Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів рендерингу web-сайтів
для пошукової оптимізації сайтів (SEO)
(тема)

Виконав:
студент 2 курсу, групи ІПЗм-22-3

Брухтій С.С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. каф.Ревенчук І.А.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Брухтію Сергію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження методів рендерингу web-сайтів для пошукової оптимізації сайтів (SEO)»
Затверджена наказом по університету від 29.03. 2024р. № 230 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2024
3. Вихідні дані до роботи Дослідження сучасних методів рендерингу, класифікація, практична перевірка, React, NodeJS
4. Перелік питань, що потрібно опрацювати в роботі
аналіз існуючих методів рендерингу web-сайтів для пошукової оптимізації сайтів (SEO), проблеми та постановка задачі, встановлення критеріїв до програмного забезпечення для дослідження, проведення експериментів та аналіз отриманих результатів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.04.2024	*виконано*
2	Аналіз предметної галузі та постановка задачі	01.04 – 10.04.24	*виконано*
3	Аналіз та вибір алгоритмів для дослідження	10.04 – 14.04.24	*виконано*
4	Аналіз та моделювання предметної області	14.04 – 20.04.24	*виконано*
5	Планування експериментів	20.04 – 29.04.24	*виконано*
6	Програмна реалізація кожного з обраних для дослідження API	29.04 – 10.05.24	*виконано*
7	Експериментальні дослідження	10.05 – 14.05.24	*виконано*
8	Аналіз результатів експериментальних досліджень	14.05 – 16.05.24	*виконано*
10	Підготовка пояснювальної записки	16.05 – 25.05.24	*виконано*
11	Підготовка презентації та доповіді	25.05 – 30.05.24	*виконано*
12	Перевірка на плагіат	30.05 – 03.06.24	*виконано*
13	Нормоконтроль	03.06 – 08.06.24	*виконано*
14	Рецензування	08.06 – 12.06.24	*виконано*
15	Попередній захист	12.06.2024	*виконано*
16	Занесення диплома в електронний архів	15.06.2023	*виконано*
17	Допуск до захисту у зав. кафедри	20.06.2024	*виконано*

Дата видачі завдання 01.04 2024р.

Студент _____ Брухтій С.С.
(підпис)

Керівник кваліфікаційної роботи _____ доц. каф. Ревенчук І.А.
(підпис)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 66 с., 24 рис., 3 табл., 15 джерел.

РЕНДЕРІНГ ВЕБ-САЙТІВ, SEO, REACT, NODEJS

Об’єкт дослідження – процес рендерингу веб-сайтів для SEO.

Метою роботи є дослідження процесу рендерингу веб-сайтів, сучасних методів рендерингу, їх переваги та недоліки, виявлення оптимальних рішень для кожного з методів.

У результаті кваліфікаційної роботи були створені веб сторінки за допомогою яких було проведено дослідження сучасних методів рендерингу веб-сайтів для пошукової системи SEO.

WEB PAGE RENDERING, SEO, REACT, NODEJS

Object of study - the process of rendering web pages for SEO.

The aim of the work is to investigate the process of rendering websites, modern rendering methods, their advantages and disadvantages, and identifying optimal solutions for each method.

As a result of the qualification work, web pages were created to conduct research on modern rendering methods for search engine optimization (SEO).

Заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElArKhNURE.

Я, Брухтій Сергій Сергійович, студент гр. ІПЗм-22-3, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів рендерингу web-сайтів для пошукової оптимізації сайтів (SEO)», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У сучасному цифровому світі, веб-сайти стали невід'ємною частиною бізнесу, освіти та соціального взаємодії. У цьому контексті ефективність рендерингу веб-сайтів є ключовим фактором для оптимізації пошукових систем (SEO). Рендеринг веб-сайтів визначає, як пошукові системи індексують та розуміють вміст веб-сайтів, що безпосередньо впливає на їх видимість у пошукових запитах. Від цього залежить, наскільки швидко та правильно користувачі зможуть знайти потрібну інформацію у мережі.

З розвитком і активним використанням динамічних веб-технологій, включаючи JavaScript, AJAX і сучасні фреймворки, важливо забезпечити ефективний рендеринг для SEO.

Динамічний контент, який створюється на стороні клієнта, може бути складним для індексації пошуковими системами, що ускладнює просування веб-сайтів. Це означає, що веб-сайти мають бути не тільки візуально привабливими для користувачів, але й легкодоступними для пошукових роботів.

Статичний рендеринг (Static Site Generation, SSG), серверний рендеринг (Server-Side Rendering, SSR) та клієнтський рендеринг (Client-Side Rendering, CSR) відіграють важливу роль у цьому процесі, кожен з них має свої переваги та недоліки у контексті SEO.

Дана робота спрямована на дослідження сучасних методів рендерингу та їх аналіз з метою дослідити сучасні методи рендерингу веб-сайтів, зосереджуючись на їх впливі на SEO. Метою дослідження є знаходження оптимального та продуктивного вибору типу рендерингу при певних умовах. Вибір оптимального методу рендерингу є важливим для досягнення високих показників продуктивності та ефективності веб-сайтів, що забезпечить їх конкурентоспроможність у пошукових системах.

1 АНАЛІЗ ПРОБЛЕМНОЇ ГАЛУЗІ

1.1 Аналіз пошукової оптимізації веб сторінок

Розглянемо що таке SEO і як працює:

SEO (Search Engine Optimization) – це комплекс стратегій та методик, спрямованих на підвищення видимості веб-сайту у пошукових системах, таких як Google, Bing чи Yahoo. Мета SEO полягає в тому, щоб зробити сайт більш привабливим для пошукових систем, що, в свою чергу, може збільшити кількість органічного (неоплачуваного) трафіку на сайт.

SEO працює за допомогою ряду стратегій та технік, які допомагають підвищити ранг веб-сайту у пошуковій видачі, тим самим збільшуючи його видимість для користувачів, які шукають відповідні терміни чи фрази. Їх можна поділити на зовнішню та внутрішню оптимізацію [9].

Зовнішня оптимізація відбувається поза межами самого сайту з метою підвищення його авторитету, рейтингу та видимості в пошукових системах. Зовнішня оптимізація включає лінкбілдінг (створення зовнішніх посилань на сайт), активність у соціальних мережах, відгуки, рейтинги для залучення уваги та авторитетності до сайту.

Внутрішня оптимізація - це пошукова оптимізація самого сайту, включаючи його структуру, вміст, код та технічні аспекти.

Ось як це відбувається:

- вибір ключових слів - це релевантні ключові слова, які люди використовують при пошуку в Інтернеті. Визначивши ці ключові слова, SEO-фахівці вбудовують їх у веб-контент, заголовки, мета-описи, URL-адреси тощо;
- ооптимізація вмісту - створення високоякісного, цінного контенту, який природно включає ключові слова. Контент має бути корисним та інформативним, щоб відповідати на запити користувачів;

– побудова беклінків: Залучення зворотних посилань з інших респектабельних сайтів є важливою складовою SEO. Беклінки діють як "голоси довіри", підвищуючи авторитет сайту в очах пошукових систем;
– технічне SEO - включає оптимізацію структури сайту та його коду для поліпшення індексації та завантаження. Це означає забезпечення швидкої швидкості завантаження сторінок, мобільної адаптивності, безпечних протоколів (HTTPS) та чистого, структурованого коду.

SEO – це тривалий процес, який вимагає регулярних зусиль та адаптації до змін у алгоритмах пошукових систем та поведінці користувачів. Завдяки ефективному SEO, веб-сайт може залучати більше відповідного трафіку, підвищувати свою видимість і, як наслідок, збільшувати конверсію та доходи.

Один з важилів для ефективної внутрішньої оптимізації, а саме в технічній частині– є рендеринг веб-сайтів, який і є цілю дослідження в цій роботі.

Рендеринг веб-сайтів має велике значення для SEO, але там існують проблеми які можуть вплинути використання веб-сайту в SEO. Перша проблема - це складність індексації веб-сайтів, які використовують важкі JavaScript-фреймворки. Ці технології можуть ускладнити завдання пошукових роботів при аналізі вмісту сайту, що впливає на його видимість у пошукових системах.

Ще однією проблемою є швидкість завантаження сайтів, особливо при використанні рендерингу на стороні клієнта. Пошукові системи віддають перевагу швидким сайтам, тому потрібно знаходити баланс між функціональністю та швидкістю завантаження.

1.2 Аналіз підходів технологій для рендерингу web-сайтів

Client-Side Rendering (CSR) означає рендеринг сторінок безпосередньо в браузері за допомогою JavaScript. Вся логіка, отримання даних, шаблонізація та маршрутизація обробляються на клієнті (двигуні браузера).

Основні етапи CSR методу (див. рис. 1.1):

– компіляція коду, яка включає мініфікацію, транспіляцію і збирання ресурсів, таких як JavaScript, CSS і зображення. Після компіляції;

- згенерований код та ресурси завантажуються на сервер, що буде хостити веб-додаток;
- запит: Клієнт надсилає запит на сервер для отримання веб-сторінки. А відповідь (яка включає HTML, CSS, JS);
- після компіляції згенерований код та ресурси завантажуються на сервер, що буде хостити веб-додаток;
- отримання даних: Клієнт може також запитати додаткові дані, які зазвичай передаються у форматі JSON;
- відповідь (JSON): Сервер надсилає запитувані дані клієнту у форматі JSON;
- ініціалізація додатку: Коли клієнт отримує весь необхідний код та дані, відбувається ініціалізація веб-додатку;
- генерація сторінки: Клієнт генерує сторінки динамічно за допомогою отриманого JavaScript та даних;
- відображення: врешті, генерована сторінка відображається у вікні браузера користувача.

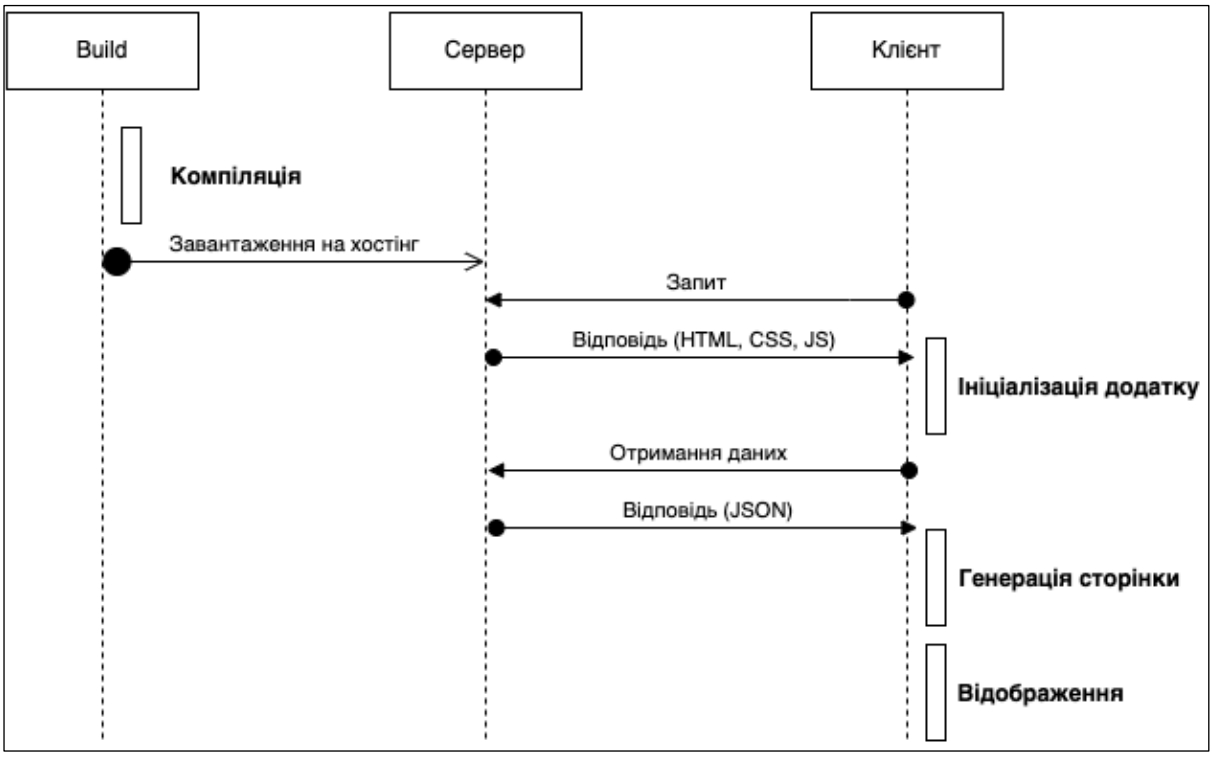

Рисунок 1.1 – Client-Side Rendering процес роботи (за даними [8])

Переваги CSR включають плавну інтерактивність і швидкі відповіді на дії користувачів після початкового завантаження, а також можливість створення багатих інтерактивних додатків. Однак, цей підхід може призводити до сповільненого початкового завантаження сторінки, оскільки весь JavaScript потрібно завантажити та виконати перед тим, як користувач побачить повноцінний вміст. Також це може створити проблеми з SEO, оскільки пошуковим системам може бути складніше індексувати динамічний вміст.

