МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Барба Давид Ігорович

Інтернет магазин з використанням технології React/Online store using React technology

спеціальність: 123 – Комп'ютерна інженерія
освітньо-професійна програма – Комп'ютерна інженерія

Кваліфікаційна робота

Виконав: студент групи КІ-41
Барба Давид Ігорович

Науковий керівник
к.т.н., О.Й. Піцун

ТЕРНОПІЛЬ-2023

РЕЗЮМЕ

Кваліфікаційна робота на тему «Програмний засіб проектування структурованої кабельної системи мережі» зі спеціальності 123 «Комп'ютерна інженерія» освітнього ступеня «бакалавр» містить 55 сторінок пояснюючої записки, 19 рисунків, 12 таблиць, 3 додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою кваліфікаційної роботи є розроблення інтернет магазину з використанням технології React/Online store using React technology.

Методи дослідження включають методи розробки односторінкових веб-сайтів і структуризація веб-застосунку за дорпромогою розробки компонент, методи оптимізації завантаження, адаптивне розміщення елементів веб-сайту.

Досліджено час завантаження та оптимізація веб-сайту, при первинному завантаженню веб-додатку на робочих станціях користувача. Розроблено алгоритм відображення елементів продукції під окремого користувача. Проведено його алгоритмізацію. Перевагою створеного алгоритму є адаптивність під кожного користувача ,що підвищує користувацький досвід.

Розроблено серверну частину для забезпечення запитів з front-end добази даних для отримання користувачів та елементів продукції та обробляти різні операції над даними. Для серверної реалізації було використано стек MERN, що складається з MongoDB, Express.js, React та Node.js.

Ключові слова: ДИНАМІЧНИЙ ВЕБ-САЙТ, ОНЛАЙН МАГАЗИН, ОДНОСТОРІНКОВИЙ ВЕБ-ДОДАТОК.

RESUME

Qualification work on the topic "Software tool for designing a structured cabling system of a network" in the specialty 123 "Computer Engineering" of the educational degree "Bachelor" contains 55 pages of explanatory note, 19 figures, 12 tables, 3 appendices. The volume of graphic material is 2 sheets of A3 format.

The purpose of the qualification work is to develop an online store using React technology/Online store using React technology.

Research methods include methods of developing single-page websites and structuring a web application according to the component development approach, methods of loading optimization, adaptive placement of website elements.

The loading time and optimization of the website during the initial download of the web application on user workstations are investigated. An algorithm for displaying product elements for an individual user has been developed. Its algorithmization was carried out. The advantage of the created algorithm is adaptability for each user, which increases the user experience.

The server side was developed to provide queries from the front-end data adder to retrieve users and product items and process various data operations. The MERN stack consisting of MongoDB, Express.js, React, and Node.js was used for the server implementation.

Keywords: DYNAMIC WEBSITE, ONLINE STORE, SINGLE-PAGE WEB APPLICATION.

ЗМІСТ

Перелік умовних скорочень 10
Вступ 11
Аналіз веб-ресурсів реалізації інтернет–магазину 14
1.1 Інструменти розробки динамічних веб-сайтів 14
1.2 Аналіз сайтів-аналогів 19
1.3 Аналіз елементів веб-сайту 24
1.4 Висновки та постановка задачі 29
2 Алгоритми роботи веб-сайту інтернет магазину 32
2.1 Алгоритми авторизації користувача 32
2.2 Алгоритм формування товарів 37
2.3 Структура бази даних 42
3 Програмна реалізація веб-сайту 49
3.1 Структура веб-сайту використані патерни та архітектурні стилі 49
3.2 Розміщення компонент на веб-сайті 57
3.3 Тестування та порівняльний аналіз 64
4 Техніко-економічний розділ 73
4.1 Розрахунок витрат на розробку програмного забезпечення 73
4.2 Розрахунок можливої ціни поректу 77
4.3 Розрахунок показників економічної ефективності 82
Висновки 86
Список використаних джерел 87
Додаток А Лістинг файлу «DataBase» 92
Додаток Б Довідка про використання 95
Додаток В Сертифікат участі в конференції 96

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

MERN	–	MongoDB ,Express ,React ,Node.js
HTML	–	Hyper text mark up language
CSS	–	Cascading Style Sheets
API	–	Application Programming Interface
SEO	–	Search Engine Optimization
UX	–	User Experience
UI	–	User Interface
БД	–	База даних
СУБД	–	Система управління базами даних
LAMP	–	LINUX,APACHE,MySQL,php
AJAX	–	Asynchronous JavaScript and XML
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
DB	–	Data base

ВСТУП

У сучасному світі надзвичайно важливо використовувати сучасні технології веб-розробки, які надають безліч переваг порівняно з застарілими технологіями.Зі стрімким розвитком технологій, веб-розробка пройшла довгий шлях з часів початку Інтернету. Пройшли ті часи, коли статичний HTML та CSS були єдиними доступними інструментами для створення веб-сайтів. Сьогодні, з'явилися сучасні технології, такі як React Redux, що пропонують низку переваг порівняно зі старішими фреймворками для веб-розробки. У сучасному цифровому світі наявність сильного онлайн-присутності є невід'ємною складовою успіху будь-якого бізнесу, який прагне залишатися конкурентоздатним. З кожним днем все більше і більше споживачів звертаються до Інтернету для придбання товарів та послуг, тому створення веб-магазинів, які забезпечують безшовний досвід покупок, стає дедалі важливішим завданням для бізнесів. Одним з найочевидніших відділів веб-розробки, які можуть скористатися новітніми технологіями, є веб-магазини. Веб-магазини вимагають швидкої загрузки сайту і оптимізації для індексації пошуковими системами. З такою великою кількістю конкуренції на ринку електронної комерції, повільно завантажені сайти або сайти зі складною навігацією можуть призвести до втрати продажів та зменшення задоволеності клієнтів. На щастя, сучасні технології, такі як React, Redux і Node.js, надають ряд функцій, які можуть бути використані для створення приємного досвіду покупок. Ці технології дозволяють швидше завантаження сторінок, легшу навігацію та поліпшену функціональність, що призводить до сайту, який є не тільки дружнім користувачу, але й ефективним. Крім поліпшення досвіду користувача, використання новітніх технологій також може призвести до кращих результатів в рейтингуванні пошуковими системами. Пошукові системи, такі як Google, надають перевагу сайтам, які оптимізовані та використовують сучасні технології розробки веб-сайтів.

З появою нових та інноваційних технологій, таких як React Redux, веб-розробники тепер мають доступ до потужних інструментів, які дозволяють їм

створювати складні та динамічні веб-додатки з легкістю. Ці сучасні технології пропонують безліч переваг над старішими фреймворками та інструментами, включаючи покращену продуктивність, більшу гнучкість та покращені користувацькі враження. За допомогою використання сучасних технологій веб-розробки, бізнес може створювати веб-сайти та додатки, які працюють швидше, ефективніше та є більш користувацько-орієнтованими, ніж будь-коли раніше. Це може призвести до збільшення трафіку, вищих конверсій та міцнішої онлайн-присутності, що всі ці чинники можуть сприяти більшому успіху та прибутковості в довгостроковій перспективі. В цьому контексті очевидно, що використання сучасної технології веб-розробки - це не просто питання підтримки останніх тенденцій.

Одним з найочевидніших відділів веб-розробки, які можуть скористатися новітніми технологіями, є веб-магазини. Це через те, що веб-магазини вимагають швидкої загрузки сайту і оптимізації для індексації пошуковими системами. З такою великою кількістю конкуренції на ринку електронної комерції, повільно завантажені сайти або сайти зі складною навігацією можуть призвести до втрати продажів та зменшення задоволеності клієнтів. На щастя, сучасні технології, такі як React, Redux і Node.js, надають ряд функцій, які можуть бути використані для створення безшовного досвіду покупок. Однією з найбільш значущих переваг сучасних технологій веб-розробки є їхній підхід до розробки. Нові технології надають можливість розробникам створювати додатки відповідно до концепції "компонентної архітектури", що дозволяє розбити додаток на менші, більш логічні модулі. Це полегшує розробку, тестування та підтримку додатків, що зменшує витрати на розробку та забезпечує високу якість продукту. Іншою важливою перевагою є зменшення часу розробки та випуску додатків.

Дослідження фокусується на аналізі веб-сторінок інтернет-магазину, розробленого з використанням стеку MERN. Стек MERN - це популярний фреймворк для веб-розробки, який поєднує MongoDB, Express.js, React та Node.js для створення ефективних та динамічних веб-додатків. Вивчаючи веб-сторінки цього інтернет-магазину, дослідники мають на меті оцінити загальний користувацький досвід, оцінити продуктивність та швидкість реагування веб-

сайту, а також визначити будь-які потенційні сфери для вдосконалення з точки зору дизайну, функціональності та взаємодії з користувачами. Дослідження має на меті надати цінну інформацію про ефективність стеку MERN у створенні інтернет-магазину та спрямувати майбутні зусилля з розробки.

Метою дослідження є вивчення різних підходів до розробки динамічних веб-сайтів, зокрема, інтернет-магазинів, з використанням стеку MERN. Стек MERN, який включає MongoDB, Express.js, React та Node.js, забезпечує надійний та масштабований фреймворк для створення динамічних та інтерактивних веб-додатків. Вивчаючи підходи, що використовуються при розробці інтернет-магазинів зі стеком MERN, дослідники прагнуть виявити найкращі практики, архітектурні патерни та ефективні стратегії для покращення функціональності, продуктивності та користувацького досвіду таких веб-сайтів. Дослідження має на меті внести цінний вклад у сферу веб-розробки та надати рекомендації для розробників, які бажають створювати динамічні інтернет-магазини з використанням стеку MERN.

Перший розділ цього присвячений аналізу веб-ресурсів, придатних для реалізації інтернет-магазину. Будуть розглянуті різні інструменти динамічної розробки веб-сайтів, щоб визначити їх потенціал у створенні надійних та інтерактивних онлайн-платформ. Завдяки поглибленому аналізу існуючих сайтів-аналогів будуть визначені ключові особливості та функціональні можливості, що сприятимуть успіху інтернет-магазину. Крім того, будуть досліджені різні елементи веб-сайту, включаючи навігацію, користувацький інтерфейс та організацію контенту, щоб отримати уявлення про ефективні принципи веб-дизайну. Результати цього аналізу послужать основою для визначення чітких цілей і завдань у наступних розділах.

АНАЛІЗ ВЕБ-РЕСУРСІВ РЕАЛІЗАЦІЇ ІНТЕРНЕТ–МАГАЗИНУ

1.1 Інструменти розробки динамічних веб-сайтів

Зважаючи на наш цифровий світ, інтернет став важливим інструментом для бізнесу, а реалізація інтернет-магазину стає все більш популярною серед підприємств різного розміру. Однак, з такою великою кількістю інтернет-магазинів, дуже важливо провести аналіз веб-ресурсів перед реалізацією інтернет-магазину. Один з важливих аспектів для розгляду - це дизайн інтернет-магазину. Дизайн має бути привабливим візуально, легким у використанні та зрозумілим. Заплутаний або незрозумілий дизайн може відвернути потенційних клієнтів та негативно вплинути на бізнес [1]. Також важливо забезпечити оптимізацію веб-сайту для мобільних пристроїв, оскільки все більше людей використовує телефони та планшети для перегляду та покупок в Інтернеті. Ще одним важливим аспектом для аналізу є функціональність інтернет-магазину. Інтернет-магазин має мати безпечну та надійну систему обробки платежів, а також ефективну та точну систему управління запасами. Також важливо мати чіткий та зрозумілий опис продуктів та інформацію про ціни, щоб уникнути непорозумінь та покращити досвід користувачів. Розробка веб-магазину включає різноманітні етапи і процеси, від концептуалізації до розгортання. Планування та концептуалізація: Це перший етап процесу розробки веб-магазину, де визначаються цілі, завдання та обсяг проекту. Він включає проведення маркетингових досліджень, визначення цільової аудиторії, розробку карти сайту та визначення функцій та можливостей веб-магазину.

Планування та концептуалізація є важливими етапами в розробці будь-якого проекту, включаючи дизайн та розробку веб-сайту. Ці етапи є основою, на якій будується решта проекту, і сильна фаза планування та концептуалізації може підготувати проект до успіху. Етап планування передбачає визначення цілей проекту, ідентифікацію зацікавлених сторін, визначення обсягу проекту та створення плану проекту. Важливо чітко визначити мету та цілі проекту, а також

ключові показники результативності, за допомогою яких буде вимірюватися його успіх. На цьому етапі також визначаються ресурси, необхідні для проекту, такі як персонал, бюджет та інструменти, і створюється графік проекту.

Етап концептуалізації передбачає брейнштормінг ідей для проекту, визначення унікальної пропозиції продажу проекту та створення концепту для проекту. Цей етап має вирішальне значення для розуміння цільової аудиторії, їх потреб та уподобань. Це розуміння використовується для розробки концепту, який задовольняє потреби цільової аудиторії, в той час як він відрізняє проект від його конкурентів. Під час етапу планування та концептуалізації дизайну та розробки веб-сайту важливо враховувати фактори, такі як користувацький досвід, оптимізація пошукових систем та масштабованість [2]. Чітке розуміння цих факторів допоможе забезпечити те, що веб-сайт є зручним у використанні, швидким у завантаженні та легко масштабується при зростанні бізнесу. Крім того, етап планування та концептуалізації також передбачає визначення архітектури веб-сайту та його функціональності. Веб-сайт повинен мати логічну структуру та бути простим у використанні для користувачів. Функціональність веб-сайту повинна відповідати меті та цілям проекту та задовольняти потреби користувачів. Нарешті, етап планування та концептуалізації включає в себе визначення контенту веб-сайту. Контент повинен бути привабливим та цікавим для користувачів, а також повинен бути оптимізований для пошукових систем. Важливо також враховувати потреби користувачів та інформацію, яка їм потрібна, та забезпечити, що контент веб-сайту відповідає цим потребам. Взагалі, етап планування та концептуалізації є важливими етапами в розробці веб-сайту. Чітке визначення мети та цілей проекту, а також концепту та контенту веб-сайту, допоможуть забезпечити успіх проекту та задовольнення потреб користувачів.

Дизайн та розробка веб-сайту, фаза планування та концептуалізації є абсолютно критичною. Ця фаза є основою всього проекту та встановлює тон для всього, що буде відбуватися далі. Якщо планування та концептуалізація будуть виконані належним чином, то це може зробити відмінність між успішним веб-сайтом, який залучає та радує свою аудиторію, та веб-сайтом, який не

задовольняє своїх цілей та провалюється. Тож, що саме включає фаза планування та концептуалізації? По-перше, все починається з визначення цілей проекту. Що повинен досягнути веб-сайт [3]? Чи це електронний комерційний сайт, призначений для продажу продуктів? Сайт-портфоліо, призначений для демонстрації роботи дизайнера? Блог, який надає корисну інформацію читачам? Чітко визначаючи цілі веб-сайту, команда проекту може забезпечити, що кожне рішення, що буде прийнято, спрямоване на досягнення цих цілей.

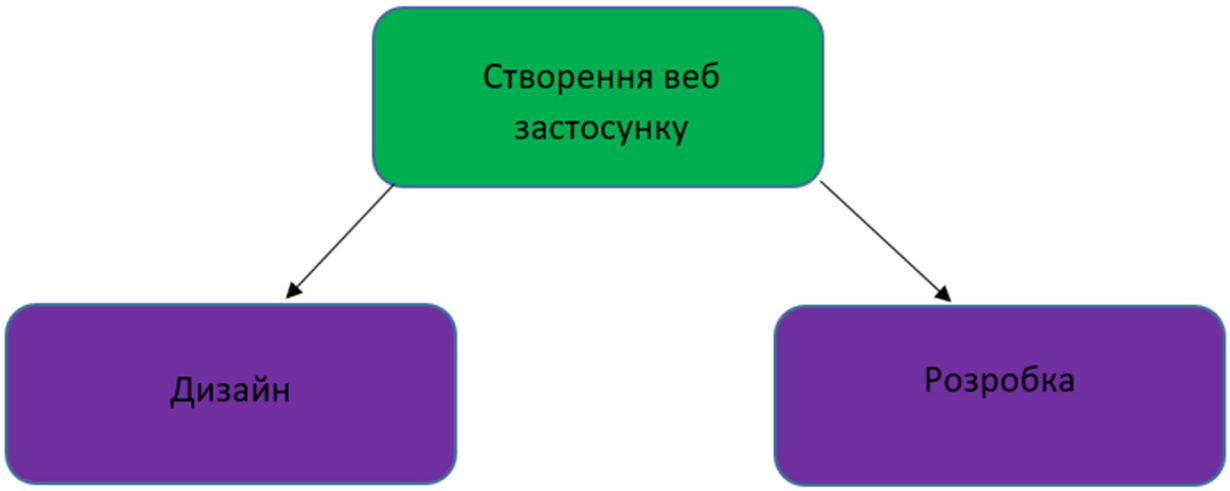

Рисунок 1.1 – Етапи створення веб-сайту

Дизайн: На цьому етапі створюється дизайн веб-магазину, включаючи інтерфейс користувача (UI) та досвід користувача (UX). Дизайн UI включає створення візуально привабливого веб-сайту, який легко навігується та використовується. Дизайн UX спрямований на створення безперешкодного та інтуїтивно зрозумілого досвіду користувача, оптимізуючи макет та можливості веб-сайту. Дизайн є важливим етапом у розробці веб-магазину. На цьому етапі створюються інтерфейс користувача (UI) та досвід користувача (UX), які є ключовими елементами успішної роботи веб-магазину. Дизайн UI включає створення візуально привабливого веб-сайту, який легко навігується та використовується [4]. Дизайн повинен бути не тільки естетичним, але й функціональним. Дизайнер повинен забезпечити зручну навігацію веб-сайтом, щоб користувачі могли швидко знайти те, що їх цікавить. Кольорова гамма,

типографіка та графічний дизайн повинні відображати бренд та створювати єдиний дизайн-концепт. З іншого боку, дизайн UX спрямований на створення безперешкодного та інтуїтивно зрозумілого досвіду користувача, оптимізуючи макет та можливості веб-сайту. Гарний дизайн UX повинен надавати пріоритет потребам та очікуванням користувачів, забезпечуючи легкість використання та навігації на веб-сайті. Дизайнер повинен враховувати різноманітні фактори, такі як цільова аудиторія, маршрут користувача та цілі веб-сайту. Щоб досягти унікального та цікавого дизайну, дизайнер повинен звертати увагу на найновіші тенденції та інновації у веб-дизайні. Він повинен бути креативним та думати нестандартно, використовуючи унікальні елементи, що роблять веб-сайт виділятися серед конкурентів. Однак, дуже важливо знайти баланс між інноваціями та функціональністю, щоб забезпечити, що веб-сайт виконує свою функцію. Отже, успішність веб-магазину в значній мірі залежить від його дизайну, зокрема UI та UX. Дизайнер повинен створити візуально привабливий веб-сайт, що легко навігується, з безперешкодним досвідом користувача. Надаючи пріоритет потребам та очікуванням користувачів та слідкуючи за останніми тенденціями дизайну, дизайнер може створити унікальний та захоплюючий веб-магазин. Дизайн є важливим аспектом успіху будь-якого веб-магазину. Він може зробити користувальницький досвід успішним або невдалим і, в кінці кінців, визначити, чи здійснять користувачі покупку. Щоб створити дизайн, який буде одночасно цікавим і унікальним, дизайнерам потрібно знайти баланс між естетикою та функціональністю.

Сучасні веб-магазини також набагато більше наголошують на естетиці та брендуванні. Сьогодні веб-магазини створені для того, щоб бути візуально привабливими, з високоякісними зображеннями та графікою, які демонструють товари та створюють згуртований ідентифікаційний стиль бренду. Наприклад, багато веб-магазинів модного одягу використовують великі, високоякісні зображення, що дозволяють користувачам побачити товар детальніше та отримати уявлення про його стиль та якість [5]. Ще одним значним відмінністю між старомодними веб-сайтами та сучасними веб-магазинами є використання високотехнологічних рішень та інтерактивних функцій. Наприклад, багато веб-

магазинів сьогодні використовують чат-ботів та віртуальних помічників, щоб забезпечити персоналізовану підтримку клієнтів, а інші використовують доповнену реальність, щоб дозволити користувачам побачити, як буде виглядати товар в їхньому будинку перед покупкою.

