

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Розробка системи для продажу товарів для тварин»

Виконав студент групи ІПЗ-19-1
НЕЧУХРАНИЙ Олег Ігорович

Керівник роботи:
МАРЧУК Галина Вікторівна

Рецензент:
САВІЦЬКИЙ Роман Святославович

Житомир – 2023

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **НЕЧУХРАНИЙ Олег Ігорович**

Керівник роботи: **МАРЧУК Галина Вікторівна**

Тема роботи: **«Розробка системи для продажу товарів для тварин»,**
затверджена Наказом закладу вищої освіти від **«14» лютого 2023 р., №70с**

Вихідні дані для роботи: вивчення технічних можливостей розробки системи для продажу товарів для тварин а також розробка архітектури та функціональності системи.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Марчук Г.В.	20.02.2023р.	20.02.2023р.
2	Марчук Г.В.	20.04.2023р.	20.04.2023р.
3	Марчук Г.В.	16.05.2023р.	16.05.2023р.

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з програмного комплексу системи для продажу товарів для тварин та пояснювальної записки. Пояснювальна записка до випускної кваліфікаційної роботи містить 67 сторінки, 60 ілюстрацій та 10 сторінок додатків.

Метою роботи є розробка системи для продажу товарів для тварин а також для ведення статистики проданих товарів та рекомендації для щомісячної закупівлі проданого товару.

В роботі визначені головні завдання для розробки системи, проаналізовано методи вирішення поставленої задачі, обґрунтовано вибір засобів розробки backend та frontend складових готового вебдодатку. Наведено алгоритм роботи готового продукту, взаємодію основних компонентів а також архітектуру вебдодатку.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, FRONTEND, BACKEND, СИСТЕМА ДЛЯ ПРОДАЖУ ТОВАРІВ ДЛЯ ТВАРИН

					ІПЗ.КР.Б-121-23-ІПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Нечухраний О.І.			Система для продажу товарів для тварин. Пояснювальна записка		Літ.	Арк.
Керівник		Марчук. Г. В.						2
							Житомирська політехніка, група ІПЗ-19-1	
н. контр.		Тичина О.П.						
Зав. каф.		Морозов А.В.						

ABSTRACT

Bachelor's graduate work consists of software package a software complex of a system for the sale of animal products and a documentation. Documentation for bachelor's graduate work contain 67 pages, 60 illustrations and 10 additional pages.

The purpose of graduation work is to develop a system for the sale of products for animals, as well as for keeping statistics of sold products and recommendations for the monthly purchase of sold products.

Graduation work contains the main tasks for the development of the system, analyzes the methods of solving the given task, substantiates the choice of tools for the development of backend and frontend components of the finished web application. The algorithm of the finished product, the interaction of the main components, and the architecture of the web application are given.

KEYWORDS: WEB APPLICATION, FRONTEND, BACKEND, SYSTEM FOR SALE OF PET PRODUCTS.

					ИП3.КР.Б-121-23-ИЗ	
						3

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	5
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ РОЗРОБКИ СИСТЕМИ ПРОДАЖУ ТОВАРІВ ДЛЯ ТВАРИН	7
1.1 Постановка задачі	7
1.2 Аналіз аналогів програмного продукту	8
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	13
Висновки до першого розділу	14
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ ПРОДАЖУ ТОВАРІВ ДЛЯ ТВАРИН	15
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	15
2.2 Розробка бази даних системи	18
2.3 Проектування та реалізація алгоритмів роботи системи	20
2.4 Реалізація функціональних можливостей системи продажу зоотоварів	28
Висновки до другого розділу	30
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМИ ДЛЯ ПРОДАЖУ ТОВАРІВ ДЛЯ ТВАРИН.....	31
3.1 Порядок встановлення та налаштування параметрів системи	31
3.2 Структура інтерфейсу системи для продажу товарів для тварин	32
3.3 Тестування роботи програмного комплексу	45
Висновки до третього розділу	52
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	54
ДОДАТКИ.....	56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

FE – FrontEnd

BE – BackEnd

ПЗ – програмне забезпечення

JS – JavaScript

CRUD – create, read, update, delete

CRD – create, read, delete

CRU – create, read, update

API – Application Programming Interface

FAQ - Frequently Asked Question(s)

					ІПЗ.КР.Б-121-23-ПЗ	
						5

ВСТУП

Актуальність теми. На сьогоднішній день інтернет-магазини є невід'ємною складовою в житті майже кожної людини, будь то вебдодаток ресторану де користувач може замовити собі доставку їжі чи вебмагазин для продажу будь-якого товару фактично без винятків. Один з видів таких магазинів є зоомагазини, де господар може замовити товари для свого улюбленця.

З боку ж власника такого магазину буде дуже зручно мати систему яка контролює статистику по кількості проданих товарів.

Метою кваліфікаційної роботи є розробка системи для продажу товарів для тварин.

Встановлена мета обумовлює наступні завдання:

- Ознайомитись з вже існуючими зоомагазинами.
- Ознайомити з методами розробки аналогічного програмного забезпечення.
- Скласти план по розробці програмного продукту.
- Обрати технології для розробки програмного продукту.
- Розробити вебдодаток системи для продажу товарів для тварин.

Об'єктом дослідження є технології розробки системи для продажу товарів для тварин.

Предметом дослідження є розробка архітектури та функціональності системи. Визначення основних модулів, їхньої взаємодії, розробка алгоритмів та процесів, які забезпечують роботу системи

За темою випускної кваліфікаційної роботи було опубліковано тези: Нечухраний О.І., Марчук Г.В. Моделювання роботи он-лайн магазину. XIII Міжнародна науково-технічна конференція. Інформаційні комп'ютерні технології - 2023. 30-31 березня 2023 р. м. Житомир.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ РОЗРОБКИ СИСТЕМИ ПРОДАЖУ ТОВАРІВ ДЛЯ ТВАРИН

1.1 Постановка задачі

Задачею кваліфікаційної роботи є створення системи для продажу товарів для тварин, за допомогою якої з боку покупця буде можливим огляд наявних товарів, взаємодія з ними та їх замовлення з подальшою оплатою при отриманні а також з боку власника буде можливість відстежувати кількість проданих товарів.

Готовий вебдодаток повинен мати наступні функції:

Для покупця:

- перегляд товарів;
- пошук товару;
- фільтрація товарів;
- сортування товарів;
- додання товару в кошик;
- оформлення замовлення;
- можливість створення власного акаунту;
- Вхід/Вихід з акаунту;
- перегляд останніх замовлень.

Для власника:

- наявні всі можливості звичайного користувача
- перегляд статистики покупки товарів.

Для адміністратора:

- наявні всі можливості користувача;
- CRD по товарах;
- додавати користувачів з різними рівнями доступу;
- оновлення статусу замовлення з автоматичним відсиленням електронного письма користувачу;

1.2 Аналіз аналогів програмного продукту

На сьогоднішній день в мережі Інтернет представлено велику кількість інтернет-магазинів для домашніх тварин. Ці магазини можуть мати зовсім різний список товарів, що продають, наприклад магазини які продають лише корма, магазини які продають лише нехарчові товари або ж магазини які поєднують всі товари у собі, також існують магазини від виробника товарів, в яких наведено лише товари даної фірми.

В процесі виконання кваліфікаційної роботи було розглянуто декілька прикладів:

1) Інтернет-магазин «E-ZOO» (рис 1.1).

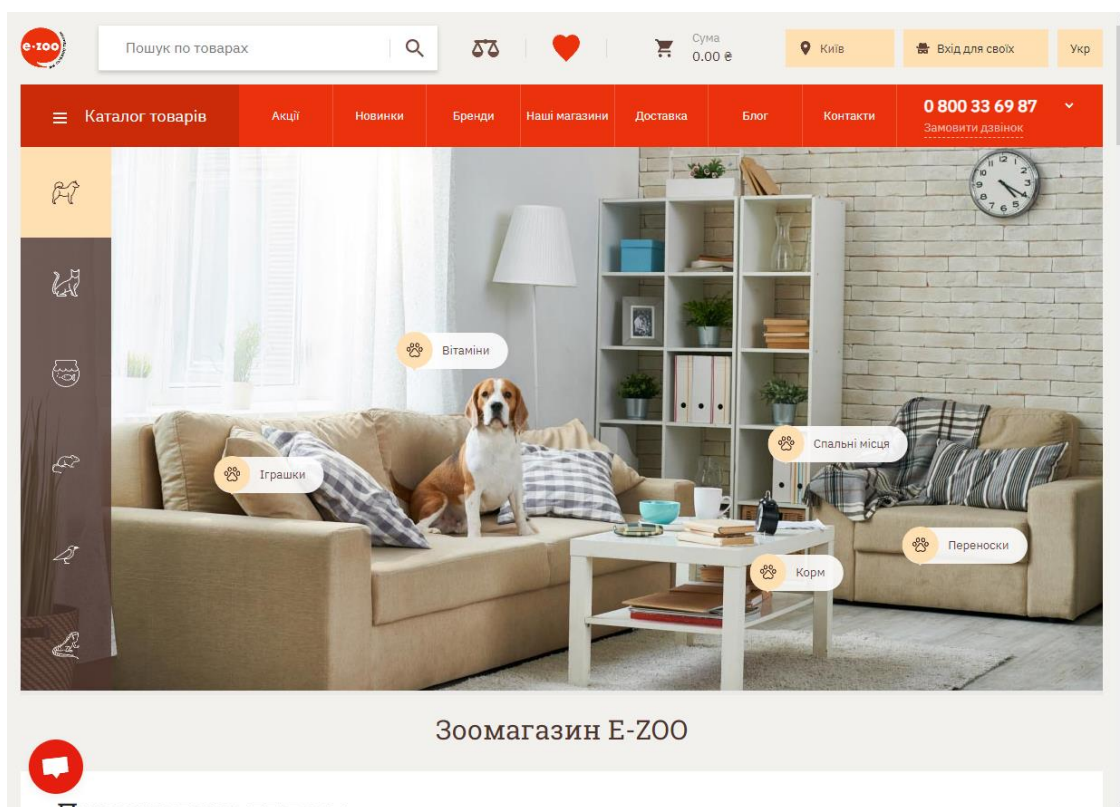


Рис. 1.1. Інтерфейс «E-ZOO» [1]

Позитивні елементи:

- цікавий та зручний інтерфейс;
- широкий вибір товарів;
- інтерактивне меню для вибору товару для улюбленця;
- зручне оформлення доставки товару;
- можливість порівняти товари.

Недоліки:

- нагромадження зайвої інформації;
- винесення інформацію з FAQ на головну сторінку;
- нав'язливе вікно «Хочете ми вам зателефонуємо?».

2) Інтернет-магазин «MasterZoo» (рис.1.2)

Позитивні елементи:

- зручна фільтрація товарів;
- зручне оформлення замовлення;
- можливість порівняти товари.

Недоліки:

- поганий вибір шрифту для оформлення вебдодатку;
- неприємне зору поєднання великої кількості кольорів.

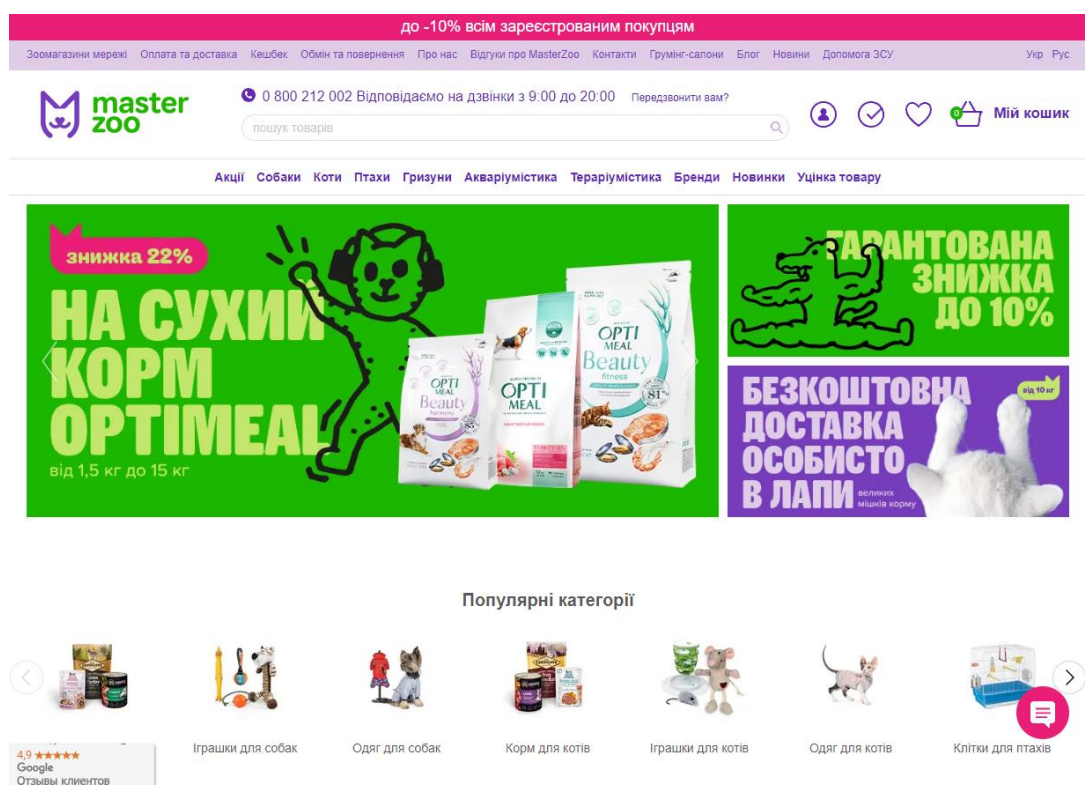


Рис. 1.2 Інтерфейс «MasterZoo» [2]

3) Інтернет-магазин «Zooshop.in.ua» (рис 1.3)

Позитивні елементи:

- приємне оформлення сторінок;
- зручне оформлення замовлення.

Недоліки:

- перемішана локалізація українською та російською мовами;
- обмежена фільтрація товарів.
- наявні елементи тексту які неможливо прочитати через фоновий колір

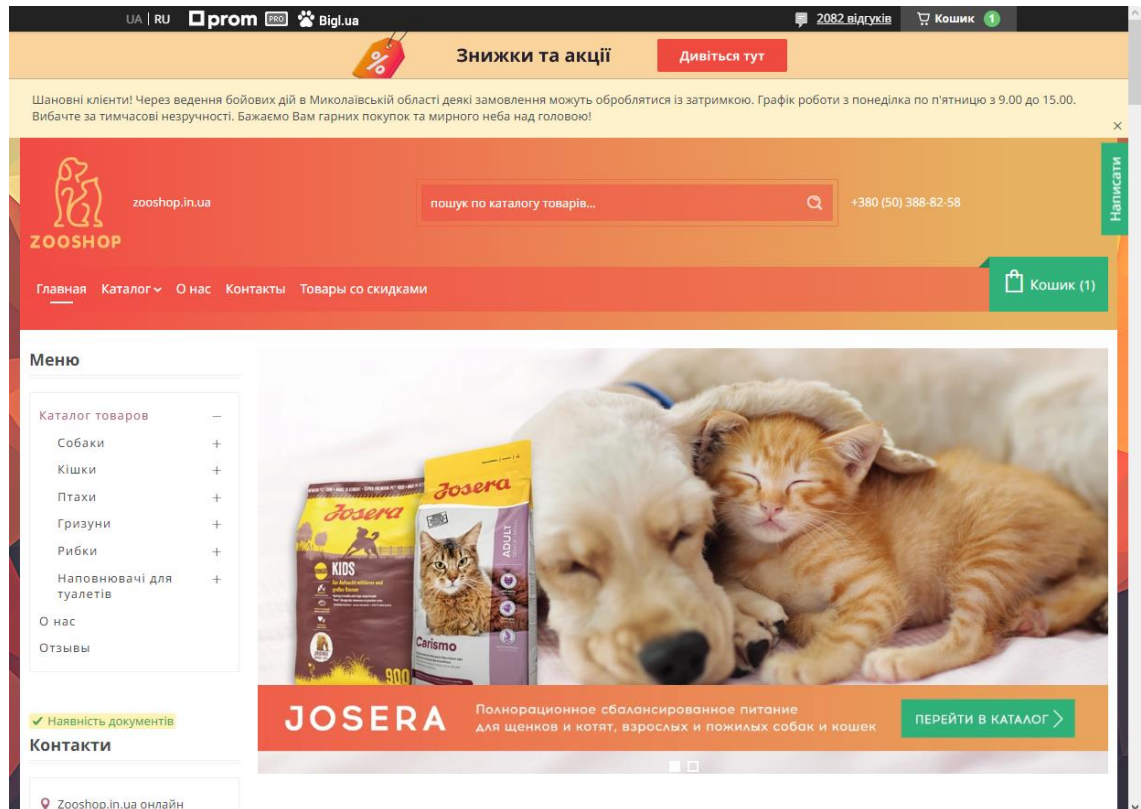


Рис. 1.3 Інтерфейс «Zooshop.in.ua» [3]

4) Интернет-магазин «Гавкер М'явський» (рис 1.4)

Недоліки:

- не виявлено.

Переваги:

- зручний та лаконічний інтерфейс;
- комфортне оформлення замовлення.

Гавкер
М'явський

Найкращі ціни на
найкращі товари
для тварин

Оберіть товар для
вашого улюбленця



Онлайн оплата



Безкоштовна доставка кур'єром*



Система знижок



Гарантія якості

Хіти продажу

ХІТ



Артикул: 45404

СУПЕРІУМ Панація, протипаразитарні
таблетки для котів 0,5-2 кг

138 грн
В наявності

ХІТ



Артикул: lckymatorange

Lucky Mat - Килимок-годівець
силиконовий для повільного годуван...