Для оптимізації CSR можуть бути використані такі техніки, як розбиття коду на частини (Code splitting [3]), ліниве завантаження (Lazy loading [4]) та інші методи поліпшення продуктивності, щоб зменшити навантаження на браузер та покращити загальний досвід користувача.

Server-Side Rendering (SSR) – це метод, в якому вміст веб-сторінки генерується на сервері перед тим, як він буде відправлений до клієнта. В SSR, сервер обробляє запити, виконує весь необхідний код, звертається до баз даних або зовнішніх API для отримання даних, і генерує кінцевий HTML відповіді.

Основні етапи SSR методу (див. рис. 1.2):

- компіляція коду;
- завантаження на веб-сервер;
- запит: клієнт надсилає запит на сервер для отримання веб-сторінки;
- отримання даних: сервер отримує запит, обробляє його і, за необхідності, отримує додаткові дані з бази даних або з інших сервісів;
- генерація сторінки: сервер генерує HTML сторінку на основі запиту та отриманих даних, часто використовуючи шаблони та серверні скрипти;
- відповідь: генерований HTML, разом з необхідними CSS та JavaScript файлами, надсилається назад клієнту;
- відображення: клієнт отримує HTML та відображає веб-сторінку, використовуючи дані з відповіді сервера;
- гідрація (якщо використовується React чи Vue): клієнтський JavaScript "гідрує" сторінку, додаючи обробники подій та активуючи додаткову інтерактивність.

Переваги SSR включають швидке відображення контенту, оскільки браузер отримує вже сформований HTML, що особливо корисно для користувачів з повільними інтернет-з'єднаннями або для оптимізації відображення на мобільних пристроях. Крім того, SSR може покращити SEO, оскільки пошукові системи мають легший доступ до статичного HTML-вмісту.

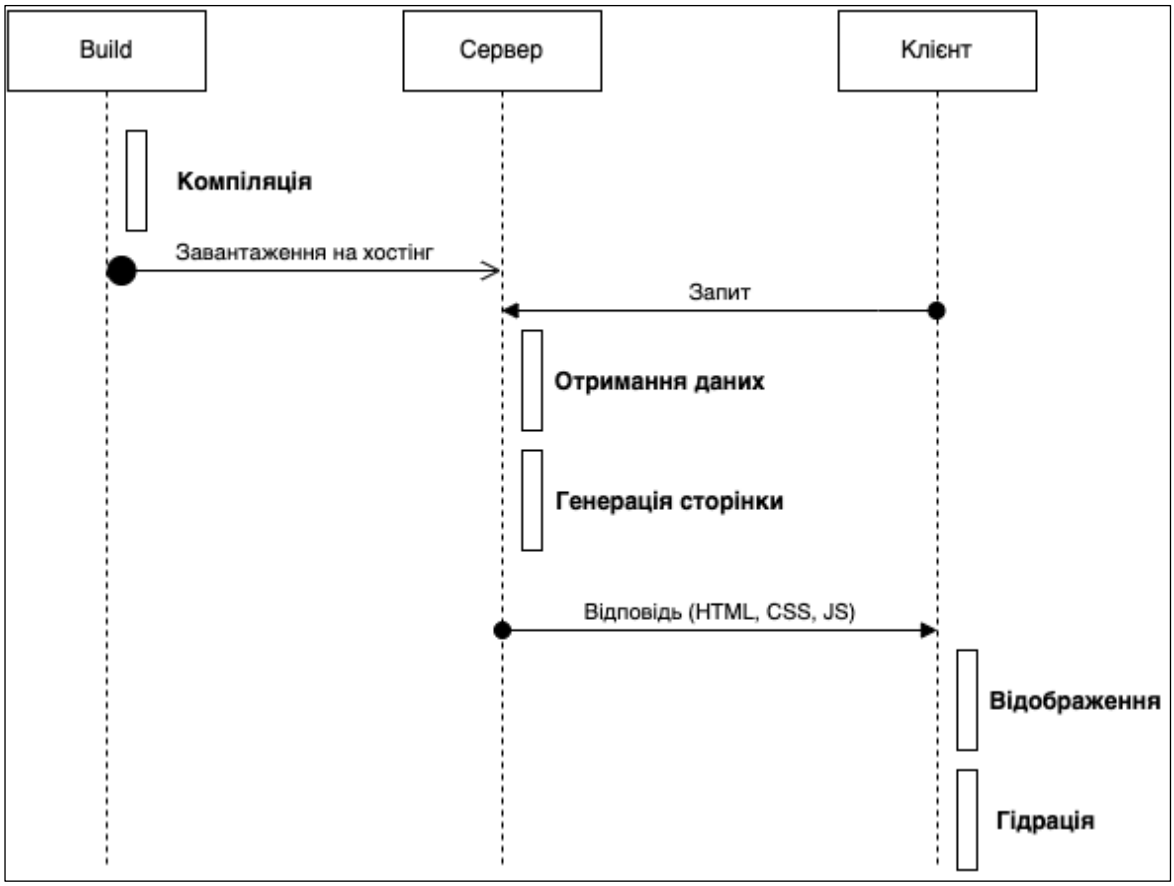

Рисунок 1.2 – Server-Side Rendering процес роботи (за даними [8])

Однак, SSR може також призвести до збільшеного навантаження на сервер, оскільки він має обробляти кожен запит і генерувати вміст при кожному запиті користувача. Також, це може обмежити або ускладнити деякі інтерактивні можливості, які легше реалізувати за допомогою Client-Side Rendering (CSR).

Static Site Generation (SSG) - це метод веб-розробки, в якому всі сторінки веб-сайту згенеровані під час процесу збірки проекту. Відмінно від динамічних сайтів, статичні сторінки не вимагають серверної обробки на момент запиту користувача. Замість цього, вони вже повністю сформовані і можуть бути розгорнуті на сервері.

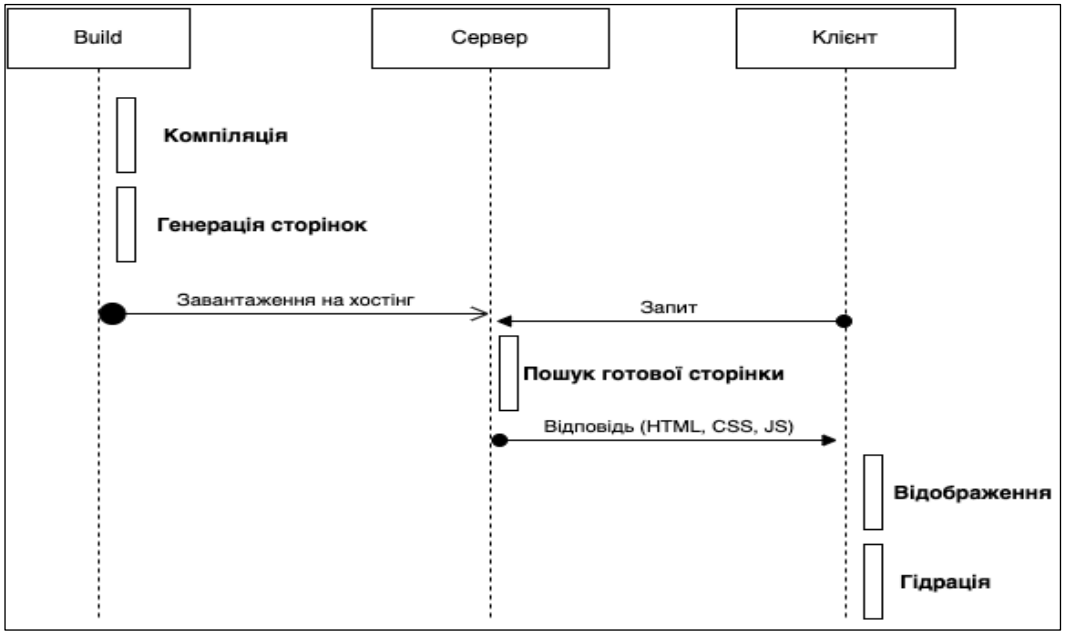

Рисунок 1.3 – Server-Side Rendering процес роботи (за даними [8])

SSG пропонує високу швидкість завантаження та поліпшену безпеку, оскільки сторінки веб-сайту генеруються заздалегідь і служать як готові HTML-файли. Це елімінує необхідність в обробці серверних запитів в реальному часі, що може значно зменшити час відгуку та потенційні точки входу для веб-атак. SSG також спрощує розгортання та хостинг, дозволяючи використовувати недорогі та швидкі рішення на базі CDN (Content delivery network [5]). З точки зору SEO, статичні сайти часто мають перевагу, оскільки вони легко індексуються пошуковими системами.

Недоліками SSG є динамічний вміст та інтерактивність. Сторінки, що часто оновлюються або містять велику кількість індивідуалізованого вмісту, можуть зіткнутися з труднощами, оскільки кожне оновлення вимагає повної перебудови та розгортання сайту. Це може виявитися часовитратним і неефективним для великих додатків, які потребують швидких оновлень.

Також, інтерактивність, яка залежить від серверної логіки, вимагатиме додаткових клієнт-серверних запитів, які можуть погіршити користувацький досвід. Для вирішення цих питань може знадобитися гібридний підхід з динамічним рендерингом на стороні клієнта або використанням серверних

функцій, що вносить додаткову складність у процес розробки та підтримки веб-сайту.

1.3 Проблеми ренденгу web-сайтів

Різні типи рендерингу веб-сайтів мають свої специфічні проблеми, що впливають на пошукову оптимізацію. Розглянемо кожний з типів рендерингу.

Client-Side Rendering:

- індексація контенту: так як виконується JavaScript для генерації контенту. Пошукові роботи можуть мати труднощі з індексацією контенту, який генерується динамічно. Сучасні пошукові роботи, здатні виконувати JavaScript, але цей процес може бути неповним або затриманим, що призводить до неповної індексації;
- повільне початкове завантаження: перший запит може бути повільним, оскільки браузеру потрібно завантажити і виконати JavaScript, щоб згенерувати контент;
- проблеми з мета-тегами: перший запит може бути повільним, оскільки браузеру потрібно завантажити і виконати JavaScript, щоб згенерувати контент.

Server-Side Rendering:

- навантаження на сервер: для кожного запиту потрібна генерації HTML на сервері, що може створювати навантаження на сервер, особливо при великій кількості одночасних запитів;
- проблеми з кешуванням: кешування динамічного контенту може бути складним, оскільки кожен запит може генерувати унікальний HTML;
- складність реалізації: є складнішою порівняно з CSR Це вимагає додаткових зусиль для налаштування та підтримки.

Static Site Generation (SSG):

- доступний тільки статичний контент: у випадку коли потрібна будь яка зміна контенту, SSG вимагає регенерацію та повторне розгортання;

– час генерації: генерація великої кількості сторінок може займати значний час, уповільнюючи процес розгортання і оновлення сайту.

1.4 Постановка задачі

Основною цілю є пошук оптимальних шляхів використання технологій веб-рендерингу для підвищення ефективності SEO. Для досягнення цієї цілі потрібно виконати наступні завдання:

– зібрати та систематизувати інформацію про різні підходи до рендерингу веб-сайтів, зокрема статичний, серверний та клієнтський рендеринг;
– аналізувати плюси та мінуси кожного методу рендерингу з точки зору SEO, включаючи їх вплив на швидкість індексації та ранжування;
– розробити методику оцінки ефективності різних методів рендерингу в контексті SEO;
– провести експериментальні дослідження з метою порівняння різних методів рендерингу, що застосовуються у веб-розробці;
– виокремити найкращі практики оптимізації рендерингу для пошукових систем.

2 ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ РІШЕНЬ ДЛЯ РЕНДЕРИНГУ ТА ОПТИМІЗАЦІЇ

2.1 Процес рендерингу веб сторінок

Рендеринг – це процес, який використовується у веб-розробці і перетворює код веб-сайту на інтерактивні сторінки, які користувачі бачать, коли відвідують веб-сайт. Цей термін зазвичай відноситься до використання кодів HTML, CSS та JavaScript. Процес завершується за допомогою двигуна рендерингу, програмного забезпечення, яке веб-браузер використовує для відображення веб-сторінки.

Якщо відображати схематично то кроки які виконує двигун, можна зазначити на рисунку 2.1.

