



Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра
**Інформаційна система управління життєвим циклом електронного
маркетингу (за зразком Amazon)**
на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту:

№	П.І.Б.	Ролі	Група
1	Бабчинський Олексій Павлович	Backend / DevOps	КН-П-213
2	Франчук Богдан Максимович	Backend / Frontend	КН-П-211
3	Мо Ассунсау Іванна	Frontend	КН-П-211

Автор звіту: Мо Ассунсау Іванна

Керівник	Доцент кафедри ІТ та ФП, к.т.н., Суліма М.М.
----------	---

АНОТАЦІЯ

Мо Ассунсау І. Система управління контентом електронної комерції (на прикладі сервісу Amazon). – випускна кваліфікаційна робота на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». – Одеський технологічний університет «ШАГ», Одеса, 2025.

Робота присвячена розробці та впровадженню клієнтської частини інформаційної системи електронної комерції, орієнтованої на забезпечення зручної, функціональної та адаптивної взаємодії користувача з онлайн-магазином. Основним завданням стало створення сучасного вебінтерфейсу, який дозволяє користувачам здійснювати пошук товарів, перегляд характеристик, управління кошиком, оформлення замовлень, авторизацію та персоналізацію профілю.

Актуальність теми визначається стрімким розвитком цифрової торгівлі, зростанням попиту на онлайн-платформи для продажу товарів, а також необхідністю забезпечення стабільного, масштабованого та зручного вебінтерфейсу для кінцевого користувача. Основні вимоги до таких систем включають швидкодію, адаптивність, надійність обробки даних та зручність навігації на різних пристроях.

Об'єктом дослідження виступає клієнтська частина багаторівневої вебсистеми, що реалізує функціональність сучасного інтернет-магазину. Предметом дослідження є методи організації логіки інтерфейсу, керування станом застосунку, а також взаємодії з серверною частиною для забезпечення коректної роботи всіх бізнес-процесів.

У рамках реалізації було вирішено такі задачі:

- обрати відповідний стек технологій для забезпечення типізації, перевірки даних та стабільності розробки;
- спроектувати архітектуру клієнтської частини за принципами компонентності та повторного використання коду;
- реалізувати основні функціональні сторінки: головну, каталог товарів, товарну картку, кошик, оформлення замовлення, профіль користувача;
- створити систему керування станом застосунку з централізованим зберіганням даних;
- впровадити механізми валідації форм і обробки помилок на

клієнтській стороні;

- протестувати інтерфейс на різних пристроях та забезпечити його коректне відображення відповідно до вимог адаптивності.

У процесі роботи були застосовані сучасні підходи до побудови SPA-застосунків, включно з використанням бібліотеки компонентів, системи типізації, інструментів для збирання та перевірки коду. Інтерфейс спроектовано з урахуванням принципів UI/UX-дизайну, що забезпечує зручну та логічну взаємодію користувача із системою.

Методи дослідження включали аналіз існуючих підходів до створення вебінтерфейсів, практичне застосування клієнт-серверної архітектури, а також модульне тестування та адаптацію інтерфейсу під різні платформи. Результатом є стабільна, функціональна та гнучка система, придатна до подальшого масштабування та інтеграції з іншими підсистемами електронної комерції.

ЗМІСТ

ЗМІСТ	4
ВСТУП	5
ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ	6
РОЗДІЛ 1	
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ	10
РОЗДІЛ 2	
РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ	12
РОЗДІЛ 3	
ОЦІНКА	18
ВИСНОВКИ	21
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	23

ВСТУП

Стрімкий розвиток цифрових технологій та поширення електронної комерції змінюють уявлення про сучасну торгівлю. Онлайн-магазини стали невід'ємною частиною глобального ринку, відкриваючи нові можливості для бізнесу і створюючи нові вимоги до якості користувацького інтерфейсу. Важливою складовою успіху таких платформ є ефективно реалізована фронтенд-частина – зовнішній вигляд і логіка взаємодії з користувачем, що безпосередньо впливає на зручність навігації, швидкість роботи сайту та загальний користувацький досвід.

