Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інтернет магазину техніки з використанням React та TypeScript

Виконав(ла):студент(ка) 4 курсу, групи СП-43
спеціальності 121
«інженерія програмного забезпечення»
(шифр і назва спеціальності)

Свинарчук П.А.
(підпис) (прізвище та ініціали)

Керівник Михалик Д.М.
(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент Чайковський А.В.
(підпис) (прізвище та ініціали)

Тернопіль 2025

АНОТАЦІЯ

Кваліфікаційна робота бакалавра на тему «Розробка інтернет магазину техніки з використанням React та TypeScript» виконана Свинарчуком Павлом Андрійовичом, студентомТернопільського національного технічного університету імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-43.

Відомості про обсяг:

Робота присвячена створенню інтернет-магазину техніки з використанням бібліотеки React та мови програмування TypeScript.Інтернет-магазини у сучасному світі стали невід'ємною частиною бізнесу та щоденного життя людей. Вони значно спрощують процес купівлі товарів, дозволяючи кожному користувачеві здійснювати необхідні покупки, не виходячи з дому.

У вступі вказано мету роботи: розробка функціонального, зручного та адаптивного вебзастосунку, що дозволяє користувачам переглядати каталог техніки, додавати товари до кошика та оформлювати замовлення. Завданнями роботи є вивчення історії, основних принципів фронт-енд розробки, можливостей Reactта TypeScript, розробка та тестування вебсайту.

Перший розділ охоплює теоретичні основи фронт-енду, описує його історію, основні принциписемантична структура, такі як, компонентний підхід, адаптивність дизайну.

Другий розділ детально розглядає бібліотеку React, її основні концепції, такі як компонентний підхід, хуки, віртуальний DOM та односпрямований потік даних. Окрім того, аналізуються основні елементи екосистеми React. Також розглядаються переваги TypeScriptнад JavaScript, і його взаємодію з React.

Третій розділ містить практичну частину роботи: опис прототипу проєкту "Nice gadgets", який реалізує концепцію сучасного інтернет-магазину техніки. У цьому розділі представлено етапи створення вебзастосунку, починаючи з вибору редактора коду та структури проєкту, аж до побудови повноцінного інтерфейсу з

урахуванням принципів адаптивного дизайну. Детально розглянуто розробку архітектури застосунку, побудову основних компонентів, реалізацію маршрутизації, динамічного контенту, взаємодії з користувачем та обробки стану.

Окрему увагу приділено аналізу структури сторінок магазину, принципам їх компонування, а також використаним інструментам екосистеми React і TypeScript. В межах практичної реалізації продемонстровано приклади створення каталогу товарів, сторінок деталей продукту, функціоналу кошика, обраного та фільтрації.

Розділ завершується тестуванням вебсайту "Nice gadgets" на різних типах пристроїв (смартфонах, планшетах, ноутбуках, ПК) і в популярних браузерах. Метою тестування є перевірка коректної роботи інтерфейсу, адаптивності, сумісності та зручності користування. У результаті було підтверджено, що вебплатформа відповідає сучасним вимогам веброзробки і забезпечує якісний користувацький досвід.

ABSTRACT

The Bachelor's qualification thesis entitled "Development of an Electronics Online Store Using React and TypeScript" was completed byPavlo Andriyovych Svinarchuk, a student at Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Group SP-43.

Scope and content:

This work is dedicated to the development of an electronics online store using the React library and the TypeScript programming language. In today's world, online stores have become an integral part of business and everyday life. They greatly simplify the process of purchasing goods, allowing users to shop from the comfort of their homes.

The introduction defines the goal of the work: to develop a functional, user-friendly, and adaptive web application that allows users to browse a catalog of electronics, add products to a shopping cart, and place orders. The tasks include studying the history and key principles of front-end development, exploring the capabilities of React and TypeScript, and designing and testing the website.

The first chapter covers the theoretical foundations of front-end development, outlining its history and core principles, such as semantic structure, component-based architecture, and responsive design.

The second chapter provides a detailed overview of the React library, focusing on its core concepts like the component-based approach, hooks, the virtual DOM, and unidirectional data flow. It also examines the main elements of the React ecosystem. Additionally, the chapter discusses the advantages of TypeScript over JavaScript and its integration with React.

The third chapter presents the practical part of the thesis: a description of the "Nice gadgets" project prototype, which implements the concept of a modern electronics online store. This chapter outlines the stages of building the web

application—from choosing the code editor and organizing the project structure to creating a complete user interface based on the principles of responsive design. It covers the development of the application’s architecture, construction of core components, implementation of routing, dynamic content, user interaction, and state management.

Special attention is given to the analysis of the store’s page structure, principles of layout composition, and the tools used from the React and TypeScript ecosystem. The practical part includes examples of creating a product catalog, product detail pages, shopping cart functionality, favorites, and filtering.

The chapter concludes with testing the "Nice gadgets" website on various device types (smartphones, tablets, laptops, PCs) and in popular web browsers. The purpose of this testing is to verify correct interface behavior, responsiveness, compatibility, and user convenience. As a result, it was confirmed that the web platform meets modern web development standards and provides a high-quality user experience.

ЗМІСТ

ВСТУП

У наш час існує безліч сучасних технологій, які активно впроваджуються в усі сфери діяльності людини. Бізнес, звісно, не є винятком. Використання інформаційних технологій дозволяє значно покращити ефективність роботи підприємств, зробити їх сервіси зручнішими для клієнтів, а подекуди — просто необхідними для забезпечення конкурентоспроможності на ринку.

Із кінця XX століття розвиток інформаційних технологій докорінно змінив економічну картину світу. Постійне зростання конкуренції змушує підприємства впроваджувати нові технологічні рішення, вдосконалювати існуючі інформаційні системи та розширювати свою присутність у цифровому просторі. В сучасному середовищі відсутність онлайн-діяльності часто сприймається як недолік, що може вплинути на репутацію компанії та зменшити її шанси на успіх.

Особливої актуальності інтернет-комерція набула в умовах останніх глобальних подій — пандемії COVID-19 та воєнного стану в Україні. Ці обставини суттєво обмежили фізичну присутність людей у громадських місцях, зробили небезпечними традиційні способи здійснення покупок. Як наслідок, все більше споживачів надають перевагу онлайн-шопінгу, адже це зручно, швидко й безпечно.

Сучасні інтернет-магазини дають змогу користувачам легко знайти потрібний товар, перевірити його наявність, ознайомитися з характеристиками, порівняти ціни та здійснити покупку — не виходячи з дому. Це не тільки заощаджує час, а й значно підвищує рівень обслуговування.

У зв’язку з цим, розробка зручних, ефективних і візуально привабливих інтернет-магазинів стає невід’ємною частиною цифрової стратегії будь-якого бізнесу. Метою цієї дипломної роботи є створення сучасного веб-застосунку інтернет-магазину із використанням бібліотеки React та мови програмування TypeScript**.** Використання цих технологій дозволяє реалізувати продуктивний,

масштабований та легкий у підтримці веб-застосунок, що відповідає вимогам сучасного ринку.

Мета роботи – застосувати кращі практики фронтенд розробки для реалізації повноцінного вебсайту інтрнет-магазину техніки з використанням бібліотеки Reactта мови програмування TypeScript. Особливий акцент зроблено на забезпеченні оптимального відображення інтерфейсу на різних пристроях і розмірах екрана з урахуванням сучасних підходів у веброзробці.

