дозволяє користувачам працювати на різних операційних системах, таких як Windows, macOS та Linux, забезпечуючи єдність середовища незалежно від обраної платформи. Це сприяє зручності роботи в умовах розподіленого розробництва та дозволяє зосередитися на розвитку програмного забезпечення, уникнувши перешкод, пов'язаних з обмеженнями вибору операційної системи.

У Visual Studio Code відзначається також висока ефективність у розробці веб-сайтів. Його інтегровані інструменти для веб-розробки дозволяють швидко створювати та відлагоджувати різноманітні веб-додатки та сайти. Вбудовані розширення для роботи з мовами програмування, такими як HTML, CSS та JavaScript, забезпечують розширені можливості автодоповнення коду, синтаксичного підсвічування та перевірки синтаксису, що допомагає зберегти час та знизити кількість помилок.

Крім того, Visual Studio Code інтегрується з різноманітними інструментами для розробки веб-сайтів, такими як платформи для фронтенд-розробки (наприклад, React, Angular, Vue.js), серверні фреймворки (наприклад, Node.js, Express.js) та системи управління версіями (наприклад, Git). Це дозволяє зручно і ефективно працювати з повним стеком технологій, необхідних для розробки сучасних веб-додатків.

Наявність інтегрованих інструментів для відлагодження та налагодження веб-додатків, таких як відлагоджувальник JavaScript та інші розширення, дозволяє ефективно виявляти та виправляти помилки під час розробки. Комбінація цих можливостей робить Visual Studio Code привабливим вибором для розробників, що працюють у сфері веб-розробки, надаючи їм потужний та зручний інструментарій для реалізації своїх ідей та проєктів.

2.3 React/JavaScript

Реакт (React.js) є відкритим JavaScript фреймворком, призначеним для розробки інтерактивних веб-інтерфейсів. Використання React у веб-розробці є важливим з точки зору підвищення ефективності, оскільки цей фреймворк спрощує створення складних інтерфейсів шляхом розділення їх на незалежні компоненти. Він базується на концепції віртуального DOM (Document Object Model), що дозволяє швидко відображати зміни на сторінці без необхідності перерендерингу всього документа.

Застосування React у веб-сайті для продажу побутової техніки забезпечує багато переваг. Наприклад, він дозволяє створювати розширювальні та перевикористовувальні компоненти, такі як каруселі для відображення товарів, форми для замовлення та фільтри для швидкого пошуку товарів за різними параметрами. Крім того, завдяки можливості використання JSX (розширеного синтаксису JavaScript), розробники можуть легко комбінувати HTML-розмітку з JavaScript кодом, що спрощує розробку складних інтерфейсів.

До інших переваг React відноситься його ефективне управління станом додатку. Завдяки вбудованим засобам керування станом, таким як стейт і прописи, розробники можуть легко відслідковувати зміни даних та оновлювати інтерфейс відповідно до них без необхідності вручну маніпулювати DOM. Це особливо важливо для веб-сайтів із великою кількістю динамічного контенту, таких як інтернет-магазини, оскільки воно дозволяє покращити продуктивність та відповідати на зміни користувацьких дій миттєво. Таким чином, використання

React у веб-сайті продажу побутової техніки може значно полегшити процес розробки та забезпечити високий рівень користувацького досвіду. JavaScript (JS) є ключовою складовою веб-розробки, і його поєднання з React.js створює потужний інструментарій для створення інтерактивних веб-сайтів. JS використовується для програмування логіки веб-сторінок, відповіді на події користувача та взаємодії з сервером. У поєднанні з React, JavaScript дозволяє створювати динамічні та ефективні веб-інтерфейси.

На веб-сайті продажу побутової техніки JavaScript використовується для різноманітних завдань. Наприклад, він може бути використаний для взаємодії з елементами інтерфейсу, такими як кнопки та поля введення, збір даних введених користувачем та відправлення їх на сервер для обробки. Крім того, JS може використовуватися для анімації елементів, щоб зробити веб-сайт більш привабливим та користувацьки орієнтованим. У контексті React.js, JavaScript також використовується для створення та управління компонентами. JavaScript дозволяє також використовувати сучасні підходи до програмування, такі як асинхронні запити (за допомогою async/await), для покращення продуктивності та ефективності веб-сайту.

Узагальнюючи, JavaScript у поєднанні з React.js є потужним інструментом для розробки веб-сайтів, що пропонують широкий спектр функціональності та забезпечують високий рівень користувацького досвіду. Він дозволяє створювати динамічні, інтерактивні та ефективні веб-інтерфейси, що відповідають сучасним вимогам та очікуванням користувачів.

2.4 Схема бази даних

Для створення бази даних нашого проекту було ретельно проаналізувано різні системи управління базами даних і вирішили обрати PostgreSQL. Цей вибір був обґрунтований кількома ключовими факторами.

Перш за все, PostgreSQL відома своєю високою надійністю та стабільністю, що є критично важливим для веб-сайтів, які повинні працювати

безперебійно. Крім того, ця система підтримує широкий спектр функцій, включаючи розширені типи даних, роботу з геоданими та текстовий пошук. Це дозволяє нам ефективно реалізовувати різноманітні вимоги нашого проекту.

Ще однією важливою перевагою PostgreSQL є підтримка формату JSON. Це дає змогу гнучко працювати з даними та швидко реагувати на зміни, що виникають у процесі розробки. Нарешті, PostgreSQL є програмним забезпеченням з відкритим вихідним кодом. Це означає, що ми маємо доступ до вихідного коду та можемо модифікувати його відповідно до наших потреб. Активна спільнота розробників і користувачів також є великою перевагою, оскільки вони можуть надати підтримку і допомогти у вирішенні проблем [4].

Таким чином, обрання PostgreSQL для проекту обумовлено її надійністю, розширеними можливостями, підтримкою JSON і відкритим вихідним кодом, що разом забезпечує оптимальну продуктивність і гнучкість у роботі з вибраною базою даних.

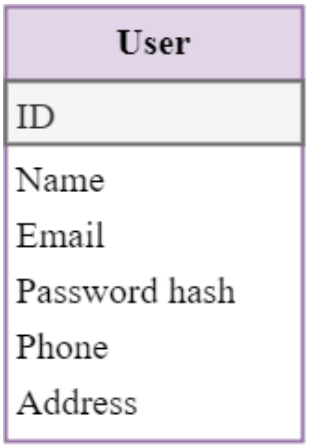

Рисунок 2.3 – Таблиця Користувачі

Користувачі: ця таблиця важлива для управління доступом до різних ресурсів сайту. Атрибути таблиці:

1. ID: серійний унікальний ідентифікатор, який автоматично збільшується.
2. Ім'я: текстове поле, що містить повне ім'я користувача.

3. Email: унікальне текстове поле, яке використовується для входу на сайт.

4. Хеш пароля: захищене поле, в якому зберігається хеш-код пароля користувача.

5. Номер телефону: текстове поле, що містить номер телефону контактної особи.

6. Адреса: текстове поле, що містить адресу користувача. Електронна пошта: унікальне текстове поле, яке використовується для входу на веб сайт.

Рисунок 2.4 – Таблиця Продукти

Товари (Products): таблиця відіграє ключову роль у каталогізації та інвентаризації товарів, доступних для продажу:

1. ID товару: серійний унікальний ідентифікатор.

2. Назва: текстове поле, що містить назву товару.

3. Опис: текстове поле з детальним описом товару.

4. Ціна: числове поле, яке містить ціну товару.

5. Категорія: текстове поле, що указує категорію, до якої належить конкретний вид товару.

6. Кількість на складі: числове поле, яке вказує доступну кількість товару на складі.

Рисунок 2.5 – Таблиця Замовлення

Замовлення (Orders): таблиця забезпечує збереження інформації про замовлення, що виконуються через сайт:

1. ID замовлення: серійний унікальний ідентифікатор.
2. ID користувача: зовнішній ключ, що пов'язує замовлення з конкретним користувачем.
3. Дата замовлення: дата і час створення замовлення.
4. Загальна сума: загальна вартість замовлення.
5. Статус: текстове поле, що вказує поточний статус замовлення.

Ці таблиці взаємодіють між собою через зовнішні ключі, що зображено на рисунку 2.6.

Рисунок 2.6 – Схема зв’язку таблиць БД

Для полегшення узгодженості та управління базою даних таблиця "Замовлення" пов'язана з таблицею "Користувачі" за допомогою зовнішнього ключа. Кожне замовлення пов'язане з конкретним користувачем, який його

розмістив, що дозволяє аналізувати замовлення та відстежувати активність користувача на веб сайті.

Зовнішній ключ "ID користувача" в таблиці "Замовлення" посилається на унікальний ID користувача в таблиці "Користувачі". Цей зв'язок дозволяє легко знайти таку інформацію, як ім'я, електронна пошта, номер телефону та інші контактні дані користувача, який розмістив кожне замовлення.

2.5 Веб-фреймворк Django

Back-end (серверна частина) веб-сайту розроблена на мові програмування Python. Вибір Python є стратегічним рішенням, оскільки він має простий і легкий для читання код, що полегшує розробку та підтримку проектів. Python також пропонує широкий спектр бібліотек і фреймворків, які дозволяють ефективно використовувати його в багатьох сферах програмування.

