ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Вебсайт інтернет-магазину з продажу самокатів з використанням багатокритеріальних фільтрів та рекомендаційної системи»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп’ютерні науки»
Освітня програма «Комп’ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-21
Іван КИРИЛЛОВ
Керівник: зав. кафедри комп’ютерних наук
к.е.н., с.н.с., доцент
Сергій МІЧКІВСЬКИЙ

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2025 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри
комп'ютерних наук
_Сергій МІЧКІВСЬКИЙ
«_____»_____20___р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кириллов Іван Вячеславович

Тема роботи	Вебсайт інтернет-магазину з продажу самокатів з використанням багатокритеріальних фільтрів та рекомендаційної системи
Номер та дата наказу про затвердження теми	№121-7 від 24 грудня 2024 року
Коротка постанова завдання	Розробити веб-сайт інтернет-магазину для продажу самокатів для залучення більшої кількості покупців. Реалізувати використання багатокритеріальних фільтрів та рекомендаційну систему з використанням відповідних методів (наприклад, використання колаборативної фільтрації)
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1. М. Даниленко і І. Колесник, «МЕТОДИ РОЗРОБКИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ», *ІТКІ*, вип. 52, вип. 3, с. 10–15, Груд 2021. DOI: https://doi.org/10.31649/1999-9941-2021-52-3-10-15 2. Hodovychenko, M. A., & Gorbatenko, A. A. (2023). Recommender systems: models, challenges and opportunities. *Вісник сучасних інформаційних технологій*, *6*(4), 308–319. https://doi.org/10.15276/hait.06.2023.20
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має передбачити теоретичне, системотехнічне або експериментальне дослідження складного спеціалізованого завдання або практичної проблеми в галузі комп'ютерних наук, яке характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій

Дата видачі завдання 27 грудня 2024 р.

Керівник _____Сергій МІЧКІВСЬКИЙ

Здобувач освітнього ступеня бакалавра _____ Іван КИРИЛЛОВ

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
	Підготовчий етап		
1	Вибір напряму дослідження	02.12.2024 р.	*виконано*
2	Формування теми та призначення керівника	16.12.2024 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	23.12.2024 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	27.12.2024 р.	*виконано*
	Основний етап		
5	Розробка концепції кваліфікаційної роботи	13.01.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	20.01.2025 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	10.03.2025 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	24.03.2025 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	31.03-04.04.2025 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	07.04.2025 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	14.04.2025 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	21.04.2025 р.	*виконано*
	Завершальний етап		
13	Представлення рукопису для перевірки на плагіат	28.04-04.05.2025 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	05.05-11.05.2025 р.	*виконано*
15	Передзахист кваліфікаційної роботи	12.05-16.05.2025 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	19.05-06.06.2025 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	09.06-15.06.2025 р.	*виконано*
18	Доопрацювання доповіді та презентації для захисту	09.06-15.06.2025 р.	*виконано*
19	Захист кваліфікаційної роботи	16.06-22.06.2025 р.	*виконано*

Керівник _____ Сергій МІЧКІВСЬКИЙ

Здобувач освітнього ступеня бакалавра _____ Іван КИРИЛЛОВ

Кириллов І.В. Вебсайт інтернет-магазину з продажу самокатів з використанням багатокритеріальних фільтрів та рекомендаційної системи.

Кваліфікаційна робота за спеціальністю 122 – Комп'ютерні науки (освітня програма – Комп'ютерні науки) СО Бакалавр – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2025.

Описано розробку веб-сайт інтернет-магазину з продажу самокатів, який забезпечує пошук з використанням багатокритеріальних фільтрів та рекомендаційної системи.

Ключові слова: веб-сайт, інтернет-застосунок, багатокритеріальний фільтр.

Рис. 33. Бібліограф.: 26 найм.

Kyryllov I.V. Website of an online store selling scooters, using multi-criteria filters and a recommendation system.

Qualification work in the specialty 122 - Computer Science (educational program - Computer Science) Bachelor's degree - Higher Educational Institution "University of Economics and Law "KROK", Educational and Scientific Institute of Information and Communication Technologies, Department of Computer Science, Kyiv, 2025.

The development of a website for an online store selling scooters, which provides a search using multi-criteria filters and a recommendation system, is described.

Keywords: website, Internet application, multi-criteria filter

Fig. 33. Bibliography: 26 Items.

ЗМІСТ

ВСТУП 6

РОЗДІЛ 1 ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ВЕБСАЙТУ 8

1.1 Опис предметної області 8

1.2 Визначення потенційних конкурентних переваг 8

1.3 Постановка завдання на кваліфікаційну роботу 11

Висновки до розділу 1 13

РОЗДІЛ 2 ПРОЕКТУВАННЯ ВЕБ-САЙТУ 14

2.1 Проектування структури 14

2.2 Моделювання даних 14

2.3 Проектування інтерфейсу 17

2.4 Моделювання процесів 17

Висновки до розділу 2 19

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ВЕБ-САЙТУ 21

3.1 Особливості реалізації 21

3.2 Основні компоненти програмного продукту 24

3.3 Тестування 39

3.4 Використання 39

Висновки до розділу 3 40

ВИСНОВКИ 41

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 42

ВСТУП

Актуальність теми. У сучасному світі електронна комерція активно розвивається, і покупці все більше віддають перевагу інтернет-магазинам. На даний момент майже неможливо уяви собі магазин який не має свого веб-сайту. Майже кожен магазин має свій веб-сайт, адже це допомагає покупцям замовляти товари абсолютно з будь-якої точки світу. Через наявність великого асортименту товарів, користувачі можуть стискатися з труднощами при виборі необхідного їм товару. В тому числі і при виборі самокатів.

Для рішення даної проблеми, є застосування багатокритеріальних фільтрів та рекомендаційної системи та веб-сайті, які можуть допомогти покупцям у пошуку та придбанні їхнього товару. Веб-сайт інтернет-магазину дозволяє знайти підходящу модель самокату який підходить для потреб користувача. Для зручності пошуку користувачі можуть використовувати фільтри, які допоможуть їм знайти модель самокату, який буде відповідати їхнім потребам. Також їм може допомогти рекомендаційна система інтернет-магазину.

Використання багатокритеріальних фільтрів та рекомендаційної системи відповідає сучасним трендам в даній сфері, вони допомагають користувачу полегшити його пошук та вибір потрібного йому товару. А продавцю, визначити який товар, бренд є в тренді, для подальшого аналізу ринку, з метою покращення прибутку.

Мета дослідження полягає у створенні вебсайту інтернет-магазину з продажу самокатів з використанням багатокритеріальних фільтрів та рекомендаційної системи.

Завдання дослідження:

- аналіз аналогів;
- створення дизайну веб-сайту;
- проектування та розробка бази даних;
- розробка багатокритеріальних фільтрів;

- розробка рекомендаційної системи;
- розробка програмного забезпечення вебсайту інтернет-магазину з продажу самокатів з використанням багатокритеріальних фільтрів та рекомендаційної системи.

Об’єкт дослідження – процес пошуку, вибору та продажу самокатів користувачу.

Предмет дослідження – самокати та їх продаж через інтернет-магазин з використання багатокритеріальних фільтрів та рекомендаційної системи.

Методологічна основа розробки. Розробка інтернет-магазину основуючись на аналізі ринку, конкурентів та цільової аудиторії. Проведення порівняння вже існуючих аналогів. Визначення їхніх переваг та недоліків.

Практичне значення. Розробка веб-сайту інтернет-магазину з продажу самокатів для покращення продажів та збільшення кількості покупців. Задля цього реалізувати функції багатокритеріальних фільтрів та рекомендаційну систему.

Структура та обсяг пояснювальної оцінки. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку посилань (26 найменувань). Пояснювальна записка містить 33 рисунків. Загальний обсяг пояснювальної записки складає 44 сторінок, основний зміст викладено на 41 сторінках.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ВЕБСАЙТУ

1.1 Опис предметної області

На сьогоднішній день, все більшу і більшу популярність набирають самокати, як транспорт пересування в міських умовах. В наші дні, коли частину дня ми вимушені стояти в заторах, самокат допоможе вам у пересуванні по місту. Адже дістатись до вашого місця призначення можна набагато швидше по тротуару. Також, це значно економніше, адже коштує він, у порівнянні з автомобілем, набагато дешевше. Також вони до переваг самокатів можна віднести їхню компактність, адже вони не потребують великого приміщення для зберігання. До того ж, вони легкі у керуванні, і вам не потрібно буде довго навчатись їздити на них.[1]

Для більш зручного пошуку найбільш підходящого вам самокату, вам допоможе інтернет-магазин. Інтернет-магазини дуже популярні на даний момент. Це дешевий спосіб для початку власного бізнесу, адже це набагато дешевше ніж утримувати персонал та орендування переміщень. Також це значно спрощує графік роботи, адже клієнти зможуть формувати їхні замовлення в будь-який час доби [2].

За допомогою веб-сайту користувачі можуть переглядати доступні для них товари, переглядати їхні характеристики та ціни.

1.2 Визначення потенційних конкурентних переваг

Розглянемо деякі аналоги інтернет-магазинів для продажу самокатів.

SAMOKAT (рис. 1.1) – інтернет-магазин, який спеціалізується на продажі самокатів, велосипедів, запчастин та аксесуарів. Пропонують різноманітні акції та знижки на їхні товари. Включають безкоштовну доставку при замовлені від 3000 гривень. Також мають окремий розділ з тестами та оглядам власної продукції, що допомагає клієнтам обрати необхідний ним товар. Серед недоліків можна виділити застарілий дизайн.

