МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему:

«Обгрунтування методів оптимізації клієнтської частини веб-сайту»

Виконав: студент 6 курсу
спеціальності 126 «Інформаційні
системи та технології»

Оліховська С.В.
(прізвище та ініціали)

Керівник: Желєзняк А.М.
(прізвище та ініціали)

ДУБЛЯНИ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти другий (магістерський)
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
“____” _____202 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Оліховська Софія Віталіївна
(прізвище, ім’я, по батькові)

1. Тема роботи «*Обгрунтування методів оптимізації клієнтської частини веб-сайту*»
керівник роботи *к. е н., доцент., Желєзняк А.М.*
(наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НУП №616/к-с від 12.09.2024 р.

2. Строк подання студентом роботи 2.12.2024 р.

3. Вихідні дані: *аналітичні дані роботи та характеристика об’єкту дослідження, опис бібліотек мов програмування, науково-технічна і довідкова література.*
4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз стану питання в теорії та практиці та постановка завдання
2. Обґрунтування, вибір та реалізація інструментарію вирішення задачі
3. Результати вирішення задачі
4. Охорона праці та безпека в надзвичайних ситуаціях
5. Визначення ефективності
Висновки
Бібліографічний список
5. Перелік графічного матеріалу
Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3, 5	*Желєзняк А.М., доцент кафедри інформаційних технологій*			
4	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*			

7. Дата видачі завдання *1.03.2024*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Отримання завдання. Вивчення рекомендованої літератури по темі роботи. Написання першого розділу*	*01.03.2024 – 31.05.2024*	
2	*Проектування та опис технічного завдання, обґрунтування та вибір інструментарію реалізації проекту (написання другого розділу).*	*01.06.2024 – 31.08.2024*	
3	*Програмна реалізація поставленого завдання (написання третього розділу)*	*01.09.2024 – 18.10.2024*	
4	*Написання розділу «Охорона праці та безпека у надзвичайних ситуаціях»*	*19.10.2024 – 04.11.2024*	
5	*Оцінка ефективності поставленого завдання (виконання п'ятого розділу)*	*05.11.2024 – 18.11.2024*	
6	*Завершення оформлення основної частини, написання висновків та підготовка презентаційного матеріалу*	*19.11.2024 – 02.12.2024*	
7	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*03.12.2024 – 16.12.2024*	

Студент _____ *Оліховська С. В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Желєзняк А.М.*
(підпис) (прізвище та ініціали)

УДК 004.738.1-048.34

Обґрунтування методів оптимізації клієнтської частини веб-сайту.

Оліховська С.В. Кафедра інформаційних технологій - Дубляни, Львівський НУП, 2024.

Кваліфікаційна робота: 75 сторінок текст. част., 42 рис., 23 джерела.

Наведено теоретичні основи оптимізації клієнтської частини веб-сайту. Проведено огляд прикладних рішень, включаючи технології мініфікації файлів, оптимізації графічних елементів, асинхронного завантаження скриптів, а також сучасних підходів до адаптивного дизайну.

Обґрунтовано перспективи використання інструментів Google Lighthouse та інших для вдосконалення процесу аналізу продуктивності веб-сайтів і підвищення якості їх роботи.

Запропоновано ефективні методи оптимізації клієнтської частини сайту.

Здійснено аналіз травматичних ситуацій при виконанні різних робіт у сфері використанням комп’ютерної техніки, викладено питання охорони праці.

Здійснено обґрунтування та вибір методів і технологій оптимізації клієнтської частини веб-сайту. Проаналізовано перспективи використання адаптивного дизайну та сучасних підходів до покращення швидкості завантаження сторінок. Проаналізовано та визначено показники ефективності запропонованих методів щодо вдосконалення процесу, які сприяють зменшенню часу завантаження сайту, підвищенню лояльності користувачів і якості роботи веб-ресурсу. Здійснено аналіз охорони праці та ризиків появи травматичних ситуацій у процесі роботи з комп’ютерною технікою, запропоновано рекомендації щодо їхнього уникнення.

Ключові слова: клієнтська частина, технологія, сайт, веб-розробка.

ЗМІСТ

ВСТУП

Щодня мільярди людей взаємодіють із веб-ресурсами, виконуючи широкий спектр завдань: від пошуку інформації до здійснення покупок, спілкування та розваг. У цьому контексті зручність і швидкість доступу до веб-сайтів визначають конкурентоспроможність онлайн-сервісів. Навіть незначні затримки у завантаженні сторінок можуть призводити до втрати користувачів, що особливо критично для комерційних проєктів.

Успішне функціонування веб-ресурсу залежить від багатьох чинників, одним із ключових є оптимізація клієнтської частини. Це дозволяє не лише покращити швидкість завантаження сторінок, а й забезпечити стабільну роботу сайту, незалежно від умов користувачів.

Таким чином, розробка та впровадження ефективних оптимізаційних рішень стали критично важливим завданням для веб-індустрії. Це вимагає аналізу сучасних методів, розуміння принципів роботи веб-технологій і застосування новітніх інструментів для забезпечення максимальної продуктивності, що сприяє покращенню користувацького досвіду та підвищенню конкурентоспроможності веб-ресурсів.

Метою кваліфікаційної роботи є впровадження та реалізація оптимізаційних рішень, спрямованих на підвищення ефективності клієнтської частини веб-сайту, забезпечення швидкого завантаження сторінок, поліпшення користувацького досвіду та адаптації ресурсу до різноманітних пристроїв.

Актуальність теми цієї роботи зумовлена зростаючою потребою забезпечити надійну та ефективну роботу веб-ресурсів у відповідь на збільшення кількості користувачів і їхніх вимог до швидкості та зручності взаємодії.

В ході виконання кваліфікаційної роботи були поставлені наступні завдання:

1. Провести аналіз сучасних підходів до оптимізації клієнтської частини веб-сайтів.

2. Оцінити стан і продуктивність клієнтської частини на основі практичного кейсу.

3. Визначити та впровадити методи оптимізації, які дозволять підвищити швидкодію та зручність використання сайту.

4. Оцінити вплив впроваджених оптимізацій на загальну продуктивність сайту, зручність його використання та задоволеність користувачів.

5. Розглянути питання охорони праці при виконанні робіт у сфері веб-розробки.

Загалом дипломна робота складається з п’яти розділів. В першому розділі розкриті теоретичні основи оптимізації клієнтської частини веб-сайтів, а також оглянуто сучасні тенденції в цій сфері. У другому розділі наведено обґрунтування вибору інструментів та технологій для реалізації поставленої задачі. Третій розділ присвячено впровадженню оптимізаційних заходів і оцінці їх впливу на продуктивність веб-сайту. У четвертому розділі висвітлені питання охорони праці. В п’ятому розділі здійснено оцінку ефективності проведених оптимізацій.

Наукова новизна кваліфікаційної роботи полягає у використанні комплексного підходу до оптимізації клієнтської частини веб-сайту з акцентом на інтеграцію сучасних методів і технологій для досягнення максимальної продуктивності та зручності використання веб-ресурсу.

Під час практичної підготовки та виконання кваліфікаційної роботи результати були апробовані на Міжнародній студентській науковій конференції «Студентська молодь і науковий прогрес в АПК», матеріали якої були опубліковані у вигляді тез:

1. Оліховська С. В. Методи оптимізації клієнтської частини веб-сайту. *Тези доповідей Міжнародного студентського наукового форуму 4-6 жовтня 2024 р*. С. 362

РОЗДІЛ 1.
АНАЛІЗ СТАНУ ПИТАННЯ В ТЕОРІЇ ТА ПРАКТИЦІ ТА ПОСТАНОВКА ЗАВДАННЯ

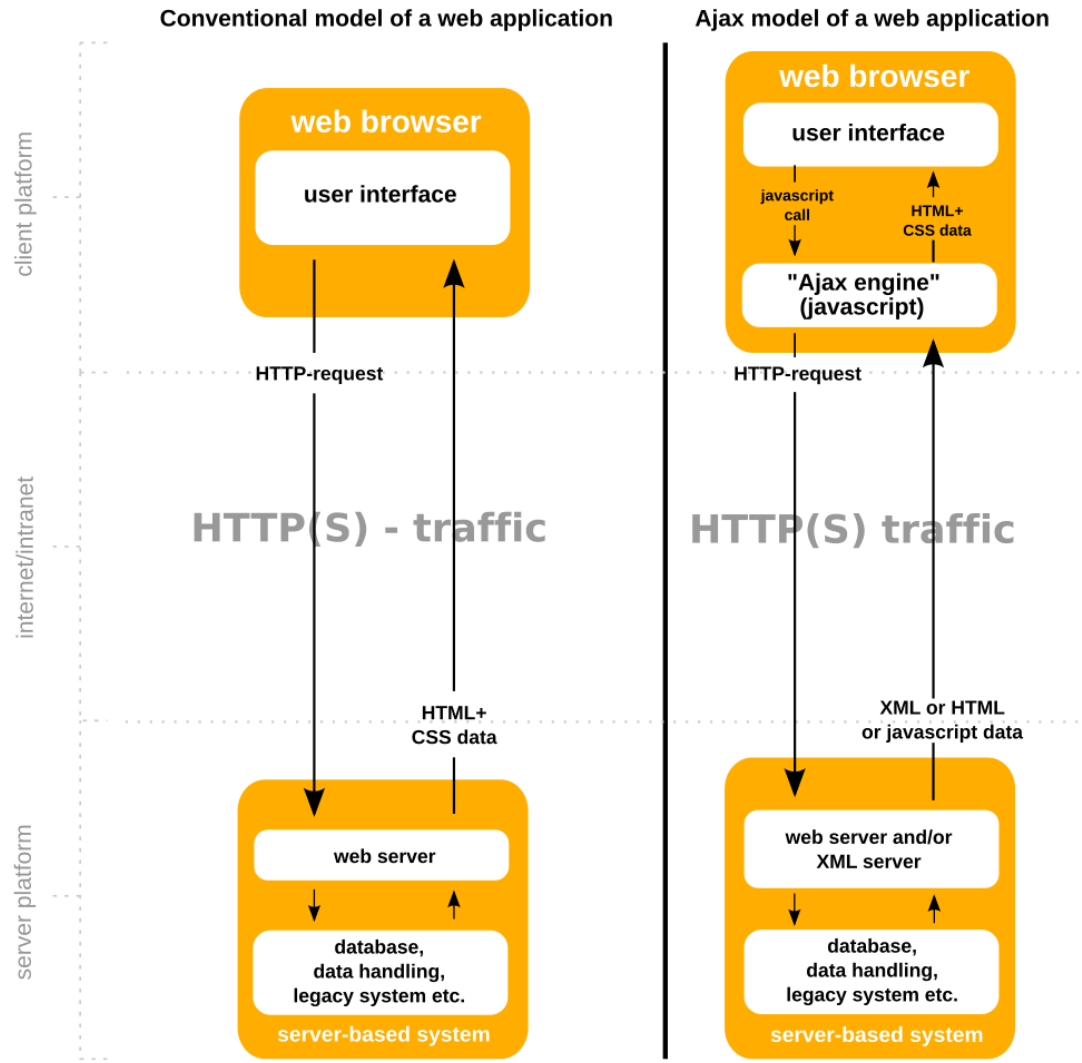
1.1 Огляд наукових досліджень у сфері розробки клієнтської частини веб-сайту

Наукові дослідження, присвячені розробці клієнтської частини веб-сайтів, розвиваються в кількох напрямках, зокрема забезпечення продуктивності, зручності використання, адаптивності та безпеки інтерфейсів. Ці аспекти є ключовими для створення веб-додатків, що відповідають сучасним вимогам користувачів. Першочергові дослідження у сфері веб-розробки зосередилися на розв’язанні проблем взаємодії між користувачем і системою, що особливо важливо для frontend-розробки.

Перші підходи до спрощення роботи з JavaScript, мовою програмування, яка є основою для клієнтської частини, запропонували такі науковці, як Джон Резіг, який створив бібліотеку jQuery, що стала надзвичайно популярною у 2006 році і значно змінила підхід до обробки подій та маніпуляції з HTML DOM. У своїх дослідженнях та розробках Резіг підкреслив важливість абстрагування низькорівневих аспектів JavaScript, що дозволило розробникам зосередитись на функціональності, а не на деталях реалізації [1]. Бібліотека jQuery забезпечила простий доступ до функцій, що раніше потребували складного кодування, і стала основою для багатьох сучасних підходів у веб-розробці.

Пізніші дослідження в області клієнтської частини веб-сайтів сконцентрувалися на питаннях оптимізації продуктивності, зокрема через удосконалення способів передачі даних між клієнтом і сервером. Дуглас Крокфорд зробив значний внесок у розвиток клієнтської архітектури через розробку формату JSON (JavaScript Object Notation), який замінив XML у багатьох застосунках як основний формат обміну даними. JSON суттєво прискорив процес взаємодії між клієнтом і сервером, що дозволило

використовувати асинхронні запити, які не переривають роботу інтерфейсу. Це стало ключовим етапом у розвитку AJAX-технологій (див. рис. 1.1), які активно досліджуються і застосовуються для підвищення зручності користувача [2].



а) б)

Рисунок 1.1 – Модель класичних застосунків для мережі (а) в прямому порівнянні із застосуванням AJAX-технологій (б). [3]

У сфері розробки односторінкових веб-додатків (SPA) суттєвий внесок зробили такі науковці та фахівці, як Джордан Валке, що працював над створенням React, та Еван Ю, засновник Vue.js. Зокрема, роботи Валке були

спрямовані на оптимізацію роботи з інтерфейсами через введення концепції "віртуального DOM", що дозволяє скоротити час оновлення елементів сторінки і підвищити продуктивність додатків.

Еван Ю, розробивши Vue, створив фреймворк, орієнтований на поступову інтеграцію, що забезпечує зручність для SPA-додатків середнього розміру та інтерактивних інтерфейсів. Vue використовує компонентну архітектуру та віртуальний DOM, як і React, але його особливістю є легкість у вивченні і простота в інтеграції з іншими проектами. Vue став особливо популярним серед розробників, які створюють легкі SPA-додатки або інтерактивні інтерфейси на існуючих платформах.

Іншою важливою сферою досліджень у розробці клієнтської частини є адаптивність інтерфейсів, що набуває особливого значення з огляду на зростання мобільного трафіку. Наприклад, дослідження Джеффрі Зельдмана в веб-дизайні, показали, що адаптивний дизайн покращує користувацький досвід і підвищує видимість веб-сайтів у пошукових системах, що є важливим аспектом SEO-оптимізації [4]. Зельдман особливо наголошував на важливості CSS-фреймворків, таких як Bootstrap, що сприяють швидкій адаптації дизайну під різні роздільності екранів за допомогою медіа-запитів та гнучких сіток.

Безпека клієнтської частини також є важливим напрямом досліджень. Наприклад, Гері Макгроу, автор книги *Software Security: Building Security In*, описує підходи до захисту від XSS (cross-site scripting) та CSRF (cross-site request forgery) атак. Макгроу акцентує на важливості застосування політик безпеки та використанні механізмів перевірки даних для захисту даних клієнтів від поширених вразливостей [5].

Оптимізація процесів рендерингу та компіляції коду також досліджувалася широко. Наприклад, Кайл Сімпсон, автор відомих книг серії *You Don't Know JS*, аналізує інструменти, такі як Webpack і Babel, для мінімізації часу завантаження веб-сторінок та підвищення ефективності виконання JavaScript-коду. Використання Webpack дозволяє групувати і

мінімізувати скрипти, а Babel забезпечує кросбраузерну підтримку сучасного коду, що покращує продуктивність веб-додатків [6].

Таким чином, наукові дослідження у сфері розробки клієнтської частини веб-сайтів охоплюють широкий спектр тем, від забезпечення зручності користувачів і адаптивності інтерфейсів до оптимізації продуктивності та безпеки. Завдяки таким дослідженням створюються нові підходи і технології, що дозволяють розробляти високоякісні веб-застосунки, які відповідають зростаючим потребам ринку і користувачів.

1.2 Теоретичні основи оптимізації клієнтської частини веб-сайту

Оптимізація клієнтської частини веб-сайту є важливою складовою створення продуктивних, зручних у користуванні та економічно ефективних веб-ресурсів. Розвиток технологій та збільшення кількості відвідувачів, які використовують різноманітні пристрої для доступу до інтернету, висувають нові вимоги до швидкодії та якості веб-сайтів. Користувачі очікують миттєвого завантаження сторінок і стабільної роботи, незалежно від їхніх технічних характеристик чи пропускної здатності мережі. Це створює підґрунтя для дослідження різних методів оптимізації, які спрямовані на покращення функціонування клієнтської частини та підвищення задоволеності користувачів.

Клієнтська частина (або frontend) веб-сайту – це та частина веб-додатка, яка безпосередньо відображається користувачеві та з якою він взаємодіє через браузер. Вона включає весь видимий інтерфейс, включаючи тексти, зображення, форми, кнопки, меню навігації тощо, а також містить логіку, яка забезпечує інтерактивність і дизайн. Сучасний frontend постійно обробляє дані, щоб створити для користувача швидкий і гнучкий інтерфейс. Але при зростанні складності сайтів клієнтська частина часто стає більш ресурсомісткою і потребує оптимізації для зменшення навантаження на сервери і полегшення доступу для відвідувачів. [7]

Оптимізація клієнтської частини веб-сайту вимагає глибокого розуміння основних принципів функціонування та взаємодії основних елементів, таких як HTML, CSS і JavaScript, які забезпечують швидкість, інтерактивність та естетичність веб-додатка. HTML – це мова розмітки, яка відповідає за структуру сторінки, CSS - мова стилів, яка визначає зовнішній вигляд сторінки, а JavaScript (мова програмування, що забезпечує динамічну взаємодію) додає інтерактивність. При завантаженні сторінки браузер спочатку отримує HTML-файл, потім завантажує і обробляє CSS і JavaScript. Далі, він формує Document Object Model (див. рис. 1.2) — дерево елементів HTML-документа, а також об'єктну модель CSS (CSSOM), де відображені всі стилі. На основі цих структур браузер об'єднує DOM і CSSOM у рендер-дерево, після чого виконує фазу рендерингу та виводить контент на екран. [8]

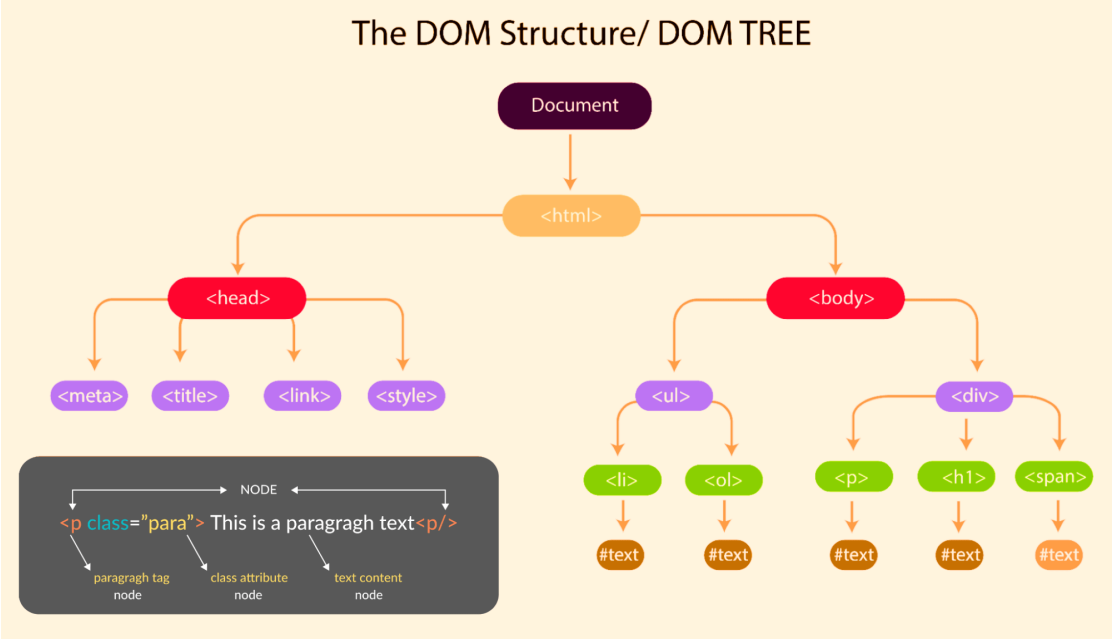


Рисунок 1.2 – Приклад DOM структури сайту. [9]

Навантаження на клієнтську частину зростає з кожним новим елементом на веб-сторінці, збільшуючи обсяг завантажуваних даних і ускладнюючи процеси рендерингу. Середній час завантаження сторінок є критичним для досвіду користувачів: дослідження показують, що кожна секунда затримки може призвести до відмови користувачів від відвідування, особливо на

мобільних пристроях. Тому, оптимізація клієнтської частини спрямована на те, щоб забезпечити миттєву реакцію системи на дії користувачів і мінімізувати ресурси, необхідні для обробки запитів.

