МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем
_____Юрій КОНДРАТЕНКО
«____»_____ 20___ р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ІНФОРМАЦІЙНА СИСТЕМА АНАЛІЗУ ТА МОНІТОРИНГУ ПРОДУКТИВНОСТІ WEB-ЗАСТОСУНКІВ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Володимир БЛИНЦОВ
«___» _____червня____ 2025 р.

Керівник канд. пед. наук, доцент _____ Надія БОЛЮБАШ
«____» _____червня____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем
_____ Юрій КОНДРАТЕНКО
«____» _____ 2025 р.

З А В Д А Н Н Я

на кваліфікаційну роботу здобувача

Блинцова Володимира Володимировича

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи «Інформаційна система аналізу та моніторингу продуктивності web-застосунків».

Керівник роботи: Болюбаш Надія Миколаївна, доцент кафедри інтелектуальних інформаційних систем, канд. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «18» листопада 2024 р. № _321_

2. Строк представлення кваліфікаційної роботи студентом «_18_» червня_ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: система аналізу продуктивності вебресурсів на базі швидкого консольного застосунку з паралельним запуском операцій для автоматизованого тестування, інтегрованого з вебзастосунком для візуалізації результатів аналізу; предметна сфера аудиту продуктивності web-застосунків в рамках APM-систем, методи аналізу й

моніторингу їх продуктивності, а також сукупність вебресурсів для тестування системи.

4. Перелік питань, що підлягають розробці: проведення аналізу та дослідження сучасних підходів і програмних рішень у сфері аудиту продуктивності вебресурсів; обґрунтування вибору стеку технологій та методів аналізу й моніторингу продуктивності вебзастосунків для розробки інформаційної системи; здійснення проєктування, програмної реалізації і тестування системи аналізу й моніторингу продуктивності web-застосунків та оцінка її якості.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

Керівник роботи _____ Надія БОЛЮБАШ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Володимир БЛИНЦОВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «_18_» ___грудня__ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

кваліфікаційної роботи

Тема: Інформаційна система аналізу та моніторингу продуктивності web-застосунків

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми КРБ. Подання заяви на затвердження теми КРБ	25.09.2024	17.12.2024	Виконано
2	Отримання завдання на виконання КРБ	18.12.2024	20.12.2024	Виконано
3	Аналіз предметної сфери аудиту продуктивності web-застосунків та постановка задачі	21.12.2024	30.01.2025	Виконано
4	Огляд літератури за темою дослідження підходів до аудиту продуктивності web-застосунків, виявлення методів, моделей та метрик аналізу продуктивності вебресурсів	31.01.2025	17.02.2025	Виконано
5	Вибір технологій та інструментальних засобів розробки системи	18.02.2025	28.02.2025	Виконано
6	Створення дизайну, проєктування та програмна реалізація, тестування	01.03.2025	15.04.2025	Виконано
7	Робота над розділами кваліфікаційної роботи	16.04.2025	15.05.2025	Виконано
8	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	Виконано
9	Корегування роботи за результатами попереднього захисту	17.05.2025	29.05.2025	Виконано
10	Другий попередній захист КР	30.05.2025	30.05.2025	Виконано
11	Остаточне оформлення КР та слайдів доповіді до захисту	31.06.2025	10.06.2025	Виконано
12	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	11.06.2025	11.06.2025	Виконано

Керівник роботи _____ Надія БОЛЮБАШ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Володимир БЛИНЦОВ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану

« 30 » січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 402 ЧНУ ім. Петра Могили

Блинцова Володимира Володимировича

на тему: «**ІНФОРМАЦІЙНА СИСТЕМА АНАЛІЗУ ТА МОНІТОРИНГУ ПРОДУКТИВНОСТІ WEB-ЗАСТОСУНКІВ**»

Дана кваліфікаційна робота присвячена розробці системи аналізу та низькорівневого тестування продуктивності web-застосунків на етапі їх експлуатації з урахуванням геолокалізації користувачів ресурсу. Що є актуальним в умовах високих темпів цифровізації усіх сфер сучасного суспільства, оскільки ключем для успіху вебресурсів, які розробляють IT-компанії, є їх висока продуктивність на різних платформах у будь-якій точці земної кулі.

Об’єкт роботи – процес аудиту вебресурсів.

Предмет роботи – програмні засоби та методи аналізу і моніторингу продуктивності web-застосунків.

Мета роботи – підвищення ефективності моніторингу продуктивності web-застосунків шляхом розробки інформаційної системи із вбудованими методами відстеження показників продуктивності та формування рекомендацій їх оптимізації.

Структура кваліфікаційної роботи включає вступ, чотири розділи, висновки та додатки. У першому розділі розкрито теоретичні аспекти аналізу та моніторингу продуктивності вебресурсів, досліджено сучасні підходи та програмні рішення у цій сфері. У другому розділі обґрунтовано вибір стеку технологій та методів аналізу й моніторингу продуктивності вебзастосунків для розробки інформаційної системи. У третьому розділі розкрито проєктування інформаційної системи. У четвертому розділі описано програмну реалізацію і тестування системи аналізу й моніторингу продуктивності web-застосунків та оцінено її якість.

Кваліфікаційна робота містить 103 сторінок (без додатків), 44 рисунків, 7 таблиць, 36 джерел та 4 додатків.

Ключові слова: оцінка продуктивності, управління продуктивністю застосунків, SEO аудит, технічний аналіз.

ABSTRACT

to the qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

Blyntsov Volodymyr

«INFORMATION SYSTEM FOR ANALYZING AND MONITORING PERFORMANCE OF WEB APPLICATIONS»

This bachelor's qualification work is devoted to the development of a system for analyzing and low-level testing the performance of web applications at the stage of their operation, taking into account the geolocation of resource users. Which is relevant in the conditions of high rates of digitalization of all spheres of modern society, since the key to the success of web resources developed by IT companies is their high performance on various platforms anywhere in the world.

Object of work – web resource audit process.

Subject of work – software and methods for analyzing and monitoring web application performance.

The purpose of this work is to increasing the efficiency of web application performance monitoring by developing an information system based with built-in methods for tracking performance indicators and generating recommendations for their optimization.

The structure of the bachelor's work includes an introduction, four sections, conclusions and appendices. The first section reveals the theoretical aspects of analyzing and monitoring the performance of web resources, explores modern approaches and software solutions in this area. The second section justifies the choice of a stack of technologies and methods for analyzing and monitoring the performance of web applications for the development of an information system. The third section reveals the design of the information system. The fourth section describes the software implementation and testing of the system for analyzing and monitoring the performance of web applications and assesses its quality.

The qualification work contains 103 pages (without appendices), 44 figures, 7 tables, 36 sources and 4 appendices.

Keywords: performance score, application performance management (APM), SEO audit, technical analysis.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ТА – Технічний аналіз

ПЗ – Програмне забезпечення

API – Application Programming Interface

APM – Application Performance Management

CI/CD – Continuous Integration / Continuous Delivery

CLI – Command Line Interface

CLS – Cumulative Layout Shift

FCP – First Contentful Paint

INP – Interaction to Next Paint

LCP – Largest Contentful Paint

SEO – Search Engine Optimization

TTFB – Time to First Bite

ВСТУП

Актуальність. У сучасних умовах цифровізації економіки вебресурси стають основною платформою для надання послуг, продажу товарів та взаємодії з клієнтами. Вебзастосунки використовуються як у простих системах керування контентом (CMS), так і у складних порталах, CRM-системах, онлайн-магазинах, освітніх платформах, соціальних мережах та інших інтерактивних сервісах. Зростання ролі вебтехнологій у світовій економіці підвищує попит на швидкодію та стабільність вебресурсів, адже саме ці показники визначають комфорт користування та рівень задоволеності клієнтів.

Продуктивність вебзастосунку безпосередньо впливає на користувацький досвід: кілька секунд затримки завантаження сторінки може призвести до втрати клієнтів і зниження конверсії. У висококонкурентному середовищі будь-яке уповільнення здатне відштовхнути потенційних користувачів, а ресурси, що опинилися на другій або третій сторінці пошукової видачі, можуть залишитися непоміченими. Для підтримки системи у належному стані та забезпечення високої продуктивності використовується відповідний інструмент, що є частиною APM систем, який дозволяє здійснювати аналіз та моніторинг продуктивності вебресурсів у реальному часі, забезпечуючи виявлення та усунення технічних недоліків на ранніх етапах.

З метою забезпечення точного та комплексного аналізу доцільно застосовувати системи з інтегрованим підходом до збору та аналізу даних. Інформаційна система повинна не лише фіксувати показники швидкодії та ефективності, а й виявляти закономірності у їх зміні з часом, що дозволить своєчасно вживати заходів для підтримки стабільності вебресурсу. Алгоритми аналізу мають базуватися на сучасних методах вимірювання показників продуктивності, включаючи моделі обробки даних, що забезпечать гнучкість та точність результатів аудиту. Проте можливості інтеграції різних алгоритмів у єдине середовище не є дослідженими у повній мірі.

Це обумовило **мету роботи**, яка полягає у підвищенні ефективності моніторингу продуктивності web-застосунків шляхом розробки інформаційної системи із вбудованими методами відстеження показників продуктивності та формування рекомендацій їх оптимізації.

Відповідно до поставленої мети було сформульовано **завдання**:

– провести аналіз, дослідити сучасні підходи та програмні рішення у сфері аудиту продуктивності вебресурсів;

– обґрунтувати вибір стеку технологій та методів аналізу й моніторингу продуктивності вебзастосунків для розробки інформаційної системи;

– здійснити проєктування, програмну реалізацію і тестування системи аналізу й моніторингу продуктивності web-застосунків та оцінити її якість.

Об’єкт роботи – процес аудиту вебресурсів.

Предмет роботи – програмні засоби та методи аналізу і моніторингу продуктивності web-застосунків.

Методологічною основою дослідження є загальнонаукові методи, методи технічного аналізу продуктивності вебресурсів, методи оцінки показників SEO, Performance Score та Best Practise вебресурсів, які дозволили комплексно вивчити предмет і об’єкт роботи, дослідити основні аспекти аудиту продуктивності вебресурсів.

Практичне значення отриманих результатів полягає в тому, що CLI-застосунок розробленої інформаційної системи дозволяє підвищити ефективність аналізу та моніторингу вебресурсів у різних середовищах і на різних апаратних платформах і може бути інтегрованим в інші інструменти аудиту.

Структура кваліфікаційної роботи. Відповідно до мети, завдань і предмета роботи, кваліфікаційна робота складається із вступу, чотирьох розділів, висновку, списку використаних джерел та 4 додатків. Загальний обсяг роботи – 115 сторінок, із них основного тексту – 103 сторінок. Кількість використаних джерел – 36.

1 ТЕОРЕТИЧНІ ЗАСАДИ АНАЛІЗУ ТА МОНІТОРИНГУ ПРОДУКТИВНОСТІ WEB-РЕСУРСІВ

1.1 APM-системи управління продуктивністю вебзастосунків

У сучасному світі цифрових технологій, де кожна мілісекунда має значення, безперебійна робота вебзастосунків стала критично важливим фактором для процвітання будь-якого бізнесу. Сучасні вебресурси є не лише віртуальними вітринами, вони виступають ключовими каналами комунікації та взаємодії з клієнтами. Через ці цифрові платформи користувачі знайомляться з брендами, отримують інформацію, здійснюють покупки та формують свої враження про компанії. Будь-які перебої у функціонуванні сайту, зависання чи повільне завантаження сторінок можуть миттєво відштовхнути потенційних клієнтів і призвести до відчутних фінансових втрат.

Значення продуктивності вебсайтів неодноразово підкреслювалося провідними експертами галузі по всьому світу. Зокрема на щорічній конференції Performance.now, яка проводиться в Амстердамі, експерти наголошують на критичній необхідності покращення швидкості та надійності вебресурсів, адже проблеми з продуктивністю виходять далеко за межі негативного впливу на користувацький досвід. Такі проблеми безпосередньо позначаються на ключових бізнес-метриках, включаючи позиції в пошуковій видачі, коефіцієнт конверсії, клікабельність реклами та рентабельність інвестицій у рекламу. Таким чином, повільний вебсайт не лише зменшує потік органічних відвідувачів, але й знижує ефективність платних рекламних кампаній, одночасно збільшуючи витрати на залучення нових клієнтів [1].

