МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ЕКОНОМІКО-ПРАВНИЧИЙ ФАХОВИЙ КОЛЕДЖ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ»

Циклова комісія математичних дисциплін та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «РОЗРОБКА СИСТЕМИ ПРОДАЖУ СПОРТИВНИХ ТОВАРІВ»

Виконав: здобувач освіти 4 курсу, групи К121-21
Спеціальність 121 Інженерія програмного забезпечення
(шифр і спеціальність)
Владислав ЧУМАК
(ім'я та ПРІЗВИЩЕ)
Керівник Роман КОТОВ
(ім'я та ПРІЗВИЩЕ)
Рецензент доцент кафедри захисту інформації НУ Запорізька Політехніка, к.т.н., доцент Ганна НЕЛАСА
(посада, науковий ступінь, вчене звання, ім'я та ПРІЗВИЩЕ)

Запоріжжя
2025

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ВСП «Економіко-правничий фаховий коледж ЗНУ»

Освітньо-професійний ступінь фаховий молодший бакалавр

Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ

Голова циклової комісії математичних дисциплін та інформаційних технологій

_____ Тетяна СМОЛЯНКОВА
(підпис)

“ 14 ” червня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ЧУМАКУ Владиславу Миколайовичу
(ПРІЗВИЩЕ ім’я та по–батькові)

1. Тема роботи «Розробка системи продажу спортивних товарів»

Керівник роботи КОТОВ Роман Олександрович
(посада, науковий ступінь, вчене звання, ПРІЗВИЩЕ ім’я по-батькові)

Затверджені наказом ВСП ЕПФК ЗНУ від « 26 » листопада 2024 р. № 1852-с

2. Строк подання студентом роботи 13.06.2025

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

4. Зміст розрахунково– пояснювальної записки (перелік питань, які потрібно розробити)
1.Сучасні інформаційні системи.
2. Розробка проекту інформаційної системи.
3. Програмна реалізація інформаційної системи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація до захисту

6. Консультанти розділів роботи

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

7. Дата видачі завдання 01.12.2024

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи:		
	Постановка задачі	Грудень 2024	виконано
2.	Збір вихідних даних, обробка методичних		
	та теоретичних джерел	Січень 2025	виконано
3.	Розробка першого розділу:		
	Огляд систем продажу та технологій для їхнього створення	Лютий 2025	виконано
4.	Розробка другого розділу:		
	Розробка проєкту системи продажу та вибір технологій	Квітень 2025	виконано
5.	Розробка третього розділу:		
	Розробка застосунку	Травень 2025	виконано
6.	Оформлення і нормоконтроль кваліфікаційної роботи та перевірка на плагіат	Червень 2025	виконано
7.	Захист кваліфікаційної роботи	20.06.2025	виконано

Здобувач освіти _____ (підпис) Владислав ЧУМАК (ім'я ПРІЗВИЩЕ)

Керівник роботи _____ (підпис) Роман КОТОВ (ім'я ПРІЗВИЩЕ)

Нормоконтроль пройдено

Нормоконтролер _____ (підпис) Юлія БОРИСОВСЬКА (ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота: 44 сторінки, 12 ілюстрацій, 13 джерел, 2 додатки.

Об’єкт дослідження – інструменти та засоби розробки системи продажу спортивних товарів.

Предмет дослідження – система продажу спортивних товарів.

Мета роботи – розробка системи продажу спортивних товарів.

У кваліфікаційній роботі розглянуто та проаналізовано сучасні системи продажу, їх приклади та інструменти і засоби для створення систем продажу.

В ході дослідження було розроблено систему продажу спортивних товарів у вигляді вебсайту, створеному з використанням HTML, CSS, JavaScript, React, для інтерфейсу і Node.js, Express для опису серверної частини вебдодатку у середовищі Visual Studio Code.

Розроблений програмний продукт є результатом ретельного дослідження теми кваліфікаційної роботи, опанування засобів створення дизайн-макетів, тривалого та поглибленого вивчення програмування і вебверстки.

Система продажу створена для надання можливості пошуку та придбання спортивних товарів.

В ході виконання кваліфікаційної роботи було досягнуто мети дослідження – створення автоматизованої системи продажу, яка дозволяє переглядати, шукати і замовляти спортивний інвентар та вести облік спортивних товарів та їх категорій.

СИСТЕМА ПРОДАЖУ, ВЕБСАЙТ, HTML, CSS, JAVASCRIPT, REACT, NODE.JS, ДИЗАЙН, СПОРТИВНІ ТОВАРИ.

SUMMARY

Qualification work: 44 pages, 12 illustrations, 13 sources, 2 appendices.

The object of the study is tools and means for developing a sports goods sales system.

The subject of the study is a sports goods sales system.

The purpose of the work is to develop a sports goods sales system.

The qualification work examines and analyzes modern sales systems, their examples, and tools and means for creating sales systems.

During the study, a sports goods sales system was developed in the form of a website created using HTML, CSS, JavaScript, React for the interface and Node.js, Express for describing the server part of the web application in the Visual Studio Code environment.

The developed software product is the result of a thorough study of the topic of the qualification work, mastering the means of creating design layouts, long-term and in-depth study of programming and web layout.

The sales system was created to provide the ability to search and purchase sports goods.

During the qualification work, the research goal was achieved - the creation of an automated sales system that allows you to view, search for and order sports equipment and keep track of sports goods and their categories.

SALES SYSTEM, WEBSITE, HTML, CSS, JAVASCRIPT, REACT, NODE.JS, DESIGN, SPORTING GOODS.

ЗМІСТ

Завдання на кваліфікаційну роботу студенту 2
Реферат 4
Summary 5
Вступ 7
1 Сучасні системи продажу 8
1.1 Поняття системи продажу 8
1.2 Призначення інтернет-магазину 8
1.3 Популярні інтернет-магазини та системи продажу 8
1.4 Складові інтернет-магазину 9
1.5 Функціонал розглянутих аналогів 10
1.6 Інструменти та засоби розробки систем продажу 11
1.7 Висновки до розділу 1 12
2 Розробка проєкту системи продажу спортивних товарів 14
2.1 Технічне завдання 14
2.2 Визначення засобів розробки системи 16
2.3 Етапи створення системи продажів 17
2.4 Створення діаграми прецедентів 17
2.5 Розробка дизайн-макету вебсайту 20
2.6 Висновки до розділу 2 22
3 Програмна реалізація системи 23
3.1 Проєктування та реалізація бази даних 23
3.2 Підготовка проєкту 24
3.3 Клієнтська частина 25
3.4 Серверна частина 33
3.5 Тестування 34
3.6 Завантаження на GitHub 36
3.7 Висновок до розділу 3 37
Висновок 38
Перелік використаних джерел 39
Додаток А 41
Додаток Б 43

ВСТУП

На даний момент великий відсоток придбань здійснюються онлайн. Системи для продажів товарів виконують завдання такі завдання як:

- спрощення процесу придбання товарів;
- спрощення обліку товарів;
- налагодження спілкування між клієнтом та постачальником.

Метою кваліфікаційної роботи є дослідження таких систем та розробка власної системи продажів спортивних товарів.

Об'єктом дослідження є методи розробки систем продажу та обліку товарів.

Кваліфікаційна робота присвячена розробці системи продажу спортивних товарів. Її призначенням є надання користувачам можливості пошуку та замовлення спортивних товарів за допомогою інтерфейсу вебсайту.

