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

Випускна кваліфікаційна роботи бакалавра

«СИСТЕМА ЕЛЕКТРОННОЇ КОМЕРЦІЇ ВЕБ-САЙТУ ТЕХНІКИ З СИСТЕМОЮ УПРАВЛІННЯ ДЛЯ АДМІНІСТРАТОРІВ»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проекту

№	П.І.Б. Група
1	Волков Юрій Ігорович КН-П-202
2	Чермалих Олексій Олександрович КН-П-202
3	Бондаренко Данило Олександрович КН-П-202

Автор звіту

_____ Чермалих Олексій Олександрович

Одеса – 2024

АНОТАЦІЯ

УДК 004.9
Ч-45

Чермалих О.О. Платформа електронної комерції, система управління. Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп’ютерні науки». - Одеса: ОТУШ, 2024 - 39 с.

Дипломний проект присвячений розробці та впровадженню платформи електронної комерції для продажу електронної техніки, а також системи керування магазином для власника бізнесу. У проекті реалізовано серверну частину, яка забезпечує зв'язок між інтернет-магазином, адміністративним додатком та базою даних для передачі даних.

Основні можливості сайту включають реєстрацію та авторизацію користувачів, зручний пошук товарів, відгуки та оформлення замовлень. Для адміністратора платформи створено окрему кросплатформну програму, яка дозволяє легко додавати або видаляти товари, обробляти замовлення, переглядати та видаляти відгуки.

Серверна частина не тільки передає дані між сайтом та програмою, але й генерує токени авторизації.

ЗМІСТ

ВСТУП

У світі інформаційних технологій електронна комерція стала невід'ємною частиною життя багатьох людей. Онлайн торгівля проникає у різні сфери бізнесу, полегшуючи процес купівлі та продажу товарів і послуг.

Метою цього проекту є створення комплексного рішення, яке спростить процеси управління.

Наш проект називається TechX і ділиться на три підпроекти:

1. **TechX Store** - сучасний і зручний магазин електроніки, який дозволяє споживачам легко знаходити товари, забезпечуючи широкий вибір продукції. Користувачі можуть редагувати свій профіль, переглядати історію замовлень, додавати товари в обрані, залишати відгуки та багато іншого.
2. **TechX Compass** - кросплатформна програма, яка дозволяє власнику сайту зручно керувати магазином.
3. **TechX Server** - обробляє запити для роботи з магазином (TechX Store), TechX Compass з базою даних.

TECHX STORE

TechX Store - інформаційна система е-комерції для магазину електроніки, яка використовує фреймворк Next.js.

Було проведено порівняльний аналіз технологій створення фронтенд додатків: Angular, React, Vue. Next.js виділяється завдяки своїй вбудованій підтримці SSR (Server-Side Rendering) та SSG (Static Site Generation), спрощеній файловій маршрутизації та високій продуктивності. Angular та Vue також пропонують багато можливостей, але їх налаштування та використання можуть вимагати більше зусиль.

Next.js – це фреймворк React для створення повнофункціональних веб-застосунків. Використовуються компоненти React для створення інтерфейсів користувача і Next.js для додаткових функцій і оптимізації. Завдяки Next.js можна зосередитися на створенні програми, а не витрачати час на налаштування.

React - це бібліотека для розробки інтерфейсу користувача. React.js спрощує розробку UI, роблячи код більш модульним, гнучким та легко підтримуваним.

У нашому проекті ми користуємось Tailwind CSS – це CSS фреймворк, який допомагає швидко верстати компоненти.

Також ми користуємось UI-бібліотеками shadcn/ui та Headless.ui. Завдяки цим бібліотекам ми можемо легко та швидко створювати різні компоненти.

Ми також використовуємо бібліотеку анімацій Framer Motion, яка дозволяє швидко налаштовувати анімації та створювати красиві ефекти.

Головна сторінка

Головна сторінка програми відображається одразу після переходу користувача на сайт

Каталог

У хедері програми реалізовано каталог товарів, який динамічно завантажує дані з бази даних. Він представлений у вигляді випадаючого списку, що відображає дані, відсортовані за категоріями.

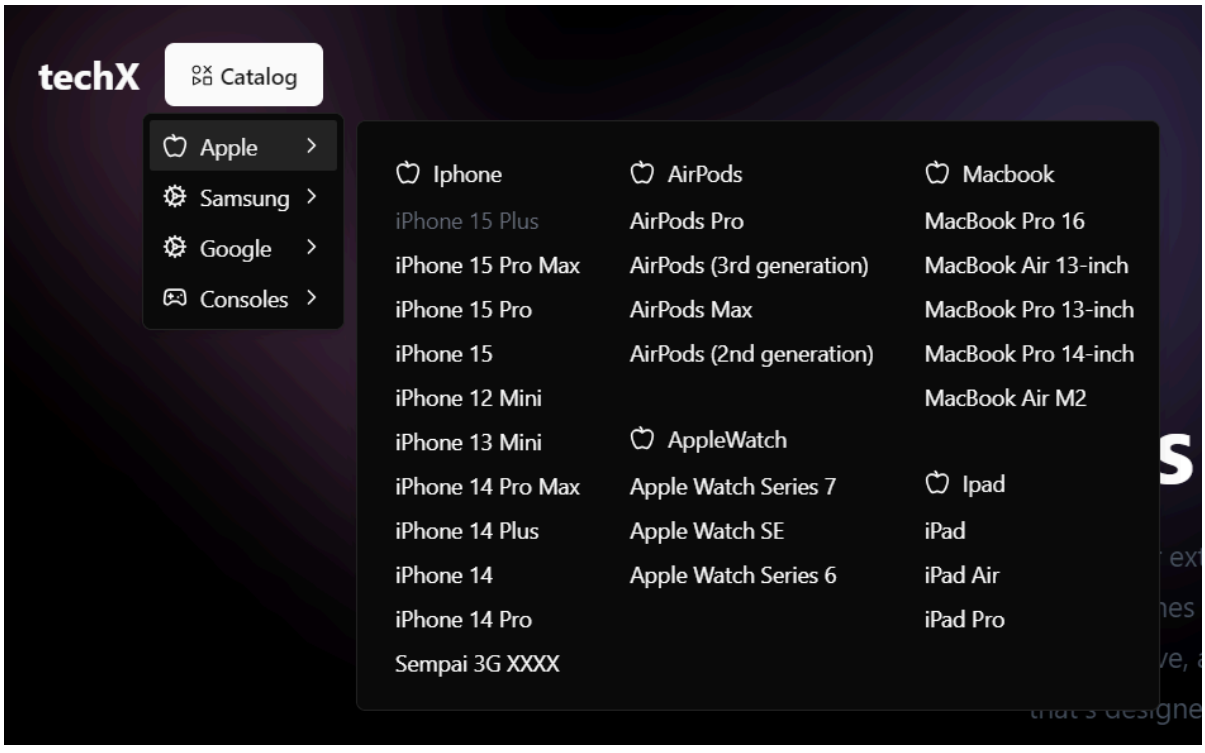

Рисунок 1 - Каталог

При виборі певної моделі товару користувач буде перенаправлений на сторінку product-detail.

Пошук

Праворуч від каталогу товарів розміщено іконку пошуку. При натисканні на іконку користувачеві відображається модальне вікно з рядком.
При натисканні на іконку пошуку користувачеві відображається модальне вікно з пошуковою сторінкою. Пошук на сайті здійснюється динамічно, запит на сервер для пошуку відображається з кожною зміною значення пошукового рядка. Цей функціонал працює в такий спосіб.