Час до першого рендерингу (First Contentful Paint, FCP), час до повної взаємодії (Time to Interactive, TTI) і час до повного завантаження (Load Time) — це ключові показники продуктивності веб-сторінок, що відображають різні аспекти завантаження та готовності сторінки до використання. [10]

Час до першого рендерингу (First Contentful Paint, FCP) відображає момент, коли браузер уперше показує користувачеві будь-який видимий елемент сторінки, такий як текст, зображення або інший графічний елемент. Це перший значний показник у завантаженні сторінки, оскільки він сигналізує користувачеві, що сторінка почала завантажуватися. На FCP впливає низка факторів, серед яких швидкість мережі, розмір і кількість завантажуваних ресурсів, структура HTML-коду та стильових файлів CSS, а також обробка скриптів JavaScript. Чим більший обсяг HTML або чим складніша структура CSS і DOM (Document Object Model), тим більше часу потрібно браузеру для побудови та відображення перших елементів. Крім того, час FCP може зрости через великий обсяг або неналежну оптимізацію зображень і відео, що впливають на затримку рендерингу. Це особливо актуально для сторінок з великою кількістю мультимедійних елементів, таких як фотографії чи іконки, які вимагають ресурсомісткого завантаження.

Час до повної взаємодії (Time to Interactive, TTI) визначає момент, коли сторінка стає повністю інтерактивною, тобто коли користувач може без затримок взаємодіяти з усіма елементами інтерфейсу, наприклад, клікаючи на кнопки, заповнюючи форми або переміщаючись по меню. TTI є важливим показником, оскільки до цього моменту частина контенту може бути видимою, але не реагувати на дії користувача, що може призвести до незадоволення та залишення сторінки. На TTI впливають, передусім, JavaScript і загальна продуктивність скриптів, які часто блокують взаємодію до повного виконання. Великі чи некоректно завантажені файли JavaScript, які

відповідають за функціональність інтерфейсу, можуть створити так звані "блокуючі процеси", які не дозволяють користувачу скористатися елементами сторінки, поки всі необхідні скрипти не будуть виконані. Крім того, ефективність роботи браузера, зокрема його здатність до паралельної обробки завдань, також значно впливає на TTI, оскільки зменшує час очікування завершення основних процесів. Якщо браузер не встигає обробити всі JavaScript-скрипти паралельно, це призводить до подовження часу до повної взаємодії.

Час до повного завантаження (Load Time) визначає момент, коли всі ресурси на сторінці, включаючи зображення, відео, CSS, JavaScript, а також сторонні ресурси, як-от рекламні блоки чи аналітичні скрипти, повністю завантажені та готові до використання. Load Time є фінальним етапом завантаження сторінки і відображає повне завершення всіх мережевих і обчислювальних процесів. На Load Time впливають такі фактори, як розмір та кількість завантажуваних файлів, пропускна здатність мережі, продуктивність сервера, з якого завантажуються ресурси, а також використання кешування та CDN (мереж доставки контенту). Великі ресурси, що завантажуються без компресії, такі як зображення високої роздільної здатності або об’ємні JavaScript-бібліотеки, можуть значно збільшити час завантаження. Крім того, завантаження сторонніх скриптів, особливо якщо вони пов'язані з третіми сервісами (рекламні мережі, аналітика, соціальні віджети), також впливають на повне завантаження сторінки, оскільки їх завантаження контролюється зовнішніми серверами, на які власники сайту не мають прямого впливу. [11]

З точки зору зручності користування, оптимізація клієнтської частини є також значущим фактором. Інтуїтивно зрозумілий інтерфейс з швидким часом відгуку створює сприятливі умови для взаємодії, допомагаючи користувачам швидко знаходити потрібну інформацію або здійснювати необхідні дії. Затримки в завантаженні або нестабільність роботи клієнтської частини можуть призвести до негативного досвіду, що в свою чергу може зменшити кількість активних користувачів. Сучасні технології дозволяють

впроваджувати розумні стратегії оптимізації, такі як адаптивна розмітка (адаптивні макети та медіазапити в CSS), що забезпечує відображення контенту на різних пристроях, адаптуючи розмір шрифтів, зображень та інших елементів.

У підсумку, оптимізація клієнтської частини веб-сайту є важливим аспектом для створення швидких, зручних і продуктивних веб-додатків. Оптимізована клієнтська частина не тільки покращує досвід користувачів, але й дозволяє зменшити навантаження на сервери, зменшуючи обсяг запитів до них, і є одним із ключових елементів успішного впровадження сучасних веб-рішень.

1.3. Приклади реалізації клієнтських частин

При розгляді прикладів реалізації клієнтської частини веб-сайтів на основі популярних платформ можна проаналізувати ряд сайтів з інноваційним підходом до оптимізації інтерфейсу і взаємодії. Це дозволить глибше зрозуміти, як принципи ефективного фронтенду застосовуються для покращення продуктивності та зручності користування. Нижче описано конкретні приклади реалізації клієнтських частин, таких як Facebook та Amazon. Кожен із них використовує різні підходи до оптимізації для досягнення найкращих результатів.

Amazon є одним із найвідоміших і найуспішніших інтернет-магазинів у світі, і його клієнтська частина — це яскравий приклад комплексної, зручної та оптимізованої реалізації інтерфейсу, розробленої для максимального комфорту користувачів. Кожен елемент сайту Amazon продуманий так, щоб допомогти користувачеві швидко знайти потрібний товар, отримати всю необхідну інформацію про нього та оформити замовлення з мінімальними зусиллями.

На головній сторінці Amazon можна побачити багатий вміст, що включає банери, слайдери з товарами, рекомендовані категорії та різні

персоналізовані блоки. Ці елементи розташовані так, щоб відображати ключові пропозиції магазину, проте вони не відволікають від основних розділів, таких як меню пошуку та навігації, які завжди залишаються доступними у верхній частині сторінки.

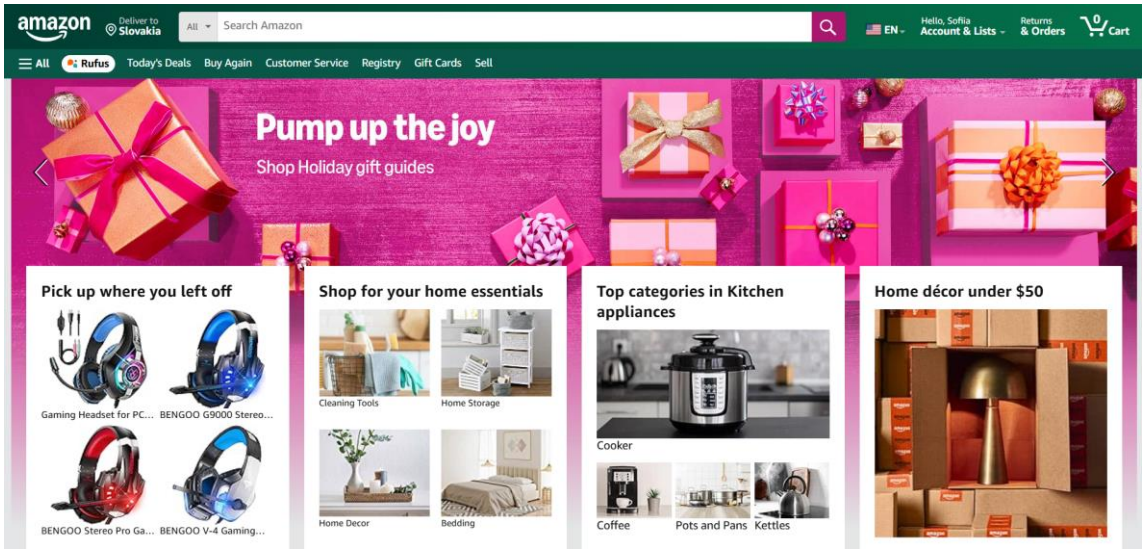


Рисунок 1.3 – Головна сторінка Amazon

У верхньому меню розміщений зручний пошуковий рядок, який відіграє центральну роль у роботі сайту, дозволяючи швидко знайти потрібний товар. Інтерфейс передбачає автозаповнення та підказки, що значно скорочує час пошуку. На цьому ж рівні розташовані елементи входу до облікового запису, кошика, списку побажань та доступу до історії замовлень. Усі ці компоненти зроблені максимально доступними для користувача, з метою швидкого переходу до часто використовуваних функцій.

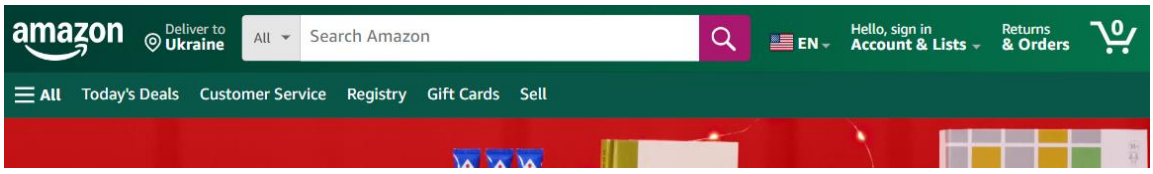


Рисунок 1.4 – Верхнє меню Amazon

Сторінка товару на Amazon побудована таким чином, щоб надати всю інформацію, необхідну для здійснення покупки, на одному екрані. Зліва розташовані зображення товару, які можна переглядати, збільшуючи кожне з них, щоб детально розглянути товар. Праворуч — ключова інформація, така як назва, ціна, доступність і кнопки для додавання товару в кошик або в список бажань. Що нижче, то більш детальна інформація доступна, зокрема опис товару, характеристики, відгуки користувачів та інформація про доставку.

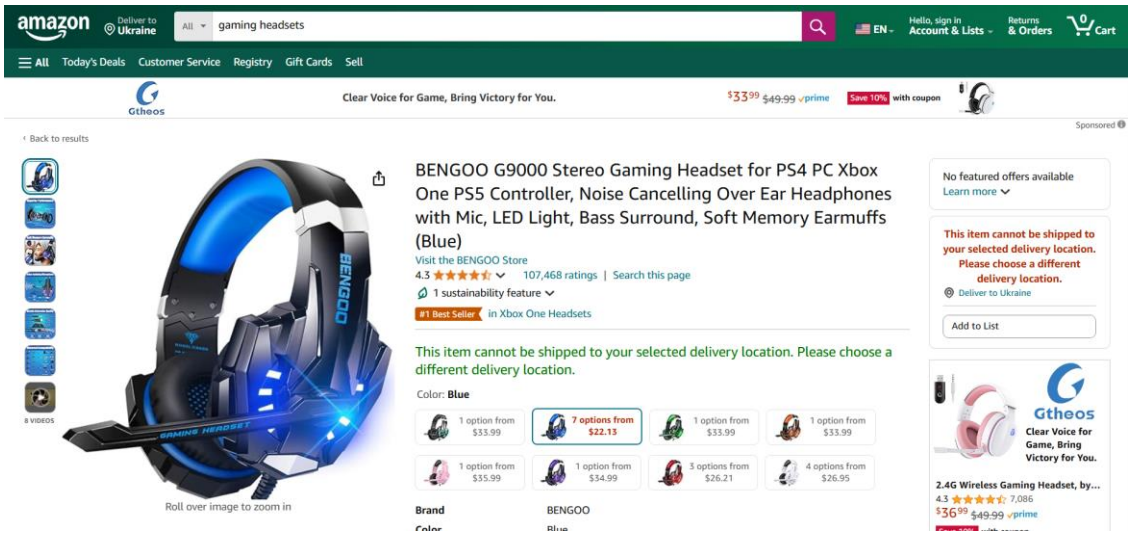


Рисунок 1.5 – Сторінка товару Amazon

Amazon активно використовує алгоритми для персоналізації рекомендацій, щоб допомогти користувачам знаходити цікаві товари. Наприклад, блоки «Customers who bought this item also bought» або «Recommended for you» з’являються на основі історії переглядів та покупок.

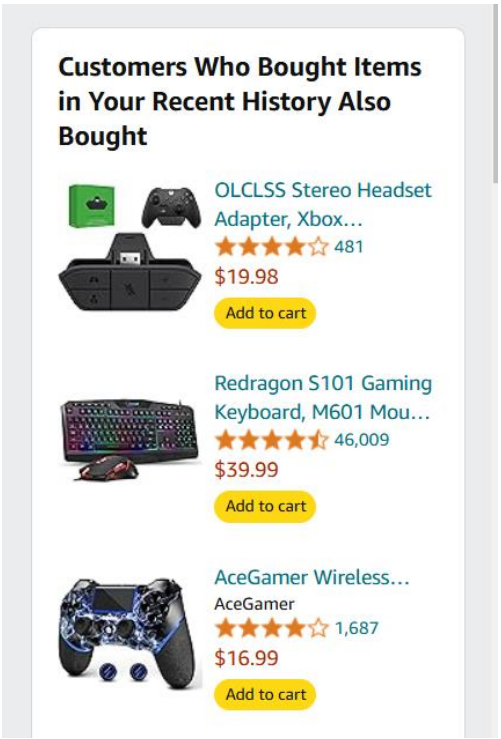


Рисунок 1.6 – Блок з персоналізованими рекомендаціями Amazon

Розділ із відгуками користувачів є ще одним важливим елементом, оскільки допомагає покупцям приймати обґрунтовані рішення. Відгуки мають сортування за релевантністю, і Amazon виділяє відгуки з найкориснішими оцінками.

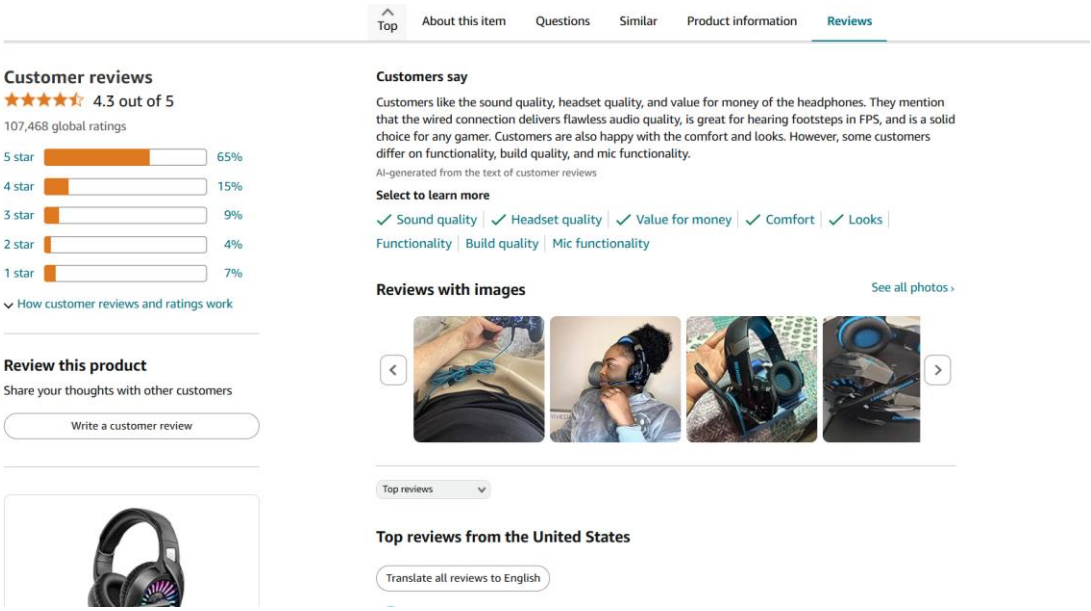


Рисунок 1.7 – Блок відгуків Amazon

Кошик на Amazon побудований так, щоб дозволити швидко переглянути всі додані товари з деталями, включаючи кількість, ціну та варіанти доставки. Процес оформлення замовлення також дуже простий і оптимізований для швидкого проходження користувачем усіх необхідних кроків. Кожен етап розділено, щоб уникнути плутанини, що є важливою частиною зручного користувацького досвіду.

Важливим елементом є швидка реакція інтерфейсу при оформленні замовлення. Клієнтська частина Amazon оптимізована так, щоб забезпечити швидкий і плавний перехід між етапами замовлення, а також мінімізувати час завантаження на кожному кроці цього процесу. Завдяки оптимізації JavaScript-коду для динамічного оновлення сторінок та використанню технології кешування, користувач відчуває плавний досвід покупки без затримок.

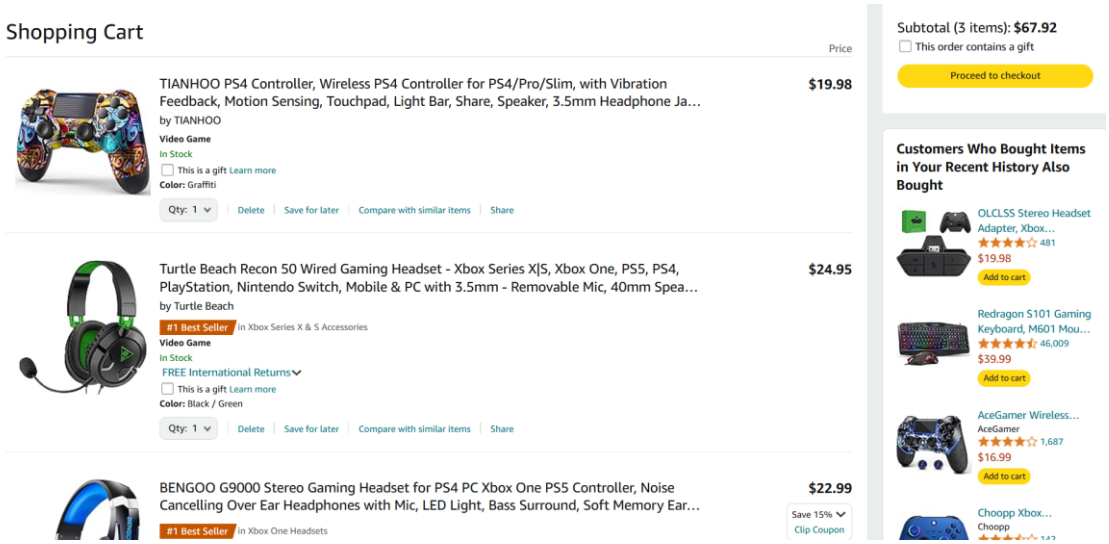


Рисунок 1.8 – Блок корзини Amazon

Amazon розроблений як для настільних комп’ютерів, так і для мобільних пристроїв, і його адаптивний дизайн дозволяє комфортно користуватися сайтом на різних екранах. Зокрема, мобільна версія зберігає основні функції, такі як пошуковий рядок, меню категорій і кошик, доступні в зручному форматі.