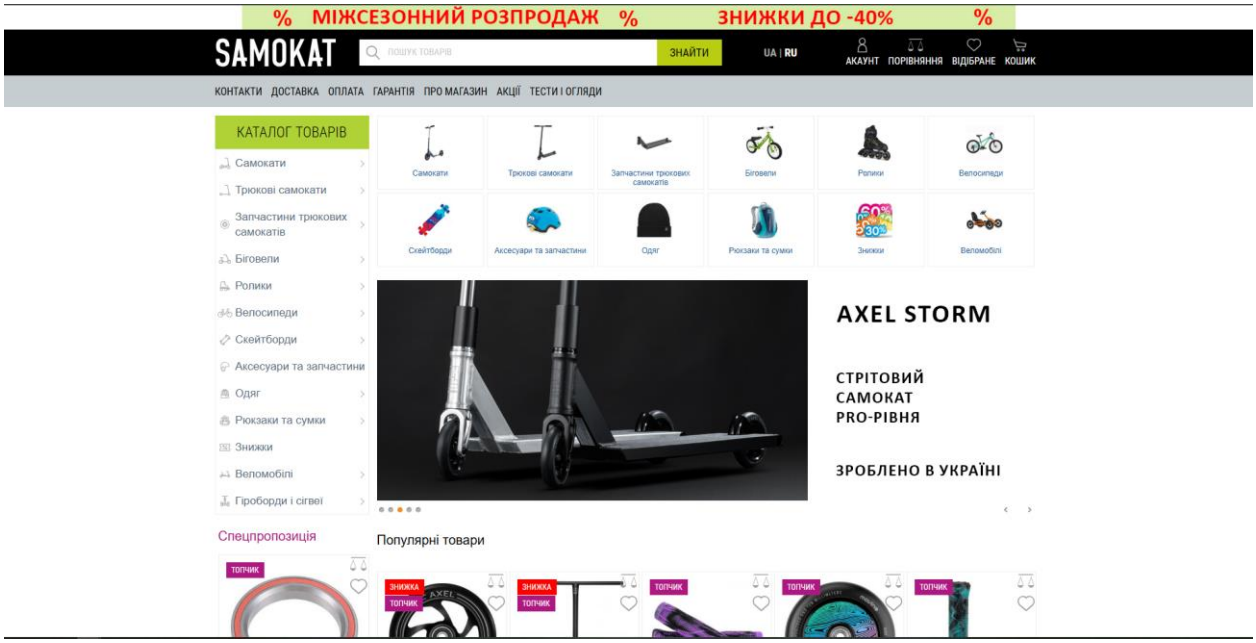

Рисунок 1.1 – Інтернет-магазин Samokat

Джерело [3].

Candy Boards(рис. 1.2) – інтернет-магазин, який спеціалізується на продажі персонально транспорту та аксесуарів для активного відпочинку. Мають великий асортимент товарів та співпрацюють з відомими брендами. Також мають окремий блог, де публікуються новини, огляди товарів та корисні поради. Також доступна безкоштовно доставка при замовленні від 2500 гривень. Серед недоліків можна виділити відсутність порівняльної статистики.

Flash Skate(рис. 1.3) – інтернет-магазин, спеціалізується на продажі активного транспорту та аксесуарів для активного відпочинку. Має наявність безкоштовної доставки при замовленні від 2500 гривень. Присутня можливість оплати частинами. Надає інформації про акції, новинки та хіти продажів. Відзначається велика надійність та довіра до магазину, більше 10 років на ринку з позитивними відгуками. Серед недоліків даного веб-сайту можна виділити повільне завантаження сторінок сайту.

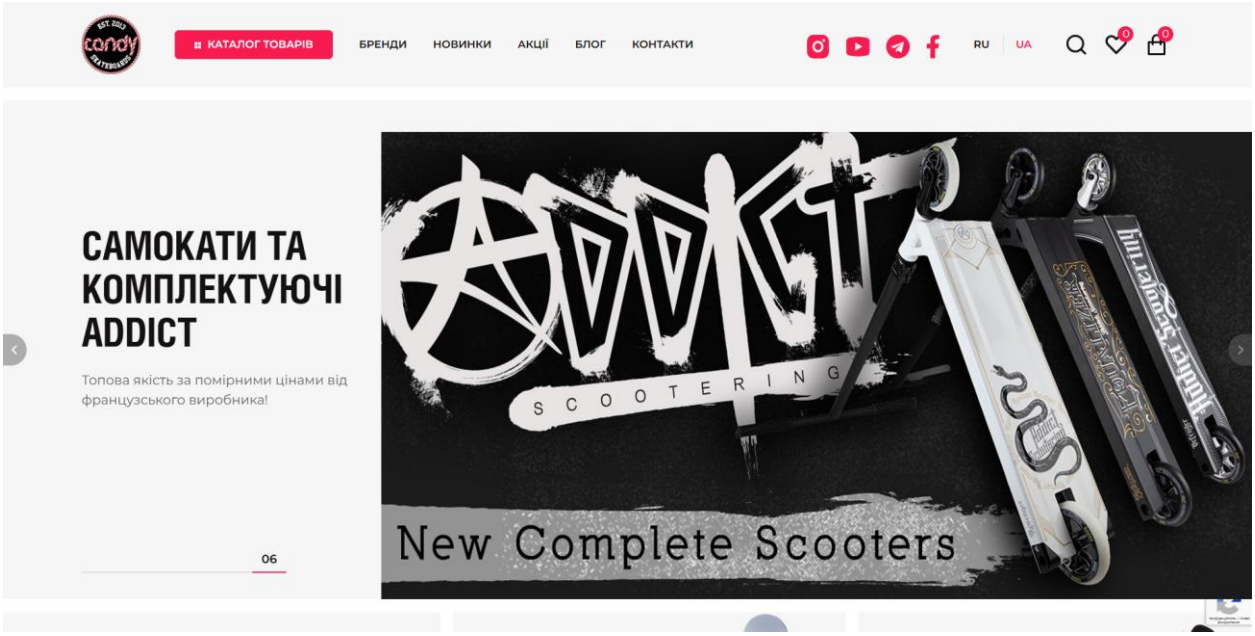

Рисунок 1.2 – Інтернет-магазин Candy Boards

Джерело [4].

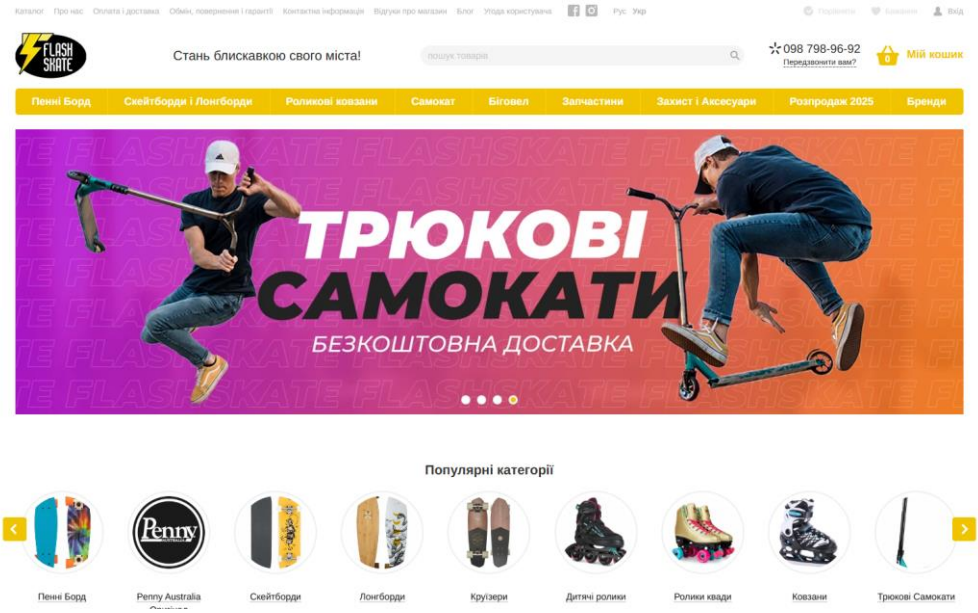

Рисунок 1.3 – Інтернет-магазин Flash Skate

Джерело [5].

Інтернет-магазин «ScootZone» може надавати порівняльну статистику його продуктів для користувачів, що полегшить вибір користувачу при виборі

товару. Також запровадить безкоштовну доставку, оскільки не матиме фізичних магазинів для придбання або видачі товару на руки. Перевагою також буде слугувати рекомендаційна система, яка буде допомагати користувачам при виборі товару. Також, величезний вплив матиме простий інтерфейс, який буде зрозумілий та максимально простий для користувача.

Ці переваги можуть допомогти знайти своїх користувачів, та підвищити продажі, що допоможе нав’язати конкуренцію іншим інтернет-магазинам.

1.3 Постановка завдання на кваліфікаційну роботу

Завданням кваліфікаційної роботи є розробка інтернет-магазину, яка включає розробку дизайну, який буде приємний та зрозумілий для користувача. Під час виконання роботи буде проаналізовано потенційних конкурентів для визначення основних функцій та потенційних переваг які мають бути реалізовані в роботі.

Потрібно буде реалізувати функцію реєстрацію та авторизації користувачів на веб-сайті. Це потребувати для більш детального перегляду інформації, яку буде повертати нам веб-сайт (профіль користувача, дані користувача, історію замовлень, збережені товари).

Обов’язковою буде реалізація багатокритеріальних фільтрів для більш швидкого та зручного пошуку необхідного товару.

Вхідною інформацією будуть слугувати дані користувача при реєстрації, оцінки та відгуки. Вихідною ж інформацією будуть слугувати характеристики та ціни товарів, відгуки та рейтинги інших покупців, історія замовлень.

Для успішної реалізації веб-сайту інтернет-магазину, потрібно реалізувати коректну роботу користувача з серверною частиною та базою даних. Для визначення ми використовуємо функціональні та нефункцінальні вимоги.

Функціональні та нефункціональні вимоги. Аналіз вимог – це важливий процес, який допомагає в оцінці системного та програмного проекту. Аналіз вимог поділяється та функціональні та нефункціональні вимоги.[7]

Функціональні вимоги – це базові можливості, які повинна пропонувати система. Вони обов’язково повинні бути включені в систему. Вони сформовані у вигляді даних, які потрібно надати системі, виконуваної операції та очікуваного результату. Вони сформовані користувачем, які ми побачимо в кінцевому продукті. Зазвичай функціональні вимоги описують поведінку системи за певних умов.[6][7]

Функціональні вимоги:

- перегляд товарів – користувач повинен мати змогу переглядати усі товари, які доступні в каталозі магазину;
- реєстрація користувача – користувач повинен мати змогу для реєстрації на сайті, для подальшого оформлення покупок
- авторизація користувача – користувач повинен мати змогу авторизуватись на зареєстрованому аккаунті, для подальшого оформлення покупок та перегляду історії придбань;
- фільтрація товарів – користувач повинен мати змогу використовувати наявні функції фільтрації товарів, для більш гнучкого, зручного та швидкого пошуку необхідного товару;
- оформлення замовлень – користувач повинен мати змогу оформлювати замовлення на сайті.

В той же час, нефункціональні вимоги не пов’язані з функціональністю системи. Нефункціональні вимоги визначають, як система повинна працювати. Вони мають вирішальне значення для забезпечення зручності, надійності та ефективності системи, часто впливаючи на загальний користувальницький досвід. Їх також називають не поведінковими вимогами.[6][7]

Нефукнціональні вимоги:

- маршрутизація – користувач повинен швидко переходити між сторінками магазину, не чекати повільного завантаження сторінок на сайті;

- простий інтерфейс – користувач не повинен губитись на сайті, він має бути зрозумілим для користувача;
- безпека – користувач повинен бути певен у забезпеченні безпечного зберігання його персональних даних.

