НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок «Онлайн-ринок електронної комерції для цифрових продуктів»

Виконав студентка IV курсу, групи ІП-з01
(шифр групи)
Іваницька Оксана Вадимівна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Полупан Ю. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент, к.т.н. Шимкович В.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ –2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Іваницькій Оксані Вадимівні
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок «Онлайн-ринок електронної комерції для цифрових продуктів»

керівник проєкту Полупан Юлія Вікторівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 27 »квітня 2024 р. № 2113-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема структурна компонентів програмного забезпечення _____

3) Схема бази даних _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «11» березня 2024 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв’язання задачі	06.04.2024	
3	Постановка та формалізація задачі	13.04.2024	
4	Розробка інформаційного забезпечення	20.04.2024	
5	Алгоритмізація задачі	27.04.2024	
6	Обґрунтування вибору використаних технічних засобів	04.05.2024	
7	Розробка програмного забезпечення	18.05.2024	
8	Налагодження програми	26.05.2024	
9	Виконання графічних документів	29.05.2024	
10	Оформлення пояснювальної записки	30.05.2024	
11	Подання ДП на попередній захист	02.06.2024	
12	Подання ДП рецензенту	12.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Оксана ІВАНИЦЬКА
(підпис) (ініціали, прізвище)

Керівник _____ Юлія ПОЛУПАН
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 30 таблиць, 42 рисунки та 21 джерело – загалом 77 сторінок

Мета дипломного проєкту: вивчити особливості та специфіку створення вебзастосунків для купівлі та продажу цифрових продуктів.

Об’єкт дослідження: програмні інструменти здійснення електронної комерції.

Предмет дослідження: процеси розроблення, впровадження, супроводження і забезпечення якості програмного забезпечення на прикладі вебзастосунку для здійснення електронної комерції.

У першому розділі представлено загальні положення про стан ринку електронної комерції в Україні та світі. Наведено визначення цифрових продуктів, переваги їх продажу на онлайн-ринках, такі як: однакові можливості для торгівлі для великих та невеликих фірм, цілодобова доступність, гнучкість. Як приклад представлено такі платформи, як Amazon, GooglePlay, Rozetka.

Проведено порівняння монолітної та мікросервісної архітектур. На основі зробленого порівняння та наявних для розробки ресурсів було прийнято рішення використовувати монолітну архітектуру.

Представлено опис бізнес-процесів розробки у вигляді BPMN схем, покрокову послідовність кожного процесу.

У другому розділі визначено основні функції додатку у вигляді діаграми варіантів використання. Загальною функцією було визначено забезпечення здійснення публікації своїх товарів для продавців та купівлі даних продуктів для споживачів. Також функціонал вебзастосунку передбачає пошук, сортування та додавання товарів до списку обраних.

Представлено перелік функціональних вимог до вебзастосунку. Для візуалізації покриття розробки тестами була використана матриця трасування вимог.

Також наведено список нефункціональних вимог, серед яких: безпека, кросбраузерність, простота у використанні.

У третьому розділі описано архітектуру розробки, наведено її наочне зображення на діаграмі компонентів.

Розглянуто існуючі засоби розробки, які можуть бути використані для створення вебдодатків, серед яких було обрано JavaScript, Next.js і MongoDB.

Описано основні моменти розробки клієнтської та серверної частин додатку. У вигляді ER-моделі зображено структуру БД.

Описано основні утиліти, бібліотеки та ресурси, що використовуються у розробці веб-додатку. Проведено аналіз безпеки даних. Передбачено перевірку аутентифікації користувача, безпечне зберігання пароля у вигляді шифру. Реалізовано основний функціонал веб-забезпечення.

У четвертому розділі виконано тестування програмного забезпечення з метою перевірки правильності роботи додатку та виправлення помилок. Було описано покрокове використання розробленого вебдодатку.

У п’ятому розділі описані стратегії та практики розгортання програм Next.js. Були також наведені основні практики для оптимізації процесу розгортання та підвищення продуктивності та надійності програми Next.js, такі як використання змінних середовища та Git. Після цього було наведено опис покрокового розгортання розробленого програмного забезпечення.

КЛЮЧОВІ СЛОВА: ЦИФРОВИЙ ТОВАР, МАРКЕТПЛЕЙС, ВЕБЗАСТОСУВАННЯ, ЕЛЕКТРОННА КОМЕРЦІЯ.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 30 tables, 42 figures and 21 sources – in total 77 pages.

The purpose of the diploma project is to study the features and specifics of creating web applications for buying and selling digital products.

The object of study: software tools for e-commerce.

The subject of the study: the processes of development, implementation, maintenance and quality assurance of software on the example of a web application for e-commerce.

In the first section general information about the e-commerce market in Ukraine and the world were described. The definition of digital products, the advantages of selling them on online markets, such as: equal trading opportunities for large and small firms, round-the-clock availability, flexibility, were given. As an example, such platforms as Amazon, Google Play, Rozetka were presented.

A comparison of monolithic and microservice architectures was made. Based on the comparison made and the resources available for development, a decision was made to use a monolithic architecture.

A description of business processes in the form of BPMN schemes and a step-by-step sequence of each process were presented.

In the second section the main functions of the application were defined in the form of a use case diagram. The general function was defined as ensuring the publication of their goods for sellers and the purchase of these products for consumers. Also, the functionality of the web application includes searching, sorting and adding products to the list of favorites.

A list of functional requirements for the application was presented. A requirement tracing matrix was used to visualize the development coverage by tests. A list of non-functional requirements was also given.

In the third section the architecture was described, its visual representation on the component diagram was given.

The development tools that can be used to create web applications were considered, among which JavaScript, Next.js and MongoDB were chosen.

The development of the client and server parts of the application were described. Database ER diagram was also developed.

Utilities, libraries, and resources used in web application development were described. A data security analysis was conducted. The main functionality of the web application has been implemented.

In the fourth section test plan and the results of the testing were submitted. The step-by-step use of the developed web application was described.

In the fifth section strategies and practices for deploying Next.js applications were described. Basic practices for optimizing the deployment process and improving Next.js application performance and reliability were also covered, such as using environment variables and Git. After that, a description of the step-by-step deployment of the developed software was provided.

KEYWORDS: DIGITAL PRODUCT, MARKETPLACE, WEB APPLICATION, E-COMMERCE.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-з0102.045440.01.91	Технічне завдання	17	
3	А4	КПІ.ІП-з0102.045440.02.81	Пояснювальна записка	77	
4	А4	КПІ.ІП-з0102.045440.03.12	Текст програми	41	
5	А4	КПІ.ІП-з0102.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-з0102.045440.05.34	Керівництво користувача	15	
7	А3	КПІ.ІП-з0102.045440.06.99	Графічний матеріал	3	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-з0102.045440.00.90		
Розроб.		Іваницька О.В.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Полупан Ю.В.					
Н.контр.		Стельмах О.П.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-з01	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2024 р.

ВЕБЗАСТОСУНОК «ОНЛАЙН-РИНОК ЕЛЕКТРОННОЇ КОМЕРЦІЇ ДЛЯ ЦИФРОВИХ ПРОДУКТІВ»

Технічне завдання

КПІ.ІП-з0102.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юлія ПОЛУПАН

Нормоконтроль:

_____ Олександр СТЕЛЬМАХ

Виконавець:

_____ Оксана ІВАНИЦЬКА

Київ – 2024

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок «Онлайн-ринок електронної комерції для цифрових продуктів».

Галузь застосування:

Наведене технічне завдання поширюється на розробку вебзастосунку «Онлайн-ринок електронної комерції для цифрових продуктів» [КПІ.ІП-з0102.045440.01.91], котрий використовується для онлайн-торгівлі та призначений для виробників та споживачів цифрових товарів.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки вебзастосунку «Онлайн-ринок електронної комерції для цифрових продуктів» є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Запропоноване програмне забезпечення з електронної комерції призначене для спрощення процесу купівлі та продажу цифрових продуктів, надаючи інструменти для ефективної взаємодії між продавцями та покупцями та гарантуючи однаковий доступ до інформації для всіх учасників.

Метою розробки є створення програмного засобу, що буде використовуватися для купівлі та продажу цифрових товарів та допоможе продавцям залучити більше клієнтів, автоматизувати продажі та делегувати логістику.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу:

– керування додатком через дії користувача;
– введення даних користувачем, відповідь системи;
– відображення даних, введених користувачем;
– відображення вказівок щодо навігації по сайту;
– інформування користувача у разі помилки реєстрації, входу;
– дизайн повинен бути адаптований під будь-які розміри монітору.

Вигляд екранних форм наведений на рисунках 4.1 – 4.6.

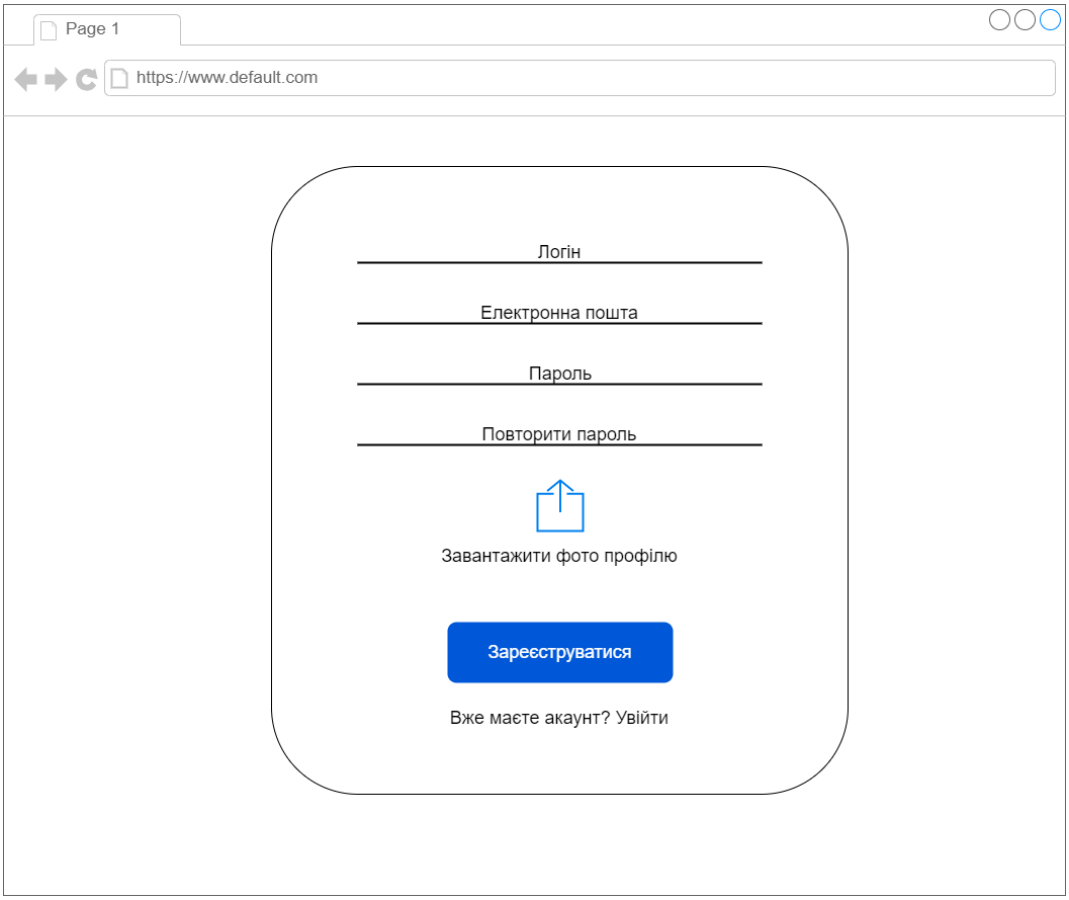


Рисунок 4.1 – Екранна форма сторінки реєстрації

Після натискання кнопки «Зареєструватися» користувач переходить на сторінку входу.

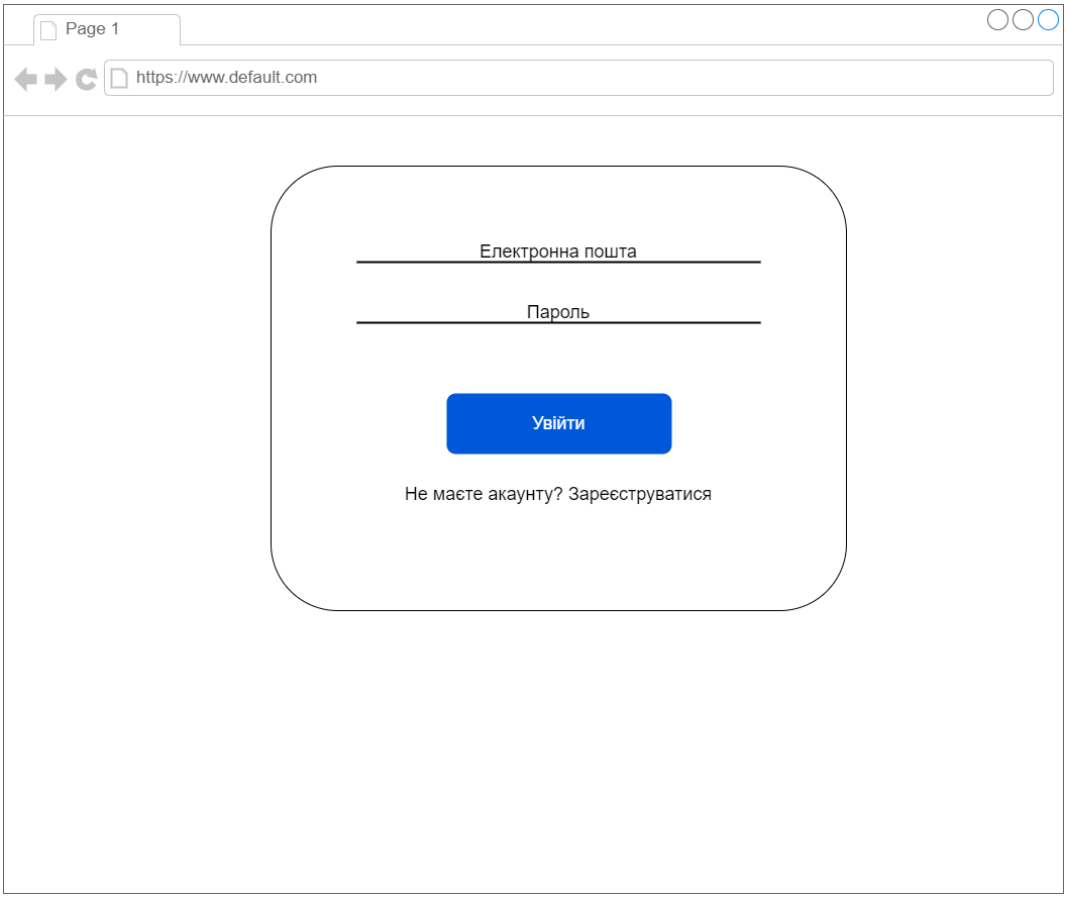


Рисунок 4.2 – Екранна форма сторінки входу

Після натискання кнопки «Увійти» користувач переходить на домашню сторінку.

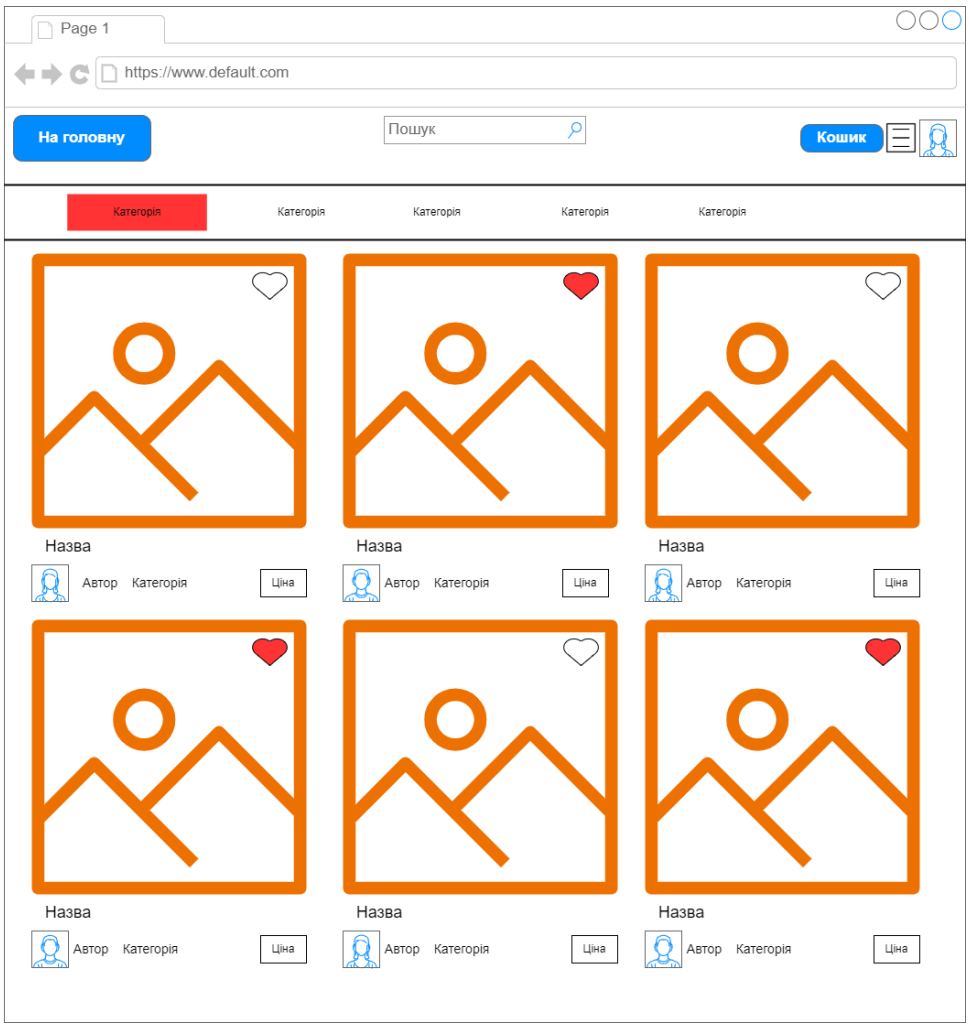


Рисунок 4.3 – Екранна форма домашньої сторінки

Якщо користувач натискає на зображення або назву товару, він перенаправляється на сторінку товару. Натиснувши кнопку «Кошик», користувач переходить на сторінку кошика.

При натисканні кнопки меню у правому верхньому кутку, відкривається меню навігації, за яким можна перейти на сторінку створення товару, або вийти з облікового запису. Для незареєстрованого користувача доступні опції увійти та зареєструватися.

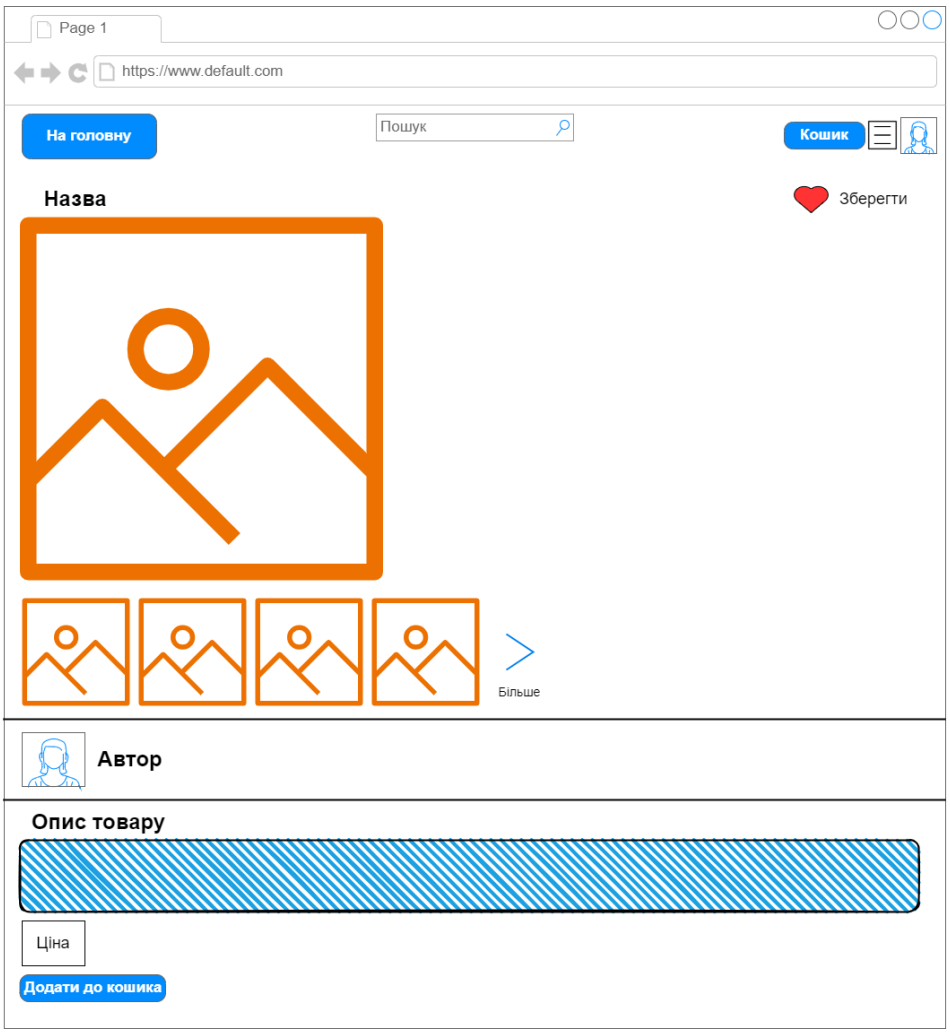


Рисунок 4.4 – Екранна форма сторінки товару

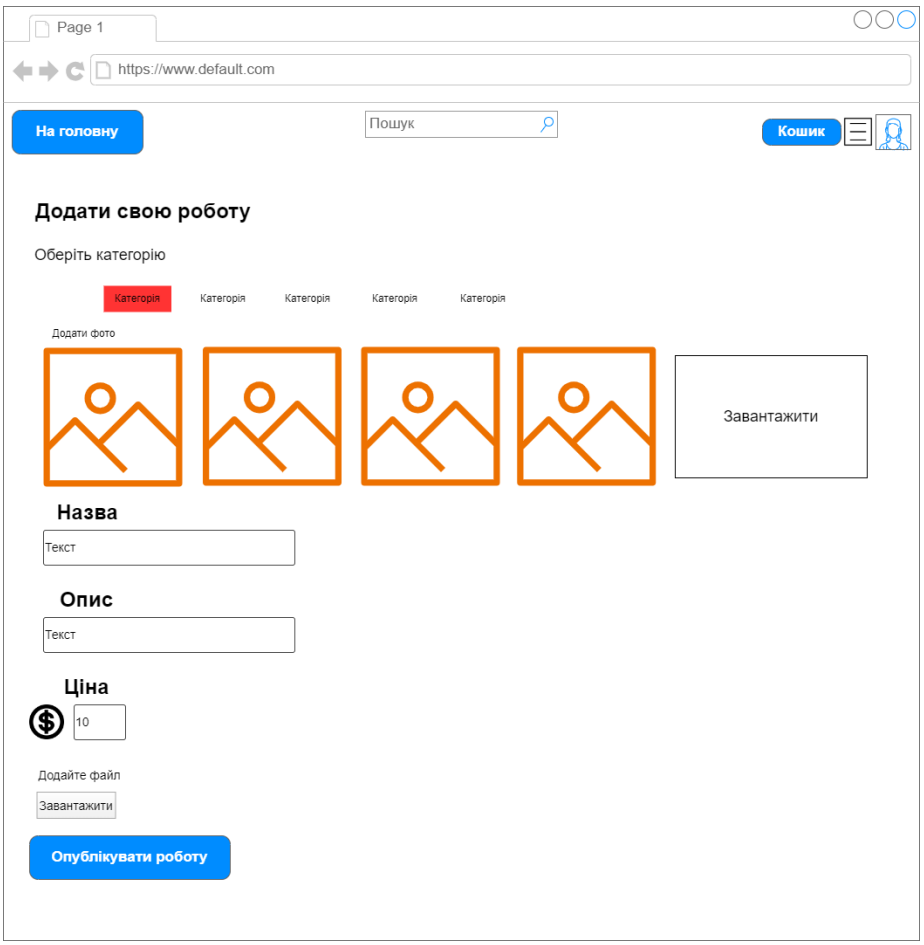


Рисунок 4.5 – Екранна форма сторінки створення товару

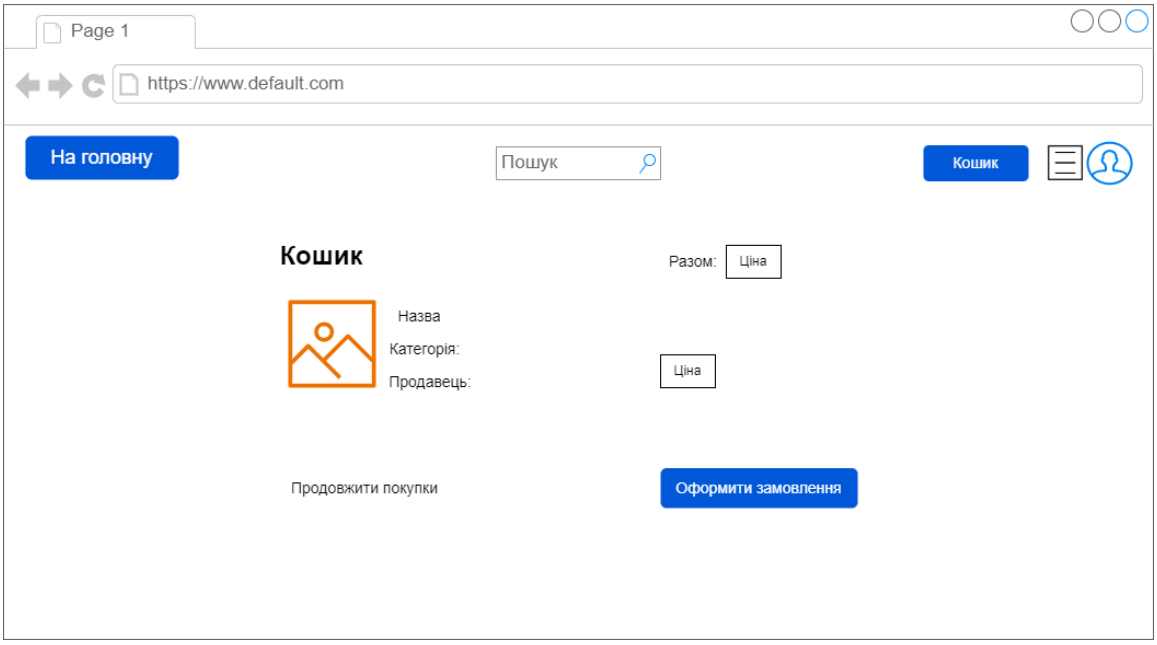


Рисунок 4.6 – Екранна форма сторінки кошика

4.1.2 Для незареєстрованого користувача:

- створення профілю користувача, при реєстрації користувач вводить наступні дані: логін, електронна пошта, пароль, підтвердження паролю;
- входу у профіль користувача, при вході користувач вводить електронну пошту і пароль;
- перегляду існуючих товарів на головній сторінці: назва продукту, ціна, категорія, зображення;
- сортування товарів на головній сторінці за категорією;
- пошуку товарів на головній сторінці за назвою;
- перегляду повної інформації про товар на сторінці товару: назва продукту, ціна, категорія, зображення, текстовий опис.

4.1.2.1 Для зареєстрованого користувача-продавця:

- завантаження товару в застосунок разом з документом, що підтверджує авторське право на товар;
- додавання зображення, назви, текстового опису товару, категорії, ціни;
- продажу товару;

4.1.2.2 Для зареєстрованого користувача-покупця:

- додавання товару в кошик;
- перегляд стану кошика покупця: назва, зображення, ціна доданих продуктів, загальна вартість продуктів у кошику;
- здійснення оплати за товар;
- отримання інформації про результат проведення оплати товару;
- завантаження придбаного товару;
- додавання товару в обране;
- подання скарги адміністратору у випадку порушення авторського права.

4.1.3 Для адміністратора системи:

– надання статусу товару: чекає перевірки або верифіковано;
– видалення товару.

4.1.4 Додаткові вимоги:

– групування товарів за категоріями;
– завантаження товару покупцем можливе лише після успішної оплати;
– зареєстрований користувач може бути одночасно і продавцем, і покупцем;
– товари стають доступними для купівлі лише після верифікації адміністратором.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних, захист паролів, даних в локальному сховищі користувача. База даних повинна мати ефективний захист від несанкціонованого доступу та дозволяти розмежовувати права доступу до даних для різних категорій користувачів.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (версії від Windows 7 і вище).

Вебдодаток повинен забезпечувати коректне відображення даних в наступних браузерах: Opera, Edge, Google Chrome.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі:

- пароль повинен містити не менше 8 символів;
- електронна пошта має відповідати шаблону (містити @);
- графічні файли: jpg, png, gif, svg, psd, eps, ico;
- текстові файли: pdf, epub, fb2, txt;
- шрифти : cff, eot, otf, ttf.
- архіви: zip, rar.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі:

- графічні файли: jpg, png, gif, svg, psd, eps, ico;

- текстові файли: pdf, epub, fb2, txt;
- шрифти : cff, eot, otf, ttf.
- архіви: zip, rar.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування JavaScript та з використанням HTML 5, CSS 3, Next.js 14, React.js.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Node.js з використанням Visual Studio Code.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді файлів js, tsx, css.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;

– технічне завдання;

– текст програми;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;

– схема бази даних;

– архітектура програмного забезпечення;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Вебзастосунок «Онлайн-ринок електронної комерції для цифрових продуктів»

КПІ.ІП-з0102.045440.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
SDK	–	Software development kit
IT	–	Інформаційні технології
ER	–	Entity-Relation diagram
ОС	–	Операційна система.
БД	–	База даних.
СУБД	–	Система управління базами даних.
ПЗ	–	Програмне забезпечення
JSON	–	JavaScript Object Notation – запис об'єктів JavaScript.
DBaaS	–	База даних як сервіс.

ВСТУП

Цифрові продукти споживають усі – від музики до відео, від електронних книг до онлайн-курсів. Через їх популярність і простоту розповсюдження багато підприємців запускають лінійки цифрових товарів як доповнення до фізичних товарів чи послуг. Окрім того, їх можна створювати один раз і багаторазово продавати різним клієнтам.

Для підприємців продаж цифрових товарів має масу переваг. Товар завжди в наявності, не потрібно шукати місце для зберігання товарів, крім того, в більшості випадків покупець отримує товар відразу після оплати.

Використання маркетплейсу для купівлі цифрових товарів надасть покупцям можливість проводити порівняння цін від різних продавців, обираючи найкращий із доступних варіантів, а також можливість придбання різних товарів в одному місці та забезпечення миттєвого доступу до товару одразу після оплати, наприклад, шляхом завантаження файлів. Для підприємців маркетплейс стане гарним варіантом отримання стабільних продажів на старті бізнесу.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Цифрові продукти — це нематеріальні товари, які існують лише у електронній формі. Вони зберігаються на комп’ютерах, серверах, у хмарних сховищах та поширюються через посилання на завантаження файлів.

Велика кількість конкурентих переваг цифрових товарів робить їх продаж привабливим для бізнесу. Серед них:

1) популярність і простота розповсюдження. Інтернет — це безмежний ринок, де можна охопити покупців у всьому світі без плутанини через кордони чи логістику доставки. Створений цифровий продукт не може закінчитися або зіпсуватися, на відміну від фізичного товару. Таким чином, його можна багаторазово продавати і поширювати між клієнтами.

2) автоматизація. Процес торгівлі, від обробки замовлення до доставки, часто можна повністю автоматизувати, звільняючи час і ресурси [1].

3) низькі накладні витрати. Відсутня потреба у фізичних сховищах для зберігання та повторюваних затрат підвищує рентабельність. Витрати на початкові інвестиції у створення цифрових продуктів і послуг високі, тоді як граничні витрати на їх поширення низькі. Така динаміка витрат дозволяє отримати великий обсяг виробництва з меншими середніми витратами на одиницю продукції [2].

Купівля та продаж цифрових товарів здійснюється за допомогою онлайн-ринків та онлайн-магазинів.

Онлайн-ринок (або онлайн-маркетплейс) – це тип сайту електронної комерції, де інформація про продукти чи послуги надається третіми сторонами, а транзакції обробляються операторами ринку.

Перевагами маркетплейсу для продавців та споживачів є:

1) однакові можливості для торгівлі як для великих, так і для невеликих фірм. Маючи низький бар’єр входу, ринок цифрових продуктів надає підприємцям швидкий доступ до перспективного ринку, який зростає

прискореними темпами. Маркетплейс цифрових продуктів надає підприємцям-початківцям можливість отримувати реалістичні короткострокові прибутки, а також дозволяє їм поширити свій бренд.

2) гнучкість. Ринок цифрових продуктів потенційно може працювати будь-де, потрібно лише підключення до Інтернету. Це розширює варіанти вибору місця розташування головного офісу та забезпечує потенціал для масштабування та розширення бізнесу у всьому світі.

3) доступність. Товари та інформацію про них можна отримати у режимі реального часу цілодобово.

Однак існують і потенційні ризики, найголовнішим з яких є загроза шахрайства. Якщо рівень безпеки додатку буде низький, споживачі та виробники можуть стати жертвами комп'ютерних злочинців. Окрім того, продукти на легальних ринках можуть пропонуватися паралельно через незаконні піратські канали [3]. Тому чимало уваги потрібно приділити забезпеченню захисту інтелектуальної власності та персональної інформації про клієнтів.

Одним з рішень цих проблем є створення безпечної та надійної платформи, здатної протистояти онлайн-загрозам.

Електронна комерція є, загалом, перспективним для бізнесу напрямком, що швидко розвивається. Протягом 2014 - 2020 рр. він зріс практично у 3 рази. Аналітики передбачають, що під кінець 2024 року загальний прибуток від цієї сфери зросте до 6388 млрд. дол. США [4].

Після 2022 року, стан електронної комерції в Україні погіршився. Згідно з дослідженням Promodo, компанії, задіяні в цій сфері, зазнали значних збитків, деякі втратили понад 80% трафіку та майже весь прибуток. Але поступово діяльність фірм відновлюється, виробники пристосовуються до нових умов. Підприємства все більше переходять на використання нових технологій електронної комерції. Такий підхід забезпечує стійкість та дозволяє їм залишатися конкурентоспроможними в період невизначеності.

Популярною серед ключових рітейлерів є тенденція до розширення їх інтернет-магазинів у маркетплейси [5].

Майбутнім шляхом розвитку даної сфери може бути збільшення кількості спеціалізованих маркетплейсів, які б відповідали лише за цифрову продукцію, на локальному українському ринку. Успішне зростання даного напрямку є корисним для досягнення таких цілей, як:

- забезпечення надходження інвестицій в національну економіку та нарощування її потужності;
- розвиток об'єму внутрішнього IT-ринку, що загалом покращить рівень життя населення;
- забезпечення безбар'єрного поширення онлайн-комерції та прискорення інтеграції України в міжнародний електронний бізнес.

1.2 Аналіз існуючих рішень

Проаналізуємо відомі на сьогодні технологічні забезпечення, що допоможуть у реалізації вебзастосунку «Онлайн-ринок електронної комерції для цифрових продуктів». Далі будуть розглянуті існуючі технічні рішення та IT-проєкти.

1.2.1 Аналіз відомих програмних продуктів

У пункті описано основні особливості трьох найпопулярніших онлайн-маркетплейсів в Україні та світі станом на 2024 рік.

Першим з них є Amazon. Amazon (рисунок 1.1) - один із найбільших міжнародних маркетплейсів, який існує вже понад 30 років. Його щоденна аудиторія у розмірі 28 млн. чоловік сприяє високому товарообігу.

Спочатку ця платформа розроблялася як локальний цифровий магазин, але трохи пізніше перетворилася на повноцінний бренд із безліччю продуктів, серед яких і маркетплейс, де можна продавати та купувати всі види товарів: одяг, техніку, меблі тощо.

Маючи кілька постачальників для кожного продукту, Amazon забезпечує конкурентоспроможні ціни, надаючи клієнтам свободу вибору кращої пропозиції. Віддаючи пріоритет якості та підтримуючи надійну систему зворотного зв'язку, Amazon забезпечує задоволення клієнтів та безпеку замовлень.

Проте варто згадати і кілька складнощів, з якими можна зіткнутися:

- складна реєстрація. Мало хто попереджає про те, що складнощі з Amazon можуть розпочатися задовго до того, як ви почнете щось продавати. Спочатку потрібно буде вивчити особливості реєстрації, щоб її пройти, подати копію поспорта, налаштувати проксі. Іноді Amazon вимагає додаткову документацію після створення облікового запису.
- постійні блокування користувачів. Amazon може заблокувати акаунт та вимагати документи практично на будь-якому етапі (від підтвердження особи до надання рахунків на товар). Також кілька разів на рік відбуваються масові блокування, які часто нічим не обґрунтовані. Вони теж не припиняться без вашої участі, тому потрібно бути готовим до того, що вам потрібно буде зв'язуватися з підтримкою не один раз. Найчастіше такі блокування відбуваються через нові заходи безпеки, оскільки деякі продавці справді підробляють документи, продають неякісний товар, порушують права на інтелектуальну власність тощо.
- мовний бар'єр. Незнання англійської мови може бути проблемою, особливо якщо вам потрібно спілкуватися з підтримкою [6].

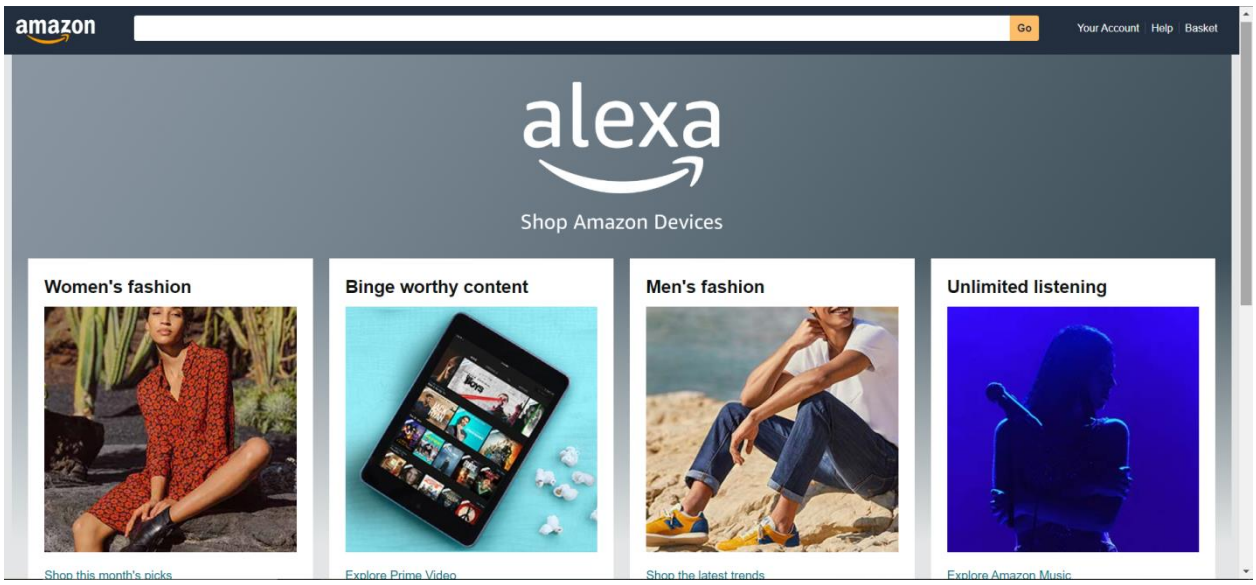


Рисунок 1.1 – Головна сторінка Amazon

Наступний продукт, GooglePlay (рисунок 1.2), офіційний магазин додатків для смартфонів Android, перетворився на важливий компонент екосистеми мобільних додатків, що надає широкий спектр програм та цифрового контенту. Як одна з найбільших у світі платформ поширення цифрових товарів, вона має як переваги, так і недоліки для користувачів та розробників.

Переваги GooglePlay:

- великий вибір програм. GooglePlay має велику різноманітність цифрових товарів, включаючи ігри, інструменти для підвищення продуктивності, книги тощо.
- зручний інтерфейс користувача.
- оновлення програм на регулярній основі. Розробники мають змогу постійно випускати оновлення додатків у GooglePlay, гарантуючи, що споживачі матимуть доступ до останніх функцій та виправлень помилок.
- увімкнені функції безпеки. GooglePlay використовує вбудовані функції безпеки для перевірки програм на наявність шкідливого програмного забезпечення та небезпечного контенту, тим самим підвищуючи безпеку користувачів та їх даних.