В роботі буде розглянуто тему й об'єкт дослідження. Буде визначено поняття систем продажу, їх призначення, описано інструменти їх розробки, а також розглянуто відомі приклади таких систем.

Також буде описано процес розробки проєкту розроблюваної системи продажу. Цей процес включає в себе формування технічного завдання та вимог, визначення інструментів розробки системи відповідно до вимог, створення діаграми прецедентів і створення дизайн-макету.

Останнім етапом буде описано, як створювалась програмна реалізація продукту на основі створеного проєкту системи.

1 СУЧАСНІ СИСТЕМИ ПРОДАЖУ

1.1 Поняття системи продажу

Система продажу – це сукупність методів та інструментів організації процесу торгівлі, що спрямовані на збільшення ефективності продажів та збільшення прибутку, як наслідок [1].

В контексті даної роботи розглядається один з таких інструментів – інтернет-магазин – платформа у мережі Інтернет, яка забезпечує пряму торгівлю постачальника зі споживачем, при цьому замовлення товарів та розміщення інформації споживача відбувається на сайті [2].

1.2 Призначення інтернет-магазину

Інтернет-магазин, як складова системи продажу, призначений для автоматизації та спрощення процесу торгівлі між постачальниками та клієнтами, що сприяє збільшенню її ефективності за рахунок того, що перегляд та пошук товарів в онлайн здійснюється значно швидше, та такі етапи, як оформлення замовлення та оплата товару автоматизовані й не вимагають залучення працівників.

1.3 Популярні інтернет-магазини та системи продажу

Було розглянуто аналоги подібних систем з метою порівняння їх між собою, визначення їх спільних рис та найбільш ефективних методів їх створення.

Для аналізу було обрано, такі відомі в Україні інтернет-магазини, як Rozetka, OLX та SEVN.

Rozetka – раніше команія, що спеціалізувалася лише на продажі електроніки, а нині – один з найбільших в Україні маркетплейсів с великою кількістю товарів з багатьох різних категорій [3].

OLX – маркетплейс, що дозволяє здійснювати торгівлю між користувачами платформи без посередників, де можна придбати як новий товар так і той, що був у використанні раніше [4].

Sevn – інтернет-магазин, який спеціалізується в основному на продажі спортивного обладнання й одягу. Також даний магазин пропонує й асортимент інструментів для дому [5].

При аналізі даних систем продажу було досліджено їх дизайн, вигляд та методи навігації, методи адаптації сайтів під різні браузери та розміри екранів, їх основні сторінки та функціональні можливості.

1.4 Складові інтернет-магазину

Для створення власного інтернет-магазину необхідно визначити його основні складові. Він містить в собі дві основні частини: вебсайт і базу даних.

Для веб-сайту потрібно розробити клієнтську частину сайту і серверну. В даному випадку серверна частина міститиме в собі код для запуску локального сервера і API – набір визначень/протоколів, створених для взаємодії програм між собою [6]. Клієнтська частина в свою чергу складатиметься з трьох основних елементів.

Головна сторінка – візитка магазину, що повинна зацікавити користувача. Дана сторінка є основною, саме її користувач побачить вперше під час відвідування сайту.

Каталог товарів – сторінка, на якій відображається весь каталог товарів. На відміну від головної сторінки, окрім простого перегляду контенту, каталог повинен надавати багато функціоналу для пошуку потрібних товарів за назвою, ціною та категоріями, сортування за рейтингом, ціною і назвою, додавання у кошик.

Кошик користувача – список всіх товарів, які користувач додав до нього при виборі у каталозі й хоче придбати в одному замовленні. Доступ до керування цим списком також надаватиметься за допомогою інтерфейсу, створеного на окремій сторінці. Мінімальним функціоналом такої сторінки буде збільшення і зменшення кількості певного продукту, видалення товарів зі списку, перегляд загальної ціни кошику й оплата, яка здійснюється натисканням кнопки.

Другою складовою є база даних – місце, в якому зберігається структурований набір інформації про всі товари, категорії товарів, про користувачів, їх замовлення та повідомлення, доступний для обробки, пошуку та аналізу [7].

1.5 Функціонал розглянутих аналогів

Було розглянуто пропонований існуючими системами продажу функціонал. Дані системи дозволяють користувачеві використовувати зручну навігацію для переміщення сторінками сайту, переглядати й шукати товари в каталозі, додавати товари до кошику та сплачувати їх, збирати інформацію про компанію, зокрема її контактну інформацію, таку як номери телефонів для зворотного зв’язку.

Особливу увагу було приділено функціоналу каталогу, як основній частині сайту. Окрім перегляду товарів він надає можливість шукати товари за

назвою, використовуючи поле пошуку, сортувати товари за декількома параметрами. Також у таких системах зазвичай є багато категорій, декілька з яких можна обрати, щоб шукати продукти тільки в межах обраних категорій.

Окрім цього було розглянуто методи навігації вебсайтів, оскільки вона повинна бути простою та інтуїтивно зрозумілою для користувачів. Навігація представлена в шапці сайту і в окремому меню, яке відкривається натисненням кнопки. Частіше за все такі меню використовуються для більш зручної навігації на мобільних пристроях з невеликим розміром екрану.

1.6 Інструменти та засоби розробки систем продажу

Далі необхідно було визначити, які інструменти використовуються для розробки таких систем.

Середовище розробки – програмне забезпечення, в якому буде вся файлова структура та код проєкту. Найпопулярнішим на даний момент середовищем розробки є Visual Studio Code, оскільки є повністю безплатним і пропонує великий функціонал. Окрім редактору коду це середовище пропонує свій термінал, зручний інтерфейс для відлагодження (дебаггінгу) коду, можливість встановлення будь-яких плагінів. Аналогами VS Code є Sublime Text, Adobe Brackets і Web Storm. Останній з них також вартий уваги, оскільки пропонує схожий з VS Code різноманітний функціонал та вбудовані інструменти й розширення.

Необхідними для розробки вебсайтів є HTML і CSS, без яких зараз не створюється жодний сайт.

HTML – гіпертекстова мова розмітки, основа для створення верстки веб-сайтів [8].

CSS – каскадна таблиця стилів, або ж мова стилів, призначена для позиціонування елементів та для стилізації веб-сайту й збільшення його привабливості та читабельності, як результат [9].

Таким же необхідним для створення клієнтської частини є JavaScript, який в основному забезпечує обробку подій в інтерфейсі. Для систем продажу JavaScript потрібен для реалізації таких дій як сортування й фільтрування списку товарів, додавання їх до кошику, надсилання запитів до бази даних, збереження даних про кошик і користувача в пам’яті браузера.

Разом з JS для реалізації користувацьких інтерфейсів використовується бібліотека React або такі фреймворки як Vue.js або Angular. Кожен з них призначений для зрозумілішої структури й значно кращої масштабованості проєкту та для спрощення процесу розробки сайту [10].

JavaScript також може бути використаний і для написання серверної частини програмного продукту завдяки Node.js – середовищі для запуску JavaScript-коду, призначеному для створення серверів та вебдодатків [11]. Разом з Node.js часто використовується фреймворк Express. Аналогом такого поєднання для розробки серверної частини є мова програмування Python з фреймворком Django.