Автоматизовані функції – це функції, які використовуються без втручання користувача. Вони використовується в результаті виконання певних подій на сайті.

Автоматизовані функції:

- перегляд товарів;
- реєстрація користувачів;
- багатокритеріальні фільтри;
- кошик та оформлення замовлення;
- рекомендаційна система.

Висновки до розділу 1

В даному розділі ми розглянули теоретичні етапи створення веб-застосунку інтернет-магазину «ScootZone». Було показано причини та етапи створення даного застосунку. Було проведено аналіз потенційних конкурентів, визначено їхні переваги та недоліки.

Визначали функціональні та нефункціональні вимоги.

Під час практичної реалізації завдання, можуть виникати зміни, оскільки під час реалізації теоретичних завдань будуть помилки, вирішення яких потребуватиме певних змін та нововведень.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ВЕБ-САЙТУ

2.1 Проектування структури

Діаграма прецедентів – це візуальне представлення, яке показує нам взаємодію між користувачем(актором) та системою. Діаграма прецедентів показує нам функціональні вимоги системи, показуючи, як різні користувачі взаємодіють з функціональними можливостями системи. Вони мають вирішальне значення для визначення обсягу та вимог системи.[8]

Актори – це суб'єкти які взаємодіють із системою. Це можуть бути користувачі, інші системи або апаратні пристрої. Правильна ідентифікація та розуміння акторів є вирішальними для точного моделювання поведінки системи.[8]

Прецеденти представляють конкретні можливості, які може робити ваша система.

На рис. 2.1 зображено діаграму прецедентів веб-сайту онлайн-магазину «ScootZone», яка показує взаємодію користувача із системою.

Діаграма послідовності – це діаграми взаємодії, вони описують, як виконуються операції. Відображають взаємодію між об'єктами в контексті спільної роботи. Вони показують порядок взаємодії, використовуючи вертикальну вісь діаграми для представлення часу.[9]

На рисунку 2.2 ми побачимо діаграму прецедентів, яка показує взаємодію між об'єктами під час виконання оформлення замовлення на сайті.

2.2 Моделювання даних

Технологія структурного аналізу та проектування (SADT) – це діаграма нотація, розроблена для того, щоб допомогти розуміти систему. Вона пропонує будівельні блоки для представлень сутностей та видів діяльності, а також різноманітні стрілки для зв'язку між блоками. SADT можна використовувати, як інструмент функціонального аналізу певного процесу.[10]

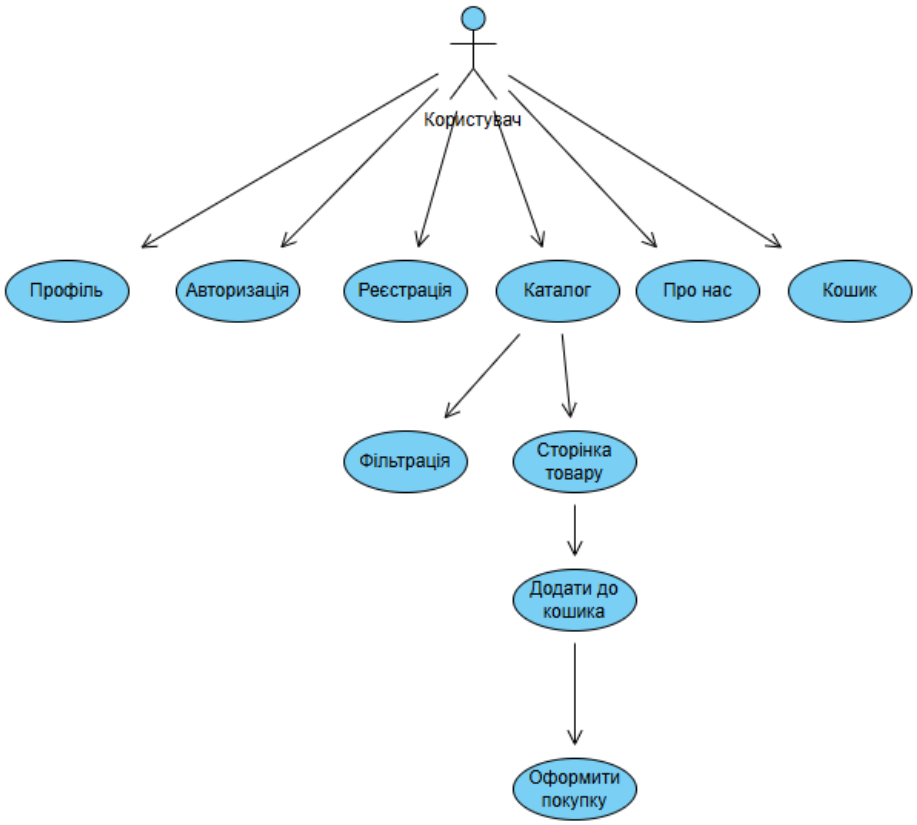

Рисунок 2.1 – Діаграма прецедентів

Джерело: розроблено автором

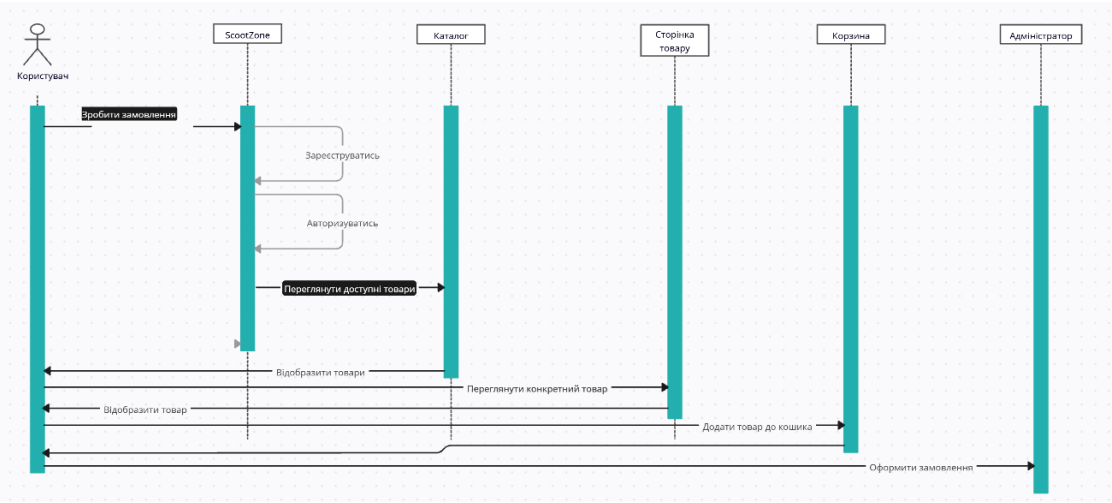

Рисунок 2.2 – Діаграма послідовності

Джерело: розроблено автором

На рис. 2.3 представлено SADT діаграму авторизації користувача на веб-сайті магазину.

Діаграма потоку користувача – це діаграма, що показує певну послідовність дій, яку виконає користувач на сайті, щоб виконати певну задачу. Це інструмент, який використовується для визначення оптимальних способів взаємодії з веб-сайтом. Чим краще буде реалізований процес руху користувача від початку до закінчення певного процесу, тим більша ймовірність, що буде забезпечений чудовий користувацький досвід. Наявність діаграми потоку користувача допомагає зрозуміти, що повинен робити продукт, в якому порядку повинна надаватись інформація користувачу, і чому кожна функція та сторінка знаходиться там, де вона є.[11][12]

На рисунку 2.4 представлено діаграму потоку користувача, під час оформлення замовлення на сайті.

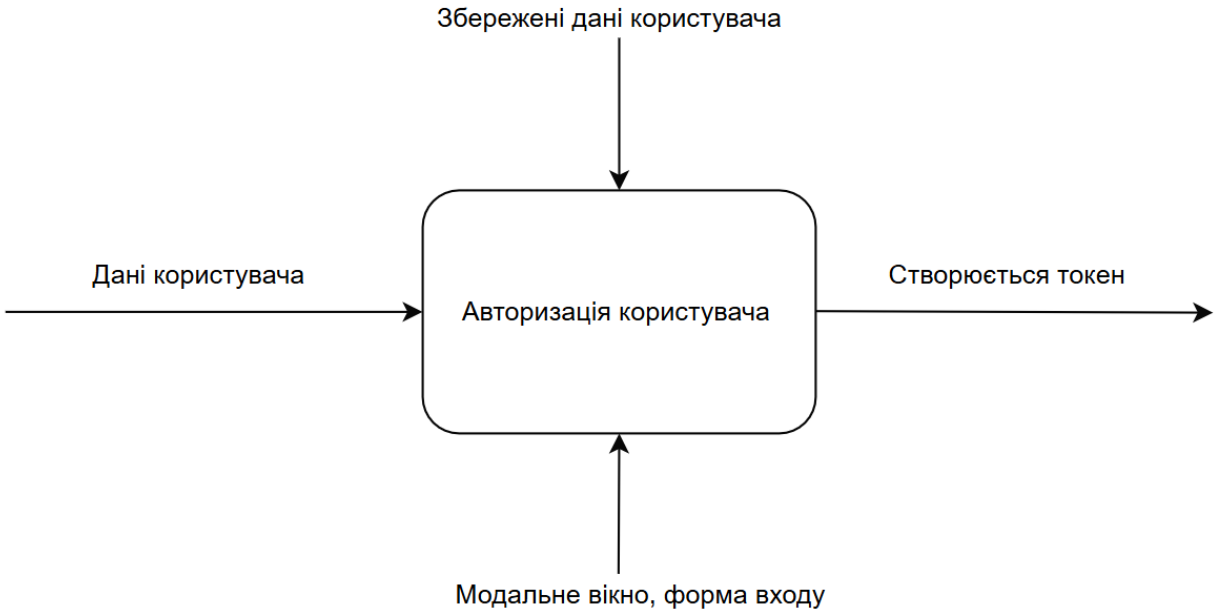

Рисунок 2.3 – Діаграма SADT

Джерело: розроблено автором

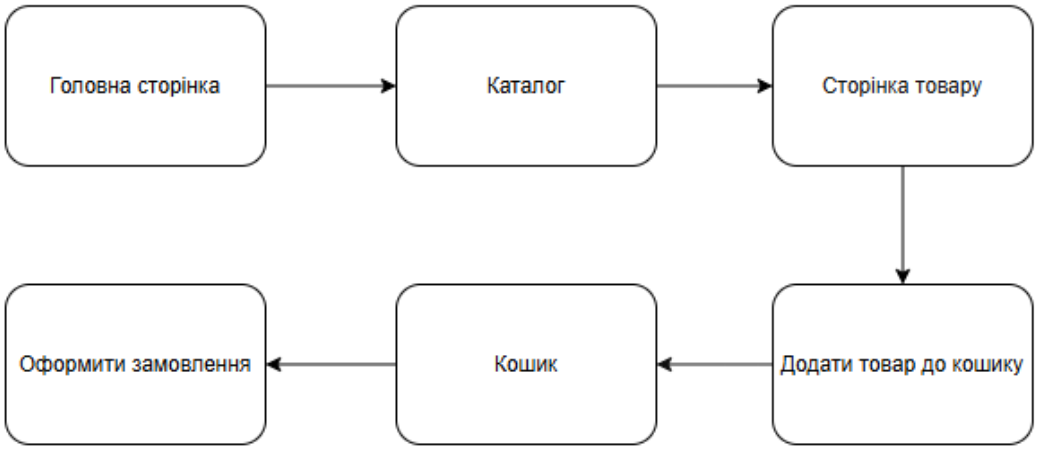

Рисунок 2.4 – Діаграма потоку користувача

Джерело: розроблено автором

2.3 Проектування інтерфейсу

Діаграма структури інтерфейсу показує зв'язок між елементами, як і з якими елементами вони взаємодіють між собою. Це архітектура, яка використовується для проектування користувацьких інтерфейсів для програмних додатків. Діаграму структури інтерфейсу можна використовувати для створення візуального дизайну поведінки та взаємодії з користувачем.[13]

На рис. 2.5 представлено діаграму структуру інтерфейсу.

