НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
“___”_____202__р.

Дипломна робота
на здобуття ступеня бакалавра

За освітньо-професійною програмою “Цифрові технології в енергетиці”
Спеціальності 122 “Комп’ютерні науки”
на тему: Платформа для електронної комерції (frontend)_____
_____
_____

Виконала: студентка 4 курсу, групи ТР-11
_____БРАТЧИКОВА Наталія Віталіївна_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: професор д.т.н., доц. Шушура Олексій Миколайович _____
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові) (підпис)

Рецензент: ____професор кафедри інженерії програмного забезпечення в енергетиці, д.т.н, професор ГАВРИЛКО Євген Володимирович___ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім’я, по батькові) (підпис)

Н.контроль ст.викладач Ольга БЕСПАЛА _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____Наталія АУШЕВА
«__»_____202__р.

ЗАВДАННЯ
на дипломну роботу студентці

БРАТЧИКОВІЙ Наталії Віталіївні
(прізвище, ім’я, по батькові)

1. Тема роботи **Платформа для електронної комерції (frontend)**
Науковий керівник ***Шушура Олексій Миколайович, професор, д.т.н., доцент***
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «02» червня 2025 р. № 1875-с
2. Термін подання роботи студентом 09.06.2025

3. Вихідні дані до роботи мова програмування javascript, утиліта typescript, бібліотека React, бібліотека Material UI, бібліотека React Router, бібліотека axios, бібліотека Redux.

4. Зміст роботи

1) Провести дослідження серед наявних аналогів
2) Аргументувати обрані засоби для реалізації програмного продукту
3) Реалізація взаємодії користувача із обліковими даними
4) Розробити інтерфейс клієнтської частини застосунку платформи для електронної комерції
5) Презентувати створений продукт

6. Орієнтовний перелік графічного (ілюстративного) матеріалу схема життєвого циклу об’єкту користувача клієнтської частини, діаграма прецедентів, приклади роботи користувача з системою.

7. Дата видачі завдання 13.09.2024

Календарний план

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	13.09.24	Виконано
2.	Постановка та вивчення задач	15 – 17.04.2025	Виконано
3.	Реалізація взаємодії користувача із обліковими даними	18.04 – 25.04.2025	Виконано
4.	Реалізація взаємодії із товарами, акаунтом продавця, реалізації функціональних можливостей продажу товарів звичайним користувачем	26.04 – 06.05.2025	Виконано
5.	Оформлення пояснювальної записки	07 – 13.05.2025	Виконано
6.	Захист програмного продукту	14.05.2025	Виконано
7.	Предзахист	27.05.2025	Виконано
8.	Захист	16 – 20.06.2025	Виконано

Студент _____ Наталія БРАТЧИКОВА
(підпис) (ім’я ПРІЗВИЩЕ)

Керівник роботи _____ Олексій ШУШУРА____
(підпис) (ім’я ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 59 сторінках, містить 32 ілюстрацій, 2 додатків, 20 джерел у переліку посилань.

Мета роботи – розробка функціональної та зручної клієнтської частини e-commerce застосунку з урахуванням ролей користувачів, реалізацією таких функціональних можливостей, як перегляду, створення, редагування товарів, оформлення замовлень і взаємодії з особистим кабінетом.

Методи та засоби: компонентний підхід, архітектура FSD, мова програмування JavaScript, бібліотека React, Redux для управління станом, React Router для маршрутизації, бібліотека компонентів Material UI, асинхронна взаємодія через REST API, адаптивна верстка з урахуванням кросбраузерності.

Результат – клієнтський застосунок, що забезпечує інтуїтивно зрозумілий інтерфейс, зручний доступ до каталогу товарів, можливість керування замовленнями та ефективну взаємодію з серверною частиною.

Ключові слова: E-COMMERCE, FRONTEND, REACT, USER INTERFACE, SPA, КОРИСТУВАЦЬКИЙ ДОСВІД.

ABSTRACT

The thesis consists of 59 pages, includes 32 illustrations, 2 appendixes, and 20 references in the bibliography.

The aim of the work is development of a functional and convenient client part of the e-commerce application taking into account user roles, implementation of functionality for viewing, creating, editing products, placing orders, and interacting with the personal account.

Methods and tools: component-based approach, FSD architecture, JavaScript programming language, React library, Redux for state management, React Router for routing, Material UI component library, asynchronous interaction via REST API, and responsive layout with cross-browser support.

Result – a client-side application that provides an intuitive user interface, convenient access to the product catalog, order management functionality, and effective interaction with the server side.

Keywords: E-COMMERCE, FRONTEND, REACT, USER INTERFACE, SPA, USER EXPERIENCE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8

ВСТУП 9

1 ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ E-COMMERCE ЗАСТОСУНКУ 11

2 АНАЛІЗ СУЧАСНИХ РІШЕНЬ ДЛЯ РЕАЛІЗАЦІЇ FRONTEND ЧАСТИНИ ЗАСТОСУНКУ З ЕЛЕКТРОННОЇ КОМЕРЦІЇ 14

2.1 Огляд готових рішень для застосунків з електронної комерції 14

2.2 Огляд популярних платформ для електронної комерції в Україні. Порівняння їх показників 18

2.3 Загальні висновки по існуючим рішенням методів розробки платформ з електронної комерції. Аналіз місць для покращення 20

3 ВИБІР ТЕХНОЛОГІЙ ТА ПІДХОДІВ ДЛЯ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ E-COMMERCE ПЛАТФОРМ 23

3.1 Надбудова TypeScript 23

3.2. Бібліотека React 24

3.3 Технологія для контролю станів Redux 26

3.4 Бібліотека axios 27

3.5 Бібліотека React Route 27

3.6 Бібліотека Material UI 28

3.7 Архітектура FSD для структуризації коду 29

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ 32

4.1 Функціональні аспекти програми 32

4.2. Технічна реалізація програми 36

5 ВЗАЄМОДІЯ КОРИСТУВАЧА З ПРОДУКТОМ..... 41
ВИСНОВКИ..... 58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 60
ДОДАТОК А..... 62
ДОДАТОК Б..... 68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AJAX – це технологія, яка дозволяє здійснювати асинхронний обмін даними з сервером без перезавантаження вебсторінки.

API (Application Programming Interface) – інтерфейс прикладного програмування, який дозволяє різним програмам або сервісам взаємодіяти між собою.

Backend – це серверна частина веб-застосунку, яка відповідає за бізнес-логіку, обробку запитів, роботу з базами даних та взаємодію з API.

E-commerce – е лектронна комерція, тобто процес купівлі та продажу товарів і послуг через інтернет.

Fetch – це вбудований метод JavaScript, який дозволяє надсилати HTTP-запити до сервера для отримання або надсилання даних у форматі JSON.

Frontend – це клієнтська частина веб-застосунку, з якою безпосередньо взаємодіє користувач.

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту, що використовується для обміну інформацією між веб-сервером і браузером користувача.

JSON (JavaScript Object Notation) – це легкий текстовий формат обміну даними, який легко читається як людиною, так і машиною.

Open-source – програмне забезпечення з відкритим кодом, який є доступним для перегляду, змін і вільного використання будь-ким.

SaaS-platforms (Software as a Service) – платформи, які надають доступ до програмного забезпечення через інтернет на основі підписки.

TS (TypeScript) – мова програмування, що є надбудовою над JavaScript, додає статичну типізацію.

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій, цифрової трансформації бізнесу та зростання впливу інтернету на економічні процеси, електронна комерція (e-commerce) посідає одне з провідних місць серед усіх напрямів господарської діяльності. Вона охоплює широкий спектр рішень – від онлайн-магазинів до комплексних платформ, що забезпечують автоматизацію бізнес-процесів, аналітику, логістику та персоналізовану взаємодію з клієнтом. У зв’язку з цим зростає потреба у створенні високоефективних, масштабованих, безпечних та орієнтованих на користувача e-commerce застосунків, які здатні відповідати сучасним вимогам як з технічної, так і з функціональної точки зору[1].

Клієнтська частина таких застосунків (frontend) відіграє ключову роль у забезпеченні позитивного досвіду взаємодії користувача з системою. Вона є тією складовою, через яку відбувається безпосередній контакт із клієнтом – саме її зручність, швидкодія, візуальна привабливість і логіка роботи часто визначають загальне враження користувача від роботи з платформою. Успішна реалізація інтерфейсу дозволяє не лише підвищити рівень задоволеності клієнтів, а й сприяє досягненню стратегічних бізнес-цілей компанії: збільшенню конверсій, лояльності користувачів, обсягу продажів і зміцненню конкурентних позицій на ринку. У зв’язку з цим особливої актуальності набуває розробка сучасних клієнтських застосунків з використанням новітніх технологій, архітектурних підходів та бібліотек, які забезпечують швидку, надійну та адаптивну роботу системи [2].

Актуальність теми обумовлюється необхідністю розробки інтерфейсів, які відповідають сучасним вимогам до функціональності, продуктивності, адаптивності безпеки та наданням користувачам того набору функціональних можливостей, які задовольнятимуть всі потреби користувача та позитивно впливатиме на користувацький досвід. Окрім того, особливого значення

набувають питання оптимізації процесів обміну даними між клієнтською та серверною частинами, інтеграції з зовнішніми сервісами, забезпечення безперервної роботи системи та підвищення її масштабованості.

Метою даної роботи є розробка функціональної та зручної клієнтської частини e-commerce застосунку з урахуванням ролей користувачів (покупців і продавців), забезпечення можливості створення, перегляду, редагування товарів, управління замовленнями та підвищення загального рівня взаємодії користувача з інтерфейсом системи. Для досягнення поставленої мети було обрано сучасний стек технологій, який дозволяє реалізувати всі ключові функції з належним рівнем продуктивності та масштабованості.

Таким чином, постановка задачі розробки клієнтської частини e-commerce застосунку є важливою практичною проблемою, вирішення якої сприяє розвитку галузі веб-розробки та задоволенню актуальних потреб ринку електронної торгівлі.

1 ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ E-COMMERCE ЗАСТОСУНКУ

Розвиток електронної комерції вимагає створення сучасних, надійних та високопродуктивних веб-застосунків, які не лише забезпечують базову функціональність замовлення товарів чи послуг, але й відповідають високим очікуванням користувачів щодо якості взаємодії. Це включає забезпечення комфортного, швидкого та інтуїтивно зрозумілого інтерфейсу, адаптивного дизайну, високої швидкості завантаження сторінок, доступності на різних пристроях, а також безперебійної роботи навіть під великим навантаженням.

Клієнтська частина e-commerce застосунку виконує роль основного засобу взаємодії між користувачем і системою, формуючи перше враження про платформу та впливаючи на загальний рівень задоволеності користувачів. Вона повинна не лише виглядати привабливо, але й функціонувати без збоїв, бути зручною в навігації та дозволяти легко реалізовувати складні бізнес-процеси. Тому її розробка є критично важливим етапом загальної архітектури проєкту, який вимагає глибокого аналізу, продуманого планування та якісного технічного виконання[2].

Мета розробки полягає у створені клієнтської частини застосунку, що дозволяє користувачам здійснювати пошук товарів, переглядати їх характеристики, додавати товари до кошика, оформлювати замовлення та взаємодіяти з особистим кабінетом у зручній, безпечній та ефективній формі.

Основні вимоги до клієнтської частини наступні:

- інтуїтивно зрозумілий інтерфейс – зрозуміла навігація, логічна структура сторінок, адаптивність під різні пристрої (десктопи, планшети, смартфони);

- висока продуктивність, а саме швидке завантаження сторінок, миттєва реакція на дії користувача завдяки оптимізації ресурсів та використанню сучасних технологій асинхронної взаємодії (ajax);
- інтеграція з серверною частиною: реалізація обміну даними через API для отримання інформації про товари, обробки замовлень, реєстрації та автентифікації користувачів;
- захист персональних даних користувачів при реєстрації, автентифікації та оформленні замовлень;
- можливість подальшого масштабування без істотних змін у базовій архітектурі застосунку.

До функціональних вимог клієнтської частини застосунку можна віднести наступні пункти:

- каталог товарів із можливістю сортування;
- сторінка детального перегляду товару;
- пошук товарів;
- кошик для управління замовленнями;
- оформлення замовлення із вибором способу доставки та оплати;
- реєстрація, авторизація та особистий кабінет користувача;
- управління профілем користувача, історія замовлень.

Технологічні вимоги платформи:

- використання сучасних фреймворків та бібліотек – React для побудови інтерфейсу;
- забезпечення state-менеджменту за допомоги Redux;
- реалізація маршрутизації всередині застосунку із використанням React Router;
- інтеграція зовнішніх бібліотек для підвищення якості інтерфейсу – Material UI;
- використання методології компонентного підходу для підтримки чистоти коду і можливості масштабування – FSD архітектури;

− аналіз сучасних рішень для реалізації frontend частини e-commerce застосунків.

Дотримання викладених вище вимог дозволить розробити клієнтський застосунок, що забезпечить зручний доступ до всіх функціональних можливостей магазину, позитивний користувацький досвід, високу швидкість роботи та надійний обмін даними з серверною частиною.

Успішна реалізація клієнтської частини напряму впливає на конкурентоспроможність інтернет-магазину, оскільки саме інтерфейс є першим і головним інструментом взаємодії з користувачем. Наявність усіх необхідних функцій, швидка та безперебійна робота, естетичний і доступний інтерфейс підвищують довіру до ресурсу, сприяють зростанню конверсій, зменшують показник відмов та забезпечують лояльність клієнтів. Тому якісний frontend є не лише технічною складовою, а й стратегічною перевагою в умовах стрімкого розвитку цифрової торгівлі.

2 АНАЛІЗ СУЧАСНИХ РІШЕНЬ ДЛЯ РЕАЛІЗАЦІЇ FRONTEND ЧАСТИНИ ЗАСТОСУНКУ З ЕЛЕКТРОННОЇ КОМЕРЦІЇ

У сучасному світі електронної комерції клієнтська частина застосунку відіграє ключову роль у формуванні користувацького досвіду та залученні клієнтів. Вибір технологій і підходів до розробки інтерфейсу напряму впливає на швидкодість, зручність використання та адаптивність платформи. У цьому розділі буде проведено аналіз сучасних рішень для реалізації клієнтської частини e-commerce застосунків, зокрема популярних платформ, готових рішень та інструментів, які забезпечують ефективну та масштабовану розробку.

2.1 Огляд готових рішень для застосунків з електронної комерції

Розробка застосунків для електронної комерції є однією з найбільш поширених та затребуваних сфер сучасної веб-розробки. Це пов’язано з тим, що все більше бізнесів переходять в онлайн-простір і потребують зручних, функціональних та масштабованих рішень для продажу товарів і послуг. З цією метою на ринку представлена велика кількість готових рішень та платформ, які відрізняються за функціональністю, можливостями кастомізації, архітектурними підходами та технологічним стеком. Завдяки такому розмаїттю розробники мають змогу обирати оптимальні інструменти для конкретних завдань, враховуючи як потреби бізнесу, так і технічні обмеження, що забезпечує ефективне створення індивідуальних електронних платформ з різними рівнями складності та масштабування[**Error! Reference source not found.**].

Сьогодні на ринку виділяють наступні платформи для розробки e-commerce застосунків:

- shopify є однією з найпопулярніших SaaS-платформ для створення повноцінних інтернет-магазинів та управління електронною комерцією. Вона надає користувачам інтуїтивно зрозумілий інтерфейс, набір готових шаблонів для швидкого старту, інтегровану систему обробки онлайн-платежів, а також доступ до великої бібліотеки сторонніх додатків і модулів. Ці інструменти дозволяють розширити функціональність магазину без необхідності глибоких технічних знань. Основною перевагою платформи є можливість швидкого запуску магазину без значних інвестицій у розробку, що робить Shopify особливо привабливою для малого та середнього бізнесу. Водночас, хоча Shopify підтримує певний рівень кастомізації, можливості для внесення складних змін у логіку роботи чи глибоку технічну оптимізацію можуть бути обмеженими через закритість платформи та залежність від внутрішньої екосистеми[5];
- magento (Adobe Commerce) – це потужна open-source платформа для створення гнучких і масштабованих рішень у сфері електронної комерції. Вона особливо популярна серед середнього та великого бізнесів, які потребують глибокої кастомізації функціональних аспектів, складної логіки роботи з товарами та інтеграції з численними зовнішніми системами. Magento забезпечує підтримку великої кількості товарів, користувачів та одночасних транзакцій, що робить її придатною для високонавантажених проектів. Однією з ключових переваг є її модульна архітектура, яка дозволяє змінювати і розширювати систему відповідно до потреб бізнесу. Крім того, платформа надає розвинені можливості персоналізації інтерфейсу, управління каталогами, ціноутворенням, SEO-оптимізацією та маркетингом. Проте розробка, налаштування та підтримка проєктів на базі Magento потребує глибоких технічних знань, значних ресурсів і досвіду роботи з її складною структурою, що може стати бар'єром для малого бізнесу або початківців у веброзробці[6];

− wooCommerce – це популярне розширення для WordPress, яке дозволяє швидко і зручно додати функціональні можливості електронної комерції до вже існуючого або нового сайту на цій CMS. Воно підходить як для малого, так і для середнього бізнесу, оскільки дозволяє створити інтернет-магазин із широкими можливостями без необхідності значних технічних знань. Перевагами WooCommerce є легка інтеграція в екосистему WordPress, інтуїтивно зрозумілий інтерфейс, величезна кількість безкоштовних і платних тем та плагінів, що дозволяють налаштувати магазин відповідно до потреб бізнесу. Активна спільнота розробників та користувачів забезпечує постійну підтримку та оновлення. Крім того, WooCommerce легко налаштовується під локальні ринки завдяки великій кількості локалізованих модулів та платіжних систем. Водночас для великих інтернет-магазинів, що обслуговують тисячі товарів або великий потік трафіку, WooCommerce може мати обмеження у продуктивності та масштабованості. Це пов'язано з тим, що платформа побудована на WordPress, який не є спеціалізованим рішенням для високонавантажених e-commerce проєктів, і вимагає ретельної оптимізації та потужного хостингу для стабільної роботи в таких умовах;

− bigCommerce – це одна з провідних SaaS-платформ, розроблена спеціально для потреб середнього та великого бізнесу, який потребує стабільного, гнучкого та масштабованого рішення для електронної комерції. Вона дозволяє запускати повнофункціональні інтернет-магазини без необхідності самостійно розгортати інфраструктуру чи займатися технічним обслуговуванням. Однією з ключових переваг BigCommerce є підтримка мультиканальних продажів, тобто можливість продавати товари не лише через власний сайт, а й на сторонніх платформах, таких як Amazon, eBay, Facebook, Instagram та інші. Платформа має вбудовану оптимізацію для пошукових систем (SEO), що дозволяє покращити видимість магазину в результатах пошуку та залучити більше органічного трафіку. Крім того, BigCommerce підтримує широкий набір інтеграцій з платіжними системами, логістичними службами, CRM і ERP-рішеннями, аналітичними

інструментами та маркетинговими платформами. Завдяки цьому вона ідеально підходить для компаній, які планують масштабувати бізнес або вже мають складні бізнес-процеси. Однак слід враховувати, що BigCommerce є платною платформою з підпискою, вартість якої залежить від обсягу продажів та функціональних потреб. Тому для невеликих проєктів або стартапів вона може виявитися фінансово менш вигідною у порівнянні з простішими рішеннями;

- openCart – це безкоштовна система управління інтернет-магазинами з відкритим вихідним кодом, яка користується популярністю серед малого та середнього бізнесу завдяки простоті використання та широкому спектру можливостей. Однією з ключових переваг платформи є інтуїтивно зрозумілий інтерфейс адміністративної панелі, що дозволяє швидко налаштовувати магазин навіть користувачам із базовими технічними знаннями. OpenCart має велику бібліотеку безкоштовних і платних модулів та шаблонів, що дає змогу розширити функціональність магазину згідно з індивідуальними потребами. Крім того, система підтримує багатомовність, мультивалютність, різні способи доставки та оплати, що робить її зручною для запуску магазинів у міжнародному масштабі. Однак, для реалізації складніших функцій, а також для глибокого кастомізування зовнішнього вигляду чи логіки роботи магазину, потрібні певні технічні знання – зокрема, володіння PHP, HTML/CSS та основами роботи з базами даних. Також важливо враховувати, що хоча базова версія OpenCart безкоштовна, вартість повноцінного проєкту може зрости через необхідність платних модулів або залучення розробників. Незважаючи на це, OpenCart залишається одним із доступних і водночас функціональних рішень для створення інтернет-магазину.

