

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ**

Кафедра кібернетики та інформатики

КВАЛІФІКАЦІЙНА РОБОТА

освітній ступінь - «Бакалавр»

**на тему: Розробка веб-сайту інтернет-магазину продажу
спортивних товарів**

Виконав:

студент спеціальності

126 «Інформаційні системи та технології»

ОП «Інформаційні системи та технології»

Чубун Микола Петрович

Керівник:

Руденко Юлія Олександрівна

к. п. н., доцент кафедри кібернетики та інформатики

Рецензент:

Дегтярьова Неля Валентинівна

к. п. н., доцент кафедри інформатики Сумського
державного педагогічного університету імені
А.С.Макаренка

Суми – 2024

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет	Економіки і менеджменту
Кафедра	Кібернетики та інформатики
Освітній ступінь	«Бакалавр»
Спеціальність	<u>126 «Інформаційні системи та технології»</u> <u>ОП «Інформаційні системи та технології»</u>

Затверджую:
Завідувач кафедри _____ Світлана АГАДЖАНОВА

« _____ » _____ 2024 ____ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Чубуну Миколі Петровичу

1. Тема роботи: Розробка веб-сайту інтернет-магазину продажу спортивних товарів структурним.

2. База практичного дослідження: Сумський НАУ.

керівник роботи Руденко Юлія Олександрівна, к.т.н., доцент

затверджена наказом по університету від _____ « _____ » 2024 р. № _____ -н

3. Строк подання студентом закінченого проекту (роботи) _____ «23» червня 2024 року

4. Вихідні дані до проекту (роботи):

Результати аналітичних досліджень зарубіжних та вітчизняних вчених, нормативні документи та чинне законодавство, Інтернет-джерела.

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

- виконати постановку задачі;
- вибрати середовище для реалізації програмного додатку;
- провести аналіз наявних рішень для вирішення проблеми;
- розробити проєкт удосконалення наявної мережі;
- розробити віртуальну модель модернізованої корпоративної мережі;
- розробити веб-сайт для продажу спортивних товарів

6. Дата видачі завдання: _____ « _____ » 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту(роботи)	Строк виконання етапів проекту	Примітка
1	Визначення об'єкту, предмету дослідження, формулювання мети та задач кваліфікаційної роботи, складання плану	січень 2023 р.	виконано
2	Підбір та вивчення літературних джерел	січень 2023 р.	виконано
3	Узагальнення теоретичного матеріалу з обраної теми дослідження та представлення чорнового варіанту першого розділу кваліфікаційної роботи	січень 2023 р.	виконано
4	Збір та обробка фактичного матеріалу, узагальнення аналізу застосування досліджуваного питання в установі (переддипломна практика)	січень - березень 2023 р.	виконано
5	Оформлення теоретичної частини кваліфікаційної роботи, узагальнення аналітичної частини, захист звіту по переддипломній практиці	березень 2023 р.	виконано
6	Оформлення практичної частини кваліфікаційної роботи, розробка веб-сайту з продажу спортивних товарів	квітень 2023 р.	виконано
7	Підготовка висновків, пропозицій та узгодження з керівником	травень 2022 р.	виконано
8	Підготовка кінцевого варіанту кваліфікаційної роботи	до 3 червня 2023 р.	виконано
9	Опрацювання зауважень керівника, перевірка на автентичність	до 6 червня 2023 р.	виконано
10	Перевірка з академічної доброчесності кваліфікаційної роботи	9 червня 2023 р..	виконано
11	Представлення кваліфікаційної роботи на кафедру, деканат	23 червня 2023 р..	виконано
12	Підготовка доповіді, презентації до кваліфікаційної роботи та її попередній захист	до 27 червня 2023 р..	виконано
13	Захист кваліфікаційної роботи	28 червня 2023 р.	виконано

Здобувач вищої освіти

Микола ЧУБУН

Керівник роботи

Юлія РУДЕНКО

Перевірку на автентичність проведено

Надія БАРАНИК

Дипломна робота допущена до захисту

Маргарита ЛИШЕНКО

АНОТАЦІЯ

Чубун М.П. Розробка веб-сайту інтернет-магазину продажу спортивних товарів.

Кваліфікаційна робота зі спеціальності 126 «Інформаційні системи та технології» ОП Інформаційні системи та технології, СНАУ, Суми-2024р. – Рукопис.

Кваліфікаційна робота присвячена дослідженню актуальних питань, пов'язаних з створенням веб-додатків. У роботі проведений аналіз функціональних вимог пов'язаних зі створенням інтернет-магазину.

Актуальність теми обумовлена зростаючою популярністю онлайн-покупок, підвищеним інтересом до здорового способу життя, розвитком технологій, зростанням конкуренції та можливістю виходу на глобальний ринок. Веб-сайт дозволяє забезпечити зручний доступ до широкого асортименту товарів, аналізувати поведінку користувачів і пропонувати персоналізовані рекомендації, що підвищує задоволеність клієнтів та сприяє їхньому залученню.

Основні завдання створення веб-сайту з продажу спортивних товарів включають розробку зручного та функціонального дизайну, який забезпечить інтуїтивний доступ до всіх категорій товарів та функцій сайту. Інтеграція з різними платіжними системами дозволить забезпечити зручність та безпеку транзакцій для клієнтів.

Результатом дослідження буде створення ефективного веб-сайту для продажу спортивних товарів, який відзначатиметься зручним і функціональним дизайном, інтеграцією з різними платіжними системами, ефективною логістикою та доставкою. Сайт буде оптимізований для пошукових систем, що збільшить його видимість та відвідуваність.

Ключові слова: Веб-сайт, інтернет-торгівля

SUMMARY

Chubun M.P. Development of an E-commerce Website for Selling Sporting Goods.

Qualification work in the specialty 126 "Information Systems and Technologies," OP Information Systems and Technologies, SNAU, Sumy-2024. – Manuscript.

The qualification work is dedicated to researching current issues related to the creation of web applications. The work includes an analysis of the functional requirements related to the creation of an e-commerce website.

The relevance of the topic is determined by the growing popularity of online shopping, increased interest in a healthy lifestyle, technological development, growing competition, and the possibility of entering the global market. The website provides convenient access to a wide range of products, analyzes user behavior, and offers personalized recommendations, which enhances customer satisfaction and engagement.

The main tasks of creating a website for selling sporting goods include developing a convenient and functional design that ensures intuitive access to all categories of products and site features. Integration with various payment systems will ensure convenience and security of transactions for customers.

The result of the research will be the creation of an efficient website for selling sporting goods, characterized by a convenient and functional design, integration with various payment systems, efficient logistics, and delivery. The site will be optimized for search engines, increasing its visibility and traffic.

Keywords: Website, e-commerce.

Зміст

ВСТУП	7
РОЗДІЛ 1. ТЕНДЕНЦІЇ РОЗВИТКУ ЕЛЕКТРОННОЇ КОМЕРЦІЇ	8
1.1 Огляд інтернет-магазинів та їх функціональності.....	8
1.2 Технології розробки веб-сайтів для інтернет-магазинів.....	11
РОЗДІЛ 2. АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ ДО ВЕБ-САЙТУ	14
2.1 Визначення основних функціональних вимог	14
2.2 Вибір архітектури та технологій	20
2.3 Проектування інтерфейсу та користувацького досвіду	33
РОЗДІЛ 3. РОЗРОБКА ВЕБ-САЙТУ ІНТЕРНЕТ-МАГАЗИНУ	39
3.1 Створення бази даних товарів та клієнтів	39
3.2 Розробка кошика покупок та системи замовлень.....	43
3.3 Розробка модуля оплати та доставки	48
3.4 Розробка адміністративного інтерфейсу для управління магазином	50
РОЗДІЛ 4. ТЕСТУВАННЯ ТА ВІДЛАГОДЖЕННЯ.....	54
4.1 Верифікація та валідація веб-сайту	54
4.2 Тестування функціональності та безпеки.....	56
4.3 Виправлення помилок та відлагодження.....	58
ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62

ВСТУП

Наш час називають "інформаційним віком". Ця назва виникла через те, що найважливішим, цінним і необхідним ресурсом є інформація. Володіння інформацією та вміння своєчасно, швидко і чітко передавати необхідну інформацію клієнтам - ключ до успіху в реаліях сучасного бізнесу. В даний момент саме Інтернет здатен оперативнo і масово передавати текст, звук, зображення і навіть відео-ролики, тобто практично будь-яку інформацію. Аудиторія мережі Інтернет зростає з кожним роком. Користувачі мережі Інтернет є об'єктом особливого інтересу для рекламодавців, оскільки факт залишається фактом - серед користувачів мережі є багато потенційних клієнтів.

Сфера послуг є однією з найбільш перспективних галузей економіки та економічної діяльності. Вона охоплює широкий спектр економічної діяльності і за своїм функціональним призначенням не є єдиною системою. У процесі комерціалізації російської економіки та при російських реаліях: великі відстані, гостра нестача часу; особливе значення набуває така важлива форма реалізації товарів та послуг, як торгівля та реклама через мережу Інтернет. Існує велика кількість способів комерційного підходу до мережі Інтернет. У мережі можна рекламувати послуги, продавати товари або представляти споживачу і те, і інше. У сучасному бізнесі багато від чого залежить від самопрезентації компанії, її позиціонування на ринку наданих послуг та здатності знаходити нових клієнтів і ринки збуту. Одним з інструментів, як іміджевих, так і маркетингових, є наявність власного сайту в мережі Інтернет. З точки зору бізнесу, Інтернет - це сучасна рекламна платформа, що дозволяє забезпечити притік клієнтів.

Веб-сайт перетворює компанію в сучасний бізнес. Веб-сайт компанії є найважливішим джерелом інформації для потенційних клієнтів та людей, чий вигляд є громадськи важливим.

РОЗДІЛ 1

ТЕНДЕНЦІЇ РОЗВИТКУ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

1.1 Огляд інтернет-магазинів та їх функціональності

Для аналізу було обрано декілька сайтів зі схожою тематикою або призначенням. Одним із таких сайтів являється інтернет-магазин Rozetka, який представлений на рис. 1.1.

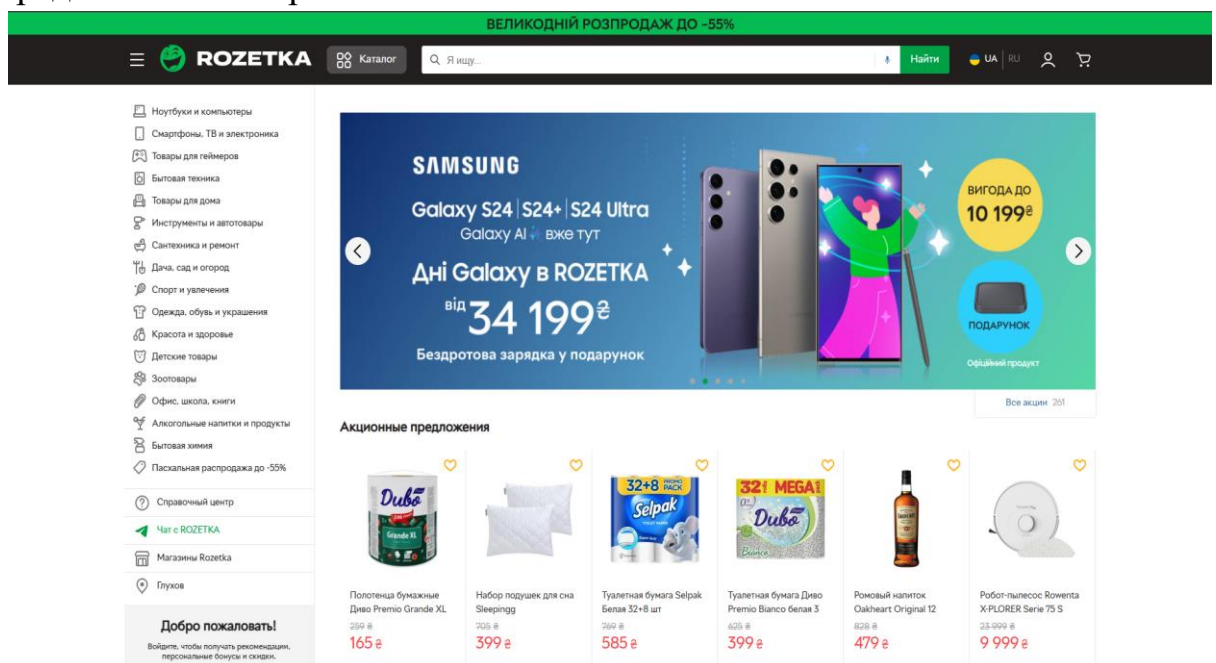


Рисунок 1.1 – Головна сторінка сайту Rozetka

Розглянемо інтернет-магазин Rozetka та його основні функціональні можливості. Сайт має розширений каталог товарів з різними категоріями та підкатегоріями. Фільтри для швидкого пошуку товарів за різними характеристиками (ціна, виробник, розмір, колір тощо). Також присутня детальна інформація про кожен товар, включаючи опис, характеристики, фотографії, відгуки користувачів та оцінки. Існує можливість додавання товарів у кошик, перегляд та зміна кількості товарів, видалення товарів з кошика. Присутня форма для введення контактних даних клієнта та адреси доставки. Можна вибрати метод оплати (готівкою, онлайн-платіжними системами,

кредитними картками тощо). Наявний кабінет користувача, можливість реєстрації і авторизації. Є можливість перегляду історії замовлень, зміни особистих даних, відкликання товарів, відгуки, бонусні програми, збереження списків бажань тощо. Присутні контентні розділи: Блог, новини, поради та статті про продукти, акції та розпродажі. Можна здійснити пошук товарів за ключовими словами. Наявні рекомендації товарів на основі попередніх покупок, переглядів, історії пошуку. Наявний адаптивний дизайн та оптимізація для мобільних пристроїв.

Тепер можна розібрати спеціалізований веб-сайт магазину по продажу спортивних товарів, одягу, добавок і т.д. Таким сайтом являється GymBeam, показаний на рис. 1.2.

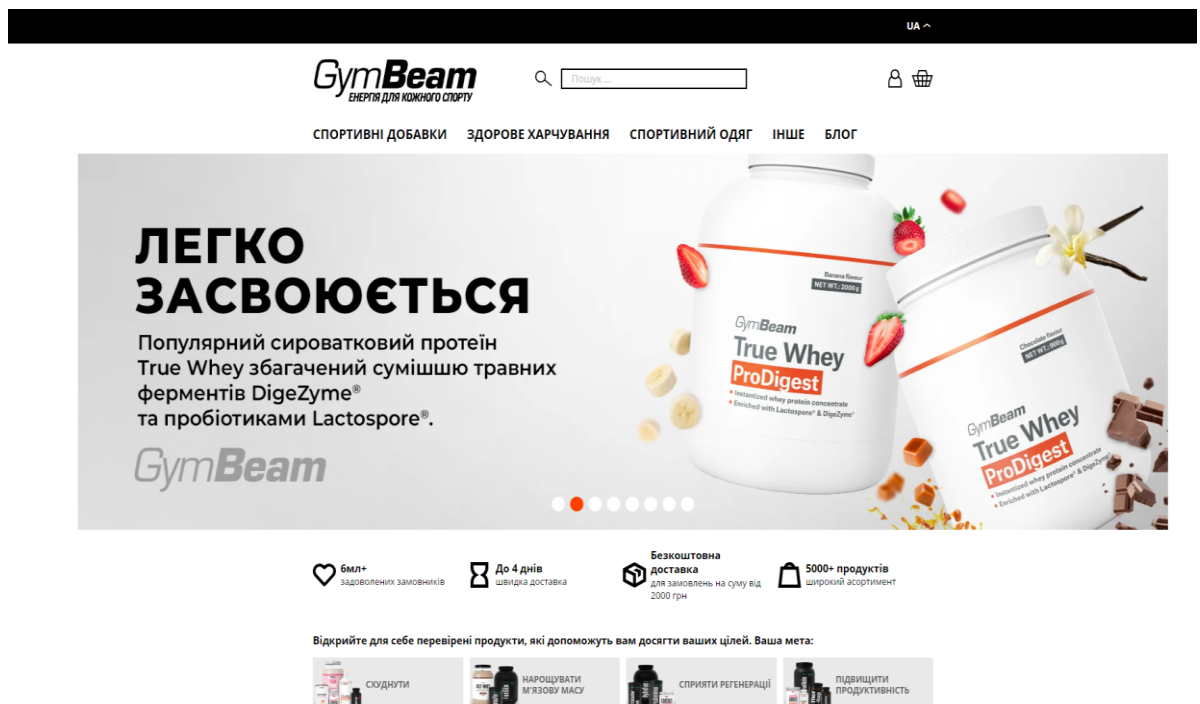


Рисунок 1.2 – Головна сторінка сайту GymBeam

GymBeam, функціонально схожий на сайт Rozetka. Сайт також має розширений каталог продуктів з розділенням на категорії, такі як протеїни, вітаміни, енергетики, спортивний одяг, аксесуари тощо. Присутні фільтри для швидкого пошуку продуктів за типом, брендом, ціною, обсягом, характеристиками тощо. Є детальна інформація про кожен продукт, включаючи

опис, склад, характеристики, фотографії, порівняння з аналогами, відгуки покупців та оцінки. Можна додавати товари до корзини, переглядати та редагувати замовлення. Вибір методів доставки та оплати, введення контактних даних та адреси доставки. Блог, статті, поради та інформація про тренування, дієтах, здоров'ї та фітнесі. Рекомендації продуктів відповідно до спортивних цілей та потреб користувача. Можливість обміну відгуками та досвідом користувачів через соціальні мережі.

