

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Веб-сайт для збору та продажу ПК»

Виконав студент групи ІПЗ-20-3
Ясен Антон Євгенійович

Керівник роботи:
Антонюк Дмитро Сергійович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЯСЕН Антон Євгенійович**

Керівник роботи: **АНТОНЮК Дмитро Сергійович**

Тема роботи: **«Веб-сайт для збору та продажу ПК»**,

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**

Вихідні дані для роботи: **використання сучасних технологій для проектування, розробки та реалізації веб-сайту для збору та продажу ПК**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Антонюк Д.С.	27.02.2024р.	27.02.2024р.
2	Антонюк Д.С.	27.02.2024р.	27.02.2024р.
3	Антонюк Д.С.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-сайту з збору та продажу ПК та пояснювальної записки. Пояснювальна записка до випускної роботи містить 73 сторінки, 29 ілюстрацій та 6 таблиць.

Метою роботи є розробка веб-сайту для збору та продажу ПК, який призначений для надання користувачам можливості зручно замовляти індивідуальні конфігурації комп'ютерів та купувати готові рішення, забезпечуючи високий рівень обслуговування та підтримки.

У роботі визначено основні завдання на розробку системи, проведено аналіз вимог та функціональності додатку, також був проведений аналіз аналогів створеного програмного продукту. Було здійснено проектування бази даних та обрано набір сучасних технологій розробки, використання яких забезпечує високу продуктивність додатку.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, ГОТОВІ КОНФІГУРАЦІЇ КОМП'ЮТЕРІВ, ІГРОВІ КОМП'ЮТЕРИ, КОМПЛЕКТУЮЧІ, REACT.JS, DATABASE.

					ІПЗ.КР.Б-121-24-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Веб-сайт для збору та продажу ПК	Літ.	Арк.	Аркушів
Розроб.	Ясен А.Є.						3	73
Керівник	Антонюк Д.С.					Житомирська політехніка, група ІПЗ-20-3		
Керівник								
Н. контр.	Тичина О.П.							
Зав. каф.	Вакалюк Т.А.							

ABSTRACT

The bachelor's thesis consists of a web application for the exchange of a website for the collection and sale of PCs and an explanatory note. The explanatory note to the final work contains 73 pages, 29 illustrations and 6 tables.

The aim of the work is the development of a website for collecting and selling PCs, which is designed to provide users with the opportunity to conveniently order individual computer configurations and buy ready-made solutions, providing a high level of service and support.

The paper defines the main tasks for developing the system, analyzed the requirements and functionality of the application, and also analyzed the analogues of the created software product. The database was designed and a set of modern development technologies was chosen, the use of which ensures high application performance.

KEY WORDS: WEB APPLICATION, READY COMPUTER CONFERENCE, GAMING COMPUTERS, REACT.JS, REDUX.

ЗМІСТ

РЕФЕРАТ	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБДОДАТКУ ДЛЯ ЗБОРУ ТА ПРОДАЖУ ПК.....	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	16
Висновки до першого розділу.....	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-САЙТУ ДЛЯ ЗБОРУ ТА ПРОДАЖУ ПК	19
2.1 Визначення варіантів використання та структури системи	19
2.2 Розробка бази даних системи	23
2.3 Проектування та реалізація алгоритмів роботи системи	27
2.4 Реалізація веб-сайту для збору та продажу ПК	29
Висновки до другого розділу.....	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБ-САЙТУ ДЛЯ ЗБОРУ ТА ПРОДАЖУ ПК.....	43
3.1 Порядок встановлення та налаштування параметрів системи.....	43
3.2 Структура інтерфейсу веб-додатку та порядок роботи	44
3.3 Тестування роботи вебдодатку	51
Висновки до третього розділу	53
ВИСНОВКИ.....	54
СПИСОК ЛІТЕРАТУРИ.....	55
ДОДАТКИ.....	57

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

ПЗ – Програмне забезпечення

БД – база даних

СУБД – система управління базами даних

					ІПЗ.КР.Б-121-24-ПЗ	Арк.
						6

ВСТУП

Актуальність теми. В сучасному світі персональні комп'ютери є невід'ємною частиною повсякденного життя. Вони використовуються для широкого спектру завдань: від роботи та навчання до розваг та творчості. З розвитком технологій вимоги до продуктивності та функціональності комп'ютерів постійно зростають, що робить питання вибору та складання ПК особливо актуальним.

Попит на індивідуалізовані рішення у сфері ПК збільшується, оскільки користувачі бажають отримати пристрої, що найкраще відповідають їхнім конкретним потребам. Ігрові комп'ютери, робочі станції для графічного дизайну, відеомонтажу та програмування вимагають різних конфігурацій, що робить стандартні готові рішення менш привабливими.

На цьому фоні розробка веб-сайту для збору та продажу ПК стає надзвичайно актуальною. Така платформа надає користувачам можливість зручно підбирати та замовляти індивідуальні конфігурації, отримувати професійні консультації, а також користуватися додатковими послугами, такими як монтаж та налаштування ПК. Важливим аспектом є також забезпечення високого рівня підтримки клієнтів, що підвищує довіру до бренду та стимулює повторні покупки.

Сучасні тенденції також вказують на зростаючу важливість зручності онлайн-покупок та персоналізації товарів. Розвиток e-commerce дозволяє значно розширити ринок та залучити клієнтів з різних регіонів, що робить створення веб-сайту з інтерактивними можливостями для складання ПК не лише актуальним, але й необхідним кроком для успішного ведення бізнесу в цифрову епоху.

Таким чином, проект веб-сайту для збору та продажу ПК відповідає поточним вимогам ринку, задовольняючи потреби користувачів у персоналізованих, високоякісних рішеннях, і водночас сприяє розвитку сучасних технологій та інновацій у сфері комп'ютерної техніки.

Метою кваліфікаційної роботи є розробка та проектування веб-сайту для збору та продажу ПК, який призначений для забезпечення можливості користувачам зручно підбирати, замовляти та купувати індивідуальні конфігурації комп'ютерів, а також готові рішення, із забезпеченням високого рівня обслуговування та підтримки без необхідності фізичного відвідування магазину.

Для реалізації поставленої мети потрібно виконати наступні завдання:

1. Створити інтуїтивно зрозумілий та зручний інтерфейс для користувачів.
2. Визначити архітектуру системи та спроектувати базу даних для зберігання даних.
3. Забезпечити можливість зручного оформлення замовлення, оплати та відстеження доставки для користувачів.

Об'єктом дослідження є процес розробки та функціонування веб-сайту для збору та продажу ПК, включаючи його технічні, функціональні та користувацькі аспекти.

Предметом дослідження є методи та інструменти, які використовуються для розробки веб-сайту для збору та продажу ПК, а також процеси, що забезпечують зручність користування, інтерактивність, безпеку та ефективність роботи платформи.

Розроблений програмний продукт можна використовувати як веб-сайт підприємств або компаній, що займаються збором та продажем ПК. Окрім поточних можливостей, веб-сайт має значний потенціал для подальшого розвитку та розширення функціоналу, що дозволить йому задовольняти зростаючі потреби ринку та пропонувати користувачам ще більше зручних та корисних функцій.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБДОДАТКУ ДЛЯ ЗБОРУ ТА ПРОДАЖУ ПК

1.1 Постановка задачі

У сучасному світі комп'ютери та їх компоненти відіграють важливу роль у повсякденному житті і бізнесі. Створення веб-сайту для збору та продажу комп'ютерів та комплектуючих є актуальним завданням, що відповідає зростаючим потребам ринку та сприяє зручності покупців та продавців.

Основною метою цього проекту є розробка інноваційного веб-сайту, який надасть користувачам можливість легко та ефективно збирати та купувати комп'ютерні системи та компоненти. В рамках цього проекту я покладаю за мету створення зручного та безпечного середовища для покупців ПК.

Для досягнення цієї мети я ставлю перед собою наступні завдання:

1. Аналіз вимог та функціональності веб-сайту:

- а). визначення потреб цільової аудиторії та ключових функціональних вимог до платформи;
- б). аналіз сучасних трендів у сфері електронної комерції та онлайн-торгівлі комп'ютерною технікою;

2. Проектування та створення архітектури веб-сайту;

- а). розробка структури бази даних для ефективного зберігання і керування продуктами;
- б). вибір та інтеграція платформи для електронної комерції з урахуванням потреб проекту;

3. Розробка дизайну та інтерфейсу користувача:

- а). створення привабливого та інтуїтивно зрозумілого інтерфейсу;
- б). адаптація дизайну під різні типи пристроїв та екрани для зручного користування на будь-яких пристроях;

4. Реалізація функціоналу веб-сайту:

- а). розробка системи реєстрації та авторизації користувачів;
- б). створення інтерфейсу для додавання та управління продуктами (ПК та компонентами);
- с). інтеграція платіжних систем для забезпечення безпечної та зручної оплати;

5. Тестування та впровадження веб-сайту

Результатом реалізації цих завдань буде функціональний веб-сайт для збору та продажу комп'ютерів та комплектуючих, який надасть користувачам зручність, безпеку та ефективність у процесі покупки та продажу ПК.

1.2 Аналіз аналогів програмного продукту

У ході дослідження ринку онлайн-платформ для збору та продажу ПК було проведено детальний аналіз конкурентів і визначено їхні переваги та недоліки. Користувачі завжди вимагають відмінного сервісу, який поєднує в собі зручність, надійність та ефективність в процесі купівлі та продажу комп'ютерної техніки. Наш веб-сайт прагне відповісти на ці потреби, забезпечуючи швидку та безпечну транзакцію, а також інтуїтивно зрозумілий і адаптивний інтерфейс для зручного користування на будь-яких пристроях.

Одним з таких додатків, який є аналогом розробленого додатку, є «brain.com.ua»[1]. Даний додаток є відомою платформою для покупки ПК яка відзначається своєю накопичувальною системою та різноманітним вибором товарів(рис. 1.1).

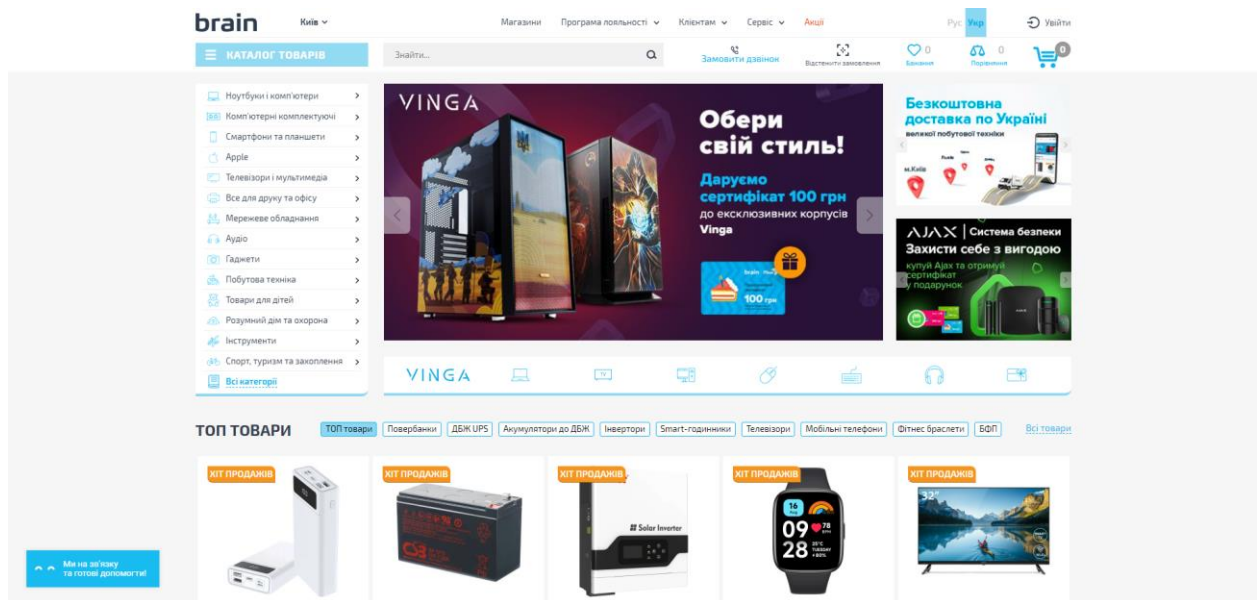


Рисунок 1.1 – Головна сторінка у веб-додатку «brain.com.ua»

Однією з ключових переваг платформи є широкий спектр товарів. Величезний асортимент включає в себе різні категорії комп'ютерної техніки та комплектуючих, що дозволяє користувачам легко знайти необхідні продукти, незалежно від їхніх потреб. Завдяки цьому, користувачі можуть швидко підібрати всі необхідні компоненти для збору персонального комп'ютера, що робить процес покупки зручним та ефективним (рис. 1.2).