Згідно даних використання Chrome, 90% часу користувач проводить на вебсторінці після її завантаження. Тому досить важливим є показник, який відображає швидкість відгуку ресурсу на дії користувача. Зокрема, аналіз, проведений Google, засвідчує, що навіть незначна затримка у 500 мс при

завантаженні сторінки призводить до втрати приблизно 20% трафіку [2]. У свою чергу, досвід Amazon показує, що кожні 100 мілісекунд затримки завантаження обертаються для компанії втратою 1% доходу [3]. Дослідження Mozilla також виявило, що значна частина користувачів залишає сайт, якщо сторінка не завантажується протягом критично короткого проміжку часу – від однієї до п'яти секунд [4]. Ці дані яскраво ілюструють прямий зв’язок між швидкістю роботи вебресурсу та його здатністю залучати й утримувати клієнтів, а отже, і генерувати прибуток. Чим продуктивніше є вебсайт, тим більше ймовірність, що потенційні користувачі залишаться та здійснять покупку або взаємодію з вмістом [5]. Навпаки, повільне завантаження вебсторінки може призвести до фрустрації й відмови від відвідування сайту, що, в свою чергу, негативно вплине на конверсії та загальні фінансові результати бізнесу. Отже, інвестування в оптимізацію ключових метрик вебресурсу не лише підвищує користувацький досвід, але й безпосередньо сприяє зростанню доходів компанії. Крім того, збереження клієнтів і підвищення їхньої лояльності можливе саме завдяки ефективності та продуктивності сайту, що в умовах сучасного висококонкурентного ринку є надзвичайно важливим фактором успіху.

Для забезпечення високої продуктивності вебресурсів існують спеціалізовані системи, які реалізують управління продуктивністю застосунків (англ. Application Performance Management, APM) (рис. 1.1). *Управління продуктивністю застосунків* має на меті забезпечити необхідні процеси та інструменти для отримання постійної та актуальної картини відповідних показників продуктивності під час операцій, а також для підтримки виявлення та вирішення інцидентів, пов’язаних з продуктивністю. Тому APM-системи є ключовими в аналізі, управлінні продуктивністю вебзастосунків та забезпеченні їх ефективної роботи. APM-системи не лише спостерігають за станом вебзастосунку, а й детально аналізують поведінку користувачів, що дозволяє зрозуміти, як саме відвідувачі взаємодіють з контентом. Це, у свою чергу, відкриває нові можливості для виявлення, діагностики та швидкого вирішення проблем продуктивності на основі зібраних

даних. Такий проактивний підхід не лише підтримує стабільність системи, а й постійно оптимізує досвід кінцевого користувача, що є надзвичайно важливим в умовах жорсткої конкуренції.

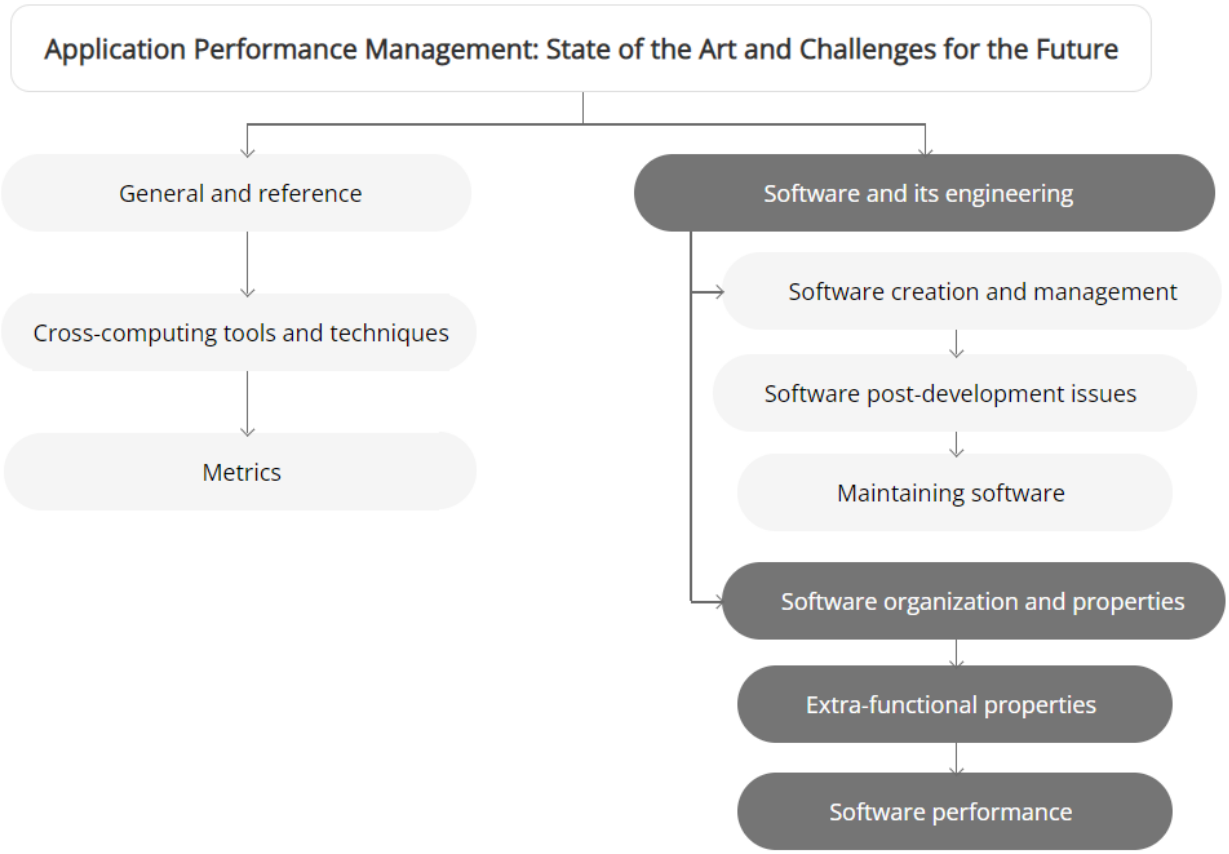


Рисунок 1.1 – Application Performance Management: сучасний стан та виклики майбутнього

З технічної точки зору, сучасні APM-рішення охоплюють декілька рівнів моніторингу, кожен з яких відіграє критичну роль у загальному управлінні продуктивністю. Це включає в себе моніторинг серверного середовища, де відстежуються ресурси, доступність та працездатність серверів, а також збір метрик, таких як завантаження сторінок, час відгуку та частота виникнення помилок. Аналіз цих показників дозволяє не лише виявляти можливі вузькі місця, але й оцінювати загальну ефективність системи. Крім того, APM-системи здійснюють детальний аналіз контенту, що завантажується, та його доступності для

користувача, що, у свою чергу, забезпечує більш глибоке розуміння впливу різних елементів вебзастосунку на загальний користувацький досвід.

APM-системи охоплюють численні аспекти роботи програмного забезпечення, що дозволяє розглядати їх як універсальні інструменти для підтримки високої продуктивності. З метою виконання завдань аналізу та моніторингу продуктивності були розроблені окремі програми, які є складовими частинами екосистем APM. Такі програми, котрі є частиною APM, не обмежуються лише технічними показниками, такими як час завантаження, відгук сервера або частота помилок. Вони також враховують фактори, що впливають на успішність вебресурсу в цілому. Наприклад, системи моніторингу включають аналіз SEO-оптимізації, що дозволяє оцінювати, як добре вебзастосунок відповідає вимогам пошукових систем, що має прямий вплив на його видимість у результатах пошуку.

Окрім цього, APM-системи враховують поведінкові фактори, такі як взаємодія користувачів з контентом, що дає можливість виявити, які елементи вебзастосунку надають найбільшу цінність відвідувачам, а які можуть потребувати поліпшення. Це особливо важливо в умовах сучасного ринку, де високий рівень мобільної адаптивності стає критично важливим. APM-системи також забезпечують відповідність ресурсів актуальним вимогам користувачів і тенденціям цифрового середовища.

Завдяки цьому комплексному підходу, аналіз продуктивності в рамках APM систем створює цілісну картину, в якій враховуються як технічні метрики, так і бізнес-показники [6]. Це дозволяє компаніям та бізнесам не лише підтримувати стабільність своїх вебзастосунків, а й приймати стратегічні рішення, спрямовані на покращення користувацького досвіду і, відповідно, оптимізацію бізнес-процесів та зростання фінансових результатів.

Проте, незважаючи на наявність великої кількості інструментів для вимірювання продуктивності вебсторінок, більшість з них мають суттєві обмеження. Зокрема, вони покладаються на синтетичні тести, які не завжди відображають реальний досвід користувачів. В умовах, коли вебресурси стають

дедалі складнішими, із великою кількістю зовнішніх залежностей та інтеграцій, традиційні інструменти часто не здатні адекватно проаналізувати вебзастосунок [7]. Крім того, веб інструменти можуть вимагати значного часу на обробку даних, оскільки результати часто генеруються з урахуванням великої кількості факторів, що ускладнює швидке прийняття рішень. Багато з цих сервісів також є платними, що може стати суттєвим фінансовим навантаженням при їх використанні.

Географічні обмеження, пов’язані з серверами таких ресурсів, можуть призвести до нерелевантних результатів, які не відображають фактичного досвіду користувачів у конкретній місцевості. Водночас, одним із суттєвих обмежень вебверсій APM-застосунків є недостатня зручність інтерфейсів та не оптимальна візуалізація даних для глибокого аналізу продуктивності. Це створює труднощі як для нетехнічних користувачів, яким важливо швидко оцінити загальний стан системи, так і для технічних спеціалістів, які потребують деталізованих звітів і засобів для виявлення причин проблем. Відсутність гнучких налаштувань подання інформації та інтерактивних інструментів ускладнює роботу з метриками, що, своєю чергою, уповільнює процес ухвалення рішень щодо оптимізації системи.

З огляду на зазначене, розробка консольного застосунку, який має інтерфейс командного рядка (англ. Command Line Interface, CLI), для локального аналізу продуктивності вебресурсів, як складової частини APM-систем, є не лише перспективним, а й стратегічно обґрунтованим рішенням. Такий підхід відповідає сучасним вимогам до гнучкості, масштабованості та ефективності інструментів технічного моніторингу. Консольний застосунок (CLI-застосунок) пропонує цілу низку важливих переваг, зокрема – високу швидкість роботи, незалежність від зовнішніх серверів, можливість повної автоматизації процесу аналізу, а також максимальну адаптацію під специфіку конкретного середовища.

На відміну від традиційних веборієнтованих APM-рішень, CLI-інструмент не потребуватиме мережевого підключення до сторонніх сервісів для обробки й візуалізації даних. Це дозволяє уникнути затримок, пов’язаних із передачею

інформації через мережу, що особливо актуально в умовах високого навантаження або обмеженої пропускної здатності каналу. CLI-застосунок виконує всі обчислення локально, що забезпечує отримання результатів у реальному часі та підвищує загальну продуктивність системи.

Крім того, такий підхід дає змогу інтегрувати інструмент безпосередньо в CI/CD (англ. Continuous Integration / Continuous Delivery), забезпечуючи автоматичний запуск перевірок на етапах деплою або тестування, а також моніторинг стану системи в реальному часі. Технологія неперервної інтеграції та доставки CI/CD є технологією автоматизації тестування та доставки нових модулів проєкту, що розробляється, зацікавленим сторонам: розробникам, аналітикам, інженерам якості та кінцевим користувачам (рис. 1.2). Це дозволяє виявляти критично низькі показники продуктивності ще до релізу продукту в продакшн (перевірки замовником), що знижує ризики та вартість усунення помилок на пізніших етапах життєвого циклу програмного забезпечення [8].

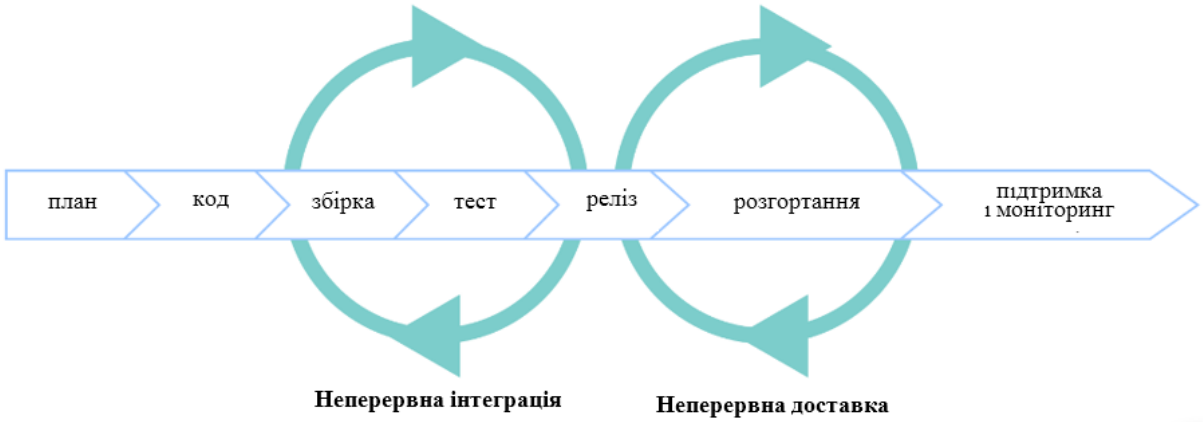


Рисунок 1.2 – Етапи CI/CD

Ще однією перевагою є низький рівень залежності від графічного інтерфейсу, що робить CLI-рішення ідеальними для серверного використання, у тому числі в середовищах з обмеженими ресурсами.