Фронтенд сучасного інтернет-магазину має відповідати високим стандартам: бути адаптивним, інтуїтивно зрозумілим і швидким у завантаженні. Візуальний вигляд, логіка фільтрації товарів, інтеграція з API для відображення актуальних даних – усе це є ключовими елементами взаємодії користувача з платформою. Успішна реалізація таких компонентів дозволяє підвищити ефективність онлайн-продажів та забезпечити позитивний досвід використання сайту.

У межах даної дипломної роботи було реалізовано фронтенд-частину інтернет-магазину з повноцінною підтримкою основних функцій електронної комерції. Основою для розробки став дизайн-макет, наданий у Figma, відповідно до якого було створено інтерфейс користувача з урахуванням принципів адаптивності та доступності. Проєкт охоплює реалізацію головної сторінки магазину, каталогу товарів, сторінки продукту, кошика, форми оформлення замовлення, а також елементів навігації та фільтрації.

Метою даного дослідження є створення зручного, ефективного та візуально привабливого інтерфейсу користувача для системи електронної комерції, який відповідає сучасним вимогам і підвищує якість взаємодії з платформою. У роботі розглянуто ключові аспекти фронтенд-розробки: структурування компонентів, використання бібліотек для управління станом, обробка запитів до API, а також забезпечення адаптивності дизайну для різних типів пристроїв.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

Загальний опис

Програмне забезпечення (ПЗ) реалізує клієнтську частину (Frontend) вебзастосунку для електронної комерції. Інтерфейс забезпечує взаємодію користувача з функціоналом онлайн-магазину через браузер, дозволяє переглядати товари, працювати з кошиком, керувати власним профілем, оформлювати замовлення та переглядати історію покупок.

Застосунок розроблений з урахуванням сучасних принципів побудови масштабованих односторінкових інтерфейсів (Single Page Application), із застосуванням TypeScript для статичної типізації, Redux для управління станом, Vite як інструменту збірки, а також SCSS для модульного стилізування компонентів.

Призначення

ПЗ орієнтоване на забезпечення зручної взаємодії між кінцевим користувачем (покупцем) та клієнтською частиною системи електронної комерції шляхом відображення, фільтрації та структурованої подачі товарів, а також реалізації функціоналу для оформлення замовлень, перегляду історії покупок, управління профілем та списком бажаного. Система має забезпечувати швидкий доступ до інформаційних блоків, інтуїтивну навігацію, адаптивність до різних пристроїв та легкість взаємодії з основними сервісами онлайн-магазину.

Вимоги до функціоналу

- **Верхня панель (Header):**
 - логотип застосунку;
 - кнопка відкриття бокового меню;
 - кнопка профілю користувача;
 - кнопка відкриття кошика;
 - поле пошуку товарів.
- **Нижня панель (Footer):**
 - розділи «Політика конфіденційності» (Privacy Policy), «Умови користування» (Terms and Conditions) та «Ліцензійна угода» (License agreement);

- розділи «Зв'язатися з нами» (Contact Us) та «Поширені запитання» (FAQ) для зворотного зв'язку з користувачами;
- посилання на соціальні мережі;
- логотип застосунку.
- **Навігація:**
 - верхня панель (Header) із кнопкою відкриття бокового меню, полем пошуку та доступом до профілю користувача;
 - бокове меню з переліком основних категорій товарів;
 - нижня панель (Footer) з доступом до інформаційних розділів і зворотного зв'язку;
 - механізм breadcrumb-навігації.
- **Головна сторінка:**
 - каруселі банерів;
 - каруселі категорій та товарів.
- **Каталог товарів:**
 - виведення списку товарів з пагінацією;
 - багаторівнева фільтрація та сортування за рейтингом, ціною та новизною;
 - можливість змінити кількість товарів, які відображаються.
- **Сторінка товару:**
 - детальний опис, характеристики та галерея зображень товару;
 - можливість додати товар до кошика з вибраною кількістю;
 - можливість додати товар до списку бажань;
 - відгуки користувачів із зірчковим рейтингом.
- **Кошик та оформлення замовлення:**
 - відображення товарів у кошику з вказаною кількістю, ціною та загальною сумою.
 - форма замовлення, що включає текстові поля вводу для імені, електронної адреси, поштової адреси та способу оплати.
 - валідація введених даних у формі перед оформленням замовлення;
 - модальне вікно підтвердження замовлення.
- **Особистий кабінет:**
 - перегляд історії замовлень та їх статусів;
 - перегляд збережених товарів у списку бажань та їхнє видалення за потреби;