Практичне значення:

Результатом роботи є створений інтернет-магазин техніки “Nicegadgets”, який реалізує ключові елементи адаптивного дизайну та служить прикладом практичної реалізації вебплатформи з урахуванням сучасних вимог до UI/UX. Проект може бути використаний як основа для комерційних або освітніх веб рішень, а також як приклад для початківців-розробників, які починають вивчення фронтенд розробки.

Структура дипломної роботи:

Дипломна робота складається з трьох розділів та висновків:

1.Теоретичні основи та інструменти фронтенд розробки – огляд історії розвитку, базових понять і технологій, порівняння сучасного та класичного фронтенду.

2.React та TypeScript та їх екосистема – вивчення основних концепцій React (компоненти, хуки, VirtualDOM), особливості інтеграції з TypeScript, огляд супутніх бібліотек.

3. Практична реалізація інтернет-магазину – поетапна розробка вебзастосунку: архітектура, компоненти, маршрутизація, обробка стану, адаптація до пристроїв, оптимізація контенту, тестування.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА ІНСТРУМЕНТИ ФРОНТЕНД-РОЗРОБКИ

1.1. Основи фронтенд розробки

Фронтенд-розробка – це галузь веброзробки, яка відповідає за створення інтерфейсу користувача, з яким безпосередньо взаємодіє відвідувач сайту або вебзастосунку. Вона базується на низці принципів та технологій, що забезпечують зручність, функціональність та доступність [6].

1.1.1. Семантика HTML

Семантична верстка – це процес написання HTML-коду із застосування тегів відповідно до їх смислового навантаження [6]. Вона дозволяє надавати тегам логічну структуру.

Основні приклади семантичних тегів:

– <header>, <footer> - верхній та нижній колонтитули сторінки.
– <nav> - блок із навігаційним меню.
– <article>, <section> - самостійні частини контенту.
– <main> - основний вміст сторінки.

За допомогою семантики ми можемо покращити SEOоптимізацію сторінки, кращу читабельність та підтримку коду, зручність для людей з різними вадами зору тощо.

1.1.2. Відокремлення структури, стилів і логіки

У сучасній веброзробці важливо дотримуватись чіткого поділу на:

– HTML – відповідає за структуру сайту.

– CSS – визначає зовнішній вигляд елементів (кольори, розміри, розташування).

– JavaScript – додає функціонал нашим елементам і логіку їх поведінки[6].

За допомогою такого розподілу відповідальностей ми получаємо зручність розробки, масштабованість, командну роботу.

1.1.3 Доступність та зручність для користувача

Доступність – це створення таких інтерфейсів, якими зможуть користуватись всі люди, включно з тими хто має порушення зору або інші фізичні обмеження [6].

Ключові принципи доступності:

– Використання правильних тегів**.** Наприклад, <label> для полів форм, <button> замість для кнопок.

– Текстові альтернативи**.** Зображення повинні мати атрибут alt для пояснення їх змісту.

– Контрастність**.** Текст повинен чітко читатися на фоні.

– Навігація з клавіатури**.** Всі елементи інтерфейсу повинні бути доступні без миші.

– ARIA-атрибути**.** Спеціальні атрибути, які допомагають допоміжним технологіям інтерпретувати динамічний контент.

1.2. Базові технології фронтенду

Фронтенд-розробка базується на використанні трьох основних технологій, які тісно взаємодіють між собою: HTML, CSS і JavaScript. Саме завдяки їм стає можливим створення вебінтерфейсів, зручних для користувачів і ефективних у роботі.

HTML (HyperText Markup Language) надає логічку структуру сторінці та використовується для створення заголовків, списків, форм тощо.

CSS (Cascading Style Sheets) відповідає за зовнішній вигляд HTML-елементів. За допомогою CSS можна змінювати кольори, шрифти, розміри, розміщення елементів, застосовувати анімації та адаптовувати інтерфейс.

JavaScript — це мова програмування, яка додає вебсторінці логіку, динаміку та інтерактивність, що дозволяє програмі реагувати на дії користувача [5].

Разом HTML, CSS і JavaScript утворюють технологічний базис фронтенд-розробки. Їх грамотне поєднання дозволяє створювати не просто вебсторінки, а повноцінні клієнтські додатки, здатні забезпечити швидкий, зручний і функціональний досвід для користувача.

1.3. Сучасні концепції фронтенду

Сучасна фронтенд-розробка приділяє значну увагу зручності користування,адаптивності та компонентному підході. Підхід до створення інтерфейсів получив багато змін, від статичних сторінок до інтерактивних додатків з багатою логікою. Це стало можливим завдяки появі сучасних фреймворків і бібліотек, таких як React, Vue, Angular [3]. Вони надають можливість створювати компонентну структуру додатку, в якій кожен елемент

інтерфейсу ізольований та можу бути повторно використаний. Також зростає популярність підходу MobileFirst, який передбачає проектування інтерфейсу спочатку для мобільних пристроїв з урахуванням обмеженого простору, а вже потім його адаптацію на більші екрани.

1.3.1. DOMі динамічність

DOM (Document Object Model) – об’єктна модель HTML-документа. Завдяки доступу до DOM, JavaScriptдозволяє змінювати структуру сторінки в реальному часу, обробляти події, здійснювати валідацію форм та динамічно фільтрувати контент [6].

1.3.2. Переваги TypeScript

TypeScript – це надбудова на JavaScript, яка дозволяє додавати типізацію. Вона допомагає запобігати помилкам ще до виконання коду, покращує авто доповнення та читаємість [4]. Особливо цінним є її використання в масштабних проектах та при роботі в команді.

```typescript
// Оголошення типу користувача (User)
type User = {
  id: number;          // Ідентифікатор користувача — число
  username: string;    // Ім'я користувача — рядок
  isAdmin: boolean;    // Чи є користувач адміністратором — булеве значення (true/false)
};

// Змінна `age` — зберігає вік користувача, тип number
let age: number = 25;

// Змінна `name` — зберігає ім'я користувача, тип string
let name: string = "Іван";

// Масив чисел `scores` — зберігає оцінки, тип масив чисел (number[])
let scores: number[] = [95, 87, 100];

// Функція `greet` — приймає ім’я користувача як аргумент і повертає вітання
function greet(userName: string): string {
  return `Привіт, ${userName}!`; // Повертає рядок з вітанням
}

```

Рисунок 1.1. демонструє приклад оголошення типів у TypeScript

1.4. Компонентний підхід у сучасній розробці (React)

React – це JavaScript-бібліотека для побудови UI, яка надає доступ до компонентного підходу [3]. Кожен елемент інтерфейсу є окремим компонентом, що містить свою логіку, стилі та розмітку. Це спрощує повторне використання, тестування та масштабування за стосунку.

1.4.1. Основні концепції React

Ключовими концепціями React є:

- Компонентність: інтерфейс складається з незалежних компонентів, які можна повторно використовувати.
- JSX (JavaScript XML)**:** синтаксис, який дозволяє писати HTML-подібний код у JavaScript. Це підвищує читабельність коду і спрощує структурування.
- Стан (state) та властивості (props)**:** state — це внутрішній стан компонента, тоді як props — це параметри, які передаються ззовні. React компоненти автоматично перерендерюються при зміні стану.
- Hooks (хуки)**:** функції, які дозволяють працювати з логікою життєвого циклу, станом та побічними ефектами в функціональних компонентах. Основні хуки: `useState`, `useEffect`, `useContext`, `useMemo`, `useCallback`.
- Односпрямований потік даних**:** дані в React передаються зверху вниз — від батьківських до дочірніх компонентів, що забезпечує передбачуваність.

Ці концепції дозволяють створювати масштабовані, модульні додатки з мінімальним дублікатом коду [3].