1.7 Висновки до розділу 1

Перший розділ кваліфікаційної роботи був присвячений збору й дослідженню теоретичного матеріалу про сучасні системи продажу шляхом аналізу відкритих джерел і вже готових подібних систем.

Для досягнення мети даного розділу в першу чергу було дано визначення системі продажу й описано їх призначення.

Наступним кроком був розгляд відомих в Україні систем продажу з метою дослідження їх інтерфейсу, функціоналу й основних складових, порівняння між собою та визначення найбільш ефективних методів розробки таких систем.

Останнім етапом був аналіз та опис зібраної під час дослідження аналогів інформації.

Таким чином було чітко визначено поняття системи продажу, проаналізовано їх складові, структуру й функціонал, було досліджено найефективніші методи та інструменти розробки даних систем. Даний етап став корисним для подальшого визначення технічних вимог й інструментів для розробки власної системи продажу.

2 РОЗРОБКА ПРОЄКТУ СИСТЕМИ ПРОДАЖУ СПОРТИВНИХ ТОВАРІВ

2.1 Технічне завдання

2.1.1 Найменування продукту та область застосування

Найменування продукту розробки: “Система продажу спортивних товарів”.

Продукт призначений для обліку та продажу спортивного інвентарю через вебсайт.

2.1.2 Підстава для розробки

Програмний продукт було розроблено на підставі наказу від “Про затвердження тем кваліфікаційних робіт студентів 4 курсу освітньо-професійного ступеня фаховий молодший бакалавр денної форми здобуття освіти ВСП «Економіко-правничий фаховий коледж ЗНУ» від 26 листопада 2024 року.

2.1.3 Призначення розробки

Призначенням розробки даної системи є створення вебсайту, який надає основну інформацію про компанію, заохочуючи людей до придбання

спортивного обладнання у неї, демонструє весь перелік інвентарю з можливістю швидкого пошуку потрібних товарів, а також має спосіб для зв’язку з компанією.

Після опису призначення розробки проєкту набагато простішою стає задача визначення технічних вимог до нього.

2.1.4 Технічні вимоги

Після визначення призначення системи було описано технічні вимоги до неї.

Сайт повинен складатися з головної сторінки, контактної форми, каталогу, кошику та форми для реєстрації. Всі перелічені частини мають бути оформлені у вигляді окремих сторінок.

Дизайн сайту повинен бути не перенасиченим, а достатньо простим та приємним для користувача. Кольорова схема має складатись з двох кольорів – білого і червоного.

Необхідно забезпечити просту й інтуїтивно зрозумілу навігацію між сторінками сайту.

Інтерфейс розроблюваного вебдодатку має коректно відображатись та бути однаково зручним на моніторі будь-якого розміру. Для цього потрібно зробити правильну стилізацію мовою CSS, а також запровадити окремі інструменти для пристроїв з невеликим розміром екрану. Наприклад, навігацію на мобільних пристроях перенести з шапки сайту в окреме меню.

В каталозі повинна відображатись інформація про назву, ціну та рейтинг кожного товару. Окрім даної інформації картки товарів повинні містити кнопку для додавання товарів у кошик.

Необхідно забезпечити можливість прискорювати пошук товарів користувачем за рахунок поля пошуку за назвою, вибору категорій, цінового діапазону та параметрів сортування у каталозі.

Сторінка кошику повинна відображати список доданих туди товарів, інформацію про кожен з них і загальну ціну кошику. Також повинна бути реалізована функція зміни кількості кожного товару. Після списку необхідно розмістити кнопку для оплати.

Завдяки чітко визначеному функціоналу системи стає значно зрозумілішим об’єм виконуваної роботи та перелік необхідних для її виконання інструментів.

2.2 Визначення засобів розробки системи

Розробляти систему було вирішено у середовищі розробки Visual Studio Code.

Для розробки клієнтської частини вебсайту є обов’язковими такі технології як гіпертекстова мова розмітки HTML, каскадна таблиця стилів CSS, мова програмування JS.

Також було вирішено використати бібліотеку для розробки користувацьких інтерфейсів React.js та препроцесор для CSS – Sass.

Для реалізації серверної частини для зв’язку з базою даних було вирішено використати Node.js з фреймворком Express.js та встановити бібліотеку Axios для спрощеного синтаксису написання запитів до серверу та бази даних.

Базу даних було створено за допомогою не реляційної системи керування базами даних MongoDB.

2.3 Етапи створення системи продажів

Для успішної розробки даного програмного продукту було визначено наступні етапи її створення:

1. Визначення мети створення програмного продукту;
2. Формування технічного завдання;
3. Формування вимог до системи;
4. Створення дизайн-макету користувацького інтерфейсу;
5. Визначення інструментів для розробки проекту;
6. Опис проєкту системи;
7. Проєктування та реалізація бази даних;
8. Створення front-end частини сайту;
9. Створення back-end – серверної частини;
10. Тестування програмного продукту;
11. Виправлення помилок та оптимізація;
12. Завантаження продукту на GitHub.

Таким чином було описано послідовність етапів розробки проєкту, після виконання якої буде досягнуто мети кваліфікаційної роботи, якою є розробка системи продажу спортивних товарів.

2.4 Створення діаграми прецедентів

Для наочного відображення сценаріїв використання програмного продукту було створено діаграму прецедентів. Перед створенням діаграми потрібно було визначити перелік акторів та прецедентів.

Перелік акторів:

- Користувач (покупець);

- Адміністратор.

Перелік прецедентів:

- Реєстрація на сайті;
- Перегляд головної сторінки сайту;
- Надсилання повідомлення через контактну форму;
- Перегляд товарів;
- Вибір категорій товарів;
- Визначення найменшої та найвищої ціни товару;
- Пошук за назвою;
- Додавання товарів у корзину;
- Змінення кількості товарів при перегляді корзини;
- Видалення товарів з корзини;
- Оплата корзини;
- Додавання товарів у БД;
- Додавання категорій у БД.

На основі даного переліку акторів та прецедентів було створено діаграму за допомогою інструменту draw.io (рисунок 2.1).



Рисунок 2.1 – Діаграма прецедентів

Таким чином було описано функціонал створюваної системи продажу й зображено його у схематичному вигляді, що значно спрощує сприйняття та розуміння того, раніше описаних вимог до системи.

2.5 Розробка дизайн-макету вебсайту

Макет сайту було створено у вебдодатку Mockflow в 2 етапи:

1. Створення Wireframe-макету;

Wireframe-макет, по-суті, є каркасом, на якому потім розроблявся сам дизайн вебсайту. На такому макеті все зображено в загальних рисах, без кольорової схеми, зображень і т.п. Так на сторінці магазину на рисунку 2.2 зображено повністю структуру сторінки, навігацію, всі потрібні для роботи каталогу кнопки, але немає жодних зображень, кольорів, а також конкретних назв, рейтингів або цін товарів.

