**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ**
**Кафедра програмної інженерії**

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ МАГАЗИНУ ПОБУТОВОЇ ТЕХНІКИ»**

Виконав: студент 4 курсу, групи 6.1219-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Я.В. Явдощен
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD, Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

**ЗАТВЕРДЖУЮ**
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 р.

# З А В Д А Н Н Я

## НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Явдощену Якову Валерійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет магазину побутової техніки

керівник роботи Столярова Анастасія Валеріївна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проєктування програмного забезпечення.
3. Реалізація та тестування програмного доповнення.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| 1 | Лісняк А.О., завідувач кафедри програмної інженерії | | |
| 2 | Лісняк А.О., завідувач кафедри програмної інженерії | | |
| 3 | Лісняк А.О., завідувач кафедри програмної інженерії | | |

7. Дата видачі завдання 07.02.2023 р.

## К А Л Е Н Д А Р Н И Й  П Л А Н

| № | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Розробка плану виконання кваліфікаційної | 08.02.2023 | |
| | роботи бакалавра. | | |
| | | | |
| 2. | Збір вихідних даних. | 24.02.2023 | |
| | | | |
| 3. | Обробка методичних та теоретичних | 10.03.2023 | |
| | джерел. | | |
| | | | |
| 4. | Розробка першого та другого розділу. | 14.04.2023 | |
| | | | |
| 5. | Розробка третього розділу та додатків. | 15.05.2023 | |
| | | | |
| 6. | Оформлення та нормоконтроль | 01.06.2023 | |
| | кваліфікаційної роботи бакалавра. | | |
| | | | |
| 7. | Захист кваліфікаційної роботи. | 22.06.2023 | |

Студент _____ Я.В. Явдощен
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Столярова
(підпис) (ініціали та прізвище)

**Нормоконтроль пройдено**

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

# РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет магазину побутової техніки»: 56 с., 26 рис., 3 табл., 15 джерел, 3 додатки.

API, FIGMA, FRAMEWORK, JAVASCRIPT, NEST.JS, REACT.JS.

Об’єкт дослідження – система, інструменти для взаємодії React та БД, засоби взаємодії програмного забезпечення з користувачем.

Мета роботи: розробити інтернет магазин для продажу побутових товарів.

Метод дослідження – моделювання, проєктування, програмний, аналітичний.

У нашому сучасному світі існує велика кількість інтернет магазинів, які пропонують побутові товари для продажу. Один з них - це мій інноваційний інтернет магазин на платформі React. Цей веб-сайт забезпечує користувачам зручну та безпечну платформу для придбання різноманітних побутових товарів. Він розроблений з використанням потужного фреймворку React, який забезпечує швидку та ефективну роботу сайту.

Один з головних переваг мого інтернет магазину – його інтуїтивно зрозумілий і легкий у використанні інтерфейс. Користувачі зможуть швидко знайти потрібні товари за допомогою розумної системи пошуку та фільтрації. Крім того, цей веб-сайт пропонує детальні описи товарів, відгуки користувачів та зручну систему оформлення замовлення.

Таким чином, за результатами роботи створено зручний та ефективний інтернет магазин з продажу побутових товарів.

# SUMMARY

Bachelor’s qualifying paper «Development of The Household Appliances Online Store»: 56 p., 26 figures, 3 tables, 15 references, 3 supplements.

API, FIGMA, FRAMEWORK, JAVASCRIPT, NEST.JS, REACT.JS.

The object of study is system, tools for interaction between React and database, means of software interaction with the user.

The aim of the study is to develop an online store for the sale of household goods.

The methods of research are modeling, design, software, analytical.

In our modern world, there are a large number of online stores that offer household goods for sale. One of them is my innovative online store on the React platform. This website provides users with a convenient and secure platform to purchase a variety of household goods. It is developed using the powerful React framework, which ensures fast and efficient website performance.

One of the main advantages of my online store is its intuitive and easy-to-use interface. Users will be able to quickly find the products they need using a smart search and filtering system. In addition, this website offers detailed product descriptions, user reviews, and a convenient checkout system.

As a result, we have created a convenient and efficient online store selling household goods.

**ЗМІСТ**

# ВСТУП

У сучасному цифровому світі інтернет-магазини стали невід'ємною частиною нашого повсякденного життя. Завдяки їх зростаючій популярності все більше людей обирають зручну та доступну форму електронної торгівлі. Однак, успішність інтернет-магазину залежить від його грамотного проєктування, UI та використання сучасних технологій. У цьому контексті ця дипломна робота присвячена розробці інтернет-магазину для продажу побутових товарів з використанням бібліотеки React.js.

Мета роботи полягає у створенні зручного та ефективного інтернет-магазину, що забезпечує взаємодію між покупцями та продавцями побутових товарів. З метою досягнення цієї мети, в роботі будуть виконані наступні завдання:

- огляд інструментів розробки;
- аналіз предметної області;
- проєктування інтернет-магазину;
- розробка користувацького інтерфейсу.

# 1 ТЕХНІЧНЕ ЗАВДАННЯ

## 1.1 Терміни та визначення

Додаток – програмне забезпечення або програма, яку можна відкрити за допомогою будь-якого браузера.

React – потужна та популярна бібліотека JavaScript, яка використовується для розробки користувацького інтерфейсу веб-додатків [1].

Користувач – людина, котра зареєстрована у системі.

Адміністратор – користувач, котрий має права на редагування додатку.

Інформаційна система (ІС) – система що реалізує функцію збору, зберігання, обробки і передачі інформації [2].

База даних (БД) – це будь-яка пов’язана між собою за певними ознаками інформація, що зберігається і організується певним чином, як правило у вигляді таблиць [3].

User experience (UX) – це підхід до розробки продуктів або послуг, який фокусується на задоволенні потреб користувачів та покращені їх взаємодії з продуктом [4].

## 1.2 Функціональні вимоги

### 1.2.1 Призначення і цілі створення додатку

Функціональне призначення додатку – реалізувати можливість покупки товару в інтернет магазині.

Експлуатаційне призначення додатку: додаток може експлуатуватися як адміністратором так і користувачем на різних рівнях доступу.

Мета створення додатку – розробка інтернет магазину для покупки побутових товарів.

### 1.2.2 Загальні функціональні можливості додатку

Додаток має надавати користувачам такі можливості:

- додавання/видалення товарів у кошику;
- додавання/видалення товарів в/з «Wishlist»;
- реєстрація в додатку;
- додавання/видалення особистих даних;
- перегляд товарів;
- вхід/вихід до/з додатку;
- оформлення замовлення товару;
- оплата товару.

Додаток має надавати адміністраторам такі можливості:

- створення та перегляд користувачів;
- перегляд статистики додатку;
- додавання категорій та підкатегорій товарів;
- додавання товарів та знижок до них.

## 1.3 Нефункціональні вимоги

До нефункціональних вимог відносяться: інтерфейс користувача, вимоги по продуктивності та вимоги до безпеки. Розглянемо їх детальніше.

*Інтерфейс користувача*. Додаток зобов’язаний відображати інтерфейс красиво та коректно як на комп’ютерах так і на мобільних девайсах.

*Вимоги до продуктивності*. Додаток повинен працювати для наступних браузерів останніх версій:

- Mozilla Firefox.
- Google Chrome.
- Opera.

Додаток повинен відображати всі сторінки не довше, ніж за 3 секунди.

Додаток повинен відправляти повідомлення не довше, ніж 5 секунд.

*Вимоги до безпеки.* Додаток зобов'язаний не давати доступ до функцій адміністрування сайту користувачам які не мають статус адміністратора.

Додаток зобов'язаний не давати доступ до функцій сайту користувачам які не зареєструвались у ньому.

### 1.4 Опис предметної області

Предметною областю є розробка додатку для покупки побутової техніки. Цей додаток повинен надати користувачам можливість покупки товарів користувачем, додавання їх в улюблене для подальшох покупки чи перегляду або додавання власної інформації для подальшого її використання при замовленні товару. Процес взаємодії з додатком проходить наступним чином. Потенційний користувач, повинен ввести адресу сайту в браузері та ввести дані у форму входу до системи. Якщо користувач вперше в інтернет магазині, він повинен спочатку зареєструватися (додати ім'я, прізвище, пошту та пароль).

Після входу в додаток користувач має можливість взаємодіяти з такими розділами як: категорії товарів, особисті дані, каталог, улюблені товари.

Процес покупки товару відбувається наступним чином. Користувач переходить в каталог товарів, додаток відображає інтерфейс фільтрації товарів. Для покупки товару користувач переходить на сторінку товару і натискає кнопку «ADD TO CART». Після цього користувач переходить на сторінку кошику де бачить товар. Потім він переходить на сторінку «Checkout», де бачить свої персональні дані, які він може змінити для доставки товару, та натискає «CONFIRM THE ORDER».

Окрім основних функцій додаток надає адміністраторам можливість переглядати статистику користувачів та їх покупки.

## 1.5 Структура додатку

Назва додатку: LaPigeon.

Опис: LaPigeon є інтернет-магазином, спеціалізованим на продажі побутових товарів. Додаток розроблений з метою надати зручну і просту платформу для покупців, які шукають товари для свого дому, офісу або інших потреб.

Основні функції додатку:

1) реєстрація та вхід користувача:
   – користувачі можуть створювати облікові записи в додатку, використовуючи свою електронну пошту або соціальні мережі;
   – вхід в обліковий запис за допомогою електронної пошти та пароля або іншого засобу автентифікації, наприклад, відбитка пальця або розпізнавання обличчя;
2) пошук товарів:
   – користувачі можуть шукати товари за категоріями, підкатегоріями, назвою або ключовими словами;
   – розширений пошук за додатковими параметрами;
3) каталог товарів:
   – користувачі можуть переглядати широкий асортимент побутових товарів, включаючи електроніку, побутову техніку, меблі, побутові прилади тощо;
   – кожен товар має сторінку з детальним описом, зображеннями, ціною, відгуками користувачів;
4) кошик покупок:
   – користувачі можуть додавати товари до свого кошика покупок;
   – можливість зміни кількості товарів, видалення товарів або збереження їх для майбутнього придбання;
5) оформлення замовлення:
   – користувачі можуть оформити замовлення на придбання товарів зі

свого кошика;
- вибір способу доставки та оплати;

6) оплата:
   - забезпечення безпеки платежів та захисту персональних фінансових даних користувачів;
7) відстеження замовлення:
   - користувачі можуть відстежувати статус своїх замовлень;
8) рейтинг та відгуки:
   - користувачі можуть залишати відгуки про придбані товари та оцінювати їх;
   - інші користувачі можуть переглядати відгуки та рейтинги, щоб зробити більш обґрунтований вибір при покупці;
9) користувацький профіль:
   - користувачі мають особисті профілі, де вони можуть зберігати свої адреси доставки, відстежувати історію замовлень, зберігати улюблені товари та налаштовувати свої уподобання.

## 1.6 Огляд інструментів розробки

### 1.6.1 Огляд JavaScript UI фреймворків

У цій секції проведено аналіз трьох популярних JavaScript-фреймворків для розроблення інтернет-магазинів із продажу побутових товарів: React, Angular і Vue. Кожен із цих фреймворків має свої унікальні особливості та переваги [5].

React дає змогу створювати компоненти користувацького інтерфейсу, які можна перевикористовувати й ефективно керувати. React використовує віртуальний DOM, що дає змогу ефективно оновлювати тільки змінені частини інтерфейсу, покращуючи продуктивність під час роботи з великими

обсягами даних. Він також має активну спільноту розробників і велику документацію, що полегшує вивчення і підтримку фреймворка [6].

Angular є повноцінним фреймворком для створення складних веб-додатків. Він заснований на TypeScript і використовує концепцію двостороннього зв'язування даних, що забезпечує потужні можливості для управління станом програми. Однак, Angular може бути складнішим для вивчення і має більш громіздку структуру порівняно з іншими фреймворками [7].

Vue є прогресивним JavaScript-фреймворком, який забезпечує простоту і гнучкість розробки. Він має невеликий розмір, що полегшує його впровадження в проєкти, і дає змогу створювати компоненти з використанням шаблонів і однофайлових компонентів. Vue також надає зручні інструменти для управління станом додатка і має хорошу продуктивність. Він швидко набирає популярності та має активну спільноту підтримки [8].