Недоліки GooglePlay:

– платформа орієнтована на користувачів Android, що звужує коло можливих продавців та покупців, що користуються іншими операційними системами.
– фрагментація екосистеми Android спричиняє чимало труднощів для розробників, оскільки вони повинні забезпечити сумісність своїх додатків з різними налаштуваннями пристрою.
– поділ доходів. GooglePlay стягує комісію у розмірі 30% з продажу програм, покупок у додатках та підписок, яка може швидко накопичуватися.
– процес розгляду. Процес затвердження програми GooglePlay може бути тривалим, що призводить до затримок в оновленні та запуску програм [7].

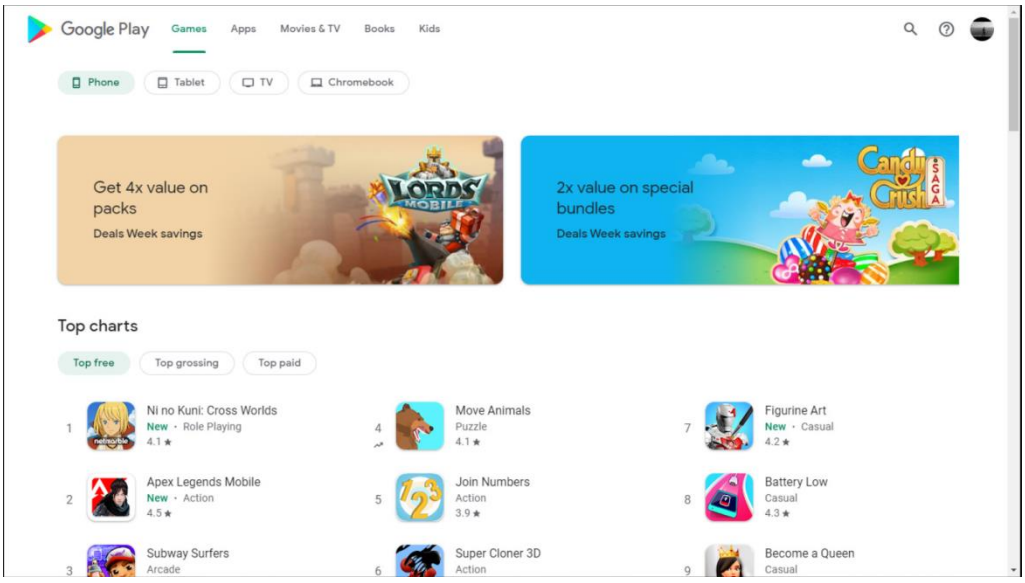


Рисунок 1.2 – Головна сторінка GooglePlay

На українському ринку цифрової комерції найпопулярнішим є інтернет-магазин Rozetka (рисунок 1.3), що розпочав свою діяльність у 2005 році. Спочатку дана платформа використовувалася для продажу техніки. Але зараз тут можна купити побутову техніку, одяг, зоотовари і навіть музичні інструменти. На сайті Rozetka представлено приблизно 4 млн. видів товарів, щоденно сайт відвідують 2,5 млн. осіб. Після 2016 року на платформі

відбулося додавання функціональності торговельного майданчика та залучення до продажу інших учасників, які тепер відповідають за 25% від усіх продажів [4].

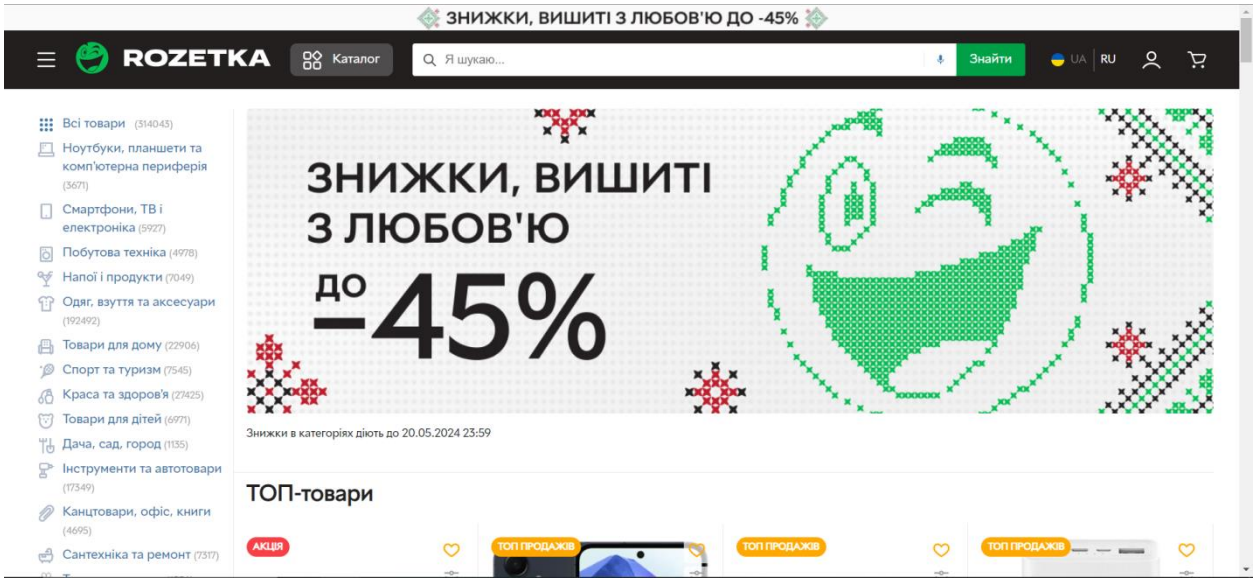


Рисунок 1.3 – Головна сторінка Rozetka

В таблиці 1.1 виконано порівняння дипломного проекту з аналогами.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Дипломний проєкт	Amazon	GooglePlay	Rozetka
Орієнтований виключно на цифрові товари	+	-	+	-
Можливість додати товар в обране	+	+	+	+
Пошук товарів на сайті	+	+	+	+
Відсутність реклами	+	-	-	-

Таким чином, у порівнянні з іншими платформами, додаток, розроблений в межах дипломної роботи є мінімалістичним, орієнтованим виключно на цифрові товари та забезпечує відсутність реклами, що покращує

комфортність та простоту в користуванні і відсутність відволікаючих сповіщень.

1.2.2 Аналіз відомихалгоритмічних та технічних рішень

Вибір правильної архітектури програмного забезпечення – запорука її успіху у майбутньому. В даному пункті розглянуто два популярні та дієві підходи – монолітну та мікросервісну архітектуру.

Почнемо із традиційної монолітної архітектури. Вона є єдиною структурою, у якій всі компоненти програми міцно пов'язані між собою. У контексті веб-застосунків ми говоримо про те, що клієнтський і серверний код знаходяться всередині одного проекту (рисунок 1.4).

Головний плюс монолітів – їхня простота та легкість розробки. Компоненти монолітної системи тісно пов'язані, тому писати та тестувати такий код порівняно легко. Єдина база коду спрощує розуміння загальної логіки програми.

До явних переваг монолітної архітектури відносяться продуктивність та ефективність. Оскільки всі компоненти виконуються у межах одного процесу, відпадає потреба у міжпроцесній взаємодії. Тобто, швидший час виконання та мінімальні часові витрати.

І ще один плюс монолітів – середовище спільно використовуваних даних. Всі компоненти мають прямий доступ до тієї ж бази даних, що дозволяє безперешкодно обмінюватися інформацією. Тим самим усувається потреба у складних механізмах синхронізації.

У монолітній архітектурі є й вади. Один із них – масштабування. Масштабувати моноліти не так вже й просто, адже вся програма масштабується як єдине ціле. У результаті, якщо додаткові ресурси потрібні лише для певних компонентів, це може вилитися в їх нераціональне використання.

Крім того, весь моноліт споживає ресурси сервера, на якому працює. А якщо у вас є складний продукт з безліччю опцій, то вся система почне

гальмувати, коли для роботи однієї з таких функцій будуть потрібні додаткові ресурси.

Моноліти часто «зав'язані» на певному стеку технологій. І додавання нових технологій або фреймворків може призвести до перепроектування всієї програми.

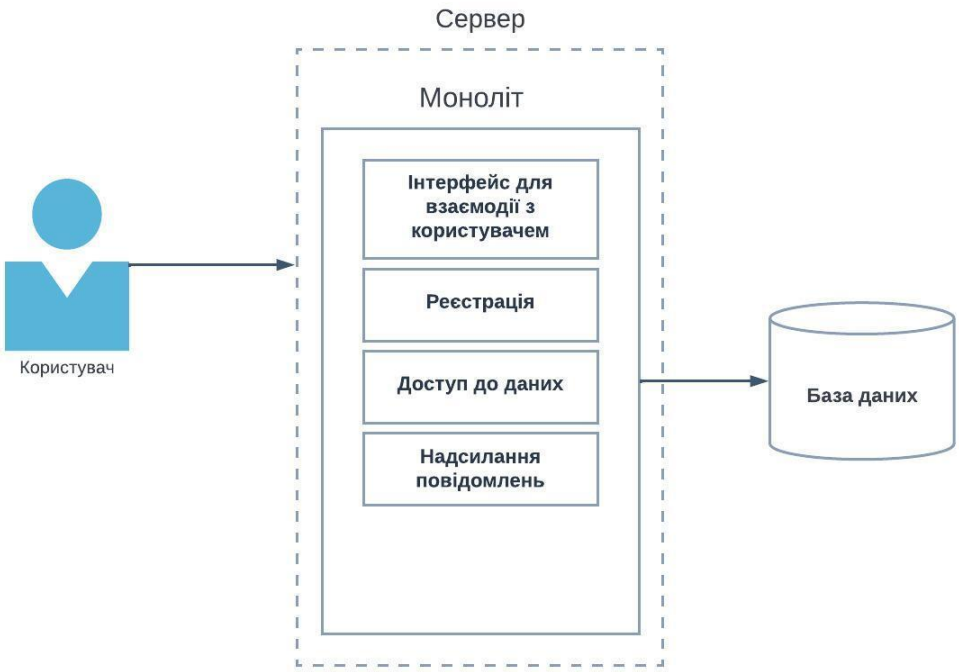


Рисунок 1.4 – Монолітна архітектура

Тепер перейдемо до мікросервісів. Вони розбивають додатки на більш дрібні автономні сервіси, які взаємодіють один з одним (рисунок 1.5).

Однією з головних переваг даної архітектури є масштабованість і гнучкість. Кожен мікросервіс можна масштабувати незалежно від інших, виходячи з його потреб.

Також для мікросервісів характерна технологічна різноманітність та автономія. Кожен сервіс працює окремо, так що можна підібрати для нього різні технології та фреймворки.

Ще однією перевагою є те, що сервіси не мають прив'язки один до одного, завдяки чому їх можна розробляти, тестувати та розгортати окремо.

Одним з недоліків мікросервісної архітектури є проблема з управлінням розподіленою системою. Для координації взаємодії між

сервісами, забезпечення узгодженості даних та обробки збоїв необхідне скрупульозне проектування.

Також слід враховувати витрати на координацію процесів. Чим більше сервісів стає, тим складніше підтримувати їхню взаємодію та узгодженість даних. Виходить, що для реалізації ефективних процесів взаємодії потрібні додаткові засоби (наприклад, API або черги повідомлень).

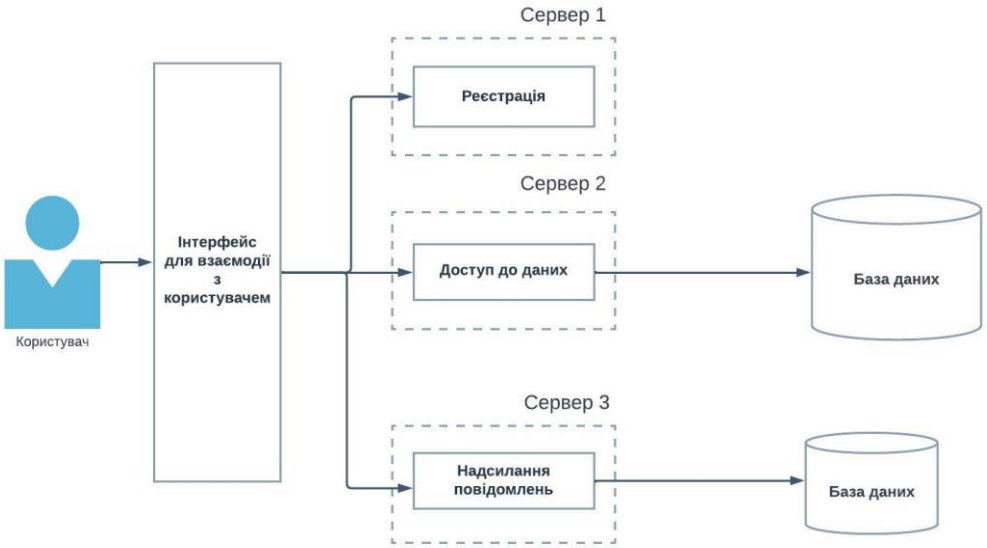


Рисунок 1.5 – Мікросервісна архітектура

Отже, монолітна архітектура пропонує простоту та ефективність; в ній використовується єдина база коду та загальні дані. Але у монолітах відзначаються явні проблеми з масштабованістю, розгортанням, обслуговуванням, а також обмежена технологічна різноманітність.

Мікросервісна архітектура пропонує масштабованість та технологічну різноманітність. Однак усе це пов'язано з низкою труднощів, пов'язаних зі складністю цієї архітектури, розподіленими системами та взаємодією між сервісами [8].

За необхідності раціонального використання ресурсів та вибору типу архітектури додатків на основі компетенцій команди розробників та масштабів створюваного програмного забезпечення було прийнято рішення використовувати монолітну архітектуру.

1.3 Опис бізнес-процесів

Далі наведемо опис послідовностей бізнес-процесів ПЗ та їх наочне представлення з використанням BPMN моделей.

Опис послідовності створення облікового запису користувача (рисунок 1.6):

– користувач переходить на сторінку реєстрації;
– користувач заповнює поля реєстрації: логін, пошта, пароль, підтвердження пароля та натискає кнопку «Зареєструватися»;
– якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відображається помилка;
– якщо дані в полях пароль та підтвердження пароля не співпадають, кнопка «Зареєструватися» не активна;
– якщо поля заповнені вірно, користувач переходить на сторінку входу.

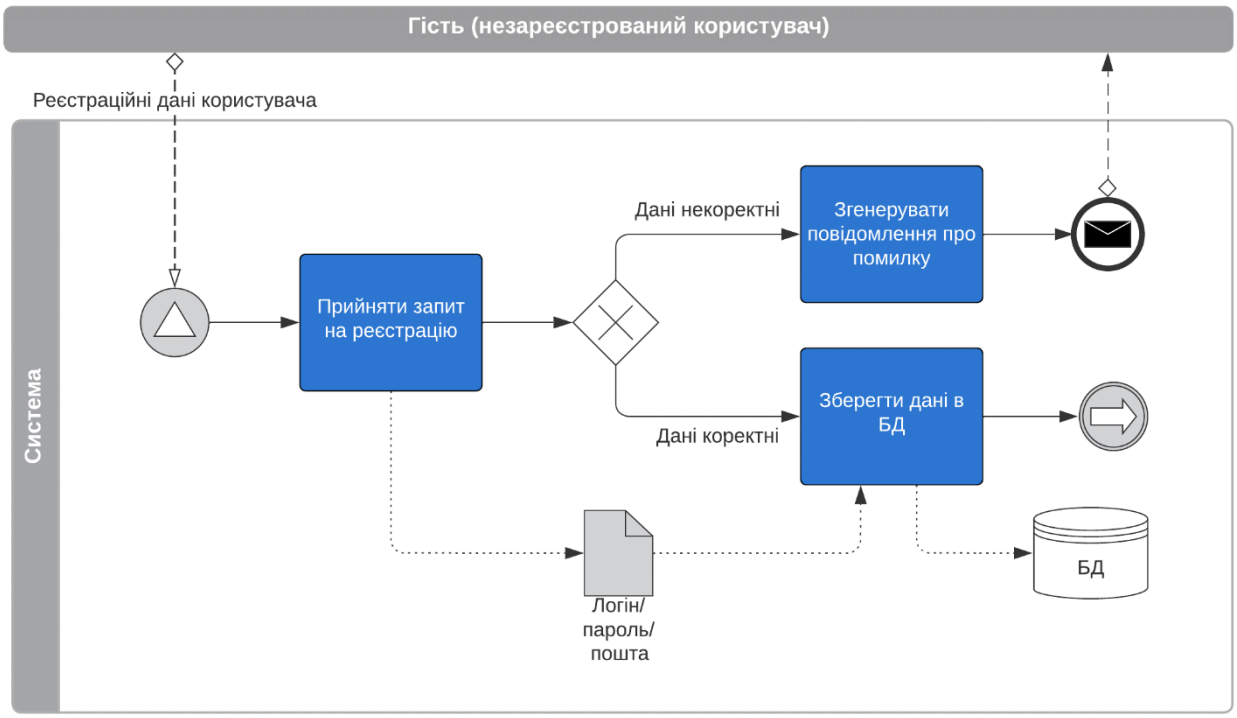


Рисунок 1.6 – Схема процесу реєстрації користувача

Опис послідовності входу в обліковий запис користувача (рисунок 1.7):

– користувач переходить на сторінку входу;

– користувач заповнює поля входу: електронна пошта і пароль, та натискає кнопку «Увійти»;
– якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відображається помилка;
– якщо поля заповнені вірно, користувач переходить на домашню сторінку.

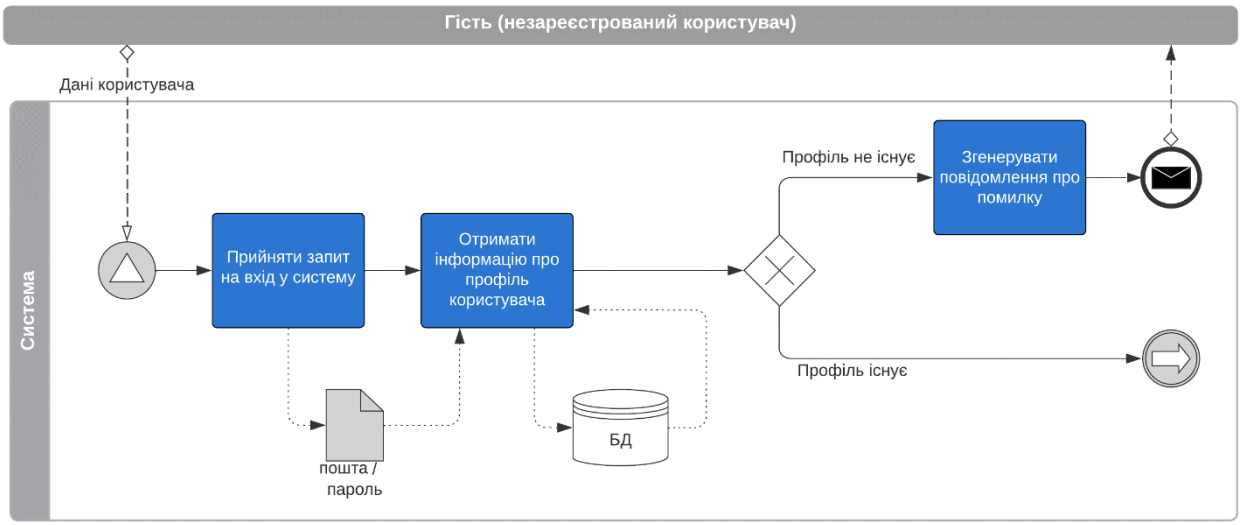


Рисунок 1.7 – Схема процесу входу користувача

Опис послідовності створення товару (рисунок 1.8):

– користувач переходить на сторінку створення товару;
– користувач заповнює відповідні поля: категорія, зображення, файли товару і документа, що підтверджує авторські права, назва, текстовий опис товару, ціна, та натискає кнопку «Зберегти»;
– якщо введені поля не відповідають шаблону заповнення, відображається помилка;
– якщо поля заповнені вірно, користувач бачить список створених ним товарів;
– якщо створений товар підтверджено адміністратором, він відображається на домашній сторінці;

– якщо створений товар не підтверджено, він не доступний для продажу.

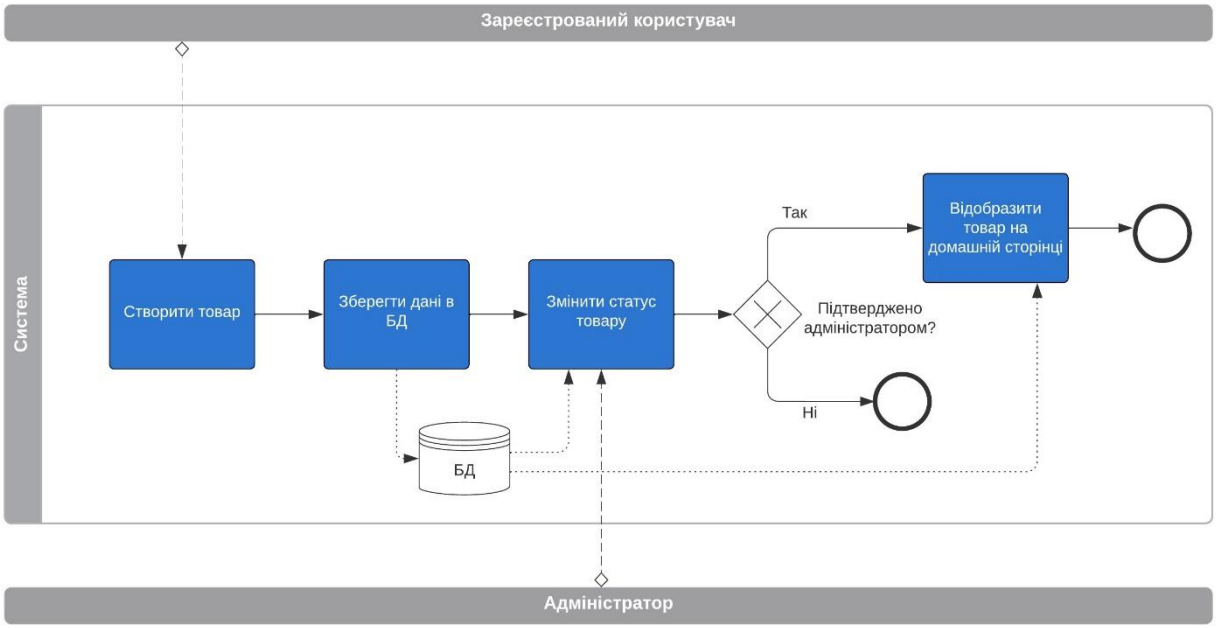


Рисунок 1.8 – Схема процесу створення продукту

Опис послідовності оновлення товару (рисунок 1.9):

– користувач переходить на сторінку товару;
– користувач натискає «Редагувати»;
– користувач заповнює відповідні поля: категорія, зображення, файл, назва, текстовий опис товару, ціна, та натискає кнопку «Зберегти»;
– якщо введені поля не відповідають шаблону заповнення, відображається помилка;
– якщо поля заповнені вірно, користувач бачить список створених ним товарів.

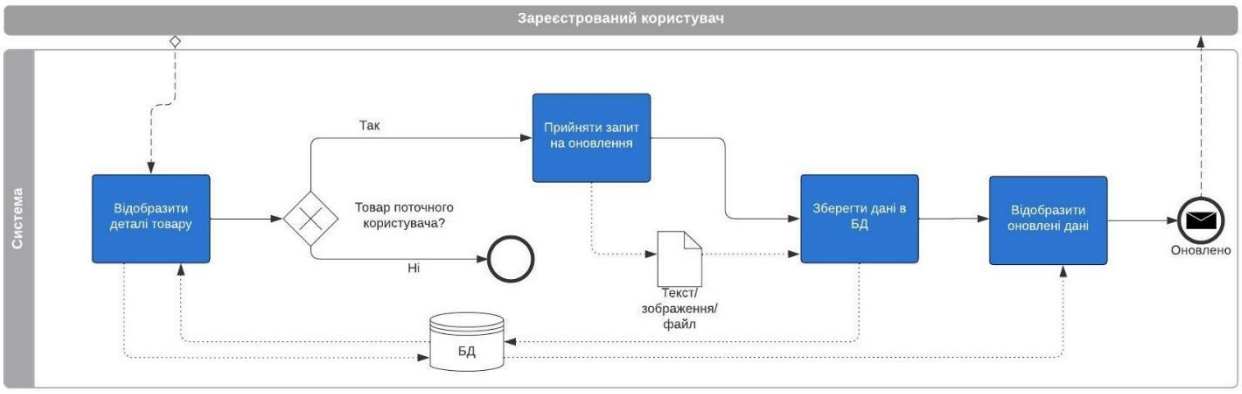


Рисунок 1.9 – Схема процесу оновлення продукту

Опис послідовності видалення товару (рисунок 1.10):

– користувач бачить список своїх товарів на домашній сторінці або на сторінці створених продуктів;
– користувач натискає кнопку «Видалити» біля обраного товару;
– користувач натискає кнопку «Підтвердити» для підтвердження видалення товару;
– обраний товар видаляється з домашньої сторінки та сторінки продуктів даного користувача.

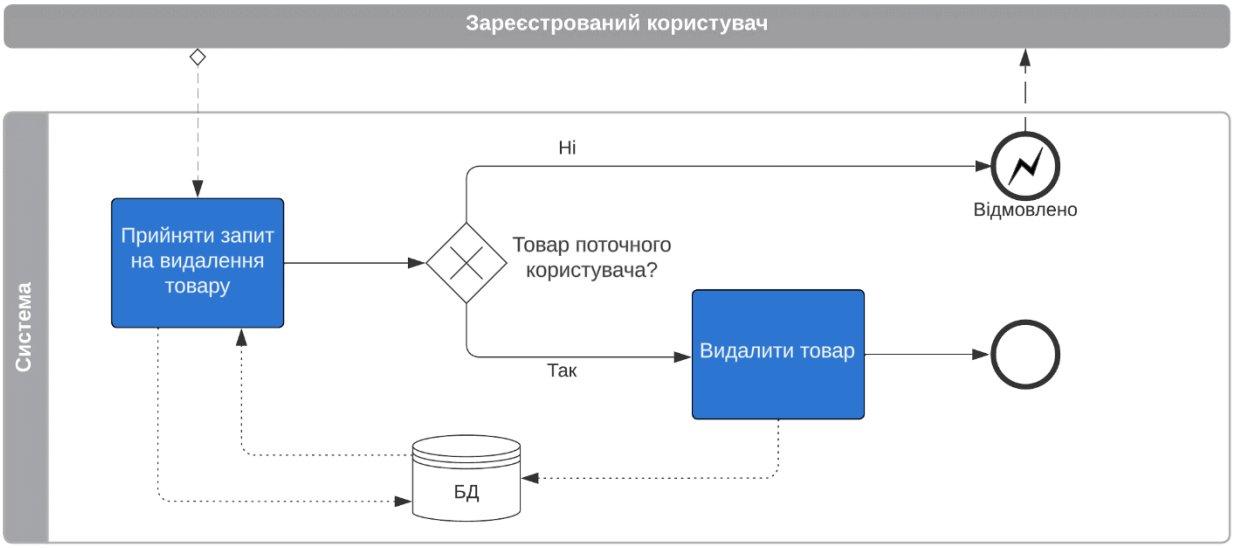


Рисунок 1.10 – Схема процесу видалення продукту

Опис послідовності додавання товару в обране (рисунок 1.11):

– користувач бачить список товарів інших продавців на домашній сторінці;

– користувач натискає кнопку «Додати до обраного» біля обраного товару;
– доданий товар відображається в списку обраного даного користувача;
– відображається кнопка «Видалити з обраного».

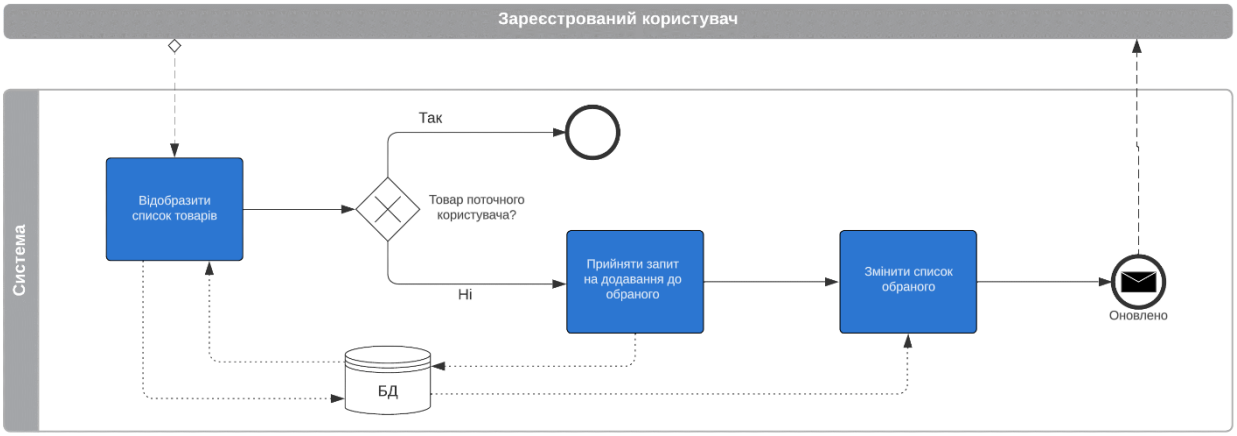


Рисунок 1.11 – Схема процесу додавання продукту в обране

Опис послідовності видалення товару з обраного (рисунок 1.12):

– користувач бачить список обраних товарів;
– користувач натискає кнопку «Видалити з обраного» біля обраного товару;
– доданий товар видаляється зі списку обраного даного користувача;
– відображається кнопка «Додати до обраного».

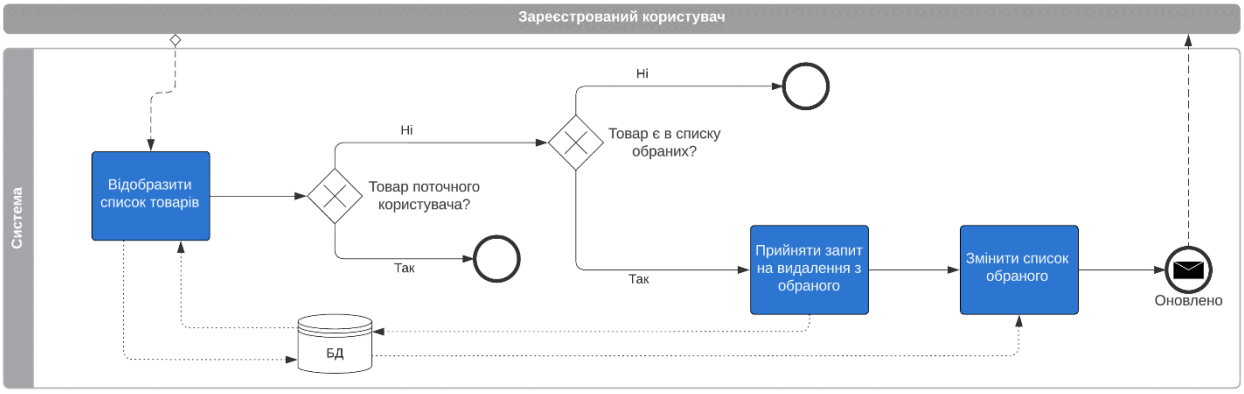


Рисунок 1.12 – Схема процесу видалення продукту в з обраного

Опис послідовності додавання товару в кошик (рисунок 1.13):

– користувач переходить на сторінку товару;

– користувач натискає кнопку «Додати в кошик»;
– якщо обраний товар вже доданий у кошик, виводиться відповідне сповіщення, стан кошику не змінюється;
– лічильник кількості товарів в кошику збільшується на 1;
– товар відображається в кошику.

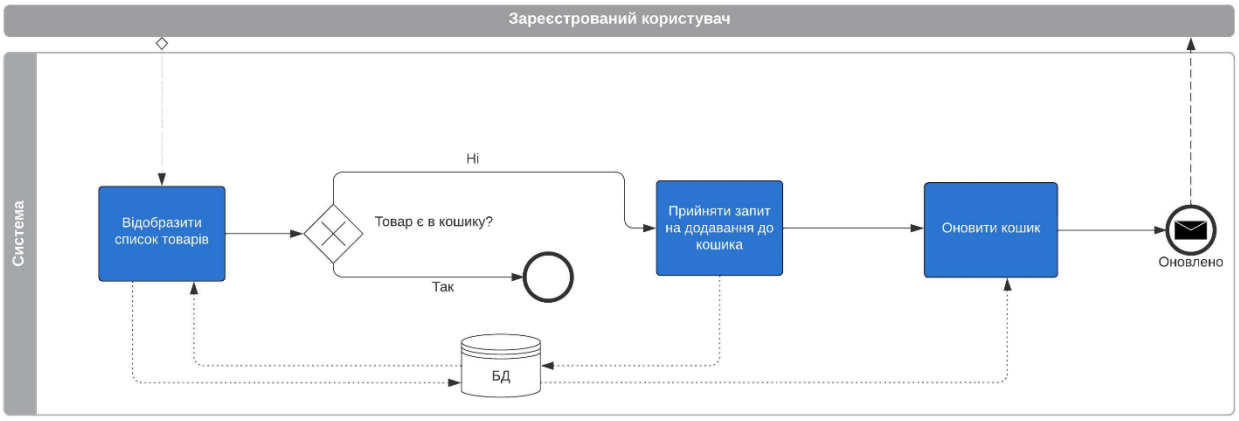


Рисунок 1.13 – Схема процесу додавання продукту в кошик

Опис послідовності видалення товару з кошика (рисунок 1.14):

– користувач переходить на сторінку кошика;
– користувач натискає кнопку «Видалити з кошика»;
– лічильник кількості товарів в кошику зменшується на 1;
– товар видаляється з кошика.

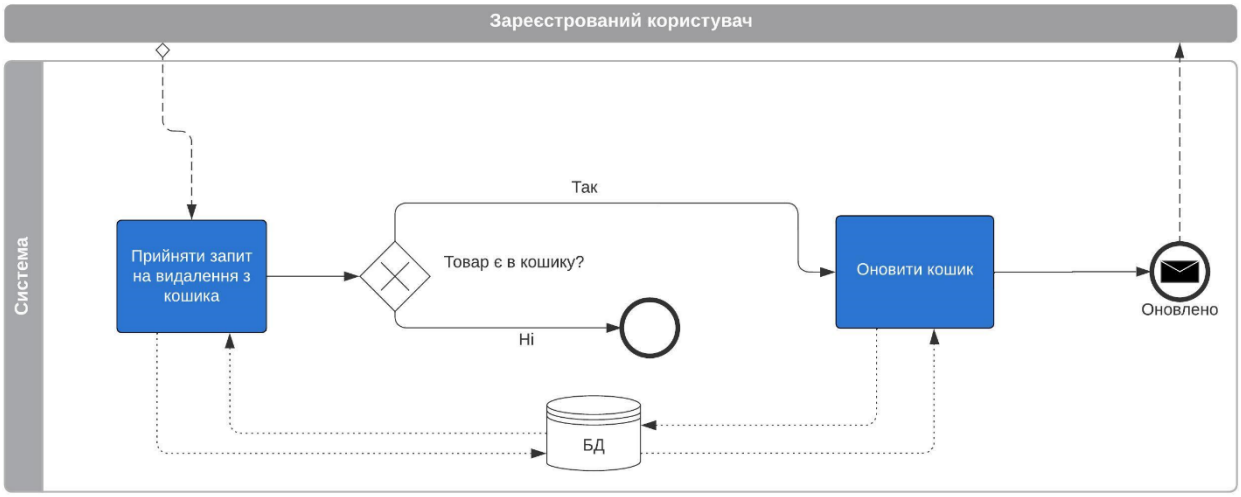


Рисунок 1.14 – Схема процесу видалення продукту з кошика

Опис послідовності здійснення пошуку товару (рисунок 1.15):

– користувач переходить на домашню сторінку;

– користувач вводить назву товару і натискає кнопку «Пошук»;
– товари, що відповідають запиту відображаються на головній сторінці.

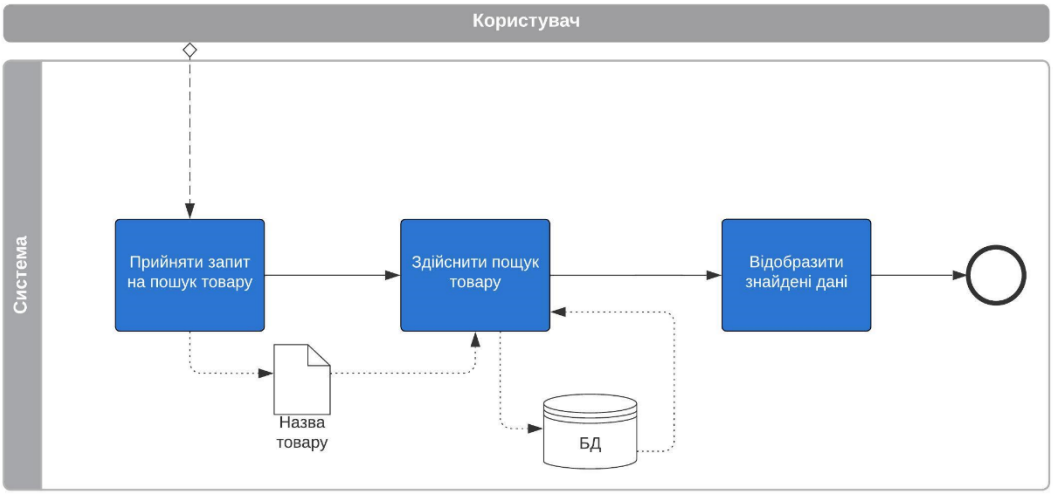


Рисунок 1.15 – Схема процесу пошуку продукту

Опис послідовності здійснення купівлі товару (рисунок 1.16):

– користувач переходить на сторінку кошика;
– користувач натискає кнопку «Оформити замовлення»;
– користувач перенаправляється на сторінку здійснення оплати;
– користувач вводить платіжні дані та натискає кнопку «Оплатити»;
– якщо оплата пройшла успішно, користувач отримує посилання на завантаження замовлення;
– якщо оплата невдала, виводиться сповіщення про помилку.

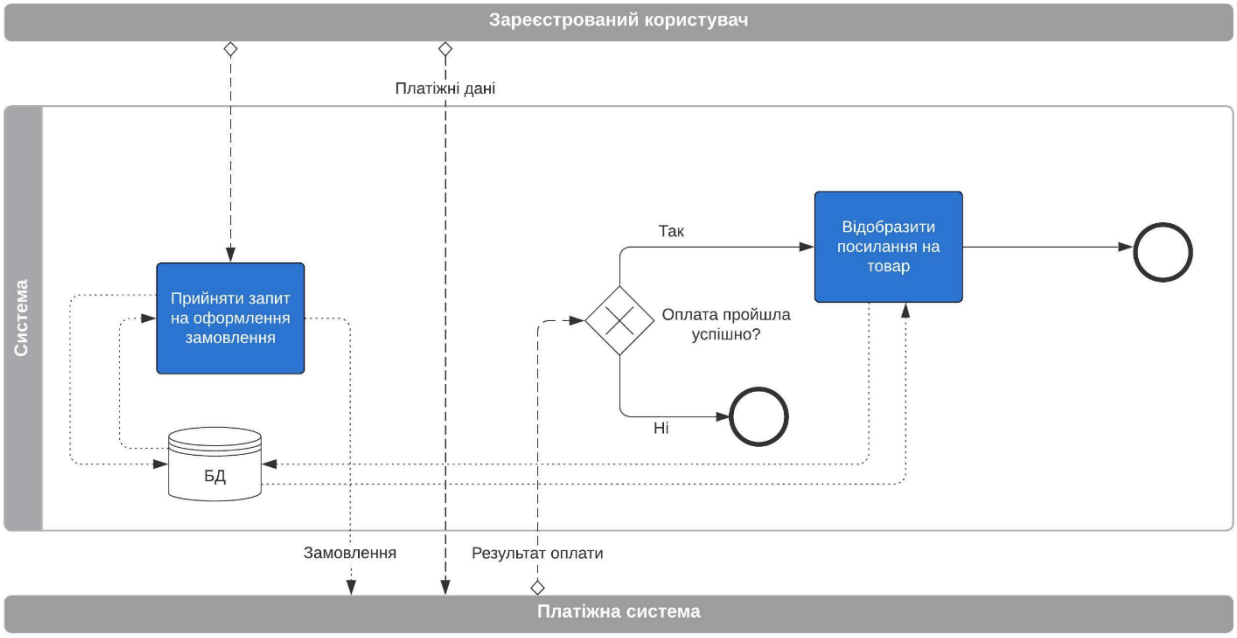


Рисунок 1.16 – Схема процесу купівлі продукту

Опис послідовності подачі скарги (рисунок 1.17):

– користувач переходить на сторінку товару;
– користувач натискає кнопку «Порушення авторських прав»;
– товар змінює статус та стає недоступним для продажу;
– адміністратор видаляє товар з системи.