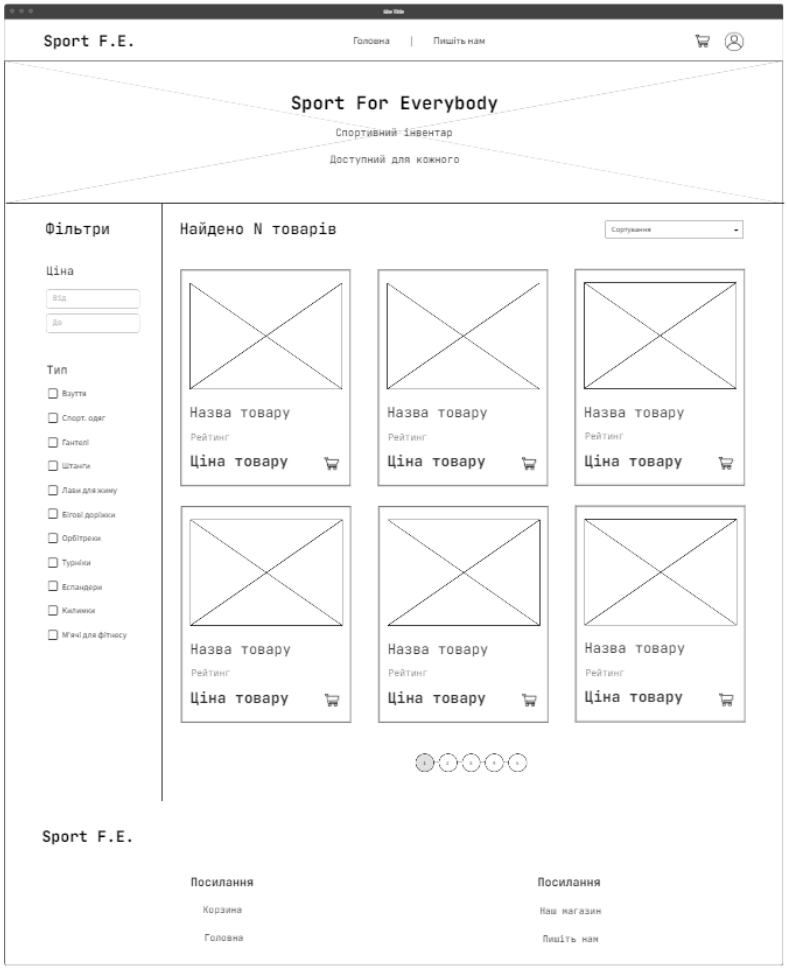


Рисунок 2.2 – Сторінка магазину (Wireframe)

2. Створення дизайн-макету.

На основі попередньої версії макету було розроблено повноцінний дизайн веб-сайту шляхом додавання кольорової схеми і зображень. Результатом розробки став дизайн-макет, частина якого зображена на рисунку 2.3. На ньому видно, що було додано конкретні зображення та значення до карток товарів. Деякі товари з макету було пізніше додано в базу даних системи.

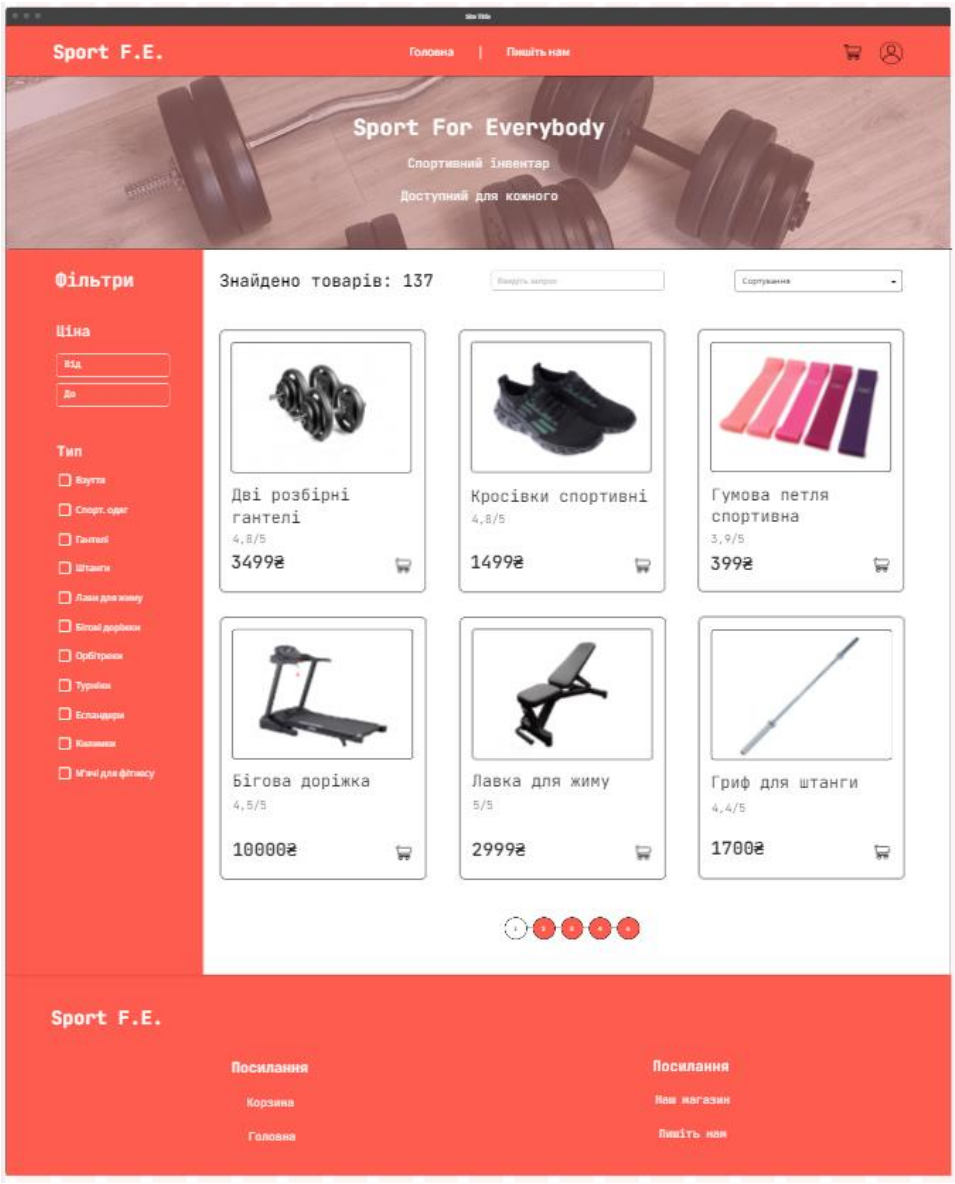


Рисунок 2.3 – Сторінка магазину

2.6 Висновки до розділу 2

Під час написання даного розділу було розроблено проєкт системи продажів спортивних товарів. Розробка проєкту була реалізована в декілька етапів.

Спершу було сформовано технічне завдання. В рамках даного етапу було визначено найменування продукту, призначення розробки, вимоги до нього. Формування вимог до проєкту допомогло визначити потрібний функціонал вебдодатку та, як наслідок, розробити решту проєкту системи.

Наступним етапом, відповідно до сформованих вимог, було підібрано перелік інструментів, необхідних для реалізації всіх вимог до даного програмного продукту.

Далі було визначено етапи розробки програмної реалізації проєкту. Перелік етапів містив основні дванадцять кроків від самого початку розробки проєкту системи до завантаження повністю готової програмної реалізації на GitHub.

Перед останнім етапом було визначено перелік акторів і прецедентів, після чого створено діаграму прецедентів. Така діаграма дозволила схематично зобразити всі сценарії використання програмного продукту та його основний функціонал, що стало дуже простим і зручним для розуміння і сприйняття.