Після уважного розгляду всіх трьох фреймворків і аналізу вимог до розроблюваного інтернет-магазину побутових товарів, було ухвалено рішення вибрати React як основний інструмент розробки. Кілька ключових факторів, які вплинули на це рішення, включають:

- простота: React надає просту і зрозумілу модель розробки на основі компонентів, що спрощує розуміння і підтримку коду;
- гнучкість: React дає змогу розробникам використовувати додаткові бібліотеки та інструменти на свій розсуд;
- продуктивність: завдяки віртуальному DOM і механізму оновлення тільки змінених компонентів, React демонструє хорошу продуктивність навіть під час роботи з великими обсягами даних (це особливо важливо для інтернет-магазину, де швидке завантаження і чуйність відіграють ключову роль);
- велике співтовариство та екосистема: React має широке співтовариство розробників, що забезпечує доступ до безлічі ресурсів, бібліотек та інструментів (це спрощує розробку, підтримку

і розширення інтернет-магазину в майбутньому).

З огляду на ці фактори і вимоги проєкту, вибір React як фреймворку для розробки інтернет-магазину з продажу побутових товарів є найбільш обґрунтованим і ефективним рішенням.

### 1.6.2 Інструменти для дизайну інтерфейсів

Під час вибору інструменту для розроблення інтерфейсу інтернет-магазину, я розглянув кілька аналогів, включно зі Sketch, Adobe XD і Figma. Однак мій вибір пав на Figma з таких причин [9]:

Веб-додаток: Figma є повністю веб-орієнтованим інструментом, що дає мені змогу працювати над проєктом із будь-якого комп’ютера з доступом в Інтернет.

Прототипування: Figma пропонує інтуїтивний інтерфейс для створення інтерактивних прототипів. Я можу додавати переходи між екранами, створювати анімації та тестувати користувацький досвід безпосередньо в рамках платформи. Це допомагає мені оцінити потік роботи і взаємодію елементів інтерфейсу до початку фактичної розробки.

Адаптивний дизайн: Figma дає змогу розробляти інтерфейси, враховуючи адаптивність на різних пристроях і роздільних здатностях екранів. Я можу створювати компоненти, які автоматично адаптуються під різні розміри екранів, що економить час і спрощує процес створення реагуючого дизайну.

Зрештою, я вибрав Figma як інструмент для розробки інтерфейсу інтернет-магазину через його веб-орієнтованість, можливості спільної роботи, прототипування і підтримки адаптивного дизайну. Ці функції забезпечують зручність, ефективність і гнучкість у процесі розробки інтерфейсу та спільної роботи з командою.

### 1.6.3 Використання API

Під час розроблення інтернет-магазину побутових товарів, було ухвалено рішення використати готовий API на Nest,js для забезпечення взаємодії із сервером [10, с. 3–14]. Після ретельного аналізу та порівняння цих інструментів, було визначено, що чек-лист хоч і є обов'язковим для даного проекту але впоратись з ним одним практично неможливо. Під час вибору API було проведено порівняння кількох популярних варіантів з урахуванням конкретних критеріїв. Розглянемо їх нижче.

*Гнучкість і функціональність*. Основним критерієм була здатність API забезпечити необхідний функціонал для розробки інтернет-магазину. Важливими функціями були управління товарами, категоріями, кошиком покупця, опрацювання замовлень і аутентифікація користувачів. API на Nest.js, який вже був розроблений, надавав усі необхідні ендпоінти та функціональність для реалізації необхідного функціоналу інтернет-магазину.

*Підтримка і документація*. Під час розробки важливо мати надійну підтримку і повноцінну документацію, щоб швидко розібратися у функціональності та використанні API. Nest.js мав активне співтовариство розробників і велику документацію, що полегшило інтеграцію та вирішення проблем, які виникали.

*Розширюваність*. Розробнику також необхідно мати можливість розширювати функціональність і впроваджувати необхідні зміни в API відповідно до вимог інтернет-магазину. API на Nest.js було розроблено з урахуванням модульності та розширюваності, що давало змогу гнучко налаштовувати та модифікувати його для задоволення потреб проєкту.

З огляду на проведене порівняння, було ухвалено рішення використовувати API на Nest.js для розробки інтернет-магазину на React. Його гнучкість, функціональність, підтримка та розширюваність є ключовими перевагами, що відповідають вимогам проєкту і забезпечують ефективну розробку та функціонування інтернет-магазину побутових товарів.

# 2 ПРОЄКТУВАННЯ

Процес проєктування є надзвичайно важливим етапом у створенні нових продуктів, систем або рішень. Його комплексність вимагає ретельного планування та виконання послідовних кроків і дій для досягнення ефективного і функціонального рішення для заданої проблеми або потреби [11].

Під час процесу проєктування додатку було витрачено значний час на розробку діаграм та вайрфреймів які незабаром будуть продемонстровані в цьому розділі. Використовуючи потужний інструмент дизайну Figma, я зміг створити високоякісні візуальні елементи і забезпечити зручну роботу з макетами. Вибір Figma як основного інструменту для проєктування був детально обгрунтований у минулому розділі.

Кожен етап процесу проєктування вимагав великої уваги і виконувався лише після завершення попередніх етапів. Це дозволило систематично просуватись у напрямку кінцевої мети і забезпечити якісний результат.

В результаті наполегливої роботи і виконання всіх необхідних етапів проєктування, можна бути впевнененим в тому, що мої рішення готові відповідати вимогам і забезпечувати задоволення потреб користувачів.

## 2.1 Діаграма прецедентів

Діаграми прецедентів є корисним інструментом під час розробки інтернет-магазину [12]. Вони допомагають ідентифікувати й описувати різні функціональні можливості системи, представляючи їх у вигляді взаємодій між акторами та прецедентами. З використанням діаграм прецедентів можна виділити основні функціональні блоки системи, визначити взаємодії між ними

і встановити пріоритети розробки. Саме цьому ця діаграма була зроблена і використана першою, що значно допомогло у плануванні всього додатку.

У підсумку, діаграма прецедентів допомогла мені створити більш структуровану і зрозумілу модель інтернет-магазину. Це дало змогу точніше визначити вимоги та цілі проєкту, а також поліпшити користувацький досвід.

На рисунку 2.1 наведено приклад схеми інтернет-магазину.


Рисунок 2.1 – Діаграма варіантів використання інтернет-магазину

Розглянемо інформацію про акторів (табл. 2.1), а також опис варіантів використання (табл. 2.2).

Таблиця 2.1 – Опис акторів

| Назва | Опис |
|---|---|
| Користувач | Користувач інтернет магазину |
| Адміністратор | Користувач, що має права «адміністратора» |
| База даних | База даних, що зберігає всю інформацію |

Таблиця 2.2 – Опис варіантів використання

| Назва | Опис |
|---|---|
| Пошук товару | Можливість пошуку товару на сайті |
| Додавання відгуків | Функція, щоб надати можливість користувачу залишати відгуки про товари |
| Авторизація | Функція входу в систему |
| Перегляд товару | Користувач має можливість переглянути товар |
| Перегляд замовлень | Адміністратор може переглядати замовлення користувачів сайту |
| Створення товару | Адміністратор може створювати нові товари |
| Додання товару в кошик | Функція для додання товару в кошик для подальшого оформлення замовлення |
| Додавання в улюблене | Функція для додавання товару в улюблені товари для подальшого покупки |
| Створення замовлення | Функція, щоб надати користувачу можливість замовляти товари |
| Реєстрація | Функція реєстрації у системі |
| Створення категорій товарів | Адміністратор може створювати різні категорії товарів для додавання товарів у систему |

## 2.2 Діаграма послідовності

Діаграма послідовності є корисним інструментом під час розробки інтернет-магазину. Вона дає змогу візуалізувати послідовність дій і комунікацію між різними об'єктами системи в певній ситуації або сценарії використання.

Оскільки покупка товару є клювочою функцією інтернет-магазину, проєктування її роботи однією з найважливіших стадій проєктування додатку що в подальшому значно полегшило її розробку.

На рисунку 2.2 описана діаграма послідовності процесу замовлення товару.

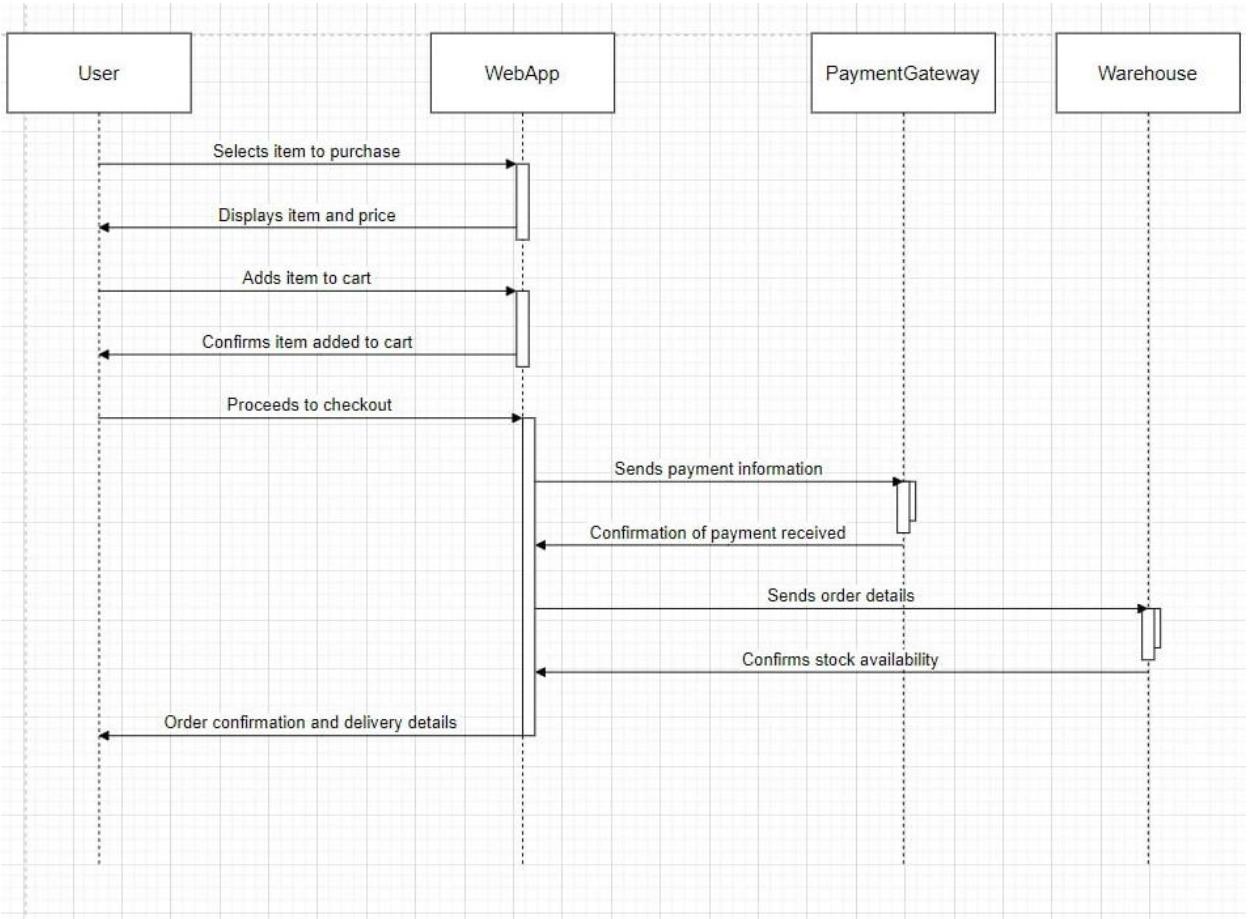


Рисунок 2.2 – Діаграма послідовності замовлення товару

## 2.3 Вайрфреми

Вайрфрейми є надзвичайно корисним інструментом під час розробки інтерфейсів та веб-сторінок [13]. Вони забезпечують простий та наочний спосіб показати структуру та компоненти інтерфейсу без зайвого акценту на дизайні та візуальному оформленні. Саме тому вони стали невід'ємною частиною процесу проєктування інтернет-магазину, дозволивши отримати повноцінне уявлення про те, як буде виглядати кінцевий додаток. За допомогою вайрфреймів була проаналізована візуалізація розміщення елементів, логіка навігації, взаємодія з користувачем та інші ключові аспекти інтерфейсу. Застосування вайрфреймів допомагло зменшити кількість непотрібних змін на етапі розробки, оскільки вони дали можливість заздалегідь побачити та обговорити основні аспекти інтерфейсу.