Таким чином, створення консольного інструменту для локального аналізу продуктивності вебресурсів дозволяє досягти високого рівня ефективності, адаптивності та контролю над технічними метриками, забезпечуючи як глибоке

занурення в проблематику продуктивності, так і оперативність реагування на виявлені відхилення, оскільки такий підхід зменшує час, необхідний для проведення аналізу. Це дозволяє швидше виявляти й усувати проблеми, покращуючи загальний досвід взаємодії користувача з вебресурсом. Розробка CLI застосунку забезпечує гнучкість і адаптивність, оскільки користувачі можуть налаштовувати аналіз відповідно до своїх конкретних потреб і вимог.

1.2 Огляд програмних рішень та досліджень у сфері аналізу продуктивності вебзастосунків

У листопаді 2024 року кількість активних вебсайтів у світі перевищила позначку в 1,1 мільярда, і ця цифра продовжує зростати [9]. Така масштабність створює вкрай висококонкурентне цифрове середовище, де кожна секунда завантаження, кожен клік і кожен відгук системи мають значення. У цих умовах продуктивність вебзастосунку стала не просто технічним аспектом, а стратегічним фактором, що визначає конкурентоспроможність ресурсу. Навіть незначні затримки можуть призвести до втрати користувачів, а отже, і прибутку. Компанії змушені не лише створювати якісний контент чи дизайн, а й забезпечувати швидкий, стабільний та безперебійний користувацький досвід на всіх пристроях і в будь-яких мережевих умовах.

Із появою інструментів вебаналітики на початку 2000-х років аналіз даних став стратегічним елементом оптимізації сайтів, оскільки APM системи та окремі технології аналітики продуктивності дозволили бізнесу отримувати детальну інформацію про різні аспекти роботи їхніх вебресурсів. Дані про трафік, поведінку користувачів і видимість сайту стали критично важливими для бізнесу. Інструменти вебаналітики забезпечують можливість приймати обґрунтовані рішення на основі даних, а не інтуїції, що значно знижує ризик помилок і дозволяє оптимізувати зусилля. Це безумовно підвищує ефективність оптимізації сайту, адже дозволяє бізнесам не лише виявляти слабкі місця, а й швидко реагувати на

недостатньо продуктивні аспекти продукту, що, в свою чергу, призводить до покращення користувацького досвіду.

Звернемося до досліджень у цій сфері, щоб визначити методи оцінки продуктивності сайту, виявити основні фактори, які впливають на продуктивність вебресурсів, а також ідентифікувати ключові показники, що відображають ефективність конкретних оптимізаційних заходів. Це дозволить отримати комплексне розуміння вимог до продуктивності, визначити потенційні проблеми та можливості для вдосконалення, а також створити основу для подальшої реалізації інформаційної системи, спрямованої на аналіз та моніторинг продуктивності вебресурсів.

На 19-й конференції AMCIS у 2013 році було представлено дослідження, метою якого було визначення ключових показників, що безпосередньо впливають на окупність бізнесу (англ. Return on Investment, ROI) та ефективність роботи вебресурсу KPI (англ. Key Performance Indicator). ROI як коефіцієнт окупності є показником повернення всіх інвестицій. Серед KPI розрізняють KPI ефективності та KPI продуктивності. *KPI ефективності* – індикатори, що відображають, наскільки результат виправдовує витрачені ресурси. *KPI продуктивності* – це показники, які демонструють, як результат співвідноситься з часом, необхідним для його досягнення.

Ефективність бізнес-процесів у вебзастосунках визначається дотриманням цілого комплексу стандартів і рекомендацій, відомих як Best Practices. Вони охоплюють критично важливі аспекти – доступність контенту, продуктивність роботи (на рівні як frontend, так і backend), безпеку даних та архітектурну стійкість системи. Їх підтримка та контроль є надійним фундаментом для забезпечення окупності бізнесу. До основних показників Best Practices вебресурсів відносять оцінку доступності, геолокації та безпеки, використання cookies та застарілих API, коректне співвідношення розмірів зображень, оцінку налаштувань метатегів та кодування символів, декларацію типу контенту [19].

У контексті підвищення ефективності роботи вебресурсу важливе значення має SEO (англ. Search Engine Optimization) оптимізація, яка охоплює як органічний, так і платний пошуковий трафік. Дослідження показують, що висока позиція сайту у видачі пошукових систем значно підвищує відвідуваність ресурсу: близько 75% користувачів не переглядають результати за межами першої сторінки. На рисунку 1.3 наведено приклад показників органічного трафіку сайту ЧНУ ім. П. Могили, визначених на 26.04.2025.

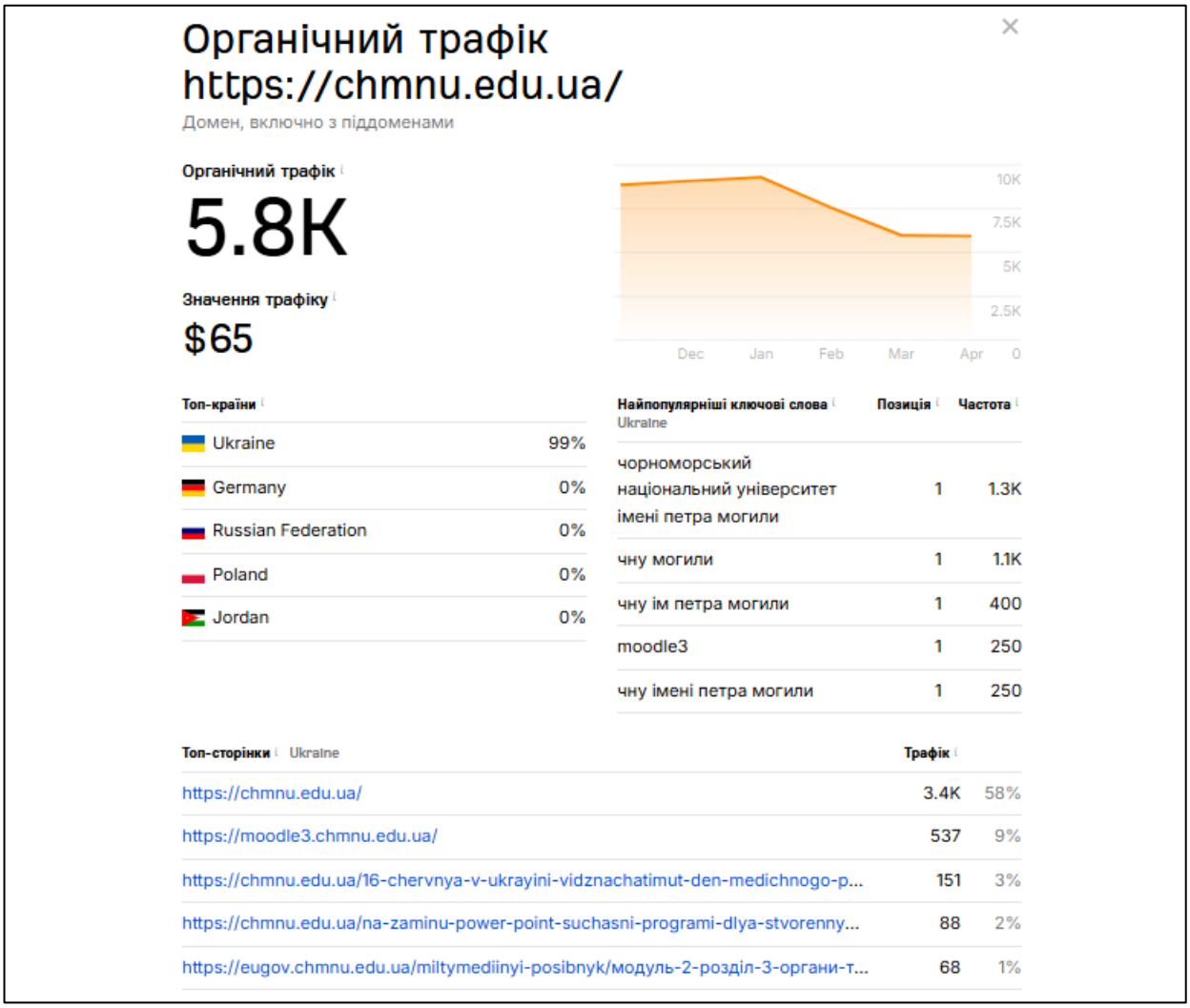


Рисунок 1.3 – Показники органічного трафіку сайту ЧНУ ім. П. Могили

Для ефективної SEO-стратегії використовуються наступні техніки оптимізації [10]:

– додавання meta title та meta description тегу;

– створення карти сайту (sitemap) для кращої індексації;

– використання заголовків <h1>-<h6> (де <h1> повинен бути представлений в кількості 1 тег на сторінку);

– налаштування індексного файлу robots.txt та його оптимізація для пошукових роботів (web crawling);

– оптимізація та використання атрибута alt для зображень.

Теж саме підтверджує більш свіже дослідження, предметом якого був вплив SEO факторів на залучення користувачів [11]. Результати цього дослідження свідчать про те, що ефективна оптимізація вебсайту для пошукових систем значно збільшує його видимість, що, в свою чергу, веде до зростання органічного трафіку. Залучення нових користувачів через поліпшення позицій у пошуковій видачі не лише покращує відвідуваність сайту, але й безпосередньо впливає на бізнес-результати, зокрема на збільшення доходів. Залучення нових клієнтів через активну SEO-стратегію формує довготривалу присутність бренду в онлайн-просторі. Орієнтований на користувача підхід до SEO не тільки збільшує корисні взаємодії на сайті, але також укріплює довіру та лояльність аудиторії, що сприяє покращенню ключових показників підприємства та підвищенню впізнаваності бренду.

Результатом обох досліджень є висновок, що ці заходи дозволяють підвищити помітність ресурсу у пошукових системах та забезпечують зростання конверсій і бізнес-показників.

Окреме дослідження, проведене науковцями з Університету Західної Аттики (Греція), акцентує увагу на важливість оптимізації першого досвіду взаємодії користувача з вебресурсом. Саме швидкість завантаження та коректне відображення контенту на початкових етапах взаємодії мають вирішальний вплив на сприйняття ресурсу. Ці фактори безпосередньо впливають на формування лояльності користувачів, продуктивність бізнесу та загальний імідж бренду [7]. Дослідження показує, що при недостатньому врахуванні цих аспектів, користувачі

можуть швидко покинути сайт, обираючи конкурентні ресурси, які забезпечують кращий досвід.

Попри значну кількість досліджень у цій сфері, все ще бракує цілісного підходу до оцінки продуктивності вебсторінок із урахуванням реальних сценаріїв використання, масштабованості та інтерактивної поведінки користувача. Проте з метою уніфікації показників для оцінки продуктивності та точнішого вимірювання, були визначені ключові метрики продуктивності завантаження контенту вебресурсу, які вимірюють реальний досвід користувача стосовно швидкості завантаження, інтерактивності та візуальної стабільності сторінки [12]:

– TTFB (Time to First Bite) – час, необхідний для отримання першого байта відповіді сервера;

– FCP (First Contentful Paint) – момент, коли браузер вперше відображає будь-який вміст (текст, зображення тощо);

– LCP (Largest Contentful Paint) – час відображення найбільшого за розміром елемента сторінки;

– CLS (Cumulative Layout Shift) – стабільність макета (сукупний зсув), що показує, наскільки змінюється положення елементів при завантаженні;

– INP (Interaction to Next Paint) – час відгуку на взаємодію користувача до моменту візуального оновлення.

Ці показники дозволяють об’єктивно оцінити загальний користувацький досвід, включаючи швидкість, стабільність інтерфейсу та якість інтерактивності. Їх впровадження у систему аналізу продуктивності вебзастосунків дозволить зробити інформаційну систему більш чутливою до реальних проблем користувачів і сприятиме глибшому розумінню впливу продуктивності на успішність продукту [13].