Для завершення повної картини створюваного продукту було розроблено макет вебсайту. Спочатку було створено Wireframe-макет, на основі якого було реалізовано фінальний дизайн-макет сайту.

В результаті розробки проєкту системи продажу спортивних товарів було описано вимоги до продукту, його функціонал, основний функціонал і етапи розробки та створено дизайн-макет всіх сторінок вебсайту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Проєктування та реалізація бази даних

В системі керування БД MongoDB була створена база даних Shop з п’ятьма колекціями:

1. Products – список товарів, доступних в інтернет-магазині.
2. Orders – замовлення, зроблені користувачами.
3. Reports – повідомлення, надіслані користувачами.
4. Users – список користувачів.
5. Categories – список категорій товарів.

Колекції Products та Categories одразу було наповнено деякою кількістю товарів і категорій. Вигляд заповненої колекції Products можна побачити на рисунку 3.1.

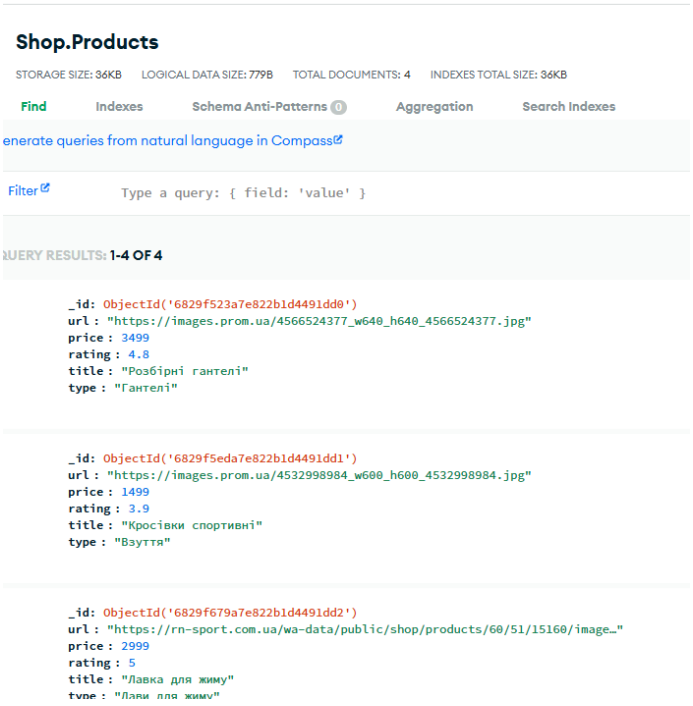


Рисунок 3.1 – Заповнення колекції Products

3.2 Підготовка проєкту

Перед розробкою веб-сайту було необхідно підготувати проєкт до роботи. Для цього було виконано декілька етапів, описаних нижче.

Створення папки проекту. На ПК було створено директорію, яка міститиме в собі всі файли вебсайту.

Ініціалізація проєкту на React. Для цього була використана така команда: npx create-react-app .

Створення віддаленого репозиторію на GitHub. Локальний репозиторій було створено автоматично в результаті виконання попереднього етапу. Далі було створено віддалений репозиторій, який знадобиться вже при закінченні роботи на проєктом для надання його на перевірку. На рисунку 3.2 зображено вигляд вікна для створення репозиторію.

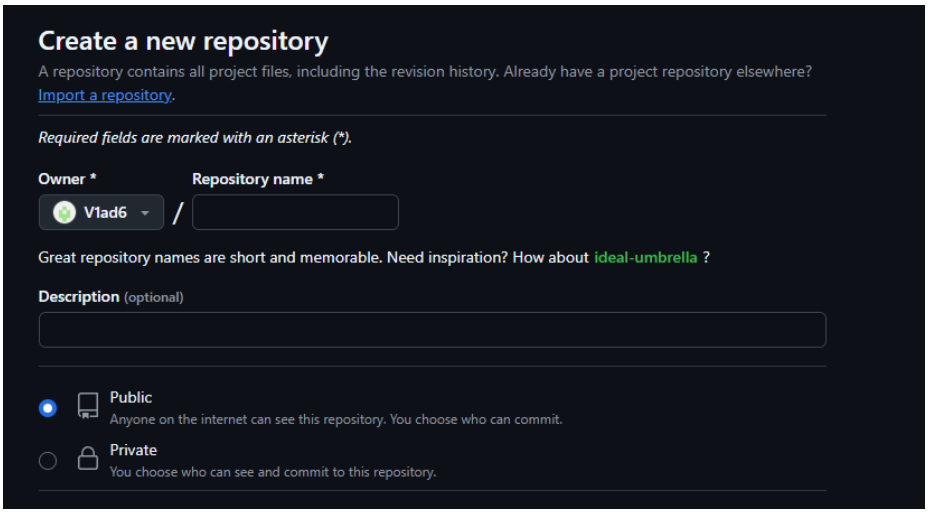


Рисунок 3.2 – Процес створення репозиторію на GitHub

З’єднання локального репозиторію і віддаленого. Після створення обох репозиторіїв було потрібно забезпечити зв’язок між ними для можливості передавати файли з одного на інший. Для цього було прописано наступні команди:

```
git add .
git commit –m “Start of project development”
git remote add origin https://github.com/V1ad6/diploma.git
git push -u origin master;
```

Завантаження sass. Хоч і завантаження препроцесору sass не було обов’язкових, він є надзвичайно корисним для розширення можливостей звичайного CSS та для спрощення написання стилів. Цей етап було виконано за допомогою такої команди: npm i sass.

Видалення зайвих файлів з файлової структури проєкту. Після завантаження всіх бібліотек та створення всієї файлової структури потрібно було очистити проєкт від зайвих файлів, створених автоматично.

3.3 Клієнтська частина

3.3.1 Створення сторінок сайту

На рисунку 3.3 зображено файли всіх сторінок вебсайту. Серед них головна й контактна сторінки, сторінка для реєстрації та входу в акаунт, каталог і кошик. Далі для кожної з цих сторінок буде своя адреса, за якою буде здійснюватись перехід до кожної з цих сторінок.

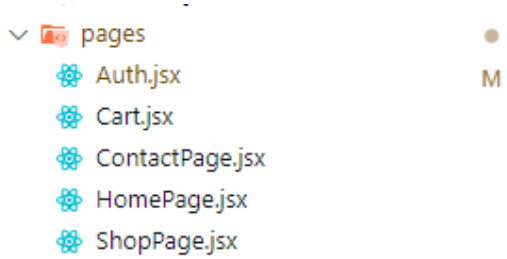


Рисунок 3.3 – Сторінки сайту

3.3.2 Запровадження можливості навігації між сторінками в межах проєкту.

Для цього потрібно було з’єднати всі сторінки в кореневому файлі вебдодатку (App.js).