1.4.2. Порівняння React з конкурентами

На ринку існує кілька популярних інструментів для створення інтерфейсів: React, Angulari Vue.js. Angular є повноцінним фреймворком з чіткою структурою і великою кількістю вбудованих інструментів, але має високу складність для новачків. Vue.js – гнучкий і простий у вивченні фреймворк, який добре підходить для малих проектів. React – це щось середнє між цими двома фреймворками – він не нав’язує свою архітектуру, легкий і зрозумілий у вивченні, що дає можливість розробнику свободу вибору технологій, тому я обрав саме React [3].

1.5. Екосистема React

React — це лише "V" у моделі MVC, тому для побудови повноцінного застосунку необхідно використовувати додаткові бібліотеки:

- React Router — стандартне рішення для реалізації маршрутизації [9]. Дозволяє налаштовувати переходи між сторінками без перезавантаження сторінки.
- Redux / Zustand / Recoil — бібліотеки для централізованого керування станом. Redux базується на принципах Flux і має складну, але масштабовану структуру. Zustand та Recoil пропонують легші API.
- React Query / SWR — інструменти для роботи з асинхронними даними. Дозволяють кешувати, оновлювати, синхронізувати дані з API.
- Axios / Fetch API — популярні засоби для надсилання HTTP-запитів.

Завдяки такій гнучкій екосистемі, розробники можуть обирати ті інструменти, які найкраще підходять для їхнього проекту [3].

1.6. Переваги використання TypeScript з React.

У контексті розробки на React TypeScript дозволяє типізувати компоненти, пропси, стани, обробники подій тощо. Це дає змогу не лише уникати помилок, а й полегшує командну розробку, оскільки структура даних стає задокументованою безпосередньо в коді.

React-компоненти, написані на TypeScript, стають більш надійними та зрозумілими. Нижче наведено приклад типізованого функціонального компонента, який приймає пропси:

```tsx
// Оголошення інтерфейсу для пропсів компонента Greeting
interface GreetingProps {
  name: string;   // Ім'я є обов'язковим рядком
  age?: number;   // Вік є необов'язковим числом (може бути відсутній)
}

// Функціональний компонент Greeting з типом пропсів GreetingProps
const Greeting: React.FC<GreetingProps> = ({ name, age }) => {
  return (
    <div>
      {/* Виводимо привітання з ім'ям користувача */}
      <h2>Привіт, {name}!</h2>

      {/* Якщо вік передано, то виводимо повідомлення про вік */}
      {age && <p>Вам {age} років.</p>}
    </div>
  );
};
```

Рисунок 1.2. Демонстрація прикладу компонента React із типізованими прописами

У цьому прикладі видно, що тип GreetingProps чітко описує очікувані властивості компонента. TypeScript гарантує, що компонент отримає лише правильні типи, а редактор коду зможе надати розробнику підказки під час написання.

1.7. Інструменти для підтримки якості коду

Для підтримки якості коду у фронтенді широко використовують інструменти ESLint та Prettier. ESLintпроводить статичний аналіз і допомогає виявити помилки ще до запуску коду, а також забезпечує дотримання стандартів стиллю. Prettierавтоматично форматує код відповідно до заданих правил, що підвищує читабельність і усуває суперечки в команді щодо форматування. Обидва інструменти інтегруються з редакторами коду, автоматизуючи контроль якості [4].

1.8. Адаптивність і стратегія MobileFirst

Підхід MobileFirstозначає, що інтерфейс проектується спочатку для мобільних пристроїв, з урахуванням обмеженого простору екрана, а потім масштабуються стилі для більших розмірів екрана. Така стратегія дозволяє зосередитись на основному контенті, забезпечити швидке завантаження сторінки і комфортну взаємодію з інтерфейсом на будь-якому пристрої. Адаптивний дизайн реалізується за допомогою CSS-медіа-запитів та гнучких макетів [8].

РОЗДІЛ 2. АНАЛІЗ ВИМОГ ДО СИСТЕМИ

2.1. Загальні положення

Перед початком розробки інтернет-магазину техніки було необхідно визначити ключові вимоги до системи, що включають як функціональні, так і не функціональні аспекти. Такий підхід дозволяє чітко окреслити задачі майбутнього продукту, уникнути неузгодженостей у процесі реалізації та забезпечити відповідність очікування користувачів і адміністратора. Система повинна забезпечувати доступ до каталогу товарів, перегляд детальної інформації, управління кошиком і оформлення замовлень. З боку адміністратора передбачено керування вмістом каталогу товарів.

Важливо було врахувати потребу в адаптивному дизайні, який гарантує коректне відображення сторінок на пристроях різної ширини. Продуктивність і швидкодія інтерфейсу також визначають якість взаємодії з користувачем, тому пріоритетом стала реалізація компонентної архітектури з використанням React.

2.2. Технологічні засади

Однією з причин вибору Reactстала його можливість оптимізовувати рендеринг за допомогою VirtualDOM [3]. Це дозволяє створювати динамічний інтерфейс, який миттєво реагує на дії користувача без необхідності перевантаження сторінки. Для зменшення кількості помилок і покращення коду було застосовано TypeScript, що забезпечує статичну типізацію та дозволяє заздалегідь виявляти потенційні проблеми під час компіляції [4].

Також під час реалізації було використано ReactRouter для налаштування маршрутизації, що дозволяє організовувати переходи між сторінками без повного оновлення інтерфейсу [9]. Такий підхід дає змогу створити єдину

SPA(SinglePageApplication), яка виглядає як повноцінний вебсайт, але працює значно швидше.

2.1. Користувацький функціонал

З боку користувача інтерфейс повинен забезпечувати зручну навігацію по сайту. Особливу увагу приділено функціоналу перегляду товарів із можливістю сортування та фільтрації. Кожен товар представлено у вигляді окремої картки з короткою інформацією, а натискання відкриває сторінку з розгорнутим описом.

Функція додавання товару до кошика реалізована так, щоб одразу надавати зворотній зв'язок користувачу, не змінюючи поточної сторінки. У процесі оформлення замовлення користувач вводить свої контактні дані, після чого підтверджує замовлення.

Рисунок 2.1. Діаграма usecase для користувача

2.4. Функціонал адміністратора

Окрему категорію вимог становить адміністративна частина системи. Вона повинна надати можливість керувати вмістом сайту – додавати нові товари, редагувати характеристики існуючих, змінювати зображення, а також видаляти позиції, які більше не продаються. Адмін – панель повинна бути простою у використанні, із перевіркою правильності введених данних, щоб уникнути помилок у виведенні інформації для покупців.

Рисунок 2.2. Діаграма usecase для адмін.

2.5. Архітектурне рішення

Програмна архітектура проекту базується на модульному підході: кожен компонент відповідає за окрему частину функціоналу. Це забезпечує легкість тестування, повторного використання коду для подальшого масштабування. Дані зберігаються в локальному localStorage, що дозволяє зберігати стан кошика навіть після оновлення сторінки. У разі розширення функціоналу система можу бути інтегрована з backend – сервером або базою даних без значних змін у клієнтському коді.

2.6. Загальні положення

Хоча система наразі працює без авторизації, вже на етапі аналізу було враховано можливість впровадження механізмів аутентифікації, шифрування персональних даних і захисту від XSS- або CSRF-атак. У подальшому можлива

реалізація особистого кабінету користувача з історією замовлень, статусом доставки та системою повідомлень.

У підсумку, проведений аналіз вимог дав змогу сформувати цілісне бачення системи, її архітектури, технічного стека та сценаріїв взаємодії, що заклали фундамент для ефективної реалізації функціоналу інтернет-магазину.