2.4 Моделювання процесів

Діаграма діяльності. Вона допомагає візуалізувати робочі процеси або види діяльності в системі. Вони показують, як різні дії пов'язані між собою і як система переходить з одного стану в інший. Пропонує чітку картину складних робочих процесів. Діаграми діяльності полегшують розуміння того, як різні елементи взаємодіють між собою у системі. Відображають порядок, в якому відбуваються дії, і чи відбуваються вони послідовно або паралельно. Допомагають пояснити, що викликає певні дії або події в системі.

Починається з початкової точки і закінчується в кінцевій точці, показуючи різні шляхи прийняття рішень на цьому шляху.[14]

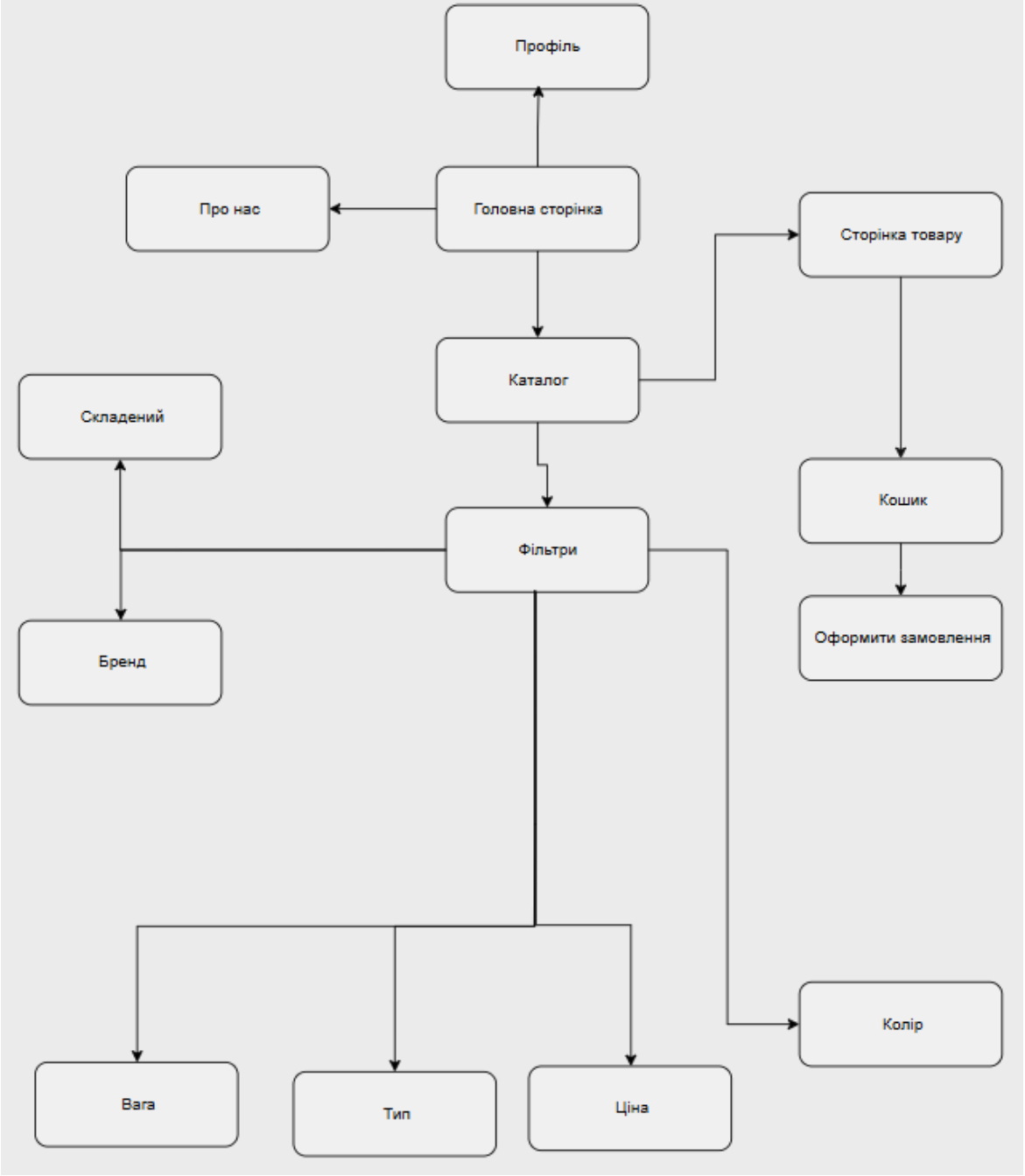

Рисунок 2.5 – Діаграма структуру інтерфейсу

Джерело: розроблено автором

На рис. 2.6 представленодіаграму діяльності, яка показує процес оформлення замовлення та інші процеси, які можуть бути виконані під час оформлення замовлення.

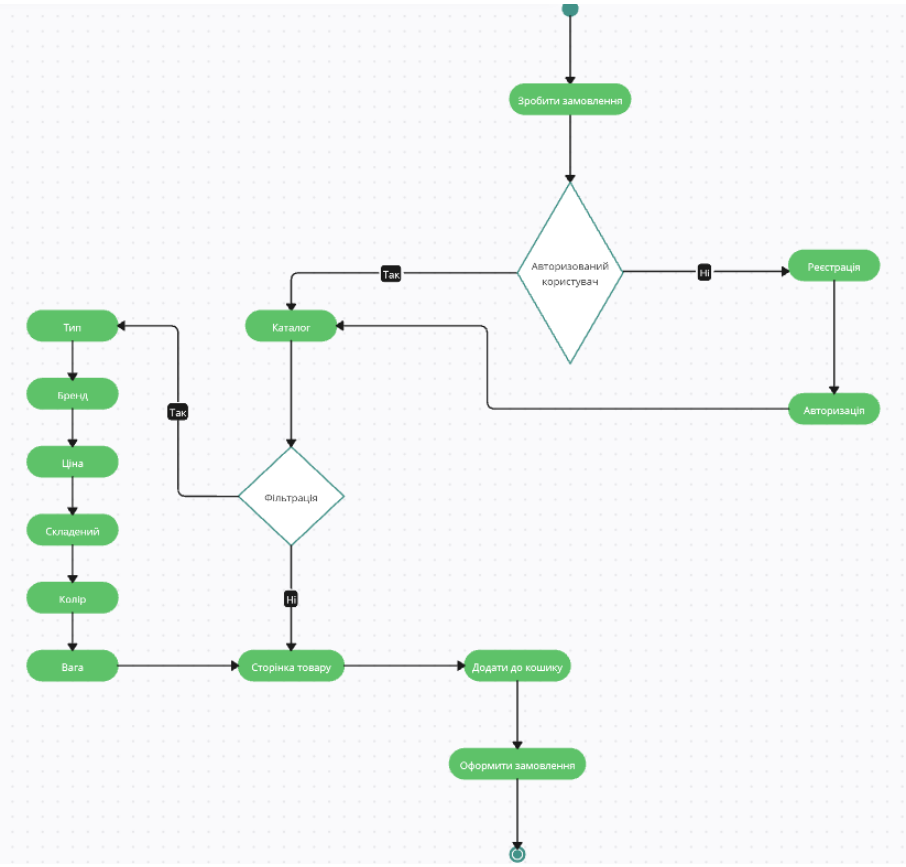

Рисунок 2.6 – Діаграма діяльності

Джерело: розроблено автором

Висновки до розділу 2

У цьому розділі ми розглянули послідовність проектування вебсайту «ScootZone». Сформували та спроектували структуру за допомогою діаграм, таких як:

- діаграма прецедентів;
- діаграма послідовності;
- діаграма класів;
- діаграма SADT;
- діаграма потоку користувача;
- діаграма структури інтерфейсу;
- діаграма діяльності.

Використавши дані діаграми, ми змогли чіткіше зрозуміти структуру веб-сайту та послідовність дій користувача. Окрема увага була приділена архітектурі веб-сайту та мапі подорожі користувача на сайті.

Результатом проектування є чітко структурована модель інформаційної системи. Побудована структура дозволяє забезпечити ефективну роботу з товарами, користувачами та замовленнями, а також забезпечує масштабованість та підтримку додаткових функцій у майбутньому.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБ-САЙТУ

3.1 Особливості реалізації

Цей розділ опише специфіку роботу, технології та реалізовані функції нашого магазину.

Архітектура сайту є одним із важливих пунктів у створення веб-сайту.

На рис. 3.1 представлено трьох рівневу архітектуру сайту.