Так в уривку коду, наведеному нижче видно, як в змінній router прописані шляхи до кожної сторінки сайту.

```
const router = createBrowserRouter([
        { path: '/', element: <HomePage/> },
        { path: '/contact', element: <ContactPage/> },
        { path: '/shop', element: <ShopPage/> },
        { path: '/cart', element: <Cart/> },
        { path: '/auth', element: <Auth/> }
]);
```

А на наступному коді змінна router використовується для того, щоб всі ці сторінки записати в компонент RouterProvider, який дозволяє відвідувати сторінки за описаними раніше шляхами.

```
<GlobalContext.Provider value={{
            items: items,
            setItems: setItems,
            cart: cart,
            setCart: setCart,
}}>
            <div className="App">
              <RouterProvider router={router}/>
            </div>
</GlobalContext.Provider>
```

3.3.3 Створення окремих компонентів

Деякі компоненти було розроблено як окремі модулі у власних файлах. Більшість з них призначено для того, щоб використовувати їх багато разів та не переписувати кожного разу заново:

- Header – шапка сайту;
- Footer – футер;
- Menu – меню для мобільних пристроїв.

Інші компоненти розроблені для декомпозиції, тобто для розбиття коду на частини для кращої читабельності, зрозумілої структури, та уникнення написання великих об’ємів коду в одному файлі:

- Product – картка товару в каталозі на сторінці магазину (код компоненту зображено нижче);
- CartProduct – відображення кожного товару в кошику;
- Filters – інструмент для фільтрування товарів для швидшого пошуку в магазині.

Далі наведено частину коду компоненту Product. В ньому замість статичних значень використовуються змінні, записані у фігурних дужках (наприклад “{title}”), щоб передавати назву, рейтинг, ціну й інші характеристики в залежності від того, який товар потрібно відобразити в картці товару в каталозі.

```
<div className="product">
            <div className="product__image">
              <img src={url} alt="product_image"/>
            </div>
            <div className="product__info">
              <h2 className="product__name">{title}</h2>
              <h4 className="product__rating">{rating}/5</h4>
            </div>
            <div className="product__purchase">
              <h2 className="product__price">{price}₴</h2>
              <button className="product__add" onClick={() =>
        addToCart(id)} disabled={cart.some(item => item[0] === id)}>
                <img src={require("../assets/cart2.jpeg")} alt=""/>
              </button>
            </div>
  </div>
```

3.3.4 Написання стилів для сторінок/компонентів

Під час розробки клієнтської частини сайту також було описано стилі всіх створених сторінок та компонентів. Цей крок в розробці був необхідним для того, щоб сайт відповідав створеному на етапі проєктування дизайн-макету.

Більша частина робота зі стилями полягала в описі кольорової схеми проєкту, визначенні кольорів фону, елементів, тексту, кнопок, визначення стилю і розміру шрифту, розмірів елементів і зображень.

Також було необхідно застосувати знання та навички роботи з Flexbox – методом верстки для розташування елементів у вигляді рядків і колонок [12]. Незважаючи на те, що порядок елементів у потоці в основному визначається HTML-версткою, без CSS стилів було б неможливим розташувати всі елементи саме так, як це зображено на дизайн-макеті. Так за допомогою властивості display зі значенням flex було розташовано список товарів праворуч від списку категорій і поля для вибору цінового діапазону, а не під ними, як це було би без стилізації.

Також окрему увагу варто було приділити медіа-запитам – засобу для адаптації інтерфейсу під різні розміри екрану. Так, у вище описаному випадку, список товарів переміщується під опції вибору категорій, коли ширина екрану стає замалою. Також товари розміщуються по три в кожному рядку, але зі зменшенням розміру монітору ця кількість знижується до двох або до одного, оскільки, наприклад, на мобільному телефоні неможливо вмістити 2-3 товари поруч в одному рядку так, щоб їх зображення та назва не були занадто малими для того, щоб їх розпізнати або прочитати.

3.3.5 Розробка параметрів сортування/фільтрування товарів на сторінці каталогу

За фільтрування та сортування товарів у каталозі магазину відповідає логіка, описана безпосередньо на сторінці магазину та в компоненті Filters.jsx.

У коді, наведеному нижче, реалізовано функцію для сортування товарів за різними параметрами, кожен з яких описаний у конструкції switch/case.

```
function sortProducts(item1, item2) {
            switch (sortMethod) {
              case "lowest-price":
                return item1.price > item2.price ? 1 : -1;
              case "highest-price":
                return item1.price < item2.price ? 1 : -1;
              case "rating":
                return item1.rating < item2.rating ? 1 : -1;
              case "name":
                return item1.title.toLowerCase().trim() >
        item2.title.toLowerCase().trim() ? 1 : -1
              default:
                return 1;
            }
 }
```

Далі дана функція використовуватиметься для того, щоб сортувати товари в залежності від значення, обраного в dropdown списку на сторінці магазину, який надає перелік можливостей сортування товарів. Вигляд даного списку на сайті зображено на рисунку 3.4.

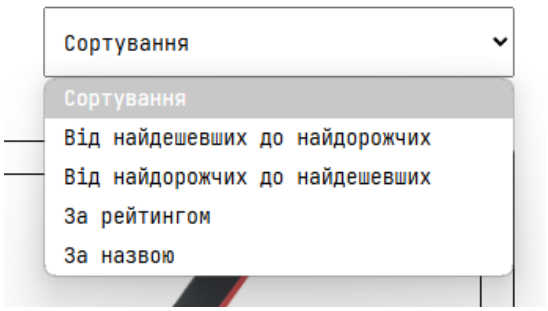


Рисунок 3.4 – Список варіантів сортування товарів

Код компоненту Filters міститься в додатку А. У компоненті містяться поля для регулювання нижньої та верхньої меж ціни шуканих товарів. Також там описано запит до серверу, який збирає всі існуючі категорії спортивного інвентарю з бази даних, які потім відображаються у компоненті й використовуються для фільтрування списку товарів за категоріями.

3.3.6 Додавання товарів до корзини

Для додавання товарів у корзину було створено кнопку на картці товару, яка далі використовую функцію для додавання товару в масив, що далі використовуватиметься для відображення змісту кошику на екрані. На рисунках 3.5 зображено вигляд даної кнопки. Вона розташовується в правому нижньому куті картки товару.



Рисунок 3.5 – Кнопка додавання товару до кошику

3.3.7 Авторизація

У вигляді окремої сторінки було реалізовано реєстрацію на сайті та вхід у створений обліковий запис. Для цього було створено файл auth.js для опису функцій реєстрації та входу, а також валідації форми.

Функція реєстрації використовує функцію валідації для перевірки правильності вказання даних. У випадку успішного проходження перевірки

ім’я, пошта й зашифрований пароль користувача надсилаються до БД і записуються в таблицю Users. Частину коду реєстрації наведено нижче.

```
const encryptedPass =
CryptoJS.AES.encrypt(e.target.elements.password.value,
process.env.REACT_APP_HASH_KEY).toString();
axios.post('http://localhost:3001/addUser', {
        username: e.target.elements.username.value,
        email: e.target.elements.mail.value,
        password: encryptedPass,
});
```

Функція входу шукає користувача за введеними іменем та поштою. Якщо користувача з такими даними не знайдено, то про це виводиться відповідне повідомлення. У випадку знаходження користувача в базі даних відбувається перевірка на вірність введеного паролю.

Шифрування і дешифрування паролю було реалізовано за допомогою бібліотеки crypto-js.

На рисунку 3.6 зображено вигляд створеної форми авторизації.