РОЗДІЛ 3. РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ НА REACTTA TYPESCRIPT

3.1. Опис прототипу інтернет-магазину

У цій роботі було реалізовано прототип інтерне-магазину з продажу мобільної техніки та аксесуарів. Основними функціями, які були реалізовані, є можливість перегляду каталогу товарів, відкриття детальної сторінки конкретного продукту, додавання товару до кошика або обраного, використання фільтрації та сортування. Весь інтерфейс було реалізовано за допомогою React, з використанням мови TypeScript. Для стилізації було обрано SCSS-модулі, що дозволяє ізолювати стилі кожного компонента і забезпечити зручне масштабування проекту.

3.2. Проєктування вебплатформи “Nicegadgets”

Проєктування вебплатформи “Nice gadgets” передбачає створення інтернет-магазину, що дозволяє користувачам переглядати, обирати та замовляти техніку, зокрема мобільні телефони та аксесуари. Основними вимогами до проєкту були: зручний користувацький інтерфейс, швидка навігація між сторінками, можливість додавання товарів у кошик і до списку обраного, а також зберігання стану користувача у браузері без необхідності авторизації.

На етапі проєктування було обрано технології, що найкраще відповідають вимогам. Для побудови інтерфейсу використовується React – бібліотека, яка дозволяє компонувати UI з окремих логічних блоків [3]. Для типізації використовувався TypeScript, що забезпечує безпечну розробку та зменшує кількість помилок у коді [4]. Маршрутизація реалізована за допомогою

бібліотеки react-router-dom, яка забезпечує швидкий перехід між сторінками без перезавантаження [9].

Дизайн застосунку проєктувався відповідно до принципів адаптивного вебдизайну. Усі компоненти інтерфейсу розроблялися з урахуванням медіазапитів, що забезпечує їхнє правильне відображення як на комп'ютерах, так і на мобільних пристроях. Основу стилів становлять SCSS-модулі, що дозволяють підтримувати ізольованість стилів, використання змінних кольорів, міксинів та вкладеності селекторів [6].

Візуальна структура платформи включає головну сторінку з банером та категоріями товарів, сторінку каталогу, де реалізовано пагінацію, фільтрацію та сортування товарів, а також сторінку з деталями кожного товару. Окрім цього, користувач має змогу переглянути обрані товари на окремій сторінці та керувати своїм кошиком. Навігація підтримується зручними маршрутами з SEO-дружніми адресами.

Для моделювання взаємодії між модулями було створено діаграми структури, що демонструють архітектурну побудову застосунку, включно з основними сторінками, компонентами, утилітами та допоміжними модулями. Усі компоненти типізовано, що підвищує стабільність роботи інтерфейсу. Наприклад, компоненти ProductCard, Banner чи Pagination отримують строго визначені пропси відповідно до заданих інтерфейсів.

Таким чином, етап проєктування дозволив сформувати чітку архітектурну основу для майбутньої реалізації інтерфейсу, яка враховує як функціональні, так і нефункціональні вимоги до сучасного онлайн-магазину. Усі елементи платформи розроблялися з акцентом на повторне використання компонентів, підтримку масштабування та зручність користувача.

3.2.1. Модель предметної області інтернет-магазину

Предметна область включає категорії товарів, окремі позиції (мобільні телефони), атрибути (бренд, ціна, колір, пам’ять), та взаємодії з користувачем: додавання в корзину, перегляд деталей, вибір кількості.

```
export interface Product {
  id: number;
  category: string;
  itemId: string;
  name: string;
  fullPrice: number;
  price: number;
  screen: string;
  capacity: string;
  color: string;
  ram: string;
  year: number;
  image: string;
}
```

Рисунок 3.1. Приклад інтерфейсу товару

3.2.2. Архітектура фронтенд-застосунку

Архітектура проекту побудована на основі компонентного підходу, що забезпечує зручне розділення логіка та можливість повторного використання коду. Уся структура організована модульно. Компоненти, що використовуються повторно (наприклад, Header, Footer, Banner, ProductCard), розміщені в окремій папці. Окремо виділено хуки, утиліти, константи та стилі.

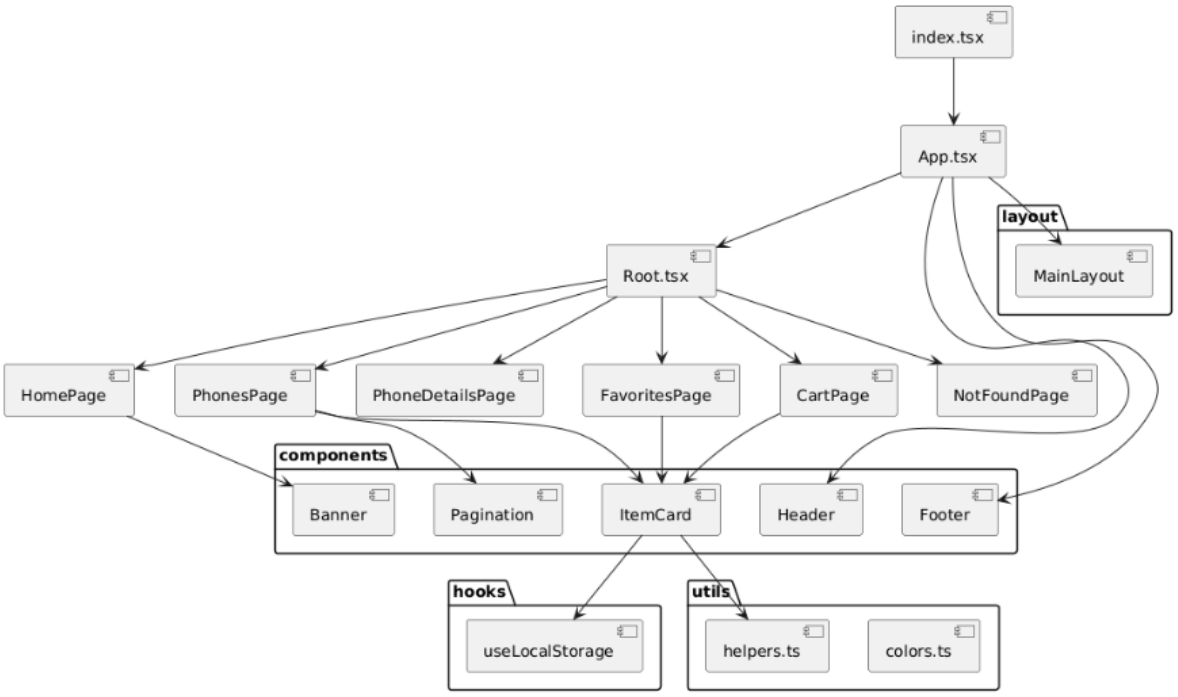

Рисунок 3.2. Діаграма компонентів

Навігація реалізована за допомогою бібліотеки ReactRouter шостої версії. Вона дає зможу будувати гнучку маршрутизацію з вкладеними маршрутами та динамічними параметрами. Основна конфігурація маршрутизатора здійснюється в окремому файлі Root.tsx. У ньому визначено маршрути до головної сторінки, сторінки з товарами, детальної інформації про товар, обране, кошик та сторінку 404.

```
export const Root = () => (
  <Router>
    <Routes>
      <Route path="/" element={<App />}>
        <Route index element={<HomePage />} />
        <Route path="home" element={<Navigate to="/" replace />} />
        <Route path="phones" element={<PhonesPage />} />
        <Route
          path="phones/:id"
          element={<ItemCard type={ProductType.PHONE} />}
        />

        <Route path="tablets" element={<TabletsPage />} />
        <Route
          path="tablets/:id"
          element={<ItemCard type={ProductType.TABLET} />}
        />

        <Route path="accessories" element={<AccessoriesPage />} />
        <Route
          path="accessories/:id"
          element={<ItemCard type={ProductType.ACCESSORIES} />}
        />
        <Route path="favourites" element={<FavouritesPage />} />
        <Route path="cart" element={<CartPage />} />

        <Route path="*" element={<NotFound />} />
      </Route>
    </Routes>
  </Router>
);
```