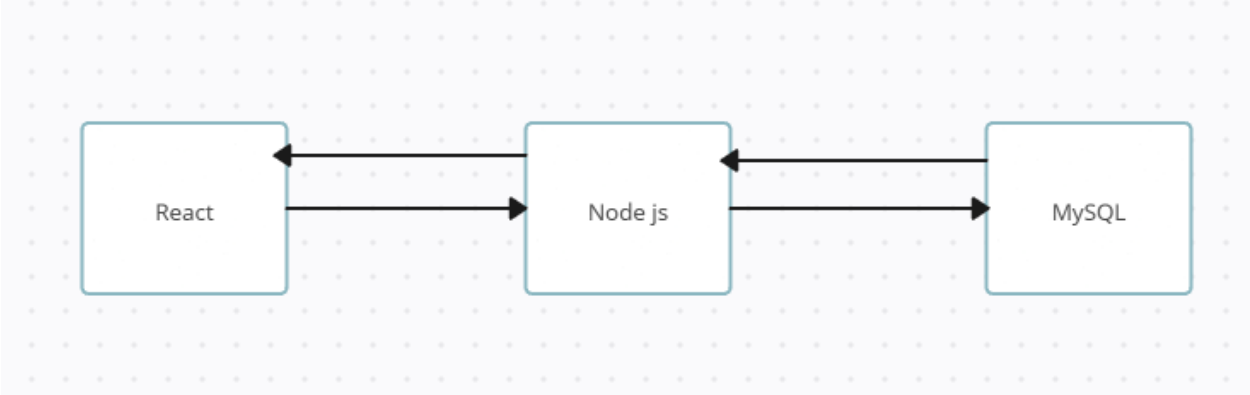

Рисунок 3.1 – Архітектура веб-сайту

Джерело: розроблено автором

React – це безкоштовна JavaScript-бібліотека з відкритим вихідним кодом, яка має на меті зробити побудову користувацьких інтерфейсів на основі компонентів. Його можна використовувати для розробки односторінкових, мобільних або серверних додатків. Ключовою перевагою є те, що React рендерить тільки ті частини сторінки, які змінилися, уникаючи непотрібного повторного рендерингу незмінних елементів DOM. Використовує мови програмування HTML, JavaScript, CSS, TyyeScript.[15]

HTML (Hypertext Markup Language) – це стандартна мова розмітки документів, призначених для відображення у браузері. Вона визначає зміст і структуру веб-контенту. Підтримується всім основними веб-браузерами,

включаючи десктопні та мобільні веб-браузери. Елемент HTML відокремлюється від іншого тексту в документів за допомогою «тегів», які складаються з імені елемента, оточеного символами < і >. Часто її доповнюють такі мови програмування, як CSS та JavaScript.[18][19]

CSS (Cascading Style Sheets) – це мова таблиць стилів, яка використовується для визначення презентації та стилю документа, написано мовою розмітки. Призначений для розділення вмісту та презентації, включаючи макет, кольори та шрифти. Розділення допомагає покращити доступність контенту, оскільки контент можна писати, не турбуючись про його представлення; забезпечити більшу гнучкість і контроль у визначенні характеристик представлення.[20]

JSX (JavaScript XML) – це спеціальний синтаксис, який використовується в React для спрощення побудови користувацьких інтерфейсів. Завдяки ньому ми можемо писати HTML код всередині JavaScript. Так ми можемо створювати компоненти інтерфейсу користувача ефективніше. Коли React обробляє JSX-код, він перетворює його на JavaScript. Потім цей код створює HTML-елементи в DOM браузера, і так відображається веб-сторінка.[22]

JavaScript є мовою програмування основною технологією Всесвітньої павутини, поряд з HTML та CSS. Є високорівневою мовою програмування, яка часто компілюється just-in-time, що відповідає стандарту ECMAScript. Має динамічну типізацію, об’єктну орієнтацію на основі прототипів та першокласні функції. Підтримує подієво-орієнтований, функціональний та імперативний стилі програмування. Має інтерфейси прикладного програмування (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних і об’єктною моделлю документа (DOM). На практиці веб-браузер або інша система виконання надає JavaScript API для вводу/виводу.[21]

Nodejs – кросплатформенне середовище виконання JavaScript з відкритим вихідним кодом. Дозволяє використовувати JavaScript для

написання інструментів командного рядка та сценаріїв на стороні сервера. Має архітектуру, керовану подіями і здатну до асинхронного вводу/виводу. Ці рішення спрямовані на оптимізацію пропускної здатності та масштабованості веб-додатків з великою кількістю операцій вводу/виводу, а також для веб-додатків, що працюють в режимі реального часу.[16]

MySQL – найпопулярніша світі система управління базами даних з відкритим вихідним кодом. Розшифровується як Structured Query Language (мова структурованих запитів). Це мова програмування, яка використовується для отримання, оновлення, видалення та інших маніпуляцій з даними в реляційних баз даних. Це реляційна база даних на основі SQL, призначена для зберігання та управління структурованими даними.[17]

SQL – мова програмування для зберігання та обробки інформації в реляційній базі даних. Оператори SQL використовуються для оновлення, видалення, пошуку та отриманні інформації з бази даних.[23]

На рис. 3.2 представлено ER діаграму бази даних нашого магазину «ScootZone».

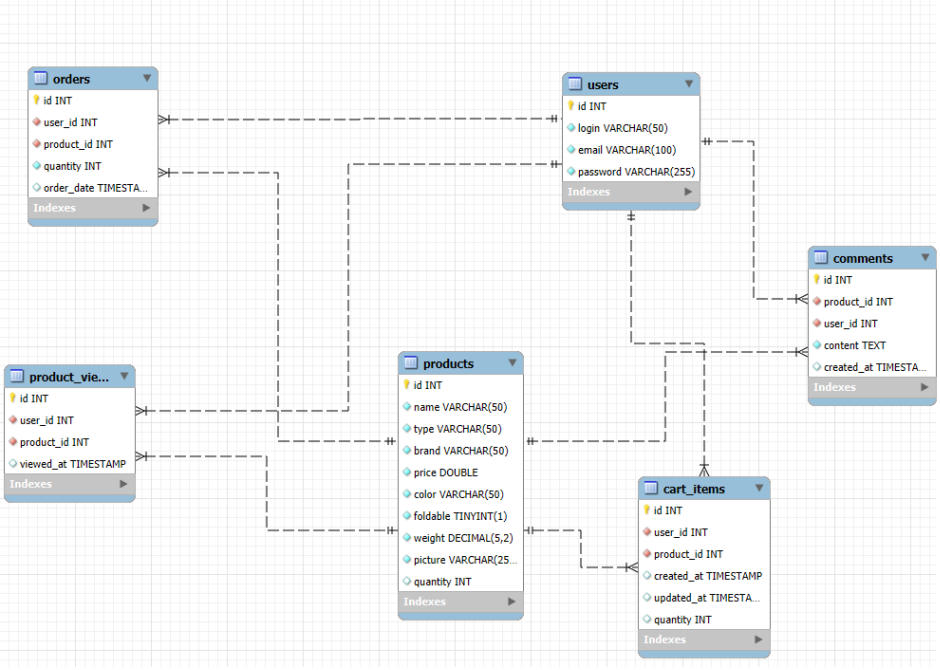

Рисунок 3.1 – ER діаграма

Джерело: розроблено автором

3.2 Основні компоненти програмного продукту

Основні компоненти складаються з таких як:

- підключення до бази даних;
- реєстрація та авторизація користувача;
- сторінка профілю;
- каталог товарів;
- фільтрація товарів;
- сторінка товару;
- кошик з оформленням замовлення;
- рекомендації на основі переглянутих товарів.

Підключення до БД. На рис. 3.2 можна код файлу db.js. В даному коді, через бібліотеку mysql2/promise використовується заповнення необхідних даних для підключення до бази даних.

На рис. 3.3 ми можемо побачити змінну pool, куди ми імпортуємо наш файл підключення до бази даних. Після цього цю змінну, ми будемо використовувати в інших методах нашої серверної частини для відправки запитів до нашої бази даних.

Тим часом на рис. 3.4 ми можемо побачити метод Listen, через який ми запускаємо роботу нашого сервера, і автоматично відбувається підключення до бази даних.

```
const mysql = require('mysql2/promise');

const pool = mysql.createPool({
  host: 'localhost',
  user: '[redacted]',
  password: [redacted],
  database: 'store',
});

module.exports = pool;
```

Рисунок 3.2 – файл db.js

Джерело: розроблено автором

```
const pool = require('./db');
```

Рисунок 3.3 – змінна «pool»

Джерело: розроблено автором

```
app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

Рисунок 3.4 – Запуск роботи серверу

Джерело: розроблено автором

Реєстрація та авторизація користувача. На рисунках 3.5 та 3.6 представлено реалізацію модального вікна авторизації та реєстрації, вони слугують для авторизації та реєстрації користувачів на сайті. А На рис. 3.7 представлено фрагмент лістингу реалізації цих вікон в середовищі Visual Studio Code.

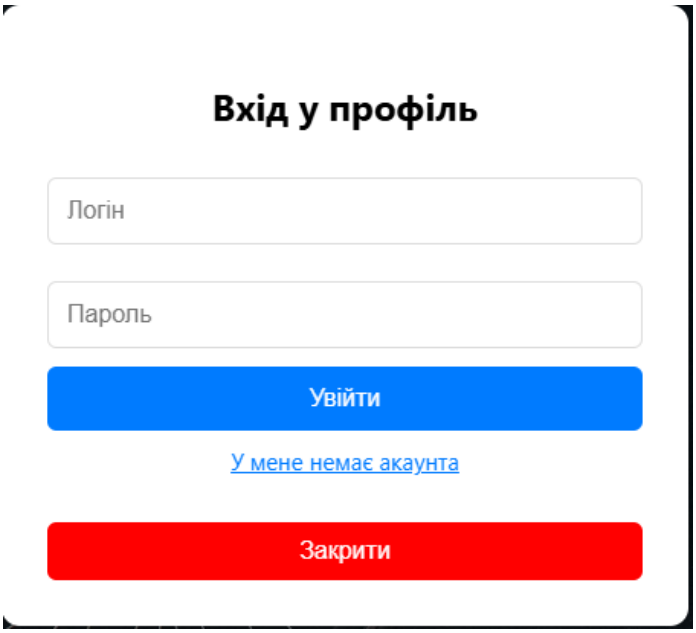

Рисунок 3.5 – Вікно авторизації

Джерело: розроблено автором

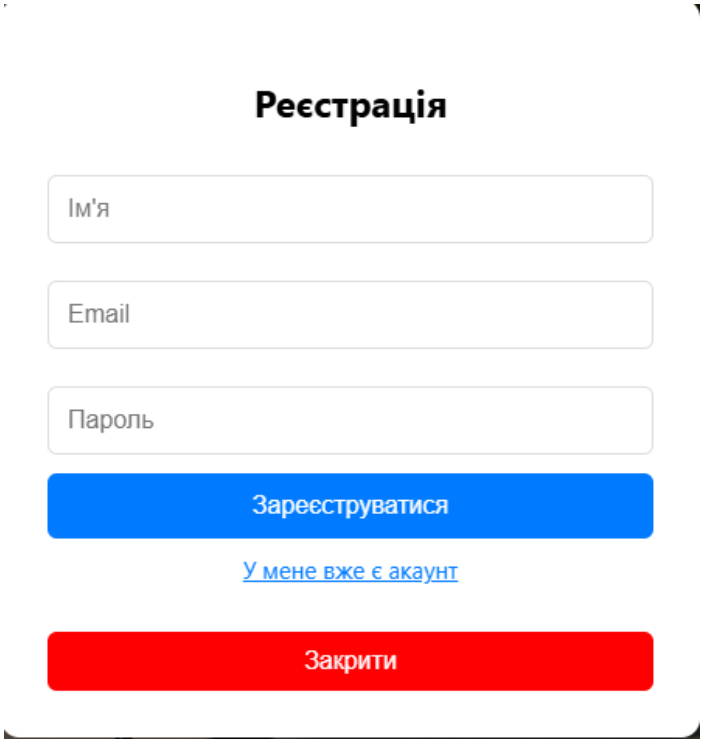

Рисунок 3.6 – Вікно реєстрації

Джерело: розроблено автором