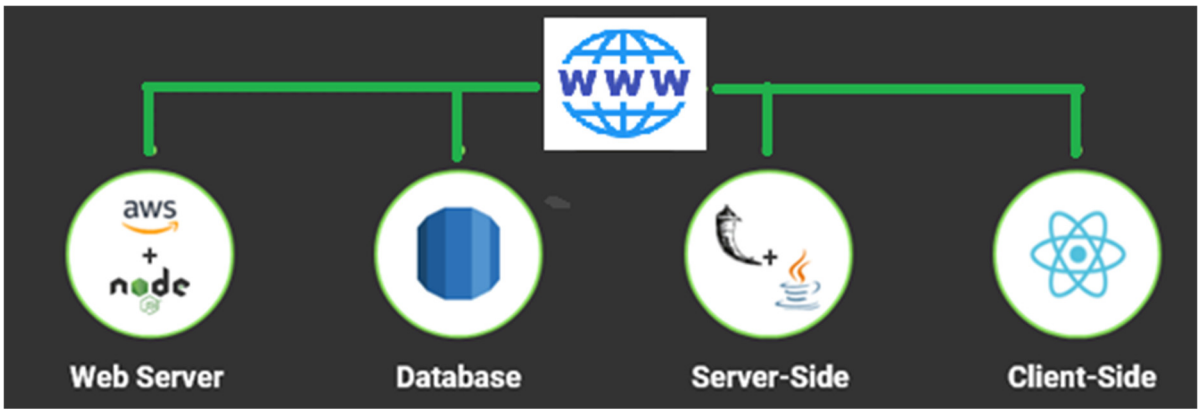

Рисунок 1.2 – Складові частини веб-сайту

Дизайн є одним з найважливіших аспектів створення високоякісного веб-магазину. Гарно розроблений веб-магазин не тільки виглядає візуально привабливо, але також забезпечує безперешкодний та легкий для користувачів досвід, що спонукає їх здійснити покупку. Однак, важливо пам'ятати, що дизайн - це лише один елемент при створенні успішного веб-магазину. Інші важливі аспекти включають якість продукту, ціноутворення, обслуговування клієнтів та маркетинг. Комбінуючи відмінний дизайн з цими ключовими елементами, власники веб-магазинів можуть створити дійсно високоякісний продукт, який виділяється на переповненому онлайн-ринку. Крім того, дизайн є постійним процесом, який потребує постійної оцінки і вдосконалення. Оскільки технології розвиваються, а поведінка користувачів змінюється, дизайнери повинні адаптувати свої дизайни, щоб відповідати новим викликам та можливостям. Залишаючись в курсі дизайнерських тенденцій та найкращих практик, власники веб-магазинів можуть забезпечити, що їх дизайни залишаються актуальними та ефективними [6].Дизайн веб-магазину відіграє важливу роль у його успіху. Завдяки пріоритетному користувацькому досвіду, естетиці та передовій технології дизайнери можуть створити високоякісний веб-магазин, який

виділяється на тлі конкурентів. В поєднанні з іншими ключовими аспектами сайту, такими як якість продукту та обслуговування клієнтів, добре спроектований веб-магазин може створити позитивний враження на користувачів та спонукати їх стати лояльними клієнтами.

1.2 Аналіз сайтів-аналогів

В сучасному висококонкурентному ландшафті електронної комерції важливо докладно вивчити конкурентів і проаналізувати їх веб-сайти. Аналізуючи веб-сайти конкурентів в галузі електронної комерції, ви можете отримати цінні уявлення про їх стратегії, сильні та слабкі сторони і використовувати цю інформацію для поліпшення свого веб-сайту і збереження конкурентної переваги.

Завдяки докладному аналізу веб-сайтів конкурентів в галузі електронної комерції, можна краще зрозуміти їх цільову аудиторію, пропозиції продуктів, стратегії ціноутворення та маркетингові тактики. Ця інформація може допомогти виявити прогалини в своїй пропозиції, коригувати свої стратегії ціноутворення для збереження конкурентоспроможності та підтримувати свої маркетингові зусилля, щоб досягти своєї цільової аудиторії більш ефективно. Додатково, аналізуючи дизайн веб-сайту, досвід користувача та навігацію конкурентів, можливо виявити області для покращення на власному веб-сайті. Виявляючи те, що працює добре для конкурентів і що ні, можна приймати обґрунтовані рішення щодо покращення власного веб-сайту та створення більш привабливого досвіду користувача. У цю еру прийняття рішень на основі даних, аналіз конкурентів в галузі електронної комерції є невід'ємною складовою успішної стратегії електронної комерції [7]. Отримуючи відомості про стратегії, сильні та слабкі сторони конкурентів, можна оптимізувати власний веб-сайт електронної

комерції, покращити досвід користувача та, нарешті, розвивати свій бізнес. Однією з ключових переваг аналізу сайтів-конкурентів електронної комерції є можливість виявлення нових можливостей для зростання та розширення. Аналізуючи пропозиції продуктів та стратегії ціноутворення конкурентів, можна виявити прогалини на ринку та розробити нові продукти або послуги, що задовольнять ці потреби. Крім того, можна отримати інформацію про нові ринки або сегменти клієнтів, які можливо пропустили та створити цілеспрямовані маркетингові кампанії, щоб досягти цих аудиторій. Ще один важливий аспект аналізу веб-сайтів конкурентів - оцінка їх дизайну та користувацького досвіду. Аналізуючи, як веб-сайти конкурентів розроблені та як вони взаємодіють зі своєю аудиторією, можна отримати цінні уявлення про те, що працює добре, а що ні. Ця інформація може допомогти створити більш ефективний дизайн веб-сайту та користувацький досвід, який резонує з аудиторією, спонукає до взаємодії та врешті-решт призводить до більшої кількості продажів. Крім того, аналіз стратегій маркетингу та реклами конкурентів також може надати цінних уявлень. Зрозумівши, як конкуренти знаходять свою цільову аудиторію, які канали використовують та який меседж працює для їхньої аудиторії, можна налаштувати власні маркетингові тактики та повідомлення, щоб краще досягати й залучати свою цільову аудиторію.

Аналіз електронних комерційних сайтів конкурентів є важливою складовою будь-якої успішної стратегії електронної комерції [8]. Здобувши уявлення про стратегії, сильні та слабкі сторони конкурента, можна виявити можливості для зростання та розширення, оптимізувати дизайн та користувацький досвід веб-сайту, та вдосконалити свої маркетингові зусилля, щоб досягати своєї цільової аудиторії більш ефективно. Регулярно аналізуючи своїх конкурентів, можна залишатися попереду конкуренції, пристосовуватися до змінних ринкових умов та досягати більшого успіху в електронній комерції.

Успішний електронний бізнес важливо тримати руку на пульсі конкурентів та регулярно аналізувати їх веб-сайти. Таким чином, можна виявити ті області, де відрізняєтеся та ті, де потрібно покращення, щоб залишатися конкурентоспроможним. Однією з перших речей, які потрібно врахувати при

порівнянні інтернет-магазину з конкурентами, є загальний досвід користувача. Чи забезпечує веб-сайт інтуїтивно зрозумілий, легкий для навігації досвід для клієнтів? Як порівнюється дизайн веб-сайту з конкурентами? Чи є області, де можна покращити вигляд та відчуття веб-сайту? Додатково, важливо проаналізувати стратегії маркетингу та реклами конкурента. Як порівнюється трафік на веб-сайті з тим, що мають конкуренти? Чи використовують конкуренти маркетингові канали, які не використовується? Аналізуючи тактики маркетингу та реклами конкурентів, можна виявити області, в яких можна покращити маркетингові зусилля та досягти своєї цільової аудиторії більш ефективно. Крім того, важливо оцінити досвід обслуговування клієнтів, який надається веб-магазином та конкурентами. Як порівняти сервіс обслуговування клієнтів з конкурентами? Чи є якісь області покращити досвід обслуговування клієнтів на веб-сайті? Регулярний аналіз веб-магазину порівняно з конкурентами допоможе визначити області , де потрібно покращитися. Ця інформація допоможе оптимізувати веб-сайт, поліпшити досвід клієнтів та, в результаті, зростити бізнес. Важливо пам'ятайтати, в сьогоднішньому висококонкурентному електронному комерційному ландшафті, важливо вести передову боротьбу з конкурентами та регулярно пристосовуватися до змін ринкових умов, щоб досягти успіху.

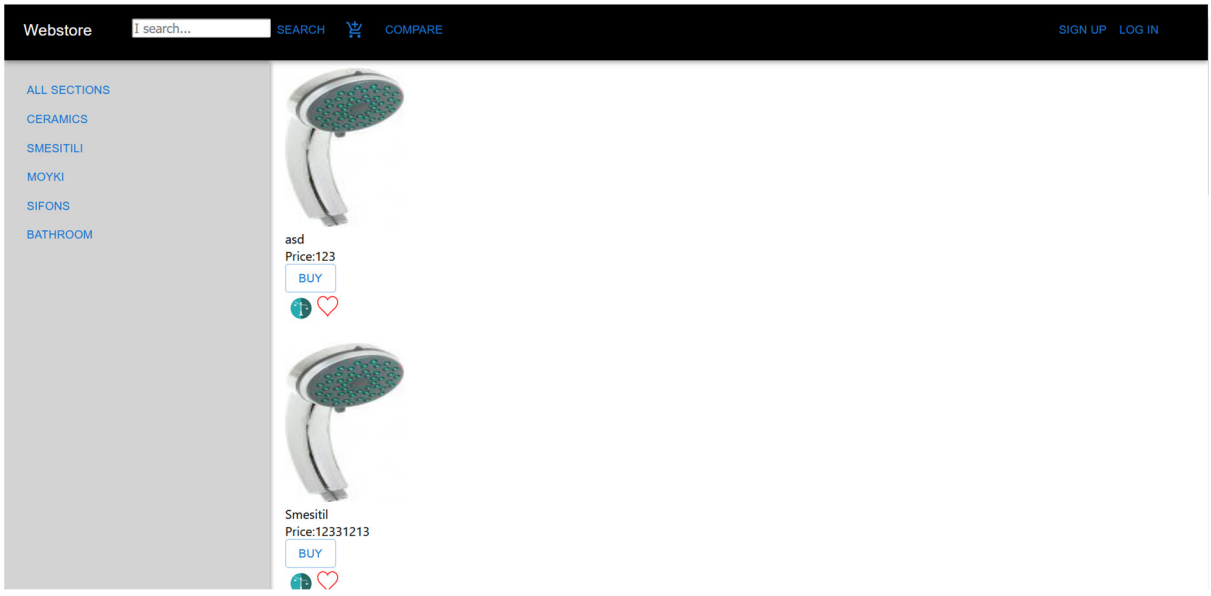

Рисунок 1.3 – Зовнішній вигляд головної сторінки розробленого сайту

Найбільший електронний магазин в Україні - Rozetka, який пропонує широкий асортимент товарів, включаючи електроніку, побутову техніку, косметику та інше [9]. Щодо технологічного стеку, використовуваного Rozetka, він базується переважно на стеку LAMP, що означає Linux, Apache, MySQL та PHP. Це загальновживаний технологічний стек для розробки динамічних веб-додатків, включаючи електронні комерційні платформи.

Rozetka використовує інші технології та фреймворки, такі як JavaScript, AJAX та jQuery, щоб покращити користувацький досвід та забезпечити безшовну функціональність на веб-сайті. Крім того, веб-сайт оптимізований для мобільних пристроїв, що є все більш важливим, оскільки все більше користувачів отримують доступ до електронних комерційних веб-сайтів через свої мобільні пристрої. Розетка також використовує інші технології та фреймворки, такі як JavaScript, AJAX та jQuery, щоб поліпшити користувацький досвід та забезпечити безшовну функціональність на веб-сайті. Крім того, веб-сайт оптимізований для мобільних пристроїв, що стає все більш важливим, оскільки все більше користувачів отримують доступ до веб-сайтів електронної комерції через свої мобільні пристрої [10]. Варто зазначити, що Розетка вкладає значні кошти у свою технологію та інфраструктуру, що відображається у швидкості завантаження веб-сайту, легкій навігації та безшовному процесі оформлення замовлення. Це спрямовано на забезпечення високоякісного користувацького досвіду, що безумовно допомогло Розетці стати найбільшою електронною комерційною платформою в Україні. В цілому, технологічний стек, що використовується Розеткою, та її акцент на надання переважного користувацького досвіду демонструють важливість інвестування у технологію та інфраструктуру для збереження конкурентоспроможності в індустрії електронної комерції.

Виправданість інвестування в технологію та інфраструктуру є критичною для збереження конкурентоспроможності в індустрії електронної комерції, і Розетка є яскравим прикладом цього.

Технологічний стек, який використовується Розеткою, включає в себе різні компоненти та рішення, які сприяють наданню високоякісного користувацького

досвіду. Це включає в себе швидкодіючу і масштабовану інфраструктуру, удосконалені системи управління контентом, особисті рекомендації, спрощений процес покупок, ефективну систему пошуку, оптимізовані мобільні додатки та інше.

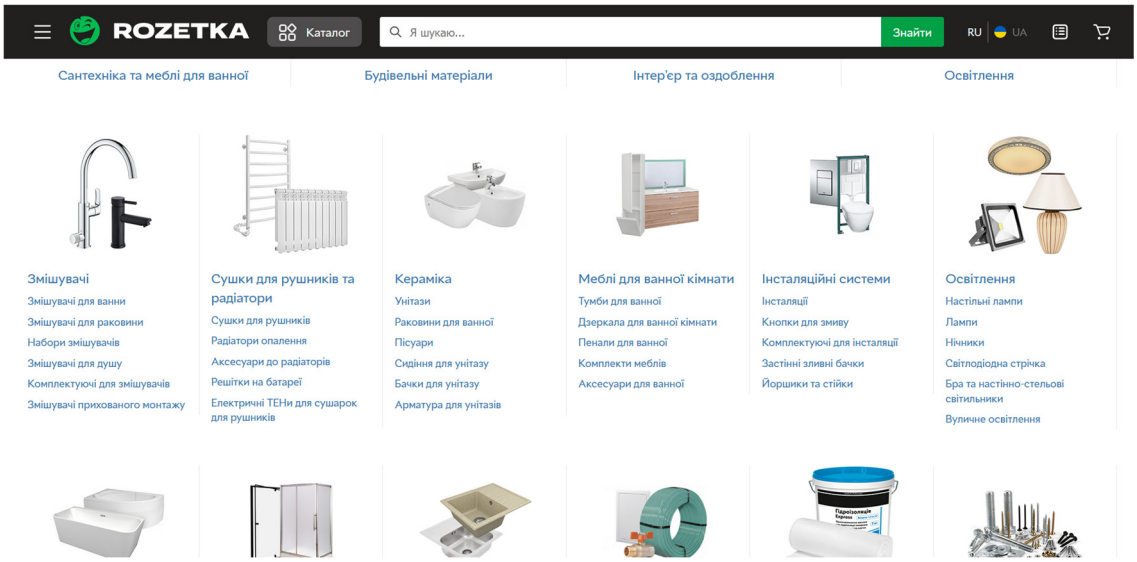

Рисунок 1.4 – Зовнішній вигляд головної сторінки сайту аналогу

Іншим значущим веб-сайтом електронної комерції в Україні є Алло, який спеціалізується на електроніці та побутовій техніці. На відміну від Розетки, технічний стек Алло базується на стеку MEAN, що означає MongoDB, Express.js, AngularJS та Node.js. MongoDB є базою даних, яка зберігає дані в документах, подібних до JSON. Express.js - це фреймворк веб-додатків для Node.js, який дозволяє розробникам створювати міцні та масштабовані веб-додатки. AngularJS є фреймворком для фронтенду, який дозволяє створювати динамічні та відгукні веб-сторінки, а Node.js є середовищем виконання JavaScript, яке дозволяє скриптувати на стороні сервера [11]. Ця технологічна структура дозволяє компанії Allo надавати високорівневий та інтерактивний веб-сайт, що є важливим для клієнтів, які хочуть купувати електроніку онлайн. Крім того, веб-сайт оптимізований для мобільних пристроїв, що є все більш важливим, оскільки все більше користувачів отримують доступ до електронної комерції через свої мобільні пристрої. В цілому використання MEAN-стеку в технологічній інфраструктурі Allo дозволяє компанії пропонувати безшовний досвід

користувачів, залишаючись конкурентоспроможною на ринку електронної комерції в Україні.

1.3 Аналіз елементів веб-сайту

Веб-магазин є складною платформою, що складається з багатьох різних елементів та компонентів, кожен з яких відіграє важливу роль у створенні ефективного та ефективного веб-сайту.

Домашня сторінка - це головна сторінка інтернет-магазину, яка слугує першою точкою контакту між бізнесом і користувачем. Це перше враження, яке користувач отримує при відвідуванні веб-сайту, і воно відіграє вирішальну роль у створенні атмосфери та передачі повідомлення про бренд [12]. Домашня сторінка зазвичай містить назву або логотип компанії, навігаційне меню, пошуковий рядок і різні розділи, які демонструють основні продукти або акції. Дизайн і макет домашньої сторінки повинні бути візуально привабливими і зручними для користувача, з чіткою ієрархією інформації та інтуїтивно зрозумілою навігацією.

Рисунок 1.5 – Панель домашньої сторінки розробленого веб-сайту

Меню навігації є важливим елементом будь-якого інтернет-магазину, оскільки воно надає відвідувачам чіткий і організований спосіб доступу до різних сторінок і продуктів на сайті. Зазвичай воно розташовується у верхній

частині головної сторінки або на бічній панелі і містить список категорій і підкатегорій, які мають відношення до пропозицій бізнесу. Дизайн і макет навігаційного меню повинні бути інтуїтивно зрозумілими і простими у використанні, з чіткою ієрархією інформації, яка полегшує відвідувачам пошук потрібного. Назви категорій повинні бути описовими та зрозумілими, а підкатегорії повинні бути чітко вкладені у свої батьківські категорії [13]. Окрім категорій продуктів, навігаційне меню може також містити посилання на важливі сторінки, такі як сторінка "Про нас", сторінка контактів і сторінка поширених запитань (FAQ).

Списки товарів є важливим компонентом будь-якого інтернет-магазину, оскільки вони надають відвідувачам детальну інформацію про товари, доступні для продажу. Ці сторінки, як правило, містять зображення товарів, опис, ціни та відгуки і призначені для демонстрації особливостей і переваг кожного товару. Зображення товару є важливим елементом оголошення, оскільки вони надають візуальне уявлення про товар і допомагають відвідувачам зрозуміти його зовнішній вигляд і функціональність. Зображення повинні бути якісними і оптимізованими для швидкого завантаження, з різними видами і ракурсами, щоб дати відвідувачам повне уявлення про товар. Опис продукту повинен бути чітким і лаконічним, надавати детальну інформацію про особливості, переваги та технічні характеристики продукту. Мова, що використовується в описах, має бути легкою для розуміння і не містити жаргонізмів, щоб відвідувачі могли прийняти обґрунтоване рішення про покупку. Інформація про ціни повинна бути на видному місці в списках товарів, з чітким зазначенням будь-яких знижок, акцій або спеціальних пропозицій [14]. Це допомагає відвідувачам зрозуміти цінність продукту і прийняти обґрунтоване рішення про те, чи варто його купувати.

Відгуки є ще одним важливим елементом оголошень про товари, оскільки вони є соціальним доказом і допомагають побудувати довіру та авторитет серед потенційних клієнтів. Відгуки повинні бути розміщені на видному місці на сторінці списку товарів, а відвідувачі повинні мати можливість фільтрувати і сортувати їх за різними критеріями, наприклад, за рейтингом або релевантністю.

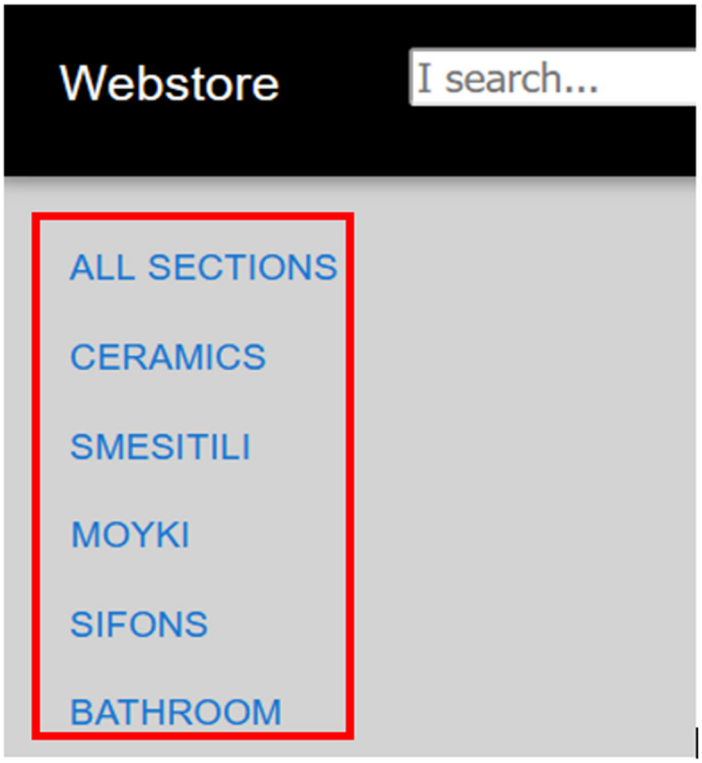

Рисунок 1.6 – Навігаційна панель розробленого веб-сайту

Загалом, списки товарів є важливим компонентом будь-якого інтернет-магазину, оскільки вони надають відвідувачам інформацію, необхідну для прийняття обґрунтованих рішень про покупку [15]. Добре продумана сторінка з переліком товарів може допомогти підвищити залученість і стимулювати продажі, що в кінцевому підсумку призведе до більш успішного і прибуткового онлайн-бізнесу.