Рисунок 3.3ю Створення маршрутизації

3.4. Ініціалізація проекту

На етапі ініціалізації проекту було створено нову структуру застосунку за допомогою інструменту Vite [7]. Цей інструмент довзоляє швидко розгорнути TypeScript-проект із підтримкою React. Створення базового шаблону відбувається за допомогою команди `npmcreatevite@latest`, після чого вибирається шаблон з підтримкою TypeScript + React. Після створення структури проекту він включає в себе основні конфігураційні файли, директорії для зберігання компонентів, сторінок, стилів, зображень та утиліт. Було встановлено всі необхідні залежності, включно з ReactRouter, SASS, TypeScript, Prettier, ESLint, classnames, та бібліотеками для покращення візуального інтерфейсу.

Після цього було налаштовано базові правила для форматування коду та перевірки на наявність помилок у файлах .eslintrc.cjs та .prettierrc [4].

Проект було структуровано за принципами чистої архітектури. У папці src було створено окремі каталоги для сторінок, спільних компонентів, хелперів, хуків, API, стилів, типів та констант. Такий підхід дозволив розділити логіку програми та забезпечити масштабованість.

3.5. Розробка інтерфейсу: компоненти, маршрутизація, адаптивність

Після створення структури було реалізовано базові компоненти інтерфейсу.

Компонент Headerмістить логотип, навігаційне меню та іконки для переходу в кошик або обране, під час створення компонентів були використані потрібні семантичні теги для кращої читабельності та структури коду, а також SEOоптимізації.

```jsx
export const Header = () => {
  const [isMenuOpen, setIsMenuOpen] = useState(false);

  const handleMenuOpen = () => {
    setIsMenuOpen(true);
  };

  const handleMenuClose = () => {
    setIsMenuOpen(false);
  };

  return (
    <header className="header">
      <div className="header_container">
        <nav className="header__nav">
          <Logo />
          <Menu />
        </nav>
        <div className="header__shopping">
          <Favourites />
          <ShoppingBag />
        </div>

        <BurgerMenu
          handleOpen={handleMenuOpen}
          handleClose={handleMenuClose}
          isOpen={isMenuOpen}
        />
      </div>
      {isMenuOpen && <MenuMobile onClose={handleMenuClose} />}
    </header>
  );
};
```

Рисунок 3.4. Лістинг компонента Header.

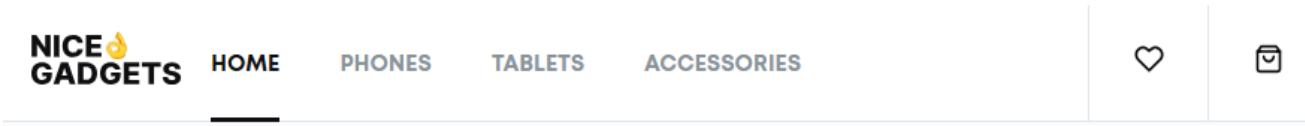

Рисунок 3.5. Зовнішній вигляд компонента Header

Компонент Footer який включає контактну інформацію та посилання на сторінки.

```jsx
export const Footer = () => {
  return (
    <div className="footer">
      <div className="footer__container">
        <Logo />
        <div className="footer__menu">
          <ul className="footer__menu-list">
            {pagesOfMenu.map(page => (
              <li className="footer__menu-item" key={page.name}>
                <a href={`#${page.name}`} className="footer__menu-link">
                  {page.name}
                </a>
              </li>
            ))}
          </ul>
        </div>
        <div className="footer__button">
          <p className="footer__button-text">Back to top</p>
          <button
            className="footer__button-back"
            onClick={() => window.scrollTo({ top: 0, behavior: 'smooth' })}
          >
            <img
              src="./img/arrow-to-top.svg"
              alt="Back to top"
              className="footer__button-back-img"
            />
          </button>
        </div>
      </div>
    </div>
  );
};
```

Рисунок 3.6. Лістинг компонента Footer.

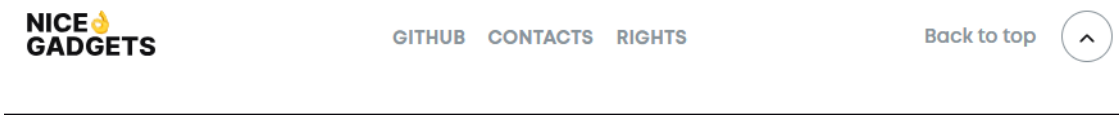

Рисунок 3.7. Зовнішній вигляд компонента Footer.

Головна сторінка HomePage відображає банер зі слайдером, популярні товари та новинки. Для реалізації слайдера було використано бібліотеку swiper, яка дозволяє реалізувати адаптивну карусель з підтримкою жестів, стрілок і пагінації. Компоненти ProductCard, Banner, Breadcrumbs, Pagination, BackButton та інші побудовані як незалежні та багаторазові блоки, що працюють з пропсами, типізовамини за допомогою TypeScript. Наприклад, ProductCard приймає дані

товару як пропси, відображає зображення, назву, ціну, колір, обсяг пам’яті та кнопки взаємодії.

```tsx
export const Root = () => (
  <Router>
    <Routes>
      <Route path="/" element={<App />}>
        <Route index element={<HomePage />} />
        <Route path="home" element={<Navigate to="/" replace />} />
        <Route path="phones" element={<PhonesPage />} />
        <Route
          path="phones/:id"
          element={<ItemCard type={ProductType.PHONE} />}
        />

        <Route path="tablets" element={<TabletsPage />} />
        <Route
          path="tablets/:id"
          element={<ItemCard type={ProductType.TABLET} />}
        />

        <Route path="accessories" element={<AccessoriesPage />} />
        <Route
          path="accessories/:id"
          element={<ItemCard type={ProductType.ACCESSORIES} />}
        />
        <Route path="favourites" element={<FavouritesPage />} />
        <Route path="cart" element={<CartPage />} />

        <Route path="*" element={<NotFound />} />
      </Route>
    </Routes>
  </Router>
);

```

Рисунок 3.8. Лістинг компонента Root

Маршрутизація була реалізована за допомогою react-router-dom. Було налаштовано базовий маршрут у файлі main.tsx, який підключає головний компонент Root. У ньому передбачено вкладені маршрути до всіх сторінок проекту. Також реалізовано обробку динамічного маршруту :phoneId, який відкриває сторіку з деталями товару.