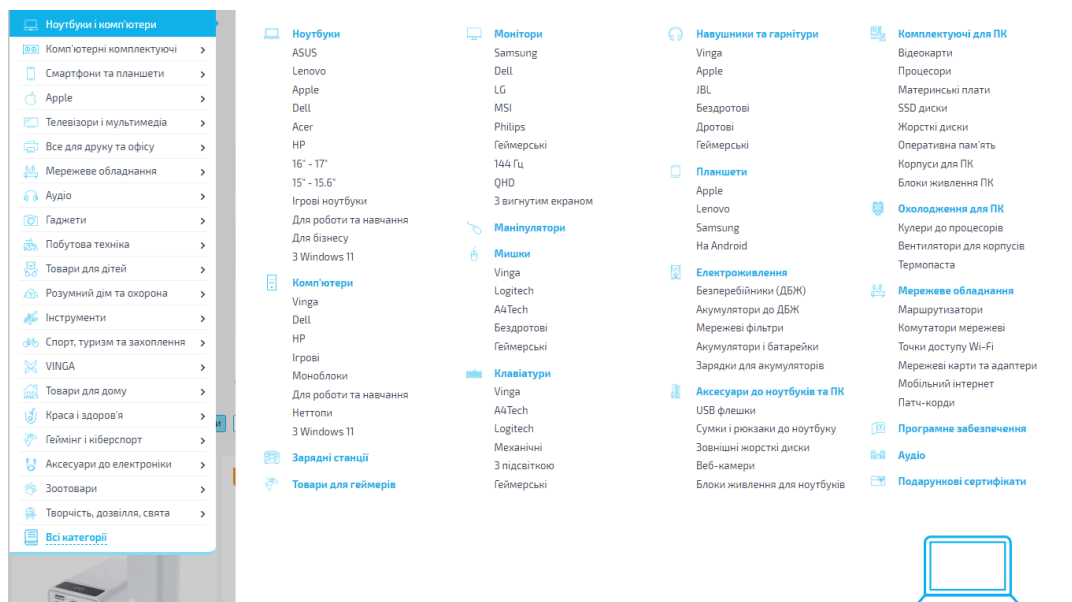


Рисунок 1.2 – Список категорій товару у веб-додатку «brain.com.ua»

Сайт brain.com.ua[1] містить велику кількість різноманітної інформації, починаючи від детальних описів товарів і закінчуючи корисними порадами

щодо вибору комп'ютерних комплектуючих для нових користувачів. Інтерфейс сайту представлений вісьмома мовами. Слід зауважити, що сайт є повністю адаптованим для різних пристроїв, включаючи мобільні телефони та планшети. Мобільна версія сайту реалізована на високому рівні, вона має привабливий та інтуїтивно зрозумілий інтерфейс, що забезпечує зручність користування для всіх категорій користувачів (рис. 1.3).

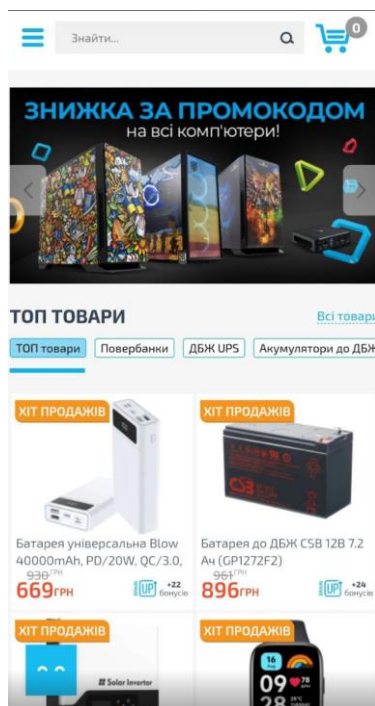


Рисунок 1.3 – Мобільна версія веб-додатку «brain.com.ua»

Іншим українським аналогом є веб-додаток «expertpc.ua»[3] (рис. 1.4).
Додаток представляє послуги лише по ПК. Він містить можливість
самостійної конфігурації ПК та редагування готових.

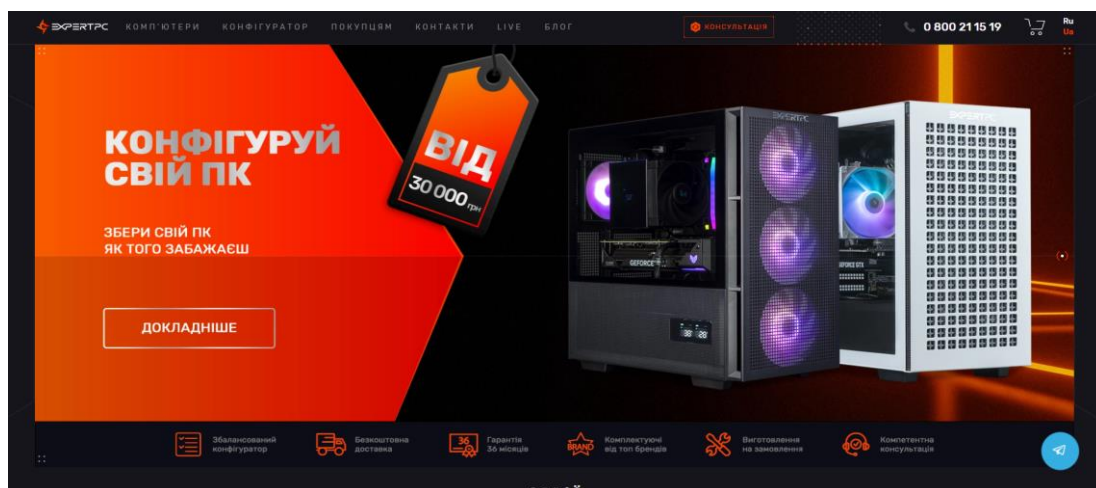
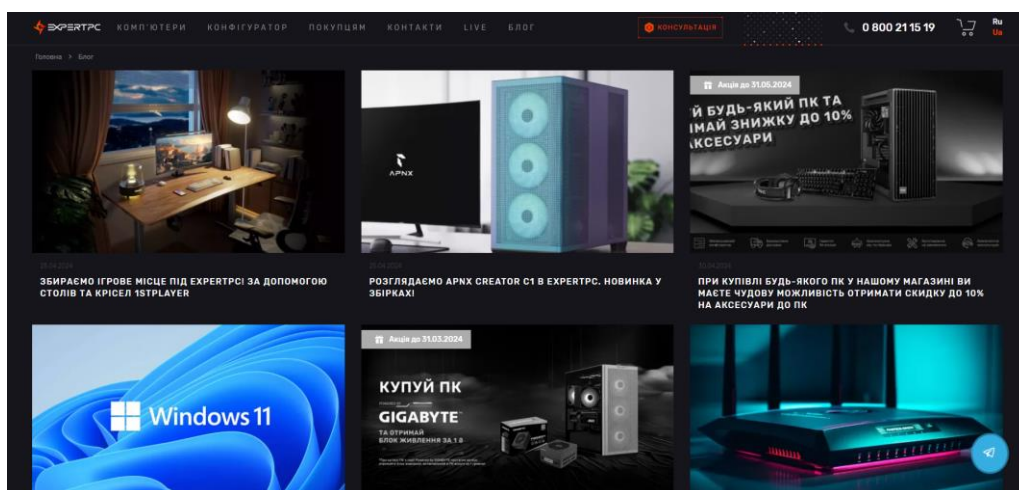


Рисунок 1.4 – Головна сторінка у веб-додатку «expertpc.ua»

Додаток має простий та зрозумілий інтерфейс, що дозволяє легко користуватися сайтом. Проте варто зауважити, що інтерфейс може здатися трохи застарілим у порівнянні з сучасними онлайн-магазинами. Додаток також надає блоги на різні новини та акції, що дозволяє користувачам бути в курсі останніх змін на ринку комп'ютерної техніки (рис. 1.5).



Останнім аналогом є український веб-додаток «gamingpc.com.ua»[2], який також призначений для продажу ПК (рис. 1.6). Основним недоліком сайту є обмежений асортимент доступних товарів. В порівнянні з конкурентами, сайт пропонує менший вибір комп'ютерної техніки та комплектуючих, що може обмежити можливості користувачів знайти ідеальну конфігурацію або необхідні компоненти для своїх потреб. Це може зменшити привабливість платформи для потенційних покупців, які шукають широкий вибір та різноманіття продуктів.



Рисунок 1.6 – Головна сторінка у веб-додатку «gamingpc.com.ua»

Додаток надає можливість придбати ПК з потрібної генерацією потужності за рахунок фільтру який відбувається через різні комп'ютерні ігри.

Для порівняння вище описаних веб-додатків було створено порівняльну таблицю (табл. 1.1).

Таблиця 1.1

Порівняння вебдодатків аналогів

Характеристика	brain.com.ua	expertpc.ua	gamingpc.com.ua
Адаптивність інтерфейсу	+	+	+
Дизайн	простий	Застарілий та привабливий	Сучасний
Навігаційне меню	+	+	+

Наявність блогу або статей	-	+	-
Додаткові товари окрім ПК	+	-	+
Відгуки клієнтів	+	+	+
Наявність інформаційного футеру	+	+	+

Аналізуючи (табл. 1.1), можна зробити висновок, що кожен з цих сервісів вирішує лише певний аспект завдань, які були поставлені перед розробленим додатком для збору та продажу ПК. Додаток «brain.com.ua»[1] виділяється простим дизайном та широким асортиментом товарів, що забезпечує користувачам можливість вибору серед різноманітних продуктів. Однак відсутність блогу або статей обмежує доступ до додаткової інформації та порад. «expertpc.ua»[3] має детальний блог та статті, що є великою перевагою для користувачів, які шукають додаткову інформацію. Проте застарілий дизайн може не відповідати очікуванням сучасних користувачів, а обмежений асортимент товарів може зменшити його привабливість. Додаток «gamingpc.com.ua»[2], пропонує сучасний дизайн та широкий асортимент товарів, що робить його привабливим для користувачів, які цінують естетику та різноманіття. Відсутність блогу або статей, однак, може обмежити корисність сайту для користувачів, які шукають додаткову інформацію.

Згідно з результатами порівняння аналогів, були виявлені всі переваги та недоліки, щоб правильно спроектувати веб-додаток для збору та продажу

ПК, який би поєднував зручний та сучасний дизайн, широкий асортимент товарів, а також наявність корисних інформаційних ресурсів для користувачів.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки програмного продукту було обрано наступні інструменти та технології.

React[4] – бібліотека для створення користувацьких інтерфейсів, яка дозволяє створювати багаторазово використовувані компоненти. Використання TypeScript у поєднанні з React забезпечує більш надійний та підтримуваний код завдяки статичній типізації.

Переваги React:

Компонентний підхід. Бібліотека надає можливість створювати багаторазово використовувані компоненти, що дозволяє спростити розробку.

Продуктивність. Завдяки віртуальному DOM, React забезпечує високу швидкість оновлення інтерфейсу.

Статична типізація. TypeScript додає типізацію до JavaScript коду, що дозволяє виявляти помилки на етапі компіляції, підвищуючи надійність коду.

Як сховище даних було вирішено використовувати базу даних MongoDB[8] з використанням ORM (Object-Relational Mapping) системи Prisma[9].

TypeScript - це мова програмування, яка є надмножиною JavaScript. Вона розроблена компанією Microsoft і позиціонується як більш розвинена версія JavaScript, що підтримує статичну типізацію. Основні особливості TypeScript:

Статична типізація. TypeScript дозволяє визначати типи для змінних, параметрів функцій, властивостей об'єктів та інших об'єктів. Це дозволяє виявляти помилки під час компіляції програми, що підвищує надійність та підтримку коду.

Класи і інтерфейси. TypeScript підтримує об'єктно-орієнтоване програмування з використанням класів і інтерфейсів, що дозволяє створювати структуровані та перевикористовувані компоненти програм.

Сумісність з JavaScript. TypeScript є надмножиною JavaScript, що означає, що весь дійсний JavaScript-код також є дійсним кодом TypeScript. Це дозволяє поступово впроваджувати TypeScript у вже існуючі JavaScript-проекти без потреби в переписуванні коду.

Інструменти для розробки. TypeScript поставляється з широким набором інструментів для підтримки розробки, таких як інтегровані середовища розробки (IDE), підтримка автоматичного доповнення коду, виявлення помилок під час розробки та інші.

Загалом, TypeScript дозволяє розробникам писати більш безпечний та структурований код, зменшуючи кількість помилок під час розробки та підвищуючи загальну продуктивність і підтримку програм.

MongoDB[8] – MongoDB є документ-орієнтованою, NoSQL базою даних, яка зберігає дані у вигляді документів у форматі BSON (Binary JSON)..

Основні особливості MongoDB:

Схема документа. MongoDB не вимагає строго визначення схеми даних. Кожен документ може мати різну структуру, що дозволяє зберігати дані різної природи в одній колекції.

Гнучкість. MongoDB підтримує динамічні запити та індекси, що дозволяє ефективно виконувати операції з даними без необхідності передумовляти схему бази даних.

Масштабованість. MongoDB підтримує горизонтальне масштабування, що дозволяє розширювати обсяги даних шляхом додавання нових серверів у кластер бази даних.

Підтримка операцій. MongoDB підтримує різноманітні операції з даними, включаючи запити CRUD (створення, читання, оновлення, видалення), агрегаційні функції, текстовий пошук та геопросторові запити.

Доступність. MongoDB забезпечує високу доступність і надійність за рахунок реплікації даних та автоматичного відновлення під час збоїв

Інтеграція. MongoDB має велику кількість бібліотек і драйверів для різних мов програмування, що спрощує розробку програм з використанням цієї бази даних

MongoDB є популярним вибором для сучасних додатків, що потребують гнучкості у зберіганні та операціях з даними, особливо коли необхідно працювати з великим обсягом невструктурованих даних.

Висновки до першого розділу

У першому розділі було проведено аналіз області дослідження та вимог до розробки програмного забезпечення. Це дозволило сформулювати основні завдання для створення вебдодатку, визначивши ключові функціональні можливості та етапи їх впровадження. Порівняння аналогів показало як переваги, так і недоліки, що сприяло правильному проектуванню додатку та визначенню необхідних функцій і особливостей для задоволення потреб користувачів. Результати дослідження інструментів дали змогу визначити оптимальний стек технологій для досягнення поставлених цілей.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-САЙТУ ДЛЯ ЗБОРУ ТА ПРОДАЖУ ПК

2.1 Визначення варіантів використання та структури системи

Для забезпечення неперервного виконання необхідних функцій і відповідності встановленим стандартам та специфікаціям, веб-сайт для збору та продажу ПК повинен задовольняти наступні високорівневі вимоги.