Кошик для покупок - це основна функція веб-сайтів електронної комерції, яка дозволяє клієнтам додавати товари, які вони хочуть придбати, і переходити до оформлення замовлення, щоб завершити процес покупки [16]. Це простий і зручний спосіб для клієнтів вибрати і зберігати кілька товарів, які вони хочуть купити, в одному місці, перш ніж завершити покупку.

Функція кошика має кілька переваг як для клієнтів, так і для бізнесу. Для клієнтів вона робить процес покупки більш зручним і ефективним, дозволяючи їм вибирати кілька продуктів і відстежувати свої покупки. Для бізнесу це

допомагає збільшити продажі, заохочуючи клієнтів додавати більше товарів до кошика і забезпечуючи спрощений процес оформлення замовлення.

Рядок пошуку - це фундаментальна функція будь-якого веб-сайту електронної комерції. Він дозволяє клієнтам шукати певні продукти або ключові слова в інтернет-магазині, полегшуючи їм пошук того, що вони шукають. Рядок пошуку є важливим інструментом як для клієнтів, так і для підприємств електронної комерції, оскільки він допомагає поліпшити загальний користувацький досвід і збільшити продажі [17]. Пошуковий рядок може надати цінну інформацію про поведінку та вподобання покупців. Аналізуючи пошукові терміни, які використовують покупці, підприємства електронної комерції можуть визначити тенденції, популярні товари та сфери, де їм може знадобитися вдосконалити свої товарні пропозиції або пошукову функціональність.

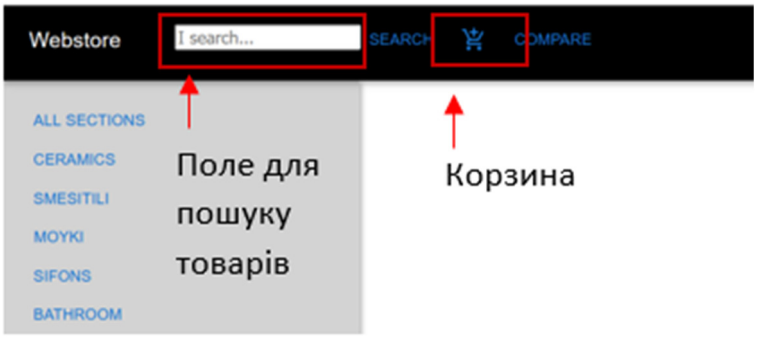

Рисунок 1.7 – Панель пошуку та корзина розробленого веб-сайту

Функція облікового запису клієнта є важливим аспектом сучасних платформ електронної комерції, що дозволяє клієнтам створювати та керувати своїми обліковими записами на веб-сайті або в додатку. Ця функція пропонує широкий спектр переваг для клієнтів, включаючи можливість зберігати інформацію про доставку та оплату, переглядати історію замовлень та отримувати ексклюзивні пропозиції. Однією з головних переваг облікових записів клієнтів є зручність, яку вони пропонують клієнтам. Дозволяючи клієнтам зберігати свою інформацію про доставку та оплату, вони можуть легко здійснювати майбутні покупки без необхідності щоразу вводити свої дані

повторно [18]. Це економить час і зусилля, а також робить процес купівлі більш спрощеним і ефективним.

У багатьох інтернет-магазинах користувачам пропонуються пов'язані товари або рекомендовані товари на основі їхньої історії переглядів і покупок. Це часто робиться за допомогою алгоритмів, які аналізують поведінку користувача і відповідно до неї надають рекомендації.Ці рекомендації можуть відображатися різними способами, наприклад "Клієнти, які купили цей товар, також купили", де показані товари, які зазвичай купують разом з товаром, який користувач переглядає в даний момент.

Списки бажань - це корисна функція для будь-якого інтернет-магазину, оскільки вони дозволяють користувачам зберігати товари, які вони зацікавлені придбати пізніше. Створюючи список бажань, користувачі можуть відстежувати товари, які вони планують придбати, без необхідності проходити весь процес купівлі одразу. Списки бажань, як правило, знаходяться в кабінеті користувача, де він може легко отримати доступ до збережених товарів і керувати ними.

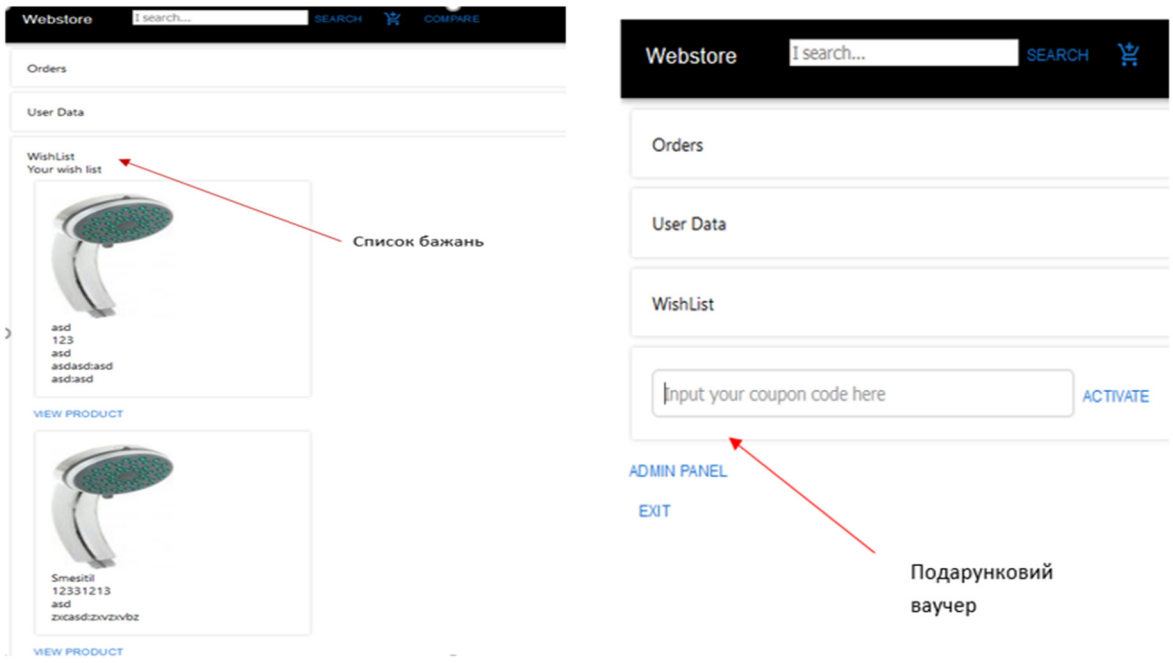

Рисунок 1.8 – Панель списку бажань та поле ввода подарункового ваучеру розробленого веб-сайту

Подарункові картки або ваучери є популярною функцією в багатьох інтернет-магазинах, оскільки вони дозволяють клієнтам купувати товари для себе або в якості подарунка для когось іншого. Ці картки або ваучери зазвичай купуються на певну суму і можуть бути використані для придбання будь-якого товару в інтернет-магазині [19]. У кабінеті користувача подарункові картки або ваучери зазвичай зберігаються у вигляді балансу або кредиту. Користувачі можуть переглянути свій баланс і обміняти свою подарункову картку або ваучер на будь-який товар в інтернет-магазині під час оформлення замовлення.

1.4 Висновки та постановка задачі

При розробці інтернет-магазину з використанням React, Redux, Node.js та Express важливо враховувати висновки та завдання, які необхідно виконати. Ось деякі важливі компоненти та міркування, які слід мати на увазі: дизайн веб-сайту, каталог продукції, кошик для покупок, процес оформлення замовлення, управління користувачами, управління замовленнями,безпека веб-застосунку.

З точки зору розробки, веб-сайт повинен бути побудований з використанням фреймворку React для фронтенду, з Redux для управління станом. Бекенд повинен бути побудований з використанням Node.js та Express. Веб-сайт повинен бути розроблений з модульною архітектурою, з компонентами, які можна повторно використовувати та поширювати на різних сторінках. Щоб забезпечити масштабованість та зручність обслуговування веб-сайту, він повинен бути побудований з використанням найкращих практик, таких як розбиття коду, ліниве завантаження та кешування [20]. Тестування та налагодження слід проводити протягом усього процесу розробки, щоб виявити та виправити будь-які проблеми на ранній стадії.

Мета роботи – розробка динамічного веб сайту інтернет-магазину засобами

Для досягнення мети розробки динамічного веб-сайту інтернет-магазину засобами MERN стеку, потрібно вирішити наступні задачі:

Провести порівняльний аналіз сайтів-аналогів: Для цього необхідно провести дослідження і вивчити існуючі інтернет-магазини, які працюють у схожій ніші або мають схожий функціонал. Аналізуючи їх, можна визначити переваги та недоліки конкурентів, з'ясувати, які функції та можливості вони надають своїм клієнтам. Також необхідно звернути увагу на їх дизайн, навігацію, корзину, систему оплати, оформлення замовлення, систему управління контентом та інші аспекти [21]. Це допоможе виокремити основні вимоги до веб-сайту та знайти можливості для його покращення, розробки унікальних функцій та підвищення конкурентоспроможності магазину.

Провести порівняльний аналіз програмних засобів розробки веб-сайтів: Існує безліч програмних засобів та технологій для розробки веб-сайтів. Необхідно провести детальний аналіз різних стеків технологій, таких як MERN (MongoDB, Express.js, React.js, Node.js), MEAN (MongoDB, Express.js, AngularJS, Node.js), LAMP (Linux, Apache, MySQL, PHP) та інших. Необхідно порівняти їх переваги та недоліки, можливості для розширення та масштабування, ефективність роботи з базами даних, швидкодію, безпеку та інші фактори. Необхідно врахувати особливості проекту, його масштаби, потенційну кількість користувачів, вимоги до швидкодії та безпеки для вибору найбільш підходящого стеку технологій. Також треба звернути увагу на наявність підтримки та ресурсів для розробки у вибраному стеку технологій, оскільки це може значно спростити роботу та допомогти вирішити можливі проблеми під час розробки та підтримки веб-сайту.

Розробити алгоритм авторизації користувачів в системі: Потрібно розробити механізм, який дозволить користувачам створювати облікові записи, входити в систему з допомогою унікального логіну та пароля, а також відновлювати забуті паролі [22]. Алгоритм авторизації повинен забезпечувати безпеку, шифрування паролів та застосування механізмів перевірки достовірності введених даних.

Розробити алгоритм формування товарів на веб-сторінці: Необхідно розробити алгоритм, який динамічно формує веб-сторінку з відображенням товарів на основі даних з бази даних. Цей алгоритм повинен враховувати фільтри, сортування та пошук, які надаються користувачам для вибору потрібних товарів. Важливо, щоб алгоритм був ефективним і швидким, щоб забезпечити зручне переглядання та навігацію по товарах.

Розробити структуру бази даних: Потрібно розробити структуру бази даних, яка відображає всі необхідні сутності, такі як товари, категорії, замовлення, користувачі та інші, а також зв'язки між ними. Необхідно визначити атрибути для кожної сутності і встановити правильні залежності та обмеження для забезпечення цілісності даних. Крім того, важливо розробити ефективні запити до бази даних для швидкого доступу та маніпуляцій з даними.

Програмно реалізувати веб-сайт інтернет-магазину: На основі вибраного стеку технологій (MERN) потрібно розробити програмний код, який реалізує функціональність інтернет-магазину [23]. Це включає розробку серверної логіки, маршрутизацію, взаємодію з базою даних, реалізацію корзини, обробку замовлень, авторизацію користувачів, відображення товарів на веб-сторінках та інші функції, необхідні для роботи інтернет-магазину.

Провести порівняльний аналіз з аналогами: Для досягнення конкурентної переваги важливо провести порівняльний аналіз існуючих аналогічних інтернет-магазинів. Потрібно дослідити їх функціональність, дизайн, швидкодію, відгуки користувачів та інші аспекти [24]. Цей аналіз допоможе виявити переваги та недоліки конкурентів і зрозуміти, як можна покращити свій веб-сайт, щоб залучити і задовольнити більше клієнтів.

Розробити алгоритм персоналізованої рекомендації товарів: Потрібно створити алгоритм, який на основі взаємодії користувача з веб-сторінкою та аналізу його попередніх виборів і демографічних даних забезпечує персоналізовані рекомендації товарів.

2 АЛГОРИТМИ РОБОТИ ВЕБ-САЙТУ ІНТЕРНЕТ МАГАЗИНУ

2.1 Алгоритми авторизації користувача

Авторизація - це процес перевірки того, що користувач має необхідні дозволи для доступу до певної інформації або виконання певних дій на веб-сайті. Це важлива частина безпеки веб-сайту, оскільки вона гарантує, що тільки авторизовані користувачі можуть отримати доступ до конфіденційної інформації або виконати дії, які можуть поставити під загрозу безпеку веб-сайту.

Алгоритм авторизації визначає правила та процедури перевірки особи користувача та його повноважень [25]. Зазвичай він включає комбінацію облікових даних для входу, таких як ім'я користувача та пароль, а також більш просунуті методи, такі як двофакторна автентифікація або біометрична ідентифікація.

Алгоритм авторизації важливий для веб-сайтів з кількох причин. По-перше, він допомагає запобігти несанкціонованому доступу до конфіденційної інформації. Наприклад, якщо на веб-сайті зберігаються дані користувачів, такі як особиста інформація або фінансові дані, дуже важливо, щоб ця інформація була доступна лише авторизованим користувачам. Надійний алгоритм авторизації гарантує, що тільки ті, хто має відповідні дозволи, можуть отримати доступ до цієї інформації.

Авторизація у веб-застосунку ділить користувачів на дві категорії: звичайний користувач і адмін.

Другою причиною важливості алгоритму авторизації є забезпечення контролю над функціоналом та можливостями, доступними користувачам. Завдяки правильно налаштованому алгоритму, адміністратор може встановлювати обмеження та рівні доступу для різних категорій користувачів, дозволяючи адаптувати функціонал веб-сайту до потреб та правил, встановлених власником або адміністратором.

Рисунок 2.1 – Категорії користувачів розробленого веб-сайту

Користувач інтернет-магазину має можливість переглядати, шукати і купувати продукти або послуги, які доступні для продажу на платформі. Користувач також має можливість створити обліковий запис в інтернет-магазині, який може надати додаткові функції та переваги.

Після створення облікового запису користувач зможе переглядати історію своїх замовлень, відстежувати статус поточних замовлень, а також керувати своєю особистою інформацією та платіжними реквізитами [26]. Він також може залишати відгуки та рейтинги для придбаних товарів, допомагаючи іншим користувачам приймати обґрунтовані рішення щодо власних покупок.

У веб-магазині адміністратор, відіграє вирішальну роль в управлінні та підтримці платформи. Адміністратори мають широкий спектр прав і обов'язків, включаючи можливість додавати і видаляти продукти, керувати обліковими записами користувачів і контролювати загальну функціональність сайту.

Одним із ключових обов'язків адміністратора є забезпечення безпеки та стабільності роботи інтернет-магазину. Це може включати в себе впровадження заходів безпеки для захисту даних користувачів, моніторинг сайту на предмет будь-якої підозрілої активності та вирішення будь-яких технічних проблем, що виникають [27]. Адміністратори також можуть відповідати за оновлення програмного забезпечення та інфраструктури сайту, щоб забезпечити його актуальність і повну функціональність.

Окрім цих технічних обов'язків, адміністратори також відіграють важливу роль у повсякденній роботі інтернет-магазину. Це може включати управління замовленнями, обробку платежів і відповіді на запити клієнтів. Адміністратори також можуть відповідати за створення та підтримку рекламних кампаній, таких як розпродажі та знижки, для залучення трафіку на сайт і збільшення продажів.

Сайт інтернет-магазину створений на основі стеку MERN. Сайт створений на основі стеку MERN - це складний веб-додаток, який вимагає декількох алгоритмів для правильної роботи. Стек MERN - це комбінація чотирьох різних технологій: MongoDB, Express.js, React.js та Node.js [28]. Кожна з цих технологій відіграє певну роль у розробці сайту інтернет-магазину.

Алгоритм авторизації на сайті, побудований на основі стеку MERN, може використовувати різні методи індентифікації та перевірки дозволів. Наприклад, при реєстрації нового користувача, йому можуть бути надані унікальні облікові дані, такі як ім'я користувача та пароль, які потім зберігаються в базі даних MongoDB.

Для перевірки дозволів користувача при авторизації, серверний компонент, побудований з використанням Express.js та Node.js, може виконувати перевірку правильності введеного пароля та відповідність ім'я користувача і пароля, збережених у базі даних. При успішній перевірці, сервер генерує токен аутентифікації, який передається клієнту.

На стороні клієнта, побудованого з використанням React.js, зберігається отриманий токен, який використовується для авторизованого доступу до захищених ресурсів і функцій [29]. Крім того, можуть використовуватися додаткові методи безпеки, такі як двофакторна автентифікація, яка може вимагати підтвердження шляхом введення додаткового коду або використання біометричних даних. Додаткові методи безпеки забезпечують додатковий рівень захисту, щоб гарантувати безпечний і надійний доступ до конфіденційної інформації та функціональних можливостей.

Рисунок 2.2 – Блок-схема авторизації користувача

Коли користувач вперше потрапляє на сторінку авторизації інтернет-магазину, він бачить форму, яка зазвичай запитує його облікові дані для входу. Ця форма містить поля для введення електронної пошти або імені користувача, а також його пароля.

Користувач повинен ввести правильні облікові дані для входу, щоб продовжити. Якщо користувач ще не зареєстрував обліковий запис в інтернет-магазині, його може бути перенаправлено на сторінку реєстрації, де він зможе створити обліковий запис і налаштувати свої облікові дані.

Коли користувач вперше потрапляє на сторінку реєстрації, він, бачить форму, яка запитує його особисту інформацію, таку як ім'я, адреса електронної

пошти та пароль [30]. Користувачеві потрібно вибрати унікальне ім'я користувача та пароль, які він буде використовувати для входу в інтернет-магазин в майбутньому.

Реєстраційна форма може також попросити користувачів надати додаткові дані, такі як адреса доставки або платіжна інформація. Ця інформація може бути використана для спрощення процесу оформлення замовлення та полегшення придбання товарів у веб-магазині.

Коли користувач потрапляє на сторінку входу, він бачить форму, в якій потрібно ввести своє ім'я користувача та пароль. Ця форма може також містити додаткові функції безпеки, такі як CAPTCHA, щоб запобігти спробам автоматизованих ботів увійти в систему.

Після того, як користувач введе свої облікові дані і надішле форму, інтернет-магазин перевірить його особу і надасть йому доступ до свого облікового запису. Якщо ім'я користувача або пароль невірні, інтернет-магазин відображає повідомлення про помилку з проханням повторити спробу або скинути пароль.

```
app.post('/login', async (req, res) => {
    const { email, password } = req.body;

    try {  // Find the user with the given email and password
        const user = await User.findOne({ email, password });

        if (user) { // Login successful
            res.json({ success: true });
        } else {
            // Login failed
            res.json({ success: false, message: 'Invalid email or password' });
        }
    } catch (error) {// Something went wrong
        res.status(500).json({ error: error.message });
    }
});
```

Рисунок 2.3 – Код авторизації написаний за допомогою Express та MongoDB

Код визначає обробник маршруту Express.js для кінцевої точки /login, яка приймає POST-запити. При отриманні POST-запиту він витягує поля email і пароль з тіла запиту, використовуючи деструктуризацію присвоєння. [31]

Якщо під час запиту до бази даних сталася помилка, сервер повертає відповідь 500 з повідомленням про помилку, що вказує на те, що пішло не так.

Код визначає схему Mongoose для об'єкта користувача, який буде зберігатися в базі даних MongoDB. Об'єкт UserSchema визначає декілька властивостей для об'єкта користувача:

fullName: Обов'язкова властивість рядка, яка представляє повне ім'я користувача.

email: Обов'язкова властивість рядка, яка представляє адресу електронної пошти користувача. Ця властивість є унікальною, що означає, що кожна електронна адреса може бути використана лише один раз. [32]

passwordHash: Обов'язкова властивість рядка, яка представляє хешований пароль користувача. Ця властивість є не відкритим текстом, а хеш-значенням пароля, яке обчислюється за допомогою алгоритму хешування.

orders: Властивість масиву, яка представляє історію замовлень користувача. За замовчуванням ця властивість має значення порожнього масиву.

avatarUrl: Рядкова властивість, яка представляє URL-адресу зображення аватара користувача.