Останнім ми розглянемо сайт Way4You, який як і GymBeam спеціалізується на продажі спортивних товарів. Головна сторінка веб-сайту представлена на рис. 1.3.

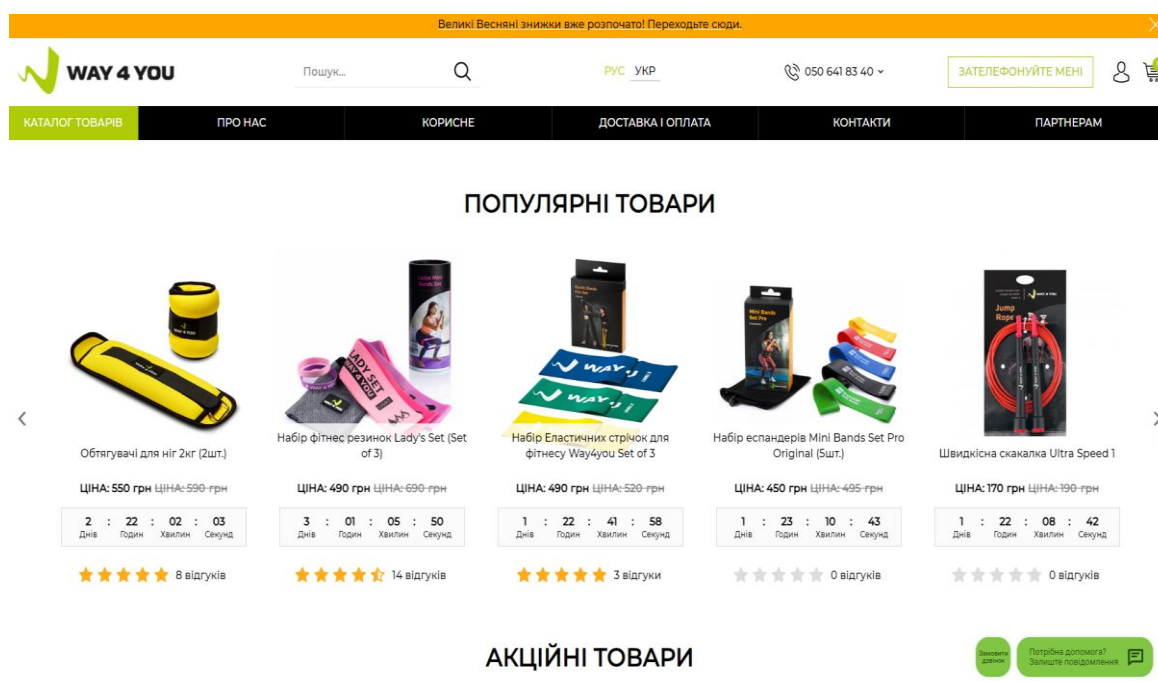


Рисунок 1.3 – Головна сторінка сайту Way4You

Цей веб-сайт має весь функціонал попередніх двох сайтів, але головна його відмінність це наявність блогу та інформаційного розділу. В блозі наводяться інформаційні статті, поради щодо тренувань, дієти.

Отже, сучасні інтернет-магазини надають широкий спектр функціональності, яка спрямована на полегшення процесу покупок для клієнтів та ефективне управління бізнесом для власників магазинів. Основні пункти, які можна виділити:

1. Каталог товарів: Широкий та добре структурований каталог з функціоналом фільтрації дозволяє швидко знаходити потрібні товари.
2. Сторінки товарів: Детальна інформація, фотографії, відгуки покупців допомагають у прийнятті рішення про покупку.
3. Корзина та оформлення замовлення: Зручний процес додавання товарів до корзини та оформлення замовлення з вибором методів доставки та оплати.
4. Особистий кабінет: Можливість реєстрації, перегляду історії замовлень, зміни особистих даних та отримання бонусів.
5. Доставка та підтримка: Різні варіанти доставки, відстеження замовлень, контактна інформація для зв'язку з підтримкою.
6. Інформаційні розділи: Блоги, статті та поради щодо товарів та здорового способу життя.
7. Мобільна версія: Адаптивний дизайн для зручності користувачів на мобільних пристроях.

Ці функції сприяють покращенню користувацького досвіду, збільшенню продажів та покращенню взаємодії з клієнтами, що робить інтернет-магазини ефективними та конкурентоспроможними на ринку електронної торгівлі.

1.2 Технології розробки веб-сайтів для інтернет-магазинів

Побудова веб-сайтів є складним та важливим процесом, що вимагає систематичного підходу та використання відповідних методів. Розглянемо декілька основних технологій розробки веб-сайтів.

HTML (HyperText Markup Language) - це мова розмітки, яка використовується для створення структури веб-сторінок. За допомогою HTML ви вказуєте браузеру, як відображати різні елементи на сторінці, такі як заголовки, абзаци, зображення, посилання тощо. Вона складається з різних тегів, які визначають тип контенту і його властивості. HTML вважається базовою

мовою для створення веб-сторінок, і вона часто використовується разом з CSS для оформлення сторінки і JavaScript для додавання інтерактивності.

CSS (Cascading Style Sheets) - це мова опису стилів, яка використовується для задання зовнішнього вигляду веб-сторінок, наприклад, кольору, шрифту, розміру тексту, відступів, вирівнювання елементів тощо. За допомогою CSS ви можете контролювати вигляд і розташування різних елементів HTML на сторінці, що дозволяє створювати привабливі та структуровані веб-дизайни. CSS застосовується разом з HTML для оформлення веб-сторінок, і він може бути вбудованим у HTML файл, знаходитися в окремому файлі стилів або використовуватися динамічно з допомогою JavaScript.

JavaScript - це високорівнева, інтерпретована мова програмування, яка використовується для розробки динамічного змісту інтерактивних веб-сторінок. Вона дозволяє додавати різноманітні функції та ефекти на веб-сторінки, такі як анімація, валідація форм, взаємодія з користувачем (наприклад, реагування на події введення миші або клавіатури), динамічне оновлення вмісту без перезавантаження сторінки, і багато іншого. JavaScript використовується разом з HTML і CSS для створення повноцінних веб-додатків і веб-сайтів. Він може виконуватися безпосередньо в браузері користувача або на серверному боці за допомогою платформи Node.js. JavaScript підтримується практично всіма сучасними браузерами і є однією з найпоширеніших мов програмування веб-розробки. Фреймворки та бібліотеки: Наприклад, React, Angular, Vue.js для побудови складних клієнтських додатків; Bootstrap, Foundation для швидкого створення адаптивного дизайну.

Система управління контентом (CMS) - це програмне забезпечення, яке дозволяє зручно створювати, редагувати, організовувати та публікувати вміст на веб-сайті без необхідності глибокого знання програмування або веб-технологій. CMS дозволяють користувачам керувати різними аспектами веб-сайту, такими як текстовий контент, зображення, відео, аудіо, блоги, форуми, коментарі, а також вибирати дизайн та оформлення сторінок. Деякі відомі приклади CMS включають WordPress, Joomla, Drupal, Shopify (для електронної комерції) та

багато інших. Користувачі можуть вибрати CMS відповідно до їхніх потреб і рівня технічних знань.

Backend мови програмування - це мови, які використовуються для розробки серверної частини веб-додатків. Ці мови відповідають за обробку запитів від клієнтів (браузерів або інших додатків) і виконання логіки, необхідної для відображення або обробки даних. До цих мов відносять: PHP, Python (Django, Flask), Ruby (Ruby on Rails), Node.js (JavaScript на сервері) для розробки серверної логіки та взаємодії з базою даних.

База даних - це структурована колекція даних, яка зберігається в електронній формі на комп'ютері або сервері та призначена для ефективного організованого доступу, оновлення та аналізу даних. Вона є важливою складовою сучасних інформаційних систем і веб-додатків, оскільки дозволяє зберігати, використовувати та керувати великим обсягом даних. Наприклад, MySQL, PostgreSQL, MongoDB, що використовуються для зберігання та обробки даних, що використовуються на веб-сайті.

Хмарні платформи та хостинг - це сервіси, які надають інфраструктуру для зберігання, керування та розгортання веб-додатків і ресурсів в Інтернеті. Основні відмінності між ними полягають у масштабованості, рівні доступу та керуванні, а також вартості. Прикладами хмарних платформ є Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, DigitalOcean для розгортання та керування веб-додатками в Інтернеті.

Веб-сервер - це програмне забезпечення, яке використовується для обробки та відповіді на HTTP-запити від клієнтів, таких як веб-браузери, і надсилання веб-сторінок або іншого вмісту назад до цих клієнтів. Основна функція веб-сервера - це надання веб-сторінок користувачам через Інтернет. Наприклад, Apache, Nginx, які відповідають за обробку запитів та доставку веб-сторінок користувачам.

РОЗДІЛ 2

АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ ДО ВЕБ-САЙТУ

2.1 Визначення основних функціональних вимог

Основні функціональні вимоги до створення сайту можуть значно варіюватися залежно від конкретної сфери діяльності сайту, його цілей та аудиторії. Прикладом реалізованих функціональних вимог виступатиме сайт GymBeam. Отже, виділимо основні функціональні вимоги до веб-сайту:

1.Каталог товарів: Система, що дозволяє класифікувати товари за категоріями (спортивні види, бренди, розміри тощо) та швидко знаходити потрібні товари. Каталог товарів повинен бути організований ієрархічно, з можливістю створення категорій та підкатегорій. Також у каталозі повинна бути можливість фільтрувати товари за різними параметрами, такими як бренд, ціновий діапазон, розмір, кольори, тип товару (наприклад, одяг, взуття, аксесуари) і т.д. Це дозволить користувачам швидко знаходити потрібні товари. Кожен товар у каталозі повинен мати як мінімум одне зображення високої якості, яке чітко відображає товар. Крім того, кожен товар повинен мати детальний опис, включаючи характеристики, матеріали, розміри, інструкції з використання, якщо потрібно. Користувачам можна дозволити додавати товари у власний список "Обране" для подальшого зручного перегляду та покупки. Приклад реалізації даних вимоги зображені на рис. 2.1. та 2.2.

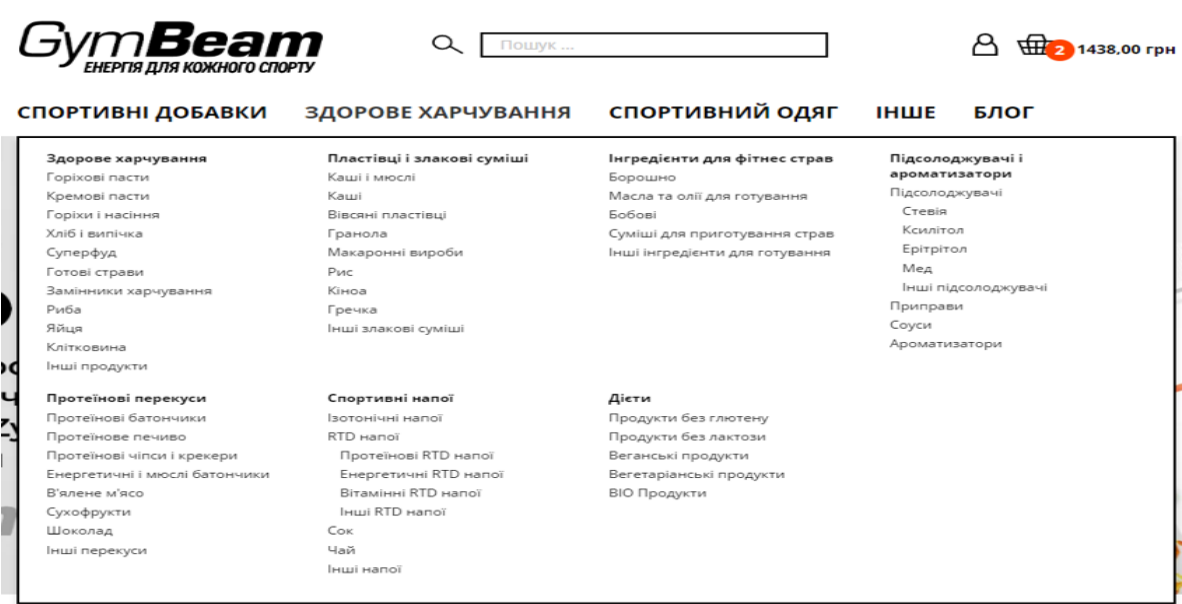


Рисунок 2.1 – Реалізація категорій товару

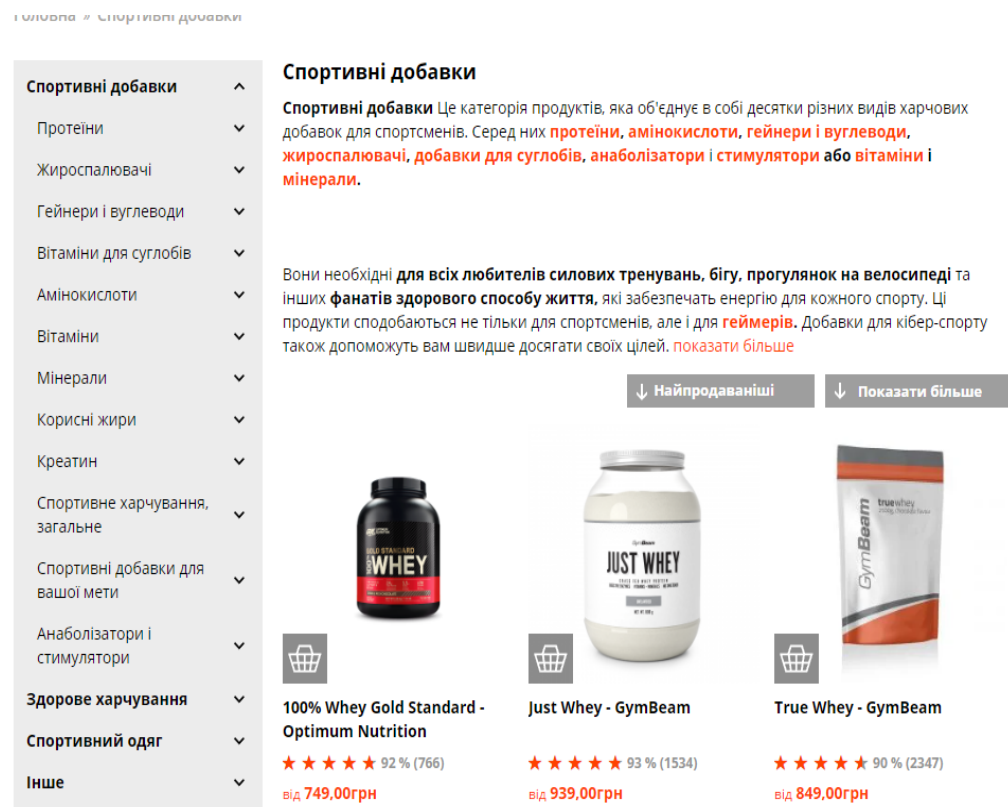


Рисунок 2.2 – Реалізація відповідних фільтрів.

2.Кошик і замовлення: Функціонал для додавання товарів у кошик, редагування кількості товарів та оформлення замовлення з доставкою та способом оплати. Кошик і замовлення на сайті з продажу спортивних товарів

мають бути зручними та інтуїтивно зрозумілими для користувачів. Кошик повинен дозволяти додавати товари з каталогу з можливістю зміни кількості, видалення товарів, а також відображення загальної вартості з урахуванням податків і доставки. При оформленні замовлення користувачеві має бути доступна форма для введення необхідних даних для доставки та оплати, вибір доступних методів доставки і оплати, а також можливість перевірити та підтвердити замовлення перед його підтвердженням і оплатою. Приклад кошику показаний на рис. 2.2.

[Виберіть ПОДАРУНОК!](#)

Кошик	Вартість доставки	Оплата
-------	-------------------	--------

Елемент	Опис	Ціна	Кількість	Всього до сплати	
	Вітамін С 1000 мг - GymBeam Таблетки: 90 табл	249,00грн	- 1 +	249,00грн	
	Компресійна футболка HG Armor Comp LS White - Under Armour Розмір одягу: M	1189,00 грн	- 1 +	1189,00 грн	

З продуктом також купують

Ваше замовлення
 всього до сплати 1 438,00ГРН

ДО СПЛАТИ 1 438,00ГРН

ДОДАТИ ПРОМОКОД: ▼

Замовте ще на 1762,00 грн та отримайте безкоштовно 2 пробники та сюрприз-подарунок до свого замовлення

Замовте ще на 562,00грн та отримайте безкоштовну доставку

Рисунок 2.3 – Вигляд кошику та оформлення замовлення.

3.Пошук і фільтрація: Пошук і фільтрація на сайті з продажу спортивних товарів відіграють ключову роль у зручному та швидкому пошуку потрібних товарів для користувачів. Функція пошуку повинна дозволити вводити ключові слова або фрази, а також використовувати автодоповнення для швидкого вибору товарів. Фільтрація товарів повинна бути розширеною і включати параметри, такі як категорії товарів, бренди, розміри, кольори, цінові діапазони, наявність на складі, рейтинг товарів та інші характеристики, які дозволяють користувачам точно визначити свої вимоги до товару. Додатково, система фільтрації повинна

бути оптимізована для швидкого відображення результатів і забезпечення плавного взаємодії з користувачем навіть при великій кількості товарів у каталозі.



Рисунок 2.4 – Вигляд пошуку та фільтрів товару.