Інше актуальне дослідження у сфері продуктивності вебресурсів зосереджується на важливості комплексного контролю та аудиту технічних характеристик, що мають прямий вплив на користувацький досвід. Автор дослідження підкреслює, що продуктивність не обмежується лише швидкістю

завантаження, а включає також якість реалізації frontend архітектури, безпеки та доступності [14]. У роботі акцентовано увагу на важливості моніторингу таких показників:

– використання застарілих API (англ. deprecated API): впливає на стабільність і безпеку вебзастосунку;

– оптимізація зображень: дозволяє зменшити об’єм трафіку та покращити швидкість завантаження;

– перевірка сторонніх cookie (англ. third-party cookies): необхідна для забезпечення конфіденційності та відповідності вимогам безпеки;

– правильне визначення HTML doctype: впливає на коректну інтерпретацію сторінки браузером;

– визначення кодування символів (англ. charset): забезпечує правильне відображення текстової інформації;

– розмір шрифтів (англ. legible font sizes): критично важливий для доступності контенту, особливо на мобільних пристроях;

– наявність source maps: полегшує дебаг і підтримку коду, зокрема при використанні збірників (webpack, parcel тощо);

– перевірка безпеки (англ. security checks): включає дотримання політик HTTPS, CSP, та інші елементи захисту;

– аудит доступності (англ. accessibility): визначає, наскільки ресурс зручний для людей з обмеженими можливостями.

Аналіз цих параметрів дає змогу виявляти технічні недоліки на ранніх етапах розробки (або вже в цілісних системах), забезпечуючи стабільну, швидку та безпечну роботу вебресурсу з урахуванням потреб усіх категорій користувачів.

Отже, для ефективного аналізу та моніторингу продуктивності вебресурсів доцільно поєднувати показники технічних аспектів, best practices, performance-метрик та SEO-оптимізації, що разом забезпечують комплексну оцінку якості та ефективності роботи сайту.

Окрім зазначених ключових метрик та аспектів, що забезпечують комплексну оцінку якості та ефективності роботи сайту, варто відзначити, що дослідники у низці робіт з аналізу конкретних вебресурсів звертають увагу і на інші, більш специфічні параметри. Наприклад, деякі дослідження розширюють розуміння SEO. У сфері Performance також використовуються різноманітні метрики, проте, з огляду на численні дослідження та рекомендації провідних експертів, зокрема Google, у кваліфікаційній роботі було обрано ряд ключових показників, які визнано найбільш релевантними для оцінки реального користувацького досвіду.

У процесі аналізу вебресурсів надзвичайно важливим етапом є проведення детального технічного аналізу. Технічний аудит дозволяє не лише оцінити поточний стан програмного продукту з точки зору коректності його роботи, а й глибоко проаналізувати внутрішню архітектуру, використані технології та підходи до розробки. Це дає змогу виявити архітектурні недоліки, неефективні рішення, ризики для масштабованості, стабільності та продуктивності системи. Також технічний аудит дозволяє ідентифікувати критичні місця, які можуть стати причиною збоїв чи безпекових загроз у майбутньому.

Таким чином, незважаючи на можливість аналізу широкого спектру метрик, для цілей даної роботи було обрано саме ті показники з категорій Performance, SEO, Best Practices та технічного аналізу, які відповідають критеріям комплексного, повноцінного та всебічного аналізу (перелічені вище). Цей вибір ґрунтується на їхній здатності підтримувати систему в критично важливому стані, виступати основними локаторами ключових проблем та відображати реальний стан вебресурсу, що підтверджується дослідженнями науковців та провідних галузевих експертів.

Перейдемо до аналізу програмних рішень у сфері моніторингу продуктивності вебресурсів. На сьогоднішній день існує багато сервісів, які перевіряють продуктивність вебзастосунків. Деякі з них є безкоштовними, проте більшість вимагають підписки або оплати за доступ до розширених функцій. Ці інструменти дозволяють комплексно оцінювати технічний стан ресурсів,

відповідність сучасним стандартам та виявляти вузькі місця, які можуть негативно вплинути на досвід користувача та окупність бізнесу.

Здійснимо огляд найпоширеніших інструментів для проведення аудиту вебресурсів, до яких відносять: SE Ranking, WebPageTest, GTmetrix, Pingdom Tools та PageSpeed Insights. Проаналізуємо їх можливості на прикладі дослідження продуктивності вебресурсу у сфері трейдингу, що активно розвивається – https://justmarkets.com. Для проєктів у цій галузі показники продуктивності мають критичне значення, оскільки вони безпосередньо впливають на рівень конверсії та залучення нових клієнтів, що є важливим для зростання доходів.

Після проведення аналізу за допомогою платформи SE Ranking, основний результат стосувався лише показників продуктивності (рис. 1.4), отриманих через стороннє API від Google (Lighthouse).

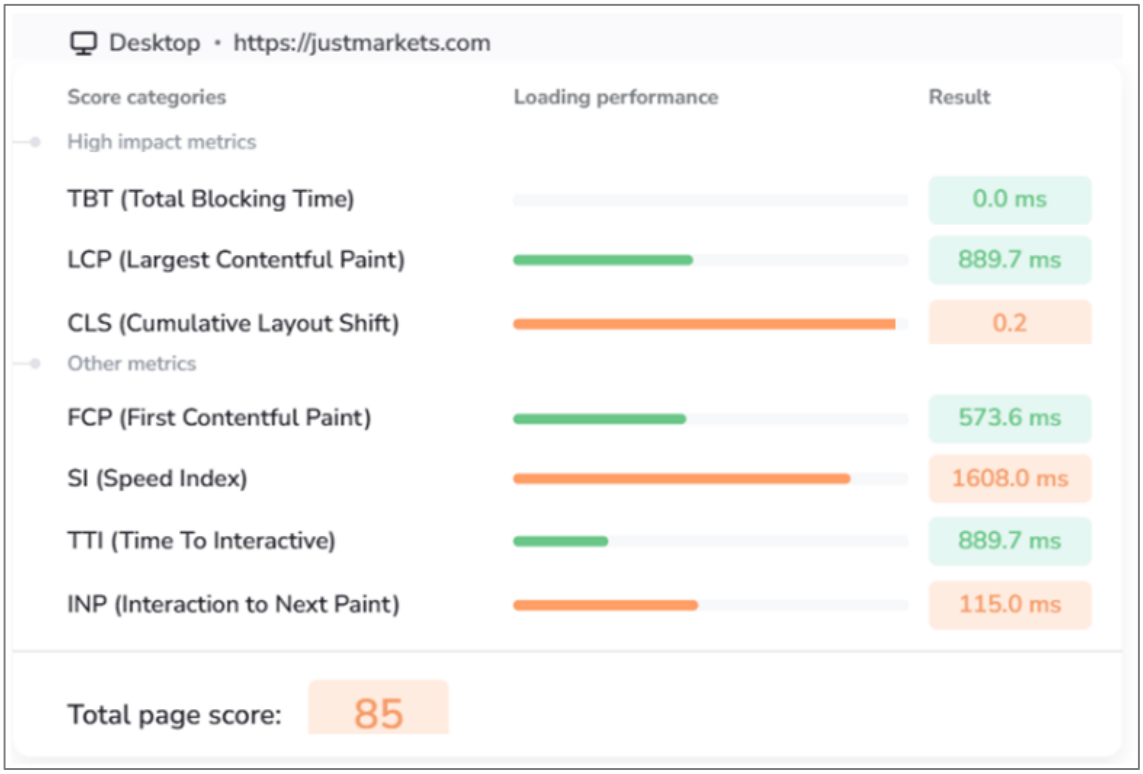


Рисунок 1.4 – Результат аудиту продуктивності від SE Ranking

На жаль, платформа SE Ranking не надає детального аналізу інших ключових метрик. Основний моніторинг SEO-показників доступний лише після придбання

підписки, що незручно для користувача, якому необхідно швидко отримати ключові результати аналізу. Про можливість інтеграції з іншими інструментами також не йдеться.

Наступним інструментом для дослідження аудитів у веб є WebPageTest. Цей інструмент подає результати глибокого аналізу завантаження сторінки з можливістю вибору декількох регіонів, типів браузера, типу мережі та пристрою (рис. 1.5). Дає змогу переглядати поетапний процес рендерингу та виявляти «вузькі місця» продуктивності, але відсутність інших, ключових метрик також не гарантує повної картини аналізу вебресурсу.

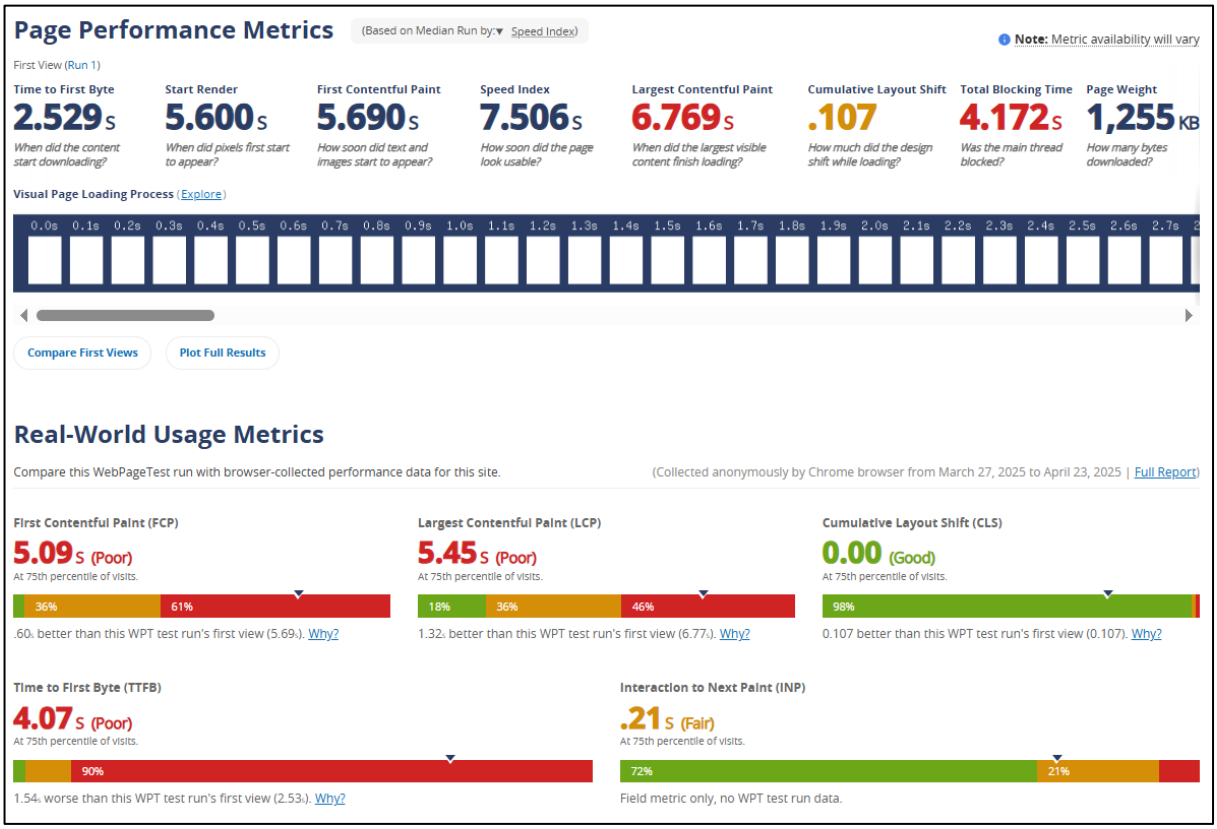


Рисунок 1.5 – Результат аудиту продуктивності від WebPageTest

PageSpeed Insights – це сервіс від Google, який оцінює продуктивність вебсторінок як на десктопах, так і на мобільних пристроях, базуючись на методології Lighthouse (API) (рис. 1.6). Він надає детальні рекомендації щодо покращення швидкості та відповідності кращим практикам веб-розробки.

Цей інструмент є фундаментом для ґрунтовного аналізу і здобув значну популярність на сучасному ринку APM в контексті аналізу вебзастосунків. Існує також CLI-застосунок, хоча можливості інтеграції та автоматизації обмежені. Сервіс зарекомендував себе як надійний та ефективний інструмент завдяки його здатності надавати конкретні дані та рекомендації для оптимізації, що безпосередньо впливають на користувацький досвід.