2.2 Огляд популярних платформ для електронної комерції в Україні. Порівняння їх показників

Ринок електронної комерції в Україні активно зростає, і з ним зростає кількість великих маркетплейсів, що пропонують можливості як для споживачів, так і для бізнесу. Кожна з платформ має власну технічну архітектуру, яка визначає їхню продуктивність, масштабованість та зручність використання [**Error! Reference source not found.**].

Основні платформи в Україні, які користуються популярністю:

1. Rozetka – найбільший та найвідоміший онлайн-ритейлер в Україні, який охоплює широкий спектр товарних категорій: від побутової техніки, електроніки, одягу й аксесуарів – до товарів для дому, косметики та навіть продуктів харчування. Платформа об’єднує мільйони покупців і продавців, виступаючи універсальним маркетплейсом для щоденних покупок. Завдяки зручному інтерфейсу, швидкій доставці та широкій мережі пунктів видачі, Rozetka утримує лідерські позиції на ринку e-commerce. Станом на березень 2025 року, щомісячний трафік ресурсу становить близько 57,76 млн відвідувань, що свідчить про стабільно високу довіру користувачів і потужну присутність на цифровому ринку України. [7].

Особливості даної платформи полягають в наступному:

- власна служба доставки;
- можливість продажу як від імені продавця, так і за моделлю комісійної торгівлі;
- понад 4 млн товарів у каталозі.

2. Prom.ua – один із провідних маркетплейсів в Україні, який активно функціонує як у B2C (Business-to-Customer), так і в B2B (Business-to-Business) сегментах. Платформа об'єднує десятки тисяч підприємців, малий та середній бізнес, а також кінцевих споживачів, забезпечуючи ефективну взаємодію між

продавцями та покупцями. На Prom.ua представлені мільйони товарів у різних категоріях – від промислового обладнання до товарів повсякденного вжитку, що робить сервіс універсальним інструментом для онлайн-торгівлі. Платформа пропонує зручну панель управління, SEO-оптимізовані міні-сайти для продавців, а також інтеграцію з платіжними системами та службами доставки. Станом на березень 2025 року щомісячний трафік сайту становить близько 36,85 млн відвідувань, що підкреслює його значущість у структурі українського e-commerce простору.

Особливостями даної платформи є:

- Понад 100 млн товарів від тисяч підприємців;
- Інтегрований конструктор сайтів для продавців;
- Можливість додавання зовнішнього домену до магазину.

3 OLX.ua – найбільший онлайн-маркетплейс в Україні, який об’єднує в собі функціональність традиційної дошки оголошень і можливості сучасної платформи електронної комерції. Сервіс дозволяє як приватним особам, так і представникам малого та середнього бізнесу швидко та зручно розміщувати оголошення, продавати, купувати або обмінюватися товарами та послугами в різних категоріях — від нерухомості та транспорту до побутової техніки та вакансій. Простий і доступний інтерфейс, широкий охват аудиторії, можливість спілкування напряму між продавцем і покупцем, а також підтримка мобільного застосунку роблять OLX надзвичайно популярною платформою серед українських користувачів. Станом на березень 2025 року сайт отримує близько 52,9 мільйона відвідувань на місяць, що забезпечує йому перше місце в категорії "Оголошення" в Україні та 517-у позицію у глобальному рейтингу вебресурсів, що свідчить про стабільний попит і активну участь користувачів у платформі.

2.3 Загальні висновки по існуючим рішенням методів розробки платформ з електронної комерції. Аналіз місць для покращення

Сучасні платформи електронної комерції в Україні, зокрема Rozetka, Prom.ua та OLX.ua, наочно демонструють різноманіття у підходах до розробки, впровадження та підтримки веб-систем. Вони використовують різні технологічні стеки, що охоплюють як класичні рішення, так і сучасні тенденції, включаючи застосування монолітної, мікросервісної або headless-архітектури. Кожна платформа обирає власну стратегію з урахуванням масштабів проєкту, потреб цільової аудиторії та бізнес-завдань.

Rozetka, як лідер українського онлайн-ритейлу, реалізує комплексну архітектуру з потужним внутрішнім каталогом, логістичною інфраструктурою та підтримкою високого навантаження. Prom.ua орієнтується як на роздрібного, так і на оптового клієнта, пропонуючи торговим майданчикам готові рішення для онлайн-присутності, що передбачає адаптивні модулі та інтеграцію з CRM-системами. OLX.ua, у свою чергу, зосереджується на C2C-сегменті, поєднуючи простоту оголошень з базовими можливостями електронної торгівлі. В кожному з випадків особлива увага приділяється користувацькому досвіду, інтуїтивній навігації, безпеці даних та оптимізації швидкодії.

Незважаючи на функціональні відмінності та різні підходи до реалізації, всі зазначені сервіси ґрунтуються на використанні сучасних веб-технологій, що дозволяють досягати високої стабільності, масштабованості й адаптивності. Це забезпечує ефективне обслуговування мільйонів користувачів, збереження високої доступності платформи навіть при пікових навантаженнях, а також реалізацію різноманітних сценаріїв використання — від класичної купівлі-продажу товарів до професійного онлайн-бізнесу та B2B-взаємодії. В результаті такі платформи не лише задовольняють поточні потреби користувачів, а й

формують стандарти розвитку e-commerce в Україні.Загалом, існують три основні архітектурні підходи до розробки e-commerce платформ:

- монолітна архітектура: це підхід, за якого всі компоненти системи (інтерфейс, логіка, база даних тощо) інтегровані в єдину, нероздільну програмну структуру. Такий підхід спрощує початкову розробку, тестування та розгортання, оскільки всі частини взаємодіють усередині одного середовища. Однак із розростанням функціональних можливостей система стає менш гнучкою: ускладнюється масштабування, впровадження нових функцій або змін, зростає ризик помилок при внесенні змін у код, а оновлення потребують перезбирання та перезапуску всього застосунку;
- мікросервісна архітектура: це підхід до розробки програмного забезпечення, при якому система розбивається на набір незалежних сервісів, кожен з яких виконує окрему функцію та взаємодіє з іншими через API. Така структура забезпечує високу гнучкість, спрощує масштабування окремих компонентів, прискорює впровадження нових функцій і дозволяє командам працювати автономно над різними частинами системи. Проте мікросервісний підхід вимагає складнішої інфраструктури, надійної системи комунікації між сервісами, грамотного управління розгортанням та забезпечення консистентності даних, що може збільшувати складність підтримки та розробки;
- headless: передбачає повне розділення фронтенду та бекенду, що дозволяє створювати гнучкі, персоналізовані та багаторазово використовувані інтерфейси користувача, які взаємодіють із серверною частиною через API. Такий підхід забезпечує високу масштабованість, адаптивність до різних платформ (веб, мобільні додатки) та швидке впровадження змін в інтерфейсі без модифікації логіки бекенду. Проте, headless-архітектура потребує вищих витрат на реалізацію, складніших механізмів синхронізації, безпеки та підтримки, що слід враховувати при виборі моделі.

Незважаючи на численні досягнення та активний розвиток електронної комерції в Україні, залишається низка аспектів, у яких вітчизняні платформи

мають значний потенціал для подальшого вдосконалення. Зокрема, це стосується підвищення рівня зручності користувацького інтерфейсу, оптимізації навігації, персоналізації сервісів відповідно до потреб конкретного клієнта та збільшення гнучкості функціональних можливостей для його ролі, а також забезпечення більш швидкого доступу до інформації. Крім того, варто звернути увагу на покращення мобільної адаптації та впровадження інноваційних рішень, які сприятимуть формуванню більш привабливого, комфортного та ефективного користувацького досвіду.

3 ВИБІР ТЕХНОЛОГІЙ ТА ПІДХОДІВ ДЛЯ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ E-COMMERCE ПЛАТФОРМ

При виборі технологій для розробки клієнтського інтерфейсу доцільно враховувати наступні фактори:

- відповідність функціональним вимогам, здатність технології ефективно реалізовувати поставлені завдання та забезпечувати необхідну функціональність.
- масштабованість та підтримка, можливість подальшого розвитку та підтримки проєкту, включаючи наявність активної спільноти та регулярних оновлень.
- екосистема та сумісність, наявність широкого спектру бібліотек і інструментів, що полегшують розробку, а також їх сумісність між собою.
- продуктивність та ефективне використання ресурсів як на стороні сервера, так і на пристроях кінцевих користувачів, що забезпечує швидке завантаження та плавну взаємодію.
- безпека та надійність, здатність технології забезпечувати захист даних користувачів та стабільну роботу платформи.

З огляду на вищезазначені міркування, для розробки клієнтської частини застосунку, окрім базових веб-технологій, було обрано стек, що включає React, TypeScript та Redux.

3.1 Надбудова TypeScript

Утиліта TypeScript – це потужне розширення (надмножина) мови JavaScript, яке додає підтримку статичної типізації, об'єктно-орієнтованого підходу,

інтерфейсів та інших сучасних можливостей, що роблять процес розробки більш надійним, структурованим і контрольованим. Ця мова була створена для того, щоб полегшити створення масштабованих веб-додатків, які з часом потребуватимуть підтримки, розширення та командної співпраці [9].

TypeScript компілюється у звичайний JavaScript, який може виконуватись у будь-якому браузері чи середовищі виконання на зразок Node.js, що робить його повністю сумісним з існуючими JavaScript-проєктами. Основна перевага TypeScript полягає в наявності статичної типізації, яка дозволяє розробникам виявляти помилки ще на етапі написання коду – ще до запуску додатку. Це не лише підвищує якість програмного продукту, але й значно економить час у процесі тестування та налагодження.

Завдяки типізації код стає більш зрозумілим, передбачуваним і легшим для читання та подальшої підтримки, особливо в командній роботі або на великих проектах. Можна легко визначати очікувані типи параметрів функцій, повертати типи результатів, описувати структури об'єктів тощо. Це знижує кількість помилок, пов’язаних з неправильним використанням змінних, і дозволяє зменшити ризик неочікуваної поведінки застосунку[10].

Використання TypeScript у поєднанні з React є однією з найпоширеніших і найефективніших практик у сучасній веб-розробці. Така комбінація дозволяє створювати більш стабільні та масштабовані frontend-рішення, де компоненти мають чітко визначені props та стани, що значно полегшує роботу з ними в довгостроковій перспективі. Завдяки такому підходу код стає більш читабельним, зрозумілим, безпечним і значно легше піддається рефакторинг[**Error! Reference source not found.**].

3.2. Бібліотека React

Бібліотека **React** – це популярна JavaScript-бібліотека, призначена для створення сучасних, зручних та ефективних інтерфейсів користувача. Вона дозволяє розробникам створювати динамічні та інтерактивні веб-додатки, що реагують на дії користувача без необхідності повного перезавантаження сторінки. Її компонентна архітектура надає можливість організовувати код у вигляді незалежних блоків, які можна повторно використовувати в різних частинах застосунку. Це значно полегшує обслуговування, масштабування та розширення великих проектів, а також сприяє підвищенню якості коду [9].

Завдяки використанню віртуального DOM (Document Object Model), React забезпечує оптимізоване оновлення інтерфейсу, дозволяючи оновлювати лише ті елементи сторінки, які дійсно змінилися. Це суттєво покращує продуктивність застосунку, що особливо важливо у випадку e-commerce платформ, де кількість взаємодій користувача з інтерфейсом може бути дуже великою. Такий підхід дозволяє досягати високої швидкості відгуку та зменшувати навантаження на браузер.

Серед ключових переваг React варто виділити широку підтримку з боку спільноти розробників, що забезпечує швидке вирішення проблем, велику кількість навчальних матеріалів, сторонніх бібліотек і рішень для інтеграції. React добре поєднується з іншими UI-бібліотеками, дозволяє гнучко будувати інтерфейси, а його проста архітектура, заснована на компонентному підході, робить його зручним у вивченні навіть для початківців.

Крім того, React забезпечує високу продуктивність роботи інтерфейсу, підтримує численні підходи до оптимізації продуктивності (наприклад, мемоізацію компонентів), має швидкий механізм рендерингу, а також надає великий набір готових інструментів для реалізації поширених задач. Це значно пришвидшує процес розробки, зменшує кількість помилок та дозволяє зосередитися на створенні якісного та привабливого користувацького досвіду [11].

3.3 Технологія для контролю станів Redux

Технологія redux – це бібліотека для керування станом JavaScript-додатків. Найчастіше використовується з React, але також може працювати з іншими фреймворками або навіть з чистим JS [**Error! Reference source not found.**].

Redux дозволяє зберігати єдиний глобальний стан додатку у централізованому місці та ефективно керувати всіма змінами цього стану через передбачуваний механізм – дії (actions) та редуктори (reducers). Такий підхід дозволяє уникнути хаосу при обробці станів у великих застосунках, зменшити дублювання логіки та забезпечити кращу керованість даними. Крім того, централізація даних сприяє простішому відстеженню помилок, легшому дебагу та забезпечує прозорість у потоці даних, що надходять у компоненти інтерфейсу.

Це також дозволяє виділити головні сутності системи та зберігати їх необхідні стани у зручному та логічно впорядкованому вигляді, який можна використовувати у відповідних компонентах застосунку. Такий підхід набагато полегшує контроль над процесами, що відбуваються у застосунку, дозволяє розробникам краще координувати логіку відображення та поведінку інтерфейсу. У результаті структура коду стає менш перезавантаженою, більш зрозумілою та масштабованою, що особливо важливо при роботі над складними проектами у сфері електронної комерції.

Усі перераховані вище переваги роблять цю технологію доцільним вибором при створенні застосунків з електронної комерції, де необхідна висока надійність, продуктивність і чіткий контроль за станом усіх частин інтерфейсу та бізнес-логіки. Redux ідеально підходить для реалізації таких вимог, що пояснює його популярність серед розробників сучасних веб-застосунків.[13].

3.4 Бібліотека axios

Axios – це популярна JavaScript-бібліотека для виконання HTTP-запитів із клієнтської або серверної частини застосунку. Вона базується на промісах і забезпечує простий та зручний синтаксис для обміну даними з API. Axios широко використовується у frontend-розробці, зокрема в додатках на React, Vue, Angular та інших фреймворках [14].

Однією з ключових переваг Axios є автоматичне перетворення відповідей у формат JSON, зручна обробка помилок, підтримка налаштування заголовків, перехоплення запитів і відповідей (interceptors), а також можливість надсилання запитів з авторизацією.

Axios також підтримує роботу з асинхронними операціями через async/await, що підвищує зручність і читабельність коду. У контексті e-commerce або інших сучасних веб-додатків Axios є важливим інструментом для взаємодії з сервером: отримання списків товарів, надсилання форм замовлення, автентифікація користувача, оновлення інформації тощо.

Завдяки своїй гнучкості та активній спільноті, Axios залишається одним із найпопулярніших рішень для роботи з HTTP-запитами у веб-розробці.

3.5 Бібліотека React Route

Бібліотека React Router – це офіційна бібліотека для реалізації маршрутизації в застосунках, побудованих на React. Вона дозволяє створювати односторінкові застосунки (SPA), у яких навігація між різними сторінками чи розділами відбувається без повного перезавантаження сторінки. Це забезпечує більш плавний, швидкий і зручний користувацький досвід.

React Router надає набір інструментів для створення динамічних маршрутів, що залежать від URL-адреси, підтримує вкладену маршрутизацію, перенаправлення, захищені маршрути (наприклад, для авторизованих користувачів), обробку помилок при переході, та багато іншого. Основними компонентами бібліотеки є BrowserRouter, Routes, Route, Link, Navigate тощо.