200 грн
В наявності

ХІТ



Артикул: K9K9PSC47201

Беззернові ласощі для дорослих собак
всіх порід з білим окрасом шерсті...

167 грн
В наявності

ХІТ



Артикул: TX-23473

Пакети Тріхіе для прибирання за
собаками з ароматом лимона, 4 руло...

64 грн
В наявності

Рис. 1.4 Інтерфейс «Гавкер М'явський» [4]

5) Інтернет-магазин «Pethouse» (рис. 1.5)

Недоліки:

- невдале оформлення посилань на навігаційній панелі

Переваги:

- зручне сортування товарів;
- приємний ненав'язливий дизайн;
- секція з акційними товарами;
- секція з корисними порадами щодо товарів.

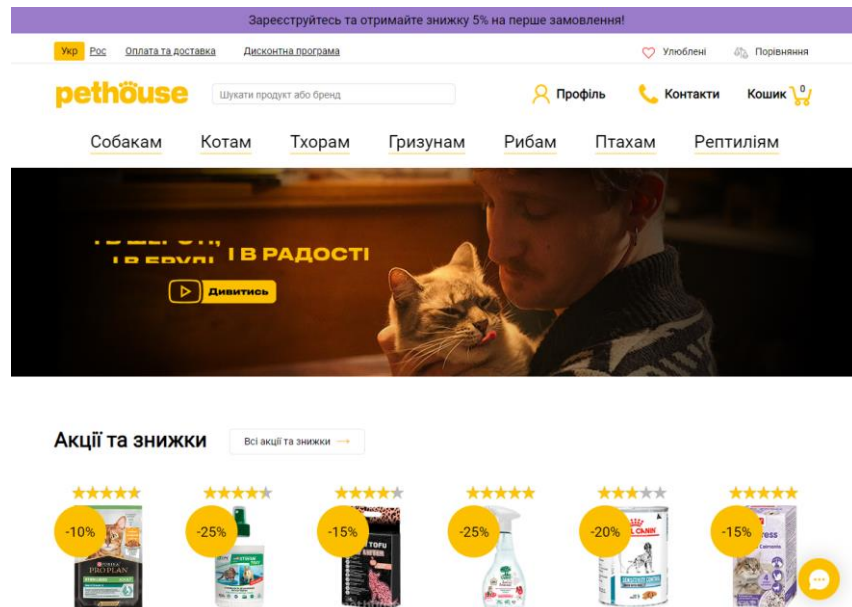


Рис. 1.5 Інтерфейс «Pethouse» [5]

Кожен з переглянутих прикладів має свої переваги та недоліки які впливають на думку користувача щодо цих платформ. Також з них було набуто ідей щодо функціоналу готового вебдодатку, а саме:

- розробити сторінку з відображенням всіх товарів;
- розробити фільтрацію товарів;
- розробити сортування товарів;
- розробити пошук товарів за будь-яким параметром;
- розробити сторінку деталей товару;
- розробити сторінку кошику;
- розробити сторінку оформлення замовлення;
- розробити сторінку перегляду замовлень користувача;
- розробити різнорівневе розмежування доступу до можливостей вебдодатку;
- розробити реєстрацію користувачів;
- розробити авторизацію користувачів;
- розробити сторінку статистики;
- розробити сторінку додавання товару;
- розробити сторінку перегляду всіх замовлень
- розробити можливість оновлювати статус замовлення.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Існує велика кількість мов програмування які можуть допомогти в розробці вебдодатку, наприклад: Python [6], PHP [7], JavaScript [8] та багато інших.

Так само існує багато баз даних, в яких можна буде зберігати всю інформацію.

Для розробки вебдодатку було обрано наступний стек технологій:

- не реляційна БД MongoDB Atlas [9, 10];
- середовище виконання Node.JS, разом із фреймворком Express [11, 12];
- фреймворк Next JS [13, 14, 15];
- контейнер контролю стану Redux [16, 17];
- Cloudinary для контролю медіа-ресурсів [18].
- EmailJs [19]

Отже, MongoDB була обрана через високу доступність та надійність, масштабованість, безпеку та простоту використання.

Node.js – через швидкість, асинхронність, масштабованість, багату бібліотеку модулів.

Next Js – через легке використання, вбудовану маршрутизацію, підтримку статичної генерації, широкі можливості і звісно ж Server-side rendering, який покращує швидкість завантаження та поліпшує індексацію пошуковими системами.

Cloudinary була обрана через легкість підключення до хмарного сховища з метою збереження зображень.

EmailJS – це сервіс який дозволяє відправляти електронні листи без налаштування власного сервера. Він надає зручний та легкий спосіб взаємодії з поштовими адресами.

Також для розробки специфічних можливостей вебдодатку використовуються різні бібліотеки, наприклад, «react-chartjs-2» [20] яка є React-обгорткою для бібліотеки візуалізації даних Chart.js та значно полегшує використання її функцій. Дана бібліотека містить у собі компоненти завдяки яким можна створювати різні типи діаграм.

Для розміщення на сервері необхідні наступні мінімальні апаратні вимоги:

- Процесор: Intel Core i3 або аналогічний.
- Оперативна пам'ять: 2 ГБ.
- Жорсткий диск: 50 ГБ вільного місця на жорсткому диску.
- Мережеве підключення: хороша швидкість Інтернету.

Для того, щоб користувач мав змогу користатися вебдодатком необхідні наступні характеристики:

- Процесор: 1 ГГц або швидше.
- Оперативна пам'ять: 1 ГБ або більше.
- Операційна система: Windows 7 або вище, Mac OS X 10.9 або вище, iOS 12 або вище, Android 6.0 або вище, Linux.

Отже, все вище наведене - це мінімально необхідні вимоги для інструментальних засобів та апаратного забезпечення, для подальшої розробки та розгортання системи продажу товарів для тварин. Вони не є занадто великими, на сьогоднішній день кожен має у кишені пристрій, що може спокійно запускати вебдодаток, що розробляється.

Висновки до першого розділу

Отже, в першому розділі було проаналізовано потреби та поставлено задачу до кваліфікаційної роботи бакалавра. Переглянуто проекти по схожій тематиці та визначено їх основні переваги та недоліки.

Обрано та обґрунтовано технології для розробки вебдодатку та описано мінімальні вимоги щодо апаратного забезпечення.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ ПРОДАЖУ ТОВАРІВ ДЛЯ ТВАРИН

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Для опису проекту було розроблено чотири діаграми UML [21], а саме «Діаграма варіантів використання» (рис.2.1), «Діаграма активності» (рис.2.2), «Діаграма компонентів» (рис.2.3).

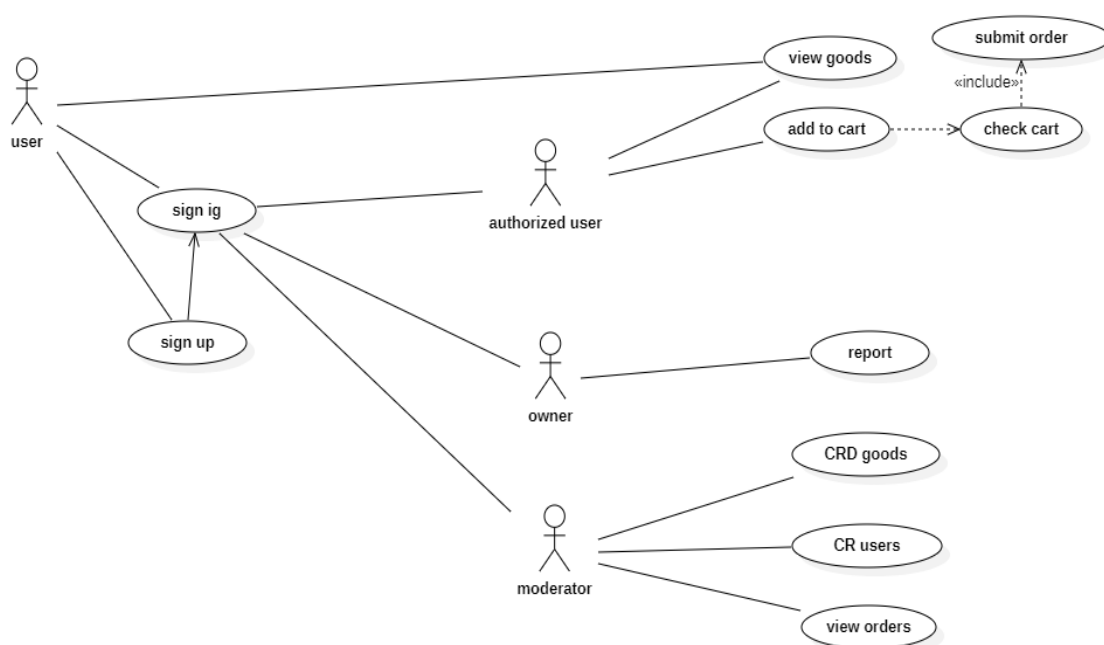


Рис. 2.1. Діаграма варіантів використання

На діаграмі варіантів використання (див. рис.2.1) зображено класичний варіант вебдодатку для продажу товарів. Починається все з неавторизованого користувача, який має всі ті ж можливості, які наявні в авторизованого звичайного користувача, окрім того, що при оформленні замовлення йому необхідно буде ввести контактну електронну адресу, щоб отримувати інформацію про статус замовлення. В процесі авторизації користувач розподіляється на три групи, згідно рівня доступу, а саме:

- власник, який після авторизації все ще має можливості звичайного користувача, але у верхній частині сайту з'являється можливість відвідати

сторінку зі звітністю про кількість замовлень у різних статусах а також про кількість куплених товарів по категоріях та типах товару;

- користувач, якого повертає на сторінку з товарами але йому не потрібно вводити свій адрес електронної пошти при оформленні замовлення;
- користувача, який є адміністратором і який може переглядати замовлення інших користувачів, додавати або видаляти наявні товари, переглядати замовлення та змінювати їх статус а також створювати нових користувачів з можливістю змінити їх рівень доступу.

Діаграма активності (рис.2.2) поділена на три частини – клієнтську, частину адміністратора та власника. Перевірка який саме користувач авторизувався відбуватиметься згідно з правами доступу наданими на акаунт. Кожна з них має свої можливості:

1) Клієнтська частина:

- перегляд товарів які представлені на сайті магазину;
- фільтрація та сортування товарів за певними критеріями;
- додавати обрані товари в кошик;
- переглядати та редагувати зміст кошика;
- оформлення замовлення (у випадку коли користувач авторизований в нього не буде відображатися поле для вводу адреси електронної пошти).

2) Адміністраторська частина матиме в собі наступні можливості:

- всі можливості надані користувачу;
- видалення товарів;
- додавання нових товарів;
- додавання нового користувача з різними рівнями доступу;
- оновлення замовлень (оновлення статусу виконання замовлення).

3) Частина власника: при авторизації власника в нього є можливість відвідати сторінку зі звітністю про продажі.

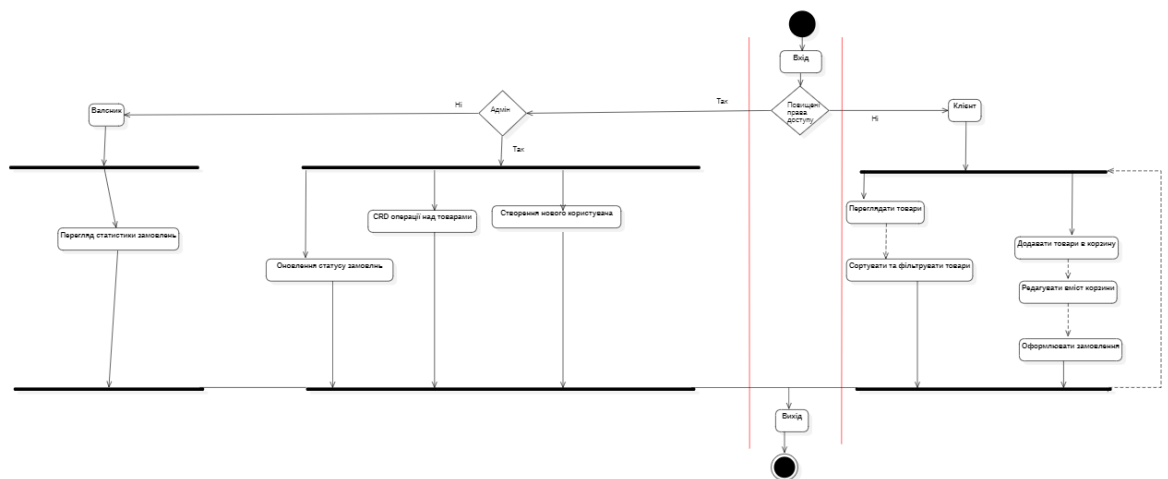


Рис. 2.2. Діаграма активності

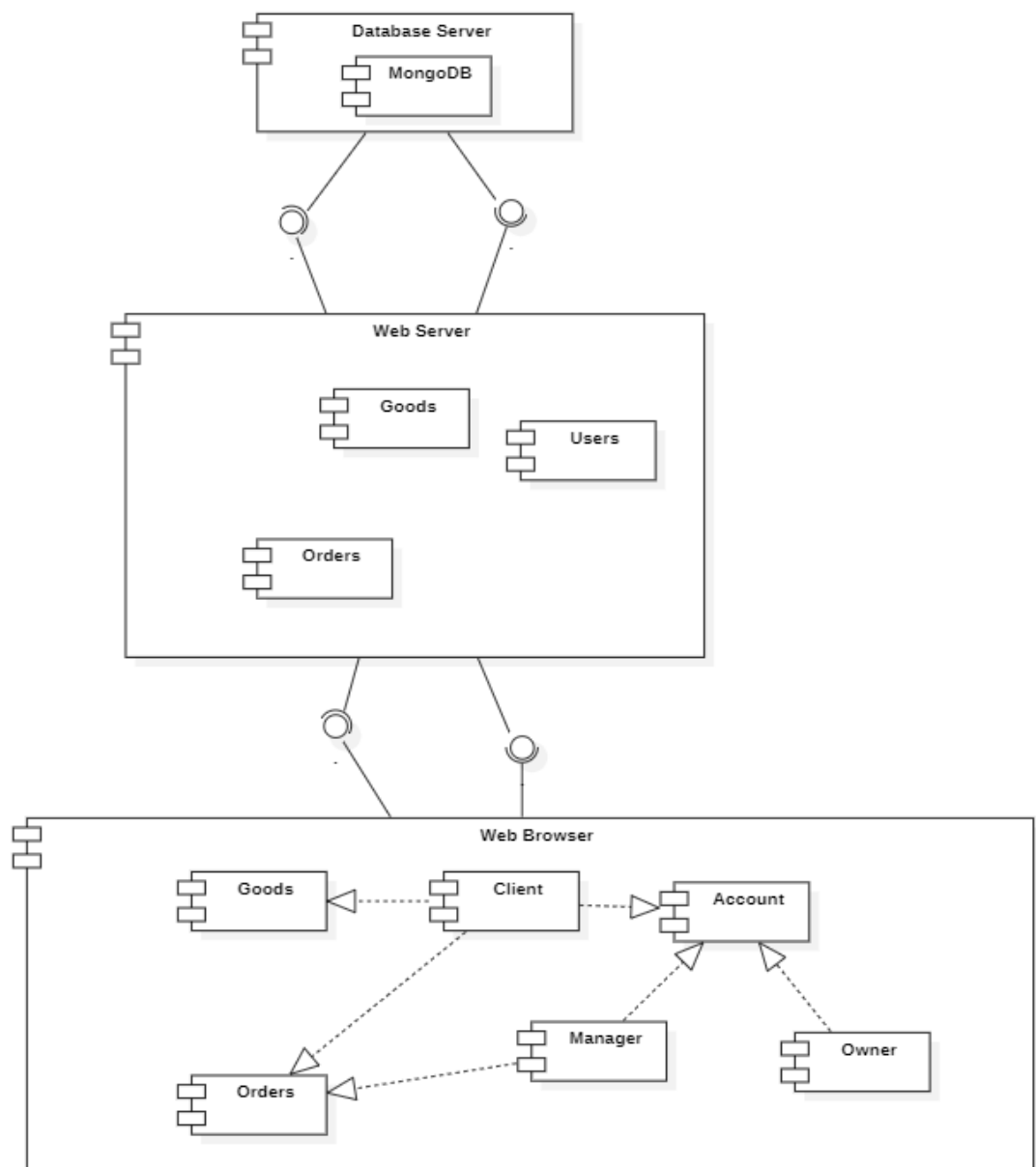


Рис. 2.3. Діаграма компонентів

На діаграмі компонентів (див. рис.2.3) зображено три основних компоненти, а саме сервер бази даних, вебсервер та веббраузер. Вебсервер на діаграмі грає роль посередника, який збирає інформацію з бази даних та передає її до веббраузеру, який в свою чергу вже надає можливість переглядати товари, оформлювати замовлення та переглядати список замовлень в акаунті користувача, також адміністратор сайту, за потреби, може отримувати інформацію про замовлення звичайних користувачів а власник магазину матиме змогу переглядати інформацію про загальну кількість замовлень.

2.2 Розробка бази даних системи

База даних проекту складається з трьох колекцій, а саме: колекція користувачів, колекція товарів та колекція замовлень. Оскільки при створенні першого запису в колекцію їй можна задати модель то в коді програми було створено три наступних моделі.