2.2 Алгоритм формування товарів

У веб-магазині, побудованому з використанням стеку MERN, алгоритм формування списку товарів є критично важливим компонентом додатку. Стек MERN, який складається з MongoDB, Express, React і Node.js, є популярним і потужним інструментом для побудови повностекових веб-додатків.

Алгоритм формування списку товарів у веб-магазині на основі стеку MERN складається з декількох ключових кроків. Спочатку додаток отримує дані про товари з бази даних MongoDB за допомогою бібліотеки Mongoose . Потім дані про товари фільтруються і сортуються на основі критеріїв пошуку

користувача, таких як категорія, ціновий діапазон і пошук за ключовими словами.

Після того, як дані про товари відфільтровані та відсортовані, вони передаються до React-компоненту, що відповідає за відображення списку товарів. Цей компонент використовує різноманітні можливості React, такі як стан та пропси, щоб динамічно відобразити список товарів на основі відфільтрованих та відсортованих даних. [33]

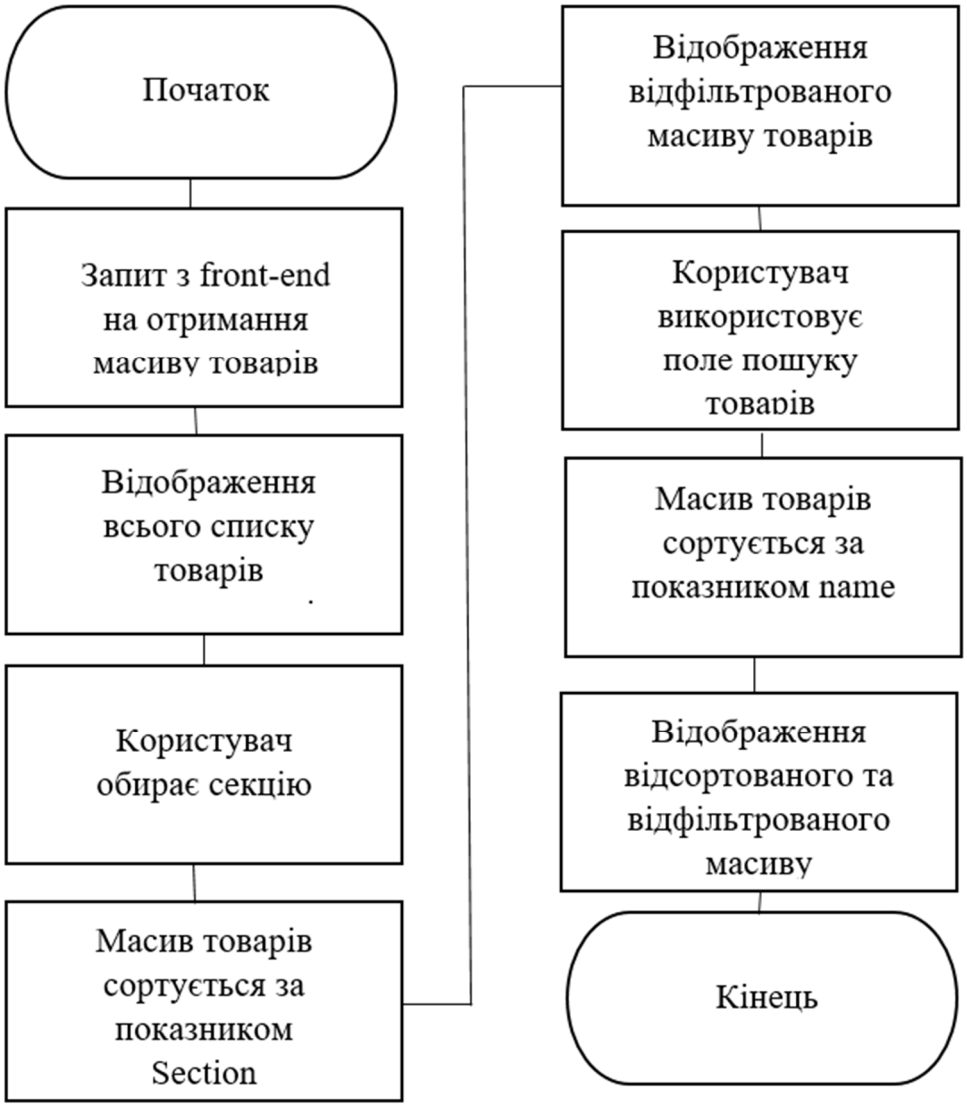

Рисунок 2.4 – Блок-схема формування товарів для окремого користувача

Запит до бази даних MongoDB з фронтенду React-додатку є поширеною вимогою для багатьох веб-додатків. Щоб отримати список товарів з бази даних MongoDB за допомогою React, потрібно виконати кілька кроків.

React-компонент, що відповідає за відображення списку товарів, повинен зробити HTTP-запит до Express-сервера. Цей запит повинен містити будь-які параметри, необхідні для фільтрації або сортування списку товарів, наприклад, пошуковий запит або діапазон цін. [34]

Запити до бази даних MongoDB можна робити безпосередньо з фронтенду за допомогою JavaScript, зазвичай це не рекомендується. У більшості випадків безпечніше і масштабованіше виконувати запити до бази даних на стороні сервера за допомогою таких інструментів, як Express і Mongoose.

```
const [products, setProducts] = useState([]);

useEffect(() => {
    fetch('/api/products?searchTerm=mySearchTerm')
        .then(response => response.json())
        .then(data => setProducts(data));
}, []);
```

Рисунок 2.5 – Програмний код запита до бази даних

Наведений вище фрагмент коду демонструє функціональний React-компонент ProductList, який отримує дані з кінцевої точки API за допомогою методу fetch(), а потім використовує хуки useState() та useEffect() для керування станом компонента.

```
├── State: products (Array) - Stores the retrieved products from the API.
├── State: sort (String) - Stores the selected sort option.
├── State: filter (String) - Stores the selected filter option.
│
├── useEffect - Runs when sort or filter changes:
│   ├── axios.get - Makes an HTTP GET request to the '/products' endpoint with the
selected sort and filter parameters.
│   │   ├── Success: setProducts - Updates the products state with the retrieved data.
│   │   └── Error: console.log - Logs any encountered errors.
│   └── Dependency Array: [sort, filter]
│
└── Render:
    ├── <h1>Product List</h1> - Heading for the product list.
    ├── <form> - Form to select sort and filter options.
    │   ├── Sort dropdown: <select> - Dropdown to select sort option by price.
    │   │   ├── Options: None, Low to high, High to low.
    │   │   ├── value={sort} - Binds the value to the sort state.
    │   │   └── onChange - Updates the sort state when the selection changes.
    │   └── Filter dropdown: <select> - Dropdown to select filter option by category.
    │       ├── Options: All, Electronics, Clothing, Home.
    │       ├── value={filter} - Binds the value to the filter state.
    │       └── onChange - Updates the filter state when the selection changes.
    │
    └── <ul> - Unordered list to display the product items.
        └── {products.map} - Maps over the products array and generates a list item
for each product.
            ├── <h2>{product.name}</h2> - Displays the product name.
            ├── <p>{product.description}</p> - Displays the product description.
            ├── <p>{product.price}</p> - Displays the product price.
            └── <p>{product.category}</p> - Displays the product category.
```

Рисунок 2.6 – Графічна схема сортування товарів у розробленому веб-сайті

У цьому React-компоненті спочатку імпортується необхідні бібліотеки - React, useState і useEffect з бібліотеки 'react', а також аксіоми для створення HTTP-запитів. Потім визначається компонент функції ProductList, яка буде повертати список продуктів, отриманих з API. [35]

Алгоритм отримує дані про товари з бази даних MongoDB за допомогою API Express.js і відображає їх в інтуїтивно зрозумілому та зручному для користувача вигляді за допомогою React.js. Бекенд отримує дані та надсилає їх до фронтенду, який потім відображає їх у браузері. Код організовано у два окремі файли, один для бекенду, а інший для фронтенду, щоб зберегти логіку окремо та легко підтримувати.

Алгоритм кошика для покупок: Цей алгоритм дозволяє користувачам додавати товари до кошика та підраховує загальну вартість замовлення, включаючи податки та вартість доставки. Алгоритм використовує MongoDB для зберігання вмісту кошика та цін на товари. Node.js, серверне середовище виконання JavaScript, використовується для обробки розрахунків і відповідного оновлення кошика. React.js використовується для відображення кошика та його вмісту в режимі реального часу, дозволяючи користувачам бачити загальну вартість замовлення, коли вони додають або видаляють товари з кошика.

Алгоритм імпортує необхідні бібліотеки/модулі (Express, body-parser і Mongoose), встановлює з'єднання з базою даних MongoDB під назвою "shopping-cart" на локальній машині та імпортує модель "Cart" з файлу "./models/cart" для подальшого використання в додатку.

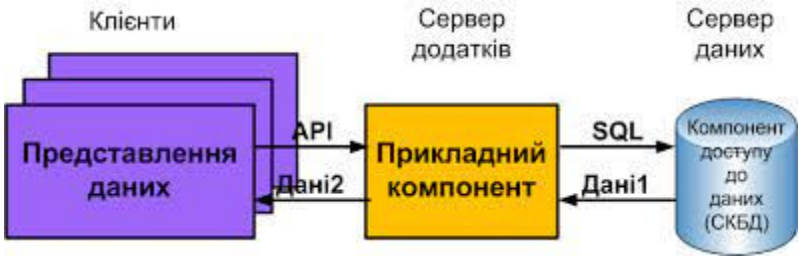

Рисунок 2.7 – Схема потоку даних веб-магазину

Якщо запит виконано успішно, викликається функція setCart, яка оновлює стан кошика новим елементом [36]. Ця функція спочатку створює копію масиву cartItems, а потім або оновлює кількість існуючого товару, або додає новий товар до масиву. Потім вона обчислює новий проміжний підсумок, податок і загальну суму для кошика і повертає об'єкт з цими значеннями та оновлений масив cartItems.

```
    </div>
<div>
    <label htmlFor="productId">Product ID:</label>
    <input type="text" id="productId" value={productId} onChange={e =>
setProductId(e.target.value)} />
</div>
<div>
    <label htmlFor="quantity">Quantity:</label>
    <input type="number" id="quantity" value={quantity} onChange={e =>
setQuantity(e.target.value)} />
</div>
<button onClick={addToCart}>Add to Cart</button>
<div>
    <h2>Cart Items:</h2>
    {cart.cartItems && cart.cartItems.map(item => (
        <div key={item._id}>
            <p>{item.name} - {item.quantity} x {item.price}</p>
            <button onClick={() => removeFromCart(item._id)}>Remove</button>
        </div>
    ))}
    <p>Subtotal: {cart.subtotal}</p>
    <p>Tax: {cart.tax}</p>
    <p>Total: {cart.total}</p>
</div>
</div>
);
};

export default ShoppingCart;
```

Рисунок 2.8 – Код кошика покупок

Компонент ShoppingCart також відображає поточні товари в кошику разом з їхньою назвою, кількістю та ціною, а також кнопку для видалення товару з кошика. Також відображається проміжний підсумок, податок і загальна сума для кошика.

Хук useEffect використовується для отримання поточного вмісту кошика з сервера під час монтування компонента. Ці дані зберігаються в об'єкті стану кошика, який має властивості для елементів кошика, проміжного підсумку, податку та загальної суми. [37]

Перевагою використання React для створення інтернет-магазину є можливість забезпечити плавний та безперешкодний користувацький досвід. Завдяки віртуальному DOM та ефективному рендерингу React користувачі можуть переглядати різні розділи та елементи без жодних затримок. Це може

зробити весь процес перегляду набагато приємнішим, що призведе до більшої задоволеності клієнтів та збільшення продажів.

```
const ItemSectionSelector = ({ items, setDisplayedItems }) => {
    const handleSectionChange = (e) => {
        const selectedSection = e.target.value;
        setSection(selectedSection);
        if (selectedSection === 'all') {
            setDisplayedItems(items);
        } else {
            const filteredItems = items.filter((item) => item.section ===
selectedSection);
            setDisplayedItems(filteredItems);
            <label htmlFor="section-selector">Choose a section:</label>
            <select id="section-selector" value={section}
onChange={handleSectionChange}>
                <option value="all">All Items</option>
                <option value=" "></option>
                <option value=" "></option>
                <option value=" "></option>
```

Рисунок 2.9 – Код селектора для фільтрації відображення товарів

Наведений вище код є прикладом селектора в React, який дозволяє користувачам фільтрувати масив елементів на основі вибраної секції. Компонент ItemSectionSelector приймає два пропси: items, який представляє весь масив елементів, і setDisplayedItems, який є функцією, що оновлює масив відображуваних елементів на основі вибраної секції.

2.3 Структура бази даних

MongoDB - популярна NoSQL-база даних, яку часто використовують у стеку MERN (MongoDB, Express, React, Node.js) для розробки веб-магазинів.

MongoDB зберігає дані в JSON-подібних документах, що робить її ідеальним рішенням для зберігання даних у веб-магазині [38]. Дані в MongoDB організовані в колекції, і кожен документ в колекції може мати власну унікальну структуру. Ця гнучкість особливо корисна для розробки інтернет-магазинів, оскільки дозволяє розробникам зберігати та отримувати дані у спосіб, який має сенс для їхнього конкретного випадку використання.

При розробці інтернет-магазину з MongoDB розробники можуть скористатися багатьма його можливостями, серед яких

Масштабованість: MongoDB розроблена з можливістю горизонтального масштабування, що означає, що вона може обробляти великі обсяги даних і трафіку без шкоди для продуктивності. Це робить її ідеальним вибором для інтернет-магазинів, яким потрібно обробляти великі обсяги трафіку та продажів.

Гнучкість: гнучка модель даних MongoDB дозволяє легко зберігати та отримувати дані у спосіб, який має сенс для конкретного випадку використання. Це може бути особливо корисно для інтернет-магазинів, які часто мають складні структури даних і взаємозв'язки.

Продуктивність: MongoDB відома своїми високими швидкостями читання і запису, що може допомогти підвищити продуктивність інтернет-магазинів. Крім того, можливості індексування MongoDB можуть допомогти розробникам швидко отримувати дані з великих колекцій.

Легка інтеграція: MongoDB легко інтегрується з іншими технологіями стеку MERN, такими як Node.js та React, що робить її популярним вибором для розробки веб-магазинів.

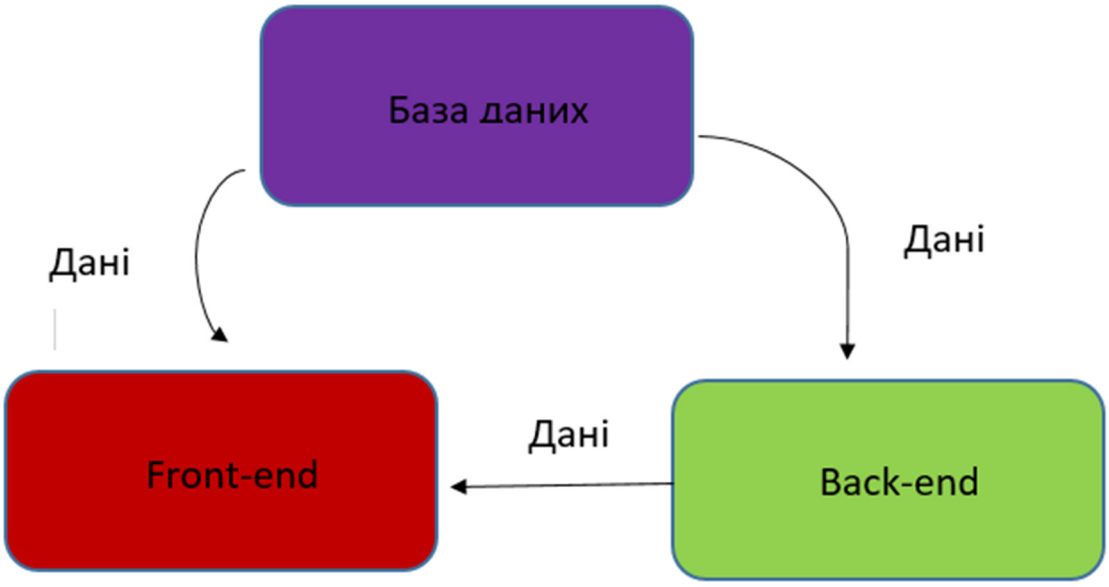

Рисунок 2.10 – Схема потоку даних веб-магазину

MongoDB - це технологія баз даних, яка добре підходить для розробки веб-магазинів у стеку MERN. Її гнучкість, масштабованість, продуктивність та легка

інтеграція роблять її популярним вибором для розробників, які створюють веб-магазини, що потребують обробки великих обсягів даних та трафіку. [39]

Реляційні та нереляційні бази даних - це два різні типи систем управління базами даних, які використовуються для зберігання, організації та пошуку даних. Основна відмінність між цими двома типами баз даних полягає в їхній

Відмінність між цими двома типами баз даних - їх масштабованість. Реляційні бази даних призначені для вертикального масштабування, що означає додавання більшої обчислювальної потужності на одному сервері. Такий підхід має обмеження з точки зору масштабованості, оскільки він може стати дорогим і непрактичним для дуже великих наборів даних. На противагу цьому, нереляційні бази даних призначені для горизонтального масштабування, що означає додавання більшої кількості серверів до кластеру [40]. Такий підхід забезпечує практично необмежену масштабованість, що робить їх добре придатними для роботи з великими наборами даних.

Схеми забезпечують рівень безпеки та перевірки даних. Визначаючи структуру і типи даних для кожного поля, схеми можуть допомогти запобігти ін'єкційним атакам і гарантувати, що дані будуть належним чином відформатовані і перевірені перед тим, як зберігатися в базі даних.

Схеми MongoDB забезпечують визначену структуру для документів у колекції, гарантуючи узгодженість і точність даних, а також полегшуючи обробку змін у структурі даних. Вони також забезпечують додатковий рівень безпеки та перевірки даних, допомагаючи запобігти атакам і забезпечити цілісність даних.

Крім масштабованості і схем, ще одна важлива відмінність між реляційними і нереляційними базами даних полягає у способі організації даних. У реляційних базах даних дані організовані у взаємозв'язані таблиці зі структурованими реляціями між ними. Нереляційні бази даних, зокрема MongoDB, використовують неструктурований або напівструктурований формат збереження даних, такий як документи JSON.

Нереляційні бази даних, зокрема MongoDB, зазвичай надають більшу гнучкість у роботі з даними, оскільки дозволяють зберігати різнородну

структуру документів у колекціях. Це дозволяє швидше здійснювати зміни у структурі даних без необхідності перебудови всієї бази даних.

```
const UserSchema=new mongoose.Schema({
    fullName:{
        type:String ,
        required:true
    },
    email:{
        type:String,
        required: true,
        unique:true
    },
    passwordHash:{
        type:String,
        required:true
    },
    orders:{
        type:Array,
        default:[],
    },
    avatarUrl:String
},{
    timestamps:true,
})
export default mongoose.model('User',UserSchema);
```

Рисунок 2.11 – Схема бази даних користувача веб-додатку

Код є визначенням схеми для користувача в додатку веб-магазину, побудованому за допомогою стеку MERN. Схема визначена за допомогою бібліотеки Mongoose, яка є інструментом об'єктно-документного відображення (ODM) для MongoDB.

UserSchema визначає поля, які будуть зберігатися для кожного користувача в базі даних MongoDB. Схема включає поля для повного імені користувача, електронної пошти, хешу пароля, замовлень та URL-адреси аватара. [41]

Поле passwordHash має тип String і також позначене як обов'язкове. Це поле використовується для зберігання хешованої версії пароля користувача, що гарантує, що пароль користувача не буде зберігатися в базі даних у вигляді простого тексту.

Поле orders має тип Array і за замовчуванням є порожнім масивом. Це поле використовується для зберігання всіх замовлень, які користувач розмістив у веб-магазині.

```
UserSchema
├── fullName: String (required)
├── email: String (required, unique)
├── passwordHash: String (required)
├── orders: Array (default: [])
├── avatarUrl: String
└── timestamps: true (automatically adds createdAt and updatedAt fields)
```

Рисунок 2.12 –Представлення схеми MongoDB для моделі User

Поле avatarUrl має тип String і використовується для зберігання URL-адреси зображення аватара користувача. [42]

Опція timestamps:true у визначенні схеми додає до кожного документа користувача два поля - createdAt і updatedAt, які автоматично заповнюються мітками часу, коли документ було створено і коли його було востаннє оновлено.

Визначення схеми забезпечує структуру для зберігання інформації про користувача в базі даних MongoDB і гарантує, що дані будуть належним чином перевірені та захищені.