Використання React Router є надзвичайно важливим для створення повноцінних веб-додатків, особливо в контексті e-commerce платформ, де користувач може переміщатися між каталогом товарів, сторінкою товару, кошиком, оформленням замовлення, особистим кабінетом тощо. Бібліотека добре інтегрується з іншими популярними технологіями, такими як Redux, React Query або TypeScript, що дозволяє будувати масштабовані, підтримувані та гнучкі системи.

Таким чином, React Router є невіддільною частиною сучасної frontend-розробки, яка дозволяє створювати зручну та логічну навігацію в межах веб-застосунку[15].

3.6 Бібліотека Material UI

Material UI – це популярна, широко використовувана бібліотека компонентів інтерфейсу користувача для React, яка реалізує принципи сучасного дизайну Material Design, розроблені компанією Google. Ця бібліотека надає розробникам великий набір заздалегідь стилізованих, зручних у використанні UI-компонентів, таких як кнопки, текстові поля, форми, діалогові вікна, сповіщення, таблиці, вкладки, навігаційні панелі тощо. Завдяки цьому розробники можуть суттєво скоротити час створення інтерфейсу веб-застосунків, зберігаючи при цьому єдність стилю та професійний зовнішній вигляд користувацького інтерфейсу [17].

Окрім базових функціональних можливостей, Material UI вирізняється високою гнучкістю та широкими можливостями кастомізації, що дозволяє адаптувати кожен компонент відповідно до конкретних вимог та дизайну проєкту. Компоненти підтримують адаптивну верстку, що гарантує коректне відображення елементів інтерфейсу на мобільних, планшетних та десктопних пристроях. Це надзвичайно важливо в умовах мультиплатформеного використання e-commerce застосунків.

Ще однією перевагою Material UI є її відмінна інтеграція з сучасними інструментами та технологіями веброзробки, такими як TypeScript, бібліотеки для керування станом (наприклад, Redux або MobX), а також інструменти для серверного рендерингу чи тестування. Бібліотека також підтримує темізацію – можливість швидко змінювати вигляд застосунку, налаштовуючи кольори, шрифти, інтервали тощо, що дозволяє гнучко адаптувати дизайн під фірмовий стиль компанії або переваги цільової аудиторії.

У контексті створення застосунків для електронної комерції, використання Material UI надає широкі можливості для створення зрозумілого, сучасного та привабливого інтерфейсу користувача. Такий підхід не тільки покращує користувацький досвід, а й сприяє досягненню бізнес-цілей компанії, оскільки інтуїтивно зрозумілий та візуально привабливий інтерфейс стимулює користувачів до довшого перебування на сайті, полегшує навігацію та сприяє збільшенню конверсій.

3.7 Архітектура FSD для структуризації коду

Архітектура FSD (Feature-Sliced Design) – це сучасний підхід до організації frontend-коду, який набув особливої популярності в розробці React-застосунків. Вона базується на принципі поділу додатку на незалежні функціональні “шари” або “слайси”, кожен з яких відповідає за конкретну бізнес-функцію чи

особливість. Такий підхід сприяє підвищенню масштабованості, підтримуваності та зрозумілості коду, а також полегшує командну роботу та повторне використання компонентів у різних частинах проєкту. [18]. Візуалізація основного концепту даної архітектури представлена на рисунку 3.1

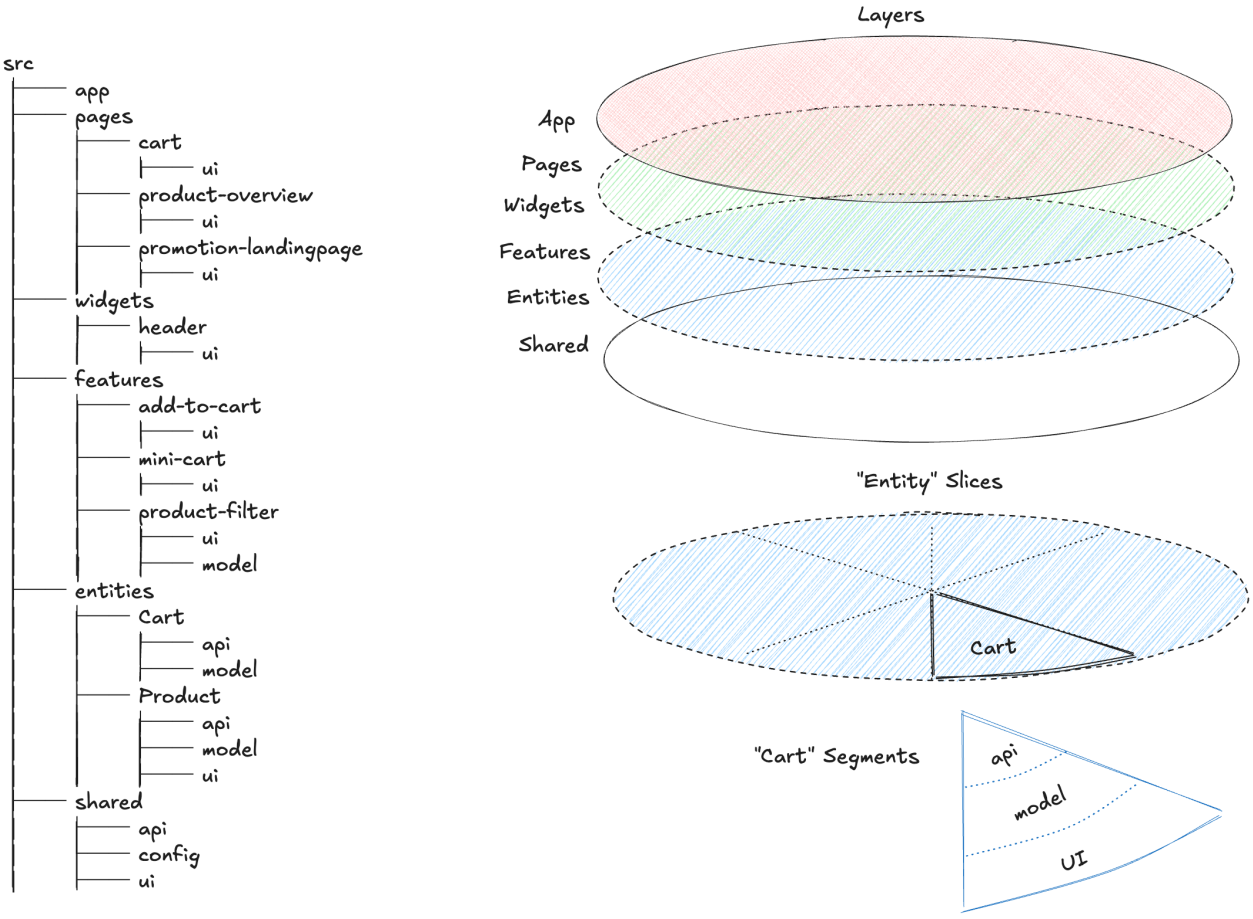

Рисунок 3.1 – Візуалізація архітектури FSD

Архітектура FSD допомагає масштабувати проєкти, зберігати читабельність та структурувати код відповідно до бізнес-функціональності. Основними перевагами даної архітектури є:

- чітке структурування проекту на основі задач, що відображає реальні функціональні одиниці продукту (shared) [**Error! Reference source not found.**];
- масштабованість, адже проект набуває більш структурованого та зрозумілого вигляду (entities);

- чітке розділення відповідальностей, адже є поділ на визначенні шари (спільні функції, базові сутності, окремі функції, складні та компоненти, що перевикористовуються) (widgets);
- шар представлення – сторінки застосунку (pages);
- шар загальних налаштування застосунку (app).

Ця архітектура є чудовим вибором для розробки сучасного програмного забезпечення, особливо у випадках, коли важливими є легкість у підтримці, масштабованість і чітка модульна структура. Вона дозволяє ефективно організувати кодову базу, розподілити функціональні обов'язки між різними модулями та спростити процес тестування й оновлення системи. Таке структуроване розділення полегшує командну роботу над проєктом, дозволяє уникати дублювання логіки та підвищує загальну продуктивність команди розробників.

У контексті створення e-commerce застосунку це має особливе значення, оскільки такі платформи зазвичай включають в себе велику кількість функціональних блоків – каталог товарів, кошик, система реєстрації та авторизації, обробка замовлень, адміністрування тощо. Кожен з цих блоків може бути реалізований у вигляді окремого модуля, що значно спрощує як поточну розробку, так і подальшу масштабну підтримку або розширення функціональності системи. Таким чином, обрана архітектура виступає надійною основою для розробки високоякісного застосунку для електронної комерції[1].

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

У ході реалізації системи було створено клієнтську частину застосунку для електронної комерції з використанням сучасних технологій: React для побудови інтерфейсу, React Router для навігації, axios для роботи з API, TypeScript для типізації та Redux для управління станом та Material UI для стилізації компонентів. Цей стек забезпечив високу продуктивність, зручність у масштабуванні та підтримці коду.

4.1 Функціональні аспекти програми

Головною метою роботи було створення якісного, зручного та функціонального UI-інтерфейсу для клієнтської частини застосунку в сфері електронної комерції. В сучасних умовах високої конкуренції на ринку онлайн-продажів інтерфейс користувача відіграє надзвичайно важливу роль, оскільки саме він безпосередньо впливає на комфорт, швидкість і загальний користувацький досвід. Тому було поставлено за завдання розробити інтерфейс, який би відповідав сучасним вимогам зручності, адаптивності до різних пристроїв, функціональності та безпеки.

Для досягнення цієї мети було проведено детальний аналіз основних функціональних вимог, що висуваються до клієнтської частини застосунку. Ці вимоги охоплюють ключові аспекти взаємодії користувачів із системою – від авторизації, перегляду товарів, їх пошуку, до управління кошиком та оформлення замовлень. Крім того, було враховано потребу в персоналізації інтерфейсу, забезпеченні стабільності роботи та підтримці і масштабованості системи.

Всі основні функціональні вимоги, які стали базою для розробки, систематизовані і представлені в таблиці4.1. Вона чітко відображає, які саме можливості має надавати клієнтська частина застосунку.

Таблиця 4.1 – функціональні та не функціональні вимоги

№	Функціональні вимоги	Не функціональні вимоги
1	Сторінка детального перегляду товару	Швидке завантаження сторінок
2	Можливість здійснювати пошук товарів	Локалізація інтерфейсу
3	Кошик для управління замовленнями	Кросбраузерність
4	Оформлення замовлення	Адаптивність
5	Реєстрація, авторизація та особистий кабінет користувача	
6	Управління профілем користувача, історія замовлень	
7	Можливість додавати продукти для продавця та звичайного користувача	
8	Відображення різних каталогів товарів відповідно до категорії та відповідно до того, ким були створені товари	

Загальну репрезентацію функціональних можливостей, які покриваються в даному застосунку можна побачити на діаграмі, показаній на рисунку 4.1. Вона відображає основні варіанти використання, які допустимі в межах даної платформи.

Така візуалізація дозволяє краще зрозуміти основні функціональні можливості платформи та ключові моменти взаємодії користувача з програмою.

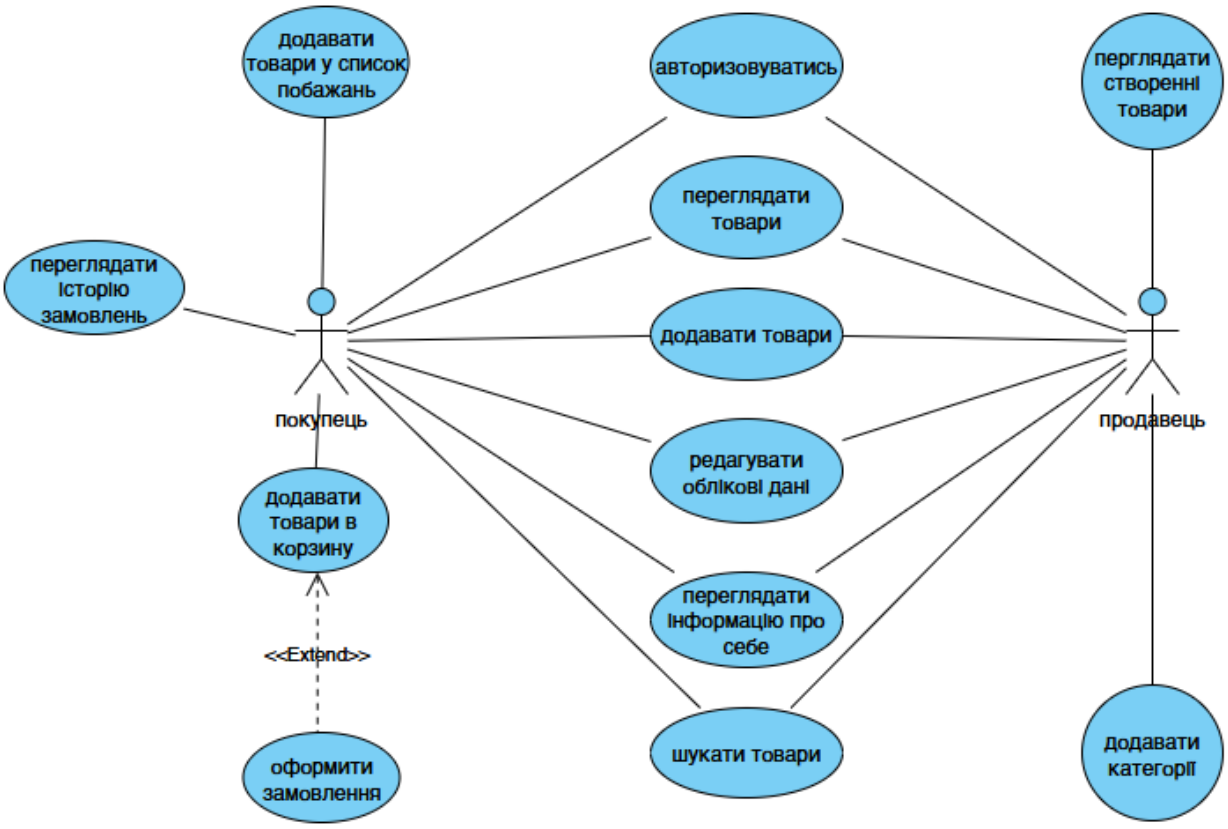

Рисунок 4.1 – Діаграма прицедентів продукту

На рисунку 4.1 показані основні етапи взаємодії 2-х типів користувачів із системою – покупця та продавця. Більш детальний опис функціональних можливостей представленої діаграми:

1. Продавець:

- реєструвати акаунт із роллю продавця;
- входити в акаунт;
- створювати товари;
- створювати категорії;
- переглядати створені ним товари;
- здійснювати пошук товарів (по категоріям, назвах, параметрам);
- редагувати інформацію в особистому кабінеті.

2. Покупець:

- створювати акаунт;

- входити в акаунт;
- переглядати товари наявні в магазині (по категоріям);
- здійснювати пошук товарів (по категоріям, назвах, параметрам);
- переглядати інформацію про товар;
- здійснювати покупку – додавати товар до корзини;
- додавати товар до списку побажань щоб мати змогу повернутись до нього пізніше;
- оформлювати замовлення;
- переглядати оформлені замовлення;
- редагувати інформацію про себе.

Таким чином, діаграма демонструє основні сценарії взаємодії двох ключових ролей у системі – продавця та покупця, окреслюючи перелік функціональних можливостей, доступних для кожного типу користувача. Вона відображає життєвий цикл основних дій, які можуть виконуватись у межах системи, починаючи з моменту реєстрації та входу в акаунт, до взаємодії з товарами, керування персональними даними та оформлення замовлень. Продавець отримує доступ до інструментів керування товарами, створення категорій, а також персоналізації свого профілю, що дозволяє йому повноцінно управляти своєю комерційною діяльністю в межах платформи. Зі свого боку, покупець має змогу зручно переглядати каталог товарів, використовувати пошук, додавати товари до кошика або списку бажань, оформлювати замовлення та слідкувати за їх статусом, а також змінювати інформацію у власному кабінеті.

Загалом, діаграма охоплює всі базові процеси, необхідні для повноцінного функціонування e-commerce-платформи, включаючи як адміністративні можливості продавців, так і зручність для кінцевих користувачів. Такий підхід дозволяє забезпечити ефективну взаємодію між обома сторонами платформи, створюючи гнучку й зрозумілу систему, яка адаптована до потреб кожної ролі. Це сприяє підвищенню користувацького досвіду, стимулює залучення нових

користувачів і дозволяє платформі ефективно функціонувати у конкурентному цифровому середовищі.

4.2. Технічна реалізація програми

Основою для розробки даної платформи електронної комерції стала ретельно продумана архітектура програми, яка забезпечує стабільність, масштабованість та зручність у подальшій підтримці. Одним із ключових аспектів стало ефективне управління станами, що дозволяє централізовано керувати всіма важливими даними додатку та забезпечує передбачувану взаємодію між його частинами.

У центрі моделі даних платформи знаходяться три основні об'єкти:

1. Користувач – відповідає за автентифікацію, персональні дані, історію замовлень, список побажань, кошик та інші персоналізовані функції. Управління його станом дозволяє забезпечити безперебійну взаємодію користувача з додатком, зокрема збереження сесії, захист приватної інформації та персоналізований контент;
2. Категорії – використовуються для структурування товарів і реалізації пошуку та навігації. Стан категорій визначає, які підгрупи товарів відображаються, а також дозволяє швидко оновлювати структуру каталогу в реальному часі;
3. Продукти – головний інформаційний блок eCommerce-системи, що містить детальні відомості про товари, їх характеристики, наявність, знижки тощо. Коректне управління станом товарів дозволяє забезпечити їх актуальність на frontend, швидкий рендер карток товарів, а також ефективну взаємодію з кошиком, замовленнями та іншими сервісами.

Інтеграція цих об'єктів через глобальний стан за допомогою Redux дозволяє досягти узгодженості даних між різними частинами програми, зменшити

дублювання логіки, забезпечити зручну синхронізацію з backend та гарантувати високу продуктивність інтерфейсу навіть за значного навантаження.

Для наочного розуміння яким чином на клієнтській стороні застосунку відбувається контроль над потоком даних, було наведено схему життєвого циклу об’єкту користувача – рисунок 4.2.