В моделі(рис.2.4) буде зберігатися необхідна інформація про користувача, така як електронна адреса, ім'я чи нікнейм, пароль та рівень доступу користувача.

```
const UserSchema = new Schema({
  email: {
    type: String,
    required: true,
    unique: true
  },
  name: {
    type: String,
    required: true,
  },
  password: {
    type: String,
    required: true,
  },
  access: {
    type: Number,
    default: 0
  }
})
```

Рис. 2.4. Модель користувача

В моделі (рис.2.5) описується товар, а саме: назва товару, картинка товару, ціна товару, тип товару (наприклад корм для котів чи собак), виробник та опис товару (найчастіше опис являє собою склад товару).

```
const GoodsSchema = new Schema({
  name: {
    type: String,
    required: true,
    unique: true
  },
  image: {
    type: String,
    required: true,
  },
  price: {
    type: Number,
    required: true,
  },
  type:{
    type: String,
    required: true
  },
  manufacturer: {
    type: String,
    required: true
  },
  fullDescription: {
    type: String,
    required: true
  }
})
```

Рис. 2.5. Модель товару

В колекції (рис.2.6) буде зберігатися інформація про всі замовлення, тому вона містить у собі інформацію про електронну адресу замовника, контактна інформація замовника, всі товари які він замовив, підсумкова ціна замовлення та дата коли було здійснено замовлення.

```
const OrdersSchema = new Schema({
  userEmail: {
    type: String,
    required: true
  },
  userInfo: {
    userEmail: String,
    city: String,
    mailBox: String,
    fname: String,
    lname: String,
    phone: String,
  },
  orderedGoods: [{
    id: String,
    quantity: Number
  }],
  totalPrice: {
    type: Number,
    required: true
  },
  date: {
    type: Date,
    default: Date.now
  },
  status: {
    type: String,
    default: 'Обробка'
  }
})
```

Рис. 2.6. Модель замовлень

2.3 Проектування та реалізація алгоритмів роботи системи

В ході розробки системи для продажу товарів для тварин було виділено такі основні алгоритми для взаємодії з її користувачами:

1. Алгоритм отримання та відображення інформації про товари.
2. Алгоритм авторизації користувача в системі.
3. Алгоритм оформлення замовлення.

4. Алгоритм утворення статистики.
5. Алгоритм оновлення статусу замовлення.
6. Алгоритм роботи кошика.

Перші три алгоритми реалізовувались за допомогою бібліотеки управління станами React-Redux, котра виступила посередником між FE та BE частинами системи.

Перший з наведених вище алгоритмів в коді виглядає наступним чином (рис. 2.7 – рис. 2.8):

```
fetchAsync: (id) => async (dispatch) => {
  dispatch(storeActions.startFetching());

  let response;
  if (id) {
    response = await api.products.fetchById(id);
  } else {
    response = await api.products.fetch();
  }
  if (response.status === 200) {
    dispatch(storeActions.fill(response.data));
  } else {
    const error = {
      status: response.status
    };
    dispatch(storeActions.setFetchingError(error));
  }
  dispatch(storeActions.stopFetching());
},
```

Рис. 2.7. Частина коду

В цьому коді ми звертаємося до API товарів в залежності чи є в нас id товару та отримуємо в результаті або список усіх товарів або інформацію про конкретний товар. Це використовується щоб відобразити всі товари на головній сторінці вебдодатку та відобразити більш детальну інформацію на сторінці конкретного товару.

```

router.get('/get-goods', async (req, res) => {
  try {
    const goods = await Goods.find();
    res.json(goods);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

router.get('/get-goods/:id', async (req, res) => {
  try {
    const id = req.params.id;
    const goods = await Goods.findById(id);
    if (!goods) {
      return res.status(404).json({ message: 'Goods not found' });
    }
    return res.json(goods);
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: 'Server error' });
  }
});

```

Рис. 2.8. Частина коду

В залежності який із варіантів було викликано в коді (див. рис. 2.7) викликається один з двох наведених запитів і повертається або список всіх товарів або інформація про конкретний товар.

Другий алгоритм (рис. 2.9) звертається до бази даних та шукає там користувача по введеній електронній пошті а потім порівнює пароль який ввів користувач із паролем який записано в базі даних і у випадку якщо пароль вірний інформація про користувача відправляється на FE.

На клієнтській частині програми отримана інформація про користувача записується в стан Redux (рис. 2.10) та використовується на кожній сторінці вебдодатку

```

router.post('/sign-in', cors(), async (req, res) => {
  const { email, password } = req.body;
  const user = await users.findOne({ email });

  if (!user) {
    return res.status(401).json({ message: 'Authentication failed' });
  }

  const isPasswordMatch = user.password === password;

  if (!isPasswordMatch) {
    return res.status(401).json({ message: 'Authentication failed' });
  }

  res.status(200).json({ message: 'Authentication successful', user: user });
});

```

Рис. 2.9. Частина коду

```

authAsync: (data) => async (dispatch) => {
  dispatch(userActions.startFetching());

  try {
    const response = await api.users.auth(data);

    if (response.status === 200) {
      dispatch(userActions.fill(response.data));
    } else {
      const error = {
        status: response.status
      };
      dispatch(userActions.setFetchingError(error));
    }
  } catch (error) {
    dispatch(userActions.setAuthError(error));
  }

  dispatch(userActions.stopFetching());
},

```

Рис. 2.10. Частина коду

Третій алгоритм (рис. 2.11 – – рис. 2.12) спочатку збирає всю необхідну інформацію про замовлення в об'єкт який після натискання кнопки «Замовити» запише його в стан Redux а потім відправить до BE де замовлення буде збережено до бази даних.

```
const onSubmit = async (data) => {
  const dataToAdd = {
    userEmail: userData?.user?.email || data.userEmail,
    userInfo: data,
    orderedGoods: newList,
    totalPrice: getTotalPrice(),
    date: new Date()
  }
  try {
    addOrder(dataToAdd)
    router.push('/')
  } catch (err) {
    console.error(err);
  }
};
```

Рис. 2.11. Частина коду

```
addAsync: (data) => async (dispatch) => {
  dispatch(ordersActions.startFetching());

  try {
    const response = await api.orders.order(data);

    if (response.status === 200) {
      dispatch(ordersActions.fill(response.data));
    } else {
      const error = {
        status: response.status
      };
      dispatch(ordersActions.setFetchingError(error));
    }
  } catch (error) {
    dispatch(ordersActions.setFetchingError(error));
  }

  dispatch(ordersActions.stopFetching());
},
```

Рис. 2.12. Частина коду

```

router.post('/add-order', async (req, res) => {
  try {
    const {
      userEmail,
      userInfo,
      orderedGoods,
      totalPrice,
      date,
      status
    } = req.body;
    const newOrder = new Orders({
      userEmail,
      userInfo,
      orderedGoods,
      totalPrice: parseFloat(totalPrice),
      date,
      status
    });
    const savedOrders = await newOrder.save();
    res.json(savedOrders);
  } catch (error) {
    console.error(error);
    res.status(500).send('Server error');
  }
});

```

Рис. 2.13. Частина коду

Четвертий алгоритм був розроблений з мінімальним використанням React-Redux, в ньому прописана логіка для отримання інформації про товари, фільтрація згідно з id замовлених товарів та врахування кількості замовлених одиниць кожного товару (рис.2.14):

```

const updateManufacturerAndTypesCounts = () => {
  const countsManufacturers = {};
  const countsTypes = {};

  uniqueManufacturers.forEach(manufacturer => {
    countsManufacturers[manufacturer] = 0;
  });

  uniqueTypes.forEach(type => {
    countsTypes[type] = 0;
  });

  pageData?.forEach(item => {
    item?.orderedGoods.forEach(itemId => {
      const product = data.find(product => product._id === itemId.id);

      if (product) {
        countsManufacturers[product.manufacturer] += itemId.quantity;
        countsTypes[product.type] += itemId.quantity;
      }
    });
  });

  setManufacturerCounts(countsManufacturers);
  setTypeCounts(countsTypes);
};

```

Рис. 2.14. Логіка підрахунку статистики

					ІПЗ.КР.Б-121-23-ПЗ	
						25

П'ятий алгоритм оновлює статус замовлення коли адміністратор змінює його а також надсилає(рис. 2.15 – рис. 2.18):

```
const onSubmit = (formData) => {
  if (selectedOption) {
    updateOrder(data?._id, formData)
    sendEmail(data, formData)
  }
}
```

Рис. 2.15. Зміна статусу

```
const sendEmail = (data, foemData) => {
  emailjs.send(
    serviceId,
    templateID,
    { send_to: data?.userEmail, status: foemData.status },
    publicKey
  )
  .then((response) => {
    console.log('Email sent successfully:', response);
  })
  .catch((error) => {
    console.error('Error sending email:', error);
  });
};
```

Рис. 2.16. Частина коду

Частина коду указана на рисунку 2.16 відповідає за надсилання письма на електронну пошту замовника повідомляючи йому актуальний статус замовлення. Водночас із цим оновлений статус записується у стан Redux та відправляється на BE де оновлює замовлення в базі даних.

```
updateAsync: (id, data) => async (dispatch) => {
  try {
    const response = await api.orders.updateStatus(id, data);

    if (response.status === 200) {
      console.log("cool")
    } else {
      const error = {
        status: response.status
      };
      dispatch(ordersActions.setFetchingError(error));
    }
  } catch (error) {
    dispatch(ordersActions.setFetchingError(error));
  }
}
```

Рис. 2.17. Частина коду

```

router.put('/update-order/:id', async (req, res) => {
  try {
    const { status } = req.body;
    const orders = await Orders.findByIdAndUpdate(
      req.params.id,
      { status },
      { new: true }
    );
    if (!orders) {
      res.status(404).json({ message: 'order not found' });
    } else {
      res.status(200).json(orders);
    }
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Server Error' });
  }
});

```

Рис. 2.18. Частина коду

Шостий алгоритм (рис. 2.19 – рис. 2.20) являє собою стандартну роботу кошика, який повинен бути у будь-якій системі, яка спеціалізується на продажі товарів.

```

const cartSlice = createSlice({
  name: 'cart',
  initialState: [],
  reducers: {
    addToCart: (state, action) => {
      const itemExists = state.find((item) => item._id === action.payload._id)
      if (itemExists) {
        itemExists.quantity++;
      } else {
        state.push({ ...action.payload, quantity: 1 });
      }
    },
    incrementQuantity: (state, action) => {
      const item = state.find((item) => item._id === action.payload);
      item.quantity++;
    },
    decrementQuantity: (state, action) => {
      const item = state.find((item) => item._id === action.payload);
      if (item.quantity === 1) {
        const index = state.findIndex((item) => item._id === action.payload);
        state.splice(index, 1);
      } else {
        item.quantity--;
      }
    },
    removeFromCart: (state, action) => {
      const index = state.findIndex((item) => item._id === action.payload);
      state.splice(index, 1);
    },
    removeAllFromCart: (state) => {
      state = []
    },
  },
});

```

Рис. 2.19. Частина коду

На рисунку 2.19 описано зріз в Redux який контролює стан кошика та оновлює його вміст при викликах різних функцій, що представлені у цьому зрізі.

```
{cartData.map((item) => (
  <div className={styles.body} key={item._id}>
    <div className={styles.image}>
      <Image src={item.image} height="90" width="65" alt='image' />
    </div>
    <p>{item.name}</p>
    <p>{item.price} грн</p>
    <p>{item.quantity}</p>
    <div className={styles.buttons}>
      <button onClick={() => addQuantity(item._id)}>
        +
      </button>
      <button onClick={() => removeQuantity(item._id)}>
        -
      </button>
      <button onClick={() => deleteFromCart(item._id)}>
        x
      </button>
    </div>
    <p>{((item.quantity * item.price).toFixed(2))} грн</p>
  </div>
))}
<h2>Взагалом: {getTotalPrice()} грн</h2>
```

Рис. 2.20. Частина коду

На рисунку 2.20 зображено алгоритм візуалізації наповнення кошика у зрозумілому для користувача вигляді.

2.4 Реалізація функціональних можливостей системи продажу зоотоварів

Під час розробки системи для продажу товарів для тварин було додано декілька найважливіших в роботі магазину функціональних можливостей а саме:

1) Каталог товарів – модель товарів в базі даних в якій зберігається вся інформація про товари доступні у вебдодатку, ВЕ складова яка отримує повний каталог товарів та передає його на клієнтську частину вебдодатку та FE частина яка обробляє отриману інформацію та виводить її для користувача у приємному вигляді.

2) Додавання товарів у кошик – розроблена можливість для користувача додавати обрані товари у кошик покупок з якого в подальшому

список замовлених товарів та інформація про користувача-замовника потрапляє до бази даних.

3) Керування замовленнями – з боку адміністратора сайту він може отримувати інформацію про всі замовлення або шукати замовлення певного користувача та керувати їх станом змінюючи статус замовлення на один із варіантів, при зміні статусу замовлення до замовника буде надіслано електронне письмо. Можливі статуси замовлення:

- Обробка.
- Виконується.
- Відправлено.
- Відмінено.

4) Користувацький обліковий запис – система забезпечує можливість реєстрації та авторизації користувача як для звичайного клієнта так і для адміністратора, в якого наявна можливість зареєструвати нового користувача з певним рівнем доступу.

5) Створення нових товарів – у адміністратора сайту наявна можливість додавати нові товари в базу даних вебдодатку. Для цього йому необхідно перейти на відповідну сторінку та заповнити необхідні поля для вводу а також завантажити фотографію товару.

6) Видалення товарів – адміністратор, за потреби, може видаляти товари з бази даних вебдодатку, для цього необхідно натиснути на іконку видалення на картці товару, яку бачить тільки користувач з рівнем доступу адміністратора.

7) Перегляд статистики – користувач з рівнем доступу власника може відвідати сторінку статистики по замовленнях. На ній наявно 3 кругові діаграми: перша показує актуальну кількість замовлень поділяючи їх по статусах; друга показує скільки товарів кожного виробника було замовлено; третя демонструє кількість замовлених товарів за типами цих товарів.

Висновки до другого розділу

Отже, в другому розділі було розроблено чотири діаграми, а саме «Діаграма варіантів використання», «Діаграма розгортання», «Діаграма компонентів» та «Діаграма активності».

Було описано базу даних та розроблено взаємодію з нею. Було проаналізовано та описано основні алгоритми та спроектовано базовий функціонал системи для продажу товарів для тварин.

					ІПЗ.КР.Б-121-23-ПЗ	
						30

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМИ ДЛЯ ПРОДАЖУ ТОВАРІВ ДЛЯ ТВАРИН

3.1 Порядок встановлення та налаштування параметрів системи

Встановлення та налаштування системи потребує розгортання всіх її компонентів та системи.

Схема розгортання (діаграмі розгортання) представлена на рисунку 3.1

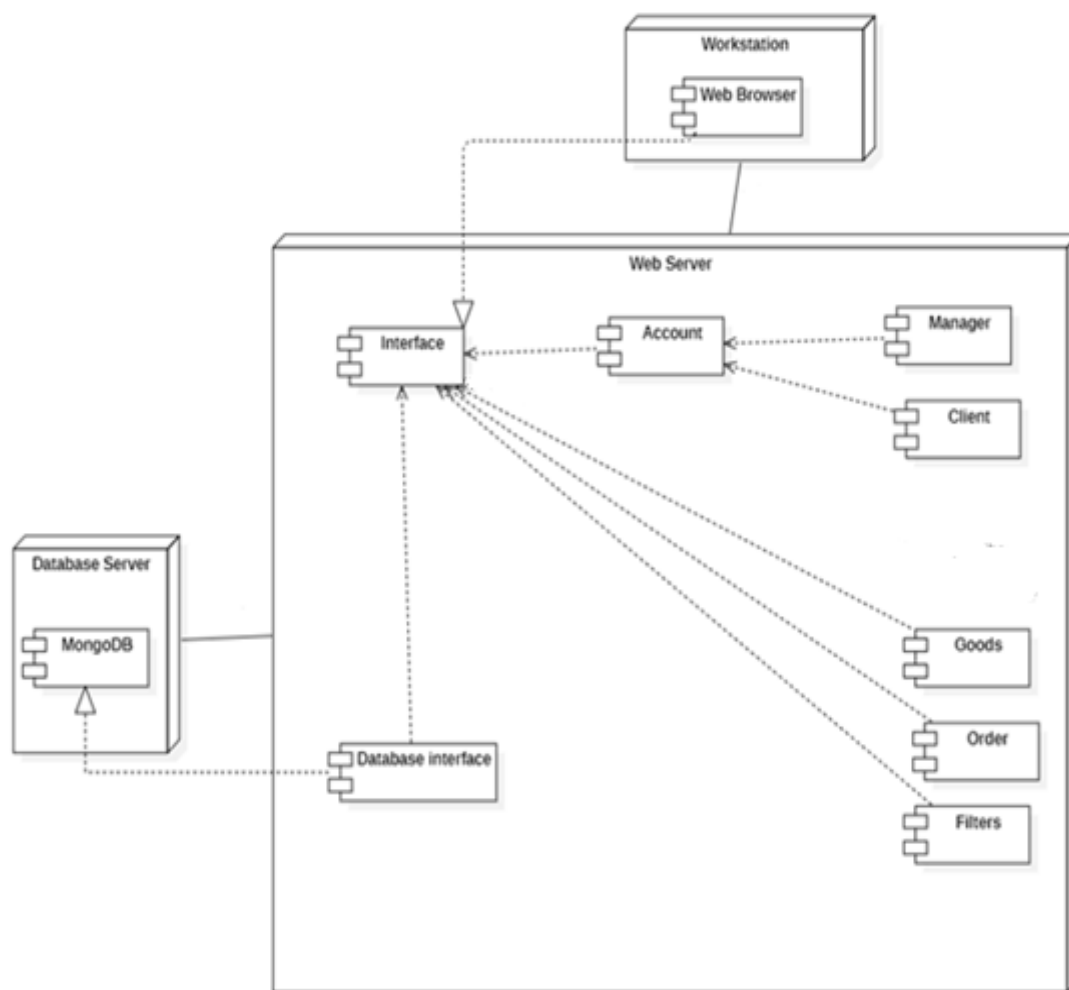


Рис. 3.1. Діаграма розгортання

Оскільки в розробці було використано MongoDB Atlas в якості бази даних, то вона не потребує розгортання оскільки вже є хмарною.