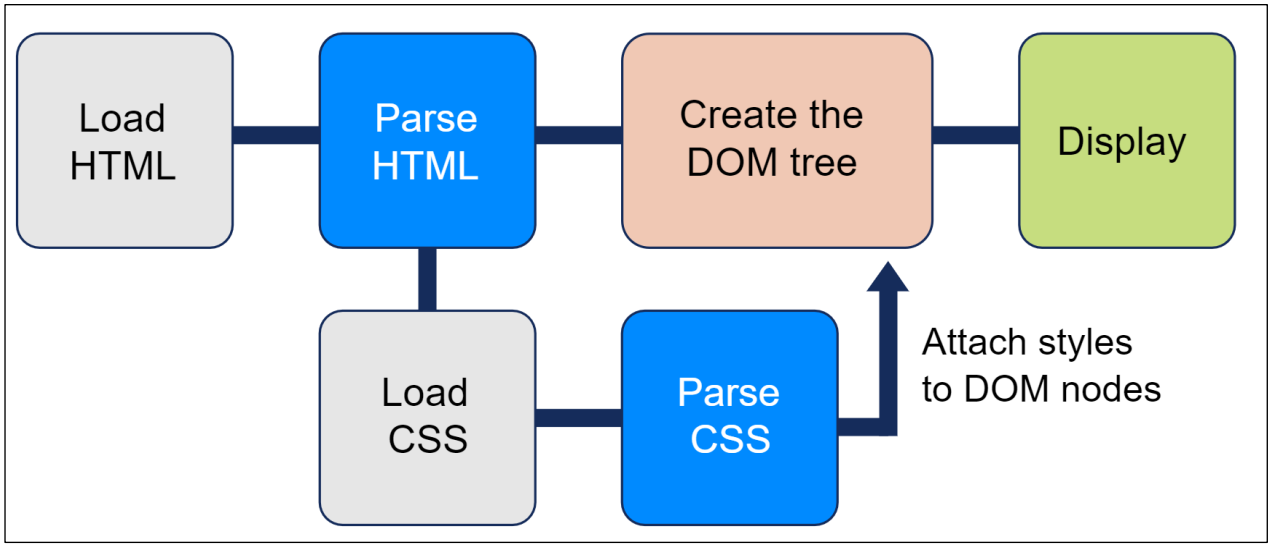

Рисунок 2.1 – Кроки рендерингу веб сторінки в двигуні браузера (за даними [2])

Розглянемо рисунок 2.1 більш детально, та опишемо кроки:

- HTML отримується з сервера та обробляється в DOM;
- стилі завантажуються та аналізуються в CSSOM;
- рендер-дерево створюється з використанням DOM та CSSOM;
- браузер створює макет для кожного елемента рендер-дерева з його індивідуальними координатами, використовуючи метод потоку, який вимагає лише одного проходження для розміщення всіх елементів, у порівнянні з методом таблиць, що вимагає більше одного проходження;

– інформація відображається у вікні браузера у його остаточній формі через останній етап процесу, який також відомий як розфарбовування.

Далі може бути проведено детальне дослідження процесу рендерингу, з акцентом на його оптимізацію для SEO. Це може включати аналіз різних методів рендерингу, таких як Server-Side Rendering (SSR), Static Site Generation (SSG) і Client-Side Rendering. Їх вплив на швидкість завантаження сторінки, індексацію та ранжування пошуковими системами. Результати такого дослідження можуть бути використані для розробки рекомендацій по покращенню SEO через технічні аспекти рендерингу веб-сторінок.

2.2 Аналіз рішень для рендерингу веб сторінок

Враховуючи сучасні тенденції, та популярні інструменти зазначимо що React займає вагове місце у сучасних фреймворках. На базі React розроблено існує кілька популярних та сучасних інструментів, які допомагають в реалізації різних методів рендирингу веб-сторінок.

Razzle, забезпечує серверний рендеринг та інструменти для налаштування розробки без потреби в налаштуванні Webpack або Babel. Завдяки підтримці SSR, Razzle дозволяє генерувати HTML на сервері, що покращує продуктивність та SEO.

Remix підтримує як SSR, так і CSR, що дозволяє обирати найкращий метод рендерингу залежно від потреб проекту. З часом до нього була додана можливість підтримки SSG. Відносно новий фреймворк, який ще перевіряється на практиці.

Next.js – є потужним фреймворком, який підтримує як SSR, так і SSG. Цей фреймворк дозволяє розробникам створювати динамічні веб-додатки з високою продуктивністю та покращеною SEO. Завдяки можливості генерувати HTML на сервері для кожного запиту SSR. Next.js є універсальним рішенням, яке підходить для створення як динамічних додатків, так і статичних сайтів, що робить його одним з найгнучкіших інструментів.

Враховуючи кожен з цих інструментів створений на базі React, вирішено не використовувати жоден з них, враховуючи що результати дослідження можуть бути різними якщо тестувати декілька з них.

Також, можемо зазначити що React версії 18, включає в себе можливість використовувати внутрішній метод «Routes», який допомагає в налагоджені SSR.

2.3 Гібридні методи рендерингу та регідрація

Гібридні методи рендерингу поєднують переваги серверного рендерингу (SSR) та клієнтського рендерингу (CSR), намагаючись забезпечити баланс між швидкістю завантаження, інтерактивністю та оптимізацією для пошукових систем.

Гібридний рендеринг дозволяє використовувати переваги швидкого індексування вмісту пошуковими системами завдяки SSR, в той час як CSR забезпечує багатий користувацький інтерфейс та інтерактивність, характерні для одно сторінкових застосунків (SPA). Це робить гібридні методи особливо ефективними для сучасних веб-додатків, які прагнуть поєднати користувацький досвід традиційних веб-сайтів з інтерактивністю SPA.

Регідрація – це процес, в якому статично відрендерений HTML, створений на сервері, доповнюється динамічною функціональністю на клієнтській стороні за допомогою JavaScript.

Це значить, що елементи інтерфейсу, такі як кнопки або форми, отримують повну функціональність і можуть взаємодіяти з користувачем. Регідрація також дозволяє додати більш складну поведінку, яка залежить від динамічних даних або стану користувача.

SSR with (Re)gidration це поєднання серверного рендеринга та клієнтського рендерингу через регідрацію (rehydration). Спочатку сервер створює повністю сформований HTML веб-сторінки, забезпечуючи швидке відображення вмісту, коли користувач вперше завантажує сторінку. Цей відрендерений контент відправляється до браузера, де він відразу відображається, покращуючи видимість сайту в пошукових системах.

Після того, як сторінка відображена, вступає в дію процес регідрації. Клієнтський JavaScript починає працювати та динамічно змінює статичний HTML, пов'язуючи обробники подій та активуючи іншу інтерактивність. Це означає, що веб-сторінка може реагувати на дії користувача в реальному часі, такі як кліки по посиланнях, заповнення форм і т.д., без необхідності повторного завантаження з сервера.

Такий гібридний підхід дає можливість миттєво показувати вміст, а потім взаємодіяти з багатим інтерактивним інтерфейсом, що згодом завантажується і виконується. Він об'єднує найкращі сторони SSR для пошукової оптимізації та швидкості, а також CSR для динамічності та зручності користувачів. Однак, цей метод також може збільшити складність розробки, оскільки потребує узгодження серверного та клієнтського коду, а також забезпечення ефективної синхронізації даних між ними.

CSR with Prerendering (або Client-Side Rendering із попереднім рендерингом) – це метод розробки веб-додатків, де веб-сторінка спочатку генерується на сервері в статичному вигляді, а потім спрацьовує динамічна функціональність на стороні клієнта.

В цьому методі основний HTML вміст веб-сторінки зазвичай рендериться на сервері, але це не повноцінний SSR, оскільки метою є створення базової версії сторінки, яка може бути індексована пошуковими системами та швидко відображена користувачам. Це забезпечує ініціальне завантаження сторінки без затримок, що можуть виникнути під час завантаження та виконання JavaScript.

Після завантаження статичного вмісту, JavaScript фреймворки на стороні клієнта виконують "регідрацію" сторінки, додаючи інтерактивність та обробку подій. У цей момент веб-сторінка перетворюється на SPA (Single-Page Application), де подальша навігація та взаємодія з користувачем відбувається без перезавантаження сторінок.

Цей підхід дозволяє поєднувати швидкість статичного сайту з інтерактивністю традиційних веб-додатків. Також, він підвищує видимість веб-сторінок в пошукових системах, оскільки сторінка містить необхідний вміст в

момент, коли пошукова система здійснює її сканування. Однак, як і у випадку з іншими гібридними методами, CSR with Prerendering може додати додаткової складності в процес розробки та потребує додаткової оптимізації для забезпечення плавної взаємодії між статичним і динамічним вмістом.

2.4 Методи пошукової оптимізації і їх зв'язок з методами рендерінгу

Методи рендерингу веб-сайтів, мають безпосередній вплив на ключові аспекти пошукової оптимізації. Основні фактори, що пов'язують рендеринг з SEO, включають індексацію контенту та швидкість завантаження сторінок.

Індексація контенту – це процес під час якого пошукові системи, виявляють, зберігають і організовують веб-сторінки для того, щоб їх можна було знайти та відобразити в результатах пошукових систем[11]. Коли пошукова система індексує веб-сторінку, вона додає цю сторінку до своєї бази даних і робить її доступною для користувачів, які шукають релевантні ключові слова або фрази.

Різні методи рендерингу веб-сторінок по-різному впливають на процес індексації контенту пошуковими системами.

У CSR початковий HTML, отриманий пошуковими роботами, містить лише базову структуру сторінки, а весь контент з'являється тільки після виконання JavaScript. Пошукові роботи можуть мати труднощі з виконанням JavaScript або можуть взагалі його не виконувати. Це негативно впливає на видимість сторінки у пошукових результатах.

У SSR індексація проходить краще, тому що пошукові роботи отримують повний HTML-код, що значно полегшує індексацію контенту. Пошукові роботи можуть швидко і ефективно проаналізувати і зберегти контент у своїх базах даних. Це забезпечує краще ранжування і видимість у пошукових системах, оскільки контент доступний для індексації відразу після завантаження сторінки.

Але найкращі результати для індексації надає SSG так як, пошукові роботи отримують готові статичні сторінки з повним HTML-кодом, що забезпечує легку і швидку індексацію.

Наступним фактором є швидкість завантаження сторінок – це час, який потрібен для повного завантаження веб-сторінки в браузері користувача. Це критично важливий аспект для пошукової оптимізації SEO, тому що пошукові системи, використовують швидкість завантаження сторінок як один з факторів ранжування. Швидкі сторінки зазвичай отримують кращі позиції у результатах пошуку.

Основними показниками швидкості завантаження сторінок є [12]:

- Time to First Byte (TTFB) – час, який проходить з моменту відправки запиту клієнтом до отримання першого байта відповіді від сервера;
- First Contentful Paint (FCP) – час, який потрібен для відображення першого елемента контенту на сторінці;
- Largest Contentful Paint (LCP) – час, потрібний для завантаження найбільшого елемента контенту на сторінці;
- First Input Delay (FID) – час, який проходить з моменту першої взаємодії користувача зі сторінкою;
- Cumulative Layout Shift (CLS) – ступінь несподіваних змін у макеті сторінки під час її завантаження.

Підсумовуючи, CSR має повільне початкове завантаження JavaScript файлу тому показники LCP та FID можуть мати найнижчі показники в порівняні з іншими.

У SSR, HTML який згенерований на сервері, може відображатись швидше, що зменшує час до першого відображення контенту. Але серверна обробка впливає на TTFB та FID.

Для SSG, головною проблемою може стати LCP у випадку великий обсяг статичних ресурсів може затримувати відображення найбільшого елементу контенту.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Засоби проведення дослідження

Спираючись на сучасні тренди у веб розробці, а саме на самий популярний фреймворк за статистикою на StackOverflow (Рис. 1.1), можемо зробити висновок що для загального тестування методу рендерингу на клієнтська стороні, ми можемо використовувати як просто JavaScript так і React фреймворк, щоб проаналізувати різницю у нагрузці. Тож буде можливо проаналізувати вплив фреймворків на рендеринг.

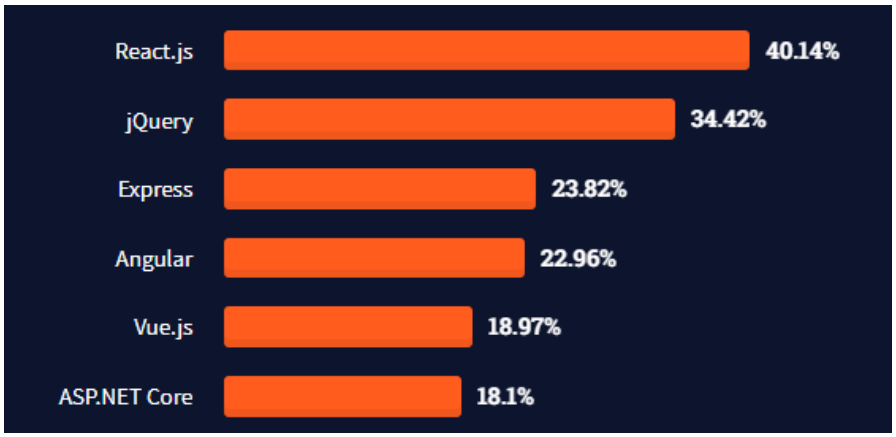

Рисунок 3.1 – Популярні фреймворки (за даними [1])

React – це декларативний, ефективний та гнучкий JavaScript фреймворк для створення користувацьких інтерфейсів, який розроблений і підтримується Facebook. Він дозволяє розробникам створювати великі веб-додатки, що можуть змінювати дані, не перезавантажуючи сторінку. Головною особливістю React є компонентний підхід до розробки, де інтерфейс поділяється на незалежні частини, які можна перевикористовувати і управляти кожну з них окремо.