- меню налаштувань: зміна імені, пароллю, електронної адреси користувача, можливість вийти з акаунта або повністю його видалити.
- модальні вікна для редагування персональних даних.
- **Авторизація та реєстрація:**
 - форми входу, реєстрації та відновлення пароллю з валідацією даних;
 - підтвердження електронної адреси та зміна пароллю шляхом введення коду.
- **Юридична інформація:**
 - сторінки «Політика конфіденційності» (Privacy Policy), «Умови користування» (Terms and Conditions) та «Ліцензійна угода» (License agreement).
- **Модальні вікна:**
 - підтвердження дій користувача (наприклад, видалення акаунта або товару зі списку бажаного).

Архітектура

ПЗ має бути спроектоване за принципами односторінкового застосунку (Single Page Application), побудованого на основі бібліотеки React із використанням мови TypeScript. Збірка проєкту здійснюється за допомогою Vite. Для управління станом застосовується Redux Toolkit, а для валідації форм і запитів – бібліотека Zod. Стилзація компонентів реалізована з використанням SCSS із підтримкою змінних та медіа-запитів.

Архітектура проєкту передбачає модульну структуру з поділом на:

- **Інтерфейсні компоненти:** кнопки, поля вводу, каруселі, модальні вікна, елементи фільтрації, сортування та інші елементи інтерфейсу, що використовуються на сторінках застосунку;
- **Сторінки:** окремі компоненти для головної сторінки, каталогу товарів, сторінки продукту, сторінки оформлення замовлення, профілю користувача та сторінки з кодом помилки 404;
- **Типи даних:** визначені за допомогою TypeScript для забезпечення строгої типізації даних у всіх модулях застосунку;
- **Схеми валідації:** реалізовані за допомогою бібліотеки Zod для перевірки даних форм;

- **Глобальний стан:** Redux-slices для управління станом акаунта користувача, категорій, товарів, кошика та інших функціональних блоків;
- **Утиліти:** набір допоміжних функцій, що спрощують логіку взаємодії з API та форматування значень;
- **Стилі:** оформлення інтерфейсу за допомогою SCSS із підтримкою змінних для гнучкої кастомізації зовнішнього вигляду;
- **Ресурси:** набір іконок, зображень та інших медіа-файлів, які використовуються для візуального оформлення інтерфейсу.

Застосунок має бути інтегрований з backend-сервером через REST API. Усі запити повинні бути асинхронними, реалізованими через createAsyncThunk з обробкою статусів (pending, fulfilled, rejected). Має бути реалізована обгортка над запитами, яка включає:

- автоматичне додавання JWT-токенів до заголовків;
- механізм повторної авторизації при втраті сесії.

Уся структура має бути побудована таким чином, щоб забезпечити масштабованість проєкту, легкість розширення функціоналу, повторне використання компонентів та модулів, а також зручність підтримки в довготривалій перспективі.

Технології

- **React:** створення UI на основі компонентного підходу.
- **TypeScript:** типізація даних, що підвищує стабільність коду.
- **Vite:** швидка збірка та розробка.
- **Redux Toolkit:** керування глобальним станом.
- **Zod:** декларативна валідація форм.
- **SCSS:** модульна стилізація з підтримкою змінних та медіа-запитів.

РОЗДІЛ 1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

У ході розробки клієнтської частини вебзастосунку електронної комерції було реалізовано функціональний інтерфейс користувача, що відповідає сучасним вимогам до зручності, швидкодії та масштабованості. Основою проєкту став односторінковий застосунок (Single Page

Application), побудований із використанням бібліотеки React та мови TypeScript, що забезпечило гнучкість структури та високу якість коду.