User:
Id
fullname
email
password
orders
Avatar icon

Таблиця 2.1 – Схема користучача у вигляді таблиці

У веб-магазині, побудованому на MongoDB, дуже важливо мати чітко визначену схему для товарів, які будуть продаватися. Схема визначає структуру даних, які будуть зберігатися в базі даних, і гарантує, що дані будуть послідовними і точними. [43]

При розробці схеми для товарів у веб-магазині важливо визначити поля, які є важливими для товарів, що будуть перераховані та продані. Деякі з важливих полів, які повинні бути включені в схему, є наступними:

```
const NewItemSchema = new mongoose.Schema({
    fullName:{
        type:String ,
    price:{
        type:Number,
    description:{
        type:String
    characteristics:{
        type:Array,
    viewCounts:{
        type:Number,
    section:{
        type:String,
    image: {
        data: Buffer,
},{
    timestamps:true,
});
export default mongoose.model('NewItem',NewItemSchema);
```

Рисунок 2.13 – Схема елементу товару в базі даних

Фрагмент коду визначає схему для нового елемента в базі даних MongoDB за допомогою бібліотеки Mongoose, спеціально для веб-магазину, побудованого за допомогою стеку MERN. Схема називається "NewItemSchema" і визначає структуру документа, який буде зберігатися в колекції "NewItem".[44]

Включивши ці поля в "NewItemSchema", інтернет-магазин, побудований за допомогою стеку MERN, може гарантувати, що вся необхідна інформація буде збережена для кожного елемента, що полегшить клієнтам перегляд і придбання товарів. Крім того, наявність чітко визначеної схеми може полегшити розробку

та підтримку інтернет-магазину, оскільки вона забезпечує чітку структуру для даних, що зберігаються.

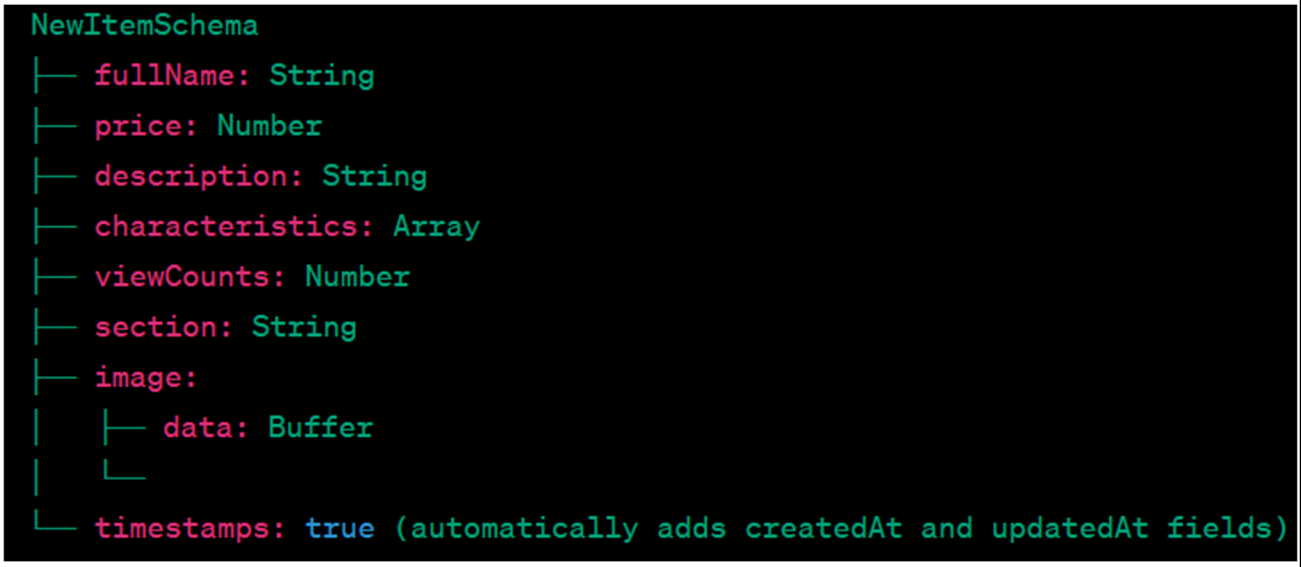

Рисунок 2.14 – Представлення схеми MongoDB для моделі елементу товару

графічне представлення схеми MongoDB для моделі NewItem

Алгоритм оформлення замовлення: Цей алгоритм обробляє процес оформлення замовлення, який включає збір інформації про користувача, обробку платежів і відправку електронних листів з підтвердженням [45]. Алгоритм використовує Node.js для обробки платежів і MongoDB для зберігання інформації про замовлення. Express.js використовується для створення API, який отримує платіжну інформацію від платіжного шлюзу та оновлює статус замовлення. React.js використовується для відображення сторінки підтвердження замовлення та надсилання імейлів з підтвердженням користувачеві та власнику магазину.

Алгоритм підтримки користувачів: Цей алгоритм забезпечує клієнтську підтримку користувачів, дозволяючи їм зв'язатися з власниками магазину зі своїми питаннями або проблемами. Він використовує Node.js для обробки запитів на підтримку та MongoDB для зберігання даних підтримки. [46]

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ

3.1 Структура веб-сайту використані патерни та архітектурні стилі

Стек MERN, до якого входять MongoDB, Express.js, React.js та Node.js, є потужною комбінацією технологій, які можна використовувати для створення веб-додатків та інтернет-магазинів. При створенні інтернет-магазину з використанням стеку MERN дуже важливо звернути увагу на структуру та патерни додатку, оскільки вони можуть суттєво вплинути на продуктивність та зручність використання інтернет-магазину.

Однією з основних переваг використання стеку MERN є його здатність ефективно обробляти велику кількість даних. Однак, якщо структура інтернет-магазину не оптимізована, це може негативно вплинути на продуктивність і час завантаження інтернет-магазину. Тому важливо проектувати схему бази даних, кінцеві точки API та React-компоненти таким чином, щоб максимізувати продуктивність та ефективність.

Структура та патерни веб-магазину, побудованого з використанням стеку MERN, також можуть впливати на взаємодію з користувачем. Добре структурований інтернет-магазин буде інтуїтивно зрозумілим і легким для навігації, що може призвести до збільшення залученості користувачів і конверсії. З іншого боку, погано структурований інтернет-магазин може заплутати і розчарувати користувачів, що може призвести до високого показника відмов і втрачених продажів.

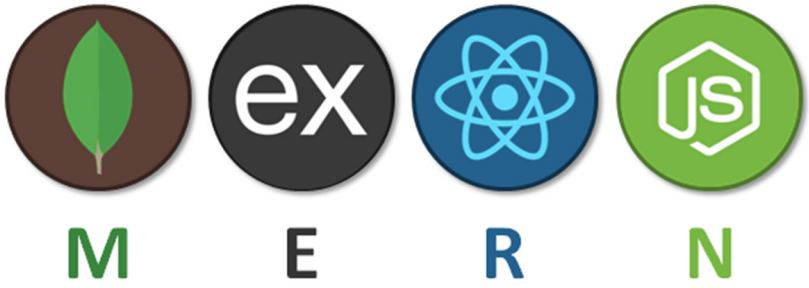

Рисунок 3.1 – Стек технологій використаний для розробки веб-додатку

Важливим аспектом добре структурованого інтернет-магазину є його здатність до масштабування. У міру зростання інтернет-магазину важливо мати масштабовану архітектуру, здатну впоратися зі зростаючим трафіком і обсягом даних. Дотримуючись найкращих практик розробки стеку MERN, таких як модуляризація коду та використання ефективних структур даних, розробники можуть гарантувати, що веб-магазин залишатиметься масштабованим та ефективним. [47]

Окрім масштабованої архітектури, оптимізація продуктивності бази даних має вирішальне значення для зростаючого інтернет-магазину. Впровадження належної індексації, механізмів кешування та методів оптимізації запитів може значно покращити час відгуку та ефективно обробляти збільшені обсяги даних.

Крім того, використання методів балансування навантаження між декількома серверами або хмарними екземплярами може ефективно розподілити трафік і запобігти перевантаженню серверів у пікові періоди.

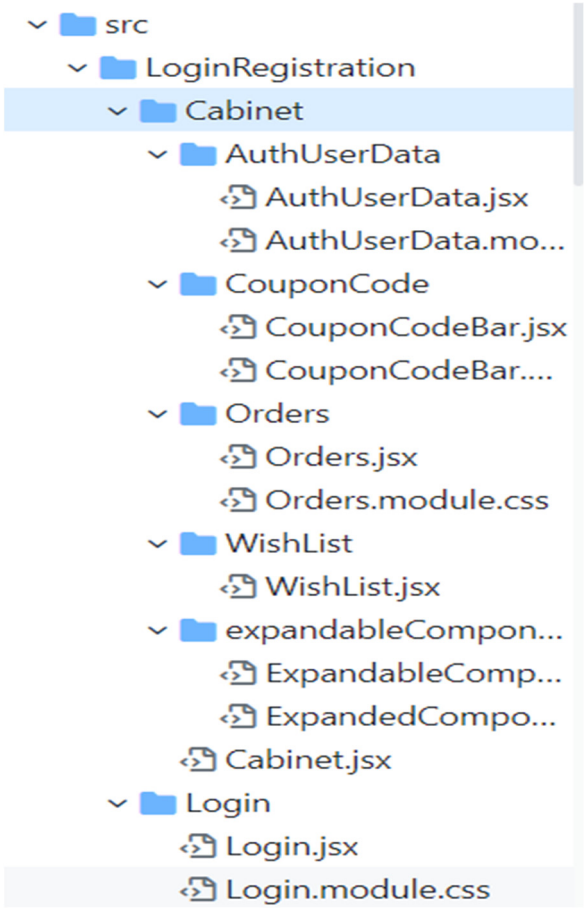

Рисунок 3.2 –Архітектурний шаблон веб-додатку

React - це популярна інтерфейсна JavaScript-бібліотека, яка широко використовується для створення веб-додатків. Однією з ключових концепцій React є використання компонентів - самодостатніх, багаторазових частин коду, які можна легко комбінувати для побудови складних користувацьких інтерфейсів. [48]

Компоненти - це потужна функція React, яка дозволяє розробникам створювати модульний, багаторазовий код, який можна використовувати в різних частинах додатку. Компоненти можуть бути простими, наприклад, кнопка або поле введення, або більш складними, наприклад, модальне вікно або форма. Розбиваючи додаток на менші багаторазові компоненти, розробники можуть створювати більш зручний, масштабований та ефективний код.

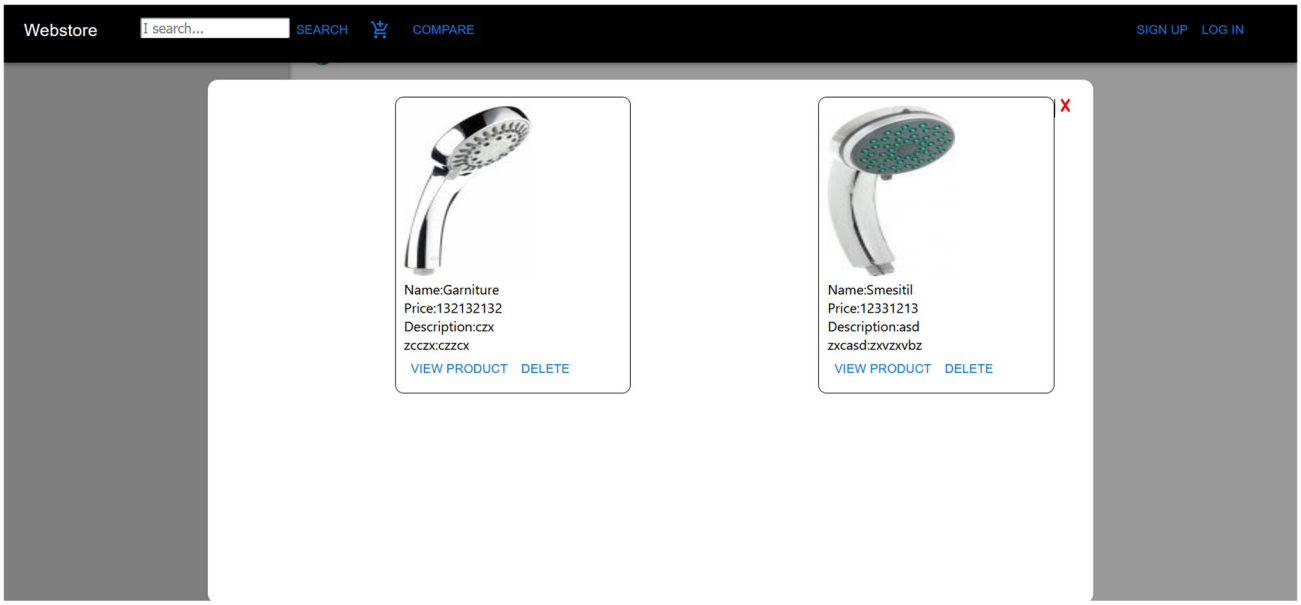

Рисунок 3.3 –Спливаюче вікно розробленого веб-сайту

Компоненти в React мають ієрархічну структуру, де компонент верхнього рівня містить підкомпоненти, які можна повторно використовувати та комбінувати для створення більш складних компонентів. Така ієрархія полегшує створення та керування компонентами, а також допомагає тримати код організованим та читабельним. [49]

Однією з переваг використання компонентів у React є те, що їх можна легко поширювати та повторно використовувати в різних додатках або частинах додатку. Це може заощадити багато часу і зусиль, оскільки розробники можуть уникнути дублювання коду і зосередитися на створенні нової функціональності.

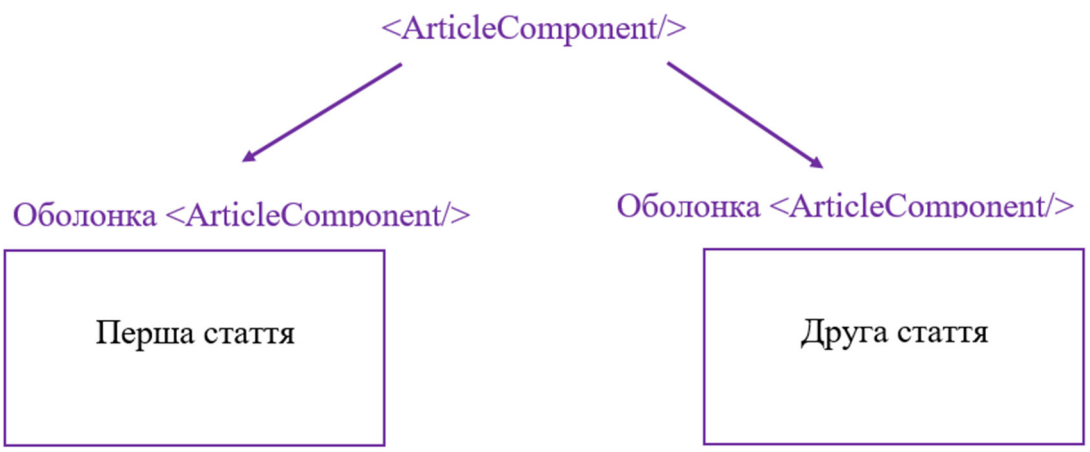

Рисунок 3.4 – Графічне відображення повторного використання компонентів

У React "state" - це вбудована функція, яка дозволяє компонентам керувати та зберігати дані, які можна оновлювати та маніпулювати ними з часом. По суті, стан представляє поточний стан компонента і може бути використаний для відображення динамічного контенту та реагування на дії користувача.

State є важливою концепцією в React, оскільки дозволяє розробникам створювати інтерактивні та динамічні користувацькі інтерфейси. Наприклад, компонент, що представляє форму, може мати об'єкт стану, який зберігає значення, введені користувачем, і оновлює інтерфейс в реальному часі, коли користувач вводить дані [50]. Іншим прикладом може бути компонент, що представляє кошик для покупок, який відстежує товари, додані користувачем, і динамічно оновлює загальну вартість, коли користувач додає або видаляє товари.

Важливо зазначити, що стан є специфічним для компонента і не передається між компонентами, якщо тільки не передається через "пропси". Це означає, що зміни, внесені до стану одного компонента, не впливають на стан іншого, забезпечуючи ізоляцію та модульність.

Використання стану в React дозволяє розробникам з легкістю створювати динамічні, адаптивні та інтерактивні користувацькі інтерфейси. Визначаючи та оновлюючи стан компонента, розробники можуть створювати компоненти інтерфейсу, які реагують на дії користувача та забезпечують безперебійний користувацький досвід.

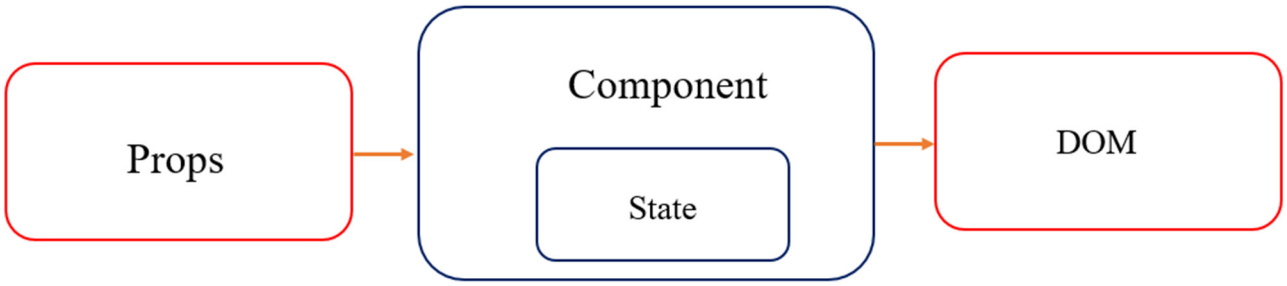

Рисунок 3.5 –Схема надходження даних до компонентів

Redux - це популярна бібліотека управління станами, що використовується в React-додатках, особливо у великих проектах, таких як веб-магазини. Вона допомагає керувати станом додатку у передбачуваний та організований спосіб, а також спрощує процес управління даними між різними компонентами.

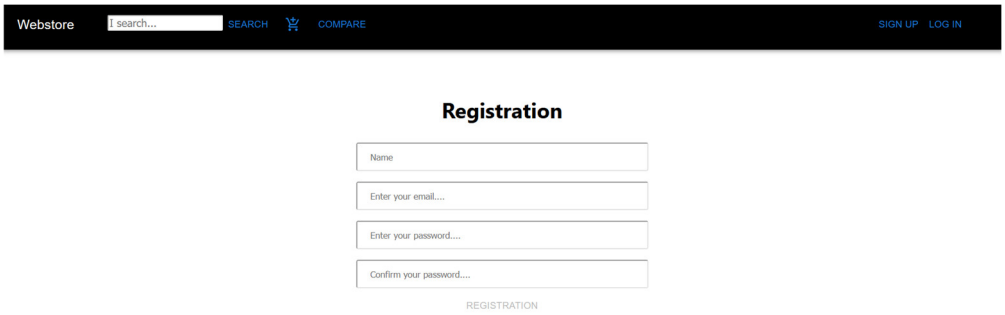

Рисунок 3.6 –Сторінка реєстрації користувача у розробленому веб-сайті

Однією з ключових переваг використання Redux у веб-магазині є те, що він може допомогти підвищити продуктивність додатку. Зберігаючи дані в центральному місці, додаток може уникнути непотрібних повторних

рендерингів і підвищити загальну швидкість і ефективність роботи програми. [52]

Перевагою використання Redux у веб-магазині є те, що він спрощує процес налагодження та тестування. За допомогою Redux розробники можуть легко відстежувати потік даних по всьому додатку і виявляти потенційні проблеми, що полегшує пошук і усунення несправностей.

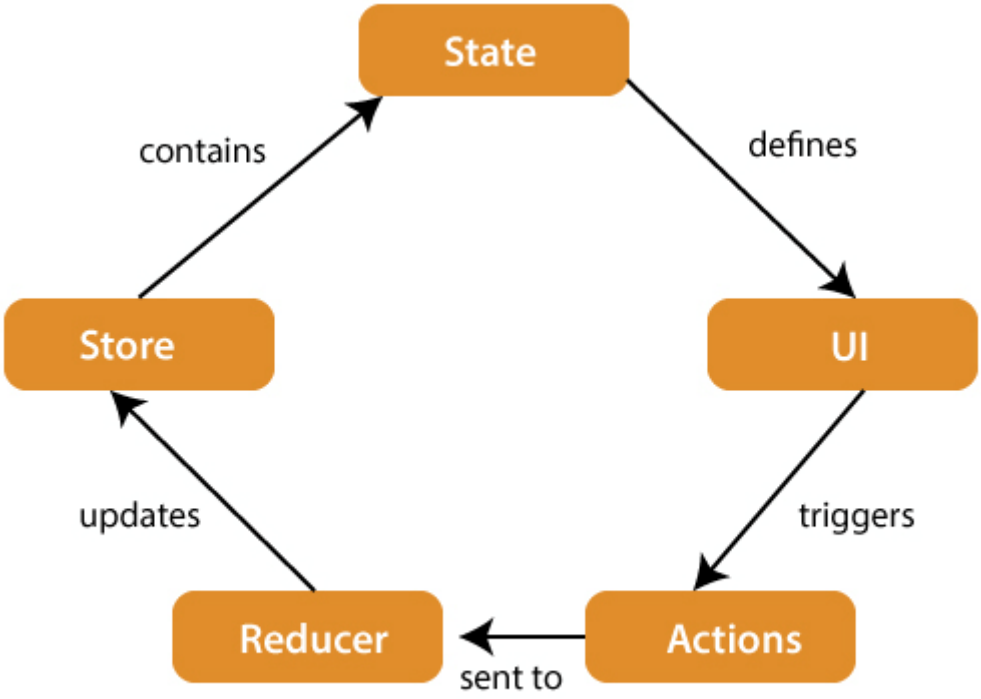

Рисунок 3.7 –Поток даних в комбінації використання React/Redux

У Redux редуктори - це функції, які визначають, як має оновлюватися стан додатку у відповідь на різні дії. Коли виконується дія, вона передається до редуктора разом з поточним станом програми, а редуктор повертає новий стан, який відображає зміни, спричинені дією.