Серед готових рішень для спрощення процесу розробки веб-додатків виділяється фреймворк Django. Він надає широкий спектр інструментів для швидкого та ефективного створення веб-додатків. Однією з головних переваг Django є вбудований ORM (Object-Relational Mapping), який дозволяє розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід. Це робить взаємодію з базами даних більш інтуїтивно зрозумілою і менш залежною від конкретної СУБД.

Django також надає низку вбудованих інструментів для спрощення розробки веб-додатків. Серед них – готові інструменти для аутентифікації користувачів, обробки форм введення даних, створення панелей адміністрування тощо. Крім того, Django дотримується концепції Model-View-Controller (MVC), яка забезпечує більшу структурованість і простоту обслуговування, розбиваючи веб-додатки на логічні компоненти.

Однією з ключових переваг Django є вбудовані механізми безпеки, які захищають веб-додатки від різних атак, таких як CSRF, XSS, SQL ін'єкції та інші.

За допомогою Django ці механізми можна легко налаштувати для забезпечення високого рівня безпеки веб-додатків.

Загалом, Django – це потужний інструмент для розробки веб-додатків, який дозволяє швидко та ефективно створювати високоякісні проекти. Його гнучкість, продуктивність і безпека роблять його ідеальним для нашого проекту з продажу побутової електроніки. Ще однією перевагою Django є велика та активна спільнота розробників, які постійно розвивають фреймворк, надають підтримку та допомагають вирішувати проблеми. Це забезпечує стабільність та актуальність фреймворку з часом, а також можливість швидко знаходити рішення технічних проблем та питань, які можуть виникнути під час розробки веб-додатку.

Висновки до розділу 2

Результатом виконання цього розділу став проєкт веб-сайту для продажу побутової техніки. У ньому було описано структуру веб-сайту, визначено його основні компоненти та їх взаємозв’язок. Також було визначено інструментальні засоби розробки веб-сайту. Для реалізації веб-сайту було обрано фреймворки Django та React. Також у цьому розділі було визначено структуру та зв’язки в базі даних веб-сайту.

РОЗДІЛ 3. ПРОГРАМНА РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБ-САЙТУ

3.1 Розробка інтерфейсу веб-сайту

Сторінка реєстрації представлена на рисунку 3.1.

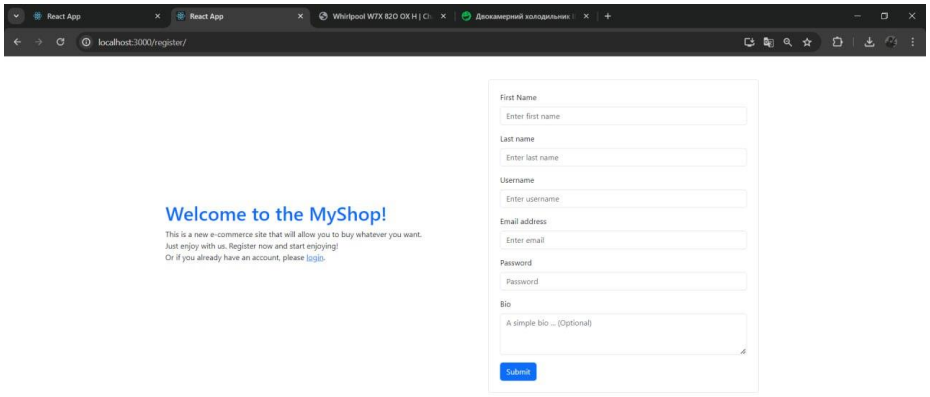

Рисунок 3.1 – Сторінка реєстрації на веб-сайті

Зображення показує інтерфейс реєстрації для веб-платформи "MyShop". Цей інтерфейс є чистим та зручним для користувача, створеним для полегшення навігації потенційних користувачів, які бажають створити обліковий запис. З лівого боку екрану знаходиться привітальне повідомлення, яке говорить: "Ласкаво просимо до MyShop!" за яким слідує короткий опис, що інформує відвідувачів, що реєстрація дозволить їм купувати все, що вони хочуть. Також надається можливість для користувачів, які вже мають обліковий запис, увійти в систему розробленого веб сайту магазину.

Праворуч на екрані розташована сама форма реєстрації з полями для введення особистої інформації, включаючи ім'я, прізвище, ім'я користувача та адресу електронної пошти. Також є поле для створення пароля та необов'язкове поле, де нові користувачі можуть додати коротку біографію про себе. Кожне поле

містить текст-заповнювач, що керує користувачів щодо необхідної інформації. Синя кнопка "Надіслати" розташована внизу форми, запрошуючи користувачів завершити процес реєстрації.

Загальний дизайн є мінімалістичним з достатньою кількістю білого простору, що покращує читабельність та зосередження на деталях реєстрації. Інтерфейс здається орієнтованим на користувача, маючи на меті надання потенційним клієнтам безпроблемного досвіду реєстрації, що наближає їх на крок ближче до вивчення та покупки товарів з MyShop. Сторінка авторизації представлена на рисунку 3.2.

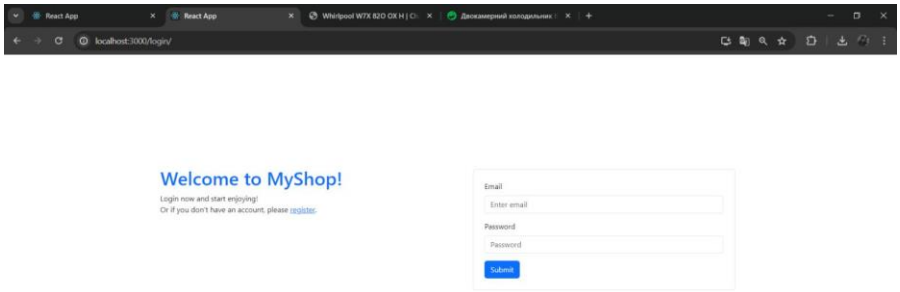

Рисунок 3.2 – Сторінка авторизації на веб-сайті

На зображенні, представлено інтерфейс авторизації для веб-сайту під назвою "MyShop". Це простий та зручний дизайн, який забезпечує легкий доступ відвідувачів до їхніх облікових записів або реєстрацію, якщо вони нові. Фон є однотонним, що зосереджує увагу користувача на формі входу. Для тих, хто ще не має облікового запису, є підказка, яка направляє їх на сторінку реєстрації у системі магазину.

Сама форма входу складається з двох основних полів введення: одне для адреси електронної пошти та інше для пароля. Кожне поле чітко позначено, що робить інтуїтивно зрозумілим для користувачів, де вводити свою інформацію. Кнопка "Надіслати" розташована нижче цих полів, готова до натискання після

введення користувачем своїх облікових даних. Цей стрункий дизайн забезпечує швидкий вхід користувачів без будь-яких проблем або плутанини. Інтерфейс редагування профілю представлено на рисунку 3.3.

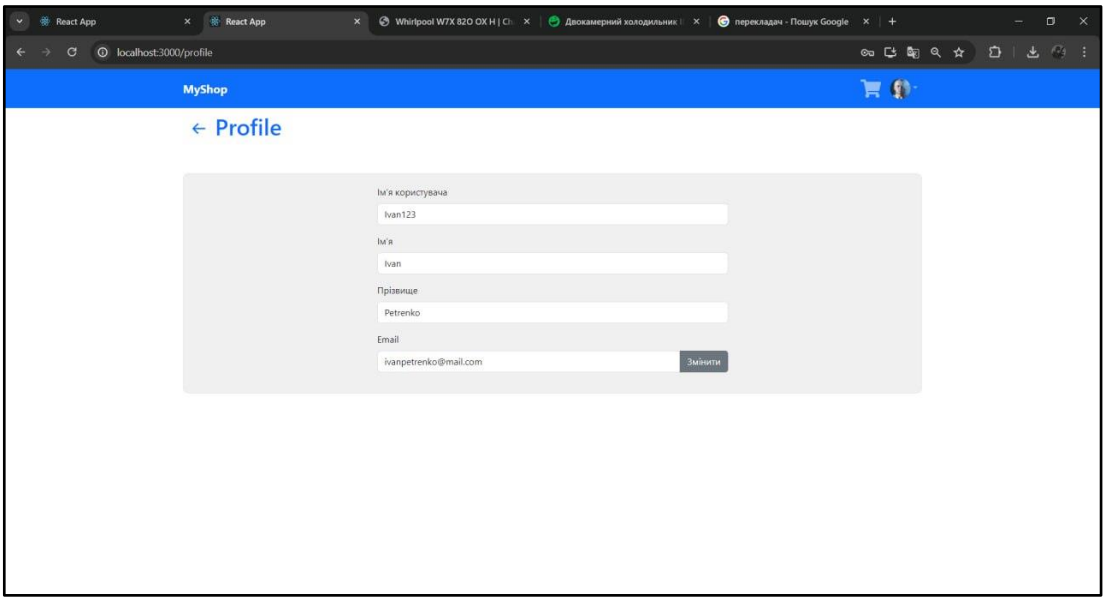

Рисунок 3.3 – Сторінка параметрів акаунту

На зображенні відображено інтерфейс налаштувань користувацького профілю для веб-сайту або додатка, який, схоже, належить до онлайн магазину "MyShop". Цей інтерфейс забезпечує простий та зручний доступ до особистих налаштувань користувача, дозволяючи легко керувати інформацією профілю та персоналізувати досвід користування. Відкрита сторінка "Профіль" містить поля для введення та редагування особистих даних, таких як ім'я, прізвище та електронна пошта, з плейсхолдерами, які надають користувачам зрозумілі підказки щодо введення інформації.