Інтерфейс системи охоплює повний цикл користувацької взаємодії – від перегляду товарів до оформлення замовлення. У межах реалізації було створено адаптивний дизайн, що коректно відображається на різних пристроях, включаючи мобільні телефони та планшети. Для покращення продуктивності та оптимізації часу розробки використовувався інструмент Vite, який забезпечує швидку збірку та гарячий перезапуск у режимі розробника.

Застосунок було реалізовано з використанням глобального управління станом через Redux Toolkit, що дозволило централізовано обробляти дані кошика, акаунта, відгуків, товарів, категорій, замовлень і списку бажаного. Для валідації форм та даних, отриманих з API, застосовувалась бібліотека Zod – гнучке рішення для перевірки типів і структури даних на клієнтському боці.

Система умовно поділяється на такі функціональні блоки:

- **Головна сторінка:** містить вітальні банери, блоки категорій та товарів;
- **Каталог товарів:** підтримує фільтрацію товарів за ключовими параметрами, сортування та перегортання сторінок;
- **Сторінка товару:** містить опис, характеристики, галерею зображень, рейтинги й відгуки про товар;
- **Кошик:** забезпечує підрахунок підсумкової вартості та надає можливість перейти до оформлення замовлення;
- **Оформлення замовлення:** містить форму, що має валідацію введених даних і підтвердження дії;
- **Особистий кабінет:** підтримує перегляд історії замовлень, зміну особистих даних та керування списком бажаного;
- **Юридичні сторінки:** містять політику конфіденційності, ліцензійну угоду та умови використання.

Архітектура проєкту побудована на модульному принципі, що передбачає поділ коду на компоненти, сторінки, схеми типів, утиліти, глобальний стан і стилі. Усі компоненти мають локальні стилі, написані з

використанням SCSS з підтримкою змінних і медіа-запитів, що забезпечує гнучкість дизайну.

Для зручності підтримки і масштабування система організована у вигляді логічних блоків: components, pages, state, types, utils, assets, styles. Такий підхід дозволяє ефективно структурувати код, забезпечує легкість внесення змін, повторне використання елементів та масштабованість проєкту.

Таким чином, у результаті виконаної роботи було створено повнофункціональний клієнтський застосунок, який охоплює ключові аспекти взаємодії користувача з системою електронної комерції, відповідає сучасним стандартам UI/UX, підтримує адаптивність і забезпечує ефективну інтеграцію з backend-сервісом через REST API.

РОЗДІЛ 2

РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

Вибір технологій і підхід до реалізації

Клієнтська частина застосунку була реалізована з використанням сучасних вебтехнологій, зокрема React, TypeScript, Redux Toolkit, Zod, SCSS та Vite. Основою архітектури обрано модель SPA (Single Page Application), що дозволяє досягти високої швидкості взаємодії користувача з інтерфейсом, без необхідності повного перезавантаження сторінки.

Під час розробки враховувалися принципи модульності, повторного використання компонентів, розділення відповідальностей, а також зручності масштабування.

Структура та логіка сторінок

Було реалізовано набір основних сторінок, кожна з яких відповідає певному функціональному сценарію:

- **Головна сторінка** містить інтерактивні слайдери з банерами, категоріями та товарами. Контент завантажується динамічно через API-запити до backend.
- **Каталог товарів** реалізований із підтримкою пошуку, сортування, пагінації та багаторівневої фільтрації. Всі фільтри мають реактивну поведінку: при зміні значення автоматично виконується запит до API.

```
export const getCategories = createAsyncThunk(
  "categories/getCategories",
  async (_, { rejectWithValue }) => {
    try {
      const response = await fetch(
        `${import.meta.env.VITE_API_URL}/categories`,
        {}
      );

      if (!response.ok) {
        const errorData = await response.json();
        return rejectWithValue(errorData);
      }

      const result = await response.json();
      return result.items;
    }
    catch (error: any) {
      return rejectWithValue(error.message);
    }
  }
);
```

```
export const getProductsByCategory = createAsyncThunk(
  "products/getProductsByCategory",
  async (
    params: { categoryId: string; filters?: { [key: string]: string[] } },
    { rejectWithValue }
  ) => {
    try {
      const urlParams = new URLSearchParams();
      urlParams.append("CategoryId", params.categoryId);

      const plainKeys = ["MinPrice", "MaxPrice", "Rating"];

      if (params.filters) {
        for (const [key, values] of Object.entries(params.filters)) {
          const paramName = plainKeys.includes(key)
            ? key
            : `Filters[${key}]`;
          values.forEach(val => urlParams.append(paramName, val));
        }
      }

      const response = await fetch(
        `${import.meta.env.VITE_API_URL}/products?${urlParams.toString()}`
      );

      if (!response.ok) {
        const errorData = await response.json();
        return rejectWithValue(errorData);
      }

      const result = await response.json();
      return result.items;
    }
    catch (error: any) {
      return rejectWithValue(error.message);
    }
  }
);
```