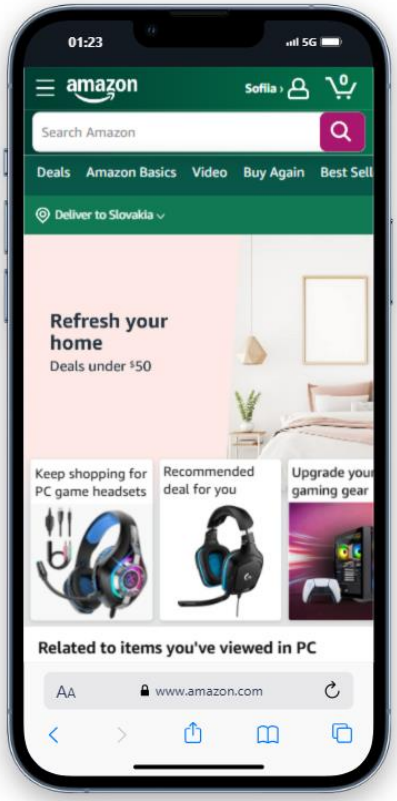


Рисунок 1.9 – Мобільна версія Amazon

Дизайн сайту також передбачає використання стратегії lazy loading (відкладеного завантаження) для зображень, що дозволяє швидко завантажувати лише перші необхідні елементи сторінки, а інші — по мірі прокручування сторінки користувачем. Завдяки цьому сторінка завантажується набагато швидше, оскільки браузер не повинен обробляти всі зображення одночасно.

Amazon також використовує метод пріоритезації ресурсів — спершу завантажуються важливі для користувача компоненти (такими є, наприклад, зображення товарів на сторінці каталогу). Додаткові ресурси (як-от рекомендації чи додаткові відгуки) завантажуються поступово, не блокуючи рендеринг основних елементів сторінки. Це дозволяє зменшити час, необхідний для першого рендерингу сторінки, і забезпечує плавність користувацького досвіду. Зображення та медіа-файли оптимізовані за

допомогою різних форматів (наприклад, WebP), що дозволяє значно зменшити їх розмір при збереженні високої якості.

Amazon активно використовує мережу доставки контенту (CDN), що дозволяє користувачам із різних куточків світу отримувати дані з найближчого до них сервера. Це значно покращує швидкість завантаження сторінок, оскільки зменшується час, необхідний для передачі інформації від сервера до користувача.

Кожен файл JavaScript та CSS, який завантажується на сторінці, ретельно оптимізується: об’єднуються файли, щоб зменшити кількість запитів до сервера, а також застосовуються методи стиснення для зменшення їхнього розміру.

Важливим елементом є також використання передових технологій для контролю за інтерактивністю на сторінках. Amazon активно використовує фреймворки та бібліотеки, які підтримують швидку взаємодію, як React, що дозволяє значно зменшити час на рендеринг інтерфейсу. Завдяки цьому, елементи на сторінках Amazon не перезавантажуються повністю, а лише оновлюються частинами, що забезпечує відчуття миттєвого відгуку на дії користувача.

Крім того, для збільшення швидкості взаємодії з користувачем активно використовуються технології передвиконання запитів, такі як HTTP/2 і навіть HTTP/3. Вони дозволяють одночасно обробляти кілька запитів через один канал зв'язку, зменшуючи затримки, які виникають при необхідності завантажити кілька ресурсів одночасно.

Facebook — це одна з найпопулярніших соціальних мереж у світі, яка активно використовує передові технології для забезпечення ефективної та швидкої клієнтської частини свого веб-сайту. Інтерфейс Facebook розроблений таким чином, щоб користувачі могли легко взаємодіяти з контентом, не відволікаючись на технічні деталі, при цьому підтримується висока швидкість завантаження сторінок і плавність переходів між різними функціями.

Інтерфейс Facebook побудований таким чином, щоб забезпечити максимально комфортне та швидке використання, орієнтуючись на потреби користувачів соціальної мережі, де основним є постійний потік новин, інтеракція з іншими користувачами та зручний доступ до всіх функцій. На головній сторінці Facebook розташована стрічка новин, яка є основним елементом інтерфейсу, з якої користувачі можуть переглядати пости своїх друзів, новини, відео, фото та інший контент. Кожен пост супроводжується кнопками для взаємодії — лайк, коментар, поширення, що дає користувачам можливість миттєво реагувати на будь-який контент. Стрічка новин постійно оновлюється, і завдяки технології lazy loading, контент завантажується поступово, що забезпечує швидке завантаження сторінки та відсутність затримок при перегляді. Це особливо помітно, коли користувачі прокручують стрічку новин — нові пости з'являються лише коли вони наближаються до кінця видимої частини сторінки.

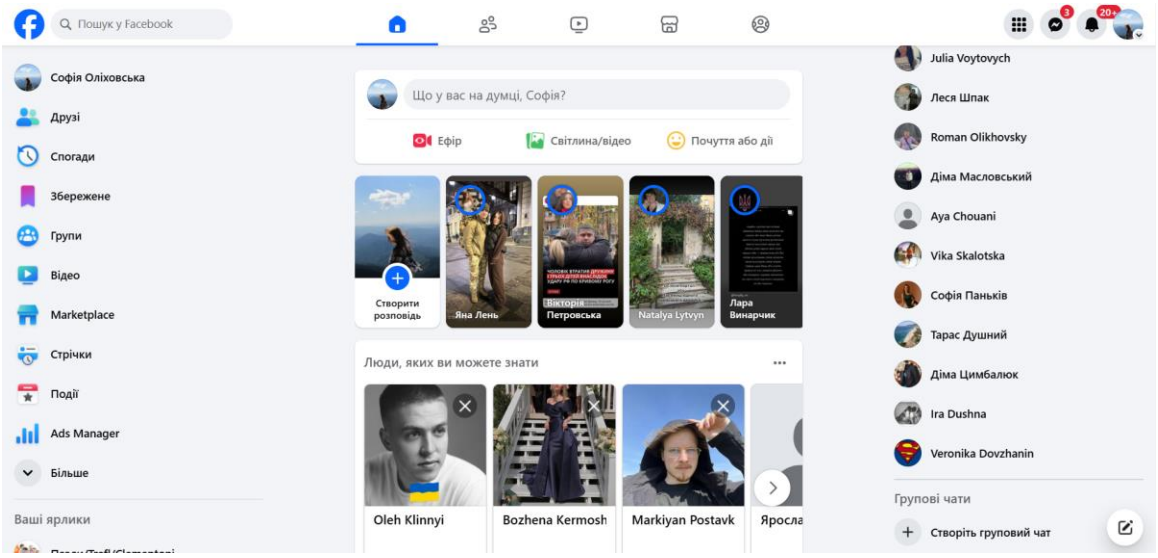


Рисунок 1.10 – Головна сторінка Facebook

У верхній частині сторінки Facebook розміщується меню навігації, що включає головні функції: профіль користувача, повідомлення, сповіщення, а також доступ до списку друзів і створення нових постів. Меню інтуїтивно зрозуміле, а доступ до ключових функцій максимально швидкий і зручний.

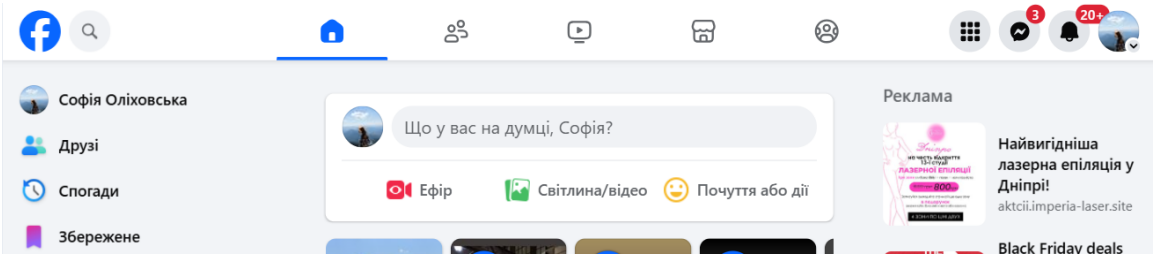


Рисунок 1.11 – Меню навігації Facebook

Наприклад, при натисканні на іконки повідомлень чи сповіщень відкривається випадаюче вікно, де миттєво відображаються нові повідомлення чи активності. Це дозволяє користувачу без зайвих кроків отримати необхідну інформацію.

Крім того, важливим елементом інтерфейсу є система повідомлень. Facebook використовує систему спливаючих вікон і оновлення в реальному часі, яка дозволяє користувачам миттєво дізнаватися про нові повідомлення, коментарі або запити. Це створює відчуття, що користувач завжди в курсі останніх подій.

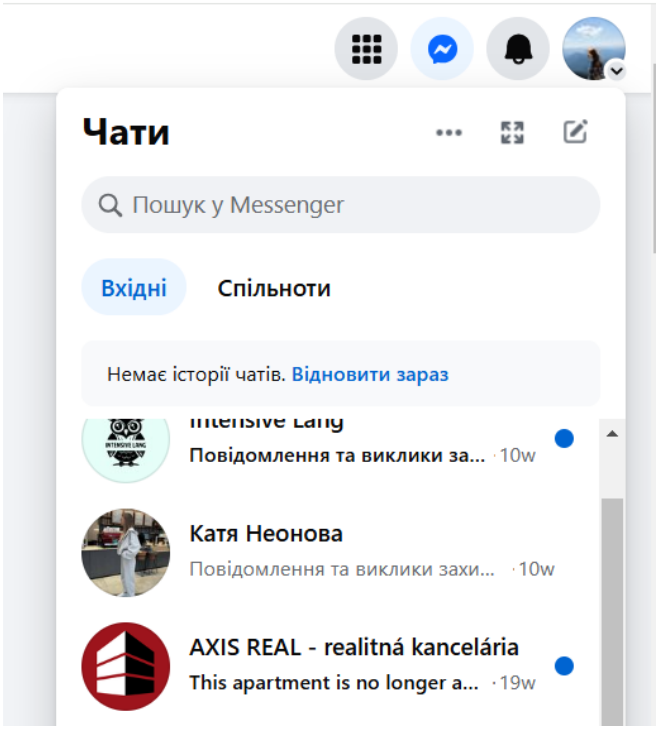


Рисунок 1.12 – Вкладка повідомлень Facebook

Інтерфейс Facebook також добре оптимізований для мобільних пристроїв. Мобільна версія сайту адаптована до розміру екрана смартфона або планшета, при цьому зберігається зручність у використанні та доступність усіх основних функцій, таких як стрічка новин, повідомлення та меню навігації. Особливо важливою є можливість швидко реагувати на повідомлення, коментарі та оновлення через мобільний додаток або мобільну версію сайту.

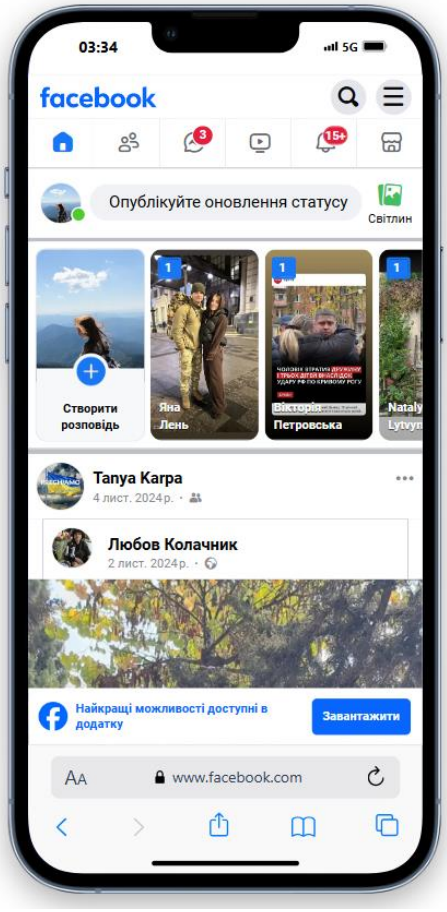


Рисунок 1.13 – Мобільна версія Facebook

Система взаємодії з контентом у Facebook включає інтерактивні елементи, які оновлюються без необхідності перезавантаження сторінки. Наприклад, коли користувач ставить лайк або коментує публікацію, ці елементи відразу відображаються без додаткового завантаження сторінки.

Одним із важливих елементів оптимізації інтерфейсу є ретельне управління медіа-контентом. Всі зображення, відео та інші медіа-файли

автоматично адаптуються до розміру екрану, що дозволяє зменшити розмір файлів, зберігаючи їх якість. Це важливо, оскільки медіа-контент займає значну частину загального обсягу трафіку на сайті. Використання адаптивних зображень дає змогу браузерам завантажувати зображення відповідно до роздільної здатності екрану користувача, що забезпечує економію трафіку та підвищення швидкості завантаження.

Facebook також активно використовує кешування для оптимізації швидкості завантаження контенту. Кешування в браузері дозволяє зберігати деякі елементи сайту на комп'ютері користувача, щоб при наступному відвідуванні ці елементи не завантажувались заново. Крім того, для зменшення часу завантаження часто використовуються ресурси з мереж доставки контенту (CDN), що дозволяє отримувати контент з найближчого до користувача сервера.

Загалом, обидва сайти застосовують схожі технології для оптимізації клієнтської частини, такі як кешування, CDN, асинхронне завантаження і адаптивні зображення. Однак їхній підхід до оптимізації відрізняється залежно від основних завдань платформи. Amazon надає пріоритет швидкому завантаженню товарів, зображень та інформації, що дозволяє швидко завершити покупку. Facebook, у свою чергу, орієнтований на швидку і безперервну взаємодію з контентом, тому його оптимізація спрямована на забезпечення плавності перегляду стрічки новин та швидкої комунікації між користувачами. У підсумку, обидва сайти є відмінними прикладами того, як сучасні технології оптимізації можуть допомогти забезпечити високий рівень користувацького досвіду, зберігаючи при цьому високу швидкість завантаження та зручний інтерфейс для різних типів діяльності в Інтернеті.

РОЗДІЛ 2.
ОБҐРУНТУВАННЯ, ВИБІР ТА РЕАЛІЗАЦІЯ ІНСТРУМЕНТАРІЮ ВИРІШЕННЯ ЗАДАЧІ

2.1 Застосування мініфікації та оптимізації розмірів файлів

Зменшення розміру ресурсів і мініфікація є важливими методами оптимізації клієнтської частини веб-сайту, які забезпечують скорочення обсягу переданих даних і пришвидшують завантаження веб-сторінки. Ці методи направлені на оптимізацію кожного файлу, який передається з сервера до браузера користувача, та включають різноманітні техніки для зменшення їхнього розміру без втрати функціональності або якості.

Першим кроком у зменшенні розміру ресурсів є використання мініфікації. Мініфікація — це процес видалення з файлів CSS, JavaScript і HTML усіх непотрібних символів, які не впливають на функціональність коду. У HTML, CSS і JavaScript файлах часто містяться пробіли, символи переносу рядка, табуляції та коментарі, які полегшують розробку і читаємість коду для програмістів, але водночас збільшують обсяг файлу. Мініфікація автоматично видаляє ці символи, перетворюючи файл у більш компактний і швидший для завантаження. Це дозволяє скоротити обсяги даних, що передаються через мережу, та знизити навантаження на сервери і користувацькі пристрої. Різниця в розмірах між оригінальними і мініфікованими файлами може бути значною — іноді до 30-40%, залежно від коду. Особливо це важливо для мобільних пристроїв і повільних мережевих з’єднань, де будь-яка економія обсягу даних може істотно покращити швидкість роботи сайту. [10]

Приклад мініфікації:

Оригінальний код:

```
function calculateSum(a, b) {
    let result = a + b;
    console.log('Сума:', result);
    return result;
}
```

Рисунок 2.1 – Приклад немініфікованого коду

Мініфікований код:

```
function calculateSum(a,b){let c=a+b;console.log('Сума:',c);return c;}
```

Рисунок 2.2 – Приклад мініфікованого коду

Окрім звичайної мініфікації, зменшення розміру ресурсів передбачає і більш глибокі підходи. Наприклад, у JavaScript-файлах, які часто складаються з великої кількості функцій і змінних, можна скористатися таким процесом як обфускація. Вона не лише зменшує розмір, але й робить код менш зрозумілим для сторонніх осіб. Під час обфускації замінюються імена змінних, функцій та інших елементів на коротші або менш зрозумілі, що дозволяє зекономити ще більше місця. Наприклад, замість повного імені змінної userInformationObject можна використовувати коротке uIO. Хоча це робить код менш зручним для розробників, обфускація є корисною технікою для підвищення ефективності і безпеки одночасно.

Приклад обфускації:

До обфускації:

```
function displayMessage(userName) {
    console.log(`Привіт, ${userName}!`);
    }
```

Рисунок 2.3 – Приклад коду до обфускації

Після обфускації:

```
function a(b){console.log(`Привіт, ${b}!`);}
```

Рисунок 2.4 – Приклад коду після обфускації

Ще один метод зменшення розміру ресурсів — це використання CDN (Content Delivery Network), яка дозволяє зберігати і доставляти файли з серверів, що розташовані ближче до користувачів. Використання CDN допомагає оптимізувати передачу даних, але також може бути корисним для скорочення розміру файлів, що завантажуються, оскільки CDN часто підтримують автоматичну мініфікацію і стиснення файлів.

Приклад використання CDN:

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

За допомогою цього коду, час завантаження завдяки кешуванню бібліотеки на сервері CDN став менший.

Стискання ресурсів є ще одним аспектом, який слід розглянути в рамках зменшення розміру файлів. Наприклад, використання таких алгоритмів як gzip або Brotli може значно знизити обсяги переданих даних. Сервери, налаштовані на стискання файлів, передають файли вже у стислому вигляді, що дозволяє браузерам швидко отримати і відобразити їх. Стискання HTML, CSS і JavaScript файлів за допомогою gzip або Brotli може зменшити їхній обсяг у 2-3 рази, що особливо корисно для великих ресурсів. Brotli, який є новішим алгоритмом, забезпечує навіть вищий рівень стиснення в порівнянні з gzip і підтримується більшістю сучасних браузерів. Це означає, що, якщо на сервері увімкнено підтримку Brotli, можна значно зменшити розмір переданих файлів. [11]

Використання таких інструментів, як Webpack, Parcel або Gulp, також допомагає в автоматизації процесу мініфікації та стиснення. Вони

забезпечують ефективне управління ресурсами та дозволяють налаштувати процеси мініфікації, стискання та об’єднання файлів для досягнення максимального зменшення розміру. Webpack, наприклад, підтримує плагіни для автоматичної мініфікації та обфускації JavaScript-коду, що робить його потужним інструментом для зниження обсягу файлів. Gulp і Parcel також підтримують процеси мініфікації і надають можливість розробникам легко інтегрувати їх у свій робочий процес. [12]

Ще одним аспектом оптимізації клієнтської частини сайту є об’єднання файлів. Замість того, щоб передавати окремі файли CSS або JavaScript, їх можна об’єднати в один великий файл, що зменшує кількість HTTP-запитів, які браузер робить до сервера. Менша кількість запитів дозволяє браузеру швидше завантажити і обробити сторінку, що покращує загальну швидкодію.