Користувач вводить текст у спеціальне поле пошуку на сайті. Це поле має placeholder, щоб вказати користувачеві його призначення. Кожна зміна відстежується функцією
handleChangeSearch . Ця функція отримує текст який користувач вводить.
Якщо поле введення порожнє, подальша обробка не виконується. Якщо текст введено, функція надсилає запит для пошуку.
Запит, який містить введений користувачем текст, надсилається на сервер.

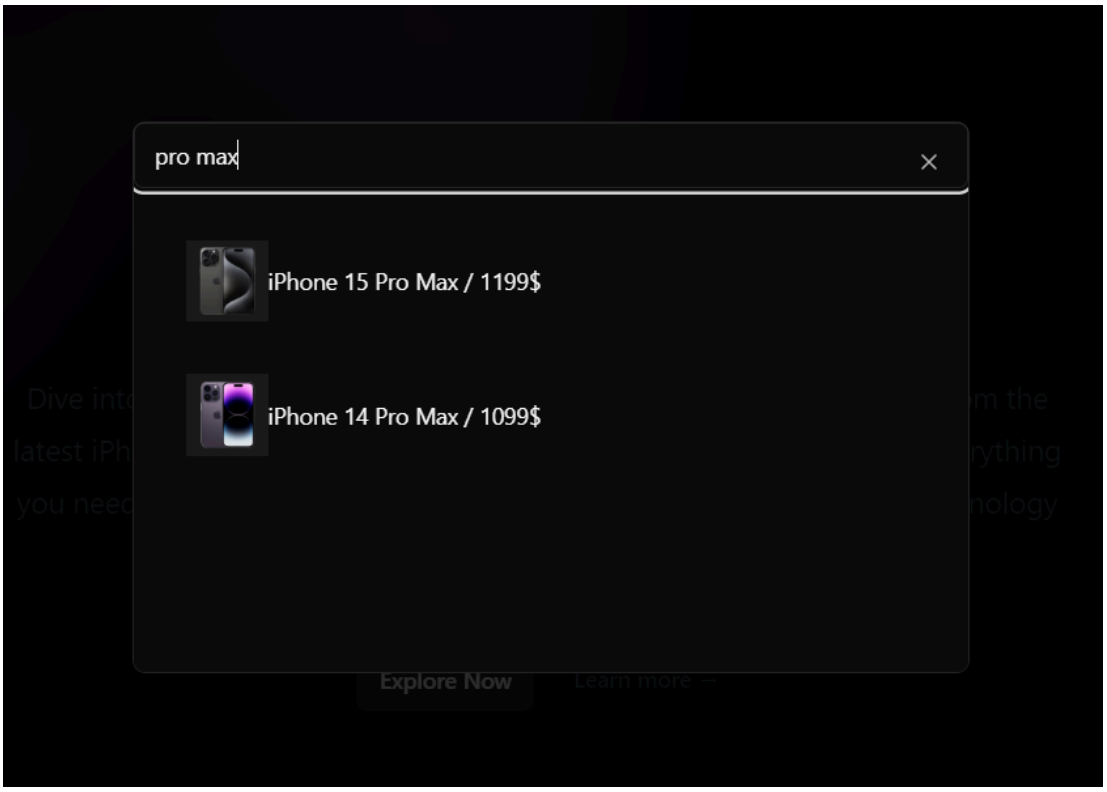

Рисунок 2 - Відображення результатів пошуку

Сервер отримує запит і виконує пошук товарів, які відповідають введеному тексту. Результати пошуку відправляються у вигляді даних у форматі JSON де зберігаються такі поля як зображення, ціна і модель.

Отримані з сервера дані використовуються для оновлення інтерфейсу сайту. Результати пошуку відображаються користувачеві, що дозволяє йому швидко знаходити потрібні товари. Обробка помилок у випадку, якщо при надсиланні або обробці запиту відбувається помилка, вона фіксується.

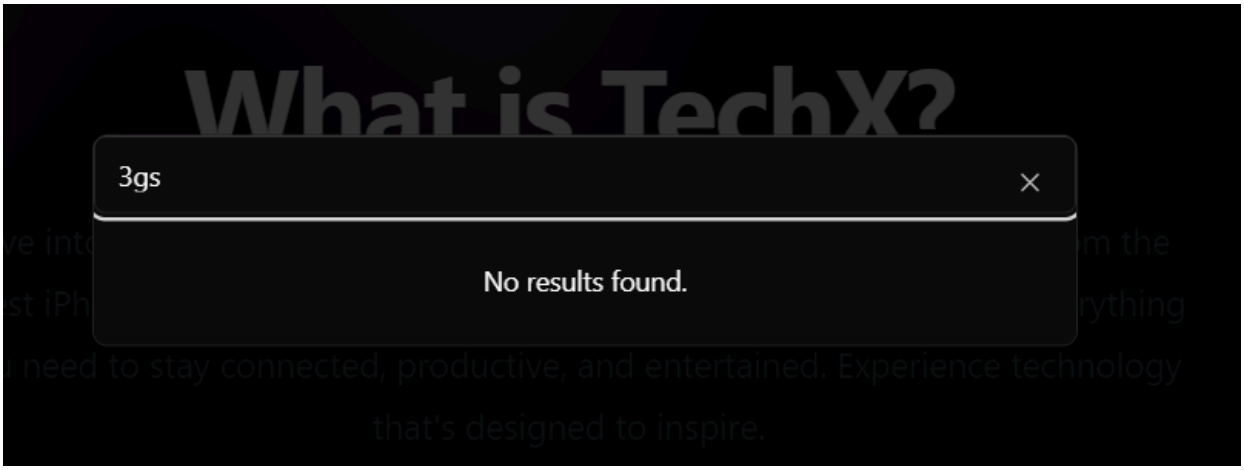

Рисунок 3 - Відображення неіснуючого товару

```
const handleChangeSearch = (e) =>
{
        const query = e.target.value.trim();
        if (query === "")
                return;
        else
                Search(query);
};

const Search = async(q) =>
{
```

```
    search_results.length = 0;
    try
    {
        const response = await
            fetch(`https://techx-server.tech:443/SearchForProducts`,
        {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({query: q}),
        });
        const data = await response.json();
        SetSearchResults(data);
    }
    catch (error) { console.error("Error searching:", error); }
};
```

Кошик

Реалізація кошика товарів на сайті дозволяє користувачам керувати своїми покупками. Користувачі можуть додавати товари в кошик та видаляти, бачити вартість замовлення та оформлювати його. Використання локального сховища забезпечує збереження даних кошика між сесіями, що покращує загальний досвід користувача і робить процес покупок більш зручним і ефективним.

При завантаженні компонента кошика він ініціалізується даними з локального сховища. Якщо в локальному сховищі є збережені товари, вони завантажуються в стан компонента.

Для синхронізації даних кошика з локальним сховищем кожні кілька секунд відбувається оновлення стану компонента. Це забезпечує актуальність даних кошика, якщо вони були змінені .

Користувач може видаляти товари з корзини. Для цього передбачена функція видалення. Після видалення товар більше не відображається у кошику.

Загальна вартість усіх товарів у кошику підраховується автоматично.

Коли користувач готовий оформити замовлення, він може натиснути кнопку Checkout, яка перенаправляє його на сторінку оформлення замовлення. Якщо користувач хоче видалити всі товари з кошика, він може натиснути кнопку "Clear all", що очищає кошик та локальне сховище.