Адаптивний дизайн реалізовано за допомогою SCSSта CSS-модулів. Кожен компонент має окремий файл стилів, і стилі написано з урахуванням

медіа запитів для різних розмірів екранів. Це забезпечує коректне відображення сайту як на десктопних, так і на мобільних пристроях. Наприклад, сторінка PhonePage використовує грід-сітку, яка адаптується до ширини екрана. Для кнопок, зображень і блоків було застосовано змінні кольорів, які знаходять у файлі utils.scss, це добавляє зручність і можливість швидкої змінни властивостей елемента на сторінці.

```scss
// colors
$primary: #0f0f11;
$secondary: #89939a;
$icons_and_placeholders: #b4bdc3;
$elements: #e2e6e9;
$white: #fff;
$gray-button: #222;
$accent: #f86800;
$background: #fafbfc;

// media
$tablet: 1199px;
$mobile: 640px;

// fonts
$font-family: 'Mont';
$font-weight-regular: 600;
$font-weight-medium: 700;
$font-weight-bolt: 800;

// fixed sizes
$header-height: 64px;
$footer-height: 96px;
```

Рисунок 3.9. Лістинг утиліти _vars

3.6. Тестування UI прототипу платформи “Nicegadgets”

Метою тестування прототипу платформи “Nicegadgets” є забезпечення її коректного функціонування та відображення на різних типах пристроїв. Одним з головних завдань є перевірка адаптивності дизайну, що гарантує зміну структури інтерфейсу відповідно до ширини екрану, орієнтації пристрою та роздільною здатності, зберігаючи при цьому зручність і доступність користування.

Також тестування охоплює функціональні аспекти вебплатформи – перевіряється правильна робота інтерактивних елементів, таких як кнопки, меню, слайдери, списки товарів, фільтрації, додавання до кошика чи обраного.

Оцінюється зручність переходу між сторінками, стабільність маршрутизації та коректне реагування на дії користувача.

Особливу увагу приділено перевірці цілісності візуального інтерфейсу – тобто правильному відображенню компонентів UI: зображень, заголовків, описів товарів, цін, кнопок і піктограм. Незалежно від розміру екрана контент має залишатись читабельним, логічно розташованим і при цьому бути привабливим з точки зору дизайну.

Для перевірки сумісності застосунку проводились тести у різних веббраузерах: GoogleChrome, MozillaFirefox, Safari, MicrosoftEdge. Також врахувались платформи – Windows, macOS, Android та iOS. Це дозволило переконатись, що всі елементи вебінтерфейсу працюють стабільно в різному середовищі, без збоїв чи спотворень.

Тестування проводилось на таких ключових сторінках, як головна, сторінка каталогу, сторінка товару, кошик і сторінка обране. Під час тестування використовувались інструменти ChromeDevToolsдля емалювання пристроїв з різною шириною екрана, а також перевірка вручну на фізичних пристроях – ноутбуках, планшетах і смартфонах.