Приклад об’єднання файлів:

Окремі файли:

```
<link href="styles/header.css" rel="stylesheet">
<link href="styles/footer.css" rel="stylesheet">
<script src="scripts/slider.js"></script>
<script src="scripts/menu.js"></script>
```

Об’єднані файл:

```
<link href="styles/all-styles.css" rel="stylesheet">
<script src="scripts/all-scripts.js"></script>
```

Таким чином, зменшення розміру ресурсів сайту і мініфікація є важливими етапами оптимізації, що забезпечують зменшення обсягів даних, які передаються до кінцевого користувача. Це дозволяє підвищити швидкість завантаження сторінки, знизити навантаження на сервери і покращити зручність користування сайтом, особливо для користувачів із повільним інтернет-з’єднанням або мобільних пристроїв з обмеженими ресурсами.

2.2. Асинхронне завантаження скриптів і відкладене завантаження

Асинхронне завантаження скриптів і відкладене завантаження є одними з ключових методів оптимізації клієнтської частини, що дозволяють зменшити затримки під час завантаження веб-сторінки та забезпечити більш плавний і швидкий користувацький досвід. У веб-розробці використання цих підходів є вкрай важливим, оскільки вони допомагають мінімізувати вплив великих і часто складних JavaScript-файлів, які можуть затримувати відображення основного вмісту сторінки. Коли браузер починає завантажувати HTML-документ, усі ресурси, пов’язані з ним, такі як стилі та скрипти, обробляються у певному порядку. За звичайних умов браузер зупиняється для обробки JavaScript, що може затримувати завантаження решти сторінки. У великих та складних веб-додатках це може викликати суттєві затримки у відображенні інтерфейсу, тому методи асинхронного завантаження та відкладеного завантаження стали важливими інструментами для веб-розробників.

Асинхронне завантаження скриптів дозволяє браузеру продовжувати обробку інших ресурсів сторінки, поки завантажується та виконується JavaScript. У HTML використовується атрибут async, який забезпечує таке асинхронне завантаження. При встановленому async браузер не чекає на завантаження скрипту для відображення інших елементів сторінки, що дозволяє швидше показати вміст для користувача. Скрипти, завантажені асинхронно, можуть виконуватися у будь-якому порядку, залежно від того, який з них завантажився першим. Це особливо корисно для скриптів, що не залежать один від одного і не мають значного впливу на початковий рендеринг сторінки. Наприклад, аналітичні або маркетингові скрипти, що збирають дані для подальшого аналізу, але не змінюють вигляд або функціонал сайту, є ідеальними кандидатами для асинхронного завантаження, оскільки вони не втручаються в основний процес відображення сторінки.

Приклад використання асинхронного завантаження скриптів:

<script async src="analytics.js"></script>

<script async src="marketing.js"></script>

Відкладене завантаження або "ліниве" завантаження (lazy loading) спрямоване на завантаження скриптів лише тоді, коли вони необхідні для певної дії чи взаємодії з користувачем. Це дозволяє уникнути початкового завантаження великих обсягів даних, поки вони не знадобляться користувачу. Наприклад, якщо на сторінці присутні функції, що активуються тільки після взаємодії користувача (натискання кнопки або скролінг), то їхні скрипти можна завантажити лише тоді, коли ця взаємодія відбувається. Це економить час при першому завантаженні сторінки і покращує користувацький досвід, зменшуючи час до взаємодії (Time to Interactive), що є критичним показником для продуктивності. [13]

Поєднання асинхронного та відкладеного завантаження також допомагає зменшити витрати на обробку ресурсів, що є особливо важливим для мобільних користувачів, у яких обсяг оперативної пам’яті та пропускна здатність мережі можуть бути обмеженими. Браузер виконує лише ті скрипти, які необхідні для основного інтерфейсу, а інші скрипти завантажуються поступово, коли користувач починає взаємодіяти з додатковими елементами або функціоналом. Це дозволяє суттєво підвищити ефективність використання ресурсів браузера та уникнути надмірного навантаження на пристрої з обмеженими ресурсами.

Технології асинхронного та відкладеного завантаження підтримуються у багатьох сучасних фреймворках та бібліотеках для розробки веб-додатків, таких як React, Vue.js та Angular, що пропонують власні методи реалізації цих підходів. Зокрема, ці фреймворки підтримують так зване "динамічне імпортування" компонентів, яке дозволяє завантажувати лише ті частини коду, які потрібні в певний момент, що сприяє економії пам’яті та ресурсів. Такий підхід значно покращує взаємодію користувача з веб-додатком, оскільки зменшується час очікування при завантаженні сторінки.

Приклад динамічного імпортування JavaScript (Lazy Loading у фреймворках):

```
document.getElementById('loadFeature').addEventListener('click', () => { import('./feature.js').then(module => {
    module.initFeature();
            }).catch(error => {
    console.error('Помилка завантаження:', error);
  });
  });
```

Рисунок 2.5 - Зразок реалізації динамічного завантаження функціоналу в JavaScript

У цій ситуації модуль feature.js завантажується тільки після натискання кнопки.

З погляду SEO та користувацького досвіду, ці методи також мають велике значення, оскільки сторінки, що швидко завантажуються, мають більший шанс отримати вищі позиції у результатах пошуку. Відповідно до оновлених вимог пошукових систем, швидкодія сторінки є важливим чинником ранжування. Сайти, які використовують асинхронне та відкладене завантаження для покращення швидкодії, зазвичай забезпечують кращу видимість у результатах пошуку і водночас пропонують кращий користувацький досвід, що знижує показники відмов та підвищує ймовірність повторного відвідування. [14]

Асинхронне завантаження та відкладене завантаження також дозволяють знизити загальний обсяг трафіку, який передається між клієнтом та сервером, що є особливо актуальним для великих проєктів з глобальною аудиторією. Зниження обсягу переданих даних зменшує ризик мережевих затримок, підвищує стійкість системи до високих навантажень і дозволяє швидше адаптуватися до зміни мережевих умов. Для сайтів та додатків, що обслуговують користувачів з різних географічних локацій, такі оптимізації є важливими для забезпечення стабільної швидкодії.

Загалом, впровадження асинхронного та відкладеного завантаження є однією з основних стратегій для покращення продуктивності сучасних веб-додатків. Вони дозволяють прискорити час завантаження сторінок, покращити

досвід користувачів, зменшити навантаження на сервер і забезпечити ефективне використання мережевих ресурсів, що робить ці методи надзвичайно важливими для сучасних веб-розробників.

2.3 Оптимізація зображень та графічних елементів

Оптимізація зображень та графічних елементів є критичним аспектом для підвищення швидкодії веб-сайтів, особливо у випадках, коли вони містять великий обсяг візуальних матеріалів. У сучасному веб-дизайні зображення займають значну частку обсягу даних, що завантажуються під час відвідування сторінки, і можуть суттєво вплинути на час завантаження сайту. Якщо зображення не оптимізовані, це може призводити до збільшення трафіку, зниження продуктивності та затримок у відображенні контенту. З огляду на те, що швидкість завантаження є важливим фактором ранжування сайтів у пошукових системах, а користувачі все частіше обирають перегляд з мобільних пристроїв з обмеженими ресурсами, оптимізація графічних елементів є особливо актуальною [15].

Одним із ключових методів оптимізації зображень є вибір оптимального формату файлу. У сучасному веб-дизайні найпоширенішими форматами зображень є JPEG, PNG та GIF, кожен з яких має свої особливості. JPEG є оптимальним для зображень з великою кількістю кольорів і складною деталізацією, таких як фотографії, оскільки підтримує стиснення з втратами, що дозволяє значно зменшити розмір файлу без суттєвих змін у якості зображення. PNG, який підтримує стиснення без втрат, більше підходить для зображень з прозорим фоном та графічних елементів з малим числом кольорів, таких як логотипи. GIF, хоча і використовується для анімацій, має обмеження в палітрі кольорів і підходить для простих ілюстрацій та іконок. Проте останнім часом також набирають популярність нові формати, як-от WebP та AVIF, що забезпечують ефективніше стиснення, менший розмір файлу і підтримують високу якість, а також прозорість і анімацію. [16]

Наступним етапом оптимізації є стиснення зображень, яке передбачає зменшення розміру файлу без суттєвих змін у візуальній якості. Існують два основні типи стиснення: стиснення з втратами і без втрат. При стисненні з втратами зображення частково втрачає деталі, які менш помітні для людського ока, але значно зменшують розмір файлу. Це особливо корисно для великих зображень та фотографій, де якість не є пріоритетною, як у випадку з фонами чи ілюстраціями. Стиснення без втрат дозволяє зберегти високу якість, але зменшує розмір файлу не так ефективно. Різні онлайн-інструменти та програми, такі як TinyPNG, JPEG Optimizer і ImageOptim, дозволяють стискати зображення та контролювати ступінь стиснення, що сприяє вибору оптимального балансу між розміром та якістю. [17]

Роздільна здатність та розміри зображень також впливають на швидкість завантаження сторінок. Зображення з високою роздільною здатністю займають значний обсяг пам’яті і часто є зайвими для відображення на екранах користувачів. Оптимізація зображень під конкретну роздільну здатність дозволяє зменшити розмір файлів і прискорити завантаження. Наприклад, для мобільних пристроїв можна використовувати зображення меншої роздільної здатності, а для великих екранів – із вищою. Використання адаптивних зображень, які змінюються залежно від розміру екрану та пристрою, є поширеною практикою у веб-розробці. HTML5 атрибут srcset дозволяє вказувати браузеру кілька варіантів зображення з різною роздільною здатністю, що дозволяє автоматично вибрати оптимальний варіант залежно від розміру екрана. Це значно покращує продуктивність, оскільки браузер завантажує лише необхідну версію зображення.

Приклад використання адаптивних зображень:

```
<img
src="image-small.jpg"
srcset="
image-small.jpg 480w,
image-medium.jpg 800w,
```

```
image-large.jpg 1200w"
sizes="(max-width: 600px) 480px,
(max-width: 1200px) 800px,
1200px"
alt="Оптимізоване зображення">
```

srcset дозволяє браузеру вибирати зображення відповідного розміру залежно від роздільної здатності екрана.

Атрибут sizes вказує бажаний розмір зображення залежно від ширини екрана.

Відкладене завантаження зображень, або lazy loading, є ще одним ефективним методом оптимізації. Цей підхід передбачає завантаження зображень лише тоді, коли вони потрапляють у поле зору користувача. Таким чином, зображення, що знаходяться нижче на сторінці і не видимі при початковому завантаженні, не завантажуються до моменту, поки користувач не дійде до них, що дозволяє зменшити початковий обсяг даних, які передаються мережею, і прискорити завантаження основного контенту. Використання відкладеного завантаження може значно знизити навантаження на сервер та забезпечити плавне відображення сторінки, особливо при наявності великої кількості зображень, що актуально для інтернет-магазинів або галерей. [18]

Приклад відкладеного завантаження зображень:

```
<img src="large-image.jpg" alt="Example" loading="lazy">
```

Рисунок 2.6 – Приклад коду з відкладеним завантаженням зображення

Атрибут loading="lazy" інтегрований у сучасні браузери і дозволяє відкладати завантаження зображень до моменту, коли вони з’являться у видимій області користувача.

Окрім того, використання CSS-спрайтів може бути корисним при роботі з графічними елементами інтерфейсу. Спрайти об'єднують кілька невеликих зображень, таких як іконки або графічні елементи, в один файл, який завантажується одночасно. Це зменшує кількість запитів до сервера, що сприяє підвищенню швидкості завантаження сторінки, оскільки кожен HTTP-запит до сервера супроводжується затримкою. CSS-спрайти дозволяють браузеру завантажувати один великий файл, а не кілька маленьких, що також зменшує трафік. [15]

Крім технічних методів, важливо враховувати, як розташування та використання зображень впливають на загальний дизайн сайту. Забезпечення узгодженості між розміром зображень та інтерфейсом дозволяє не лише зменшити завантаження сторінки, а й створює привабливий візуальний вигляд, який не перевантажує користувача надмірним обсягом інформації. Правильне використання зображень і графіки сприяє кращій організації контенту і забезпечує, щоб користувачі швидко і легко знаходили потрібну інформацію.

Окрім розглянутих методів, також можна використовувати техніку стиснення кольорової палітри. Це означає, що зображення з великою кількістю кольорів можуть бути перетворені на зображення з меншою кількістю кольорів, що значно зменшує розмір файлу без втрати візуальної якості. Така оптимізація ефективна для іконок, логотипів і простих ілюстрацій, які не вимагають високої кольорової деталізації. [16]

Зрештою, важливим аспектом є також оптимізація формату SVG для векторної графіки, що забезпечує високу якість зображень при низькому обсязі даних. Векторні зображення, на відміну від растрових, не втрачають якості при масштабуванні, що робить SVG ідеальним вибором для логотипів, іконок та інших графічних елементів. SVG може бути додатково оптимізованим за допомогою інструментів, які видаляють непотрібні елементи коду та мінімізують файл, що сприяє більш швидкому завантаженню.

Таким чином, оптимізація зображень та графічних елементів відіграє ключову роль у підвищенні продуктивності сучасних веб-сайтів, сприяючи як економії ресурсів, так і покращенню загального користувацького досвіду. Правильне застосування форматів, стиснення, адаптивного завантаження та інших технік дозволяє веб-додаткам бути швидшими, легшими та більш привабливими для користувачів, що є важливим у контексті швидкозмінного інтернет-середовища.

2.4 Адаптивний дизайн та оптимізація візуального інтерфейсу веб-сайту

У сучасному веб-дизайні адаптивність і візуальна оптимізація стали фундаментальними аспектами для забезпечення зручності користувачів. Різноманітність пристроїв, якими користуються відвідувачі сайтів, вимагає гнучких рішень, які дозволяють забезпечити однакову функціональність і якість взаємодії на мобільних телефонах, планшетах, десктопах та інших пристроях [17]. Впровадження адаптивного дизайну дозволяє веб-додаткам ефективно підлаштовуватися під різні розміри екранів і характеристики пристроїв, забезпечуючи високий рівень доступності інформації.

Адаптивний дизайн полягає у створенні інтерфейсу, який автоматично адаптується до параметрів пристрою користувача, таких як роздільна здатність екрана, орієнтація та щільність пікселів. Це досягається за допомогою гнучких сіток, медіа-запитів і гнучких графічних елементів. Гнучкі сітки дозволяють структурувати контент у вигляді пропорційних блоків, які змінюють свої розміри відповідно до ширини екрана, тоді як медіа-запити в CSS забезпечують можливість змінювати стилі залежно від характеристик пристрою. Наприклад, мобільна версія сайту може містити спрощену навігацію, більші елементи управління і оптимізований текст, що покращує взаємодію для користувачів із сенсорними екранами.

Приклад використання медіа-запитів:

```css
body {
    font-family: Arial, sans-serif;
    font-size: 16px;
}

@media (max-width: 768px) {
    body {
        font-size: 14px;
    }
}

@media (max-width: 480px) {
    body {
        font-size: 12px;
    }
}
```

Рисунок 2.7 – Приклад використання медіа-запитів

Шрифт автоматично зменшується на мобільних пристроях, забезпечуючи зручність читання.

Візуальна оптимізація також відіграє важливу роль у забезпеченні якісного користувацького досвіду. Використання CSS-анімацій дозволяє додати інтерактивність і динаміку до інтерфейсу, що сприяє кращому сприйняттю сайту. Наприклад, плавні переходи між сторінками, анімація кнопок або ефекти появи елементів можуть підвищити візуальну привабливість без шкоди для продуктивності. Важливою перевагою CSS-анімацій є їхня оптимізація для роботи у веб-браузерах за допомогою апаратного прискорення, що мінімізує вплив на швидкодію навіть на пристроях із обмеженими ресурсами. [20]

Приклад CSS анімації для кнопок:

<button class="animated-button">Купити</button>

.animated-button {

background-color: #007bff;

color: white;

padding: 10px 20px;

```
font-size: 18px;
border: none;
cursor: pointer;
border-radius: 5px;
line-height: 1.2;
transition: transform 0.3s ease,
background-color 0.3s ease;
}
.animated-button:hover {
transform: scale(1.1);
background-color: #0056b3;
}
```

При наведенні курсору кнопка збільшується і змінює колір:

(а) (б)

Рисунок 2.8 – Приклад кнопки до наведення курсора (а) та після (б)

Для складніших інтерактивних елементів використовуються оптимізовані JavaScript-бібліотеки, які надають додаткові можливості для створення анімацій та обробки користувацьких дій. Такі бібліотеки, як GSAP або Anime.js, забезпечують високий рівень контролю над процесом анімації, дозволяючи досягати плавності руху та уникати затримок [21]. Важливо враховувати, що надмірне використання анімацій і складних візуальних ефектів може негативно вплинути на продуктивність, тому під час розробки слід дотримуватися балансу між функціональністю і швидкодією.

Окрім технічних аспектів адаптивності та анімацій, важливим є забезпечення швидкого доступу до ключових функцій сайту. Елементи, такі як кнопки "Купити", "Зареєструватися" або "Зв'язатися з нами", повинні бути розташовані таким чином, щоб користувачі могли швидко їх знайти та використовувати. Це досягається завдяки продуманому розташуванню елементів на сторінці, контрастному дизайну та застосуванню підказок, які підкреслюють важливість цих дій. Наприклад, використання великих кнопок із чітким текстом і ефектами наведення дозволяє виділити їх серед іншого контенту і забезпечити інтуїтивно зрозуміле управління.

Оптимізований дизайн також враховує зменшення когнітивного навантаження на користувачів. Це передбачає мінімалістичний підхід до організації контенту, де кожен елемент має чітке призначення і не перевантажує інтерфейс. Простота візуального представлення, узгодженість кольорової палітри та вибір легких для читання шрифтів створюють комфортні умови для сприйняття інформації. Адаптивний дизайн повинен враховувати різницю у способах взаємодії з сайтом на різних пристроях: для сенсорних екранів важливо збільшити розмір інтерактивних елементів, а для десктопів — забезпечити підтримку точного наведення курсором. [22]

Таким чином, використання адаптивного дизайну в поєднанні з сучасними методами візуальної оптимізації дозволяє створювати веб-сайти, які є зручними, функціональними та привабливими для користувачів на будь-якому пристрої. Такий підхід забезпечує високу продуктивність, комфортне використання та відповідність вимогам сучасного веб-дизайну, що є критично важливим у контексті зростаючої конкуренції в інтернет-середовищі.

2.5 Огляд популярних інструментів для аналізу та оптимізації

Сучасні інструменти для аналізу продуктивності веб-сайтів розроблені для виявлення вузьких місць і обчислення основних показників продуктивності, таких як час до першого рендерингу (FCP), час до повної взаємодії (TTI) та загальний час завантаження сторінки (Load Time). Серед найбільш популярних інструментів варто виділити Google Lighthouse, PageSpeed Insights, WebPageTest, а також розширені можливості для моніторингу та аналізу від Google Analytics, що допомагає відслідковувати поведінку користувачів і їх реакції на швидкодію веб-сторінок. [23]

Google Lighthouse, інтегрований у Chrome DevTools, забезпечує всебічний аналіз продуктивності, зокрема вимірює час до першого рендерингу (First Contentful Paint, FCP), час до повної взаємодії (Time to Interactive, TTI) та обчислює загальний час завантаження сторінки (Load Time). Lighthouse відстежує і тестує сайт на таких ключових параметрах, як продуктивність, доступність, відповідність прогресивним веб-додаткам (PWA), SEO та застосування найкращих практик. Інструмент генерує інтегрований звіт з детальними індикаторами продуктивності й рекомендаціями щодо поліпшення. Наприклад, він може порадити мініфікацію JavaScript, використання сучасних форматів зображень (WebP, AVIF), оптимізацію завантаження CSS, уникнення блокуючих ресурсів та оптимізацію під час роботи в мобільних умовах. Результати Lighthouse, представлені у вигляді конкретних показників та рекомендацій, легко впроваджуються в робочий процес розробки й дозволяють розробникам контролювати прогрес оптимізації з кожним запуском.