Візуально кошик відображає всі товари, які до нього додані, включаючи зображення, назви та ціни товарів. Кожен товар має кнопку видалення, що дозволяє користувачеві видалити його з кошика. У нижній частині компонента розташовані кнопки Checkout та Clear all. Якщо кошик порожній, з'являється повідомлення "Your cart is empty".

```
useEffect(() =>
{
	SetStoredArray(JSON.parse(localStorage.getItem("Cart")) || []);
}, []);

useEffect(() =>
{
	const interval = setInterval(() =>
	{
		const cartData = JSON.parse(localStorage.getItem("Cart")) || [];
		SetStoredArray(cartData);
	}, 1000);
	return () => clearInterval(interval);
 }, [last_updated]);

const RemoveFromCart = (index) =>
{
	const updated_cart = [...stored_array];
	updated_cart.splice(index, 1);
	localStorage.setItem("Cart", JSON.stringify(updated_cart));
	SetStoredArray(updated_cart);
};

const CountTotal = () =>
{
	let total = 0;
	for (const i_price of stored_array)
```

```
                total += parseFloat(i_price.price);
        return total;
};

const GoCheckout = () => { window.location.href = "/checkout"; };

const ClearAllFromCart = (e) =>
{
        e.stopPropagation();
        localStorage.removeItem("Cart");
        stored_array.length = 0;
        SetStoredArray([]);
};
```

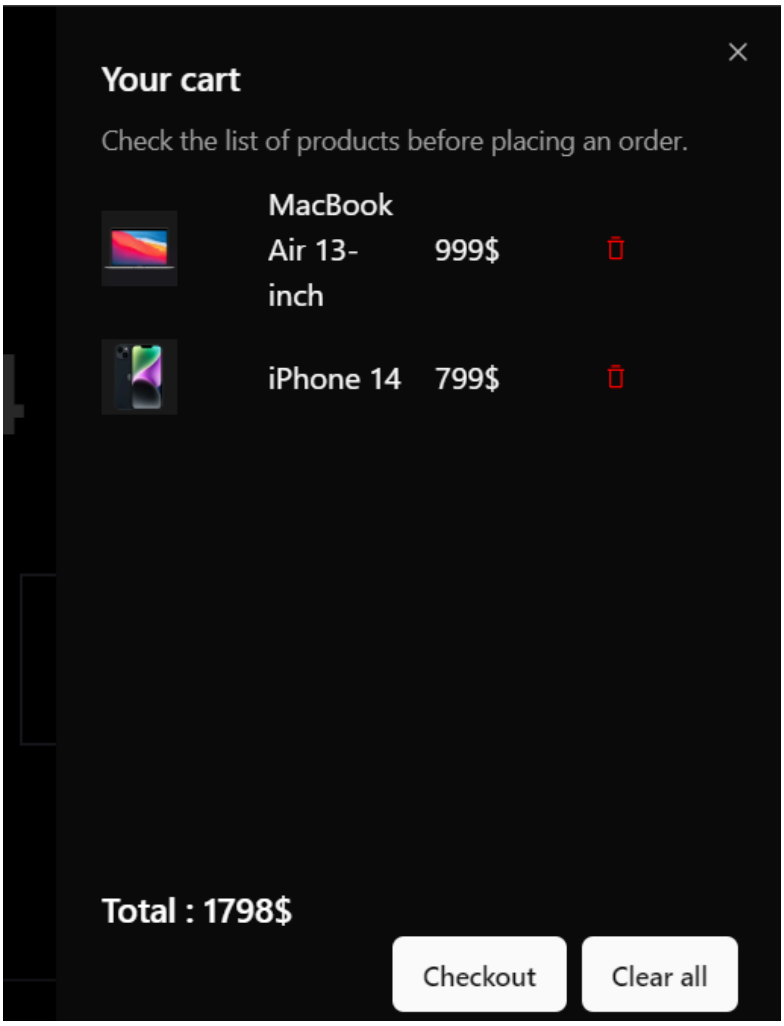

Рисунок 4 - Відображення продуктів у кошику

Реєстрація та авторизація

Біля кошика відображається значок користувача. При натисканні на цю іконку користувачеві відображається список, що випадає, і пропонується 2 варіанти. Реєстрація відображає сторінку signup, а авторизація на сторінку sign in.

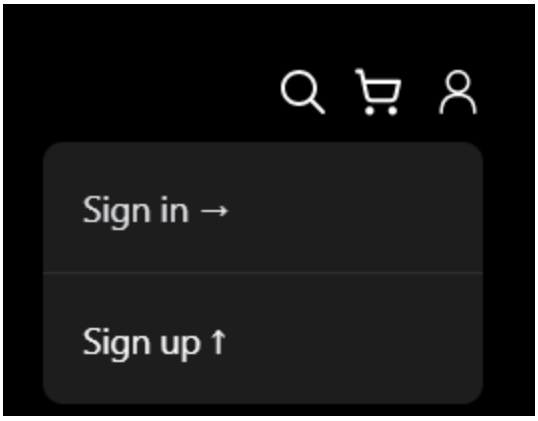

Рисунок 5 - Авторизація та Реєстрація

Каталог товарів

Каталог товарів різних категорій візуально є grid-сіткою із зображеннями та назвами категорій товарів, таких як iPhone, AirPods, Apple Watch та інші. Користувач може натиснути на будь-яку з категорій, щоб перейти до відповідних товарів. При рендері компонента, створюється сітка, що складається з декількох колонок. Кількість колонок залежить від ширини екрана пристрою користувача та регулюється класами з фреймворку Tailwind CSS. Наприклад, на невеликих екранах використовують дві колонки, а на великих — чотири. Кожен осередок сітки є посиланням на сторінку з товарами певної категорії.

При натисканні на таке посилання користувач перенаправляється на сторінку product з передачею параметрів. Це досягається використанням компонента Link з Next.js, який дозволяє виконувати навігацію всередині програми. Всередині кожного посилання розміщується компонент ProductCatalog, який відображає зображення та назву категорії товару. Зображення та назва передаються компонент як параметри. Зображення завантажуються за вказаними URL-адресами, а заголовки категорій представлені у вигляді тексту.