```
{isModalOpen && (
  <div className="modal-overlay" onClick={() => setIsModalOpen(false)}>
    <div className="modal-content" onClick={(e) => e.stopPropagation()}>
      {isLogin ? (
        <>
          <h2>Вхід у профіль</h2>
          <input type="text" placeholder="Логін" value={username} onChange={(e) => setUsername(e.target.value)} />
          <input type="password" placeholder="Пароль" value={password} onChange={(e) => setPasssword(e.target.value)} />
          <button className="login-btn" onClick={handleLogin}>Увійти</button>
          <p onClick={() => setIsLogin(false)} className="switch-link">
            У мене немає акаунта
          </p>
        </>
      ) : (
        <>
          <h2>Реєстрація</h2>
          <input type="text" placeholder="Ім'я" value={username} onChange={(e) => setUsername(e.target.value)} />
          <input type="text" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)} />
          <input type="password" placeholder="Пароль" value={password} onChange={(e) => setPasssword(e.target.value)} />
          <button className="login-btn" onClick={handleRegister}>Зареєструватися</button>
          <p onClick={() => setIsLogin(true)} className="switch-link">
            У мене вже є акаунт
          </p>
        </>
      )}
      <button className="close-btn" onClick={() => setIsModalOpen(false)}>
        Закрити
      </button>
    </div>
  </div>
)}
```

Рисунок 3.7 – Програмний код модального вікна

Джерело: розроблено автором

На рис. 3.8 представлено фрагмент лістингу функції реєстрації користувача. Для цього ми використовуємо метод post. Спочатку ми отримуємо вхідні дані з нашого вікна реєстрації. Додана перевірка на введення усіх полів, якщо хоча б одне поле буде порожнє – повертатиметься помилка 400 (Bad Request). Після цього для забезпечення безпечного зберігання пароля, відбувається його хешування за допомогою функції bcryptjs. Після цього ми формуємо наш sql-запит, який ми відправляємо на сервер. При успішному виконанні функції, повертатиметься код 201 (Created). Також додана перевірка на логін або email користувача. Якщо користувач з такою поштою або логіном вже існує, повертатиметься помилка 409 (Conflict).

```
app.post('/api/register', async (req, res) => {
  const { login, email, password } = req.body;
  if (!login || !email || !password) {
    return res.status(400).json({ error: 'Всі поля обов'язкові' });
  }

  try {
    const hashedPassword = bcrypt.hashSync(password, 8);
    const query = 'INSERT INTO users (login, email, password) VALUES (?, ?, ?)';
    await pool.query(query, [login, email, hashedPassword]);

    res.status(201).json({ message: 'Користувач зареєстрований' });
  } catch (err) {
    if (err.code === 'ER_DUP_ENTRY') {
      return res.status(409).json({ error: 'Користувач з таким логіном або email вже існує' });
    }
    res.status(500).json({ error: err.message });
  }
});
```

Рисунок 3.8 – Функція реєстрації користувача

Джерело: розроблено автором

На рис. 3.9 ми можемо побачити функцію авторизації користувача. Для цього ми використовуємо метод post. Спочатку він отримує вхідні дані з полів форми авторизації. Якщо поля порожні – повертається помилка 400 (Bad Request). Після цього ми виконуємо sql-запит для пошуку користувача з введеним логіном. Якщо користувач з таким логіном не був знайдений –

повертається помилка 401 (Unauthorized). Потім перевіряється пароль, якщо пароль не співпадає – повертається помилка 401 (Unauthorized). Далі відбувається генерація JWT токена, з даними нашого користувача, завдяки якому буде відбуватись доступ до функцій для авторизованих користувачів.

```
app.post('/api/login', async (req, res) => {
  const { login, password } = req.body;
  if (!login || !password) {
    return res.status(400).json({ error: 'Логін і пароль обов'язкові' });
  }

  try {
    const [results] = await pool.query('SELECT * FROM users WHERE login = ?', [login]);
    if (results.length === 0) {
      return res.status(401).json({ error: 'Користувача не знайдено' });
    }

    const user = results[0];
    const isMatch = bcrypt.compareSync(password, user.password);

    if (!isMatch) {
      return res.status(401).json({ error: 'Невірний пароль' });
    }

    const token = jwt.sign({ id: user.id, login: user.login }, SECRET_KEY, { expiresIn: '2h' });

    res.json({ message: 'Вхід успішний', token });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

Рисунок 3.9 – Функція авторизації користувача

Джерело: розроблено автором

Сторінка профілю користувача. На рис. 3.10 ми можемо побачити функцію отримання даних нашого користувача. Спочатку відбувається перевірка наявності токена, якщо токен дійсний, то ми додаємо в об'єкти «user» та «req» його дані. Якщо токен недійсний або відсутній – повертає помилку 401 (Unauthorized). Далі ми отримуємо id нашого користувача з токена. Після цього ми виконуємо запит до нашої бази даних. Якщо користувач не був знайдений – повертається помилка 404. Якщо такий користувач існує в базі даних – повертаються його дані.

```
app.get('/api/profile', verifyToken, async (req, res) => {
  const userId = req.user.id;
  try {
    const [results] = await pool.query('SELECT login, email FROM users WHERE id = ?', [userId]);
    if (results.length === 0) return res.status(404).json({ error: 'Користувача не знайдено' });

    res.json({ user: results[0] });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

Рисунок 3.10 – Отримання даних користувача

Джерело: розроблено автором

Каталог товарів. Сторінка каталогу товарів використовується для відображення наявних товарів в магазині.

На рис. 3.11 ми можемо побачити сторінку каталогу товарів магазину ScootZone.

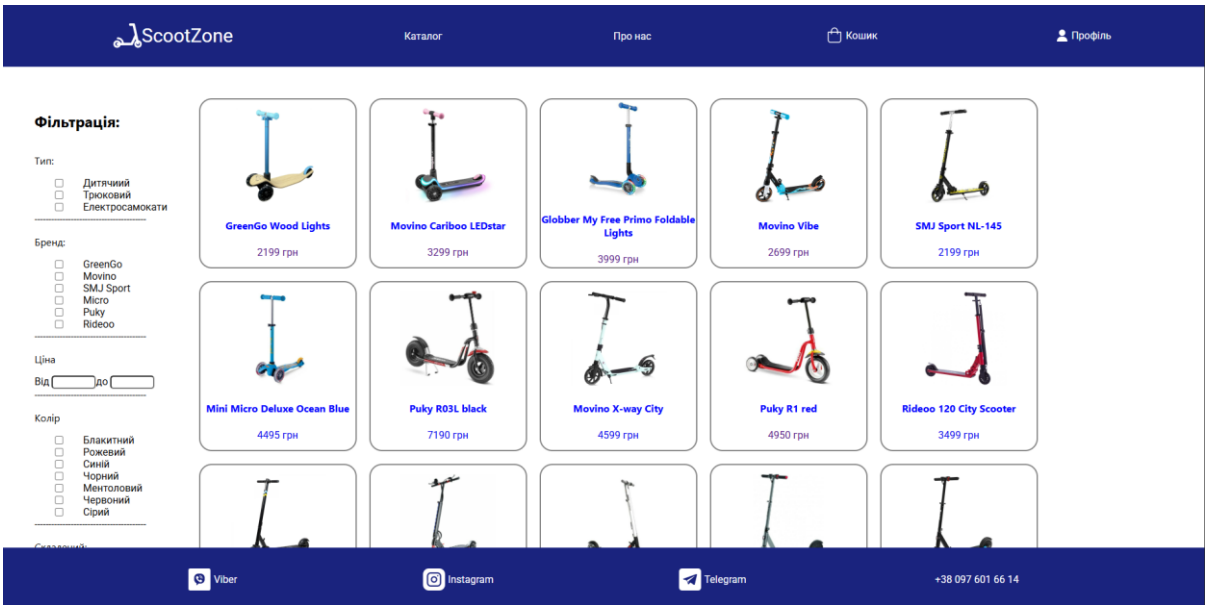

Рисунок 3.11 – Сторінка каталогу товарів

Джерело: розроблено автором

На рис. 3.12 ми можемо побачити код відображення товарів в секторі main на сторінці каталогу товарів. На рис. 3.13 ми можемо побачити метод get, через який ми отримуємо дані про товари. В даному методі, ми бачимо що

після того як ми отримуємо get-запит з клієнтської частини, використовується наша змінна pool, для відправки відповідного sql-запиту до бази даних. Після цього ми повертаємо результат у об’єкт відповіді «res» у форматі json. Якщо сталась помилка, то виводиться повідомлення про помилку в консоль.

```
{/* Каталог */}
<div className="product-list">
  {products.map((product) => (
    <div className="product-card" key={product.id}>
      <Link className="react-link" to={`/product/${product.id}`}>
        <img src={product.picture} alt={product.name} />
        <h4 className='product-name'>{product.name}</h4>
        <span>{product.price} грн</span>
      </Link>
    </div>
  ))}
</div>
```

Рисунок 3.12 – Вивід на екран наявних товарів

Джерело: розроблено автором

```
app.get('/api/products', async (req, res) => {
  try {
    const [results] = await pool.query('SELECT * FROM products');
    res.json(results);
  } catch (err) {
    console.error("Помилка при запиті продуктів:", err.message);
    res.status(500).json({ error: err.message });
  }
});
```

Рисунок 3.13 – Отримання даних про товари

Джерело: розроблено автором

Фільтрація товарів. На рис. 3.14 ми можемо побачити функції:

– handleCheckboxChange – функція виконання фільтрації за параметрами: тип, бренд, колір та складений товар чи ні;

– handleInputChange – функція виконання фільтрація за параметрами ціни та ваги товару.