Web Server, який представляє собою інтерфейс програми з можливістю отримувати/додавати інформацію до кластеру бази даних та оброблювати її для подальшого відображення для користувача.

Work station – це робоча станція користувача, використовуючи яку він може, за допомогою різних браузерів, отримати доступ до системи продажу товарів для тварин.

3.2 Структура інтерфейсу системи для продажу товарів для тварин

Під час першого заходу на сторінку користувач одразу бачить основний інтерфейс вебдодатку, який містить у собі навігаційну панель, на якій можна побачити назву сайту, кнопку яка пропонує авторизуватися та корзину з кількістю предметів доданих у неї (рис. 3.2).

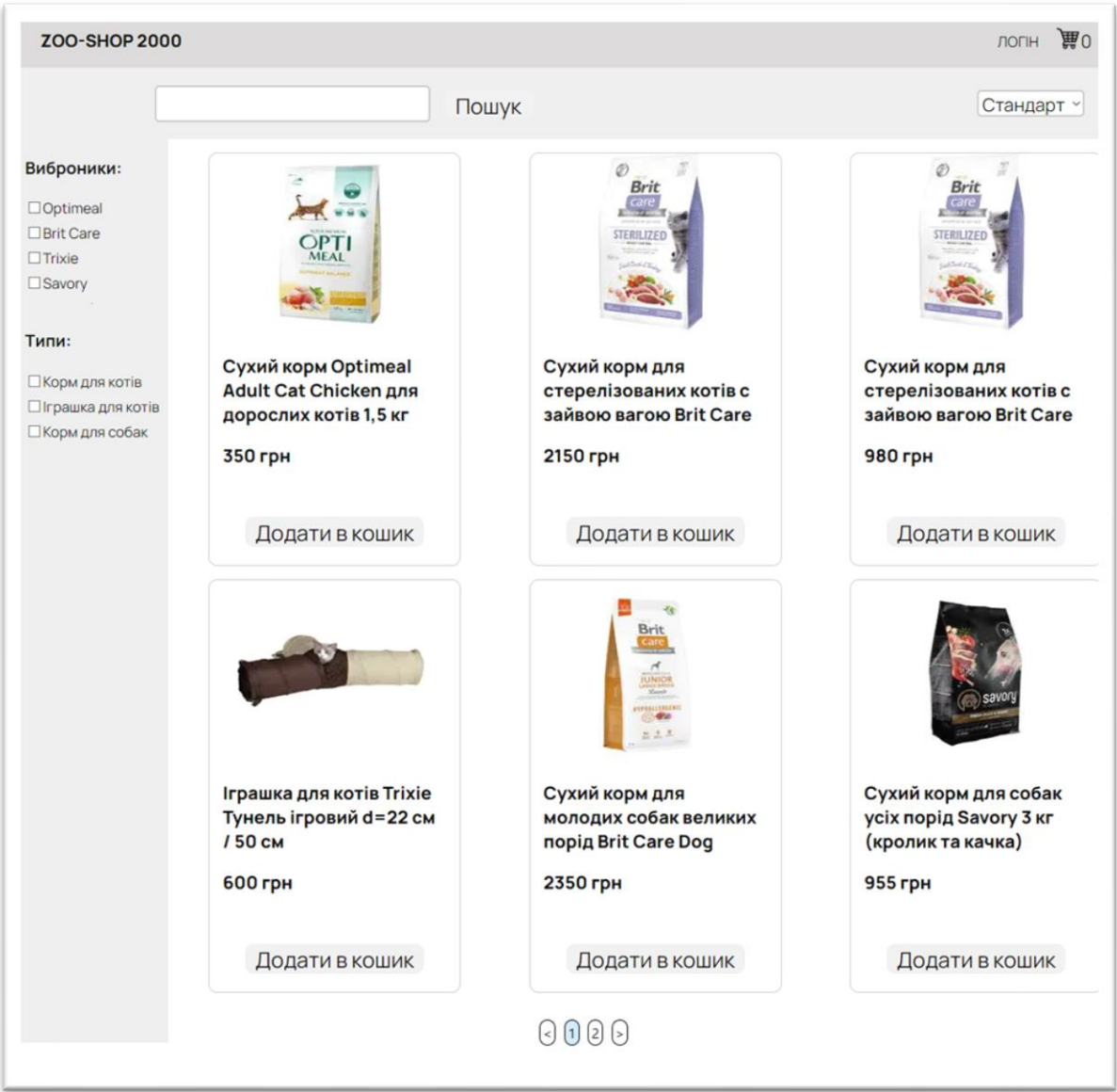


Рис. 3.2. Інтерфейс головної сторінки

Також на рисунку 3.2 можна побачити поле для пошуку товарів за назвою, ціною, виробником, типом товару або ж словами які наявні в описі товару. Окремо виділяються фільтри та сортування товару(рис 3.3 – рис 3.5). Фільтри зроблені динамічно, тобто вони оновлюються кожен раз коли оновлюється сторінка, тому у випадку якщо під час користування вебдодатком з боку адміністратора буде видалено останній товар який відповідає конкретному типу або виробнику то даний пункт зникне при наступному оновленні сторінки звичайним користувачем. Сортування представлене на сайті в двох варіантах: за зростанням або за спаданням цін.

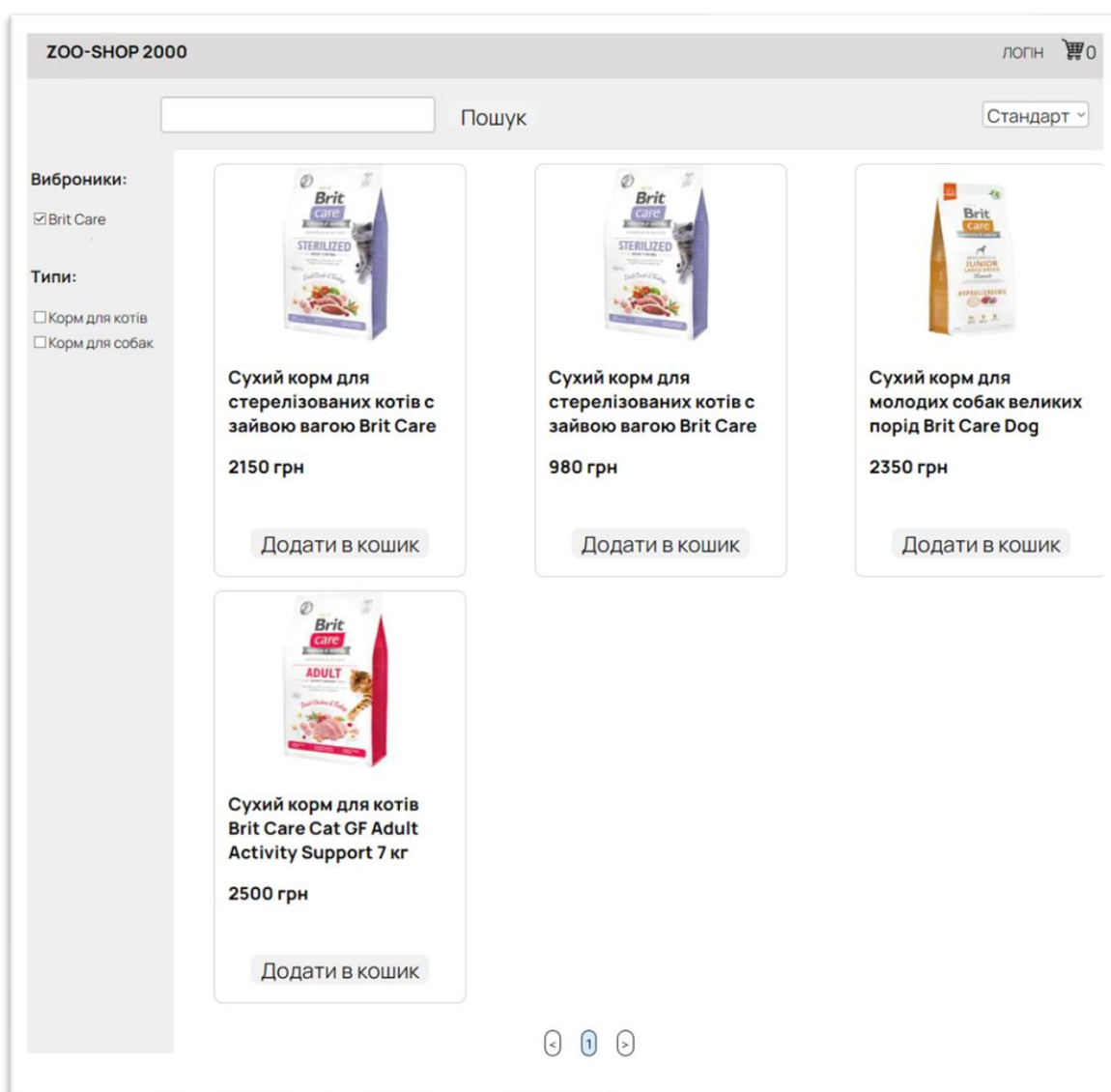


Рис. 3.3. Відфільтровані товари за виробником

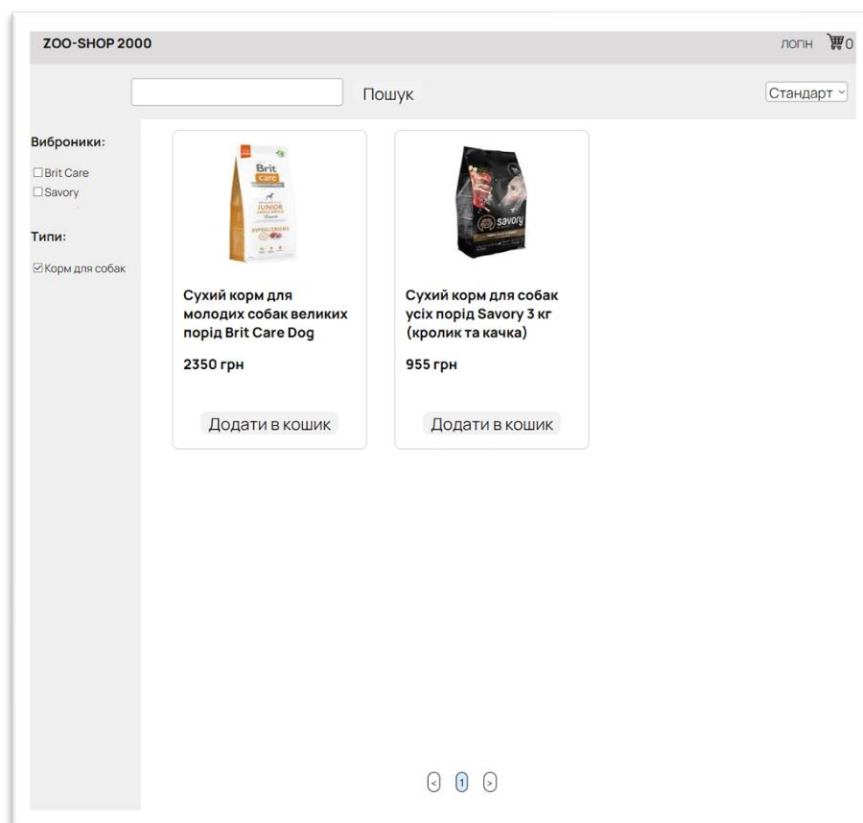


Рис. 3.4. Відфільтровані товари за типом товару

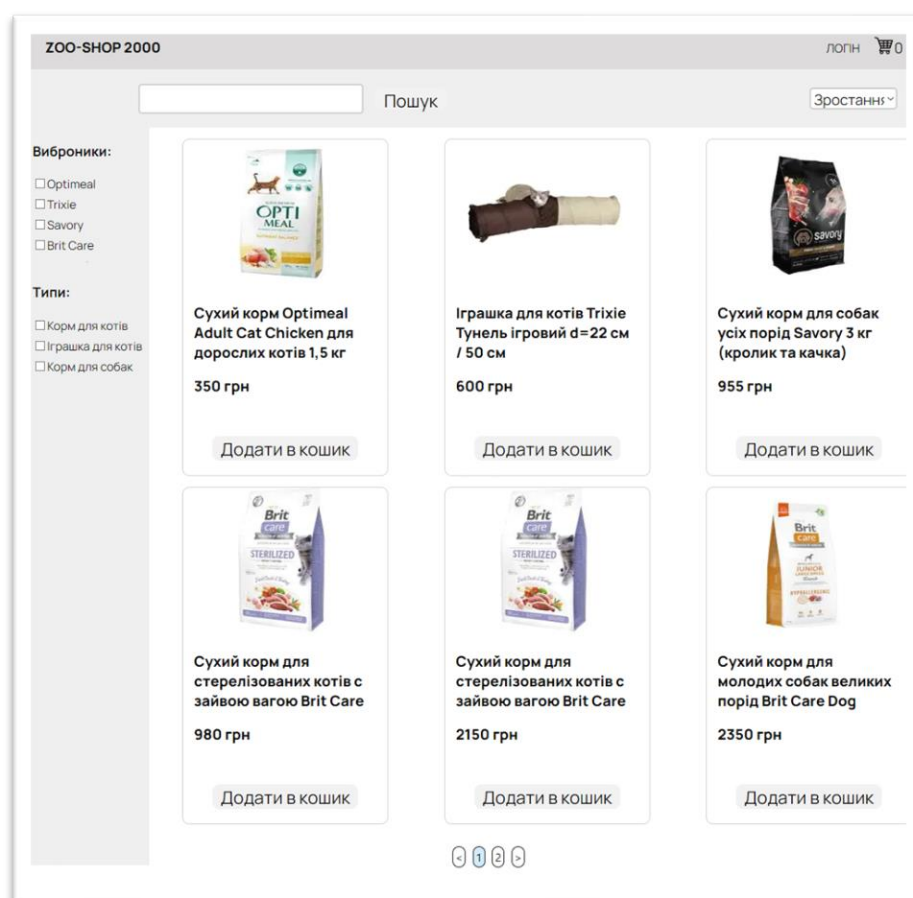


Рис. 3.5. Відсортовані товари за зростанням ціни

Далі у користувача є наступні можливості: авторизуватись, додати товар в кошик або переглянути більш детальну інформацію про товар.

При натисканні на карточку товару, користувач потрапляє на сторінку з більшою кількістю інформації про цей товар(рис. 3.6).

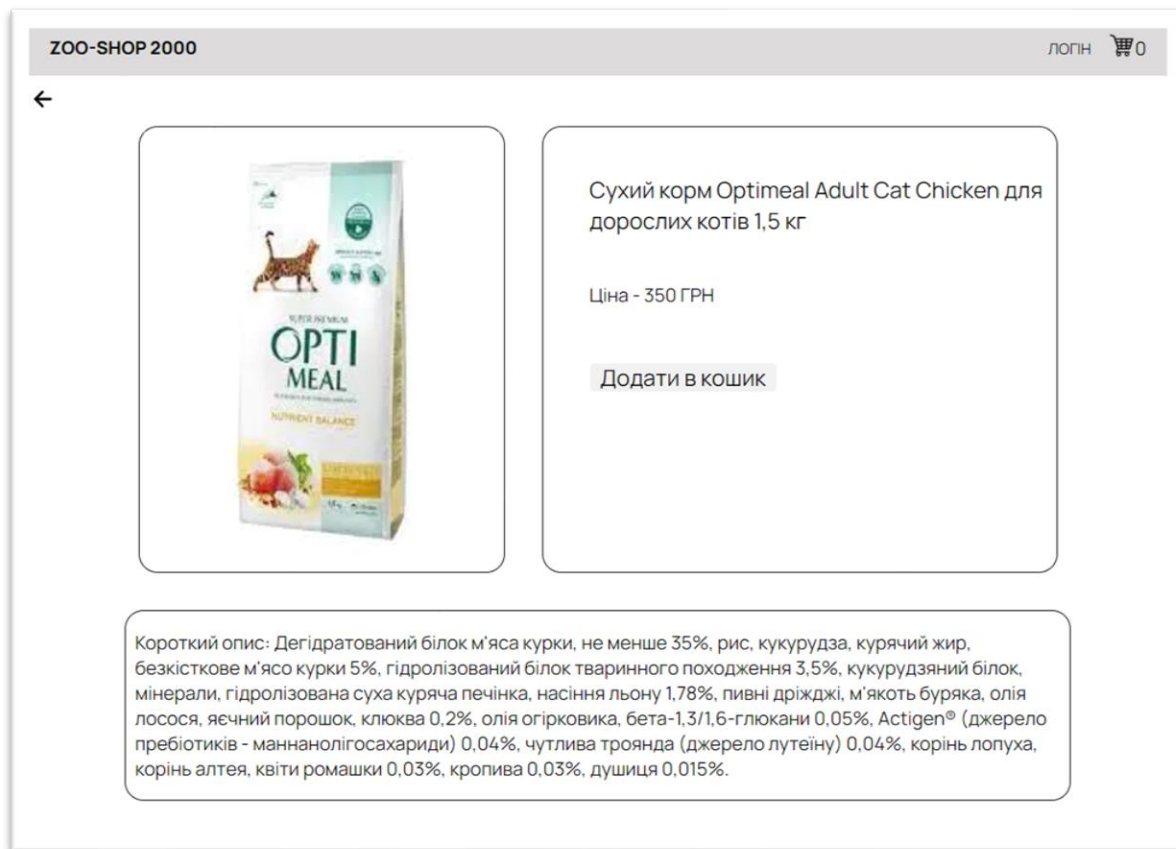


Рис. 3.6. Сторінка товару з описом

Зазвичай на подібних сторінках користувач може побачити додаткову інформацію про товар, яка не наведена на карточці товару, наприклад додатковий короткий опис товару або побачити повну назву товару яка може не міститися в карточці, найчастіше у назві буде вказана його вага.