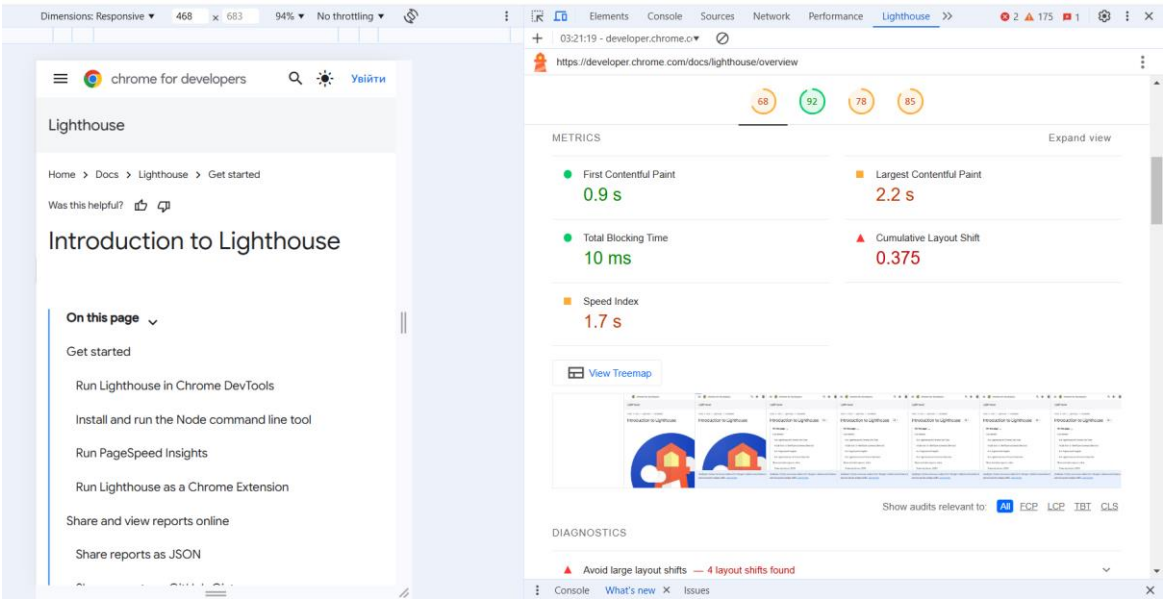


Рисунок 2.9 – Приклад аналізу сторінки за допомогою Google Lighthouse

PageSpeed Insights один із найвідоміших інструментів від Google для комплексного аналізу швидкодії як десктопної, так і мобільної версій веб-сайтів. Він використовує дані з Chrome User Experience Report (CrUX), що дозволяє аналізувати сайт з урахуванням реальних умов використання, включаючи повільні мережі та різноманітні пристрої. PageSpeed Insights надає детальний аналіз ключових показників продуктивності, таких як FCP, Largest Contentful Paint (LCP), Total Blocking Time (TBT) та Cumulative Layout Shift (CLS). Крім того, він вказує на конкретні аспекти, що потребують покращення, та пропонує рекомендації: використання стиснення даних (gzip, Brotli), кешування ресурсів, асинхронне завантаження скриптів, lazy-loading для зображень і видалення непотрібних запитів. Інструмент також присвоює оцінку від 0 до 100 для кожного проаналізованого сайту, що дає змогу розробникам швидко оцінити загальну швидкодію та зрозуміти, на чому зосередитися для оптимізації.

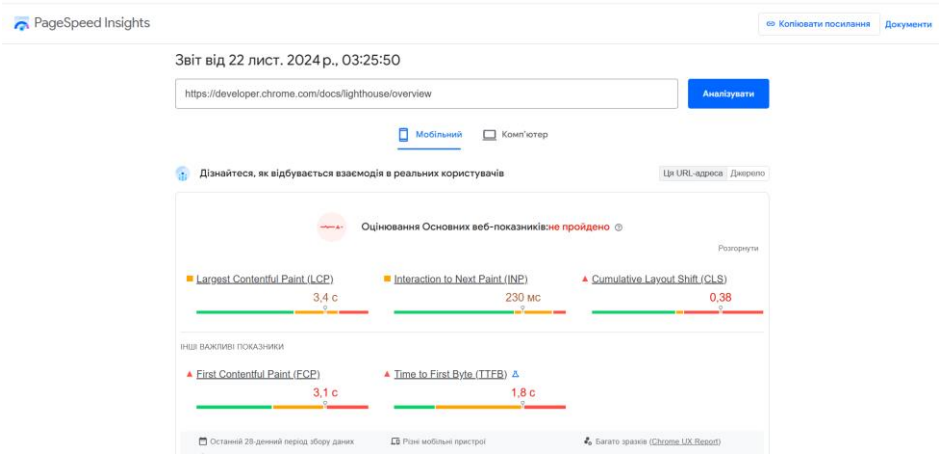


Рисунок 2.10 – Приклад аналізу сторінки за допомогою PageSpeed Insights

WebPageTest є розширеним інструментом, що дозволяє тестувати продуктивність сайтів при різних мережевих умовах, регіонах, швидкості мережі та на різних пристроях. Він надає детальну інформацію про завантаження сторінок, включаючи водоспадну діаграму (Waterfall View), яка показує, як і в якій послідовності завантажуються ресурси. Така детальна візуалізація дозволяє виявити затримки завантаження, зайві запити або проблемні ресурси, що блокують рендеринг сторінки. WebPageTest також вимірює такі показники, як час до першого байта (Time to First Byte, TTFB), час до відображення першого контенту та інші, що дозволяють зрозуміти технічні аспекти завантаження і точніше оптимізувати ресурс. WebPageTest є особливо корисним для великих або складних веб-проєктів, де точність та деталізація результатів критично важливі для успішного впровадження змін.

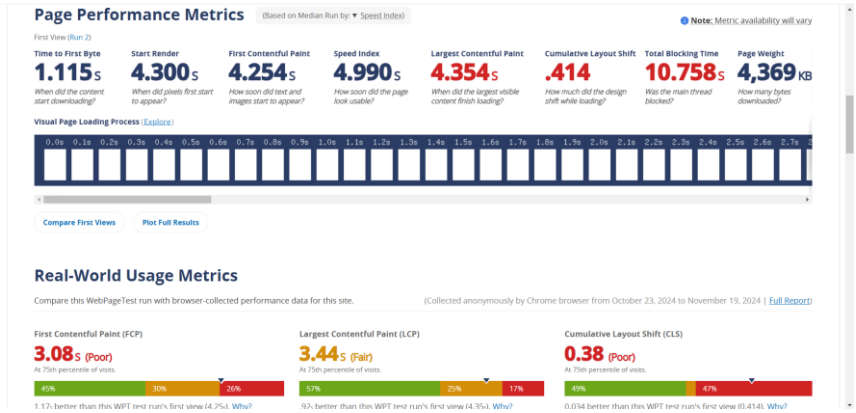


Рисунок 2.11 – Приклад аналізу сторінки за допомогою WebPageTest

Chrome DevTools — вбудований інструмент Google Chrome для розробників, що дозволяє проводити багаторівневий аналіз продуктивності та вивчати роботу внутрішніх компонентів сайту. Використовуючи вкладки «Performance», «Network» та «Coverage», можна в реальному часі відстежувати мережеві запити, аналізувати розмір файлів, завантажених скриптів та стилів, визначати блокуючі ресурси та переглядати, які файли використовуються недостатньо ефективно. Вкладка «Performance» дозволяє запускати профілювання, щоб отримати інформацію про рендеринг, виконання скриптів та затримки. Інструмент також дає можливість перевірити, як виглядає сайт при повільному інтернет-з'єднанні або на мобільному пристрої, що дозволяє детально вивчити вплив оптимізацій на продуктивність у реальних умовах.

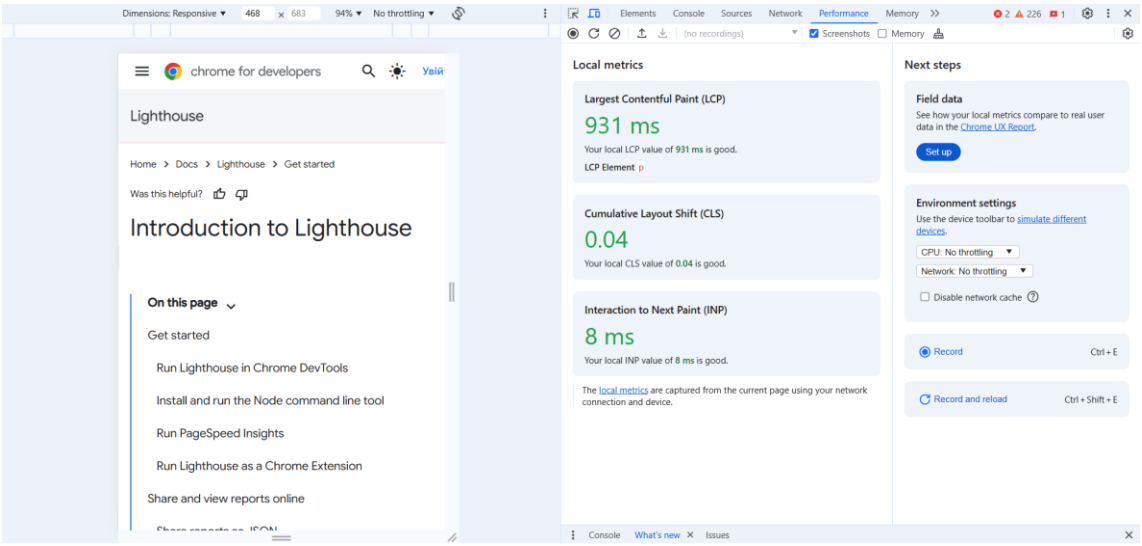


Рисунок 2.12 – Приклад аналізу сторінки за допомогою Chrome DevTools

Google Analytics є аналітичним інструментом, який відслідковує поведінку користувачів на веб-сайті. Він надає важливу інформацію про показники відмов, тривалість сеансів, шляхи навігації, що дозволяє оцінити, як користувачі взаємодіють зі сторінкою. Відстежуючи такі метрики, як час на сторінці, шляхи переходів та відмови, Google Analytics допомагає зрозуміти, як зміни в оптимізації впливають на поведінку користувачів. Розробники

можуть використовувати аналітичні звіти для визначення, які елементи потребують оптимізації, чи втрачає сайт відвідувачів через повільну швидкодію та як саме різні покращення впливають на загальний користувацький досвід. Google Analytics є важливим інструментом для тривалого моніторингу продуктивності веб-сайтів та забезпечення їх відповідності очікуванням користувачів.

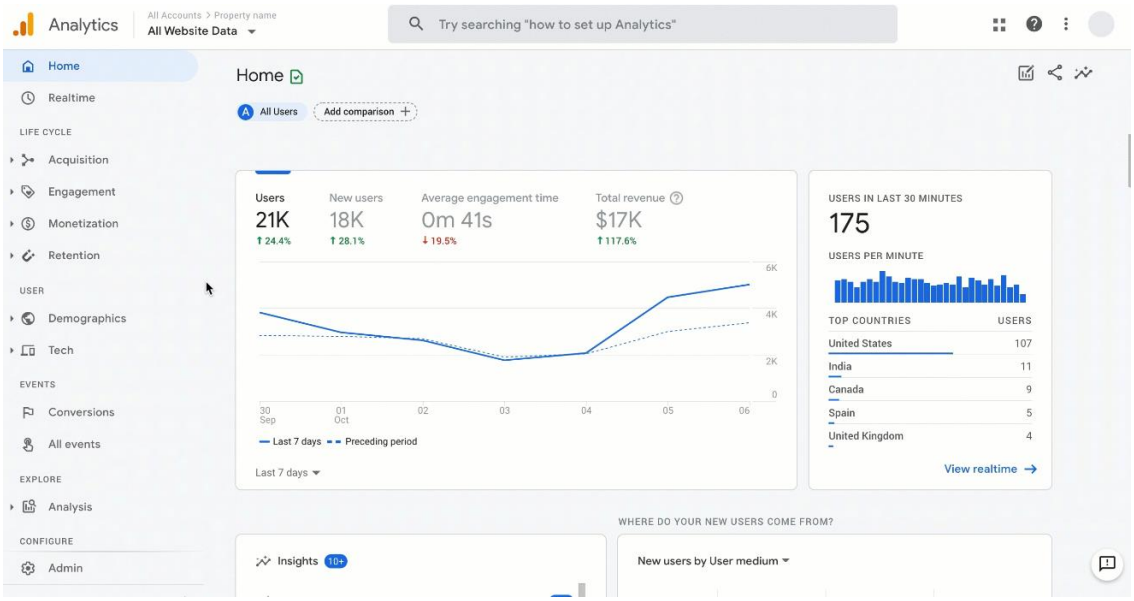


Рисунок 2.13 – Приклад аналізу сторінки за допомогою Google Analytics

Інструменти для аналізу та оптимізації клієнтської частини веб-сайтів допомагають не тільки покращити швидкодію, але й забезпечують кращий користувацький досвід. Кожен з розглянутих інструментів пропонує унікальні функції та можливості, які дозволяють глибше зрозуміти специфіку роботи сайту, виявити ключові показники та реалізувати ефективні оптимізаційні стратегії.

Застосування інструментів для аналізу продуктивності клієнтської частини допомагає розробникам глибше зрозуміти поведінку свого сайту в реальних умовах і швидко вирішити проблеми, що впливають на швидкість завантаження та взаємодію. Вибір інструменту залежить від потреб проєкту: для швидкої оцінки підходять інструменти типу PageSpeed Insights, для

глибшого аналізу з регіональними варіаціями — WebPageTest, а для детального профілювання і дослідження внутрішньої структури сторінок — Chrome DevTools. Кожен з інструментів має власні особливості, і їхнє комбінування дозволяє досягти максимальної точності в діагностиці і покращенні продуктивності клієнтської частини веб-сайтів.

РОЗДІЛ 3.
РЕЗУЛЬТАТИ ВИРІШЕННЯ ЗАДАЧІ

3.1 Обґрунтування вибору інструменту для аналізу клієнтської частини веб-сайту

Оптимізація клієнтської частини веб-сайту є одним із ключових етапів у забезпеченні його швидкодії, доступності та ефективної взаємодії з користувачами. Для досягнення цих цілей важливо використовувати інструменти, які забезпечують всебічний аналіз технічних характеристик сайту, виявляють проблемні місця та пропонують рекомендації щодо їх усунення. У нашому випадку мова йде про сайт **Marcho** — сучасний веб-ресурс, орієнтований на продаж стильного одягу для різних категорій споживачів. Цей сайт включає інтерактивні елементи, велику кількість зображень товарів, адаптивний дизайн і функціональні можливості для зручної навігації, що робить його ідеальним прикладом для аналізу й оптимізації клієнтської частини.

Серед інструментів, які були оглянуті (Google Lighthouse, PageSpeed Insights, WebPageTest, Chrome DevTools і Google Analytics), для аналізу веб-сайту Marcho був обраний Google Lighthouse. Цей вибір обґрунтований його широкими можливостями й здатністю надавати інтегрований звіт із детальними рекомендаціями для вирішення виявлених проблем.

Google Lighthouse забезпечує комплексний аналіз ключових аспектів продуктивності веб-сайту: швидкість завантаження, доступність, оптимізацію для пошукових систем (SEO), прогресивність веб-додатків (PWA) та найкращі практики розробки. Для сайту Marcho, який має велике графічне навантаження через зображення товарів та інтерактивні елементи, Lighthouse є оптимальним вибором, оскільки дозволяє оцінити всі ці параметри в комплексі.

PageSpeed Insights, хоча й також розроблений Google, пропонує лише основні показники швидкодії та короткі рекомендації. У порівнянні з

Lighthouse, цей інструмент обмежується аналізом швидкості завантаження сторінок і не надає глибокого аналізу доступності або SEO-оптимізації. Для такого багатофункціонального сайту, як Marcho, цього недостатньо.

WebPageTest пропонує потужні функції для тестування швидкодії, включаючи моделювання різних типів підключення та тестування з різних географічних точок. Проте основна увага цього інструмента зосереджена саме на продуктивності, тоді як Lighthouse оцінює також доступність та SEO, що є критичними для комерційних сайтів, таких як Marcho.

Інструмент ***Chrome DevTools*** надає розробникам гнучкий інтерфейс для ручного аналізу продуктивності, однак він не пропонує автоматизованого аналізу та інтегрованих рекомендацій. Для сайту Marcho, який потребує швидкого отримання конкретних висновків і дій, Lighthouse виглядає значно зручнішим.

Google Analytics, у свою чергу, більше орієнтований на аналіз поведінки користувачів і маркетингову ефективність. Попри те, що він є цінним для відстеження тенденцій відвідуваності та продажів, Google Analytics не проводить технічний аудит клієнтської частини сайту.

Для сайту, такого як Marcho, важливо досягти балансу між привабливим дизайном, багатофункціональністю та швидкодією. Google Lighthouse дозволяє виявити такі проблеми, як надмірно великі зображення, блокуючі скрипти, неефективне кешування ресурсів тощо. Завдяки автоматизованому аналізу та рекомендаціям Lighthouse ми можемо виявити конкретні аспекти, які потребують оптимізації.

Таким чином, Google Lighthouse був обраний для аналізу клієнтської частини веб-сайту Marcho через його здатність забезпечувати комплексний, автоматизований і багатогранний аналіз. Його переваги перед іншими інструментами дозволяють не лише ідентифікувати проблеми, а й запропонувати конкретні рішення для їх усунення.

3.2Аналіз клієнтської частини веб-сайту за допомогою обраного інструменту

Для оцінки продуктивності, доступності та зручності використання клієнтської частини веб-сайту було застосовано інструмент Google Lighthouse. Цей аналіз дозволив ідентифікувати ключові проблеми, що впливають на роботу ресурсу, та визначити напрямки для його оптимізації.

Аналіз клієнтської частини веб-сайту Marcho за допомогою Google Lighthouse виявив кілька важливих проблем, які знижують ефективність роботи сайту та його загальну зручність для користувачів.

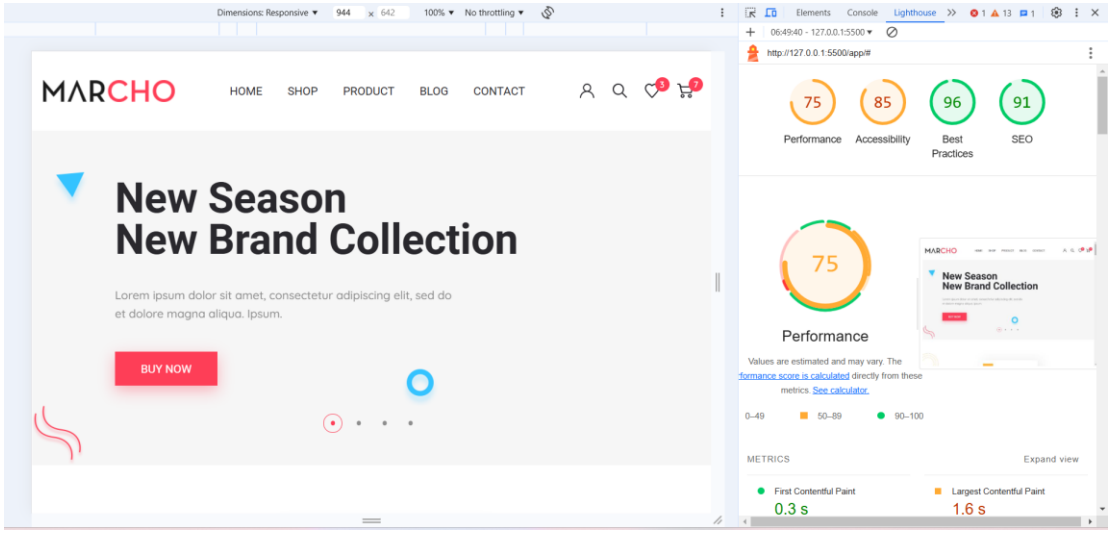


Рисунок 3.1 – Результат аналізу сайту Marcho за допомогою Google Lighthouse

Першою проблемою є недостатня адаптивність сайту. Хоча він працює на більших екранах досить добре, при перегляді на мобільних пристроях і планшетах помітні серйозні недоліки. Всі елементи сайту не адаптуються до різних розмірів екранів. Це викликає необхідність у прокручуванні сторінки по горизонталі, що є незручним для користувачів, особливо на маленьких екранах смартфонів (див. рис. 3.2).