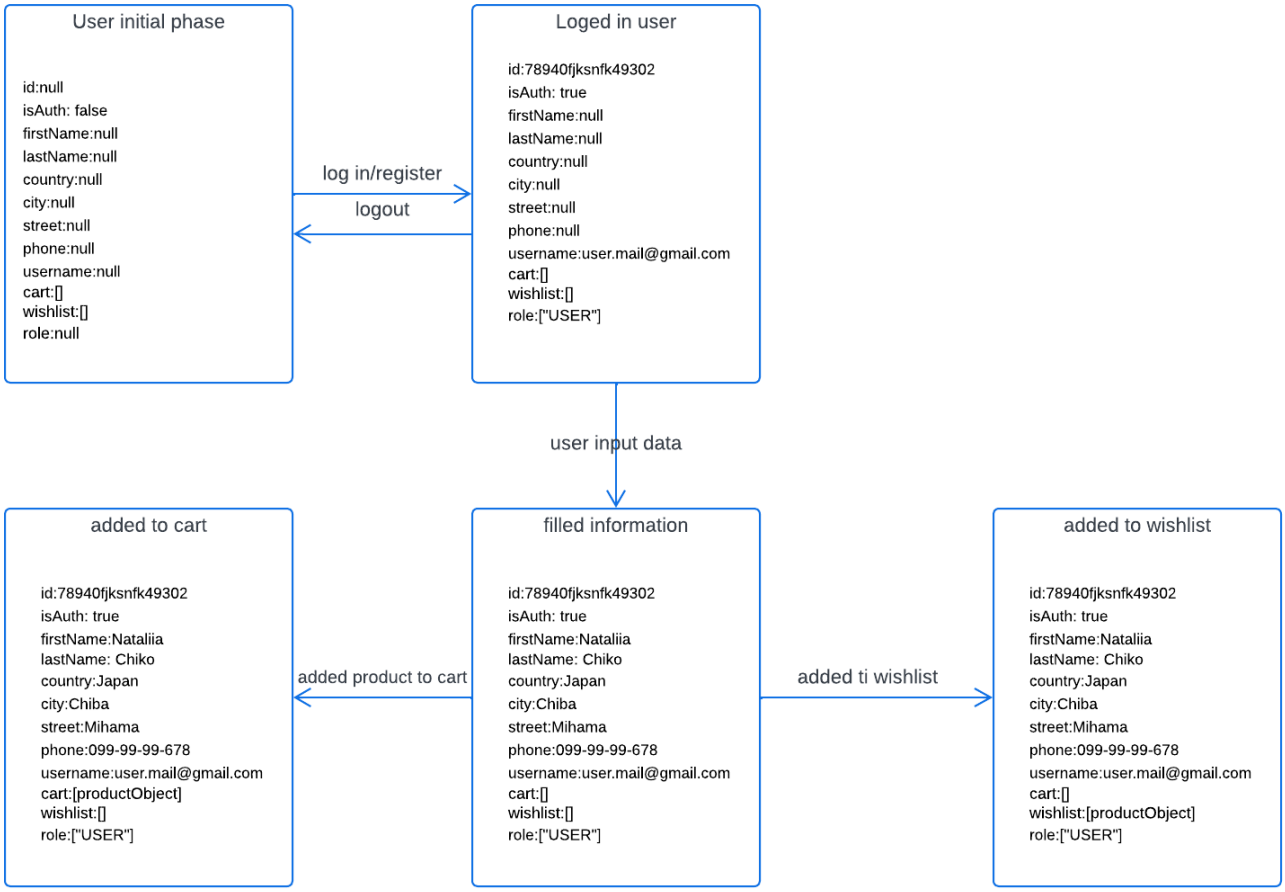

Рисунок 4.2 – Схема життєвого циклу об’єкту користувача

На рисунку 4.2 показано основні етапи життєвого циклу об’єкту користувача:

– початковий етап: етап, коли користувач ще не авторизований у системі. Всі його поля заповнені зануленими значеннями;

– авторизований етап: етап, який отримує об’єкт користувача, після реєстрації або входу в обліковий запис. В тілі об’єкта відбувається заповнення таких полів, як пошта, id та статус isAuth змінюється на позитивний;

– етап заповненого користувача: після заповнення користувачем своїх облікових даних в особистому акаунті, відповідні дані про нього на момент сесії зберігаються в об’єкті. Це полегшує їх відображення в ui-компонентах;

– етап додання в корзину: після додавання в корзину товарів, дані про це оновлюються і в об’єкті користувача;

– етап додання в список побажань: після додавання товарів у список побажань, аналогічно до кошика, вони з’являються у списку побажань в об’єкті користувача.

Для побудови інтерфейсу застосунку було створено окремі сторінки та багаторазові компоненти, які відповідають за відображення конкретного контенту та функціональність на кожному етапі взаємодії користувача з платформою. Такий підхід сприяє модульності, полегшує повторне використання коду та спрощує обслуговування проекту в майбутньому.

Однією з ключових технічних складових є реалізація системи маршрутизації – routing. Для цього було використано бібліотеку React Router, яка забезпечує ефективну навігацію між сторінками без перезавантаження браузера, що реалізує SPA-підхід. У межах цієї системи було створено:

− чітку структуру маршрутів для всіх основних сторінок (головна, каталог, сторінка товару, авторизація тощо);

− логіку динамічних маршрутів (наприклад, сторінки товару за id);

− систему перенаправлень – для обробки випадків, коли користувач звертається до неіснуючого або захищеного ресурсу без відповідних прав доступу.

Щодо обміну даними з сервером, то у програмі реалізовано шар сервісів (API services), які відповідають за взаємодію з backend-API. Їх основні методи показані на рисунку 4.3.

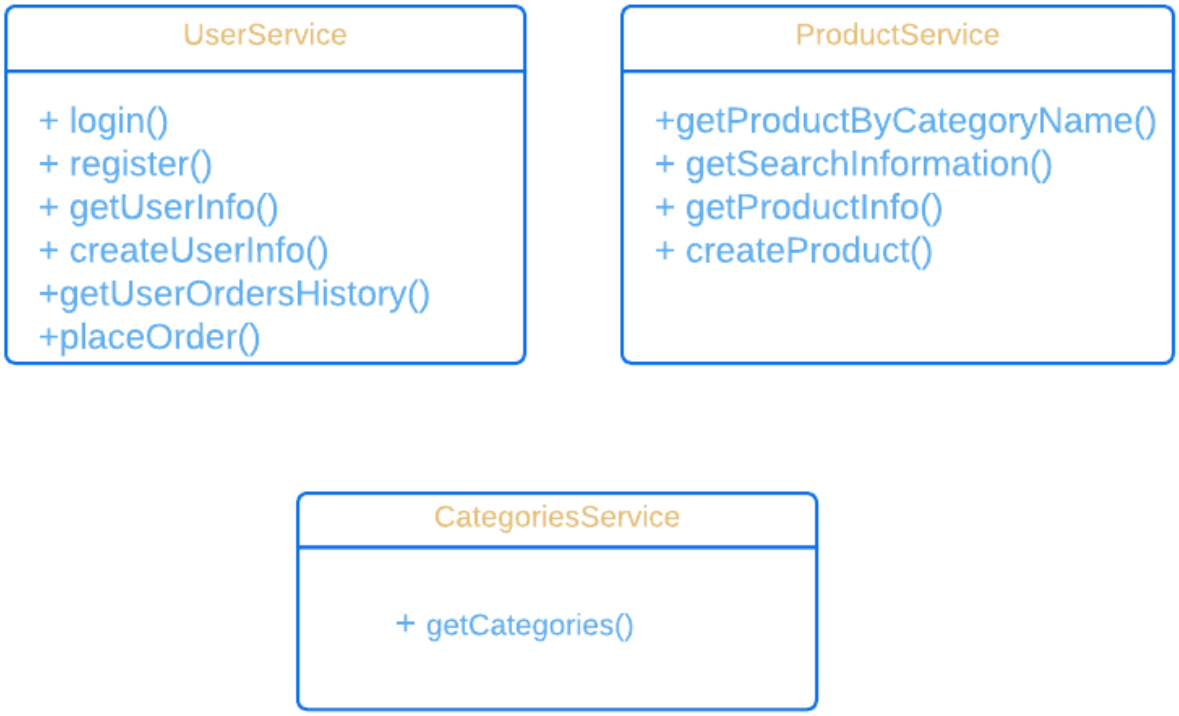

Рисунок 4.3 – Діаграма класів для сервісів

Як видно з рисунку 4.3, ці класи відповідають за комунікацію frontend та backend частини шляхом надсилання post та get запитів.

Ці сервіси інкапсулюють логіку HTTP-запитів і забезпечують централізовану точку доступу до серверних ресурсів. Для зручності та ефективності було використано бібліотеку Axios, яка надає більш компактний та читабельний синтаксис у порівнянні з нативним fetch, а також підтримує розширені можливості, такі як:

- перехоплення запитів та відповідей (interceptors);
- глобальне налаштування заголовків (headers);
- обробку помилок;
- автоматичне перетворення JSON.

Такий розподіл обов’язків між компонентами, маршрутизацією та сервісами дозволяє досягти чіткої структури коду, легкості масштабування та високої продуктивності під час розробки та обслуговування застосунку.

Крім того, централізація запитів у сервісах дає можливість реалізувати додаткові механізми безпеки та зручності – наприклад, автоматичне додавання токена авторизації до кожного запиту або повторну спробу запиту у разі помилки типу 401. Це усуває дублювання коду та покращує користувацький досвід, адже помилки обробляються уніфіковано, а дані отримуються стабільно.

5 ВЗАЄМОДІЯ КОРИСТУВАЧА З ПРОДУКТОМ

Для початку роботи із застосунком необхідно відкрити його у браузері, оскільки він є веб-додатком, що працює у середовищі інтернет-браузера без необхідності попередньої інсталяції на пристрій користувача. Цей спосіб надає зручність доступу до функціональних можливостей системи з будь-якого пристрою, що має доступ до мережі Інтернет.

Після відкриття у браузері, на користувача очікує головна сторінка сайту, яка є відправною точкою для подальшої навігації по ресурсах застосунку. Головна сторінка зазвичай містить основні категорії товарів або послуг, навігаційне меню, панель пошуку та інші елементи, що забезпечують комфортний початок взаємодії з платформою. Саме з цієї сторінки користувач може перейти до пошуку потрібного товару, ознайомитися з пропозиціями або розпочати процес оформлення замовлення (рисунок 5.1).

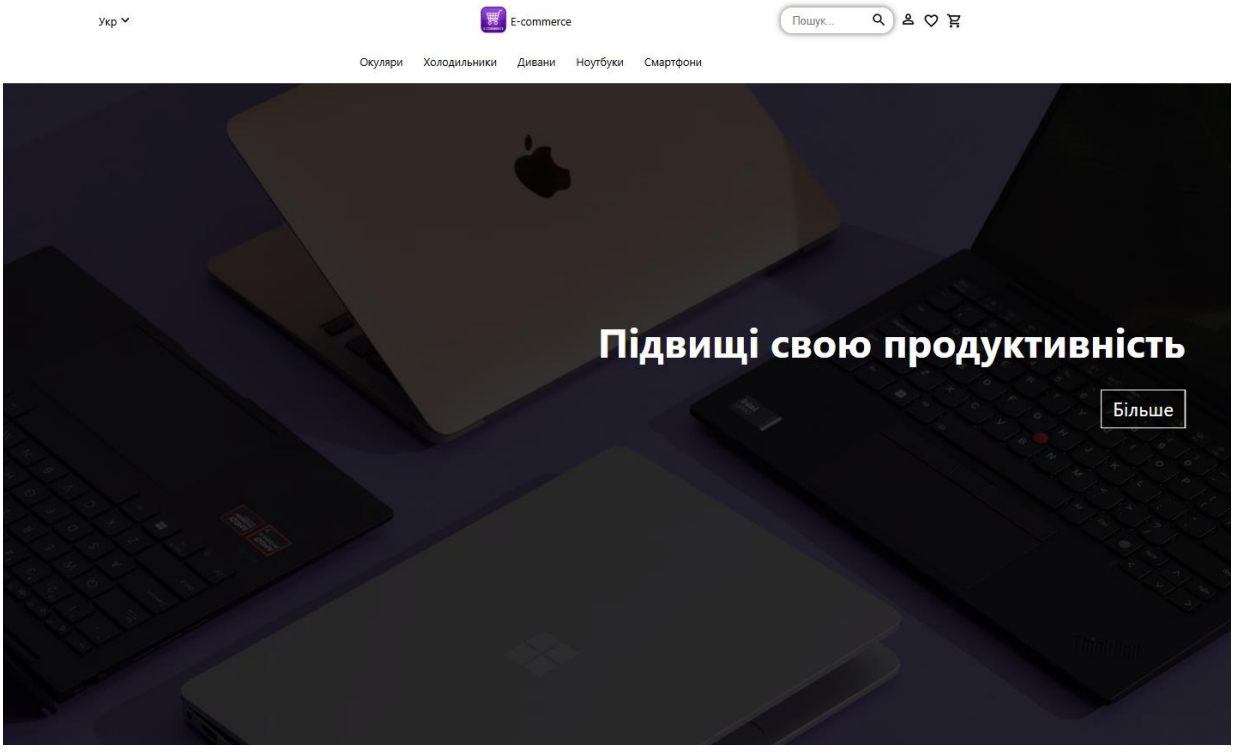

Рисунок 5.1 – Головна сторінка продукту

На головній сторінці продукту (рисунок 5.1) відображено категорії товарів, селектор для зміни мови сайту, поле для пошуку товарів по сайту, поле для взаємодії з особистим кабінетом, а також кошик та список побажань користувача.

На даному етапі користувач може змінювати мову магазину, переглядати товари та здійснювати пошук необхідних йому позицій. Для зміни мови достатньо у випадаючому списку обрати іншу мову (рисунок 5.2).

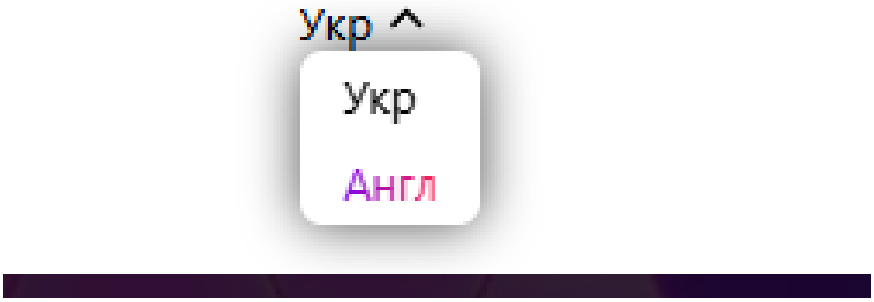

Рисунок 5.2 – Зміна мови інтерфейсу

На рисунку 5.3 показан результат зміни мови інтерфейсу на англійську.

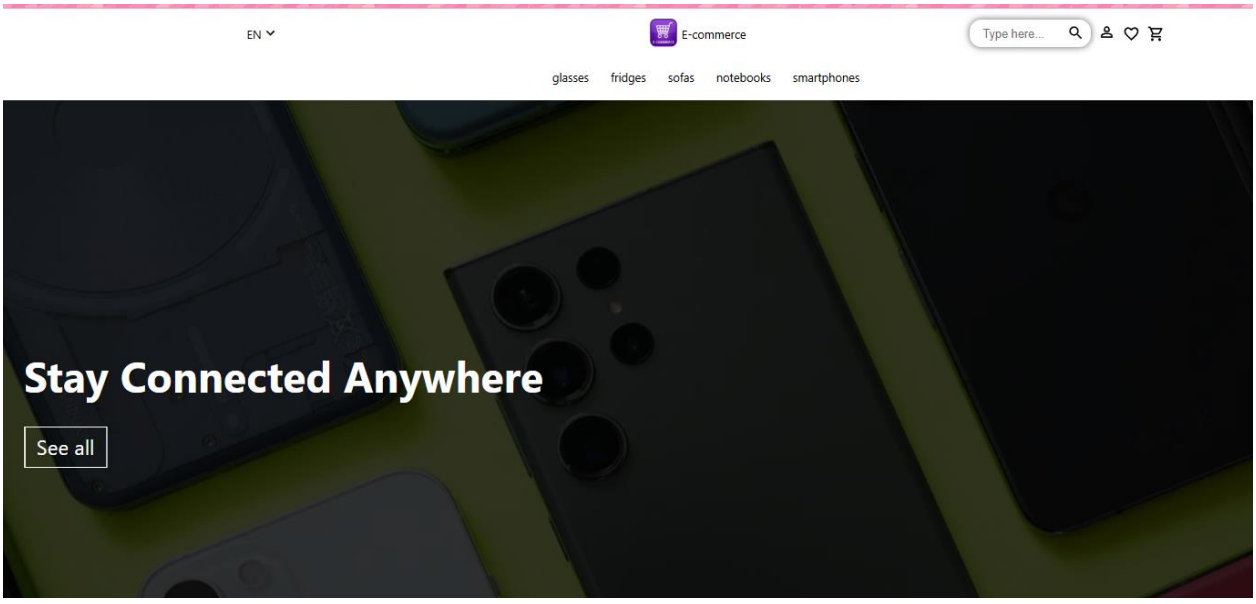

Рисунок 5.3 – Результат зміни мови сайту

Також покристувач може авторизуватися в системі. Для цього йому необхідно перейти до секції, яка відповідає за кабінет користувача (рисунок 5.4).

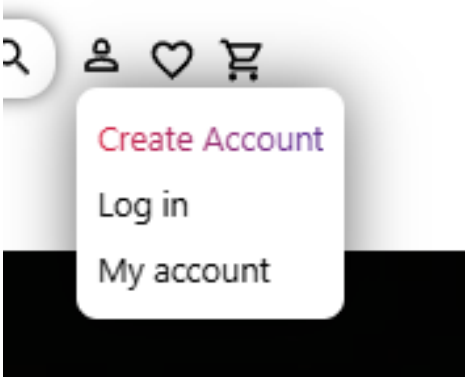

Рисунок 5.4 – Панель взаємодії із кабінетом користувача

Після обрання опції «створити акаунт» (рисунок 5.4), користувача перенаправляє на сторінку для реєстрації (рисунок 5.5).