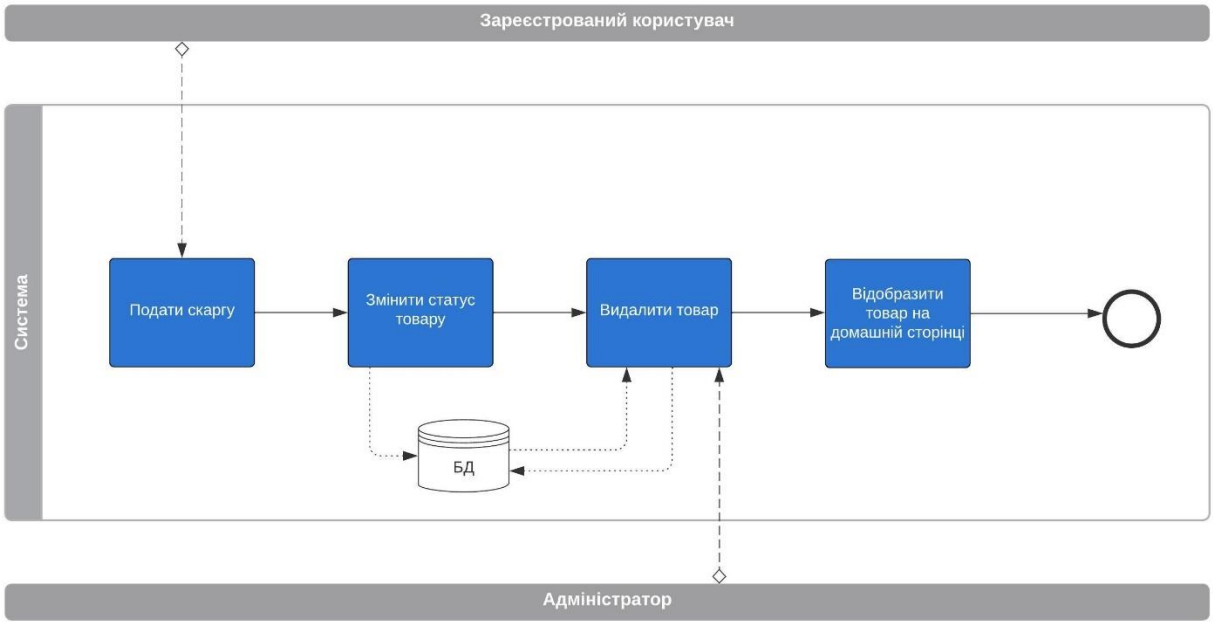


Рисунок 1.17 – Схема процесу подачі скарги

1.4 Постановка задачі

Метою розробки є створення програмного засобу, що буде використовуватися для купівлі та продажу цифрових продуктів. Запропоноване програмне забезпечення з електронної комерції покликане спростити процес купівлі та продажу цифрових продуктів, надаючи інструменти для ефективної взаємодії між продавцями та покупцями та гарантує однаковий доступ до інформації для всіх учасників.

Функціональність платформи охоплює пошук необхідного товару та інтерактивні каталоги, які надають можливість користувачам легко знаходити продукцію за визначеними параметрами. Це значно спрощує процес вибору та прийняття рішення для покупців. Таким чином, запропонований вебзастосунок сприяє більш ефективному розвитку ринку цифрових продуктів, робить його доступним та вигідним для всіх учасників.

Об'єктом дослідження є програмне забезпечення для електронної комерції.

Предметом дослідження є процеси розроблення, впровадження, супроводження і забезпечення якості програмного забезпечення.

Проаналізувавши існуючі рішення предметної області, з'ясувавши їхні недоліки та переваги, можна виділити наступні задачі в даному дипломному проекті:

1. розробити функціональні та нефункціональні вимоги до програмного забезпечення з урахуванням недоліків та переваг існуючих рішень;
2. розробити програмне забезпечення з наступними можливостями:
   2.1. авторизація;
   2.2. створення товарів;
   2.3. купівля товарів;
   2.5. додавання товарів у список обраного;
   2.6. здійснення пошуку товарів;
   2.7. сортування товарів за категоріями;

3. провести тестування створеного ПЗ та впевнитись в його якості;
4. виконати розгортання програмного забезпечення.

Висновки до розділу

У розділі 1 було представлено загальні положення про стан ринку електронної комерції в Україні та світі. Наведено визначення цифрових продуктів, переваги їх продажу на онлайн-ринках, такі як: однакові можливості для торгівлі для великих та невеликих фірм, цілодобова доступність, гнучкість. Як приклад представлено такі платформи, як Amazon, GooglePlay, Rozetka.

Майбутнім шляхом розвитку даної сфери було запропоноване збільшення кількості спеціалізованих маркетплейсів на локальному українському ринку. Успішне зростання даного напрямку може забезпечити розвиток об'єму внутрішнього IT-ринку, що загалом покращить рівня життя населення.

Було проведено порівняння монолітної та мікросервісної архітектур. Перевагами монолітної архітектури є те, що вона простіша і проектуванні, але складніша при подальшому масштабуванні. Мікросервісна ж архітектура краще забезпечує масштабованість та технологічну різноманітність, однак потребує більше часу на планування ефективної взаємодії між сервісами.

На основі зробленого порівняння та наявних для розробки ресурсів було прийнято рішення використовувати монолітну архітектуру.

Було представлено опис бізнес-процесів розробки у вигляді BPMN схем, покрокову послідовність кожного процесу.

Було визначено мету роботи: створення засобу для реалізації онлайн-торгівлі цифровими товарами, що посприяє більш ефективному розвитку ринку цифрових продуктів.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією додатку є забезпечення здійснення публікації своїх товарів для продавців та купівлі даних продуктів для споживачів. Більше функцій можна побачити на рисунку 2.1.

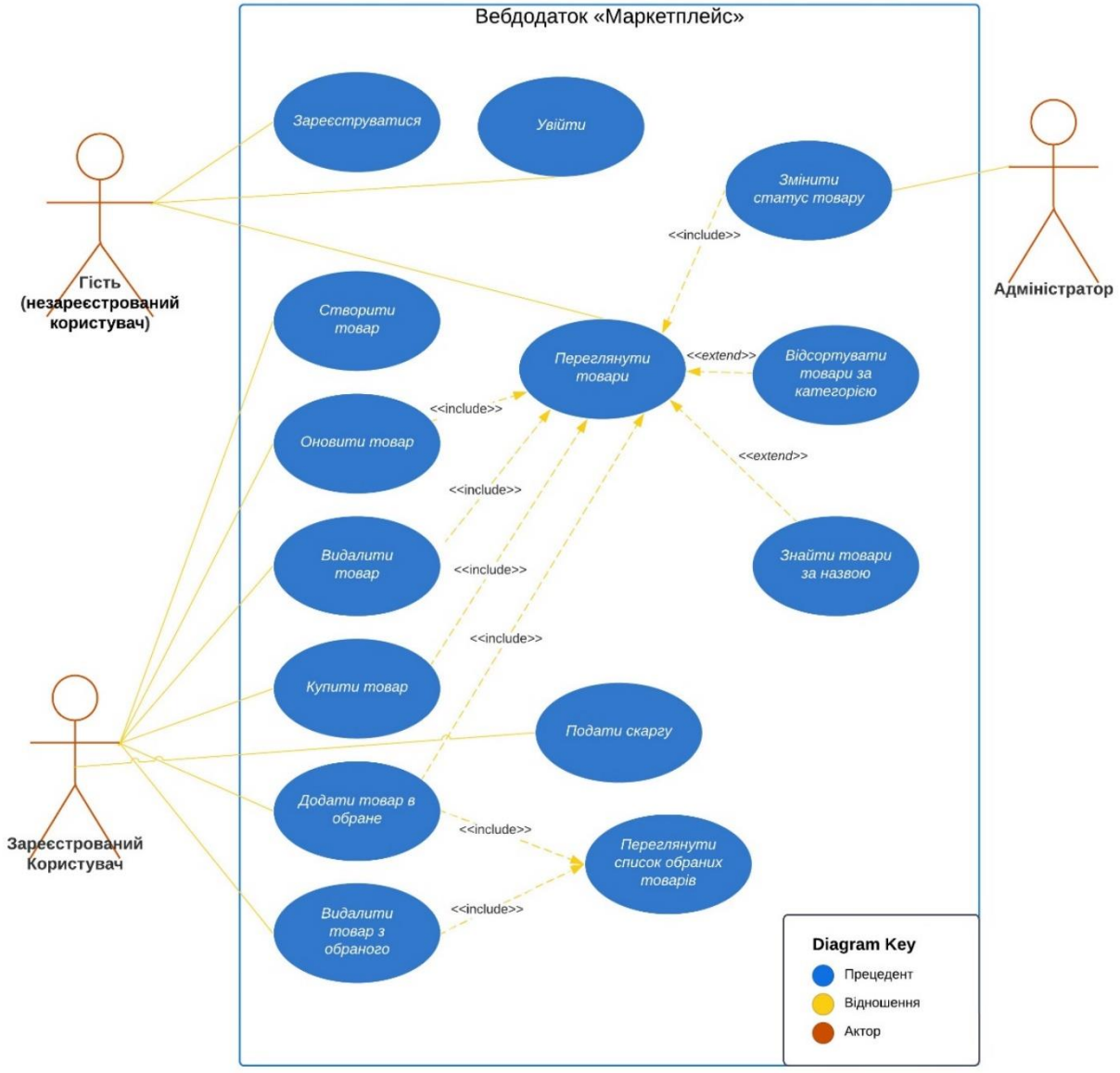


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1 - 2.8 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 - Варіант використання UC-1

Usecasename	Зареєструватися
Usecase ID	UC-01
Goals	Створення в системі облікового запису користувача.
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зареєструватися.
Pre-conditions	-
FlowofEvents	Користувач переходить на сторінку реєстрації. Користувач вводить логін, електронну пошту, пароль та повторно пароль для його підтвердження в поля для реєстрації. Після заповнення даних користувач натискає кнопку реєстрації і перенаправляється на сторінку входу.
Extension	У випадку введення некоректних даних виводиться повідомлення про помилку. Якщо дані в полях пароль та підтвердження пароля не співпадають, кнопка «Зареєструватися» стає неактивною.
Post-Condition	Створення облікового запису користувача, перехід на сторінку входу.

Таблиця 2.2 - Варіант використання UC-2

Usecasename	Увійти
Usecase ID	UC-02
Goals	Вхід користувача у свій обліковий запис в системі.
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає увійти в систему.
Pre-conditions	Користувач має обліковий запис, але вийшов із системи.
FlowofEvents	Користувач переходить на сторінку входу. Користувач заповнює поля входу: електронна пошта і пароль, та натискає кнопку «Увійти». Користувач переходить на домашню сторінку.
Extension	Якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відображається повідомлення про помилку.
Post-Condition	Вхід користувача у свій обліковий запис, перехід на домашню сторінку.

Таблиця 2.3 - Варіант використання UC-3

Usecasename	Створити товар
Usecase ID	UC-03
Goals	Додавання товару в систему поточним користувачем.
Actors	Зареєстрований користувач
Trigger	Користувач бажає створити товар в системі.
Pre-conditions	Користувач увійшов у свій обліковий запис.
FlowofEvents	Користувач переходить на сторінку створення товару. Користувач заповнює відповідні поля: категорія, зображення, файл товару та документ, що підтверджує авторські права, назва, текстовий опис товару, ціна, та натискає кнопку «Зберегти». Користувач бачить список створених ним товарів.
Extension	Якщо введені поля не відповідають шаблону заповнення, відображається помилка.
Post-Condition	Створений товар відображається на сторінці адміністратора зі статусом «Чекає підтвердження».

Таблиця 2.4 - Варіант використання UC-4

Usecasename	Оновити товар
Usecase ID	UC-04
Goals	Оновлення інформації про створений товар.
Actors	Зареєстрований користувач
Trigger	Користувач бажає оновити інформацію про свій товар.
Pre-conditions	Користувач увійшов у свій обліковий запис. Користувач створив даний товар.
FlowofEvents	Користувач переходить на сторінку товару. Користувач натискає «Редагувати». Користувач заповнює відповідні поля: категорія, зображення, файл, назва, текстовий опис товару, ціна, та натискає кнопку «Зберегти». Користувач бачить список створених ним товарів.
Extension	Якщо введені поля не відповідають шаблону заповнення, відображається помилка. Оновлюються лише ті поля, які були змінені користувачем.
Post-Condition	Інформацію про товар змінено. Користувач переходить на сторінку зі списком своїх товарів.

Таблиця 2.5 - Варіант використання UC-5

Usecasename	Видалити товар
Usecase ID	UC-05
Goals	Видалення із системи створеного користувачем товару.
Actors	Зареєстрований користувач
Trigger	Користувач бажає видалити обраний товар.
Pre-conditions	Користувач увійшов у свій обліковий запис, користувач створив обраний товар.
FlowofEvents	Користувач бачить список своїх товарів на домашній сторінці або на сторінці створених продуктів. Користувач натискає кнопку «Видалити» біля обраного товару. Користувач натискає кнопку «Підтвердити» для підтвердження видалення товару. Товар видаляється з домашньої сторінки та сторінки продуктів даного користувача.
Extension	Якщо користувач не створював даний товар, кнопка «Видалити» не відображається.
Post-Condition	Товар видаляється із системи та більше не відображається на домашній сторінці та на сторінці товарів цього користувача.

Таблиця 2.6 - Варіант використання UC-6

Usecasename	Купити товар
Usecase ID	UC-06
Goals	Здійснення купівлі користувачем товару.
Actors	Зареєстрований користувач
Trigger	Користувач бажає придбати обраний товар.
Pre-conditions	Користувач увійшов у свій обліковий запис.
FlowofEvents	Користувач переходить на сторінку кошика. Користувач натискає кнопку «Оформити замовлення». Користувач перенаправляється на сторінку здійснення оплати. Користувач вводить платіжні дані та натискає кнопку «Оплатити». Користувач переходить на сторінку з результатом оплати.
Extension	Якщо оплата невдала, виводиться сповіщення про помилку.
Post-Condition	Користувачеві надається посилання на придбаний товар.

Таблиця 2.7 - Варіант використання UC-7

Usecasename	Додати товар в обране
Usecase ID	UC-07
Goals	Додавання товару в список улюблених товарів користувача.
Actors	Зареєстрований користувач
Trigger	Користувач бажає зберегти даний товар в окремий список улюблених продуктів.
Pre-conditions	Користувач увійшов у свій обліковий запис. Даний товар не знаходиться в списку обраних.
FlowofEvents	Користувач бачить список товарів інших продавців на домашній сторінці. Користувач натискає кнопку «Додати до обраного» біля обраного товару. Доданий товар відображається в списку обраного даного користувача.
Extension	Якщо користувач створив даний товар, кнопка «Додати до обраного» не відображається.
Post-Condition	Користувач бачить доданий товар в списку улюблених товарів. Відображається кнопка «Видалити з обраного».

Таблиця 2.8 - Варіант використання UC-8

Usecasename	Видалити товар з обраного
Usecase ID	UC-08
Goals	Видалення товару зі списку улюблених товарів користувача.
Actors	Зареєстрований користувач
Trigger	Користувач бажає видалити даний товар зі списку улюблених продуктів.
Pre-conditions	Користувач увійшов у свій обліковий запис. Даний товар знаходиться в списку обраних. Відображається кнопка «Видалити з обраного».
FlowofEvents	Користувач бачить список обраних товарів. Користувач натискає кнопку «Видалити з обраного» біля обраного товару.
Extension	-
Post-Condition	Доданий товар видаляється зі списку обраного даного користувача. Відображається кнопка «Додати до обраного».

Таблиця 2.9 - Варіант використання UC-9

Use case name	Подати скаргу на товар
Use case ID	UC-09
Goals	Видалення товару зі списку доступних для продажу.
Actors	Зареєстрований користувач
Trigger	Користувач бажає подати скаргу на товар.
Pre-conditions	Користувач увійшов у свій обліковий запис. Користувач перейшов на сторінку товару.
Flow of Events	Користувач натискає кнопку «Порушення авторських прав» біля обраного товару. Товар більше не відображається на домашній сторінці. Товар відображається на сторінці адміністратора.
Extension	-
Post-Condition	Товар більше не відображається на домашній сторінці.

Таблиця 2.10 - Варіант використання UC-10

Use case name	Зміна статусу товару
Use case ID	UC-10
Goals	Підтвердження або відмовлення у можливості продажу товару.
Actors	Адміністратор
Trigger	Новий товар додано до системи або на товар подано скаргу.
Pre-conditions	Даний товар знаходиться на сторінці адміністратора.
Flow of Events	Адміністратор обирає статус продукту та натискає «Змінити статус». Якщо статус «Підтверджено», товар відображається на домашній сторінці і стає доступним для продажу.
Extension	-
Post-Condition	Статус товару змінено.

2.2 Розроблення функціональних вимог

У таблиці 2.11 наведено загальну модель вимог, а таблиця 2.12 містить опис функціональних вимог до ПЗ. Для візуалізації покриття розробки необхідним тестуванням використана матриця трасування вимог (таблиця 2.13).

Таблиця 2.11 – Загальна модель вимог

№	Назва	ID вимоги	Пріоритети	Ризики
1	Перегляд сторінки реєстрації	FR-1	Середній	Низький
2	Перегляд сторінки входу	FR-2	Середній	Низький
3	Реєстрація користувача	FR-3	Високий	Високий
4	Авторизація користувача	FR-4	Високий	Високий
5	Вихід із облікового запису	FR-5	Високий	Середній
6	Перегляд домашньої сторінки	FR-6	Середній	Низький
7	Пошук товарів за назвою	FR-7	Низький	Низький
8	Сортування товарів за категорією	FR-8	Низький	Низький
9	Створення товару	FR-9	Високий	Середній
10	Перегляд сторінки товару	FR-10	Низький	Низький
11	Оновлення товару	FR-11	Середній	Середній
12	Видалення товару	FR-12	Низький	Середній
13	Додавання товару в кошик	FR-13	Середній	Середній
14	Видалення товару з кошика	FR-14	Низький	Низький
15	Перегляд сторінки кошика	FR-15	Середній	Низький
16	Купівля товару	FR-16	Високий	Високий
17	Додавання товару в обране	FR-17	Низький	Низький
18	Видалення товару з обраного	FR-18	Низький	Низький
19	Подання скарги на товар	FR-19	Середній	Середній
20	Зміна статусу товару	FR-20	Високий	Низький

Таблиця 2.12 – Перелік функціональних вимог

ID вимоги	Назва та опис
FR-1	Перегляд сторінки реєстрації. Користувач бачить сторінку реєстрації. На ній він може здійснити реєстрацію або перейти на сторінку входу.

Продовження таблиці 2.12

FR-2	Перегляд сторінки входу. Користувач бачить сторінку реєстрації. На ній він може здійснити авторизацію або перейти на сторінку реєстрації.
FR-3	Реєстрація користувача. Система повинна надавати можливість реєстрації користувачеві шляхом введення логіну, пошти, пароля.
FR-4	Авторизація користувача. Система повинна надавати можливість входу в систему користувачеві шляхом введення логіну і пароля.
FR-5	Вихід із облікового запису. Система повинна надавати користувачеві можливість вийти з облікового запису.
FR-6	Перегляд домашньої сторінки. Користувач бачить домашню сторінку додатку. На ній він може переглядати список існуючих товарів, їхню назву, ціну, автора та категорію. Можливі варіанти відображення товарів, відфільтрованих за назвою або категорією.
FR-7	Пошук товарів за назвою. Система повинна надавати користувачеві можливість переглядати на домашній сторінці товар за введеною користувачем назвою, якщо даний товар існує в системі.
FR-8	Сортування товарів за категорією. Система повинна надавати користувачеві можливість переглядати на домашній сторінці товар лише обраної категорії.
FR-9	Створення товару. Система повинна надавати користувачеві можливість публікування товару шляхом введення його назви, текстового опису, ціни та додавання зображення і файлу. Разом з файлом має бути завантажено документ, що підтверджує авторські права.

Продовження таблиці 2.12

FR-10	Перегляд сторінки товару. Користувач бачить сторінку товару. На ній він може переглядати деталі даного товару: назву, текстовий опис, категорію, автора, ціну. Тут же можливо додати товар у кошик.
FR-11	Оновлення товару. Система повинна надавати можливість оновлення інформації про товар, раніше створений користувачем.
FR-12	Видалення товару. Система повинна надавати зареєстрованому користувачеві можливість видалення створеного ним товару.
FR-13	Додавання товару в кошик. Система повинна надавати зареєстрованому користувачеві можливість додавання товару, який він бажає придбати, в кошик.
FR-14	Видалення товару з кошика. Система повинна надавати зареєстрованому користувачеві можливість видалення товару з кошика.
FR-15	Перегляд сторінки кошика. Зареєстрований користувач бачить сторінку кошика. На ній він може переглядати список товарів в кошику, їх назву, категорію, автора, ціну, загальну ціну. Тут він може перейти на сторінку здійснення оплати замовлення.
FR-16	Купівля товару. Система повинна надавати зареєстрованому користувачеві можливість перейти до платіжної системи, щоб оплатити замовлення. Після успішної оплати він може завантажити придбаний товар.
FR-17	Додавання товару в обране. Система повинна надавати зареєстрованому користувачеві можливість додавати товар в список улюблених товарів.

Продовження таблиці 2.12

FR-18	Видалення товару з обраного. Система повинна надавати зареєстрованому користувачеві можливість видалити товар зі списку улюблених товарів.
FR-19	Подання скарги на товар. Система повинна надавати користувачеві можливість подати скаргу адміністратору на порушення товаром авторських прав.
FR-20	Зміна статусу товару. Система повинна надавати користувачеві з правами адміністратора можливість змінити статус товару.

Таблиця 2.13 – Матриця трасування вимог

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10
FR-1	+									
FR-2		+								
FR-3	+									
FR-4		+								
FR-5	+	+								
FR-6			+	+	+		+			
FR-7			+	+	+		+			
FR-8			+	+	+					
FR-9			+							
FR-10			+	+	+	+				
FR-11				+						
FR-12					+					
FR-13						+				
FR-14						+				
FR-15						+				
FR-16						+				
FR-17							+			
FR-18								+		
FR-19			+						+	+
FR-20			+						+	+

2.3 Розроблення нефункціональних вимог

Задоволення програмним забезпеченням нефункціональних вимог допомагає гарантувати, що додаток відповідає потребам користувачів та

здатен виконувати призначенні функції. Вони визначають наскільки ефективно ПЗ реалізовуватиме функціональні вимоги.

Основними нефункціональними вимогами до додатку, що розробляється, є:

- безпека – система має бути захищена від несанкціонованого доступу;
- стійкість до навантажень, якщо велика кількість користувачів намагаються зайти водночас;
- простота в технічному обслуговуванні та оновленні;
- кросбраузерність – вебдодаток має однаково відображатися і працювати в різних браузерах: Chrome, Edge, Opera;
- надійність – система має працювати без помилок та збоїв;
- юзабіліті – система має представляти користувачам легкий і простий для використання інтерфейс;
- відповідність – програмне забезпечення має відповідати нормам чинного законодавства.

Висновки до розділу

У розділі 2 було визначено основні функції додатку у вигляді діаграми варіантів використання. Загальною функцією було визначено забезпечення здійснення публікації своїх товарів для продавців та купівлі даних продуктів для споживачів. Також функціонал вебзастосунку передбачає пошук, сортування та додавання товарів до списку обраних (вішліст).

Було представлено перелік функціональних вимог до вебзастосунку. Для візуалізації покриття розробки тестами була використана матриця трасування вимог.

Також було наведено список нефункціональних вимог, серед яких: безпека, кросбраузерність, простота у використанні.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

При розроблений застосунку було використано концепцію клієнт-серверного типу архітектури. Її загальний абстрактний вигляд наведено на рисунку 3.1.

Основними елементами архітектури клієнт-сервер є:

1) Клієнт (frontend). Його призначення – надати користувачу зручний інтерфейс для взаємодії з сервером. Він здатен використовувати специфічні для браузера функціональні можливості, забезпечувати інтерактивність додатку та обробку подій.

2) Сервер (backend). Його основне завдання – обробка запитів від клієнта. Цей компонент використовується для тих частин програми, які не вимагають прямої взаємодії з користувачем, наприклад підключення до бази даних.

3) База даних. Сюди записуються усі дані, якими оперує застосунок.

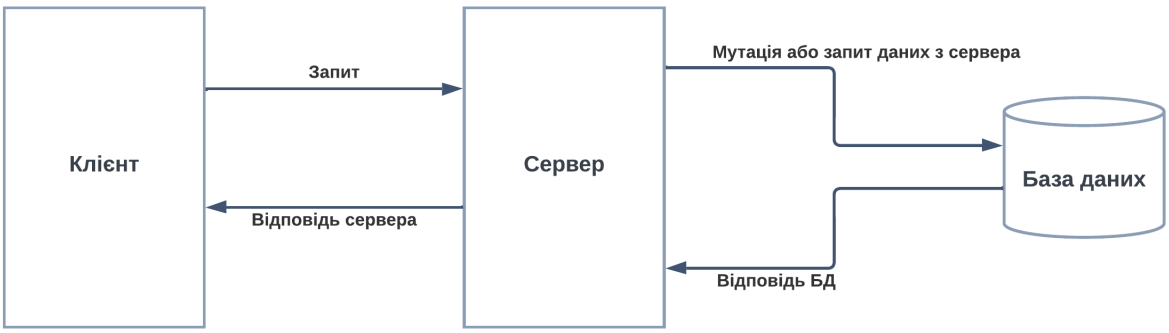


Рисунок 3.1 – Клієнт-серверна архітектура

Користувач на стороні клієнта робить запит на отримання інформації з сервера. Сервер отримує запит і перевіряє, чи має користувач права на отримання запитуваних даних. Завдяки цьому ми не можемо користуватися обліковими записами своїх друзів у соціальних мережах або отримувати інформацію про банківські рахунки інших людей. Після перевірки, сервер

надсилає запит до бази даних. Там він виконується, і відповідь повертається спочатку до серверу, потім надсилається клієнту [9].

Клієнт-серверна архітектура дозволяє знизити навантаження на клієнтські пристрої, оскільки вся бізнес-логіка виноситься на сервер [10].

Схема створеної архітектури для веб-додатку зображена на діаграмі компонентів (рисунок 3.2).

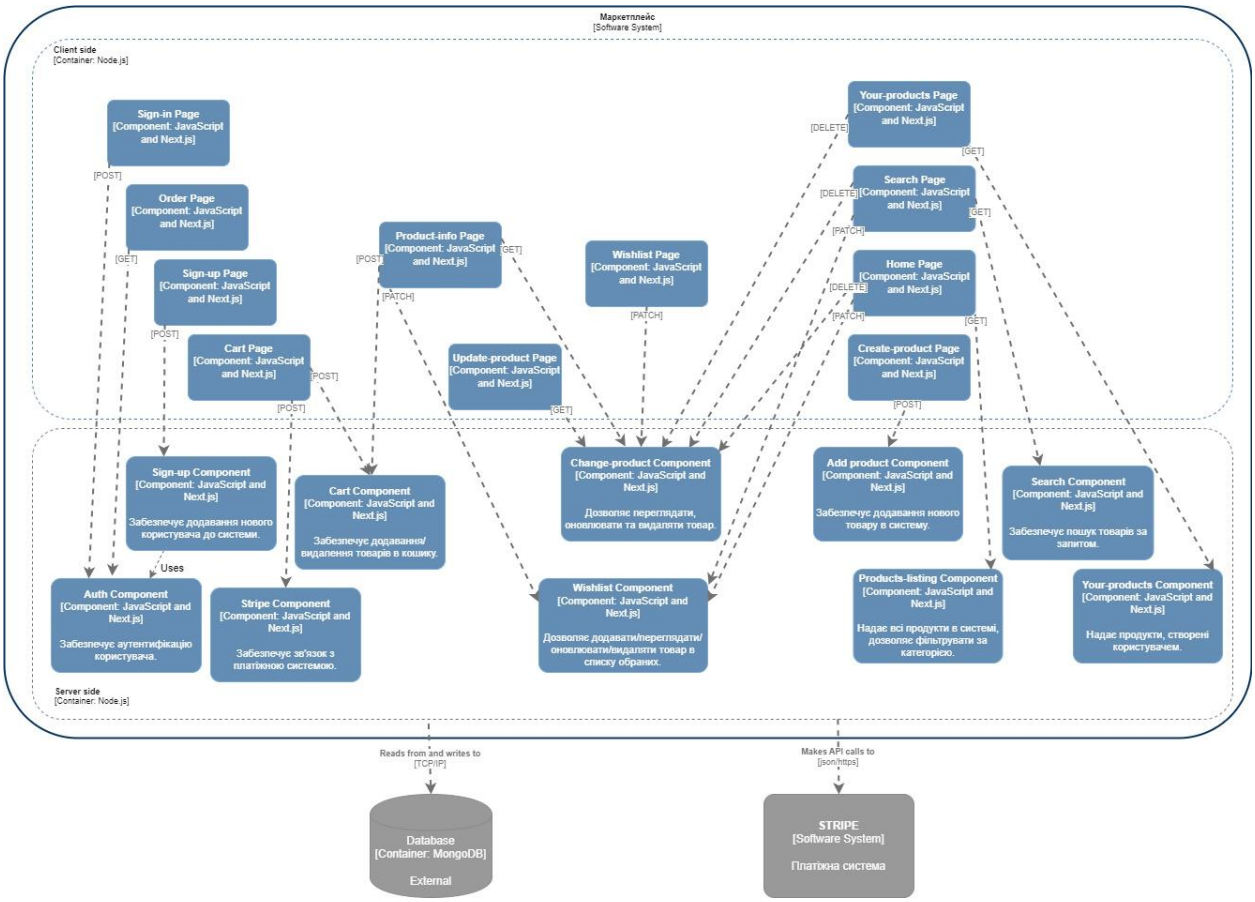


Рисунок 3.2 – Схема архітектури застосунку «Маркетплейс»

Для створення нових облікових записів була використана однофакторна аутентифікація. Користувачам пропонується ввести унікальну комбінацію електронної пошти та пароля для входу в систему.

Для зберігання даних було вирішено використовувати хмарне сховище MongoDBAtlas.

Надійність — один із головних пріоритетів у роботі з будь-якими даними. Хмарні бази даних (DBaaS) є досить надійними. DBaaS розгортаються в дата-центрах, захищених від зовнішніх атак та знеструмлення. Як правило, на випадок виходу з ладу обладнання завжди

готова репліка, щоб користувач не помітив жодних змін. Всі DBaaS пропонують послуги резервного копіювання, щонайменше щотижневі повні бекапи входять у будь-який тариф абсолютно безкоштовно.

DBaaS дозволяють повністю зосередитися на проекті, не відволікаючись на підготовку серверного обладнання, розгортання, налаштування та подальше адміністрування бази даних. Провайдер інфраструктури бере на себе всі налаштування, що покликані забезпечити безпеку та безперебійну роботу. Таким чином, DBaaS дозволяє просто користуватися інструментом, не витрачаючи час на його обслуговування [11].

В якості платіжної системи було обрано Stripe. Stripe - сервіс, який допомагає оптимізувати процес здійснення оплати за допомогою пластикових карток. Stripe вирішує всі проблеми платежів, включаючи зберігання даних карток, періодичні платежі та виплати на банківський рахунок. У Stripe висока репутація у світі Online-бізнесу, в першу чергу через високу надійність і безпеку, а також завдяки гарній документації та легкості інтеграції з сайтами[12].

3.2 Обґрунтування вибору засобів розробки

Далі розглянемо перелік засобів розробки, які можуть бути використані для створення вебдодатку, та оберемо з них ті, які підходять для конструювання нашого програмного забезпечення.

3.2.1 JavaScript для веброзробки

JavaScript - це легка, інтерпретована, прототипно-орієнтована мова з динамічною типізацією. Найбільш широке застосування вона знаходить як мова сценаріїв веб-сторінок, але також використовується як мова загального призначення (у тому числі для розробки на стороні сервера) на базі програмної платформи Node.js.

До головних особливостей цієї мови програмування належать:

1. Динамічна типізація. Тобто тип даних визначатиметься лише тоді, коли змінній присвоїться значення.

2. Гнучка робота із функціями. У JS функції можна не тільки виконувати, але ще й повертати функції з функцій, передавати функції як параметри іншим функціям і надавати функції як значення змінних.

3. JavaScript підтримується усіма сучасними браузерами.

Крім того, важливою особливістю JavaScript є його розвинена інфраструктура. На сьогоднішній день розробники можуть працювати з великою кількістю допоміжних бібліотек та фреймворків (найпопулярнішими з них є React, Angular та Vue).

Що стосується сфер застосування, в першу чергу, мова JavaScript широко використовується у веб-розробці. За допомогою JS можна написати будь-які веб-додатки і це значно підвищує попит програмістів, які володіють цією мовою. Популярність JavaScript зумовлена безліччю факторів. Наприклад: незамінність при розробці веб-сайтів та додатків, висока швидкість роботи та продуктивність, велика кількість інструментів та розвинена інфраструктура, відносна простота та легкість вивчення [13].

3.2.2 ASP.NET для веброзробки

ASP.NET - це фреймворк для розробки вебзастосунків, створений корпорацією Microsoft. У світі веброзробки ASP.NET займає важливе місце, забезпечуючи розробникам інструменти для створення масштабних та сучасних додатків.

В ASP.NET реалізовано модель об'єктно-орієнтованого програмування. Завдяки цьому вона дає розробнику доступ до всіх об'єктів у .NET Framework.

Перевагою даного фреймворку є Інтеграція з VisualStudio. Тісна взаємодія з цим середовищем розробки забезпечує інтуїтивність, багато доступних інструментів, та автоматизацію, що спрощує створення та налагодження вебдодатків.

З іншого боку, якщо сайт розробляється без використання VisualStudio, кожна сторінка компілюється окремо під час відправлення запиту. Це дещо знижує продуктивність невеликих сайтів[14].

3.2.3 Node.jsдля веброзробки

З огляду на описані вище переваги та недоліки ASP.NET та JavaScript, було прийнято рішення реалізовувати проект на мові JavaScript. Платформою для запусків клієнт-серверних додатків на JavaScript є Node.js.

Node.js — це програмне середовище, яке дозволяє запускати програми, написані на мові Javascript, поза браузером. Історично програми, написані на Javascript, на відміну від інших мов програмування, можна було запустити тільки в браузерах.

При розробці Node.js за основу був взятий рух виконання JavaScript під назвою V8, який був створений компанією Google і використовувався в браузері GoogleChrome.

За допомогою Node.js можна написати не тільки фронтенд, але й серверну частину вебдодатків. Тобто, це означає, що для розроблення всього вебзастосунку можна використовувати єдиний «стек», що робить процес програмування та обслуговування набагато швидшими і легшими.

Важливою особливістю Node.js є асинхронність. Сервер, створений з використанням Node.js, не повинен чекати, поки дані повернуться, при виконанні різних внутрішніх запитів. При цьому він також має неблокуючий механізм введення-виведення. Тоюто кілька різних процесів можуть виконуватись паралельно, не блокуючи один одного. Обидві ці властивості роблять Node.js швидким і допомагають розробити гнучкий та динамічний інтерфейс для користувача.

Node.js є дуже перспективною технологією, адже її використовують багато відомих компаній, таких як Netflix, LinkedIn та інші [15].

3.2.4 Angular для веброзробки

Наступним етапом є вибір фреймворку для розробки додатку.

Angular – це фреймворк від компанії Google для створення односторінкових веб-додатків (SPA) мовами програмування TypeScript та JavaScript.

Angular допомагає прив'язувати компоненти програми один до одного, передавати дані, анімувати інтерфейси та ін. Для простих проектів його функціональність може бути надмірною, але для складних SPA-додатків вона незамінна.

Фреймворк дозволяє створювати не лише вебзастосунки. З його допомогою можна писати код, який може бути адаптований до іншого середовища. Наприклад, програма зможе працювати в мобільній або десктопній операційній системі. За допомогою Angular навіть можна створити програму для доповненої реальності.

Тим не менш, Angular вважається одним із найскладніших фронтенд-фреймворків. Його нелегко вивчити з нуля самостійно. Крім того, для початку роботи потрібно знати не тільки "чистий" JavaScript, але і TypeScript, який на ньому заснований [16].

3.2.5 Next.jsдля веброзробки

Next.js - потужний JavaScript-фреймворк та новаторське рішення для створення сучасних веб-додатків на основі React, створеного компанією Vercel.

Next.js пропонує можливості попереднього рендерингу сторінок на стороні сервера, що дозволяє прискорити завантаження та покращити SEO за рахунок передачі HTML-контенту безпосередньо до браузера. Фреймворк автоматично розділяє код на дрібніші, керовані частини. Це дозволяє завантажувати лише необхідні компоненти, скорочуючи час початкового завантаження сторінки та підвищуючи її продуктивність. Він легко

інтегрується з різними бібліотеками, що спрощує стилізацію та позбавляє необхідності використання додаткових налаштувань та інструментів.

Створення роутів у додатку Next.js не потребує особливих зусиль. Структура папок та файлів визначає шляхи доступу клієнтської частини до компонентів програми. Також Next.js підтримує ключові слова у назвах файлової системи для поділу поведінки та контексту сервера та клієнта. Ця можливість полегшує інтеграцію функціональності бекенда з фронтендом, забезпечуючи динамічну роботу з даними.

Заважаючи на наведені переваги, легку оптимізацію продуктивності і простоту інтеграція з сучасними технологіями, було вирішено використовувати Next.js при розробці дипломного проекту [17].

3.2.6 OracleDatabase для роботи з даними

OracleDatabase - це об'єктно-реляційна система управління базами даних (СУБД) від компанії Oracle. Вона використовується для створення структури бази даних, її наповнення, редагування та відображення інформації.

OracleDatabase забезпечує створення резервних копій бази даних та відновлення даних. Завдяки їй можна легко відновити базу даних до потрібного стану. Проте для цього знадобиться додаткове місце для зберігання та архіваційні механізми.

OracleDatabase забезпечує високу швидкість роботи та здатна обробляти великі бази даних.

Тим не менш, одним із великих недоліків OracleDatabase є її складність. Oracle не рекомендується використовувати без достатніх технічних знань і навичок роботи з OracleDatabase. Oracle також не підійде для тих, хто шукає просту у використанні та основних функціях базу даних.

Oracle зазвичай вимагає більше зусиль для керування деякими операціями.

Таким чином, OracleDatabase корисна лише під час роботи з великими базами даних. Для малих та середніх компаній, де потрібні невеликі бази даних, Oracle не рекомендується[18].

3.2.7 MongoDB для роботи з даними

Для здійснення розробки була обрана NoSQL-база даних MongoDB.

MongoDB - це документоорієнтована система управління базами даних, яка зберігає дані у вигляді JSON-документів. В основі її роботи лежать концепції документів та колекцій.

Вона може працювати на невеликому об'ємі пам'яті, динамічно виділяючи та вивільняючи оперативну пам'ять залежно від потреб інших процесів. Крім того, MongoDB вважається найкращою NoSQL-базою даних завдяки своїй високій продуктивності та доступності.

Також, її перевагами є повнофункціональна мова запитів і документо-орієнтованість. У MongoDB для ефективного виконання запитів використовуються вторинні індекси (Indexes), які дозволяють упорядкувати дані щодо певного поля, що згодом прискорить пошук.

Всі дані MongoDB у сховищі (або взагалі в операційній системі) зашифровуються. Це дає гарантію, що лише автентифіковані процеси матимуть доступ до захищених даних.

До стандартних застосувань для MongoDB відносяться каталоги товарів, мобільні додатки, управління інформаційними ресурсами, персоналізація в реальному часі, а також додатки, що забезпечують індивідуальне подання даних у декількох системах[19].

3.3 Конструювання програмного забезпечення

Спершу розробимо сторінки та компоненти сторінок додатку.

Використовуючи Next.js ми можемо інтегрувати клієнтські та серверні компоненти в одному додатку.

Запропонована фреймворком структура забезпечує маршрутизацію за файловою системою програми. Стартовою точкою є папка зі спеціальною назвою App, всередині якої кожна папка є маршрутом у навігації програми. Кожен компонент-сторінка називається зарезервованим словом page. Такий підхід до організації роутів підвищує продуктивність праці розробників, адже, в порівнянні з типовою React-розробкою, зникає необхідність вручну створювати та прописувати роутинг у компонентах, що до того ж покращує читабельність коду.

Спроектуємо динамічну маршрутизацію за допомогою дужок [] в імені папки. Це дозволить створювати параметризовані веб-адреси, що будуть використовуватися для переходу на сторінку обраного продукту, виведення результатів пошуку товарів. Повну систему роутингу можна побачити на рисунку 3.3.