Додаючи товар в кошик користувач може побачити збільшення числового індикатору товару в кошику та натиснувши на іконку перейти до сторінки з товарами які були обрані користувачем (рис. 3.7):

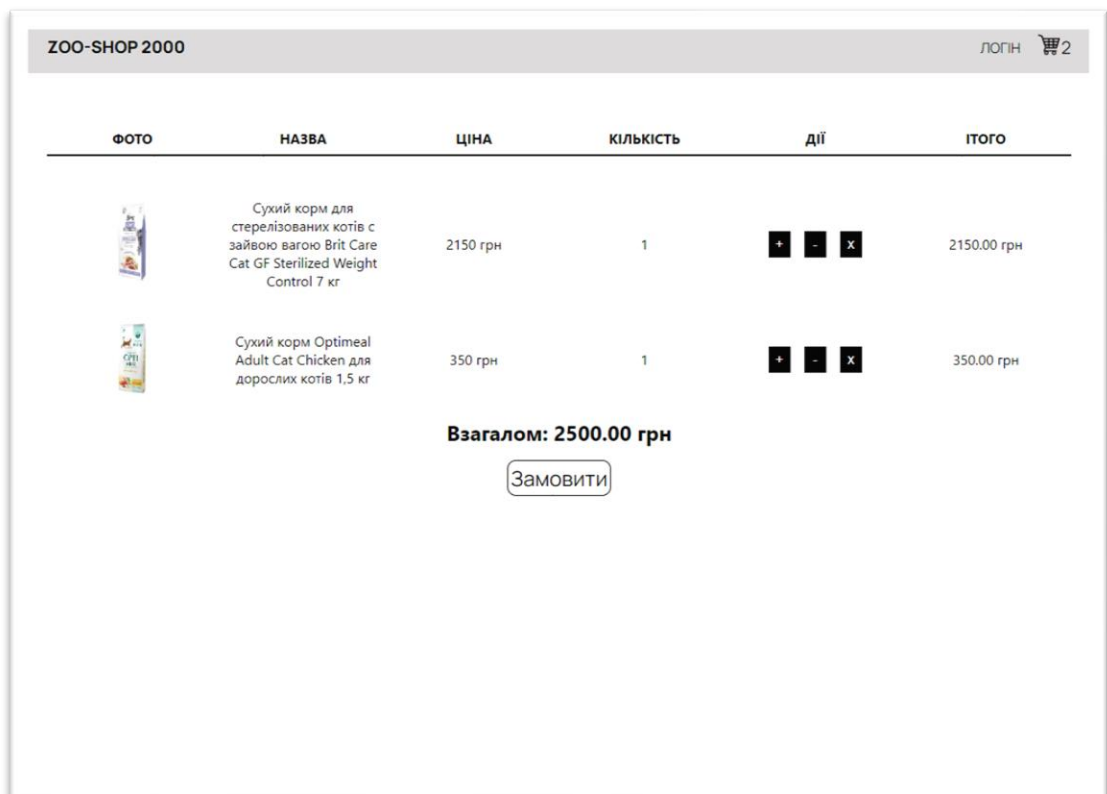


Рис. 3.7. Сторінка кошику

На рисунку 3.7 зображено інтерфейс кошику, де у вигляді таблиці представлені всі товари які обрав користувач. В колонці «Дії» користувач може збільшити або зменшити кількість однакових товарів які він обрав для замовлення. Також в тій самій колонці він може видалити товар з кошику якщо не бажає його купувати.

Знизу кошика вказана загальна ціна товарів та доступна кнопка для оформлення покупки, яка веде на наступну сторінку в ланцюгу замовлення товарів.

Сторінка оформлення замовлення представлена на рисунку 3.8. Перш за все треба звернути увагу на поле з електронною адресою, яке на рисунку 3.8 позначено червоним прямокутником. Якщо користувач буде авторизованим це поле не з'явиться. Всі інші поля обов'язкові до заповнення. Після натискання кнопки «Замовити» замовлення потрапляє до бази даних. У випадку відміни замовлення воно залишається в базі даних зі статусом «Відмінено».

ZOO-SHOP 2000

ЛОГІН 2

Email:

Ім'я:

Фамілія:

Номер телефону:

Місто:

№ Нової Пошти:

Замовити

Рис. 3.8. Сторінка оформлення замовлення

Останньою можливістю неавторизованого користувача є можливість авторизуватися. При натисканні кнопки «Логін» користувач потрапляє на сторінку авторизації (рис. 3.9).

ZOO-SHOP 2000

ЛОГІН 2

Пошта:

Пароль:

Вхід

Якщо потрібен акаунт - [Регістрація](#)

Рис. 3.9. Сторінка авторизації

На цій сторінці користувач може авторизуватися або, якщо не має акаунту, перейти на сторінку реєстрації (рис 3.10).

На сторінці реєстрації користувачу необхідно ввести базові данні для створення свого акаунту. При натисканні «Продовжити» якщо всі поля заповнені вірно користувач буде автоматично авторизований.

ZOO-SHOP 2000

ЛОГІН  2

Нікнейм:

Пошта:

Пароль:

Продовжити

Рис 3.10. Сторінка реєстрації

Після авторизації користувач потрапляє на головну сторінку але з деякими видозмінами, які залежать від рівню доступу користувача. У випадку коли користувач не має жодного підвищеного рівню доступу його головна сторінка матиме наступний вигляд (рис. 3.11).

ZOO-SHOP 2000

ОСОБИСТИЙ КАБІНЕТ ВИЙТИ  0

Пошук

Стандарт ▾

Виброніки:

☐ Optimeal
 ☐ Brit Care
 ☐ Trixie
 ☐ Savory

Типи:

☐ Корм для котів
 ☐ Іграшка для котів
 ☐ Корм для собак



Сухий корм Optimeal Adult Cat Chicken для дорослих котів 1,5 кг
 350 грн

Додати в кошик



Сухий корм для стерелізованих котів с зайвою вагою Brit Care
 2150 грн

Додати в кошик



Сухий корм для стерелізованих котів с зайвою вагою Brit Care
 980 грн

Додати в кошик



Іграшка для котів Trixie Тунель ігровий d=22 см / 50 см
 600 грн

Додати в кошик



Сухий корм для молодих собак великих порід Brit Care Dog
 2350 грн

Додати в кошик



Сухий корм для собак усіх порід Savory 3 кг (кролик та качка)
 955 грн

Додати в кошик

< 1 2 >

Рис. 3.11. Головна сторінка після авторизації звичайного користувача

					ІПЗ.КР.Б-121-23-ПЗ	38

Як можна побачити в навігаційній панелі змінилися написи, тепер користувач може вийти зі свого аккаунту або переглянути особистий кабінет де присутня інформація про всі замовлення зроблені цим користувачем (рис 3.12).

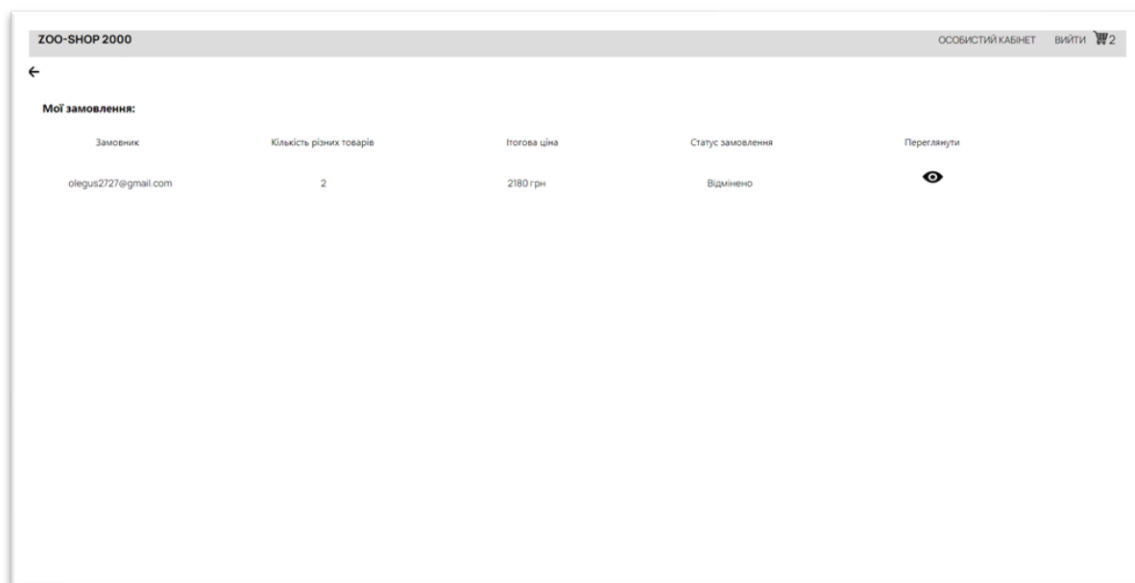


Рис. 3.12. Особистий кабінет користувача

В кабінеті користувач може побачити табличку із своїми замовленнями. У даній таблиці наведена основна інформація про замовлення, а саме: замовник, кількість різних товарів що замовлялась, загальну ціну за замовлення, статус замовлення та натиснувши на іконку «Око» можна потрапити на сторінку на якій наведено всі замовлені товари (рис. 3.13).

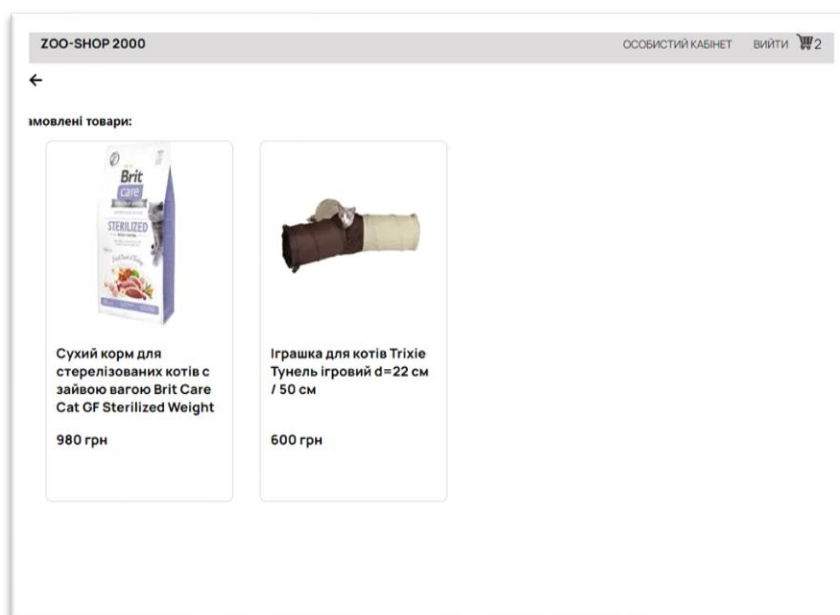


Рис. 3.13. Особистий кабінет користувача(детальніше про замовлення)

Для звичайного користувача це весь функціонал вебдодатку. Наступний тип користувача – власник, це також користувач, тому йому доступні всі можливості звичайного користувача, а також наявна додаткова можливість – перегляд статистики по замовленням, для цього йому необхідно натиснути на кнопку «Переглянути статистику» на навігаційній панелі (рис. 3.14).

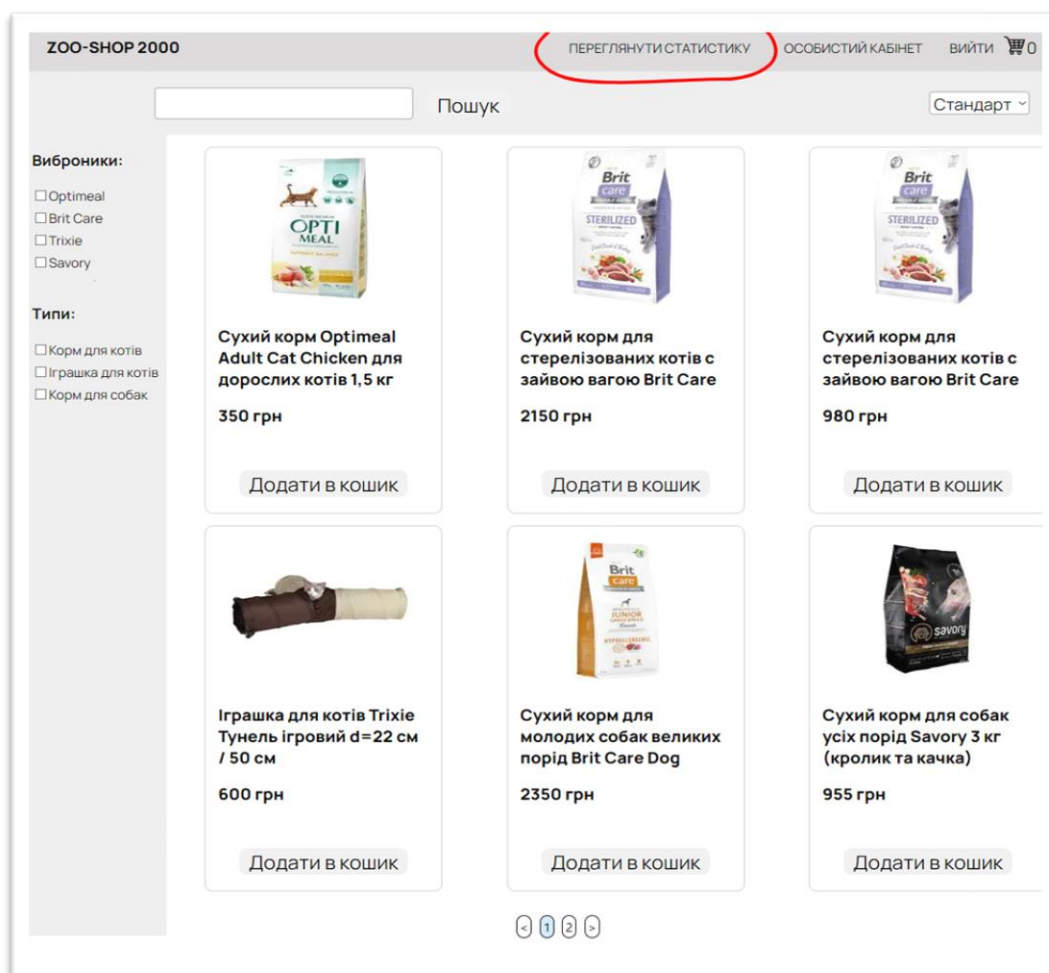


Рис 3.14. Кнопка «Переглянути статистику»

На сторінці статистики можна побачити три кругові діаграми, що показують загальну статистику

На першій діаграмі можна побачити актуальну кількість замовлень різних статусів (рис. 3.15). На другій діаграмі - кількість замовлень по виробнику (рис. 3.16). На третій діаграмі - кількість замовлень по типу товарів (рис. 3.16).