Кнопка "Змінити" пропонує зручний спосіб збереження внесених змін або переходу до додаткових опцій редагування профілю. Навігаційний елемент у вигляді стрілки "Назад" забезпечує легке повернення до попередньої сторінки або головного екрану, що робить навігацію інтуїтивно зрозумілою та зручною. Загалом, інтерфейс налаштувань користувацького профілю виглядає організованим та ефективним, спрямованим на забезпечення користувачам контролю над їхньою особистою інформацією та покращення загального досвіду

користування сайтом "MyShop". Інтерфейс каталогів товарів представлено на рисунку 3.4.

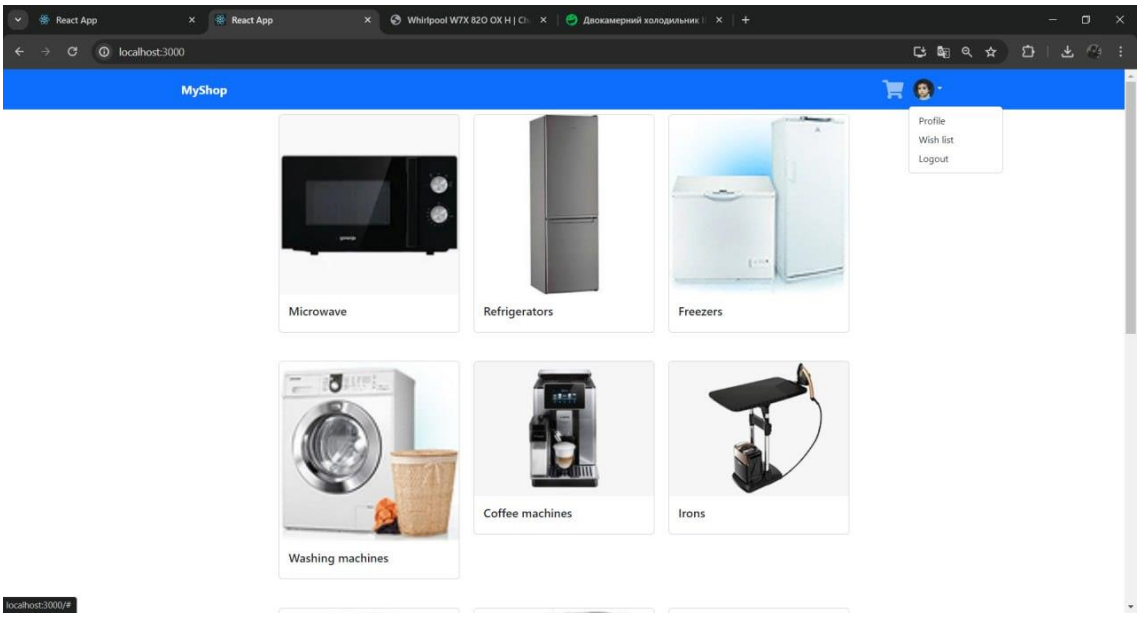

Рисунок 3.4 – Каталоги товарів

Зображення демонструє інтерфейс каталогу товарів для онлайн-магазину “MyShop”. Цей інтерфейс виглядає чистим та організованим, забезпечуючи легкий доступ до різних категорій товарів. У центрі сторінки розташовані зображення шести категорій товарів: мікрохвильові печі, холодильники, морозильники, пральні машини, кавоварки та праски. Кожна категорія супроводжується назвою, що дозволяє користувачам швидко знайти товар.

У верхньому правому кутку інтерфейсу розташоване меню користувача з опціями “Профіль”, “Список бажань” та “Вийти”, що надає додаткові можливості для персоналізації досвіду покупок. Логотип “MyShop” розташований у верхній частині сторінки, надаючи ідентифікацію бренду та сприяючи навігації. Загалом, інтерфейс каталогу товарів виглядає інтуїтивно зрозумілим та зручним для користувачів, що прагнуть здійснити покупки в онлайн-магазині “MyShop”. Інтерфейс товарів представлено на рисунку 3.5.

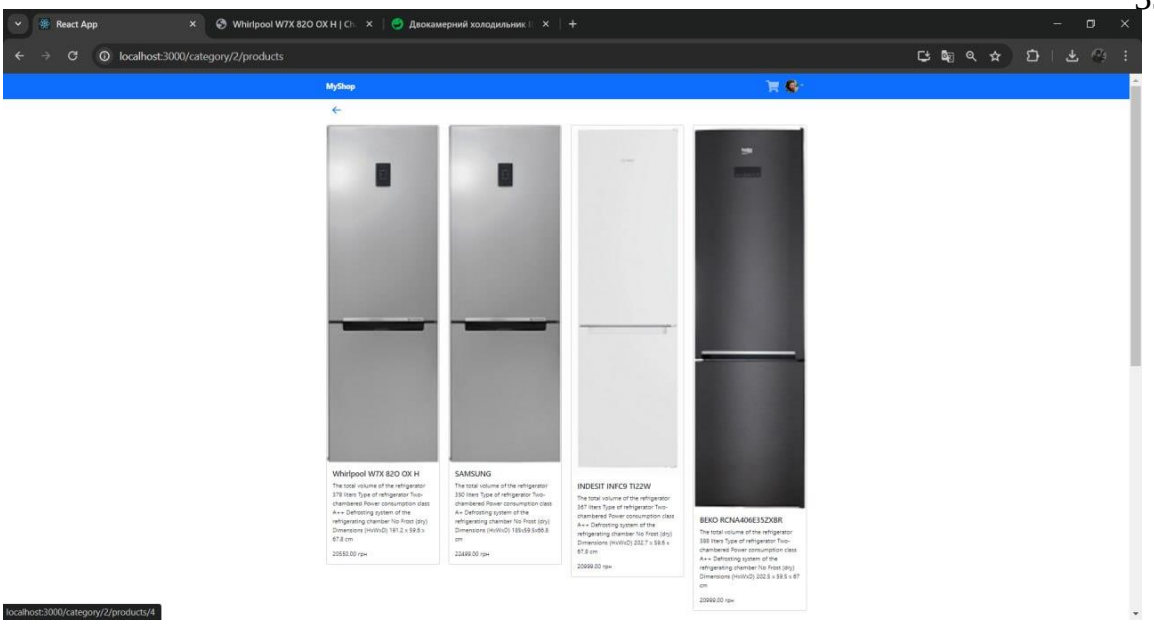

Рисунок 3.5 – Каталог холодильників

На зображенні відображено інтерфейс каталогу холодильників на веб-сайті. Цей інтерфейс показує чотири моделі холодильників різних брендів та моделей, які розташовані поруч один з одним для легкого порівняння. Кожен холодильник супроводжується коротким описом, що містить назву бренду та моделі, а також основну інформацію про продукт. Зображення холодильників високої якості дозволяють користувачам детально розглянути дизайн кожного з них і обрати той, що сподобався найбільше.

Інтерфейс має темну тему з білим текстом для легкого читання. У верхньому правому кутку веб-сторінки видно навігаційні іконки, які вказують на опції для взаємодії користувача. Це може включати перехід до профілю користувача, список бажань або вихід з облікового запису. Загалом, інтерфейс каталогу холодильників здається зручним та інтуїтивно зрозумілим, забезпечуючи користувачам приємний досвід вибору та порівняння між собою різних моделей холодильників.

3.2 Тестування функціоналу веб-сайту

На рисунку 3.6 представлено додавання товару в кошик.

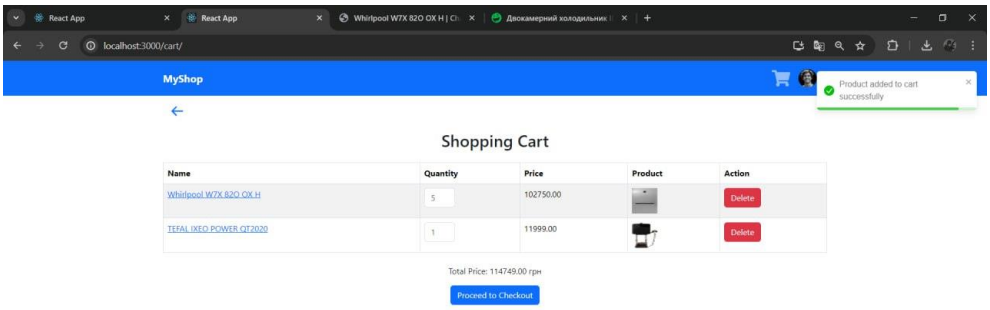

Рисунок 3.6 – Додавання товару в кошик

Користувач додав товар до свого кошика, як це видно з повідомлення в правому верхньому куті, що говорить "Product added to cart successfully". У кошику зараз два товари: "Whirlpool WTW8700EC" та "IDEAL HIGH POWER GRINDER", кожен з яких має свою кількість та ціну. Існує можливість видалити товар з кошика, натиснувши червоний кнопку "Delete". Загальна сума покупки також вказана на екрані. Внизу екрану розташована кнопка "Proceed to Checkout", яка дозволяє користувачеві перейти до наступного етапу оформлення покупки. Цей інтерфейс забезпечує зручне та інтуїтивно зрозуміле управління покупками, дозволяючи користувачам легко додавати, переглядати та видаляти товари з кошика, а також продовжувати процес оформлення замовлення.