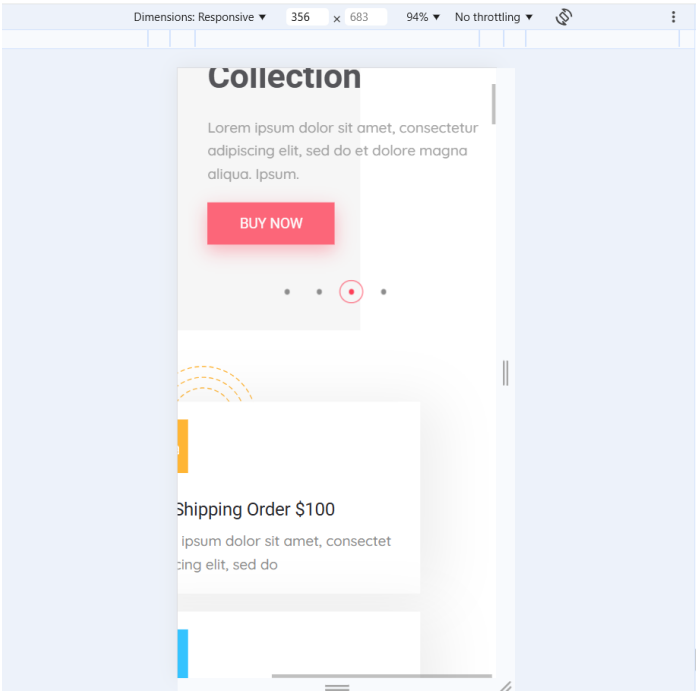


Рисунок 3.2 – Приклад неадаптованого блоку банера та умов сайту Marcho

Шрифти на сайті теж виглядають надто великими для малих екранів, що призводить до погіршення сприйняття текстової інформації та зменшення зручності взаємодії з контентом (див. рис. 3.3).

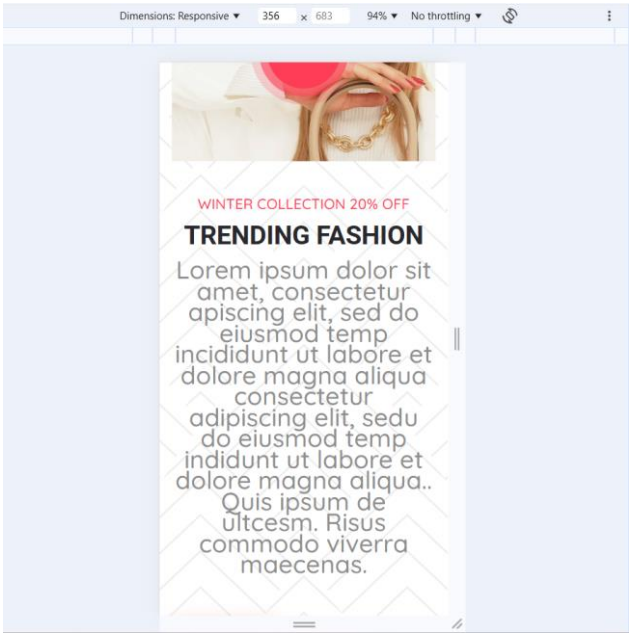


Рисунок 3.3 – Приклад проблеми адаптації шрифтів для малих екранів на сайті Marcho

Ще однією суттєвою проблемою є використання зображень у форматі JPG. Формат JPG, хоч і є оптимальним для деяких типів зображень, таких як іконки або логотипи, не є найбільш ефективним для фотографій товарів чи банерів, де більш доречним було б використання формату WebP. Це дозволило б зменшити розмір зображень без втрати якості, що значно підвищило б швидкість завантаження сайту. Також важливо зазначити, що багато елементів на сайті не мають вказаної фіксованої ширини та висоти. Це може призвести до того, що браузер не може точно визначити розміри елементів до їхнього завантаження, що, у свою чергу, спричиняє додаткові затримки в рендерингу сторінки. Відсутність вказівки цих параметрів ускладнює оптимізацію відображення контенту браузером, що призводить до збільшення часу завантаження і зниження загальної продуктивності сайту (див. рис. 3.4).

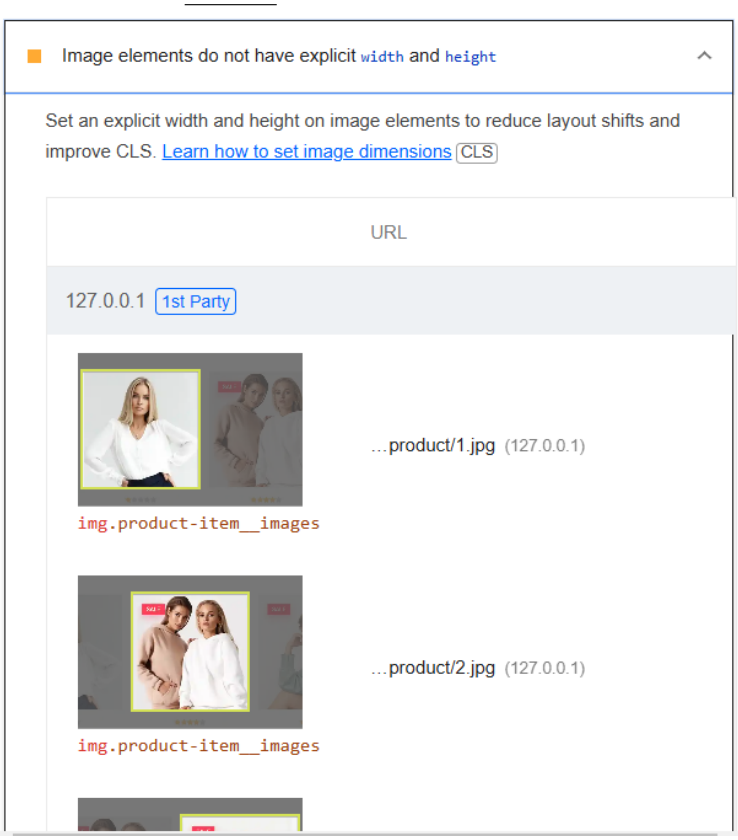


Рисунок 3.4 – Попередження через відсутність тегів width та height

Також сайт страждає від відсутності оптимізації в плані завантаження скриптів. Всі JavaScript-скрипти на сайті завантажуються синхронно, тобто

браузер чекає завершення завантаження кожного скрипта перед завантаженням інших ресурсів. Це створює значні затримки в рендерингу сторінки, оскільки без асинхронного завантаження сторінка не може повністю відображатися, поки скрипти не будуть повністю завантажені та виконані. Для того, щоб покращити цей процес, необхідно використовувати атрибути async для завантаження скриптів, що дозволить браузеру відображати сторінку паралельно з завантаженням і виконанням JavaScript, зменшуючи час до першого відображення контенту.

Додатково, сайту не вистачає візуальних індикаторів взаємодії з користувачем, таких як ховери на кнопках, що може заплутати користувача, який не розуміє, що ці елементи є клікабельними. Кнопки та інші інтерактивні елементи повинні змінювати свій вигляд при наведенні курсора або натисканні, щоб користувачі чітко розуміли, що їх можна натискати. Відсутність таких візуальних ефектів значно знижує зручність інтерфейсу і може призвести до негативного досвіду користувачів, особливо на нових або незнайомих сайтах (див. рис. 3.5).

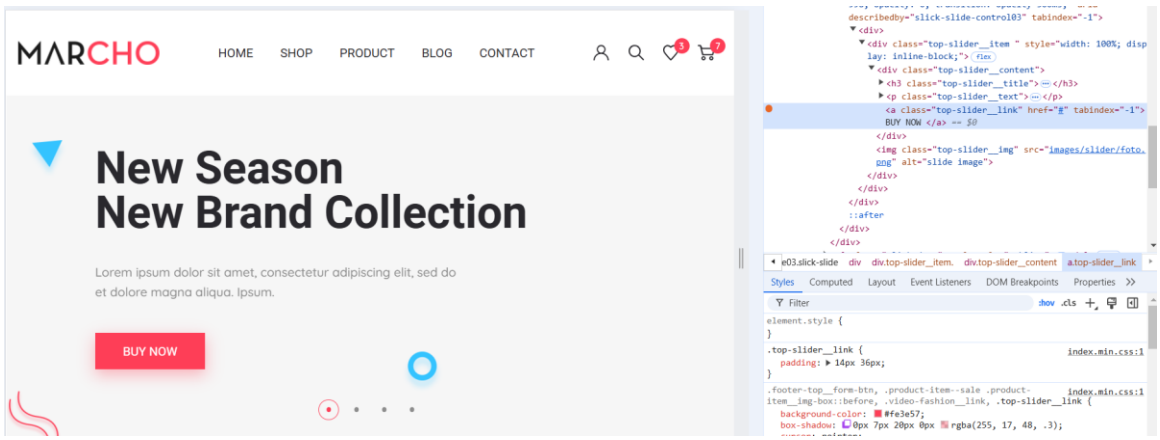


Рисунок 3.5 – Відсутність ховера при наведенні на кнопку BUY NOW

Ще однією значною проблемою є те, що файл Java Script на сайті не мініфікований. Відсутність мініфікації цього файлу призводить до його надмірного розміру, що в свою чергу збільшує час завантаження сторінки (див. рис. 3.6).

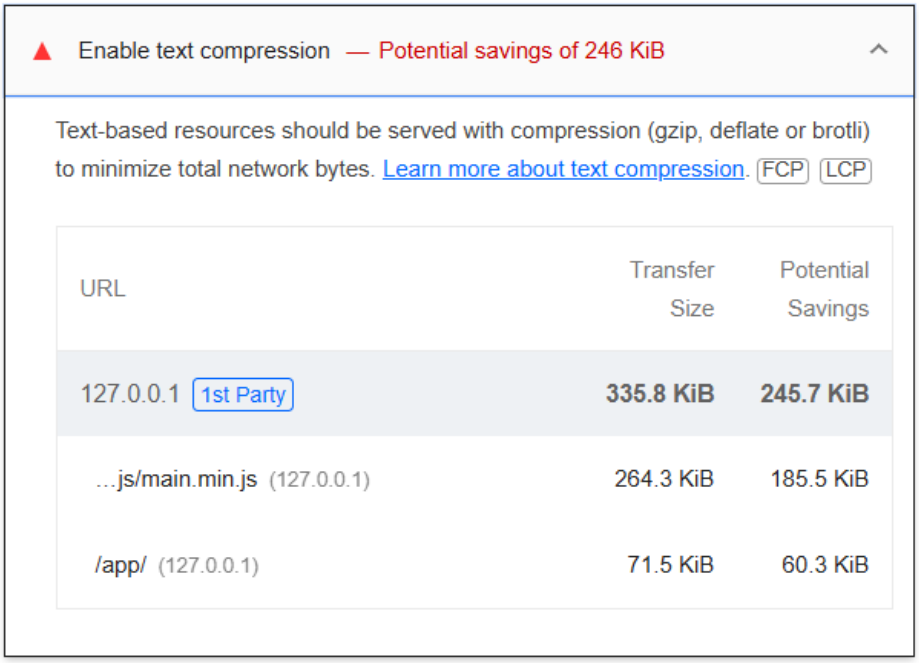


Рисунок 3.6 – Проблема відсутності мініфікації файлу

Таким чином, основні проблеми веб-сайту Marcho пов'язані з неефективним використанням ресурсів, відсутністю адаптивності для мобільних пристроїв і недостатньою оптимізацією графічних елементів. Усунення цих проблем дозволить значно покращити швидкість завантаження сторінки, зручність навігації та загальний досвід користувачів на сайті.

Додатково до перерахованих проблем, варто звернути увагу на навігаційне меню сайту, яке не є закріпленим в верхній частині сторінки. Це може створювати значні труднощі для користувачів, особливо при перегляді сайту на великих сторінках при прокручуванні контенту. Відсутність фіксованого меню означає, що користувачам доводиться прокручувати сторінку до самого верху, щоб знову отримати доступ до основних розділів сайту. У сучасних веб-дизайнах закріплене меню, яке завжди залишається видимим, навіть при прокручуванні контенту, є важливим елементом зручності. Воно дозволяє користувачам швидко переходити між розділами сайту, не витрачаючи час на пошук навігаційних елементів після кожного прокручування (див. рис. 3.7).

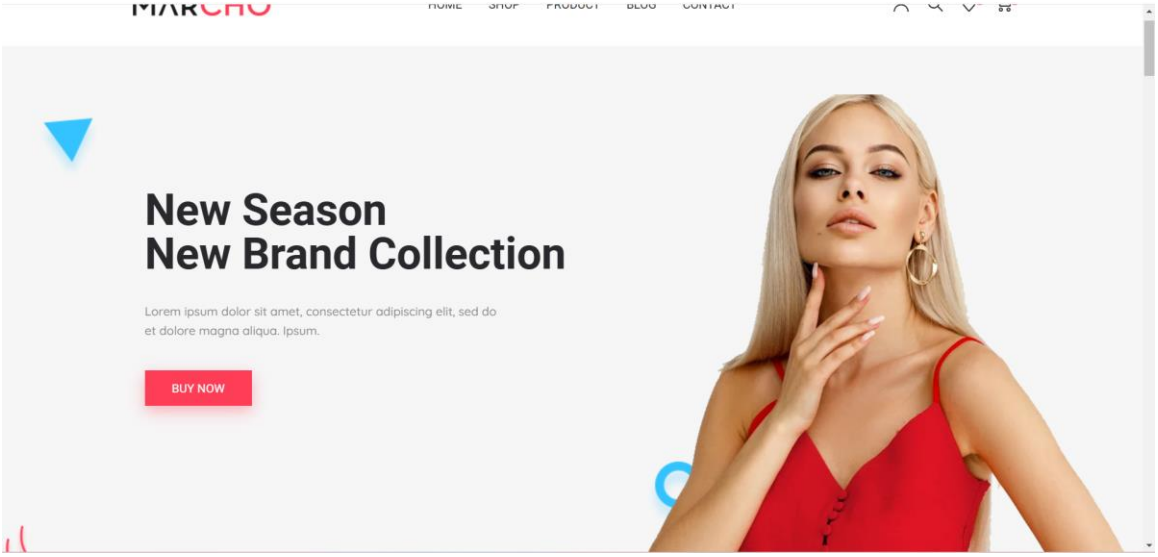


Рисунок 3.7 – Відсутність фіксованого меню на сайті Marcho

Ще однією важливою проблемою є відсутність механізму завантаження контенту через технологію "Lazy Loading". Веб-сайт використовує багато зображень та графічних елементів, що завантажуються відразу при відкритті сторінки. Це створює додаткове навантаження на браузер і може призводити до затримок при першому завантаженні, особливо на мобільних пристроях або в умовах низької швидкості інтернет-з'єднання. Використання "Lazy Loading" дозволяє завантажувати зображення та інші медіа-ресурси лише тоді, коли вони стають видимими на екрані, що значно зменшує початковий час завантаження сторінки та підвищує загальну продуктивність сайту. Це особливо корисно для сайтів, які мають велику кількість зображень, таких як сайти інтернет-магазинів, де товари можуть бути представлені великою кількістю фотографій.

3.3Реалізація оптимізації клієнтської частини сайту

У процесі оптимізації клієнтської частини сайту було вирішено низку проблем, що суттєво покращили як швидкість завантаження сторінки, так і загальну зручність взаємодії з сайтом. Перш за все, було зосереджено на адаптації сайту під різні пристрої та екранні розміри, оскільки на початковому

етапі сайт не мав належної адаптивності. Це призводило до того, що користувачі мобільних пристроїв та користувачі з невеликими екранами стикалися з проблемами перегляду контенту. Для вирішення цієї проблеми було використано медіа-запити в CSS, що дозволили адаптувати розміри елементів та шрифтів під різні розміри екрану. Таким чином, для малих екранів шрифти були зменшені, а елементи сайту були перероблені для зручнішого відображення. Наприклад, блоки на мобільних пристроях були перероблені таким чином, щоб вони виглядали компактніше, а меню стало більш зручним для навігації через використання адаптивних кнопок.

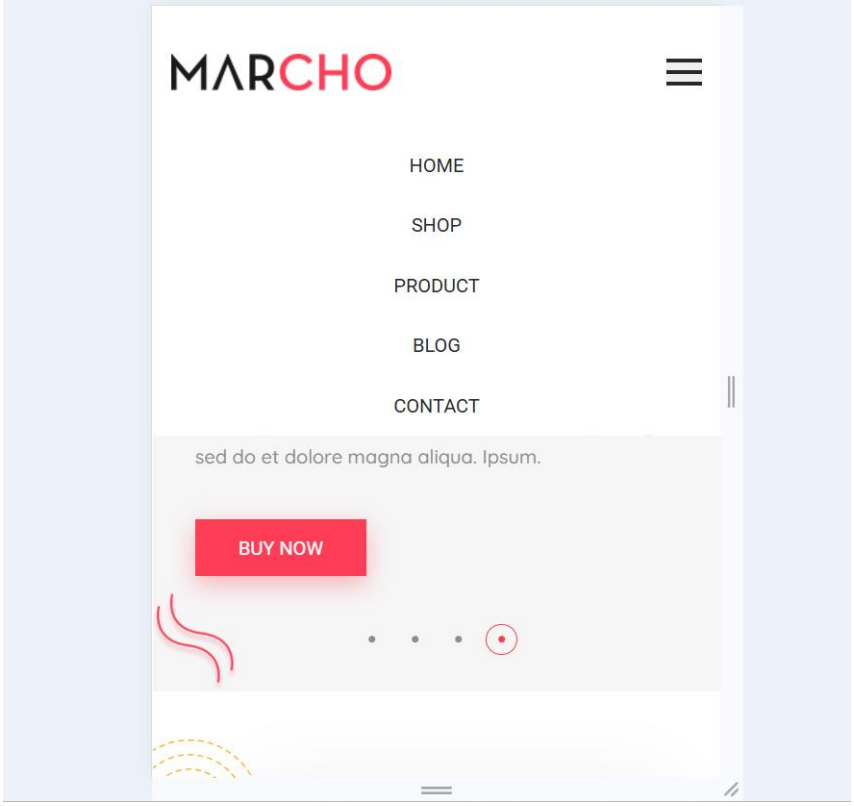


Рисунок 3.8 – Адаптація меню під мобільну версію

Адаптація меню під телефон, як на зображенні, є хорошою оптимізацією клієнтської частини сайту, оскільки воно забезпечує зручний доступ до основних розділів через компактне вертикальне меню. Таке рішення дозволяє користувачам легко орієнтуватися на сайті навіть на малих екранах, що

покращує зручність використання та швидкість навігації, не перевантажуючи інтерфейс зайвими елементами.