- **Сторінка товару** надає детальну інформацію про продукт, включаючи галерею зображень, технічні характеристики, зірочковий рейтинг, відгуки користувачів та можливість додати товар до кошика або списку бажаного.
- **Сторінка оформлення замовлення** реалізована як покрокова форма з валідацією кожного поля. Дані користувача, товари з кошика та

введена інформація передаються до backend через захищений POST-запит.

- **Особистий кабінет** дозволяє користувачу переглядати історію замовлень, керувати збереженими товарами, а також змінювати персональні дані (ім'я, електронна адреса, пароль тощо).

Стейт-менеджмент та взаємодія з API

Для керування глобальним станом застосунку було використано Redux Toolkit.

У рамках структури Redux реалізовано окремі slice-модулі для:

- профілю користувача (profile-slice);
- категорій (categories-slice);
- товарів (products-slice);
- кошика (cart-slice);
- відгуків (reviews-slice).

Запити до API виконуються через createAsyncThunk та обробляються з урахуванням помилок, статусів завантаження, а також збереження відповідей у сховищі Redux.

Захищені маршрути (ProtectedRoute)

Для обмеження доступу до певних сторінок застосунку залежно від автентифікації користувача було реалізовано компонент ProtectedRoute. Його основне призначення – забезпечення умовного рендерингу сторінок залежно від наявності авторизаційного токена або інших критеріїв доступу.

Цей компонент обгортає маршрути, які повинні бути доступними лише для зареєстрованих користувачів.

```

import React from "react";
import { Navigate, Outlet, useLocation } from "react-router-dom";
import { ADMIN_TOKEN } from "../utils/auth/authToken";

interface ProtectedRouteProps {
  children?: React.ReactNode;
  adminOnly?: boolean;
}

const ProtectedRoute: React.FC<ProtectedRouteProps> = ({ children, adminOnly = false }) => {
  const location = useLocation();

  const accessToken = localStorage.getItem("accessToken");

  const isAuthorized = adminOnly ? !!ADMIN_TOKEN : !!accessToken;

  if (!isAuthorized) {
    const redirectTo = adminOnly ? "/admin/login" : "/";
    return <Navigate to={redirectTo} state={{ from: location }} replace />;
  }

  return (
    <>
      {children ? children : <Outlet />}
    </>
  );
};

export default ProtectedRoute;

```

Було розроблено власну обгортку над fetch, яка автоматично додає JWT-токени до заголовків запитів та обробляє випадки 401/403 помилок, що дозволяє централізовано підтримувати механізм авторизації.

Валідація та обробка помилок

Для всіх форм і запитів було створено відповідні Zod-схеми. Це дозволило:

- забезпечити декларативну перевірку коректності введених даних (довжина пароля, формат email, обов'язковість полів);
- виводити локалізовані повідомлення про помилки безпосередньо в UI;
- уникнути надсилання некоректних запитів до backend.

Крім того, уся взаємодія з API супроводжується централізованою обробкою помилок (через custom hook), що дозволяє відображати повідомлення користувачу у разі збоїв мережі, помилок авторизації чи інших несправностей.