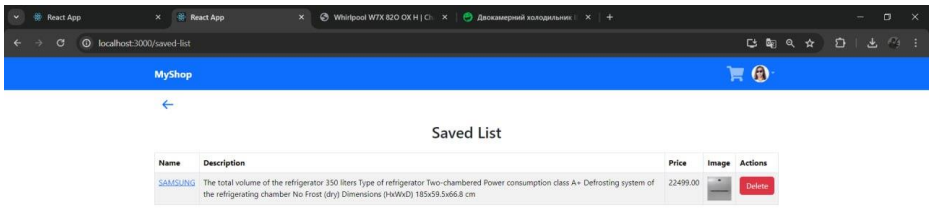

Рисунок 3.7 – Збережені товари

На зображенні 3.7 відображено інтерфейс користувача веб-сайту "MyShop", де користувач має можливість додавати товари до свого списку

збережених. Ця функція дозволяє користувачам зберігати товари, які їм подобаються, для швидкого доступу в майбутньому або для порівняння з іншими товарами. На інтерфейсі видно, що користувач вибрав певний товар, і після натискання кнопки "Save" або "Add to Saved", товар був успішно доданий до списку збережених, про що свідчить повідомлення на екрані. Також присутній список збережених товарів, де можна переглянути всі додані раніше товари, їх ціни, описи та зображення. Користувачі можуть видаляти товари зі списку збережених, якщо вони більше не зацікавлені в них, використовуючи відповідну кнопку "Delete" або "Remove". Цей інтерфейс спрощує процес планування покупок та дозволяє користувачам створювати персоналізований список бажаних товарів, який можна змінювати в будь-який час.

3.3 Аналіз результатів тестування

Аналіз результатів тестування веб-сайтів для продажу побутової техніки виявив, що вони демонструють високий рівень функціональності та ефективності. Під час проведення тестів було перевірено різноманітні аспекти веб-сайтів, включаючи функції пошуку, додавання товарів до кошика, оформлення замовлення та оплату, і усі ці функції виявилися цілком працездатними та надійними.

Перший етап аналізу стосувався функції пошуку. Тестування виявило, що пошукові запити користувачів обробляються ефективно та точно, результати відповідають введеним критеріям і відображаються коректно. Це забезпечує зручність та швидкість користувачам у пошуку необхідних товарів.

Другий етап аналізу охоплював функцію додавання товарів до кошика. Під час тестування було виявлено, що користувачі можуть безперешкодно додавати обрані товари до свого кошика, а інтерфейс кошика забезпечує зручність у перегляді та керуванні доданими товарами.

Третій етап аналізу включав функції оформлення замовлення та оплати. Під час тестування було перевірено, що процес оформлення замовлення є

інтуїтивно зрозумілим та безпроблемним, а система оплати працює надійно і забезпечує безпеку та конфіденційність даних користувачів.

Усі ці результати свідчать про те, що веб-сайт для продажу побутової техніки функціонують належним чином і готові до використання користувачами. Вони забезпечують зручний та ефективний спосіб покупки товарів онлайн і відповідають сучасним стандартам у веб-розробці.

Висновок до 3 розділу

Результатом виконання цього розділу є веб-сайт для продажу побутової техніки. Для цього було реалізовано зручний інтерфейс користувача веб-сайту, який є інтуїтивно зрозумілим. Також в цьому розділі було здійснено тестування веб-сайту методом чорного ящика. Суть тестування полягало в тому, щоб перевірити функціональність веб-сайту з точки зору користувача. Відповідно з результатами тестування було зроблено висновок про те, що веб-сайт працює коректно та виконує свій функціонал.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є розроблений веб-сайт для продажу побутової техніки.

У першому розділі було проаналізовано аналогічні розробки. Визначено їх функціонал та переваги і недоліки. В результаті аналізу було визначено функціональні вимоги до створеного веб-сайту для продажу побутової техніки.

У другому розділі було визначено проєкт майбутнього веб-сайту. Було обрано інструментальні засоби для розробки веб-сайту: фреймворки React та Django, мова програмування JavaScript, Python. Також було визначено структуру бази даних веб-сайту, зв’язок між сутностями бази даних. Крім того, було визначено структуру самого веб-сайту – взаємодію його складових.

У третьому розділі було описано інтерфейс користувача веб-сайту. Визначено яким чином реалізовано інтерфейс. Також було протестовано роботу веб-сайту. Тестування проводилось методом чорного ящика, тобто з точки зору користувача програмного продукту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Foxtrot: веб-сайт. URL: https://www.foxtrot.com.ua (дата звернення 18.04.2024)

2. Mojo.ua: веб-сайт. URL: https://www.moyo.ua/ua (дата звернення 18.04.2024)

3. Rozetka: веб-сайт. URL: https://rozetka.com.ua/ (дата звернення: 18.04.2024)

4. Promoter. База даних MySQL: веб-сайт. URL: https://promoter.net.ua/articles/baza-danix-mysql.html (дата звернення: 18.04.2024)

5. Visual Studio Code. Початок роботи: веб-сайт. URL: https://code.visualstudio.com/docs (дата звернення: 18.04.2024)

6. Django. Знайомство з Django: веб-сайт. URL: https://www.djangoproject.com (дата звернення: 19.04.2024)

7. QALight. Тестування веб-проєктів: основні етапи та поради: веб-сайт. URL: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi (дата звернення: 19.04.2024)

8. Кайл Сімпсон «Ви не знаєте JS. Типи та граматичні конструкції» / М .: Видавничий дім «Пітер Прес», 2019. – 240 с.

9. Девід Фланаган «JavaScript. Детальний посібник» / М .: Видавничий дім «Пітер Прес», 2013. – 1081 с.

10. Довідник CSS атрибутів: веб-сайт. URL: https://html-css.co.ua/dovidnuk-css-atrubytiv/ (дата звернення: 20.04.2024)

11. Програмування веб-застосувань (фронт-енд та бек-енд): книга / Роман Мельник. – Львів: Львівська Політехніка, 2018. – 248 с

12. The Art of Software Testing: довідник / Гленфорд Маєрс. – Нью-Йорк: John Wiley & Sons Inc,1979. -240 с.

13. Memrise: веб-сайт. URL: https://www.memrise.com/ (дата звернення 18.04.2024)

14. Codecademy: веб-сайт. URL: https://www.codecademy.com/learn (дата звернення: 18.04.2024)

15. John Daket "HTML and CSS: Design and Build Websites", 2016. – 480 с.

16. Програмування веб-застосувань (фронт-енд та бек-енд): книга / Роман Мельник. – Львів: Львівська Політехніка, 2018. – 248 с.

17. The Art of Software Testing: довідник / Гленфорд Маєрс. – Нью-Йорк: John Wiley & Sons Inc,1979. -240 с.

18. Web-портал: «Stackoverflow»: веб-сайт. URL: https://stackoverflow.com/ (дата звернення: 18.04.2024)

19. Морзе Н.В., Бази даних у навчальному процесі. Навчальний посібник. – Київ: ТОВ Редакція “Комп’ютер”, 2007. – 120с.

20. Борис Чорний «Професійний TypeScript. Розробка масштабованих JavaScript-додатків» / М .: Видавничий дім «Print2print», 2021. - 352 с.

21. «Що таке модульне програмування?»: веб-сайт. URL: https://uk.theastrologypage.com/modular-programming (дата звернення: 23.04.2024)

22. Шевчук С. В. Українське ділове мовлення: Навч. посібник. — К.: Літера, 2003 ISBN 966-7543-25-0 – 73с.

23. Закон України "Про охорону праці" Законодавство України про охорону праці , т.1.- К. - 1995. - 558 с.

24. Державний реєстр міжгалузевих і галузевих нормативних актів про охорону праці (реєстр ДНАОП). – К.: 1998. – 240.