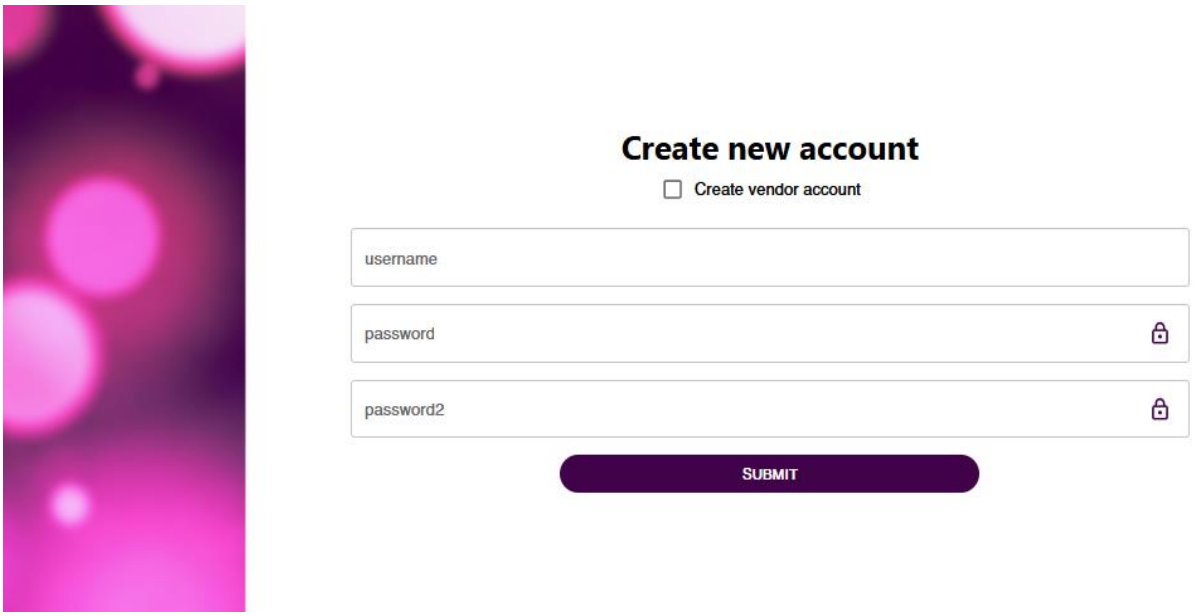

Рисунок 5.5 – Форма для реєстрації користувача

На рисунку 5.5 видно, що окрім базових полів для введення паролю та логіну користувача, є ще чекбокс для створення акаунт продавця. В цьому прикладі як раз створемо такий акаунт (рисунок 5.6).

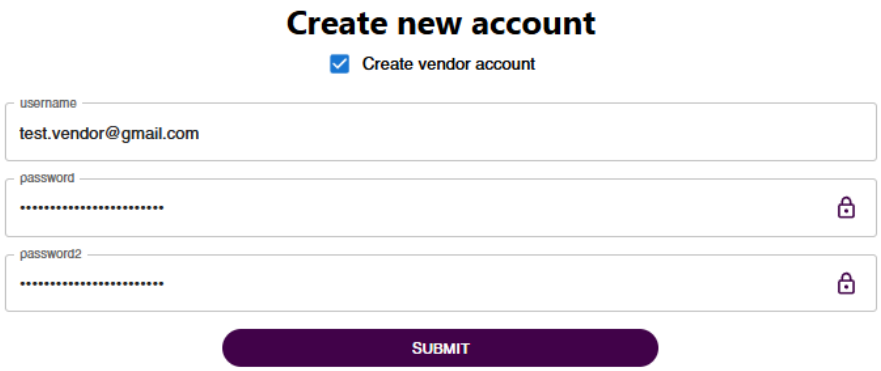

Рисунок 5.6 – Створення акаунта продавця

Після натискання на кнопку «надіслати користувача перенаправить на сторінку із його акаунтом», як показано на рисунку 5.7.

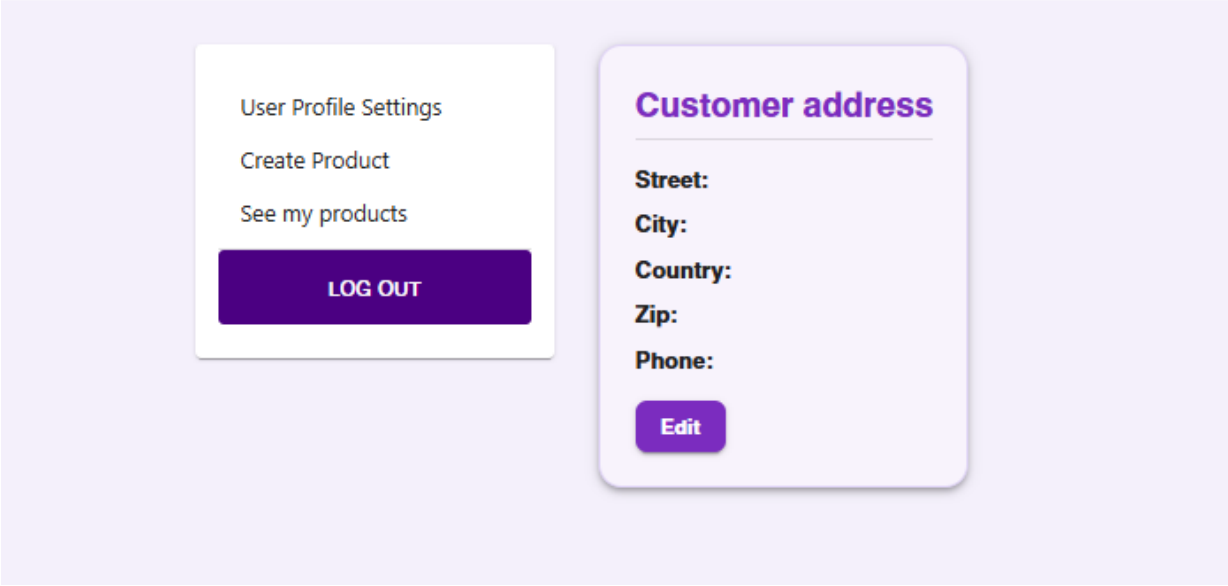

Рисунок 5.6 – Створений профіль користувача

З самого початку у користувача пустий профіль (адже всі ці поля не були додані до форми реєстрації, щоб не ускладнювати форму та сприйняття

користувача). Тут він може редагувати власні данні, натиснувши на кнопку редагувати (рисунок 5.7).

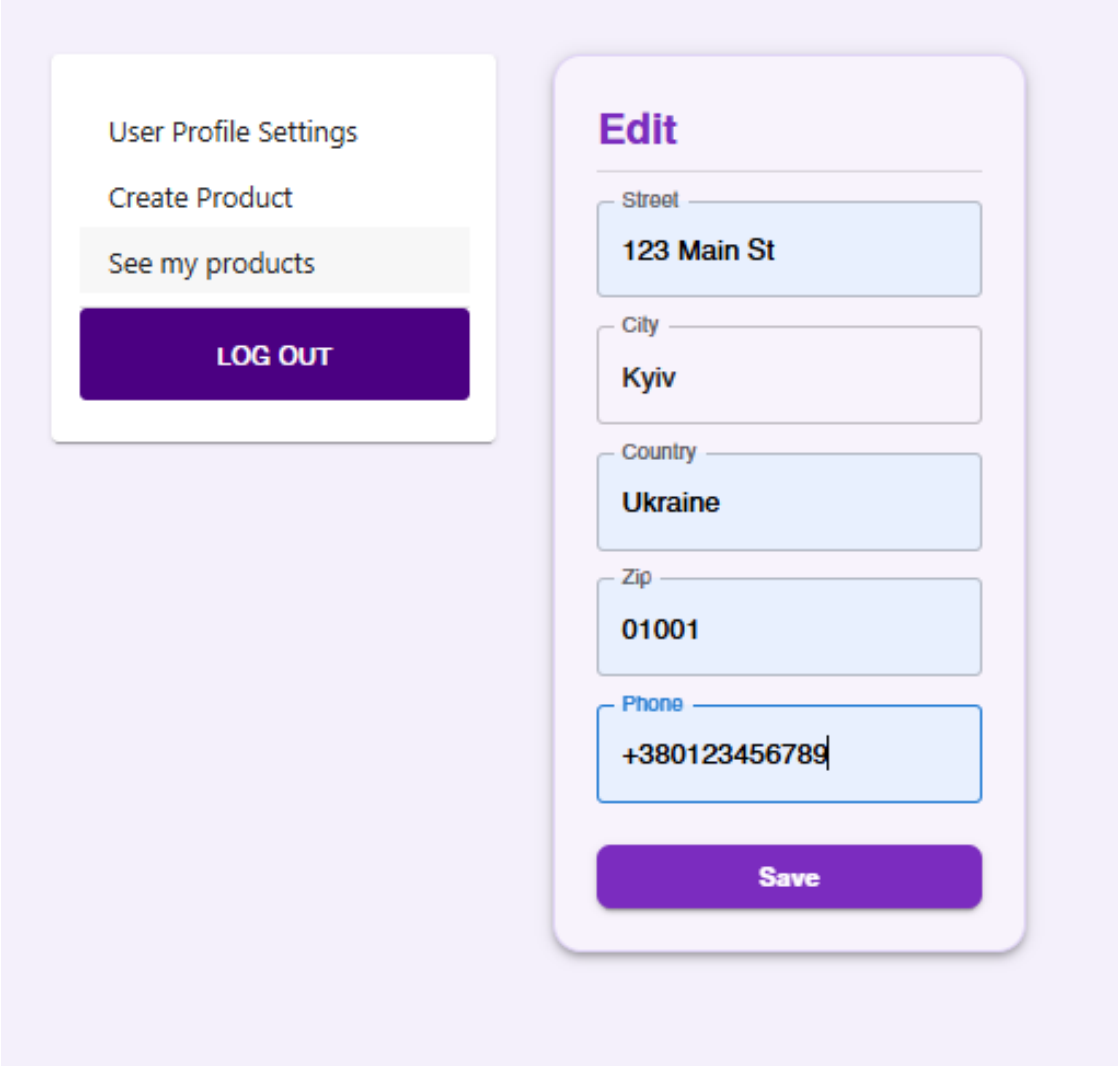

Рисунок 5.7 – Редагування даниз профілю користувача

Далі достатньо натиснути на кнопку «Зберегти», яка розміщується внизу форми редагування профілю, після чого внесені користувачем зміни будуть автоматично оброблені системою. У результаті цих дій оновлені дані зберігаються на сервері, і користувач автоматично бачить актуальну інформацію без потреби оновлення сторінки вручну. Таким чином, нові дані відображаються на сторінці профілю, що підтверджує успішне збереження змін і дозволяє переконатися у коректності введеної інформації. (рисунок 5.8).

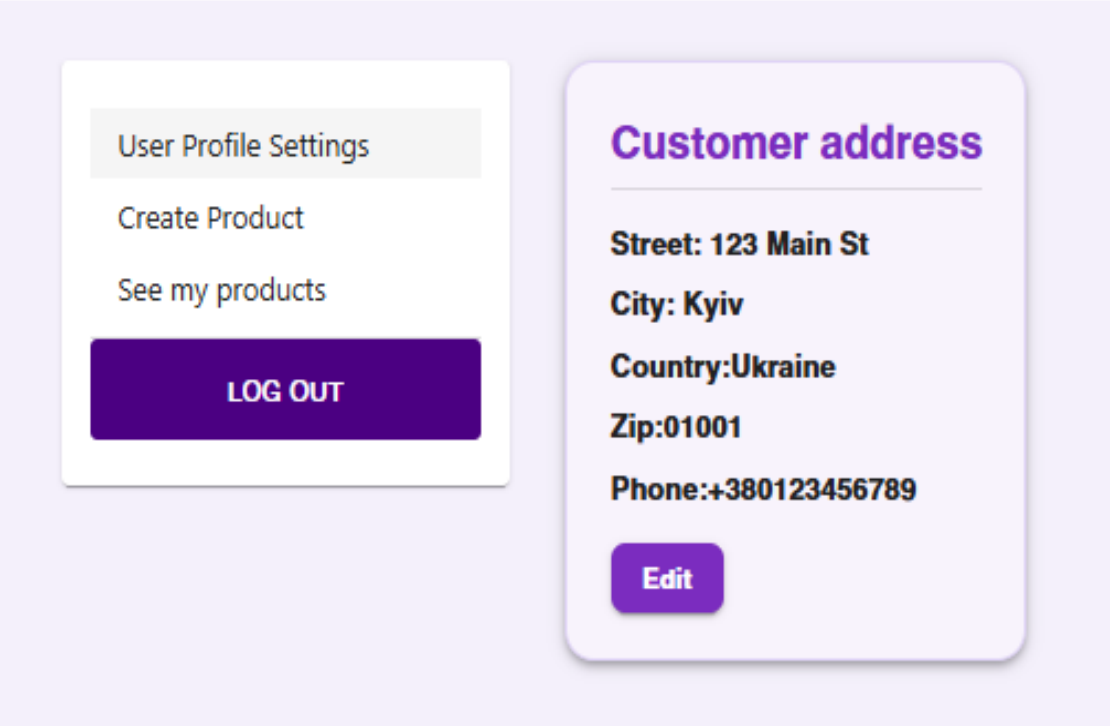

Рисунок 5.8 – Відображення редагованих даних користувача

На рисунку 5.8 також видно бокову панель навігації, яка є важливим елементом інтерфейсу користувача. Ця панель містить перелік доступних функціональних можливостей, які дозволяють здійснювати основні дії, необхідні для керування товарними позиціями та персональними даними. Зокрема, серед наявних опцій передбачено можливість редагування особистих даних продавця, створення нового товару для подальшого розміщення на платформі, а також перегляд переліку вже створених товарів з можливістю їх редагування або видалення.

Для демонстрації одного з таких функціональних сценаріїв взаємодії користувача з інтерфейсом розглянемо перехід до вкладки «Створити товар», яка відкривається при натисканні відповідної кнопки на боковій панелі. На рисунку 5.9 зображено зовнішній вигляд цієї вкладки, яка дозволяє заповнити необхідну інформацію про новий товар: назву, опис, ціну, категорію, а також завантажити

відповідне зображення. Це забезпечує зручний спосіб для продавця швидко додати нову товарну позицію до системи.

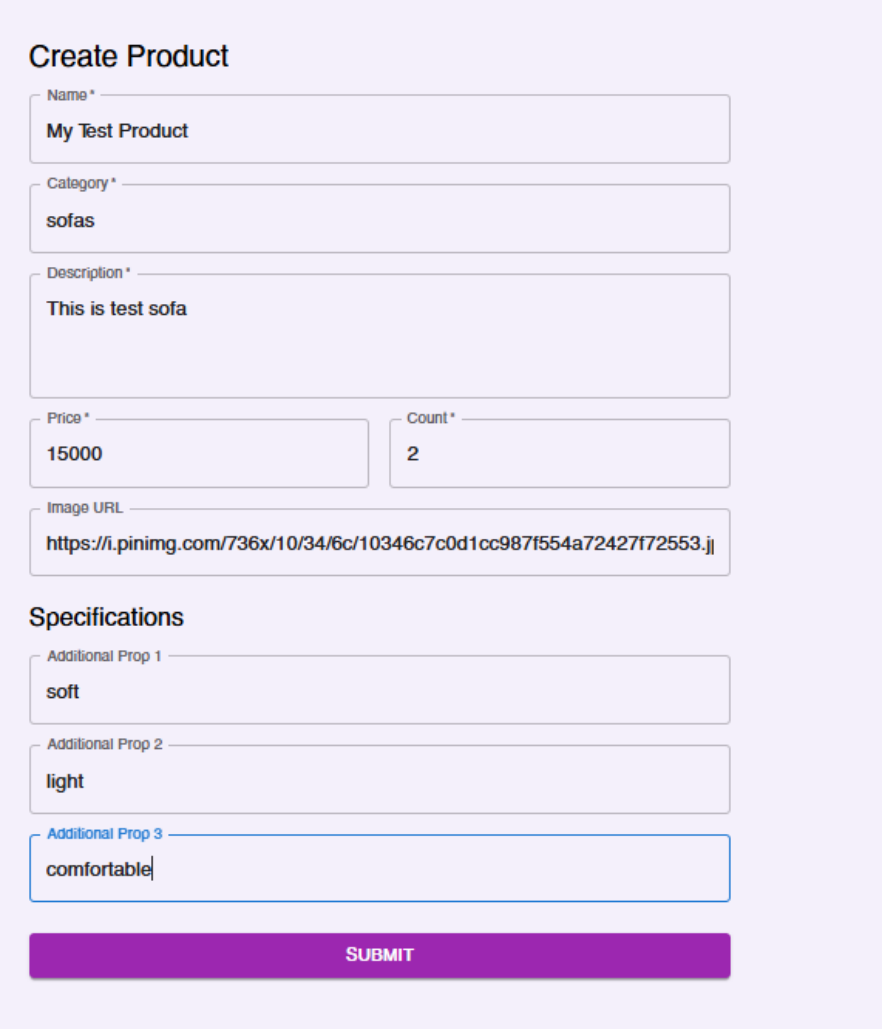

Рисунок 5.9 – Створення продукту

Після натискання на кнопку «відправити», у нас з’явиться модальне вікно, що сповіщає про успішне додання товару – рисунок 5.10.

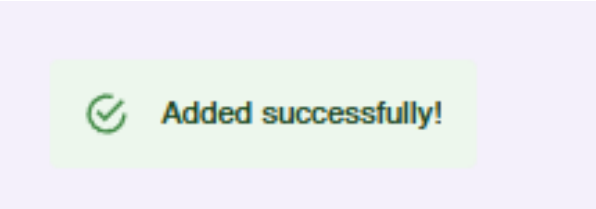

Рисунок 5.10 – Сповіщеня про успішне додання товару

Тепер можемо переглянути додані нами товари, перейшовши до владки «Переглянути мої товари» – рисунок 5.11.