4.Особистий кабінет користувача: Особистий кабінет користувача на сайті з продажу спортивних товарів - це інтерфейс, який дозволяє зареєстрованим користувачам отримати персоналізований досвід взаємодії з сайтом. У кабінеті користувач може переглядати та редагувати свої особисті дані, такі як ім'я, адреса, контактні дані, а також змінювати пароль. Також у кабінеті користувач може переглядати історію своїх замовлень, статуси доставки, отримувати повідомлення про акції та спеціальні пропозиції, керувати своїм списком бажань або зберігати товари для подальшого придбання. Додатково, можливість зворотнього зв'язку з підтримкою клієнтів чи відправка запитів з особистого кабінету також може підвищити зручність користування сайтом для клієнтів. Коротко кажучи, особистий кабінет користувача забезпечує зручний та персоналізований доступ до всіх сервісів та функціоналу сайту, що підвищує задоволення клієнтів і покращує їхній досвід покупок. Вигляд особистого кабінету показаний на рис. 2.5.

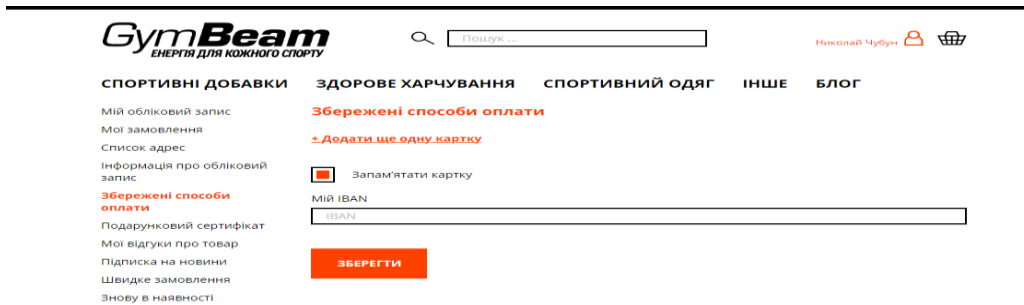


Рисунок 2.5 – Особистий кабінет користувача.

5. Оплата і доставка: Оплата і доставка на сайті з продажу спортивних товарів - це дві ключові складові, які впливають на задоволення клієнтів та їхнє довірче співвідношення з брендом. У вигляді оплати, важливо мати різноманітні методи платежу, такі як кредитні картки, платіжні системи, електронні гроші, а також можливість оплати при отриманні (післяплата), що підвищує комфорт для клієнтів. Для доставки, ефективна організація та розумні тарифи, вибір різних служб доставки з можливістю відстеження замовлення, а також оперативне вирішення питань, пов'язаних з доставкою (наприклад, затримка, втрата товару), грають важливу роль у задоволенні клієнтів та їхній лояльності до бренду. Крім того, повідомлення про статус замовлення та чітка інформація щодо термінів доставки сприяють створенню позитивного досвіду покупця та підвищують його довіру до компанії. Вигляд оплати та вибору адреси доставки зображений на рис. 2.6.

Кошик	Вартість доставки	Оплата
СПОСІБ ОПЛАТИ		
☐ VISA	Оплата картою онлайн ?	безкоштовно
☐ Google Pay	Google Pay ?	безкоштовно
☐	Оплата за реквізитами ?	безкоштовно
☐	Оплата при отриманні замовлення ?	59,00грн
☒ Я хочу додати платіжну адресу Микола Петрович Чубун Партизан Глухівщини, 3 Глухів (Сумська обл.), 00000 Україна +380993325231 РЕДАГУВАТИ		
Ваше замовлення ТОВАРУ НА СУМУ 889,00грн ВАРТІСТЬ ДОСТАВКИ 89,00грн НОВА ПОШТА - ПОШТОВЕ ВІДДІЛЕННЯ - НОВА ПОШТА ДО СПЛАТИ 978,00грн Адреса отримувача: Микола Петрович Чубун Партизан Глухівщини, 3 Глухів (Сумська обл.), 00000 Україна +380993325231 Спосіб доставки: Нова Пошта - Поштове відділення - Нова пошта Ціна: 89,00грн Selected pickup place: Глухів (Сумська обл.) 1 (Відділення №1: вул. Партизан Глухівщини, 3)		
ДОДАТИ ПРОМОКОД:		
☐ Я згоден з умовами використання та політикою конфіденційності. Перед закінченням замовлення необхідно підтвердити всі пункти.		
ПОВЕРНУТИСЯ ПІДТВЕРДИТИ ЗАМОВЛЕННЯ		

Рисунок 2.6 – Вигляд оплати та вибору адреси доставки.

6.Відгуки та рейтинги: Відгуки та рейтинги на сайті з продажу спортивних товарів є невід’ємною частиною взаємодії з клієнтами та формуванням довіри до бренду. Вони дозволяють клієнтам висловлювати свої враження та думки щодо придбаних товарів, що є важливим джерелом інформації для інших покупців. Позитивні відгуки стимулюють інших клієнтів до покупок, підвищують довіру до бренду та сприяють формуванню позитивного іміджу компанії. Рейтинги товарів допомагають клієнтам швидко оцінити якість товару, зручність використання та загальну задоволеність. Важливо створити систему, яка б забезпечувала реальність та достовірність відгуків, уникати фейкових оглядів та реагувати на негативні відгуки, пропонуючи вирішення проблеми або компенсацію. Працюючи з відгуками та рейтингами як з інструментами для взаємодії з клієнтами, компанія може підвищити якість своїх товарів і послуг, покращити обслуговування та збільшити лояльність клієнтів.

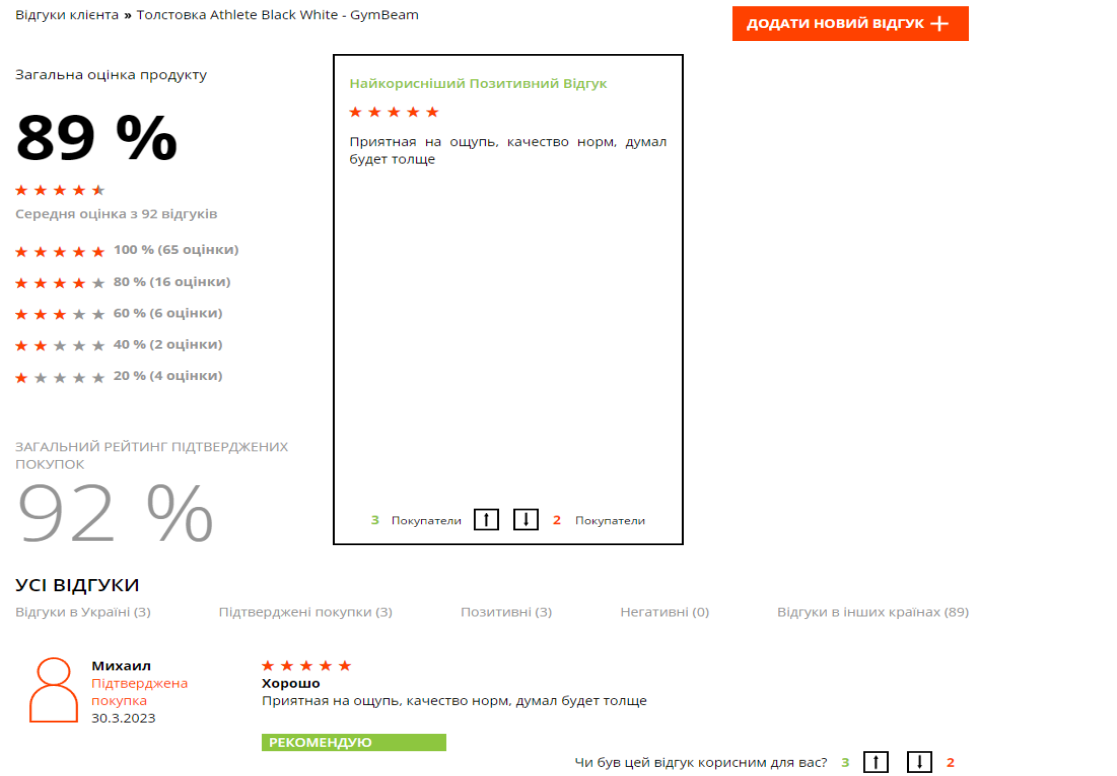


Рисунок 2.7 – Відгуки та рейтинг товару.

7.Зручний дизайн і мобільна сумісність: Зручний дизайн і мобільна сумісність є критично важливими аспектами для сайту з продажу спортивних

товарів, оскільки вони забезпечують зручність і доступність для широкого кола користувачів. Для початку, дизайн повинен бути чистим, інтуїтивно зрозумілим і естетично приємним, щоб стимулювати користувачів до подальшого перегляду товарів і оформлення замовлення. Мобільна сумісність включає в себе адаптивний дизайн, який забезпечує оптимальний вигляд та функціональність сайту на різних пристроях, таких як смартфони та планшети. Це означає адаптувати розміри шрифтів, розташування елементів і функціонал для зручного користування на малих екранах. Підтримка технологій для швидкої завантаження сторінок і оптимізація взаємодії з екранами дотикового керування також грають важливу роль у створенні позитивного враження від користування сайтом та підвищенні конверсії у вигляді покупок.

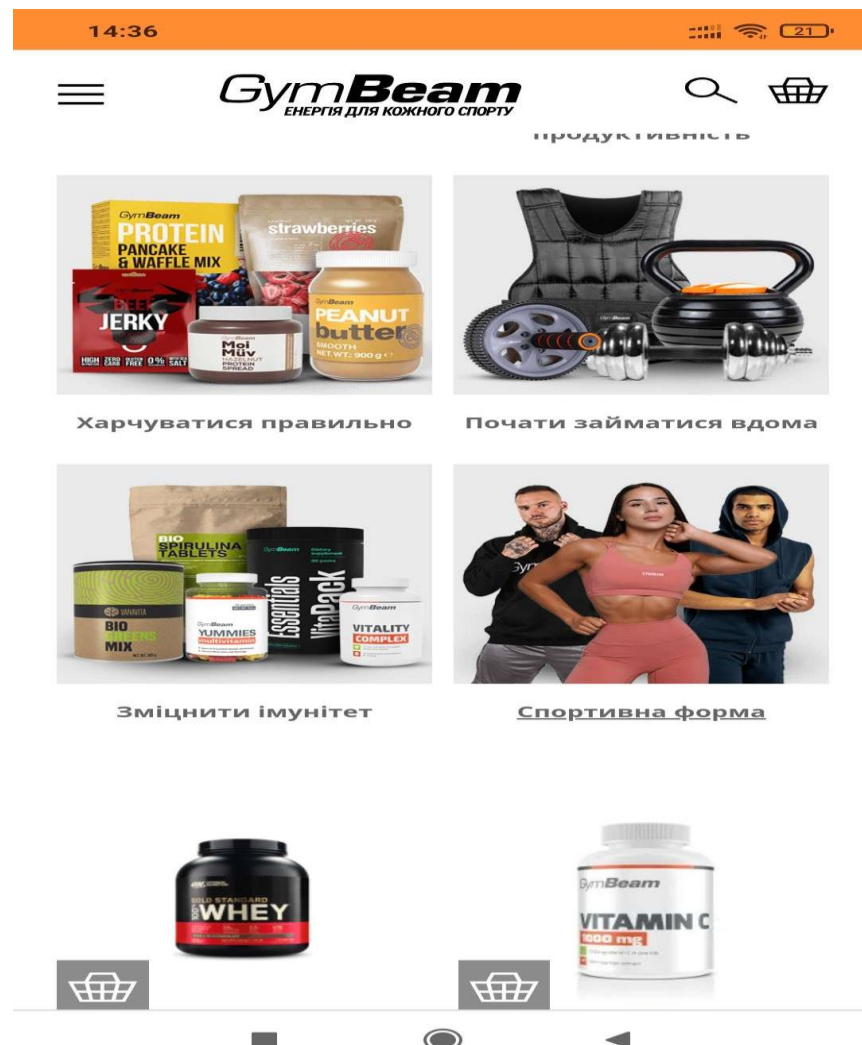


Рисунок 2.8 – Адаптація сайту під мобільні пристрої.

Відштовхуючись від моїх вмінь та тонкощів реалізації і буде розроблений веб-сайт згідно функціональних вимог.

2.2 Вибір архітектури та технологій

Для розробки веб-сайту був вибраний стек технологій MERN. Він включає в себе MongoDB, Express.js, React.js та Node.js. Проектування MERN додатків включає кілька етапів, які допомагають створити функціональний та ефективний веб-додаток. Ось основні етапи проектування MERN додатків:

1. Аналіз та визначення вимог: У цьому етапі проводиться детальний аналіз вимог до додатка. Визначаються функціональні та нефункціональні вимоги, вимоги до інтерфейсу користувача, бази даних та інші технічні вимоги.
2. Проектування архітектури: На цьому етапі визначається загальна структура додатка та його архітектура. Обираються відповідні компоненти MERN стеку та встановлюються зв'язки між ними. Розробляється схема бази даних та визначаються API точки доступу.
3. Розробка серверної частини (Back-end): У цьому етапі розробляються серверні компоненти за допомогою Node.js та Express.js. Встановлюються маршрутизація, обробка запитів та з'єднання з базою даних MongoDB. Реалізується бізнес-логіка додатка та виконуються потрібні перевірки та аутентифікація.
4. Розробка клієнтської частини (Front-end): На цьому етапі розробляється клієнтська частина додатка з використанням React.js. Створюються компоненти інтерфейсу користувача, реалізується логіка взаємодії з сервером через API запити. Додається стилізація та реалізуються функціональні можливості додатка.
5. Інтеграція та тестування: На цьому етапі проводиться інтеграція серверної та клієнтської частин додатка. Виконуються ретельні тести для перевірки працездатності, виявлення помилок та оптимізації.

Тестування може включати автоматичні тести, тестування одиниць, функціональні та інші види тестів.

6. Розгортання та підтримка: Після успішного завершення тестування додаток готовий до розгортання на веб-сервері. Встановлюються потрібні середовища та конфігурації для роботи додатка у виробничому середовищі. Після розгортання забезпечується підтримка та виправлення помилок, а також можлива подальша розширення функціоналу.

Враховуючи ці етапи проектування, розробка MERN додатків стає структурованим та ефективним процесом, що дозволяє створити потужні та масштабовані веб-додатки для різних сфер, у тому числі й медичної галузі.

Розглянемо мови розмітки та стилів, ними являються HTML та CSS.

HTML (HyperText Markup Language) - це мова розмітки, яка використовується для створення структури та вмісту веб-сторінок. Вона визначає різні елементи та їхню взаємодію на веб-сторінці. HTML використовується для створення структури веб-сторінок. Він складається з набору тегів, які описують різні елементи на сторінці. Основні теги HTML включають теги заголовків, параграфів, списків, посилань, зображень та таблиць. Крім того, HTML дозволяє вбудовувати мультимедійні елементи, форми введення даних, відео та аудіо.

CSS (Cascading Style Sheets) - це мова стилізації, яка використовується для визначення зовнішнього вигляду елементів HTML. CSS використовується для стилізації веб-сторінок. Він визначає зовнішній вигляд елементів, таких як кольори, шрифти, розташування, розміри тощо. CSS працює з HTML, використовуючи селектори для вибору елементів та задання стилів для цих елементів. Він дозволяє створювати привабливий та однорідний дизайн для вебсторінок, приклад з джерела наведено нижче на рис.2.9.

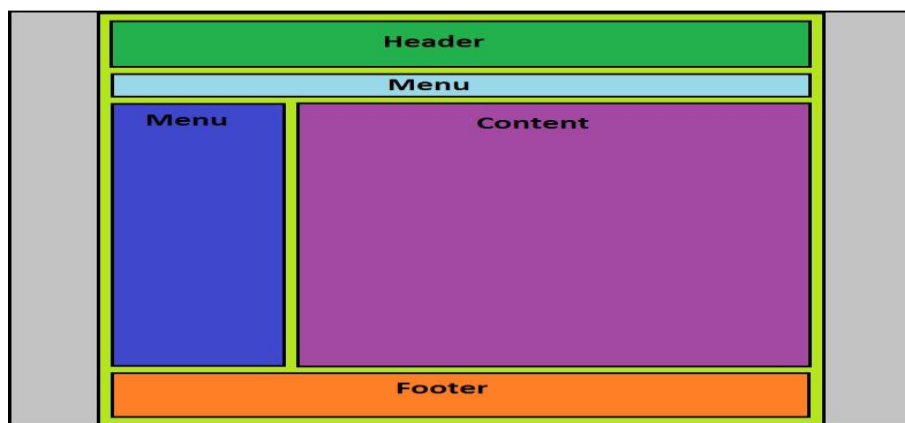


Рисунок 2.9. – Приклад розмітки веб-сторінки

Мовою програмування в нашому проєкті буде служити JavaScript, розглянемо цю мову та її особливості. JavaScript - це потужна мова програмування, яка використовується для розробки як клієнтської, так і серверної частини веб-додатків. Вона надає можливості для взаємодії з користувачем, обробки подій, маніпуляції веб-сторінками та взаємодії з сервером без перезавантаження сторінки. Ця мова програмування має динамічний характер, є Об'єктно-орієнтованою та базується на першокласних функціях. Вона була заснована в 1995 році для того щоб взаємодіяти з Java в браузерах.

JavaScript є основною мовою для веб-розробки, оскільки вона дозволяє створювати динамічні та інтерактивні веб-сторінки. Завдяки своїй широкій підтримці в браузерах, JavaScript став стандартом для реалізації різних функцій, таких як перевірка валідності форм, анімація, валідація даних та багато іншого.

Однією з найбільших переваг JavaScript є його гнучкість і розширюваність. Багато бібліотек та фреймворків, таких як React, Angular та Vue.js, побудовані на основі JavaScript, що дозволяє розробникам швидко створювати складні та масштабовані вебдодатки.