Редуктори є важливим інструментом для налагодження та тестування додатків у Redux. Вивчаючи вихідні дані функції-редуктора, розробники можуть легко побачити, як змінився стан програми з часом, що полегшує виявлення та виправлення проблем.

```
let reducers = combineReducers({
    Header: HeaderReducer,
    Users: UserReducer,
    Items:itemsReducer
});
export let store = legacy_createStore(reducers);
```

Рисунок 3.8 – Структура Store в веб-додатку

У веб-магазині на основі Redux дані зазвичай зберігаються в центральному контейнері станів, відомому як сховище. Коли виконується дія, сховище передає цю дію і поточний стан додатку відповідному редуктору, який повертає новий стан, що відображає зміни, викликані дією.

У веб-магазині на основі Redux дані надходять зі сховища до редукторів, а потім до HOCs і компонентів через компонент Provider. Звідти дані зберігаються в локальному стані компонента, де їх можна використовувати для управління специфічними для компонента даними і змінами стану [53]. Використовуючи цю архітектуру, розробники можуть створювати високомасштабовані, ефективні та модульні веб-магазини, які легко підтримувати та оновлювати з часом.

Центральний репозиторій у веб-магазині на основі Redux діє як єдине джерело істини для даних додатку. Він забезпечує узгодженість і дозволяє декільком компонентам отримувати доступ до даних і змінювати їх у передбачуваний спосіб.

Коли в додатку виконується дія, вона запускає відповідну функцію редуктора, яка оновлює стан у незмінний спосіб. Ця незмінність гарантує збереження попереднього стану, що полегшує відстеження змін і реалізацію таких функцій, як налагодження в часі.

Управління потоком даних і змінами стану, Redux також надає функціональність проміжного програмного забезпечення, що дозволяє розробникам перехоплювати і змінювати дії до того, як вони досягнуть редукторів, забезпечуючи розширені можливості, такі як ведення журналів, асинхронні операції та інтеграції з API.

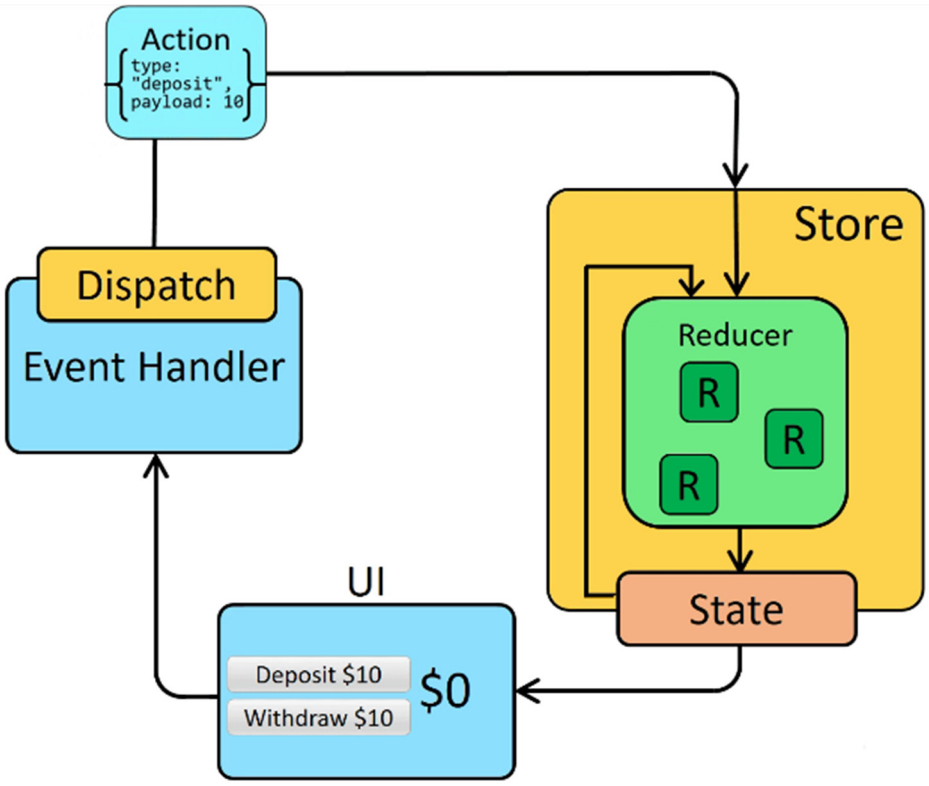

Рисунок 3.9 –Архітектура роботи Redux

Використання редукторів та менеджерів станів, таких як Redux в React, є надзвичайно важливим для розробки масштабованих та підтримуваних додатків. Ці інструменти допомагають централізувати та організувати стан додатку, полегшуючи його керування та налагодження. Вони також полегшують потік даних між компонентами, дозволяючи створити більш впорядковану та ефективну архітектуру додатку [54]. Використовуючи редуктори та менеджери станів, розробники можуть покращити загальну продуктивність та надійність своїх React-додатків.

HOC можуть отримувати оновлений стан як пропси і передавати його своїм обгорнутим компонентам, дозволяючи їм отримувати доступ до відповідних даних і запускати повторний рендеринг за необхідності. Це забезпечує більш ефективний процес рендерингу, оскільки компоненти оновлюються лише тоді, коли змінюються їхні специфічні дані.

3.2 Розміщення компонент на веб-сайті

У контексті розробки інтерфейсу на React, розміщення та розмір заголовків, нижнього колонтитула, навігаційної панелі та основних компонентів контенту є важливими факторами для створення адаптивного та зручного для користувача веб-сайту. [55]

React - це популярний фреймворк для розробки інтерфейсів, який дозволяє розробникам створювати багаторазові компоненти, які можна легко інтегрувати в різні частини веб-сайту. З React важливо дотримуватися найкращих практик розміщення та розмірів компонентів, щоб забезпечити цілісний та послідовний користувацький досвід.

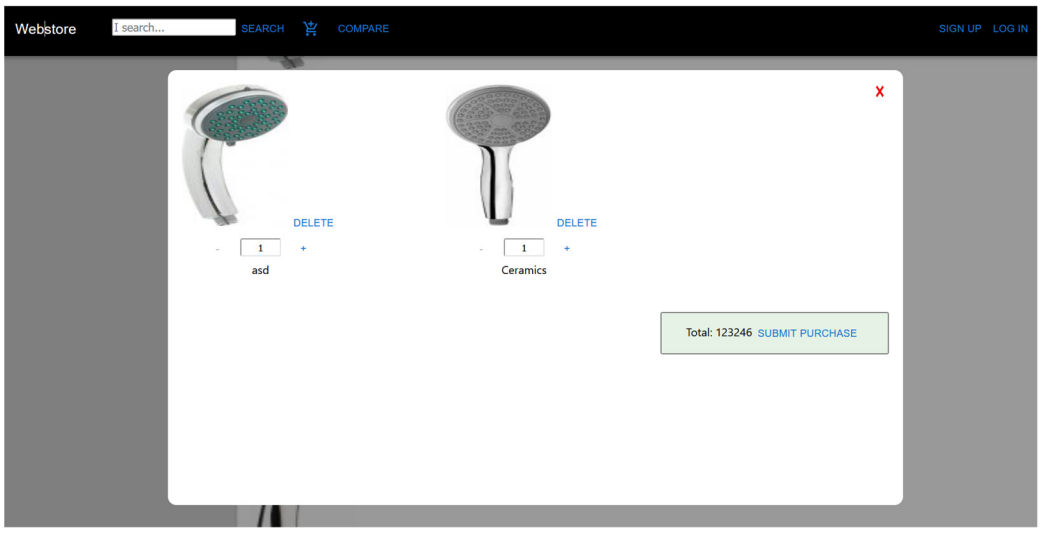

Рисунок 3.10 –Спливаюче вікно кошика покупок розробленного веб-сайту

React-розробники повинні розглянути можливість розміщення його у верхній частині сторінки, оскільки саме там користувачі очікують його знайти. Крім того, хедер повинен бути адаптивним, а його розмір і макет повинні підлаштовуватися під різні розміри екранів і типи пристроїв.

Нижній колонтитул слід розміщувати внизу сторінки, а його розмір слід ретельно продумати, щоб він не заважав основній частині контенту [56]. Панель навігації повинна бути розміщена на видному місці і спроектована так, щоб нею

було легко користуватися, з чіткими і лаконічними позначками, які направляють користувачів до потрібної їм інформації.

В основній частині контенту має бути достатньо місця для відображення основного контенту веб-сайту. Розмір і розташування основної частини контенту залежать від конкретних потреб веб-сайту, але важливо, щоб вона була легкодоступною і зручною для користувачів на всіх пристроях.

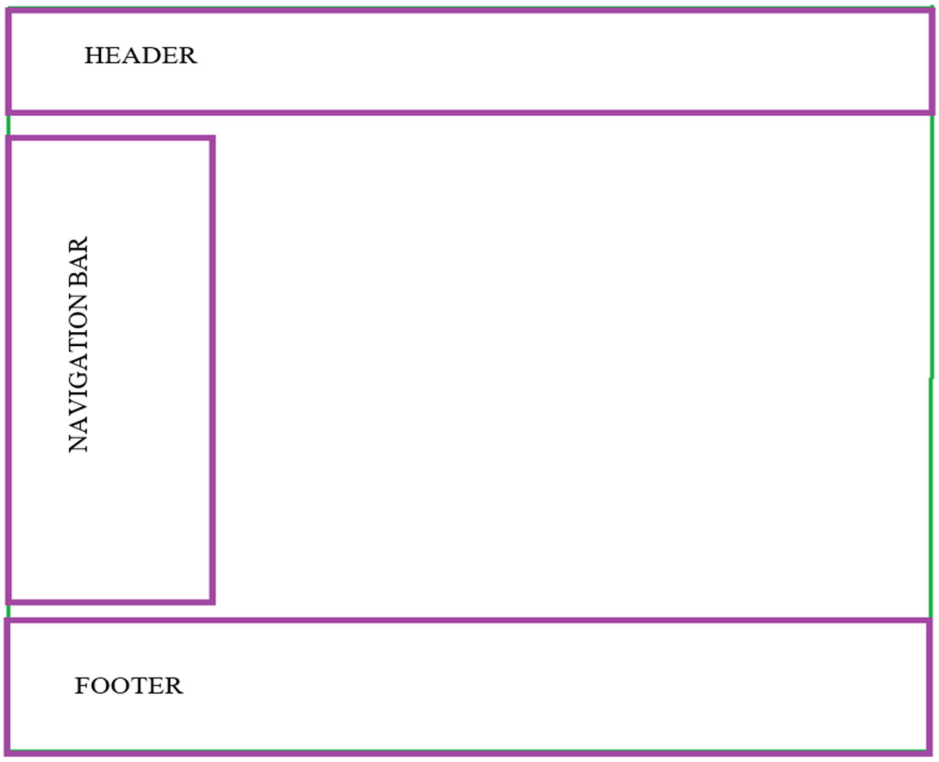

Рисунок 3.11 –Макет частин розміщення веб-додатку

Сіткова верстка - це потужний і гнучкий інструмент, який можна використовувати для створення правильного макета веб-сторінки. За допомогою CSS Grid розробники можуть легко створювати складні та адаптивні макети, які підлаштовуються під різні розміри екранів і типи пристроїв.

Використовуючи сітковий макет, розробники можуть визначати рядки і стовпці, а також вказувати, як різні елементи повинні бути розміщені в цих рядках і стовпцях. Це дозволяє точно контролювати макет веб-сторінки і гарантує, що дизайн буде послідовним і візуально привабливим.

Однією з найбільших переваг використання сітки є те, що вона дозволяє розробникам створювати адаптивні дизайни, які адаптуються до різних розмірів екрану [57]. Визначаючи різні шаблони сітки для різних розмірів екранів, розробники можуть гарантувати, що їхні макети чудово виглядатимуть на будь-яких пристроях - від невеликих мобільних до великих десктопних екранів.

Перевага використання сітки полягає в тому, що вона дозволяє легко створювати складні та асиметричні макети. Завдяки можливості визначати різні розміри рядків і стовпців, а також можливості перекривати елементи, розробники можуть створювати унікальні та візуально цікаві дизайни, яких було б складно досягти за допомогою інших методів верстки.

Використання сіткової верстки - це ефективний спосіб досягти правильного розташування елементів на веб-сторінці. Використовуючи потужні інструменти та гнучкість CSS Grid, розробники можуть створювати візуально привабливі, адаптивні та динамічні макети, які покращують взаємодію з користувачем і допомагають виділити веб-сайт з натовпу.

```
.App {
    display: grid;
    grid-template-areas:
    "header header"
    "content content "
    "footer footer";
    grid-template-rows: 70px 1fr 60px;
    grid-template-columns: 22% 1fr;
    height: 100vh;
}
.header{
    grid-area:header;
    position: sticky;
}
.content{ grid-area:content;}
.Footer{ grid-area:footer;}
```

Рисунок 3.12 –Код файлу CSS розміщення частин веб-додатку

Об'єктна модель документа, або скорочено DOM, - це програмний інтерфейс для веб-документів. Вона представляє сторінку таким чином, що програми можуть змінювати структуру, стиль і зміст документа.

Кожен HTML-елемент на веб-сторінці представлений у вигляді вузла в дереві DOM. Ці вузли з'єднані один з одним в ієрархічній структурі, де верхнім вузлом є вузол документа. Вузол document представляє весь HTML-документ, а всі інші вузли є нащадками цього вузла.

Кожен вузол у DOM-дереві має набір властивостей, які описують його атрибути, такі як ім'я тегу, клас, ідентифікатор і стиль [58]. Ці властивості можна отримати і змінити за допомогою JavaScript, що дозволяє розробникам динамічно маніпулювати вмістом і стилем веб-сторінок.

У DOM-дереві є кілька типів вузлів, зокрема вузли елементів, текстові вузли, вузли коментарів і вузли атрибутів. Вузли елементів представляють елементи HTML, такі як , <p>, <img> і <a>. Текстові вузли представляють текстовий вміст всередині HTML-елемента, а вузли коментарів - коментарі, які додаються до вихідного коду HTML. Вузли атрибутів представляють атрибути HTML-елемента, наприклад, атрибут href елемента <a>.

Кожен вузол елемента в DOM-дереві може мати дочірні вузли, які можуть бути іншими вузлами елементів, текстовими вузлами або вузлами коментарів. Дочірні вузли розташовуються у відношенні батько-дочірній, де батьківський вузол - це вузол, який містить дочірній вузол. Наприклад, елемент <body> є батьківським вузлом для всіх інших елементів на веб-сторінці, оскільки він містить всі інші елементи. [59]

DOM (Document Object Model) - це деревоподібна структура, яка представляє елементи HTML на веб-сторінці. DOM оновлюється щоразу, коли змінюється стан веб-сторінки, наприклад, коли користувач взаємодіє зі сторінкою або коли дані отримуються з API. Однак оновлення DOM може бути повільним процесом, особливо для великих і складних веб-додатків.

Віртуальний DOM у React - це абстрактний шар між реальним DOM та кодом розробника. Щоразу, коли відбувається зміна стану React-компонента, React створює нове віртуальне DOM-дерево, яке є полегшеною копією реального DOM. Потім React порівнює нове віртуальне DOM-дерево з попереднім і обчислює різницю між ними. Цей процес називається "узгодженням".

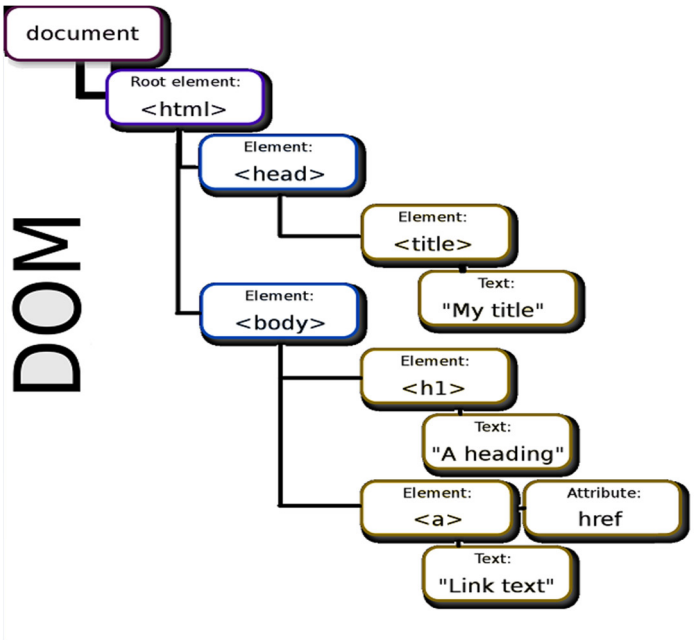

Рисунок 3.13 –Структура DOM дерева

React обчислив різницю між старим і новим віртуальним DOM-деревом, він оновлює лише необхідні частини фактичного DOM, а не оновлює весь DOM. Такий підхід набагато швидший, ніж оновлення всього DOM, оскільки вносяться лише необхідні зміни.

Віртуальний DOM також забезпечує більш ефективний спосіб оновлення стану веб-додатку. Коли користувач взаємодіє з веб-сторінкою, React оновлює віртуальне дерево DOM, а не реальний DOM [60]. Це дозволяє React виконувати багаторазові оновлення стану веб-сторінки без необхідності кожного разу оновлювати реальний DOM.

Модель об'єктів документа (DOM) - це ієрархічна структура, яка представляє елементи веб-сторінки у вигляді вузлів. Кожен вузол у дереві DOM має набір властивостей, які описують його атрибути, такі як ім'я тегу, клас, ідентифікатор і стиль. Ці властивості можна переглядати і змінювати за допомогою JavaScript, що дозволяє розробникам динамічно маніпулювати вмістом і стилем веб-сторінок.

Властивість імені тегу DOM-вузла описує HTML-тег, який було використано для створення вузла. Наприклад, елемент матиме властивість

імені тегу "div". Доступ до властивості name тегу можна отримати за допомогою властивості nodeName в JavaScript.

```
let element = document.getElementById("my-element");
console.log(element.nodeName); // outputs "DIV"
```

Рисунок 3.14 – Доступ до властивості name тегу

Властивість class DOM-вузла описує клас або класи CSS, які застосовуються до вузла. До одного вузла можна застосувати декілька класів, розділяючи їх пробілами. Доступ до властивості class можна отримати і змінити за допомогою властивості className в JavaScript.

```
let element = document.getElementById("my-element");
element.className = "new-class";
```

Рисунок 3.15 – Модифікація властивості класу

Властивість ID вузла DOM описує унікальний ідентифікатор, який присвоюється вузлу. Ідентифікатори часто використовуються для націлювання на певні елементи за допомогою JavaScript або CSS. Доступ до властивості ID можна отримати та змінити за допомогою властивості id у JavaScript.

```
let element = document.getElementById("my-element");
element.className = "new-class";
```

Рисунок 3.16 – Доступ до властивості ID

Властивість style вузла DOM описує вбудований стиль елемента. Вбудовані стилі - це CSS-стилі, які застосовуються безпосередньо до елемента за допомогою атрибута style. Доступ до властивості style можна отримати і змінити за допомогою властивості style в JavaScript. [61]

На додаток до цих властивостей, DOM-вузли також мають ряд інших властивостей, які описують їх атрибути, такі як атрибут href елемента <a>, атрибут src елемента <img> і атрибут value елемента <input>. Ці властивості

можна отримати і змінити за допомогою JavaScript так само, як і властивості, описані вище.