ДОДАТКИ

Додаток А

Код api-module

```
from django.db import models
from apps.abstract.managers import AbstractManager
class AbstractModel(models.Model):
    id = models.AutoField(primary_key=True)
    created_date = models.DateTimeField(auto_now_add=True)
    updated_date = models.DateTimeField(auto_now=True)
    objects = AbstractManager()
    class Meta:
        abstract = True

from rest_framework import generics, filters
class AbstractListCreateView(generics.ListCreateAPIView):
    filter_backends = [filters.OrderingFilter]
    ordering_fields = ["updated_date", "created_date"]
    ordering = ["-updated_date"]

from rest_framework import serializers
class AbstractSerializer(serializers.ModelSerializer):
    id = serializers.IntegerField(read_only=True)
    created_date = serializers.DateTimeField(read_only=True)
    updated_date = serializers.DateTimeField(read_only=True)

from django.db import models
from django.core.exceptions import ObjectDoesNotExist
from django.http import Http404
class AbstractManager(models.Manager):
    def get_object_by_id(self, id):
        try:
            instance = self.get(id=id)
            return instance
        except (ObjectDoesNotExist, ValueError, TypeError):
            return Http404
```

```
from rest_framework.pagination import PageNumberPagination
from rest_framework.response import Response
class ProjectPagination(PageNumberPagination):
    page_size = 6
    page_size_query_param = "page_size"
    max_page_size = 18
    def get_paginated_response(self, data):
        return Response(
            {
                "total_items": self.page.paginator.count,
                "total_pages": self.page.paginator.num_pages,
                "current": self.page.number,
                "next": self.get_next_link(),
                "previous": self.get_previous_link(),
                "results": data,
            }
        )

from django.urls import path, include
urlpatterns = [
    # /api/...
    path("auth/", include("apps.authentication.api.urls")),
    path("users/", include("apps.user.api.urls")),
    path("category/", include("apps.product.api.urls")),
    path("orders/", include("apps.order.api.urls")),
    path("saved-list/", include("apps.wishlist.api.urls")),
    path("cart/", include("apps.cart.api.urls")),
]

from django.apps import AppConfig
class AuthenticationConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "apps.authentication"
from django.contrib.auth.models import update_last_login
from rest_framework import serializers
from rest_framework_simplejwt.serializers import
TokenObtainPairSerializer
from rest_framework_simplejwt.settings import api_settings
```

```
from apps.user.models import User
from apps.user.api.serializers import UserSerializer
class UserRegistrationSerializer(serializers.ModelSerializer):
    password = serializers.CharField(write_only=True)
    class Meta:
        model = User
        fields = ["email", "password", "first_name", "last_name",
"username"]
    def create(self, validated_data):
        return User.objects.create_user(**validated_data)
class UserLoginSerializer(TokenObtainPairSerializer):
    def validate(self, attrs):
        data = super().validate(attrs)
        refresh = self.get_token(self.user)
        data["user"] = UserSerializer(self.user).data
        data["refresh"] = str(refresh)
        data["access"] = str(refresh.access_token)
        if api_settings.UPDATE_LAST_LOGIN:
            update_last_login(None, self.user)
        return data

from django.urls import path
from apps.authentication.api.views import (
    UserRegisterView,
    UserLoginView,
    TokenRefreshView,
)
urlpatterns = [
    # /api/auth/...
    path("register/", UserRegisterView.as_view()),
    path("login/", UserLoginView.as_view()),
    path("refresh/", TokenRefreshView.as_view()),
]

from rest_framework import generics, status
from rest_framework.permissions import AllowAny
from rest_framework.response import Response
from rest_framework_simplejwt.tokens import RefreshToken
from rest_framework_simplejwt.exceptions import TokenError, InvalidToken
```

```
from rest_framework_simplejwt.views import TokenRefreshView
from apps.authentication.api.serializers import (
    UserRegistrationSerializer,
    UserLoginSerializer,
)
class UserRegisterView(generics.CreateAPIView):
    serializer_class = UserRegistrationSerializer
    permission_classes = [AllowAny]
    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.save()
        refresh = RefreshToken.for_user(user)
        res = {
            "refresh": str(refresh),
            "access": str(refresh.access_token),
        }
        return Response(
            {
                "user": serializer.data,
                "refresh": res["refresh"],
                "token": res["access"],
            },
            status=status.HTTP_201_CREATED,
        )
class UserLoginView(generics.CreateAPIView):
    serializer_class = UserLoginSerializer
    permission_classes = [AllowAny]
    def create(self, request, *args, **kwargs):
        serializer = self.serializer_class(data=request.data)
        try:
            serializer.is_valid(raise_exception=True)
        except TokenError as e:
            raise InvalidToken(e.args[0])
        return                          Response(serializer.validated_data,
status=status.HTTP_200_OK)
class TokenRefreshView(generics.CreateAPIView, TokenRefreshView):
    permission_classes = [AllowAny]
```

```
    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        try:
            serializer.is_valid(raise_exception=True)
        except TokenError as e:
            raise InvalidToken(e.args[0])
        return                          Response(serializer.validated_data,
status=status.HTTP_200_OK)

from django.apps import AppConfig
class CartConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "apps.cart"

from django.db import models
from apps.abstract.models import AbstractModel
from apps.cart.managers import CartManager
class Cart(AbstractModel):
    user = models.ForeignKey(to="user.User", on_delete=models.CASCADE)
    product           =           models.ForeignKey(to="product.Product",
on_delete=models.CASCADE)
    objects = CartManager()
    def __str__(self):
        return f"Cart {self.id}"
from rest_framework import serializers
from apps.product.models import Product
from apps.cart.models import Cart
class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = Product
        fields   =   ["id",   "name",   "description",   "price",   "image",
"quantity"]
class CartSerializer(serializers.ModelSerializer):
    product_info = ProductSerializer(source="product", read_only=True)
    class Meta:
        model = Cart
        fields = [
            "id",
            "user",
```

```
            "product",
            "product_info",
            "created_date",
            "updated_date",
        ]
class CartDetailSerializer(serializers.ModelSerializer):
    product_info = ProductSerializer(source="product", read_only=True)
    quantity = serializers.IntegerField(write_only=True)
    class Meta:
        model = Cart
        fields = [
            "id",
            "user",
            "product",
            "product_info",
            "quantity",
            "created_date",
            "updated_date",
        ]
    def update(self, instance, validated_data):
        quantity = validated_data.pop("quantity", None)
        product = instance.product
        if quantity is not None:
            product.quantity = quantity
            product.save()
        return super().update(instance, validated_data)
from django.urls import path
from apps.cart.api.views import CartListView, CartDetailView
urlpatterns = [
    # /api/cart/...
    path("", CartListView.as_view(), name="api_cart_list"),
    path("<int:pk>/", CartDetailView.as_view(), name="api_cart_detail"),
]
from rest_framework import generics, permissions, status
from rest_framework.response import Response
from apps.cart.models import Cart
from apps.cart.api.serializers import CartSerializer, CartDetailSerializer
```

```
class CartListView(generics.ListCreateAPIView):
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = CartSerializer
    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        self.perform_create(serializer)
        headers = self.get_success_headers(serializer.data)
        return Response(
            serializer.data,              status=status.HTTP_201_CREATED,
headers=headers
        )
    def get_queryset(self):
        user = self.request.user
        if user.is_authenticated:
            return Cart.objects.filter(user=user)
        else:
            return Cart.objects.none()
class CartDetailView(generics.RetrieveUpdateDestroyAPIView):
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = CartDetailSerializer
    queryset = Cart.objects.all()
from django.apps import AppConfig
class OrderConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "apps.order"
from django.db import models
from apps.abstract.models import AbstractModel
from apps.order.managers import OrderManager
class Order(AbstractModel):
    STATUS_CHOICES = [
        ("Pending", "Pending"),
        ("Processing", "Processing"),