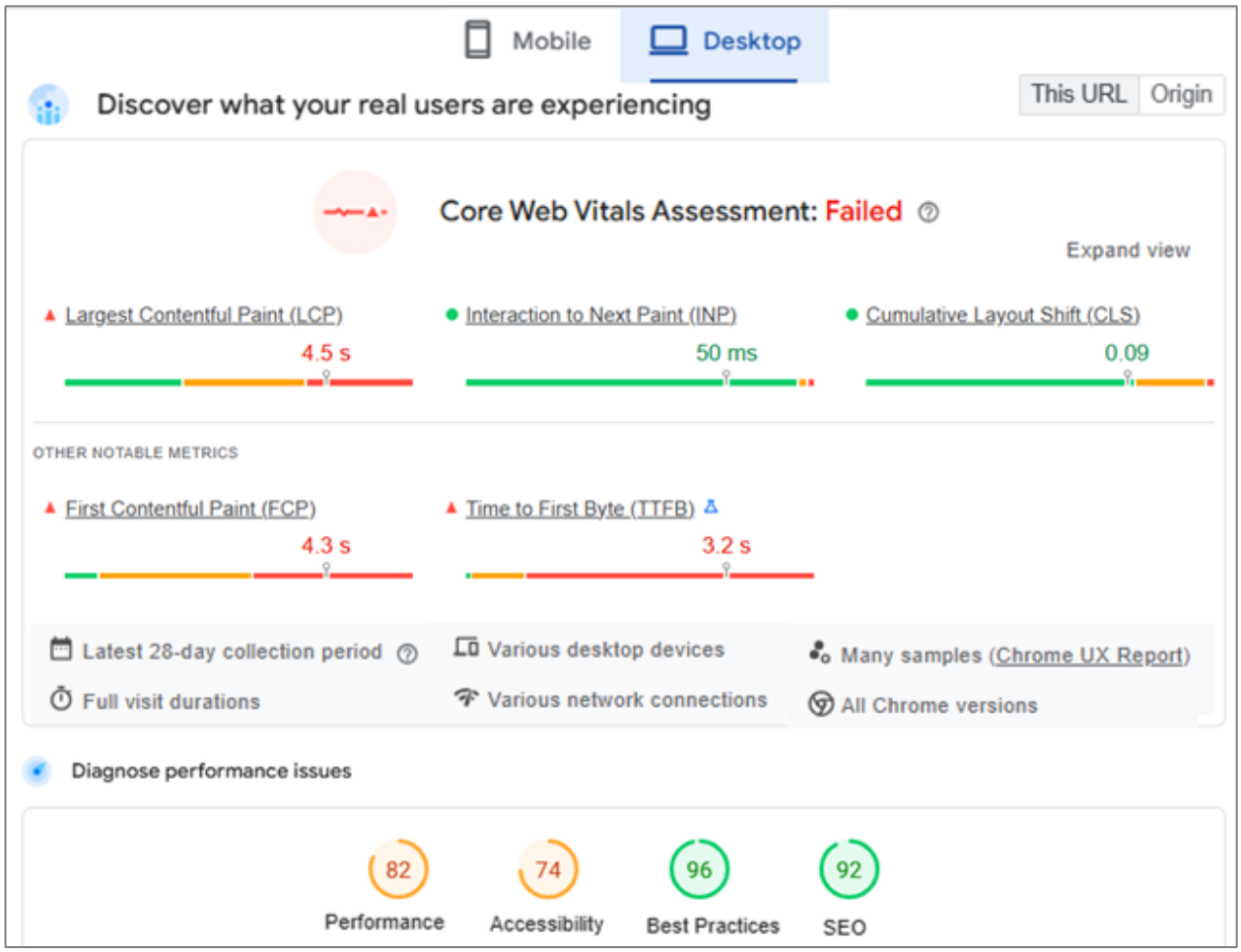


Рисунок 1.6 – Результат аудиту продуктивності від PageSpeed Insight

GTmetrix – це популярний інструмент, який забезпечує візуалізацію етапів завантаження сторінки та надає рейтинги за основними показниками продуктивності (LCP, TTI, CLS). Він зручний для відстеження динаміки змін після оптимізації (рис. 1.7). Однак, GTmetrix орієнтований виключно на аналіз

продуктивності завантаження сторінки. Інші критичні показники, які є важливими для комплексного SEO-аналізу, Best Practise та інше в ньому відсутні.

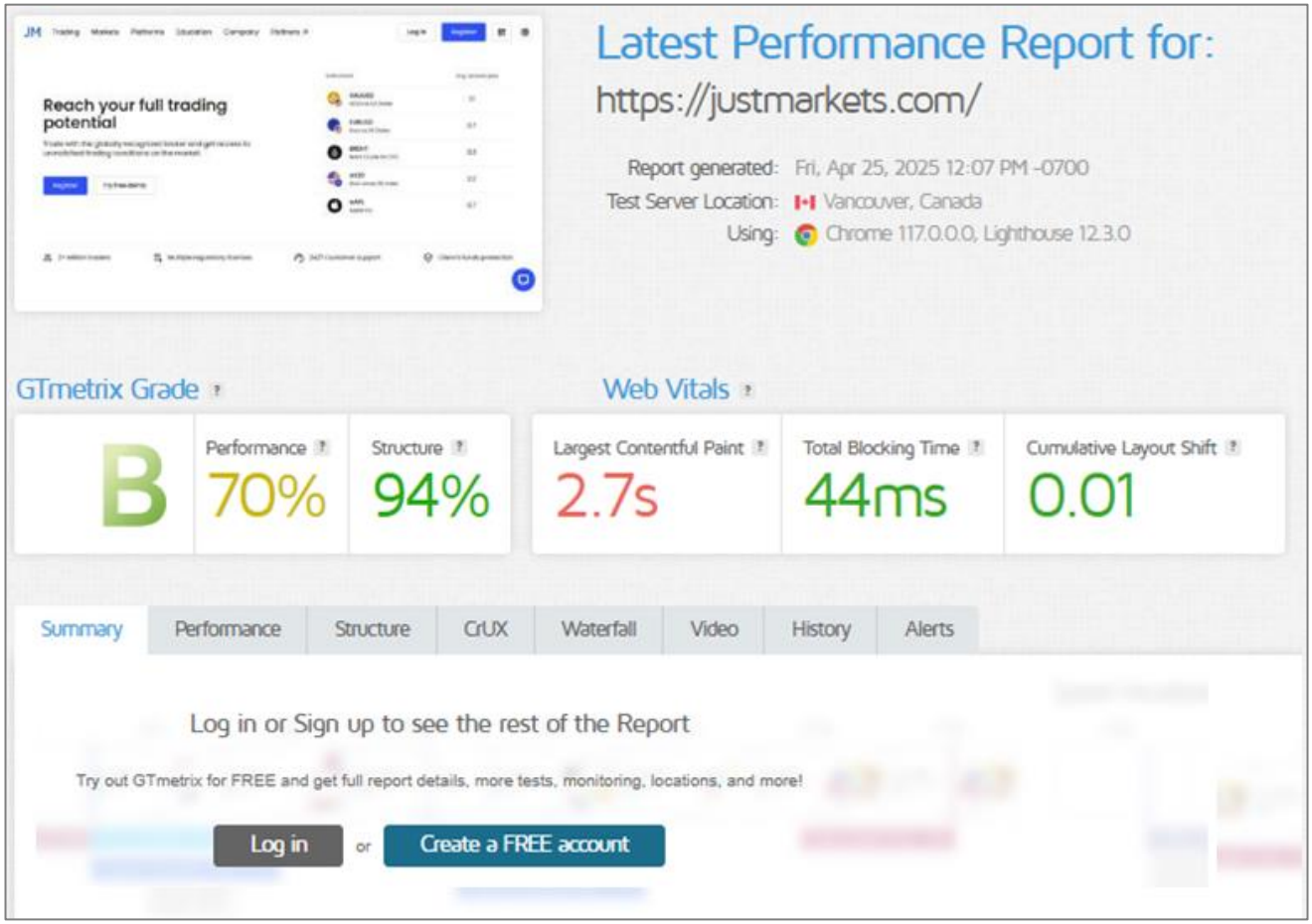
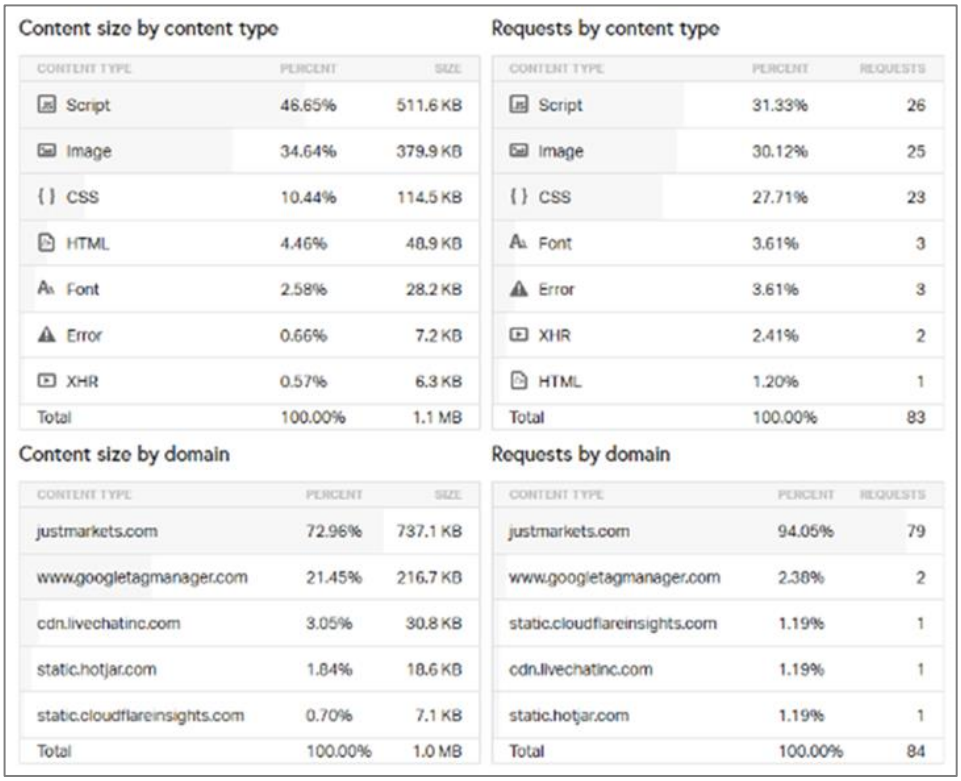


Рисунок 1.7 – Результат аудиту продуктивності від GTmetrix

Pingdom Tools – простий та швидкий сервіс для перевірки часу відгуку сервера, загального часу завантаження сторінки та розміру контенту. Він добре підходить для базового аналізу та швидкої діагностики, але його функціонал обмежений. Для глибшого аналізу та подальшої оптимізації вебзастосунків цей інструмент недостатній, оскільки, як і в багатьох інших подібних сервісах, в ньому відсутні ключові показники, необхідні для комплексного оцінювання продуктивності (рис. 1.8).



Content size by content type

CONTENT TYPE	PERCENT	SIZE
Script	46.65%	511.6 KB
Image	34.64%	379.9 KB
CSS	10.44%	114.5 KB
HTML	4.46%	48.9 KB
Font	2.58%	28.2 KB
Error	0.66%	7.2 KB
XHR	0.57%	6.3 KB
Total	100.00%	1.1 MB

Requests by content type

CONTENT TYPE	PERCENT	REQUESTS
Script	31.33%	26
Image	30.12%	25
CSS	27.71%	23
Font	3.61%	3
Error	3.61%	3
XHR	2.41%	2
HTML	1.20%	1
Total	100.00%	83

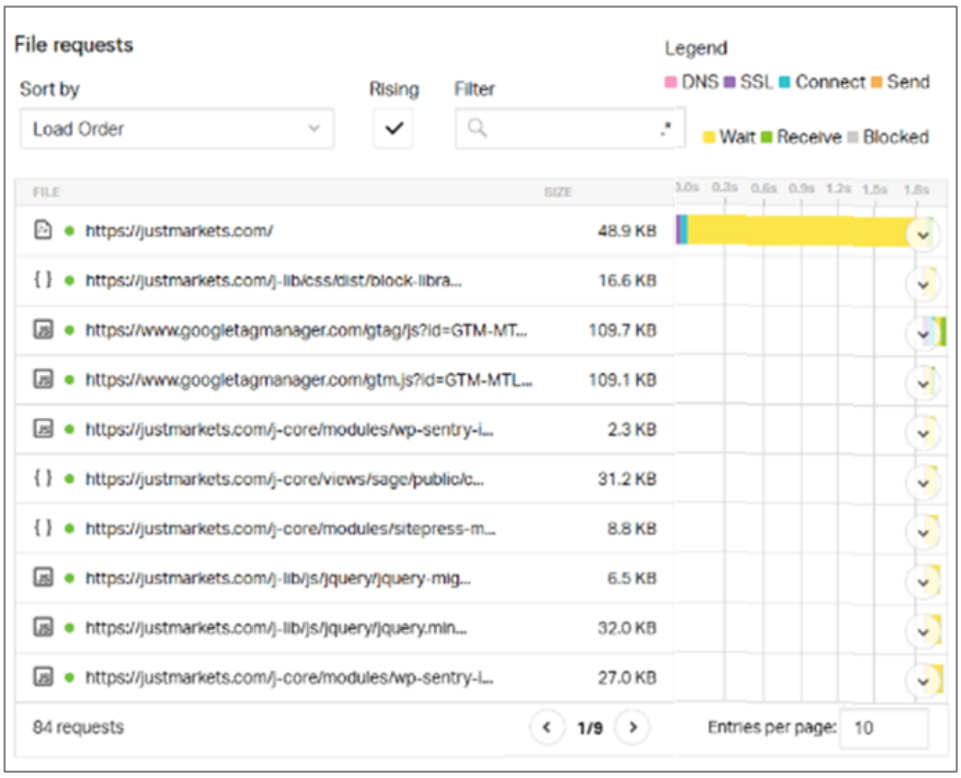
Content size by domain

CONTENT TYPE	PERCENT	SIZE
justmarkets.com	72.96%	737.1 KB
www.googletagmanager.com	21.45%	216.7 KB
cdn.livechatinc.com	3.05%	30.8 KB
static.hotjar.com	1.84%	18.6 KB
static.cloudflareinsights.com	0.70%	7.1 KB
Total	100.00%	1.0 MB

Requests by domain

CONTENT TYPE	PERCENT	REQUESTS
justmarkets.com	94.05%	79
www.googletagmanager.com	2.38%	2
static.cloudflareinsights.com	1.19%	1
cdn.livechatinc.com	1.19%	1
static.hotjar.com	1.19%	1
Total	100.00%	84

а) Content size and Requests



б) File requests

Рисунок 1.8 – Результат аудиту продуктивності від Pingdom Tools

Незважаючи на наявність широкого спектра вебінструментів для аналізу продуктивності вебресурсів, вони мають низку обмежень згаданих у параграфі 1.1, зокрема: обмежений доступ до метрик, відсутність глибоких рекомендацій, низьку гнучкість налаштувань, використання сторонніх API та обмежену кількість безкоштовних аудитів, що обмежують їх ефективне використання як для технічних фахівців, так і для нетехнічних користувачів.