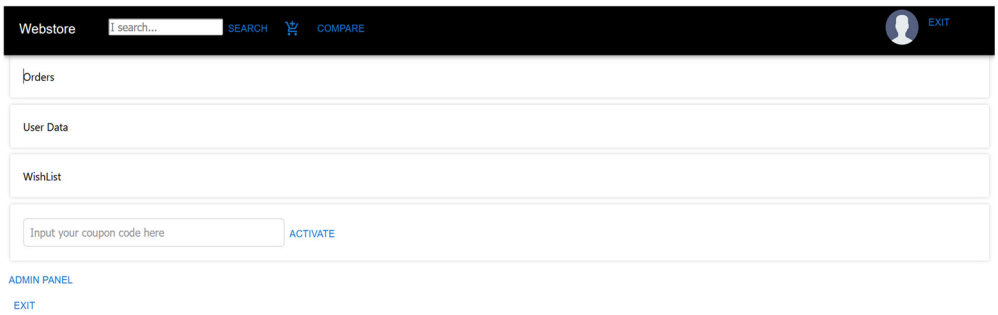

Рисунок 3.17 –Сторінка користувача веб-сайту

Керування властивостями DOM-вузлів за допомогою JavaScript є потужним інструментом для веб-розробників, оскільки дозволяє їм створювати динамічні та інтерактивні веб-сторінки. Наприклад, розробники можуть використовувати JavaScript для зміни текстового вмісту елемента, зміни класу або ідентифікатора елемента або модифікації вбудованого стилю елемента. Розуміючи властивості DOM-вузлів і те, як маніпулювати ними за допомогою JavaScript, розробники можуть створювати більш динамічні та цікаві для користувачів веб-сторінки.

Вузли - це елементи, які мають спільний батьківський вузол і розташовані на одному рівні в DOM-дереві. Це вузли, які прилягають один до одного і не вкладені один в одного. Доступ до дочірніх вузлів і маніпулювання ними можна здійснювати за допомогою JavaScript, що дозволяє розробникам динамічно змінювати вміст і структуру веб-сторінки.

Модель об'єктів документа (DOM) відіграє ключову роль у сучасній веб-розробці, забезпечуючи структуроване представлення вмісту та макета веб-сторінки, до якого можна отримати доступ і маніпулювати ним програмно за допомогою JavaScript. DOM дозволяє розробникам динамічно змінювати вміст і стиль веб-сторінок у режимі реального часу, що дає змогу створювати цікаві, інтерактивні та адаптивні веб-додатки. [63]

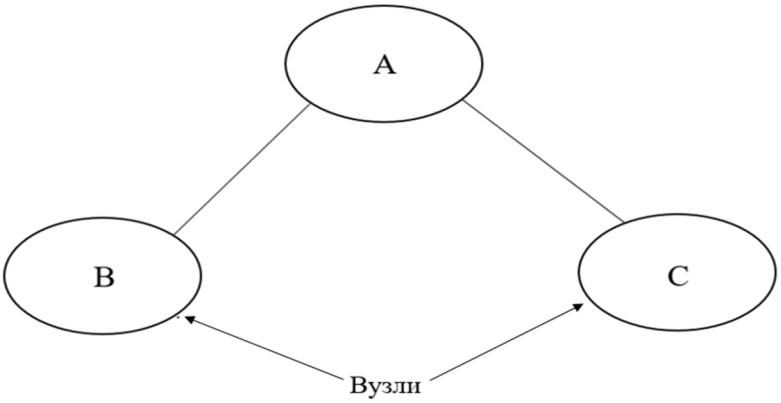

Рисунок 3.18 – Графічне представдення схеми вузлів

Розуміючи структуру елементів DOM, веб-розробники можуть отримувати доступ до властивостей і атрибутів окремих елементів і маніпулювати ними, а також переміщатися по ієрархічній структурі дерева DOM для маніпулювання більш складними структурами сторінок. Ці знання необхідні для створення динамічних та інтерактивних веб-сторінок, які можуть реагувати на дії користувача, оновлюватися в режимі реального часу і забезпечувати більш цікавий та захоплюючий користувацький досвід.

Крім того, використовуючи можливості DOM, розробники можуть створювати кастомні веб-додатки, пристосовані до потреб своїх користувачів, які можуть надавати персоналізований досвід і функціональність. Незалежно від того, чи створюється простий веб-сайт, чи складний веб-додаток, розуміння DOM є критично важливим для веб-розробників, які хочуть створювати сучасні, динамічні та інтерактивні веб-додатки.

3.3 Тестування та порівняльний аналіз

Під час розробки інтернет-магазину на основі стеку MERN тестування та порівняльний аналіз є важливими етапами, які гарантують, що додаток функціонує належним чином і відповідає вимогам проекту.

Тестування включає в себе запуск набору автоматизованих і ручних тестів, щоб переконатися, що функції і функціональність інтернет-магазину працюють так, як задумано. Автоматизовані тести можуть включати модульні, інтеграційні та наскрізні тести. Ручне тестування передбачає, що людина переміщується по веб-магазину і взаємодіє з його функціями, щоб переконатися, що вони працюють коректно.

Порівняльний аналіз передбачає оцінку продуктивності інтернет-магазину в порівнянні з конкурентами на ринку. Це може включати порівняння цін, функцій, користувацького досвіду та загального дизайну. Порівняльний аналіз може допомогти визначити сфери, в яких інтернет-магазин може покращити свої показники і диференціювати себе від конкурентів.

Тестування є невід'ємною частиною будь-якого процесу розробки програмного забезпечення, і це справедливо і для веб-розробки з використанням стеку MERN. При розробці інтернет-магазину з використанням стеку MERN тестування стає ще більш важливим, оскільки воно включає в себе кілька взаємопов'язаних технологій, які працюють разом, щоб забезпечити бездоганний користувацький досвід.

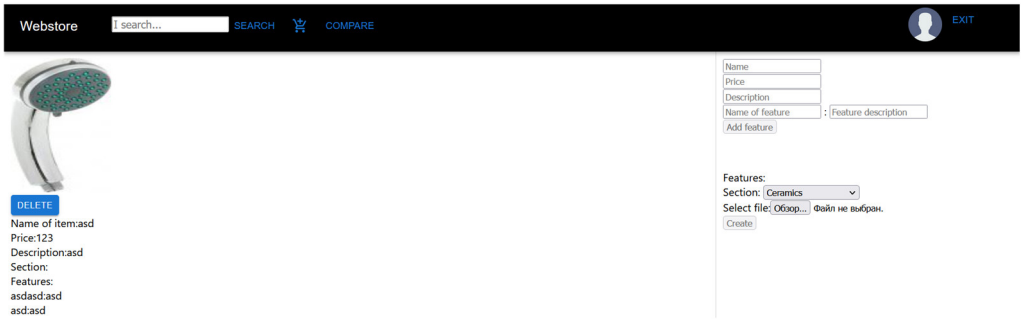

Рисунок 3.19 - Сторінка адміністратора для додавання, редагування та видалення товарів

Тестування допомагає виявити баги та помилки на ранній стадії процесу розробки, що може заощадити багато часу та зусиль у довгостроковій перспективі. Тестуючи різні компоненти інтернет-магазину, розробники можуть

переконатися, що додаток функціонує належним чином і відповідає вимогам бізнесу та його клієнтів.

Однією з важливих переваг тестування в MERN-стеку веб-розробки є те, що воно надає можливість порівняти продуктивність різних технологій і фреймворків. Наприклад, порівнюючи продуктивність різних пакетів Node.js або фронтенд-фреймворків, таких як React, розробники можуть приймати обґрунтовані рішення про те, які технології використовувати, виходячи з конкретних потреб проекту.

Рисунок 3.20 – Етапи розробки веб-додатку

Перевагою тестування є те, що воно допомагає забезпечити безпеку інтернет-магазину. Зі збільшенням кількості кібератак, спрямованих на веб-додатки, стало вкрай важливо тестувати безпеку веб-додатків, щоб виявити потенційні вразливості та виправити їх до того, як вони можуть бути використані.

Тестування є критично важливою частиною веб-розробки з використанням стеку MERN, оскільки воно допомагає забезпечити якість, надійність і безпеку інтернет-магазину [64]. Тестуючи різні компоненти програми, розробники можуть виявляти і виправляти помилки на ранній стадії розробки, що призводить до більш плавного процесу розробки і більш високої якості кінцевого продукту.

Юніт-тести мають важливе значення при розробці стека MERN, оскільки вони допомагають розробникам виявляти помилки на ранніх стадіях процесу розробки. Це полегшує виправлення проблем у міру їх виникнення і допомагає забезпечити стабільність і надійність кодової бази. Інтеграційні тести також мають вирішальне значення, оскільки вони перевіряють взаємодію між різними частинами кодової бази, що може бути складно протестувати лише за допомогою

модульних тестів. Наскрізні тести є важливими, оскільки вони перевіряють всю систему, що є необхідною умовою для забезпечення функціонування інтернет-магазину за призначенням.

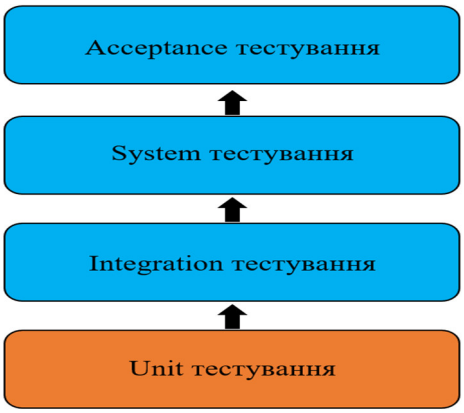

Рисунок 3.21 – Етапи тестування веб-додатку

Використовуючи різні типи тестів у веб-розробці за допомогою стеку MERN, розробники можуть бути більш впевненими в якості та надійності свого інтернет-магазину. Тестування допомагає виявити баги та помилки на ранній стадії розробки, що знижує ризик дорогих помилок і гарантує стабільну та надійну роботу інтернет-магазину для кінцевих користувачів.

Один із способів порівняти швидкість роботи декількох веб-сайтів - це провести серію тестів і порівнянь. Для цього можна використовувати такі інструменти, як PageSpeed Insights від Google або GTmetrix, щоб виміряти час завантаження і показники продуктивності. Порівнюючи ці показники на різних веб-сайтах, можна отримати уявлення про те, які сайти є найшвидшими та найефективнішими.

Інший підхід полягає у проведенні користувацького тестування і спостереженні за тим, як швидко користувачі можуть виконувати завдання на кожному веб-сайті. Для цього можна попросити користувачів виконати певні дії, наприклад, здійснити покупку або заповнити форму, і виміряти, скільки часу вони витрачають на виконання кожного завдання. Порівнюючи результати на

різних веб-сайтах, можна визначити, які з них є найбільш зручними та ефективними.

Багато факторів можуть впливати на швидкість та ефективність веб-сайту. Від дизайну і верстки до кодування і хостингу - існує незліченна кількість змінних, які можуть впливати на швидкість завантаження і реакції сайту. Однак, застосовуючи систематичний підхід до порівняння декількох веб-сайтів, заснований на даних, можна визначити, які з них є найбільш ефективними та зручними для користувача, і використовувати цю інформацію для майбутніх удосконалень та оптимізацій.

Рисунок 3.22 – Доступ до властивості name тегу

Існує кілька способів перевірити швидкість і продуктивність веб-сайтів, які можуть допомогти власникам і розробникам веб-сайтів визначити області для поліпшення і забезпечити оптимальну роботу сайту. Ось кілька методів:

Використання інструментів для тестування швидкості сайту: Існує кілька безкоштовних інструментів для тестування швидкості сайту, таких як PageSpeed Insights від Google, GTmetrix і Pingdom. Ці інструменти аналізують швидкість завантаження сайту, визначають причини будь-яких уповільнень і надають пропозиції щодо покращення.

Впровадження користувацького тестування: Спостереження за користувачами під час навігації та використання веб-сайту може дати цінну інформацію про те, наскільки швидко і легко вони можуть виконувати завдання. Проведення користувацького тестування з різноманітною групою користувачів

може допомогти виявити будь-які проблеми з продуктивністю або юзабіліті сайту, які можуть впливати на продуктивність.

Перевірка коду і дизайну сайту: Код і дизайн веб-сайту можуть впливати на його швидкість і продуктивність. Оптимізація коду та дизайну для швидкості та ефективності може допомогти покращити продуктивність веб-сайту. Це може включати стиснення зображень, зменшення кількості HTTP-запитів і мінімізацію коду та розміру файлів.

Щоб порівняти швидкість і продуктивність веб-сайту, який був розробилений, з іншими веб-сайтами, можна скористатися безкоштовними інструментами тестування швидкості веб-сайтів, такими як PageSpeed Insights від Google, GTmetrix і Pingdom. Ці інструменти надають об'єктивні вимірювання часу завантаження і показників продуктивності, а також пропонують пропозиції щодо поліпшення. Використовуючи ці інструменти, ви можете визначити області для оптимізації та забезпечити швидку та ефективну роботу веб-сайту.

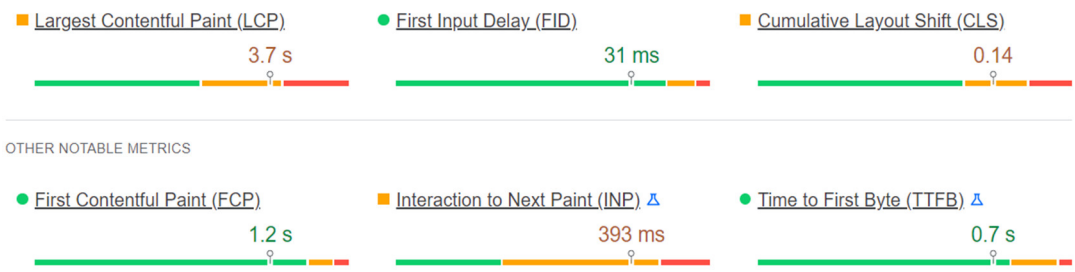

Рисунок 3.23 – Результати оцінки аналога до створеного веб-сайту «Rozetka»

Аналіз даних аналітики веб-сайту: Аналіз даних аналітики веб-сайту, таких як показник відмов, тривалість сеансу та кількість переглядів сторінок, може дати уявлення про те, як користувачі взаємодіють з веб-сайтом.

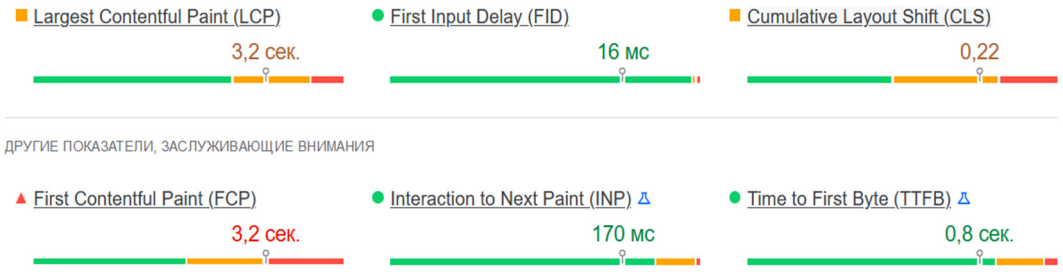

Рисунок 3.24 – Результати оцінки аналога до створеного веб-сайту «Lateram»

Для покращення FCP, власники веб-сайтів можуть оптимізувати час відгуку сервера, мінімізувати кількість HTML, CSS і JavaScript, необхідних для відображення сторінки, а також оптимізувати зображення та інші медіафайли.

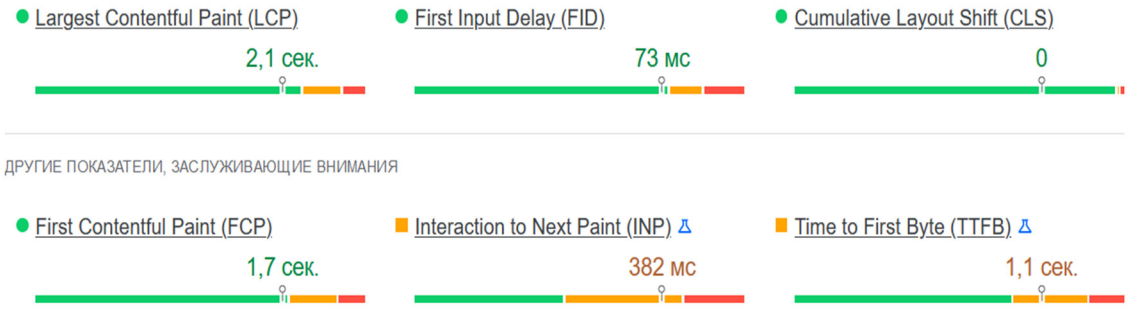

Рисунок 3.25 – Результати оцінки розробленого веб додатку

Largest Contentful Paint (LCP) вимірює час, необхідний для відображення найбільшого елемента контенту.

FCP вимірюється в секундах і зазвичай відображається як час від моменту, коли користувач ініціює запит, до моменту, коли з'являється перший фрагмент контенту. Хороший час FCP зазвичай становить менше 1 секунди.

Interaction to Next Paint (INP) вимірює час, необхідний веб-сторінці для реагування на дії користувача, такі як натискання кнопки або переходу за посиланням, і для відображення нового візуального стану на сторінці.

INP є важливою метрикою, оскільки вона вимірює, наскільки швидко веб-сайт реагує на введення даних користувачем і показує результат, що може

суттєво вплинути на користувацький досвід. Швидший час INP може покращити користувацький досвід і підвищити рівень залученості та конверсії.

Час до першого байта (TTFB) вимірює час, необхідний для того, щоб сервер почав надсилати дані у відповідь на запит користувача.

TTFB вимірюється в секундах і зазвичай відображається як час від моменту, коли користувач ініціює запит, до моменту, коли сервер починає надсилати перший байт даних. Хороший час TTFB зазвичай не перевищує 200 мілісекунд.

Щоб покращити TTFB, можна оптимізувати конфігурацію сервера, мінімізувати кількість запитів, необхідних для відображення сторінки, і використовувати мережі доставки контенту (CDN), щоб зменшити відстань між сервером і користувачем. Покращуючи TTFB, власники веб-сайтів можуть поліпшити користувацький досвід і підвищити рівень залученості та конверсії.

Назва застосунку	Rozetka	Веб-сайт стек MERN	Lateram
LCP	2.1s	3.7s	3.2s
FID	73ms	31ms	16ms
CLS	0	0.14	0.22
FCP	1.7s	1.2s	3.2s
INP	382ms	393ms	170ms
TTFB	1.1s	0.7s	0.8s

Таблиця 3.1 – Порівняльний аналіз ефективності розробленого веб-додатку з сайтами аналогами

Розробка веб-магазину на стеку MERN включає кілька етапів, від проектування та створення користувацького інтерфейсу до реалізації складної

бізнес-логіки та обробки даних. Протягом усього цього процесу тестування та порівняльний аналіз відіграють вирішальну роль у забезпеченні відповідності кінцевого продукту високим стандартам сучасної веб-розробки.

Тестування - це важливий етап процесу розробки, який передбачає перевірку функціональності, надійності та продуктивності інтернет-магазину. Проводячи комплексне тестування, розробники можуть виявити потенційні проблеми і внести поліпшення для оптимізації продуктивності, швидкості і загального користувацького досвіду інтернет-магазину. Тестування може включати в себе такі методи, як модульне тестування, інтеграційне тестування, тестування продуктивності та тестування сприйняття користувачами, серед інших.

Тестування і порівняльний аналіз є критично важливими елементами при розробці інтернет-магазину на стеку MERN. Ці методи допомагають гарантувати, що кінцевий продукт буде якісним, оптимізованим для продуктивності та здатним забезпечити винятковий користувацький досвід. Використовуючи ці методи, розробники можуть створювати успішні інтернет-магазини, які сприяють залученню користувачів, збільшенню доходів і зростанню бізнесу.

Поєднання тестування та бенчмаркінгу при розробці інтернет-магазину на основі стеку MERN дозволяє розробникам створювати відшліфований та надійний продукт, який відповідає очікуванням сучасних веб-користувачів.

Ретельне тестування дозволяє розробникам виявити та виправити будь-які функціональні проблеми або проблеми з продуктивністю на ранній стадії циклу розробки, запобігаючи тому, щоб потенційні проблеми дійшли до кінцевих користувачів і негативно вплинули на їхній досвід.

4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

Розрахунок вартості розробки програмного забезпечення є життєво важливим аспектом у визначенні фінансової життєздатності проекту. Для отримання оцінки трудовитрат необхідна ретельна оцінка факторів, пов'язаних з процесом розробки, включаючи аналіз завдання, дослідження алгоритмів, програмування, налагодження програми та підготовку документації. Для оцінки економічної доцільності розробки були розраховані витрати, пов'язані з налагодженням програми на персональному комп'ютері, та визначені витрати на підготовку документації. За допомогою відповідних формул були розраховані різні аспекти, пов'язані з собівартістю розробки, що дозволило оцінити очікувані прибутки від розробки. Таким чином, було визначено термін окупності розробки та загальну економічну доцільність проекту.

4.1 Розрахунок витрат на розробку програмного забезпечення

Складність процесу розробки визначається обраною темою розробки, що передбачає оцінку декількох факторів: різні етапи, необхідні для створення веб-ресурсу, тип завдань, а також рівень зусиль, необхідних протягом усього процесу розробки. Перш ніж приступити до роботи, необхідно провести попередню підготовку, створивши технічний план.