Бізнес вимоги

1. Основні цілі: проект створюється для забезпечення ефективного управління запасами комп'ютерної техніки та забезпечення точності і актуальності інвентарю. Метою є оптимізація процесів управління товарами та забезпечення достатньої наявності товарів для клієнтів
2. Представлення проекту: проект буде реалізовано у вигляді веб-платформи з інтуїтивно зрозумілим і привабливим інтерфейсом, що спростить процеси пошуку, вибору та покупки комп'ютерної техніки.

Вимоги користувачів

1. Можливість перегляду списку доступних комп'ютерних комплектуючих.
2. Можливість вибрати потрібну комплектуючу і переглянути її поточну наявність та ціну.
3. Можливість створити заявку на покупку комп'ютерних комплектуючих.
4. Можливість переглядати список створених покупок та їх статус.
5. Можливість залишати відгуки про отримані комплектуючі та переглядати відгуки інших користувачів.

Вимоги персоналу вебдодатку

1. Можливість додавати нові товари з автоматичним отриманням їх характеристик і зображень.

2. Можливість встановлювати ціни на товари та налаштовувати комісії за продаж.
3. Можливість відстежувати та управляти поточними замовленнями і продажами ПК.

Характеристика об'єкта комп'ютеризації

Розроблений веб-сайт для збору та продажу ПК надає користувачам можливість зручно та швидко переглядати доступні комп'ютерні системи, знайомитись з їх характеристиками та властивостями через інтуїтивний і зручний інтерфейс.

Функціональні вимоги

1. Вхід до системи: система має забезпечувати можливість реєстрації та входу користувачів.
2. Замовлення комп'ютерної техніки: користувач повинен мати можливість вибирати необхідні компоненти та пристрої, переглядати їх поточну ціну та вказувати необхідну інформацію для створення замовлення.
3. Управління замовленнями: Адміністратор має змогу переглядати та керувати поточними замовленнями.

Нефункціональні вимоги

1. Сприйняття - час, необхідний для освоєння звичайним користувачам – 10 хвилин, а для просунутих користувачів – 3-5 хвилин;
 - Час очікування відповіді на звичайні завдання не перевищує 5 секунд, а на складні завдання - не більше 10 секунд.
2. Надійність – час, витрачений на обслуговування системи, не повинен перевищувати 5% від загального часу її роботи.
 – середній час безвідмовної роботи становить 90 робочих днів.
 – Норма помилок та дефектів в роботі системи складає 3 помилки на десять тисяч рядків коду.

3. Продуктивність – система повинна забезпечувати можливість одночасної роботи мінімум з 100 користувачами, які взаємодіють з загальною базою даних одночасно.

Аналіз вимог користувачів дав підстави для визначення варіантів використання додатку для збору та продажу ПК. На (рис. 2.1) наведено діаграму використання програмної системи.

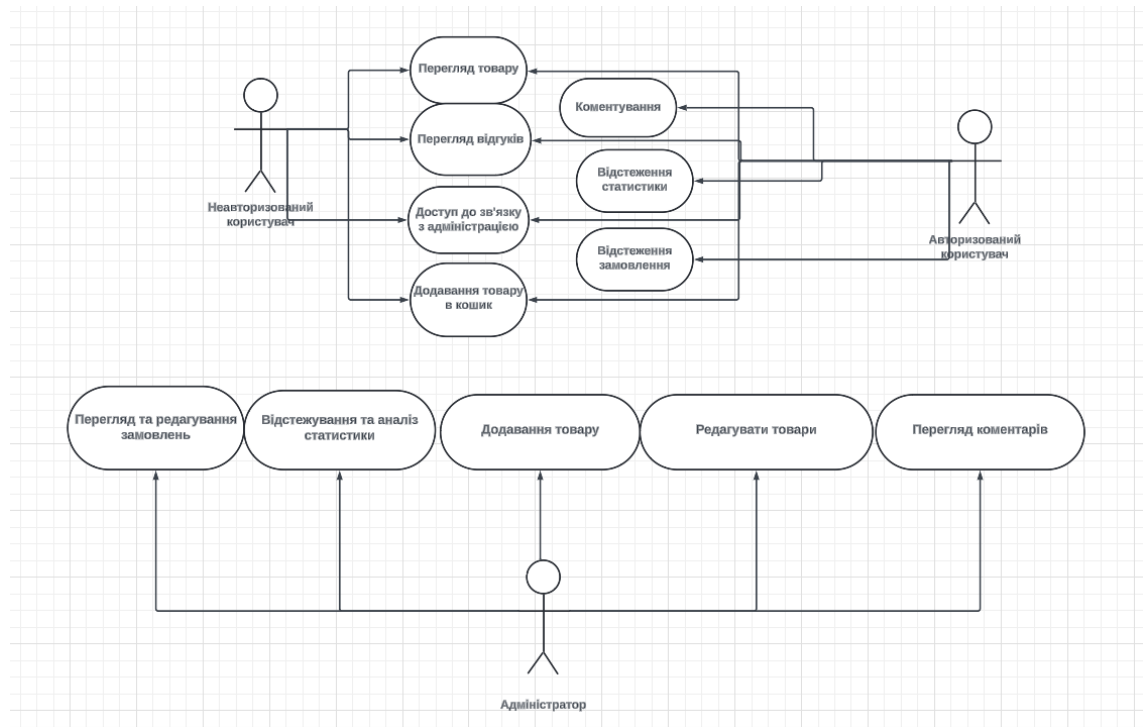


Рис. 2.1 Діаграма варіантів використання додатку

Ретельний аналіз функціональних вимог допоміг визначити основні сутності, необхідні для створення додатку. У результаті було створено діаграму сутностей (рис. 2.2), яка відображає моделі даних.

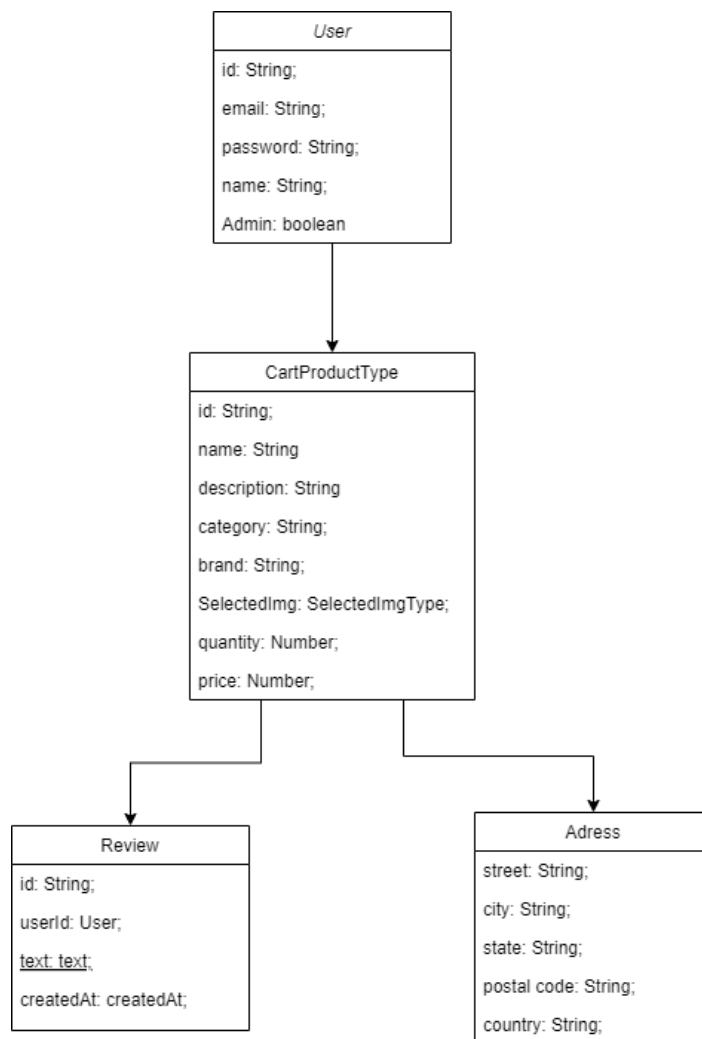


Рис. 2.2 Діаграма сутностей вебдодатку для збору та продажу ПК

Діаграма класів для вебсайту збору та продажу ПК включає дев'ять основних сутностей, кожна з яких відіграє важливу роль у функціонуванні системи та взаємодії з користувачами.

Клас "User" зберігає інформацію про користувача, включаючи унікальний ідентифікатор, ім'я, номер телефону, електронну пошту, пароль та роль (наприклад, адміністратор або клієнт). Цей клас пов'язаний з багатьма іншими сутностями, оскільки користувачі взаємодіють з різними аспектами системи.

Клас "CartProductType" зберігає інформацію про товар який обрав клієнт, включаючи унікальний ідентифікатор, назву, опис, категорію товару, бренд, зображення, наявність та ціну:

Клас "Reviews" зберігає відгуки користувачів про товари на сайті. Кожен відгук містить унікальний ідентифікатор, текст відгуку, оцінку, ідентифікатор товару, до якого він належить, ідентифікатор користувача, що залишив відгук, та дату створення:

Клас "Address" зберігає інформацію про адреси користувачів та доставок. Кожна адреса має унікальний ідентифікатор, ідентифікатор користувача, вулицю, місто, штат/область, поштовий індекс, країну та тип адреси (наприклад, домашня або робоча). Цей клас дозволяє зберігати кілька адрес для одного користувача та пов'язаний із замовленнями для забезпечення точної доставки.

2.2 Розробка бази даних системи

Для керування та зберігання даних використовується СУБД MongoDB[8], яка забезпечує високу продуктивність і масштабованість, а також дозволяє легко зберігати та отримувати дані.

База даних містить п'ять таблиць:

1. User – таблиця, яка представляє користувачів системи.
2. Order – таблиця, яка містить в собі інформацію про статус замовлень клієнтів.
3. Product – таблиця, що містить в собі інформацію про різні товари.
4. Review – таблиця, яка містить відгуки клієнтів.
5. Account – таблиця, яка містить користувачів які зареєструвалися через Google account.

Опис кожної таблиці наведений у таблицях 2.1 – 2.5.

Таблиця 2.1

Опис таблиці Users

Поле	Тип даних	Опис
_id	ObjectID	Ідентифікатор користувача.
name	String	Ім'я користувача
email	String	Електронна пошта користувача
hashedPassword	String	Пароль користувача
createdAt	ISODate	Дата коли було створено обліковий запис
updatedAt	ISODate	Дата коли обліковий запис було редаговано
role	String	Статус аккаунта

Таблиця 2.2

Опис таблиці Product

Поле	Тип даних	Опис
_id	string	Ідентифікація товару
name	string	Назва товару
description	string	Опис товару
price	number	Ціна за одиницю товару
brand	string	Бренд товару
category	string	Категорія до якої відноситься товар
inStock	boolean	Наявність товару
images	Array of strings	Зображення товару

Таблиця 2.3

Опис таблиці Order

Поле	Тип даних	Опис
_id	ObjectId	Ідентифікатор замовлення
userId	ObjectId	Ідентифікатор користувача, який зробив замовлення
amount	Number	Ціна замовлення
currency	String	Валюта замовлення
status	String	Стату замовлення
deliveryStatus	String	Стан доставки
createDate	ISODate	Дата створення замовлення
paymentIntentId	String	Ідентифікатор платіжної операції у Stipe
products	Array	Кількість товарів
address	Object	Адреса отримувача

Таблиця 2.4

Опис таблиці Review

Поле	Тип даних	Опис
id	string	Ідентифікатор відгуку
user	user	Користувач, який створив відгук
text	string	Текст відгуку
createdAt	datetime	Дата створення відгуку

Таблиця 2.5

Опис таблиці Account

Поле	Тип даних	Опис
_id	String	Унікальний ідентифікатор документа в базі даних MongoDB.
userId	String	Ідентифікатор користувача, з яким пов'язані дані

Продовження табл. 2.5

type	String	Тип аутентифікації
provider	String	Постачальник, який надає аутентифікаційні дані
providerAccountId	String	Ідентифікатор облікового запису, наданий постачальником
access_token	String	Токен доступу для аутентифікації користувача
expires_at	Number	Час завершення дії токена доступу, вказаний в Unix часі
token_type	String	Тип токена
scope	String	Права доступу, надані токену доступу
id_token	String	Токен ідентифікації, який містить інформацію про користувача з протоколу OAuth

З даних таблиць було сформовано діаграму бази даних (рис. 2.4):

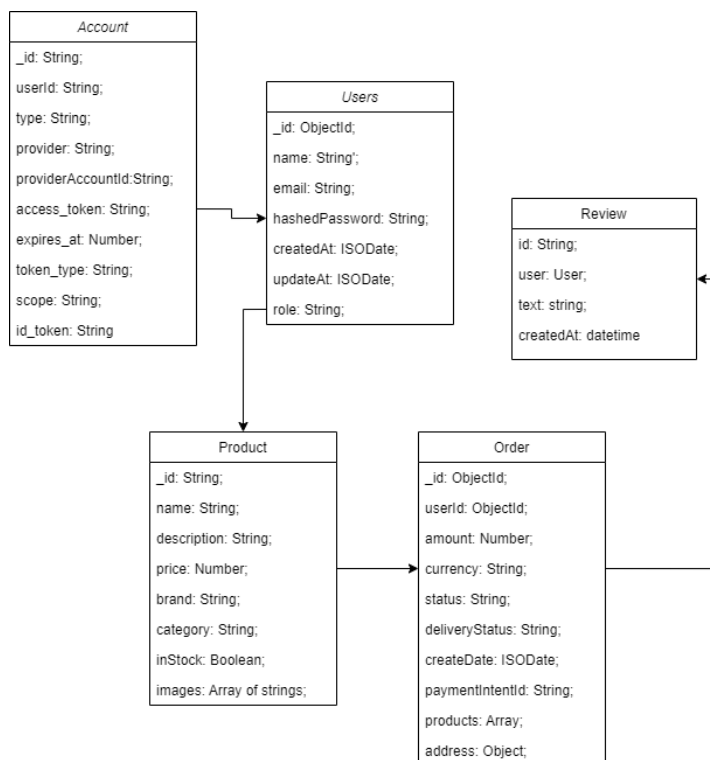


Рис. 2.4 Діаграма бази даних

2.3 Проектування та реалізація алгоритмів роботи системи

Вебдодаток дозволяє користувачам зареєструватись та увійти у свої облікові записи. Система надає можливість кожному зареєстрованому користувачу створювати заявки на покупку товару. Після заповнення відповідної форми, запит додається в базу даних, після чого адміністратор може керувати заявкою, приймаючи або відхиляючи її. Кількість зібраних компонентів та вартість розраховується на формі, а у випадку успішного продажу, вибрані товари відправляються на вказану користувачем адресу.