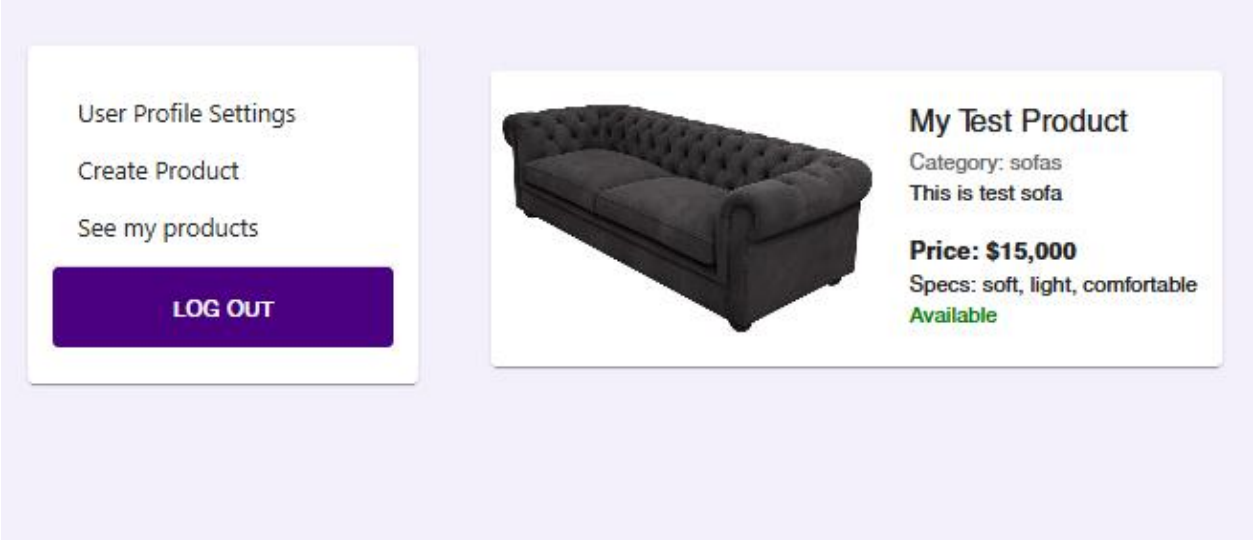

Рисунок 5.11 – Список створених користувачем товарів

Також ми можемо переглянути наш товар у відповідній категорії. Скористаємося для цього пошуком на напишемо назву нашого товару в полі для пошуку. Результати виконання показані на рисунку 5.12.

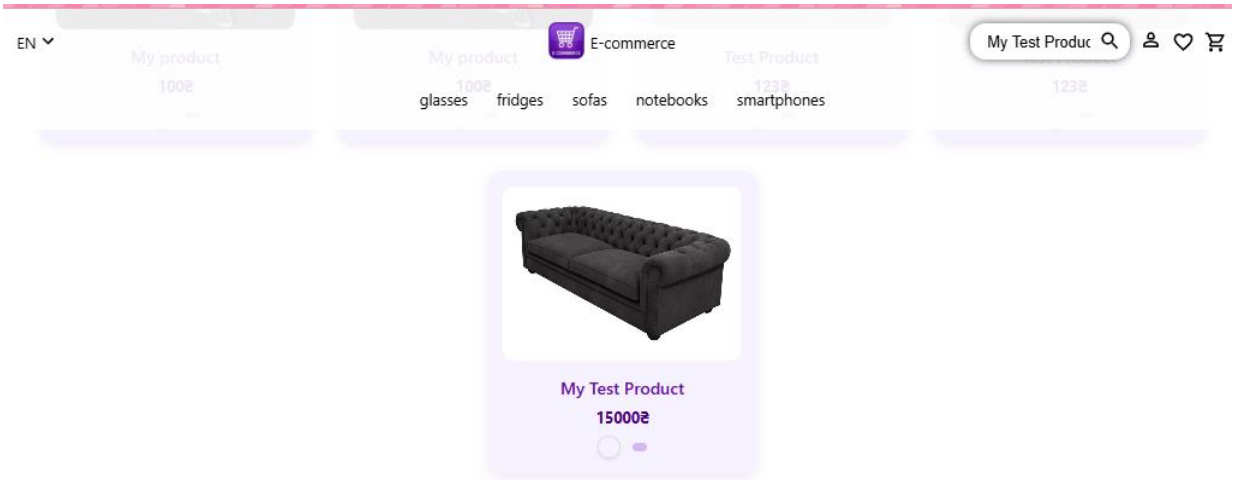

Рисунок 5.12 – Результати виконання пошуку в програмі

Також ми можемо знайти створений товар безпосередньо в категорії, серед інших товарів, як показано на рисунку 5.13.

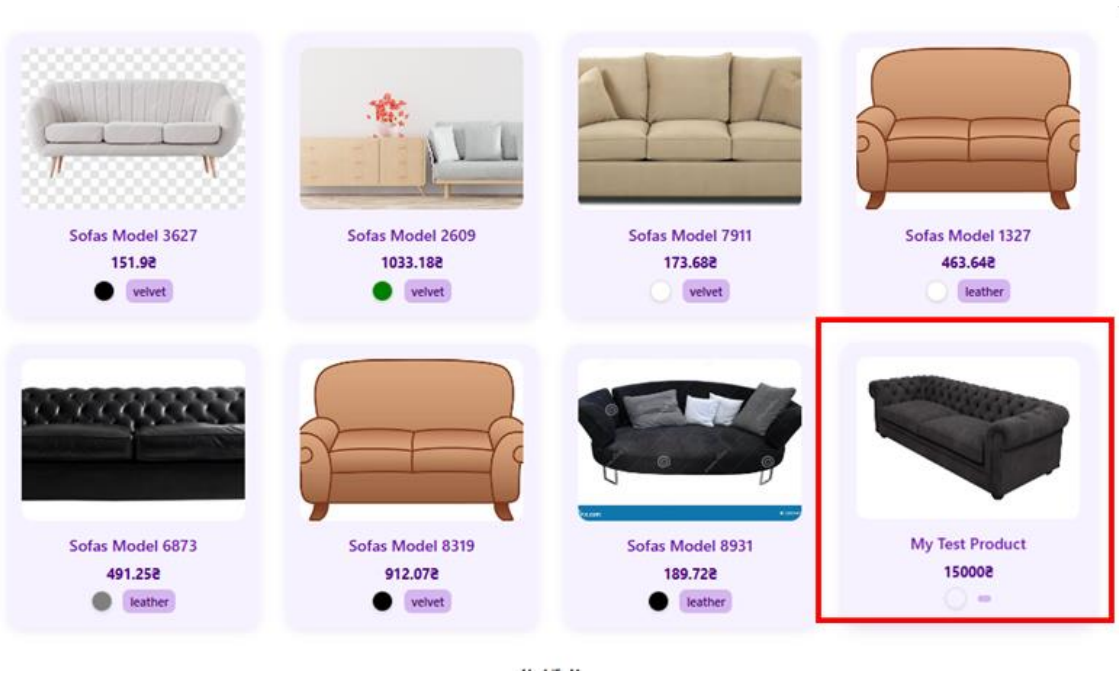

Рисунок 5.13 – Доданий товар серед інших в обраній категорії

Далі розглягнемо функціональні можливості звичайного користувача. Почнімо з того, що залогінимосі в його акаунт як показано на рисунку 5.14.

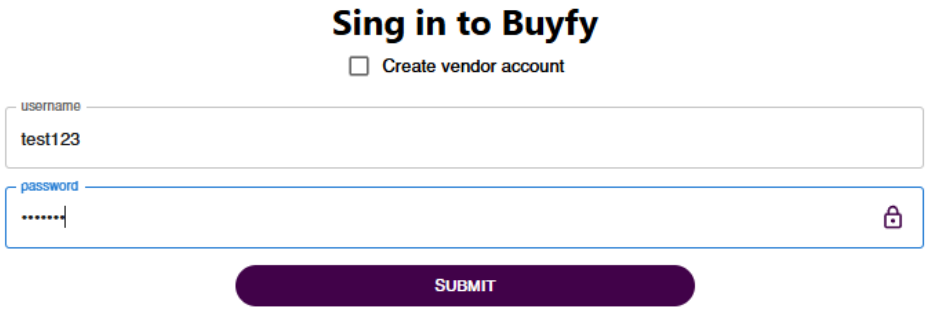

Рисунок 5.15 – Авторизація користувача в застосунку

Після успішної авторизації користувача перенаправляє на сторінку особистого кабінету (рисунок 5.16). Подібно до продавця у користувача є можливість редагувати власні дані та створювати товари (якщо є бажання продати вживаний товар гарної якості). Також тут у користувача є поле «Замовлення», де відображатимуться всі його замовлення.

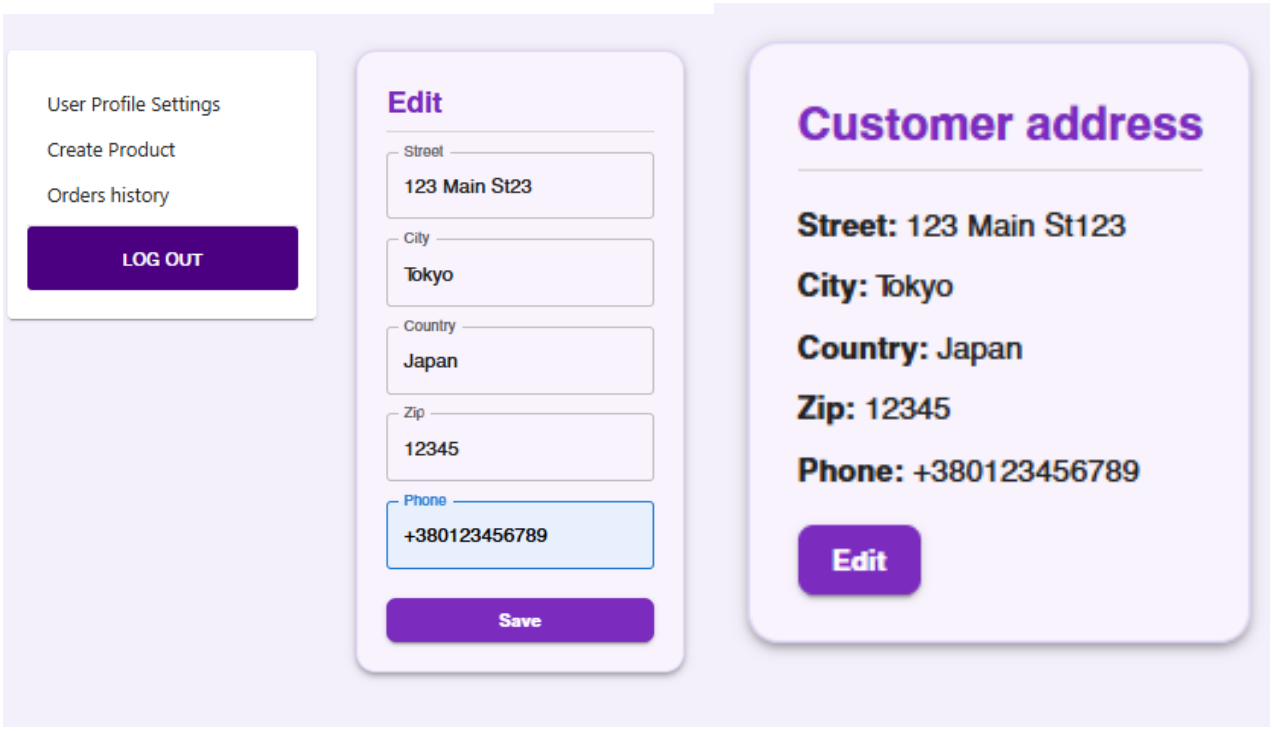

Рисунок 5.16 – Редагування даних користувача

Також як видно із меню для акаунту користувача, він може створювати продукти. Ці товари будуть відображатися окремо від тих, що створюють користувачі із роллю «продавці». Приклад створення товару показано на рисунку 5.17.

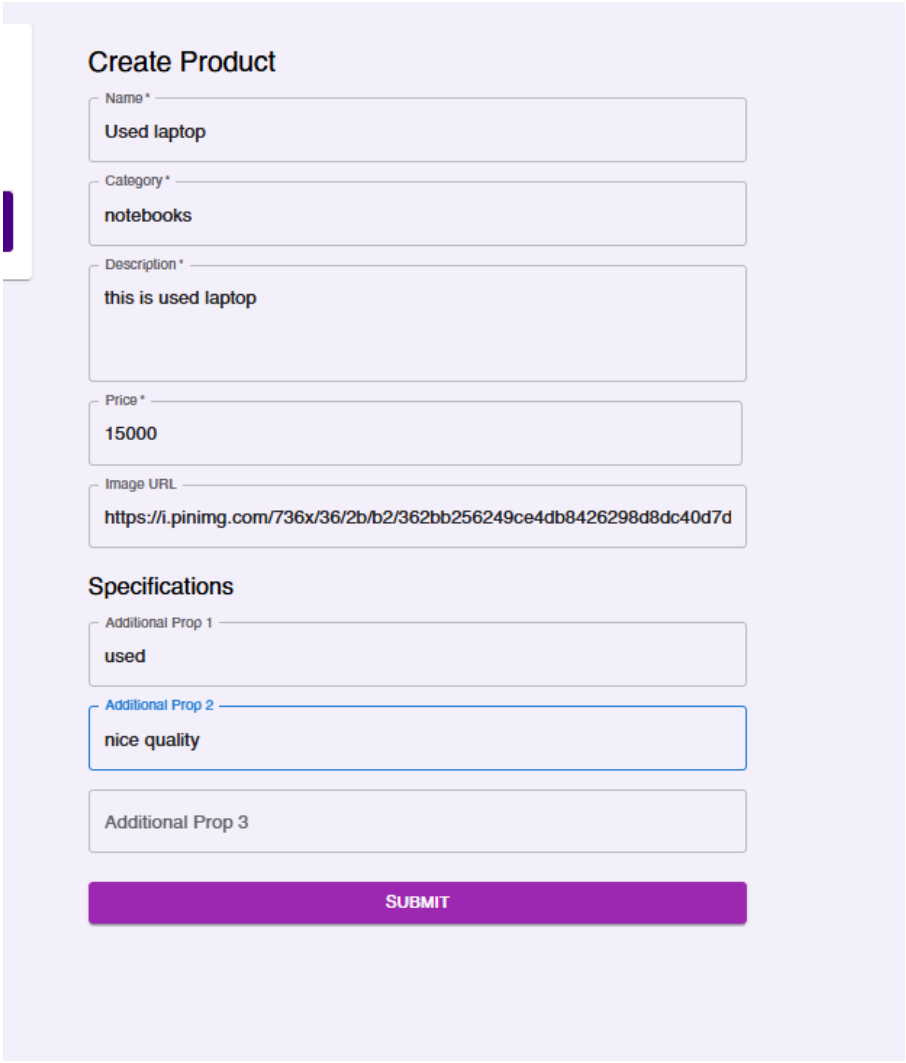

Рисунок 5.17 – Створення товару звичайним користувачем

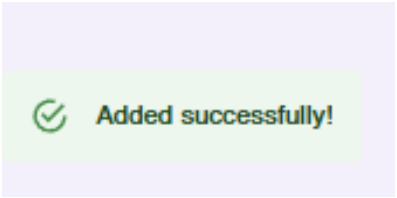

Рисунок 5.18 – Повідомлення про успішне створення

Що побачити створений товар, скористаємося пошуком, результати якого можна побачити на рисунку 5.19.

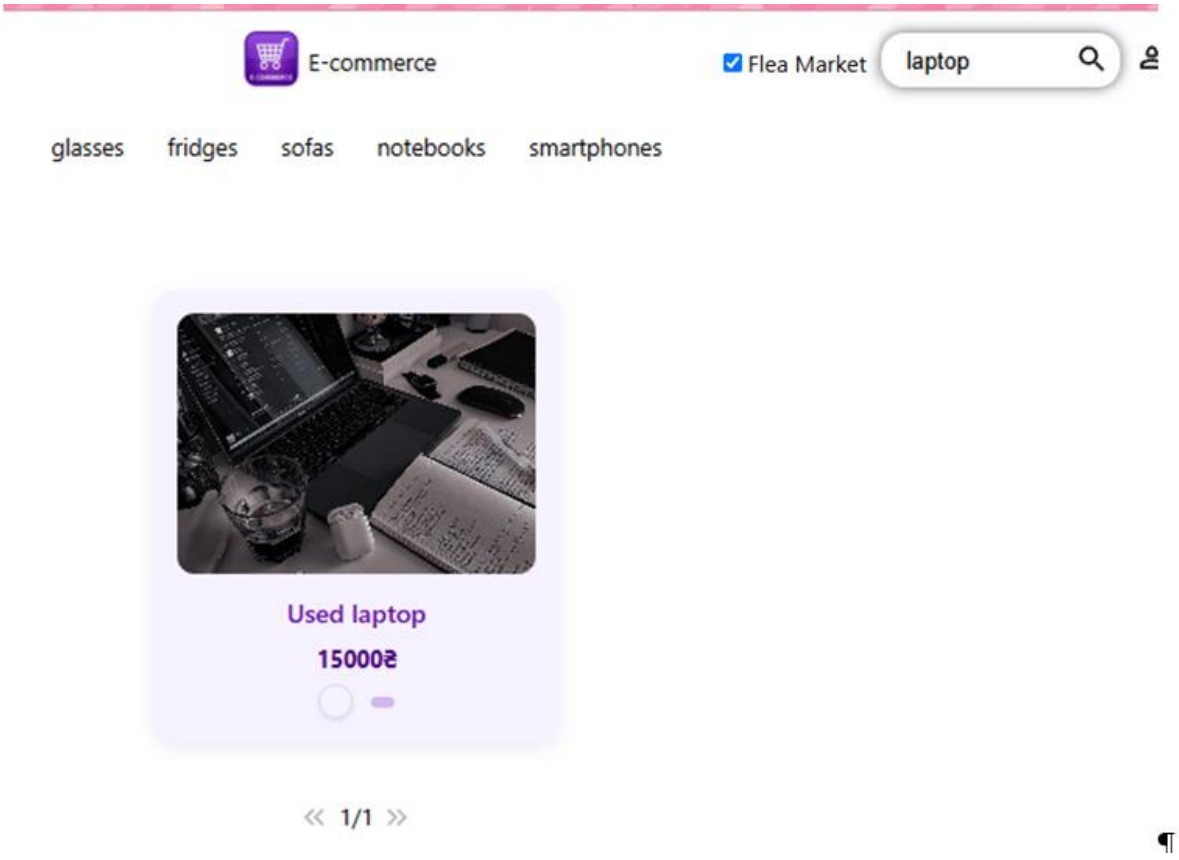

Рисунок 5.19 – Результати пошуку створеного товару користувачем

Також користувач може перейти на сторінку товару, щоб переглянути продукт (рисунок 5.20). На цій сторінці відображається детальна інформація про товар, включаючи назву, опис, зображення, ціну, доступну категорію. Це дозволяє користувачу ознайомитися з усією необхідною інформацією перед прийняттям рішення щодо покупки чи додавання товару до списку бажань.