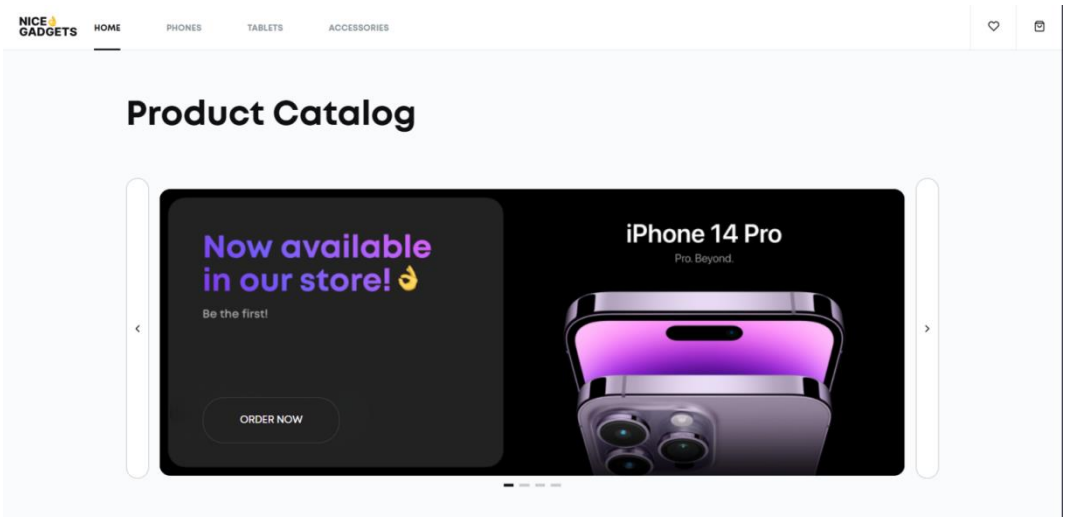

Рисунок 3.10. Головна сторінка платформи “Nicegadgets” у десктопному вигляді

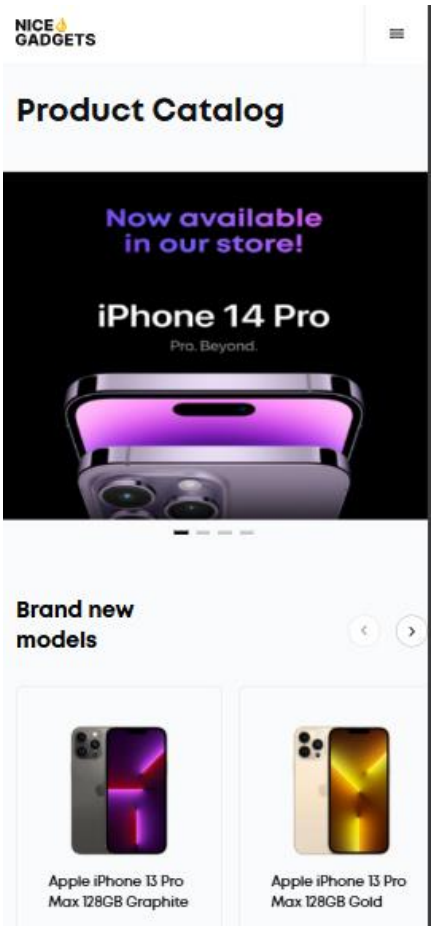

Рисунок 3.11. Головна сторінка платформи “Nicegadgets” у мобільному вигляді

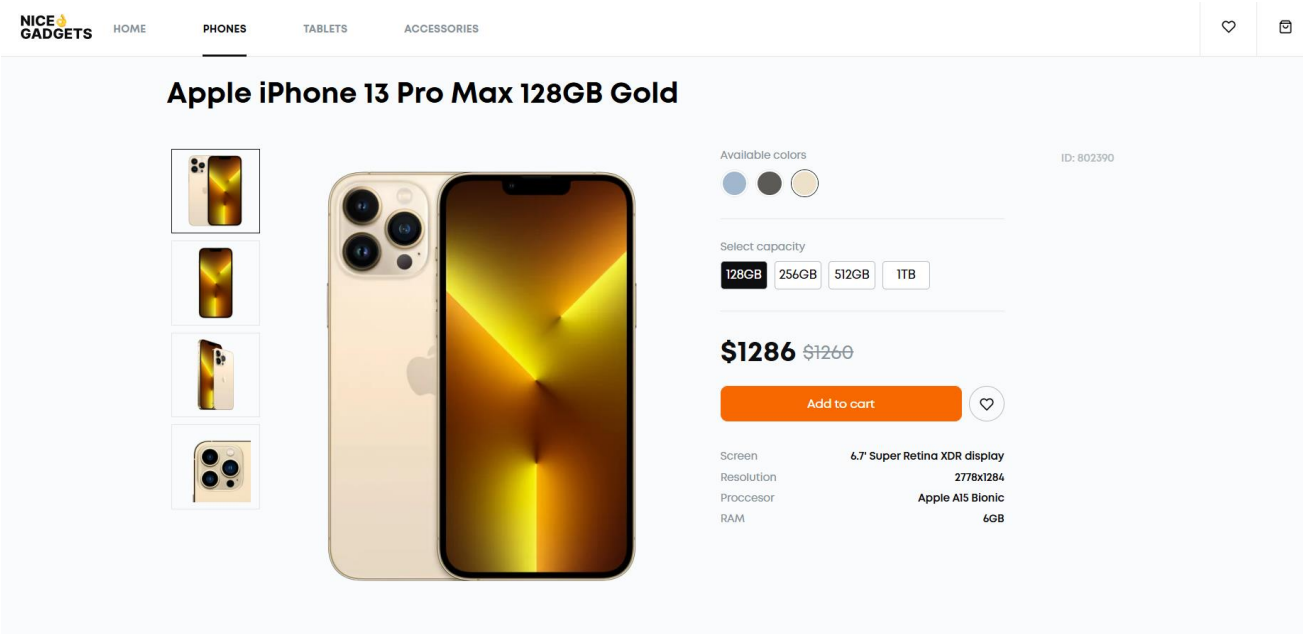

Рисунок 3.12. Сторінка товару в десктопному вигляді

Рисунок 3.13. Сторінка товару в мобільному вигляді

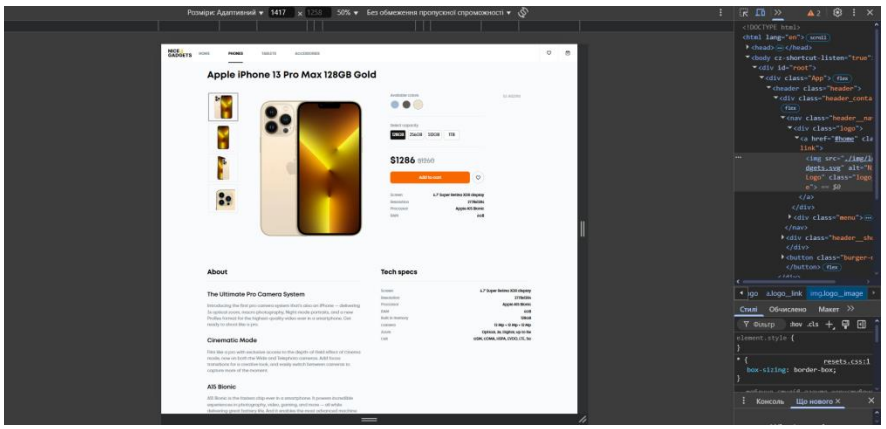

Рисунок 3.14. Перевірка адаптивності через ChromeDevTools

В результаті тестування було встановлено, що:

- дизайн коректно адаптується під різні екрани;
- компоненти залишаються функціональними і не накладаються;
- анімації, перемикання товарів та фільтрація працюють без затримок;

– усі іконки, зображення та стилі відображаються відповідно до дизайну.

Окремо було виявлено, що в браузері Firefox виникали незначні проблеми з відображенням SVG-іконок, що потребувало додаткового виправлення шляхів до ресурсів. Після оптимізації цієї частини всі компоненти інтерфейсу почали коректно відображатися і в цьому середовищі.

Таким чином, результати тестування підтверджують, що вебплатформа “Nice gadgets” відповідає сучасним вимогам щодо адаптивного дизайну, сумісності з різними браузерами та операційними системами, забезпечуючи якісний користувацький досвід.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

4.1. Комплексний аналіз життєдіяльності людини

У процесі професійної діяльності особливого значення набуває питання безпеки життєдіяльності людини. Забезпечення належних умов стабільної та ефективної життєдіяльності є ключовим чинником збереження здоров'я підвищення працездатності та мінімізації ризику виникнення надзвичайних ситуацій. У межах будь-якого виробничого або навчального процесу можуть виникати зовнішні та внутрішні загрози, які впливають на фізичний, психоемоційний, соціальний та екологічний стани людини. Через це виникає потреба у проведенні комплексного аналізу життєдіяльності людини.

Комплексний аналіз життєдіяльності людини – це систематизоване вивчення чинників умов і процесів які впливають на її стан у різних сферах: на роботі, у побуті, у навчанні та у взаємодії з навколишнім середовищем. Такий аналіз дозволяє виявити небезпеки, й розробити заходи щодо їх усунення або зменшення впливу на організм людини [11].

До основних груп чинників що враховуються під час проведення аналізу належить: психологічні, соціально-економічні, екологічні, техногенно-виробничі, фізіологічні та психофізіологічні чинники [10].

До психологічних чинників належать чинники які визначають психоемоційний стан людини: рівень стресу, міжособистісні відносини, психологічна сумісність, монотонність або надмірна складність завдань. Негативний вплив цих чинників може привезти до психічного перенавантаження, професійного вигорання, зниження концентрації, збільшення ризику помилок у роботі та зниження працездатності.

До соціально-економічних чинників належать умови соціального забезпечення людини: рівень заробітної плати, стабільність зайнятості, наявність соціальних гарантій, можливість професійного зростання, доступ до медичних і побутових послуг. Незадовільний рівень соціально-економічних умов може

спричинити зростання соціальної напруги, зниження життєвої мотивації, погіршення психологічного стану та загального рівня життєдіяльності.

До екологічних чинників належать умови навколишнього середовища, в яких перебуває людина: рівень забруднення повітря, води, ґрунту, наявність промислових викидів, шумових або радіаційних джерел. Негативний вплив таких чинників може викликати хронічні захворювання, алергічні реакції, зниження імунітету та загального фізичного стану.

До техногенно-виробничих чинників належать загрози, пов’язані з використанням технічних засобів, машин, механізмів, хімічних речовин, електроустановок, а також ризики аварій, пожеж, вибухів або поломок обладнання. За умови недотримання техніки безпеки ці чинники можуть становити серйозну загрозу життю та здоров’ю людини, призводити до травм, професійних захворювань або летальних випадків.

До фізіологічних та психофізіологічних чинників належать умови, які безпосередньо впливають на фізичне самопочуття людини: мікроклімат, освітлення, рівень шуму, вібрації, наявність електромагнітного випромінювання, а також режим праці та відпочинку. Несприятливі фізіологічні умови можуть спричинити перевтому, зниження продуктивності, погіршення загального стану здоров’я та розвиток професійних захворювань.

Комплексний аналіз життєдіяльності людини дозволяє виявити основні чинники, що впливають на стан здоров’я, безпеку та працездатність у процесі професійної діяльності. Урахування психологічних, соціально-економічних, екологічних, техногенних та фізіологічних аспектів дає змогу своєчасно ідентифікувати потенційні ризики та впроваджувати ефективні заходи для створення безпечного та комфортного середовища [11].

4.2Використання та застосування питання інженерної психології

Інженерна психологія відіграє ключову роль у системі охорони праці, оскільки дозволяє врахувати психологічні та когнітивні особливості людини при розробці технічних систем, організації робочих місць і процесів. Застосування інженерної психології сприяє зменшенню впливу людського фактора на безпеку, підвищенню продуктивності та зниженню кількості помилок.

Відповідно до ДСТУ 22787-2016 "Системи управління охороною праці. Загальні вимоги", система управління охороною праці повинна передбачати врахування факторів, що впливають на поведінку працівників, зокрема їх психофізіологічний стан, що є основою для формування заходів профілактики виробничого травматизму [13].

Закон України «Про охорону праці» закріплює обов’язок роботодавця забезпечувати умови, які гарантують безпеку і здоров’я працівників з урахуванням як фізичних, так і психологічних аспектів [14].