Для проектування та реалізації алгоритмів роботи веб-сайту для збору та продажу ПК, розглянемо основні етапи та функціональні можливості системи. Нижче наведено алгоритм роботи системи відповідно до діаграми діяльності (рис. 2.5).

Головна сторінка

1. Користувач потрапляє на головну сторінку сайту
2. Користувач може переглядати доступні товари

Реєстрація та авторизація

1. Якщо у користувача немає облікового запису, він реєструється, заповнюючи просту форму
2. Після успішної реєстрації користувач авторизується
3. Якщо авторизація не вдалася, користувач повертається до форми реєстрації

Створення замовлення

1. Після авторизації користувач може створювати замовлення, обираючи компоненти для свого майбутнього ПК
2. Оплачує замовлення

Отримання замовлення

1. Після оплати користувач отримує замовлення
2. Якщо користувач не авторизований, він може отримати замовлення

Залишити коментар

1. Якщо користувач зареєстрований, після отримання замовлення користувач може залишити коментар щодо товару або послуги

Профіль адміністратора

1. Перегляд та відслідковування замовлень
2. Відслідковування статистики
3. Додавання та редагування товарів
4. Перегляд коментарів користувачів

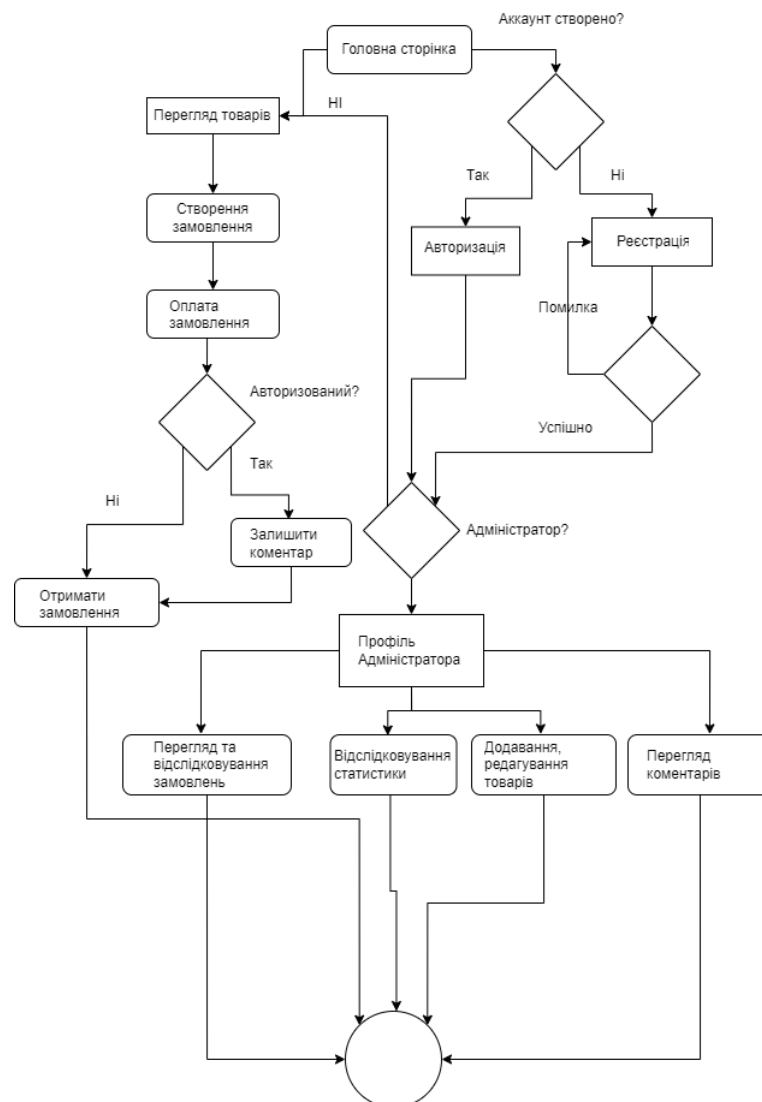


Рис. 2.5 Діаграма діяльності веб-сайту

2.4 Реалізація веб-сайту для збору та продажу ПК

Почнемо з реалізації модуля реєстрації в додатку. Спочатку створимо форму, де користувачі зможуть вводити всі необхідні дані для реєстрації.

```
const RegisterForm: React.FC<RegisterFormProps> = ({ currentUser }) => {
  const [isLoading, setIsLoading] = useState(false);
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm<FieldValues>({
    defaultValues: {
      name: "",
      email: "",
      password: "",
    },
  });

  const router = useRouter();

  useEffect(() => {
    if (currentUser) {
      router.push("/cart");
      router.refresh();
    }
  }, []);
```

Після реєстрації “useEffect”[17] Виконує перевірку, чи є currentUser. Якщо користувач авторизований (currentUser не null або undefined), перенаправляє його на сторінку /cart та оновлює сторінку.

```
const onSubmit: SubmitHandler<FieldValues> = (data) => {
  setIsLoading(true);

  axios
    .post("/api/register", data)
    .then(() => {
      toast.success("Обліковий запис створено");

      signIn("credentials", {
        email: data.email,
        password: data.password,
        redirect: false,
      }).then((callback) => {
```

```

    if (callback?.ok) {
      router.push("/cart");
      router.refresh();
      toast.success("Ви увійшли");
    }

    if (callback?.error) {
      toast.error(callback.error);
    }
  });
});
.catch(() => toast.error("Щось пішло не так"))
.finally(() => {
  setIsLoading(false);
});
});

```

Цей код доповнює попередній компонент форми реєстрації функцією `onSubmit`, яка обробляє подання форми. Коли користувач натискає кнопку для подання форми, викликається функція `onSubmit`. Дані форми відправляються на сервер за допомогою `axios.post`. Якщо реєстрація успішна, користувача інформують про успішне створення облікового запису і виконують автоматичний вхід, у випадку коли виникає помилка під час реєстрації або входу, користувача інформують про це.

```

<Heading title="Зареєструйтеся в Top-PC" />
<Button
  outline
  label="Продовжити через Google"
  icon={AiOutlineGoogle}
  onClick={() => {
    signIn("google");
  }}
/>
<hr className="bg-slate-300 w-full h-px" />
<Input
  id="name"
  label="Ім'я"
  disabled={isLoading}
  register={register}
  errors={errors}
  required
/>
<Input
  id="email"
  label="Email"
  disabled={isLoading}

```

```

    register={register}
    errors={errors}
    required
  />
  <Input
    id="password"
    label="Пароль"
    disabled={isLoading}
    register={register}
    errors={errors}
    required
    type="password"
  />
  <Button
    label={isLoading ? "Завантаження" : "Зареєструватися"}
    onClick={handleSubmit(onSubmit)}
  />
  <p className="text-sm">
    Вже є аккаунт?{" "}
    <Link className="underline" href="/login">
      Увійти
    </Link>
  </p>
</>
);
};

```

Цей код відповідає за відображення форми реєстрації в додатку. Він використовує компоненти, такі як Heading, Button, Input і Link, щоб створити інтерфейс реєстрації користувачів.

Також існує кнопка "Продовжити через Google", яка автоматично підтягує всю інформацію про клієнта з Google аккаунту та надсилає її на сервер, що значно прискорює процес реєстрації.

```

const CartClient: React.FC<CartClientProps> = ({ currentUser }) => {
  const { cartProducts, handleClearCart, cartTotalAmount } = useCart();

  const router = useRouter();

  if (!cartProducts || cartProducts.length === 0) {
    return (
      <div className="flex flex-col items-center">
        <div className="text-2xl">Ваш кошик порожній</div>
        <div>
          <Link
            href={"/"}
            className="
              text-slate-500 flex items-center gap-1
              mt-2

```

```

    "
  >
    <MdArrowBack />
    <span>Почати покупки</span>
  </Link>
</div>
</div>
);
}

```

```

return (
  <div>
    <Heading title="Кошик" center />
    <div
      className="grid
        grid-cols-5
        text-xs
        gap-4
        pb-2
        items-center
        mt-8
      "
    >

```

Цей код відповідає за відображення кошика покупок на веб-сайті для збору та продажу ПК. Він використовує компоненти та хуки, такі як Heading, Link, useCart, і useRouter, щоб створити інтерфейс кошика.

Компонент перевіряє, чи є товари в кошику (cartProducts). Якщо кошик порожній, користувач бачить повідомлення "Ваш кошик порожній" і кнопку для повернення до покупок, яка перенаправляє на головну сторінку. Це реалізовано за допомогою компонента Link, який веде на головну сторінку.

Якщо ж товари в кошику присутні, компонент відображає заголовок "Кошик" за допомогою компонента Heading, а також сітку для відображення товарів у кошику.

Сітка продуктів налаштована за допомогою класів Tailwind CSS, таких як grid, grid-cols-5, gap-4, pb-2, items-center, і mt-8, що дозволяє зручно розмістити товари в кошику та забезпечити гарний візуальний вигляд.

```

    <div className="col-span-2 justify-self-start">ТОВАР</div>
    <div className="justify-self-center">ЦІНА</div>
    <div className="justify-self-center">КІЛЬКІСТЬ</div>
    <div className="justify-self-end">ВСЬОГО</div>
  </div>

```

```

<div>
  {cartProducts &&
    cartProducts.map((item) => {
      return <ItemContent key={item.id} item={item} />;
    })}
</div>
<div className="border-t-[1.5px] border-slate-200 py-4 flex justify-between gap-4">
  <div className="w-[90px]">
    <Button
      label="Очистити кошик"
      onClick={() => {
        handleClearCart();
      }}
      small
      outline
    />
  </div>

```

Цей код створює інтерфейс корзини покупок у додатку. Він використовує ряд класів для налаштування вигляду елементів.

Код використовує клас "col-span-2", щоб розташувати перший елемент на дві колонки в сітці. Текст у всіх інших елементів вирівнюється за допомогою класів "justify-self-start", "justify-self-center" та "justify-self-end".

Після заголовків "ТОВАР", "ЦІНА", "КІЛЬКІСТЬ" і "ВСЬОГО" відображається список товарів, що знаходяться в корзині. Кожен товар представлений за допомогою компонента <ItemContent>, який отримує унікальний ключ "key" та дані про товар через властивість "item".

У нижній частині інтерфейсу знаходиться кнопка "Очистити кошик". При натисканні виконується функція handleClearCart(), яка очищає вміст кошика. Кнопка оформлена за допомогою компонента Button з атрибутами label="Очистити кошик", onClick для обробки натискання, small для визначення маленького розміру кнопки та outline для відображення контуру кнопки.

Цей інтерфейс корзини покупок призначений для зручного відображення та управління товарами, що дозволяє користувачам швидко переглядати, змінювати кількість або видаляти товари, а також легко очищати кошик за допомогою відповідної кнопки.

```

<div className="text-sm flex flex-col gap-1 items-start">
  <div className="flex justify-between w-full text-base font-semibold">
    <span>Проміжний підсумок</span>
    <span>{formatPrice(cartTotalAmount)}</span>
  </div>
  <p className="text-slate-500">
    Податки та вартість доставки розраховуються під час оформлення
    замовлення
  </p>
  <Button
    label={currentUser ? "Оплатити" : "Увійти в касу"}
    outline={currentUser ? false : true}
    onClick={() => {
      currentUser ? router.push("/checkout") : router.push("/login");
    }}
  />
  <Link
    href={"/"}
    className="
text-slate-500 flex items-center gap-1
mt-2
"
  >
    <MdArrowBack />
    <span>Продовжити покупки</span>
  </Link>
</div>
</div>
</div>
);
};

export default CartClient;

```

Цей фрагмент інтерфейсу корзини покупок забезпечує зручний доступ до інформації про покупки, можливість оплати або входу в систему для завершення замовлення, а також швидкий доступ до продовження покупок на головній сторінці.

```

export async function POST(request: Request){
  const currentUser = await getCurrentUser();

  if(!currentUser){
    return NextResponse.error();
  }

  const body = await request.json()
  const {comment, rating, product, userId} = body;

```

```

const deliveredOrder = currentUser?.orders.some(order => order.products.find(item => item.id === product.id) && order.deliveryStatus === 'delivered')

const userReview = product?.reviews.find((review: Review) =>{
  return review.userId === currentUser.id
}))

if(userReview || !deliveredOrder){
  return NextResponse.error()
}

const review = await prisma?.review.create({
  data: {
    comment,
    rating,
    productId: product.id,
    userId
  }
})

return NextResponse.json(review)
}