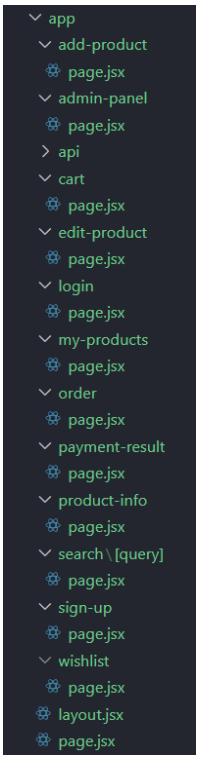


Рисунок 3.3 – Структура сторінок додатку

Оскільки Next.js працює на платформі Node.js, RouteHandlers дозволяють створювати API ендпоінти. Для цього ми розмістимо файли в папці app/api. Виклик ендпоінтів також підпорядковується правилу файлової системи, де адреса визначається назвами папок з логікою API (рисунок 3.4).

Ця логіка буде існувати виключно на стороні сервера і не збільшить розмір клієнтської програми. Ця можливість спрощує розробку бекенда та забезпечує безперешкодну взаємодію між клієнтом та сервером.

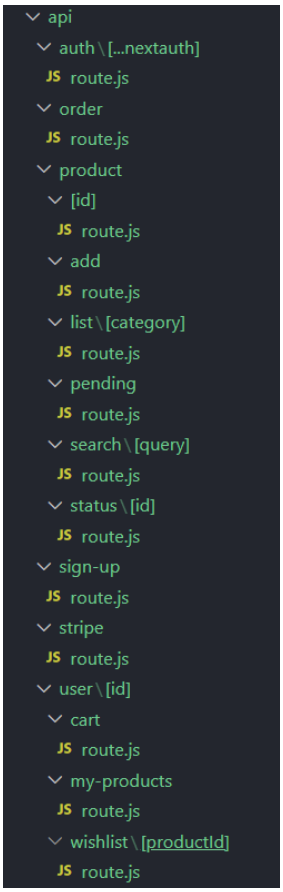


Рисунок 3.4 – Маршрутизація в додатку

В якості бази даних використовується MongoDB. Опис колекцій та їх поля наведено у таблицях 3.1 – 3.4. ER діаграма сутностей наведена на рисунку 3.5.

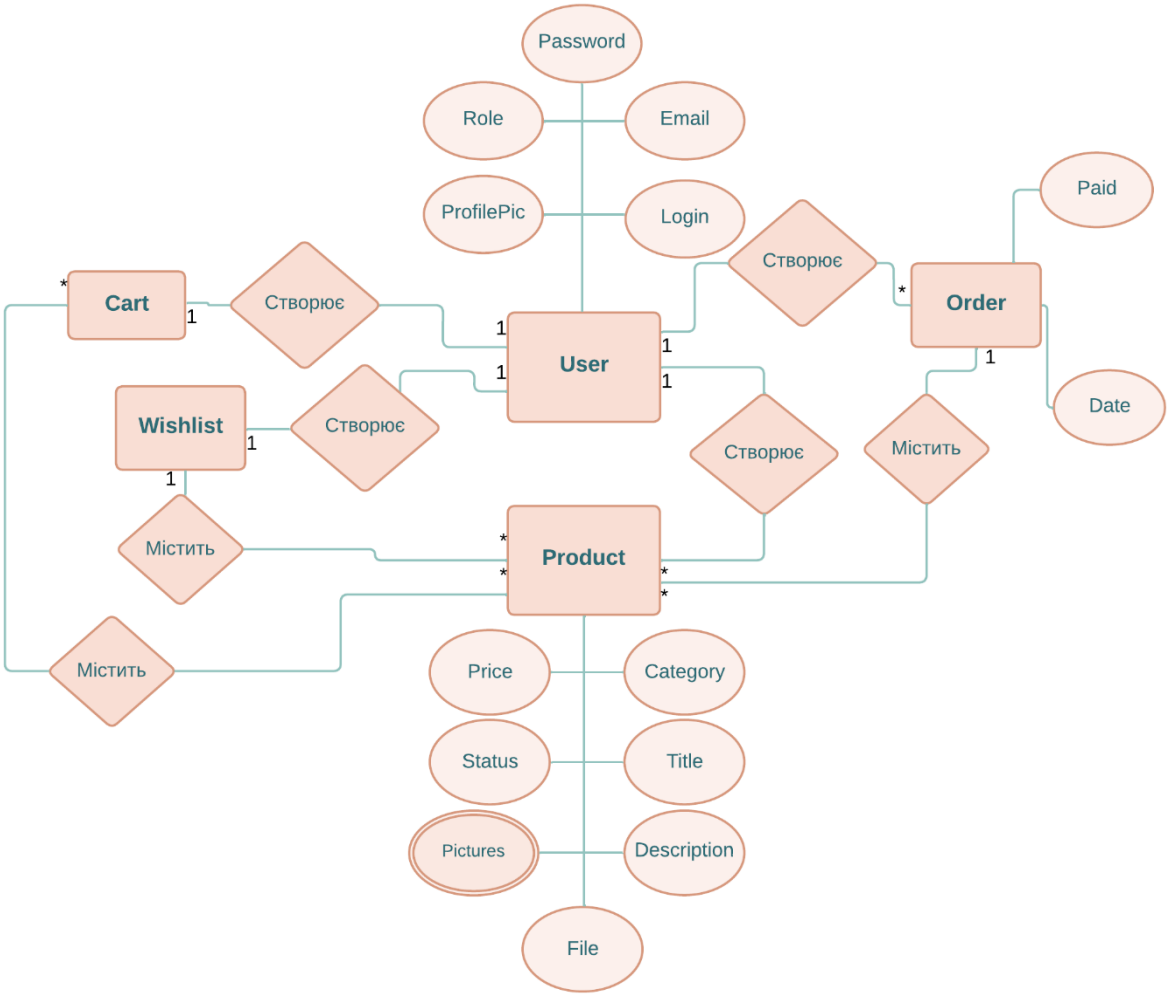


Рисунок 3.5– ER модель сутностей

Таблиця 3.1 – Опис колекції users

Назва поля	Тип даних	Опис
id	ObjectId	Ідентифікатор користувача.
login	String	Ім’я користувача.
email	String	Електронна пошта користувача.
password	String	Пароль користувача.
profilePic	String	Посилання на зображення профілю користувача.

Продовження таблиці 3.1

Wishlist	Array	Список ідентифікаторів товарів, що знаходяться в обраному.
Cart	Array	Список ідентифікаторів товарів, що знаходяться в кошику.
Orders	Array	Список замовлень користувача.

Таблиця 3.2 – Опис об’єкта orders колекції users

Назва поля	Тип даних	Опис
id	String	Ідентифікатор замовлення.
user	String	Ідентифікатор користувача.
orders	Array	Список товарів в замовленні користувача.
paid	Number	Вартість замовлення.
date	Date	Дата проведення замовлення

Таблиця 3.3 – Опис об’єкта orderItems колекції users

Назва поля	Тип даних	Опис
productId	String	Ідентифікатор товару. Один користувач може мати 1 і більше товарів в замовленні.
title	String	Назва товару.
price	Number	Ціна товару.
picture	String	Посилання на зображення товару.

Таблиця 3.4 – Опис колекції products

Назва поля	Тип даних	Опис
id	ObjectId	Ідентифікатор товару.
createdBy	ObjectId	Ідентифікатор користувача, що створив даний товар. Один користувач може мати 0 і більше створених товарів.
category	String	Категорія товару.
title	String	Назва товару.
description	String	Текстовий опис товару.
price	Number	Ціна товару.
productPics	Array	Список посилань на зображення товару.
productFie	String	Посилання на файл товару.
status	String	Статус товару.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.5.

Таблиця 3.5 – Опис утиліт

№	Назва утиліти	Опис застосування
1	VisualStudioCode	Редактор вихідного коду, середовище розробки застосунку.
2	MongoDBAtlas	Хмарний сервіс бази даних.
3	stripe	Бібліотека StripeNode забезпечує зручний доступ до Stripe API із програм, написаних на JavaScript.
4	bcryptjs	Бібліотека пакетів, яка дозволяє хешувати паролі в NodeJS.

Продовження таблиці 3.5

5	Mongoose	Представляє ODM-бібліотеку (ObjectDataModelling) для роботи з MongoDB.
6	next-auth	Бібліотека для автентифікації, керування сесіями та створення облікових записів користувачів.

Для наповнення додатку контентом було використано графічні матеріали, що перебувають у вільному доступі в мережі Інтернет (таблиця 3.6).

Таблиця 3.6 – Опис ресурсів

№ п/п	Назва ресурсу	Опис застосування
1	pexels.com	Ресурс безкоштовних зображень.
2	mui.com	ReactMaterialIcons, ресурс безкоштовних піктограм.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

У сучасну цифрову епоху забезпечення надійних заходів безпеки для автентифікації користувачів є надзвичайно важливим. Автентифікація є необхідною функцією для будь-якої програми, яка вимагає від користувачів надання своїх даних, таких як електронна пошта та пароль, для доступу до програми. Для забезпечення захисту даних користувача та безпечної реєстрації було використано NextAuth.js.

Next-auth — це модуль автентифікації для Next.js, який керує токенами. Ці токени використовуються для надання дозволу на доступ до захищених ресурсів.

Тепер давайте детальніше розглянемо процес автентифікації Next.js (рисунок 3.6).

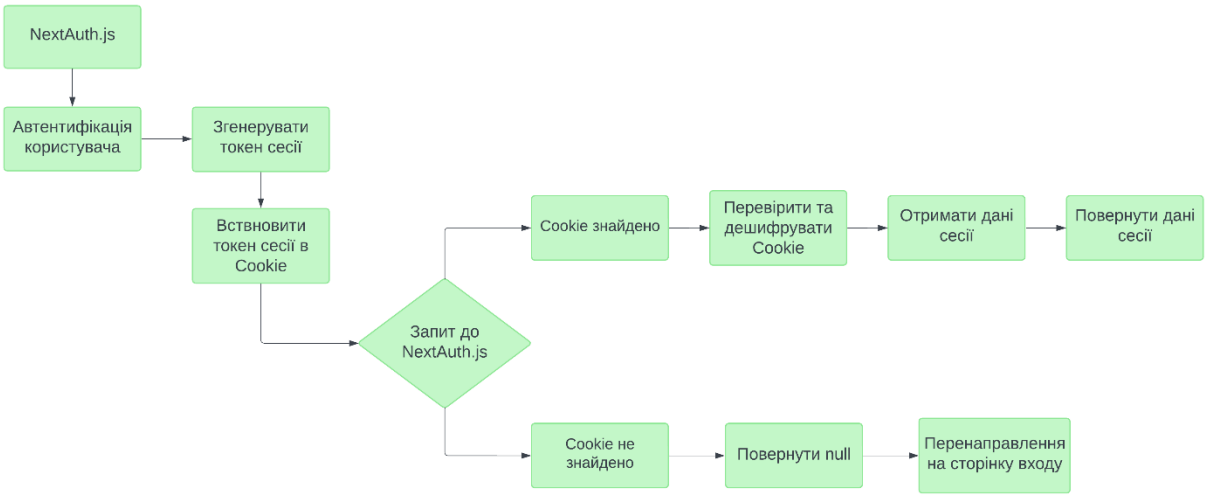


Рисунок 3.6 – Процес автентифікації з NextAuth.js

Автентифікація користувача передбачає надання облікових даних для входу або використання джерела автентифікації третьої сторони. Після авторизації NextAuth.js створює sessiontoken, збережений у файлі cookie у браузері користувача.

Згодом запити надсилаються від клієнта до сервера NextAuth.js. NextAuth.js виявляє файл cookiesession у вхідному запиті та отримує sessiontoken після його перевірки та розшифрування.

Потім NextAuth.js отримує відповідні дані активної сесії користувача за допомогою пов’язаного sessiontoken. Сервер надсилає дані сесії клієнту після отримання їх із сервера або розміщення в файлі cookie.

Якщо користувач не авторизований, замість даних сеансу повертається порожній об’єкт (nullorundefined).

Для отримання даних про користувача, автентифікованого в системі, використовується useSession() хук (рисунок 3.7).

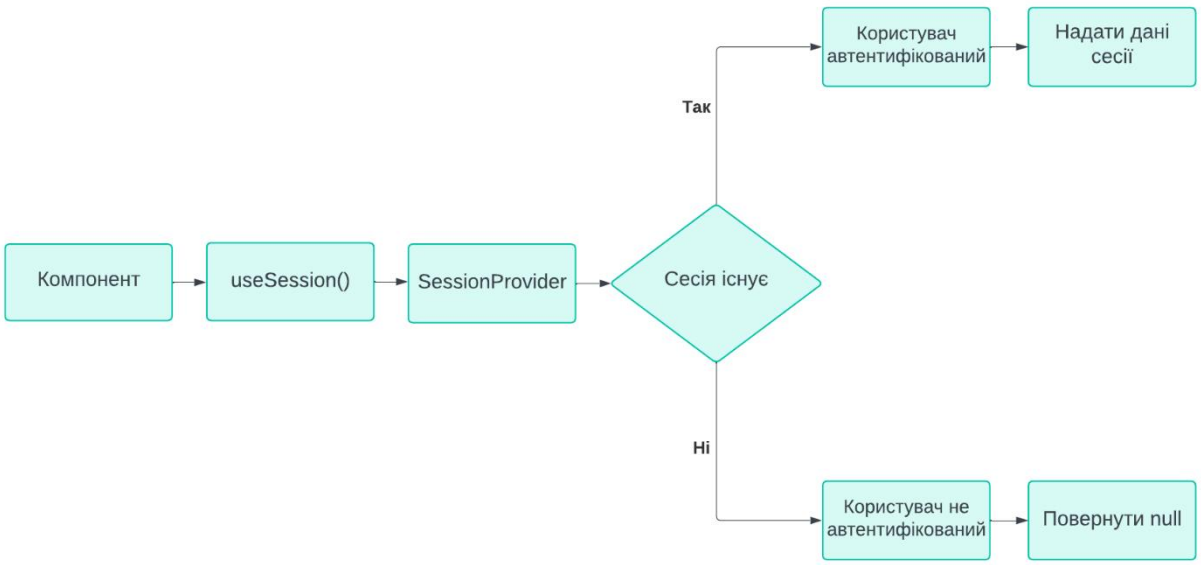


Рисунок 3.7 – Процес доступу до даних сесії з useSession()

Щоб отримати доступ до даних сесії та статусу автентифікації, компонент викликає хук useSession. Функція getSession викликається провайдером sessionprovider, який визначає чи сесія активна, тобто чи користувач автентифікований. Якщо сесія активна, компоненту надається доступ до захищених даних користувача. Якщо сесія неактивна, користувач не пройшов автентифікацію, і замість захищеної інформації повертається значення null.

Також для безпечного зберігання пароля було використано бібліотеку bcrypt. bcrypt - це функція хешування паролів, що використовується для їх захисту в базах даних, щоб їх не зміг використовувати той, хто отримає доступ до вихідного кешу. При введенні щойно створеного пароля він передається у функцію хешування. Оскільки хеш є односторонньою операцією, отримати назад хешований пароль неможливо. Коли користувач намагається увійти в систему або потрібно перевірити пароль, вихідне значення поля, отримане під час його введення, хешується, і два хеші порівнюються. Якщо хеші однакові, то й пароль має бути однаковим.

Висновки до розділу

У розділі 3 дипломної роботи було описано архітектуру розробки, наведено її наочне зображення на діаграмі компонентів.

Було розглянуто існуючі засоби розробки, які можуть бути використані для створення вебдодатків, серед яких було обрано JavaScript, Next.js і MongoDB.

Описано основні моменти розробки клієнтської та серверної частин додатку. У вигляді ER-моделі зображено структуру БД.

Описано основні утиліти, бібліотеки та ресурси, що використовуються у розробці веб-додатку. Проведено аналіз безпеки даних. Передбачено перевірку аутентифікації користувача, безпечне зберігання пароля у вигляді шифру. Реалізовано основний функціонал веб-забезпечення.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Якість ПЗ - це всі властивості розробленого застосування, за якими можна визначити, чи спроможна система задовольнити запити замовника, висловлені у вигляді вимог до програмного забезпечення [20].

Метою тестування є перевірка правильності роботи додатку, знаходження та виправлення помилок.

Метриками для оцінки якості ПЗ є:

- функціональна повнота -основних функцій системи достатньо для вирішення завдань відповідно до призначення ПЗ;
- сумісність -ПЗ може взаємодіяти з більшістю браузерів (GoogleChrome, Microsoft Edge, Opera);
- захищеність -у програмі реалізовано можливість запобігати несанкціонованому доступу до даних.
- зручність застосування -логічні концепції розробленого додатку зрозумілі для розпізнавання та застосування.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було проведено мануальне тестування додатку, опис тестів наведено у таблицях 4.1 – 4.10.

Таблиця 4.1 – Тест 01

Тест	Реєстрація користувача
Номер тесту	01
Початковий стан системи	Користувач знаходиться на сторінці реєстрації.
Вхідні дані	Логін, електронна пошта, пароль, підтвердження пароля.

Продовження таблиці 4.1

Опис проведення тесту	У відповідні поля вводяться: логін, електронна пошта, пароль, які до цього не були зареєстровані в системі. Пароль вводиться повторно в поле для підтвердження пароля. Після цього натискається кнопка «Зареєструватися».
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку входу.
Фактичний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку входу.

Таблиця 4.2 – Тест 02

Тест	Вхід користувача
Номер тесту	02
Початковий стан системи	Користувач знаходиться на сторінці входу.
Вхідні дані	Електронна пошта, пароль.
Опис проведення тесту	У відповідні поля вводяться: електронна пошта, яка до цього була зареєстрована в системі, пароль. Після цього натискається кнопка «Увійти».
Очікуваний результат	Вхід проходить успішно, користувач перенаправляється на домашню сторінку.
Фактичний результат	Вхід проходить успішно, користувач перенаправляється на домашню сторінку.

Таблиця 4.3 – Тест 03

Тест	Створення товару
Номер тесту	03
Початковий стан системи	Користувач знаходиться на сторінці створення товару.
Вхідні дані	Категорія,назва, текстовий опис товару, ціна, файли зображення товару .jpg, файл товару і документ, згруповані в .zip.
Опис проведення тесту	У відповідні поля вводяться: назва, текстовий опис товару, ціна, додаються файли, зі списку обирається категорія «Зображення». Натискається кнопка «Створити товар».
Очікуваний результат	Створення товару проходить успішно, користувач перенаправляється на сторінку зі списком своїх товарів. Відображається статус «Товар чекає підтвердження».
Фактичний результат	Створення товару проходить успішно, користувач перенаправляється на сторінку зі списком своїх товарів. Відображається статус «Товар чекає підтвердження».

Таблиця 4.4 – Тест 04

Тест	Зміна статусу товару
Номер тесту	04
Початковий стан системи	Користувач з правами адміністратора знаходиться на сторінці панелі адміністратора. На сторінці є список товарів зі статусом «Товар чекає підтвердження».
Вхідні дані	-
Опис проведення тесту	Натискається картка товару. Користувач перенаправляється на сторінку деталей товару.Користувач у випадаючому списку обирає статус «Підтверджено».
Очікуваний результат	Статус товару змінюється на «Підтверджено». Картка товару більше не відображається на сторінці адміністратора та з’являється на домашній сторінці.
Фактичний результат	Статус товару змінюється на «Підтверджено». Картка товару більше не відображається на сторінці адміністратора та з’являється на домашній сторінці.

Таблиця 4.5 – Тест 05

Тест	Додавання товару в обране
Номер тесту	05
Початковий стан системи	Користувач знаходиться на домашній сторінці. Даний товар не знаходиться в списку обраних.
Вхідні дані	-
Опис проведення тесту	Натискається кнопка «Додати до обраного».
Очікуваний результат	Доданий товар відображається в списку обраного даного користувача. Кнопка «Додати до обраного» змінюється на «Видалити з обраного».
Фактичний результат	Доданий товар відображається в списку обраного даного користувача. Кнопка «Додати до обраного» змінюється на «Видалити з обраного».

Таблиця 4.6 – Тест 06

Тест	Видалення товару з обраного
Номер тесту	06
Початковий стан системи	Користувач знаходиться на домашній сторінці. Даний товар знаходиться в списку обраних.
Вхідні дані	-
Опис проведення тесту	Натискається кнопка «Видалити зобраного».

Продовження таблиці 4.6

Очікуваний результат	Доданий товар видаляється зі списку обраного даного користувача. Кнопка «Видалити з обраного» змінюється на «Додати до обраного».
Фактичний результат	Доданий товар видаляється зі списку обраного даного користувача. Кнопка «Видалити з обраного» змінюється на «Додати до обраного».

Таблиця 4.7 – Тест 07

Тест	Оновлення товару
Номер тесту	07
Початковий стан системи	Користувач знаходиться на сторінці оновлення товару.
Вхідні дані	Назва, текстовий опис товару, ціна, файли зображення товару .jpg.
Опис проведення тесту	У відповідні поля вводяться: назва, текстовий опис товару, ціна, додаються файли зображення товару. Натискається кнопка «Оновити товар».
Очікуваний результат	Оновлення товару проходить успішно, користувач перенаправляється на сторінку зі списком своїх товарів.
Фактичний результат	Оновлення товару проходить успішно, користувач перенаправляється на сторінку зі списком своїх товарів.

Таблиця 4.8 – Тест 08

Тест	Видалення товару
Номер тесту	08
Початковий стан системи	Користувач знаходиться на домашній сторінці.
Вхідні дані	-
Опис проведення тесту	Натискається кнопка «Видалити». Користувач натискає кнопку «Підтвердити» для підтвердження видалення товару.
Очікуваний результат	Товар видаляється із системи та більше не відображається на домашній сторінці та на сторінці товарів цього користувача.
Фактичний результат	Товар видаляється із системи та більше не відображається на домашній сторінці та на сторінці товарів цього користувача.

Таблиця 4.9 – Тест 09

Тест	Додавання товару до кошика
Номер тесту	09
Початковий стан	Користувач знаходиться на сторінці товару.

Продовження таблиці 4.9

Вхідні дані	-
Опис проведення тесту	Натискається кнопка «Додати в кошик». Натискається кнопка «Перейти до кошика».
Очікуваний результат	Лічильник кошика збільшується на 1. Товар відображається на сторінці кошика.
Фактичний результат	Лічильник кошика збільшується на 1. Товар відображається на сторінці кошика.

Таблиця 4.10 – Тест 10

Тест	Оплата замовлення
Номер тесту	10
Початковий стан системи	Користувач знаходиться на сторінці кошика. В кошику відображаються додані товари.
Вхідні дані	Номер картки
Опис проведення тесту	Натискається кнопка «Оформити замовлення». Відбувається перехід на сторінку платіжної системи. Вводяться дані картки для оплати. Натискається кнопка «Оплатити». Відбувається перенаправлення на сторінку з результатом оплати. Натискається кнопка «Завантажити».
Очікуваний результат	Ідентифікатор замовлення, вартість, дата проведення оплати та посилання на завантаження відображаються на сторінці замовлень користувача. Файл завантажився.
Фактичний результат	Ідентифікатор замовлення, вартість, дата проведення оплати та посилання на завантаження відображаються на сторінці замовлень користувача. Файл завантажився.

4.3 Опис контрольного прикладу

Спочатку створимо нового користувача на сторінці реєстрації (рисунок 4.1). Якщо паролі не співпадають, кнопка «Зареєструватися» залишається неактивною.

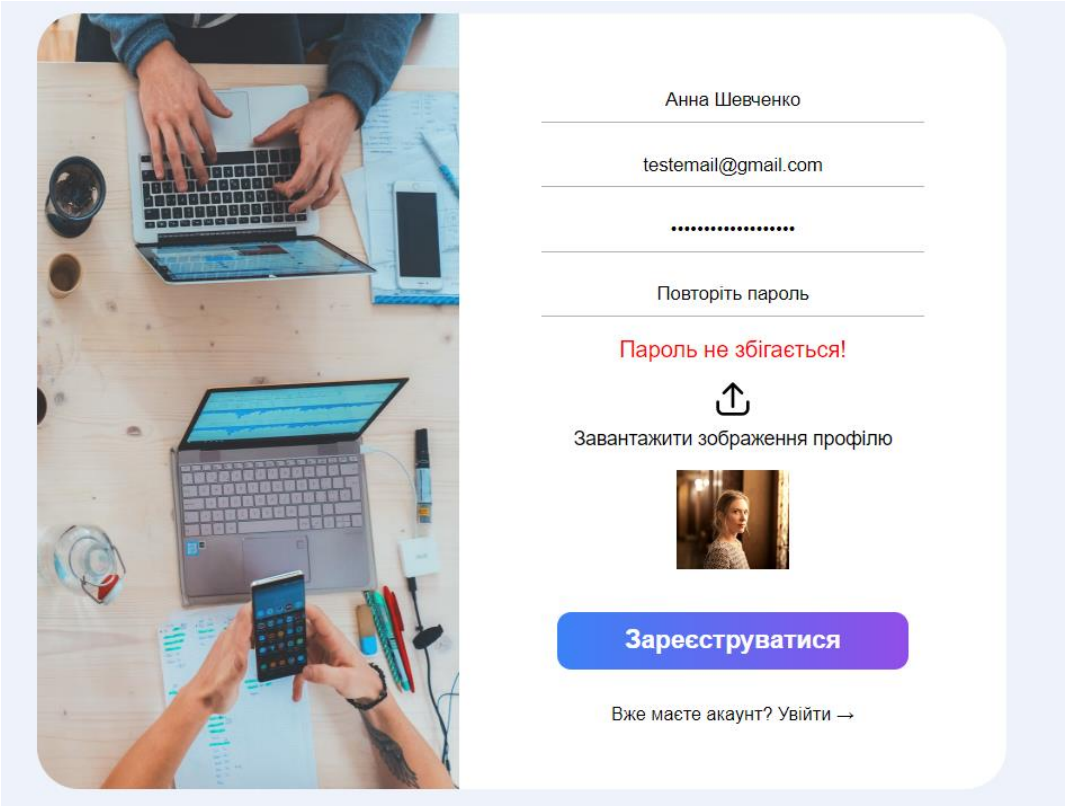


Рисунок 4.1 – Заповнення форми на сторінці реєстрації

Після чого натискаємо кнопку «Зареєструватися» і переходимо на сторінку входу (рисунок 4.2). Вводимо електронну пошту і пароль та натискаємо кнопку «Увійти».

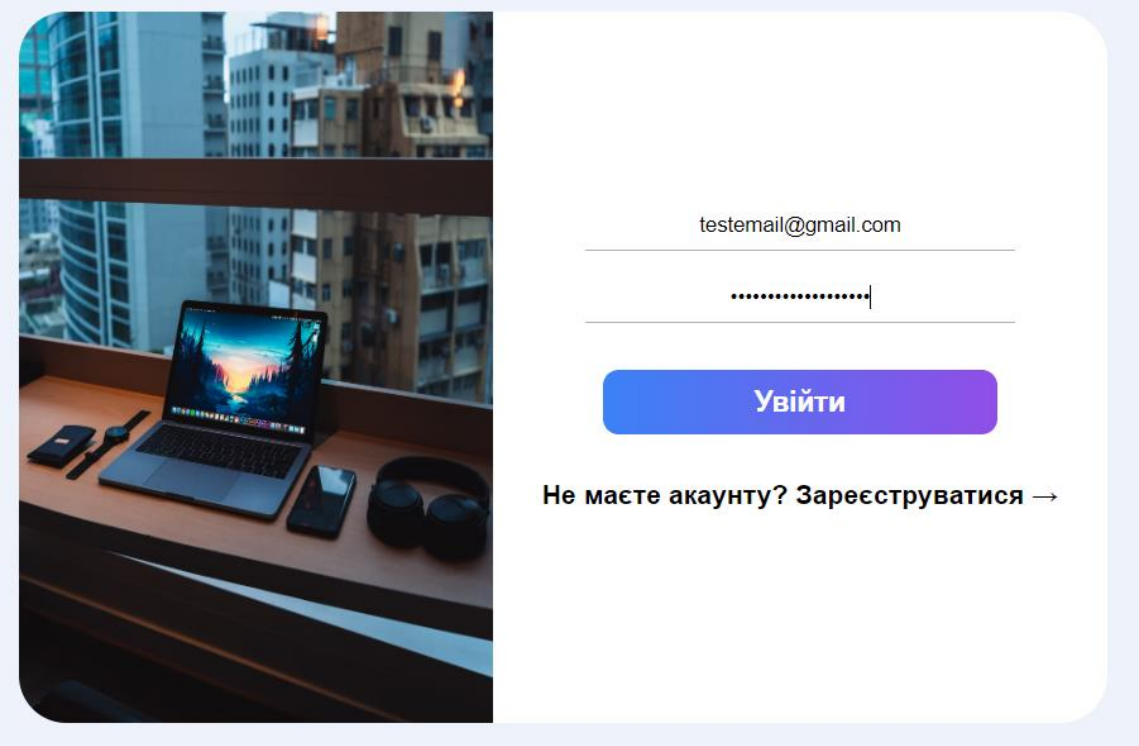


Рисунок 4.2 – Виконання входу на сторінці входу

Переходимо на домашню сторінку (рисунок 4.3). Там відображаються товари користувачів. При натисканні на категорію товари фільтруються за обраною категорією.На панелі у верхній частині сторінки відображаютьсялоготип, при натисканні на який користувач переходить на домашню сторінку, пошук, кошик з лічильником товарів в кошику, зображення профілю користувача. При натисканні кнопки меню відкривається меню навігації.

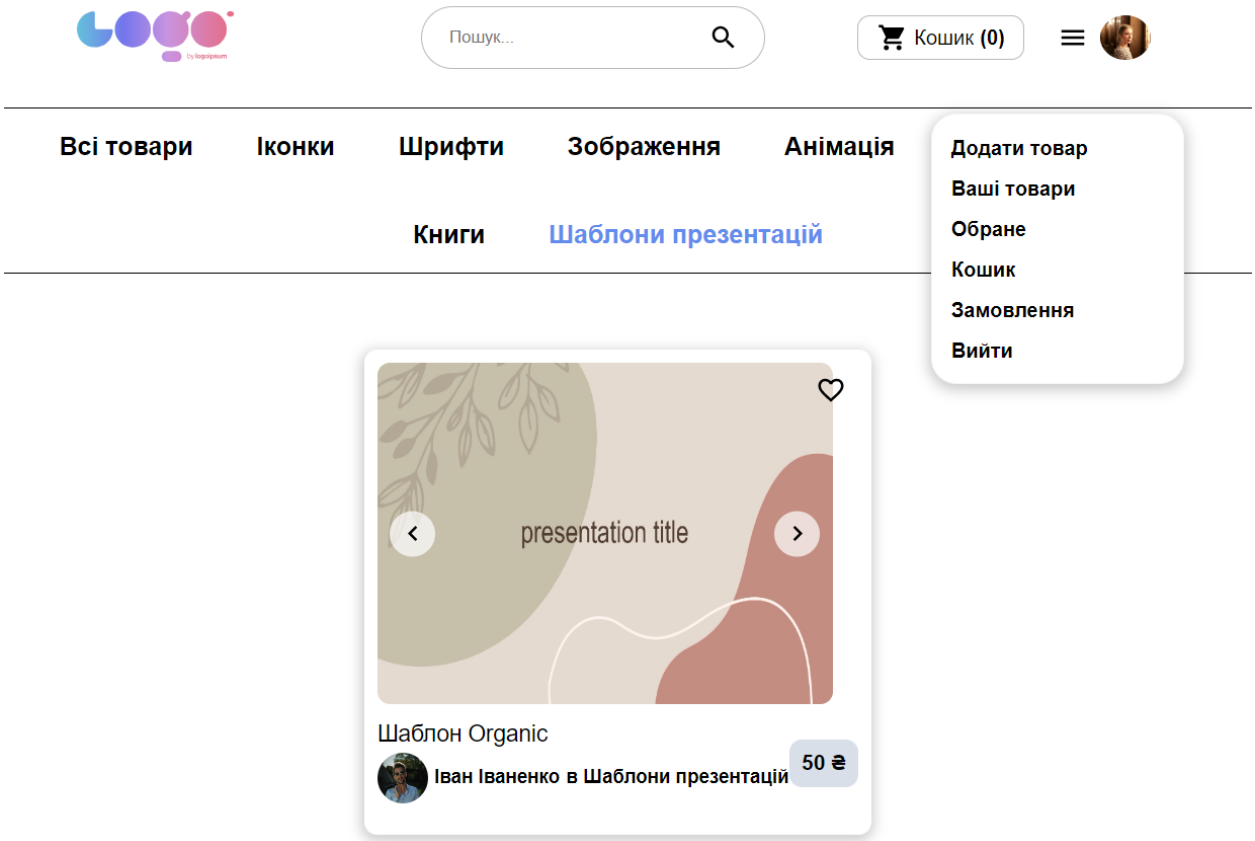


Рисунок 4.3 – Домашня сторінка

Натискаємо «Створити товар» та переходимо на сторінку додавання нового товару (рисунок 4.4). Обираємо категорію, вводимо назву, опис, ціну. Додаємо зображення та файл. Файл додається у вигляді архіву з документом, що підтверджує авторське право на цифровий продукт. Натискаємо «Додати товар».

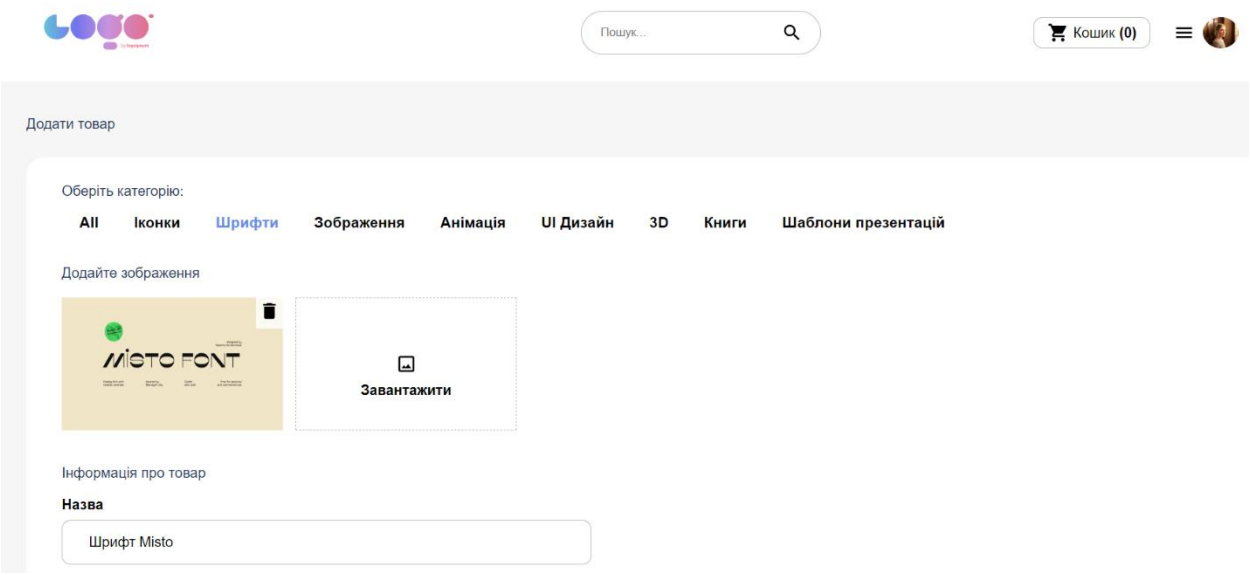


Рисунок 4.4 – Сторінка створення товару

Після цього автоматично переходимо на сторінку створених користувачем товарів, де ми можемо переглянути коротку інформацію про товар (рисунок 4.5).

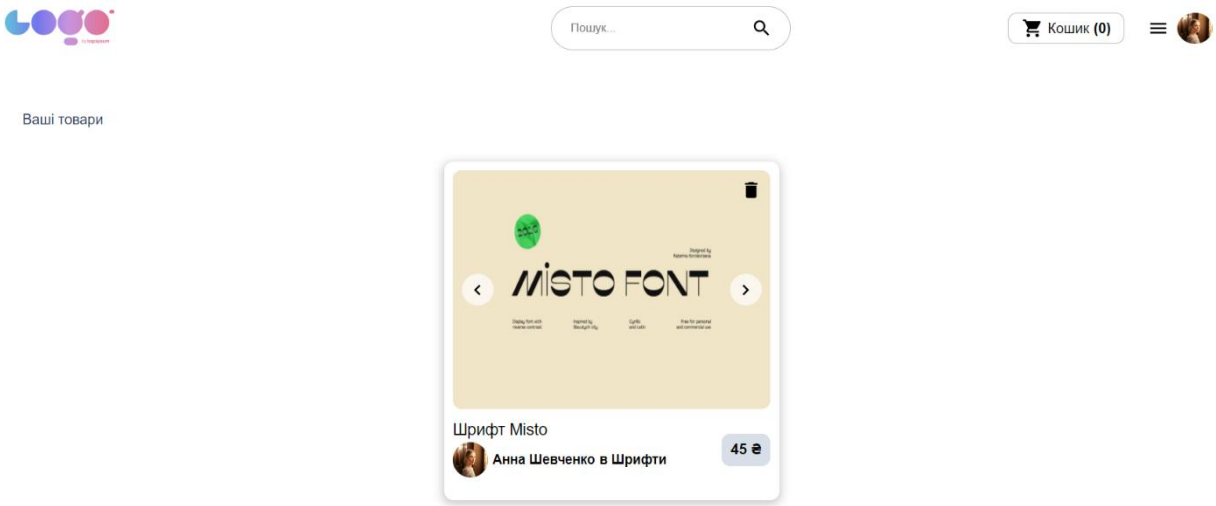


Рисунок 4.5 – Сторінка товарів користувача

Натиснувши на картку товару, переходимо на сторінку товару (рисунок 4.6).

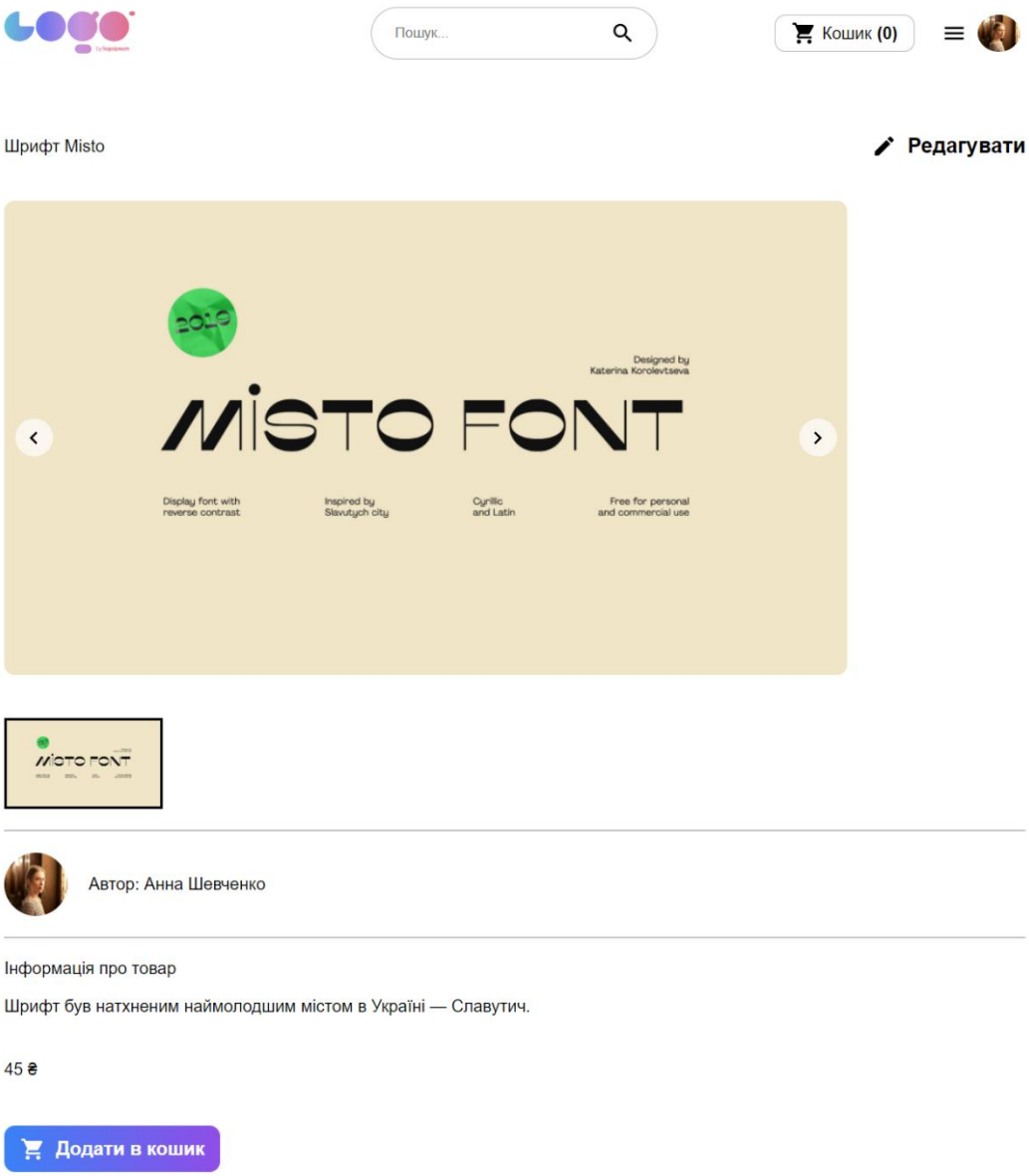


Рисунок 4.6 – Сторінка товару

Для користувача, який є автором даного продукту, доступна опція «Редагувати товар». Обравши її, ми переходимо на сторінку оновлення товару (рисунок 4.7).