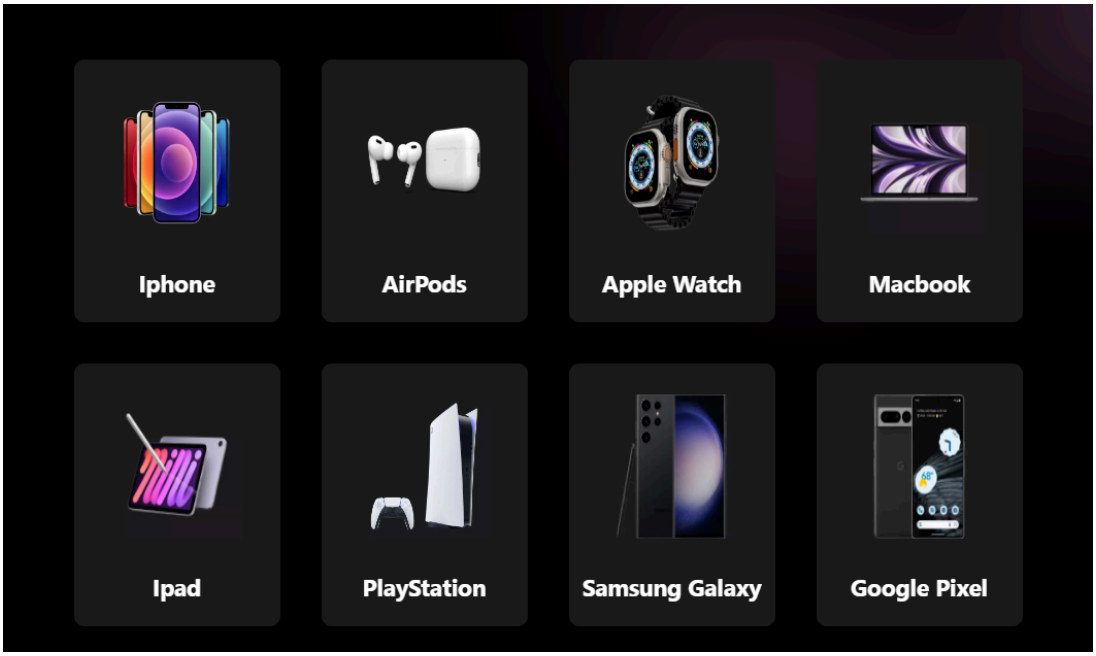

Рисунок 6 - Каталог товарів

```
<Link href={{ pathname: "/product", query: { type: "Iphone" } }}>
        <ProductCatalog
          image="https://img.jabko.ua/image/cache/home_cats/image%20154full.png.webp"
          title="Iphone"/>
</Link>
<Link href={{ pathname: "/product", query: { type: "AirPods" } }}>
        <ProductCatalog
```

```
          image="https://img.jabko.ua/image/cache/cataloge-2/main-page-2/pods-2full.png.webp"
           title="AirPods"/>
</Link>
<Link href={{ pathname: "/product", query: { type: "AppleWatch" } }}>
          <ProductCatalog
           image="https://img.jabko.ua/image/cache/home_cats/image%20155full.png.webp"
           title="Apple Watch"/>
</Link>
<Link href={{ pathname: "/product", query: { type: "Macbook" } }}>
          <ProductCatalog
          image="https://img.jabko.ua/image/cache/home_cats/macbook-air-spacegray-gallery1-2f
          ull.jpg.webp"
           title="Macbook"
/>
</Link>
<Link href={{ pathname: "/product", query: { type: "Ipad" } }}>
          <ProductCatalog
           image="https://img.jabko.ua/image/cache/home_cats/ipad-pcfull.jpg.webp"
           title="Ipad"/>
</Link>
<Link href={{ pathname: "/product", query: { type: "Console" } }}>
          <ProductCatalog
          image="https://img.jabko.ua/image/cache/cataloge-2/main-page-2/2020-07-31%2010.30.
          31%201-1397x1397full.jpg.webp"
          title="PlayStation"/>
</Link>
```

Таким чином, користувач бачить сітку із зображеннями та назвами категорій товарів. При натисканні на категорію відбувається перенаправлення на сторінку з товарами обраної категорії, що дозволяє користувачеві легко і швидко знаходити товари. Це покращує навігацію по сайту та робить його більш зрозумілим та зручним для користувачів.

Карусель рекомендацій

Компонент каруселі товарів відповідає за динамічне відображення списку товарів у вигляді каруселі, що дозволяє користувачеві перегортати пропозиції.

При завантаженні компонента за допомогою хука useEffect відбувається асинхронний запит до сервера для отримання даних про товари, які будуть відображатися в каруселі. Якщо дані успішно отримані, вони зберігаються.
Другий хук useEffect відповідає за адаптацію відображення каруселі залежно від ширини екрана. Він визначає, скільки товарів буде показано в одному слайді каруселі.

Для мобільних пристроїв відображаються два товари. Для планшетів відображаються три продукти. Для маленьких ноутбуків відображаються чотири товари. Для великих екранів та настільних комп'ютерів також відображаються чотири товари.

Для цього встановлюється слухач події зміни розміру вікна, який оновлює стан carouselSlideWidth під час зміни ширини екрана.

```
useEffect(() =>
{
	const ToGetData = async () =>
	{
		try
		{
			const formatted_data = await fetch(
					"https://techx-server.tech:443/GettingDataForCarusel",
			{
				method: "POST",
				headers: { "Content-Type": "application/json" },
			});
			if (formatted_data.ok)
				SetCData(await formatted_data.json());
		}
		catch (error) { console.error("Error fetching data:", error); }
	};
```

```
        ToGetData();
}, []);
```

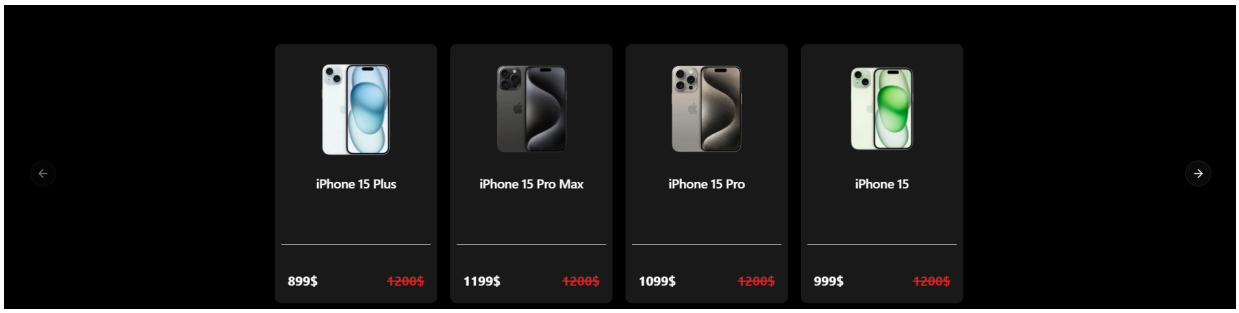

Рисунок 7 - Карусель

Карусель рендериться з використанням компонента Carousel, який має кілька плагінів, включаючи автоплей, що дозволяє автоматично перегортати слайди кожні 10 секунд. Кожен слайд формується на основі даних, які поділяються на групи.

Всередині кожного слайду товари відображаються за допомогою компонента Product Cards, який приймає параметри image, title, color та price. При натисканні товару користувач перенаправляється на сторінку з деталями товару з передачею параметра id товару.

Для навігації по каруселі використовуються компоненти CarouselPrevious та CarouselNext, які дозволяють перегортати слайди вручну.

Сторінка реєстрації

Сторінка реєстрації розроблена для надання користувачам можливості створити обліковий запис, необхідний для використання сервісу. Вона складається з кількох компонентів, що використовуються для збору даних користувача, та механізмів обробки цих даних. Основна логіка сторінки реалізується через функціональний компонент `Signup`, який використовує

хуки для відстеження даних форми, станів помилок та відображення модального вікна підтвердження пошти.

При надсиланні форми виконується функція `handleSumbit`, яка перевіряє чи заповнені всі обов'язкові поля. Потім надсилається запит на сервер для перевірки існування користувача за вказаною адресою електронної пошти. Якщо користувач з такою адресою вже існує, на сторінці відображається попередження. Інакше відкривається модальне вікно для введення коду підтвердження пошти, надісланого на вказану адресу. Після отримання коду підтвердження, користувач пропонує ввести його в модальне вікно.

Функція handleVerifyClick перевіряє правильність введеного коду і реєструє нового користувача при успішній верифікації та створює сесію. Сторінка також передбачає відображення помилок, які можуть виникнути в процесі реєстрації для інформування користувача про можливі проблеми.

```
const handleSumbit = async (e) =>
{
	e.preventDefault();
	if (!name || !email || !password)
	{
		setError("All fields are necessary.");
		setShowAlert(true);
		return;
	}
	try
	{
		const ResUserExists = await fetch(
			"https://techx-server.tech:443/CheckUserExists",
		{
			method: "POST",
			headers: { "Content-Type": "application/json" },
			body: JSON.stringify({ email }),
		});

		const { existing_user } = await ResUserExists.json();
		if (existing_user)
```

```
                {
                        showAlert(true);
                        setError("User exists");
                        console.error("User exists", error);
                        return;
                }
                else
                {
                        SetisModalConfirmMailOpen(true);
                        const SendConf = await fetch(
                                "https://techx-server.tech:443/SendConfirmationCodeEmail",
                        {
                                method: "POST",
                                headers: { "Content-Type": "application/json" },
                                body: JSON.stringify({ email }),
                        });
                        const { conf } = await SendConf.json();
                        SetConfU(conf);
                }
        }
        catch (error) {console.log("Error during registration: ", error);}
};

const handleInputChange = (index, e) =>
{
        const input = e.target;
        if (input.value.length === 1)
        {
                if (index < input_confirm_refs.length - 1)
                        input_confirm_refs[index + 1].current.focus();
        }
 };

const handleClearClick = () =>
{
        input_confirm_refs.forEach((ref) => { ref.current.value = ""; });
        input_confirm_refs[0].current.focus();
 };
```

```
const handleVerifyClick = async () =>
{
	const code = input_confirm_refs.map((ref) => ref.current.value).join("");


	if (conf_u === code)
	{
		const res = await fetch("https://techx-server.tech:443/NewUser",
		{
			method: "POST",
			headers: { "Content-Type": "application/json" },
			body: JSON.stringify({name,email,password: await
			bcrypt.hash(password, 10),
		}),
	});
}
```