```

Ця функція POST призначена для обробки POST-запитів. Вона перевіряє, чи авторизований користувач і чи доставлене замовлення з продуктом. Якщо умови виконуються, вона додає новий відгук у базу даних і повертає його як JSON. В іншому випадку вона повертає помилку.

```

return (
  <>
    <Heading title="Додати продукт" center />
    <Input
      id="name"
      label="Назва"
      disabled={isLoading}
      register={register}
      errors={errors}
      required
    />
    <Input
      id="price"
      label="Ціна"
      disabled={isLoading}
      register={register}
      errors={errors}
      type="number"
      required
    />
    <Input

```

```

        id="brand"
        label="Бренд"
        disabled={isLoading}
        register={register}
        errors={errors}
        required
    />
    <TextArea
        id="description"
        label="Опис"
        disabled={isLoading}
        register={register}
        errors={errors}
        required
    />
    <CustomCheckBox
        id="inStock"
        register={register}
        label="Цей товар в наявності"
    />

```

Цей фрагмент коду створює форму для введення даних нового продукту у додатку. Вона включає поля для назви, ціни, бренду, опису та вибору наявності товару. Кожне поле має відповідні мітки та можливість валідації.

```

<div className="w-full font-medium">
  <div className="mb-2 font-semibold">Оберіть категорію</div>
  <div className="grid grid-cols-2 md:grid-cols-3 gap-3 max-h[50vh] overflow-y-auto">
    {categories.map((item) => {
      if (item.label === "All") {
        return null;
      }

      return (
        <div key={item.label} className="col-span">
          <CategoryInput
            onClick={(category) => setCustomValue("category", category)}
            selected={category === item.label}
            label={item.label}
            icon={item.icon}
          />
        </div>
      );
    })}
  </div>
</div>

```

Цей блок коду відображає список категорій у вигляді сітки з можливістю прокрутки. Кожна категорія представлена компонентом CategoryInput, який дозволяє вибрати категорію за допомогою клікабельного елемента з міткою та іконкою.

```
<div className="w-full flex flex-col flex-wrap gap-4">
  <div>
    <div className="font-bold">
      Виберіть доступні кольори продукту та завантажте їх зображення.
    </div>
    <div className="text-sm">
      Ви повинні завантажити зображення для кожного вибраного кольору,
      інакше ваш вибір кольору буде проігноровано.
    </div>
  </div>
  <div className="grid grid-cols-2 gap-3">
    {colors.map((item, index) => {
      return (
        <SelectColor
          key={index}
          item={item}
          addImageToState={addImageToState}
          removeImageFromState={removeImageFromState}
          isProductCreated={isProductCreated}
        />
      );
    })}
  </div>
</div>
<Button
  label={isLoading ? "Завантаження..." : "Додати продукт"}
  onClick={handleSubmit(onSubmit)}
/>
</>
);
};
```

Наведений приклад коду відображення форми додавання продукту. Input, TextArea, CustomCheckBox: Ці компоненти використовуються для збору введених користувачем даних. Вони передають атрибути id, label, disabled, register, errors, required для обробки форми. SelectColor відображає список доступних кольорів для продукту, дозволяючи користувачу вибрати кольори та завантажити зображення для кожного кольору. Компонент AddProductForm збирає дані про назву, ціну, бренд, опис, наявність, категорію та кольори

продукту від користувача, використовуючи керувані компоненти для введення даних. Після заповнення форми користувачем та натискання кнопки "Додати продукт", дані відправляються на сервер для подальшого оброблення.

```
const OrdersClient: React.FC<OrdersClientProps> = ({ orders }) => {
  const router = useRouter();
  let rows: any = [];

  if (orders) {
    rows = orders.map((order) => {
      return {
        id: order.id,
        customer: order.user.name,
        amount: formatPrice(order.amount / 100),
        paymentStatus: order.status,
        date: moment(order.createDate).fromNow(),
        deliveryStatus: order.deliveryStatus,
      };
    });
  }
}
```

Компонент OrdersClient відповідає за відображення списку замовлень у вигляді таблиці. Він отримує список замовлень через пропси, мапить їх у масив rows, де кожен елемент містить дані про замовлення, які відображаються у таблиці з колонками.

Кожна колонка має своє визначене поле (field), заголовок (headerName), ширину (width) та можливість відображення спеціального вмісту через renderCell, який у випадку колонки "Ціна(UAH)" стилізує вміст шрифтом жирного шрифту та кольором тексту slate-800.

```
const columns: GridColDef[] = [
  { field: "id", headerName: "ID", width: 220 },
  { field: "customer", headerName: "Ім'я клієнта", width: 130 },
  {
    field: "amount",
    headerName: "Ціна(UAH)",
    width: 130,
    renderCell: (params) => {
      return (
        <div className="font-bold text-slate-800">{params.row.amount}</div>
      );
    },
  },
  {
    field: "paymentStatus",
    headerName: "Статус платежу",
  },
];
```

```

width: 130,
renderCell: (params) => {
  return (
    <div>
      {params.row.paymentStatus === "pending" ? (
        <Status
          text="в очікуванні"
          icon={MdAccessTimeFilled}
          bg="bg-slate-200"
          color="text-slate-700"
        />
      ) : params.row.paymentStatus === "complete" ? (
        <Status
          text="завершено"
          icon={MdDone}
          bg="bg-green-200"
          color="text-green-700"
        />
      ) : (
        <></>
      )}
    </div>
  );
},
},

```

Цей масив `columns` налаштовує вигляд колонок для таблиці замовлень. Він включає поля для ID замовлення, імені клієнта, ціни в UAH та статусу платежу, з функціями для відображення спеціалізованого вмісту.

```

{
  field: "deliveryStatus",
  headerName: "Статус доставки",
  width: 130,
  renderCell: (params) => {
    return (
      <div>
        {params.row.deliveryStatus === "pending" ? (
          <Status
            text="в очікуванні"
            icon={MdAccessTimeFilled}
            bg="bg-slate-200"
            color="text-slate-700"
          />
        ) : params.row.deliveryStatus === "dispatched" ? (
          <Status
            text="відправлено"
            icon={MdDeliveryDining}
            bg="bg-purple-200"
            color="text-purple-700"
          />
        ) : (
          <></>
        )}
      </div>
    );
  },
}

```

```

        />
      ) : params.row.deliveryStatus === "delivered" ? (
        <Status
          text="доставлено"
          icon={MdDone}
          bg="bg-green-200"
          color="text-green-700"
        />
      ) : (
        <></>
      )}
    </div>
  );
},
{
  field: "date",
  headerName: "Дата",
  width: 130,
},
];

```

Цей масив `columns` конфігурує вигляд колонок для таблиці замовлень. Він включає колонку "Статус доставки", яка використовується для відображення текстового статусу доставки залежно від значення `deliveryStatus` в кожному рядку замовлення. Також є колонка "Дата", що показує дату створення кожного замовлення.

```

{
  field: "action",
  headerName: "Дії",
  width: 200,
  renderCell: (params) => {
    return (
      <div className="flex justify-between gap-4 w-full">
        <ActionBtn
          icon={MdRemoveRedEye}
          onClick={() => {
            router.push(`/order/${params.row.id}`);
          }}
        />
      </div>
    );
  },
},
];

return (
  <div className="max-w-[1150px] m-auto text-xl">
    <div className="mb-4 mt-8">

```

```

        <Heading title="Замовлення" center />
    </div>
    <div style={{ height: 600, width: "100%" }}>
        <DataGrid
            rows={rows}
            columns={columns}
            initialState={{
                pagination: {
                    paginationModel: { page: 0, pageSize: 9 },
                },
            }}
            pageSizeOptions={[9, 20]}
            checkboxSelection
            disableRowSelectionOnClick
        />
    </div>
</div>
);
};

```

Таблиця DataGrid відображає рядки з даними замовлень, дозволяє вибір рядків та навігацію до деталей кожного замовлення за допомогою кнопки "Дії". Визначається змінна rows, яка буде масивом об'єктів, що містять дані кожного замовлення, якщо orders не є null або undefined. Кожне замовлення мапиться на об'єкт з властивостями id, customer, amount, paymentStatus, date, deliveryStatus. Визначається масив columns для налаштування кожної колонки таблиці замовлень. Компонент повертає JSX, що містить заголовок, таблицю DataGrid з визначеними rows та columns, які відображаються на сторінці.

Висновки до другого розділу

Проектування та реалізація веб-сайту для збору та продажу персональних комп'ютерів орієнтувалася на створення ефективної системи обміну цифрових активів за принципом SWAP. Основні цілі включали забезпечення швидкого та безпечного обміну активів, автоматизацію цього процесу та створення інтуїтивно зрозумілого інтерфейсу для користувачів.

Проект включав аналіз функціональних вимог, розробку алгоритмів для створення активів, обробки заявок на обмін, виконання переказів та оновлення статусів заявок. Були розглянуті всі необхідні функції системи для забезпечення безперебійної роботи та відповідності стандартам.

Для полегшення взаємодії з системою було створено зручний веб-інтерфейс, використовуючи сучасні дизайнерські підходи. Була спроектована структура бази даних для ефективного зберігання і обробки даних про заявки на обмін, користувачів та активи, що була візуалізована через діаграму бази даних.

Велика увага була приділена аспектам безпеки, зокрема захисту персональних даних користувачів та запобіганню зловживанням. Усі ці заходи спільно допомогли створити надійний та функціонально розширюваний вебдодаток, який задовольняє потреби користувачів у зручній покупці персональних комп'ютерів через інтуїтивно зрозумілий і безпечний інтерфейс.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБ-САЙТУ ДЛЯ ЗБОРУ ТА ПРОДАЖУ ПК

3.1 Порядок встановлення та налаштування параметрів системи

Встановлення та налаштування параметрів веб-сайту для збору та продажу персональних комп'ютерів включає роботу з різними складовими, кожна з яких потребує окремого розгортання та налаштування. Для ілюстрації взаємозв'язків між цими компонентами створена діаграма розгортання, яка допомагає чітко відобразити структуру системи та процеси взаємодії між ними (рис. 3.1).

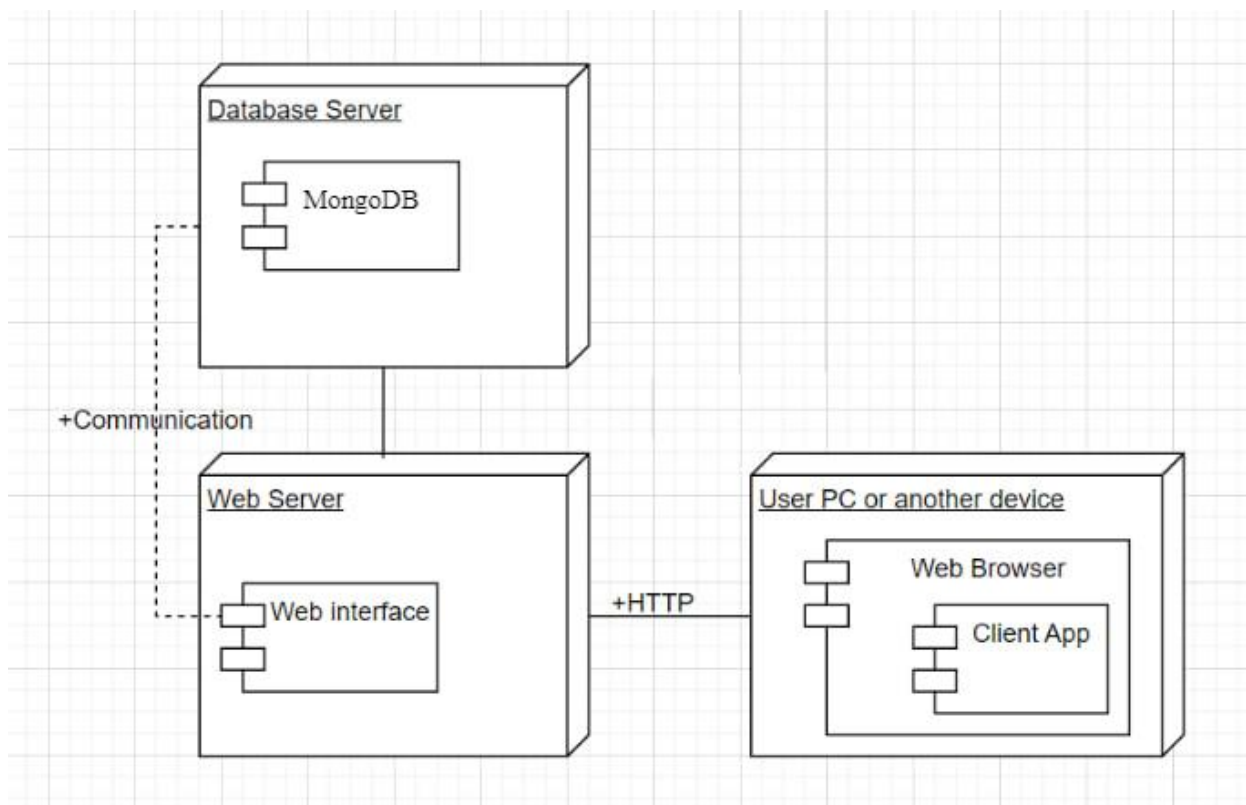


Рис. 3.1 Діаграма розгортання вебдодатку з обміну цифрових активів

Ця діаграма розгортання показує систему, що складається з трьох основних вузлів: вебсервера, сервера баз даних і клієнтського пристрою. Клієнтський пристрій взаємодіє з вебсервером за допомогою протоколу HTTP, що забезпечує обмін даними у форматі запит-відповідь на основі стандартів

TCP, забезпечуючи надійну передачу даних без втрат. Взаємодія між вебсервером і сервером баз даних здійснюється через протокол TCP/IP, що забезпечує стабільний та безперервний обмін інформацією.

3.2 Структура інтерфейсу веб-додатку та порядок роботи

Користувач, який відкриває веб-додаток, миттєво переходить на головну сторінку, де відразу бачить каталог товарів та доступні елементи навігації (рис. 3.2).