Реалізація користувацьких сценаріїв

Серед найбільш значущих реалізованих сценаріїв:

- **Додавання та видалення товарів у кошику:** реалізовано підтримку зміни кількості товарів, розрахунок підсумкової суми та збереження стану кошика у `localStorage`;
- **Оформлення замовлення:** створено форму з покроковим введенням даних, що підтримує автозаповнення для авторизованих користувачів;
- **Список бажаного:** реалізовано можливість зберігати товари навіть без входу в акаунт (через `localStorage`), з подальшою синхронізацією після авторизації;
- **Пошук товарів:** реалізовано `input` з `debounce`-логікою, що дозволяє не перевантажувати сервер зайвими запитами;
- **Система відгуків:** користувачі можуть залишати оцінку та коментар до товару, відредагувати або видалити свої відгуки.

Компонентний підхід та стилізація

Усі елементи інтерфейсу реалізовано у вигляді реюзабельних React-компонентів з типізацією `props` через `TypeScript`. Стилiзація виконана за допомогою `SCSS`-модулів, із підтримкою тем, адаптивної верстки та змінних.

Було реалізовано окремі компоненти для:

- кнопок, інпутів, чекбоксів, селектів;
- каруселей, карток товарів, банерів;
- модальних вікон, повідомлень про помилки/успіх, підтверджень дій;
- `skeleton`-завантажувачів для покращення UX при очікуванні відповіді з API.

Оптимізація продуктивності

Застосунок зібрано за допомогою `Vite`, що забезпечує швидкий старт розробки та оптимізацію фінальної збірки.

Було реалізовано наступні покращення:

- **Code splitting** – сторінки завантажуються динамічно (`lazy loading`);

- **Мемоізація компонентів** – за допомогою useCallback;

```
const getPercent = useCallback(  
  (value: number) =>  
    Math.round(  
      ((value - props.initialMinPrice) /  
        (props.initialMaxPrice - props.initialMinPrice)) *  
        100  
    ),  
  [props.initialMinPrice, props.initialMaxPrice]  
);
```

- **Кешування запитів** – збереження результатів у Redux та localStorage.

РОЗДІЛ 3

ОЦІНКА

Відповідність технічному завданню

У процесі розробки клієнтської частини системи було дотримано всіх вимог, визначених у технічному завданні. Основна мета – створити повноцінний інтерфейс електронної комерції, що охоплює повний цикл взаємодії користувача – від пошуку товарів до оформлення замовлення – була повністю реалізована.

Особливу увагу приділено стабільності роботи, зручності навігації, адаптивності для різних пристроїв, а також масштабованості архітектури на майбутнє.

Оцінка реалізованої архітектури

Обраний підхід SPA (Single Page Application) на основі React у поєднанні з TypeScript та Redux Toolkit продемонстрував ефективність при реалізації складної системи з численними сценаріями користувацької взаємодії. Модульна архітектура, розділення на slice-модулі для управління станом, а також типізація API-структур дозволили забезпечити контроль над логікою застосунку навіть на етапі активного розширення.

Застосування бібліотеки Zod для валідації форм у поєднанні з типами TypeScript покращило якість перевірки даних на стороні клієнта та зменшило кількість помилок при взаємодії з backend.

Також досягнуто розділення відповідальностей у структурі проєкту: компоненти, сторінки, утиліти, стилі та типи винесені в окремі модулі, що полегшило навігацію, тестування та подальше обслуговування системи.

Адаптивність та користувацький досвід

Завдяки використанню SCSS-модулів з підтримкою медіа-запитів було реалізовано адаптивний дизайн, що дозволяє комфортно користуватися сайтом на мобільних пристроях, планшетах та десктопах. Інтерфейс адаптується до ширини екрана, змінює структуру відображення фільтрів, карток товарів та модальних вікон без втрати функціональності.

Користувацький досвід (UX) був підсилений за рахунок таких рішень, як skeleton-завантажувачі, підтвердження дій через модальні вікна,

збереження кошика в localStorage, динамічна пагінація, інтерактивні сповіщення та зручна breadcrumb-навігація.

Продуктивність та оптимізація

Проект зібрано з використанням Vite, що забезпечує швидке інкрементальне збирання та гаряче оновлення під час розробки.