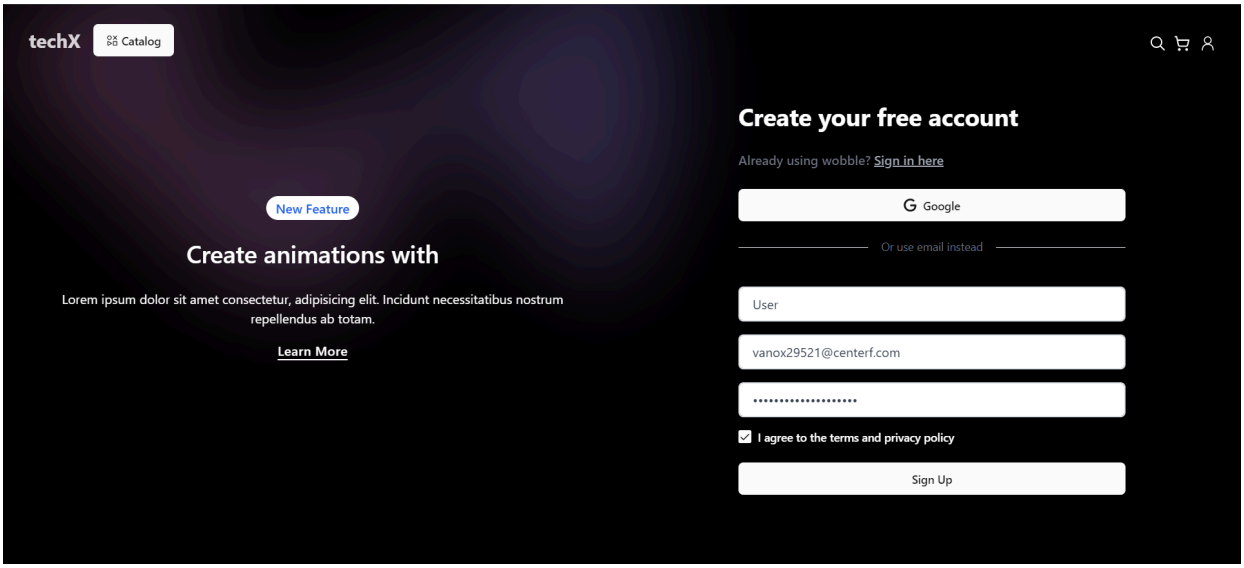

Рисунок 8 - Сторінка реєстрації

Сторінка авторизації

На сторінці авторизації користувач бачить форму, де йому пропонується ввести свою адресу електронної пошти та пароль.

Коли користувач надсилає форму, відбувається перевірка заповненості обох полів. Якщо якесь із полів залишається порожнім, з'являється повідомлення про помилку, що вказує на необхідність заповнення усіх полів. Після заповнення форми дані надсилаються на сервер для перевірки наявності користувача із вказаною адресою електронної пошти.

Якщо користувач із вказаною адресою електронної пошти існує у базі даних, відбувається додаткова перевірка введеного пароля. Для цього також надсилається запит на сервер, де пароль порівнюється з хешованим паролем. У разі успішної аутентифікації створюється токен сесії, який зберігається у локальному сховищі браузера користувача. Цей токен використовується для ідентифікації користувача при наступних запитах до захищених ресурсів програми.

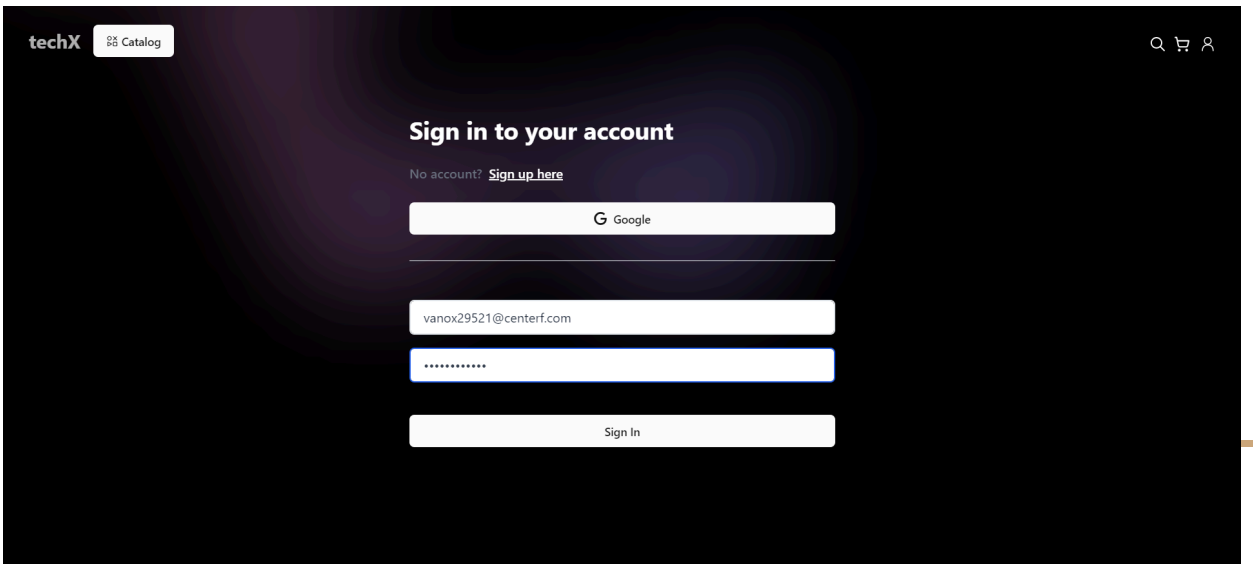

Рисунок 9 - Сторінка авторизації

```
const handleSumbit = async (e) =>
{
	e.preventDefault();
	if (!email || !password)
	{
		setError("All fields are necessary.");
		setShowAlert(true);
		return;
	}
	try
	{
		const ResUserExists = await fetch(
					"https://techx-server.tech:443/CheckUserExists",
		{
			method: "POST",
			headers: { "Content-Type": "application/json" },
			body: JSON.stringify({ email }),
		});
		const { existing_user } = await ResUserExists.json();
		if (existing_user)
		{
			const ResProofPass = await fetch(
				"https://techx-server.tech:443/ProofPass",
			{
				method: "POST",
				headers: { "Content-Type": "application/json" },
				body: JSON.stringify({ email, password }),
```

```
});
const { success } = await ResProofPass.json();
if (success)
{
        try
        {
                const reply_token = await fetch(
                        "https://techx-server.tech:443/GenerateToken",
                {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                        body: JSON.stringify({ email }),
                });
                const { token } = await reply_token.json();
                localStorage.setItem("token", token);
                const CreateSessionResponse = await fetch(
                        "https://techx-server.tech:443/CreateSession",
                {
                        method: "POST",
                        headers: { "Content-Type": "application/json",
                                        Authorization: `Bearer ${token}`},
                        body: JSON.stringify({ email }),
                });

        const create_session_data = await CreateSessionResponse.json();
        if (create_session_data)
        {.
                window.location.href = "/";
                console.log("Sessia is complite");
        }
        else
                console.log("Sessia is not complite");
}
catch (error)
{
        setError("Error during signing the token.");
        setShowAlert(true);
        console.error("Error during signing the token:", error);
}
```

```
                }
                else
                {
                        setShowAlert(true);
                        setError("Password is incorrect");
                        console.error("SIGN IN: false", error);
                        return;
                }
        }
        else
        {
                 setShowAlert(true);
                setError("User not found");
                console.error("User not found");
                return;
        }
}
catch (error) { console.log("Error during registration: ", error); }
};
```

Якщо введені дані не збігаються з даними з бази даних (наприклад, неправильний пароль), або користувач із вказаною адресою електронної пошти не знайдено, відображається повідомлення про помилку, що інформує користувача про проблему автентифікації.