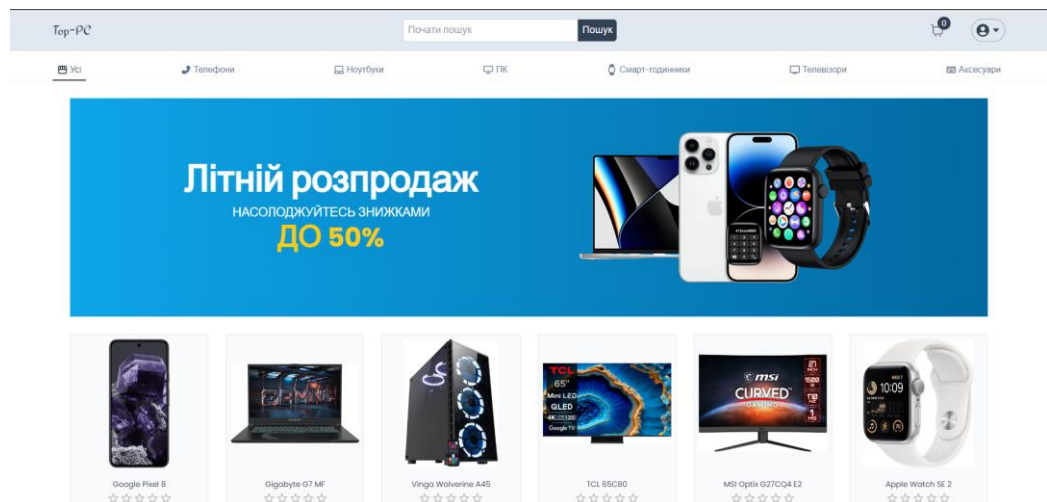


Рис. 3.2 Головна сторінка

Клієнт може проскролити нижче то побачити різноманітну інформацію про сайт та команду. (рис. 3.3).

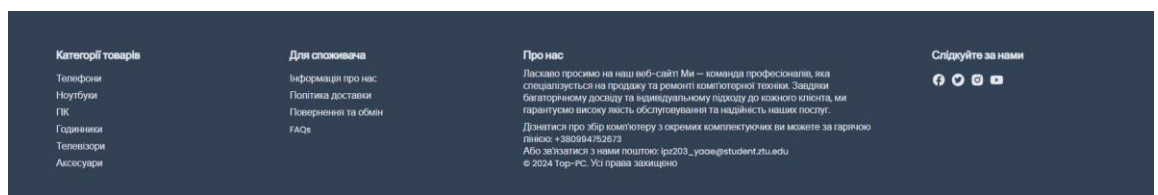


Рис. 3.3 Додаткова інформація на головній сторінці

Розглянемо який інтерфейс бачить користувач коли хоче обрати товар. При натисканні клієнт бачить інформацію про товар, його категорію, назву, колір, чи є він на складі та його ціну(рис. 3.4).

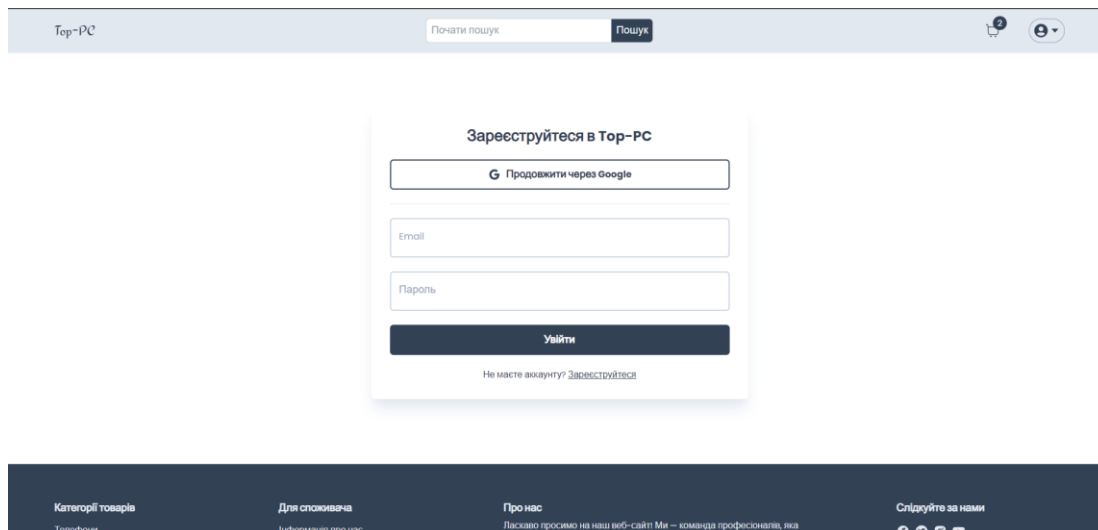


Рис. 3.6 Відображення інтерфейсу реєстрації

Зареєструємо клієнта в додатку, для цього потрібно перейти на клікабельний напис “Зареєструйтеся” (рис. 3.7).

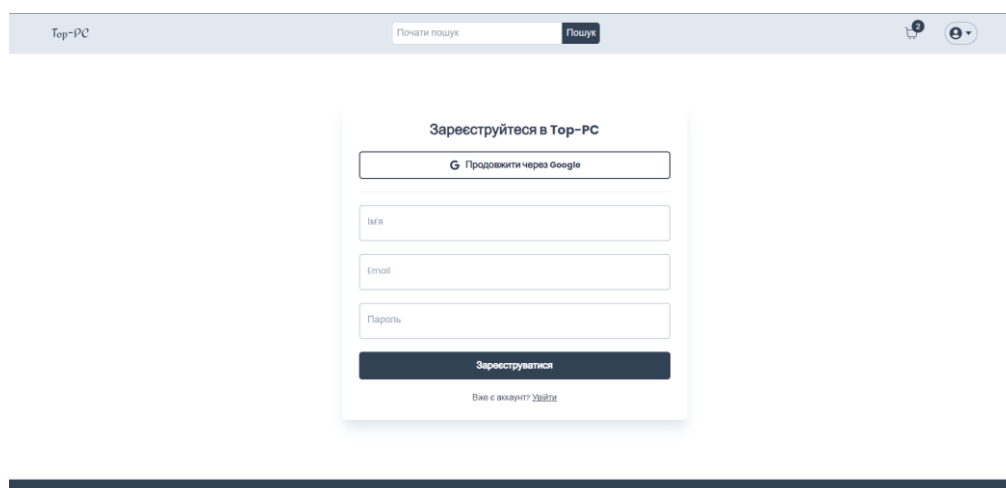


Рис. 3.7 Сторінка реєстрації

У випадку заповненої форми буде відображатися значок успіху (рис. 3.9).

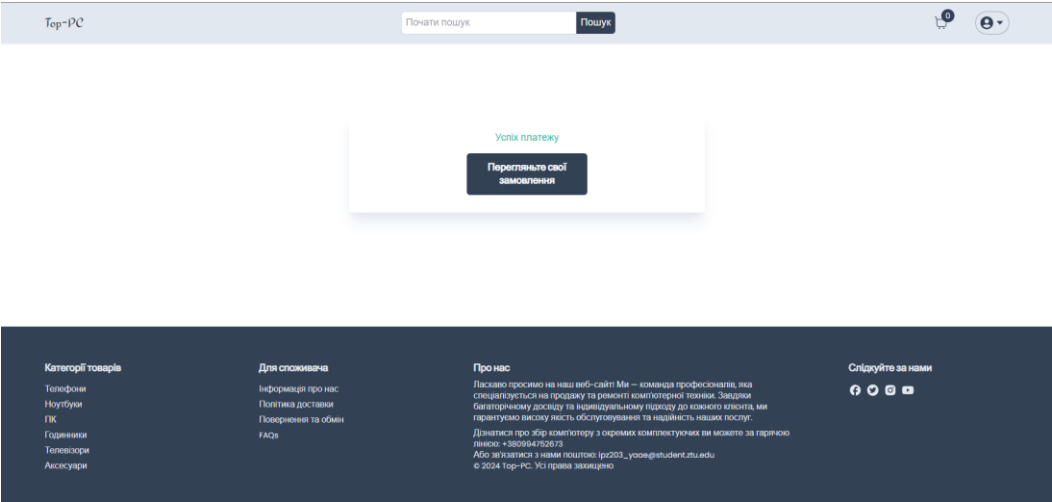


Рис. 3.12 вікно з посиланням на замовлення

Після оплати клієнт може переглядати статус свого замовлення(рис. 3.13)

Замовлення

☐	ID	Ім'я клієнта	Ціна(UAH)	Статус платежу	Статус доставки	Дата	Дії
☐	66740fc340c794ab899d3d6	Антон	30 000,00 грн	в очікуванні	в очікуванні	3 minutes ago	

Рис. 3.13 вікно з посиланням на замовлення

При авторизації адміністратора в додатку, йому відкривається доступ до всіх сторінок додатку, а також сторінки пов'язані з адмініструванням(рис. 3.14).

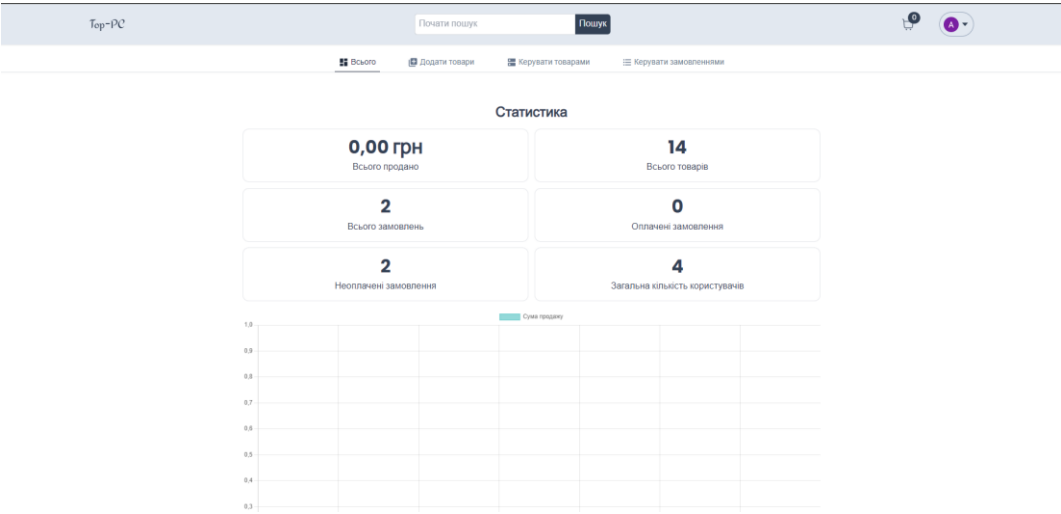


Рис. 3.14 Сторінка адміністратора

Сторінка «Додати товар» містить в собі поля з інформацією про товар, його категорію та колір (рис. 3.15) та (рис. 3.16).

Рис. 3.15 Сторінка Додавання нового товару

Рис. 3.16 Сторінка Додавання нового товару

“Керування товарами” містить таблиці з інформаціями про товар який додано в додаток (рис. 3.16), в товар можна внести зміни або видалити його.

☐	ID	Назва	Ціна(UAH)	Категорія	Бренд	в наявності	Дії
☐	667400d940cf794ab899d3c4	Google Pixel 8	25 000,00 грн	Телефони	Google	в наявності	
☐	667401a240cf794ab899d3c5	Apple iPhone 15 Pro Max 2	50 000,00 грн	Телефони	Apple	в наявності	
☐	6674058140cf794ab899d3c6	Gigabyte G7 MF	40 000,00 грн	Ноутбуки	Gigabyte	в наявності	
☐	6674069540cf794ab899d3c8	LOQ 15APH8	40 000,00 грн	Ноутбуки	Lenovo	в наявності	
☐	6674073140cf794ab899d3c9	Artline Gaming X47	27,00 грн	ПК	Artline	немає в наявноі	
☐	667407a440cf794ab899d3ca	Vinga Wolverine A45	41 000,00 грн	ПК	Vinga	в наявності	
☐	6674087440cf794ab899d3cb	Apple Watch SE 2	8 000,00 грн	Смарт-го...	Apple	в наявності	
☐	667408f240cf794ab899d3cc	TCL 65C80	50 000,00 грн	Телевізори	TCL	в наявності	
☐	6674097c40cf794ab899d3cd	Sapphire Radeon RX 6650 X...	16 400,00 грн	Аксесуари	Sapphire	немає в наявноі	

Розглянемо процес керування замовлень (рис. 3.17). Адміністратор може міняти статус замовлення з недоставленого на доставлений (рис. 3.18) та відправлено (рис. 3.19).

Рис. 3.17 Сторінка керування замовлень

Рис. 3.18 Зміна статусу на доставлено

Рис. 3.19 Зміна статусу на відправлено

3.3 Тестування роботи вебдодатку

Тестування програмного забезпечення є важливою частиною процесу розробки веб-сайту для збору та продажу ПК. Цей процес спрямований на виявлення помилок і дефектів, що дозволяє покращити якість продукту перед його випуском на ринок. Основні цілі тестування включають забезпечення надійності, функціональності та безпеки додатку, щоб користувачі могли максимально ефективно використовувати його можливості.

Для тестування функціоналу збору та продажу комп'ютерних конфігурацій використовувалися методи "причина-наслідок" та "аналіз граничних значень". Користувачі проводили тестування, вибираючи різні варіанти комплектацій ПК та змінюючи кількість та параметри компонентів. Очікувані результати полягали у правильному отриманні та відображенні деталей конфігурацій, що автоматично оновлювалися при зміні вибраної комплектації.

Метод "передбачення помилок" допоміг ідентифікувати та виправити проблеми взаємодії користувачів з інтерфейсом, а також перевірити, як система реагує на неправильні або відсутні дані.