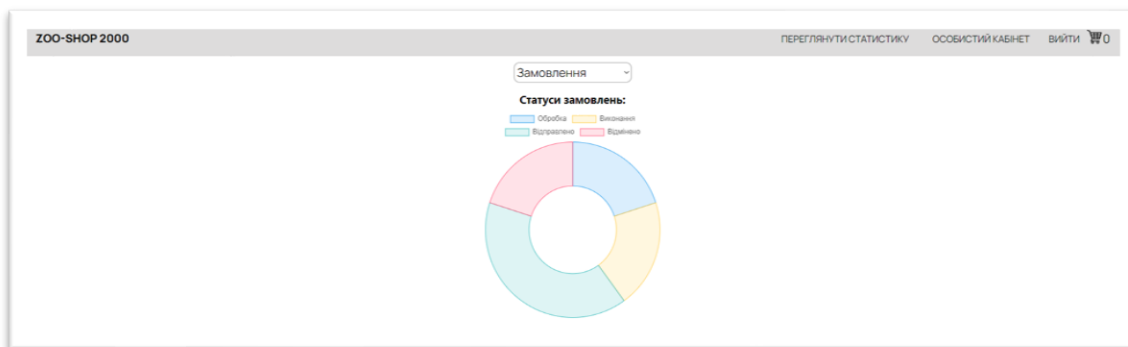


Рис. 3.15. Статистика по замовленням

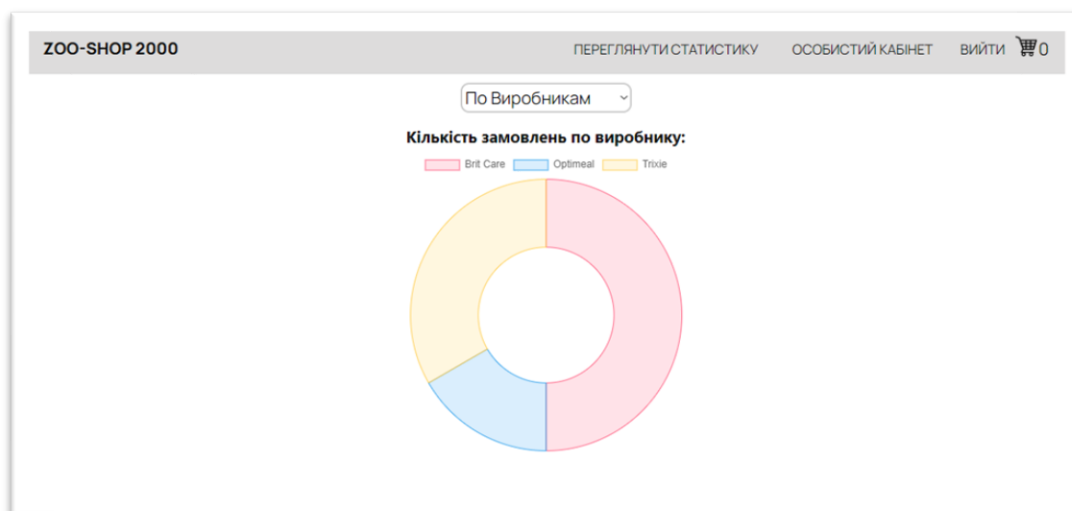


Рис. 3.16. Статистика по виробникам

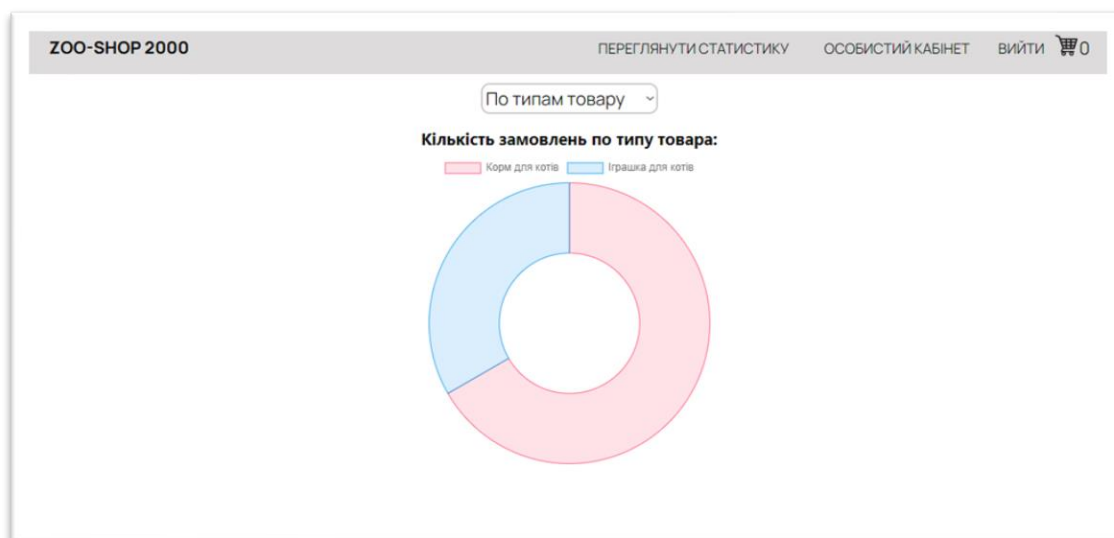


Рис. 3.17. Статистика по типам товару

Останній тип користувача – адміністратор, в нього присутні найширші можливості серед усіх типів користувачів. Він може видаляти товари а також найширшу навігаційну панель з різними можливостями (рис. 3.18).

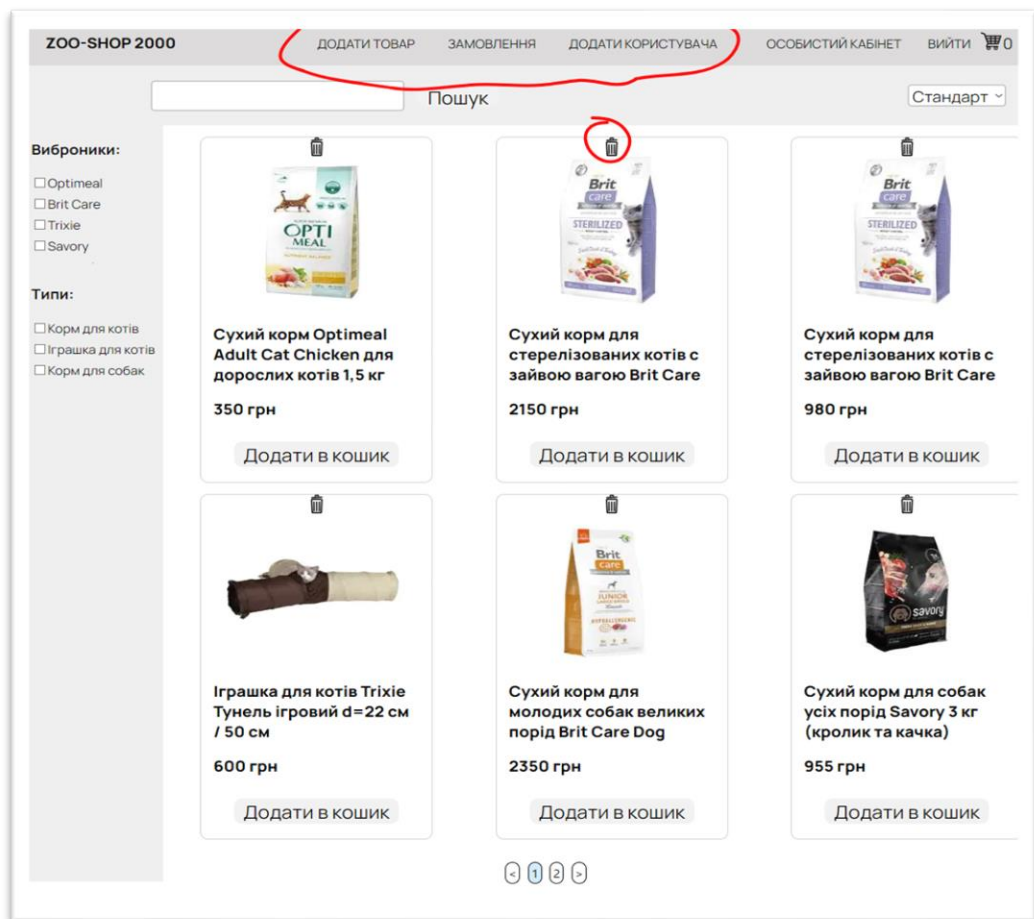


Рис. 3.18. Головна сторінка, вид адміністратора

На рисунку 3.18 червоним обведено основні можливості адміністратора, а саме «додати товар», «додати користувача», «замовлення».

Іконка смітника на карточці з товаром дозволяє видалити товар. Кнопки «додати товар», «додати користувача» та «замовлення» переносять на відповідні сторінки. При переході на сторінку «замовлення» адміністратор бачить наступну табличку (рис. 3.19).

ZOO-SHOP 2000					
ДОДАТИ ТОВАР ЗАМОВЛЕННЯ ДОДАТИ КОРИСТУВАЧА ОСОБИСТИЙ КАБІНЕТ ВИЙТИ					
Замовлення					
Пошук					
Замовник	Кількість різних товарів	Ітогова ціна	Статус замовлення	Переглянути	
ipz191_noi@student.ztu.edu.ua	2	2500 грн	Виконання	👁	
owner@owner.com	1	2150 грн	Обробка	👁	
olegus2727@gmail.com	2	2180 грн	Відмінено	👁	

Рис. 3.19. Сторінка всіх замовлень

Дана сторінка (див. рис. 3.19) містить у собі табличку із всіма замовленнями користувачів.

Адміністратор може переглянути деталі замовлення. Проте найголовніша особливість даної сторінки полягає в тому, що адміністратор може змінювати статуси замовлень (рис. 3.20). Можливі статуси замовлення: обробка, виконується, відправлено, відмінено.

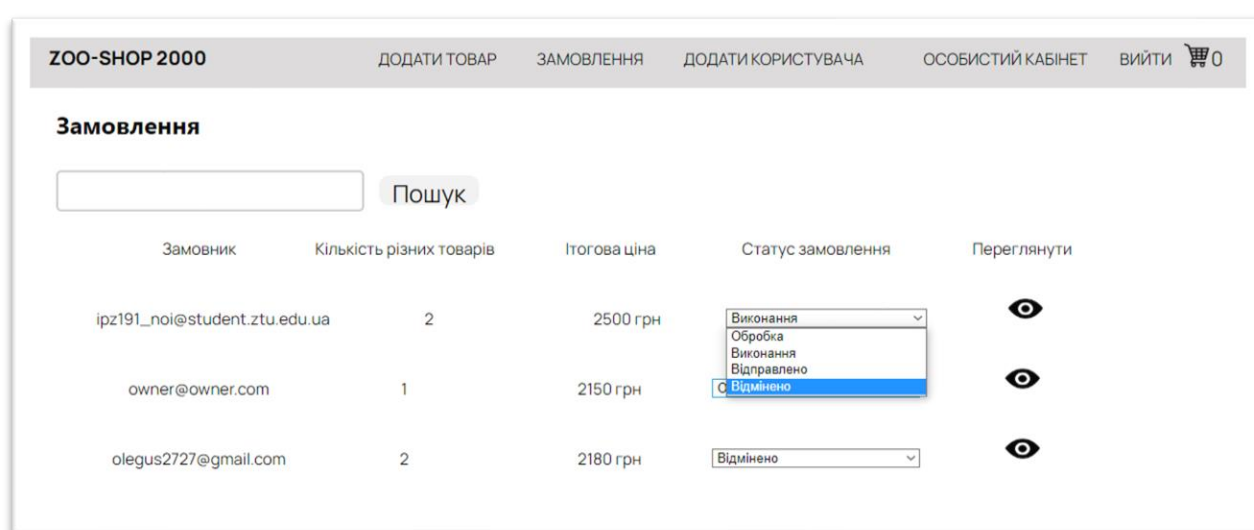


Рис. 3.20. Варіант статусів замовлення

Коли статус замовлення змінюється користувачу чис замовлення було оновлено на електрону пошту приходить письмо з актуальним статусом замовлення (рис. 3.21).

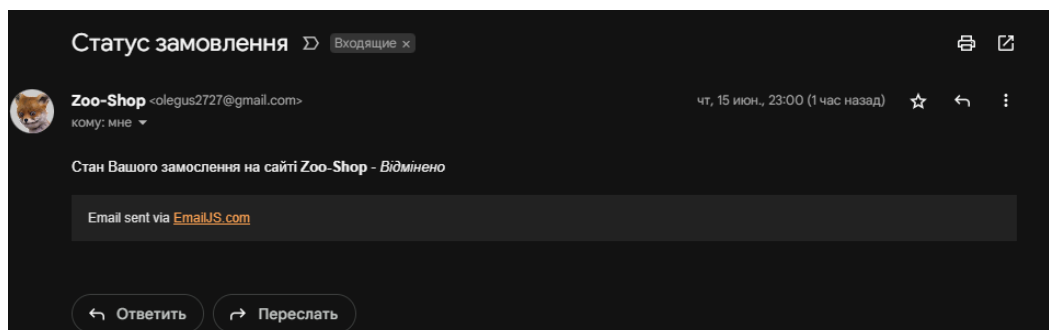


Рис. 3.21. Письмо із статусом замовлення

Також на сторінці замовлень наявне поле пошуку замовлень по замовнику, ціні, кількості різних товарів у замовленні або по статусу замовлення (рис. 3.22).

Рис. 3.22. Пошук по фразі «відмі»

При переході на сторінку додавання товару (рис. 3.23) адміністратору необхідно заповнити кожне поле для додання нового товару який одразу ж з'явиться на сторінці з товарами.

Рис. 3.23. Форма створення товару

Також адміністратор може додати нового користувача (рис. 3.24), змінивши його рівень доступу, тобто додати як звичайного користувача так і нового адміністратора або власника.

Рис. 3.24. Форма додавання користувача.

Отже, можемо дійти висновку, звертаючи увагу на вище наведений функціонал, у вебдодатку реалізовано можливості, котрі є як і життєво-необхідні для будь-якого сайту з продажу товарів, так і необов'язкові для спокійного функціонування програми.

3.3 Тестування роботи програмного комплексу

Для тестування роботи готового вебдодатку був обраний інструмент розробника Lighthouse [22], який наявний в доступі серед інших інструментів розробника на будь-якій сторінці Google Chrome. Він надає коротку інформацію по наступним параметрам: ефективність, спеціальні можливості, оптимальні методи, оптимізація пошукових систем. Під час тестування були перевірені всі сторінки вебдодатку (рис. 3.25 – рис. 3.35).

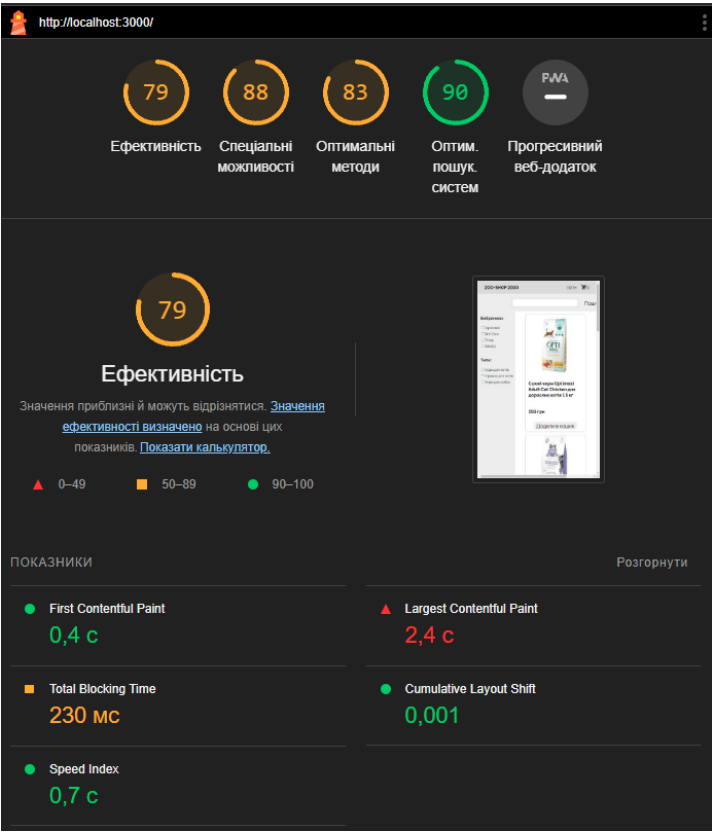


Рис. 3.25. Головна сторінка

На рисунку 3.25 показники в межах норми.

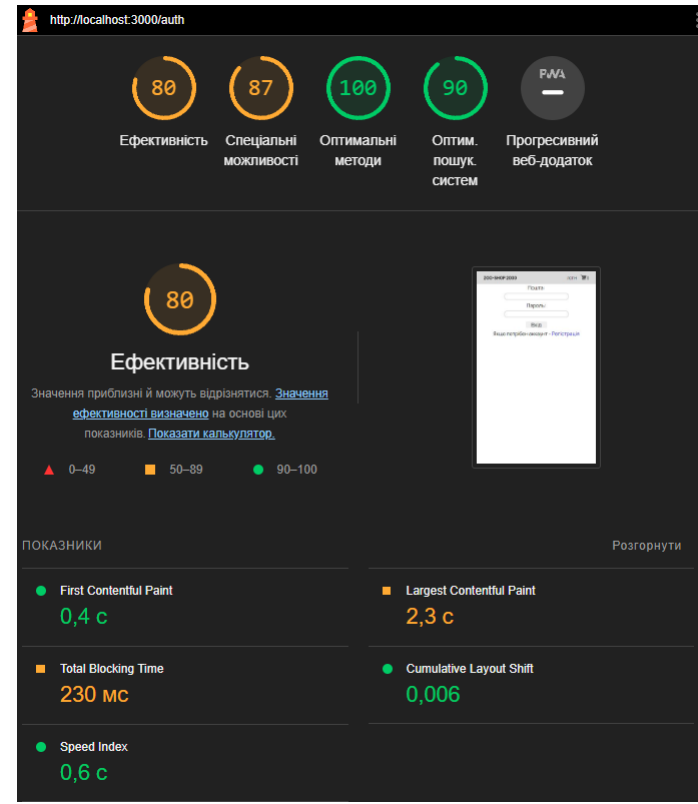


Рис. 3.26. Сторінка авторизації

На рисунку 3.26 показники в межах норми.