Після проведення детального огляду провідних сучасних рішень для аналізу та моніторингу продуктивності вебзастосунків, існуючі інструменти, хоч і надають певну цінність, часто мають значні обмеження, які можуть ускладнити або навіть унеможливити повноцінний та глибокий аналіз. Ці обмеження стосуються як спектра охоплених метрик, так і можливостей налаштування та інтерпретації результатів.

Таким чином, більшість існуючих інструментів не враховують широкий спектр критично важливих метрик, що робить їх недостатньо ефективними для комплексного аналізу та оптимізації продуктивності вебзастосунків. Фактично, вони часто представляють лише окремі фрагменти з більш широкого спектра APM систем для аналізу, не надаючи цілісної картини. Це обмежує їх корисність для розробників та оптимізаторів, які прагнуть досягти максимальної продуктивності своїх веб-ресурсів.

Саме тому розробка власного проєкту в цій галузі стає актуальною необхідністю. Такий проєкт може усунути недоліки вже існуючих вебінструментів, об’єднуючи широкий спектр критично важливих метрик, таких як Performance, SEO, Best Practices та технічні аналіз. Це рішення пропонує не лише детальний аналіз, але й конкретні рекомендації для покращення, підтримку запуску аналізу продуктивності вебсторінки з різних географічних локацій, а також можливість швидкого та зручного запуску аналізу в одному універсальному та безкоштовному форматі. Такий підхід дозволить отримати максимально повне уявлення про стан вебзастосунку та ефективно працювати над його оптимізацією.

1.3 Постановка задачі

У сучасних умовах розвитку вебтехнологій критично важливим є забезпечення високої продуктивності вебресурсів, їхньої технічної оптимізації, відповідності сучасним стандартам SEO та найкращим практикам розробки. Від швидкості завантаження, коректності верстки, адаптивності та технічної якості ресурсу значною мірою залежить як досвід користувачів, так і успішність бізнес-процесів компаній.

Об’єкт роботи – процес аудиту вебресурсів.

Предмет роботи – програмні засоби та методи аналізу і моніторингу продуктивності web-застосунків.

Мета роботи полягає у підвищенні ефективності моніторингу продуктивності web-застосунків шляхом розробки інформаційної системи із вбудованими методами відстеження показників продуктивності та формування рекомендацій їх оптимізації.

Для розробки інформаційної системи необхідно вирішити такі завдання:

– провести аналіз, дослідити сучасні підходи та програмні рішення у сфері аудиту продуктивності вебресурсів;

– обґрунтувати вибір стеку технологій та методів аналізу й моніторингу продуктивності вебзастосунків для розробки інформаційної системи;

– здійснити проєктування, програмну реалізацію і тестування системи аналізу й моніторингу продуктивності web-застосунків та оцінити її якість.

Робота в інформаційній системі аналізу та моніторингу продуктивності базується на основі розробленого консольного CLI-застосунку для комплексного аналізу продуктивності, SEO-оптимізації та технічного стану вебресурсів, орієнтованого на швидке, автономне тестування у реальному середовищі без потреби у сторонніх онлайн-сервісах. У разі застосування при розробці IT-продуктів технології неперервної інтеграції та доставки CI/CD, CLI-застосунок

забезпечує моніторинг стану вебресурсу в реальному часі та оперативні реагування на виявлені відхилення.

CLI-застосунок повинен мати наступну функціональність:

– мати зрозумілий інтерфейс репорту, що дозволить легко інтегрувати її у процес розробки і використання сторонніми контрібьютерами;

– бути простим у використанні та швидким у виконанні тестів (аудитів), оперативно реагуючи на відхилення від оптимальних значень показників продуктивності вебзастосунків;

– надавати можливість симулювати роботу вебресурсу з десктопної та мобільної платформи (двома водночас), автоматично аналізувати зібрані дані;

– надавати рекомендації щодо покращення продуктивності вебресурсу.

Тестування та оцінку якості роботи системи необхідно провести на реальних прикладах вебресурсів та забезпечити стабільність роботи у різних середовищах і на різних апаратних платформах. Передбачено також запустити CLI-застосунок як Open-Source для загального доступу, що дозволить користувачам вдосконалювати систему та інтегрувати її в інші інструменти для аналізу та моніторингу продуктивності.

Для покращення візуального відображення результатів аудиту передбачено інтеграцію розробленого CLI-застосунку до вебзастосунку інформаційної системи. В результаті буде створено комплексний інструмент, який дозволить оперативно та ефективно проводити аналіз продуктивності вебзастосунків на відповідність ключовим показникам якості та бізнес параметрам.

Висновки до розділу 1

У першому розділі було здійснено дослідження предметної області, що стосується аналізу продуктивності вебзастосунків. Установлено, що продуктивність вебзастосунків є одним із ключових чинників, які безпосередньо впливають на досвід користувачів, показники SEO, лояльність клієнтів і, як

наслідок, – на загальні бізнес-результати компаній. Таким чином, забезпечення високої продуктивності вебсайтів є важливим аспектом для досягнення конкурентних переваг і збереження позитивного іміджу компанії.

Аналіз сучасних досліджень та існуючих рішень показав, що переважна більшість вебінструментів для аналізу продуктивності мають низку обмежень: синтетичний характер тестування, обмеження за геолокацією, тривалий час очікування, відсутність гнучких налаштувань і обмежений доступ до повного спектра метрик. Ці інструменти, хоча й широко використовуються, не завжди здатні надати повну картину щодо продуктивності вебсайтів в реальних умовах експлуатації.

З огляду на це, було обґрунтовано доцільність створення інформаційної системи на основі консольного CLI-застосунку, здатного локально виконувати детальний аналіз продуктивності вебсторінок. Такий підхід забезпечує значну швидкість, гнучкість та масштабованість. Крім того, можливість локального тестування дає змогу більш точно врахувати всі аспекти роботи вебзастосунку, без необхідності покладатися на зовнішні сервіси з обмеженими функціями. Для візуального відображення результатів CLI-застосунок доцільно інтегрувати з вебзастосунком.

Таким чином, користувачі та сторонні компанії зможуть отримати комплексний підхід до оцінки продуктивності власних веб-рішень, не витрачаючи кошти на підписки та обмеження сторонніх сервісів. Інформаційна система дозволить проводити повноцінний аудит продуктивності вебсайтів, допомагаючи виявляти вузькі місця вебсистем та надаючи повноцінний аналіз, забезпечуючи користувачів детальною інформацією про роботу їхніх застосунків. Спираючись на виявлені аспекти, програма пропонуватиме рекомендації, які дозволяють покращувати власні показники та оптимізувати роботу вебресурсі.

2 МЕТОДИ АУДИТУ WEB-ЗАСТОСУНКІВ

2.1 Методи оцінки продуктивності вебресурсів

Швидкість завантаження вебсторінок є критичним фактором для утримання користувачів та забезпечення позитивного досвіду взаємодії з ресурсом. Якщо сайт завантажується занадто довго, ймовірність того, що користувач залишить його ще до повного завантаження, зростає в рази. Навіть коротка затримка в кілька сотень мілісекунд може негативно вплинути на ставлення користувачів до вебресурсу, що, в свою чергу, може призвести до зростання показника відмов та зниження конверсії. Крім того, в умовах сучасного цифрового середовища швидкість завантаження є одним із важливих критеріїв, за якими пошукові системи, такі як Google, Bing, Yahoo та інші оцінюють та ранжують сайти в результатах видачі.

Ефективність вебресурсу визначається не лише тим, наскільки швидко він відкривається, але й тим, наскільки зручно і стабільно відбувається подальша взаємодія користувача із контентом. Важливо забезпечити не тільки швидке завантаження основного вмісту, але й миттєву відповідь на дії користувача та відсутність небажаних візуальних змін на сторінці під час використання [15]. Таким чином, комплексна оцінка продуктивності повинна враховувати кілька різних аспектів, що разом формують загальне враження від роботи з вебресурсом.

Для об’єктивного аналізу цих характеристик використовуються спеціалізовані метрики, які дають змогу оцінити як швидкість завантаження контенту, так і якість взаємодії з інтерфейсом. Серед основних метрик, що застосовуються для вимірювання продуктивності вебсторінок, вибір яких було обґрунтовано у параграфі 1.2 є: LCP, INP, CLS, FCP та TTFB. Вони дозволяють комплексно охопити різні етапи завантаження сторінки та оцінити її готовність до повноцінної взаємодії з користувачем. Розглянемо більш детально метрики вимірювання продуктивності завантаження контенту (рис. 2.1):

– *час відповіді сервера TTFB*: цей показник є критично важливим, оскільки це перший індикатор швидкості роботи ресурсу, який визначає час, необхідний для отримання першого байта відповіді від сервера;

– *перша значуща поява контенту FCP*: цей показник демонструє, коли користувач починає бачити на екрані браузера будь-який вміст (текст, зображення тощо), що є важливим для формування позитивного враження у користувача;

– *найбільша значуща поява контенту LCP*: цей показник є важливим для оцінки швидкості завантаження основного контенту, оскільки показує час відображення найбільшого за розміром елемента на сторінці браузера;

– *сумарний зсув верстки/макету CLS*: це показник стабільності, який характеризує, наскільки змінюється позиція елементів на екрані під час завантаження. Високий CLS може призводити до негативного досвіду взаємодії, оскільки елементи, які «підскакують», можуть заважати користувачеві при здійсненні навігації вебресурсом;

– *взаємодія до наступного відображення INP*: новий показник, введений у 2024 році, який відображає час відгуку на взаємодію користувача з елементами вмісту сторінки. Цей показник допомагає виміряти, наскільки швидко вебресурс відповідає на запити користувачів, що суттєво впливає на загальний досвід.

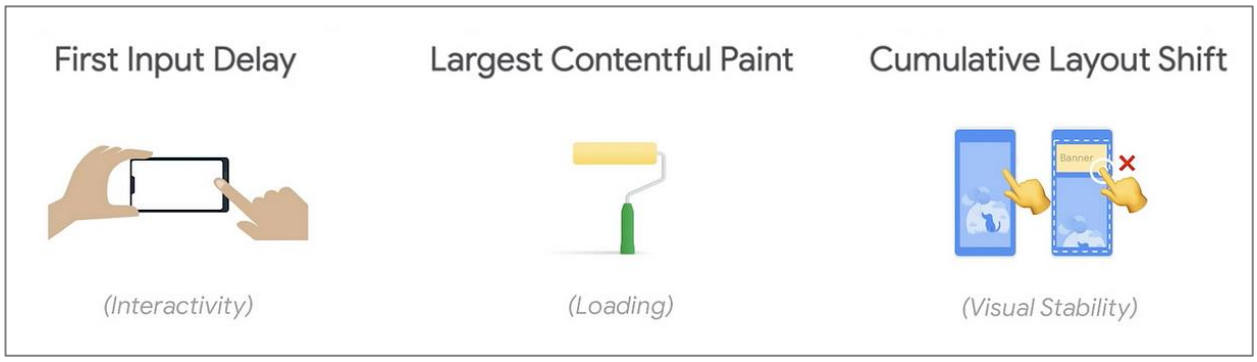


Рисунок 2.1 – Ключові показники performance вебресурсів

Визначивши ключові метрики для оцінки продуктивності вебресурсів, перейдемо до опису формального математичного апарату, що дозволить провести

комплексний аналіз цих показників. Це допоможе не лише кількісно оцінити ефективність завантаження та взаємодії з користувачами, а й виявити потенційні проблемні зони, які потребують оптимізації.