Основні етапи тестування:

Функціональне тестування

1. Тестування реєстрації та авторизації: Перевірка можливості створення нового облікового запису, успішної авторизації та обробки помилок при введенні неправильних даних
2. Тестування процесу створення замовлення: Включає перевірку вибору компонентів, додавання їх до кошика, оформлення замовлення та здійснення оплати. Кожен етап перевірявся на коректність обробки даних та відображення відповідних повідомлень користувачеві.
3. Тестування функціоналу адміністрування: Адміністратори перевіряли можливість додавання, редагування та видалення товарів, перегляду коментарів та управління замовленнями

Тестування інтерфейсу

1. Перевірка зручності користування: Користувачі оцінювали, наскільки легко орієнтуватися на сайті, знаходити необхідну інформацію та виконувати дії (створення замовлення, перегляд товарів, залишення відгуків тощо)
2. Тестування на різних пристроях: Веб-сайт перевірявся на різних пристроях (комп'ютери, планшети, смартфони) для забезпечення коректного відображення та функціонування на кожному з них

Тестування безпеки

1. Перевірка автентифікації та авторизації: Забезпечення надійного захисту даних користувачів при вході до системи, зберігання облікових даних та використання шифрування
2. Перевірка наявності вразливостей: Виконання тестів на SQL-ін'єкції, XSS-атаки та інші можливі вразливості, які можуть поставити під загрозу безпеку системи та даних користувачів

Навантажувальне тестування

1. Перевірка продуктивності: Визначення, як система справляється з великою кількістю одночасних запитів та користувачів. Перевірка часу відгуку, стабільності роботи при пікових навантаженнях та ефективності використання ресурсів сервера
2. Стрес-тестування: Імітація екстремальних умов для визначення, наскільки стійка система до надзвичайних навантажень і як вона відновлюється після збоїв.

Регресійне тестування.

1. Перевірка змін та оновлень: Після внесення змін або додавання нових функцій до веб-додатку проводиться тестування для впевненості, що ці зміни не вплинули на вже існуючий функціонал. Кожна нова версія додатку проходить ретельну перевірку, щоб переконатися у відсутності нових помилок.

Завдяки ретельному тестуванню вдалося значно підвищити надійність, безпеку та зручність користування веб-додатком. Це забезпечило високий рівень задоволеності користувачів та готовність системи до запуску на ринок.

Тестування є безперервним процесом, що триває і після запуску продукту. Збір зворотного зв'язку від користувачів та моніторинг роботи системи дозволяють вчасно виявляти та усувати нові проблеми, забезпечуючи постійне вдосконалення та розвиток веб-додатку.

Висновки до третього розділу

У цьому розділі детально розглянуто процес встановлення та налаштування параметрів системи для веб-сайту збору та продажу персональних комп'ютерів. Описано послідовність розгортання системи та взаємодію її компонентів, що сприяло чіткому визначенню архітектури рішення. Для ілюстрації взаємодії компонентів використано діаграму розгортання, яка демонструє, як різні частини системи взаємодіють між собою для забезпечення ефективної роботи веб-додатку.

Також детально описано структуру інтерфейсу веб-додатку. Розглянуто основні функції користувачів, такі як реєстрація та створення замовлень. Окрему увагу приділено можливостям адміністратора, які включають додавання, редагування та видалення товарів, а також управління замовлень користувачів.

Також висвітлено процес тестування веб-додатку, наведено конкретні приклади використання методів тестування, що допомогли виявити та виправити потенційні проблеми функціональності та взаємодії користувачів з системою.

ВИСНОВКИ

Під час виконання дипломної роботи було розроблено та реалізовано веб-сайт для збору та продажу персональних комп'ютерів. На початкових етапах проведено детальний аналіз потреб користувачів та існуючих ринкових рішень, що сприяло формулюванню основних вимог до розроблюваного веб-додатку.

Проектування веб-додатку включало в себе аналіз варіантів використання та розробку всіх можливих сценаріїв взаємодії користувачів з системою. Створена структура бази даних була спроектована для ефективного зберігання та обробки інформації користувачів, їх замовлення та товари на сайті.

Основні функції користувачів, включаючи реєстрацію, створення замовлень та додавання відгуків, а також можливості адміністратора, які включають управління замовленнями, були реалізовані з урахуванням зручності та інтуїтивності інтерфейсу. Користувачі мають можливість легко та ефективно купляти комп'ютери та комплектуючі частини без зайвих складнощів.

У процесі розробки були застосовані методи тестування, такі як "причина-наслідок", "аналіз граничних значень" та "передбачення помилок", що сприяло виявленню і виправленню помилок під час розробки та поліпшенню загальної якості веб-додатку.

Отже, веб-сайт для збору та продажу персональних комп'ютерів є ефективним інструментом для користувачів, який забезпечує зручну та надійну платформу для придбання комп'ютерної техніки

СПИСОК ЛІТЕРАТУРИ

1. Brain [Електронний ресурс]. Режим доступу до ресурсу:
<https://brain.com.ua/ukr/>
2. Gamingpc [Електронний ресурс]. Режим доступу до ресурсу:
<https://gamingpc.com.ua/katalog>
3. Expertpc [Електронний ресурс]. Режим доступу до ресурсу:
<https://expertpc.ua/>
4. React.js [Електронний ресурс]. Режим доступу до ресурсу:
<https://uk.reactjs.org/docs/getting-started.html>
5. React hot toast [Електронний ресурс]. Режим доступу до ресурсу:
<https://react-hot-toast.com/>
6. Firebase [Електронний ресурс]. Режим доступу до ресурсу:
<https://firebase.google.com/>
7. CSS [Електронний ресурс]. Режим доступу до ресурсу:
<https://devdocs.io/css/>
8. MongoDB [Електронний ресурс]. Режим доступу до ресурсу:
<https://cloud.mongodb.com/>
9. Prisma [Електронний ресурс]. Режим доступу до ресурсу:
<https://www.prisma.io/docs/orm/overview/databases/sqlite>
10. react-router-dom [Електронний ресурс]. Режим доступу до ресурсу:
<https://reactrouter.com/en/main>
11. Stripe [Електронний ресурс]. Режим доступу до ресурсу:
<https://stripe.com/>
12. Moment.js [Електронний ресурс]. Режим доступу до ресурсу:
<https://momentjs.com/docs/>
13. Redux Toolkit [Електронний ресурс]. Режим доступу до ресурсу:
<https://redux-toolkit.js.org/>
14. React verifies [Електронний ресурс]. Режим доступу до ресурсу:
<https://react.dev/learn/lifecycle-of-reactive-effects#how-react-verifies-that-your-effect-can-re-synchronize>

15. React Redux [Электронный ресурс]. Режим доступа до ресурсу:
<https://react-redux.js.org/>

16. React useState [Электронный ресурс]. Режим доступа до ресурсу:
<https://react.dev/reference/react/useState>

17. React useEffect [Электронный ресурс]. Режим доступа до ресурсу:
<https://react.dev/reference/react/useEffect>

18. Types of testing [Электронный ресурс]. Режим доступа до ресурсу:
<https://training.gatestlab.com/blog/technical-articles/review-the-types-of-testing/>

19. Draw io [Электронный ресурс]. Режим доступа до ресурсу:
<https://app.diagrams.net/>

20. Lucid [Электронный ресурс]. Режим доступа до ресурсу:
<https://lucid.app/>

ДОДАТКИ

					ІІЗ.КР.Б-121-24-ІІЗ	Арк.
						57

Додаток Footer.tsx

```
import Link from "next/link";
import Container from "../Container";
import FooterList from "../FooterList";
import { MdFacebook } from "react-icons/md";
import {
  AiFillTwitterCircle,
  AiFillInstagram,
  AiFillYoutube,
} from "react-icons/ai";

const Footer = () => {
  return (
    <footer className="bg-slate-700 text-slate-200 text-sm mt-16">
      <Container>
        <div className="flex flex-col md:flex-row justify-between pt-16 pb-8">
          <FooterList>
            <h3 className="text-base font-bold mb-2">Категорії товарів</h3>
            <Link href="http://localhost:3000/?category=%D0%A2%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD%D0%B8">
              Телефони
            </Link>
            <Link href="http://localhost:3000/?category=%D0%9D%D0%BE%D1%83%D1%82%D0%B1%D1%83%D0%BA%D0%B8">
              Ноутбуки
            </Link>
            <Link href="http://localhost:3000/?category=%D0%9F%D0%9A">ПК</Link>
            <Link href="http://localhost:3000/?category=%D0%A1%D0%BC%D0%B0%D1%80%D1%82-%D0%B3%D0%BE%D0%B4%D0%B8%D0%BD%D0%BD%D0%B8%D0%BA%D0%B8">
              Годинники
            </Link>
            <Link href="http://localhost:3000/?category=%D0%A2%D0%B5%D0%BB%D0%B5%D0%B2%D1%96%D0%B7%D0%BE%D1%80%D0%B8">
              Телевізори
            </Link>
            <Link href="http://localhost:3000/?category=%D0%90%D0%BA%D1%81%D0%B5%D1%81%D1%83%D0%B0%D1%80%D0%B8">
              Аксесуари
            </Link>
          </FooterList>
          <FooterList>
            <h3 className="text-base font-bold mb-2">Для споживача</h3>
            <Link href="https://ztu.edu.ua/">Інформація про нас</Link>
            <Link href="https://ztu.edu.ua/">Політика доставки</Link>
            <Link href="https://ztu.edu.ua/">Повернення та обмін</Link>
            <Link href="https://ztu.edu.ua/">FAQs</Link>
          </FooterList>
          <div className="w-full md:w-1/3 mb-6 md:mb-0">
            <h3 className="text-base font-bold mb-2">Про нас</h3>

```

```

    <p className="mb-2">
      Ласкаво просимо на наш веб-сайт! Ми – команда професіоналів, яка
      спеціалізується на продажу та ремонті комп'ютерної техніки.
      Завдяки багаторічному досвіду та індивідуальному підходу до
      кожного клієнта, ми гарантуємо високу якість обслуговування та
      надійність наших послуг.
    </p>
    <p>
      Дізнатися про збір комп'ютеру з окремих комплектуючих ви можете за
      гарячою лінією: +380994752673
    </p>
    <p>Або зв'язатися з нами поштою: ipz203_yaae@student.ztu.edu</p>
    <p>&copy; {new Date().getFullYear()} Top-PC. Усі права захищено</p>
  </div>
  <FooterList>
    <h3 className="text-base font-bold mb-2">Слідкуйте за нами</h3>
    <div className="flex gap-2">
      <Link href="https://www.facebook.com/ztueduua/">
        <MdFacebook size={24} />
      </Link>
      <Link href="https://x.com/ztueduua">
        <AiFillTwitterCircle size={24} />
      </Link>
      <Link href="https://www.instagram.com/ztueduua/">
        <AiFillInstagram size={24} />
      </Link>
      <Link href="https://www.youtube.com/channel/UCXcSq2MLmApgKFdf-
bOzFNqw">
        <AiFillYoutube size={24} />
      </Link>
    </div>
  </FooterList>
</div>
</Container>
</footer>
);
};

export default Footer;

```

Додаток ManageOrders

```

"use client";

import { Order, User } from "@prisma/client";
import { DataGrid, GridColDef } from "@mui/x-data-grid";
import { formatPrice } from "@utils/formatPrice";
import Heading from "@app/components/Heading";

```

```

import Status from "@app/components/Status";
import {
  MdAccessTimeFilled,
  MdDeliveryDining,
  MdDone,
  MdRemoveRedEye,
} from "react-icons/md";
import ActionBtn from "@app/components/ActionBtn";
import { useCallback } from "react";
import axios from "axios";
import toast from "react-hot-toast";
import { useRouter } from "next/navigation";
import moment from "moment";

interface ManageOrdersClientProps {
  orders: ExtendedOrder[];
}

type ExtendedOrder = Order & {
  user: User;
};

const ManageOrdersClient: React.FC<ManageOrdersClientProps> = ({ orders }) => {
  const router = useRouter();
  let rows: any = [];

  if (orders) {
    rows = orders.map((order) => {
      return {
        id: order.id,
        customer: order.user.name,
        amount: formatPrice(order.amount / 100),
        paymentStatus: order.status,
        date: moment(order.createDate).fromNow(),
        deliveryStatus: order.deliveryStatus,
      };
    });
  }

  const columns: GridColDef[] = [
    { field: "id", headerName: "ID", width: 220 },
    { field: "customer", headerName: "Ім'я клієнта", width: 130 },
    {
      field: "amount",
      headerName: "Ціна(UAH)",
      width: 130,
      renderCell: (params) => {
        return (
          <div className="font-bold text-slate-800">{params.row.amount}</div>
        );
      },
    },
  ],

```

```

    },
    {
      field: "paymentStatus",
      headerName: "Статус платежу",
      width: 130,
      renderCell: (params) => {
        return (
          <div>
            {params.row.paymentStatus === "pending" ? (
              <Status
                text="pending"
                icon={MdAccessTimeFilled}
                bg="bg-slate-200"
                color="text-slate-700"
              />
            ) : params.row.paymentStatus === "complete" ? (
              <Status
                text="completed"
                icon={MdDone}
                bg="bg-green-200"
                color="text-green-700"
              />
            ) : (
              <></>
            )}
          </div>
        );
      },
    },
    {
      field: "deliveryStatus",
      headerName: "Статус доставки",
      width: 130,
      renderCell: (params) => {
        return (
          <div>
            {params.row.deliveryStatus === "pending" ? (
              <Status
                text="В очікуванні"
                icon={MdAccessTimeFilled}
                bg="bg-slate-200"
                color="text-slate-700"
              />
            ) : params.row.deliveryStatus === "dispatched" ? (
              <Status
                text="Відправлено"
                icon={MdDeliveryDining}
                bg="bg-purple-200"
                color="text-purple-700"
              />
            ) : params.row.deliveryStatus === "delivered" ? (