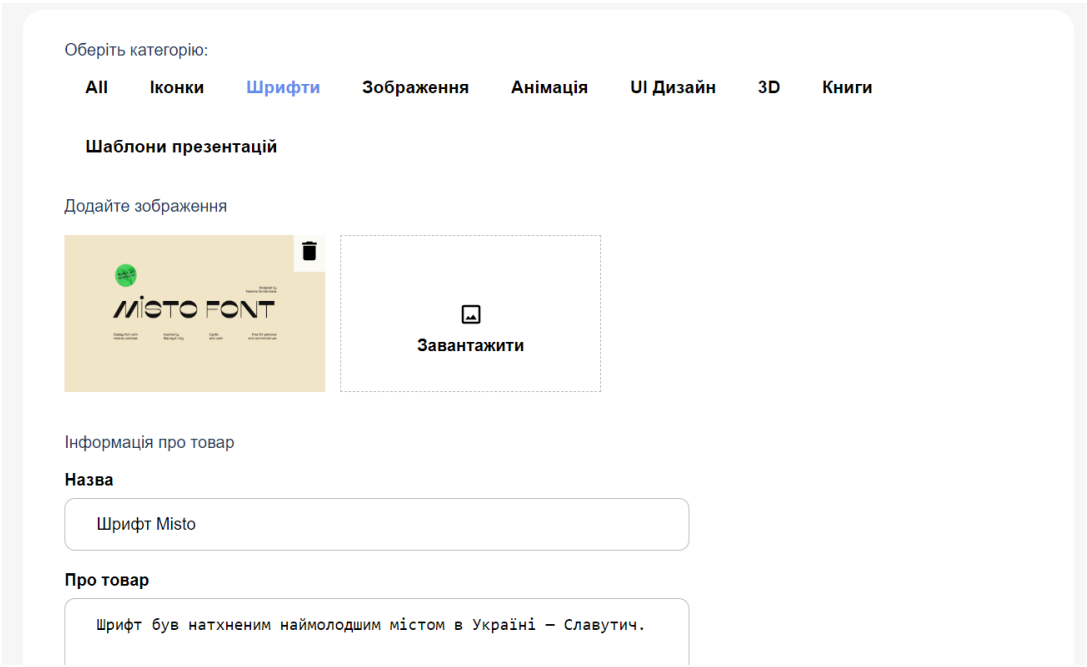


Рисунок 4.7 – Сторінка оновлення товару

Тут можливо змінити назву, категорію, опис, ціну та зображення. Натиснувши кнопку «Зберегти», ми переходимо на сторінку створених користувачем товарів, де відображається товар зі зміненою інформацією.

Оскільки щойно створений товар ще не підтверджений адміністратором, він не відображається на домашній сторінці. Для зміни статусу товару увійдемо в обліковий запис адміністратора. Для такого користувача, на домашній сторінці відображається кнопка «Панель адміністратора» (рисунок 4.8).

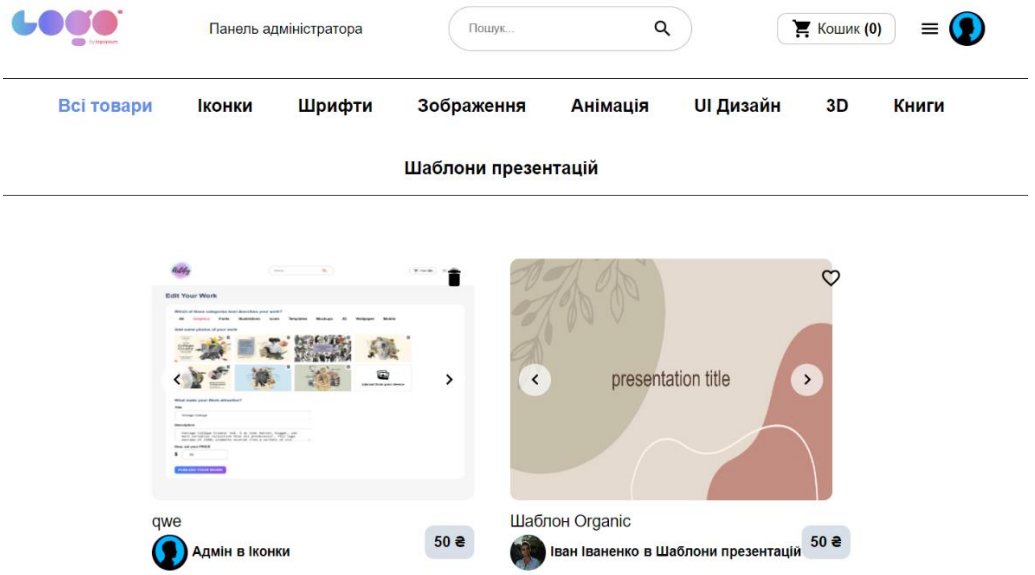


Рисунок 4.8 – Домашня сторінка

Натиснувши її, переходимо на сторінку адміністратора, де відображаються всі товари що чекають перевірки (рисунок 4.9).

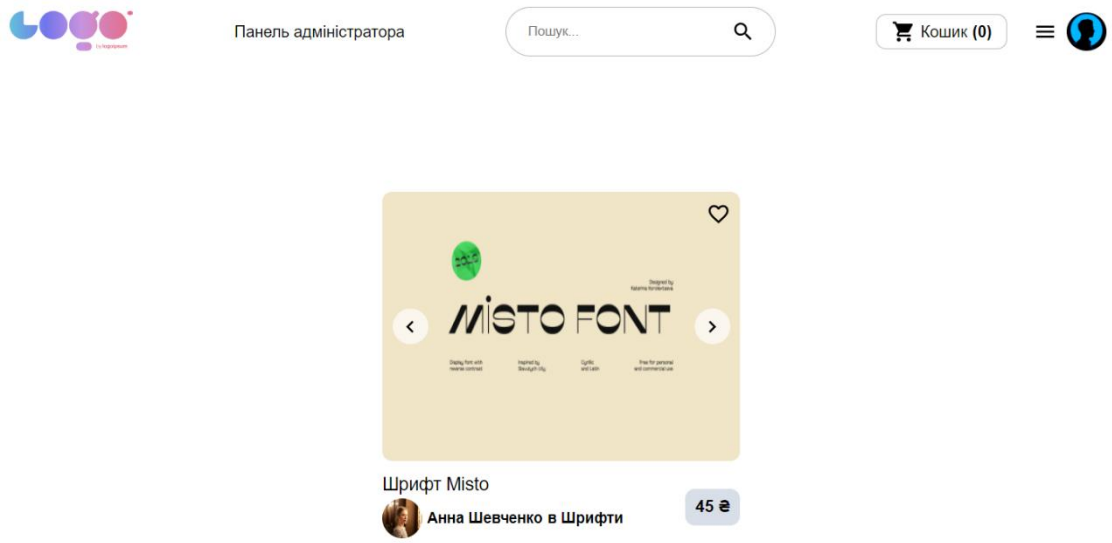


Рисунок 4.9 – Сторінка адміністратора

Натиснувши на товар, переходимо на сторінку товару та змінюємо статус на «Підтверджено» (рисунок 4.10).

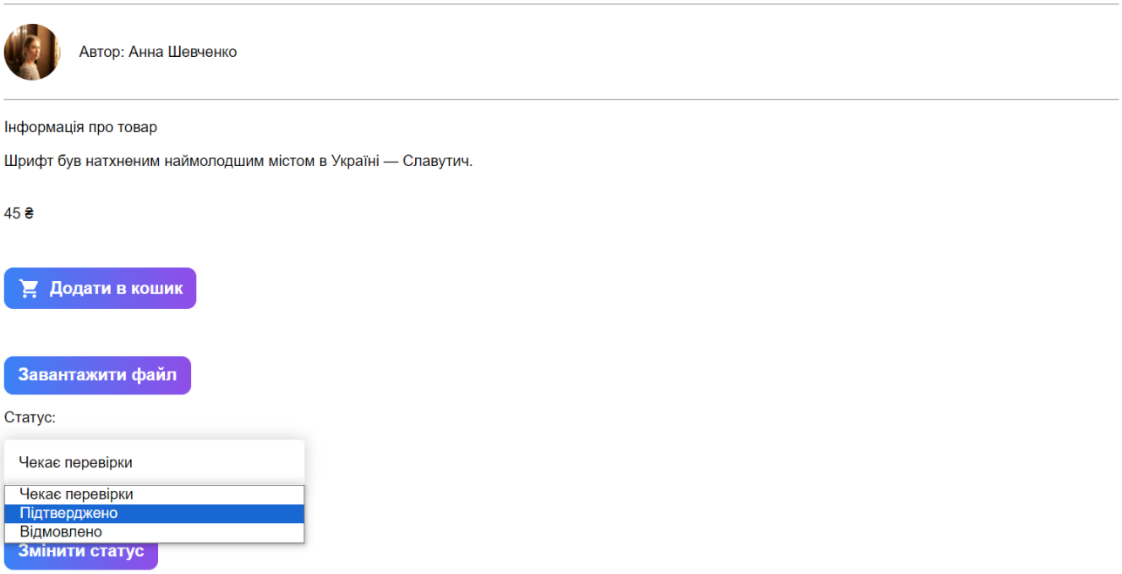


Рисунок 4.10 – Зміна статусу товару

Після цього товар відображається на домашній сторінці (рисунок 4.11). Автор товару може його видалити, натиснувши відповідну кнопку біля зображення на картці товару.

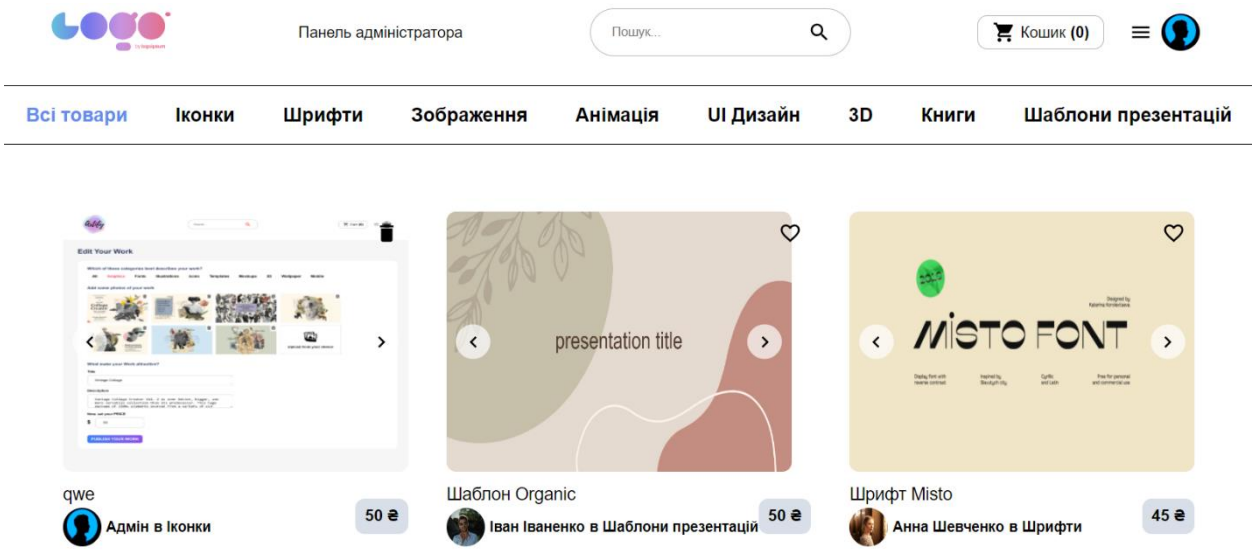


Рисунок 4.11 – Домашня сторінка зі створеним товаром

Виконаємо пошук товару за назвою. Для цього введемо у відповідне поле назву та натиснемо кнопку пошук. Знайдений товар відображається на сторінці результатів пошуку (рисунок 4.12).

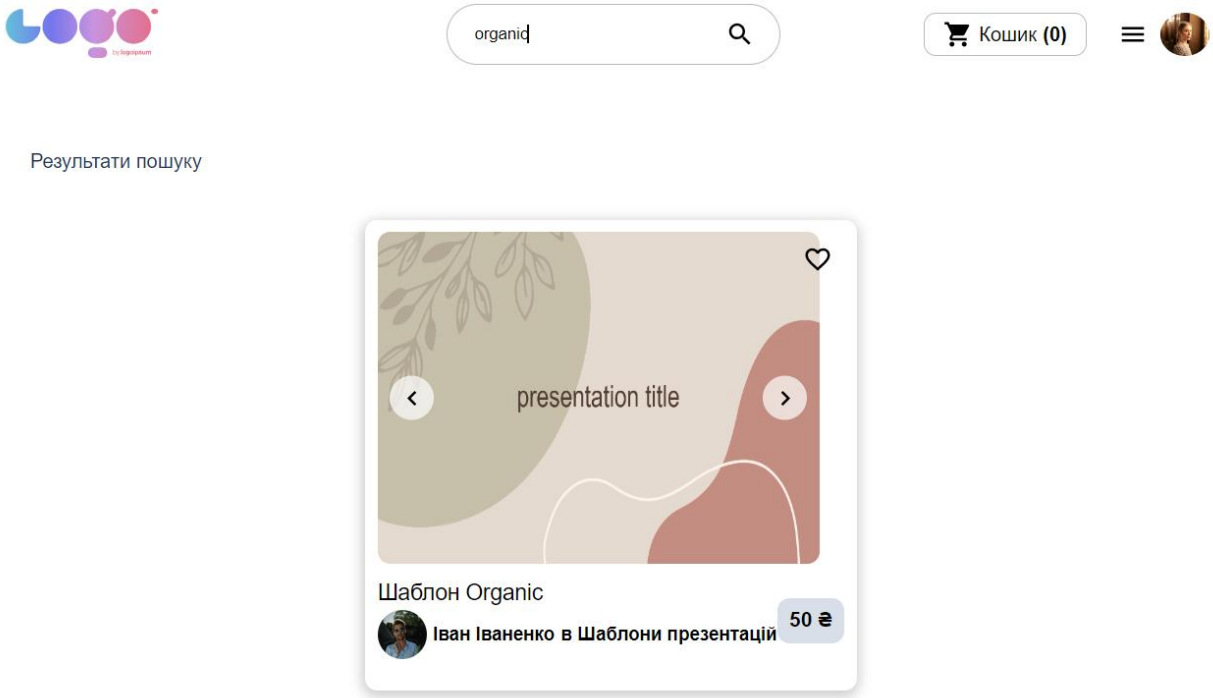


Рисунок 4.12 – Результат пошуку

Перейдемо на сторінку знайденого товару та натиснемо кнопку «Додати до кошика». При повторному натисканні кнопки виводиться повідомлення про те, що даний товар вже знаходиться в кошику (рисунок 4.13).

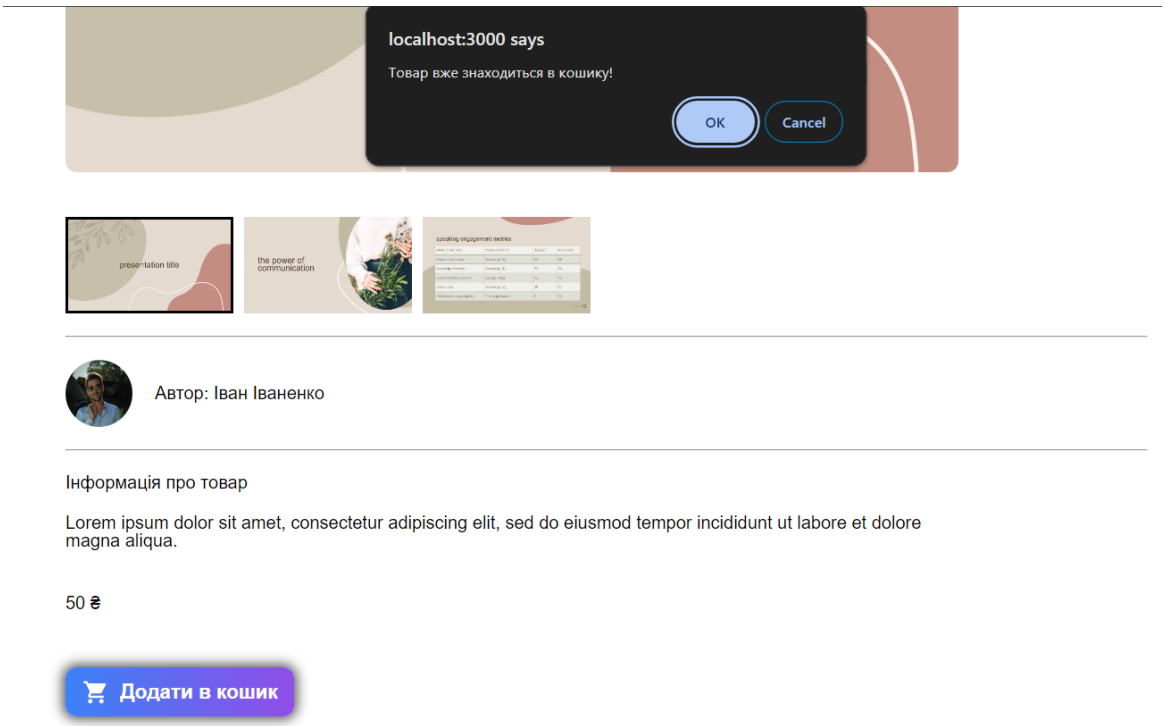


Рисунок 4.13 – Додавання до кошика

Натискаємо на кнопку кошика і переходимо на сторінку кошика користувача (рисунок 4.14). Тут ми бачимо інформацію про додані товари, суму до сплати. При натисканні кнопки видалення, товар видаляється з кошика.

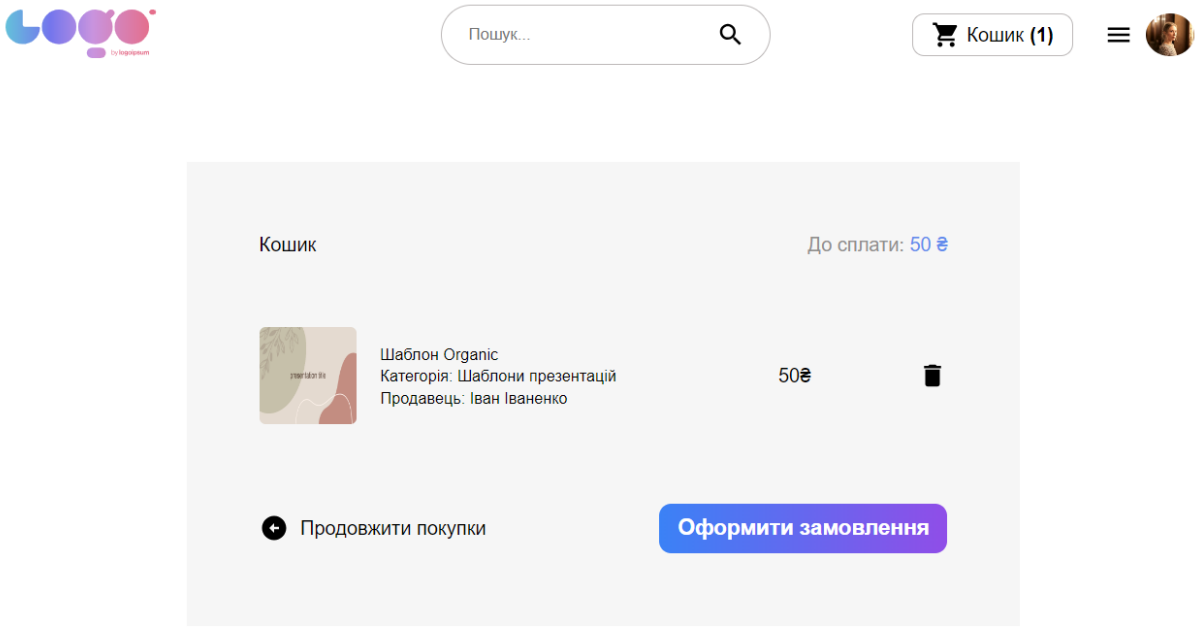


Рисунок 4.14 – Кошик

Натискаємо «Оформити замовлення» та переходимо на сторінку платіжної системи, де ми можемо ввести платіжні дані та натиснути «Оплатити» (рисунок 4.15).

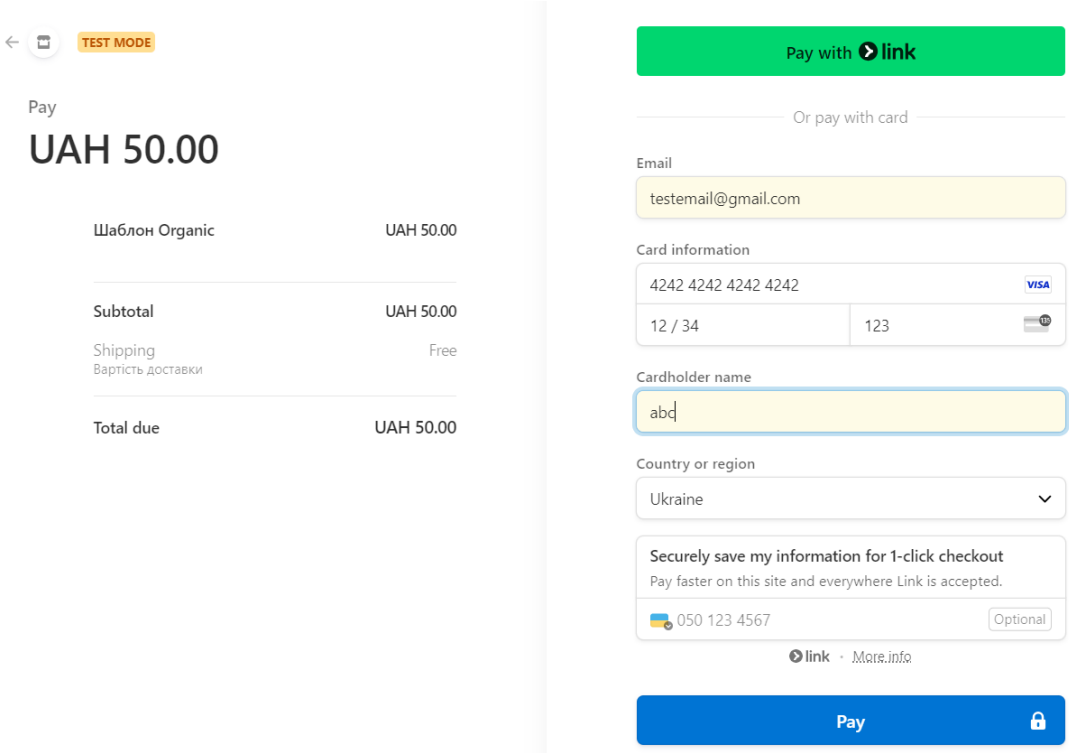


Рисунок 4.15 – Оплата товару

Після цього переходимо на сторінку з замовленнями, де ми можемо завантажити придбаний товар (рисунок 4.16).

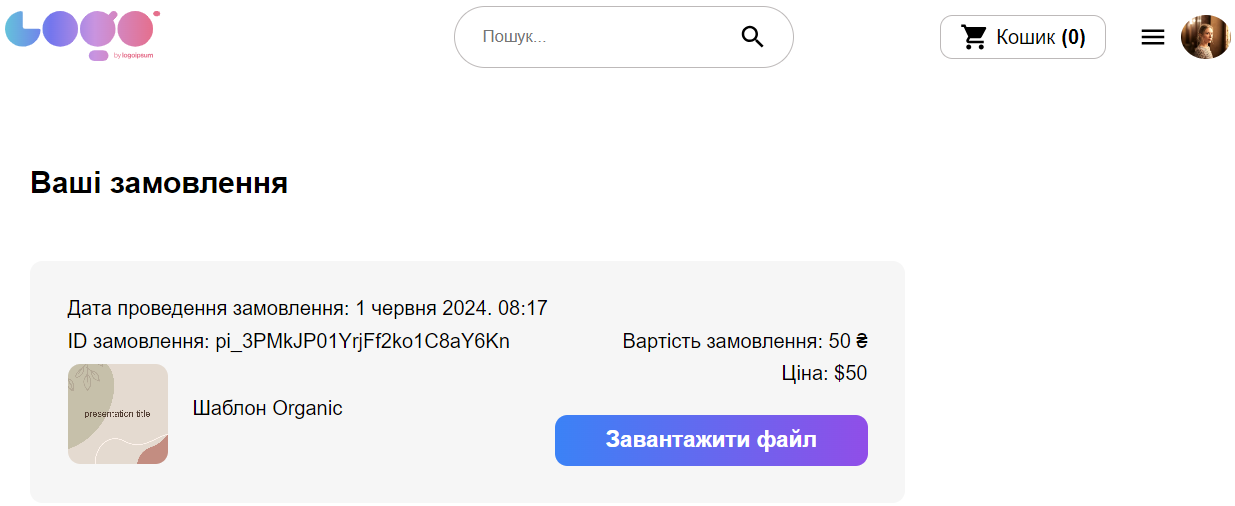


Рисунок 4.16 – Сторінка замовлення

Перебуваючи на домашній сторінці, натискаємо на ім’я або зображення профілю користувача, щоб перейти на сторінку з товарами даного

користувача (рисунок 4.17). Натиснувши на кнопку «Додати в обране», ми додамо бажаний товар в свій список обраних товарів. Якщо натиснути на кнопку повторно, товар видалиться з обраного.

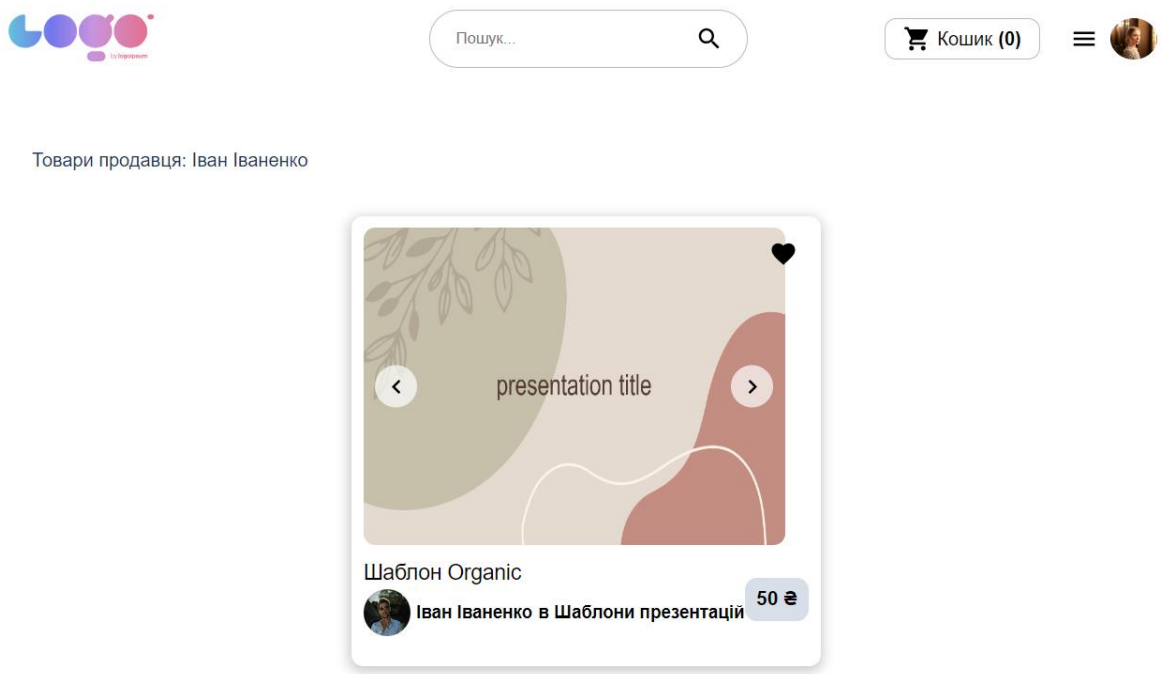


Рисунок 4.17 – Сторінка з товарами користувача

На сторінці товару доступна кнопка «Порушення авторського права». При її натисканні товар стає недоступним для додавання в кошик, видаляється з домашньої сторінки та відображається на сторінці адміністратора. Якщо виявлено порушення, адміністратор видаляє товар.

Висновки до розділу

У розділі 4 було виконано тестування програмного забезпечення з метою перевірки правильності роботи додатку та виправлення помилок. Якість розробки було проаналізовано з урахуванням таких характеристик, як: функціональна повнота, сумісність з різними браузерами, захищеність даних та зручність інтерфейсу.

Було проведено мануальне тестування додатку. У відповідних таблицях порівнювалися очікуваний та фактичний результат дій користувача.

Було описано покрокове використання розробленого вебдодатку: послідовність створення нового облікового запису, входу в систему, створення, оновлення, видалення та купівлі товару.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Розгортання програми Next.js є важливим кроком у життєвому циклі вебдодатку. Процес розгортання включає:

– розміщення програми на платформі хостингу;

– налаштування;

– забезпечення безперебійної роботи для користувачів у всьому світі.

Розгортання додатку дозволяє забезпечити:

1) Доступність.Користувачі повинні мати доступ до проекту та безперешкодно використовувати його функції.
2) Тестування реальними користувачами.Розгортання проекту є єдиним способом перевірити, чи він вирішує потрібну проблему. Це дозволяє тестувати та виправляти помилки реальних користувачів і даних, а також виправляти програму, щоб вона працювала належним чином.
3) Масштабованість і надійність.Розгортання програми дозволяє додатку масштабуватися, працювати з більшою кількістю користувачів і збільшувати трафік.

Першим доступним варіантом розгортання є статичний хостинг. Простий статичний вебсайт із незначним рендерингом на стороні сервера можна розмістити на таких платформах, як Netlify або GitHubPages. Статичний хостинг не потребує керування базою даних або обробки на стороні сервера. Це буде дешевше для невеликих програм і вебсайтів з низьким трафіком. Крім того, статичні файли менш вразливі до типових атак вебдодатків.

Наступний варіант — ServerlessFunctions. Проекти Next.js можна інтегрувати з ServerlessFunctions, такими як AWS Lambda або Vercel. Цей варіант розгортання допомагає обробляти надсилання форми, виклики API та інші обчислення на стороні сервера масштабованим і дешевим способом.

Також при розгортанні програми Next.js слід враховувати використання:

- змінних середовища для керування конфіденційною інформацією API та конфігурацією.
- Git для відстежування змін у кодовій базі[21].

Для розгортання додатку використаємо Vercel. Vercel створено розробниками Next.js.

Спочатку потрібно створити обліковий запис Vercel. Після цього на інформаційній панелі Vercel створюється новий проект за допомогою опції «Продовжити з GitHub». У списку репозиторіїв обирається потрібний, у нашому випадку це digit-market, та імпортується. Для нашого додатку було обрано Next.js як фреймворк.

Після цього, у розділі «Змінні середовища» додаються всі змінні середовища з локального файлу .env.

Після того, як підготовка завершена, відбувається розгортання додатку. Застосунок стає доступним за власним доменом.

5.2 Супровід програмного забезпечення

Інструкція користувача міститься в окремому документі.

Технічна підтримка передбачає оновлення існуючих сторінок при зміні вимог до додатку, а також створення нових сторінок і компонентів, оптимізацію роботи веб-застосування, усунення вірусів та неполадок, розширення інтеграції із зовнішніми ресурсами.

Графічний супровід передбачає оновлення існуючих та створення нових панелей, кнопок, форм, піктограм тощо; підбір та обробку зображень, шрифтів, дизайну кольорового оформлення тощо.

Висновки до розділу

У розділі 5були описані різні стратегії та практики розгортання програм Next.js. Були наведені сильні сторони стратегій статичного хостингу та ServerlessFunctions.

Були також наведені основні практики для оптимізації процесу розгортання та підвищення продуктивності та надійності програми Next.js, такі як використання змінних середовища та Git.

Після цього було наведено опис покрокового розгортання розробленого програмного забезпечення. Описано зміст технічної підтримки додатку: оновлення, створення нових сторінок і компонентів, оптимізацію роботи веб-застосування, усунення неполадок тощо.

Також доступна графічна підтримка веб-застосування, що передбачає оновлення компонентів, їх розташування на сторінці та зовнішнього вигляду.

ВИСНОВКИ

Дана дипломна робота присвячена створенню вебдодатку «Онлайн-ринок електронної комерції для цифрових продуктів», котрий використовується для онлайн-торгівлі та призначений для виробників та споживачів цифрових товарів.

За результатом виконання дипломної роботи було проаналізовано існуючі рішення предметної області: Amazon, Google Play, Rozetka. З'ясовано їхні недоліки, такі як: висока конкуренція, високі комісії, тривалий процес розгляду.

Для розробки програмного забезпечення з урахуванням недоліків існуючих рішень, було визначено функціональні та нефункціональні вимоги нового продукту, такі як: забезпечення процесу купівлі та продажу товарів, перевірка якості продуктів, їх сортування та пошук, а також реалізація можливості запобігати несанкціонованому доступу до даних. Це дасть змогу значно підвищити якість роботи з розробленою системою.

В якості середовища розробки обрано Visual Studio Code. Визначено інструменти для реалізації клієнтської та серверної частин системи. Додаток написаний за допомогою JavaScript, CSS та Next.js на платформі Node.js.

Проведено огляд систем управління базами даних. Для обробки даних користувачів було обрано MongoDBAtlas.

Вирішено всі поставлені перед розробкою задачі:

- основних функцій розробленої системи достатньо для вирішення завдань відповідно до призначення ПЗ: авторизація, створення товарів, купівля товарів, додавання товарів у список обраного, здійснення пошуку товарів, сортування товарів за категоріями.
- логічні концепції розробленого додатку зрозумілі для розпізнавання та застосування.

Було наведено приклад використання програми: реєстрація користувача, створення облікового запису користувача використання можливостей розробленого веб-застосування.

Для перевірки відповідності ПЗ висунутим вимогам та для забезпечення його якості було проведено мануальне тестування. Після реалізації застосунку він був протестований у різних браузерах, з різними розмірами екранів щоб переконатися, що додаток коректно працює на різних пристроях.

Для розгортання додатку запропоновано сервіс Vercel, який максимально пристосований до розгортання додатків на Next.js, оскільки створений однією командою розробників.

Вебзастосунок покликаний, з одного боку, забезпечити рівний доступ користувачів послуг до ринку, з іншого сформувати передумовидля розвитку конкурентного середовища.

Серед переваг програмного забезпечення є покращення ситуації з фінансовою доступністю за рахунок зняття географічних обмежень для купівлі та продажу товарів. При цьому користувачі отримають дистанційний доступ до фінансових послуг у режимі 24/7та широку лінійку цифрових продуктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Park. P. From pixels to profits: How to sell digital products online // Unbounce. URL: https://unbounce.com/ecommerce/how-to-sell-digital-products/ (дата звернення: 15.04.2024).

2) Janković M., Dimitrijević D., Milicevic R. Market and market structures of digital products. *Knowledge International Journal*. 2018. Vol. 28, P. 217-222. DOI: 10.35120/kij2801217j.

3) Патраманська Л.Ю. Електронна комерція: переваги та недоліки. *Ефективна економіка.* 2015. №11. URL: http://www.economy.nayka.com.ua/?op=1&z=4505 (дата звернення: 16.04.2024).

4) Іваненко Л.М. Маркетплейси як об’єктивний наслідок розвитку електронної комерції. *Економіка і організація управління.* 2021. № 4. С. 178-187. DOI: 10.31558/2307-2318.2021.4.16.

5) Кубліцька О. Ринок електронної комерції в Україні: сучасний стан та тенденції повоєнного відновлення. *Проблеми і перспективи економіки та управління*. 2023. № 3. С. 98–108. DOI: 10.25140/2411-5215-2023-3(35)-98-108.

6) Feld-Jakobsen H. Selling on Amazon vs an Ecommerce Website: Pros & Cons // Vaimo. URL: https://www.vaimo.com/blog/pros-cons-amazon-ecommerce/ (дата звернення: 16.04.2024).

7) Truly A. What is Google Play? // AndroidPolice. URL: https://www.androidpolice.com/google-play-guide/ (дата звернення: 16.04.2024).

8) Doglio F. Monolith vs Microservice Architecture: A Comparison // Camunda. URL: https://camunda.com/blog/2023/08/monolith-vs-microservice-architecture-comparison/ (дата звернення: 16.04.2024).

9) Esterkin J. Client-Server Architecture // OpenClassrooms. URL: https://openclassrooms.com/en/courses/6397806-design-your-software-architecture-using-industry-standard-patterns/6896156-client-server-architecture (дата звернення: 17.04.2024).

10) Войтко В.В., Денисюк П.М. Особливості розробки серверних додатків клієнт-серверної архітектури. *Електронні інформаційні ресурси: створення, використання*: збірник матеріалів Міжнародної науково-практичної Інтернет-конференції (Вінниця, 12 грудня 2014 р.). Вінниця. 2014. С. 96-100.

11) DBaaS Benefits: Pros And Cons // ScaleGrid. URL: https://scalegrid.io/blog/dbaas-pros-cons/#:~:text=DBaaS%20offers%20key%20benefits%20that,without%20its%20set%20of%20challenges (дата звернення: 18.04.2024).

12) Tsymbaliuk I. Overview of Stripe's Payments System: Benefits and Features // rates.fm. URL: https://rates.fm/payment-systems/overview-of-stripes-payments-system-benefits-and-features/ (дата звернення: 18.04.2024).

13) Advantages and Disadvantages of JavaScript // GeeksforGeeks. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-javascript/ (дата звернення: 19.04.2024).

14) Advantages and Disadvantages of ASP.NET // Yuhiro. URL: https://www.software-developer-india.com/advantages-and-disadvantages-of-asp-net/ (дата звернення: 19.04.2024).

15) Sharma I. Advantages of Node js // TatvaSoft. URL: https://www.tatvasoft.com/outsourcing/2021/07/advantages-of-node-js.html (дата звернення: 19.04.2024).

16) The Advantages and Disadvantages of AngularJS // Guru. URL: https://www.guru.com/blog/the-advantages-and-disadvantages-of-angularjs/ (дата звернення: 19.04.2024).

17) What is Next.js and its benefits? // Medium. URL: https://medium.com/@asiandigitalhub/what-is-next-js-and-its-benefits-8b13aab56bfd#:~:text=without%20sacrificing%20performance.-,Next.,handle%20versatile%20and%20scalable%20projects (дата звернення: 20.04.2024).

18) Singh H. Oracle Database Advantages, Disadvantages and Features // NineHertz. URL: https://theninehertz.com/blog/advantages-of-using-oracle-database (дата звернення: 22.04.2024).

19) Advantages of MongoDB // MongoDB. URL: https://www.mongodb.com/resources/compare/advantages-of-mongodb (дата звернення: 22.04.2024).

20) Плескач В.Л., Затонацька Т.Г. Інформаційні системи і технології на підприємствах: підручник. Київ: Знання, 2011. 718 с.

21) Okeke T. How to Choose a Deployment Strategy for Your Next.js Application // freeCodeCamp. URL: https://www.freecodecamp.org/news/deployment-strategies-for-nextjs-apps/ (дата звернення: 08.05.2024).

UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016334638

Дата перевірки:
08.06.2024 11:14:51 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 11:29:31 EEST

ID користувача:
76913

Назва документа: **ІП-з01_Іваницька_ПЗ**

Кількість сторінок: **72** Кількість слів: **9948** Кількість символів: **81585** Розмір файлу: **6.81 MB** ID файлу: **1016135115**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

12.5%
Схожість

Найбільша схожість: **3.37%** з джерелом з Бібліотеки (ID файлу: **1016116309**)

4.97% Джерела з Інтернету	226	Сторінка 74
11.3% Джерела з Бібліотеки	413	Сторінка 76

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	8
Підозріле форматування	15 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК «ОНЛАЙН-РИНОК ЕЛЕКТРОННОЇ КОМЕРЦІЇ ДЛЯ ЦИФРОВИХ ПРОДУКТІВ»

Текст програми

КПІ.ІП-з0102.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юлія ПОЛУПАН

Нормоконтроль:
_____ Олександр СТЕЛЬМАХ

Виконавець:
_____ Оксана ІВАНИЦЬКА

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://github.com/iva-oks/digit-market

Файл sign-up/page.jsx
Реалізація реєстрації користувача на стороні клієнта

```
 "use client";

import { useState } from "react";
import { useRouter } from "next/navigation";
import { useEffect } from "react";
import Link from "next/link";
import "@styles/Signup.css";

const SignUp = () => {
  const navRouter = useRouter();

  const [userForm, setUserForm] = useState({
    login: "",
    email: "",
    password: "",
    confirmPass: "",
    profilePic: null,
  });

  const handleChange = (event) => {
    event.preventDefault();
    const { name: key, value, files: pictures } = event.target;
    setUserForm({
      ...userForm,
      [key]: value,
      [key]: key === "profilePic" ? pictures[0] : value,
    });
  };

  // для перевірки чи підтвердження паролю співпадає
  const [checkPass, setCheckPass] = useState(true);
  useEffect(() => {
    setCheckPass(userForm.password === userForm.confirmPass);
  });

  const handleSubmit = async (event) => {
    event.preventDefault();

    try {
      // об'єкт містить введені дані про користувача разом з файлами
      const signInForm = new FormData();

      for (let property in userForm) {
        signInForm.append(property, userForm[property]);
      }

      const fetchSignUp = await fetch("/api/sign-up/", {
        method: "POST",
        body: signInForm,
      });

      if (fetchSignUp.ok) {
        navRouter.push("/login");
      }
```

```
    } catch (error) {
      console.error(error);
    }
  };



  return (
    <main className="signup">
      <img src="/images/signup.jpg" alt="" className="signup-card" />
      <section className="container">
        <form className="form" onSubmit={handleSubmit}>
          <input
            type="text"
            name="login"
            required
            value={userForm.login}
            placeholder="Логін"
            onChange={handleChange}
          />
          <input
            type="email"
            name="email"
            required
            value={userForm.email}
            placeholder="Електронна пошта"
            onChange={handleChange}
          />
          <input
            type="password"
            name="password"
            required
            value={userForm.password}
            placeholder="Пароль"
            onChange={handleChange}
          />
          <input
            type="password"
            name="confirmPass"
            required
            value={userForm.confirmPass}
            placeholder="Повторіть пароль"
            onChange={handleChange}
          />
          {/* якщо підтвердження паролю не збігається з паролем, виводиться
повідомлення про помилку*/}
          {!checkPass && <p>Пароль не збігається!</p>}
          <label htmlFor="profile-pic">
            <img src="/images/profile.png" alt="зображення профілю" />
            <p>Завантажити зображення профілю</p>
          </label>
          <input
            id=" profile-pic"
            type="file"
            accept="image/*"
            name="profilePic"
            required
            onChange={handleChange}

          />
          {/*відобразити завантажене зображення профілю */}
          {userForm.profilePic && (
            <img
              src={URL.createObjectURL(userForm.profilePic)}
```

```
              alt="зображення профілю"

            />
          )}
          <button type="submit" disabled={!checkPass}>
            Зареєструватися
          </button>
        </form>
        <Link href="/login">Вже маєте акаунт? Увійти →</Link>
      </section>
    </main>
  );
};

export default SignUp;
```

Файл login/page.jsx

Реалізація входу користувача в систему на стороні клієнта

```
"use client";

import "@styles/Login.css";
import { signIn } from "next-auth/react";
import Link from "next/link";
import { useRouter } from "next/navigation";
import { useState } from "react";

const Login = () => {
  const navRouter = useRouter();
  // поля для входу - електронна пошта і пароль
  const [email, setEmail] = useState("");
  const [pass, setPass] = useState("");
  // повідомлення про помилку, якщо пароль не співпадає
  const [matchPass, setMatchPass] = useState("");

  const handleSubmit = async (event) => {
    event.preventDefault();

    try {
      const signInResponse = await signIn("credentials", {
        redirect: false,
        email: email,
        password: pass,
      });

      if (signInResponse.ok) {
        navRouter.push("/");
      } else {
        setMatchPass("Некоректно введені дані!");
      }
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <main className="signin">
      <img src="/images/signin.jpg" alt="" className=" signin-card" />
      <section className="container">
        <form className="form" onSubmit={handleSubmit}>
          <input
            type="email"
            required
```

```
            name="email"
            value={email}
            placeholder="Електронна пошта"
            onChange={(event) => setEmail(event.target.value)}
          />
          <input
            type="password"
            required
            name="password"
            value={pass}
            placeholder="Пароль"
            onChange={(e) => setPass(e.target.value)}
          />
          {/* якщо підтвердження паролю не співпадає з паролем, виводиться
повідомлення про помилку */}
          {matchPass && <p className="pass-error">{matchPass}</p>}
          <button type="submit">Увійти</button>
        </form>
        <Link href="/sign-up">Не маєте акаунту? Зареєструватися →</Link>
      </div>
    </main>
  );
};

export default Login;
```

Файл search/[query]/page.jsx

Реалізація пошуку товару на стороні клієнта

```
"use client";
import Downloading from "@components/Downloading";
import Header from "@components/Header";
import ProductList from "@components/ProductList";
import { useParams } from "next/navigation";
import React, { useState, useEffect } from "react";
import "@styles/Search.css";

const SearchResults = () => {
  // запит пошуку
  const { query } = useParams();

  const [load, setLoad] = useState(true);

  // список знайдених товарів
  const [products, setProducts] = useState([]);

  const getProducts = async () => {
    try {
      const results = await fetch(`/api/product/search/${query}`, {
        // method: "GET",
      });

      const finded = await results.json();
      setProducts(finded);
      setLoad(false);
    } catch (error) {
      console.error(error);
    }
  };

  useEffect(() => {
    getProducts();
  }, [query]);
```

```
  return load ? (
    <Downloading />
  ) : (
    <>
      <Header />

      <h1>Результати пошуку</h1>

      <ProductList productCards={products} />
    </>
  );
};

export default SearchResults;
```

Файл wishlist/page.jsx

Реалізація додавання товару до обраного на стороні клієнта

```
"use client";

import Header from "@components/Header";
import Downloading from "@components/Downloading";
import ProductList from "@components/ProductList";
import { useSession } from "next-auth/react";
import "@styles/Wishlist.css";

const Wishlist = () => {
  // отримуємо інформацію про автентифікованого користувача
  const { data: sessionData } = useSession();
  const wishlist = sessionData?.user?.wishlist;

  return !sessionData ? (
    <Downloading />
  ) : (
    <>
      <Header />

      <h1>Обране</h1>

      <ProductList productCards={wishlist} />
    </>
  );
};

export default Wishlist;
```

Файл cart/page.jsx

Відображення сторінки кошика та реалізація оформлення замовлення на стороні клієнта

```
"use client";

import Header from "@components/Header";
import { useSession } from "next-auth/react";
import Downloading from "@components/Downloading";
import DeleteIcon from "@mui/icons-material/Delete";
import connectToStripe from "@lib/getStripe";
import Link from "next/link";
import ArrowCircleLeftIcon from "@mui/icons-material/ArrowCircleLeft";
import "@styles/Cart.css";
```

```
const Cart = () => {
  // отримуємо інформацію про автентифікованого користувача
  const { data: sessionData, update } = useSession();
  const shoppingCart = sessionData?.user?.cart;
  const userId = sessionData?.user?._id;

  // оновлення стану кошика при додаванні/видаленні з нього товарів
  const updateShoppingCart = async (shoppingCart) => {
    const fetchCart = await fetch(`/api/user/${userId}/cart`, {
      method: "POST",
      body: JSON.stringify({ cart: shoppingCart }),
    });
    const shoppingCartData = await fetchCart.json();
    update({ user: { cart: shoppingCartData } });
  };

  // підрахунок загальної вартості замовлення
  const calcSum = (shoppingCart) => {
    return shoppingCart?.reduce((total, product) => {
      return total + product.price;
    }, 0);
  };
  const sum = calcSum(shoppingCart);

  // видалення товару з кошика
  const del = (delProduct) => {
    const updatedCart = shoppingCart.filter(
      (product) => product.productId !== delProduct.productId
    );
    updateShoppingCart(updatedCart);
  };

  // перехід на сторінку платіжної системи для здійснення покупки
  const handleBuy = async () => {
    const stripe = await connectToStripe();

    const fetchStripe = await fetch("/api/stripe", {
      method: "POST",
      body: JSON.stringify({ cart: shoppingCart, userId }),
    });

    if (fetchStripe.statusCode === 500) {
      // якщо підключення не вдалося
      return;
    }

    // якщо підключення вдалося
    const stripeData = await fetchStripe.json();

    const redirect = stripe.redirectToCheckout({ sessionId: stripeData.id });

    if (redirect.error) {
      console.log(redirect.error.message);
      // toast.error("Something went wrong");
    }
  };

  return !sessionData?.user?.cart ? (
    <Downloading />
  ) : (
    <>
      <Header />
```

```
      <main>
        <div className="cart-container">
          <div className="top">
            <h1>Кошик</h1>
            <h2>
              До сплати: <p>{sum} ₴</p>
            </h2>
          </div>

          {shoppingCart?.length === 0 && <h3>Кошик порожній</h3>}

          {/* Якщо в кошику є товари, вивести деталі товару */}
          {shoppingCart?.length > 0 && (
            <section className="products">
              {shoppingCart?.map((product, i) => (
                <div className="product" key={i}>
                  <div className="product_info">
                    <img src={product.picture} alt="зображення товару" />
                      <p>{product.title}</p>
                      <p>Категорія: {product.category}</p>
                      <p>Продавець: {product.createdBy.login}</p>
                  </div>

                  <div className="pay">
                    <p>{product.price}₴</p>
                  </div>

                  <div className="delete">
                    <DeleteIcon onClick={() => del(product)} />
                  </div>
                </div>
              ))}

              <div className="back">
                <Link href="/">
                  <ArrowCircleLeftIcon />
                  Продовжити покупки
                </Link>
                <button onClick={handleBuy}>Оформити замовлення</button>
              </div>
            </section>
          )}
        </div>
      </main>
    </>
  );
};

export default Cart;
```

Файл my-products/page.jsx

Відображення сторінки зі списком товарів користувача

```
"use client";

import Downloading from "@components/Downloading";
import Header from "@components/Header";
import ProductList from "@components/ProductList";
import { useSession } from "next-auth/react";
import { useSearchParams } from "next/navigation";
import React, { useEffect, useState } from "react";
import "@styles/MyProducts.css";
```

```
const MyProducts = () => {
  const [load, setLoad] = useState(true);
  // отримуємо інформацію про автентифікованого користувача
  const { data: sessionData } = useSession();
  const currentId = sessionData?.user?._id;
  // отримуємо id користувача з бази даних
  const searchParams = useSearchParams();
  const DBUserId = searchParams.get("id");
  const [currentUser, setCurrentUser] = useState({});

  // список товарів, створених користувачем
  const [products, setProducts] = useState([]);
  useEffect(() => {
    const getProducts = async () => {
      const response = await fetch(`api/user/${DBUserId}/my-products`, {
        method: "GET",
        headers: {
          "Content-Type": "application/json",
        },
      });
      const myProducts = await response.json();
      setProducts(myProducts.products);
      setCurrentUser(myProducts.user);
      setLoad(false);
    };

    if (DBUserId) {
      getProducts();
    }
  }, [DBUserId]);

  return load ? (
    <Downloading />
  ) : (
    <>
      <Header />
      {/*  для автора товару */}
      {currentId === DBUserId && <h1>Ваші товари</h1>}
      {/*  для інших користувачів */}
      {currentId !== DBUserId && (
        <h1>Товари продавця: {currentUser.login}</h1>
      )}

      <ProductList productCards={products} />
    </>
  );
};

export default MyProducts;
```

Файл order/page.jsx

Реалізація завантаження придбаного товару та перегляду інформації про замовлення.

```
"use client";

import Header from "@components/Header";
import { useSession } from "next-auth/react";
import "@styles/Order.css";

const Order = () => {
  // отримуємо інформацію про автентифікованого користувача
```

```
  const { data: sessionData } = useSession();
  const orders = sessionData?.user?.orders;

  // створення посилання на завантаження файлу
  const fileLink = (path) => {
    const fileName = path.split("/").pop();
    const a = document.createElement("a");
    a.href = path;
    a.setAttribute("download", fileName);
    document.body.appendChild(a);
    a.click();
    a.remove();
  };

Const orderDate = new Date(order.date);

  return (
    <>
      <Header />
      <main>
        <h1>Ваші замовлення</h1>
        <section className="orders">
          {orders?.map((order, i) => (
            <div className="order-info" key={i}>
              <p>
                Дата проведення замовлення:
                { orderDate.toString() }
              </p>
              <div className="title">
                <p>ID замовлення: {order.id}</p>
                <p>Вартість замовлення: {order.paid} ₴</p>
              </div>

              <div className="products">
                {order.products.map((product, i) => (
                  <div className="product" key={i}>
                    <div className="card">
                      <img src={product.picture} alt="зображення товару" />
                      <div className="details">
                        <h4>{product.title}</h4>
                      </div>
                    </div>

                    <div className="paid">
                      <h3>Ціна: ${product.price}</h3>

                      <button
                        className="download-button"
                        onClick={() => {
                          fileLink(product.productFile);
                        }}
                      >
                        Завантажити файл
                      </button>
                    </div>
                  </div>
                ))}
              </div>
            </div>
          ))}
        </section>
      </main>
    </>
  );
```

```
};

export default Order;
```

Файл admin-panel/page.jsx

Відображення сторінки адміністратора

```
"use client";

import Downloading from "@components/Downloading";
import Header from "@components/Header";
import ProductList from "@components/ProductList";
import { useParams } from "next/navigation";
import React, { useState, useEffect } from "react";
import "@styles/Admin.css";

const AdminPanel = () => {
  const [load, setLoad] = useState(true);

  // список товарів, що чекають перевірки
  const [products, setProducts] = useState([]);

  const getProducts = async () => {
    try {
      const results = await fetch(`/api/product/pending/`, {
        // method: "GET",
      });

      const finded = await results.json();
      setProducts(finded);
      setLoad(false);
    } catch (error) {
      console.error(error);
    }
  };

  useEffect(() => {
    getProducts();
  }, [0]);

  return load ? (
    <Downloading />
  ) : (
    <>
      <Header />

      <h1>Панель адміністратора</h1>

      <ProductList productCards={products} />
    </>
  );
};

export default AdminPanel;
```

Файл payment-result/page.jsx

Реалізація отримання відповіді про успішну оплату на стороні клієнта

```
import "@styles/PaymentResult.css";
import Link from "next/link";

const PaymentResult = () => {
```

```
  return (
    <main>
      <h1>Оплата пройшла успішно</h1>
      <p>Дякуємо за покупку!</p>
      <Link href="/order">Переглянути замовлення</Link>
    </main>
  );
};

export default PaymentResult;
```

Файл add-product/page.jsx

Реалізація створення товару на стороні клієнта.

```
"use client";

import React, { useState } from "react";
import Form from "@components/Form";
import Header from "@components/Header";
import { useSession } from "next-auth/react";
import { useRouter } from "next/navigation";

const AddProduct = () => {
  // отримуємо інформацію про автентифікованого користувача
  const { data: sessionData } = useSession();
  const navRouter = useRouter();

  const [product, setProduct] = useState({
    createdBy: "",
    title: "",
    description: "",
    category: "",
    price: "",
    pictures: [],
    files: [],
  });

  // зареєстрований користувач автоматично встановлюється автором товару
  if (sessionData) {
    product.createdBy = sessionData?.user?._id;
  }

  const handleSubmit = async (event) => {
    event.preventDefault();
    try {
      // об'єкт містить введені дані про продукт разом з файлами
      const productFormData = new FormData();

      for (let property in product) {
        productFormData.append(property, product[property]);
      }

      // заповнюємо список зображень товару
      product.pictures.forEach((picture) => {
        productFormData.append("productPics", picture);
      });

      product.files.forEach((file) => {
        productFormData.append("productFile", file);
      });

      const fetchAddProduct = await fetch("/api/product/add", {
        method: "POST",
```

```
        body: productFormData,
      });

      // якщо створення товару пройшло успішно, користувач перенаправляється
на сторінку своїх товарів
      if (fetchAddProduct.ok) {
        navRouter.push(`/my-products?id=${sessionData?.user?._id}`);
      }
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <>
      <Header />
      <Form
        product={product}
        setProduct={setProduct}
        handleSubmit={handleSubmit}
      />
    </>
  );
};

export default AddProduct;
```

Файл edit-product/page.jsx

Реалізація оновлення товару на стороні клієнта.

```
"use client";
import Header from "@components/Header";
import { useSearchParams } from "next/navigation";
import React, { useEffect, useState } from "react";
import Downloading from "@components/Downloading";
import Form from "@components/Form";
import { useRouter } from "next/navigation";
import { useSession } from "next-auth/react";

const EditProduct = () => {
  const navRouter = useRouter();
  // отримуємо інформацію про автентифікованого користувача
  const { data: sessionData } = useSession();

  // отримуємо інформацію про товар
  const urlParams = useSearchParams();
  const productId = urlParams.get("id");

  const [product, setProduct] = useState({
    title: "",
    category: "",
    description: "",
    price: "",
    pictures: [],
  });

  // для відображення анімації завантаження
  const [load, setLoad] = useState(true);

  useEffect(() => {
    const getProductInfo = async () => {
      const fetchProduct = await fetch(`api/product/${productId}`, {
        // method: "GET",
```

```
        // headers: {
        //   "Content-Type": "application/json",
        // },
      });
      const productInfo = await fetchProduct.json();

      setProduct({
        title: productInfo.title,
        category: productInfo.category,
        description: productInfo.description,
        price: productInfo.price,
        pictures: productInfo.productPics,
      });

      setLoad(false);
    };

    if (productId) {
      getProductInfo();
    }
  }, [productId]);

  const handleSubmit = async (event) => {
    event.preventDefault();

    try {
      // об'єкт містить введені дані про продукт разом з файлами
      const productFormData = new FormData();

      for (let property in product) {
        productFormData.append(property, product[property]);
      }

      product.pictures.forEach((picture) => {
        productFormData.append("productPics", picture);
      });

      // запит на збереження оновлених даних у базі даних
      const editFetch = await fetch(`/api/product/${productId}`, {
        method: "PATCH",
        body: productFormData,
      });

      // якщо збережено успішно перенаправлення на сторінку товарів
користувача
      if (editFetch.ok) {
        navRouter.push(`/my-products?id=${sessionData?.user?._id}`);
      }
    } catch (error) {
      console.error(error);
    }
  };

  return load ? (
    <Downloading />
  ) : (
    <>
      <Header />
      <Form
        // type="Edit"
        product={product}
        setProduct={setProduct}
        handleSubmit={handleSubmit}
      />
```

```
    </>
  );
};

export default EditProduct;
```

Файл product-info/page.jsx

Відображення сторінки з інформацією про товар та реалізація додавання товару в кошик, зміни статусу товару на стороні клієнта.

```
"use client";

import "@styles/ProductInfo.css";
import { useSearchParams } from "next/navigation";
import { useEffect, useState } from "react";
import Downloading from "@components/Downloading";
import Header from "@components/Header";
import ModeEditIcon from "@mui/icons-material/ModeEdit";
import FavoriteIcon from "@mui/icons-material/Favorite";
import FavoriteBorderIcon from "@mui/icons-material/FavoriteBorder";
import KeyboardArrowLeftIcon from "@mui/icons-material/KeyboardArrowLeft";
import KeyboardArrowRightIcon from "@mui/icons-material/KeyboardArrowRight";
import ShoppingCartIcon from "@mui/icons-material/ShoppingCart";
import { useSession } from "next-auth/react";
import { useRouter } from "next/navigation";

// сторінка з детальною інформацією про товар
const ProductInfo = () => {
  const navRouter = useRouter();

  // отримуємо інформацію про автентифікованого користувача
  const { data: sessionData, update } = useSession();
  const userId = sessionData?.user?._id;
  const cart = sessionData?.user?.cart;
  const role = sessionData?.user?.role;

  // отримуємо інформацію про товар
  const urlParams = useSearchParams();
  const productId = urlParams.get("id");

  const [product, setProduct] = useState({});

  // для відображення анімації завантаження
  const [load, setLoad] = useState(true);

  useEffect(() => {
    const getProductInfo = async () => {
      const fetchProduct = await fetch(`api/product/${productId}`, {
        // method: "GET",
      });
      const productInfo = await fetchProduct.json();
      setProduct(productInfo);
      setLoad(false);
    };

    if (productId) {
      getProductInfo();
    }
  }, [productId]);

  // Слайдер для зображень
  const [index, setIndex] = useState(0);
```

```
// наступне зображення
const nextPicture = () => {
  setIndex((currIndex) => (currIndex + 1) % product.productPics.length);
};

// попереднє зображення
const prevPicture = () => {
  setIndex(
    (currIndex) =>
      (currIndex - 1 + product.productPics.length) %
      product.productPics.length
  );
};

// показати більше зображень
const [shownPictures, setShownPictures] = useState(4);

const getMorePictures = () => {
  setShownPictures(product.productPics.length);
};

// відобразити зображення на слайдері
const [currentPicture, setCurrentPicture] = useState(0);

const handleShowPicture = (i) => {
  setCurrentPicture(i);
  setIndex(i);
};

// отримати інформацію про список обраного користувача
const wishlist = sessionData?.user?.wishlist;

// додати товар до списку обраного
const isWished = wishlist?.find(
  (wishedProduct) => wishedProduct?._id === product._id
);

// якщо користувач не автентифікований, опція додати до обраного не активна
const checkAddToWish = async () => {
  if (!sessionData) {
    navRouter.push("/login");
    return;
  }

  // зберегти оновлений вішліст
  const updatedUser = await fetch(
    `api/user/${userId}/wishlist/${product._id}`,
    {
      method: "PATCH",
    }
  );

  // оновити дані про вішліст в сесії поточного користувача
  const updatedData = await updatedUser.json();
  update({ user: { wishlist: updatedData.wishlist } });
};

// перевірка, чи товар вже додано до кошика
const isAddedToCart = cart?.find(
  (product) => product?.productId === productId
);

// додати товар в кошик
```

```
const addToCart = async () => {
  // якщо користувач не автентифікований, опція додати до кошика не активна
  if (!sessionData) {
    navRouter.push("/login");
    return;
  }

  const addedProduct = {
    productId: productId,
    title: product.title,
    createdBy: product.createdBy,
    category: product.category,
    price: product.price,
    picture: product.productPics[0],
    file: product.productFile[0],
  };

  if (!isAddedToCart) {
    const updatedCart = [...cart, addedProduct];

    // зберігаюмо дані про оновлений кошик в базу даних
    try {
      await fetch(`/api/user/${userId}/cart`, {
        method: "POST",
        // headers: {
        //   "Content-Type": "application/json",
        // },
        body: JSON.stringify({ cart: updatedCart }),
      });
      update({ user: { cart: updatedCart } });
    } catch (error) {
      console.error(error);
    }
  } else {
    confirm("Товар вже знаходиться в кошику!");
    return;
  }
};


// створення посилання на завантаження файлу
const fileLink = (path) => {
  const fileName = path.split("/").pop();
  const a = document.createElement("a");
  a.href = path;
  a.setAttribute("download", fileName);
  document.body.appendChild(a);
  a.click();
  a.remove();
};

// зміна статусу
const [productStatus, setProductStatus] = useState("pending");

const handleSubmit = async (event) => {
  event.preventDefault();
  try {
    const productFormData = new FormData();
    productFormData.append("status", productStatus);

    // запит на збереження статусу у базі даних
    const statusFetch = await fetch(`/api/product/status/${productId}`, {
      method: "PATCH",
      body: productFormData,
```

```
      });

      // якщо збережено успішно
      if (statusFetch.ok) {
        window.location.reload();
      }
    } catch (error) {
      console.error(error);
    }
  };

  return load ? (
    <Downloading />
  ) : (
    <>
      <Header />
      <main>
        <div className="container">
          <h1>{product.title}</h1>
          {product?.createdBy?._id === userId ? (
            // для автора товару відображається кнопка Редагувати товар
            <div
              onClick={() => {
                navRouter.push(`/edit-product?id=${productId}`);
              }}
            >

              <ModeEditIcon />
              <p>Редагувати</p>
            </div>
          ) : (
            // для інших користувачів відображається кнопка Додати до
обраного
            <div onClick={checkAddToWish}>
              {isWished ? (
                <FavoriteIcon />
              ) : (
                <FavoriteBorderIcon />
              )}
              <p>В обране</p>
            </div>
          )}
        </div>

        <div className="slider">
          <div
            className="slides">
            {product.productPics?.map((picture, i) => (
              <div className="slide" key={i}>
                <img src={picture} alt="зображення товару" />
                <div
                  className="prev"
                  onClick={(event) => prevPicture(event)}
                >
                  <KeyboardArrowLeftIcon />
                </div>
                <div
                  className="next"
                  onClick={(event) => nextPicture(event)}
                >
                  <KeyboardArrowRightIcon />
                </div>
              </div>
            ))}
```

```
          </div>
        </div>

        <div className="pictures">
          {product.productPics?.slice(0, shownPictures).map((picture, i) => (
            <img
              src={picture}
              alt="зображення товару"
              key={i}
              onClick={() => handleShowPicture(i)}
              className={currentPicture === i ? "current" : ""}
            />
          ))}

          {/* якщо зображень більше ніж 4, доступна кнопка Більше */}
          {shownPictures < product.productPics.length && (
            <div className="more" onClick={getMorePictures}>
              <KeyboardArrowRightIcon />
              Більше
            </div>
          )}
        </div>

        <hr />

        <div className="user-profile">
          {/* при натисканні на фото автора товару, перенаправлення на
сторінку товарів даного автора */}
          <img
            src={product.createdBy.profilePic}
            alt="користувач"
            onClick={() =>
              navRouter.push(`/my-products?id=${product.createdBy._id}`)
            }
          />
          <h3>Автор: {product.createdBy.login}</h3>
        </div>

        <hr />

        <h3>Інформація про товар</h3>
        <p>{product.description}</p>

        <h1>{product.price} ₴</h1>
        <button type="submit" onClick={addToCart}>
          <ShoppingCartIcon />
          Додати в кошик
        </button>

        {role === "admin" && (
          <>
            <button
              className="download-button"
              onClick={() => {
                fileLink(product.productFile[0]);
              }}
            >
              Завантажити файл
            </button>

            <button
              className="complaint-button"
              onClick={ handleFileComplaint }
            >
```

```
              Порушення авторських прав
            </button>

            <br />
            {/* змінити статус */}
            <form onSubmit={handleSubmit}>
              <div className="select">
                <label htmlFor="standard-select">Статус:</label>
                <br />
                <br />
                <select
                  id="standard-select"
                  name="status"
                  value={productStatus}
                  onChange={(e) => {
                    const selectedStatus = e.target.value;
                    setProductStatus(selectedStatus);
                  }}
                >
                  <option value="pending">Чекає перевірки</option>
                  <option value="approved">Підтверджено</option>
                  <option value="denied">Відмовлено</option>
                </select>
              </div>

              <button className="submit_btn" type="submit">
                Змінити статус
              </button>
            </form>
          </>
        )}
      </main>
    </>
  );
};

export default ProductInfo;
```

Файл page.jsx

Домашня сторінка додатку

```
import Content from "@components/Content";
import Header from "@components/Header";

const Home = () => {
  return (
    <>
      <Header />
      <Content />
    </>
  );
};

export default Home;
```

Файл layout.jsx

Точка входу в додаток

```
import "@styles/globals.css";
import Provider from "@components/Provider";

export const metadata = {
  title: "Маркетплейс",
  description: "Ваш Маркетплейс для цифрових продуктів",
```

```
};

const layout = ({ children }) => {
  return (
    <html lang="ua">
      <body>
        <Provider>
          <main>{children}</main>
        </Provider>
      </body>
    </html>
  );
};

export default layout;
```

Файл api/sign-up/route.js

Реалізація реєстрації користувача в системі на стороні сервера

```
import { mongoDB } from "@mongodb/database";
import User from "@models/User";
import { NextResponse } from "next/server";
import { hash } from "bcryptjs";
import { writeFile } from "fs/promises";

/* Реєстрація користувача */
export async function POST(request) {
  try {
    // Підключення до бази даних
    await mongoDB();

    // Отримуємо дані з форми реєстрації
    const userFormData = await request.formData();
    const login = userFormData.get("login");
    const email = userFormData.get("email");
    const password = userFormData.get("password");
    const profilePic = userFormData.get("profilePic");

    // зберігаємо зображення на диск
    const temp = await profilePic.arrayBuffer();
    const profilePicBuffer = Buffer.from(temp);
    // const profilePicPath = `C:/Projects/digit-
market/public/uploads/${profilePic.name}`;
    await writeFile(
      `C:/Projects/digit-market/public/uploads/${profilePic.name}`,
      profilePicBuffer
    );

    // перевіряємо, чи користувач вже існує в базі даних
    const isEmailExist = await User.findOne({ email });
    if (isEmailExist) {
      return NextResponse.json(
        { message: "Обліковий запис вже існує" },
        { status: 409 }
      );
    }

    // хешування паролю
    const passHashed = await hash(password, 8);

    // створення нового користувача
    const user = new User({
      login: login,
```

```
      email,
      password: passHashed,
      profilePic: `/uploads/${profilePic.name}`,
    });

    // збереження створеного користувача в базі даних
    await user.save();

    // при успішному створенні повертається повідомлення і статус 200
    return NextResponse.json(
      { message: "Обліковий запис успішно створено", user: user },
      { status: 200 }
    );
  } catch (error) {
    console.error(error);
    return NextResponse.json(
      { message: "Помилка при створенні облікового запису" },
      { status: 500 }
    );
  }
}
```

Файл api/auth/route.js

Реалізація входу користувача в систему на стороні сервера

```
import NextAuth from "next-auth";
import CredentialsProvider from "next-auth/providers/credentials";
import { compare } from "bcryptjs";

import { mongoDB } from "@mongodb/database";
import User from "@models/User";

const authentification = NextAuth({
  providers: [
    CredentialsProvider({
      name: "Credentials",
      async authorize(credentials, req) {
        if (!credentials.email) {
          throw new Error("Неправильна пошта!");
        } else if (!credentials.password) {
          throw new Error("Неправильний пароль!");
        }

        /* При вході виконується пошук користувача в базі даних */
        await mongoDB();
        const user = await User.findOne({ email: credentials.email });
        /* Якщо електронну пошту не знайдено */
        if (!user) {
          throw new Error("Неправильна пошта!");
        }
        /* Чи співпадають паролі в сесії користувача і в базі даних */
        const comparePassword = await compare(
          credentials.password,
          user.password
        );
        /* Якщо пароль не знайдено */
        if (!comparePassword) {
          throw new Error("Неправильний пароль!");
        }

        return user;
      },
    }),
```

```
  ],

  secret: process.env.AUTHENTIFICATION_SECRET,

  callbacks: {
    // доступ до даних користувача, автентифікованого в поточній сесії
    async session({ session: authSession }) {
      const authUser = await User.findOne({ email: authSession.user.email });
      authSession.user.id = authUser._id.toString();

      // об'єднує дані про користувача з сесії та з бази даних
      authSession.user = { ...authSession.user, ...authUser._doc };

      return authSession;
    },
  },
});

export { authentification as GET, authentification as POST };
```

Файл api/order/route.js

Реалізація створення нового замовлення на стороні сервера

```
import User from "@models/User";
import { mongoDB } from "@mongodb/database";

const stripe = require("stripe")(process.env.NEXT_PUBLIC_STRIPE_SECRET_KEY);

// отримуємо список товарів в кошику
async function getProductsInCart(productsFromStripe) {
  return new Promise((result, reject) => {
    let productsInCart = [];

    productsFromStripe?.data?.forEach(async (p) => {
      const product = await stripe.products.retrieve(p.price.product);
      const productId = product.metadata.productId;
      const productFile = product.metadata.productFile;

      productsInCart.push({
        productId,
        title: product.name,
        price: p.price.unit_amount_decimal / 100,
        picture: product.images[0], // назва поля за замовченням в stripe
        productFile: productFile,
      });

      if (productsInCart.length === productsFromStripe?.data?.length) {
        result(productsInCart);
      }
    });
  });
}

export const POST = async (request, res) => {
  try {
    const requestBody = await request.text();
    const stripeHeader = request.headers.get("stripe-signature");

    // stripe реагуватиме на event успішної оплати та у відповідь
генеруватиметься замовлення
    const successEvent = stripe.webhooks.constructEvent(
      requestBody,
      stripeHeader,
```

```
      process.env.STRIPE_WEBHOOK_SECRET
    );

    if (successEvent.type === "checkout.session.completed") {
      const successSession = successEvent.data.object;

      const orderInfo = await stripe.checkout.sessions.listLineItems(
        successEvent.data.object.id
      );

      // отримуємо інформацію про товари в замовленні
      const orderProducts = await getProductsInCart(orderInfo);
      const userId = successSession.client_reference_id;

      // вартість замовлення
      const paid = successSession.amount_total / 100;

      const order = {
        id: successSession.payment_intent,
        user: userId,
        products: orderProducts,
        paid: paid,
        date: new Date(),
      };

      // зберігаємо замовлення в базі даних
      await mongoDB();
      // додаємо замовлення в список замовлень знайденого користувача
      const user = await User.findById(userId);
      user.orders.push(order);
      // обнуляємо кошик користувача
      user.cart = [];
      await user.save();

      return new Response(JSON.stringify({ received: true }), { status: 200
});
    }
  } catch (error) {
    console.error(error);
    return new Response("Не вдалося створити замовлення", { status: 500 });
  }
};
```

Файл api/user/[id]/cart/route.js

Реалізація функції додавання товарів в кошик

```
import { mongoDB } from "@mongodb/database";
import User from "@models/User";

export const POST = async (request, { params }) => {
  // збереження кошика користувача в бд
  try {
    const { cart: productsInCart } = await request.json();
    // підключення до бази даних
    await mongoDB();
    // знаходження користувача по id
    const id = params.id;
    const user = await User.findById(id);
    // збереження кошика
    user.cart = productsInCart;
    await user.save();
    return new Response(JSON.stringify(user.cart), { status: 200 });
  } catch (error) {
    console.error(error);
    return new Response("Помилка при оновленні кошика", { status: 500 });
```

```
  }
};
```

Файл api/user/[id]/my-products/route.js

Отримання товарів створених користувачем

```
import Product from "@models/Product";
import User from "@models/User";
import { mongoDB } from "@mongodb/database";

export const GET = async (request, { params }) => {
  try {
    // підключення до БД
    await mongoDB();
    // знаходження користувача в базі даних
    const user = await User.findById(params.id);
    // знаходження всіх товарів, у яких createdBy - знайдений користувач
    const products = await Product.find({ createdBy: params.id }).populate(
      "createdBy"
    );

    user.products = products;
    await user.save();

    // повертає товари, створені користувачем
    return new Response(JSON.stringify({ user: user, products: products }), {
      status: 200,
    });
  } catch (error) {
    return new Response("Помилка при отриманні створених товарів
користувача", {
      status: 500,
    });
  }
};
```

Файл api/user/[id]/wishlist/route.js

Реалізація функції додавання товарів в обране

```
import Product from "@models/Product";
import User from "@models/User";
import { mongoDB } from "@mongodb/database";

export const PATCH = async (request, { params }) => {
  try {
    // отримуємо id користувача та товару з запиту
    const userId = params.id;
    const productId = params.productId;

    // підключення до БД
    await mongoDB();

    // отримуємо дані про користувача та товар з БД
    const user = await User.findById(userId);
    const product = await Product.findById(productId).populate("createdBy");

    // перевіряємо, чи знаходиться товар в списку обраного
    const wishedProduct = user.wishlist.find(
      (product) => product._id.toString() === productId
    );

    // якщо товар вже знаходиться в бажаному
    if (wishedProduct) {
      // товар видаляється з бажаного
```

```
      user.wishlist = user.wishlist.filter(
        (product) => product._id.toString() !== productId
      );
      await user.save();
      return new Response(
        JSON.stringify({
          message: "Товар видалено зі списку обраного",
          wishlist: user.wishlist,
        }),
        { status: 200 }
      );
    } else {
      // якщо товар не знаходиться в бажаному, він додається в список
      user.wishlist.push(product);
      await user.save();
      return new Response(
        JSON.stringify({
          message: "Товар додано до списку обраного",
          wishlist: user.wishlist,
        }),
        { status: 200 }
      );
    }
  } catch (error) {
    console.error(error);
    return new Response("Помилка при оновленні списку обраного", {
      status: 500,
    });
  }
};
```

Файл api/product/[id]/route.js

Реалізація функції оновлення товарів

```
import Product from "@models/Product";
import { mongoDB } from "@mongodb/database";

// отримуємо товар з бази даних
export const GET = async (request, { params }) => {
  try {
    // підключення до БД
    await mongoDB();

    const product = await Product.findById(params.id).populate("createdBy");
    if (!product) return new Response("Товар не знайдено", { status: 404 });

    return new Response(JSON.stringify(product), { status: 200 });
  } catch (err) {
    return new Response("Помилка при отриманні товару", { status: 500 });
  }
};

// оновлення товару
export const PATCH = async (request, { params }) => {
  try {
    await mongoDB();

    const productFormData = await request.formData();

    // Отримуємо передані дані
    const title = productFormData.get("title");
    const createdBy = productFormData.get("createdBy");
    const category = productFormData.get("category");
    const description = productFormData.get("description");
    const price = productFormData.get("price");
```

```
    // Отримуємо список зображень
    const productPics = [];
    const pictures = productFormData.getAll("productPics");

    // Збереження зображень
    for (const picture of pictures) {
      // Якщо додано нове зображення - тип Object
      if (picture instanceof Object) {
        // зберігаємо зображення на диск
        const temp = await picture.arrayBuffer();
        const productPicBuffer = Buffer.from(temp);
        await writeFile(
          `C:/Projects/digit-market/public/uploads/${picture.name}`,
          productPicBuffer
        );

        // Додаємо посилання на зображення в базу даних
        productPics.push(`/uploads/${picture.name}`);
      } else {
        // Якщо зображення залишилось без змін - тип String
        productPics.push(picture);
      }
    }

    // Знаходимо товар для оновлення в базі даних
    const findProduct = await Product.findById(params.id);
    if (!findProduct) {
      return new Response("Товар не знайдено", { status: 404 });
    }

    // Оновлюємо дані
    findProduct.title = title;
    findProduct.category = category;
    findProduct.description = description;
    findProduct.price = price;
    findProduct.productPics = productPics;
    // запис та збереження даних
    await findProduct.save();

    return new Response("Товар оновлено", { status: 200 });
  } catch (error) {
    console.error(error);
    return new Response("Помилка при оновленні товару", { status: 500 });
  }
};

// видалення товару
export const DELETE = async (request, { params }) => {
  try {
    await mongoDB();
    await Product.findByIdAndDelete(params.id);

    return new Response("Товар успішно видалено", { status: 200 });
  } catch (error) {
    console.error(error);
    return new Response("Помилка при видаленні товару", { status: 500 });
  }
};
```

Файл api/product/add/route.js

Реалізація функції створення товарів

```
import { mongoDB } from "@mongodb/database";
import { writeFile } from "fs/promises";
import Product from "@models/Product";

export async function POST(request) {
  try {
    // підключення до БД
    await mongoDB();

    // Отримуємо передані дані
    const productFormData = await request.formData();
    const title = productFormData.get("title");
    const createdBy = productFormData.get("createdBy");
    const category = productFormData.get("category");
    const description = productFormData.get("description");
    const price = productFormData.get("price");

    // Отримуємо зображення товару
    const productPics = [];
    const pictures = productFormData.getAll("productPics");
    // Отримуємо файли товару
    const files = productFormData.getAll("productFile");

    // Збереження зображень
    for (const picture of pictures) {
      // зберігаємо зображення на диск
      const temp = await picture.arrayBuffer();
      const productPicBuffer = Buffer.from(temp);

      await writeFile(
        `C:/Projects/digit-market/public/uploads/${picture.name}`,
        productPicBuffer
      );

      // Додаємо посилання на зображення в базу даних
      productPics.push(`/uploads/${picture.name}`);
    }

    // Збереження файлу
    const productFile = [];
    for (const myfile of files) {
      const fileBytes = await myfile.arrayBuffer();
      const fileBuffer = Buffer.from(fileBytes);
      const fileP = `C:/Projects/digit-market/public/uploads/${myfile.name}`;
      await writeFile(fileP, fileBuffer);
      productFile.push(`/uploads/${myfile.name}`);
    }

    // Створення нового товару
    const newProduct = new Product({
      title,
      createdBy,
      category,
      description,
      price,
      productPics: productPics,
      productFile: productFile,
    });
    // Збереження створеного товару в БД
    await newProduct.save();

    return new Response(JSON.stringify(newProduct), { status: 200 });
  } catch (error) {
    console.error(error);
```

```
    return new Response("Помилка при створенні товару", { status: 500 });
  }
}
```