Розглянемо вайрфрейми головної сторінки (рис. 2.3), сторінки продукту (рис. 2.4), кошику (рис. 2.5) та сторінки розрахунку (рис. 2.6).

Під час розробки вайрфрейму головної сторінки (див. рис. 2.3) був проведений аналіз конкурентів, який допоміг більш детально розглянути яку інформацію ми хочемо продемонструвати користувачу. Готовий варіант сторінки можна буде побачити на рисунку 3.8. Головна сторінка була розділена на декілька частин:

Шапка сайту – як і у всіх конкурентів, шапка сайту повинна бути основним інструментом навігації, і наш додаток не став винятком. Вона була розділена на 2 частини. Верхня частина містила додаткову інформацію для користувача та можливість вибору міста. Нижня ж частина відповідала повністю за навігацію по сторінці. Тут були розташовані пошук товарів, вибір мови, можливість перейти до налаштування профілю користувача або зареєструватися, якщо він ще не мав облікового запису. Пізніше було прийнято рішення відображати шапку та підвал сайту на кожній сторінці, що значно спрощувало користування магазином для потенційних клієнтів.

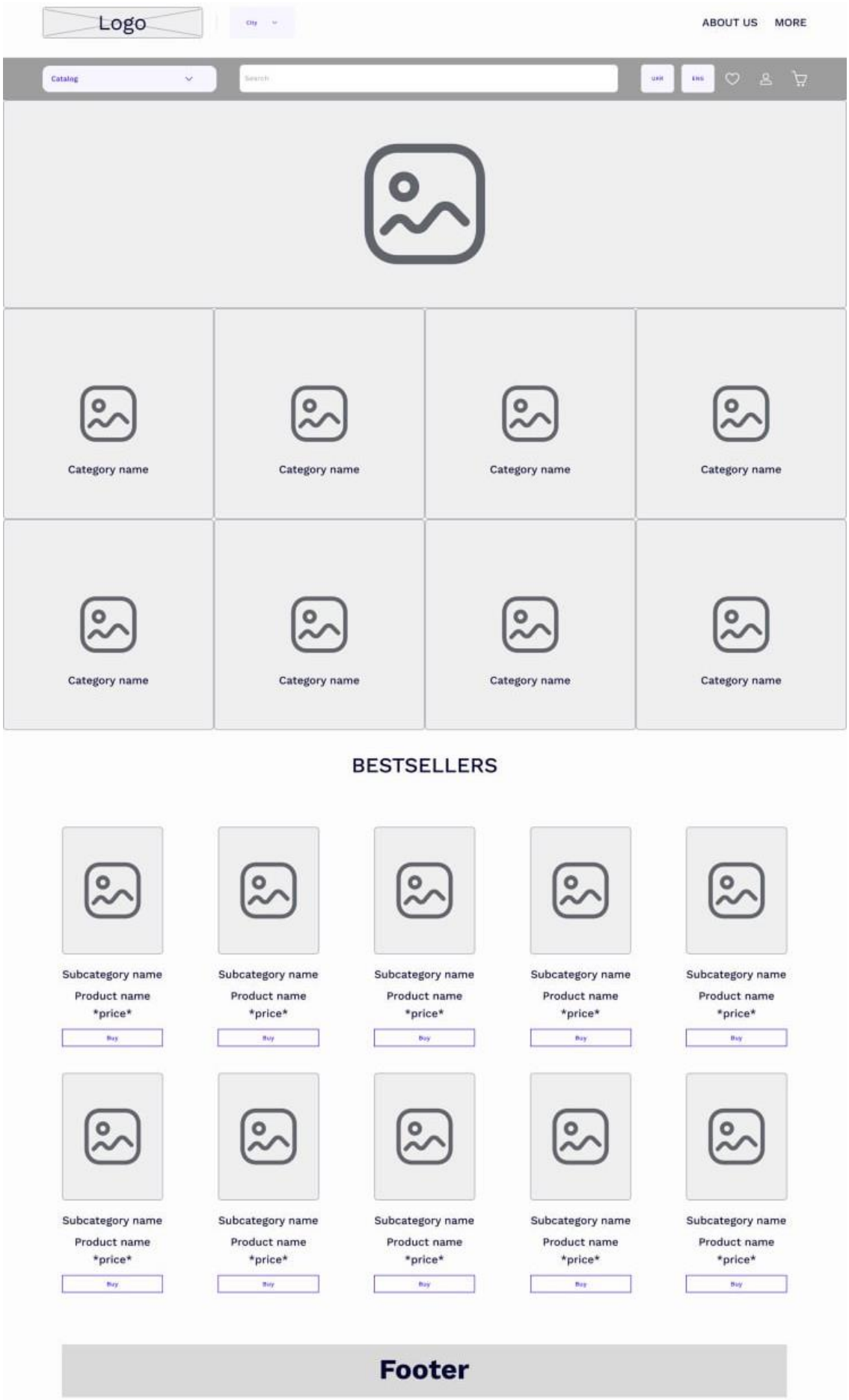


Рисунок 2.3 – Вайрфрейм головної сторінки інтернет-магазину

Контент сайту повинен був надавати можливість користувачам одразу бачити товари та їх ціни, і це було зроблено. Також було прийнято рішення додатково розмістити категорії товарів з ілюстраціями, щоб користувачам було легше орієнтуватися на сайті.

Підвал сайту, як і у всіх інтернет-магазинах, повинен містити всі контакти з компанією, що керує проєктом, та навігацію до всіх доступних сторінок. На даному етапі проєкту не було необхідності детального проєктування підвалу, тому була продемонстрована лише його наявність.

Наступним вайрфреймом стала сторінка продукту (рис. 2.4). Вона була найбільш складною через велику кількість інформації, яку потрібно було продемонструвати користувачу, тому її деталізація була значно вищою, ніж у інших вайрфреймах проєкту.

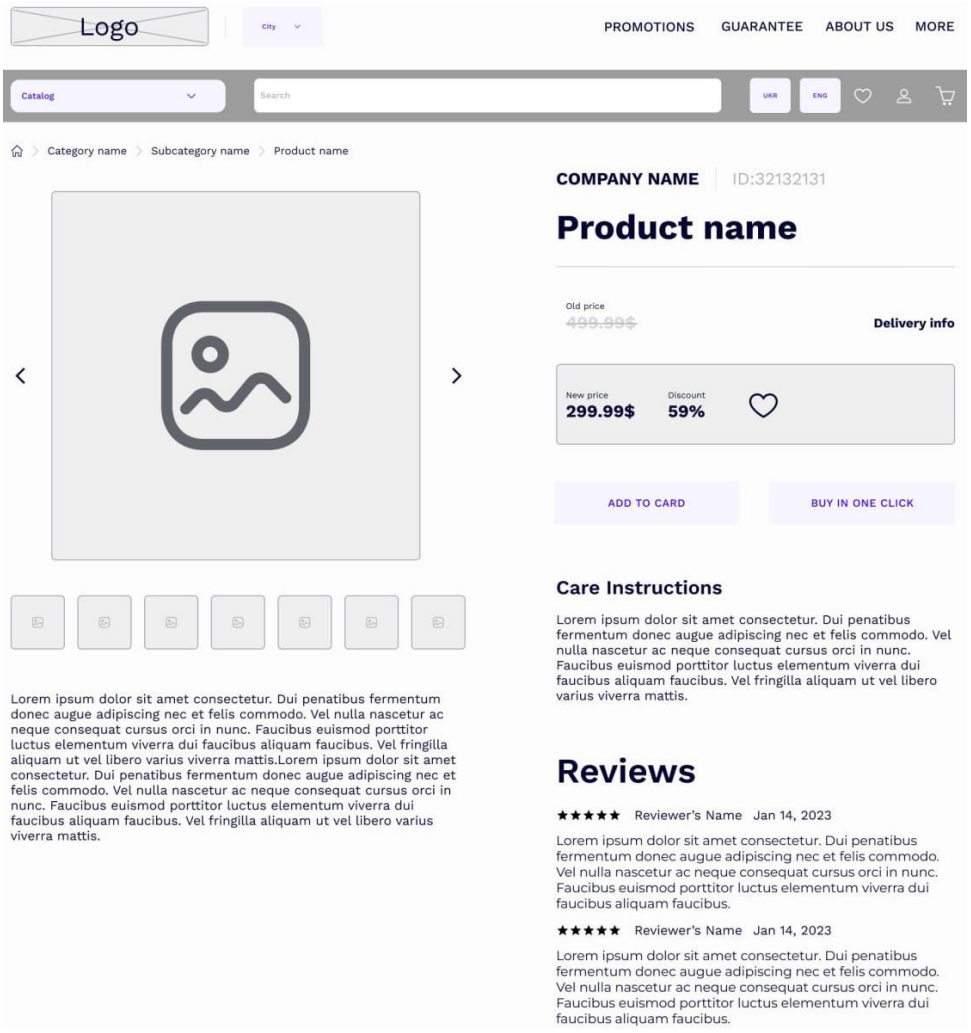


Рисунок 2.4 – Вайрфрейм сторінки продукту інтернет-магазину

Ця сторінка містить зображення товару, опис, назву, ціну та знижку, у видапку якщо вона є, а також відгуки та можливість придбання товару. Кожен з цих компонентів має велике значення, тому детальний опис сторінки продукту став обов'язковим кроком. Її реалізацію можна буде побачити на рисунку 3.13.

Логічним продовженням сторінки продукту стала сторінка кошика (рис. 2.5). Розробка цього вайрфрейму була найпростішою через малу кількість деталей.

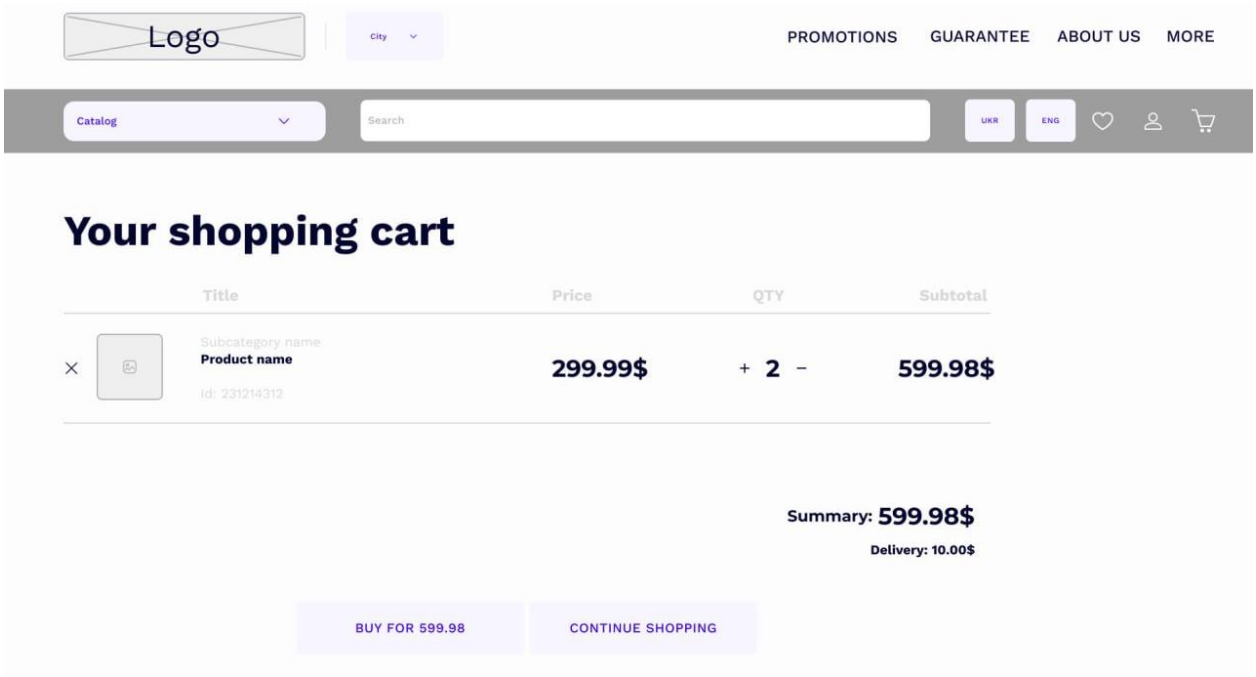


Рисунок 2.5 – Вайрфрейм сторінки кошику інтернет-магазину

Головним компонентом сторінки кошика є інформація про товар, який був доданий, а саме: фотографія товару; назва категорії товару; назва товару; ціна; кількість.