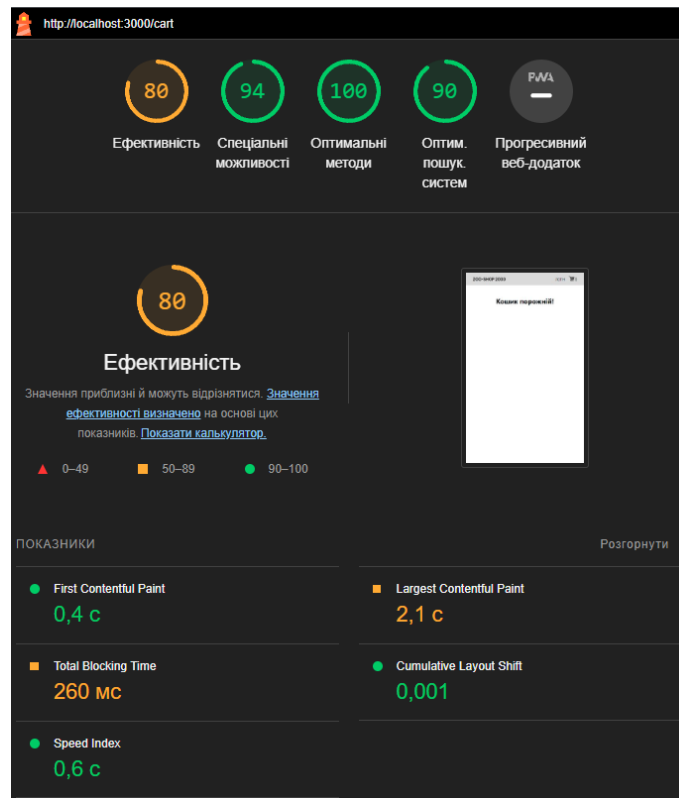


Рис. 3.27. Сторінка кошику

На рисунку 3.27 показники в межах норми.

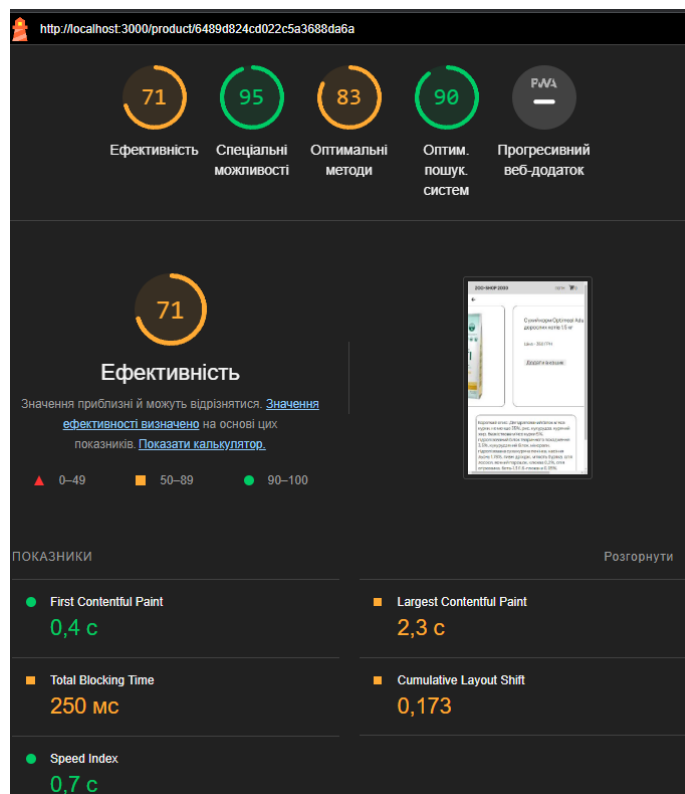


Рис. 3.28. Сторінка товару

На рисунку 3.28 показники в межах норми.

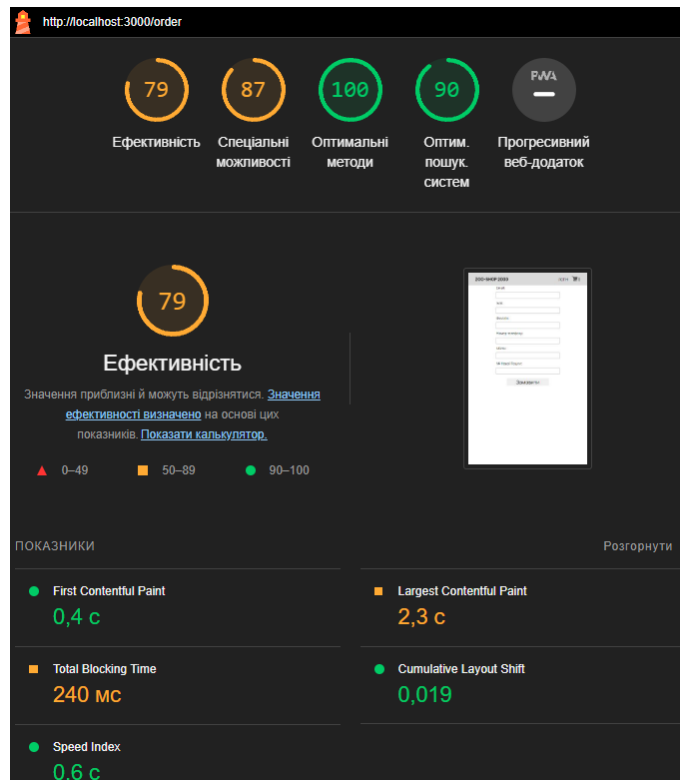


Рис. 3.29. Сторінка оформлення замовлення

На рисунку 3.29 показники в межах норми.

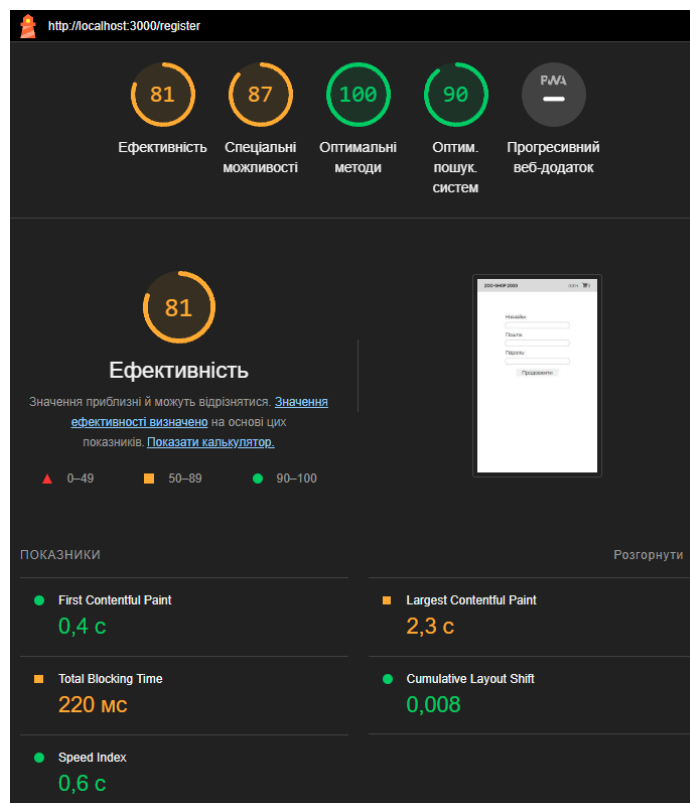


Рис. 3.30. Сторінка реєстрації

На рисунку 3.30 показники в межах норми.

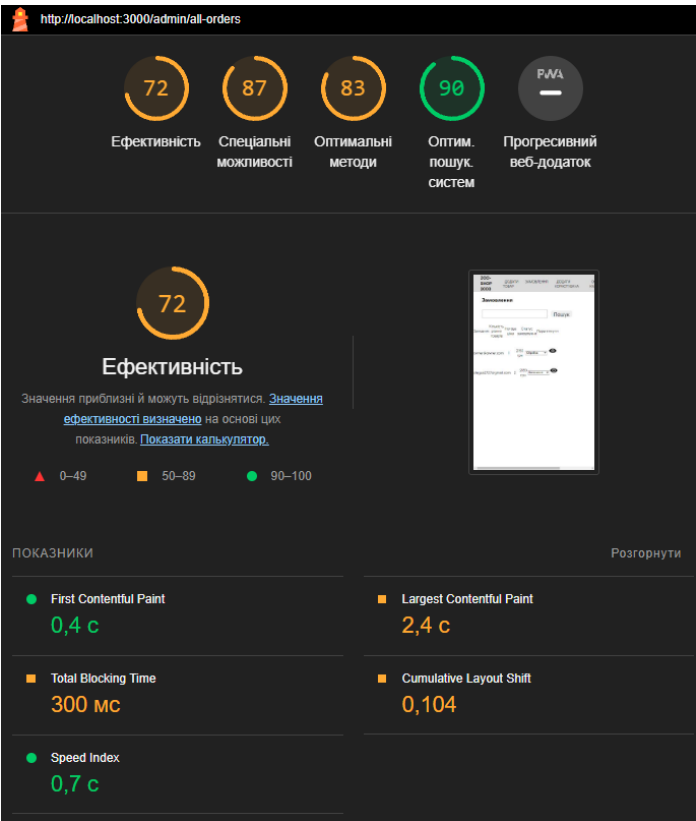


Рис. 3.31. Сторінка всіх замовлень

На рисунку 3.31 показники в межах норми.

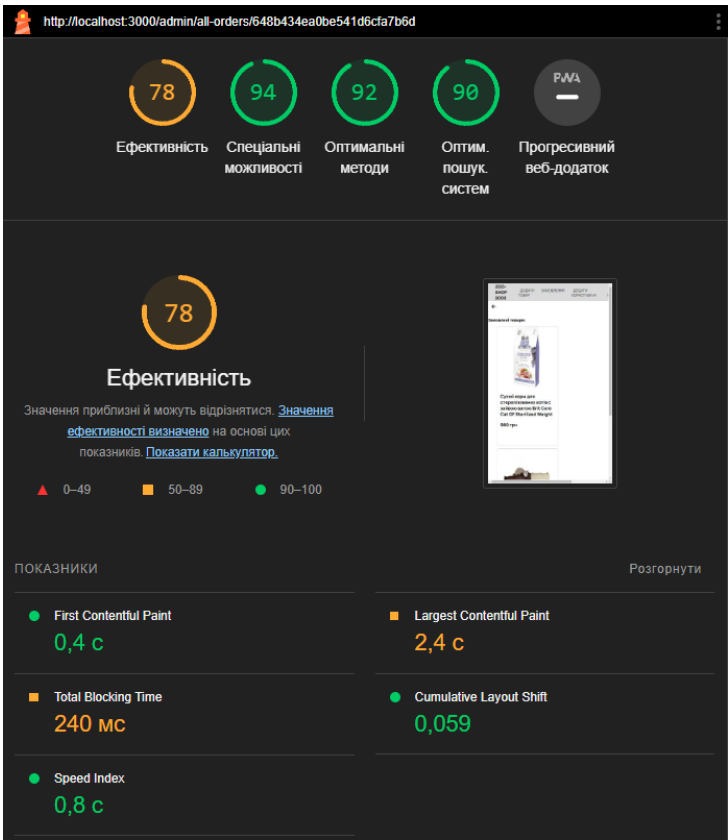


Рис. 3.32. Сторінка конкретного замовлень

На рисунку 3.32 показники в межах норми.

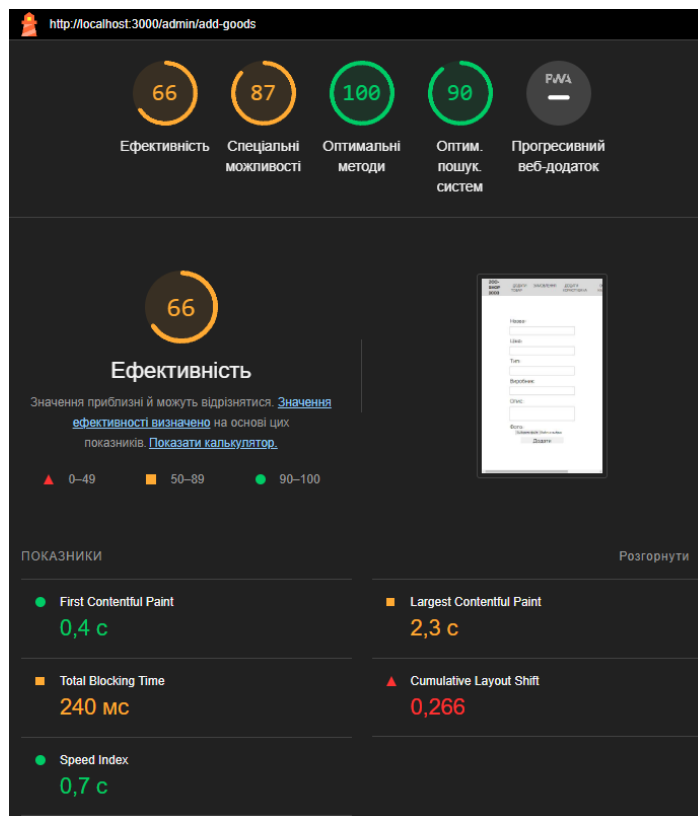


Рис. 3.33. Сторінка додавання нового товару

На рисунку 3.33 показники в межах норми.

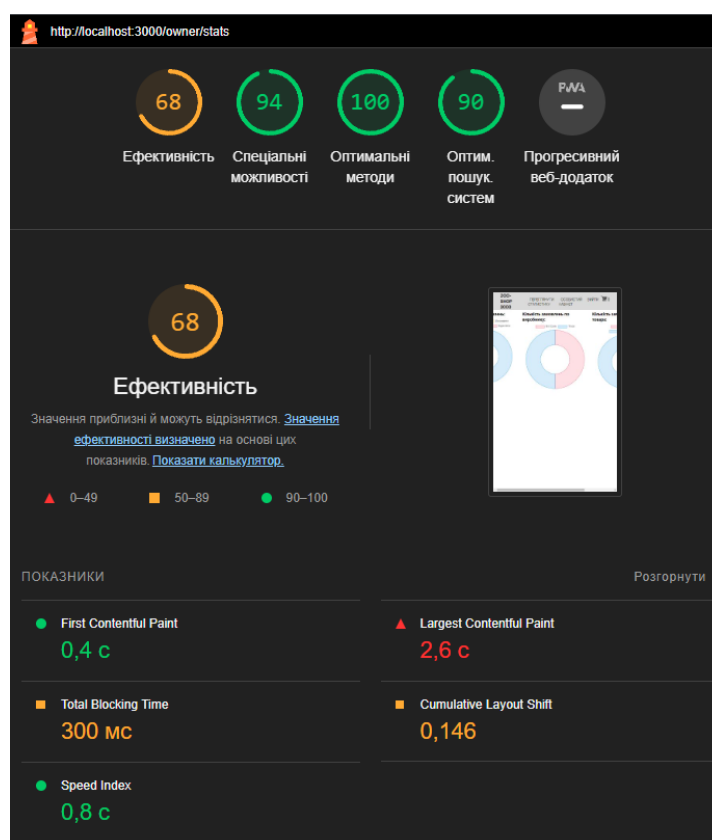


Рис. 3.34. Сторінка статистики

На рисунку 3.34 показники в межах норми.

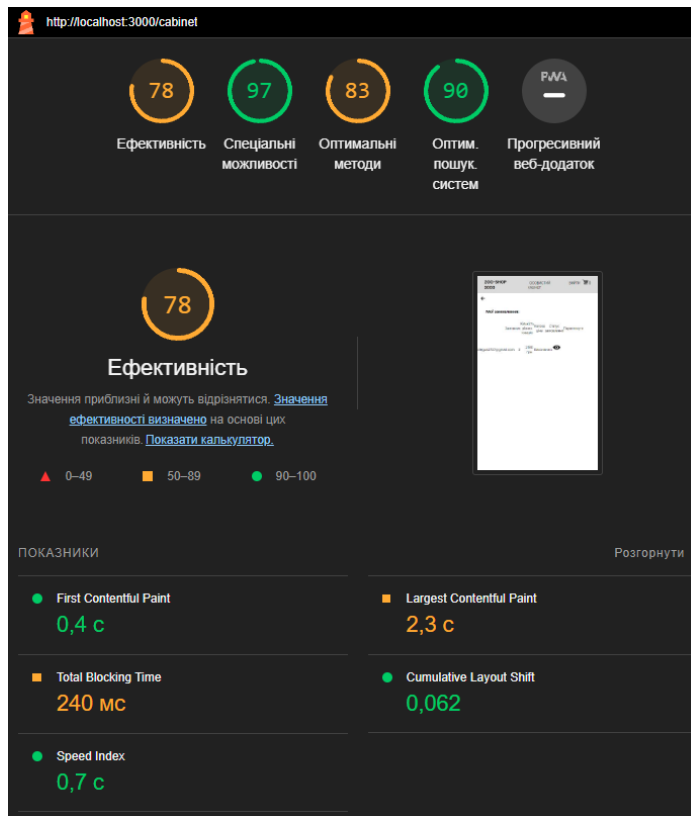


Рис. 3.35. Сторінка особистий кабінет

На рисунку 3.35 показники в межах норми.

Підсумовуючи вище наведені рисунки, було проведено тестування готового програмного продукту та отримано задовільний результат.

Кожна сторінка має чудовий результат з оптимізації для пошукових систем, отже, ймовірність того, що пошукові системи браузерів покажуть розроблений вебдодаток раніше ніж деякі інші сайти за схожою тематикою, більша.

Також, всі сторінки вебдодатку мають задовільний результат доступності, тому взаємодія користувача із сайтом буде на рівні інтуїції.

Висновки до третього розділу

Отже, було представлено та описано діаграму розгортання.

Було описано повний функціонал вебдодатку починаючи із головної сторінки і торкаючись всіх інших сторінок готового проекту.

Також було проведено тести за допомогою Lighthouse в браузері Google Chrom та отримано задовільний результат.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано систему для продажу товарів для тварин.

В ході підготовки до реалізації готового продукту було проаналізовано потреби та поставлено задачу до кваліфікаційної роботи, переглянуто проекти по схожій тематиці та визначено їх основні переваги та недоліки. Обрано та обґрунтовано технології для розробки вебдодатку та описано мінімальні вимоги щодо апаратного забезпечення.