Файл api/product/list/[category]/route.js

Реалізація функції отримання товарів за категорією

```
import { mongoDB } from "@mongodb/database";
import Product from "@models/Product";

export const GET = async (request, { params }) => {
  try {
    let products;
    // підключення до БД
    await mongoDB();

    const { category: selected } = params;

    // отримання з бази даних товарів та користувачів
    if (selected !== "All") {
      // пошук товарів обраної категорії
      products = await Product.find({ category: selected }).populate(
        "createdBy"
      );
    } else {
      // всі товари
      products = await Product.find().populate("createdBy");
    }


    return new Response(JSON.stringify(products), { status: 200 });
  } catch (error) {
    console.error(error);
    return new Response("Помилка при отриманні товарів", { status: 500 });
  }
};
```

Файл api/product/status/[id]/route.js

Реалізація функції зміни статусу товарів

```
import Product from "@models/Product";
import { mongoDB } from "@mongodb/database";

// оновлення статусу
export const PATCH = async (request, { params }) => {
  try {
    await mongoDB();

    const productFormData = await request.formData();

    // Отримуємо передані дані
    const status = productFormData.get("status");

    // Знаходимо товар для оновлення в базі даних
    const findProduct = await Product.findById(params.id);
    if (!findProduct) {
      return new Response("Товар не знайдено", { status: 404 });
    }

    // Оновлюємо дані
    findProduct.status = status;

    // запис та збереження даних
    await findProduct.save();
```

```
    return new Response("Статус оновлено", { status: 200 });
  } catch (error) {
    console.error(error);
    return new Response("Помилка при оновленні статусу", { status: 500 });
  }
};
```

Файл api/product/search/[query]/route.js
Реалізація функції пошуку товарів

```
import Product from "@models/Product";
import { mongoDB } from "@mongodb/database";

export const GET = async (request, { params }) => {
  // отримання результатів пошуку
  try {
    // підключення до БД
    await mongoDB();

    const { query: productName } = params;
    let products = [];

    if (productName === "all") {
      // пошук за категорією
      products = await Product.find().populate("createdBy");
    } else {
      // пошук за назвою товару
      products = await Product.find({
        $or: [
          { category: { $regex: productName, $options: "i" } }, // без
врахування регістра
          { title: { $regex: productName, $options: "i" } },
        ],
      }).populate("createdBy");
    }

    if (!products)
      return new Response("Товар за запитом не знайдено", { status: 404 });

    return new Response(JSON.stringify(products), { status: 200 });
  } catch (error) {
    console.error(error);
    return new Response("Помилка пошуку товару", { status: 500 });
  }
};
```

Компоненти:

Файл Content.jsx
Використовується на головній сторінці.

```
"use client";

import { categories } from "@data";
import ProductList from "./ProductList";
import { useEffect, useState } from "react";
import Downloading from "./Downloading";
import "@styles/Content.css";
```

```
const Content = () => {
  // Якщо елементи завантажуються, на сторінці відображається анімація
завантаження
  const [loader, setLoader] = useState(true);
  // Обрана категорія для фільтрації
  const [activeCat, setActiveCat] = useState("All");
  // Список всіх товарів за категорією
  const [productList, setProductList] = useState([]);

  const getProductList = async () => {
    const result = await fetch(`/api/product/list/${activeCat}`);
    const productList = await result.json();
    setProductList(productList);
    setLoader(false);
  };

  useEffect(() => {
    getProductList();
  }, [activeCat]);

  return loader ? (
    <Downloading />
  ) : (
    <>
      <section className="select-category">
        {categories?.map((category, i) => (
          <p
            key={i}
            onClick={() => setActiveCat(category)}
            className={`${category === activeCat ? "current" : ""}`}
          >
            {category === "All" ? "Всі товари" : category}
          </p>
        ))}
      </section>

      <ProductList productCards={productList} />
    </>
  );
};

export default Content;
```

Файл Header.jsx

Містить навігацію та пошук.

```
"use client";
import SearchIcon from "@mui/icons-material/Search";
import ShoppingCartIcon from "@mui/icons-material/ShoppingCart";
import AccountCircleIcon from "@mui/icons-material/AccountCircle";
import MenuIcon from "@mui/icons-material/Menu";
import { signOut as logout, useSession } from "next-auth/react";
import Link from "next/link";
import React, { useState } from "react";
import { useRouter } from "next/navigation";
import "@styles/Header.scss";


const Header = () => {
  const navRouter = useRouter();

  // отримуємо інформацію про автентифікованого користувача
```

```
  const { data: sessionData } = useSession();
  const user = sessionData?.user;
  // отримуємо інформацію про продукти в кошику користувача
  const shoppingCart = user?.cart;
  const role = user?.role;

  // опції в меню навігації залежать від того, користувач автентифікований,
чи ні
  const [nav, setNav] = useState(false);

  // запит при пошуку товару
  const [searchQuery, setSearchQuery] = useState("");
  // перенаправлення на сторінку результатів пошуку
  const searchForProduct = async () => {
    navRouter.push(`/search/${searchQuery}`);
  };

  // після натискання кнопки вийти користувач перенаправляється на сторінку
входу
  const handleExit = async () => {
    logout({ to: "/login" });
  };

  return (
    <header className="header">
      <a href="/">
         <img src="logo.svg" alt="На головну" />
      </a>

      <div className="adminButton">
        {role === "admin" && (
          <Link href="/admin-panel">Панель адміністратора</Link>
        )}
      </div>

      <div className="search">
        <input
          type="text"
          placeholder="Пошук..."
          value={searchQuery}
          onChange={(e) => setSearchQuery(e.target.value)}
        />
        <SearchIcon color="blue" onClick={searchForProduct} />
      </div>

      <div className="user">
        {/* якщо користувач автентифікований, відображається кнопка кошика
*/}
        {user && (
          <Link href="/cart" className="cart">
            <ShoppingCartIcon color="gray" />
            Кошик <small>({shoppingCart?.length})</small>
          </Link>
        )}

        {/* Кнопка відобразити меню навігації */}
        <button className="nav" onClick={() => setNav(!nav)}>
          <MenuIcon />
          {!user ? (

            <AccountCircleIcon color="gray" />
          ) : (
            <img
              src={user.profilePic}
```

```
              alt="користувач"

            />
          )}
        </button>

        {/* гість може  Увійти або Зареєструватися*/}
        {nav && !user && (
          <div className="nav">
            <Link href="/sign-up">Зареєструватися</Link>
            <Link href="/login">Увійти</Link>
          </div>
        )}

        {/* Навігація по сайту доступна зареєстрованому користувачу*/}
        {nav && user && (
          <div className="nav">
            <Link href="/add-product">Додати товар</Link>
            <Link href={`/my-products?id=${user._id}`}>Ваші товари</Link>
            <Link href="/wishlist">Обране</Link>
            <Link href="/cart">Кошик</Link>
            <Link href="/order">Замовлення</Link>
            <a onClick={handleExit}>Вийти</a>
          </div>
        )}
      </div>
    </header>
  );
};

export default Header;
```

Файл ProductList.jsx

Відповідає за відображення списку товарів.

```
import "@styles/ProductList.css";
import ProductCard from "./ProductCard";

const ProductList = ({ productCards }) => {
  return (
    <div className="list">
      {productCards.map((product) => (
        <ProductCard key={product._id} product={product} />
      ))}
    </div>
  );
};

export default ProductList;
```

Файл ProductCard.jsx

Відповідає за відображення товару, додавання товару в обране, видалення товару.

```
import KeyboardArrowLeftIcon from "@mui/icons-material/KeyboardArrowLeft";
import KeyboardArrowRightIcon from "@mui/icons-material/KeyboardArrowRight";
import DeleteIcon from "@mui/icons-material/Delete";
import FavoriteIcon from "@mui/icons-material/Favorite";
import FavoriteBorderIcon from "@mui/icons-material/FavoriteBorder";
import { useSession } from "next-auth/react";
import { useRouter } from "next/navigation";
import { useState } from "react";
import "@styles/ProductCard.css";
```

```
// превью товару на домашній сторінці
const ProductCard = ({ product }) => {
  const navRouter = useRouter();

  // отримуємо інформацію про зареєстрованого користувача
  const { data: userSession, update } = useSession();
  const userId = userSession?.user?._id;
  const wishlist = userSession?.user?.wishlist;

  // видалити товар
  const handleDelProduct = async () => {
    const confirmDelete = confirm("Ви точно бажаєте видалити товар?");

    // товар видаляється лише після підтвердження
    if (confirmDelete) {
      try {
        await fetch(`api/product/${product._id}`, {
          method: "DELETE",
        });
      } catch (error) {
        console.error(error);
      }
      window.location.reload();
    }
  };

  // Слайдер для зображень
  const [index, setIndex] = useState(0);

  // наступне зображення
  const nextPicture = () => {
    setIndex((currIndex) => (currIndex + 1) % product.productPics.length);
  };

  // попереднє зображення
  const prevPicture = () => {
    setIndex(
      (currIndex) =>
        (currIndex - 1 + product.productPics.length) %
        product.productPics.length
    );
  };

  // чи додано товар до обраного
  const isWished = wishlist?.find(
    (wishedProduct) => wishedProduct?._id === product._id
  );

  // якщо користувач не автентифікований, опція додати до обраного не активна
  const checkAddToWish = async () => {
    if (!userSession) {
      navRouter.push("/login");
      return;
    }

    // зберегти оновлений вішліст
    const updatedUser = await fetch(
      `api/user/${userId}/wishlist/${product._id}`,
      {
        method: "PATCH",
      }
    ).json();
```

```
    // оновити дані про вішліст в сесії поточного користувача
    update({ user: { wishlist: updatedUser.wishlist } });
  };

  return (
    <section
      className="card"
      onClick={() => {
        navRouter.push(`/product-info?id=${product._id}`);
      }}
    >
      {/* Слайдер з зображеннями */}
      <div className="container">
        <div
          className="slides"
        >
          {product.productPics?.map((picture, i) => (
            <div className="picture" key={i}>
              <img src={picture} alt="зображення товару" />
              <div
                className="prev"
                onClick={(event) => {
                  event.stopPropagation();
                  prevPicture(event);
                }}
              >
                <KeyboardArrowLeftIcon />
              </div>
              <div
                className="next"
                onClick={(event) => {
                  event.stopPropagation();
                  nextPicture(event);
                }}
              >
                <KeyboardArrowRightIcon />
              </div>
            </div>
          ))}
        </div>
      </div>

      {/* Деталі товару */}
      <div className="product">
        <div>
          <h3>{product.title}</h3>
          <div className="user">
            <img src={product.createdBy.profilePic} alt="автор" />
            <span>{product.createdBy.login}</span>
            <span>в {product.category}</span>
          </div>
        </div>
        <div className="paid">{product.price} ₴</div>
      </div>

      {/* якщо поточний користувач створив даний товар, доступна опція
видалення */}
      {userId === product?.createdBy._id ? (
        <div
          className="delete"
          onClick={(event) => {
            event.stopPropagation();
            handleDelProduct();
```

```
          }}
        >
          <DeleteIcon />
        </div>
      ) : (
        <div
          className="icon"
          onClick={(e) => {
            e.stopPropagation();
            checkAddToWish();
          }}
        >
          {/* якщо поточний користувач не є автором товару, доступна опція
додати до обраного/видалити з обраного */}
          {isWished ? (
            <FavoriteIcon />
          ) : (
            <FavoriteBorderIcon />
          )}
        </div>
      )}
    </section>
  );
};

export default ProductCard;
```

Файл Form.jsx

Відповідає за додавання/оновлення/видалення товару.

```
import { categories } from "@data";
import CropOriginalIcon from "@mui/icons-material/CropOriginal";
import DeleteIcon from "@mui/icons-material/Delete";
import AttachFileIcon from "@mui/icons-material/AttachFile";
import "@styles/Form.scss";

const Form = ({ product, setProduct, handleSubmit }) => {
  // завантажене зображення додається в список та відображається на сторінці
  const handleAddPic = (e) => {
    const newPic = e.target.files;
    setProduct((prevProduct) => {
      return {
        ...prevProduct, // всі попередньо додані зображення
        pictures: [...prevProduct.pictures, ...newPic],
      };
    });
  };

  // видалене зображення видаляється зі списку по індексу та не
відображається на сторінці
  const handleDelPic = (i) => {
    setProduct((prevProduct) => {
      return {
        ...prevProduct,
        pictures: prevProduct.pictures.filter(
          (_, currentIndex) => currentIndex !== i
        ),
      };
    });
  };

  // завантаження файлу товару
  const handleAddFiles = (e) => {
    const newFiles = e.target.files;
```

```
  setProduct((prevProduct) => {
    return {
      ...prevProduct,
      files: [...prevProduct.files, ...newFiles],
    };
  });
};

const handleChange = (e) => {
  const { name, value } = e.target;
  setProduct((prevProduct) => {
    return {
      ...prevProduct,
      [name]: value,
    };
  });
};

return (
  <div className="container">
    <h1> Додати товар</h1>
    <form onSubmit={handleSubmit}>
      <h3>Оберіть категорію:</h3>
      <div className="categories">
        {categories?.map((category, i) => (
          <p
            key={i}
            className={`${product.category === category ? "current" : ""}`}
            onClick={() => {
              setProduct({ ...product, category: category });
            }}
          >
            {category}
          </p>
        ))}
      </div>

      <h3>Додайте зображення</h3>
      {/* якщо список зображень порожній */}
      {product.pictures.length < 1 && (
        <div className="photos">
          <label htmlFor="pictureInput" className="alone">
            <div className="icon">
              <CropOriginalIcon color="black" />
            </div>
            <p>Завантажити</p>
          </label>
          <input
            id="pictureInput"
            type="file"
            accept="image/*"
            onChange={handleAddPic}
            multiple
          />
        </div>
      )}

      {/* якщо вже є додані зображення, вони відображаються */}
      {product.pictures.length > 0 && (
        <div className="pictures">
          {product?.pictures?.map((picture, i) => (
            <div key={i} className="picture">
              {picture instanceof Object ? (
```

```
                  // зображення, завантажені з пристрою - об'єкти (при
створенні товару)
                  <img
                    src={URL.createObjectURL(picture)}
                    alt="зображення товару"
                  />
                ) : (
                  // зображення, завантажені з бази даних - string (при
редагуванні товару)
                  <img src={picture} alt="зображення товару" />
                )}
                <button type="button" onClick={() => handleDelPic(i)}>
                  <DeleteIcon color="black" />
                </button>
              </div>
            ))}

            <label htmlFor="picture">
              <div className="icon">
                <CropOriginalIcon color="black" />
              </div>
              <p>Завантажити</p>
            </label>
            <input
              id="picture"
              type="file"
              accept="image/*"
              onChange={handleAddPic}
              multiple
            />
          </div>
        )}

        <h3>Інформація про товар</h3>
        <div className="description">
          <p>Назва</p>
          <input
            type="text"
            placeholder="Назва товару"
            onChange={handleChange}
            name="title"
            value={product.title}
            required
          />
          <p>Про товар</p>
          <textarea
            cols={4}
            rows={20}
            type="text"
            placeholder="Опис товару"
            onChange={handleChange}
            name="description"
            value={product.description}
            required
          />
          <p>Вкажіть ціну</p>
          <span>₴</span>
          <input
            type="number"
            placeholder="Ціна"
            onChange={handleChange}
            name="price"
            value={product.price}
            required
```

```
            className="price"
          />
        </div>

        <h3 className="addFile">
          Додайте файл. Обов'язково додайте документ, що підтверджує
авторське право.
        </h3>
        <div className="files">
          <label htmlFor="file" className="file-label">
            <div className="icon">
              <AttachFileIcon color="black" />
            </div>
            <p>Завантажити</p>
          </label>
          <input
            id="file"
            type="file"
            style={{ display: "none" }}
            accept="application/*"
            onChange={handleAddFiles}
          />
        </div>

        <button className="submit_btn" type="submit">
          Опублікувати
        </button>
      </form>
    </div>
  );
};

export default Form;
```

Допоміжні файли:

Файл database.js

Відповідає за підключення до бази даних.

```
import mongoose from "mongoose";

// Відслідковує підключення до бази даних.
let isConnToDb = false;

export const mongoDB = async () => {
  mongoose.set("strictQuery", true);

  if (isConnToDb) {
    return;
  }

  try {
    await mongoose.connect(process.env.MONGODB_CONNECTION_STRING, {
      dbName: "digit-market",
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });

    isConnToDb = true;

  } catch (error) {
    console.error(error);
```

```
  }
};
```

Файл User.js

Містить схему даних для користувача.

```
import { Schema, model, models } from "mongoose";

const SchemaForUsers = new Schema({
  // Містить ім'я користувача в системі. Обов'язкове, унікальне
  login: {
    type: String,
    required: [true, "Поле логін обов'язкове для заповнення"],
    unique: [true, "Логін вже існує"],
  },
  // Містить електронну пошту користувача. Обов'язкове, унікальне
  email: {
    type: String,
    required: [true, "Поле електронна пошта обов'язкове для заповнення"],
    unique: [true, "Електронна пошта вже існує"],
  },
  // Містить пароль користувача. Обов'язкове
  password: {
    type: String,
    required: [true, "Поле пароль обов'язкове для заповнення"],
  },
  // Містить роль користувача
  role: {
    type: String,
    default: "user",
  },
  // Містить посилання на зображення профілю користувача
  profilePic: {
    type: String,
    //required: [true, "Profile image is required"],
  },
  // Містить список товарів, створених користувачем. Список об'єктів
  products: {
    type: Array,
    default: [],
  },
  // Містить список замовлень користувача. Список об'єктів
  orders: {
    type: Array,
    default: [],
  },
  // Містить список товарів в кошику користувача. Список об'єктів
  cart: {
    type: Array,
    default: [],
  },
  // Містить список обраних товарів користувача. Список об'єктів
  wishlist: {
    type: Array,
    default: [],
  },
});

// Створити нову модель Користувач, якщо дана модель ще не створена
const User = models.User || model("User", SchemaForUsers);

export default User;
```