За замовчуванням всі товари додаються з кількістю в один товар, але на цьому етапі їх кількість можна змінити, натиснувши на відповідну іконку, або повністю видалити.

Реалізацію сторінки кошика можна побачити на рисунку 3.14.

Сторінка розрахунку (див. рис. 2.6) хоч і є останньою у списку, але і

кількість інформації яку вона в собі несе значно вища за попередні оскільки на цьому етапі наш додаток повинен підтягувати абсолютно всі дані користувача. Як інформація про товар, яка була описана у минулому абзаці, так і інформація про самого користувача.

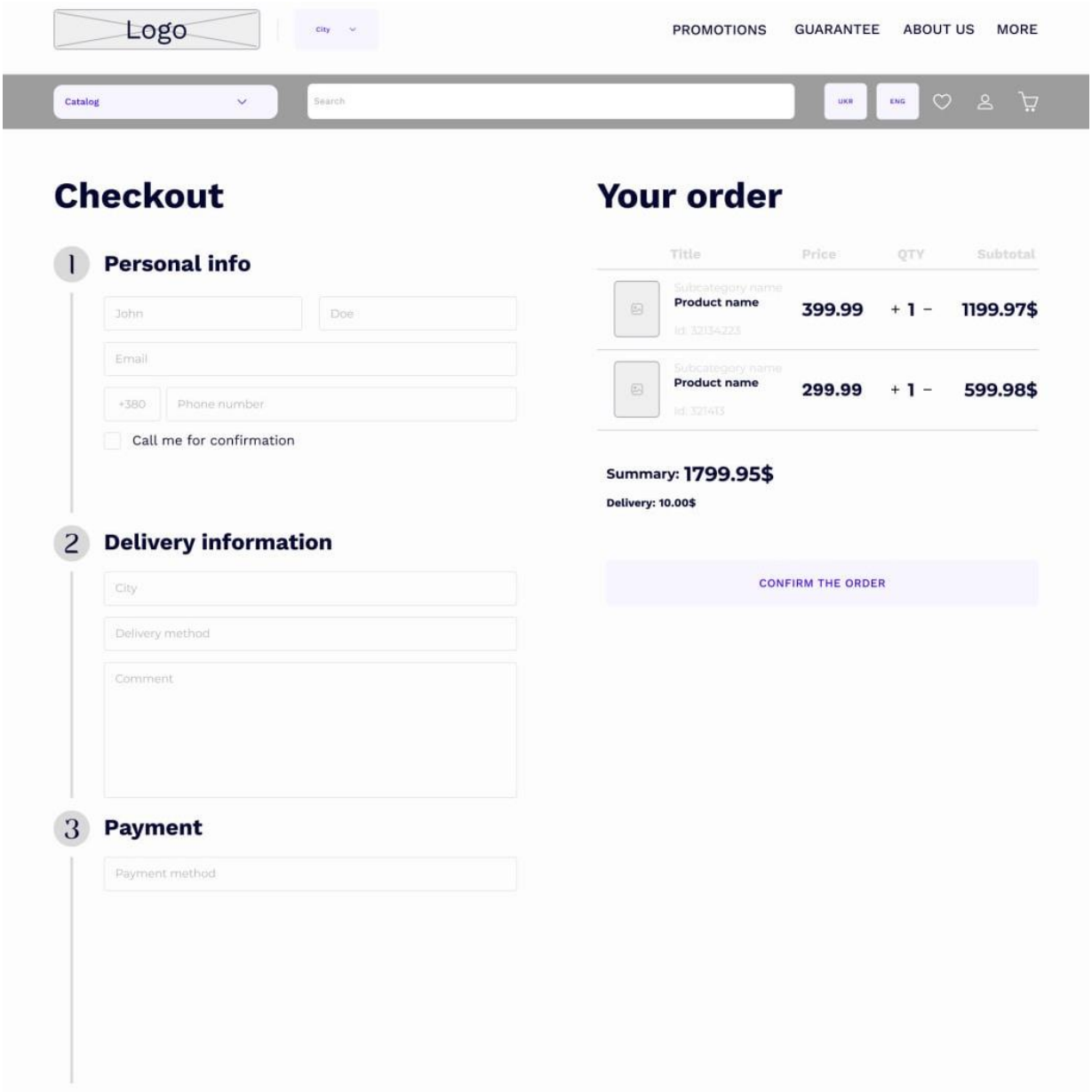


Рисунок 2.6 – Вайрфрейм сторінки розрахунку інтернет-магазину

Цей вайрфрейм є логічним завершенням основного сценарію та став вирішуючим фактором, оскільки одразу після його малювання була почата робота над етапом розробки продукту. Реалізацію цього вайрфрейму можна побачити на рисунку 3.15.

# 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

Однією з головних переваг React, який було використано під час розроблення додатку, є величезна кількість корисних бібліотек, які в комбінації з хуками значно прискорили процес розроблення інтернет-магазину. Нижче будуть коротко описані деякі з них, а в наступних пунктах третього розділу детально розглянуто користь їх застосування.

Axios – це бібліотека JavaScript, яка використовується для здійснення HTTP-запитів з браузера або середовища Node.js. Вона надає простий та зручний інтерфейс для взаємодії з API та отримання даних з сервера [14].

React Hook Form – це бібліотека для керування станом і валідації форм у React з використанням хуків. Вона надає простий та інтуїтивно зрозумілий спосіб створення форм, опрацювання користувацького введення, валідації даних і надсилання форми.

Yup – це бібліотека валідації схеми даних для JavaScript. Вона дає змогу визначати та застосовувати правила валідації до даних, включно з перевіркою типів, обов'язкових полів, довжини рядка, формату дати та інших варіантів валідації.

## 3.1 Структура проєкту

Каталог проєкту є складним (див. рис. 3.1).

Каталог client містить основний код програми.

Файл client/public/index.html – основна html-сторінка програми.

Каталог client/src/components містить всі React компоненти додатку.

Файл client/src/TokenContext.js відповідає за зберігання токену та відображення елементів веб сторінки які від нього залежать.

Файл client/src/App.js – головний файл який відповідає за роутінг та

відображення компонентів.

Каталог Backend тримає у собі всю логіку бекенду.

Файл client/src/utils/axios/instance.js відповідає за відправку даних на бекенд.

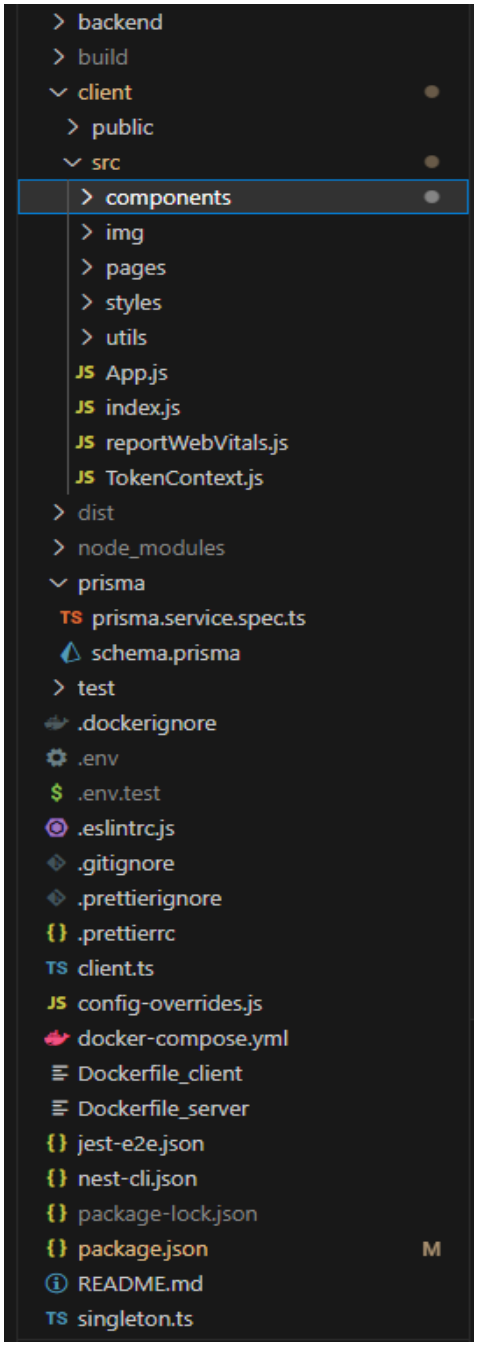


Рисунок 3.1 – Каталог проєкту

Отже, проєкт містить багато підкаталогів та файлів які підтримують функціювання системи в цілому.

### 3.2 React-компоненти

У кожного програміста різних підхід при розробці застосунків тому що у кожного є свої переваги та недоліки. Тим не менш, є деякі кроки які гарантовано треба пройти. Ось як виглядав процес створення майже кожного компонента у цій роботі:

- створення папки для кожного компонента;
- створеняя файлу jsx та css;
- імпортування стилів з css файлу та бібліотек для використання React хуків;
- написання jsx коду та css до нього;
- додавання цього компоненту у app.js файл для роумінгу;
- підв’зування компоненту до ендпоінту та його тестування;

Візуалізація процесу та приклад коду компонента наведено у Додатку А.

### 3.3 Процес створення картки товару

В пункті 3.2 вже описано, як створювався кожен компонент React у додатку. Картка товару не була виключенням, оскільки вона була і є ключовим компонентом будь-якого інтернет-магазину, і спочатку створювалась саме за допомогою кроків, описаних у попередньому пункті.

Головною задачею картки товару є не тільки відображення основної інформації про товар у стислому вигляді, а й надання користувачу можливості додати цей товар у кошик або до улюблених товарів. Нижче наведено приклад написання цих функцій за допомогою хуків useState (рис. 3.2).

Представлений код перевіряє, чи натиснув користувач відповідну кнопку та відображає змінений компонент. Оскільки користувач повинен мати змогу одразу видаляти предмети з улюбленого, була створена перевірка на наявність цього товару у базі, та видалення його за унікальним

ідентифікатором. Візуалізація процесу та приклад повного коду наведено у Додатку Б.

```
const [isAddedToCart, setIsAddedToCart] = React.useState(false);
 const [isAddingToCart, setIsAddingToCart] = React.useState(false);
 const [isInWishlist, setIsInWishlist] = React.useState(itsInWishlist);
 const handleAddToCartClick = () => {
  if (!isAddingToCart) {
   setIsAddingToCart(true);
   onAddToCart()
    .then(() => {
     setIsAddedToCart(true);
     setIsAddingToCart(false);
    })
    .catch((error) => {
     console.error(error);
     setIsAddingToCart(false);
    });
  }
 };
 const handleAddToWishlistClick = () => {
  if (!isInWishlist) {
   onAddToWishlist()
    .then(() => {
     setIsInWishlist(true);
    })
    .catch((error) => {
     console.error(error);
    });
  }
 };

 const handleRemoveFromWishlist = () => {
  if (isInWishlist) {
   onRemoveFromWishlist()
    .then(() => {
     setIsInWishlist(false);
    })
    .catch((error) => {
     console.error(error);
    });
  }
 };
```

Рисунок 3.2 – Приклад додавання картки товару у кошик та улюблене

### 3.4 Процес валідації форм

Є безліч способів для валідації форм. Користувач повинен бачити в чому помилка та система повинна винагороджувати користувача за кожний клік в додатку. Таким чином користувач відчуває що система добре працює.

У випадку цього додатку були використані деякі бібліотеки для валідації а саме React-hook-form та Yup більше детально про які було написано на початку розділу. Вони дуже легкі і вже мають всі необхідні методи які можна кастомізувати в залежності від бажань. Нижче приведено приклад використання Yup (рис. 3.3).

```
const schema = yup.object().shape({
  email: yup
    .string()
    .email('Invalid email')
    .lowercase()
    .trim()
    .max(255, 'Email must be at most 255 characters')
    .required('Email is required'),
  password: yup
    .string()
    .required('Password is required')
    .min(8, 'Password must be at least 8 characters')
    .max(20, 'Password must be at most 20 characters')
.matches(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*()_+\-=\[\]{};':"\\|,.<>\/?])/,
      'Must contain at least one uppercase, one lowercase, one digit and one special character'
    ),
});
```

Рисунок 3.3 – Приклад валідації даних на сторінці логіну

Представлений код примаймає в себе дані введені у поле та валідує їх за допомогою методів які обробляють цю інформацію. Ця функція процює за

рахунок бібліотеки Yup, котра була детально розглянута на початку розділу.