Обробка сеансів

Важливо, що у додатку всі дані користувача передаються серверу лише з використанням токенів. Це забезпечує безпеку переданих даних та

захищає їх. Вся інформація, що передається до сервера, захищена шляхом передачі через токени, забезпечуючи безпеку та надійність даних.

Функція PullOutOfSession починає свою роботу з перевірки наявності токена у локальному сховищі браузера. Якщо токен присутній, надсилається запит на сервер для перевірки підпису токена. Для підтвердження автентичності токена використовується механізм надсилання запиту з даними токена на сервер. У разі успішної перевірки підпису токена сервером, функція продовжує виконання та надсилає запит на сервер для вилучення даних користувача з його сеансу. Отримані дані користувача зберігаються і повертаються функцією, щоб бути використаними у подальшій роботі програми.

У випадку, якщо перевірка токена не вдалася або токен відсутній, відбувається очищення даних про сеанс користувача з локального сховища, щоб запобігти несанкціонованому доступу до даних.

```
export const PullOutOfSession = async () =>
{
	if (localStorage.getItem("token") == null)
		return false;

	const CheckSignature = await fetch( "https://techx-server.tech:443/CheckToken",
	{
		method: "POST",
		headers: { "Content-Type": "application/json",
		 Authorization: `Bearer ${localStorage.getItem("token")}`,
	  },
	});
	const { success } = await CheckSignature.json();
	if (success)
	{
		console.log("TOKEN IS TRUE");
		const PullData = await fetch("https://techx-server.tech:443/PullOutOfSession",
		{
			method: "POST",
			headers:
			{
```

```
                        "Content-Type": "application/json",
                        Authorization: `Bearer ${localStorage.getItem("token")}`,
                    },
                });
            const { user_data } = await PullData.json();
            console.log(user_data);
            if (user_data != null)
                return user_data;
            }
            else
                localStorage.clear("token");
        return false;
};
```

Сторінка Product Menu

Сторінка створена для спрощення процесу вибору продуктів із каталогу шляхом надання користувачеві можливості застосовувати фільтри. Метою розробки даного компонента є підвищення зручності інтерфейсу користувача та забезпечення ефективного пошуку та вибору продуктів.

Компонент Product Selection надає наступний функціонал:

1. Фільтрування продуктів: Користувачі можуть фільтрувати продукти за параметрами, такими як пам'ять, колір та версія. Для кожного із цих параметрів надаються відповідні опції вибору.
2. Отримання даних про продукти: Під час завантаження компонента надсилається запит на сервер для отримання даних про продукти певного типу. Отримані дані використовуються для відображення наявних продуктів.
3. Відображення відфільтрованих продуктів: Після застосування фільтрів відображається список продуктів, які відповідають вибраним критерієм фільтрації. Кожен продукт представлений у вигляді картки із зображенням, найменуванням, характеристиками та ціною.
4. Перехід до детальної інформації про продукт: При натисканні на картку продукту користувач перенаправляється на сторінку з детальною інформацією вибраного продукта.

Процес роботи компонента Product Selection описується так. У компоненті використовуються хуки useState для зберігання поточних значень фільтрів пам'яті, кольору та версії, а також для зберігання даних про продукти. Початкові значення фільтрів встановлюються порожніми масивами. Опції для фільтрів зберігаються у попередньо заданих масивах.

Під час завантаження компонента надсилається запит на сервер для отримання даних про продукти певного типу. Отримані дані обробляються та зберігаються у стані компонента.

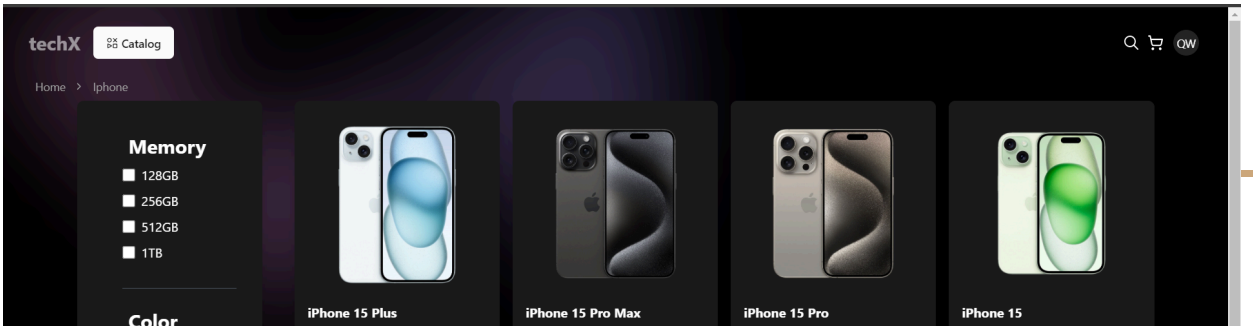

Рисунок 10 - Сторінка product-menu

Користувач вибирає необхідні параметри фільтрації, після чого стан фільтрів оновлюються за допомогою обробників подій handleCheckboxChange. Фільтрування продуктів здійснюється за допомогою функції filterProducts, яка повертає масив відфільтрованих продуктів на основі вибраних параметрів.

Відфільтровані продукти відображаються у вигляді карток на сторінці. При натисканні на картку користувач перенаправляється на сторінку з детальною інформацією про вибраний продукт. Елементи інтерфейсу стилізовані відповідно до сучасних дизайн-стандартів, що забезпечує зручність використання компонента і підвищує привабливість інтерфейсу користувача.

```
useEffect(() =>
{
	const ToGetData = async (type_p) =>
	{
		setProducts([]);
		try
		{
			const formatted_data = await fetch(
				`https://techx-server.tech:443/GetDataForListProduct/${type_p}`,
				{
					method: "POST",
					headers: { "Content-Type": "application/json" },
```

```
                });


                  if (formatted_data.ok)
                        setProducts(await formatted_data.json());
            }
            catch (error) { console.error("Error fetching data:", error); }
        };
        const params = new URLSearchParams(window.location.search);
        const product_type = params.get("type");
        if (product_type !== null)
        {
            switch (product_type)
            {
                case "Iphone":
                    ToGetData("Iphone");
                    break;
                case "AirPods":
                    ToGetData("AirPods");
                    break;
                case "AppleWatch":
                    ToGetData("AppleWatch");
                    break;
                case "Macbook":
                    ToGetData("Macbook");
                    break;
                case "Ipad":
                    ToGetData("Ipad");
                    break;
                case "Console":
                    ToGetData("Console");
                    break;
            }
        }
        else
            window.location.href = "/404";
}, []);
```

Сторінка Product Details

Компонент Product Details розроблений для надання користувачам детальної інформації про вибрані продукти, а також для забезпечення різних дій з ними, таких як додавання до обраного, написання відгуків та додавання до кошика.