Враховуючи що React дуже потужний інструмент для створення односторінкових веб-сайтів, з динамічним рендерингом, ми використаємо його для генерації контенту сторінки на стороні клієнта.

Для аналізу роботи на Серверній стороні був обраний NodeJS.

Node.js – це потужна крос-платформна середовища виконання для JavaScript, яка дозволяє розробникам використовувати JavaScript для написання

серверного коду. Вона була створена і заснована на двигуні V8 від Google, який виконує JavaScript дуже швидко, компілюючи його в машинний код.

Для хостингу серверної частини на Node.js була обрана платформа – Heroku. Ця платформа дозволяє розгортати, управляти та масштабувати додатки.

Webpack – це потужний інструмент для сучасної веб-розробки, який використовується як модульний збірник (module bundler) для JavaScript додатків. Буде використовуватись для створення білдів на різних методах. Також буде проаналізований його вплив на методи рендерингу.

Для хостингу клієнтської сторони обрано Netlify. Netlify є потужною платформою для розробки, розгортання та управління веб-сайтами та веб-додатками, що базуються на JAMstack архітектурі. Вона надає широкий спектр інструментів для автоматизації процесів CI/CD які будуть корисні під час дослідження.

Тестування різних методів рендерингу буде проводитись за допомогою зручних інструментів, таких як:

GTMetrix – це популярний онлайн-інструмент, який використовується для аналізу продуктивності та швидкості вебсайтів. Він надає детальні звіти про різні аспекти продуктивності вебсайту.

Google PageSpeed Insights – інструмент, який аналізує швидкість завантаження вашого сайту і рекомендує як її покращити.

3.2 Веб-сайт сторінка для тестування

Створення веб-сторінки для тестування є базовим етапом у процесі дослідження методів рендерингу веб-сайтів для пошукової оптимізації (SEO), які будуть проводитись а рамках цієї роботи.

Така сторінка повинна бути спеціально розроблена для викликання різних сценаріїв рендерингу та оцінки їх впливу на індексацію пошуковими системами. Вона повинна містити різноманітні типи контенту, такі як текст, зображення, відео та інші медіа-елементи, а також різні структурні і функціональні компоненти, включаючи заголовки, списки, форми та кнопки.

Створюючи таку сторінку ми повинні розглядати такі атрибути:

- структуру контенту – ієрархію заголовків (h1-h6), абзаци, списки і таблиці для перевірки структурної семантики та її впливу на SEO;
- використання ключових слів – вбудування відповідних ключових слів у вміст для аналізу їх розпізнавання та ранжування пошуковими системами;
- додавання метаданих – налаштування мета-тегів, таких як title, description, та атрибути alt для зображень, щоб оцінити, як вони впливають на SEO;
- зовнішні та внутрішні посилання: посилання на інші сторінки сайту та зовнішні ресурси для виміру їх впливу на SEO;
- JavaScript-інтерактивність: Інтегруємо динамічні JavaScript-елементи для тестування клієнтського рендерингу (це важливо для тестування CSR).

Спираючись на всі попередні зазначення, можемо зробити висновок, що веб-сторінка, яка схожа на магазин продажу товарів, з певним асортиментом товарів на одній сторінці, підходить для тестування усіх методів рендерингу, та всі зазначені атрибути. Кожен товар на сторінці буде компонентом для Client-Side Rendering або один компонент який динамічно генерує всі товари на сторінці. Тим самим ми покриємо перевірку динамічного рендерингу.

Також не менш важливим є кількість HTML-елементів або в цілому різних компонентів на сторінці. Можливо тільки здогадуватись як різні методи рендерингу будуть працювати на схожих сторінках але з різною кількістю компонентів. Тому дослідження схожих сторінок, але з різною кількістю контента, буде доцільним, щоб проаналізувати, в яких випадках, той чи інший метод будуть більш ефективними.

Тому ми будемо перевіряти одну сторінку, з такою кількістю однакових компонентів: 100, 250, 500, 1000. Таким чином ми можемо побачити не тільки ефективність, переваги та недоліки кожного з методів рендерингу, але й оцінити, при яких значеннях, є сенс змінювати метод або намагатись оптимізувати механізми для рендерингу сторінки.

3.3 Критерії оцінки процесу рендерингу

Оцінка методів рендерингу для SEO включає кілька ключових метрик, таких як: Time to Interactive (TTI), First Contentful Paint (FCP), Time to First Byte (TTFB) та Largest Contentful Paint (LCP).

Time to Interactive (TTI) – це метрика, яка вимірює час від моменту запиту сторінки користувачем до моменту, коли сторінка стає повністю інтерактивною та здатною відповідати на взаємодії користувача, такі як кліки або введення даних. Це не просто час завантаження візуального вмісту, але й готовність сторінки до повноцінної взаємодії. Високий TTI може вказувати на проблеми з JavaScript, які блокують основний потік, або на велику кількість невикористаного коду.

First Contentful Paint (FCP) вимірює час від початку завантаження сторінки до моменту, коли браузер відмальовує перший шматок вмісту, який може бути текстом, зображенням або SVG. Це відображає швидкість, з якою вміст починає з'являтися на сторінці, даючи користувачу перші візуальні відомості про завантаження вмісту. Чим швидше відбувається FCP, тим краще користувацький досвід.

Time to First Byte (TTFB) – це час, який проходить від моменту зробленого користувачем запиту до отримання першого байту відповіді сервером. Ця метрика важлива, оскільки вона відображає швидкість відгуку сервера, що включає час мережевої затримки та швидкість обробки запиту сервером. Оптимальний TTFB вказує на те, що сервер швидко обробляє та відповідає на запити, що є важливим для загальної продуктивності веб-сайту.

First Input Delay (FID) це польова метрика, яку не можливо визначити під час дослідження оскільки для цього потрібні реальні користувачі. Тому інструменти Google Page Speed та GTmetrix не можуть її проаналізувати. Замість цієї метрики ми звернемо увагу на LCP.

Largest Contentful Paint (LCP) - це метрика, яка вимірює час, необхідний для відображення найбільшого видимого елемента в області перегляду після початку

завантаження сторінки. Вона є важливою для оцінки швидкості завантаження сторінки.

Ці метрики можна вимірюватись за допомогою GTmetrix. Також, в рамках дослідження буде використаний Google PageSpeed Insights, який забезпечує аналіз швидкості завантаження сторінок і дає рекомендації щодо їх оптимізації.

3.4 Реалізація Веб-застосунків для дослідження

Згідно з поставленим завданням, розроблена структуру веб-додатку, яка забезпечує можливість дослідити сучасні методи рендирингу на практиці, отримати показники які важливі для SEO.

Структура компонентів, повинна включати в себе всі можливі DOM-елементи які будуть мати вплив на пошук у SEO, як зазначено у пункті 3.2. Тому був створений один компонент, який включає в себе необхідні DOM-елементи. Та буде використаний для певної кількості таких компонентів. Прототип сторінки можна побачити на рисунку 3.2

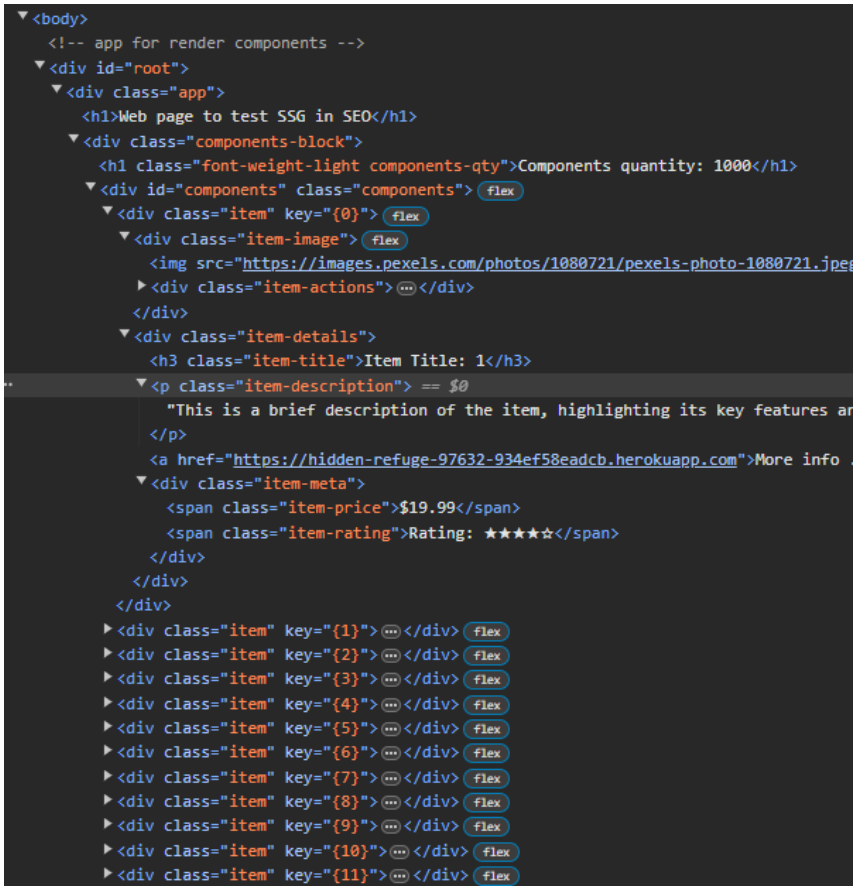

Рисунок 3.2 – Прототип сторінки для дослідження (рисунок створено самостійно)

Враховуючи що під час ініціалізації веб-сторінки у браузері, та обробки клієнтської частини, а також те що деякі частини клієнтської сторони будуть загружатись для кожного з видів рендерингу були створені окремі веб-додатки. Кожен з них має можливість перевірки різної кількості компонентів. Цей підхід одразу буде більш детальним та чистим для дослідження.

Також, в кожен тип рендерингу були додані мета-теги, які допомагають SEO ініціалізувати веб сторінку (див рис. 3.3)

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="CSR SEO test">
    <meta name="keywords" content="HTML, CSS, JavaScript, SEO">
    <meta name="author" content="Serhii Brukhtii">
    <meta name="google-site-verification" content="qyvTUdO3TFe4Lf0TeasCRvG-
    <title>CSR SEO Test</title>
</head>
```

Рисунок 3.3 – Прототип сторінки мета-теги (рисунок створено самостійно)

Це є важливою складовою для внутрішньої оптимізації, тому що пошукові роботи, пов’язують веб сторінку в базі даних з цими мета тегами.

4 ОПИС ЕКСПЕРЕМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Реалізація Веб-застосунків для дослідження

Для реалізації SSR та SSG був розроблений Node.js сервіс, який працює з POST запитами, в якому можна запросити потрібну нам кількість компонентів.

Спочатку був створений звичайний JavaScript модуль, який використовується для формування HTML структури для наших запитів. Функція «getComponent» повертає компонент в якому включені всі можливі та необхідні теги для SEO (див. рис. 4.1). За допомогою «getComponentBlock» ми створюємо блок компонентів який включає в себе ту кількість компонентів яку ми запросили (див. рис. 4.2).

```
const getComponent = (index) =>
    `<div class="item" key={${index}}>
      <div class="item-image">
        <img src="${imagesLinks[index % 10]}" alt="Item Name"/>
        <div class="item-actions">
          <button type="button" class="add-to-cart-btn">Add to Car
        </div>
      </div>
      <div class="item-details">
        <h3 class="item-title">Item Title: ${index + 1}</h3>
        <p class="item-description">This is a brief description of
        <a href="https://hidden-refuge-97632-934ef58eadcb.herokuap
        <div class="item-meta">
          <span class="item-price">$19.99</span>
          <span class="item-rating">Rating: ★★★★☆</span>
        </div>
      </div>
  </div>`;
```

Рисунок 4.1 – Генерація одного компоненту (рисунок створено самостійно)

```
export const ComponentsBlock = ({ componentsQty }) => {

    const getComponents = () => {
      const arrayWithObjects = Array.from({ length: componentsQty }, (_, index) => (getCompo
      return arrayWithObjects;
    }

  return (
      <div className="components-block">
      <h1 className="font-weight-light components-qty">Components quantity: {componentsQty}<
        <div id="components" className="components">
          {getComponents()}
        </div>
      </div>
  );
};
```

Рисунок 4.2 – Генерація блоку компонентів (рисунок створено самостійно)

Для реалізації запиту в SSR, створений POST запит «getSSR», який отримує необхідну кількість компонентів, генерує їх за допомогою «getComponentBlock», та повертає готовий HTML (див. рис. 4.3).