Коли користувач переглядатиме сторінку створеного ним товару, кнопки «додати до кошика» та «додати до списку побажань» будуть заблоковані. Це зроблено з міркувань логіки платформи: немає сенсу додавати до кошика чи списку бажань товар, який сам користувач і виставив на продаж. Такий підхід покращує UX, оскільки запобігає плутанині в діях користувача та дозволяє чітко розділити ролі на платформі.

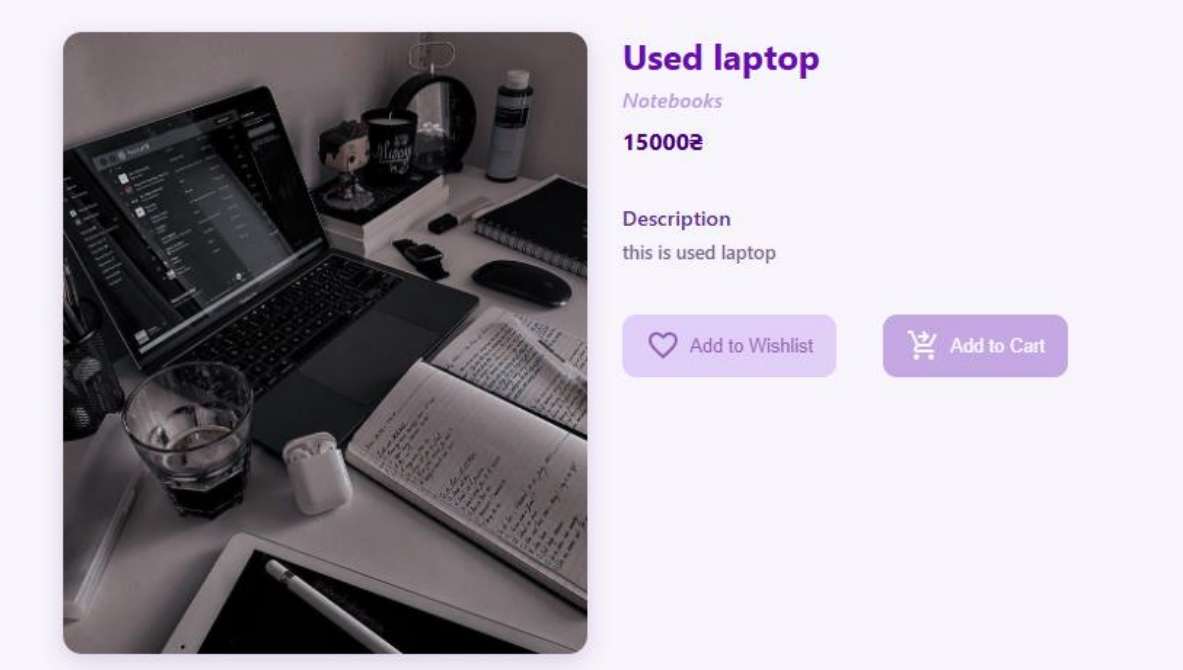

Рисунок 5.20 – Перегляд сторінку створеного товару користувачем

Також користувач може переглядати й інші товари, створені продавцями. Для цього йому достатньо відкрити наявні категорії або скористатись пошуком за ключовими словами, що значно полегшує навігацію по платформі. Категорії формуються на основі товарів, створених продавцями, що дозволяє підтримувати актуальну структуру асортименту.

Інтерфейс пошуку реалізовано таким чином, щоб користувач міг швидко знайти необхідні позиції за назвою, описом або іншими параметрами товару. Це створює зручні умови для ознайомлення з повним переліком пропозицій, наданих іншими користувачами з роллю продавця.

Приклад перегляду списку товарів відповідної категорії, доданої продавцями, показано на рисунку 5.21.

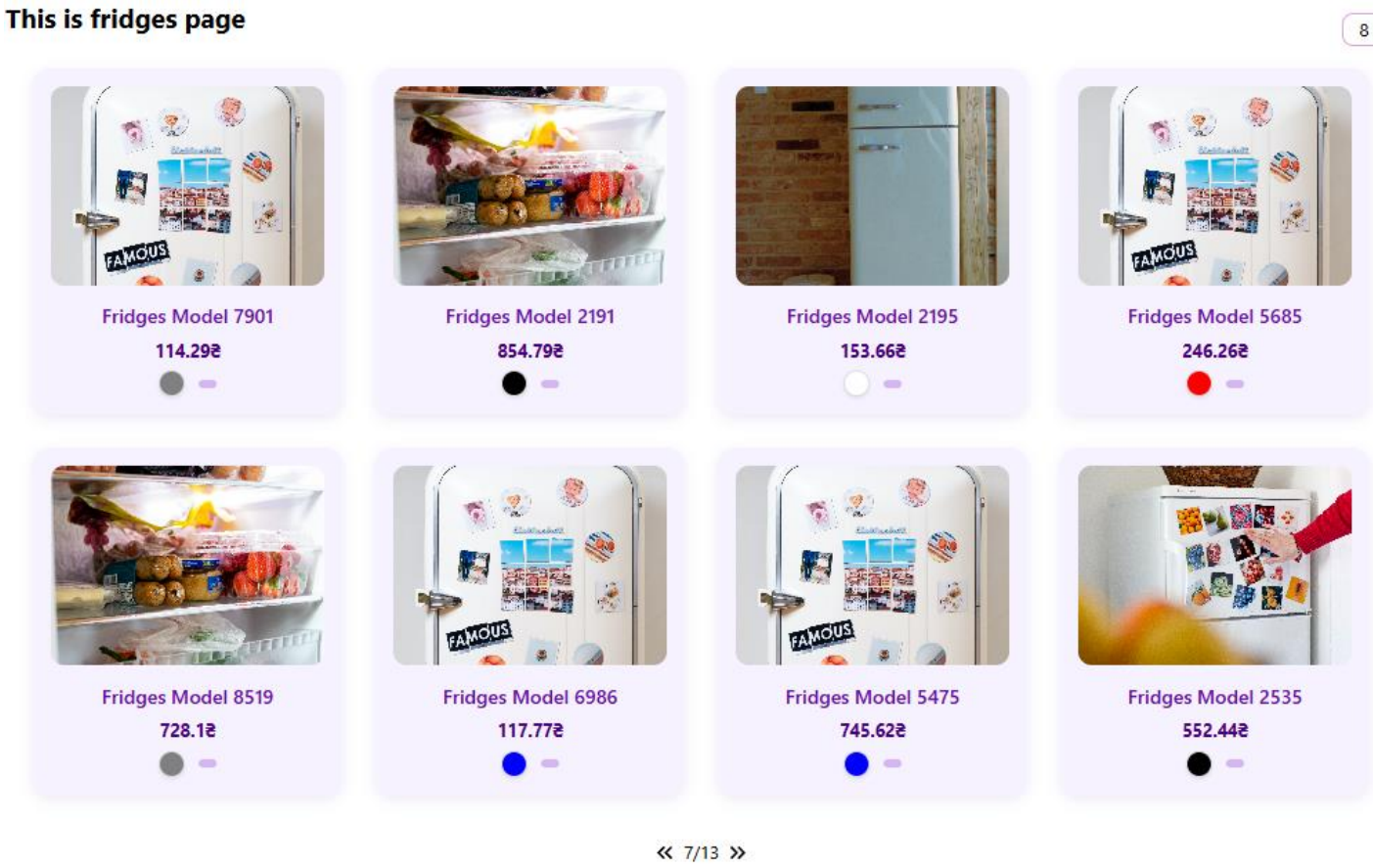

Рисунок 5.21 – Список товарі в категоії

На рисунку 5.21 можна побачити, як відображається кожен товар у вигляді окремої картки з зображенням, назвою, ціною. Завдяки такому підходу, платформа забезпечує інтуїтивну та привабливу взаємодію для кінцевого користувача.

Користувач також має можливість переглядати будь-який товар окремо, перейшовши безпосередньо на його детальну сторінку. У такому випадку кнопки з функцією «додати» до кошика або списку будуть активні та готові до використання, що дозволяє швидко і зручно здійснити додавання вибраного товару. Це можна побачити на прикладі, зображеному на рисунку 5.22, де чітко показано стан активності кнопок після переходу на сторінку товару.

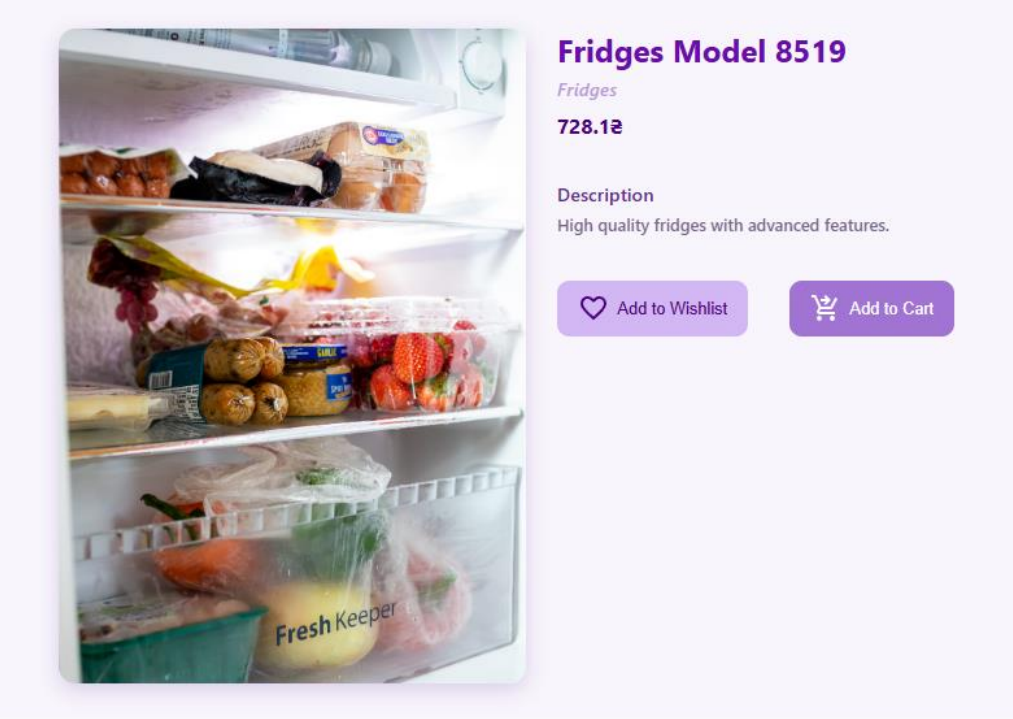

Рисунок 5.22 – Детальна сторінка продукту із доступними кнопками

Даний товар користувач може додати до корзини та списку побажань. Початкові стани корзини та списку побажань показані на рисунку 5.23.

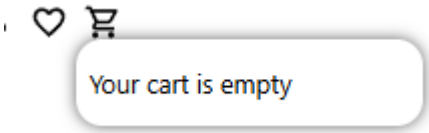

Рисунок 5.23 – Початкові стани списку побажань та корзини

Стани списку побажань та корзини, коли до них додані товари можна побачити на рисунку 5.24 та 5.25 відповідно.

Рисунок 5.24 – Активний стан списку побажань

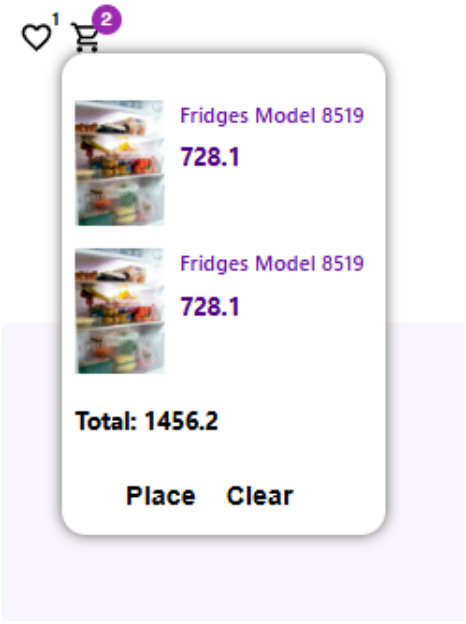

Рисунок 5.25 – Активний стан корзини, коли до неї додані товари

На рисунку 5.25 показан міні кошик, в якому відображені обрані користувачем товари. Окрім самих товарів тут ще показана загальна вартість та кнопки «очистити кошик» та «оформити замовлення».

При натисненні на «очистити кошик» всі товари з кошика буде видалено – рисунок 5.26.

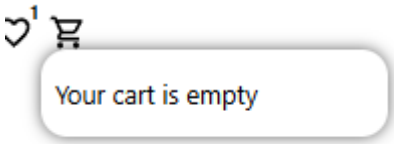

Рисунок 5.26 – Очищений кошик

Проте, якщо натиснути на кнопку «оформити замовлення», то корзина також очиститься, проте з’явиться модальне вікно про успішне оформлення замовлення – рисунок 5.27.

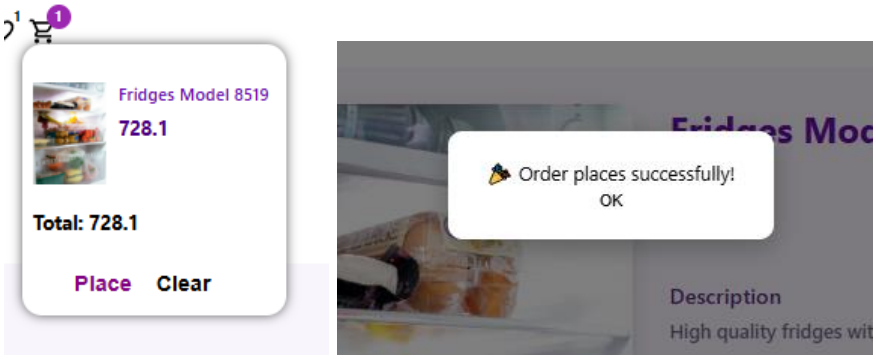

Рисунок 5.27 – Оформлення замовлення

Далі користувач може перейти до особистого кабінету та у вкладці «історія замовлень» переглянути замовлення, де відобразиться поточне замовлення із статусом «відкорито» – рисунок 5.28.

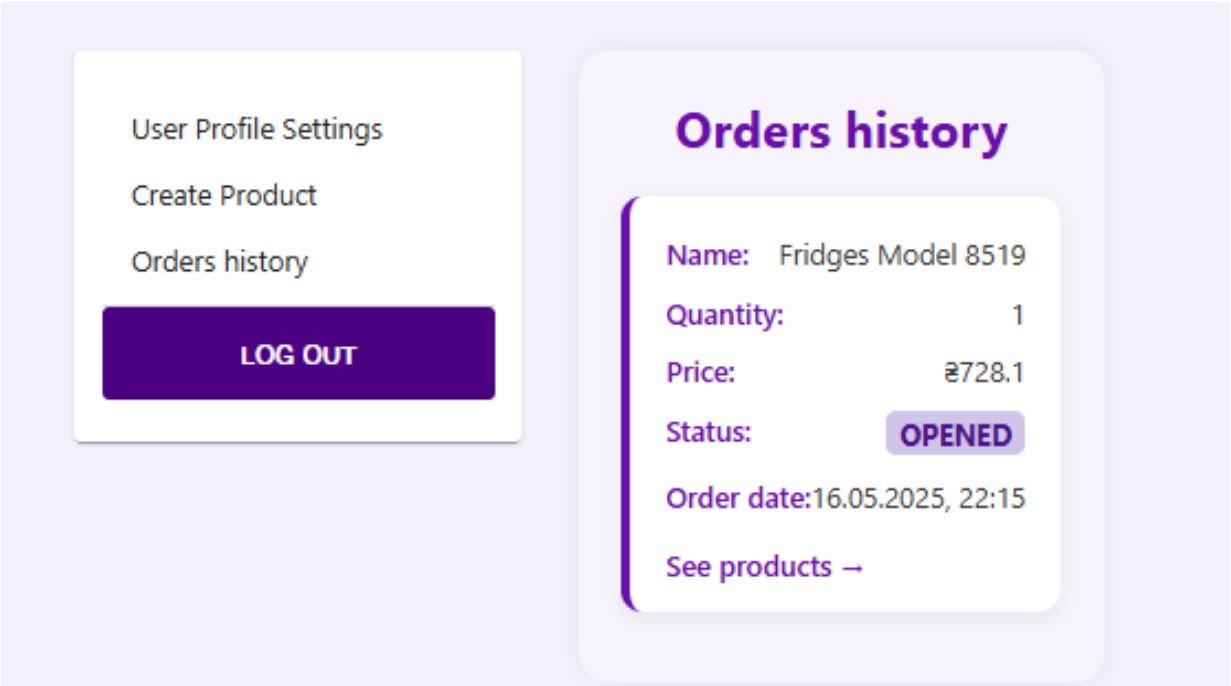

Рисунок 5.28 – Відображення історії замовлення користувача

Після виконання усіх необхідних йому дій користувач може вийти з ситеми, натиснувши на кнопку «вийти».