При ініціалізації компонента відбувається встановлення початкових значень різних станів, таких як ємність, ідентифікатор продукту, відображення контенту для авторизованих користувачів і дані про продукт.

У компоненті реалізована функція, яка дозволяє користувачам додавати або видаляти продукти зі списку вибраних. При натисканні на відповідну кнопку стан "liked" змінюється і надсилається запит на сервер для оновлення списку вибраних. Також є функція handleSendReview, яка обробляє надсилання відгуків про продукт. Залежно від статусу авторизації користувача відгук надсилається із зазначенням його імені або електронної пошти. Для додавання продукту в кошик реалізовано функцію handleAddToCart. Під час натискання на відповідну кнопку вибраний продукт додається до кошика та

з'являється повідомлення про додавання.

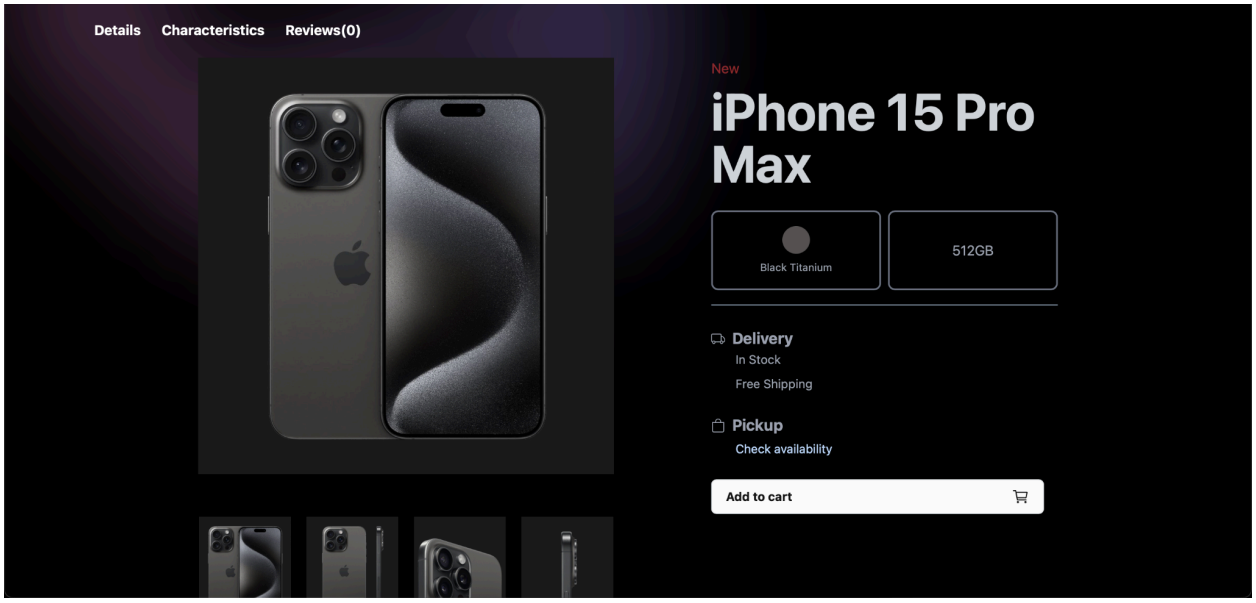

Рисунок 11 - Сторінка product-details

При завантаженні компонента виконуються запити на сервер для отримання інформації про продукт та його відгуки. Отримані дані зберігаються та використовуються для відображення на сторінці. Для керування повідомленнями про дії користувача використовуються стани showAlert та message. Анімації елементів інтерфейсу реалізовані з використанням бібліотеки Framer Motion, що надає інтерактивності та привабливості користувальницькому досвіду.