Файл Product.js

Містить схему даних для товару.

```
import { Schema, model, models } from "mongoose";

const SchemaForProducts = new Schema({
  // Містить id користувача, що створив товар
  createdBy: {
    type: Schema.Types.ObjectId,
    ref: "User",
  },
  // Містить категорію товару. Обов'язкове
  category: {
    type: String,
    required: [true, "Категорія обов'язкова"],
  },
  // Містить назву товару. Обов'язкове
  title: {
    type: String,
    required: [true, "Назва обов'язкова"],
  },
  // Містить текстовий опис товару. Обов'язкове
  description: {
    type: String,
    required: [true, "Опис товару обов'язковий"],
  },
  // Містить статус товару
  status: {
    type: String,
    default: "pending",
  },
  // Містить ціну товару. Обов'язкове
  price: {
    type: Number,
    required: [true, "Ціна обов'язкова"],
  },
  // Містить список посилань на зображення товару
  productPics: [{ type: String }],
  // Містить список посилань на файли товару
  productFile: [{ type: String }],
});

// Створити нову модель Товар, якщо дана модель ще не створена
const Product = models.Product || model("Product", SchemaForProducts);
export default Product;
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК «ОНЛАЙН-РИНОК ЕЛЕКТРОННОЇ КОМЕРЦІЇ ДЛЯ ЦИФРОВИХ ПРОДУКТІВ»

Програма та методика тестування

КПІ.ІП-з0102.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юлія ПОЛУПАН

Нормоконтроль:
_____ Олександр СТЕЛЬМАХ

Виконавець:
_____ Оксана ІВАНИЦЬКА

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є вебзастосунок «Онлайн-ринок електронної комерції для цифрових продуктів», створений на мові програмування JavaScript за допомогою Node.js, фреймворку Next.js та з використанням MongoDB Atlas.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог, а саме: процеси авторизації, створення товарів, купівлі товарів, додавання товарів у список обраного, здійснення пошуку товарів, сортування товарів за категоріями;

– перевірка збереження даних;

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Edge);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

– тестування «чорної скриньки» – об’єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;

– тестування «сірої скриньки» – об’єктом тестування тут є деякі особливості внутрішньої поведінки програми. Перевіряється коректність вихідних даних при заданих вхідних, застосовується для тестування окремих алгоритмів (функцій).

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Тестування API проводиться з використанням інструментарію Postman.

Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- ручне тестування на відповідність функціональним вимогам;
- тестування на пристроях з різною роздільною здатністю екрану;
- тестування в різних браузерах;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування інтерфейсу користувача;
- тестування зручності використання;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК «ОНЛАЙН-РИНОК ЕЛЕКТРОННОЇ КОМЕРЦІЇ ДЛЯ ЦИФРОВИХ ПРОДУКТІВ»

Керівництво користувача

КПІ.ІП-з0102.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юлія ПОЛУПАН

Нормоконтроль:
_____ Олександр СТЕЛЬМАХ

Виконавець:
_____ Оксана ІВАНИЦЬКА

Київ – 2024

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Вебдодаток «Онлайн-ринок електронної комерції для цифрових продуктів» використовується для онлайн-торгівлі та призначений для виробників та споживачів цифрових товарів.

Програма забезпечує можливість виконання наступних функцій:

- авторизація,
- створення товарів,
- купівля товарів,
- додавання товарів у список обраного,
- здійснення пошуку товарів,
- сортування товарів за категоріями.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Програмне забезпечення працює на IBM-сумісних персональних комп‘ютерах під управлінням операційних систем сімейства WIN32 (версії від Windows 7 і вище) із встановленим браузером Google Chrome або Opera.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

2.2 Завантаження застосунку

Для початку роботи з програмним забезпеченням, оператор повинен відкрити браузер та ввести у адресний рядок ip-адресу або доменне ім’я серверу, на якому розгорнуто програмне забезпечення. Запуск браузера проводиться способами, детальна інформація про які наведена у керівництві користувача операційної системи. Для отримання IP-адреси або доменного імені серверу, на якому розгорнуто програмне забезпечення, оператору слід звернутись до суб'єкта, що проводив розгортання програмного забезпечення.

2.3 Перевірка коректної роботи

У разі успішного розгортання додатку, після переходу за адресою вебдодатку, повинна відобразитись домашня сторінка додатку. Натискаючи на посилання, користувач має змогу перейти на сторінки сайту.

3 ВИКОНАННЯ ПРОГРАМИ

Перш за все користувачу, що вперше на сайті необхідно зареєструватись, для цього треба перейти на сторінку реєстрації (рисунок 3.1). У кожне поле необхідно ввести дані, відповідно до інструкції над кожною формою введення. Якщо паролі не співпадають, кнопка «Зареєструватися» залишається неактивною та на сторінці виводиться відповідне повідомлення.

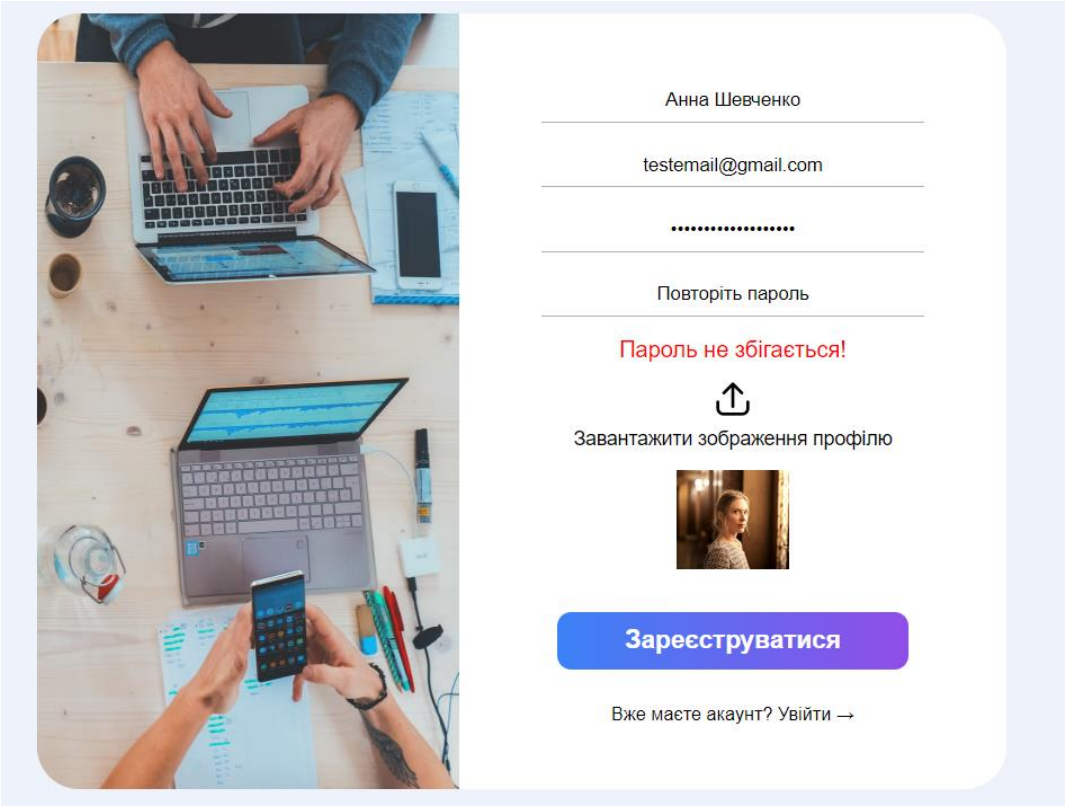


Рисунок 3.1 – Заповнення форми на сторінці реєстрації

Після чого натискається кнопка «Зареєструватися» і користувач переходить на сторінку входу (рисунок 3.2). Для входу необхідно ввести електронну пошту і пароль та натиснути кнопку «Увійти». Якщо виводиться повідомлення про помилку, можливо користувач з такими даними вже існує. Для вирішення проблеми користувачеві треба змінити вказані поля та спробувати ще раз.

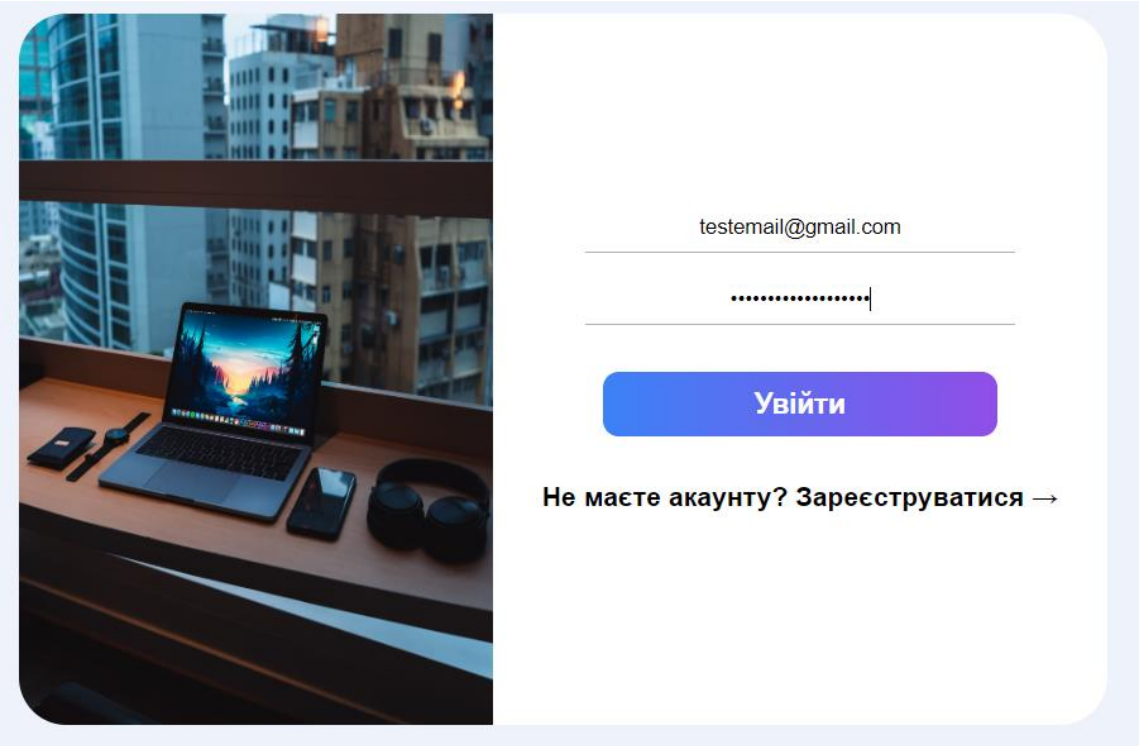


Рисунок 3.2 – Виконання входу на сторінці входу

Після цього користувач перенаправляється на домашню сторінку (рисунок 3.3). Там відображаються товари користувачів. При натисканні на категорію товари фільтруються за обраною категорією. На панелі у верхній частині сторінки відображаються логотип, при натисканні на який користувач переходить на домашню сторінку, пошук, кошик з лічильником товарів в кошику, зображення профілю користувача. При натисканні кнопки меню відкривається меню навігації.

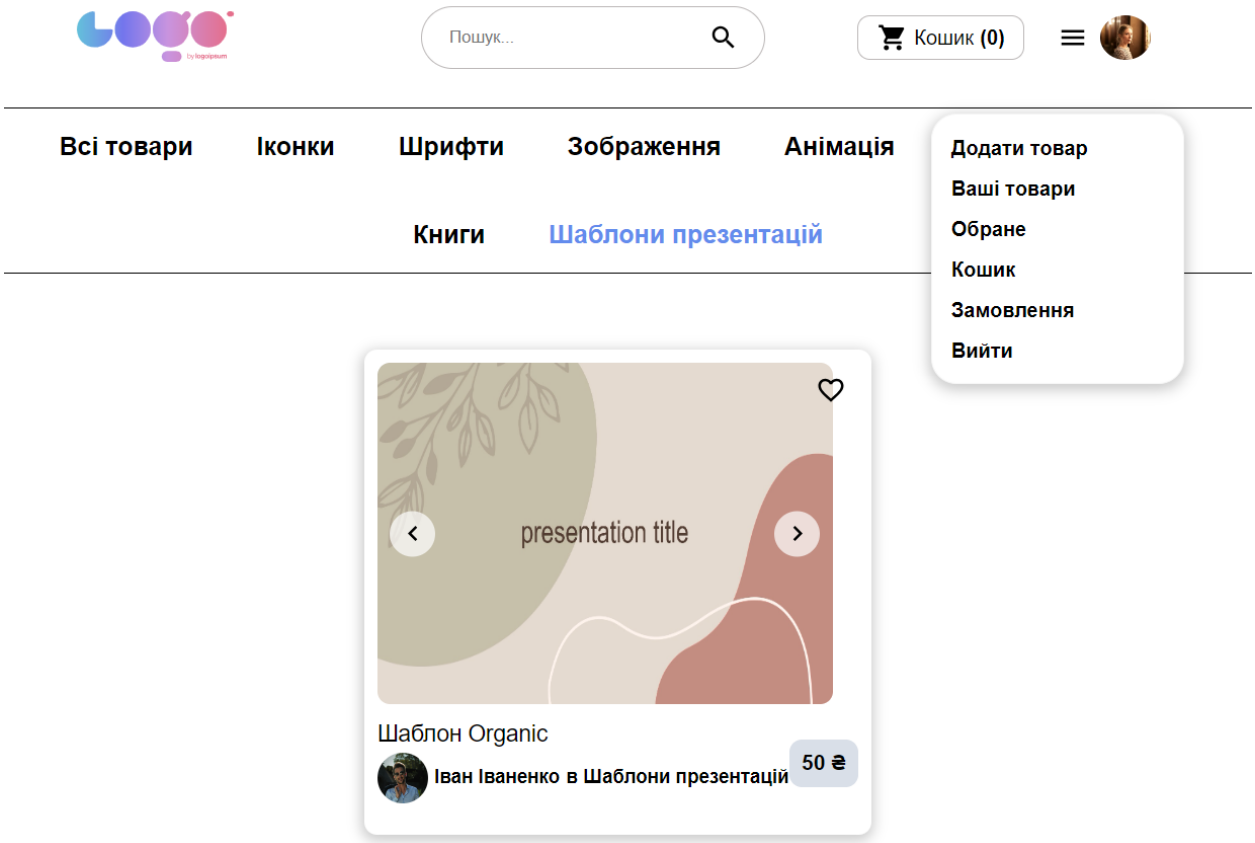


Рисунок 3.3 – Домашня сторінка

Натиснувши «Додати товар», користувач переходить на сторінку додавання нового товару (рисунок 3.4). Тут необхідно обрати категорію, ввести назву, опис, ціну. Додати зображення та файл. Файл додається у вигляді архіву з документом, що підтверджує авторське право на цифровий продукт. Щоб створити товар, необхідно натиснути «Додати товар».

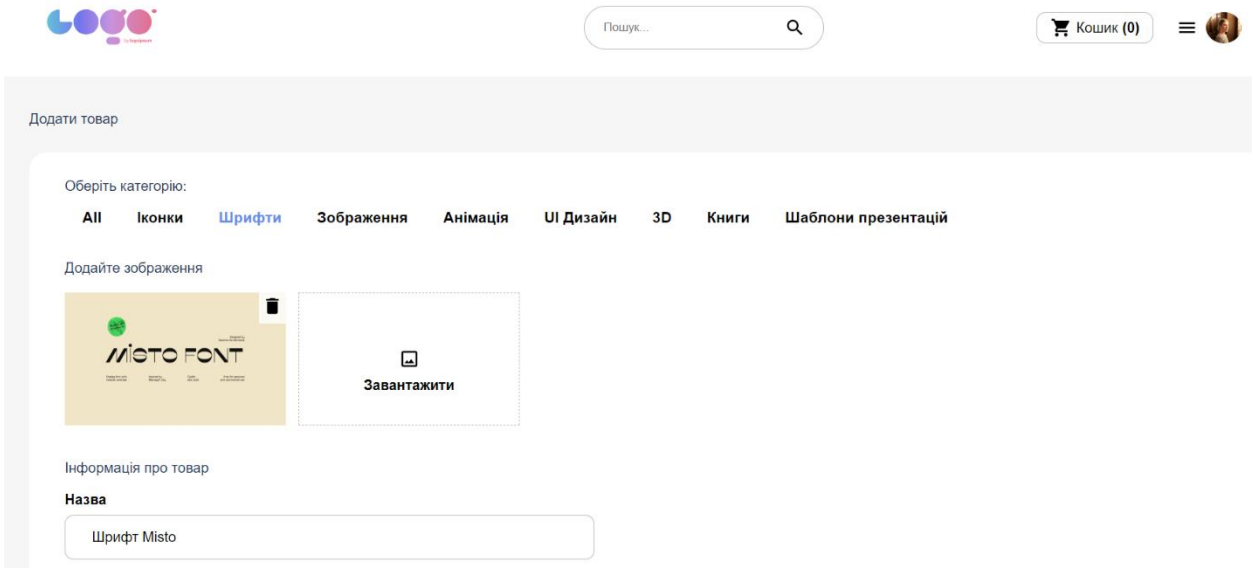


Рисунок 3.4 – Сторінка створення товару

Після цього користувач автоматично переходить на сторінку створених товарів, де може переглянути коротку інформацію про товар (рисунок 3.5).

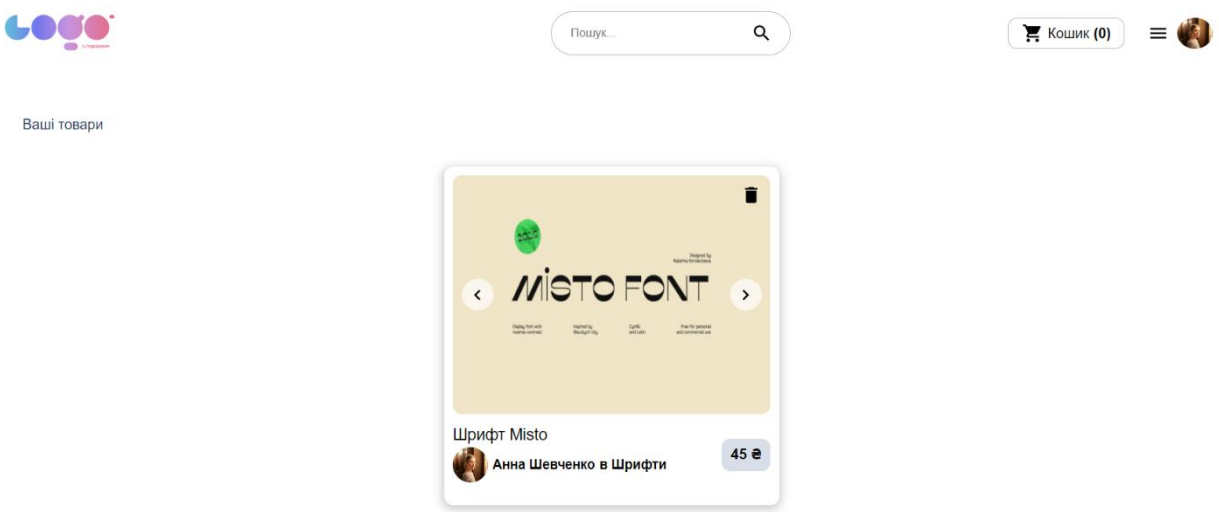


Рисунок 3.5 – Сторінка товарів користувача

Натиснувши на картку товару, можна перейти на сторінку товару (рисунок 3.6).

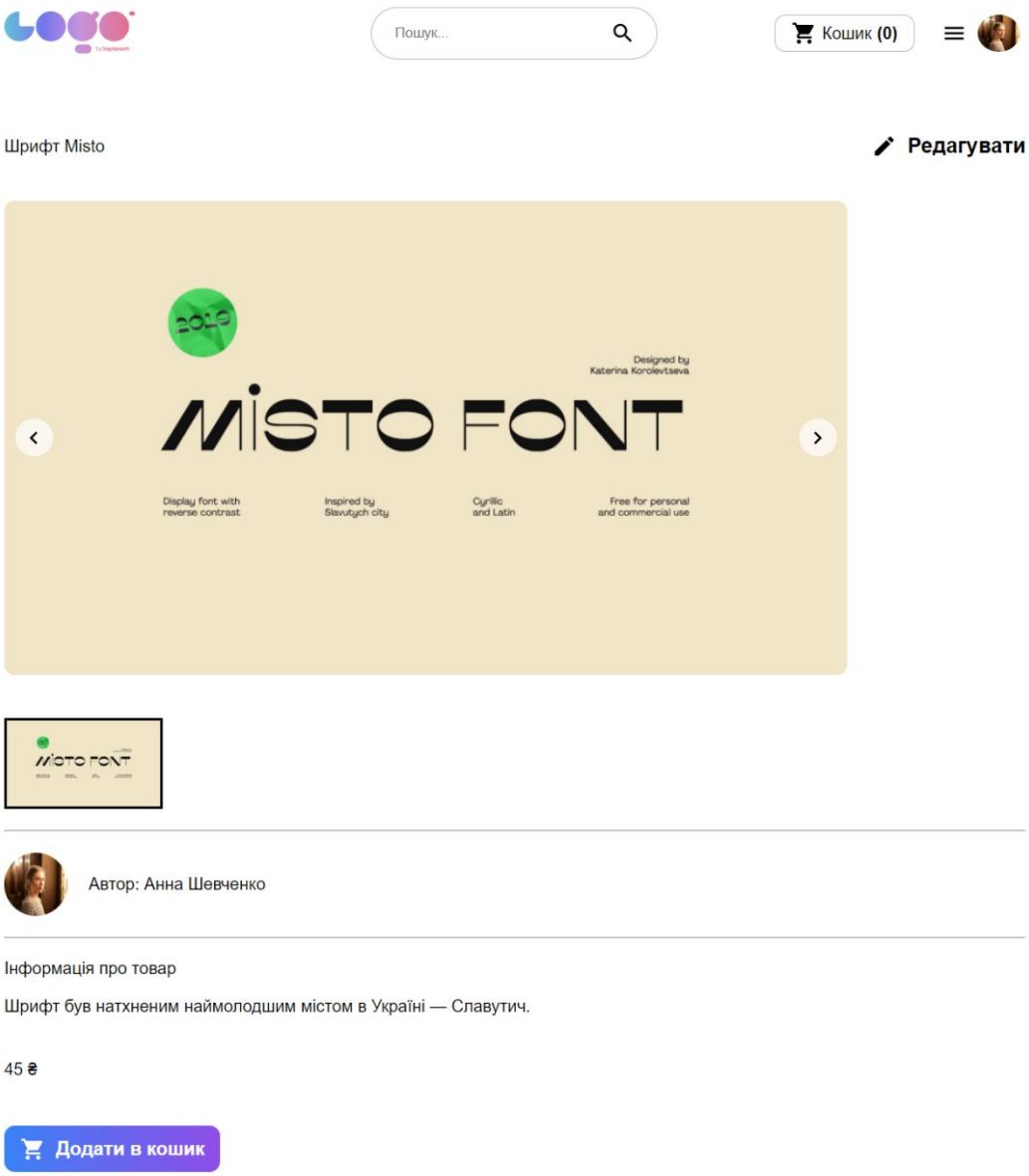


Рисунок 3.6 – Сторінка товару

Для користувача, який є автором даного продукту, доступна опція «Редагувати товар». Обравши її, можна перейти на сторінку оновлення товару (рисунок 3.7).

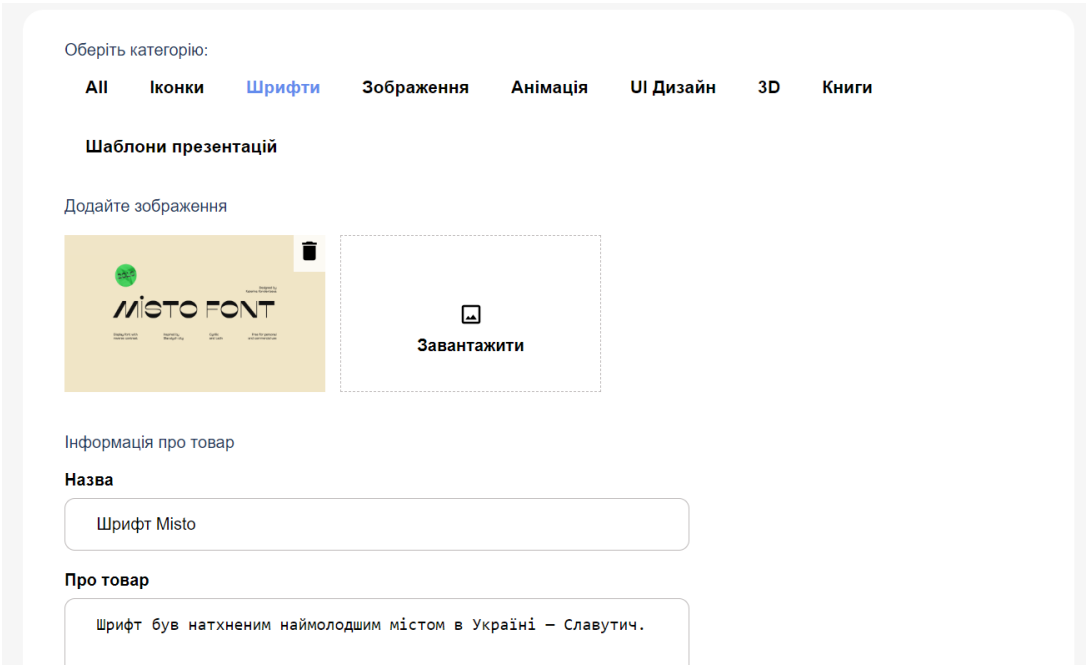


Рисунок 3.7 – Сторінка оновлення товару

Тут можливо змінити назву, категорію, опис, ціну та зображення. Натиснувши кнопку «Зберегти», користувач переходить на сторінку створених товарів, де відображається товар зі зміненою інформацією.

Оскільки щойно створений товар ще не підтверджений адміністратором, він не відображається на домашній сторінці.

Для користувача з правами адміністратора на домашній сторінці відображається кнопка «Панель адміністратора» (рисунок 3.8).

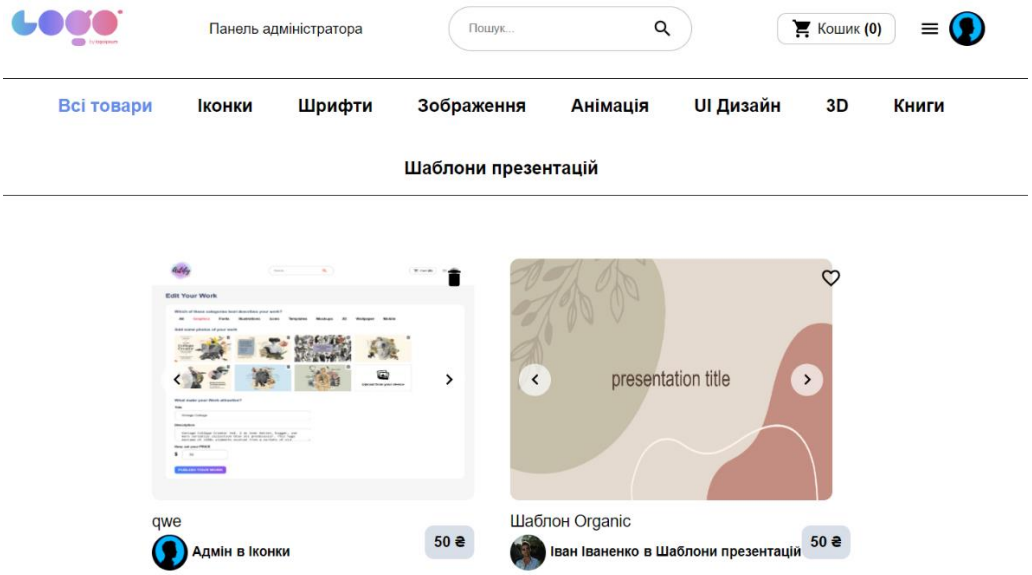


Рисунок 3.8 – Домашня сторінка

Натиснувши її, користувач переходить на сторінку адміністратора, де відображаються всі товари що чекають перевірки (рисунок 3.9).

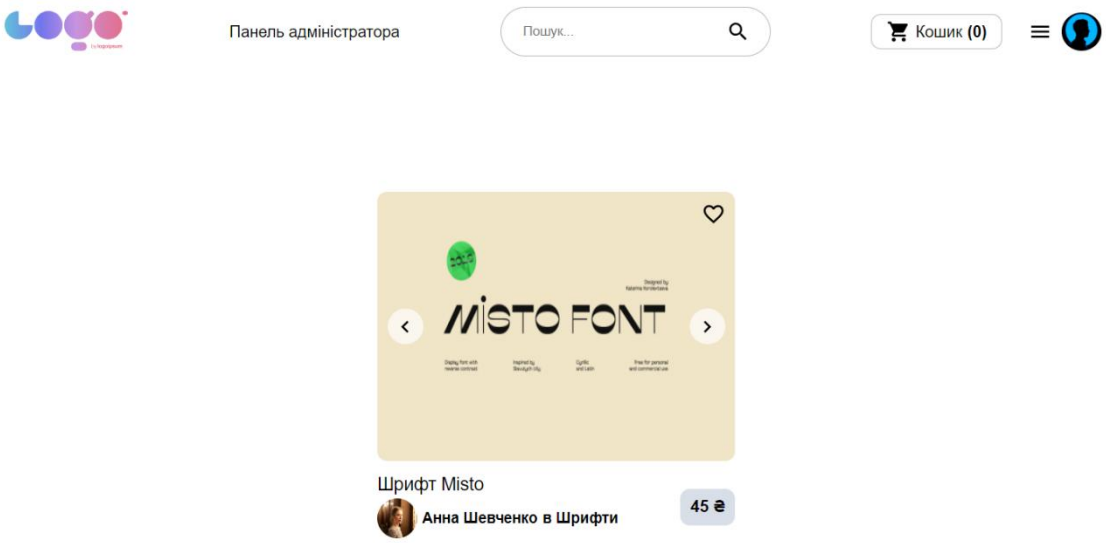


Рисунок 3.9 – Сторінка адміністратора

Натиснувши на товар, користувач переходить на сторінку товару та змінює статус на «Підтверджено» (рисунок 3.10).

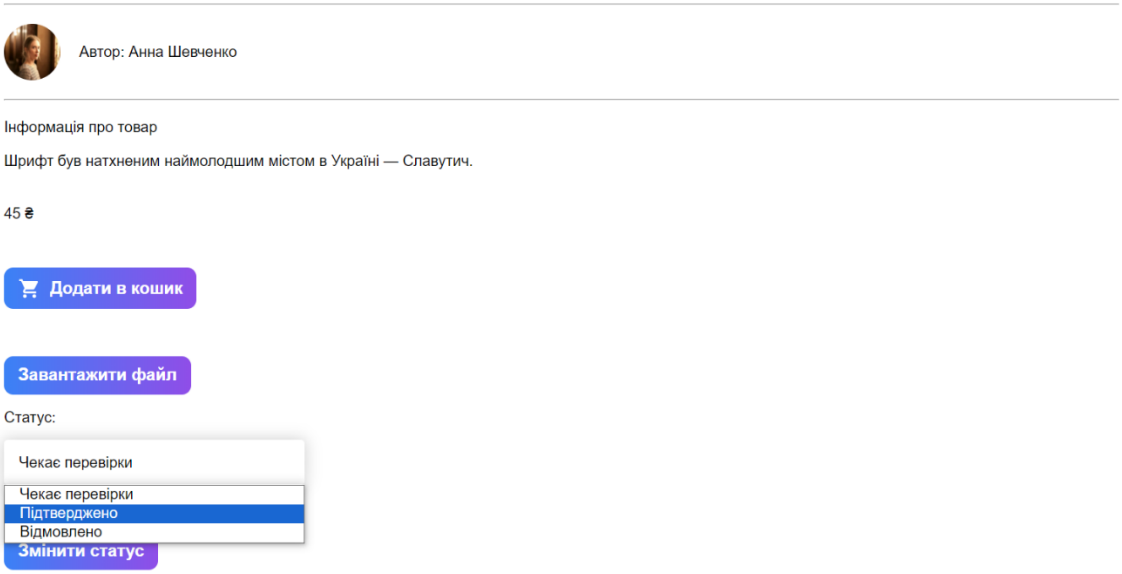


Рисунок 3.10 – Зміна статусу товару

Після цього товар відображається на домашній сторінці (рисунок 3.11). Автор товару може його видалити, натиснувши відповідну кнопку біля зображення на картці товару.

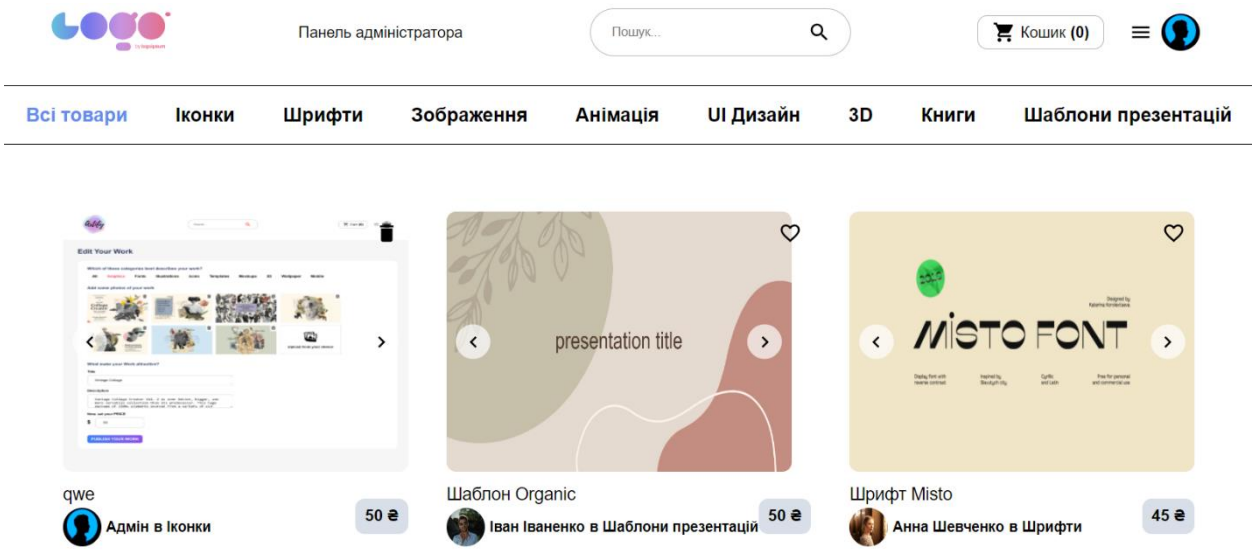


Рисунок 3.11 – Домашня сторінка зі створеним товаром

На домашній сторінці можливо виконати пошук товару за назвою. Для цього необхідно ввести у відповідне поле назву та натиснути кнопку «Пошук». Знайдений товар відображається на сторінці результатів пошуку (рисунок 3.12).

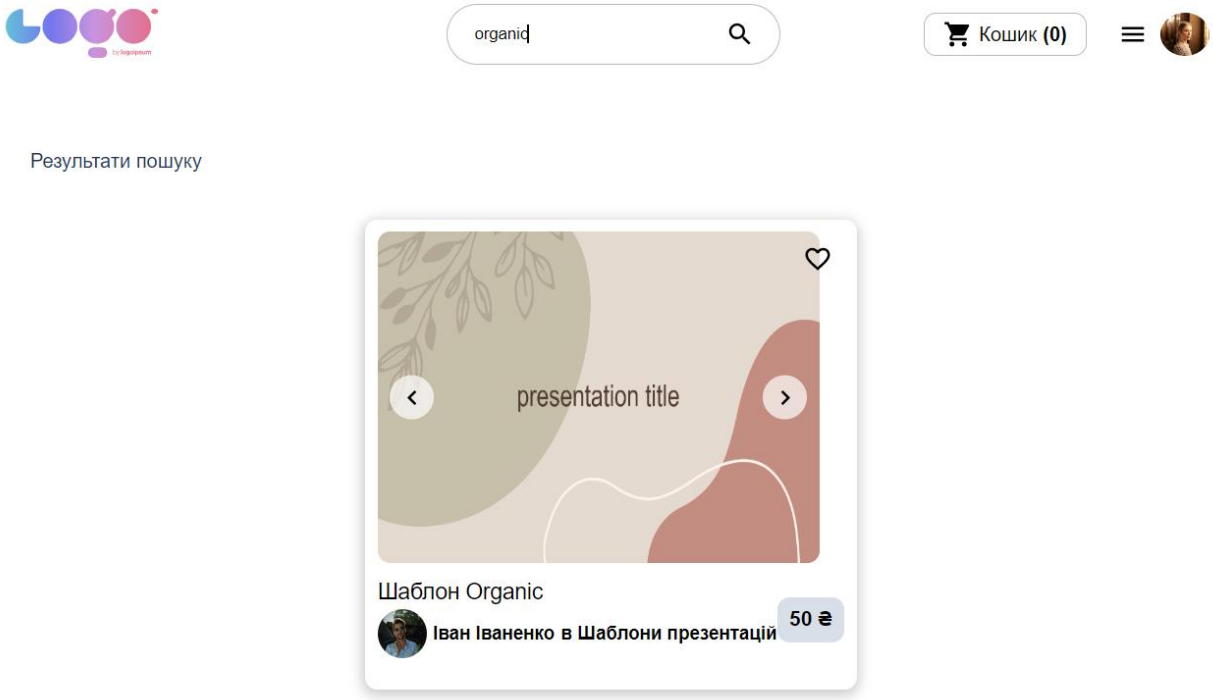


Рисунок 3.12 – Результат пошуку

Щоб додати товар до кошика, необхідно перейти на сторінку знайденого товару та натиснути кнопку «Додати до кошика». При

повторному натисканні кнопки виводиться повідомлення про те, що даний товар вже знаходиться в кошику (рисунок 3.13).

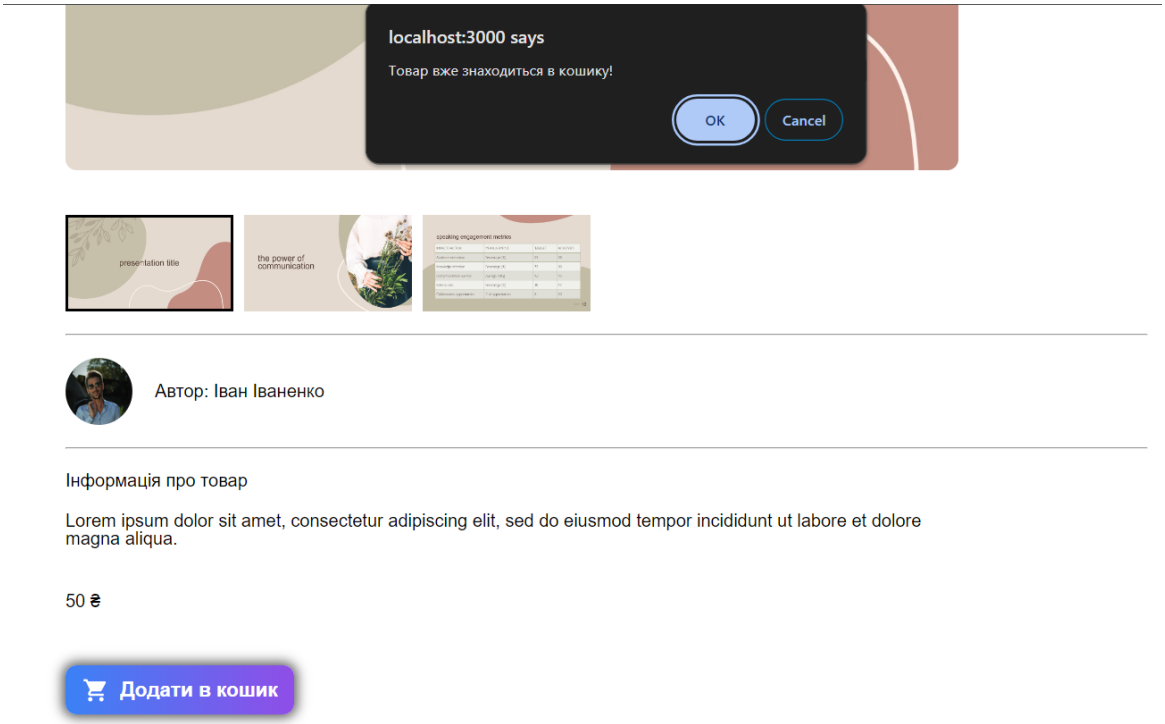


Рисунок 3.13 – Додавання до кошика

При натисканні кнопки кошика відбувається перенаправлення на сторінку кошика користувача (рисунок 3.14). Тут можна бачити інформацію про додані товари, суму до сплати. При натисканні кнопки видалення, товар видаляється з кошика.

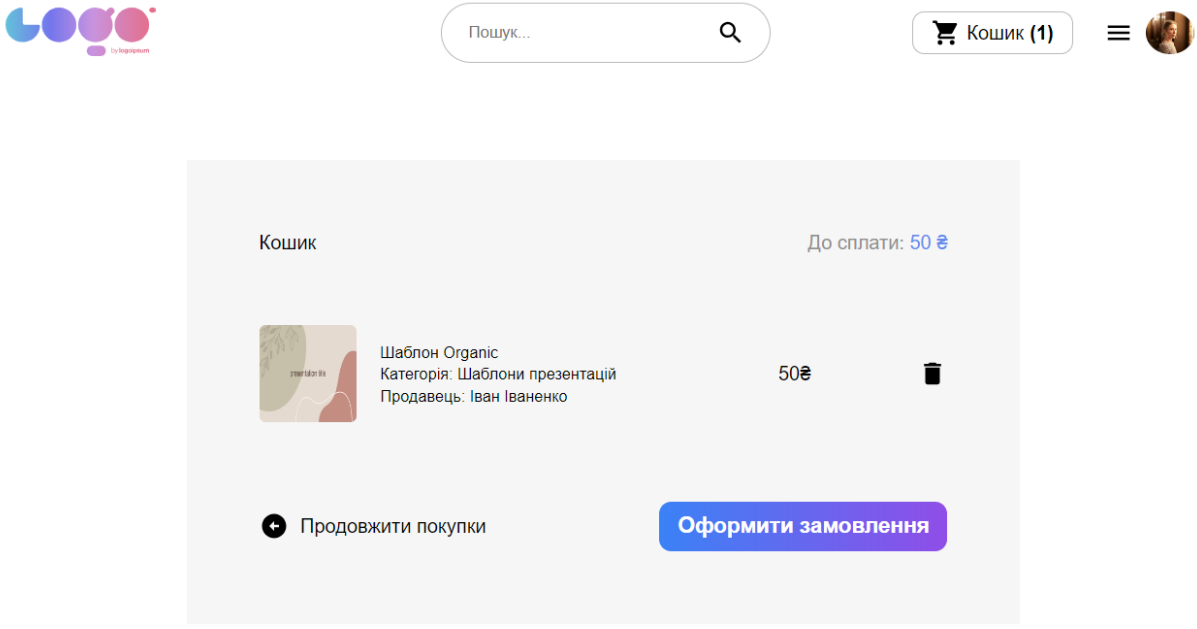


Рисунок 3.14 – Кошик

Коли користувач натискає кнопку «Оформити замовлення», він переходить на сторінку платіжної системи, де може ввести платіжні дані та натиснути «Оплатити» (рисунок 3.15).

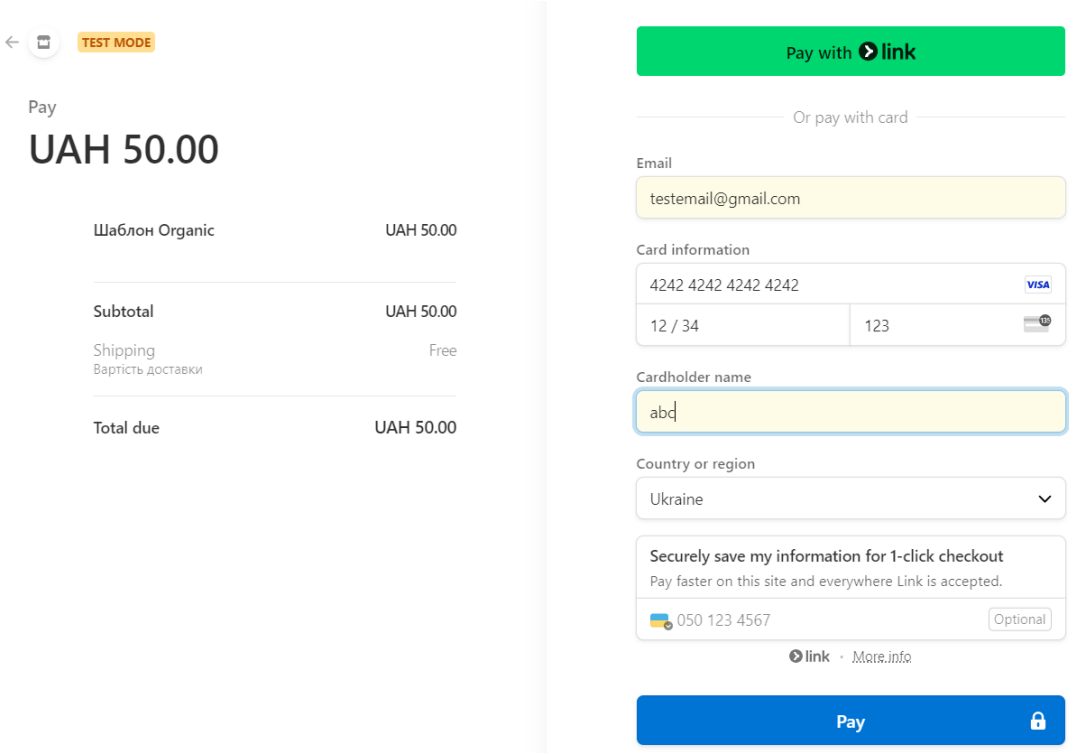


Рисунок 3.15 – Оплата товару

Після цього користувач переходить на сторінку з замовленнями, де може завантажити придбаний товар, натиснувши відповідне посилання (рисунок 3.16).

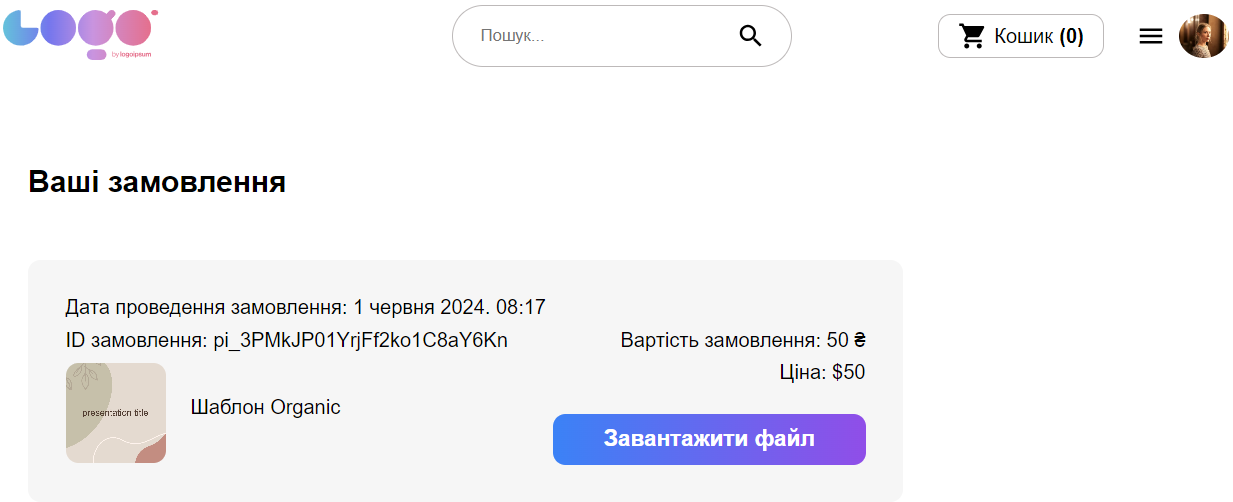


Рисунок 3.16 – Сторінка замовлення

Перебуваючи на домашній сторінці, користувач може натиснути на ім’я або зображення профілю користувача, щоб перейти на сторінку з товарами даного користувача (рисунок 3.17). Натиснувши на кнопку «Додати в обране», можна додати бажаний товар в свій список обраних товарів. Якщо натиснути на кнопку повторно, товар видалиться з обраного.

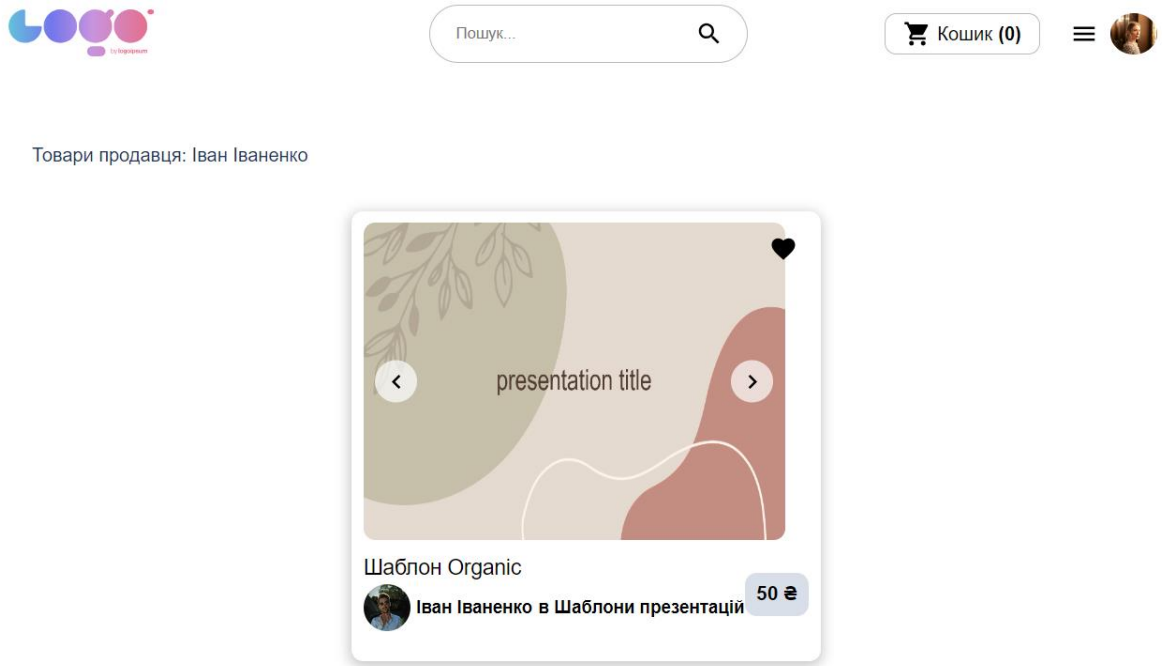


Рисунок 3.17 – Сторінка з товарами користувача

На сторінці товару доступна кнопка «Порушення авторського права». При її натисканні товар стає недоступним для додавання в кошик, видаляється з домашньої сторінки та відображається на сторінці адміністратора. Якщо виявлено порушення, адміністратор видаляє товар.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК «ОНЛАЙН-РИНОК ЕЛЕКТРОННОЇ КОМЕРЦІЇ ДЛЯ ЦИФРОВИХ ПРОДУКТІВ»

Графічний матеріал

КПІ.ІП-з0102.045440.06.99

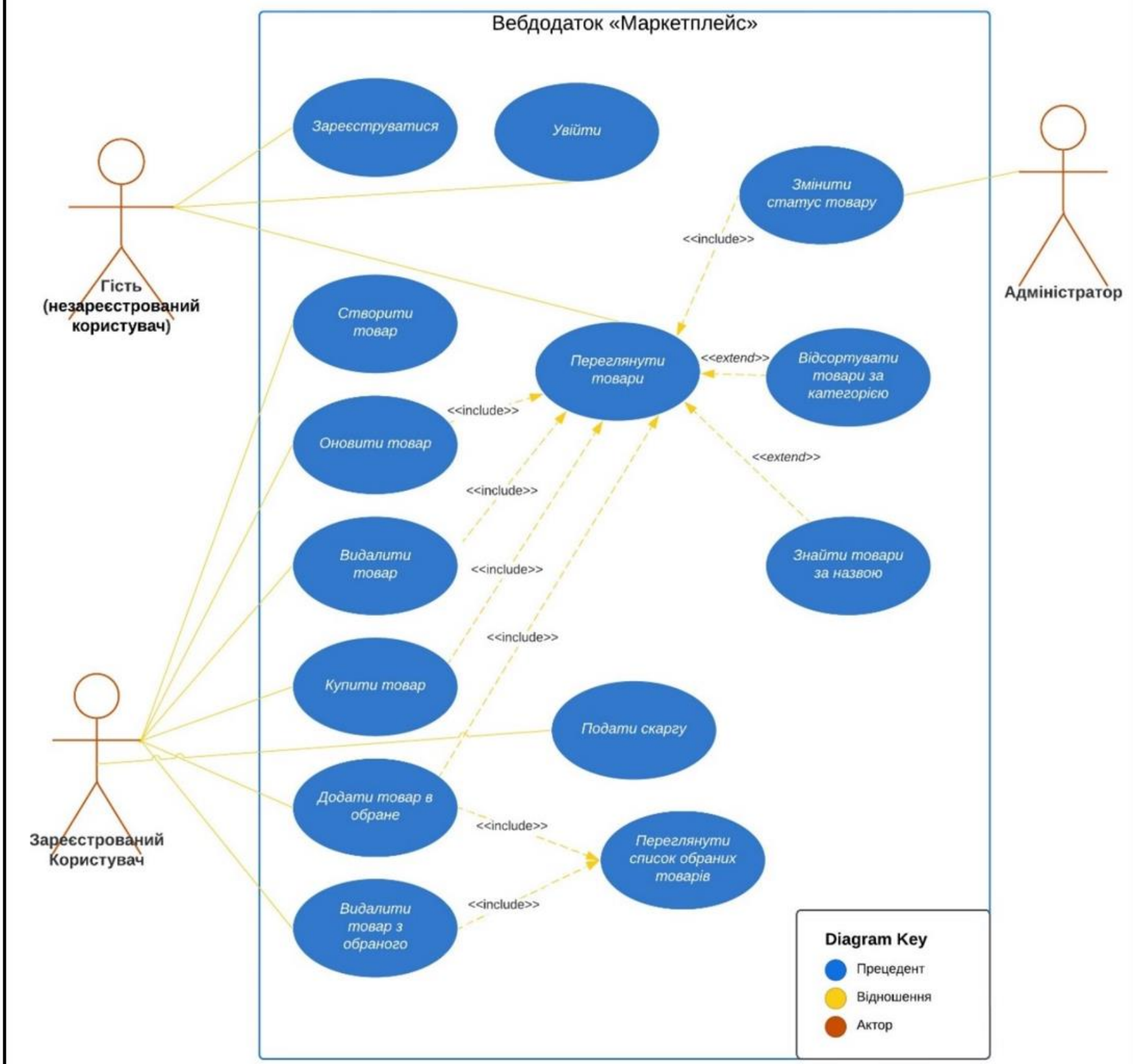
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Юлія ПОЛУПАН

Нормоконтроль:
_____Олександр СТЕЛЬМАХ

Виконавець:
_____ Оксана ІВАНИЦЬКА

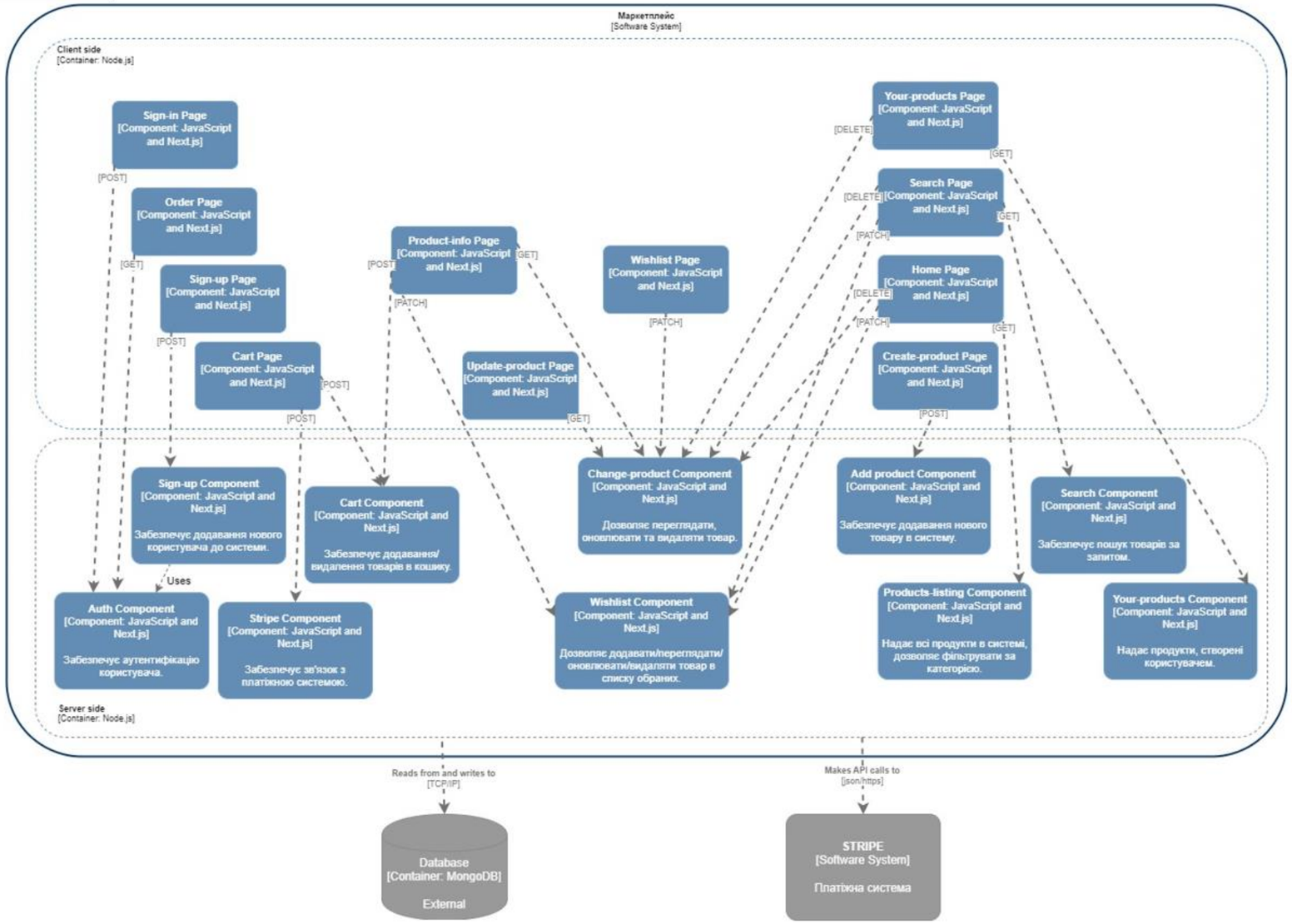
Київ – 2024

КПІ.ІП-з0102.045440.06.99.ССВ



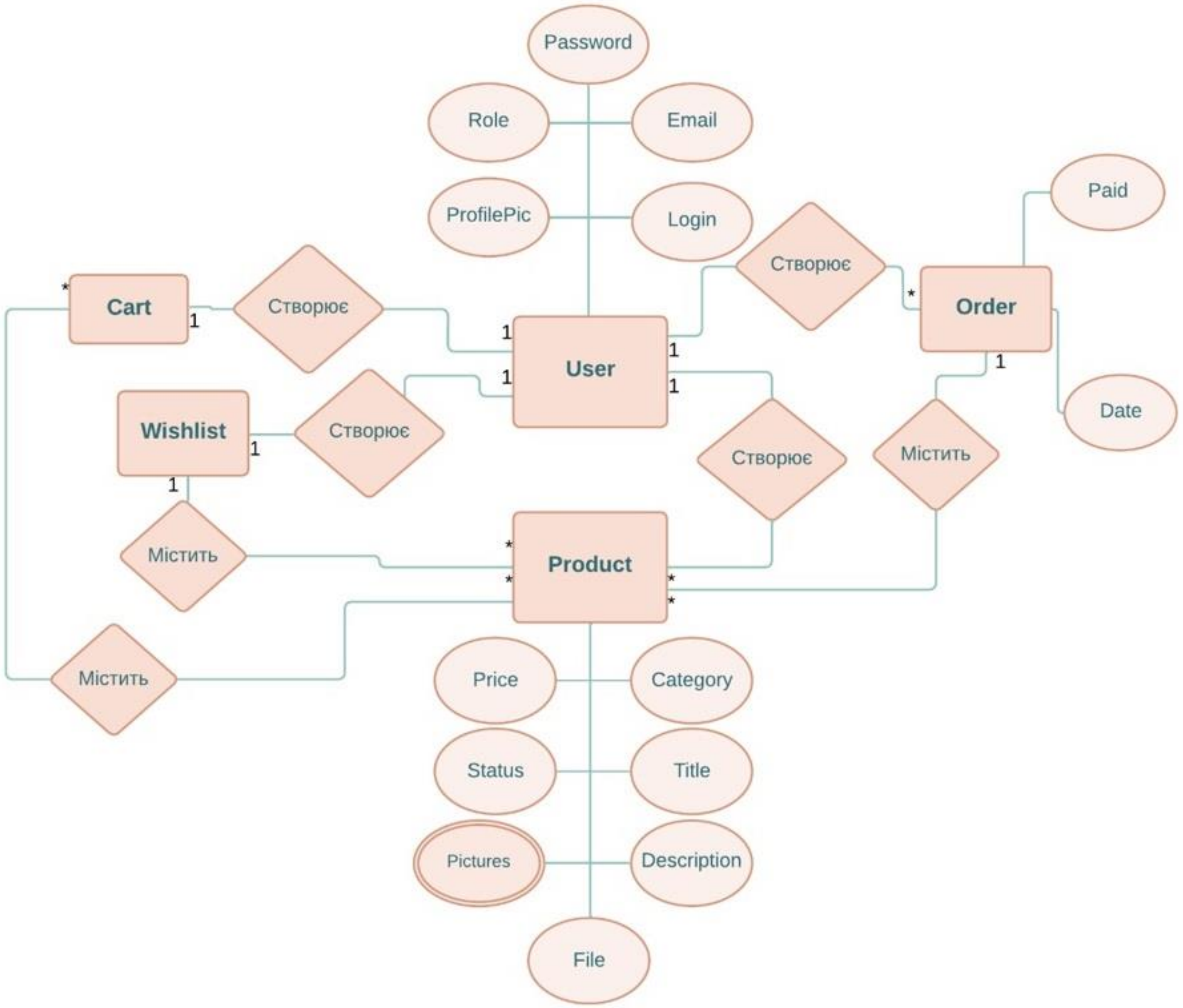
					КПІ.ІП-з0102.045440.06.99.ССВ
					Схема структурна варіантів використань
Зм.	Арк.	№ докум.	Підп.	Дата	Лит. / Маса / Масштаб
Розробив		Іваницька О.В.			Аркуш / Аркушів
Перевірив		Полупан Ю.В.			Вебзастосунок «Онлайн-ринок електронної комерції для цифрових продуктів»
Т. контр.					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-з01
Н. контр.		Стельмах О.П.			
Затвердив		Жаріков Е.В.			

КПІ.ІП-з0102.045440.06.99.ССМ



					КПІ.ІП-з0102.045440.06.99.ССМ			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення			
Розробив		Іваницька О.В.				Аркуш	Аркушів	
Перевірив		Полупан Ю. В.						
Т. контр.					Вебзастосунок «Онлайн-ринок електронної комерції для цифрових продуктів»	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-з01		
Н. контр.		Стельмах О. П.						
Затвердив		Жаріков Е.В.						

КПІ.ІП-з0102.045440.06.99.СБД



					КПІ.ІП-з0102.045440.06.99.СБД			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних			
Розробив		Іваницька О.В.						
Перевірив		Полупан Ю. В.						
Т. контр.						Аркуш		Аркушів
					Вебзастосунок «Онлайн-ринок електронної комерції для цифрових продуктів»	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-з01		
Н. контр.		Стельмах О. П.						
Затвердив		Жаріков Е.В.						