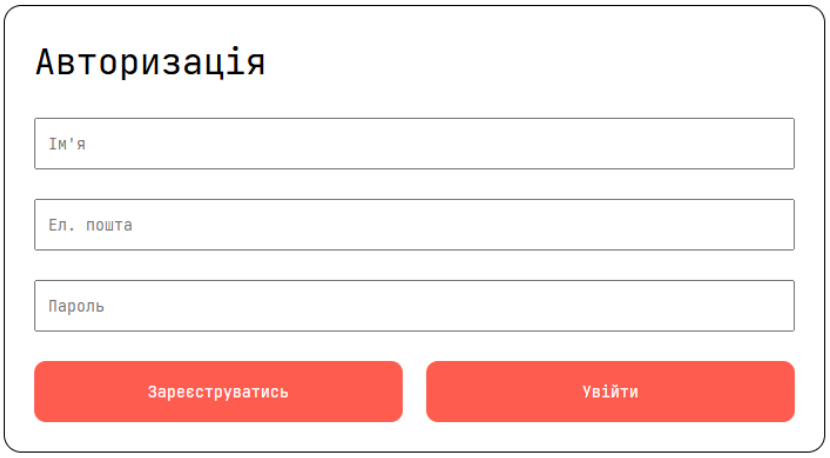


Рисунок 3.6 – Форма авторизації

3.4 Серверна частина

Серверна частина додатку, була реалізована повністю мовою JavaScript на Node.js з використанням фреймворку Express. Для її реалізації потрібно було 2 файли: server.js та models.js.

Server.js є основним файлом, який створює та запускає локальний сервер та описує API для взаємодії з базою даних на MongoDB.

Додаток Б містить реалізацію API ендпоінтів у файлі server.js для отримання даних про товари і категорії, а також для надсилання даних про замовлення та відгуки користувачів з контактної форми.

Ендпоінти в API – URL-адреси, які є точками взаємодії між двома між двома API [13].

В коді server.js є звернення до моделей (productModel, categoryModel, orderModel, reportModel). Це моделі даних, які додаток запитує у БД або надсилає до неї. Всі вони реалізовані у файлі model.js. Схему даних для товарів описано в коді, наведеному нижче.

```
const ProductSchema = new mongoose.Schema({
  _id: mongoose.Schema.Types.ObjectId,
  url: String,
  price: Number,
  rating: Number,
  title: String,
  type: String
});
export const productModel = mongoose.model('products', ProductSchema, 'Products');
```

3.5 Тестування

Для тестування було вручну перевірено роботу функціоналу проєкту, використано інструменти розробника в браузері Google Chrome для перевірки повідомлень в консолі та перевірки адаптивності верстки, а також проєкт було запущено в інших браузерах для перевірки кросплатформеності сайту.

На рисунку 3.7 продемонстровано вигляд сторінки магазину на планшеті моделі Ipad Air.

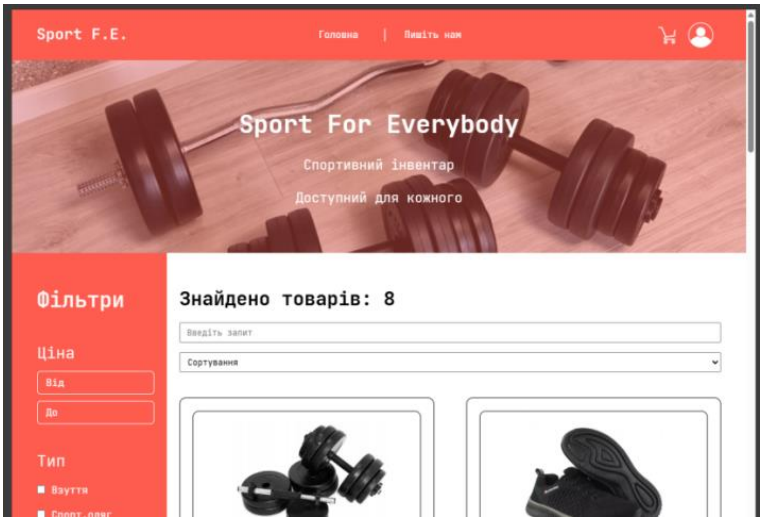


Рисунок 3.7 – Вигляд сайту на Ipad Air

Також було необхідно було перевірити коректність відображення сайту на маленьких екранах, оскільки більшість відвідувачів будуть переглядати сайт саме з телефону. На рисунку 3.8 зображено вигляд сайту на Samsung Galaxy S9. Можна побачити, як замість навігації в шапці сайту додається кнопка зверху праворуч для відкриття меню, яке і містить в собі дану навігацію, оскільки з такою шириною екрану неможливо вмістити посилання на інші сторінки.



Рисунок 3.8 – Вигляд сайту на екрані смартфону

Кросплатформеність сайту було перевірено за допомогою перегляду сайту в браузерах Google Chrome, Microsoft Edge і Mozilla Firefox. Так на рисунку 3.9 зображено вигляд сторінки в браузері Microsoft Edge.

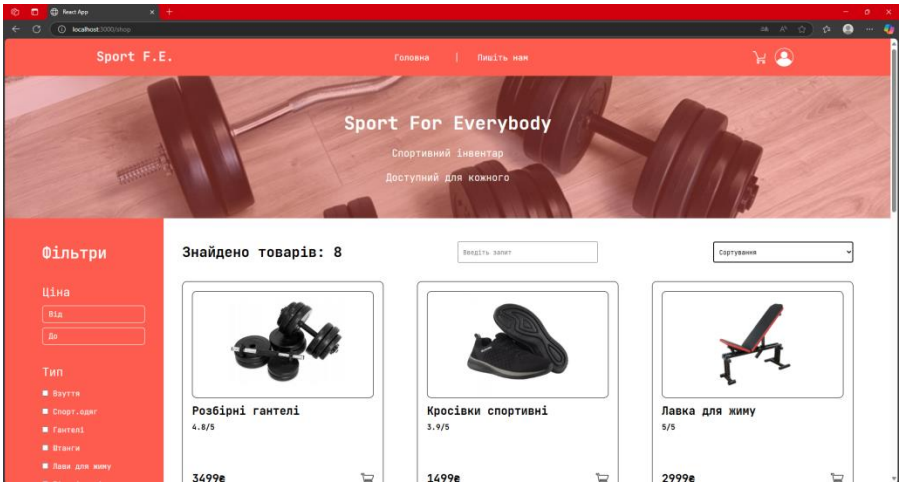


Рисунок 3.9 – Вигляд сайту в браузері Microsoft Edge

3.6 Завантаження на GitHub

Після останніх змін в проекті останнім етапом було завантаження проєкту на GitHub.

Спершу було підготовлено всі змінені файли до комміту (до збереження нової версії проєкту). Для цього було використано команду git add, де замість шляхів до директорій, які потрібно зберігти, вказано крапку для того, щоб зберегти одразу всі зміни в проєкті: *git add .*

Після цього збережено нову версію проєкту – зроблено комміт шляхом виконання команди git commit з параметром –m “Ready cart”, щоб надати цій версії проєкту свою назву: *git commit –m “Ready cart”;*

Останнім кроком було завантаження даної версії проєкту до репозиторію на GitHub: *git push.*

3.7 Висновок до розділу 3

Останній розділ був присвячений програмній реалізації системи продажу спортивних товарів.

Перед початком розробки вебсайту було створено нереляційну базу даних у MongoDB та всі колекції в ній, дві з яких було заповнено даними.