```
.post('/getSSR', (req, res) => {
  const componentsQty = req.body.componentsQty;

  const componentBlock = getComponentsBlock({ componentsQty });
  res.send(componentBlock);
})
```

Рисунок 4.3 – POST запит для SSR (рисунок створено самостійно)

В реалізації запиту в SSG була додана можливість повернення вже готового HTML файлу. Для цього був написаний ще один скріпт, який генерує статичні HTML сторінки. В результаті ми маємо необхідну нам згенеровану кількість компонентів у статичному вигляді на стороні сервера(див. рис. 4.4).

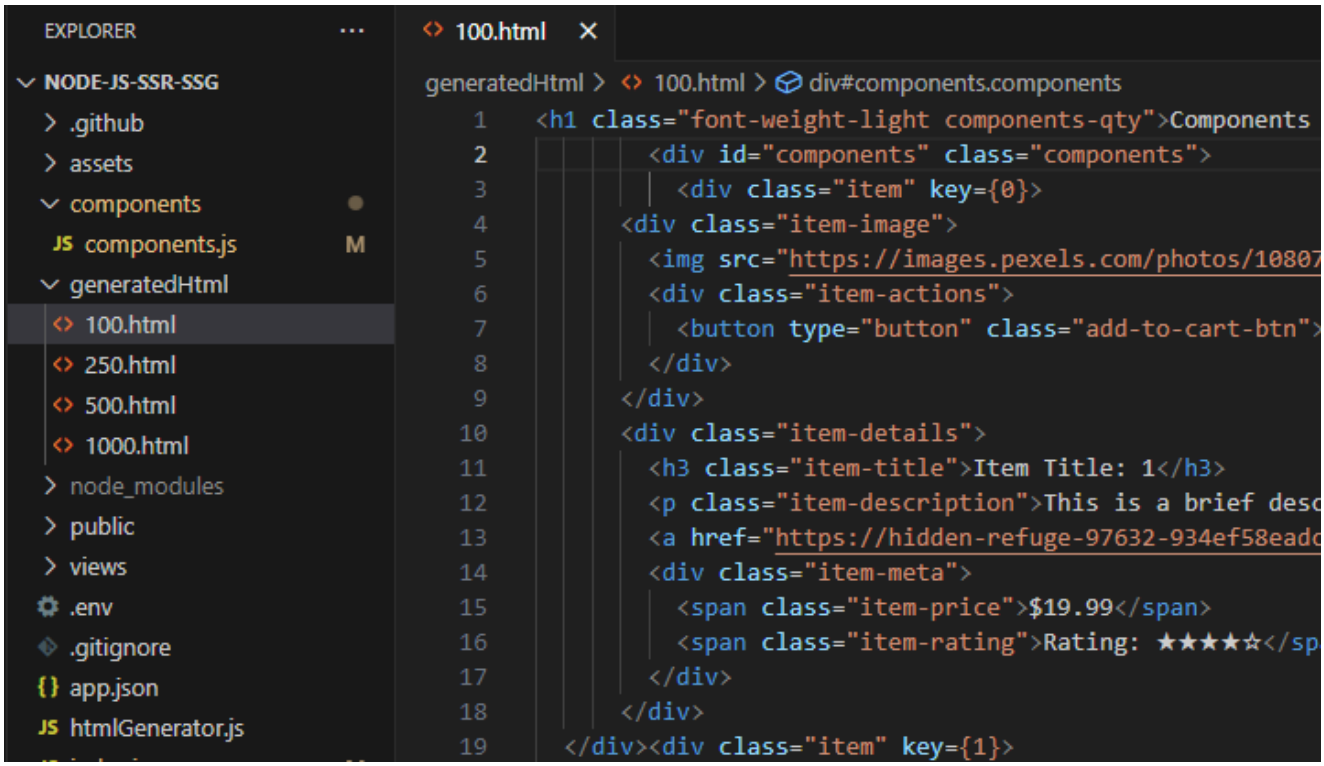

Рисунок 4.4 – Сгенеровані HTML файли для SSG (рисунок створено самостійно)

Щоб кліент міг запросити HTML файли, реалізовано ще один POST запит «getSSG», який очікує тільки 100, 250, 500, 1000 елементів (див. рис. 4.5).

```
.post('/getSSG', (req, res) => {
  const componentsQty = req.body.componentsQty;

  const file = `generatedHtml/${componentsQty}.html`;
  res.sendFile(file, {root: __dirname })
})
```

Рисунок 4.5 – POST запит для SSG (рисунок створено самостійно)

Після реалізації необхідних нам запитів в Node.js, ця логіка була розгорнута на Heroku платформі. Для перевірки праце здібності цього рішення, використаємо Postman для створення POST запитів (див. рис. 4.6).

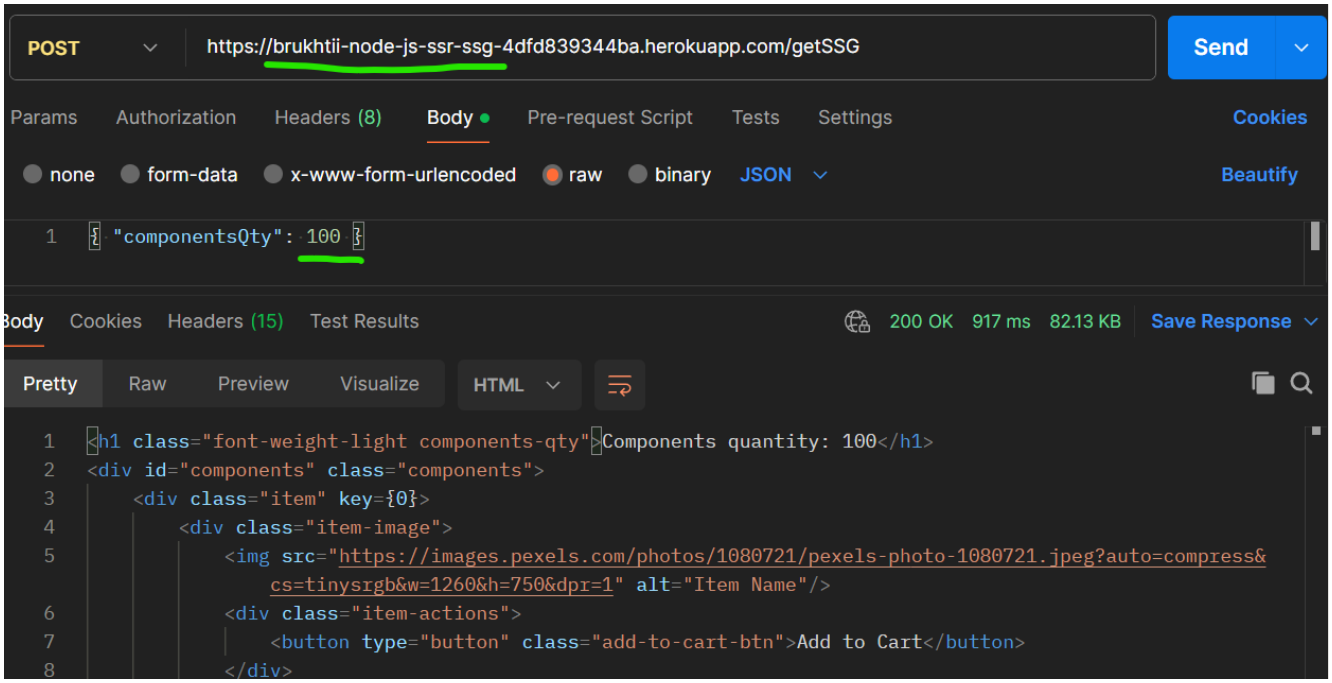

Рисунок 4.6 – тестування POST запиту для SSG (рисунок створено самостійно)

В результаті отримано Server Side сервіс який ми можемо використати вже на стороні клієта для дослідження SSR та SSG.

Для реалізації клієнта який рендерить сторінку за допомогою SSR був розроблений простий клієнт. Використовуючи можливості React та Webpack, який після ініціалізації на стороні браузера, робить запит щоб отримати певну кількість компонентів з реалізованого сервісу у пункті 3.4.1.

«App.js» - головний файл який запускає React застосунок. Включає в себе загальні елементи та «ComponensBlock» компонент, який виконує ключову роль для дослідження. Саме «ComponensBlock» відповідає за рендеринг кількості компонентів які будуть досліджуватись (див. рис. 4.7). Він включає запит для отримання згенерованих компонентів в Node.js та їх рендерінг після отримання.

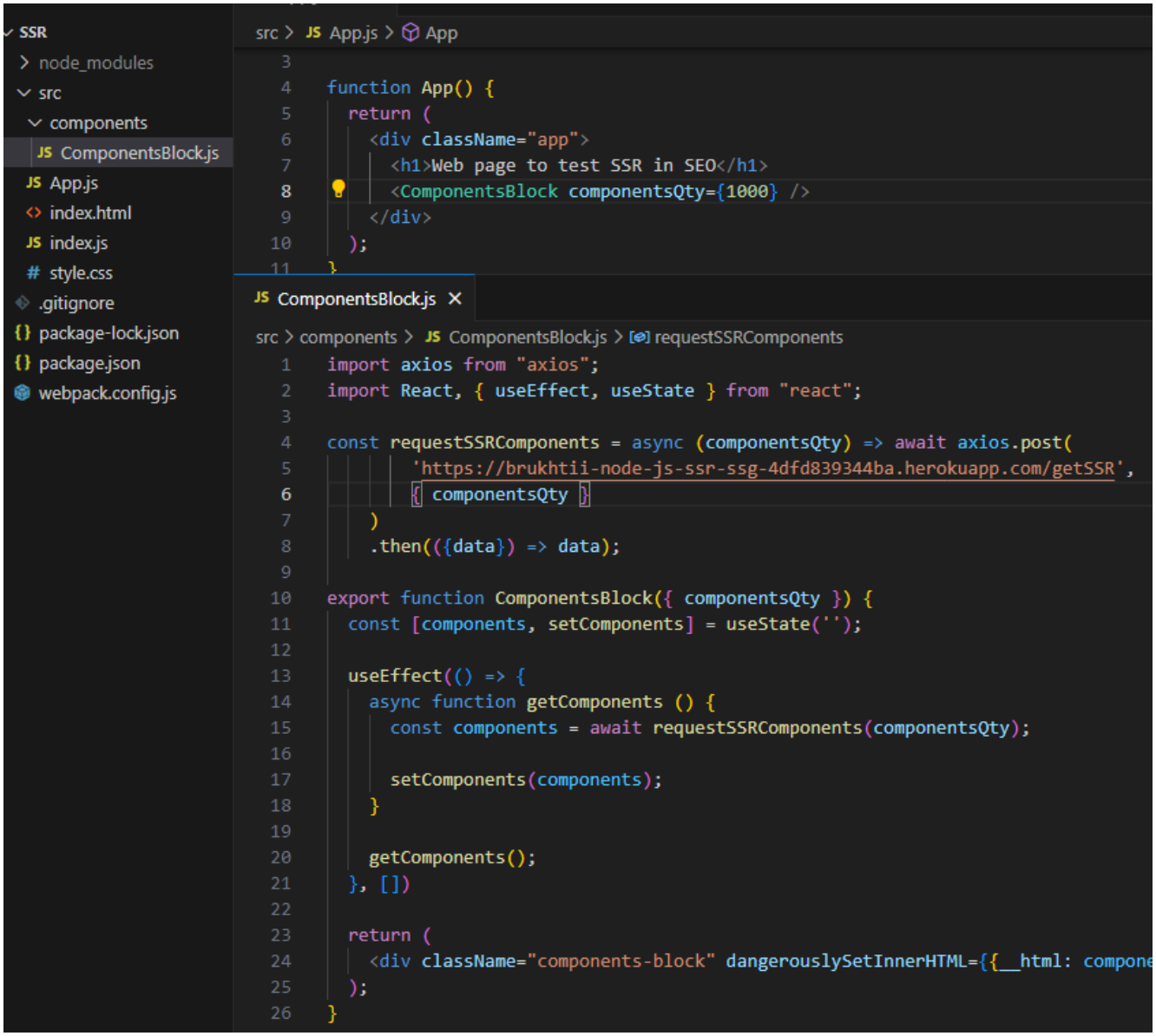

Рисунок 4.7 – Реалізація SSR клієнта (рисунок створено самостійно)

Для реалізації клієнта який рендерить сторінку за допомогою SSG, був пере використаний код з SSR. Змінивши у «ComponensBlock» запит на POST «getSSG» який повертає вже згенерований HTML (див. рис. 4.8).