```
const handleCheckboxChange = (category, value) => {
  setFilters(prev => {
    const current = prev[category] || [];
    const updated = current.includes(value)
      ? current.filter(v => v !== value)
      : [...current, value];

    const newFilters = { ...prev };

    if (updated.length === 0) {
      delete newFilters[category];
    } else {
      newFilters[category] = updated;
    }

    return newFilters;
  });
};

const handleInputChange = (e, key) => {
  setFilters(prev => ({
    ...prev,
    [key]: e.target.value
  }));
};
```

Рисунок 3.14 – Функції фільтрації товарів

Джерело: розроблено автором

На рис. 3.15 ми можемо побачити метод post, через який ми отримуємо дані про товари, які були відфільтровані клієнтом на клієнтській частині. Покрокове пояснення методу:

- спочатку в даному методі ми отримуємо фільтри з тіла запиту «req.body»;
- далі ми вказуємо нашу початкову заготовку нашого SQL-запиту в змінній «query»;
- далі ми використовуємо оператор if для додавання наших фільтрів до початкового запиту;
- після цього ми отримуємо наш фінальний запит, який відправляємо до нашої бази даних;

– база даних повертає нам результат у вигляді json файлу.

```
app.post('/api/products/filter', async (req, res) => {
  const {
    type = [], brand = [], color = [], foldable = [],
    priceMin, priceMax,
    weightMin, weightMax
  } = req.body;

  let query = 'SELECT * FROM products WHERE 1=1';
  const params = [];

  if (type.length) {
    query += ` AND type IN (${type.map(() => '?').join(',')})`;
    params.push(...type);
  }

  if (brand.length) {
    query += ` AND brand IN (${brand.map(() => '?').join(',')})`;
    params.push(...brand);
  }

  if (color.length) {
    query += ` AND color IN (${color.map(() => '?').join(',')})`;
    params.push(...color);
  }

  if (foldable.length) {
    query += ` AND foldable IN (${foldable.map(() => '?').join(',')})`;
    params.push(...foldable);
  }

  if (priceMin) {
    query += ' AND price >= ?';
    params.push(priceMin);
  }

  if (priceMax) {
    query += ' AND price <= ?';
    params.push(priceMax);
  }

  if (weightMin) {
    query += ' AND weight >= ?';
    params.push(weightMin);
  }

  if (weightMax) {
    query += ' AND weight <= ?';
    params.push(weightMax);
  }

  try {
    const [results] = await pool.query(query, params);
    res.json(results);
  } catch (err) {
    console.error('Помилка при фільтрації товарів:', err.message);
    res.status(500).json({ error: 'Помилка сервера при фільтрації' });
  }
});
```

Рисунок 3.15 – Фільтрація товарів

Джерело: розроблено автором

Сторінка товару. На рис. 3.16 ми можемо побачити сторінку товару нашого магазину. На рис. 3.17 ми можемо побачити метод отримання даних про конкретний товар. Спочатку за допомогою властивості req.params ми отримуємо id нашого товару. Після цього ми виконуємо запит з пошуком

товару за нашим id. Якщо товар не був знайдений, то повертається статус 404 (Not Found). Якщо товар був знайдений, то ми повертаємо об’єкт одного товару у форматі JSON. Якщо відбулась помилка при запиті до бази даних, то в консолі повернеться статус 500 (Internal Server Error).

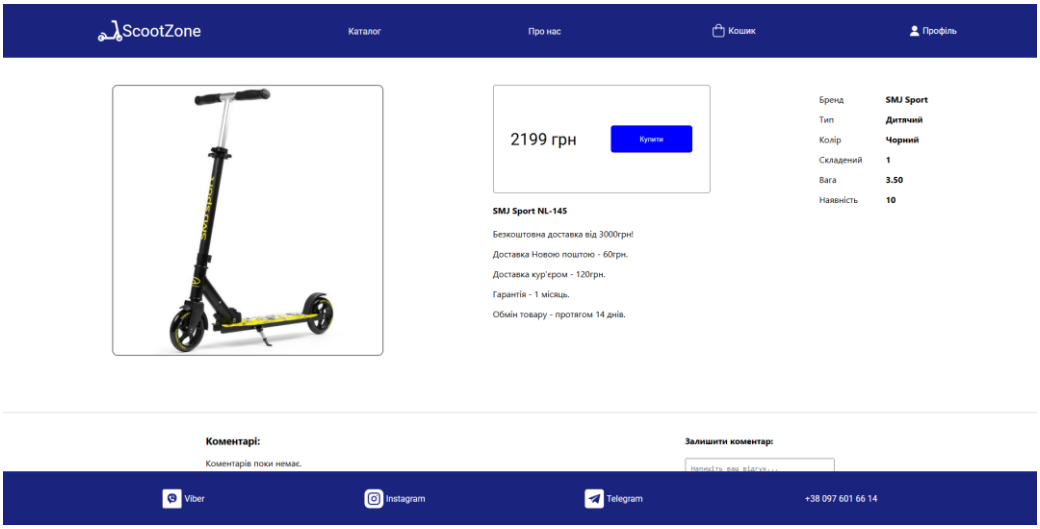

Рисунок 3.16 – Сторінка товару

Джерело: розроблено автором

Сторінка кошику. На сторінці кошику ми можемо побачити такі елементи сторінки як:

На рис. 3.18 ми можемо побачити сторінку кошика нашого магазину.

На рис. 3.19 ми можемо побачити код, який показує нам, як виводиться на екран товари, які були додані до кошика.

На рис. 3.20 ми можемо побачити метод, який додає товар до кошика. Для цього ми використовуємо метод post. Спочатку ми отримуємо дані, які ми будемо додавати до нашого кошика. Після цього ми формуємо запит до нашої бази даних. Я випадку успішного додавання товару до кошика, повертаються статус 200 (OK).

```
app.get('/api/products/:id', async (req, res) => {
  const { id } = req.params;
  console.log(`Запит до продукту з id: ${id}`);

  const query = 'SELECT * FROM products WHERE id = ?';

  try {
    const [results] = await pool.query(query, [id]);

    if (results.length === 0) {
      return res.status(404).json({ message: 'Товар не знайдено' });
    }

    res.json(results[0]); // повертаємо один товар
  } catch (err) {
    console.error('Помилка при запиті до бази даних:', err.message);
    res.status(500).json({ message: 'Помилка сервера' });
  }
});
```

Рисунок 3.17 – Отримання даних конкретного товару

Джерело: розроблено автором

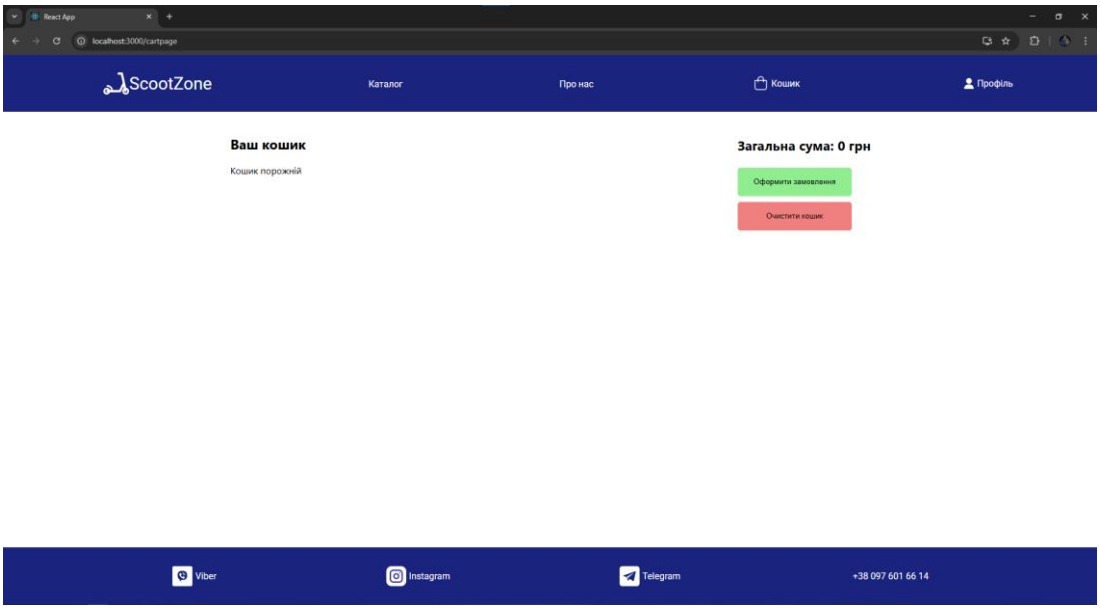

Рисунок 3.18 – Сторінка кошику магазину

Джерело: розроблено автором

```
<div>
    <h2>Ваш кошик</h2>
    {cartItems.length === 0 ? (
        <p>Кошик порожній</p>
    ) : (
        <div className="cart-items">
            {cartItems.map(item => (
                <div key={item.id} className="cart-item">
                    <img src={item.picture} alt={item.name} className="cart-image" />
                    <div className="cart-info">
                        <h4>{item.name}</h4>
                        <p>{item.price} грн</p>
                        <p>Кількість: {item.quantity}</p>
                    </div>
                </div>
            ))}
        </div>
    )}
</div>
```

Рисунок 3.19 – Вивід на екран доданих товарів

Джерело: розроблено автором

```
app.post('/api/cart', verifyToken, async (req, res) => {
  const { productId } = req.body;
  const userId = req.user.id;
  const quantity = 1;

  console.log('Додавання в кошик:', { userId, productId });

  if (!productId) {
    return res.status(400).json({ error: 'Product ID відсутній' });
  }

  const query = `
    INSERT INTO cart_items (user_id, product_id, quantity)
    VALUES (?, ?, ?)
    ON DUPLICATE KEY UPDATE quantity = quantity + 1
  `;

  try {
    await pool.query(query, [userId, productId, quantity]);
    res.status(200).json({ message: 'Товар додано до кошика' });
  } catch (err) {
    console.error('Помилка при додаванні в кошик:', err);
    return res.status(500).json({ error: err.message });
  }
});
```

Рисунок 3.20 – Додавання товару до кошика

Джерело: розроблено автором

На рис. 3.21 ми можемо побачити метод виводу доданих товарів до кошика. Для цього ми використовуємо метод get. Спочатку через req.user.id ми отримуємо id авторизованого користувача. Після цього ми виконуємо запит до нашої бази даних, для отримання усіх товарів, які були додані конкретним користувачем.