Візуалізація процесу та приклад повного коду наведено у Додатку В.

### 3.5 Тестування додатку

У процесі порівняння найвідоміших інструментів тестування для розробки інтернет-магазину, було враховано кілька аспектів. Вибір падав на такі інструменти, як ручне тестування, чек-лист та тест-кейси.

Оскільки ручне тестування є обов’язковим та регулярно проходилось як на етапі розробки так і після нього треба визначити що використувавати – чек-листи, тест-кейси чи обидва одночасно.

Після ретельного аналізу та порівняння цих інструментів, було визначено, що чек-лист хоч і є обов’язковим для даного проєкту але впоратись з ним одним практично неможливо, тому було прийнято рішення використовувати одразу декілька інструментів. Пізніше це допомогло протестувати кожну деталь інтернет-магазину окремо і впевнетись у працездатності проєкту. Основними причинами такого вибору стало те, що чек-листи призначені для поверхневого аналізу і в середніх або великих додатках їх використання є лише етапом у тестуванні за яким зазвичай йдуть тест-кейси, які відповідають за тестування кожного компонента окремо [15, с. 393–439].

Таким чином, додаток протестовано на всіх рівнях і його функціональність більше не могла ставитись під сумнів.

На рисунку 3.4 детально зображено чек-лист верстки інтернет магазину.

Наступним кроком після написання та первірки чек-листу стало тестування за допомогою тест кейсів. Оскільки реєстрація є ключовою функцією для користування нашим інтернет-магазином, тест-кейс для неї був написаний у першу чергу (див. рис. 3.5).

| | A | B | C | D |
|---|---|---|---|---|
| 1 | *Чек-лист* | *Google Chrome* | *Opera* | *Mozila Firefox* |
| 2 | **Наявність елементів сторінок** | | | |
| 3 | Перевірка наявності шапки сайту | Passed | Passed | Passed |
| 4 | Перевірка наявності головної сторінки сайту | Passed | Passed | Passed |
| 5 | Перевірка нявності підвалу сайту | Passed | Passed | Passed |
| 6 | Перевірка відповідності дизайну | Passed | Passed | Passed |
| 7 | Перевірка наявності логотипу | Passed | Passed | Passed |
| 8 | **Відображення елементів сторінки** | | | |
| 9 | Перевірка відображення шрифтів | Passed | Passed | Passed |
| 10 | Перевірка відображення кнопок, блоків | Passed | Passed | Passed |
| 11 | Перевірка відображення банерів та якість картинок вцілому | Passed | Passed | Passed |
| 12 | **Активні елементи** | | | |
| 13 | Перевірити наявність підсказок | Passed | Passed | Passed |
| 14 | Перевірити реагування елементів на наведення курсору і натискання | Passed | Passed | Passed |
| 15 | **Зміст** | | | |
| 16 | Перевірити орфографію тексту | Passed | Passed | Passed |
| 17 | | | | |
| 18 | | | | |
| 19 | | | | |

Рисунок 3.4 – Чек-лист додатку

| | |
|---|---|
| 1 | **Реєстрація** |
| 2 | Перейти на сторінку інтернет-магазину |
| 3 | Натиснути на іконку користувача |
| 4 | Ввести ім'я у поле Name |
| 5 | Ввести прізвище у поле Surname |
| 6 | Ввести валідну адресу електронної пошти у поле Email |
| 7 | Ввести валідний пароль у поле Password |
| 8 | Натиснути на кнопку "SIGN UP" |
| 9 | Перевірити що іконка користувача змінила свій колір |
| 10 | **Очікуваний результат** |
| 11 | На кожному кроці не повинно виникати помилок |
| 12 | Після натискання на кнопку SIGN UP користувача повинно перевести на сторінку з відповідним сповіщенням |
| 13 | Іконка користувача тепер веде на сторінку налаштування акаунту |
| 14 | **Пояснення** |
| 15 | Валідна адреса електронної пошти - це адреса яка відповідая стандартам формата email (наприклад, example@gmail.com |
| 16 | Валідний пароль - це пароль який повинен відповідати деяким вимогам (мін. кільк. символів - 8, обов'язкові великі літери, цифри та один спеціальний символ |

Рисунок 3.5 – Тест-кейс реєстрації в інтернет магазині

### 3.6 Джерело даних

Будь-який застосунок повинен мати ендпоінти для з'єднання серверних процесів із зовнішнім інтерфейсом і LaPigeon не став винятком. У таблиці 3.1 будуть представлені деякі з найважливіших ендпоінтів для роботи інтернет-

магазину.

Таблиця 3.1 – Основні ендпоінти

| Назва | Опис |
|---|---|
| Signup | Реєстрація користувача у системі |
| Categories | Створення категорій продуктів |
| Products | Створення продукту (потребується вже створена категорія та підкатегорія продуктів) |
| Subcategories | Створення під категорій продуктів (потребується вже створена категорія продуктів |
| Wishlist | Додавання продукту в улюблене |
| Product/id/comments | Додавання коментаря до продукту |
| Carts | Додавання продукту до кошику |
| Cart/id/confirm | Створення замовлення |

Кожен з ендпоінтів представлений в таблиці 3.1 зберігає в собі величезну кількість даних які використовуються пізніше у інших не менш важливих ендпоінтах. Взаємодія з цими ендпоінтами була скоєна за допомогою специфікації OpenAPI, що значно прискорило подальше тестування

Нижче на рисунку 3.6 буде зображено приклад відповіді сервера на запит GET до ендпоінту «Products» у форматі yaml – одному з найзручніших форматів серіалізації даних.

Як вже було зазначено у вступі дипломної роботи, в якості технології для розробки API та серверної частини був використаний Nest.js. Причини його використання були описані у пункті 1.6.3. Використання даних з цього API здійснено за допомогою бібліотеки Axios, що значно зменшило кількість потребуємого коду.

Всім запитам на сервер були задані деякі параметри, головним з яких був токен, який діставався за допомогою веб-хуку useContext. Приклад використання цього API продемонстрований на рисунку 3.7.

```
id: 4
userId: 2
totalPrice: 2000
createdAt: '2023-06-13T14:34:29.150Z'
updatedAt: '2023-06-13T14:34:29.157Z'
cartItems:
- id: 6
  cartId: 4
  productId: 1
  quantity: 1
  subTotalPrice: 2000
  createdAt: '2023-06-13T14:34:29.150Z'
  updatedAt: '2023-06-13T14:34:29.150Z'
  product:
    subcategory:
      id: 15
      name: Smartphone
      subcategoryIconId: 15
      categoryId: 2
      createdAt: '2023-06-03T19:53:19.190Z'
      updatedAt: '2023-06-03T19:53:19.488Z'
    name: iPhone 14 Pro Max
    price: 2000
    productImages:
    - id: 1
      key: 7f5f9107-f61e-439e-838a-fee9c66b6047-cart_image2 (1).png
      productId: 1
      createdAt: '2023-06-03T19:56:05.045Z'
      updatedAt: '2023-06-03T19:56:05.045Z'
```

Рисунок 3.6 – Відповідь сервера на запит до «Products»

```
import axios from "axios";
import { useContext } from "react";
import { TokenContext } from "../../TokenContext";

const AxiosInstance = () => {
  const { token } = useContext(TokenContext);

  const instance = axios.create({
    baseURL: 'http://127.0.0.1:3001',
    timeout: 1000,
    headers: {
      'X-Custom-Header': 'foobar',
      'Authorization': `Bearer ${token}`
    }
  });

  return instance;
};

export default AxiosInstance;
```

Рисунок 3.7 – Файл instance.js

### 3.7 Приклад роботи системи

Після входу в інтернет магазин, ви потрапляєте на головну сторінку (рис. 3.8). На цій сторінці можна побачити категорії товарів та картки.

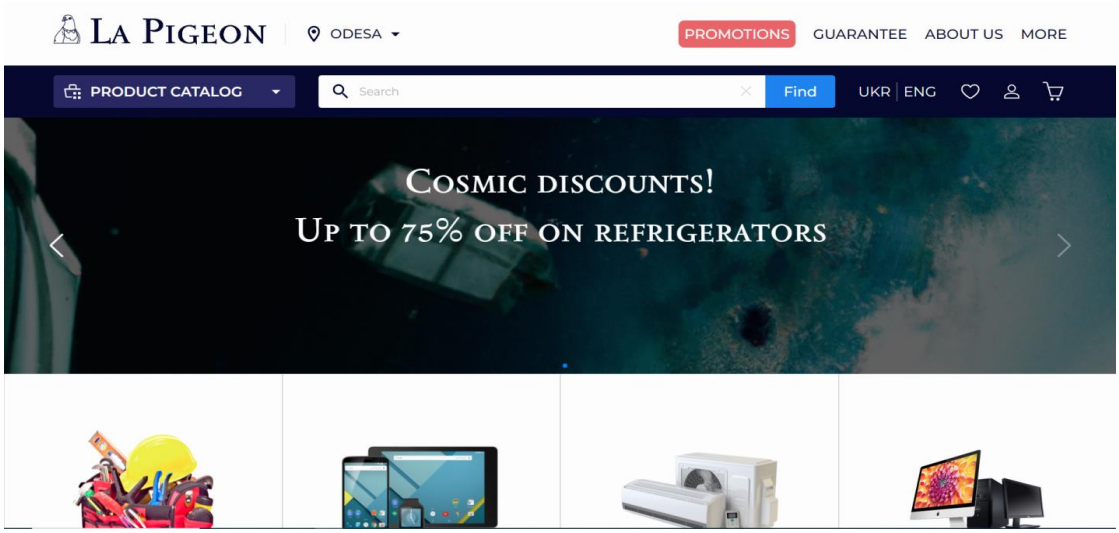


Рисунок 3.8 – Головна сторінка інтернет-магазину

Для придбання будь-якого товару потрідно зареєструватися, тому ви натискаєте на іконку користувача, і система відобразить форму реєстрації (рис. 3.9).

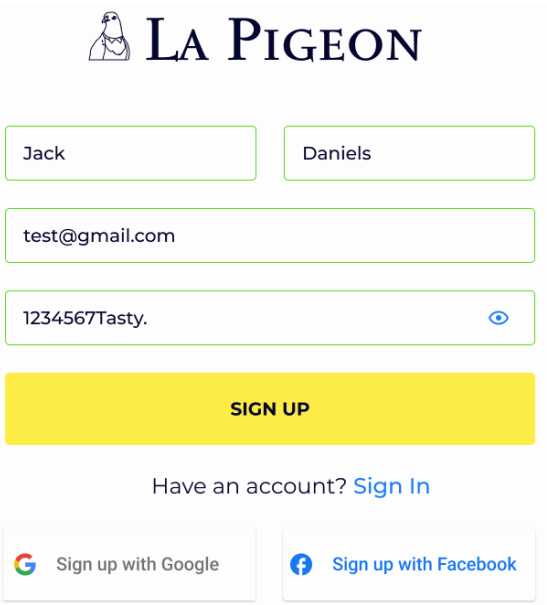


Рисунок 3.9 – Форма реєстрації

Під час заповнення форми, додаток буде автоматично перевіряти введені дані на валідність. Якщо користувач с цими даними вже є або пароль не відповідає вимогам, то система вас про це сповістить (рис. 3.10).

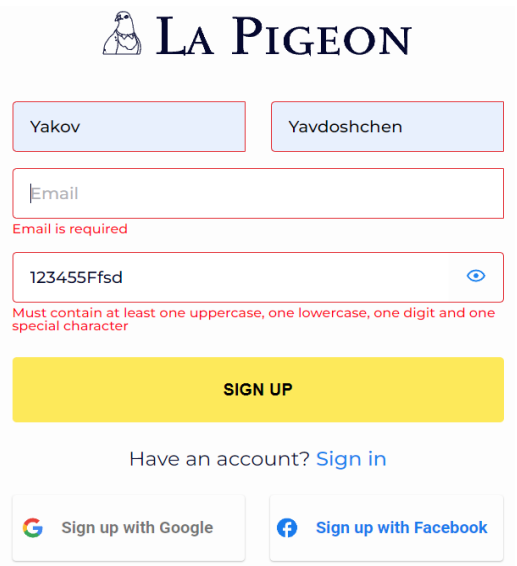


Рисунок 3.10 – Відображення помилок при введені невірних даних

Після того як користувач введе коректні та натискне кнопку «SIGN UP», він буде перенаправлений на сторінку, яка повідомляє йому про успішну реєстрацію (рис. 3.11), а кнопка, яка відкривала поп-ап реєстрації, загориться синім кольором та буде перенаправляти користувача у його особистий кабінет.