```
const requestSSRComponents = async (componentsQty) => await axios.post(
        'https://brukhtii-node-js-ssr-ssg-4dfd839344ba.herokuapp.com/getSSG',
        { componentsQty }
    )
    .then(({data}) => data);

export function ComponentsBlock({ componentsQty }) {
```

Рисунок 4.8 – Реалізація SSG клієнта (рисунок створено самостійно)

При розгляді реалізації клієнта для CSR була виконана генерація компонентів на стороні клієнта інструментом React. На рисунку 4.9 видно, що компонент «ComponentsBlock», буде згенерований в двигуні браузера.

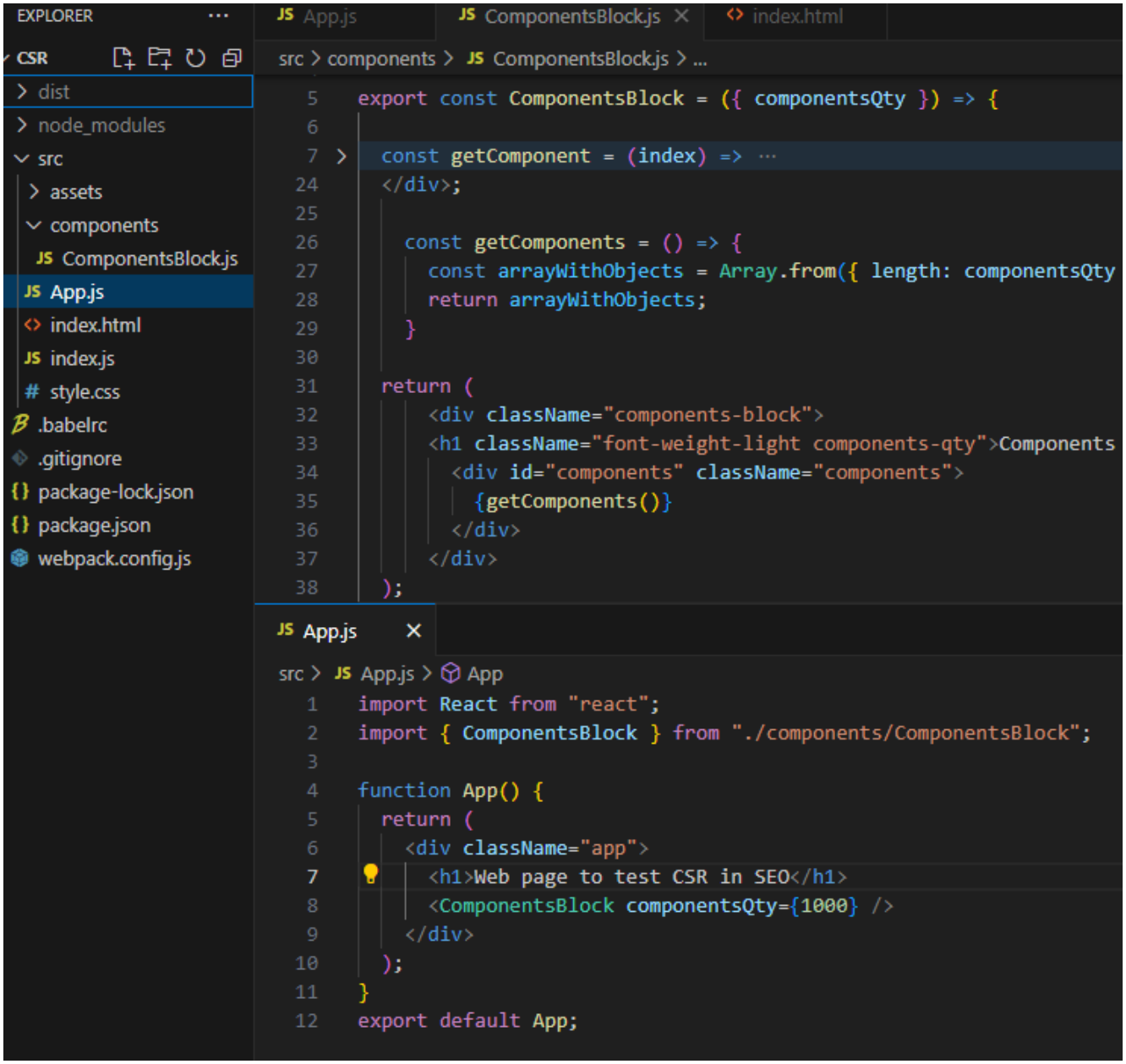

Рисунок 4.9 – Реалізація CSR клієнта (рисунок створено самостійно)

Використовуючи платформу Netlify, були розгорнуті всі необідні веб сторінки для кожного типу рендерингу та певної кількості компонентів. А саме 100, 250, 500 та 1000 (див. рис. 4.10).

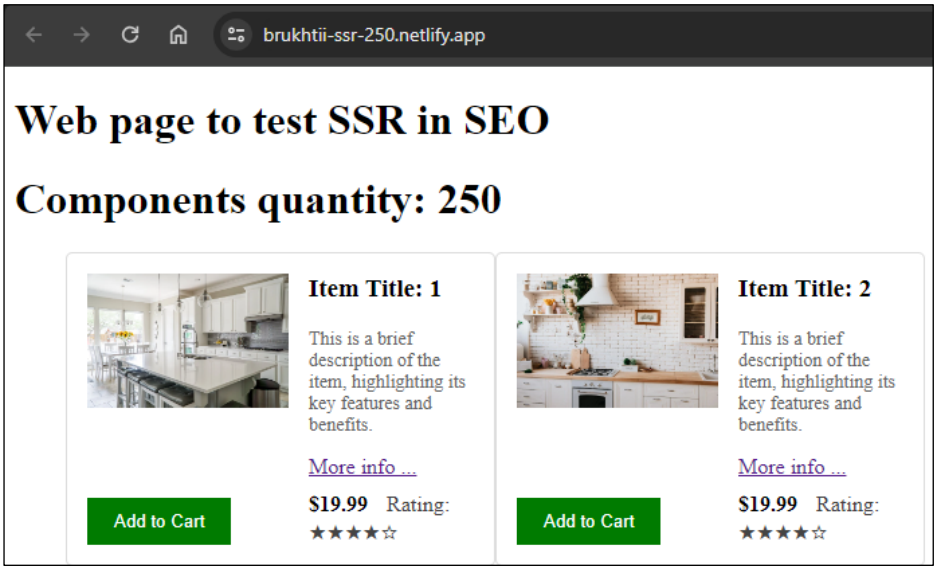

Рисунок 4.10 – Приклад хосту для SSR з 250 компонентів (рисунок створено самостійно)

4.2 Збір даних з розгорнутих веб-сторінок

Для збору необхідних критеріїв оцінки кожна веб сторінка була проаналізована в GTmetrix. Один з результатів тесту, а саме CSR з кількістю компонентів 100 зображено на рисунку 4.11.

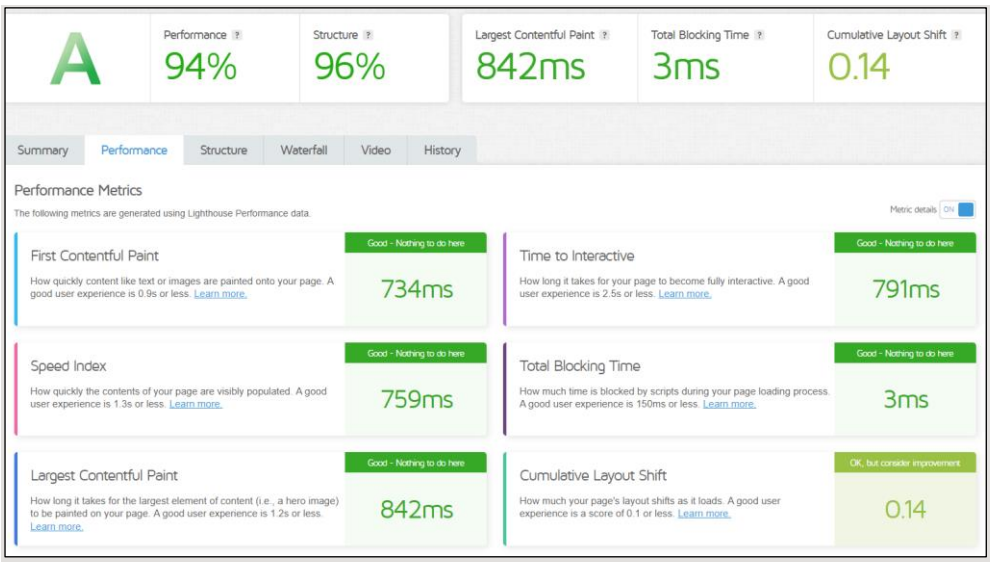

Рисунок 4.11 – Приклад результатів тесту у GTmetrix для CSR з 100 компонентами (рисунок створено самостійно)

За допомогою GTmetrix було проаналізована кожна з сторінок та результати необхідних метрик записані в таблицю (див. табл. 4.1). Всі тести збережені та доступні у Додатку Е.

Таблиця 4.1 – Узагальнююча таблиця результатів метрик

Metric/components	100	250	500	1000	positive
CSR					
Time to First Byte (TTFB)	0,186	0,186	0,191	0,198	0,20
First Contentful Paint (FCP)	0,734	0,797	0,943	1,100	0,90
Largest Contentful Paint (LCP)	0,842	0,938	0,981	1,500	1,20
Time to Interactive (TTI)	0,791	0,797	1,200	1,400	2,50
SSR					
Time to First Byte (TTFB)	0,195	0,198	0,191	0,191	0,20
First Contentful Paint (FCP)	0,568	0,690	0,593	0,597	0,90
Largest Contentful Paint (LCP)	1,400	1,700	1,800	2,200	1,20
Time to Interactive (TTI)	0,568	1,700	1,700	2,100	2,50
SSG					
Time to First Byte (TTFB)	0,195	0,202	0,191	0,191	0,20
First Contentful Paint (FCP)	0,705	0,681	0,562	0,584	0,90
Largest Contentful Paint (LCP)	1,500	1,700	1,800	2,100	1,20
Time to Interactive (TTI)	1,400	1,700	1,700	2,000	2,50

Результати в Google PageSpeed Insights, а саме показник оптимізації SEO були перевірені на показник SEO (див. рис. 4.12)

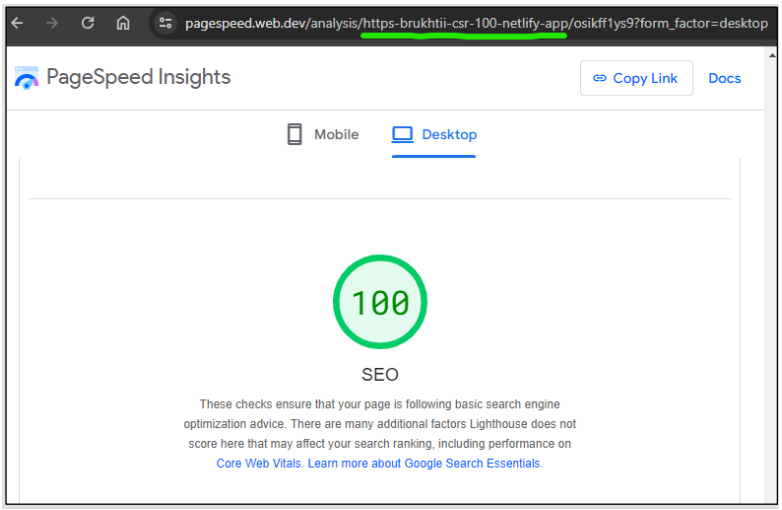

Рисунок 4.12 – Результат перевірки параметру SEO в Google PageSpeed Insights (рисунок створено самостійно)

Усі веб сторінки, в Google PageSpeed Insights показали SEO в 100%. При перевірці як цей показник рахується, було виявлено що тільки по мета-тегам. Наші тестові сторінки вже включають в себе всі необхідні. Тому спостерігати далі за цим параметром не має сенсу (див. рис. 4.12).

4.3 Аналіз отриманих результатів

Після отримання таблиці з метріками для кожного типу даних та всіх установлених компонентів було створено діаграми та проаналізуємо кожен окремий параметр для всіх типів рендирингу та кількості компонентів.

Результати TTFB метрики (див. рис 4.13)

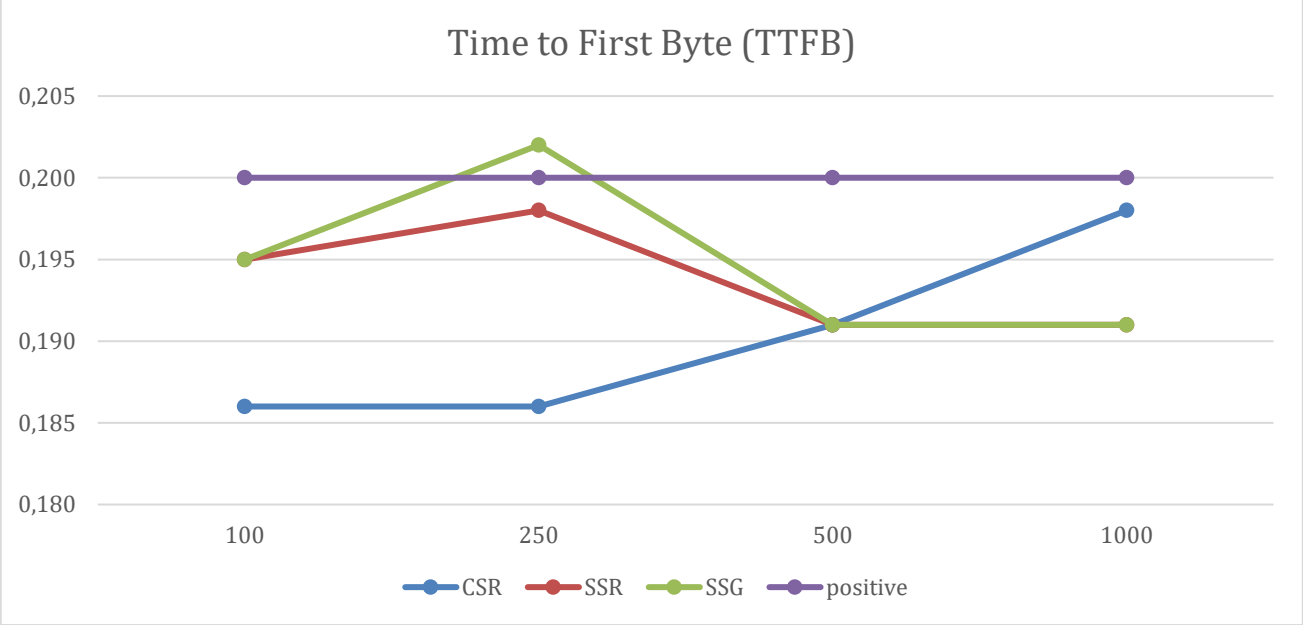

Рисунок 4.13 – Результат тесту TTFB (рисунок створено самостійно)

Для CSR час до першого байта починається з 0.186 секунди при 100 компонентах і залишається стабільним до 0.186 секунди при 250 компонентах. Потім він дещо збільшується до приблизно 0.191 секунди при 500 компонентах і зростає до 0.198 секунди при 1000 компонентах.

Для SSR початковий час до першого байта становить 0.195 секунди при 100 компонентах і дещо збільшується до 0.198 секунди при 250 компонентах. Потім час зменшується до 0.191 секунди при 500 компонентах і залишається стабільним до 0.191 секунди при 1000 компонентах.

Для SSG час до першого байта починається з 0.195 секунди при 100 компонентах, досягає піку в 0.200 секунди при 250 компонентах, а потім зменшується до 0.191 секунди при 500 компонентах і залишається стабільним до 0.191 секунди при 1000 компонентах.

Зауважемо що CSR показує найстабільніший час при малих обсягах даних, але збільшується зі збільшенням кількості компонентів. SSR і SSG типи рендерингу показують тенденцію до зниження часу до першого байта зі збільшенням кількості компонентів, що може залежити від ефективності серверного оброблення та кешування. Але це важко пояснити враховуючи що реалізація SSR та SSG для різних компонентів абсолютно однакова.