```

```

        <Status
          text="доставлено"
          icon={MdDone}
          bg="bg-green-200"
          color="text-green-700"
        />
      ) : (
        <></>
      )}
    </div>
  );
},
},
{
  field: "date",
  headerName: "Дата",
  width: 130,
},
{
  field: "action",
  headerName: "Дії",
  width: 200,
  renderCell: (params) => {
    return (
      <div className="flex justify-between gap-4 w-full">
        <ActionBtn
          icon={MdDeliveryDining}
          onClick={() => {
            handleDispatch(params.row.id);
          }}
        />
        <ActionBtn
          icon={MdDone}
          onClick={() => {
            handleDeliver(params.row.id);
          }}
        />
        <ActionBtn
          icon={MdRemoveRedEye}
          onClick={() => {
            router.push(`/order/${params.row.id}`);
          }}
        />
      </div>
    );
  },
},
];

const handleDispatch = useCallback((id: string) => {
  axios

```

```

        .put("/api/order", {
            id,
            deliveryStatus: "dispatched",
        })
        .then((res) => {
            toast.success("Замовлення відправлено");
            router.refresh();
        })
        .catch((err) => {
            toast.error("Щось пішло не так");
            console.log(err);
        });
    }, []);

const handleDeliver = useCallback((id: string) => {
    axios
        .put("/api/order", {
            id,
            deliveryStatus: "delivered",
        })
        .then((res) => {
            toast.success("Замовлення доставлено");
            router.refresh();
        })
        .catch((err) => {
            toast.error("Щось пішло не так");
            console.log(err);
        });
    }, []);

return (
    <div className="max-w-[1150px] m-auto text-xl">
        <div className="mb-4 mt-8">
            <Heading title="Керуйте замовленнями" center />
        </div>
        <div style={{ height: 600, width: "100%" }}>
            <DataGrid
                rows={rows}
                columns={columns}
                initialState={{
                    pagination: {
                        paginationModel: { page: 0, pageSize: 9 },
                    },
                }}
                pageSizeOptions={[9, 20]}
                checkboxSelection
                disableRowSelectionOnClick
            />
        </div>
    </div>
);

```

```
};
```

```
export default ManageOrdersClient;
```

Додаток AddProductForm

```
"use client";
```

```
import Button from "@app/components/Button";
import Heading from "@app/components/Heading";
import CategoryInput from "@app/components/inputs/CategoryInput";
import CustomCheckBox from "@app/components/inputs/CustomCheckBox";
import Input from "@app/components/inputs/Input";
import SelectColor from "@app/components/inputs/SelectColor";
import TextArea from "@app/components/inputs/TextArea";
import firebaseApp from "@libs/firebase";
import { categories } from "@utils/Categories";
import { colors } from "@utils/Colors";
import { useCallback, useEffect, useState } from "react";
import { FieldValues, SubmitHandler, useForm } from "react-hook-form";
import toast from "react-hot-toast";
```

```
import {
  getDownloadURL,
  getStorage,
  ref,
  uploadBytesResumable,
} from "firebase/storage";
import axios from "axios";
import { useRouter } from "next/navigation";
```

```
export type ImageType = {
  color: string;
  colorCode: string;
  image: File | null;
};
```

```
export type UploadedImageType = {
  color: string;
  colorCode: string;
  image: string;
};
```

```
const AddProductForm = () => {
  const router = useRouter();
  const [isLoading, setIsLoading] = useState(false);
  const [images, setImages] = useState<ImageType[] | null>();
  const [isProductCreated, setIsProductCreated] = useState(false);

  const {
    register,
    handleSubmit,
```

```

    setValue,
    watch,
    reset,
    formState: { errors },
  } = useForm<FieldValues>({
    defaultValues: {
      name: "",
      description: "",
      brand: "",
      category: "",
      inStock: false,
      images: [],
      price: "",
    },
  });

  useEffect(() => {
    setCustomValue("images", images);
  }, [images]);

  useEffect(() => {
    if (isProductCreated) {
      reset();
      setImages(null);
      setIsProductCreated(false);
    }
  }, [isProductCreated]);

  const onSubmit: SubmitHandler<FieldValues> = async (data) => {
    console.log("Дані продукту", data);

    setIsLoading(true);
    let uploadedImages: UploadedImageType[] = [];

    if (!data.category) {
      setIsLoading(false);
      return toast.error("Категорія не обрана!");
    }

    if (!data.images || data.images.length === 0) {
      setIsLoading(false);
      return toast.error("Зображення не вибрано!");
    }

    const handleImageUploads = async () => {
      toast("Створення продукту, зачекайте..");
      try {
        for (const item of data.images) {
          if (item.image) {
            const fileName = new Date().getTime() + "-" + item.image.name;
            const storage = getStorage(firebaseApp);

```

```

const storageRef = ref(storage, `products/${fileName}`);
const uploadTask = uploadBytesResumable(storageRef, item.image);

await new Promise<void>((resolve, reject) => {
  uploadTask.on(
    "state_changed",
    (snapshot) => {
      const progress =
        (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
      console.log("Завантаження " + progress + "%");
      switch (snapshot.state) {
        case "paused":
          console.log("Завантаження призупинено");
          break;
        case "running":
          console.log("Виконується завантаження");
          break;
      }
    },
    (error) => {
      console.log("Помилка завантаження зображення", error);
      reject(error);
    },
    () => {
      getDownloadURL(uploadTask.snapshot.ref)
        .then((downloadURL) => {
          uploadedImages.push({
            ...item,
            image: downloadURL,
          });
          console.log("Файл доступний за адресою", downloadURL);
          resolve();
        })
        .catch((error) => {
          console.log(
            "Помилка отримання URL-адреси завантаження",
            error
          );
          reject(error);
        });
    }
  );
});
} catch (error) {
  setIsLoading(false);
  console.log("Помилка обробки завантажень зображень", error);
  return toast.error("Помилка обробки завантажень зображень");
}
};

```

```

await handleImageUploads();
const productData = { ...data, images: uploadedImages };

axios
  .post("/api/product", productData)
  .then(() => {
    toast.success("Product created");
    setIsProductCreated(true);
    router.refresh();
  })
  .catch((error) => {
    toast.error("Під час збереження продукту в базі даних сталася помилка");
  })
  .finally(() => {
    setIsLoading(false);
  });
};

const category = watch("category");

const setCustomValue = (id: string, value: any) => {
  setValue(id, value, {
    shouldValidate: true,
    shouldDirty: true,
    shouldTouch: true,
  });
};

const addImageToState = useCallback((value: ImageType) => {
  setImages((prev) => {
    if (!prev) {
      return [value];
    }

    return [...prev, value];
  });
}, []);

const removeImageFromState = useCallback((value: ImageType) => {
  setImages((prev) => {
    if (prev) {
      const filteredImages = prev.filter(
        (item) => item.color !== value.color
      );
      return filteredImages;
    }

    return prev;
  });
}, []);

```

```

return (
  <>
    <Heading title="Додати продукт" center />
    <Input
      id="name"
      label="Назва"
      disabled={isLoading}
      register={register}
      errors={errors}
      required
    />
    <Input
      id="price"
      label="Ціна"
      disabled={isLoading}
      register={register}
      errors={errors}
      type="number"
      required
    />
    <Input
      id="brand"
      label="Бренд"
      disabled={isLoading}
      register={register}
      errors={errors}
      required
    />
    <TextArea
      id="description"
      label="Опис"
      disabled={isLoading}
      register={register}
      errors={errors}
      required
    />
    <CustomCheckBox
      id="inStock"
      register={register}
      label="Цей товар в наявності"
    />
    <div className="w-full font-medium">
      <div className="mb-2 font-semibold">Оберіть категорію</div>
      <div className="grid grid-cols-2 md:grid-cols-3 gap-3 max-h[50vh] over-
flow-y-auto">
        {categories.map((item) => {
          if (item.label === "All") {
            return null;
          }
        })}
      </div>
    </div>
  </>
)

```

```

        return (
          <div key={item.label} className="col-span">
            <CategoryInput
              onClick={({category}) => setCustomValue("category", category)}
              selected={category === item.label}
              label={item.label}
              icon={item.icon}
            />
          </div>
        );
      }
    </div>
  </div>
  <div className="w-full flex flex-col flex-wrap gap-4">
    <div>
      <div className="font-bold">
        Виберіть доступні кольори продукту та завантажте їх зображення.
      </div>
      <div className="text-sm">
        Ви повинні завантажити зображення для кожного вибраного кольору,
        інакше ваш вибір кольору буде проігноровано.
      </div>
    </div>
    <div className="grid grid-cols-2 gap-3">
      {colors.map((item, index) => {
        return (
          <SelectColor
            key={index}
            item={item}
            addImageToState={addImageToState}
            removeImageFromState={removeImageFromState}
            isProductCreated={isProductCreated}
          />
        );
      })}
    </div>
    <div>
      <Button
        label={isLoading ? "Завантаження..." : "Додати продукт"}
        onClick={handleSubmit(onSubmit)}
      />
    </div>
  );
};

export default AddProductForm;

```

Додаток Summary.tsx

```

"use client";

import { Order, Product, User } from "@prisma/client";
import { useEffect, useState } from "react";
import Heading from "../components/Heading";
import { formatPrice } from "@/utils/formatPrice";
import { formatNumber } from "@/utils/formatNumber";

interface SummaryProps {
  orders: Order[];
  products: Product[];
  users: User[];
}

type SummaryDataType = {
  [key: string]: {
    label: string;
    digit: number;
  };
};

const Summary: React.FC<SummaryProps> = ({ orders, products, users }) => {
  const [summaryData, setSummaryData] = useState<SummaryDataType>({
    sale: {
      label: "Всього продано",
      digit: 0,
    },
    products: {
      label: "Всього товарів",
      digit: 0,
    },
    orders: {
      label: "Всього замовлень",
      digit: 0,
    },
    paidOrders: {
      label: "Оплачені замовлення",
      digit: 0,
    },
    unpaidOrders: {
      label: "Неоплачені замовлення",
      digit: 0,
    },
    users: {
      label: "Загальна кількість користувачів",
      digit: 0,
    },
  });

  useEffect(() => {
    setSummaryData((prev) => {

```

```

let tempData = { ...prev };

const totalSale = orders.reduce((acc, item) => {
  if (item.status === "complete") {
    return acc + item.amount;
  } else return acc;
}, 0);

const paidOrders = orders.filter((order) => {
  return order.status === "complete";
});

const unpaidOrders = orders.filter((order) => {
  return order.status === "pending";
});

tempData.sale.digit = totalSale;
tempData.orders.digit = orders.length;
tempData.paidOrders.digit = paidOrders.length;
tempData.unpaidOrders.digit = unpaidOrders.length;
tempData.products.digit = products.length;
tempData.users.digit = users.length;

return tempData;
});
}, [orders, products, users]);

const summaryKeys = Object.keys(summaryData);

return (
  <div className="max-w-[1150px] m-auto">
    <div className="mb-4 mt-8">
      <Heading title="Статистика" center />
    </div>
    <div className="grid grid-cols-2 gap-3 max-h-50vh overflow-y-auto">
      {summaryKeys &&
        summaryKeys.map((key) => {
          return (
            <div
              key={key}
              className="rounded-xl border-2 p-4 flex flex-col items-center
gap-2 transition"
            >
              <div className="text-xl md:text-4xl font-bold">
                {summaryData[key].label === "Всего продано" ? (
                  <>{formatPrice(summaryData[key].digit)}</>
                ) : (
                  <>{formatNumber(summaryData[key].digit)}</>
                )}
              </div>
              <div>{summaryData[key].label}</div>
            </div>
          )
        })
      }
    </div>
  </div>
)

```

```

        </div>
      );
    }
  }
  </div>
</div>
);
};

export default Summary;

```

Додаток firebase.ts

```

import { initializeApp } from "firebase/app";

const firebaseConfig = {
  apiKey: "AIzaSyA4lcqj4Rsvf_9n-5jdjsZtxILzJdWrdxU",
  authDomain: "top-pc-c33f2.firebaseio.com",
  projectId: "top-pc-c33f2",
  storageBucket: "top-pc-c33f2.appspot.com",
  messagingSenderId: "463038968428",
  appId: "1:463038968428:web:c657591dc0a3b293f85e49",
};

const firebaseApp = initializeApp(firebaseConfig);

export default firebaseApp;

```

Додаток stripe-webhook.ts

```

import { buffer } from "micro";
import { NextApiRequest, NextApiResponse } from "next";
import { type } from "os";
import Stripe from "stripe";

export const config = {
  api: {
    bodyParser: false,
  },
};

const stripe = new Stripe(process.env.STRIPE_SECRET_KEY as string, {
  apiVersion: "2022-11-15",
});

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse
) {

```

```

const buf = await buffer(req);
const sig = req.headers["stripe-signature"];

if (!sig) {
  return res.status(400).send("Missing the stripe signature");
}

let event: Stripe.Event;

try {
  event = stripe.webhooks.constructEvent(
    buf,
    sig,
    process.env.STRIPE_WEBHOOK_SECRET!
  );
} catch (err) {
  return res.status(400).send("Webhook error" + err);
}

switch (event.type) {
  case "charge.succeeded":
    const charge: any = event.data.object as Stripe.Charge;

    if (typeof charge.payment_intent === "string") {
      await prisma?.order.update({
        where: { paymentIntentId: charge.payment_intent },
        data: { status: "complete", address: charge.shipping?.address },
      });
    }
    break;
  default:
    console.log("Unhandled event type:" + event.type);
}

res.json({ received: true });
}

```