        ("Shipped", "Shipped"),
        ("Delivered", "Delivered"),
        ("Cancelled", "Cancelled"),
    ]

    user  =  models.ForeignKey(to="user.User",  on_delete=models.CASCADE,
```

```
null=True)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default="Pending")
    shipping_address = models.TextField(blank=True, null=True)
    objects = OrderManager()
    def __str__(self):
        return f"Order {self.id}"
class OrderItem(models.Model):
    order = models.ForeignKey(
        Order, related_name="order_items", on_delete=models.CASCADE
    )
    product = models.ForeignKey(to="product.Product",
on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)
    def __str__(self):
        return f"{self.product} (Quantity: {self.quantity})"

from rest_framework import serializers
from apps.order.models import Order, OrderItem
from apps.product.models import Product
class ProductSerializer(serializers.ModelSerializer):
    image = serializers.CharField()

    class Meta:
        model = Product
        fields = ["id", "name", "price", "description", "image"]
class OrderItemSerializer(serializers.ModelSerializer):
    product = ProductSerializer()
    class Meta:
        model = OrderItem
        fields = ["product", "quantity"]
    def create(self, validated_data):
        product_data = validated_data.pop("product", None)
        product, _ = Product.objects.get_or_create(**product_data)
        order_item = OrderItem.objects.create(product=product,
**validated_data)
        return order_item
```

```
class OrderSerializer(serializers.ModelSerializer):
    order_items = OrderItemSerializer(many=True)
    class Meta:
        model = Order
        fields = ["id", "user", "status", "shipping_address",
"order_items"]
    def create(self, validated_data):
        order_items_data = validated_data.pop("order_items")
        order = Order.objects.create(**validated_data)
        for order_item_data in order_items_data:
            product_data = order_item_data.pop("product", None)
            product, _ = Product.objects.get_or_create(**product_data)
            OrderItem.objects.create(order=order, product=product,
**order_item_data)
        return order
from django.urls import path
from apps.order.api.views import OrderListView, OrderDetailView
urlpatterns = [
    # /api/orders/...
    path("", OrderListView.as_view(), name="api_orders_list"),
    path("<int:pk>/", OrderDetailView.as_view(),
name="api_order_detail"),
]

from rest_framework import generics, permissions, status
from rest_framework.response import Response
from django.core.exceptions import PermissionDenied
from apps.order.models import Order
from apps.order.api.serializers import OrderSerializer
class OrderListView(generics.ListCreateAPIView):
    queryset = Order.objects.all()
    serializer_class = OrderSerializer
    permission_classes = [permissions.IsAuthenticated]
    def get_queryset(self):
        return self.queryset.filter(user=self.request.user)
    def perform_create(self, serializer):
        serializer.save(user=self.request.user)

    def post(self, request, *args, **kwargs):
```

```
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        self.perform_create(serializer)
        headers = self.get_success_headers(serializer.data)
        return Response(
            serializer.data,              status=status.HTTP_201_CREATED,
headers=headers
        )
class OrderDetailView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Order.objects.all()
    serializer_class = OrderSerializer
    permission_classes = [permissions.IsAuthenticated]
    def perform_update(self, serializer):
        instance = serializer.save()
        if instance.user != self.request.user:
            raise PermissionDenied("You do not have permission to perform
this action.")
from django.contrib import admin
from django.utils.safestring import mark_safe
from apps.product.models import Category, Product
class ImagePreviewMixin:
    def image_preview(self, obj, width=50, height=50):
        if obj.image:
            return mark_safe(
                '<img   src="{url}"   width="{width}"   height="{height}"
/>'.format(
                    url=obj.image.url,
                    width=width,
                    height=height,
                )
            )
        else:
            return "(No image)"
    image_preview.short_description = "Image Preview"
class CategoryAdmin(admin.ModelAdmin, ImagePreviewMixin):
    list_display = ["name", "image_preview"]
    readonly_fields = ("image_preview",)
class ProductAdmin(admin.ModelAdmin, ImagePreviewMixin):
    list_display = [
```

```
        "name",
        "description",
        "price",
        "quantity",
        "category",
        "image_preview",
    ]
    readonly_fields = ("image_preview",)
admin.site.register(Category, CategoryAdmin)
admin.site.register(Product, ProductAdmin)
from django.apps import AppConfig
class ProductConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "apps.product"
from django.db import models
from apps.abstract.models import AbstractModel
from apps.product.managers import ProductManager
class Category(models.Model):
    name = models.CharField(max_length=255)
    image        =        models.ImageField(null=True,        blank=True,
upload_to="category/")

    def __str__(self):
        return self.name
class Product(AbstractModel):
    name = models.CharField(max_length=255)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    image        =        models.ImageField(null=True,        blank=True,
upload_to="products/")
    quantity = models.IntegerField(default=1)
    category = models.ForeignKey(
        Category, on_delete=models.SET_NULL, null=True, blank=True
    )
    objects = ProductManager()
    def __str__(self):
        return self.name

from rest_framework import serializers
```

```
from apps.product.models import Product, Category
class CategorySerializer(serializers.ModelSerializer):
    image = serializers.ImageField(read_only=True)
    class Meta:
        model = Category
        fields = ["id", "name", "image"]
class ProductSerializer(serializers.ModelSerializer):
    category = serializers.PrimaryKeyRelatedField(
        queryset=Category.objects.all(), required=False
    )
    class Meta:
        model = Product
        fields = [
            "id",
            "name",
            "description",
            "price",
            "image",
            "quantity",
            "category",
            "created_date",
            "updated_date",
        ]
        read_only_fields = ["quantity", "category"]
    def create(self, validated_data):
        category_pk = self.context.get("view").kwargs.get("category_pk")
        try:
            category = Category.objects.get(pk=category_pk)
        except Category.DoesNotExist:
            raise serializers.ValidationError("Category matching query
does not exist.")
        product = Product.objects.create(category=category,
**validated_data)
        return product
from django.urls import path
from apps.product.api.views import (
    CategoryListView,
    CategoryDetailView,
    ProductListView,
```

```
    ProductDetailView,
)
urlpatterns = [
    # /api/category/...
    path("", CategoryListView.as_view(), name="api_categories_list"),
    path("<int:pk>/",                        CategoryDetailView.as_view(),
name="api_category_detail"),
    path(
        "<int:category_pk>/products/",
        ProductListView.as_view(),
        name="api_products_list",
    ),
    path(
        "<int:category_pk>/products/<int:pk>/",
        ProductDetailView.as_view(),
        name="api_product_detail",
    ),
]

from django.shortcuts import get_object_or_404
from rest_framework import generics, permissions
from apps.product.models import Product, Category
from     apps.product.api.serializers     import     ProductSerializer,
CategorySerializer
class CategoryListView(generics.ListCreateAPIView):
    """
    Browse to get a list of all product categories or create a new category.
    """
    permission_classes = (permissions.AllowAny,)
    serializer_class = CategorySerializer
    queryset = Category.objects.all()
class CategoryDetailView(generics.RetrieveUpdateDestroyAPIView):
    """
    View to get, update, or delete information about a product category.
    """
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = CategorySerializer
    queryset = Category.objects.all()
class ProductListView(generics.ListCreateAPIView):
```

```
    """
    View to get a list of all products or create a new product.
    """
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = ProductSerializer
    def get_queryset(self):
        """
        Returns a list of all products that match the filter by category.
        """
        category_pk = self.kwargs.get("category_pk")
        category = get_object_or_404(Category, pk=category_pk)
        return Product.objects.filter(category=category)
class ProductDetailView(generics.RetrieveUpdateDestroyAPIView):
    """
    View to get, update, or delete product information.
    """
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = ProductSerializer
    queryset = Product.objects.all()

from django.apps import AppConfig
class UserConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "apps.user"
from django.contrib.auth.base_user import BaseUserManager
from apps.abstract.managers import AbstractManager
class UserManager(BaseUserManager, AbstractManager):
    use_in_migrations = True
    def _create_user(self, email, password, **extra_fields):
        """
        Creates and saves a User with the given email and password.
        """
        if not email:
            raise ValueError("The given email must be set")
        email = self.normalize_email(email)
        cleared_extra_fields = {
            k: v for k, v in extra_fields.items() if hasattr(self.model,
k)
        }
```

```
        user = self.model(email=email, **cleared_extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user
    def create_user(self, email, password=None, **extra_fields):
        extra_fields.setdefault("is_superuser", False)
        extra_fields.setdefault("is_staff", False)
        return self._create_user(email, password, **extra_fields)
    def create_superuser(self, email, password, **extra_fields):
        extra_fields.setdefault("is_superuser", True)
        extra_fields.setdefault("is_staff", True)
        if extra_fields.get("is_superuser") is not True:
            raise ValueError("Superuser must have is_superuser=True.")
        return self._create_user(email, password, **extra_fields)

from django.contrib.auth.models import AbstractBaseUser, PermissionsMixin
from django.db import models

from apps.abstract.models import AbstractModel
from apps.user.managers import UserManager
class User(AbstractModel, AbstractBaseUser, PermissionsMixin):
    username    =    models.CharField(db_index=True,    max_length=255,
unique=True)
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    email = models.EmailField(db_index=True, unique=True)
    bio = models.TextField(null=True)
    avatar = models.ImageField(null=True)

    wishlist      =      models.ManyToManyField(to="wishlist.Wishlist",
related_name="users")
    is_active = models.BooleanField(default=True)
    is_superuser = models.BooleanField(default=False)
    is_staff = models.BooleanField(default=False)
    is_admin = models.BooleanField(default=False)
    is_moderator = models.BooleanField(default=False)

    objects = UserManager()
    USERNAME_FIELD = "email"
```

```
    REQUIRED_FIELDS = ["username"]
    class Meta:
        verbose_name = "user"
        verbose_name_plural = "users"
    def __str__(self):
        return f"{self.email}"
    @property
    def is_privileged(self):
        return self.is_moderator or self.is_admin
    @property
    def name(self):
        return f"{self.first_name} {self.last_name}"
    def add_to_wishlist(self, product):
        """Like `product` if it hasn't been done yet"""
        return self.wishlist.add(product)
    def remove_from_wishlist(self, product):
        """Remove a like from a `product`"""
        return self.wishlist.remove(product)
    def has_added_to_wishlist(self, product):
        """Return True if the user has liked a `product`; else False"""
        return self.wishlist.filter(pk=product.pk).exists()

from apps.abstract.serializers import AbstractSerializer
from apps.user.models import User
class UserSerializer(AbstractSerializer):
    class Meta:
        model = User
        fields = [
            "id",
            "username",
            "first_name",
            "last_name",
            "email",
            "is_active",
            "created_date",
            "updated_date",
        ]
        read_only_field = ["is_active"]
class UserDetailSerializer(AbstractSerializer):
```

```
    class Meta:
        model = User
        fields = [
            "id",
            "username",
            "first_name",
            "last_name",
            "bio",
            "avatar",
            "email",
            "is_active",
            "created_date",
            "updated_date",
        ]
        read_only_field = ["is_active"]
from django.shortcuts import get_object_or_404
from rest_framework import generics
from apps.api.pagination import ProjectPagination
from apps.abstract.views import AbstractListCreateView
from apps.user.models import User
from apps.user.api.serializers import UserSerializer, UserDetailSerializer
class UsersListView(AbstractListCreateView):
    """
    List of users.
    """
    queryset = User.objects.filter(is_superuser=False).order_by("id")
    serializer_class = UserSerializer
    pagination_class = ProjectPagination
class UserDetailView(generics.RetrieveUpdateDestroyAPIView):
    """
    Retrieve, update, or delete a user by ID.
    """
    serializer_class = UserDetailSerializer
    pagination_class = ProjectPagination
    def get_object(self):
        id = self.kwargs.get("id")
        queryset = User.objects.filter(is_superuser=False)
        user = get_object_or_404(queryset, id=id)
        return user
```

```
from django.db import models
from apps.abstract.models import AbstractModel
class Wishlist(AbstractModel):
    user = models.ForeignKey(
        to="user.User",                          on_delete=models.CASCADE,
related_name="wishlists_items"
    )
    products = models.ForeignKey(
        to="product.Product",                    on_delete=models.CASCADE,
related_name="wishlists"
    )
    def __str__(self):
        return f"Wishlist of {self.user.username}"

from django.apps import AppConfig
class WishlistConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "apps.wishlist"
from rest_framework import permissions, generics, status
from rest_framework.response import Response

from apps.product.models import Product
from apps.wishlist.models import Wishlist
from apps.wishlist.api.serializers import WishlistSerializer


class SavedProductsList(generics.ListCreateAPIView):
    """
    List of saved products.
    Add a product to the saved list.
    """

    permission_classes = [permissions.IsAuthenticated]
    serializer_class = WishlistSerializer

    def get_queryset(self):
        user = self.request.user
        return Wishlist.objects.filter(user=user)
```

```
    def create(self, request, *args, **kwargs):
        user = request.user
        product_id = request.data.get("products")

        if not product_id:
            return Response(
                {"error": "Product ID must be provided."},
                status=status.HTTP_400_BAD_REQUEST,
            )

        product = Product.objects.filter(id=product_id).first()
        if not product:
            return Response(
                {"error": "Product with provided ID does not exist."},
                status=status.HTTP_400_BAD_REQUEST,
            )

        if Wishlist.objects.filter(user=user, id=product_id).exists():
            return Response(
                {"message": "Product already saved."},
                status=status.HTTP_200_OK,
            )

        wishlist_item        =        Wishlist.objects.create(user=user,
products=product)
        serializer = WishlistSerializer(wishlist_item)
        return Response(serializer.data, status=status.HTTP_201_CREATED)
class SavedProductsDestroy(generics.DestroyAPIView):
    """
    Remove the product from the saved list.
    """
    permission_classes = [permissions.IsAuthenticated]

    def destroy(self, request, *args, **kwargs):
        user = request.user
        wishlist_item_id = kwargs.get(
            "pk"
        )  # Отримати ідентифікатор запису списку бажань з URL-адреси
```

```
        if not wishlist_item_id:
            return Response(
                {"error": "Wishlist item ID must be provided."},
                status=status.HTTP_400_BAD_REQUEST,
            )
        wishlist_item    =    Wishlist.objects.filter(id=wishlist_item_id,
user=user).first()
        if not wishlist_item:
            return Response(
                {"error": "Wishlist item not found."},
                status=status.HTTP_400_BAD_REQUEST,
            )
        wishlist_item.delete()
        return Response(
            {"message": "Wishlist item removed from the saved list."},
            status=status.HTTP_204_NO_CONTENT,
        )

from django.urls import path

from       apps.wishlist.api.views       import       SavedProductsList,
SavedProductsDestroy
urlpatterns = [
    # /api/saved-list/...
    path(
        "",
        SavedProductsList.as_view(),
        name="saved_companies_create",
    ),
    path(
        "<int:pk>/",
        SavedProductsDestroy.as_view(),
        name="saved_companies_destroy",
    ),
]

from rest_framework import serializers
from apps.product.models import Product
from apps.product.api.serializers import ProductSerializer
```

```
from apps.wishlist.models import Wishlist
class WishlistSerializer(serializers.ModelSerializer):
    products = serializers.PrimaryKeyRelatedField(queryset=Product.objects.all())
    class Meta:
        model = Wishlist
        fields = ["id", "user", "products", "created_date", "updated_date"]
    def to_representation(self, instance):
        representation = super().to_representation(instance)
        product_data = ProductSerializer(instance.products).data
        representation["products"] = product_data
        return representation
```