Важливим є визначення порогових значень метрик. Ці значення виконують важливу роль у оцінці продуктивності завантаження вебресурсу, завдяки визначенню «зелених», «жовтих» та «червоних» зон. Кожен із цих кольорів має своє значення [15]:

– *зелена зона* (англ. green): показники в межах оптимальних значень, де все працює добре, що свідчить про хороший користувацький досвід та те, що вебресурс функціонує на високому рівні;

– *жовта зона* (англ. yellow): показники, які виходять за межі норми, але ще не критично, що є сигналом для команди розробників звернути увагу на показники системи з метою оптимізації, адже вони можуть перетворитися на серйозні проблеми з ефективністю вебресурсу;

– *червона зона* (англ. red): показники перебувають на небезпечному рівні, який негативно впливає на працездатність вебсайту та користувацький досвід, що може призводити до високого відсотка відмов від користування ресурсом, зниження задоволеності користувачів і, як наслідок, погіршення позицій у пошукових системах.

Для кожної метрики в рамках порогових значень визначаються відповідні штрафи, які залежать від ступеня перевищення встановлених порогів. Ці штрафи, виражені у вигляді ваг, відіграють важливу роль під час аудиту продуктивності вебсайту.

Коли показники перевищують критичні значення, штрафи зменшують загальну оцінку ресурсу, відображаючи серйозність проблем. Чим більше метрика виходить за межі норми, тим вищий штраф, що негативно впливає на фінальний результат. Це дозволяє акцентувати увагу на найбільш критичних аспектах продуктивності, що потребують термінового виправлення. Таким чином, система

штрафів слугує орієнтиром для команди розробників у покращенні вебресурсу та оптимізації користувацького досвіду.

Оцінка продуктивності вебсайту базується на загальній сумі штрафів, яка нараховується відповідно до певних метрик. Початкова оцінка складає 100 балів, і з цієї базової вартості відраховуються штрафи за кожну метрику, що перевищує встановлені порогові значення. Кожна метрика має свої критичні значення. Чим більше перевищення порогу, тим значніші штрафи. Таким чином, для кожної з метрик, що потрапила в жовту або червону зону, визначається конкретна величина штрафу, яка відображає серйозність проблеми.

Загальна формула для оцінки продуктивності (англ. Performance Score) вебресурсу S_{perf} є наступною:

$$S_{perf} = \max(100 - D), \tag{2.1}$$

де D – загальний штраф усіх метрик.

Для спрощення позначимо індекси метрик:

– L для метрики LCP;
– I для метрики INP;
– C для метрики CLS;
– F для метрики FCP;
– T для метрики TTFB.

Тоді $D = \sum_i D_i$, і формулу для оцінки продуктивності вебресурсу S_{perf} можна переписати у вигляді:

$$S_{perf} = \max(100 - \sum_i D_i), \tag{2.2}$$

де i – індекс метрики, $i \in \{L; I; C; F; T\}$;

D_i – штраф i-ї метрики.

Штраф D_i кожної метрики визначається формулою:

$$D_i = \begin{cases} X_i \cdot r_i, & \text{якщо } X_i > r_i \\ X_i \cdot y_i, & \text{якщо } X_i > y_i \\ X_i \cdot g_i, & \text{якщо } X_i > g_i \\ 0, & \text{інакше} \end{cases}, \quad (2.3)$$

де i – індекс метрики, $i \in \{L;\ I;\ C;\ F;\ T\}$;

X_i – значення i-ї метрики;

g_i – порогове значення green i-ї метрики (задовільний рівень);

y_i – порогове значення yellow i-ї метрики (середній, незадовільний рівень);

r_i – порогове значення red i-ї метрики (непридатний рівень).

Цей підхід дозволяє не лише обчислити показник продуктивності ресурсу, а й візуалізувати ступінь проблем у роботі вебсайту. Наприклад, якщо одна метрика має значне перевищення порогу, то це спричиняє суттєве зниження балів, що сигналізує про термінову необхідність оптимізації.

Імплементація описаних методів оцінки загальної продуктивності вебресурсів передбачає їх інтеграцію до загальної інформаційної системи моніторингу вебресурсу. Розрахунок підсумкового показника продуктивності Performance Score дасть змогу не лише об'єктивно оцінювати поточний стан продуктивності вебсайту, а й оперативно реагувати на критичні відхилення в роботі. Таким чином, запропонований підхід в аналізу продуктивності стане важливою складовою у підтримці стабільної, швидкої та зручної взаємодії користувачів з ресурсом, що гарантуватиме більше залучення нових користувачів та ріст бізнес-показників компаній.

2.2 Визначення SEO показників

SEO показники вебресурсу є критично важливими для бізнесу з ряду причин, які варто розглянути детальніше. По-перше, у сучасному цифровому середовищі більшість користувачів насамперед звертаються до пошукових платформ, як-от

Google чи Bing, коли потрібно знайти інформацію, послуги чи продукти. Якщо вебсайт не з'являється на першій сторінці результатів пошуку, він втрачає величезну кількість потенційних клієнтів, які могли б звернутися.

Вища позиція в результатах пошуку безпосередньо впливає на обсяг трафіку, що надходить на сайт. Як правило, перші три результати пошуку отримують найбільшу увагу від користувачів, що створює можливості для отримання більшої кількості відвідувачів. Збільшення трафіку, у свою чергу, призводить до зростання конверсій – перетворення відвідувачів у покупців або клієнтів. Завдяки ефективній SEO-оптимізації бізнес може суттєво підвищити свою прибутковість, оскільки залучення нових клієнтів через органічний пошук є значно менш витратним, ніж платна реклама [17].

Крім того, добре оптимізований вебресурс допомагає підвищити впізнаваність бренду. Коли сайт з'являється на високих позиціях у результатах пошуку, споживачі більше довіряють бренду, вважаючи його авторитетним у своїй галузі. Це дозволяє бізнесу не тільки залучати нових клієнтів, а й утримувати вже існуючих, що є критично важливим елементом для успішного розвитку.

Оцінка SEO сайту базується на кількох ключових елементах, які допомагають пошуковим системам краще зрозуміти вміст сторінки і правильно її ранжувати для пошукових ботів з метою органічного просування:

– *title*: це головний заголовок сторінки, який бачать користувачі в результатах пошуку, який має бути унікальним, точним і включати основні ключові слова вебзастосунку;

– *description*: короткий опис сторінки, який теж відображається в пошуку і його добре продумана реалізація може збільшити кількість переходів на сайт;

– *H1 tag*: основний заголовок на самій сторінці, який має відображати її головну тему та допомагає пошуковим системам визначити, про що саме йде мова на вебсторінці;

– *alt image attribute*: альтернативний текст для зображень є важливим як для SEO (описує зміст зображення), так і для доступності сайту для людей із вадами зору (англ. accessibility);

– *canonical tag*: тег, що показує пошуковим системам, яка версія сторінки є основною, щоб уникнути дублювання контенту;

– *crawlable*: означає, що сторінка відкрита для індексації і доступна для сканування пошуковими ботами (якщо доступ обмежений, сторінка може не потрапити в результати пошуку).

Для розрахунку SEO-показників вебресурсу враховано вищеописані елементи, які суттєво впливають на видимість сайту в пошукових системах. При підрахунку результативності впливу SEO на продуктивність пошуку вебресурсу для різних показників було визначено ваги у загальному обсязі впливу на ефективність оптимізації. Для формального математичного опису позначимо індекси компонентів:

– T для елемента title;

– D для елемента description;

– H для елемента H1 tag;

– A для елемента alt;

– N для елемента canonical;

– R для елемента crawlable.

Кожен із компонентів має свою вагу w_i, що відображає значущість кожного з них у загальному розрахунку SEO-показників:

– для елемента title: $w_T = 15$;

– для елемента description: $w_D = 15$;

– для елемента H1 tag: $w_H = 10$;

– для елемента alt: $w_A = 10$;

– для елемента canonical: $w_N = 10$;

– для елемента crawlable: $w_R = 10$.

Крім того, важливими є також показники валідності m_i, які визначають «справжність» (відповідність) кожного компонента певним критеріям. Валідність кожного з компонентів може набувати одного з трьох значень:

– *0 – компонент невалідний*: це означає, що даний елемент не відповідає рекомендованим стандартам або налаштуванням, що негативно вплине на SEO-оптимізацію. Наприклад, заголовок чи опис відсутні, або мають суттєві помилки, що заважають пошуковим системам точно визначити зміст сторінки та валідувати контент;

– *0.5 – компонент частково валідний*: у цьому випадку елемент є напівфункціональним, тобто вебресурс має деякі позитивні аспекти SEO, але йому бракує важливих елементів або правильної структури. Наприклад, мета-опис може бути занадто коротким або недостатньо інформативним, щоб привабити відвідувачів, або заголовок може містити ключові слова, але бути надмірно загальним;

– *1 – компонент валідний*: цей показник свідчить про те, що елемент відповідає всім вимогам та стандартам SEO, що робить його ефективним інструментом для залучення користувачів і покращення видимості сайту.

Загальна оцінка SEO визначається на основі ваги кожного окремого компонента, а також його валідності. Це дозволяє отримати комплексний показник, який відображає, наскільки ефективно сайт оптимізований для пошукових систем.

Формула для розрахунку комплексного показника SEO-оцінки є наступною:

$$S_{SEO} = \left(\frac{\sum_{i=1}^{k} w_i \cdot m_i}{\sum_{i=1}^{k} w_i}\right) \cdot 100, \qquad (2.4)$$

де i – індекс компонента, $i \in \{T;\ D;\ H;\ A;\ N;\ R\}$;

w_i – значення ваги i-го компонента;

w_i – значення показника валідності i-го компонента;

k – кількість SEO компонент.

У формулі 2.4 чисельник підсумовує добутки ваги та валідності для всіх компонентів. Чим вищі значення w_i та m_i тим більшим буде внесок компонента в загальну оцінку. Знаменник підсумовує ваги всіх компонентів, забезпечуючи нормалізацію. Це дозволяє перевести отримане значення SEO-оцінку в градацію від 0 до 100. Високий бал вказує на те, що всі важливі аспекти вебресурсу оптимізовані належним чином, тоді як нижчі оцінки сигналізують про необхідність внесення змін або покращення певних елементів.

Оцінка SEO є надзвичайно важливим інструментом для визначення рівня оптимізації вебсторінки в пошуковій видачі, оскільки вона надає чітке уявлення про те, наскільки ефективно сайт відповідає вимогам пошукових систем. Високий рівень оцінки оптимізації SEO збільшує шанси на залучення цільового трафіку та підвищує конкурентоспроможність сайту [18].

2.3 Агрегована оцінка Best Practices вебресурсів

Підтримка комплексу стандартів і рекомендацій Best Practices на рівні архітектури вебзастосунку є ключовим чинником для забезпечення стабільності роботи, безперервності бізнес-процесів та високого рівня користувацького досвіду. Best Practices дозволяють своєчасно виявляти вузькі місця у функціонуванні системи, мінімізувати ризики відмов, втрати даних або зниження продуктивності, що безпосередньо впливає на досягнення бізнес-цілей. Їх систематичне впровадження і контроль на всіх рівнях – від окремих модулів до загальної архітектури створює надійний фундамент для довгострокового розвитку бізнесу та підвищення конкурентоспроможності [19].

Розглянемо ключові показники, що оцінюють різні аспекти бізнес-процесів:

– *accessibility* (доступність): оцінює, наскільки вебзастосунок забезпечує рівний доступ незалежно від фізичних чи когнітивних можливостей, що важливо для створення інклюзивного середовища;

– *deprecated APIs*: аналізує використання застарілих програмних інтерфейсів, які важливо уникати, адже їхнє використання може призвести до нестабільності системи та проблем сумісності;

– *third party cookies*: оцінює використання сторонніх файлів cookie, які впливають на конфіденційність користувачів і можуть призвести до юридичних наслідків, якщо не дотримуються політики захисту даних;

– *image aspect ratio and resolution*: перевіряє коректність співвідношення сторін і розширення зображень, що важливо для забезпечення гарного вигляду на різних пристроях та оптимізації швидкості завантаження;

– *viewport meta tags*: оцінює правильність налаштування метатегів перегляду, що забезпечують адаптивність вебсторінок на різних екранах;

– *doctype*: аналізує декларацію типу документа, що допомагає браузерам правильно відображати контент, оскільки їх відсутність або неправильний doctype може призвести до проблем із версткою і відображенням контенту;

– *charset*: оцінює правильність налаштування кодування символів, що є критично важливим для відображення тексту в різних мовах без помилок;

– *font sizes*: перевіряє використання оптимальних розмірів шрифтів, що впливає на читабельність тексту та загальний досвід користувача;

– *source maps*: аналізує наявність карт джерел, що полегшують налагодження коду, дозволяючи розробникам легко знаходити помилки;

– *geolocation*: оцінює геолокаційні функції, які можуть підвищити персоналізацію та якість обслуговування;

– *security* (безпека): оцінює загальний рівень безпеки вебзастосунку, включаючи захист даних користувачів, відповідність стандартам шифрування, а також безпеку від кібератак різного типу.