У продакшн-версії було реалізовано:

- мемоізацію компонентів з використанням useCallback;
- оптимізацію запитів через кешування даних у Redux та локальному сховищі;
- розділення коду (code splitting), що дозволяє зменшити початковий обсяг коду.

В результаті сторінки завантажуються швидко, навіть на повільних з'єднаннях, а кількість непотрібних запитів до сервера зведена до мінімуму.

Співпраця в команді та контроль якості

Усі етапи реалізації фронтенд-частини супроводжувалися активною командною взаємодією через систему контролю версій Git. Було організовано структуру репозиторію з поділом на гілки, pull request'и та код-рев'ю, що забезпечило якість інтеграції окремих модулів.

Також застосовувались статичні інструменти перевірки коду (ESLint, Prettier), що підтримували чистоту стилю коду на всіх етапах.

Підсумкова оцінка

Функціональність, передбачена технічним завданням, реалізована повністю. Інтерфейс забезпечує надійну роботу всіх ключових сценаріїв: реєстрація, пошук товарів, перегляд деталей, фільтрація, управління кошиком, оформлення замовлення, відгуки, кабінет користувача.

Обрані технології виправдали себе в аспектах продуктивності, розширюваності та підтримуваності. Створений клієнтський застосунок є сучасним, стабільним та придатним до подальшого масштабування – як з боку інтерфейсу, так і логіки бізнес-процесів.

ВИСНОВКИ

У ході виконання дипломної роботи було повністю реалізовано клієнтської частини вебзастосунку електронної комерції, яка забезпечує повноцінну взаємодію користувача з функціональністю онлайн-магазину. Система охоплює всі основні етапи користувацького шляху: від знайомства з асортиментом до оформлення замовлення, перегляду історії покупок та керування власним профілем.

Розробка здійснювалася відповідно до сучасних принципів побудови веб-застосунків типу (Single Page Application) із використанням бібліотеки React, мови TypeScript для типізації, Redux Toolkit для управління глобальним станом, Zod для валідації форм, а також SCSS для створення адаптивного інтерфейсу з підтримкою тем і медіа-запитів.

У рамках реалізації клієнтської частини були виконані наступні ключові завдання:

- Створено модульну архітектуру інтерфейсу з поділом на сторінки, спільні компоненти, схеми валідації, утиліти та глобальні типи.
- Реалізовано функціонал перегляду каталогу з багаторівневою фільтрацією, сортуванням, пошуком та пагінацією.
- Розроблено сторінки товарів із підтримкою зображень, характеристик, відгуків та дій користувача (додавання в кошик, бажане, оцінки).
- Побудовано систему реєстрації, авторизації, підтвердження email та відновлення паролю.
- Створено повний функціонал кошика та оформлення замовлення з валідацією даних та підтвердженням дій.
- Реалізовано особистий кабінет із можливістю перегляду історії замовлень, керування збереженими товарами та налаштуванням профілю.
- Здійснено підключення до backend API та побудовано механізми обробки запитів, помилок і станів завантаження.
- Забезпечено адаптивну верстку для коректного відображення інтерфейсу на різних пристроях.

Для забезпечення високої продуктивності було використано механізми оптимізації: ліниве завантаження компонентів, кешування

даних, мемоізація функцій та фрагментація коду. Розробка проводилася в інтеграції з системою контролю версій Git із застосуванням pull request'ів та код-рев'ю.

Реалізована клієнтська частина відповідає всім вимогам, визначеним у технічному завданні, і є надійною основою для подальшого масштабування платформи. Обрані технології та підходи підтвердили свою доцільність при створенні сучасного інтерфейсу для систем електронної комерції, орієнтованого на якісний користувацький досвід.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

- React Documentation – <https://react.dev/learn>
- TypeScript Documentation – <https://www.typescriptlang.org/docs/>
- Vite Documentation – <https://vite.dev/guide/>
- Redux Toolkit Documentation – <https://redux-toolkit.js.org/introduction/getting-started>
- Zod Documentation – <https://zod.dev/?ref=dev2devio&id=yup>
- SCSS Documentation – <https://sass-lang.com/documentation/>
- Репозиторій проекту – <https://github.com/Babchinsky/AmazonKillerFront>