Результати FCP метрики (див. рис 4.14)

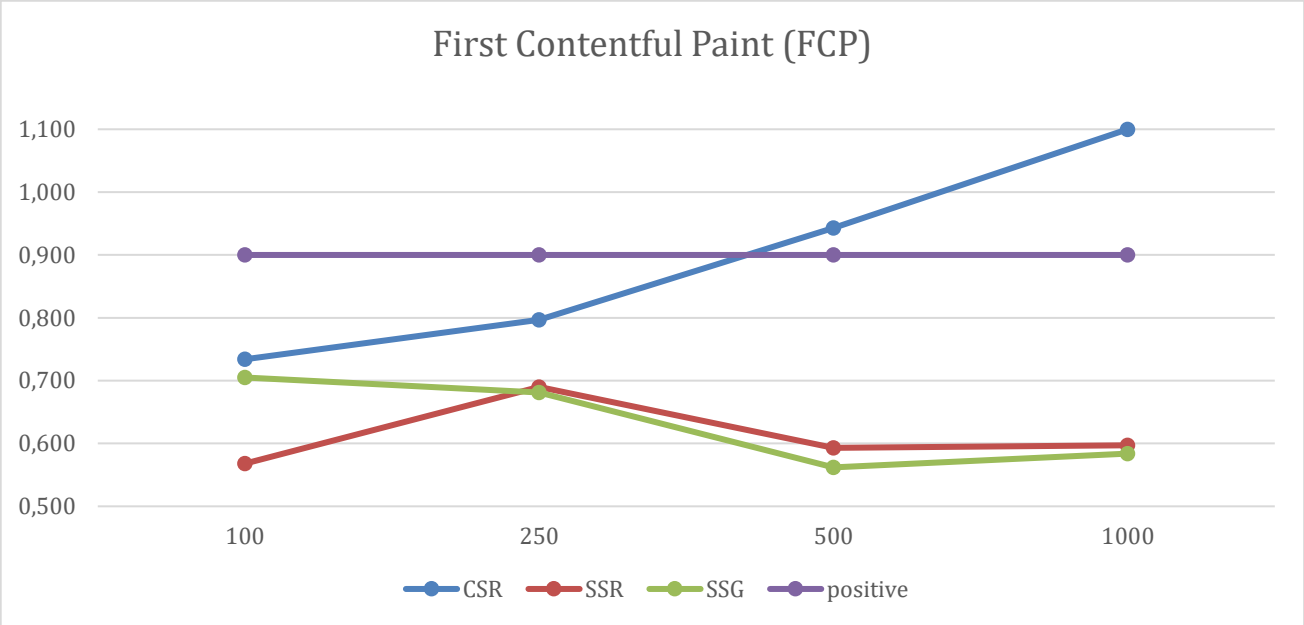

Рисунок 4.14 – Результат FCP тесту (рисунок створено самостійно)

Для CSR час до першого відображення контенту починається з 0.734 секунди при 100 компонентах і поступово збільшується до 1.1 секунди при 1000 компонентах.

Для SSR він становить 0.568 секунди при 100 компонентах, збільшується до 0.690 секунди при 250 компонентах, а потім зменшується до приблизно 0.593 секунди при 500 компонентах і залишається майже незмінним до 1000 компонентів.

Для SSG час до першого відображення контенту починається з 0.705 секунди при 100 компонентах, збільшується до 0.681 секунди при 250 компонентах, потім зменшується до 0.562 секунди при 500 компонентах і знову збільшується до 0.597 секунди при 1000 компонентах.

Узагальнюючи, можемо сказати що CSR має найгірший час до першого відображення контенту зі збільшенням кількості компонентів, тоді як SSR і SSG показують кращу продуктивність, особливо при великій кількості компонентів. SSR має дещо кращі результати для FCP при малих обсягах даних, тоді як SSG показує стабільніші результати при збільшенні кількості компонентів.

Результати LCP метрики (див. рис 4.15)

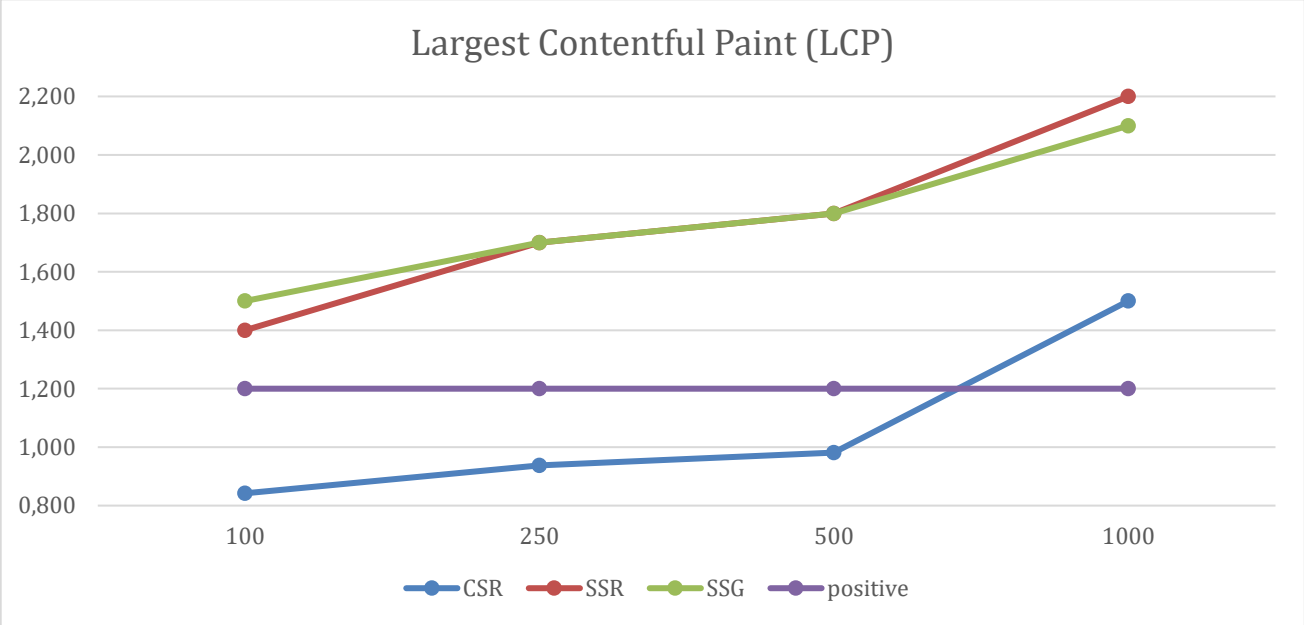

Рисунок 4.15 – Результат тесту LCP (рисунок створено самостійно)

Для CSR час до найбільшого відображення контенту починається з приблизно 0.842 секунди при 100 компонентах і залишається майже незмінним до 250 компонентів. Потім час різко збільшується до приблизно 1.5 секунд при 500 компонентах і досягає 2.1 секунд при 1000 компонентах.

Для SSR час до найбільшого відображення контенту починається з приблизно 1.4 секунд при 100 компонентах і поступово збільшується до 1.8 секунд при 250 компонентах. Потім час зростає повільніше до приблизно 1.9 секунд при 500 компонентах і досягає піку в 2.1 секунди при 1000 компонентах.

Для SSG час до найбільшого відображення контенту починається з 1.5 секунд при 100 компонентах і також поступово збільшується до 1.7 секунд при 250 компонентах. Потім час продовжує зростати до приблизно 1.8 секунд при 500 компонентах і досягає 2.0 секунд при 1000 компонентах.

Можемо зробити висновок, що CSR має найкращі результати для часу до найбільшого відображення контенту в рамках обраної кількості компонентів, тоді як SSR і SSG показують більш стабільну продуктивність зі збільшенням компонентів. Можна вважати що при ще більшій кількості компонентів, CSR буде ще менш ефективний

Результати TTI метрики (див. рис 4.16)

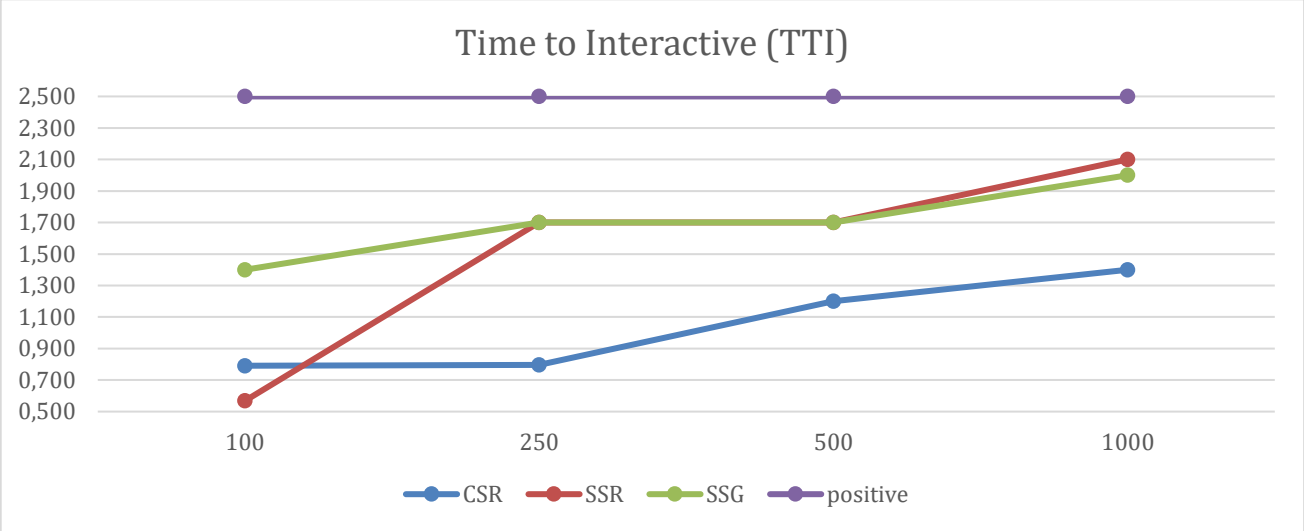

Рисунок 4.16 – Результат тесту TTI (рисунок створено самостійно)

Для CSR час до інтерактивності починається з 0.791 секунди при 100 компонентах і поступово збільшується до приблизно 2.1 секунди при 1000 компонентах.

Для SSR він становить 0.568 секунди при 100 компонентах, але різко зростає до 1.7 секунди при 250 компонентах і потім до 2.1 секунди при 1000 компонентах..

Для SSG час до інтерактивності починається з 1.4 секунди при 100 компонентах і залишається майже стабільним до 1.7 секунди при 250 і 500 компонентах, а потім досягає 2.1 секунди при 1000 компонентах.

Можемо зробити висновок що CSR найкраще працює при невеликій кількості компонентів, тоді як SSR показує значне зростання часу до інтерактивності з ростом компонентів. SSG має стабільніший час до інтерактивності при збільшенні обсягу даних, але все ж таки показує збільшення часу при великій кількості компонентів.

4.3 Оптимізація веб сторінок за допомогою комбінованих типів рендирингу методів

Для оптимізації були обрані протестовані веб сторінки з 1000 компонентів, тому що саме з цими показниками простіше оцінити вплив типу рендерингу та його комбінацій.

Найбільшою проблемою в CSR є індексація під час пошуку, що виділяє її від SSR та SSG. Тому було реалізовані комбінації CSR + SSR та CSR + SSG для більш детального аналізу впливу на внутрішню оптимізацію.

Для реалізації SSR + CSR, використовувались вже розроблені рішення у пункті 4.1, достатньо об’єднати ці методи за допомогою React компонентів. Таким чином «CSRComponentBlock» виконує рендеринг на стороні клієнту для 500 компонентів. А «SSRComponentsBlock» виконує запит для отримання згенерованого HTML та його рендеринг.