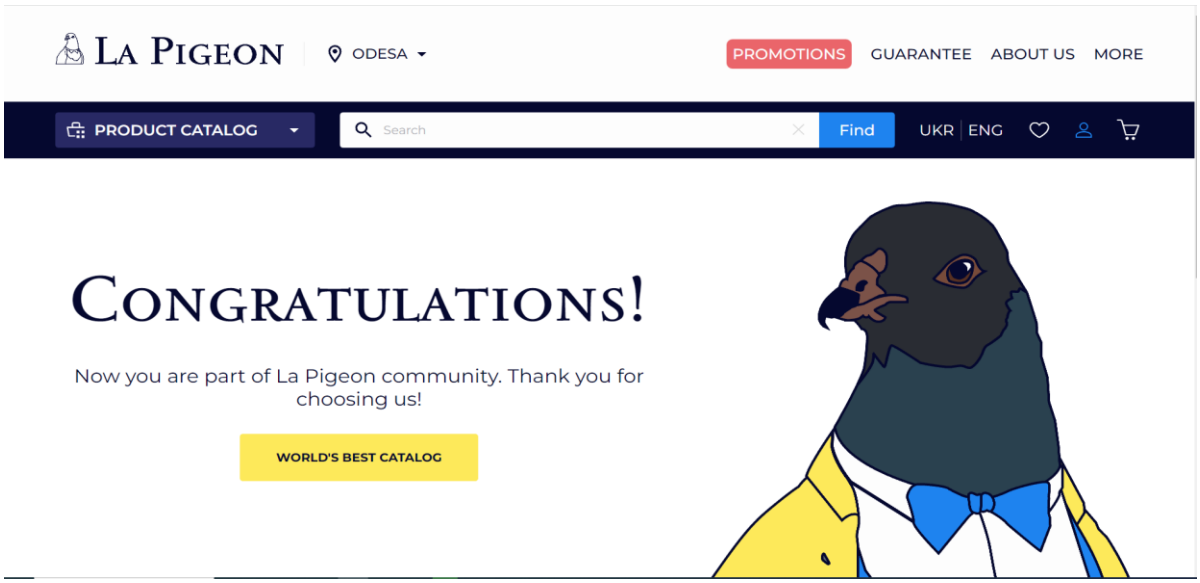


Рисунок 3.11 – Повідомлення про успішно реєстрацію

Оскільки форма реєстрації також передає токен, ніякої необхідності повторно авторизуватись на сайті  немає. Після цих кроків користувач може взаємодіяти з товарами.

При повернені на головну сторінку, додаток, як і раніше, буде відображати популярні продукти, які можна купити, натиснувши на кнопку «Buy» (рис. 3.12).

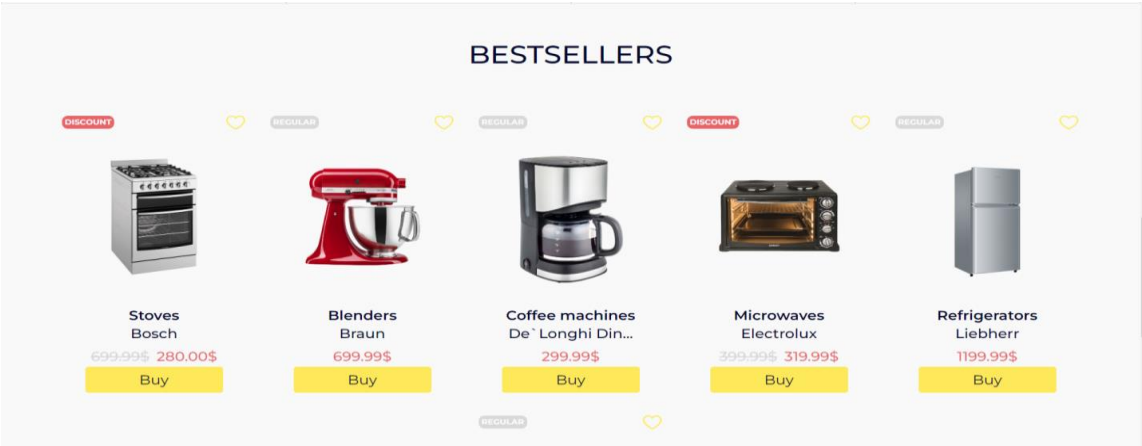


Рисунок 3.12 – Список популярних продуктів

Після того, як ви виберете товар, який вас зацікавив, натиснувши на картинку товару, ви перейдете сторінку цього товару з більш детальним описом, більшою кількістю фотографій та відгуками (рис. 3.13)

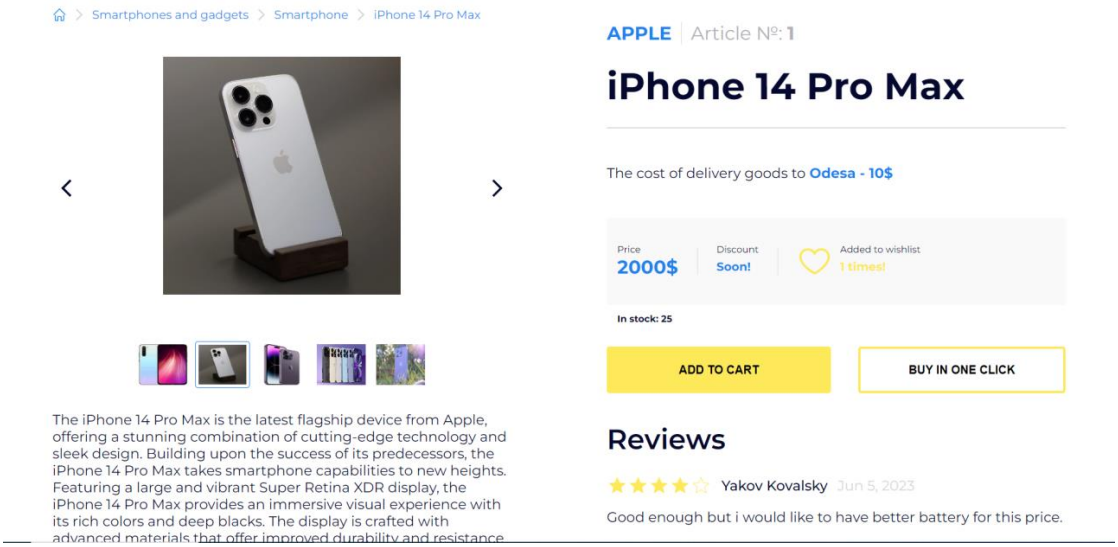


Рисунок 3.13 – Сторінка продукту

У товару є кнопка «ADD TO CART», натиснувши на котру, сервер отримає запит на додання цього продукту в кошик (рис. 3.14).

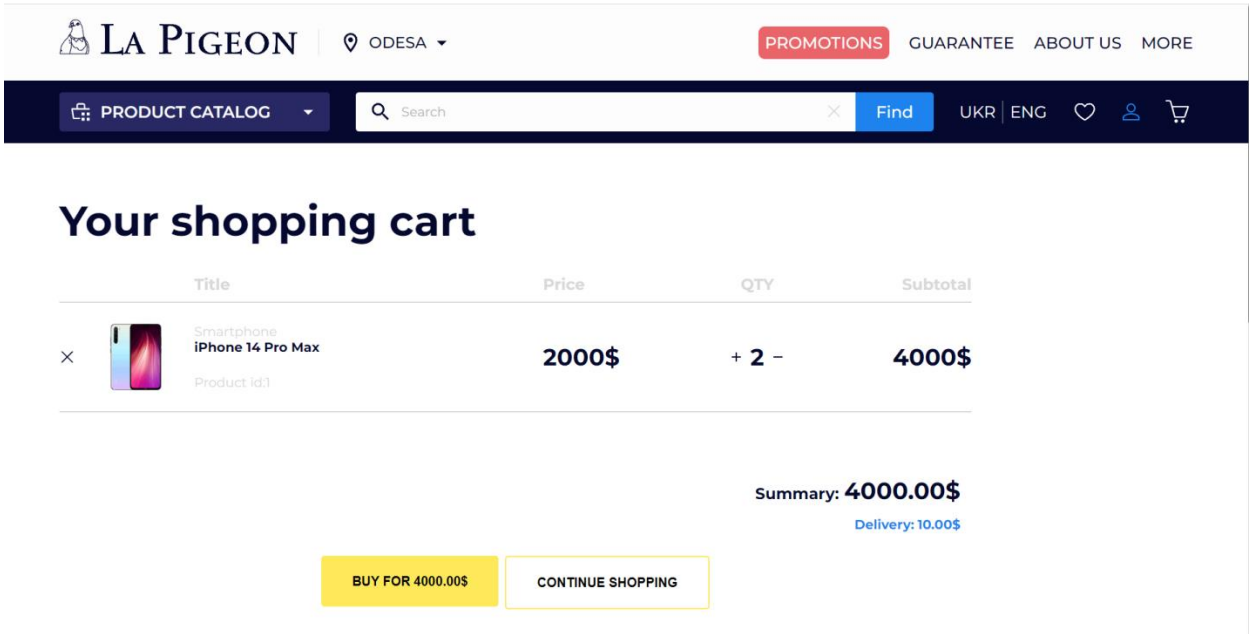


Рисунок 3.14 – Кошик з доданим продуктом

Зі сторінки кошика користувач може зовсім видалити продукт, змінити його кількість або перейти на сторінку розрахунку (рис. 3.15).

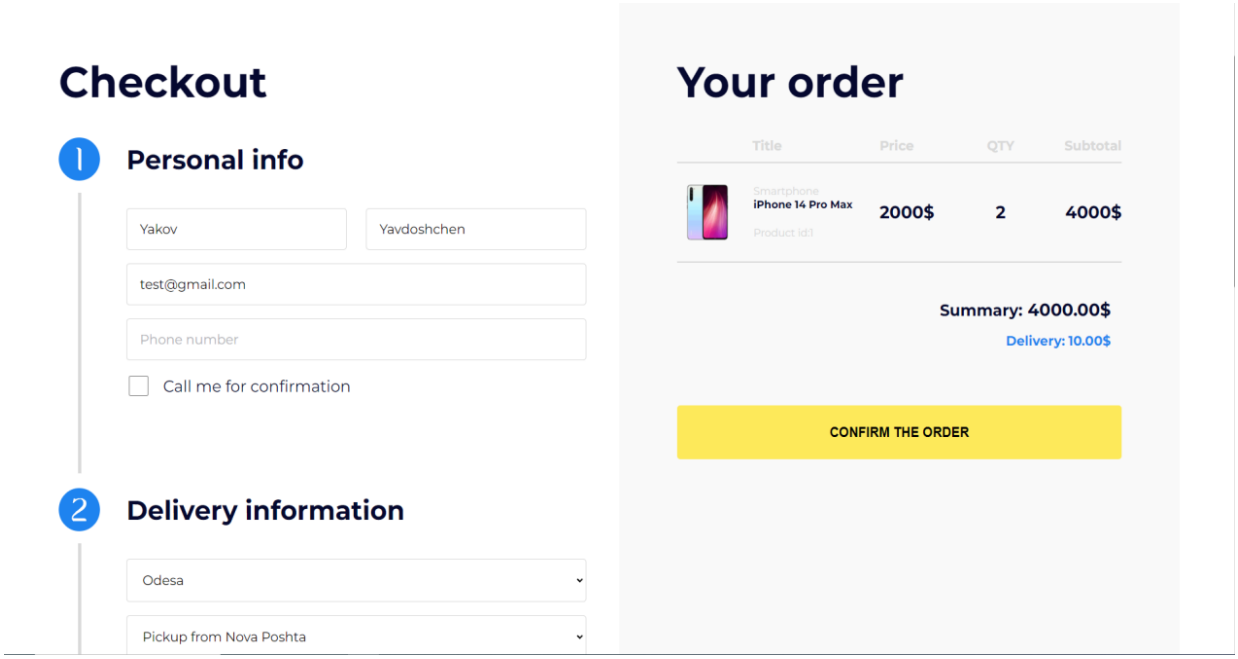


Рисунок 3.15 – Сторінка розрахунку

На сторінці рорахунку додаток підтягує всі дані, які були додані до цього. При побажані їх можна змінити. Вибравши спосіб доставки та відділення, і перевіривши товари і їх кількість ви натискаєте кнопку «CONFIRM THE ORDER», після чого ваше замовлення обробляється, а вас самих перенаправляє на сторінку замовлень (рис. 3.16).