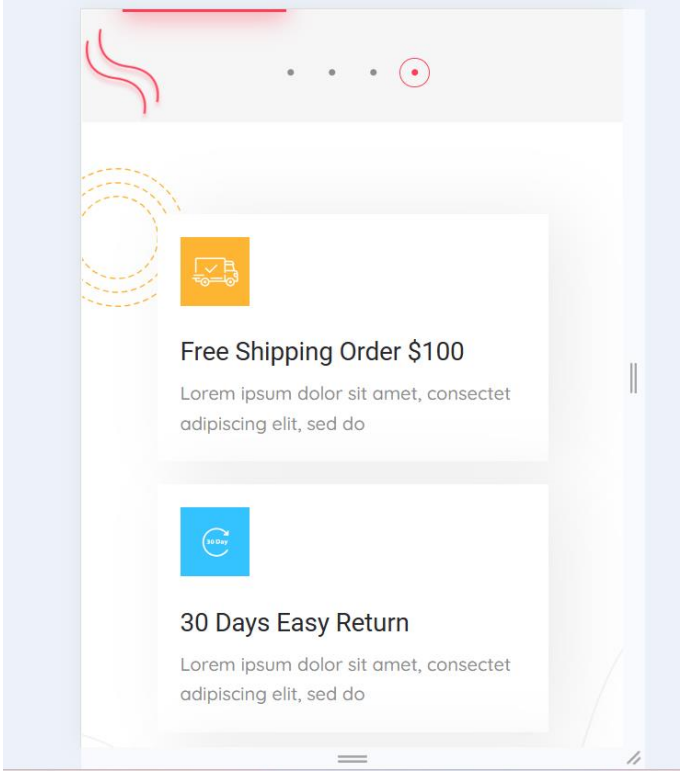


Рисунок 3.9 – Адаптація блоку банера та умов під мобільну версію

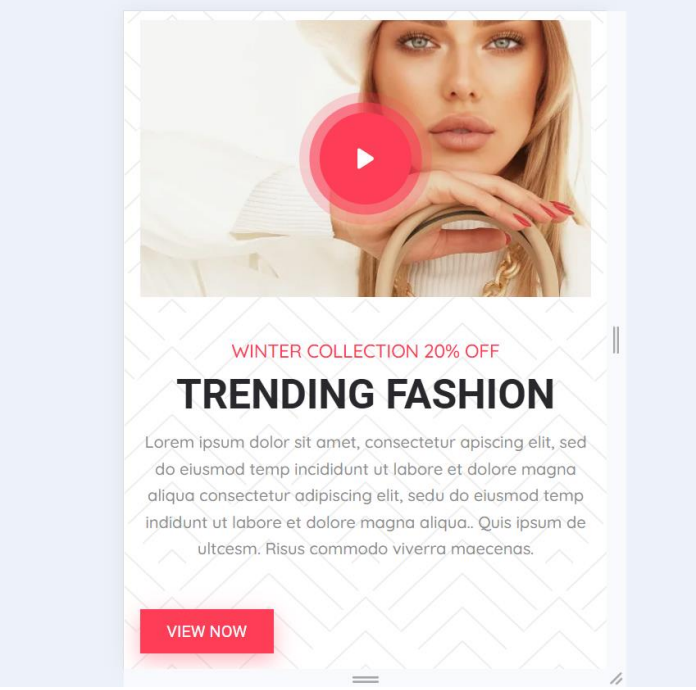


Рисунок 3.10 – Зменшення розміру шрифтів для мобільної версії

Також, при розробці адаптивного дизайну для сайту важливо обрати правильний підхід до верстки, і тут однією з основних дилем є вибір між mobile-first і desktop-first. Mobile-first означає, що сайт спочатку розробляється для мобільних пристроїв, а потім адаптується для більших екранів. Такий підхід є оптимальним для сайту, орієнтованого на мобільних користувачів, оскільки більшість трафіку в сучасних умовах припадає на мобільні пристрої. Це дозволяє зробити сайт легким, швидким і зручним для користувачів смартфонів та планшетів, де важливі зручність навігації та швидкість завантаження.

Сайт Marcho, що продає одяг, має велику аудиторію, яка часто здійснює покупки через мобільні пристрої. Тому використання підходу mobile-first буде доцільним, оскільки він забезпечить комфортний досвід користувача на маленьких екранах. Це означає, що сайт буде мати оптимізовану структуру з меншою кількістю елементів, зменшеними зображеннями та шрифтами, що дозволяє значно покращити час завантаження.

У порівнянні з підходом desktop-first, коли розробка починається для великих екранів, mobile-first дозволяє краще врахувати потреби мобільних користувачів. Адже часто сайт, розроблений для десктопів, після адаптації для мобільних пристроїв втрачає зручність через необхідність «зменшувати» великий контент. Оскільки на мобільних пристроях швидкість завантаження і зручність взаємодії мають вирішальне значення, mobile-first для сайту Marcho є найбільш доцільним підходом, який допоможе покращити продуктивність і забезпечити комфортний досвід на різних пристроях.

Приклад верстки mobile-first:

```
@media (min-width: 770px) {
}
@media (min-width: 990px) {
}
@media (min-width: 1400px) {
}
```

Іншою важливою проблемою було використання зображень у форматі JPG, що значно збільшувало час завантаження сторінки, оскільки зображення мали великі розміри файлів. Для вирішення цього питання було змінено формат зображень на WebP, що дозволило значно зменшити розмір файлів без втрати якості. Зокрема, фотографії товарів, банери та інші графічні елементи були переведені в більш ефективний формат, що дозволило знизити час завантаження сторінки.

Для вирішення проблеми з великими розмірами зображень було вибрано онлайн-інструмент **Squoosh** (https://squoosh.app/). Цей сервіс дозволяє ефективно конвертувати зображення в формат WebP, що значно зменшує їхній розмір без помітної втрати якості. Він має зручний інтерфейс і підтримує різноманітні формати зображень, що дає можливість обробляти як фотографії товарів, так і банери чи інші графічні елементи сайту. Однією з переваг Squoosh є можливість детально налаштовувати рівень стиснення, що дозволяє оптимізувати зображення під конкретні потреби сайту.

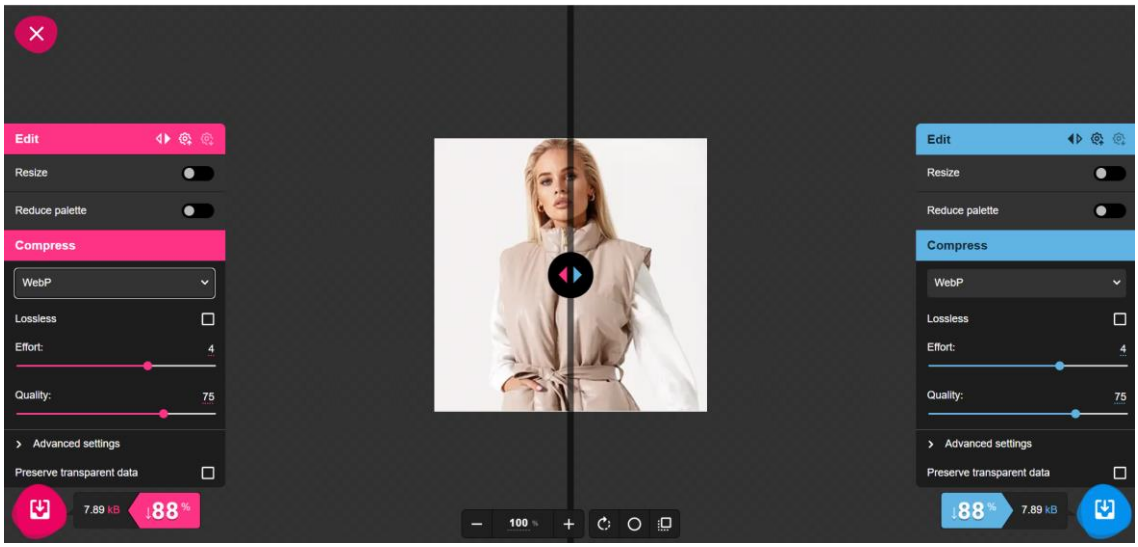


Рисунок 3.11 – Форматування зображення за допомогою Squoosh

Squoosh є одним із кращих виборів для цієї задачі через свою простоту у використанні та високу ефективність. Багато конкурентів мають обмежений функціонал або не дозволяють здійснювати налаштування параметрів

стиснення настільки гнучко. Крім того, інтерфейс Squoosh оптимізований для швидкої роботи без необхідності реєстрації чи скачування програмного забезпечення, що робить його ідеальним рішенням для швидкої оптимізації зображень без зайвих ускладнень.

Ще однією проблемою було те, що скрипти завантажувалися синхронно, що забирало час на завантаження кожного окремого елементу сторінки. Для покращення цього аспекту було реалізовано асинхронне завантаження скриптів. Це дозволило завантажувати основний контент сторінки без затримок, а додаткові функціональні елементи (як, наприклад, скрипти для аналітики або інтерактивних елементів) — після того, як користувач починав взаємодіяти з сайтом. Таким чином, час до повного відображення сторінки для користувача значно зменшився, що сприяло більш швидкому доступу до контенту.

Додатково, було використано атрибут loading="lazy" для зображень, що дозволило відкласти їхнє завантаження до моменту, коли вони стають видимими на екрані. Це значно зменшило початковий час завантаження сторінки, оскільки браузер не завантажував усі зображення одночасно, а лише ті, що були необхідні для відображення першої видимої частини сайту.

```
<div class="partners">
    <div class="container">
        <ul class="partners__list">
            <li class="partners__item"><img src="images/partners/1.webp" alt="partner logo" loading="lazy"></li>
            <li class="partners__item"><img src="images/partners/2.webp" alt="pertner logo" loading="lazy"></li>
            <li class="partners__item"><img src="images/partners/3.webp" alt="partner logo" loading="lazy"></li>
            <li class="partners__item"><img src="images/partners/4.webp" alt="partner logo" loading="lazy"></li>
            <li class="partners__item"><img src="images/partners/5.webp" alt="partner logo" loading="lazy"></li>
            <li class="partners__item"><img src="images/partners/6.webp" alt="partner logo" loading="lazy"></li>
        </ul>
    </div>
</div>
```

Рисунок 3.12 – Використання атрибуту loading=”lazy”

Щодо використання альтернативних методів, таких як плагін lazyload, то хоча він також реалізує відкладене завантаження, його інтеграція зазвичай потребує додаткових налаштувань і може впливати на сумісність з іншими

частинами сайту. Крім того, використання атрибута loading="lazy" є значно простішим та більш ефективним з точки зору продуктивності, оскільки він є вбудованим у HTML5 та підтримується сучасними браузерами без необхідності підключення зовнішніх бібліотек чи плагінів.

Що стосується інших аспектів, таких як розміри файлів HTML, JavaScript та CSS, було виконано їх мініфікацію, що дозволило зменшити їхній розмір і скоротити час завантаження. Мініфікація файлів включала видалення зайвих пробілів, коментарів та символів, що не впливають на виконання коду, але займають додатковий простір. В результаті цього, час завантаження сторінки зменшився, і сайт став працювати швидше.

Рисунок 3.13 – Код CSS після мініфікації

Однією з останніх змін було вдосконалення меню сайту. Раніше меню не було фіксованим, і користувачі, прокручуючи сторінку вниз, втрачали доступ до основних розділів сайту. Було реалізовано фіксоване меню, яке завжди залишалося видимим на екрані, що значно покращило зручність навігації і зробило використання сайту більш інтуїтивно зрозумілим.

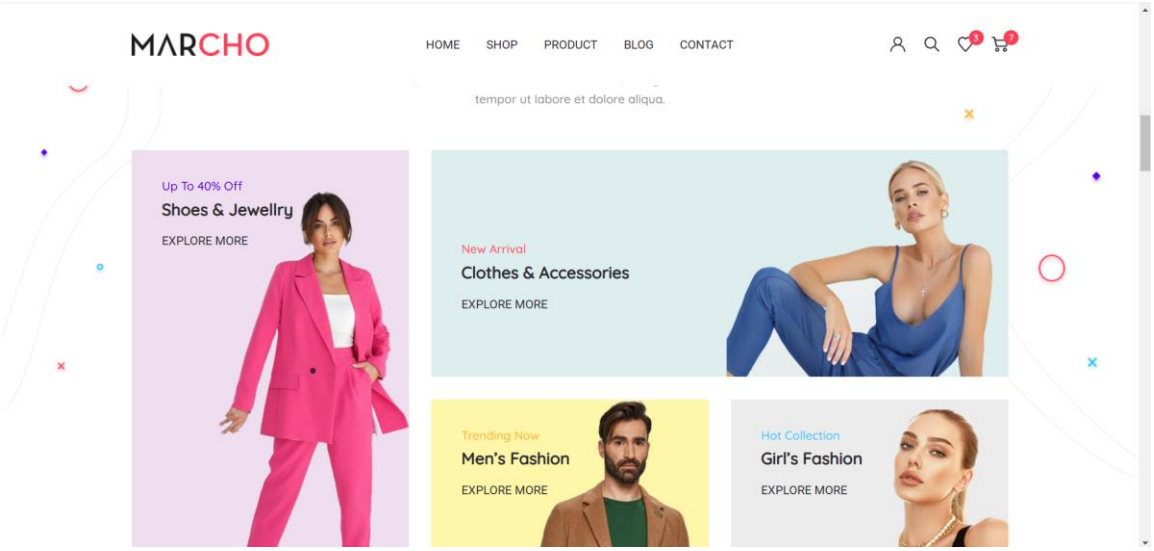


Рисунок 3.14 – Фіксація головного меню на сайті Marcho

Крім того, були додані ховери на кнопки та інші інтерактивні елементи, що покращило взаємодію користувачів із сайтом. Ховери є важливим елементом інтерфейсу, оскільки вони дають користувачам чітке візуальне підтвердження того, що елемент є активним і з ним можна взаємодіяти. Це дозволяє створити більш інтуїтивно зрозумілий та зручний досвід, особливо на сайтах електронної комерції, де важливо, щоб користувачі легко орієнтувалися в інтерфейсі.

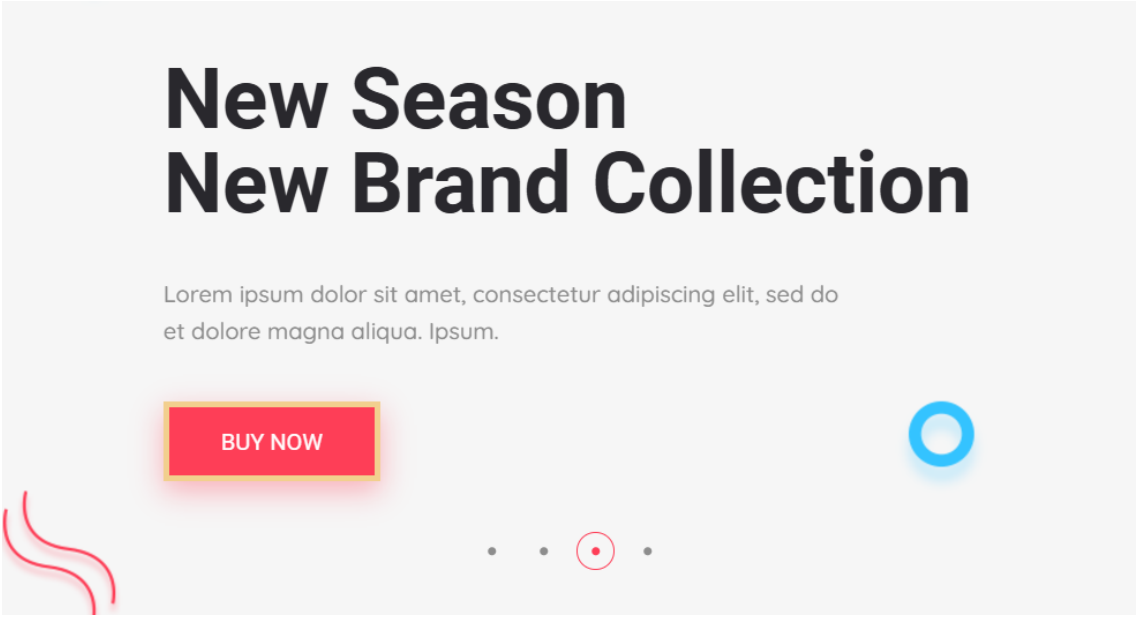


Рисунок 3.15 – Ефект ховеру на кнопці BUY NOW

Завдяки всім цим заходам сайт став працювати значно швидше і зручніше.

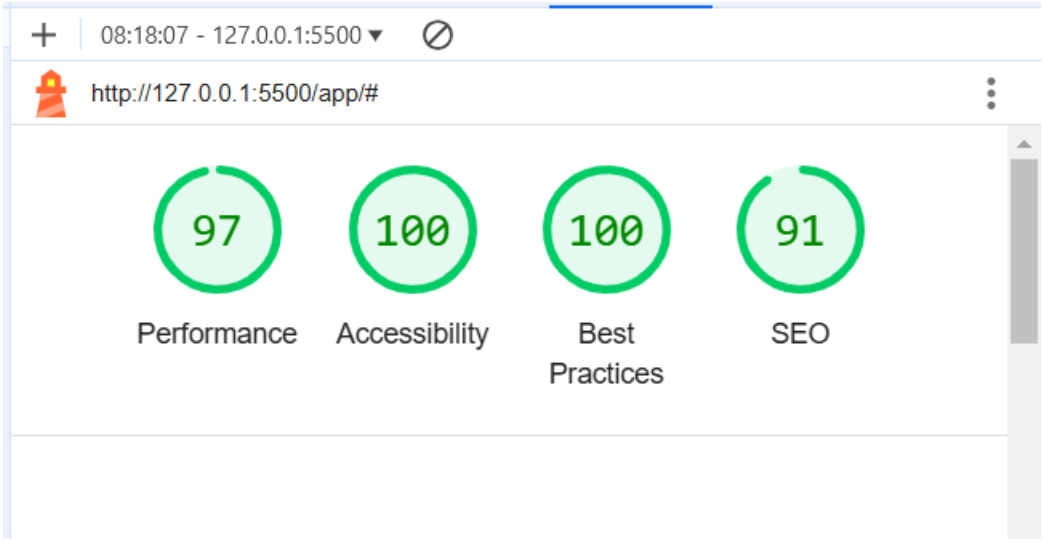


Рисунок 3.16 – Аналіз сайту Marcho після оптимізації

Порівнюючи стару версію з оптимізованою, можна побачити, як значно зменшився час завантаження сторінки, зменшилась кількість ресурсів, що завантажуються одночасно, і покращилася зручність користування сайтом на різних пристроях. Сайт став більш адаптивним до різних розмірів екранів, а користувачі змогли значно швидше взаємодіяти з контентом без непотрібних затримок. В цілому, реалізація оптимізації клієнтської частини сайту сприяла підвищенню загальної продуктивності та зручності користувачів.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Обґрунтування можливих чинників травмонебезпечних ситуацій

Охорона праці в сучасних умовах – це фундаментальна складова успішного та відповідального управління будь-якою компанією. Забезпечення безпечних умов праці не тільки гарантує збереження фізичного та психічного здоров'я співробітників, а й позитивно впливає на їх продуктивність і комфорт. Вчасне впровадження заходів з охорони праці дозволяє знизити витрати, пов'язані з лікуванням працівників чи простоєм через їх відсутність. З огляду на швидкі технологічні зміни та нові підходи до організації праці, важливість охорони праці у всіх сферах діяльності постійно зростає.