Для того, щоб почати створювати вебсайт, було підготовлено проєкт у VS Code до роботи. В рамках підготовки було завантажено шаблон React-проєкту, створено репозиторій на GitHub, встановлено препроцесор Sass і очищено файлову структуру від зайвих файлів.

Далі було розроблено інтерфейс вебсайту. Було реалізовано верстку й стилізацію сайту. Одним із завершальних етапів стилізації була адаптація інтерфейсу під різні розміри екрану. Було реалізовано функціонал каталогу і кошику на JavaScript та написано запити до серверної частини.

Було створено серверну частину сайту, яка завантажує локальний сервер і реалізує запити для зв’язку з базою даних.

Після основної частини розробки було проведення тестування та оптимізацію проєкту. Під час тестування було перевірено коректність роботи каталогу і кошику, адаптивність і кросплатформеність сайту.

Так було розроблено базу даних зі спортивними товарами та повноцінний вебсайт з користувацьким інтерфейсом і серверною частиною для обміну інформацією з БД.

ВИСНОВОК

В ході виконання кваліфікаційної роботи було досліджено тему кваліфікаційної роботи та об’єкт дослідження. Так було визначено поняття системи продажу та її призначення, розглянуто приклади сучасних систем продажу відомих в Україні. Було описано потрібні інструменти та засоби розробки для створення таких систем.

Перед програмною реалізацією було розроблено проєкт створюваної системи продажу. В рамках розробки було описано технічні вимоги та завдання, обрано середовище та інструменти розробки, а також створено дизайн-макет вебсайту.

Було розроблено програмну реалізацію системи у вигляді веб-сайту. Для цього було створено клієнтську частину сайту та сервер, створено базу даних та наповнено її даними про товари та їх категорії.

В результаті тестування розробленого програмного продукту було визначено, що програмна реалізація повністю відповідає всім вимогам, визначеним під час опису проєкту системи, після чого вебсайт було завантажено на віддалений репозиторій на GitHub.

Таким чином було проведено дослідження теми кваліфікаційної роботи та було виконано мету роботи - розробка власної системи продажу спортивних товарів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Система продажів: визначення та етапи створення – Fractus. URL: https://fractus.com.ua/uk/blog/korysni-statti/prodazhi/yak-pobuduvati-sistemu-prodazhiv/ (дата звернення: 20.05.2025).

2. Інтернет-магазин. URL: https://uk.wikipedia.org/wiki/Інтернет-магазин (дата звернення: 20.05.2025).

3. Розетка (інтернет-магазин). URL: https://uk.wikipedia.org/wiki/Розетка_(інтернет-магазин) (дата звернення: 20.05.2025).

4. Про нас. URL: https://blog.olx.ua/pro-nas-uk/ (дата звернення: 20.05.2025).

5. Хто ми і що ми робимо. URL: https://sevn.com.ua/about/ (дата звернення: 20.05.2025).

6. Прикладний програмний інтерфейс. URL: https://uk.wikipedia.org/wiki/Прикладний_програмний_інтерфейс (дата звернення: 20.05.2025).

7. Що таке база даних? Визначення, типи, особливості. URL: https://university.sigma.software/what-is-database/ (дата звернення: 20.05.2025).

8. HTML Підручник - W3Schools українською - GitHub Pages. URL: https://w3schoolsua.github.io/html/index.html#gsc.tab=0 (дата звернення: 20.05.2025).

9. CSS Підручник - W3Schools українською - GitHub Pages. URL: https://w3schoolsua.github.io/css/index.html#gsc.tab=0 (дата звернення: 20.05.2025).

10. Фреймворки у мовах програмування: навіщо служать та як їх вибрати. URL: https://cloud.itstep.org/blog/frameworks-in-programming-languages-what-are-they-for-and-how-to-choose-them#4 (дата звернення: 20.05.2025).

11. Node.js — Run JavaScript Everywhere. URL: https://nodejs.org/en (дата звернення: 20.05.2025).

12. CSS Flexbox (Flexible Box Layout). URL: https://www.w3schools.com/css/css3_flexbox.asp (дата звернення: 22.05.2025).

13. What is an API endpoint? URL: https://blog.postman.com/what-is-an-api-endpoint/ (дата звернення: 22.05.2025).

ДОДАТОК А

Лістинг коду компоненту Filters

```
function Filters({ prices, setPrices, types, setTypes }) {
  const [categories, setCategories] = useState([]);

  useEffect(() => {
    axios.get("http://localhost:3001/getCategories")
        .then(categories => setCategories(categories.data));
  }, []);

  return (
    <div className="products__filters">
      <h2>Фільтри</h2>

      <form action="" className="product__price-range">
        <h3>Ціна</h3>
        <input
          type="number"
          placeholder='Від'
          value={prices[0]}
          onChange={(e) => setPrices([e.target.value, prices[1]])}
        />
        <input
          type="number"
          placeholder='До'
```

```
        value={prices[1]}
        onChange={(e) => setPrices([prices[0], e.target.value])}
      />
    </form>

    <form
      action=""
      className="product__types"
      onChange={(e) => {
        if (types.includes(e.target.value)) {
          setTypes(prev => prev.filter(type => type !== e.target.value));
        } else {
          setTypes(prev => [...prev, e.target.value]);
        }
      }}
    >
      <h3>Тип</h3>

      {categories.map(category =>
        <label><input                                    type="checkbox"
value={category.name}/>{category.name}</label>
      )}
    </form>
  </div>
 );
}
```

ДОДАТОК Б

Реалізація API ендпоінтів у файлі server.js

```
app.get('/getProducts', (req, res) => {
  productModel.find()
        .then(products => res.json(products))
        .catch(err => res.json(err));
});

app.get('/getCategories', (req, res) => {
  categoryModel.find()
        .then(categories => res.json(categories))
        .catch(err => res.json(err));
});

app.post('/order', (req, res) => {
  const order = new orderModel({
    username: req.body.username || "unknown",
    products: req.body.products,
    totalPrice: req.body.totalPrice
  });

  order.save();
});

app.post('/sendFeedback', (req, res) => {
  const report = new reportModel({
```

```
    username: req.body.username || "unknown",
    email: req.body.email,
    text: req.body.text
  });

  report.save();
});

app.post('/addUser', (req, res) => {
  const user = new userModel({
    username: req.body.username,
    email: req.body.email,
    password: req.body.password
  });

  user.save();
});

app.get('/getUser', (req, res) => {
  if (req.query.username) {
    userModel.find({ username: req.query.username, email: req.query.email })
          .then(user => res.json(user))
          .catch(err => res.json(err));
  } else {
    userModel.find({ email: req.query.email })
           .then(user => res.json(user))
           .catch(err => res.json(err));
  }
});
```

Декларація

академічної доброчесності

здобувача освіти ВСП «Економіко-правничого фахового коледжу ЗНУ»

Я, Чумак Владислав Миколайович, здобувач освіти 4 курсу, спеціальності/освітньо-професійної програми 121 Інженерія програмного забезпечення/Розробка програмного забезпечення, групи К121-21, адреса електронної пошти v.n.chumack@gmail.com

- підтверджую, що написана мною кваліфікаційна робота на тему «Розробка системи продажу спортивних товарів» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений;

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата _____ Підпис _____ Владислав ЧУМАК

Дата _____ Підпис _____ Роман КОТОВ