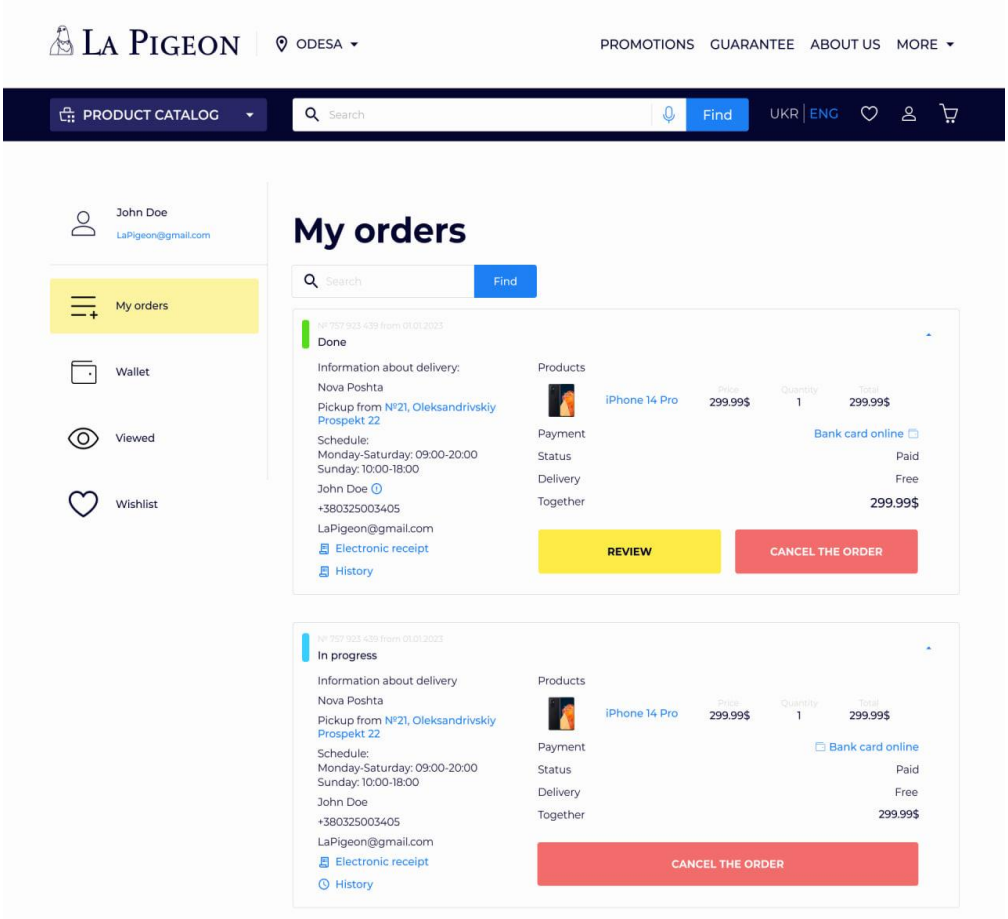


Рисунок 3.16 – Сторінка замовлень

Таким чином, після всіх цих кроків користувач додатку успішно доходить до кінця основного сценарію і досягає своєї основної мети – замовлення товару.

# ВИСНОВКИ

В результаті роботи розроблено інтернет-магазин для продажу побутових товарів. Для створення додатку був використаний React.js та такі його бібліотеки як React-hook-form, Yup, Axios та багато інших.

У відповідності з метою кваліфікаційної роботи був розроблений інтернет-магазин з використанням таких технологій як React.js (для реалізації frontend частини) та Nest.js (для реалізації backend частини).

У відповідності до поставлених задач кваліфікаційної роботи, були виконані наступні етапи створення додатку:

У першому розділі були сформовані вимоги до додатку (функціональні та нефункціональні). Також проведено огляд предметної області та аналіз інструментів розробки.

У другому розділі була спроєктована та побудована структура додатку (побудовані діаграма прецедентів та діаграма послідовностей, створені вайрфрейми ключових сторінок додатку).

У третьому розділі був реалізований та протестований сам інтернет магазин та всі його компоненти, такі як реєстрація, авторизація, додавання товару в кошик, оформлення замовлення, перегляд замовлення та багато іншого.

# ПЕРЕЛІК ПОСИЛАНЬ

1. React Developer Documentation. URL: https://legacy.reactjs.org/docs/getting-started.html (дата звернення: 13.04.2023).

2. Грицунов О. В. Інформаційні системи та технології : навчальний посібник. Харків : ХНАМГ, 2010. 7 с.

3. Харів Н. О. Бази даних та інформаційні системи : навчальний посібник. Рівне : НУВГП, 2018. 8 с.

4. Чемерис Г. Ю. UX/UI дизайн : навчальний посібник. Запоріжжя : ЗНУ, 2021. 49 с.

5. Sunderaraman P. Core JavaScript and JavaScript Frameworks. Practical Ext JS 4. Berkeley, CA, 2013. С. 1–7. URL: https://doi.org/10.1007/978-1-4302-6074-5_1 (дата звернення: 13.04.2023).

6. Yao R., Swift D. R., Perl P. C. React. Js : React. Js Programming, in 8 Hours, for Beginners, Quick Start Guide: React. Js Library Crash Course Tutorial and Exercises. Independently Published. 2021. 145 с.

7. Angular Developer Documentation. URL: https://angular.io/guide/what-is-angular (дата звернення: 21.04.2023).

8. Vue Developer Documentation. URL: https://vuejs.org/guide/introduction.html#the-progressive-framework (дата звернення: 12.05.2023).

9. Figma Documentation. URL: https://www.figma.com/best-practices/guide-to-developer-handoff/components-styles-and-documentation/ (дата звернення: 01.05.2023).

10. Magolan G., Housley P. Nest.js: A progressive Node.js Framework, CA, 2014. 303 с.

11. Коваленко О. С., Дібровська Л. М. Проектування інформаційних систем: загальні питання теорії проектування ІС. Київ : КПІ ім. Ігоря Сікорського, 2020. 8 с.

12. The Use Case Diagram / M. Seidl та ін. UML @ Classroom. Cham, 2015. С. 23–47. URL: https://doi.org/10.1007/978-3-319-12742-2_3 (дата звернення: 13.04.2023).

13. Salgado C. Sketchs, mockups, wireframes y prototipos. Mosaic. 2015. № 130. URL: https://doi.org/10.7238/m.n131.1524 (дата звернення: 13.04.2023).

14. Axios Documentation. URL: https://axios-http.com/docs/intro (дата звернення: 13.06.2023).

15. Firth A. Tools and QA. Practical Web Inclusion and Accessibility. Berkeley, CA, 2019. 470 с.

# ДОДАТОК А

## React – компонент

### А.1 Приклад створення компонента

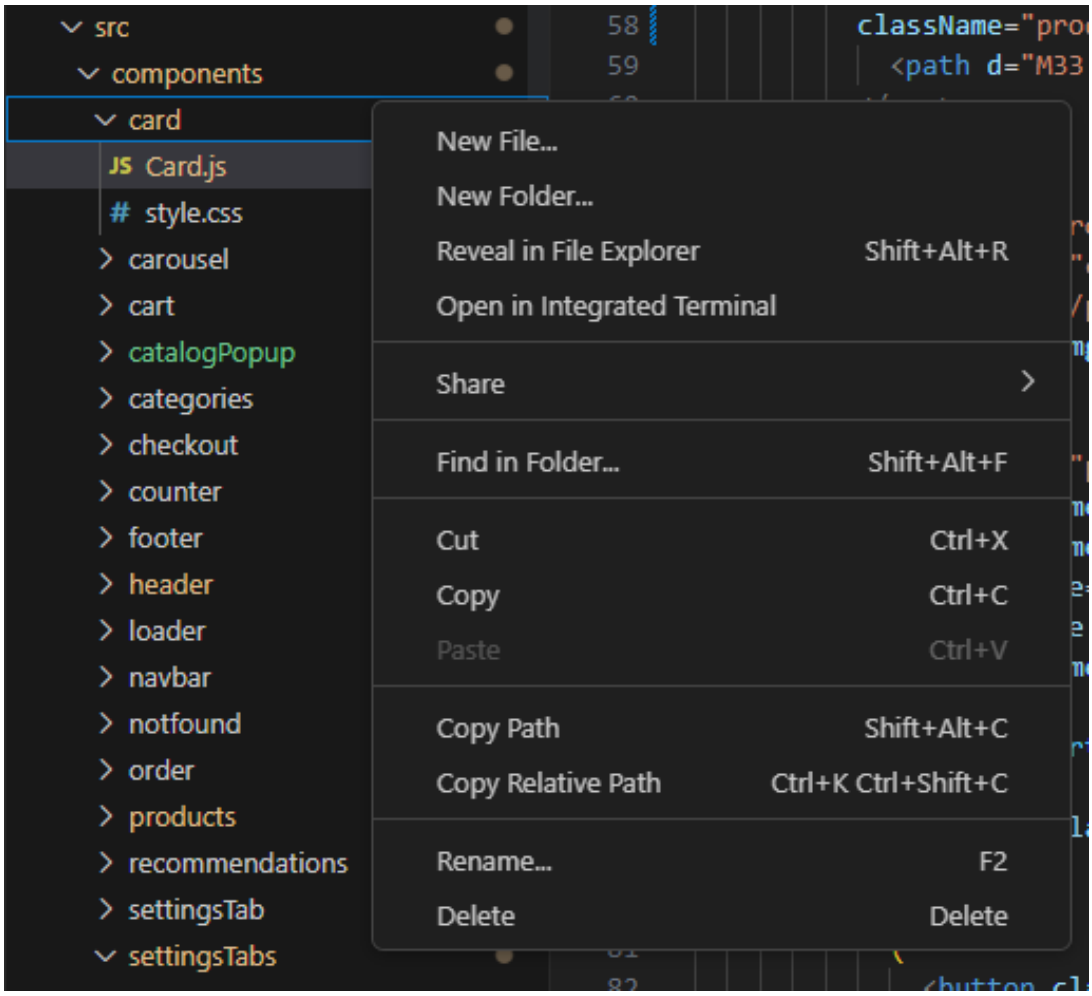


Рисунок А.1 – Створення компонента за допомогою Visual Studio Code

```
1  import './style.css';
2  import React from 'react'
3
4  import { NavLink } from 'react-router-dom';
5
```

Рисунок А.2 – Підключення стилів та бібліотек

```
function App() {

  return (
    <div className="App">
      <TokenProvider>
      <Router>
        <ScrollToTop />
        <Header />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/product" element={<Product />} />
          <Route path="/cart" element={<Cart />} />
          <Route path="/checkout" element={<Checkout />} />
          <Route path="/settings" element={<Settings />}>
            <Route path="account" element={<Account />} />
            <Route path="orders" element={<Orders />} />
            <Route path="wallet" element={<Wallet />} />
            <Route path="viewed" element={<Viewed />} />
            <Route path="wishlist" element={<Wishlist />} />
            <Route path="logout" element={<Logout />} />
          </Route>
          <Route path="/successRegistration" element={<SuccessRegistration />} />
          <Route element={<NotFound />}/>
        </Routes>
        <Footer />
      </Router>
      </TokenProvider>
    </div>
```