JavaScript також має багато ресурсів та спільнот, які забезпечують великий об'єм документації, пакетів та розширень, що допомагають розробникам прискорити процес розробки та розв'язання різних завдань. Однак, варто відзначити, що JavaScript має і свої обмеження:

- Вона є мовою зі слабою типізацією, що може викликати деякі проблеми при великих проектах з багатою логікою.
- Оскільки JavaScript виконується на клієнтській стороні, він піддається більшій вразливості і може бути підданий небажаним змінам або зламам.
- Обмежені можливості в обробці важких обчислень: JavaScript, зокрема з використанням середовища виконання Node.js, не є найкращим вибором для виконання важких обчислень, обробки великого обсягу даних, машинного навчання або складних алгоритмів.
- Обмежена підтримка старих браузерів: Деякі старі версії браузерів можуть не підтримувати останні функції та стандарти JavaScript.

Усупереч своїм недолікам, JavaScript продовжує займати провідні позиції в світі веб-розробки. Його постійний розвиток, широкі можливості та активна спільнота роблять його важливим інструментом для створення сучасних та інноваційних веб-додатків.

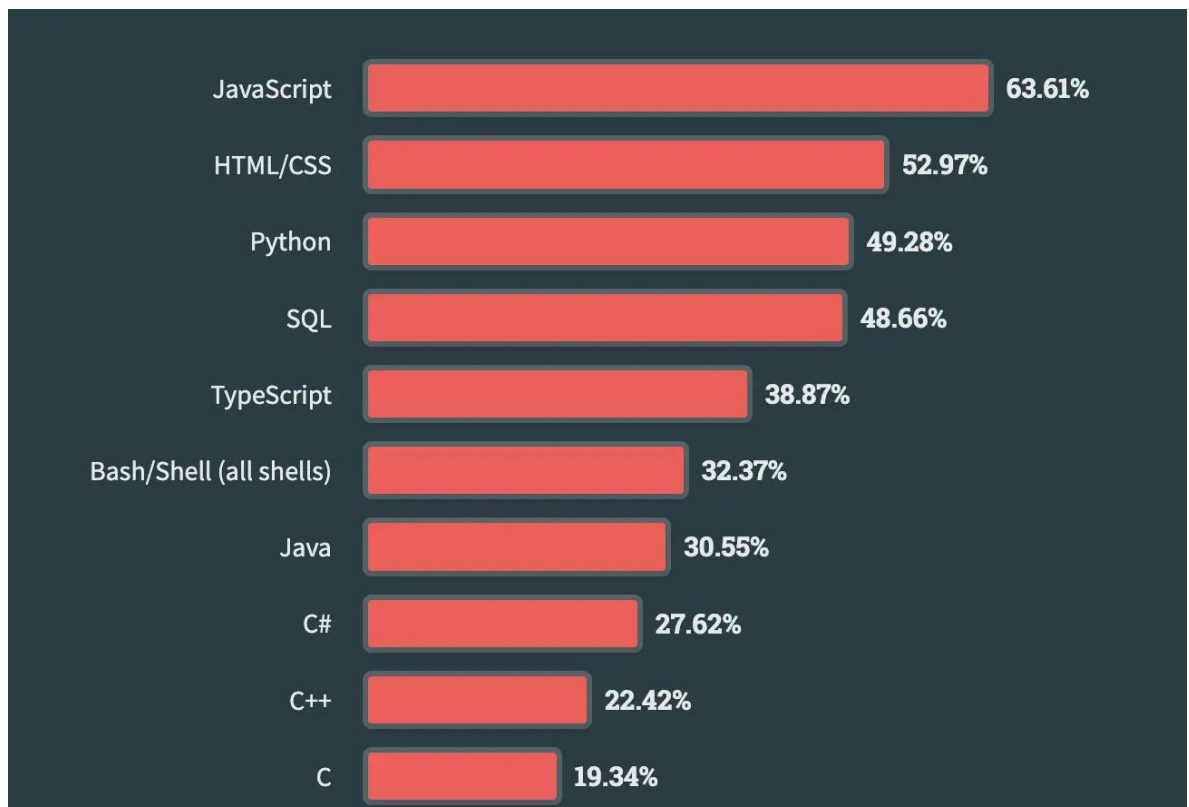


Рисунок 2.10. – 2023 Developer Survey від Stack Overflow

Відповідно до результатів опитування, які можна побачити на рис. 2.10 з джерела [4], проведеного компанією Stack Overflow у 2023 році, серед понад 70000 професійних розробників програмного забезпечення та студентів, які бажають поділитися своїми улюбленими технологіями розробки, виявлено, що 63,61% учасників назвали JavaScript своєю найчастіше використовуваною мовою програмування. Це статистично підтверджує високу популярність JavaScript як однієї з найбільш поширених технологій.

JavaScript, раніше відомий як ECMAScript, має багату історію випуску нових версій, які вводять нові можливості та поліпшення до мови програмування. Він був стандартизований в 1996 році компанією Netscape за допомогою ECMA (Ecma International - Європейська асоціація стандартизації інформаційно-комунікаційних систем). Починаючи з того часу, було випущено різні версії ECMAScript. Після випуску найбільш масштабної версії в 2015 році ECMA прагне видавати нові версії щороку. Тому ECMA змінила систему нумерації версій ECMAScript з номерів на роки випуску. Однак серед спільноти JavaScript більш популярною є система обчислення версій.

Тепер розглянемо фреймворк нашого проекту. React - це потужний та інноваційний фреймворк для розробки веб-додатків. Його основною ідеєю є створення користувацького інтерфейсу з високою ефективністю та перевикористовуванням компонентів. За допомогою React, розробка веб-додатків стає захоплюючим та ефективним процесом. Він пропонує декларативний підхід до розробки, де ви можете описувати, як ваші компоненти повинні виглядати у різних станах.

Одним з найважливіших аспектів React є віртуальний DOM, який забезпечує швидкий та ефективний рендеринг компонентів. React вміє "розуміти" лише ті зміни, які потрібно внести до DOM, тим самим забезпечуючи оптимальну продуктивність.

Крім того, React має широку екосистему із багатьма додатковими бібліотеками та інструментами, які допомагають вам у розробці. Він також

підтримує концепцію компонентного підходу, що сприяє легкості розширення та підтримці вашого коду.

Завдяки своїм перевагам та популярності, React став одним з найулюбленіших виборів для розробки сучасних веб-додатків. Використовуючи React, ви можете створювати надзвичайно масштабовані та високоякісні додатки, які здатні задовольнити навіть найвибагливіші потреби користувачів.

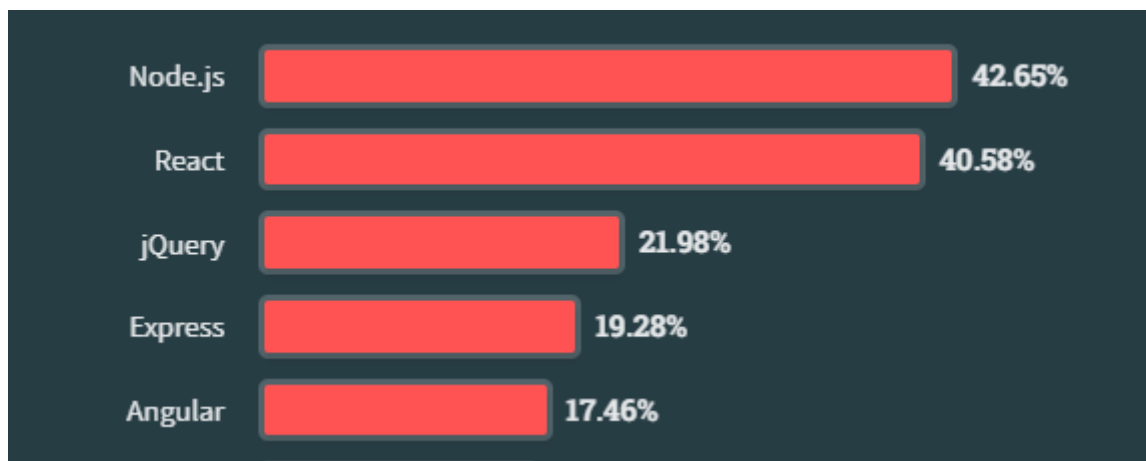


Рис. 2.11. Найживаніші веб-фреймворки у 2023 році

Знову звертаючись до статистики яку щорічно підбиває компанія Stack Overflow, на рис. 2.3 з джерела [\[4\]](#), можна побачити що React.js являється одним з найпопулярніших фреймворків серед веброзробників. Головні принципи використання React включають:

- Компонентна архітектура: React базується на ідеї розбиття інтерфейсу на невеликі та перевикористовувані компоненти. Компоненти в React використовуються для побудови складних інтерфейсів шляхом комбінування простих компонентів.
- Віртуальний DOM: React використовує віртуальний DOM, що дає змогу ефективно оновлювати тільки необхідні елементи на сторінці, зменшуючи затрати на маніпулювання реальним DOM.

- Одностороннє зв'язування даних: React пропонує одностороннє зв'язування даних, що означає, що дані відображаються від батьківського компонента до дочірніх компонентів, що полегшує керування станом додатка.
- Використання JSX: JSX є розширенням синтаксису JavaScript, яке дозволяє описувати структуру інтерфейсу у вигляді компонентів безпосередньо в коді JavaScript. Це полегшує розробку та читання коду.

Незважаючи на численні переваги, React також має кілька недоліків:

- Навчання та оволодіння React може вимагати певного часу та зусиль, особливо для розробників, які не мають досвіду з компонентною архітектурою.
- Висока складність додатків: Зі зростанням розміру та складності додатків, керування станом та взаємодія між компонентами може стати складнішою задачею.
- Екосистема та вибір інструментів: Вибір додаткових бібліотек, інструментів та підходів може бути викликом, оскільки велика кількість доступних варіантів може призвести до плутанини та необхідності додаткового дослідження.
- Першочерговість веб-розробки: React зосереджений переважно на розробці веб-інтерфейсів, тому для розробки інших типів додатків, таких як мобільні або настільні, можуть знадобитись додаткові інструменти та бібліотеки.

Незважаючи на ці недоліки, React продовжує бути популярним вибором для багатьох розробників завдяки своїм перевагам у швидкості, продуктивності та перевикористовуванні компонентів, приклади яких можна побачити на рис. 2.4 та 2.5 з джерела [\[5\]](#).



Рис. 2.12. – Приклад створення простого компоненту в React

Компоненти використовують метод `render()`, який обробляє вхідні дані та повертає вміст, що відображатиметься для користувача. У даному прикладі використовується JSX - синтаксис, схожий на XML. Звернення до переданих в компонент вхідних даних здійснюється за допомогою методу `render()` та `this.props`.

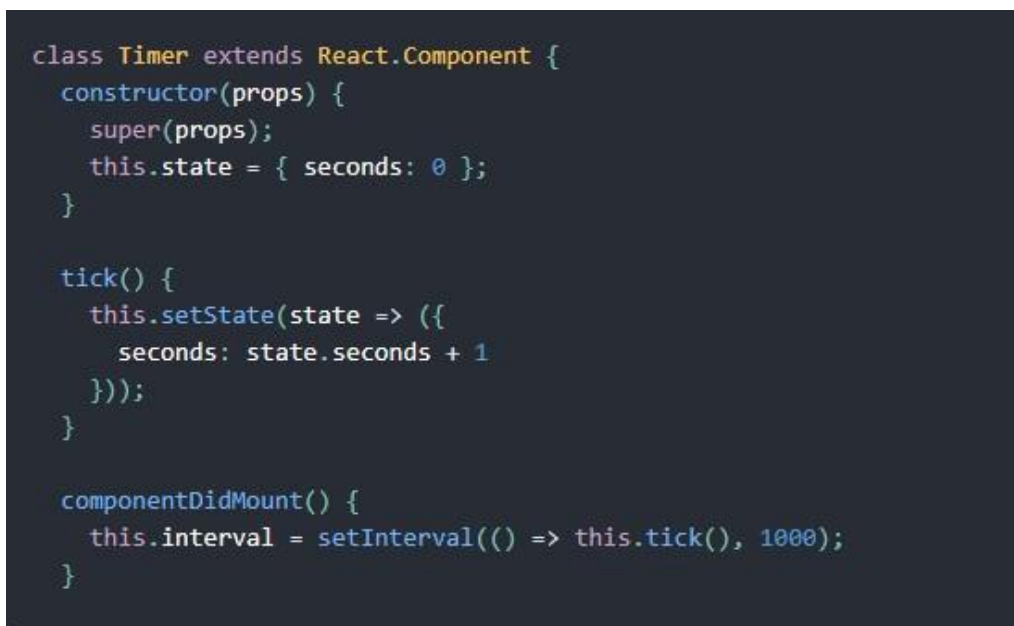


Рис. 2.13 – Приклад React компоненту зі станом

Крім прийняття вхідних даних (доступних через `this.props`), компоненти також можуть зберігати внутрішні дані в стані (`this.state`), як зображено на рис. 2.4. Коли дані стану компонента змінюються, розмітка автоматично оновлюється шляхом виклику функції `render()`.

Середовищем для виконання JavaScript в нашому проєкті буде Node.js. Node.js - це відкрите середовище виконання JavaScript, що дозволяє виконувати JavaScript-код поза межами веб-браузера. Це потужний інструмент, заснований на двигуні V8, розробленому компанією Google для виконання JavaScript в браузері Chrome. Однак, Node.js відрізняється своїм підходом до виконання коду та забезпечує високу продуктивність.

Основою Node.js є однопотокова подієва модель, що означає, що він може обробляти події асинхронно, не блокуючи виконання коду, принцип роботи можна побачити на рис. 2.6 з джерела [6]. Це робить його ідеальним вибором для написання серверних додатків та іншого програмного забезпечення,

що може вимагати великої навантаженості.

Node.js надає вбудовану бібліотеку вводу-виводу, яка дозволяє виконувати операції вводу-виводу асинхронно. Це включає роботу з мережевими запитами, файловою системою та іншими джерелами даних. Такий підхід дозволяє створювати ефективні та швидкі додатки.

Однією з найбільших переваг Node.js є його здатність обробляти багато запитів одночасно. Він використовує неблокуючий ввід-вивід та потоки, що забезпечує ефективне виконання операцій у багатопотоковому середовищі. Крім того, Node.js має широкий вибір сторонніх модулів і використовує пакетний менеджер npm, що дозволяє швидко розгортати проєкти та використовувати готові компоненти. Node.js знайшло застосування в розробці серверних додатків, мережесервісів, інструментів командного рядка, мікросервісів та іншого програмного забезпечення, де швидкість та ефективність є критичними факторами. Його популярність продовжує зростати завдяки його потужності та гнучкості.

Node Package Manager (NPM) - це пакетний менеджер для середовища Node.js. Він є стандартним інструментом, який допомагає в управлінні залежностями проєкту та установці зовнішніх пакетів. NPM дозволяє швидко знайти, завантажити, встановити та оновити пакети з репозиторію NPM, який є найбільшою колекцією відкритих пакетів для JavaScript. За допомогою NPM

розробники можуть легко управляти залежностями свого проекту. Вони можуть визначити необхідні пакети та їх версії у файлі `package.json`. NPM автоматично завантажує ці пакети з репозиторію та встановлює їх у проект. Крім того, NPM дозволяє керувати версіями пакетів, оновлювати їх та вирішувати конфлікти між залежностями. Використовуючи NPM, розробники можуть спростити розгортання та розробку своїх проектів, використовуючи готові пакети і модулі, які відповідають їх вимогам. Крім того, NPM також дозволяє розробникам публікувати власні пакети, щоб інші розробники могли їх використовувати. В цілому, NPM є важливим інструментом в екосистемі Node.js, який допомагає розробникам ефективно управляти залежностями та підтримувати структуровані проекти.

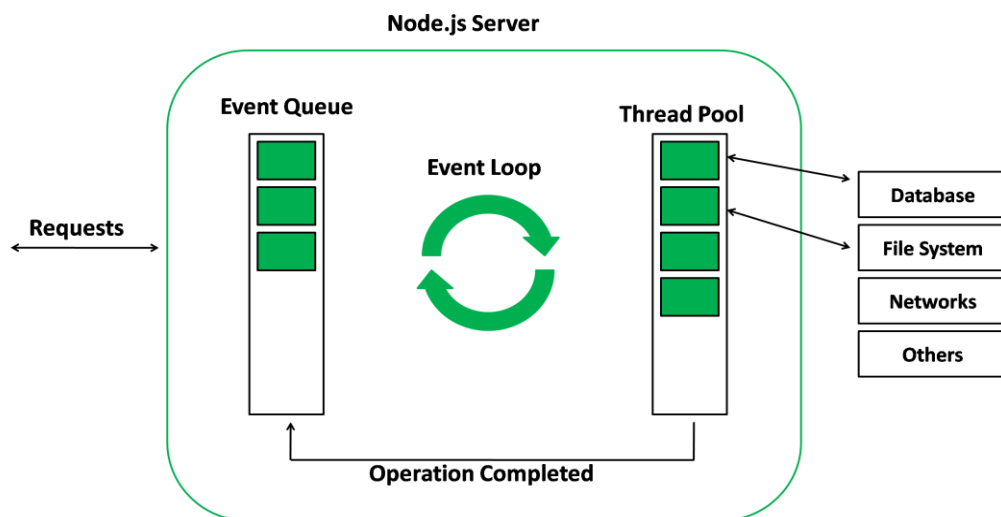


Рис. 2.14. –Принцип роботи Event loop в середовищі Node.js

Бекенд фреймворк Express - це веб-фреймворк для розробки веб-додатків на основі Node.js. Він надає простий та інтуїтивний спосіб побудови веб-серверів і API, дозволяючи розробникам швидко створювати потужні та масштабовані додатки.