В процесі розробки програмного продукту, було розроблено чотири діаграми, а саме «Діаграма варіантів використання», «Діаграма розгортання», «Діаграма компонентів» та «Діаграма активності». Було описано базу даних та розроблено взаємодію з нею. Було проаналізовано та розроблено основні алгоритми та створено базовий функціонал системи для продажу товарів для тварин.

Коли вебдодаток було завершено було описано функціональні можливості вебдодатку починаючи із головної сторінки і торкаючись всіх інших сторінок готового програмного проекту. Було проведено тестування за допомогою Lighthouse в браузері Google Chrom та отримано задовільний результат. Також було представлено та описано діаграму розгортання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] E-Zoo [Електронний ресурс] // E-Zoo – Режим доступу до ресурсу: https://e-zoo.com.ua/ua?gclid=Cj0KCQjw7aqkBhDPArisAKGa0oLybad7pDGgKdbZWkRzJtui1591PE3NzaayG_1oP7GJQSW80ucmgY8aAkbMEALw_wcB.

[2] MasterZoo [Електронний ресурс] // MasterZoo – Режим доступу до ресурсу: https://masterzoo.ua/ua/?gclid=Cj0KCQjw7aqkBhDPArisAKGa0oJtztiNVftckQaG8nj9AvkdaEh-r9jQE7PUg2vt3rYy2ZAilcvRIImUaAqJfEALw_wcB.

[3] zooshop.in.ua [Електронний ресурс] // zooshop.in.ua – Режим доступу до ресурсу: <https://zooshop.in.ua/ua/>.

[4] Гавкер М'явський [Електронний ресурс] // Гавкер М'явський – Режим доступу до ресурсу: https://gavke.pp.ua/?gclid=Cj0KCQjw7aqkBhDPArisAKGa0oK03lIX-4N-3BCik_h4B-wEGDLNc_GldZz_aCqM0AtA4IOFTajTTJYaAj3WEALw_wcB.

[5] Pathouse [Електронний ресурс] // pathouse – Режим доступу до ресурсу: <https://pethouse.ua/ua/>.

[6] Python 3.13.0a0 documentation [Електронний ресурс] // python.org. – 2023. – Режим доступу до ресурсу: <https://docs.python.org/3.13/tutorial/index.html>.

[7] PHP [Електронний ресурс] // PHP.net. – 2023. – Режим доступу до ресурсу: <https://www.php.net/docs.php>.

[8] JavaScript [Електронний ресурс] // mozilla.org. – 2023. – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

[9] What is MongoDB Atlas? [Електронний ресурс] // MongoDB – Режим доступу до ресурсу: <https://www.mongodb.com/docs/atlas/>.

[10] Mongoose [Електронний ресурс] // Mongoose – Режим доступу до ресурсу: <https://mongoosejs.com/docs/>.

[11] Using middleware [Електронний ресурс] // Express. – 2017. – Режим доступу до ресурсу: <https://expressjs.com/en/guide/using-middleware.html>.

					ІІЗ.КР.Б-121-23-ІІЗ	
						54

[12] Express [Електронний ресурс] // Express. – 2017. – Режим доступу до ресурсу: <https://expressjs.com>.

[13] Next JS [Електронний ресурс] // Vercel. – 2023. – Режим доступу до ресурсу: <https://nextjs.org/docs>.

[14] Nextjs image optimization with examples [Електронний ресурс] // refine. – 2022. – Режим доступу до ресурсу: <https://refine.dev/blog/using-next-image/#introduction>

[15] NextJS Head: Optimize MetaData [Електронний ресурс] // OH MY CRAWL – Режим доступу до ресурсу: <https://www.ohmycrawl.com/nextjs/head/>.

[16] Redux [Електронний ресурс] // Redux. – 2023. – Режим доступу до ресурсу: <https://redux.js.org>.

[17] How to use Redux in Next.js [Електронний ресурс] // LogRocket. – 2023. – Режим доступу до ресурсу: <https://blog.logrocket.com/use-redux-next-js/>.

[18] Cloudinary [Електронний ресурс] // Cloudinary. – 2023. – Режим доступу до ресурсу: <https://cloudinary.com>.

[19] EmailJS [Електронний ресурс] // EmailJS. – 2023. – Режим доступу до ресурсу: <https://www.emailjs.com/docs/>.

[20] react-chatjs-2 [Електронний ресурс] // react-chatjs-2 – Режим доступу до ресурсу: <https://react-chartjs-2.js.org>.

[21] UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] // Evergreens. – 2021. – Режим доступу до ресурсу: <https://evergreens.com.ua/ua/articles/uml-diagrams.html>.

[22] Lighthouse [Електронний ресурс] // Google. – 2022. – Режим доступу до ресурсу: <https://developer.chrome.com/docs/lighthouse/overview/>.

ДОДАТКИ

					ІІЗ.КР.Б-121-23-ІЗ	
						56

Файл stats.js – логіка збору статистики

```
import { useEffect, useState } from 'react';
import { useUserData } from '../../store/hooks/useUserData';
import { useProductData } from '../../store/hooks/useProductsData';
import { useRouter } from 'next/router';
import { useOrders } from "../../store/hooks/useOrders";
import { Chart as ChartJS, ArcElement, Tooltip, Legend } from 'chart.js';
import { Doughnut } from 'react-chartjs-2';
import Head from 'next/head';
import styles from '../../styles/Owner.module.css'

ChartJS.register(ArcElement, Tooltip, Legend);

const StatsPage = () => {
  const { userData } = useUserData()
  const { ordersData, isFetching, error, updateOrder, searchOrder } = useOrders()
  const { data } = useProductData()
  const router = useRouter()
  const [pageData, setPageData] = useState(null)
  const [statusCounts, setStatusCounts] = useState()
  const [allItemIds, setAllItemIds] = useState([]);
  const [uniqueManufacturers, setUniqueManufacturers] = useState([]);
  const [manufacturerCounts, setManufacturerCounts] = useState({});
  const [uniqueTypes, setUniqueTypes] = useState([]);
  const [typeCounts, setTypeCounts] = useState({});

  useEffect(() => {
    if (userData && userData[0] !== null && userData.user?.access !== 2) {
      router.push('/error');
    }
    if (!userData) {
      router.push('/');
    }
  }, [userData])

  useEffect(() => {
    if (ordersData.length > 0) {
      setPageData(ordersData)
    }
  }, [])

  useEffect(() => {
    if (ordersData.length >= 0) {
      setPageData(ordersData)
    }
  }, [ordersData])
}
```

```

const statusCount = {
  "Обробка": 0,
  "Виконання": 0,
  "Відправлено": 0,
  "Відмінено": 0
};

useEffect(() => {
  if (pageData && pageData[0] !== null) {
    const countStatus = pageData.reduce((count, order) => {
      const status = order.status;
      count[status] += 1;
      return count;
    }, statusCount);
    setStatusCounts(Object.values(countStatus))
  }
}, [pageData])

useEffect(() => {
  const updateData = () => {
    const updatedAllItemIds = [];

    pageData?.forEach(item => {
      if (item) {
        updatedAllItemIds.push(...item.orderedGoods);

        item?.orderedGoods.forEach(itemId => {
          const product = data.find(product => product._id === itemId.id);

          if (product && !uniqueManufacturers.includes(product.manufacturer) && !uniqueTypes.includes(product.type)) {
            setUniqueManufacturers(prevManufacturers => [...prevManufacturers, product.manufacturer]);
            setUniqueTypes(prevTypes => [...prevTypes, product.type])
          }
        });
      }
    });

    setAllItemIds(updatedAllItemIds);
  };

  const updateManufacturerAndTypesCounts = () => {
    const countsManufacturers = {};
    const countsTypes = {};

    uniqueManufacturers.forEach(manufacturer => {
      countsManufacturers[manufacturer] = 0;
    });

    uniqueTypes.forEach(type => {
      countsTypes[type] = 0
    });

    pageData?.forEach(item => {
      item?.orderedGoods.forEach(itemId => {
        const product = data.find(product => product._id === itemId.id);

        if (product) {
          countsManufacturers[product.manufacturer] += itemId.quantity;
          countsTypes[product.type] += itemId.quantity;
        }
      });
    });

    setManufacturerCounts(countsManufacturers);
    setTypeCounts(countsTypes)
  };

  updateData();
  updateManufacturerAndTypesCounts();
}, [pageData, data, uniqueManufacturers, uniqueTypes]);

```

Файл useOrders.js – хук для роботи із замовленнями

```
import { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { ordersActions } from '../actions/ordersActions';

export const useOrders = (email = null, id = null) => {
  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(ordersActions.fetchAsync(email, id));
  }, [dispatch, email]);

  const {
    ordersData,
    isFetching,
    error
  } = useSelector((state) => state.orders);

  console.log(ordersData)

  const addOrder = async (data) => {
    try {
      console.log("data in hook", data)
      dispatch(ordersActions.addAsync(data))
      console.log('Замовлення додано');
    }
    catch (error) {
      console.error('Помилка при доданні Замовлення:', error);
    }
  };

  const updateOrder = async (id, data) => {
    try {
      console.log("data", data)
      dispatch(ordersActions.updateAsync(id, data))
      console.log('Замовлення оновлено');
    }
    catch (error) {
      console.error('Помилка при оновленні Замовлення:', error);
    }
  };
};
```

```

const searchOrder = async (data) => {
  try {
    dispatch(ordersActions.searchAsync(data));
  } catch (error) {
    console.error('Помилка при пошуку:', error);
  }
};

return {
  ordersData,
  isFetching,
  error,
  addOrder,
  updateOrder,
  searchOrder
}
}

```

Файл ordersActions.js – дії з API замовлень

```

import { types } from '../types';
import { api } from '../../api/index';

export const ordersActions = Object.freeze({
  startFetching: () => {
    return {
      type: types.ORDER_START_FETCHING
    }
  },
  stopFetching: () => {
    return {
      type: types.ORDER_STOP_FETCHING
    }
  },
  fill: (payload) => {
    return {
      type: types.ORDER_FILL,
      payload
    }
  },
  setFetchingError: (error) => {
    return {
      type: types.ORDER_SET_FETCHING_ERROR,
      error: true,
      payload: error
    }
  },
});

```

```

fetchAsync: (email) => async (dispatch) => {
  dispatch(ordersActions.startFetching());

  let response;
  if (email) {
    response = await api.orders.getOrdersByUser(email);
  } else {
    response = await api.orders.getOrders()
  }
  if (response.status === 200) {
    dispatch(ordersActions.fill(response.data));
  } else {
    const error = {
      status: response.status
    };
    dispatch(ordersActions.setFetchingError(error));
  }
  dispatch(ordersActions.stopFetching());
},
addAsync: (data) => async (dispatch) => {
  dispatch(ordersActions.startFetching());

  try {
    const response = await api.orders.order(data);

    if (response.status === 200) {
      dispatch(ordersActions.fill(response.data));
    } else {
      const error = {
        status: response.status
      };
      dispatch(ordersActions.setFetchingError(error));
    }
  } catch (error) {
    dispatch(ordersActions.setFetchingError(error));
  }

  dispatch(ordersActions.stopFetching());
},
searchAsync: (data) => async (dispatch) => {
  try {
    const response = await api.orders.searchOrders(data);

    if (response.status === 200) {
      dispatch(ordersActions.fill(response.data));
    }
  }
}

```

```

    } else {
      const error = {
        status: response.status
      };
      dispatch(ordersActions.setFetchingError(error));
    }
  } catch (error) {
    dispatch(ordersActions.setFetchingError(error));
  }
},
updateAsync: (id, data) => async (dispatch) => {
  try {
    const response = await api.orders.updateStatus(id, data);

    if (response.status === 200) {
      console.log("cool")
    } else {
      const error = {
        status: response.status
      };
      dispatch(ordersActions.setFetchingError(error));
    }
  } catch (error) {
    dispatch(ordersActions.setFetchingError(error));
  }
}
})

```

Файл Filters.jsx – компонент який утворює динамічний список фільтрів

```

import React, { useState } from 'react';
import styles from "../../styles/Filter.module.css"

const CheckboxList = ({
  manufacturers,
  types,
  setCheckedItemsTypes,
  setCheckedItemsManu,
  checkedItemsManu,
  checkedItemsTypes
}) => {
  const handleChangeManu = (event) => {
    const { name, checked } = event.target;
    if (checked) {
      setCheckedItemsManu((prevItems) => [...prevItems, name]);
    } else {
      setCheckedItemsManu((prevItems) => prevItems.filter((item) => item !== name));
    }
  };
};

```

```

const handleChangeTypes = (event) => {
  const { name, checked } = event.target;
  if (checked) {
    setCheckedItemsTypes((prevItems) => [...prevItems, name]);
  } else {
    setCheckedItemsTypes((prevItems) => prevItems.filter((item) => item !== name));
  }
};

return (
  <div className={styles.container}>
    <h3>Виброніки:</h3>
    {
      manufacturers?.map((item, key) => (
        <label className={styles.input}>
          <input
            type="checkbox"
            name={item}
            checked={checkedItemsManu.includes(item)}
            onChange={handleChangeManu}
          />
          {item}
        </label>
      ))
    }
    <hr />
    <h3>Типи:</h3>
    {
      types?.map((item, key) => (
        <label className={styles.input}>
          <input
            type="checkbox"
            name={item}
            checked={checkedItemsTypes.includes(item)}
            onChange={handleChangeTypes}
          />
          {item}
        </label>
      ))
    }
  </div>
);
};

export default CheckboxList;

```

Файл AllOrders.jsx – компонент який повертає рядок з таблиці замовлень та при зміні статусу відправляє електронне письмо замовнику

```

import React, { useState, useEffect } from 'react';
import Image from 'next/image';
import emailjs from 'emailjs-com';

```

```

import { useRouter } from 'next/router';
import { useForm } from 'react-hook-form';
import styles from '../../styles/AllOrders.module.css';

const AllOrdersComponent = ({ data, updateOrder, showSelect = true }) => {
  const [selectedOption, setSelectedOption] = useState();
  const { register, handleSubmit, setValue } = useForm();
  const router = useRouter()

  const sendEmail = (data, foemData) => {
    emailjs.send(
      'service_lxgyzy',
      'template_ce9ek8q',
      { send_to: data?.userEmail, status: foemData.status },
      'T_jo2y0zXya_3gw67'
    )
      .then((response) => {
        console.log('Email sent successfully:', response);
      })
      .catch((error) => {
        console.error('Error sending email:', error);
      });
  };

  useEffect(() => {
    if (data) {
      setSelectedOption(data.status)
    }
  }, [data])

  const handleChange = (event) => {
    setSelectedOption(event.target.value);
    setValue("status", event.target.value)
  };

  const onSubmit = (formData) => {
    console.log(formData)
    if (selectedOption) {
      updateOrder(data?._id, formData)
      sendEmail(data, formData)
    }
  }

  const handleClick = () => {
    if(showSelect) {
      router.push(`/admin/all-orders/${data?._id}`)
    } else {
      router.push(`/orders/${data?._id}`)
    }
  }

  return (
    <form className={styles.body} key={data?._id} onChange={handleSubmit(onSubmit)}>
      <p>{data?.userEmail}</p>
      <p>{data?.orderedGoods.length}</p>
      <p>{data?.totalPrice} грн</p>
      {

```

```

    {
      showSelect ? (
        <select className={styles.select} name="status"
          value={selectedOption} onChange={handleChange}>
          <option className={styles.option}>Обробка</option>
          <option className={styles.option}>Виконання</option>
          <option className={styles.option}>Відправлено</option>
          <option className={styles.option}>Відмінено</option>
        </select>
      ) : (
        <p>{data?.status}</p>
      )
    }
    <div className={styles.eye} onClick={handleClick}>
      <Image src='/eye.png' height="30" width="35" alt='image' />
    </div>
  </form>
);
};

export default AllOrdersComponent;

```

Файл cart.js – сторінка кошику

```

import Image from 'next/image';
import Link from 'next/link';
import { useCart } from '../store/hooks/useCart';
import styles from '../styles/CartPage.module.css';
import Head from 'next/head';

const CartPage = () => {
  const { cartData, addQuantity, removeQuantity, deleteFromCart, removeAll, getTotalPrice } = useCart()

  console.log("cart", cartData)

  return (
    <div className={styles.container}>
      <Head>
        <title>Zoo-Shop 2000</title>
      </Head>
      {cartData.length === 0 ? (
        <h1>Кошик порожній!</h1>
      ) : (
        <>
          <div className={styles.header}>
            <div>Фото</div>
            <div>Назва</div>
            <div>Ціна</div>
            <div>Кількість</div>
            <div>Дії</div>
            <div>Ітого</div>
          </div>
          {cartData.map((item) => (
            <div className={styles.body} key={item._id}>
              <div className={styles.image}>
                <Image src={item.image} height="90" width="65" alt='image' />
              </div>
              <p>{item.name}</p>
              <p>{item.price} грн</p>
              <p>{item.quantity}</p>
              <div className={styles.buttons}>

```

```

        <button onClick={() => addQuantity(item._id)}>
          +
        </button>
        <button onClick={() => removeQuantity(item._id)}>
          -
        </button>
        <button onClick={() => deleteFromCart(item._id)}>
          x
        </button>
      </div>
      <p>{(item.quantity * item.price).toFixed(2)} грн</p>
    </div>
  </div>
  <h2>Взагалом: {getTotalPrice()} грн</h2>
  <span className={styles.order}>
    <Link href="/order">
      <span>Замовити</span>
    </Link>
  </span>
</>
  )}
</div>
);
};

export default CartPage;

```