```
function App() {
  return (
    <div className="app">
      <h1>Web page to test SSR + CSR in SEO</h1>
      <div className="modified-components">
        <CSRComponentsBlock componentsQty={500} />
        <SSRComponentsBlock componentsQty={500} />
      </div>
    </div>
  );
}
```

Рисунок 4.17 – Об’єднання CSR з SSR

Враховуючи схожість архітектури веб сторінок між SSR та CSR, був пере використаний код з рисунку 4.17, але замість «SSRComponentsBlock», був доданий «SSGComponentsBlock», який створює запит для отримання статичного HTML файлу та його рендерингу.

4.4 Аналіз результатів оптимізації в порівняні з основними методами рендирину

Використовуючи GTmetrix були протестовані веб-сторінки з гібридним методом рендирингу CSR + SSR та CSR + SSG. Дані отримані за допомогою GTmetrix та записані в таблицю 4.2 (переглянути результати можна за посиланням в додатку Е)

Таблиця 4.2 – Результати метрик гібридних підходів.

Metric/components	Value
CSR + SSR	
Time to First Byte (TTFB)	0,195
First Contentful Paint (FCP)	1,000
Largest Contentful Paint (LCP)	1,300
Time to Interactive (TTI)	2,000
CSR + SSG	
Time to First Byte (TTFB)	0,191
First Contentful Paint (FCP)	1,000
Largest Contentful Paint (LCP)	1,300
Time to Interactive (TTI)	2,000

Таблиця 4.3 – Результати метрик основних та гібридних підходів для 1000 компонентів

Metrix/rendering type	CSR	SSR	SSG	CSR + SSR	CSR + SSG	positive
Time to First Byte (TTFB)	0,198	0,191	0,191	0,195	0,191	0,20
First Contentful Paint (FCP)	1,100	0,597	0,584	1,000	1,000	0,90
Largest Contentful Paint (LCP)	1,500	2,200	2,100	1,300	1,300	1,20
Time to Interactive (TTI)	1,400	2,100	2,000	2,000	2,000	2,50

Була створена порівняльна діаграма (див. додаток Ж).

Таблиця 4.3 показує, що Time to First Byte (TTFB) має схожі значення для всіх методів рендерингу, з незначними відмінностями. Найгірший показник у CSR (0.198 секунд), тоді як SSR і SSG мають однакові найнижчі значення (0.191 секунд). Це свідчить про те, що серверні методи рендерингу можуть забезпечити швидший початковий відгук. Гібридні підходи CSR + SSR (0.195 секунд) і CSR + SSG (0.191 секунд) демонструють покращені показники порівняно з CSR. Враховуючи що JavaScript файл в досліді виконує доволі просту та одноманітну логіку (створення компонентів), він не займає багато місця. В більш приближеному до реальних веб-сайтів JavaScript файл був би більшим, та його загрузка виконувалась би довше. Тому TTFB, може виконуватись довше.

Для First Contentful Paint (FCP) спостерігається значна різниця між методами рендерингу. CSR має найгірший результат (1.100 секунд). SSR (0.597 секунд) і SSG (0.584 секунд). Гібридні підходи CSR + SSR і CSR + SSG мають 1 секунду. Маємо висновок, що гібридні підходи покращують виконання загрузки для клієнтської сторони, але значно впливають на Server Side.

Largest Contentful Paint (LCP) показує найбільші відмінності між методами рендерингу. Найгірші показники у SSR (2.200 секунд) та SSG (2.100 секунд), що є результатом великої кількості елементі, які потрібно завантажити. CSR (1.500 секунд) не дивлячись на те що в реальнмоу часі рендерить такуж кількість компонентів. Гібридні підходи CSR + SSR і CSR + SSG мають показник в 1.300 секунд, що значно покращує час LCP, хоча не наблизився до позитивної оцінки в 0,9 секунди.

Для Time to Interactive (TTI) SSR та SSG мають найгірші показники. CSR (1 має кращий результат в 1,4 секунди. Гібридні підходи CSR + SSR і CSR + SSG мають однаковий результат в 2 секунди, що вказує на поєднання затримок обох підходів. Поєднання CSR з іншими типами рендерингу, значно впливають на TTI.

CSR найкраще підходить для простих веб-додатків, які потребують високої інтерактивності, таких як соціальні мережі та панелі управління. Важливою частиною оптимізаціїї є JavaScript для зменшення часу завантаження контенту.

SSR оптимальний для сайтів, які потребують швидкого завантаження та високої видимості в пошукових системах, таких як новинні портали, блоги та e-commerce сайти. Він забезпечує швидке відображення сторінок і покращує індексацію пошуковими системами, але може створювати високе навантаження на сервер.

SSG підходить для статичних сайтів, які рідко змінюються, таких як документація, портфоліо та корпоративні сайти. SSG забезпечує високу швидкість завантаження але не підходить для веб сторінок з контентом який потрібно часто змінювати.

Гібридні методи рендерингу, можуть допомогти в оптимізаціїї складних веб-сторінок.

Client-Side Rendering (CSR) with Prerendering поєднує рендеринг HTML на клієнті з попереднім рендерингом статичного контенту для покращення продуктивності та SEO. Дозволяє сторінці бути динамічною та інтерактивною. Проте, деякі сторінки, які можуть залишатися відносно статичними, рендеряться заздалегідь під час процесу збірки. Ці попередньо рендерені сторінки зберігаються як статичні HTML-файли.

Rehydration з SSR. Спочатку HTML-сторінка рендериться на сервері і відправляється до клієнта. Клієнт отримує відрендерену HTML-сторінку одразу, що покращує початковий час завантаження та SEO. Після завантаження HTML, клієнт завантажує та виконує JavaScript додаючи інтерактивність.

ВИСНОВКИ

У роботі було проведено дослідження основних методів рендерингу веб-сайтів з метою оптимізації для пошукових систем (SEO): Client-Side Rendering (CSR), Server-Side Rendering (SSR) та Static Site Generation (SSG). Кожен з методів було протестовано на прикладі веб-сторінок з різною кількістю компонентів (100, 250, 500, 1000).

Для кожного типу рендерингу було виміряно кілька ключових метрик, які впливають на SEO, а саме: Time to Interactive (TTI), First Contentful Paint (FCP), Time to First Byte (TTFB) та Largest Contentful Paint (LCP). Дані було зібрано за допомогою інструментів GTmetrix та Google PageSpeed Insights.

CSR показав що при невеликій кількості компонентів (100, 250), показники Time to Interactive (TTI) та Largest Contentful Paint (LCP) знаходяться в межах оптимізації. Але при збільшені кількості компонентів (500, 1000), що впливає на час виконання JavaScript збільшує TTI та LCP стрімко що може негативно сказатись на оптимізації SEO.

SSR спочатку показував кращі результати для FCP та TTI, але при збільшені кількості компонентів (250, 500) час до інтерактивності значно збільшився, що було пов'язано з навантаженням на сервер. SSG показав більш стабільні результати при збільшенні кількості компонентів, збільшення основних метрик відбувалось не так стрімко.

Поєднання CSR з SSR або SSG дозволяє зберегти переваги обох підходів, забезпечуючи як високу інтерактивність, так і покращену SEO. Гібридні методи демонструють покращені результати в таких метриках, як TTFB та TTI. Але потребують більш інтеграціїї гібридних методів рендирингу, таких як Prerendering та SSR + Rehydration.

Результати дослідження демонструють, що вибір методу рендерингу залежить від кількості елементів які будуть рендериться включати важливі теги для SEO (для 500 компонентів метрики значно збільшувались) на веб-сторінці, загрузки та виконання JavaScript на клієнті.

Вибір оптимального методу рендерингу суттєво впливає на індексацію сайту в базах даних пошуковиків. Тому працючи з CSR потрібно враховувати цю проблему і оптимізовувати за допомогою Code splitting та Lazy loading, для отримання LCP (до 1,2 секунди) до та FID (до 2,5 секунд)

Загалом, результати показують, що найбільш ефективним підходом до рендерингу веб-сторінок з точки зору SEO є використання SSR та SSG, які демонструють стабільні показники продуктивності при великій кількості компонентів. CSR виявляється менш ефективним для складних та інтерактивних веб-сторінок, але може бути корисним для невеликих, швидких додатків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Most popular technologies webframe. URL: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks (дата звернення: 20.05.2024).

2. Erwin Hofman, (2022). Parsing and rendering process simplified. URL: https://www.erwinhofman.com/blog/parsing-and-rendering-process-simplified/ (дата звернення: 20.03.2024).

3. Code splitting. URL: https://developer.mozilla.org/en-US/docs/Glossary/Code_splitting (дата звернення: 20.05.2024).

4. Lazy loading. URL: https://developer.mozilla.org/en-US/docs/Web/Performance/Lazy_loading (дата звернення: 20.05.2024).

5. Content delivery network. URL: https://uk.wikipedia.org/wiki/Content_delivery_network (дата звернення: 20.05.2024).

6. Lighthouse - Google Plugin. URL: https://chromewebstore.google.com/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=ru&pli=1 (дата звернення: 20.05.2024).

7. webpagetes. URL: https://www.webpagetest.org (дата звернення: 20.03.2024).

8. Maksym Rudnyi, (2023). Порівнюємо способи генерації сторінок: CSR, SSR, SSG, ISR. URL: https://dou.ua/forums/topic/41585/ дата звернення: 20.05.2024).

9. Павленко Ю.С, (2022). Пошукова оптимізація, технології та сервіси веб аналітики. URL: https://evnuir.vnu.edu.ua/bitstream/123456789/21864/1/SEO.pdf (дата звернення: 20.05.2024).

10. Natascha Fher, (2023). Web rendering: how does it affect SEO?. URL: https://www.geotelecom.mx/en/render-web-as-affecting-web-as-affecting-seo/ (дата звернення: 20.05.2024).

11. Qamar Zaman, (2023). What is indexing in SEO?. URL: https://www.linkedin.com/pulse/what-indexing-seo-how-works-qamar-zaman (дата

звернення: 20.05.2024).

12. Philip Walton, (2023) User-centric performance metrics. URL: https://web.dev/articles/user-centric-performance-metrics (дата звернення: 20.03.2024).

13. Брухтій С.С., Ревенчук І.А.Тенденція розвитку методів рендеренгу web-сайтів для пошукової оптимізації сайтів (SEO): матеріали 28-го Міжнарод. молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 6., Харків: ХНУРЕ. 2024.С.918-919. (DOI: https://doi.org/10.30837/IYF.IIS.2024.918).

14. Мандрика М. Ревенчук І.А. Дослідження методів та інструментів мониторинга веб-сервісів: матер XV Міжнар наук.-практ конф. "Innovative Approaches to the Progressive Solution of Scientific Research Problems". Іспанія 27-29.03.2024. P.59-61.

15. Моруга Д.І., Ревенчук І.А. Unified Approach for Development of Graphical User Interface in Cross-platform Applications: XXII International Scientific and Practical Conference «Modern Scientific Research: Theoretical and Practical Aspects». Oslo, Norway 8-10.05.2024. P.73-76.