Одна з головних переваг Express - це його мінімалістичний підхід. Він надає лише основні функції і можливості, не нав'язуючи жорстких правил і обмежень. Це дає розробникам велику свободу в побудові архітектури свого додатка і виборі необхідних компонентів.

Express пропонує простий і зрозумілий API для обробки маршрутів (routes) і обробки запитів (request) та відповідей (response). Розробники можуть легко визначати маршрути, обробники запитів та виконувати різні дії, такі як отримання даних з бази даних, відправлення відповідей клієнту, рендерінг сторінок та багато іншого.

Express також підтримує велику кількість сторонніх модулів (middleware), які допомагають розширити його функціональність. Middleware - це функції, які обробляють запити перед тим, як вони досягнуть основного обробника запитів. Це дозволяє виконувати різні завдання, такі як аутентифікація, логування, обробка помилок і багато іншого.

Завдяки своїй популярності та активній спільноті, Express має велику кількість ресурсів, документацію, приклади та плагіни, що дозволяє розробникам ефективно працювати з цим фреймворком. Він використовується для розробки різноманітних веб-додатків, включаючи веб-сайти, API, мікросервіси та інші типи додатків.

Express використовує методи HTTP (GET, POST, PUT, DELETE) та шляхи до ресурсів для реалізації маршрутизації. Кожен маршрут може мати власне проміжне програмне забезпечення, яке обробляє пов'язані з ним запити. Проміжне програмне забезпечення виконує специфічні дії, такі як перевірка аутентифікації, обробка даних або бізнес-логіка. Їх можна додавати до маршрутів за допомогою методів `.use()`, `.get()` або `.post()`, залежно від потреб рівня проміжного програмного забезпечення, на якому вони застосовуються.

```
// GET method route
app.get('/', function (req, res) {
  res.send('GET request to the homepage');
});

// POST method route
app.post('/', function (req, res) {
  res.send('POST request to the homepage');
});
```

Рис. 2.15. – Приклад обробки маршрутів в Express В Express

В Express також існує можливість обробки помилок за допомогою спеціального проміжного програмного забезпечення, приклад можна побачити на рис. 2.7. Це проміжне програмне забезпечення можна додати до додатка та воно буде викликане у випадку виникнення помилки. Воно отримує аргументи, такі як помилка, об'єкт запиту та об'єкт відповіді, і здатне виконувати різні дії, такі як відправлення користувачеві повідомлень про помилку або їх журналювання.

```
function errorHandler(err, req, res, next) {  
  if (res.headersSent) {  
    return next(err);  
  }  
  res.status(500);  
  res.render('error', { error: err });  
}
```

Рис. 2.16 – Приклад обробки помилок в Express

Ну і базою даних в проєкті служить MongoDB. MongoDB - це документоорієнтована, NoSQL база даних, яка зберігає дані у вигляді документів у форматі BSON (Binary JSON). Вона використовує модель даних, яка дозволяє гнучко зберігати та організовувати дані без заданої схеми, що дає можливість швидко вносити зміни у структуру даних.

MongoDB часто використовується в парі з Mongoose. Mongoose - це об'єктно-документний (ODM) інструмент для Node.js, який надає зручний спосіб моделювання даних MongoDB у вигляді схем та взаємодії з базою даних. Він надає високорівневий API, який дозволяє розробникам зручно виконувати операції збереження, оновлення, видалення та запитів до даних MongoDB.

Ключові переваги використання MongoDB в парі з Mongoose:

- Зручне моделювання даних: Mongoose дозволяє визначати схеми для даних MongoDB, що спрощує моделювання та валідацію даних.

- Зручний API для взаємодії з базою даних: Mongoose надає високорівневий API, який спрощує виконання операцій збереження, оновлення, видалення та запитів до бази даних MongoDB.
- Middleware: Mongoose надає можливість використовувати middleware, яке дозволяє вам виконувати дії перед або після виконання операцій з даними.
- Керування відносинами між даними: MongoDB має можливість вкладеності та посилання між документами, а Mongoose надає зручний спосіб використовувати ці можливості.
- Підтримка запитів та агрегацій: MongoDB надає потужні можливості для виконання запитів та агрегаційних операцій.

Отже, сфера розробки веб-додатків дуже швидко та постійно еволюціонує. Останнім часом з'являється багато нових технологій та фреймворків і розробники мають швидко адаптовуватися під такі правила. Та навпаки, раніше вибір мови програмування був дуже обмеженим і не існувало такого варіанту який би можна було використовувати як на клієнті(фронтенд), так і на сервері(бекенд).

Проте, з появою Node.js як серверного середовища JavaScript, та багатьох фронтенд фреймворків, наприклад React, стало можливим розробляти повноцінне програмне забезпечення, використовуючи лише одну мову програмування. Тому для цього конкретного проекту вибір було зроблено на користь JavaScript та MERN-стеку, що включає в себе вищезазначені MongoDB, Express, React та Nodejs і дозволяє ефективно та зручно розробляти веб-додатки.

Основні переваги розробки використовуючи MERN-стек:

- Усі компоненти MERN-стеку використовують JavaScript, що дозволяє розробникам працювати з однією мовою на всіх рівнях стеку.
- MERN-стек підтримує прогресивний підхід до веброботки, особливо завдяки використанню React(SPA).

- MERN-стек дозволяє побудовувати масштабовані веб-додатки завдяки використанню MongoDB як бази даних, яка працює на основі документів.
- MERN-стек дозволяє використовувати багато готових модулів, пакетів та бібліотек для розширення функціональності веб-додатків.

Загалом, використання MERN-стеку надає розробникам потужні та гнучкі інструменти для швидкого розгортання веб-додатків.

2.3 Проектування інтерфейсу та користувацького досвіду

Аналіз та проектування інтерфейсу та користувацького досвіду для інтернет-магазину є критичним етапом у створенні продукту, який забезпечить задоволення клієнтів та сприятиме конверсії. Аналіз та проектування інтерфейсу і користувацького досвіду для інтернет-магазину є критичним етапом у створенні продукту, який забезпечить задоволення клієнтів та сприятиме конверсії. Поточний стан інтерфейсу вашого інтернет-магазину, безумовно, має деякі позитивні аспекти, такі як наявність хлібних крихт та фільтрів за статтю. Однак, завдання полягає в пошуку можливостей для посилення взаємодії з користувачем та покращення загального враження від використання продукту.

Розглянемо детальніше різні аспекти аналізу та проектування інтерфейсу та користувацького досвіду для вашого інтернет-магазину. Поглиблюємось у поточний стан і розглядаємо різні можливості для покращення. Для початку була розроблена діаграма робіт над веб-сайтом, були додані робітники та підвівся підрахунок ресурсів, все це зображено на рисунку 2.17.

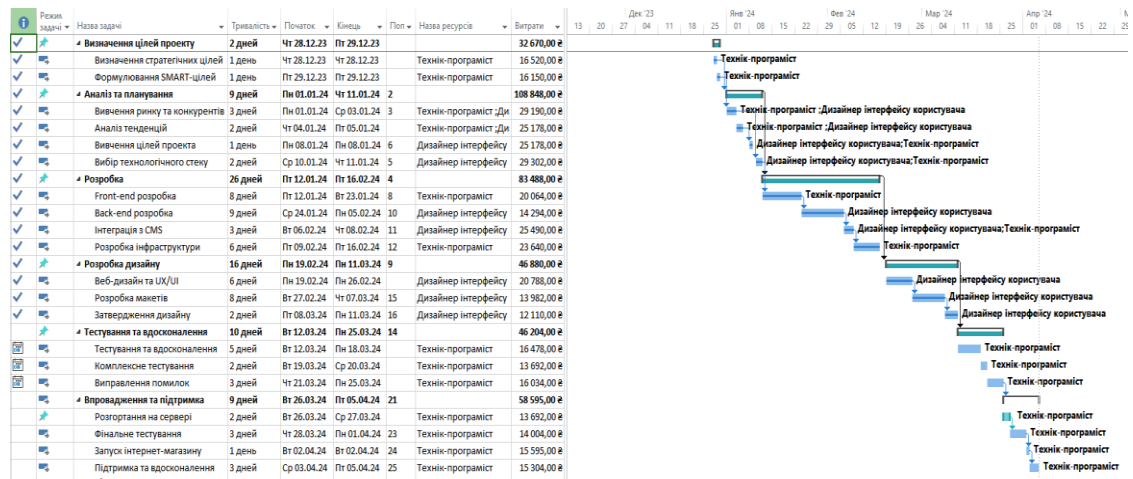


Рис. 2.17 – Роботи по створенню веб-сайту

Перейдемо до проектування інтерфейсу. Першою важливою вимогою є візуальні аспекти та анімації:

Візуальні аспекти та анімації грають важливу роль у створенні привабливого та привітного для користувачів дизайну веб-сайту. Колірна палітра, типографія, графічні елементи та композиція сторінок створюють перший враження про сайт і впливають на користувацький досвід. Важливо дотримуватися балансу між естетикою та функціональністю, забезпечуючи зручність навігації та легку сприйняття інформації. Крім того, використання анімаційних ефектів, таких як переходи між сторінками, анімація елементів при взаємодії користувача та інші мікро-анімації, допомагають зробити користувацький досвід більш динамічним та захоплюючим.

Наприклад, використання анімацій може допомогти акцентувати увагу користувача на важливих елементах, покращити візуальну структуру сторінок, зробити перехід між різними станами або функціями сайту більш плавним та природним. Важливо використовувати анімації обережно та розумно, уникаючи їх перевантаження, щоб не відволікати увагу від головного контенту та не ускладнювати користувацький досвід.

Функціональність та фільтрація:

Функціональність веб-сайту відіграє ключову роль у забезпеченні зручного та ефективного користувацького досвіду. Для інтернет-магазинів

важливо мати широкий функціонал, який дозволяє користувачам не лише швидко знаходити потрібні товари, а й легко додавати їх до кошика, оформлювати замовлення та виконувати оплату. Крім того, функціональність включає у себе можливість перегляду детальної інформації про товар, відгуків покупців та рейтингу продуктів, що сприяє прийняттю обґрунтованих рішень при покупках.

Фільтрація, у свою чергу, є важливим елементом для зручності навігації користувачів по великому асортименту товарів. Вона дозволяє впорядкувати товари за різними критеріями, такими як ціна, категорія, розмір, колір тощо, що значно спрощує пошук необхідного продукту. Розумна система фільтрації дозволяє збільшити ефективність взаємодії користувача з сайтом, забезпечуючи високий рівень задоволення від покупок та збільшуючи конверсію.

Мобільний досвід:

Мобільний досвід є надзвичайно важливим для успіху будь-якого веб-сайту, зокрема для інтернет-магазинів. У сучасному світі, коли користувачі все частіше використовують мобільні пристрої для пошуку та покупок товарів, необхідно мати адаптивний та оптимізований дизайн для мобільних пристроїв. Це означає, що інтерфейс сайту повинен бути легко читабельним на малих екранах, елементи керування (наприклад, кнопки та поля введення) повинні бути зручними для використання пальцями, а швидкість завантаження сторінок повинна бути високою для запобігання втраті уваги користувача.

Особлива увага також повинна приділятися мобільній функціональності та навігації. Користувачам має бути легко знаходити необхідні товари, використовуючи фільтри та пошукові опції, і проводити покупки без зайвих труднощів. Крім того, додаткові можливості, такі як сповіщення про акції та зручний процес оформлення замовлення, підвищують зручність та привабливість мобільного досвіду. Інвестування в мобільну оптимізацію та вдосконалення користувацького досвіду на мобільних пристроях може значно підвищити конверсію та задоволення ваших клієнтів.

Пошуковик:

Пошуковик у інтернет-магазині є одним із найважливіших інструментів для користувачів у пошуках необхідних товарів. Його функціональність та ефективність може визначити ступінь задоволення користувача та конверсію на сайті. Важливо мати потужну систему пошуку, яка здатна швидко та точно знаходити товари згідно з запитом користувача. Використання автодоповнення, фільтрація результатів, можливість сортування та додаткові опції пошуку (наприклад, пошук за брендом, розміром, ціною) допоможуть забезпечити зручність та ефективність пошукової системи.

Також важливо проводити аналіз даних пошукового запиту, щоб зрозуміти попит на певні товари та покращити пропозицію асортименту в інтернет-магазині. Регулярне оновлення та удосконалення алгоритмів пошуку допоможе забезпечити високу якість пошукового досвіду для користувачів і позитивно позначиться на їхній взаємодії з сайтом та на конверсії.

Сторінка "Not Found":

Сторінка "Not Found" або "404 Error" є важливою складовою інтерфейсу будь-якого веб-сайту. Ця сторінка відображається користувачам, коли вони спробують отримати доступ до неіснуючої сторінки або ресурсу. Оскільки це може виникати з різних причин, таких як помилковий введений URL, старі посилання, зміни в структурі сайту тощо, важливо забезпечити на сторінці "Not Found" інформативну та корисну інформацію для користувача.

Добра сторінка "Not Found" повинна містити пояснення про те, що трапилася помилка, можливі причини її виникнення, а також допомогти користувачеві знайти шлях назад на сайт або на головну сторінку. Також доцільно додати контактну інформацію або посилання на сторінку з контактами, щоб користувач міг зв'язатися з адміністратором сайту в разі потреби. Гарно оформлена та інформативна сторінка "Not Found" допоможе зберегти користувача на сайті та позитивно вплине на його враження від взаємодії з веб-ресурсом.

Корзина:

Корзина в інтернет-магазині є ключовою складовою для покупок користувачів. Це місце, де вони можуть збирати товари, які їх цікавлять, переглядати їхню кількість та вартість, а також готуватися до оформлення замовлення. Оптимальна корзина повинна бути простою та зрозумілою для використання, з можливістю додавання, видалення та зміни кількості товарів одними клацаннями. Також важливо, щоб корзина відображала повну вартість замовлення, враховуючи доставку та податки, щоб користувачі могли оцінити свої витрати перед оформленням покупки.

Поряд з цим, корзина повинна мати явні кнопки "Оформити замовлення" або "Перейти до оплати", що спрощує процес завершення покупки. Також важливо включити можливість збереження товарів у корзині для подальшого відновлення сесії, щоб користувачі могли повернутися до неї пізніше без втрати даних. Наприклад, якщо користувач залишив корзину без оформлення замовлення, повідомлення або нападка, яка нагадує про незавершене замовлення, може підтримати його і підвищити конверсію. Такий підхід допоможе створити зручний та приємний досвід користувачам при покупках в вашому інтернет-магазині.

Сторінка з акаунтом та авторизацією:

Сторінка з акаунтом та авторизацією є важливим елементом будь-якого веб-сайту, особливо для інтернет-магазинів. Це місце, де користувачі можуть увійти до свого особистого облікового запису, переглянути свої попередні замовлення, змінити особисту інформацію, включаючи адресу доставки та способи оплати, а також додати товари до списку бажань. Оптимальна сторінка з акаунтом повинна бути простою та зрозумілою для використання, з можливістю швидкої авторизації через соціальні мережі або електронну пошту.

Також важливо, щоб сторінка авторизації мала високий рівень безпеки, забезпечуючи захист особистої інформації користувачів та їхніх фінансових даних. Додаткові можливості, такі як можливість відновлення паролю, подвійна автентифікація та опції збереження сесії авторизації, сприяють зручності та безпеці використання сторінки з акаунтом.

Отже, подальше вдосконалення інтерфейсу та користувацького досвіду відкриває шлях до покращення конверсії та забезпечення задоволення клієнтів. Реалізація вищезгаданих підходів дозволить вашому інтернет-магазину зайняти відповідне місце на ринку та залучити нових клієнтів.

РОЗДІЛ 3

РОЗРОБКА ВЕБ-САЙТУ ІНТЕРНЕТ-МАГАЗИНУ

3.1 Створення бази даних товарів та клієнтів

При розробці бази даних товарів та клієнтів було використано MongoDB та Express.js.

MongoDB - це NoSQL база даних, яка дозволяє зберігати дані у вигляді документів у форматі JSON. Для інтернет-магазину ми можемо створити колекції для товарів та клієнтів. Кожен документ у колекції "Товари" може містити різні поля, такі як ідентифікатор товару, назва, опис, ціна, категорія тощо. Для забезпечення швидкості доступу та ефективного використання ресурсів можна створити індекси для поля, яке часто використовується для пошуку товарів.

Express.js використовується для створення серверної частини додатку. Ми створюємо маршрути (routes) для обробки HTTP-запитів, які приходять від клієнтської частини. Наприклад, маршрути GET можуть бути налаштовані для отримання списку товарів або детальної інформації про певний товар. Маршрути POST можуть обробляти дані, надіслані користувачем при створенні нового товару або замовлення. Крім того, Express дозволяє обробляти помилки та створювати користувацькі обробники для кожного типу помилок, що дозволяє забезпечити відповідне повідомлення користувачам у разі виникнення проблем.

На рисунку 3.1. зображена діаграма варіантів використання бази даних.