Основні напрямки застосування інженерної психології:

– Раціональне проєктування робочих місць з урахуванням психологічних особливостей працівника, що забезпечує зручність та інтуїтивність керування;

– Уникнення інформаційного перевантаження та монотонності, які можуть викликати психоемоційне виснаження та помилки в роботі;

– Оптимізація систем сигналізації, зручних для сприйняття у стресових або аварійних ситуаціях;

– Психологічна підготовка та навчання персоналу, з урахуванням особливостей сприйняття та поведінки в умовах підвищеного стресу.

ВИСНОВКИ

У ході виконаної бакалаврської кваліфікаційної роботи на тему “Розробка інтернет магазину техніки з використанням React та TypeScript” було досягнуто поставлену мету – створено адаптивний прототип вебплатформи “Nicegadgets”, що забезпечує зручний та швидкий доступ до каталогу товарів, функції додавання до кошика та обраного, а також перегляд детальної інформації про кожну позицію.

На основі теоретичного аналізу та практичних навичок здобутих за період навчання було розглянуто та закріплено сучасні принципи фронтенд – розробки, переваги компонентного підходу, а також особливості роботи з такими технологіями як React, TypeScript, SCSSта ReactRouter. Реалізація інтерфейсу на базі бібліотеки Reactдала змогу організувати структуру додатку у вигляді окремих, багаторазово використовуваних компонентів. Використання TypeScript підвищило надійність і передбачуваність коду завдяки статичній типізації.

Особлива увага приділялась адаптивному дизайну, що забезпечує коректне відображення сайту на різних типах пристроїв. Впровадження SCSS-модулів дозволило ефективно структурувати стилі та уникнути конфліктів у CSS. У рамках тестування було перевірено сумісність за стосунку в різних браузерах та на різних платформах, що підтвердило його стабільну роботу.

У перспективі проект може бути доповнений функцією авторизації, особистим кабінетом користувача, інтеграцією з backend-сервером і системою зберігання даних у базі даних. Завдяки модульній архітектурі реалізованої системи такі зміни можуть бути впроваджені без необхідності значного переписування вже створених компонентів.

Таким чином створений прототип відповідає вимогам до сучасного інтернет-магазину техніки та демонструє ефективність застосування React I TypeScripty практичній фронтенд-розробці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Duckett J. *HTML and CSS: Design and Build Websites*. Wiley, 2011.

2. Freeman E. & Robson E. *Head First JavaScript Programming*. O'Reilly Media, 2014.

3. Abramov D. *React Documentation* — https://react.dev

4. Microsoft. *TypeScript Documentation* — https://www.typescriptlang.org/docs

5. Flanagan D. *JavaScript: The Definitive Guide*. O'Reilly Media, 2020.

6. Mozilla Developer Network (MDN). *HTML, CSS, JS Guides* — https://developer.mozilla.org/

7. Vite Documentation — https://vitejs.dev/guide/

8. Marcotte E. *Responsive Web Design*. A Book Apart, 2011.

9. React Router Docs — https://reactrouter.com/en/main

10. Хмельовський В., Мотрич М., Скібчик В., Марчиниша Є., Білько Т. Охорона праці : навч. посіб. для студентів ОПП «Бакалавр». – К. : Центр учбової літератури, 2023. – 594 с.

11.Пістун І.П., Кочубей В.І. Безпека життєдіяльності. – К. : Університетська книга, 2025. – 575 с.

12.Безпека життєдіяльності. Основи охорони праці :електронний курс лекцій / І.Б. Окіпний, О.Я. Гурик. – Тернопіль : ТНТУ ім. І. Пулюя, 2025. – [Електронний ресурс] – Режим доступу: https://dl.tntu.edu.ua/index.php(доступ для зареєстрованих користувачів).

13.ДСТУ 22787-2016 Системи управління охороною праці. Загальні вимоги,[Електронний ресурс] – Режим доступу:https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=63821

14.Закон України «Про охорону праці» [Електронний ресурс] – Режим доступу:https://zakon.rada.gov.ua/laws/show/2694-12#Tex

ДОДАТКИ

ДОДАТОК А – Лістинг коду проекту

Лістингкоду 1 – файлapp.tsx

```
import React from 'react';
import './App.scss';
import { Header } from './components/Header';
import { Outlet } from 'react-router-dom';
import { Footer } from './components/Footer';
import { ProductsProvider } from
'./components/context/ProductsContext';

export const App = () => (
<div className="App">
<ProductsProvider>
<Header />
<main style={{ flexGrow: 1, background: '#fafbfc' }}>
<Outlet />
</main>
<Footer />
</ProductsProvider>
</div>
);
```

Лістингкоду 2 – файл index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-
scale=1.0" />
<title>Nice Gadgets</title>
<link rel="icon" type="image/g-icon" href="/img/logo-
title.png">
<link rel="stylesheet" href="./src/utils/resets.css">
```

```
</head>
<body>
<div id="root"></div>
<script type="module" src="/src/index.tsx"></script>
</body>
</html>
```

Лістингкоду 3 – файл_mixins.scss

```
@mixin title-of-blocks {
  font-family: $font-family;
  font-size: 32px;
  font-weight: $font-weight-bolt;

  @media (max-width: $mobile) {
    font-size: 22px;
    width: 136px;
  }
}

@mixin title-main {
  font-family: $font-family;
  font-size: 48px;
  font-weight: $font-weight-bolt;
  color: $primary;

  @media (max-width: $mobile) {
    margin: 24px 16px;
    font-size: 32px;
  }
}
```

```
@mixin title-in-item-block {
  font-family: $font-family;
  font-weight: 800;
  font-size: 22px;
  line-height: 30.8px;

  padding-bottom: 16px;
  border-bottom: 1px solid #e2e6e9;
}

@mixin menu-list {
  font-family: $font-family;
  font-size: 12px;
  font-weight: $font-weight-bolt;
  text-transform: uppercase;
  color: $secondary;
}

@mixin body-text {
  font-family: $font-family;
  font-size: 14px;
  line-height: 21px;
}

@mixin small-text {
  font-family: $font-family;
  font-size: 12px;
  line-height: 15.34px;
}
```

Лістингкоду 4 – файлRoot.tsx

```
import React from 'react';
import {
  HashRouter as Router,
  Routes,
  Route,
```

```
  Navigate,
} from 'react-router-dom';
import { App } from './App';
import { HomePage } from './components/HomePage';
import { PhonesPage } from
'./components/PhonesPage/PhonesPage';
import { ItemCard, ProductType } from './components/ItemCard';
import { TabletsPage } from
'./components/TabletsPage/TabletsPage';
import { AccessoriesPage } from
'./components/AccessoriesPage/AccessoriesPage';
import { FavouritesPage } from
'./components/FavouritesPage/FavouritesPage';
import { CartPage } from './components/CartPage';
import { NotFound } from './components/NotFound/NotFound';

export const Root = () => (
<Router>
<Routes>
<Route path="/" element={<App />}>
<Route index element={<HomePage />} />
<Route path="home" element={<Navigate to="/" replace />} />
<Route path="phones" element={<PhonesPage />} />
<Route
          path="phones/:id"
          element={<ItemCard type={ProductType.PHONE} />}
        />

<Route path="tablets" element={<TabletsPage />} />
<Route
          path="tablets/:id"
          element={<ItemCard type={ProductType.TABLET} />}
        />

<Route path="accessories" element={<AccessoriesPage />} />
<Route
          path="accessories/:id"
```

```
                element={<ItemCard type={ProductType.ACCESSORIES}
/>}
              />
      <Route path="favourites" element={<FavouritesPage />} />
      <Route path="cart" element={<CartPage />} />

      <Route path="*" element={<NotFound />} />
      </Route>
      </Routes>
      </Router>
      );
```

ДОДАТОК Б – Диск з роботою