Трудомісткість розробки ПЗ розраховують за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_d$$

де t_o - витрати праці на підготовку й опис поставленої задачі;

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ПК;

t_d - витрати праці на підготовку документації.

t_o=500

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється. Умовне число операторів (підпрограм) розраховують за формулою:

$$Q = q * C * (1 + p)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

Передбачуване число операторів для задач планування 3200

Тоді q =3200

Фактор складності програми визначається з урахуванням перетину груп складності та ступеня новизни.

В нашому випадку група складності враховуючи комплексність розробки складає:

C=1.15

Коефіцієнт кореляції програми в ході її розробки- збільшення обсягу робіт з допомогою внесення змін - у алгоритм чи програму з результатам уточнення постановок і описів її, зміни складу і структури інформації, і навіть уточнень. Для даної програми він дорівнює 0

p =0

Q=3500*1.15*1=4025

Q=4025

Витрати праці на вивчення опису задачі визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{(75..85) * k}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

Обирається в інтервалі від 1,2 до 1,5

В нашому випадку 1,3

B=1.3

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Значення коефіцієнт кваліфікації k для програміста до 2 років стажу становить 0.8

K=0.8

t_u =4025 / (20 × 0,8) = 251

Витрати праці на розробку алгоритму рішення задачі розраховують за формулою:

$$t_a = \frac{Q}{(20 \ldots 25) * k}$$

t_a = 4025 / (20 × 0,8) = 251

t_a=251

Витрати на складання програми по готовій блок-схемі розраховують за формулою:

$$t_n = \frac{Q}{(20..25) * k}$$

t_n = 4025 / (20 × 0,8) = 251

Витрати праці на налагодження програми на ПК розраховують:

- за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5) * k}$$

$t_{\text{отл}}$= 4025 / (5 × 0,8)= 1006

Витрати праці на підготовку документації розраховують за формулою:

$$t_d = t_{dp} + t_{do}$$

t_d = 280+ 210= 490

t_d = 490

де t_{dp} - трудомісткість підготовки матеріалів і рукопису, що розраховують за формулою:

$$t_{dp} = \frac{Q}{(15..20) * k}$$

t_{dp} = 4025 / (18× 0,8)= 280

t_{do} - трудомісткість редагування, друкування й оформлення документації:

$$t_{do} = 0.75 * t_{dp}$$

t_{do} = 0,75 × 280= 210

$$t = t_o + t_u + t_a + t_n + t_{\text{отл}} + t_d$$

t_o	t_u	t_a	t_n	$t_{\text{отл}}$	t_d	Загалом
50	87	251	251	1006	490	2135

4.2 Розрахунок можливої ціни поректу

Розрахунок витрат на оплату праці розробників Розрахуємо середньогодинну оплату програміста. Для цього необхідно спочатку визначити його річний фонд грошового забезпечення. Це можна зробити, знаючи місячне грошове забезпечення програміста. Воно складає приблизно 15000,00 гривень. Таким чином, річний фонд грошового забезпечення 180000 гривень.
Кількість робочих годин у році розраховуємо за формулою:
Далі розрахлвуємо кількість робочих годин на рік. Формула для цього розрахунку:

$$N_p = (N - N_n - N_в) * 8$$

де N - загальна кількість днів у році,

N_n – кількість святкових днів у році,

$N_в$- кількість вихідних днів у році.

Приймається, що кількість святкових днів у році – 14, а вихідних – 104.

N_p=(365-14-104)*8=1976

Середню погодинну заробітну плату програміста можна визначити за наступною формулою:

$$C_n = \frac{Ф_р}{N_p}$$

Ф – річний фонд грошового забезпечення.

C_n=234180000 / 1976 = 91

Витрати на оплату праці розробників програми складають:

$$\text{В}_{\text{оп}} = \text{С}_n * T_з$$

де $T_з$ - загальна кількість годин роботи програміста над проектом.

На розробку проекту витрачено 330 годин

$T_з$=1976/12*2=330

$\text{В}_{\text{оп}}$ =91,1*330=30063

$\text{В}_{\text{оп}}$ =30063

Витрати, пов'язані з розробкою програми на ПК розраховуються:

$$\text{В}_{\text{пк}} = \text{Т}_{\text{пк}} * t_c$$

$\text{В}_{\text{пк}}$=330*0.09=30

де $\text{Т}_{\text{пк}}$ - час використання ПК для розробки програми,

t_c- собівартість машинного часу обчислювальної техніки (розраховує бухгалтерія підприємства).

Собівартість однієї години роботи ПК дорівнює:

$$C_{\text{пк}} = \frac{\text{В}_\text{е}}{\Ф_{\text{пк}}}$$

де $\text{В}_\text{е}$ - річні поточні витрати на експлуатацію ПК,

$\text{В}_\text{е}$=1600

$\Ф_{\text{пк}}$- річний фонд часу корисної роботи ПК.

Щоб розрахувати річний фонд робочого часу ПК, ми можемо визначити фактичний річний час використання ПК в годинах. Це дозволить нам оцінити вартість машинного часу за годину.

Формулу для фактичного річного фонду часу роботи ПК можна переписати як:

$$\Phi_d = N_p - (\Phi_м + \Phi_{річ})$$

де $\Phi_м$ - місячний фонд часу на профілактику і ремонт ПК (час профілактики щомісячно – 5 годин),

Φ_d=1976-(5+144)= 1797 годин

$С_{пк}$=1600/1797=0.89

Річні експлуатаційні витрати на програмне забезпечення можна визначити за допомогою наступної формули:

$$В_{пк} = В_{Е_р} + В_{А_р} + В_{РЕМ_р} + В_{ДК_р} + В_{I_р}$$

де $В_{А_р}$ – річні відрахування на амортизацію,

$В_{Е_р}$ – річні витрати на електроенергію для ПК,

$В_{РЕМ_р}$ – річні витрати на ремонт ПК,

$В_{ДК_р}$ –річні витрати на додаткові комплектуючі ПК,

$В_{I_р}$ –Інші витрати.

Сума річної амортизації може бути визначена за допомогою наступної формули:

$$В_{А_р} = Ц_{ПК} * Н_А$$

де $Ц_{ПК}$ – балансова вартість ПК,

$Н_А$ – норма амортизаційних відрахувань (дорівнює 15% у квартал).

Часу витрачено на розробку 4 місяця тоді 15%*4/3=20%

Балансову вартість ПК розраховуємо за формулою:

$$Ц_{ПК} = Ц_р * (1 + К_{УН})$$

$Ц_{ПК}$=17000*(1+0.12)= 19 040

де $Ц_р$ – ринкова вартість ПК,

$Ц_р$=17000

$К_{УН}$– коефіцієнт, що враховує витрати на установку й налагодження ПК

$В_{А_р}$=19 040*10%=1904

Вартість електроенергії, яку споживає комп'ютер, можна розрахувати за наступною формулою:

$$В_{Е_р} = Р_{ПК} * Ф_{ПК} * Ц_Е * К_{ІВ}$$

де $Р_{ПК}$ – паспотна потужність ПК,

Р =0,06 кВт/год

$Ф_{ПК}$ – річний фонд корисного часу роботи ПК,

$Ф_{ПК}$ =1823

$Ц_Е$ – вартість 1 кВт/год електроенергії,

$Ц_Е$ =1,68

$К_{ІВ}$ – коефіцієнт інтенсивності використання ПК (0,7 - 1).

$К_{ІВ}$ = 0,7

$В_{Е_р}$=0,06кВт/год*1797*1,68*0.7=127

Таким чином, розрахункове значення витрат на електроенергію, що споживає ПК, складає:

- витрати на поточний і профілактичний ремонт (приймаються рівними 6% від вартості ПК):

$$В_{РЕМ_Р} = Ц_{ПК} * 0.06$$

$В_{РЕМ_Р}$=17000*0.06=1020

Витрати на додаткові компоненти відносяться до витрат, необхідних для забезпечення належного функціонування комп'ютера, і приймаються на рівні 2% від вартості комп'ютера.

$$В_{ДК_Р} = Ц_{ПК} * 0,02$$

$В_{ДК_Р}$=17000*0.02=340

- інші витрати, тобто непрямі витрати пов'язані з експлуатацією ПК (приймаються рівними 5-10% від вартості ПК):

$$В_{I_р} = Ц_{ПК} * 0,05$$

$В_{I_р}$=17000*0.05=850

$$В_{пк} = В_{E_р} + В_{A_р} + В_{РЕМ_Р} + В_{ДК_р} + В_{I_р}$$

$В_{пк}$=127+1904+1020+340+850=7953

$В_{пк}$=4241

У процесі розробки програмних систем комп'ютер використовується на різних етапах програмування, які включають в себе:

– Написання програми відповідно до попередньо створеної схеми алгоритму: ПК використовується для кодування програми на основі заздалегідь визначеного алгоритму або блок-схеми

– Налагодження програми на ПК: Комп'ютер використовується для виявлення та вирішення помилок і проблем у програмі, забезпечуючи її безперебійне і точне виконання.

– Підготовка документації до завдання: ПК використовується для створення та форматування документації, пов'язаної із завданням, включаючи проектну документацію, посібники користувача та технічні специфікації.

Таким чином, витрати машинного часу склали ($t_{маш}$):

t_o	t_u	t_a	t_n	$t_{отл}$	t_d	Загалом
50	87	251	251	1006	490	2135

$$t_{маш} = t_n + t_{отл}^k + t_d$$

$t_{маш}$=251+1006+490=1747

Вартість оплати машинного часу можна розрахувати за наступною формулою:

$$В_{маш} = t_{маш} * С_{ПК}$$

$В_{маш}$= 1747× 0.89= 1555

Загальні витрати на розробку програмного комплексу складають:

$$В_{заг} = В_{оп} + В_{маш}$$

$В_{заг}$=30063+1555=79786

4.3 Розрахунок показників економічної ефективності

За міжнародним стандартам для оцінки ефективності розробки ПЗ астосовують такі показники:

- внутрішня норма дохідності;
- чистий приведений дохід;
- рентабельність;
- термін окупності.

Показник внутрішньої дохідності характеризує величину чистого прибутку (чистого валового доходу), що припадає на одиницю інвестиційних вкладень у кожному часовому інтервалі життєвого циклу проекту. Розрахунок цього показника виконується за такою формулою:

$$\sum_{\mathrm{i=0}}^{\mathrm{T}} \frac{Di}{(1+q)^i} - \sum_{i=0}^{T} \frac{Ki}{(1+q)^i} = 0$$

де D - дохід (прибуток) у і-му періоді;

Розрахунок прибутку

Середня ціна (1 елементу продукції)	Чистий прибуток (1 елементу продукції)	Середня кількість збиту продукції	Місячний прибуток (без сплати податків)	Місячний прибуток (після сплати податків)
400 грн	150 грн	700 шт/місяць	105 000 грн	83 475 грн

Податкове законодавство України передбачає єдину стандартну ставку. Платник податку на прибуток на загальних підставах сплачує 20.5%.

Місячний прибуток після сплати налогів

105 000 -20.5%= 83 475 грн

де K- інвестиційні вкладення в і-му періоді з урахуванням інфляційних процесів;

i - періоди виконання і впровадження проекту;

T - загальний період (тривалість) життєвого циклу проекту;

q - показник внутрішньої норми дохідності.

Показник інвестиційних вкладень з урахуванням інфляційних процесів обчислюємо за формулою:

$$K_i = \varphi_i * R_i$$

де φ - коефіцієнт інфляції на поточний період;

φ=1.35

R- інвестиційні платежі в і-му періоді (капітальні вкладення).

R=31 618

Ki = 1,35*31 618=42 685

Дохід від розробки ПЗ у і-му періоді розраховуємо за формулою:

$$D_i = J_i(B_i - C_i)$$

де B - ціна продажу програмного продукту в і-му періоді;

B_{12}=1 001 700

C - собівартість програмного продукту (фактично дорівнює сумі витрат на розробку ПЗ);

C=224 500

J - кількість ПЗ.

J=1

D_{12}=1(1 001 700-224 500) = 777200

У ринкових умовах при ціновій політиці, що змінюється, показник терміну окупності є одним з головних для підприємств. Він визначається на основі величини капітальних витрат по періодах розробки програмного продукту та величини фактичних чи прогнозних доходів:

$$\sum_{i=0}^{T} Ki = \sum_{i=0}^{T} Di$$

E_{12} = 777200/ 224500= 3.46

Тоді термін окупності можна розрахувати за такою формулою:

$$E = \frac{Di}{\text{В}_{\text{заг}}}$$

$$\text{E}_{12} = 777200/\ 224500 = 3.46$$

Тоді термін окупності можна розрахувати за такою формулою:

$$\text{T} = \frac{1}{\text{E}}$$

T = 1 / 3.46= 0.29

0.29*12=3.48

Термін окупності складає 3 місяця 15 днів.

Враховуючи основні економічні показники, що стосуються розробки програмного продукту, можна зробити висновок, щодо доцільності запропонованої розробки. Якщо отримано суттєвий економічний ефект від розробки програмного продукту, а термін окупності капітальних вкладень не більший 10 років, то така розробка є економічно вигідною та конкурентоздатною на ринку подібних ІТ продуктів.Проведено аналіз техніко-економічного обґрунтування розробки програмного забезпечення. Зважаючи на термін окупності ,який складає Тр= 3.48 проект є доцільним до виконання.

ВИСНОВКИ

1.У даному дослідженні був розглянутий аналіз веб-ресурсів реалізації інтернет-магазину, а також розроблена програмна реалізації багатофункціонального динамічного веб-сайту на основі технології React.

2.У першому розділі був проведений аналіз інструментів розробки динамічних веб-сайтів, порівняння з аналогічними сайтами та аналіз елементів веб-сайту. Цей аналіз дав змогу визначити найкращі практики та функціональні вимоги для розробки інтернет-магазину.

3.У другому розділі були розглянуті алгоритми роботи веб-сайту, зокрема алгоритм роботи веб-сайту та алгоритм пошуку товарів. Також була визначена структура бази даних, яка дозволить зберігати та організовувати інформацію про товари та користувачів.

4.У третьому розділі була проведена програмна реалізація веб-сайту інтернет-магазину з використанням технології React. Була розроблена структура веб-сайту з використанням патернів та архітектурних принципів. Також була реалізована клієнтська частина, яка забезпечує зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

5.Під час розробки було проведене тестування та порівняльний аналіз розробленого веб-сайту. Це дозволило виявити та виправити можливі проблеми та недоліки в роботі веб-сайту, а також зробити порівняльний аналіз з іншими схожими веб-сайтами.

6.В результаті даного дослідження був розроблений багатофункціональний динамічний веб-сайт інтернет-магазину, який забезпечує зручну та ефективну платформу для покупців. Розроблений веб-сайт відповідає сучасним стандартам та надає користувачам зручність та надійність у використанні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flanagan, D. (2011). JavaScript: The Definitive Guide. O'Reilly Media.
2. Duckett, J. (2014). HTML & CSS: Design and Build Websites. John Wiley & Sons.
3. Gauch, S., Gauch, S., & Gauch, S. (2015). Web Design: A Complete Introduction. John Wiley & Sons.
4. Freeman, E., & Robson, E. (2014). Head First HTML and CSS. O'Reilly Media.
5. Powell, R., & Hockenberry, M. (2012). iOS Programming: The Big Nerd Ranch Guide. Big Nerd Ranch.
6. Meyer, E. A. (2018). CSS: The Definitive Guide. O'Reilly Media.
7. Holmes, J. (2014). CSS Cookbook. O'Reilly Media.
8. Pehlivanoglu, Y. (2019). Mastering Modern Web Development: Building Modern Web Applications with React, Angular, and Vue.js. Packt Publishing.
9. W3Schools. (2023). HTML Tutorial. Retrieved from https://www.w3schools.com/html/
10. W3Schools. (2023). CSS Tutorial. Retrieved from https://www.w3schools.com/css/
11. W3Schools. (2023). JavaScript Tutorial. Retrieved from https://www.w3schools.com/js/
12. Duckett, J. (2011). JavaScript and jQuery: Interactive Front-End Web Development. John Wiley & Sons.
13. Pilgrim, M. (2011). HTML5: Up and Running. O'Reilly Media.
14. Duckett, J. (2016). JavaScript and JQuery: Interactive Front-End Web Development. John Wiley & Sons.
15. Gackenheimer, A. (2015). Full Stack JavaScript Development with MEAN. Packt Publishing.
16. Powell, R., & Griesemer, G. (2012). Learning Node.js: A Hands-On Guide to Building Web Applications in JavaScript. Addison-Wesley Professional.

17. Wilson, E., & Johnson, A. (2014). Building Node Applications with MongoDB and Backbone. O'Reilly Media.
18. Kho, R. (2015). Mastering React. Packt Publishing.
19. Banks, J., & Porcello, E. (2017). Learning React: Functional Web Development with React and Redux. O'Reilly Media.
20. Freeman, A., & Robson, E. (2013). Head First JavaScript Programming. O'Reilly Media.
21. Liebman, J., & Lobanov, A. (2015). Learning Web Development with Bootstrap and AngularJS. O'Reilly Media.
22. Ducket, J. (2015). HTML and CSS: Design and Build Websites. John Wiley & Sons.
23. Spurlock, R. (2017). Mastering Bootstrap 4. Packt Publishing.
24. Duckett, J. (2015). JavaScript and jQuery: Interactive Front-End Web Development. John Wiley & Sons.
25. Zeeshan, A. (2020). Angular 10 for Enterprise-Ready Web Applications: Deliver production-ready and cloud-scale Angular web apps. Packt Publishing.
26. Goerzen, J. (2019). Foundations of Python Network Programming: The Comprehensive Guide to Building Network Applications with Python. Apress.
27. Oreilly, T. (2018). Flask Web Development with Python Tutorial Series.
28. Hunter, D., & Stienen, B. (2015). Real-World Flask Web Development: Practical Solutions for Real-world Flask Projects. Packt Publishing.
29. Grinberg, M. (2018). Flask Web Development in Python Tutorial Series.
Freeman, A., & Robson, E. (2014). Head First HTML5 Programming: Building Web Apps with JavaScript. O'Reilly Media.
30. Laing, E. (2014). Practical Web Development with Haskell. Apress.
31. Turner, B., & Berry, R. (2015). Haskell Design Patterns. Packt Publishing.
32. Mangan, J. (2014). Developing Responsive Web Applications with AJAX and jQuery. Packt Publishing.
33. Ramnath, R. (2016). Advanced R: Mastering the R programming language. Packt Publishing.

34. Vohra, D. (2015). Ruby on Rails Tutorial: Learn Web Development with Rails. Addison-Wesley Professional.
35. Hartl, M. (2012). Ruby on Rails Tutorial: Learn Web Development with Rails. Addison-Wesley Professional.
36. O'Reilly, T. (2018). Django Web Development with Python Tutorial Series.
37. Holovaty, A., & Kaplan-Moss, J. (2009). The Definitive Guide to Django: Web Development Done Right. Apress.
38. Mele, A. (2019). Angular 8: A Practical Guide for Beginners. Packt Publishing.
39. Ram, V. (2019). Angular 8: Upgrading and Mastering Angular Reactive Forms. Apress.
40. Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., & Secret, A. (1994). The World Wide Web. Communications of the ACM, 37(8), 76-82.
41. Martin, R. C. (2003). Agile Software Development: Principles, Patterns, and Practices. Prentice Hall.
42. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.
43. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.
44. Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The Unified Modeling Language User Guide (2nd ed.). Addison-Wesley Professional.
45. McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction (2nd ed.). Microsoft Press.
46. Hunt, A., & Thomas, D. (2000). The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley Professional.
47. Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. Retrieved from http://agilemanifesto.org/
48. Beck, K. (2000). Extreme Programming Explained: Embrace Change (2nd ed.). Addison-Wesley Professional.

49. Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.
50. McConnell, S. (1996). Rapid Development: Taming Wild Software Schedules. Microsoft Press.
51. Coad, P., & Yourdon, E. (1991). Object-Oriented Analysis (2nd ed.). Prentice Hall.
52. Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Addison-Wesley Professional.
53. Ambler, S. W. (2002). Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. Wiley.
54. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns CD: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.
55. Martin, R. C. (2006). Agile Principles, Patterns, and Practices in C#. Prentice Hall.
56. Freeman, E., & Freeman, E. (2004). Head First Design Patterns. O'Reilly Media.
57. Martin, R. C. (2009). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.
58. Fowler, M. (2003). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley Professional.
59. Larman, C., & Vodde, B. (2016). Large-Scale Scrum: More with LeSS. Addison-Wesley Professional.
60. Richards, M. (2002). Object-Oriented Programming with Visual Basic .NET. Apress.
61. Fowler, M. (1997). Analysis Patterns: Reusable Object Models. Addison-Wesley Professional.
62. Larman, C. (2002). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd ed.). Prentice Hall.

63. Beck, K. (2004). Test-Driven Development: By Example. Addison-Wesley Professional.
64. Cockburn, A. (2001). Writing Effective Use Cases. Addison-Wesley Professional.