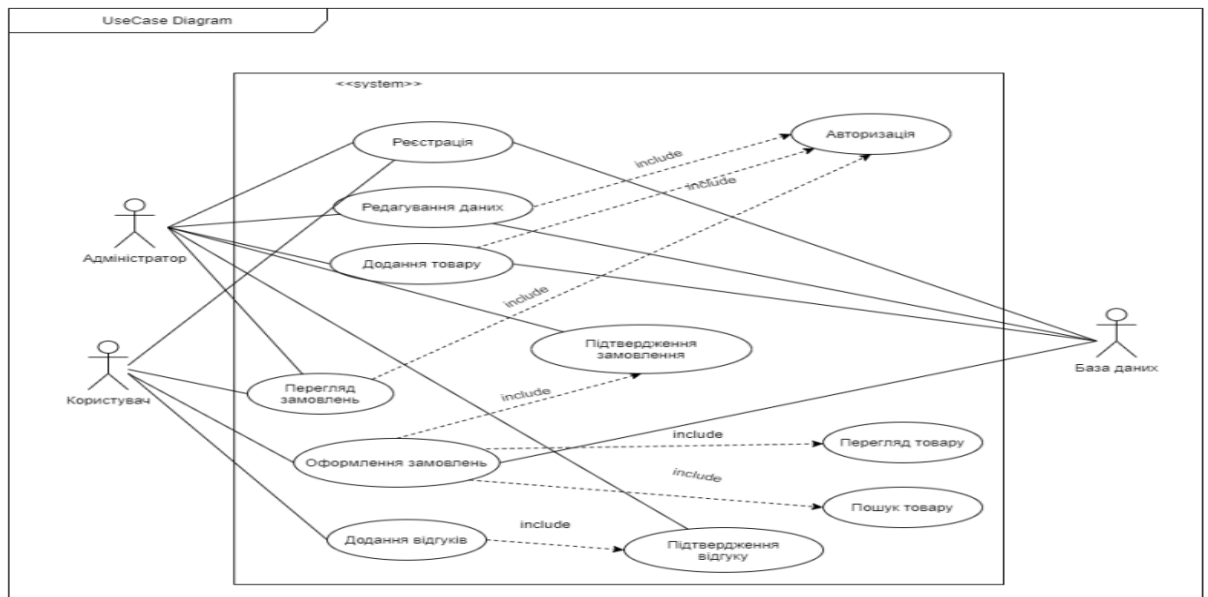


Рисунок 3.1 - Варіант використання бази даних

Для початку проведемо імпорт модулів. MongoClient це клас з модуля mongodb, який надає можливість встановлювати з'єднання з базою даних MongoDB та виконувати операції з базою даних. Mongoose це бібліотека для моделювання даних MongoDB для Node.js, яка дозволяє зручно взаємодіяти з MongoDB через об'єктно-орієнтований підхід.

```

1  import { MongoClient } from 'mongodb';
2  import mongoose from 'mongoose';
  
```

Рисунок 3.2 - Фрагмент програмного коду «Імпорт модулів»

Налаштуємо підключення до MongoDB через Atlas або інший URI, а також опцій підключення:

ConnectionString - це URI, який містить інформацію про адресу сервера бази даних та інші параметри підключення. Options - це об'єкт, який містить налаштування підключення до MongoDB, такі як використання нового парсера URL (useNewUrlParser) та єднання з єдиною топологією (useUnifiedTopology), що є стандартними опціями для підключення.

```
const connectionString = process.env.ATLAS_URI || '';
const options = {
  useNewUrlParser: true,
  useUnifiedTopology: true,
};
```

Рисунок 3.3 - Налаштування та підключення до MongoDB

Оголошуємо змінні для зберігання даних про підключення. Client: Змінна, яка зберігає екземпляр клієнта MongoClient для використання під час підключення до MongoDB. Conn: Змінна, яка зберігає посилання на базу даних MongoDB. DbProducts: Змінна, яка також зберігає посилання на базу даних "Products".

```
let client
let conn
let dbProducts
```

Рисунок 3.4 - Оголошення змінних для клієнта, з'єднання та бази даних

Прописуємо функцію для підключення до бази даних MongoDB за допомогою mongoose. Ця функція використовує mongoose.connect() для встановлення з'єднання з MongoDB за допомогою рядка підключення та опцій, визначених раніше.

```
async function connectToMongoDB() {
  try {
    await mongoose.connect(connectionString, options);
    console.log('MongoDB is connected');
  } catch (err) {
    console.error(err);
  }
}
```

Рисунок 3.5 - Функція для підключення до бази даних MongoDB за допомогою mongoose

Після цього була написана Функція для підключення до MongoDB через нативний драйвер MongoClient. Ця функція створює новий екземпляр MongoClient та встановлює з'єднання з базою даних, використовуючи URI. Після

успішного з'єднання отримується об'єкт бази даних `conn`, який також присвоюється змінній `dbProducts`.

```
async function connectToMongoClient() {
  try {
    client = new MongoClient(connectionString);
    await client.connect();
    conn = client.db("Products");
    dbProducts = conn;
  } catch (e) {
    console.error(e);
  }
}
```

Рисунок 3.6 - Функція для підключення до MongoDB через нативний драйвер MongoClient

Реалізуємо Функцію для ініціалізації підключення. Ця функція асинхронно викликає обидві попередні функції, щоб ініціалізувати підключення до бази даних.

```
async function initialize() {
  await Promise.all([connectToMongoDB(), connectToMongoClient()]);
}

initialize();
```

Рисунок 3.7 - Функція для ініціалізації підключення до бази даних

Експорт змінних `conn` та `dbProducts` для доступу до з'єднання та бази даних з інших модулів. Цей експорт дозволяє іншим модулям отримувати доступ до з'єднання та бази даних для взаємодії з даними. Експорт з'єднання `mongoose` для використання моделей та схем даних. Цей експорт дозволяє іншим модулям використовувати з'єднання `mongoose` для визначення моделей даних та роботи з ними.

```
export { conn, dbProducts };
export default mongoose.connection;
```

Рисунок 3.8 - Експорт з'єднання `mongoose` та змінних з базою даних інших модулів.

3.2 Розробка корзини та системи покупок

При створенні інтернет-магазину ключовою частиною є корзина та система покупок. Вони забезпечують зручний та ефективний спосіб додавання товарів до замовлення, його оплати та оформлення. Розробка цих компонентів вимагає детального аналізу та проектування для забезпечення відповідності потребам користувачів та бізнесу.

Основні аспекти розробки:

1. Функціональність корзини:

Розробка корзини полягає у створенні інтерфейсу, який дозволить користувачам додавати товари до списку покупок, змінювати їх кількість та видаляти. Корзина також повинна відображати загальну суму замовлення та надавати можливість продовження оформлення замовлення.

2. Управління замовленнями:

Для ефективного управління замовленнями потрібна система, яка забезпечить обробку покупок, включаючи підтвердження та відстеження стану замовлення. Розробка системи покупок включає в себе обробку оплати, виписування рахунків та сповіщення користувачів про статус їхніх замовлень.

3. Безпека та захист даних:

Оскільки користувачі магазину надають особисту та фінансову інформацію під час оформлення покупки, важливо забезпечити високий рівень безпеки та захисту цих даних. Це може бути досягнуто за допомогою шифрування даних, використання безпечного протоколу передачі даних (наприклад, HTTPS), аутентифікації користувачів та застосування інших методів захисту.

4. Інтеграція з платіжними системами:

Розробка системи покупок також включає інтеграцію з платіжними системами, такими як Stripe, PayPal або інші. Це передбачає

налаштування зв'язку з платіжними шлюзами, обробку платежів та отримання підтверджень про успішну оплату.

Важливі аспекти реалізації:

- Інтерактивний інтерфейс: Користувацький інтерфейс корзини та системи покупок повинен бути зручним та інтуїтивно зрозумілим, з можливістю миттєвої відповіді на дії користувача.
- Масштабованість: Система повинна бути готовою до масштабування, щоб забезпечити ефективну роботу при зростанні обсягів продажів та попиту.
- Тестування та валідація: Важливо провести ретельне тестування системи, щоб переконатися в її надійності та відповідності вимогам.

Перейдемо до реалізації системи корзини та покупок. Спочатку імпортуємо необхідні бібліотеки для створення. UseEffect та useState це хуки React, що дозволяють використовувати ефекти та стан компонента. UseDispatch та useSelector це хуки Redux, що дозволяють використовувати функції диспетчера та селектори для роботи зі Store. RootState: тип даних, що представляє кореневий стан Redux Store. CartItem, clearCart, removeFromCart, updateCart: типи та дії Redux, що використовуються для роботи з кошиком покупок. Styles: об'єкт стилів з CSS модуля для компонента Cart. SelectIsAuth: селектор Redux, що дозволяє перевірити, чи є користувач автентифікованим.

```
import { useEffect, useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { RootState } from '../app/store';
import { CartItem, clearCart, removeFromCart, updateCart } from '../redux/slices/cartSlice';
import styles from './Cart.module.css';
import { selectIsAuth } from '../redux/slices/authSlice';
import { useRouteChange } from '../hooks/useRouteChange';
```

Рисунок 3.9 - Імпорт необхідних бібліотек та залежностей

Визначаємо користувацьку логіку useCartLogic, яка приймає параметри isVisible (чи видимий кошик) та setIsVisible (функція для зміни видимості). Використовуємо хуки та селектори Redux та функції маршрутизації.

routeChange: користувацький хук для зміни маршруту. ShowSuccessNotification: стан для відображення повідомлення про успішну оплату. IsAuth: змінна, що показує, чи автентифікований користувач. Dispatch: функція диспетчера Redux для виклику дій (actions). CartItems: масив товарів у кошику, отриманий зі Store.

```
const useCartLogic = (isVisible: boolean, setIsVisible: React.Dispatch<React.SetStateAction<boolean>>) => {
  const routeChange = useRouteChange()
  const [showSuccessNotification, setShowSuccessNotification] = useState(false);

  const isAuth = useSelector(selectIsAuth);
  const dispatch = useDispatch();
  const cartItems: CartItem[] = useSelector((state: RootState) => state.cartItems);
```

Рисунок 3.10 - Використання хуків та селекторів Redux та функції маршрутизації

Прописуємо функцію обробки подій. handleClick: функція для зміни видимості кошика. handleCheckoutCart: функція для оформлення замовлення у кошику, перевіряє, чи користувач автентифікований. handleRemoveClick: функція для видалення товару з кошика. handleQuantityChange: функція для зміни кількості товару у кошику.

```
const handleClick = () => {
  setIsVisible(!isVisible);
};

const handleCheckoutCart = () => {
  if (isAuth === true) {
    dispatch(clearCart())
    setShowSuccessNotification(!showSuccessNotification);
  }
  else {
    setIsVisible(false)
    return routeChange('/login')
  }
}

const handleRemoveClick = (item: string) => {
  dispatch(removeFromCart(item));
};

const handleQuantityChange = (id: string, quantity: number) => {
  dispatch(updateCart({ id, quantity }));
};
```

Рисунок 3.11 - Визначення функцій обробки подій

CalculateTotal: функція для підрахунку загальної вартості товарів у кошику. Обчислення загальної вартості товарів у кошику зображено на рис. 3.11.

```
const calculateTotal = () => {
  let total = 0;
  cartItems.forEach((item) => {
    const itemQuantity = item.quantity || 1;
    total += parseFloat(item.price as any) * itemQuantity;
  });
  return total.toFixed(2);
};
```

Рисунок 3.12 - Обчислення загальної вартості товарів у кошику

Створюємо ефекти для управління стилями та поведінкою DOM, які встановлюють поведінку прокрутки сторінки та елемента кошика залежно від видимості кошика.

```
useEffect(() => {
  document.body.style.overflow = isVisible ? 'hidden' : 'auto';
  const cartElement = document.querySelector(`.${styles.cart}`) as HTMLElement | null;
  if (cartElement) {
    cartElement.style.overflow = isVisible ? 'auto' : 'hidden';
  }
  return () => {
    document.body.style.overflow = 'auto';
  };
}, [isVisible]);
```

Рисунок 3.13 - Ефекти для управління стилями та поведінкою DOM

Реалізуємо повернення об'єкта з функціями та станами для використання в компоненті. Цей об'єкт містить функції та стани, які можна використовувати в компоненті Cart для управління кошиком.

```

return {
  handleClick,
  handleRemoveClick,
  handleQuantityChange,
  calculateTotal,
  handleCheckoutCart,
  showSuccessNotification,
  setShowSuccessNotification
};
};

```

Рисунок 3.14 - Повернення об'єкта з функціями та станами для використання в компоненті

Експортуємо функції useCartLogic. Цей експорт дозволяє імпортувати useCartLogic у інших компонентах для використання логіки кошика.

```
export default useCartLogic;
```

Рисунок 3.15 - Експорт функції useCartLogic:

Кінцевий результат, вигляд корзини покупок зображений на рисунку 3.14

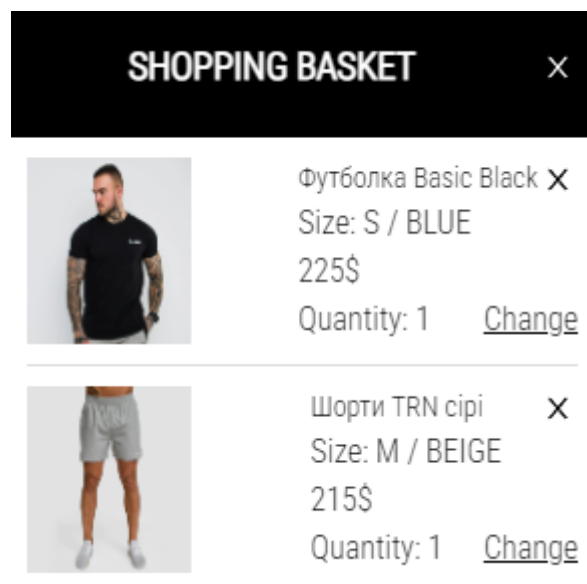


Рисунок 3.16 – корзина покупок

3.3 Розробка модуля оплати та доставки

Розробка модуля оплати та доставки для сайту інтернет-магазину є критично важливою складовою, яка безпосередньо впливає на користувацький досвід та ефективність операцій. Цей процес включає кілька ключових етапів, які необхідно виконати, щоб забезпечити безперебійне та безпечне здійснення транзакцій і доставку товарів до клієнтів.

Етап 1: Підготовка

На першому етапі необхідно визначитися з технологічними рішеннями, які будуть використані для розробки. Для серверної частини часто використовують Node.js з Express або Python з Flask/Django, оскільки ці фреймворки забезпечують гнучкість і швидкість розробки. Для клієнтської частини стандартними виборами є HTML, CSS та JavaScript. Також важливо вибрати базу даних, яка буде використовуватися для зберігання інформації про замовлення та користувачів, наприклад, MySQL, PostgreSQL або MongoDB.

Етап 2: Вибір та інтеграція платіжної системи

Вибір платіжної системи є одним із найважливіших рішень. Популярні варіанти включають Stripe, PayPal, LiqPay та WayForPay. Кожна з цих систем має свої переваги і недоліки, тому важливо вивчити їхню документацію та зрозуміти процес інтеграції. Наприклад, Stripe пропонує потужний API, який дозволяє легко налаштувати процес оплати і підтримує різні валюти та методи оплати.

Етап 3: Розробка серверної частини

Після вибору платіжної системи наступним кроком є розробка серверної частини. Почніть з ініціалізації проекту і встановлення необхідних залежностей. Наприклад, для Node.js встановіть Express, Stripe та body-parser. Створіть базовий сервер, який буде обробляти запити на створення платіжних намірів. Це включає написання маршруту, який буде викликатися при ініціюванні платежу, і використання API платіжної системи для створення платіжного наміру. Наприклад, для Stripe це виглядатиме так: створення маршруту для POST запиту

/create-payment-intent, який приймає суму платежу, створює платіжний намір і повертає client_secret, необхідний для завершення оплати на стороні клієнта.

Етап 4: Розробка клієнтської частини

Клієнтська частина включає створення HTML форми для введення платіжної інформації. Використовуйте офіційні бібліотеки платіжної системи для інтеграції платіжного інтерфейсу. Наприклад, для Stripe використовуйте Stripe.js для створення елемента картки і обробки платіжної інформації. Додайте JavaScript код, який буде відправляти дані форми на сервер для створення платіжного наміру і обробляти результат. Важливо забезпечити, щоб інтерфейс був інтуїтивно зрозумілим і зручним для користувачів.

Етап 5: Інтеграція служби доставки

Після налаштування платежів, необхідно інтегрувати служби доставки. Виберіть служби доставки, які найкраще підходять для вашого бізнесу, наприклад, Нова Пошта, Укрпошта, DHL або FedEx. Вивчіть їх API для отримання інформації про тарифи, строки доставки та відстеження посилок. Створіть маршрути на сервері для обробки запитів на створення та відстеження відправлень. Наприклад, для Нової Пошти використовуйте їх API для створення відправлень, отримання номерів накладних і статусів доставки.

Етап 6: Тестування

Тестування є критично важливим етапом розробки. Переконайтеся, що всі сценарії оплати і доставки працюють коректно. Використовуйте тестові картки та облікові записи, надані платіжними системами, для перевірки різних варіантів оплати. Також тестуйте інтеграцію з службами доставки, перевіряючи коректність створення відправлень і отримання статусів. Залучайте тестувальників для перевірки користувацького інтерфейсу і виявлення можливих проблем.

Етап 7: Впровадження і підтримка

Після успішного тестування модулі оплати та доставки можуть бути впроваджені на вашому сайті. Забезпечте, щоб ваш сервер підтримував HTTPS для безпеки даних користувачів. Після запуску надайте підтримку користувачам

і оперативно вирішуйте виникаючі проблеми. Регулярно оновлюйте модулі відповідно до змін в API платіжних систем і служб доставки.

Таким чином, розробка модуля оплати та доставки включає підготовку, вибір і інтеграцію платіжних систем та служб доставки, розробку серверної і клієнтської частини, тестування і впровадження. Кожен з цих етапів вимагає ретельного планування і виконання для забезпечення безперебійної роботи інтернет-магазину. На жаль, інтегрувати модулі оплати та доставки не вдалось.