ВИСНОВКИ

У процесі виконання даної роботи було проведено всебічне дослідження сфери електронної комерції, зокрема українських маркетплейсів, таких як Rozetka, Prom, OLX та інші. Було детально проаналізовано ключові функціональні можливості, структуру інтерфейсів, моделі взаємодії користувачів та особливості бізнес-логіки, які притаманні найбільш популярним платформам. Це стало основою для формування концепції з чітким баченням архітектури та взаємодії між складовими системи, орієнтованої на зручність, гнучкість, безпеку та розширюваність.

У межах практичної частини було реалізовано клієнтську частину e-commerce застосунку, що була побудована із використанням сучасного frontend-стека. Основу інтерфейсу складає бібліотека React, яка забезпечує компонентний підхід до розробки та високу продуктивність за рахунок використання віртуального DOM. Для управління глобальним станом застосовано Redux, що забезпечує контрольованість, передбачуваність та масштабованість поведінки застосунку. Використання TypeScript дозволило значно підвищити надійність і читабельність коду завдяки статичній типізації та підтримці інструментів рефакторингу.

Система реалізує чітке розмежування прав доступу та функціональних можливостей для двох основних категорій користувачів – звичайних покупців та продавців. Завдяки цьому вдалося створити єдиний, але універсальний інтерфейс, що адаптується під роль користувача. Звичайні користувачі можуть створювати акаунти, авторизовуватись, переглядати товари за категоріями, використовувати пошук, додавати товари до кошика або списку побажань, здійснювати покупки та переглядати історію замовлень. Крім цього, реалізована можливість додавання товару звичайним користувачем, що дозволяє йому тимчасово виступати в ролі продавця – така функція підвищує гнучкість системи та залученість користувачів.

Продавець, у свою чергу, має ширші повноваження. Він може створювати нові товари, редагувати їх характеристики (назву, опис, категорію, ціну, фото тощо), переглядати свої позиції. Вся взаємодія з товарами організована через кабінет користувача, що дозволяє ефективно управляти асортиментом.

Особливу увагу при створенні застосунку було приділено розробці інтерфейсу відповідно до принципів UX-дизайну: структура сторінок логічна, навігація проста та інтуїтивна, візуальне оформлення гармонійне, що створює сприятливі умови для комфортної взаємодії користувача із системою. Значний акцент зроблено на локальну обробку даних та збереження стану, що мінімізує ризики втрати інформації при оновленні сторінки або нестабільності мережі.

У підсумку, створений застосунок є повноцінною клієнтською частиною системи електронної комерції, яка задовольняє сучасні вимоги як до функціональних можливостей, так і до зручності користування. Він не лише демонструє приклад ефективної реалізації складного інтерфейсу на основі передових технологій, але й враховує потреби цільової аудиторії та можливості подальшого масштабування. Проєкт може бути доповнений в майбутньому доданням функції чату для обміну повідомленнями між продавцями та покупцями, інтеграцію з платіжними системами чи CRM, а також внутрішньою системою доставки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nielsen J. E-commerce user experience. Fremont, CA: Nielsen Norman Group, 2001. 424 p.

2. Kaufman J. The Personal MBA. London: Penguin, 2020. 496 p.

3. The 6 best eCommerce website building platforms for online stores in 2025. Zapier: веб-сайт. URL: https://zapier.com/blog/best-ecommerce-shopping-cart-software/ (дата звернення: 15.04.2025).

4. The 9 Best Ecommerce Platforms to Consider for Your Online Store. BigCommerce: веб-сайт. URL: https://www.bigcommerce.com/articles/ecommerce/ecommerce-platforms/ (дата звернення: 15.04.2025).

5. Ecommerce Platform Comparison Tool. Tooltester: веб-сайт. URL: https://www.tooltester.com/en/ecommerce-platforms/ (дата звернення: 15.04.2025).

6. Adobe Commerce (Magento): B2B & B2C Enterprise Solutions. Adobe: веб-сайт. URL: https://business.adobe.com/products/magento (дата звернення: 15.04.2025).

7. Ukrainian e-Commerce Trends in 2021. Good Time Invest: веб-сайт. URL: https://good-time-invest.com/blog/ukrainian-e-commerce-trends-in-2021/ (дата звернення: 16.04.2025.

8. Most Visited Retail Websites in Ukraine 2025. Semrush: веб-сайт. URL: https://www.semrush.com/trending-websites/ua/retail (дата звернення: 17.04.2025).

9. Rippon C. Learn React with TypeScript. Mumbai : Packt Publishing, 2023. 474 p.

10. Vanderkam D. Effective TypeScript. Sebastopol : O’Reilly, 2024. 569 p.

11. Manglurkar K. TypeScript 5 Design Patterns and Best Practices. Sebastopol: O’Reilly Media, 2025. 419 p.

12. Banks A., Porcello E. Learning React FUNCTIONAL WEB DEVELOPMENT WITH REACT AND REDUX. Sebastopol : O'Reilly, 2017. 350p.

13. General Redux. Redux: веб-сайт. URL: https://redux.js.org/faq/general (дата звернення: 18.04.2025).

14. Understanding Redux. LogRocket: веб-сайт. URL: https://blog.logrocket.com/understanding-redux-tutorial-examples/ (дата звернення: 19.04.2025).

15. Getting Started. Axios: веб-сайт. URL: https://axios-http.com/docs/intro (дата звернення: 18.04.2025).

16. React Router Home. React Router: веб-сайт. URL: https://reactrouter.com/home (дата звернення: 19.04.2025).

17. Material UI – Overview. Material UI: веб-сайт. URL: https://mui.com/material-ui/getting-started/ (дата звернення: 19.04.2025).

18. Gasparyan D. Feature-Sliced Design: The Ideal Frontend Architecture. Medium: веб-сайт. URL: https://medium.com/@dtgasparyan/feature-sliced-design-the-ideal-frontend-architecture-84d701ad44ba (дата звернення: 18.04.2025).

19. Feature-Sliced Design: методологія архітектури для фронтенд-проєктів. Feature-Sliced Design: веб-сайт. URL: https://feature-sliced.github.io/documentation/docs (дата звернення: 18.04.2025)

20. Abele F. Feature-Sliced Design and what we need for good frontend architecture. URL :https://www.codecentric.de/en/knowledge-hub/blog/feature-sliced-design-and-good-frontend-architecture(дата звернення: 19.04.2025).

ДОДАТОК А

ТЕКСТ ПРОГРАМНОГО МОДУЛЯ

«РЕАЛІЗАЦІЯ СТОРІНКИ КАТАЛОГУ ТОВАРІВ»

УРК. НТУУ «КПІ ім. Ігоря Сікорського» _ІАТЕ_ЦТЕ_ТР-11

Аркушів 5

Київ-2025

Програмні засоби:

– мова програмування – typescript;

– фреймворк для побудови інтерфейсів – react;

– маршрутизація сторінок – react router;

– стилізація – css-модулі;

– використання іконок – Material UI.

```
import React, { useEffect, useState } from "react";
import { useLocation, useNavigate, useParams } from "react-router-dom";
import ProductService from "../../shared/api/Product/product";
import ProductCard from "../../widgets/Product";
import './styles.css';
import { KeyboardDoubleArrowLeft, KeyboardDoubleArrowRight } from "@mui/icons-material";
function CategoryPage() {
  let { categoryName } = useParams();
  const location = useLocation();
  const navigate = useNavigate();
  const [products, setProducts] = useState<any[]>([]);
  const [currentPage, setCurrentPage] = useState(1);
  const [itemsPerPage, setItemsPerPage] = useState(8);
  const [totalPages, setTotalPages] = useState(0)
  const queryParams = new URLSearchParams(location.search);
  const page = parseInt(queryParams.get("page") || currentPage.toString(), 10);
  const limit = parseInt(queryParams.get("limit") || itemsPerPage.toString(), 10);
  useEffect(() => {
    setCurrentPage(page);
    setItemsPerPage(limit);
  }, [location.search]);
  useEffect(() => {
```

```
  const queryParams = new URLSearchParams(location.search);
 let updated = false;
 if (!queryParams.has("page")) {
  queryParams.set("page", currentPage.toString());
  updated = true;
 }
 if (!queryParams.has("limit")) {
  queryParams.set("limit", itemsPerPage.toString());
  updated = true;
 }
 if (updated) {
  navigate({ search: queryParams.toString() }, { replace: true });
 }
  const fetchProducts = async () => {
   if (categoryName) {
 const fetchedProducts = await
ProductService.getProductByCategoryName(categoryName);
    setTotalPages(Math.ceil(fetchedProducts.length / itemsPerPage));
     const initialIndex = (currentPage - 1) * itemsPerPage;
     const finishIndex = initialIndex + itemsPerPage;
     setProducts(fetchedProducts.slice(initialIndex, finishIndex));
   }
  };
  const fetchProductsSearch = async () => {
  const q = queryParams.get("q");
  if (q) {
   const searchResults = await ProductService.getSearchInformation(q);
   setTotalPages(Math.ceil(searchResults.length / itemsPerPage));
   const initialIndex = (currentPage - 1) * itemsPerPage;
```

```
        const finishIndex = initialIndex + itemsPerPage;
        setProducts(searchResults.slice(initialIndex, finishIndex));
      } else if (categoryName) {
    const fetchedProducts = await ProductService.getProductByCategoryName(categoryName);
        setTotalPages(Math.ceil(fetchedProducts.length / itemsPerPage));
        const initialIndex = (currentPage - 1) * itemsPerPage;
        const finishIndex = initialIndex + itemsPerPage;
        setProducts(fetchedProducts.slice(initialIndex, finishIndex));
      }
    };
      fetchProducts();
      fetchProductsSearch()
    }, [categoryName, currentPage, itemsPerPage, window.location.href]);
    const updateItemsInUrl = (newLimit: number) => {
      queryParams.set("limit", newLimit.toString());
      queryParams.set("page", "1");
      navigate({ search: queryParams.toString() });
    };
    const handlePerItemsUserSet = (value: number) => {
      setItemsPerPage(value);
      updateItemsInUrl(value);
    };
    const updatePageInUrl = (newPage: number) => {
        queryParams.set("page", newPage.toString());
        navigate({ search: queryParams.toString() });
    };
      return (
        <div className="page-container">
```

```
<h2>This is {categoryName} page</h2>
<div className="control-wrapper">
  <div className="buttons-control">
  <ul className="per-page-items">
    {[8, 16, 24].map((value, index) => (
      <li
       key={value}
        className={itemsPerPage === value ? "active" : ""}
        onClick={() => handlePerItemsUserSet(value)}
      >
        {value}
      </li>
    ))}
  </ul>
</div></div>
<ul className="plp">
  {products.map((product, index) => (
    <ProductCard
      key={index}
      id={product.id}
      name={product.name}
      price={product.price}
      imageUrl={product.imageUrl}
      specifications={product.specifications}
    />
  ))}
</ul>
<div className="pagination">
   <button
```

```
    onClick={() => updatePageInUrl(currentPage - 1)}
    disabled={currentPage === 1}
   >
    <KeyboardDoubleArrowLeft/>
   </button>
   <span>{currentPage}/{totalPages}</span>
   <button
    onClick={() => updatePageInUrl(currentPage + 1)}
    disabled={currentPage === totalPages}
   >
    <KeyboardDoubleArrowRight/>
   </button>
      </div>
    </div>
   );
}
export default CategoryPage;
```

ДОДАТОК Б

АПРОБАЦІЇ

УРК. НТУУ «КПІ ім. Ігоря Сікорського» _ІАТЕ_ЦТЕ_ТР-11

Аркушів 3

Київ-2025

UDC 004.8:004.93:519.6:6

BS's programme 4th year Shust K.E.
BS's programme 4th year Bratchykova N.V.
[1] Prof., doc.eng.sc. Shushura O.M.
https://scholar.google.com.ua/citations?user=beUGko0AAAAJ&hl=uk&oi=sra

[1] Igor Sikorsky Kyiv Polytechnic Institute

DEVELOPMENT OF AN E-COMMERCE SYSTEM FOR ONLINE TRADING

Statement of the problem and its relevance. With the active development of e-commerce, online platforms are becoming a key tool for the sale of goods and services. The main challenges in creating such systems are ensuring a convenient user interaction with the platform, automating order processing, and the scalability of the architecture. Special attention should be paid to the organization of the seller's cabinet, which must provide functionality for adding products, viewing sales statistics, and managing orders.

The relevance of this work is determined by the need to create a universal e-commerce system that combines functionality for both buyers and sellers, considering modern approaches to software architecture design..

Analysis of the latest research. Currently, there are many ready-made solutions for creating online stores, such as Shopify, WooCommerce, and OpenCart. Most of them are focused on creating monolithic applications or use outdated architectural approaches.

Modern approaches to building web applications, such as feature-sliced design on the frontend and module-driven architecture on the backend, significantly improve scalability, responsibility distribution, and code maintainability.

In [1], a modular architecture approach for the backend based on Spring Boot is presented, which provides flexible integration of new features. Research [2] describes the concept of the feature-sliced approach in frontend development for better structuring of business logic.

Goal formulation. The aim of this work is to create a comprehensive e-commerce system that provides a seamless and efficient experience for both buyers and sellers. Buyers will be able to browse and search for products, add items to their cart, place orders, make secure payments, and track their order history. Meanwhile, sellers will have access to a dedicated management panel that allows them to list and update products, monitor inventory, analyze sales performance, and receive detailed reports on customer behavior and market trends. Additionally, the system will incorporate modern architectural principles to ensure scalability, security, and ease of maintenance.

The main part.

The developed system consists of two independent parts:

1. Frontend – implemented using React with the feature-sliced design approach, which divides functionality into domains (features, entities, widgets, shared).
2. Backend – built on Spring Boot based on the module-driven architecture, where each functional module is responsible for a specific business process.

The system provides comprehensive functionality for both users and sellers. It includes user registration and authorization, allowing users to create accounts and protect personal data. Registered users can view their order history, manage personal information by updating contact details and profile settings. For convenient product search, the system implements keyword search and product filtering by categories. The system also offers product recommendations based on the user's previous purchases.

Sellers gain access to a personal dashboard where they can add, edit, and delete products, as well as view sales statistics to analyze their business performance.

The system features a convenient order processing mechanism, including the selection of payment and delivery methods. To enhance user interaction, a feedback and product rating functionality is implemented. The seller's dashboard includes tools for monitoring product availability and managing prices..

Figure 1 shows the overall system workflow diagram:

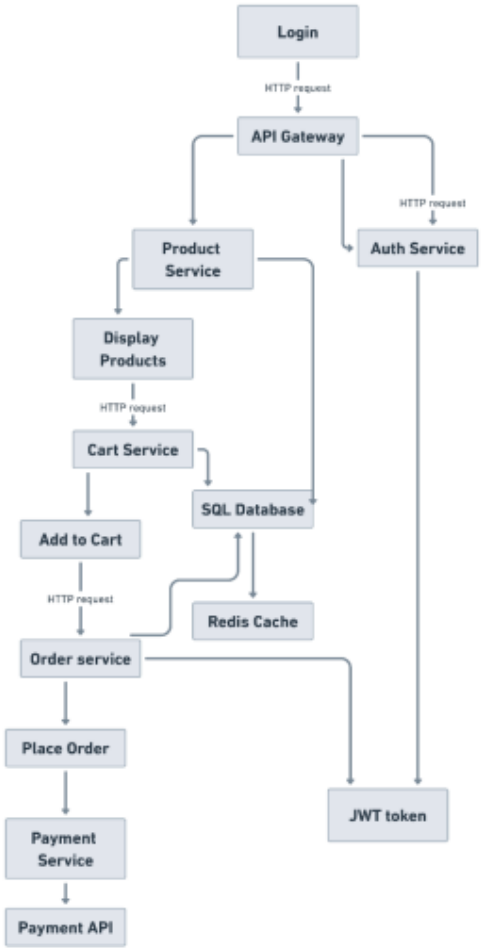

Figure 1 – Generalized diagram of the online trading system

Key points include:

1. Frontend (React, Redux, React Query)
   a) Feature-sliced structure;
   b) Interaction through Axios;

c) Query caching through Redux;
d) Authorization via JWT.

2. Backend (Spring Boot + Microservices Architecture)
   a) Security Layer for access contro;
   b) Separate services for authentication, products, cart, orders, and payments;
   c) Database (SQL);
   d) Redis for product caching;
   e) API protection through JWT.
   f) Integration with third-party payment services.
3. Main interaction scenarios:
   a) Authorization (Login);
   b) Retrieving the product list;
   c) Adding a product to the cart;
   d) Placing an order;
   e) Payment confirmation.

The e-commerce system is built on a client-server architecture with interaction through a REST API. The React frontend uses a feature-sliced approach, Axios for HTTP requests, and caching via Redux. Authorization is implemented using JWT tokens.

The Spring Boot backend follows a modular architecture, where each service is independently responsible for core functionalities such as authentication, product management, cart handling, order processing, and payments. This modularity allows for easy scalability and flexibility in extending or modifying services. Data is stored in an SQL database, providing reliable and structured storage, while Redis is used for caching products to enhance the speed and efficiency of data retrieval. API security is maintained through JWT, protecting against unauthorized access and ensuring secure data transmission.

The system is designed to handle key scenarios such as user authorization, product viewing, cart management, order placement, and payment processing. It ensures smooth user interactions by validating input data, storing necessary information in the database, and updating order statuses in real time. Additionally, the modular approach allows for continuous improvement and the seamless integration of new features as the e-commerce platform evolves.

Conclusions. The e-commerce system ensures secure and efficient interaction between users and the platform through modern technologies. It supports registration, authorization, product search and filtering, as well as personal data management. To enhance user experience, product recommendations, order history viewing, and a seller's personal dashboard for managing products and sales statistics are implemented. Integration with payment services and API protection through JWT make the system fully functional for online commerce.

References:

1. DDD Bounded Contexts and Java Modules. URL: https://www.baeldung.com/java-modules-ddd-bounded-contexts (access date 14.02.2025).

2. Feature-Sliced Design. Architectural methodology for frontend projects. URL: https://feature-sliced.design/ (access date 20.02.2025).