```
app.get('/api/cart', verifyToken, async (req, res) => {
  const userId = req.user.id;

  const query = `
    SELECT ci.id, ci.quantity, p.id AS product_id, p.name, p.price, p.picture
    FROM cart_items ci
    JOIN products p ON ci.product_id = p.id
    WHERE ci.user_id = ?
  `;

  try {
    const [results] = await pool.query(query, [userId]);
    res.status(200).json({ cart: results });
  } catch (err) {
    console.error('SQL error:', err);
    return res.status(500).json({ error: err.message });
  }
});
```

Рисунок 3.21 – Вивід на екран доданих товарів

Джерело: розроблено автором

Рекомендаційна система. На рис. 3.22 представлено сектор з рекомендованими товарами, який знаходиться на сторінці товару.

Рисунок 3.22 – Сектор рекомендаційних товарів

Джерело: розроблено автором

На рис. 3.23 знаходиться метод запису перегляду товару. Для цього ми використовуємо метод post. Спочатку через req.params.id та req.user.id ми отримуємо id товару та id користувача для формування нашого запиту для запису перегляду товарів в базу даних. Після цього ми формуємо наш запит до бази даних та посилаємо його.

```
app.post('/api/products/:id', verifyToken, async (req, res) => {
  console.log('Виконання функції запису в таблицю переглянутих товарів');
  const productId = req.params.id;
  const userId = req.user.id;

  console.log(`id Користувача: ${productId}`);
  console.log('productId:', productId);


  console.log(`Запит на запис перегляду товару: userId = ${userId}, productId = ${productId}`);

  if (!userId || !productId) {
    return res.status(400).json({ message: 'Невірні дані для запису перегляду' });
  }

  try {
    const query = 'INSERT INTO product_views (user_id, product_id) VALUES (?, ?)';
    await pool.query(query, [userId, productId]);

    console.log('Запис успішно додано в таблицю product_views');
    res.status(201).json({ message: 'Запис успішно додано в product_views' });
  } catch (err) {
    console.error('Помилка при виконанні запиту:', err);
    return res.status(500).json({ message: 'Помилка при записі перегляду товару', error: err });
  }
});
```

Рисунок 3.23 – Запис переглянутого товару

Джерело: розроблено автором

На рис. 3.24 знаходиться метод виводу рекомендованих товарів на екран. Для цього ми використовуємо метод get. Покрокове виконання методу виводу рекомендованих товарів:

- за допомогою req.user.id ми отримуємо id користувача;
- відправляємо запит до бази даних, для отримання останнього переглянутого товару і отримуємо його характеристики;
- відправляємо ще один sql-запит до бази даних. Цей запит шукає подібні товари в базі даних. Подібність (similarity) рахується як

сума збігів по типу і бренду товару. Кожен збіг дає +1 до схожості. Виключанням можуть бути товари, які вже до цього були переглянуті. Також додане обмеження до 5 товарів;

– повернення масиву з рекомендованими товарами

```
app.get('/api/recommendations', verifyToken, async (req, res) => {
  const userId = req.user.id;

  try {
    const [viewed] = await pool.query(
      `SELECT p.type, p.brand, p.color, p.foldable
       FROM product_views v
       JOIN products p ON v.product_id = p.id
       WHERE v.user_id = ?
       ORDER BY v.viewed_at DESC
       LIMIT 1`,
      [userId]
    );

    if (viewed.length === 0) {
      return res.json([]);
    }

    const { type, brand, color, foldable } = viewed[0];

    const [results] = await pool.query(
      `
      SELECT p.*,
        (
          (p.type = ?)*1 +
          (p.brand = ?)*1
        ) AS similarity
      FROM products p
      WHERE p.id NOT IN (
        SELECT product_id FROM product_views WHERE user_id = ?
      )
      ORDER BY similarity DESC
      LIMIT 5
      `,
      [type, brand, userId]
    );

    res.json(results);
  } catch (err) {
    console.error('Помилка при отриманні рекомендацій:', err);
    return res.status(500).json({ message: 'Помилка при отриманні рекомендацій', error: err });
  }
});
```

Рисунок 3.24 – Вивід рекомендованих товарів на екран

Джерело: розроблено автором

3.3 Тестування

Було проведене модульне та функціональне тестування роботи веб-сайту. Перевірки модульного тестування:

- обрахунок вартості товарів у кошику;
- функція генерації JWT токена;
- функція підключення до бази даних;
- фільтрація товарів.

Під час тестування було виявлено, що всі функції працюють правильно та без помилок.

Перевірки функціонального тестування:

- реєстрація користувача;
- авторизація користувача;
- перегляд товару;
- додавання товару до кошика;
- перегляд кошика;
- видалення товарів з кошика;
- отримання рекомендацій.

Під час проведення функціонального тестування, було виявлено, що усі функції працюють правильно та без помилок.

3.4 Використання

Розроблений веб-сайт забезпечує зручний і зрозумілий інтерфейс для взаємодії з користувачем.

Головна сторінка: користувач може перейти на сторінки каталогу, кошику, про магазин, профілю(за умови авторизації), зареєструвати та авторизувати на сайті.

Каталог: користувач може побачити наявні товари в магазині, та за допомогою фільтрації знайти відповідний йому товар.

Сторінка товару: користувач може подивитись детальні характеристики товару. За умови проходження авторизації, користувач додатково може додати товар до кошика, залишити коментар до товару та побачити рекомендовані товари(при умові відвідування 5+ товарів на сайті).

Профіль: за умови проходження авторизації, користувач може побачити свої дані при реєстрації (логін та пошта), а також побачити свою історію покупок.

Кошик: за умови авторизації на сайті, користувач може додавати до кошику товари та оформлювати замовлення.

Про нас: користувач може прочитати інформації про магазин.

Висновки до розділу 3

В даному розділі було показано за допомогою яких засобів було реалізовано веб-сайт. Було наведені функції та їх пояснення, які використовуються на нашому веб-сайті. Було проведено тестування наявних функцій, для перевірки їх роботи. Була описана інструкція для користувача користуванням сайту.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи, було зроблено опис предметної області, визначення потенційних конкурентів розробленого продукту, для виявлення потенційних переваг над конкурентами. Було постановлене завдання на виконання кваліфікаційної роботи. Були визначенні функціональні та нефункціональні вимоги до веб-сайту.

Було проведено проектування структури веб-сайту, інтерфейсу та моделювання процесів веб-сайту.

В результаті виконання кваліфікаційної роботи, був реалізований веб-сайт для продажу самокатів. Були реалізовані базові функції для комфортного користування веб-сайту користувачем.

Було виявлено, що дана ніша має свої недоліки, при усуненні яких, на виході можна отримати продукт, який зможе конкурувати з наявними конкурентами на ринку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Самокат або мотоцикл? // Geon. URL: https://geon.ua/samokat-abo-mototsycle/ (дата звернення: 24.03.2025).

2. ТОРГСОФТ. URL: https://store.torgsoft.ua/blog/jakyj-magazyn-vidkryty-v-2025/?srsltid=AfmBOoppBWLECHck9avGfCycxUEUcIcEkwkZQtd0lMX9D2LeJHSvAdtG (дата звернення: 24.03.2025).

3. Samokat. URL: https://samokat.ua/ua/ (дата звернення: 24.03.2025).

4. Candy Boards. URL: https://candyboards.com.ua/ (дата звернення: 24.03.2025).

5. Flash Skate. URL: https://flash-skate.com.ua/ua/ (дата звернення: 24.03.2025).

6. Functional and Nonfunctional Requirements: Specification and Types // URL: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/ (дата звернення: 13.05.2025).

7. Functional vs. Non Functional Requirements // URL: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/ (дата звернення: 13.05.2025).

8. Use Case Diagram – Unified Modeling Language (UML) // URL: https://www.geeksforgeeks.org/use-case-diagram/ (дата звернення: 13.05.2025)

9. What is Sequence Diagram? // URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/ (дата звернення 13.05.2025)

10. Structured Analysis and Design Technique (SADT) // URL: https://segoldmine.ppi-int.com/taxonomy/term/844 (дата звернення 13.05.2025)

11. User Flow // URL: https://www.productplan.com/glossary/user-flow/ (дата звернення 13.05.2025)

12. The ultimate guide to User Flow Diagram // URL: https://medium.com/design-bootcamp/the-ultimate-guide-to-user-flow-diagram-b108d7de10d (дата звернення 13.05.2025)

13. Interface Structure Design // URL: https://creately.com/diagram/example/hsr7wu1h2/interface-structure-design (дата звернення 13.05.2025)

14. Activity Diagrams – Unified Modeling Language (UML) // URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/ (дата звернення 13.05.2025)

15. React (software) // URL: https://www.wikiwand.com/en/articles/React_(software) (дата звернення 16.05.2025)

16. Node.js // URL: https://www.wikiwand.com/en/articles/Node.js (дата звернення 16.05.2025)

17. MySQL: Unerstanding What It Is and How It's Used // URL: https://www.oracle.com/ua/mysql/what-is-mysql/ (дата звернення 16.05.2025)

18. HTML // URL: https://www.wikiwand.com/en/articles/HTML // (дата звернення 16.05.2025)

19. HTML (Hypertext Markup Language) // URL: https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language (дата звернення 13.05.2025)

20. CSS // URL: https://www.wikiwand.com/en/articles/CSS (дата звернення 16.05.2025)

21. JavaScript // URL: https://www.wikiwand.com/en/articles/JavaScript (дата звернення 16.05.2025)

22. React JSX // URL: https://www.geeksforgeeks.org/reactjs-jsx-introduction/ (дата звернення 16.05.2025)

23. What is SQL (Structured Query Language)? // URL: https://aws.amazon.com/what-is/sql/ (дата звернення 16.05.2025)

24. Мічківський С. Microsoft Office (Word, Excel, Outlook …) : навч. посіб. / С. Мічківський, Д. Балдик, В. Головань; Східноукр. нац. ун-т ім. В. Даля, Аграр. ф-т. – Київ : [Вид-во Східноукр. нац. ун-т ім. В. Даля], 2023. – 128 с. – URL: https://dspace.snu.edu.ua/handle/123456789/1723

25. Шаповалов С.М., Мічківський С.М. Розробка рекомендаційної системи для підбору співрозмовника в соціальній мережі // XII Міжнародна науково-технічна конференція «Проблеми Інформатизації» (м. Київ, 13 грудня 2018 року). – К.: Державний університет телекомунікацій, 2018. URL: https://dut.edu.ua/uploads/l_1701_65845854.pdf

26. Разиграєв В., Мічківський С. Рекомендаційна підсистема та багаторівнева архітектура в платформах інтернет-магазинів // Матеріали IV Міжнародної науково-практичної конференції «Бізнес-аналітика: моделі, інструменти та технології». 1-3 бер. 2023. – К.: НАУ, 2023. – 548 с. – С 357-359.