3.4 Розробка адміністративного інтерфейсу для управління магазином

Під час розробки інтернет-магазину, я вивчав та впроваджував функціонал адміністративного інтерфейсу, який дозволяє легко та ефективно керувати товарами та замовленнями без необхідності втручання у базу даних вручну.

Основні функції адміністративного інтерфейсу:

1. Додавання та редагування товарів: Створено інтерфейс, що дозволяє адміністраторам легко додавати нові товари до каталогу магазину. Це включає заповнення необхідних полів, таких як назва товару, опис, ціна, зображення тощо, а також можливість редагування існуючих товарів.
2. Управління замовленнями: Реалізовано інтерфейс для перегляду та обробки замовлень. Адміністратор може переглядати деталі замовлення, включаючи інформацію про клієнта, обрані товари та загальну суму, а також відзначати їх як виконані чи скасовані.
3. Керування категоріями товарів: Для зручного організації каталогу товарів, створено можливість додавання, редагування та видалення категорій. Це дозволяє структурувати товари за різними критеріями та полегшує навігацію для користувачів.

Важливі аспекти реалізації:

- Зручний та інтуїтивно зрозумілий інтерфейс: Адміністративний інтерфейс розроблено з урахуванням потреб користувачів та забезпечує зручність та ефективність використання.
- Безпека та захист даних: Застосовано необхідні заходи безпеки для захисту важливих даних користувачів та забезпечення конфіденційності.
- Підтримка масштабування: Розроблений інтерфейс готовий до масштабування та розширення, щоб відповідати зростаючим потребам магазину та його адміністраторів.

Цей адміністративний інтерфейс дозволяє забезпечити ефективне керування магазином та забезпечити його успішну роботу на ринку електронної торгівлі.

Перейдемо до реалізації. Для початку Імпортуємо бібліотеки та компоненти. `axios`: бібліотека для здійснення HTTP-запитів. Використовується для взаємодії з сервером для надсилання та отримання даних. `ChangeEvent`, `FormEvent`, `useState`: це хуки React, які використовуються для роботи з формою. `ChangeEvent` використовується для обробки змін у введених даних, `FormEvent` - для обробки події відправки форми, `useState` - для зберігання стану компонента. `PagesInformer`: це компонент, який відображає інформацію на сторінці, наприклад, номер поточної сторінки або загальну кількість сторінок.

```
import axios from "axios";  
import { ChangeEvent, FormEvent, useState } from "react"  
import PagesInformer from "../../components/pages-informer";
```

Рисунок 3.17 - Імпорт бібліотек та компонентів

Реалізуємо Інтерфейс `FormData`. Інтерфейс визначає структуру даних, які будуть введені у формі. В цьому випадку, це дані про одяг: назва, колекція, категорія, стать, колір, розмір, URL зображення та ціна.

```
interface FormData {
  name: string;
  collection: string;
  category: string;
  gender: string;
  color: string;
  size: string;
  url: string;
  price: string;
}
```

Рисунок 3.18 - Реалізація інтерфейсу FormData

Прописуємо Компонент FormClothesCreate. Цей компонент створює форму для введення даних про одяг. useState використовується для зберігання стану форми, а саме введених даних.

```
export const FormClothesCreate = () => {
  const [formData, setFormData] = useState<FormData>({
    name: '',
    collection: '',
    category: '',
    gender: '',
    color: '',
    size: '',
    url: '',
    price: '',
  })
}
```

Рисунок 3.19 - Компонент FormClothesCreate

Створюємо обробники подій handleSubmit та handleChange. handleSubmit: обробник події відправки форми. Він перешкоджає стандартній поведінці форми, відправляє POST-запит на сервер із введеними даними про одяг. handleChange: обробник події зміни значення введеного поля форми. Він оновлює стан форми при кожній зміні.

```
const handleSubmit = async (event: FormEvent<HTMLFormElement>) => {
  event.preventDefault();
  try {
    console.log(formData);
    await axios.post('http://localhost:5172/api/Clothes', formData)
  } catch (error) {
    console.error('Error submitting data', error)
  }
}

const handleChange = (event: ChangeEvent<HTMLInputElement>) => {
  const { name, value } = event.target;
  setFormData((prevFormData) => ({
    ...prevFormData,
    [name]: value
  })))
}
```

Рисунок 3.20 - обробники подій handleSubmit та handleChange

Повернення JSX-елементів. JSX-код, який повертає компонент PagesInformer, форму з введенням даних про одяг та кнопку "Submit" для відправки форми. onSubmit={handleSubmit}: вказує, що обробник handleSubmit викликається при відправці форми. Цей код дозволяє користувачам вводити дані про одяг у форму, яка відправляє ці дані на сервер для зберігання у базі даних.

```
return (
  <>
    <PagesInformer />
    <form onSubmit={handleSubmit}>
      <input type="text" name="name" value={formData.name} onChange={handleChange} placeholder="Name" />
      <input type="text" name="collection" value={formData.collection} onChange={handleChange} placeholder="Collection" />
      <input type="text" name="category" value={formData.category} onChange={handleChange} placeholder="Category" />
      <input type="text" name="gender" value={formData.gender} onChange={handleChange} placeholder="Gender" />
      <input type="text" name="color" value={formData.color} onChange={handleChange} placeholder="Color" />
      <input type="text" name="size" value={formData.size} onChange={handleChange} placeholder="Size" />
      <input type="text" name="url" value={formData.url} onChange={handleChange} placeholder="URL" />
      <input type="text" name="price" value={formData.price} onChange={handleChange} placeholder="Price" />
      <button type="submit">Submit</button>
    </form>
  </>
)
```

Рисунок 3.21 - Повернення JSX-елементів

РОЗДІЛ 4

ТЕСТУВАННЯ ТА ВІДЛАГОДЖЕННЯ

4.1 Верифікація та валідація веб-сайту

Даний етап потрібен для перевірки коректності введених даних користувачем. Під час реєстрації нового користувача та оформлення замовлення на сайті спортивного інтернет-магазину, перевірка коректності введених даних є важливою складовою для забезпечення точності та відповідності стандартам. Наприклад, при реєстрації нового користувача система перевіряє введену електронну пошту на відповідність стандартному формату електронної пошти (наприклад, `example@example.com`) та переконується, що пароль містить достатньо символів та містить комбінацію цифр, літер верхнього та нижнього регістрів. Додатково, при введенні основних даних для товарів, таких як розмір, колір та інші параметри, також проводиться перевірка на наявність обов'язкових полів та їхню відповідність стандартам (наприклад, числове значення розміру).

Також треба враховувати захист від потенційних атак і вразливостей. З метою запобігання потенційним атакам і вразливостям, система застосовує ряд заходів. Один з них - фільтрація та обробка введених даних. Наприклад, використання механізмів, що блокують спроби вводу шкідливого коду або SQL-ін'єкцій через веб-форми на сайті. Крім того, валідація даних здійснюється на обох рівнях - на клієнтському та серверному. Це забезпечує більш надійний контроль над даними перед їх відправкою на сервер і дозволяє уникнути передачі некоректних або шкідливих даних. Додатково, система використовує автоматизовані тести для валідації даних, що дозволяє виявляти та виправляти потенційні проблеми ще до випуску нових версій сайту в продакшн. Такий комплексний підхід до верифікації та валідації даних допомагає забезпечити надійність та безпеку введених даних, що є критичним для успішного функціонування спортивного інтернет-магазину.

Перейдемо до реалізації цього етапу. Цей код використовує бібліотеку `express-validator` для валідації даних, які надходять на сервер. В контексті теми створення бази даних товарів та клієнтів, цей код допомагає перевіряти правильність введених даних перед їх збереженням у базу даних. `validateClothes` валідує дані, які надходять для створення нового товару. Кожне правило валідації вказує на необхідність заповнення певних полів товару (`name`, `category`, `gender`, `color`, `size`, `url`, `price`). Порожні значення у цих полях вважаються неприпустимими (`notEmpty()`).

```
export const validateClothes = [
  body('name').notEmpty(),
  body('collection'),
  body('category').notEmpty(),
  body('gender').notEmpty(),
  body('color').notEmpty(),
  body('size').notEmpty(),
  body('url').notEmpty(),
  body('price').notEmpty(),
];
```

Рисунок 3.19 - Повернення JSX-елементів

`loginValidation` є масивом правил валідації для даних, які надходять для авторизації користувача. `body('email', 'Неверный формат почты').isEmail()`: це правило перевіряє, чи відповідає значення поля "email" формату електронної пошти. Параметр 'Неверный формат почты' вказує на текст помилки, який буде виведений у разі невалідного формату. `body('password', 'Пароль должен быть минимум 5 символов').isLength({ min: 5 })`: це правило перевіряє, чи має значення поля "password" мінімальну довжину не менше 5 символів.

```
export const loginValidation = [
  body('email', 'Неверный формат почты').isEmail(),
  body('password', 'Пароль должен быть минимум 5 символов').isLength({ min: 5 }),
];
```

Рисунок 3.19 - Повернення JSX-елементів

registerValidation є масивом правил валідації для даних, які надходять для реєстрації нового користувача. Подібно до loginValidation, цей блок також містить правила для "email" і "password", а також правило для "fullName", яке перевіряє мінімальну довжину повного імені користувача.

```
export const registerValidation = [  
  body('email', 'Неверный формат почты').isEmail(),  
  body('password', 'Пароль должен быть минимум 5 символов').isLength({ min: 5 }),  
  body('fullName', 'Укажите имя').isLength({ min: 3 }),  
];
```

Рисунок 3.20 - Повернення JSX-елементів

У контексті створення бази даних для товарів та клієнтів, ці правила валідації допомагають гарантувати, що дані, які надходять на сервер, відповідають певним критеріям. Це важливо для запобігання зберігання некоректних або неповних даних у базі даних та для забезпечення безпеки та надійності системи.

4.2 Тестування функціональності та безпеки

Ручне тестування функціональності.

Для детальної перевірки функціональності інтернет-магазину безпосередньо з користувачем можна застосовувати різноманітні методи, такі як пряме тестування та контрольні перевірки. Це дозволяє виявити та виправити помилки, забезпечуючи таким чином високу якість і зручність використання сайту.

1.Пряме тестування: Цей метод передбачає ручну перевірку окремих функціональних можливостей інтернет-магазину. Наприклад, важливо перевірити, чи правильно працює додавання товару до кошика, оформлення замовлення, реєстрація нового користувача, відновлення паролю та інші ключові функції. Це дозволяє виявити проблеми на ранніх етапах і швидко їх усунути, забезпечуючи таким чином безперебійне функціонування сайту.

2.Контрольна перевірка: Регулярне тестування основних функцій інтернет-магазину є критично важливим. Це включає додавання нових товарів до каталогу, оновлення інформації про товари, перевірку правильності обробки замовлень та доставок, а також взаємодію з клієнтами через зворотний зв'язок. Контрольна перевірка дозволяє своєчасно виявити можливі проблеми та оперативно їх вирішити, покращуючи загальний користувацький досвід і підвищуючи рівень задоволеності клієнтів.

Ручне тестування безпеки:

Безпека даних є однією з найважливіших аспектів для інтернет-магазину, тому ручні перевірки безпеки можуть допомогти виявити потенційні проблеми та загрози. Цей процес передбачає детальне тестування на вразливості та перевірку систем авторизації та аутентифікації, що дозволяє забезпечити високий рівень захисту даних користувачів.

1.Тестування на вразливості: Цей метод включає ручну перевірку можливостей вводу даних для виявлення потенційних вразливостей, таких як SQL-ін'єкції або XSS-атаки. Тестувальники можуть спробувати ввести шкідливий код у різні форми введення, щоб переконатися, що система належним чином захищена від таких атак. Це дозволяє вчасно виявити слабкі місця в безпеці і вжити необхідних заходів для їх усунення.

2.Тестування авторизації та аутентифікації: Важливо перевірити систему авторизації та аутентифікації, включаючи перевірку прав доступу до різних частин сайту та можливість виконання незаконних дій. Це включає перевірку механізмів входу та виходу з акаунту, відновлення паролю, управління правами користувачів, а також захист від несанкціонованого доступу. Такий підхід дозволяє переконатися, що лише авторизовані користувачі мають доступ до конфіденційної інформації та можуть виконувати дозволені дії.

Застосування ручного тестування функціональності та безпеки, хоч і не є повністю автоматизованим, дозволяє забезпечити базовий рівень перевірок для вашого інтернет-магазину. Це гарантує якість та надійність продукту перед його

випуском, що, в свою чергу, підвищує довіру користувачів і забезпечує успіх вашого бізнесу.

4.3 Виправлення помилок та відлагодження

Виправлення помилок та вдосконалення є важливими етапами у процесі розробки програмного забезпечення, включаючи інтернет-магазини. Цей процес охоплює не тільки усунення виявлених помилок, але й впровадження покращень для підвищення ефективності, безпеки та зручності використання продукту. Ця діяльність дозволяє підтримувати високу якість продукту, забезпечувати його безперебійну роботу та задовольняти вимоги користувачів.

Перший крок у виправленні помилок – це їхнє виявлення та ретельний аналіз. Важливо створити ефективний механізм для збору зворотного зв'язку від користувачів, які можуть стикатися з різними проблемами під час використання інтернет-магазину. Для цього можна використовувати систему підтримки клієнтів, де користувачі можуть надсилати звіти про помилки та інші труднощі. Крім того, корисним інструментом є аналіз системних журналів, які автоматично фіксують помилки та винятки, що виникають під час роботи програми. Ці журнали надають цінну інформацію для діагностики проблем та їхнього оперативного вирішення. Окрім того, регулярне систематичне тестування, що включає функціональне, інтеграційне та регресійне тестування, дозволяє виявити нові або раніше не помічені помилки, забезпечуючи таким чином високий рівень надійності та стабільності продукту.

Після аналізу помилок важливо їх пріоритизувати відповідно до їхнього впливу на функціональність, безпеку та досвід користувача. Критичні помилки, які призводять до втрати даних, порушення безпеки або зупинки роботи сайту, мають бути виправлені в першу чергу. Це може включати проблеми з обробкою платежів, неправильне зберігання даних користувачів або вразливості, що можуть призвести до зламу системи. Середні помилки, що впливають на функціональність або зручність використання, але не є критичними, також

потребують уваги. Наприклад, це можуть бути проблеми з інтерфейсом користувача, некоректне відображення товарів або труднощі з навігацією. Маловажливі помилки, які незначно впливають на роботу сайту або користувацький досвід, можуть бути виправлені останніми, але їх також не слід залишати без уваги, адже вони можуть накопичуватися та знижувати загальну якість продукту.

Після пріоритизації помилок команди розробників приступають до їх усунення. Цей процес включає написання нового коду для додавання необхідних функцій або виправлення існуючих, що дозволяє усунути виявлені проблеми. Крім того, можуть бути внесені зміни у вже існуючий код, щоб оптимізувати його роботу або виправити помилки. Це включає рефакторинг коду, що може покращити його читабельність та підтримуваність, а також збільшити продуктивність системи. Модифікація бази даних також може бути необхідною для вирішення проблем, пов'язаних з даними. Наприклад, це може включати зміну структури бази даних, оптимізацію запитів або корекцію некоректних даних.

Після внесення змін до коду необхідно провести ретельне тестування, щоб переконатися, що виправлення не спричинили нових проблем або погіршили інші аспекти програми. Ручне тестування дозволяє перевірити нові функції та виправлення у реальних умовах, щоб переконатися, що вони працюють як очікувалося. Автоматизоване тестування використовується для перевірки великої кількості сценаріїв використання, що допомагає швидше виявити потенційні проблеми та забезпечити високу якість програмного забезпечення.

Після виправлення помилок можна зосередитися на покращенні функціональності та безпеки продукту. Оптимізація роботи сайту включає підвищення продуктивності, зменшення часу завантаження сторінок та покращення загальної швидкодії. Вдосконалення інтерфейсу користувача сприяє кращому користувацькому досвіду, забезпечуючи зручність навігації та привабливий дизайн. Впровадження додаткових заходів безпеки, таких як двофакторна аутентифікація та посилення захисту від атак типу SQL-ін'єкцій та

XSS, підвищують рівень захисту даних користувачів та забезпечують їхню довіру до продукту.

Після успішного виправлення помилок та впровадження покращень нові версії продукту можуть бути випущені для користувачів. Важливо інформувати користувачів про внесені зміни та можливі відмінності у функціональності через електронні листи, повідомлення в блозі або на сторінці новин. Це допомагає користувачам швидше адаптуватися до нових функцій та змін. Організація плавного переходу на нові версії програмного забезпечення допомагає мінімізувати можливі незручності для користувачів, забезпечуючи безперервність роботи сайту та задоволеність клієнтів.

Виправлення помилок та вдосконалення – це безперервний процес у розробці програмного забезпечення, який дозволяє підтримувати високий рівень якості, безпеки та задоволення користувачів. Це сприяє успіху вашого інтернет-магазину, підвищуючи довіру користувачів та їхню лояльність, що, в свою чергу, сприяє зростанню вашого бізнесу та його конкурентоспроможності на ринку.

ВИСНОВКИ

В кваліфікаційній роботі виконано аналітичний огляд досліджень, що-до визначеної предметної області, проаналізовано сучасні рішення, сформульовані загальні підходи до розробки веб-сайту для продажі спортивних товарів.

Створено проєкт модернізації наявної корпоративної мережі. До нього входять: діаграма використань, діаграма Ганта та мережевий графік всіх наявних робіт за допомогою програми Microsoft Project.

В ході роботи були вивчені і проаналізовані програмні засоби для розробки сайтів. Так само розглянуті схожі сайти, виявлено їх переваги та недоліки, що лягло в основу створення програмного продукту.