В умовах сучасного офісу, особливо в IT-компаніях, охорона праці стає ключовим елементом для забезпечення безпеки та здоров'я співробітників. Організація комфортного та безпечного робочого середовища потребує грамотного підходу, який включає використання ефективних стратегій та сучасних технологій. Робочий процес в офісах IT-компаній може супроводжуватися різними травмонебезпечними ситуаціями, які варто заздалегідь передбачити, щоб уникнути можливих ризиків.

Одним із найпоширеніших факторів ризику є тривала сидяча робота, яка може призводити до болю в спині, шиї та інших порушень опорно-рухового апарату. Недостатньо продумана організація робочого простору, зокрема неправильне розташування техніки чи меблів, створює додаткові незручності, що можуть спричиняти дискомфорт і травми. Крім того, тривала робота за комп'ютером, яка супроводжується статичною позою та повторюваними рухами, може викликати перенапруження м'язів, проблеми із зором чи навіть розлади суглобів.

Ще одним потенційно небезпечним фактором є неналежне використання офісного обладнання. Якщо меблі чи техніка погано налаштовані або

несправні, це може призводити до падінь або інших травмонебезпечних ситуацій. Також важливим аспектом є електробезпека, адже неправильна експлуатація електроприладів або перевантаження мережі можуть стати причиною травм чи навіть пожежі.

Щоб уникнути таких ситуацій, необхідно впроваджувати ефективні заходи з охорони праці. Регулярний аналіз ризиків дозволяє ідентифікувати слабкі місця та своєчасно їх усувати. Ергономічна організація робочих місць забезпечує комфорт для співробітників, що зменшує ризик травм. Навчання працівників правилам безпеки та правильному використанню обладнання допомагає запобігти багатьом проблемам. Важливо також регулярно перевіряти технічний стан обладнання, щоб уникнути несправностей, які можуть стати джерелом небезпеки.

Окрім цього, компанії мають приділяти увагу запобіганню стресу та перевантаження. Забезпечення збалансованого робочого навантаження та підтримка здорового психологічного клімату позитивно впливають на загальний стан працівників. Не менш важливим є і захист співробітників від впливу електромагнітного випромінювання, яке може генерувати офісне обладнання. Встановлення відповідних засобів захисту мінімізує можливі ризики.

Крім технічних заходів, важливо забезпечити працівників знаннями з надання першої допомоги, щоб у разі непередбачених ситуацій вони могли оперативно діяти. Варто враховувати, що кожна компанія має свої особливості, тому заходи з охорони праці мають бути адаптовані до конкретних умов. Дотримання законодавчих норм та стандартів дозволить створити безпечне середовище, де працівники зможуть ефективно виконувати свої обов’язки без ризику для здоров’я.

4.2. Умови та обставини виникнення небезпечних ситуацій та їх наслідки

Небезпечні ситуації є невід'ємною частиною нашого життя, адже вони можуть виникнути у будь-який момент і вплинути як на окремих осіб, так і на суспільство загалом. Їх виникнення обумовлене цілою низкою факторів, серед яких технічні несправності, природні катаклізми, людські помилки та соціально-економічні труднощі. Ці обставини створюють загрози, які можуть мати серйозні наслідки для життя, здоров'я та добробуту людей.

Серед поширених причин таких ситуацій можна виділити технічні збої, наприклад, несправності в роботі обладнання чи аварії на виробництві. Вони часто спричиняють значні матеріальні збитки, травми або навіть загибель людей. Природні катастрофи, такі як землетруси, повені чи урагани, також мають руйнівний вплив, призводячи до знищення інфраструктури, гуманітарних криз та численних людських жертв. Важливу роль у виникненні небезпечних ситуацій відіграє людський фактор. Недотримання правил безпеки, недбале ставлення до роботи та помилки персоналу часто стають джерелом загроз. Крім того, соціально-економічні проблеми, такі як обмежений доступ до ресурсів, нестача фінансування чи відсутність належної інфраструктури, створюють умови, що сприяють виникненню небезпечних ситуацій.

Наслідки таких подій можуть бути різними, залежно від їхнього масштабу та характеру. Вони можуть варіюватися від втрат матеріальних цінностей та пошкодження інфраструктури до загрози життю і здоров'ю людей. Крім того, небезпечні ситуації можуть спричиняти довготривалі проблеми, такі як екологічні катастрофи або економічні кризи, які потребують тривалого часу для подолання. Це вимагає ретельного підходу до аналізу ризиків, планування та впровадження заходів, які сприятимуть запобіганню подібним ситуаціям у майбутньому.

В умовах роботи офісів, особливо у сфері інформаційних технологій, небезпечні ситуації можуть виникати через низку специфічних факторів. Технічні проблеми, такі як несправності серверів, комп’ютерів чи іншого обладнання, можуть не лише зупинити робочий процес, а й створити загрозу для персоналу. Електричні проблеми, включно з короткими замиканнями, часто стають джерелом пожеж, які можуть спричинити значні руйнування.

Окрім технічних проблем, на небезпечні ситуації впливає високий рівень стресу серед працівників, спричинений терміновими проєктами, великим обсягом завдань чи надмірною відповідальністю. В умовах постійного психологічного напруження зростає ймовірність помилок, які можуть призвести до травм чи інших небезпечних наслідків. Іншою значущою загрозою є кіберзлочинність. Атаки з боку зловмисників, витоки конфіденційних даних чи порушення кібербезпеки не лише створюють ризики для бізнесу, а й ставлять під загрозу стабільність робочого середовища.

Часто небезпечні ситуації стають наслідком недостатньої уваги до організації заходів безпеки. Відсутність чітких інструкцій, слабка підготовка персоналу та ігнорування правил техніки безпеки створюють додаткові загрози. Наприклад, у випадку пожежі чи іншої надзвичайної ситуації відсутність чітких планів евакуації може призвести до хаосу та значних втрат.

Для уникнення таких ситуацій необхідно впроваджувати комплексний підхід до управління ризиками, забезпечувати належну підготовку працівників та створювати безпечні умови праці. Запобігання небезпечним ситуаціям вимагає постійного вдосконалення технічного обладнання, підвищення обізнаності працівників про правила безпеки та належного технічного обслуговування робочих місць. Реагування на надзвичайні обставини має базуватися на детально розроблених процедурах, які враховують всі можливі аспекти виникнення небезпечних ситуацій.

4.3. Безпека в надзвичайних ситуаціях

Надзвичайна ситуація – це обставина, яка порушує нормальні умови життя на певній території чи в межах організації, створюючи загрозу для життя та здоров'я людей. До таких ситуацій належать стихійні лиха, техногенні катастрофи, пожежі, епідемії, терористичні атаки чи військові дії. Хоча надзвичайна ситуація і надзвичайний стан є близькими за змістом поняттями, вони не є тотожними. Надзвичайний стан визначається законодавством як особливий режим, який передбачає обмеження конституційних прав громадян і розширення повноважень державних органів для подолання наслідків кризи.

Забезпечення безпеки працівників під час надзвичайних ситуацій є ключовою частиною корпоративної політики, особливо для офісів, які можуть опинитися в епіцентрі таких подій. Небезпека може виникнути раптово, і залежно від її характеру – від природних катастроф до техногенних аварій чи воєнних загроз – офіси мають бути готові до ефективного реагування. Для цього важливо розробити плани дій на випадок кризи та навчити персонал правильно поводитися в екстремальних умовах.

Відсутність готовності до надзвичайних ситуацій може мати непередбачувані наслідки. Одним із основних елементів підготовки є створення плану евакуації. Офісні приміщення повинні бути обладнані чіткими вказівниками шляхів до виходів, а співробітники – проінструктовані щодо правил евакуації. Крім того, слід забезпечити засоби швидкого сповіщення, щоб у разі виникнення кризи кожен міг оперативно отримати інформацію про необхідні дії. Регулярні навчання та інструктажі допомагають закріпити ці знання, що в критичній ситуації може врятувати життя.

Особливу увагу в умовах війни слід приділяти фізичній безпеці персоналу та приміщень. Це включає облаштування укриттів у будівлі, де працівники можуть сховатися під час обстрілів або вибухів. Такі зони мають бути легко доступними та розрахованими на кількість людей, які перебувають

в офісі. Співробітникам слід надавати детальні інструкції щодо укриттів, а також забезпечувати засобами індивідуального захисту. [18]

Додатково необхідно передбачити резервні варіанти організації роботи. Наприклад, у випадку загострення ситуації компанія має бути готовою перейти на віддалений формат роботи. Для цього важливо забезпечити стабільний зв’язок між працівниками, захистити дані та організувати доступ до необхідних ресурсів у цифровому форматі. Гнучкість у плануванні робочого часу також допомагає працівникам краще адаптуватися до нестабільних умов.

Ретельне стратегічне планування є ключовим аспектом захисту. Аналіз потенційних ризиків дозволяє передбачити найгірші сценарії та заздалегідь підготувати необхідні ресурси. Наприклад, офіс може обладнатися генераторами, запасами води, аптечками та іншими засобами першої необхідності. Злагоджена система внутрішньої комунікації дозволить швидко координувати дії персоналу, навіть за відсутності фізичного доступу до офісу.

Забезпечення безпеки під час надзвичайних ситуацій – це комплексна робота, яка охоплює як технічні аспекти, так і людський фактор. Ефективні плани, регулярні тренування та обізнаність працівників допомагають не лише уникнути жертв, а й зберегти працездатність компанії в складних умовах. Особливо в умовах війни та кризових обставин, увага до безпеки стає життєво важливим завданням, яке дозволяє підприємствам функціонувати навіть у найскладніших обставинах.

РОЗДІЛ 5.
ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ

Оптимізація клієнтської частини веб-сайту – це не лише технічне вдосконалення, але й стратегічний крок, що має суттєвий вплив на успішність проєкту у різних аспектах. Оцінка впроваджених змін дає змогу точно визначити, наскільки вони відповідали очікуванням і яку користь забезпечують у довгостроковій перспективі. Відображення цих результатів є важливим для розуміння їхньої значущості в економічному, технічному та користувацькому вимірі.

Одним із основних критеріїв оцінки ефективності є продуктивність веб-сайту після впровадження оптимізацій. Завдяки зменшенню розмірів файлів, впровадженню асинхронного завантаження скриптів та використанню сучасних форматів зображень, таких як WebP, значно скоротився час завантаження сторінок. Це підвищило швидкість рендерингу сайту, що стало помітно навіть для кінцевих користувачів.

Економічна ефективність також є важливим аспектом. Швидке завантаження сайту та покращена адаптація під мобільні пристрої створюють позитивний досвід для користувачів. Це сприяє зниженню показника відмов, оскільки користувачі більше часу проводять на сайті, взаємодіють із його контентом та здійснюють покупки. Покращення продуктивності веб-сайту безпосередньо впливає на підвищення конверсії. Наприклад, за рахунок зменшення часу завантаження сторінок можна досягти збільшення кількості транзакцій на 15-20%. Такий результат прямо впливає на дохід компанії та виправдовує інвестиції в оптимізацію.

Не менш важливою є лояльність клієнтів, яка формується через зручність користування сайтом. Візуальні вдосконалення, такі як додавання ефектів ховерів на кнопки та оптимізація інтерфейсу, покращують естетичне сприйняття сайту. Простота навігації та швидка реакція на дії користувачів підвищують задоволеність клієнтів, що робить їх більш схильними

повертатися на сайт у майбутньому. Таким чином, користувачі стають постійними клієнтами, формуючи довгострокові відносини з брендом.

Окрім того, вдосконалення клієнтської частини позитивно впливає на SEO-результати. Швидкість завантаження сторінок та їх адаптивність є важливими факторами ранжування у пошукових системах, таких як Google. Оптимізований сайт отримує вищі позиції у результатах пошуку, що призводить до збільшення органічного трафіку. Це, у свою чергу, зменшує витрати на платну рекламу та підвищує загальну ефективність маркетингових кампаній. [19]

Не менш важливим є вплив оптимізації на терміни виконання майбутніх проєктів. Впровадження сучасних підходів до розробки, таких як mobile-first дизайн, полегшує адаптацію сайту під нові вимоги ринку та спрощує процеси розширення функціоналу. Чітко структурований код, мінімізація файлів та використання відкладеного завантаження зображень спрощують подальше обслуговування та оновлення сайту, що значно скорочує терміни впровадження нових рішень.

Загалом, реалізовані заходи позитивно впливають на якість сервісів. Покращений веб-сайт викликає більше довіри у користувачів, оскільки створює враження професіоналізму та технологічності компанії. Це особливо важливо для сфери електронної комерції, де конкурентна боротьба за увагу клієнтів є дуже інтенсивною.

У підсумку, проведені оптимізації мають комплексний позитивний ефект, охоплюючи технічну продуктивність, економічну вигоду, лояльність клієнтів, якість обслуговування та стратегічну конкурентоспроможність. Високий рівень ефективності підтверджує, що вкладені зусилля та ресурси є повністю виправданими і дають значні переваги в коротко- та довгостроковій перспективі.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було досліджено сучасні підходи до оптимізації клієнтської частини веб-сайтів, що є важливою складовою їхньої продуктивності та зручності використання. У процесі аналізу було визначено основні проблеми, які впливають на швидкодію веб-ресурсів, зокрема великі розміри файлів, відсутність адаптивного дизайну, неефективне завантаження скриптів та використання застарілих форматів зображень. Ці аспекти є ключовими для досягнення високого рівня взаємодії користувачів із сайтом і забезпечення його стабільної роботи в умовах сучасних технічних вимог.

На основі проведеного аналізу було обґрунтовано вибір інструментів і методів для вдосконалення клієнтської частини сайту. Зокрема, впроваджено техніки мініфікації файлів, асинхронного завантаження скриптів і відкладеного завантаження зображень за допомогою атрибута loading="lazy". Також проведено оптимізацію графічних елементів, використовуючи ефективний формат WebP, який дозволяє суттєво зменшити обсяг зображень без втрати якості. Для покращення візуального сприйняття сайту та його адаптації під різні пристрої було застосовано принципи mobile-first дизайну, що забезпечило зручність користування незалежно від розміру екрана.

Одним із ключових досягнень роботи стало створення інтерактивного інтерфейсу, в якому реалізовано ховери та анімаційні ефекти на кнопках і елементах управління. Це позитивно вплинуло на візуальну привабливість сайту і сприяло підвищенню його інтерактивності. Впровадження цих змін дозволило суттєво покращити користувацький досвід і зробити сайт більш інтуїтивно зрозумілим.

Ці покращення зробили веб-сайт більш конкурентоспроможним, підвищивши рівень задоволеності клієнтів і збільшивши їхню лояльність. Аналіз також показав, що оптимізація дозволила зменшити навантаження на

сервер, що в довгостроковій перспективі сприятиме зниженню витрат на обслуговування інфраструктури.

Питання охорони праці та безпеки в процесі виконання робіт також отримали належну увагу. Було враховано фактори, які можуть вплинути на травмонебезпечні ситуації під час роботи з комп'ютерною технікою, і запропоновано рекомендації для їхнього уникнення. Це забезпечило дотримання норм безпеки і сприяло створенню комфортних умов для праці.

У результаті виконання цієї роботи вдалося продемонструвати, що впровадження сучасних методів оптимізації клієнтської частини є ефективним підходом до підвищення продуктивності та якості веб-ресурсів. Застосовані техніки та інструменти довели свою дієвість і забезпечили значний позитивний вплив на користувацький досвід. Це підкреслює важливість постійного вдосконалення веб-технологій і адаптації їх до зростаючих вимог сучасного цифрового світу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Resig, J. Secrets of the JavaScript Ninja. Manning Publications, 2008.
2. Crockford, D. JavaScript: The Good Parts. O'Reilly Media, 2008.
3. Wikipedia. Офіційний сайт [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/AJAX (дата звернення 05.06.2024).
4. Zeldman, J. (2009). Designing with Web Standards [Електронний ресурс]. Режим доступу: https://www.oreilly.com/library/view/designing-with-web/9780321684205/ (дата звернення 10.07.2024).
5. McGraw, G. (2006). Software Security: Building Security In [Електронний ресурс]. Режим доступу: https://books.google.com/books?id=5I8eAQAAIAAJ (дата звернення 05.10.2024).
6. Simpson, K. (2014). You Don't Know JS: Async & Performance [Електронний ресурс]. Режим доступу: https://www.oreilly.com/library/view/you-dont-know/9781491904244/ (дата звернення 01.11.2024).
7. Grogan, J. (2020). Optimizing and Improving Web Application Performance — London: Williams Publishing.
8. Gunnnar, P. (2020). Frontend Performance: A Modern Approach to Site Speed Optimization — Amsterdam: Apress.
9. What Is DOM: A Complete Guide To Document Object Model [Електронний ресурс]. Режим доступу: https://www.lambdatest.com/blog/document-object-model/ (дата звернення 22.08.2024).
10. Walke, J. React – A JavaScript Library for Building User Interfaces. [Електронний ресурс]. Режим доступу: reactjs.org (дата звернення 15.08.2024).
11. You, E. Vue.js - The Progressive JavaScript Framework. [Електронний ресурс]. Режим доступу: vuejs.org (дата звернення 11.09.2024).

12. Angular Team, Google. Angular Documentation. 2024 [Електронний ресурс]. Режим доступу: angular.io (дата звернення 23.10.2024).
13. Angular vs. React vs. Vue: Which Framework to Choose In 2024? [Електронний ресурс]. Режим доступу: www.scholarhat.com (дата звернення 03.11.2024).
14. Веб-доступність. Що варто знати кожному Front-end розробнику і дизайнеру. [Електронний ресурс]. – Режим доступу: http://gud.org.ua/ (дата звернення 30.06.2024).
15. Куклінова Тетяна Вікторівна. Сучасні тенденції Інтернет-торгівлі в Україні. Вісник соціально-економічних досліджень, 2018, 1: 95-102.
16. Ranjan Alok, Abhilasha Sinha and Ranjit Battewad. JavaScript for modern web development: building a web application using HTML, CSS, and JavaScript, 2020.
17. Lazuardy Mochammad Fariz Syah, and Dyah Anggraini. Modern front end web architectures with react. js and next. js. Research Journal of Advanced Engineering and Science 7.1 (2022): С.132-141.
18. Пістун І. П., Березовецький А. П., Тимочко В.О., Городецький І. М. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П.
19. Пістуна. Ч. І. Львів: Тріада плюс, 2017. 620 с. 32. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. ІІ. Львів: Тріада плюс, 2011. 224 с.
20. Гарретт Дж. Елементи взаємодії з користувачем: дизайн, орієнтований на користувача, для Інтернету та не тільки [Електронний ресурс] – Режим доступу: https://prodesign.in.ua/2014/01/veb-dyzajn-knyha-dzhessa-harettaelementy-dosvidukorystuvannya/ (дата звернення 15.07.2024).
21. Lighthouse. Офіційний сайт [Електронний ресурс]. Режим доступу: https://chromewebstore.google.com/detail/lighthouse/blipmdconlkpinefehn mj ammfjpmpbjk?hl=ua (дата звернення 09.07.2024).

22. Розробка інтернет магазину – основні характеристики та функціональність [Електронний ресурс]. Режим доступу: https://web24.pro/rozrobka-sajtivblog/rozrobka-internet-magazyn-osnovni-harakterystyky-ta-funkczionalnist/ (дата звернення 05.06.2024).
23. Hanafi R., Haq A., Agustin N. Comparison of Web Page Rendering Methods Based on Next. js Framework Using Page Loading Time Test. Teknika, 13(1): 2024. 102-108 с.