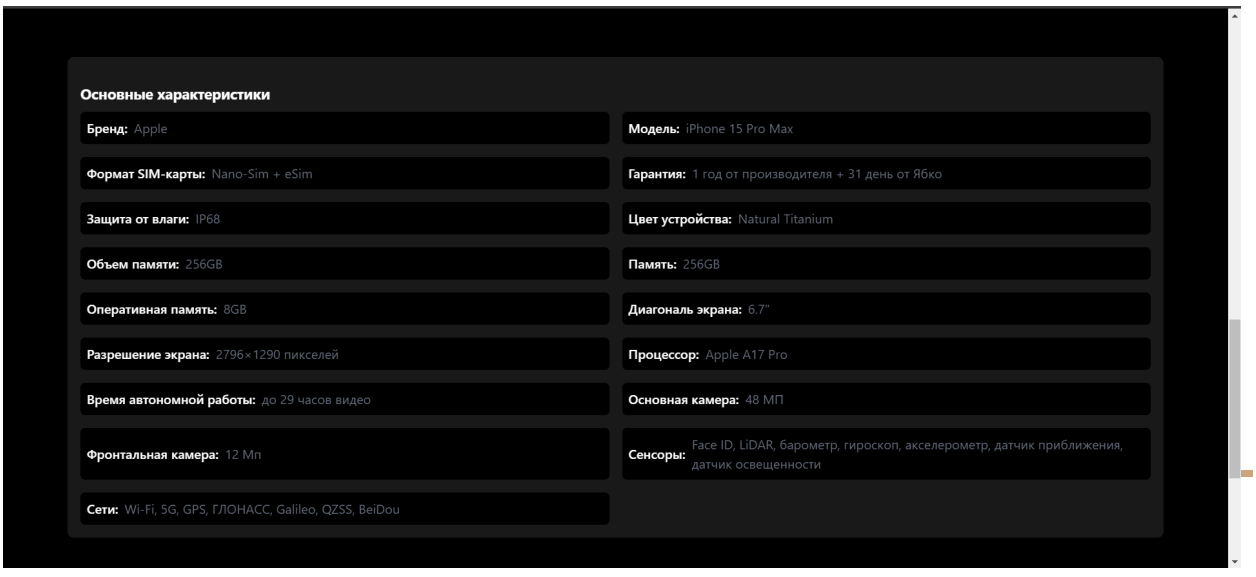

Рисунок 12 - Відображення характеристики

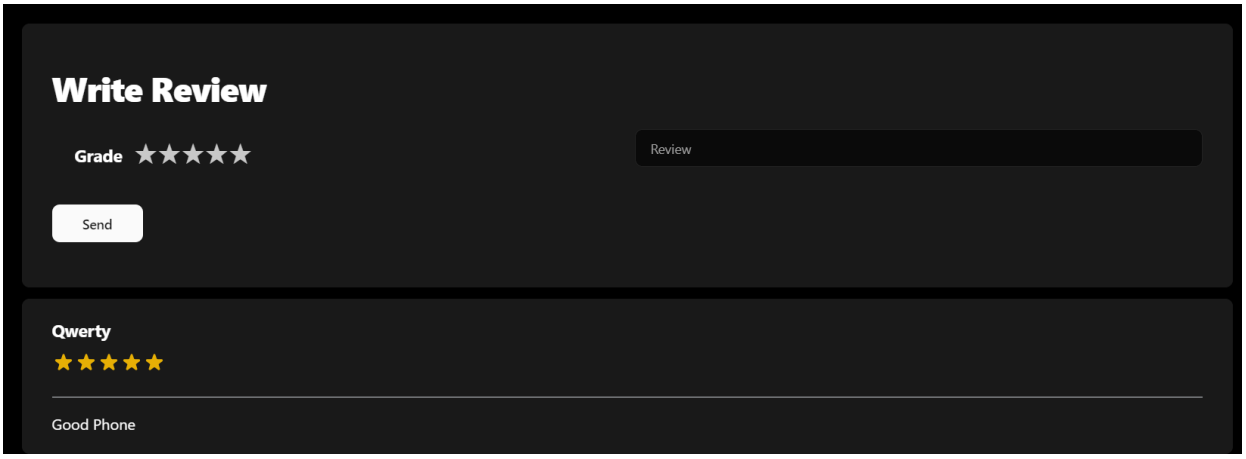

Рисунок 13 - Відображення відгуків

```
const toggleLike = async () =>
{
	setLiked(!liked);
	setShowAlert(true);
	setMessage( liked ? "Product removed from favorites" : "Product added to favorites");
	const id_product = new URLSearchParams(window.location.search).get("id");
	if (!liked)
	{
		const formatted_data = await fetch(
			`https://techx-server.tech:443/AddFavoriteProduct/${id_product}`,
		{
			method: "POST",
			headers: { "Content-Type": "application/json",
				Authorization: `Bearer ${localStorage.getItem("token")}`,
```

```
                }});
        }
        else
        {
                const formatted_data = await fetch(
                        `https://techx-server.tech:443/DeleteFavoriteProduct/${id_product}`,
                {
                        method: "POST",
                        headers: { "Content-Type": "application/json",
                                Authorization: `Bearer ${localStorage.getItem("token")}`,
                }});
        }
 };

useEffect(() =>
{
        const handleScroll = (event) =>
        {
                event.preventDefault();
                const targetId = event.target.getAttribute("href").slice(1);
                const targetElement = document.getElementById(targetId);
                if (targetElement)
                        window.scrollTo({ top: targetElement.offsetTop, behavior: "smooth" });
        };
        const scrollLinks = document.querySelectorAll('a[href^="#"]');
        scrollLinks.forEach((link) =>
        {
                link.addEventListener("click", handleScroll);
        });
        return () =>
        {
                scrollLinks.forEach((link) => { link.removeEventListener("click", handleScroll);
        });};
 }, []);




useEffect(() =>
{
        const ToGetData = async (id) =>
        {
                try
                {
```

```
                const formatted_data = await fetch(
                                `https://techx-server.tech:443/ExtractData/${id}`,
                {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                });

                if (formatted_data.ok)
                {
                        const data = await formatted_data.json();
                        SetProductData(data);
                        setSelectedImage(data.images[0]);
                }
        }
        catch (error) { console.error("Error fetching data:", error); }
};
const GetProductReviews = async () =>
{
        try
        {
                const id_product = new
                URLSearchParams(window.location.search).get("id");
                const rew = await fetch(
                        `https://techx-server.tech:443/GetProductReview/${id_product}`,
                {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                });
                const data = await rew.json();
                if (rew.ok)
                        SetAllReviews(data);
}
catch (error) { console.error("Error fetching product reviews:", error); }
};
```

Сторінка профілю

Profile Page - це компонент, призначений для відображення та керування профілем користувача на веб-платформі. Він надає користувачеві зручний інтерфейс для роботи з особистою інформацією, обраними продуктами, історією замовлень та налаштуваннями облікового запису.

Коли сторінка завантажується, компонент бере інформацію про поточного користувача та його улюблені продукти. Для кожного улюбленого продукту надсилається запит на сервер для отримання додаткової інформації.

Користувачі можуть редагувати свій обліковий запис у розділі "Account". Вони можуть змінювати своє ім'я, адресу електронної пошти, номер телефону та адресу доставки. Редагування здійснюється за допомогою діалогових вікон.

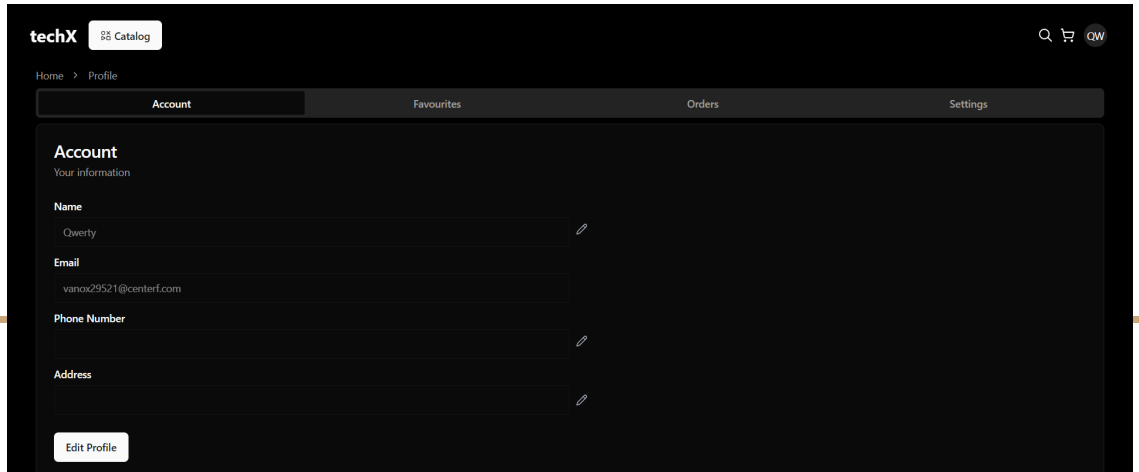

Рисунок 14 - Сторінка профілю

У розділі "Favourites" користувачі можуть керувати списком вибраних продуктів. Вони можуть додавати та видаляти продукти з обраного, а також змінювати статус лайка, який відображається у вигляді піктограми серця.

Розділ "Orders" відображає історію замовлень користувача за допомогою таблиці замовлень. Користувачі можуть переглядати деталі кожного замовлення, такі як дата, номер замовлення та статус доставки.

У розділі "Settings" користувачі можуть налаштовувати параметри свого облікового запису та персоналізувати свій досвід використання платформи. Це може включати зміну пароля, мови інтерфейсу та інших налаштувань.

ВИСНОВКИ

Було проведено аналіз інформаційних систем, що підтримують функціонування інтернет-магазинів, і виявлено, що найбільш поширеними є системи з розподіленою архітектурою для ефективного управління даними та взаємодії з користувачами. Для реалізації було обрано систему управління інтернет-магазином, яка включає три основні компоненти: фронтенд на Next.js, адмін-панель та серверну частину .

Проект TechX-Store реалізований на платформі Next.js, що забезпечує високопродуктивний та SEO-оптимізований веб-додаток для користувачів. Цей додаток дозволяє користувачам переглядати каталог товарів, здійснювати пошук, додавати товари до улюблених та оформлювати замовлення. Next.js

забезпечує швидке рендеринг сторінок та легку інтеграцію з іншими сервісами.

Адмін-панель TechX-Compass розроблена на базі Electron, що дозволяє створювати кросплатформенні настільні додатки з використанням веб-технологій.

Серверна частина TechX-Server написана на Node.js і є основним інструментом для обробки запитів, управління базою даних та реалізації бізнес-логіки. Сервер використовує MongoDB .

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

№ п.п.	**Ресурс**	**Ступінь запозичення**
1.	Next.js Розробник: Vercel; Документація: https://nextjs.org/	Використано у якості фреймворку React для платформи для розробки Web-додатку.
2.	React.js Розробник: Facebook;	Використано у якості платформи для розробки Web-додатку.

	Документація: https://react.dev/	
6.	JavaScript Розробник: Brendan Eich; Документація: https://developer.mozilla.org/en-US/docs/Web/JavaScript	Використовувався для написання системи для керування адміністраторів та сервера
7.	JSX Розробник: Jordan Walke; Документація: https://ru.legacy.reactjs.org/docs/introducing-jsx.html	Використано у якості язика програмування для розробки Web-додатку.
9.	Visual Studio Code Розробник: Microsoft; Документація: https://code.visualstudio.com/	Використовувалося як редактор коду