В якості веб-сервера було обрано платформу Node.js. Важливими критеріями для відбору стали мультиплатформеність і безкоштовне розповсюдження. Крім того, Node.js в даний момент один з найбільш популярних веб-серверів в усьому світі.

Створення розподіленої архітектури додатку підвищило надійність і продуктивність програмного продукту. Чітко визначена роль кожної з підсистем на всіх рівнях продукту в значній мірі спростила завдання проектування і реалізації системи, дозволила забезпечити модифікацію продукту і його підтримку. Розроблений веб-додаток відповідає вимогам користувачів, володіє помірними потребами в апаратних ресурсах, заснований на переносних незалежних технологіях. Програмний продукт пройшов ряд тестових випробувань на предмет виявлення помилок в проектуванні і реалізації програми. Програмний продукт відповідає вимогам і має функціональність, надійністю, мобільністю та зручністю використання.

В результаті, все це дозволяє забезпечити більш тісний контакт з клієнтами через Інтернет, надає можливість пошуку нових контактів. За допомогою даного програмного продукту клієнти мають можливість отримати потрібну інформацію про товари і магазині, не виходячи з дому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Задерейко О. В. Комп'ютерні мережі: навчальний посібник / О. В. Задерейко, Н. І. Логінова, А. А. Толокнов. – Одеса, 2022. 249 с.
2. Захист комп'ютерних мереж URL – <https://magistrweb.wordpress.com/home-2/protection/>. (Дата звернення 23.01.2023)
3. Кіщак І.Т., Шебаніна О.В., Ключник А.В., Федосова А.О. Управління проектами: навчально-методичний посібник для самостійного вивчення дисципліни. – Миколаїв: 2018. 352 с.
4. Матеріали науково-практичної конференції викладачів, аспірантів та студентів Сумського НАУ (25-28 квітня 2023 р.). – Суми, 2023. 642 с.
5. Технології XXI сторіччя: Збірник тез за матеріалами 28-ої міжнародної науково-практичної конференції (23-25 листопада 2022 р.). Ч.2. – Суми: СНАУ, 2022. 193 с.
6. Управління IT-проектами. [Книга 1. Стандарти, моделі та методи управління проектами] – Львів: «Новий Світ- 2000». 2017. 550 с.
7. About WMI URL – <https://docs.microsoft.com/en-us/windows/win32/wmisdk/about-wmi>. (Дата звернення 22.04.2023)
8. Faisal P. Create A Windows Service In C# URL - <https://www.csharpcorner.com/article/create-windows-services-in-c-sharp/>.
9. GRIDINSOFT URL – <https://gridinsoft.ua/ddos>. (Дата звернення 27.01.2023)
10. ManagementEventWatcher Class URL – <https://docs.microsoft.com/enus/dotnet/api/system.management.managementeventwatcher?view=dotnet-plat-ext3.1>. (Дата звернення 17.03.2023)
11. Rozetka URL – https://hard.rozetka.com.ua/ua/kingston_sa400s37_240g/p14606570/?gclid=CjwKCAjwyqWkBhBMEiwAp2yUFvBOlR73KtSsV-

[Ос2mH6EeOLtG12zSKK8agwZVaW2knkanknJzW_9hoCIxQQAvD_BwE.](https://rozetka.com.ua/ua/12834243/p12834243/?gclid=CjwKCAjwyqWkBhBM_EiwAp2yUFqX4ceJO1MLoywF4yZRE9cMkDHhbFHqiMEqpFRLZK80gSDX8e4ACFBoCtSkQAvD_BwE)

(Дата звернення 09.03.2023)

12.	Rozetka	URL	—
-----	---------	-----	---

[https://rozetka.com.ua/ua/12834243/p12834243/?gclid=CjwKCAjwyqWkBhBM_EiwAp2yUFqX4ceJO1MLoywF4yZRE9cMkDHhbFHqiMEqpFRLZK80gSDX8e4ACFBoCtSkQAvD_BwE.](https://rozetka.com.ua/ua/12834243/p12834243/?gclid=CjwKCAjwyqWkBhBM_EiwAp2yUFqX4ceJO1MLoywF4yZRE9cMkDHhbFHqiMEqpFRLZK80gSDX8e4ACFBoCtSkQAvD_BwE) (Дата звернення 10.03.2023)

13.	Rozetka	URL	—
-----	---------	-----	---

[https://rozetka.com.ua/ua/tp_link_tl_sg1024d/p203087/?gclid=CjwKCAjwyqWkBhBMEiwAp2yUFnn2rVZ32js4tanXsT_XRH-K73q9BnH2uJT6IQuKRSX9KrzLGFUMHxoCDBsQAvD_BwE.](https://rozetka.com.ua/ua/tp_link_tl_sg1024d/p203087/?gclid=CjwKCAjwyqWkBhBMEiwAp2yUFnn2rVZ32js4tanXsT_XRH-K73q9BnH2uJT6IQuKRSX9KrzLGFUMHxoCDBsQAvD_BwE) (Дата

звернення 11.03.2023)

14.	Rozetka	URL	—
-----	---------	-----	---

[https://rozetka.com.ua/ua/364980675/p364980675/?gclid=CjwKCAjwyqWkBhBMEiwAp2yUFoOoaiaX_6-efvfeG5ke66ZBq_H8nPQob3cVRE8MWYAKj4ohtjB3_RoCWnUQAvD_BwE.](https://rozetka.com.ua/ua/364980675/p364980675/?gclid=CjwKCAjwyqWkBhBMEiwAp2yUFoOoaiaX_6-efvfeG5ke66ZBq_H8nPQob3cVRE8MWYAKj4ohtjB3_RoCWnUQAvD_BwE)

(Дата звернення 12.03.2023)

15.	WMI	for	Detection	and	Response	URL	-
-----	-----	-----	-----------	-----	----------	-----	---

[https://www.hSDL.org/?abstract&did=795007.](https://www.hSDL.org/?abstract&did=795007)

ДОДАТКИ

Додаток А

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку інформаційної системи
«Програмний-додаток організації “ Глухівський агротехнічний фаховий
коледж Сумського НАУ ”»

ПОГОДЖЕНО:

доцент кафедри кібернетики та
інформатики

Здобувач вищої освіти групи ІСТ _____

_____ (ПІБ)

Суми 2024

1. Призначення й мета створення web-додатку

1.1 Призначення web-додатку

Web-додаток призначений для підвищення ефективності ведення погосподарського обліку в територіальних громадах.

1.2 Мета створення web-додатку

Метою створення веб-сайту служить залучення нових клієнтів, збільшення продажів, надання інформації та ресурсів, збору даних і взаємодії з користувачами.

1.3 Цільова аудиторія

Цільовою аудиторією веб-сайту для продажу спортивних товарів є спортивні ентузіасти, активні люди та спортивні команди, які шукають якісне спортивне спорядження та екіпіровку для занять різними видами спорту. Також цей сайт може бути цікавим для тренерів, фітнес клубів та організацій, що шукають обладнання для тренувань та змагань.

2 Вимоги до web-додатку

2.1 Вимоги до web-додатку в цілому

2.1.1 Вимоги до структури й функціонування web-додатку

Web-додаток повинен бути реалізований у вигляді сайту, доступного в мережі Інтернет. Сайт повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

2.1.2 Вимоги до персоналу

Вимоги до персоналу для веб-додатку включають в себе знання технічних навичок в галузі програмування, дизайну та тестування, а також вміння ефективно спілкуватися з клієнтами та колегами для ефективного управління проектом. Крім того, важливо мати аналітичні здібності для аналізу вимог користувачів та розробки стратегій покращення функціоналу додатку.

2.1.3 Вимоги до збереженні web-додатку

Уся інформація надана у web-додатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних MongoDB.

2.1.4 Вимоги до розмежування доступу

Розроблюваний web-додаток не має бути приватним.

Відповідно до прав доступу у web-додаток, користувачів можна поділити на робітників та адміністратора.

Робочі аккаунти можуть переглядати усі сторінки web-додатку, після авторизації за логіном та паролем.

Адміністратор може редагувати зовнішній вигляд та наповнення web-додатку.

2.2 Структура web-додатку

2.2.1 Навігація

Відповідно до бажаного дизайну web-додатку, для навігації, у шапці буде створена система контент меню. Меню необхідне для швидкого переміщення користувача по усім доступним сторінкам. Меню буде відображатися на всіх сторінках, щоб відвідувач міг в будь-який момент часу перейти на будь-яку сторінку web-додатку.

2.2.3 Наповнення web-додатку (контент)

Сторінки всіх розділів сайту мають бути автоматично створені на основі даних з бази даних на сервері. Редагування вмісту розділів повинно бути можливим через адміністраторський web-інтерфейс, який дозволяє редагувати інформаційний вміст сторінок сайту без необхідності програмування. Наповнення сторінок інформацією повинне бути здійснюватися за допомогою шаблонів сайту.

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Система керування контентом (адміністративна частина) повинна надавати можливість додавання, редагування й видалення вмісту сторінок.

Також повинна бути передбачена можливість додавання інформації без відображення на сайті для користувачів.

2.3.2 Функціональні вимоги

Функціональні вимоги до веб-додатку можуть включати реєстрацію та авторизацію користувачів з можливістю відновлення паролю, відображення каталогу товарів з можливістю фільтрації та сортування, систему додавання товарів до кошика та оформлення замовлення з різними методами оплати, включаючи електронні платіжні системи та банківські перекази. Також важливо передбачити функціонал для адміністраторів, який дозволить керувати товарами, замовленнями, користувачами, а також ведення статистики продажів та аналізу даних для прийняття управлінських рішень.

2.3.3 Системні вимоги

Системні вимоги до веб-сайту з продажу спортивних товарів включають наявність веб-сервера з можливістю обробки HTTP-запитів. Також необхідно мати базу даних, яка забезпечує зберігання інформації про товари, користувачів та замовлення, а також SSL-сертифікат для захисту особистої інформації користувачів та забезпечення безпеки платежів. Крім того, важливо мати резервне копіювання даних та систему моніторингу для виявлення і усунення можливих неполадок у роботі сайту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація web-додатку відбувається з використанням:

- MongoDB

- Express.js
- React.js
- Node.js

2.4.2 Вимоги до лінгвістичного забезпечення

Web-додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Веб-браузер: Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

3 Склад і зміст робіт зі створення web-додатку

Докладний опис етапів роботи зі створення web-додатку наведено в таблиці А.1.

Таблиця А.1 – Етапи створення web-додатку

№	Склад і зміст робіт	Строк розробки
1	Визначення стратегічних цілей	1 день
2	Формулювання SMART-цілей	1 день
3	Вивчення ринку та конкурентів	3 дні
5	Аналіз тенденцій	2 дні
6	Вивчення цілей проекту	1 день
7	Вибір технологічного стеку	2 дні
8	Front-end розробка	8 днів

10	Back-end розробка	9 днів
11	Інтеграція з CMS	3 дні
13	Розробка інфраструктури	6 днів
14	Веб-дизайн та UX/UI	6 днів
15	Розробка макетів	8 днів
16	Затвердження дизайну	2 дні
17	Тестування вдосконалення	5 днів
18	Комплексне тестування	2 дні
19	Виправлення помилок	3 дні
20	Розгортання на сервері	2 дні
21	Фінальне тестування	3 дні
22	Запуск інтернет-магазину	1 день
23	Підтримка та вдосконалення після запуску	3 дні
24	Загальна тривалість робіт	71 день

4. Вимоги до складу й змісту робіт із введення web-додатку в експлуатацію

Для того, щоб web-додатком могли користуватися робітники селищної ради необхідно розмістити його у мережі Інтернет, тому необхідно придбати доменне ім'я та місце на хостингу. На хостинг переноситься web-додаток і наповнення бази даних з подальшою їх доробкою. Для коректного переносу web-додатку на хостинг необхідно, щоб параметри хостинга відповідали вимогам, зазначеним у ТЗ.

Додаток Б

Код бази даних товарів та клієнтів

```
import { MongoClient } from 'mongodb';
import mongoose from 'mongoose';

const connectionString = process.env.ATLAS_URI || '';
const options = {
  useNewUrlParser: true,
  useUnifiedTopology: true,
};

let client
let conn
let dbProducts

async function connectToMongoDB() {
  try {
    await mongoose.connect(connectionString, options);
    console.log('MongoDB is connected');
  } catch (err) {
    console.error(err);
  }
}

async function connectToMongoClient() {
  try {
    client = new MongoClient(connectionString);
    await client.connect();
    conn = client.db("Products");
    dbProducts = conn;
  } catch (e) {
    console.error(e);
  }
}

async function initialize() {
  await Promise.all([connectToMongoDB(), connectToMongoClient()]);
}

initialize();

export { conn, dbProducts };
export default mongoose.connection;
```

Додаток В

Програмний код корзини та системи покупок

```
import { useEffect, useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { RootState } from '../app/store';
import { CartItem, clearCart, removeFromCart, updateCart } from '../redux/slices/cartSlice';
import styles from './Cart.module.css'
import { selectIsAuth } from '../redux/slices/authSlice';
import { useRouteChange } from '../hooks/useRouteChange';

const useCartLogic = (isVisible: boolean, setIsVisible: React.Dispatch<React.SetStateAction<boolean>>) => {
  const routeChange = useRouteChange()
  const [showSuccessNotification, setShowSuccessNotification] = useState(false);

  const isAuth = useSelector(selectIsAuth);
  const dispatch = useDispatch();
  const cartItems: CartItem[] = useSelector((state: RootState) => state.cartItems);

  const handleClick = () => {
    setIsVisible(!isVisible);
  };

  const handleCheckoutCart = () => {
    if (isAuth === true) {
      dispatch(clearCart())
      setShowSuccessNotification(!showSuccessNotification);
    }
    else {
      setIsVisible(false)
      return routeChange('/login')
    }
  }

  const handleRemoveClick = (item: string) => {
    dispatch(removeFromCart(item));
  };

  const handleQuantityChange = (id: string, quantity: number) => {
    dispatch(updateCart({ id, quantity }));
  };
}
```

```

};

const calculateTotal = () => {
  let total = 0;
  cartItems.forEach((item) => {
    const itemQuantity = item.quantity || 1;
    total += parseFloat(item.price as any) * itemQuantity;
  });
  return total.toFixed(2);
};

useEffect(() => {
  document.body.style.overflow = isVisible ? 'hidden' : 'auto';
  const cartElement = document.querySelector(`.${styles.cart}`) as HTMLElement | null;
  if (cartElement) {
    cartElement.style.overflow = isVisible ? 'auto' : 'hidden';
  }
  return () => {
    document.body.style.overflow = 'auto';
  };
}, [isVisible]);

return {
  handleClick,
  handleRemoveClick,
  handleQuantityChange,
  calculateTotal,
  handleCheckoutCart,
  showSuccessNotification,
  setShowSuccessNotification
};
};

export default useCartLogic;

```

Додаток Г

Програмний код адміністративного інтерфейсу для управління магазином

```
import axios from "axios";
import { ChangeEvent, FormEvent, useState } from "react"
import PagesInformer from "../components/pages-informer";

interface FormData {
  name: string;
  collection: string;
  category: string;
  gender: string;
  color: string;
  size: string;
  url: string;
  price: string;
}

export const FormClothesCreate = () => {
  const [formData, setFormData] = useState<FormData>({
    name: "",
    collection: "",
    category: "",
    gender: "",
    color: "",
    size: "",
    url: "",
    price: "",
  })

  const handleSubmit = async (event: FormEvent<HTMLFormElement>) => {
    event.preventDefault();
    try {
      console.log(formData);
      await axios.post('http://localhost:5172/api/Clothes', formData)
    } catch (error) {
      console.error('Error submitting data', error)
    }
  }

  const handleChange = (event: ChangeEvent<HTMLInputElement>) => {
```

```

const { name, value } = event.target;
setFormData((prevFormData) => ({
  ...prevFormData,
  [name]: value
}))

}

return (
  <>
    <PagesInformer />
    <form onSubmit={handleSubmit}>
      <input type="text" name="name" value={formData.name} onChange={handleChange} placeholder="Name" />
      <input type="text" name="collection" value={formData.collection} onChange={handleChange}
placeholder="Collection" />
      <input type="text" name="category" value={formData.category} onChange={handleChange}
placeholder="Category" />
      <input type="text" name="gender" value={formData.gender} onChange={handleChange}
placeholder="Gender" />
      <input type="text" name="color" value={formData.color} onChange={handleChange} placeholder="Color" />
      <input type="text" name="size" value={formData.size} onChange={handleChange} placeholder="Size" />
      <input type="text" name="url" value={formData.url} onChange={handleChange} placeholder="URL" />
      <input type="text" name="price" value={formData.price} onChange={handleChange} placeholder="Price" />
      <button type="submit">Submit</button>
    </form>
  </>
)
}

```

Додаток Г

Валідація та верифікація веб-сайту

```
import { body } from "express-validator";
```

```
export const validateClothes = [  
  body('name').notEmpty(),  
  body('collection'),  
  body('category').notEmpty(),  
  body('gender').notEmpty(),  
  body('color').notEmpty(),  
  body('size').notEmpty(),  
  body('url').notEmpty(),  
  body('price').notEmpty(),  
];
```

```
export const loginValidation = [  
  body('email', 'Неверный формат почты').isEmail(),  
  body('password', 'Пароль должен быть минимум 5 символов').isLength({ min: 5 }),  
];
```

```
export const registerValidation = [  
  body('email', 'Неверный формат почты').isEmail(),  
  body('password', 'Пароль должен быть минимум 5 символов').isLength({ min: 5 }),  
  body('fullName', 'Укажите имя').isLength({ min: 3 }),  
];
```