Додаток Б

ui-module

```
import React, { useState } from "react";
import { Form, Button } from "react-bootstrap";
import { useUserActions } from "../../hooks/user.actions";
function LoginForm() {
  const [validated, setValidated] = useState(false);
  const [form, setForm] = useState({
    email: "",
    password: "",
  });
  const [error, setError] = useState(null);
  const userActions = useUserActions();

  const handleSubmit = (event) => {
    event.preventDefault();
    const loginForm = event.currentTarget;
    if (loginForm.checkValidity() === false) {
      event.stopPropagation();
    }
    setValidated(true);
    const data = {
      email: form.email,
```

```
    password: form.password,
  };
  userActions.login(data).catch((err) => {
    if (err.message) {
      setError(err.request.response);
    }
  });
};
return (
  <Form
    id="registration-form"
    className="border p-4 rounded"
    noValidate
    validated={validated}
    onSubmit={handleSubmit}
  >
    <Form.Group className="mb-3">
      <Form.Label>Email</Form.Label>
      <Form.Control
        value={form.username}
        onChange={(e) => setForm({ ...form, email: e.target.value })}
        required
        type="text"
        placeholder="Enter email"
      />
      <Form.Control.Feedback type="invalid">
        This file is required.
      </Form.Control.Feedback>
    </Form.Group>
    <Form.Group className="mb-3">
      <Form.Label>Password</Form.Label>
      <Form.Control
        value={form.password}
        minLength="8"
        onChange={(e) => setForm({ ...form, password: e.target.value })}
        required
        type="password"
        placeholder="Password"
      />
```

```
        <Form.Control.Feedback type="invalid">
          Please provide a valid password.
        </Form.Control.Feedback>
      </Form.Group>
      <div      className="text-content      text-danger">{error      &&
<p>{error}</p>}</div>
      <Button variant="primary" type="submit">
        Submit
      </Button>
    </Form>
  );
}
export default LoginForm;

import React, { useState } from "react";
import { Form, Button } from "react-bootstrap";
import { useUserActions } from "../../hooks/user.actions";
function RegistrationForm() {
  const [validated, setValidated] = useState(false);
  const [form, setForm] = useState({
    username: "",
    email: "",
    password: "",
    first_name: "",
    last_name: "",
    bio: "",
  });
  const [error, setError] = useState(null);
  const userActions = useUserActions();
  const handleSubmit = (event) => {
    event.preventDefault();
    const registrationForm = event.currentTarget;

    if (registrationForm.checkValidity() === false) {
      event.stopPropagation();
    }
    setValidated(true);
    const data = {
      username: form.username,
```

```
      password: form.password,
      email: form.email,
      first_name: form.first_name,
      last_name: form.last_name,
      bio: form.bio,
    };
    userActions.register(data).catch((err) => {
      if (err.message) {
        setError(err.request.response);
      }
    });
  };
  return (
    <Form
      id="registration-form"
      className="border p-4 rounded"
      noValidate
      validated={validated}
      onSubmit={handleSubmit}
    >
      <Form.Group className="mb-3">
        <Form.Label>First Name</Form.Label>
        <Form.Control
          value={form.first_name}
          onChange={(e) => setForm({ ...form, first_name: e.target.value
})}
          required
          type="text"
          placeholder="Enter first name"
        />
        <Form.Control.Feedback type="invalid">
          This file is required.
        </Form.Control.Feedback>
      </Form.Group>
      <Form.Group className="mb-3">
        <Form.Label>Last name</Form.Label>
        <Form.Control
          value={form.last_name}
          onChange={(e) => setForm({ ...form, last_name: e.target.value
```

```
})}
            required
            type="text"
            placeholder="Enter last name"
          />
          <Form.Control.Feedback type="invalid">
            This file is required.
          </Form.Control.Feedback>
        </Form.Group>
        <Form.Group className="mb-3">
          <Form.Label>Username</Form.Label>
          <Form.Control
            value={form.username}
            onChange={(e) => setForm({ ...form, username: e.target.value })}
            required
            type="text"
            placeholder="Enter username"
          />
          <Form.Control.Feedback type="invalid">
            This file is required.
          </Form.Control.Feedback>
        </Form.Group>
        <Form.Group className="mb-3">
          <Form.Label>Email address</Form.Label>
          <Form.Control
            value={form.email}
            onChange={(e) => setForm({ ...form, email: e.target.value })}
            required
            type="email"
            placeholder="Enter email"
          />
          <Form.Control.Feedback type="invalid">
            Please provide a valid email.
          </Form.Control.Feedback>
        </Form.Group>
        <Form.Group className="mb-3">
          <Form.Label>Password</Form.Label>
          <Form.Control
            value={form.password}
```

```
          minLength="8"
          onChange={(e) => setForm({ ...form, password: e.target.value })}
          required
          type="password"
          placeholder="Password"
        />
        <Form.Control.Feedback type="invalid">
          Please provide a valid password.
        </Form.Control.Feedback>
      </Form.Group>
      <Form.Group className="mb-3">
        <Form.Label>Bio</Form.Label>
        <Form.Control
          value={form.bio}
          onChange={(e) => setForm({ ...form, bio: e.target.value })}
          as="textarea"
          rows={3}
          placeholder="A simple bio ... (Optional)"
        />
      </Form.Group>
      <div      className="text-content      text-danger">{error      &&
<p>{error}</p>}</div>
      <Button variant="primary" type="submit">
        Submit
      </Button>
    </Form>
  );
}
export default RegistrationForm;
import React from "react";
import { Navbar, Container, Image, NavDropdown, Nav } from "react-
bootstrap";
import { useNavigate, Link } from "react-router-dom";
import { FaShoppingCart } from "react-icons/fa";
import { randomAvatar } from "../utils";
function Navigationbar() {
  const navigate = useNavigate();
  const handleLogout = () => {
    localStorage.removeItem("auth");
```

```
    navigate("/login/");
  };
  const handleCartClick = () => {
    navigate("/cart/");
  };
  return (
    <Navbar bg="primary" variant="dark">
      <Container>
        <Navbar.Brand className="fw-bold" href="/">
          MyShop
        </Navbar.Brand>
        <Navbar.Collapse className="justify-content-end">
          <Nav>
            <Nav.Link onClick={handleCartClick}>
              <FaShoppingCart size={36} />
            </Nav.Link>
            <NavDropdown
              title={
                <Image
                  src={randomAvatar()}
                  roundedCircle
                  width={36}
                  height={36}
                />
              }
            >
              <NavDropdown.Item as={Link} to="/profile">
                Profile
              </NavDropdown.Item>
              <NavDropdown.Item as={Link} to="/saved-list">
                Wish list
              </NavDropdown.Item>
              <NavDropdown.Item
onClick={handleLogout}>Logout</NavDropdown.Item>
            </NavDropdown>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
```

```
  );
}
export default Navigationbar;
import React from "react";
import { Card, Button, Image } from "react-bootstrap";
import { randomAvatar } from "../utils";
function Profile(props) {
  const { user } = props;
  const handleNavigateToProfile = () => {
    // Navigate to the profile page
  };
  return (
    <Card className="border-0 p-2">
      <div className="d-flex ">
        <Image
          src={randomAvatar()}
          roundedCircle
          width={48}
          height={48}
          className="my-3 border border-primary border-2"
        />
        <Card.Body>
          <Card.Title className="fs-6">{user.name}</Card.Title>
          <Button variant="primary" onClick={handleNavigateToProfile}>
            See profile
          </Button>
        </Card.Body>
      </div>
    </Card>
  );
}
export default Profile;

import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import { Table, Button, Form } from "react-bootstrap";
import useSWR, { mutate } from "swr";
import { fetcher } from "../helpers/axios";
import { getUser, getAccessToken } from "../hooks/user.actions";
```

```
import Layout from "../components/Layout";
import BackButton from "../components/BackButton";
import Error from "../components/Error";
import Loading from "../components/Loading";

function Cart() {
  const { data: cartData, error: cartError } = useSWR(`/cart/`, fetcher);
  const [loading, setLoading] = useState(true);
  const [totalPrice, setTotalPrice] = useState(0);

  useEffect(() => {
    if (cartData) {
      setLoading(false);

      let total = 0;
      cartData.forEach((cartItem) => {
        if (
          cartItem.product_info &&
          cartItem.product_info.price &&
          cartItem.product_info.quantity
        ) {
          total          +=          cartItem.product_info.price          *
cartItem.product_info.quantity;
        }
      });
      setTotalPrice(total);
    }
  }, [cartData]);

  const handleDeleteItem = async (id) => {
    try {
      const url = `http://localhost:8000/api/cart/${id}/`;
      console.log("Sending request to:", url);
      const response = await fetch(url, {
        method: "DELETE",
        headers: {
          "Content-Type": "application/json",
          Authorization: `Bearer ${getAccessToken()}`,
        },
```

```
      body: JSON.stringify({
        user: getUser().id,
        product: id,
      }),
    });
    mutate("/cart/");
  } catch (error) {
    console.error("Error deleting item:", error);
  }
};

const handleQuantityChange = async (id, quantity) => {
  try {
    const url = `http://localhost:8000/api/cart/${id}/`;
    console.log("Sending request to:", url);
    const response = await fetch(url, {
      method: "PATCH",
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${getAccessToken()}`,
      },
      body: JSON.stringify({
        quantity: quantity,
      }),
    });
    mutate("/cart/");
  } catch (error) {
    console.error("Error updating quantity:", error);
  }
};

const handleCheckout = async () => {
  try {
    if (!cartData || cartData.length === 0) {
      console.error("No items in the cart");
      return;
    }

    const orderItems = cartData.map((item) => ({
```

```
      product: {
        name: item.product_info.name,
        description: item.product_info.description,
        price: item.product_info.price,
        image: item.product_info.image,
        quantity: item.quantity,
      },
      quantity: item.quantity,
    }));

    const payload = {
      user: getUser().id,
      status: "Pending",
      shipping_address: "",
      order_items: orderItems,
    };

    const url = "http://localhost:8000/api/orders/";
    const response = await fetch(url, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${getAccessToken()}`,
      },
      body: JSON.stringify(payload),
    });

    if (response.ok) {
    } else {
      console.error("Error creating order:", response.statusText);
    }
  } catch (error) {
    console.error("Error creating order:", error);
  }
};

if (cartError) {
  return <Error message="Error fetching cart!" />;
}
```

```
  if (loading) {
    return <Loading />;
  }

  return (
    <Layout>
      <div className="d-flex align-items-center mb-3">
        <BackButton />
      </div>
      <h2 className="text-center mb-4">Shopping Cart</h2>
      {cartData && cartData.length > 0 ? (
        <>
          <Table striped bordered hover>
            <thead>
              <tr>
                <th>Name</th>
                <th>Quantity</th>
                <th>Price</th>
                <th>Product</th>
                <th>Action</th>
              </tr>
            </thead>
            <tbody>
              {cartData.map((cartItem) => (
                <tr key={cartItem.id}>
                  <td>
                    <Link

to={`/category/${cartItem.id}/products/${cartItem.product_info.id}`}
                    >
                      {cartItem.product_info.name}
                    </Link>
                  </td>
                  <td>
                    <Form.Control
                      type="number"
                      value={cartItem.quantity}
                      onChange={(e) =>
```

```
                        handleQuantityChange(cartItem.id, e.target.value)
                      }
                      style={{ width: "60px" }}
                      placeholder={cartItem.product_info.quantity}
                    />
                  </td>
                  <td>
                    {cartItem.product_info.price &&
                      cartItem.product_info.quantity &&
                      (
                        cartItem.product_info.price *
                        cartItem.product_info.quantity
                      ).toFixed(2)}
                  </td>
                  <td>
                    <Link

to={`/category/${cartItem.id}/products/${cartItem.product_info.id}`}
                      >
                        <img
                          src={cartItem.product_info.image}
                          alt={cartItem.product_info.name}
                          style={{ width: "50px", height: "50px" }}
                        />
                      </Link>
                  </td>
                  <td>
                    <Button
                      variant="danger"
                      onClick={() => handleDeleteItem(cartItem.id)}
                    >
                      Delete
                    </Button>
                  </td>
                </tr>
              ))}
            </tbody>
          </Table>
          <div className="text-center mt-4">
```

```
            <p>Total Price: {totalPrice.toFixed(2)} грн</p>
            <Link to="/order/">
              <Button variant="primary" onClick={handleCheckout}>
                Proceed to Checkout
              </Button>
            </Link>
          </div>
        </>
      ) : (
        <div className="text-center mt-5">
          <h3>No products in the cart</h3>
        </div>
      )}
    </Layout>
  );
}
export default Cart;
```

метадані

Заголовок

Розробка сайту для продажу побутової техніки

Автор
Білецький А. Т.

Науковий керівник / Експерт
кандидат технічних наук Сергій Ващишак

підрозділ
King Danylo University

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	2

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1

КП 2

25
Довжина фрази для коефіцієнта подібності 2

5313
Кількість слів

40531
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/386/%D0%9A%D0%92%D0%90%D0%9B%D0%86%D0%A4%D0%86%D0%9A%D0%90%D0%A6%D0%86%D0%98%CC%86%D0%9D%D0%90%20%D0%A0%D0%9E%D0%91%D0%9E%D0%A2%D0%90.pdf?sequence=1	15	0.28 %
2	http://repository.ukd.edu.ua/bitstream/handle/123456789/396/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D0%A1%D1%82%D1%80%D1%96%D0%BB%D0%B5%D1%86%D1%8C%D0%BA%D0%B8%D0%B9.pdf?sequence=1	13	0.24 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/396/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D0%A1%D1%82%D1%80%D1%96%D0%BB%D0%B5%D1%86%D1%8C%D0%BA%D0%B8%D0%B9.pdf?sequence=1	12	0.23 %