Рисунок А.3 – Додавання цього компонента до app.js

## А.2 Приклад компоненту

```
import CatalogPopup from '../catalogPopup/CatalogPopup';
import Navbar from '../navbar/Navbar';
import Login from '../signup/Login';
import Registration from '../signup/Signup';
import React from 'react';

const Header = () => {
  const [loginOpened, setLoginOpened] = React.useState(false);
  const [registrationOpened, setRegistrationOpened] = React.useState(false);
  const [catalogOpened, setCatalogOpened] = React.useState(false);
```

```
  const toggleCatalog = () => {
  setCatalogOpened(!catalogOpened);
};

  return (
    <>
      {loginOpened && (
        <Login
          onCloseLogin={() => setLoginOpened(false)}
          onClickRegistration={() => {
            setRegistrationOpened(!registrationOpened);
            setLoginOpened(false);
          }}
        />
      )}
      <Navbar
        catalogOpened={catalogOpened}
        onClickCatalog={toggleCatalog}
        onClickSignup={() => {
          setRegistrationOpened(!registrationOpened);
          setLoginOpened(false);
        }}
      />
    {catalogOpened && (
      <CatalogPopup
        onCloseCatalog={toggleCatalog}
      />
    )}
      {registrationOpened && (
        <Registration
```

```
          onCloseRegistration={() => setRegistrationOpened(false)}
          onClickAccount={() => {
            setLoginOpened(!loginOpened);
            setRegistrationOpened(false);
          }}
        />
      )}
    </>
  );
};

export default Header;
```

# ДОДАТОК Б

## React – створення картки товару

```
import './style.css';
import React from 'react'
import { NavLink } from 'react-router-dom';

function Card({ name, img, category, price, onAddToCart, onAddToWishlist, discountPrice, onRemoveFromWishlist, id, itsInWishlist }) {

  const [isAddedToCart, setIsAddedToCart] = React.useState(false);
  const [isAddingToCart, setIsAddingToCart] = React.useState(false);
  const [isInWishlist, setIsInWishlist] = React.useState(itsInWishlist);

  const handleAddToCartClick = () => {
    if (!isAddingToCart) {
      setIsAddingToCart(true);
      onAddToCart()
        .then(() => {
          setIsAddedToCart(true);
          setIsAddingToCart(false);
        })
        .catch((error) => {
          console.error(error);
          setIsAddingToCart(false);
        });
    }
  };
```

```
const handleAddToWishlistClick = () => {
  if (!isInWishlist) {
    onAddToWishlist()
      .then(() => {
        setIsInWishlist(true);
      })
      .catch((error) => {
        console.error(error);
      });
  }
};

const handleRemoveFromWishlist = () => {
  if (isInWishlist) {
    onRemoveFromWishlist()
      .then(() => {
        setIsInWishlist(false);
      })
      .catch((error) => {
        console.error(error);
      });
  }
};


return (
  <li className="products__card">
    <div className="products__card-top">
                        {discountPrice ? (<span className="products-
card__status">Discount</span>) : (<span className="products-card__status--
```

```
regular">Regular</span>)}
      {isInWishlist ?
      (
         <svg onClick={handleRemoveFromWishlist}
            className="products__card-like" width="36" height="36" viewBox="0
0 36 36" fill="none" xmlns="http://www.w3.org/2000/svg">
               <path d="M33.75 13.2188C33.75 23.0625 19.1545 31.0303 18.533
31.3594C18.3691 31.4475 18.186 31.4936 18 31.4936C17.814 31.4936 17.6309
31.4475 17.467 31.3594C16.8455 31.0303 2.25 23.0625 2.25 13.2188C2.25261
10.9072 3.17202 8.69106 4.80654 7.05654C6.44106 5.42202 8.65719 4.50261
10.9688 4.5C13.8727 4.5 16.4152 5.74875 18 7.85953C19.5848 5.74875 22.1273
4.5 25.0312 4.5C27.3428 4.50261 29.5589 5.42202 31.1935 7.05654C32.828
8.69106 33.7474 10.9072 33.75 13.2188Z" fill="#FDEB46"/>
         </svg>
      ) : (
        <svg onClick={handleAddToWishlistClick}
        className="products__card-like"
        width="36"
        height="36"
        viewBox="0 0 24 24"
        fill="none"
        xmlns="http://www.w3.org/2000/svg"
      >
        <path
           d="M16.6875 3C14.7516 3 13.0566 3.8325 12 5.23969C10.9434 3.8325
9.24844 3 7.3125 3C5.77146 3.00174 4.29404 3.61468 3.20436 4.70436C2.11468
5.79404 1.50174 7.27146 1.5 8.8125C1.5 15.375 11.2303 20.6869 11.6447
20.9062C11.7539 20.965 11.876 20.9958 12 20.9958C12.124 20.9958 12.2461
20.965 12.3553 20.9062C12.7697 20.6869 22.5 15.375 22.5 8.8125C22.4983
7.27146 21.8853 5.79404 20.7956 4.70436C19.706 3.61468 18.2285 3.00174
```

```
16.6875 3ZM12 19.3875C10.2881 18.39 3 13.8459 3 8.8125C3.00149 7.66921
3.45632 6.57317 4.26475 5.76475C5.07317 4.95632 6.16921 4.50149 7.3125
4.5C9.13594 4.5 10.6669 5.47125 11.3062 7.03125C11.3628 7.16881 11.4589
7.28646 11.5824 7.36926C11.7059 7.45207 11.8513 7.49627 12 7.49627C12.1487
7.49627 12.2941 7.45207 12.4176 7.36926C12.5411 7.28646 12.6372 7.16881
12.6937 7.03125C13.3331 5.46844 14.8641 4.5 16.6875 4.5C17.8308 4.50149
18.9268 4.95632 19.7353 5.76475C20.5437 6.57317 20.9985 7.66921 21
8.8125C21 13.8384 13.71 18.3891 12 19.3875Z"
          fill="#FDEB46"
        />
       </svg>
      )}
     </div>
     <div className="product__card-content">
      <div className="card__image-container">
       <NavLink to={`/product/${id}`} className="card__image-link">
        <img src={img} alt={name} className="products__card-image" />
       </NavLink>
      </div>
      <div className="products__card-bottom">
       <span className="products__category-title">{category}</span>
       <span className="product__title">{name}</span>
       <div className="product__prices">
                    {discountPrice && <span className="product__price-
old">{price.toFixed(2)}$</span>}
                      <span className="product__price">{(discountPrice ||
price).toFixed(2)}$</span>
       </div>
       {!isAddedToCart ?
        (
```

```
                                        <button        className="product__button"
onClick={handleAddToCartClick}>Buy</button>
          )
          :
          (
                <button  className="product__button"  style={{  backgroundColor:
'#57DC19', color: "#FDFDFD"}}>Thank you!</button>
          )
        }
       </div>
     </div>
   </li>
  );
};

export default Card;
```

# ДОДАТОК В

## React – валідація

### В.1 Приклад створення компоненту з валідацією

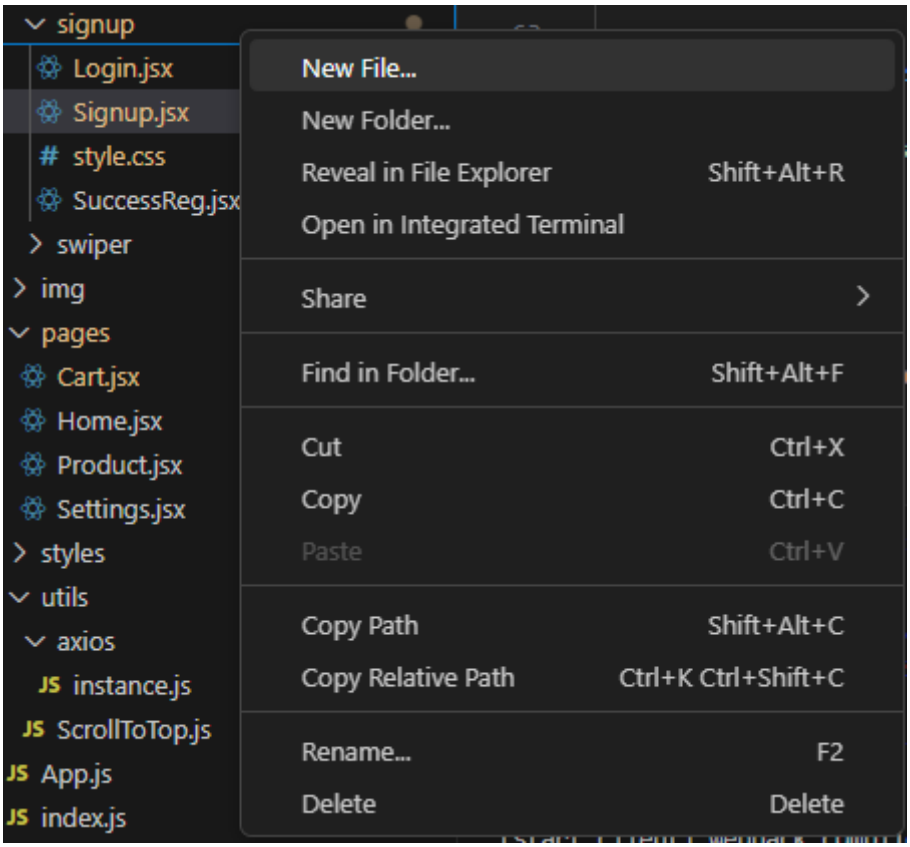


Рисунок В.1 – Створення компоненту валідації за допомогою Visual Studio Code

### В.2 Приклад моделі

```
import "./style.css"
import { useState } from "react";
import createAxiosInstance from "../../../utils/axios/instance";
import React from "react";
import { useForm } from "react-hook-form";
```

```
import * as yup from "yup";
import { yupResolver } from "@hookform/resolvers/yup";

const schema = yup.object().shape({
   firstName: yup.string(),
   lastName: yup.string(),
   phone: yup.string(),
   email: yup.string()
 });

const Account = () => {



   const instance = createAxiosInstance();
   const [isOpen, setIsOpen] = useState([false, false]);

   const togglePanel = (index) => {
        setIsOpen(prevState => prevState.map((open, i) => i === index ? !open : open));
   };

   const { register, handleSubmit } = useForm({
      resolver: yupResolver(schema),
    });

    const onSubmit = async (data) => {
      try {
        const response = await instance.patch("/account/userInfo", data);
        console.log(response.data);
      } catch (error) {
```

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



  return (
    <div>
      <section className="account__settings">
        <h1 className="account__title">
          Account
        </h1>
        <ul className="account__tabs">
          <li className="account__tab">
                    <div  className="account__tab-top"  onClick={() =>
togglePanel(0)}>
                  <h2 className="account__tab-title">
                    Personal info
                  </h2>
                      <svg  className="account__tab-button--open"  width="48"
height="48"        viewBox="0        0        48        48"        fill="none"
xmlns="http://www.w3.org/2000/svg">
                  {isOpen [0] ? (
                    <path d="M14 28L24 18L34 28H14Z" fill="#1C7FF3"/>
                    ) : (
                    <path d="M14 20L24 30L34 20H14Z" fill="#050630"/>
                    )}
                  </svg>
                </div>
                {isOpen [0] && (
```

```
                <div className="account__tab-content">
                        <form onSubmit={handleSubmit(onSubmit)} action="" className="account__content">
                    <div className="account__tab-inputs">
                       <input id="firstName" name="firstName" placeholder="First name" type="text" className="account__tab-input" {...register('firstName')}/>
                        <input id="lastName" name="lastName" placeholder="Last name" type="text" className="account__tab-input" {...register('lastName')}/>
                    </div>
                    <button type="submit" className="account__submit-button">
                       SAVE
                    </button>
                  </form>
                </div>
                )}
              </li>
              <li className="account__tab">
                        <div className="account__tab-top" onClick={() => togglePanel(1)}>
                    <h2 className="account__tab-title">
                       Contacts
                    </h2>
                        <svg className="account__tab-button--open" width="48" height="48" viewBox="0 0 48 48" fill="none" xmlns="http://www.w3.org/2000/svg">
                    {isOpen [1] ? (
                       <path d="M14 28L24 18L34 28H14Z" fill="#1C7FF3"/>
                       ) : (
                       <path d="M14 20L24 30L34 20H14Z" fill="#050630"/>
                       )}
```

```
                    </svg>
                </div>
                {isOpen[1] && (
                <div className="account__tab-content">
                        <form onSubmit={handleSubmit(onSubmit)} action="" 
className="account__content">
                    <div className="account__tab-inputs">
                            <input placeholder="Phone number" type="text" 
className="account__tab-input" {...register('phone')}/>
                                <input placeholder="Email" type="email" 
className="account__tab-input" {...register('email')}/>
                    </div>
                    <button type="submit" className="account__submit-button">
                        SAVE
                    </button>
                </form>
                </div>
                )}
            </li>
        </ul>
    </section>
  </div>
 );
}
export default Account
```