Вхідні дані Best Practices аналізу є наступними:

$$A = \{a_1, a_2, \dots, a_i, \dots, a_n\}, \tag{2.5}$$

де a_i – i-й аудиторський елемент Best Practices (вищеописані метрики);

n – загальна кількість показників, які оцінюють різні бізнес процеси.

Загальна формула оцінки Best Practices вебресурсу є наступною:

$$S_{BP} = \max\left(100 - \sum_{i=1}^{n} w_i \cdot (1 - m_i)\right), \quad (2.6)$$

де v_i – значення аудиторського елемента a_i;

де w_i – вага елемента a_i;

де m_i – коефіцієнт валідності елемента a_i.

Вага аудиту w_i відображає важливість конкретного елемента в загальному контексті оцінювання, що дозволяє правильно визначити пріоритет аудиторських метрик. Коефіцієнт валідності m_i оцінює якість і коректність кожного елемента a_i. Він може приймати одне з трьох значень:

– *1: повністю валідний*, вказує на те, що компонент є і відповідає усім вимогам;

– *0.5: частково валідний*, це означає, що елемент має певні недоліки або попередження, але загалом виконує свої функції;

– *0: невалідний*, що свідчить про те, що компонент не відповідає встановленим критеріям.

Значення для аудиту показника Best Practices формує загальну картину ефективності системи, дозволяючи приймати обґрунтовані рішення щодо оптимізації, оновлень або удосконалень.

Важливо зазначити, якщо замість булевого значення обчислення однієї з метрик a_i є числове значення для аудиту v_i, наприклад, від 0 до 100, то коефіцієнт валідності можна обчислити як відсоткове зменшення:

$$m_i = \frac{v_i}{100}. \quad (2.7)$$

Підсумковий результат – це агрегована оцінка, яка демонструє поточний стан процесів з точки зору вищезазначених метрик. Такий підхід дозволяє обґрунтовано

планувати технічні вдосконалення, оптимізувати інвестиції у розвиток та підвищувати загальну якість обслуговування користувачів, а також оцінювати, наскільки вебзастосунок є конкурентоспроможним, стійким і безпечним для користувача, гарантуючи збереження їхніх даних та надійність взаємодії з ресурсом.

2.4 Технічний аналіз web-застосунків

Проведення технічного аналізу є фундаментальним для стратегічного планування розвитку вебресурсу, оптимізації технологічної бази та підвищення ефективності бізнес-процесів. Технічний аналіз охоплює такі важливі аспекти, як:

– перевірка ефективності кешування та обробки даних, що дозволяє оптимізувати витрати ресурсів і підвищити швидкодію застосунку;

– валідація HTML DOM структури у випадку інтеграції з веб-компонентами або генерацією звітів, що забезпечує коректність візуалізації та сумісність із різними платформами;

– оптимізація роботи із зображеннями та медіа-ресурсами, що критично важливо для швидкодії та адаптивності веб-частини системи;

– перевірка адаптивності дизайну, що дозволяє гарантувати коректне відображення інтерфейсів на різноманітних пристроях та розширеннях екранів;

– аналіз логування, обробки помилок та безпеки, що допомагає мінімізувати ризики вразливостей і втрати даних.

Крім того, дослідження показують, що великі об'єкти, як-от таблиці або важкі мультимедіа-файли, негативно впливають на поступове завантаження сторінок, що може призводити до зниження задоволеності користувачів. Використання оптимізованих стратегій рендерингу – наприклад, першочергове завантаження текстового контенту та критичних стилів, а також відтерміноване завантаження важких ресурсів має бути обов'язковим елементом технічної перевірки [20].

Технічний аудит є важливою складовою загального підходу до оцінки якості вебресурсів. Він дозволяє виявити не лише явні, але й приховані проблеми, які в іншому випадку могли б призвести до стагнації системи під час масштабування чи інтеграції з іншими сервісами. Саме глибинний технічний аналіз надає можливість оцінити відповідність коду сучасним вимогам комп'ютерних наук та інженерії. Особливу увагу варто приділити тому, що проведення технічного аудиту має сенс не тільки для діючих систем, але й у рамках розробки нових проєктів, адже дозволяє:

– зменшити вартість подальшої підтримки;

– підвищити масштабованість та стійкість архітектури;

– забезпечити високу якість взаємодії кінцевих користувачів із системою;

– підготувати систему до майбутніх оновлень або інтеграцій без необхідності суттєвих переробок.

На основі результатів технічного аудиту буде формуватися набір обґрунтованих рекомендацій, спрямованих на усунення виявлених недоліків та оптимізацію роботи інформаційної системи. Під час аудиту детально аналізуються так звані struggling points – слабкі місця в архітектурі, кодовій базі, процесах обробки даних або інтеграційних механізмах. Усі знайдені проблеми фіксуються, класифікуються за рівнем критичності (наприклад, безпека, продуктивність, масштабованість) та доповнюються практичними рекомендаціями щодо їхнього усунення.

На рисунку 2.2 наведено приклад рекомендацій після технічного аудиту вебсайту від PageSpeed.

Таким чином, результати технічного аудиту слугують основою для цілеспрямованого вдосконалення системи, допомагають мінімізувати ризики в майбутньому та забезпечують стале технологічне зростання відповідно до актуальних вимог бізнесу й користувачів.

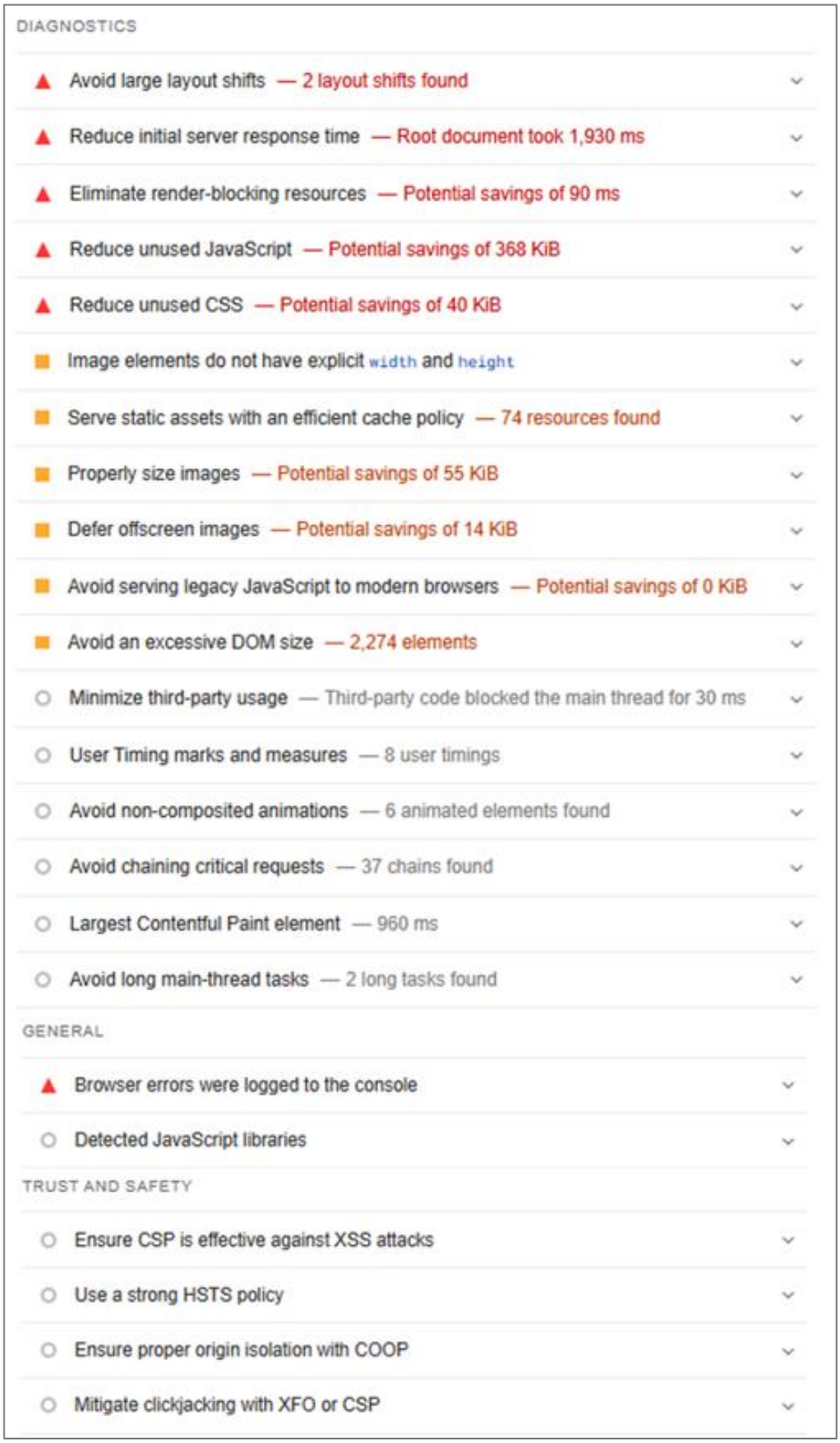


Рисунок 2.2 – Приклад рекомендацій після аудиту вебсайту від PageSpeed

Висновки до розділу 2

У другому розділі описано методи, реалізовані для аудиту вебресурсів у розробленій інформаційній системі. Ці методи слугують основою для детального та ефективного аналізу вебресурсів через інтеграцію з консольним застосунком.

Для комплексної оцінки різних етапів завантаження сторінок ресурсу було використано показники: LCP, INP, CLS, FCP та TTFB. Кожен із цих показників може мати значення з зеленої, жовтої та червоної зон, для яких визначено порогові значення та штрафи у разі перевищення порогових значень. Загальна оцінка продуктивності базується на розрахунку підсумкового показника продуктивності Performance Score.

Для оцінки SEO обґрунтовано відбір елементів, які допомагають пошуковим системам краще зрозуміти вміст сторінки і правильно її ранжувати для пошукових ботів з метою органічного просування. Загальна оцінка SEO визначається як комплексний показник, на основі ваги кожного окремого компонента, а також його валідності. У інформаційній системі моніторингу продуктивності вебресурсів передбачено також розрахунок агрегованої оцінки Best Practices вебресурсів, як підтримки комплексу стандартів і рекомендацій на рівні архітектури вебзастосунку, які оцінюють різні аспекти бізнес-процесів. У системі буде реалізовано також технічний аудит, який є не менш важливою складовою в оцінці якості вебресурсу, оптимізації технологічної бази та підвищення ефективності бізнес-процесів, оцінюючи архітектуру, безпеку та доступність системи.

Загальна оцінка продуктивності формується за допомогою компонентів, що дозволяє кількісно оцінити ефективність кожного з досліджуваних аспектів. Такий комплексний підхід забезпечує не лише об'єктивну оцінку поточного стан продуктивності вебресурсу, а й оперативне реагування на критичні відхилення в його роботі. Система надає рекомендації для покращення, зосереджуючи увагу на вдосконаленні тих елементів, що впливають на користувацький досвід. Особливістю є система штрафів, реалізована для оцінки перевищень порогових

значень метрик. Це дозволяє чітко демонструвати ступінь серйозності виявлених проблем, що надає можливість технічним фахівцям зосередити свої зусилля на найбільш критичних аспектах, які потребують термінового виправлення.

Таким комплексна оцінка вебресурсів на основі описаних методів відіграє роль не лише інструмента для глибокого аудиту, але й ефективного механізму аналізу продуктивності. Отримані результати забезпечують високий рівень якості обслуговування користувачів, відповідають потребам бізнесу та підвищують технологічну конкурентоспроможність. Крім того, система підсвічує проблеми, що виникають, з метою покращення вебресурсу. Результати аудиту, зібрані в консольному застосунку, стають важливою складовою в процесі стратегічного розвитку інформаційних систем, гарантуючи їх оптимізацію на всіх етапах життєвого циклу.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Аналіз сценаріїв використання та діаграма діяльності

На етапі моделювання інформаційної системи аналізу продуктивності вебзастосунків було розроблено діаграму сценаріїв використання для пояснення роботи системи та формування функціональних вимог до системи (рис. 3.1).

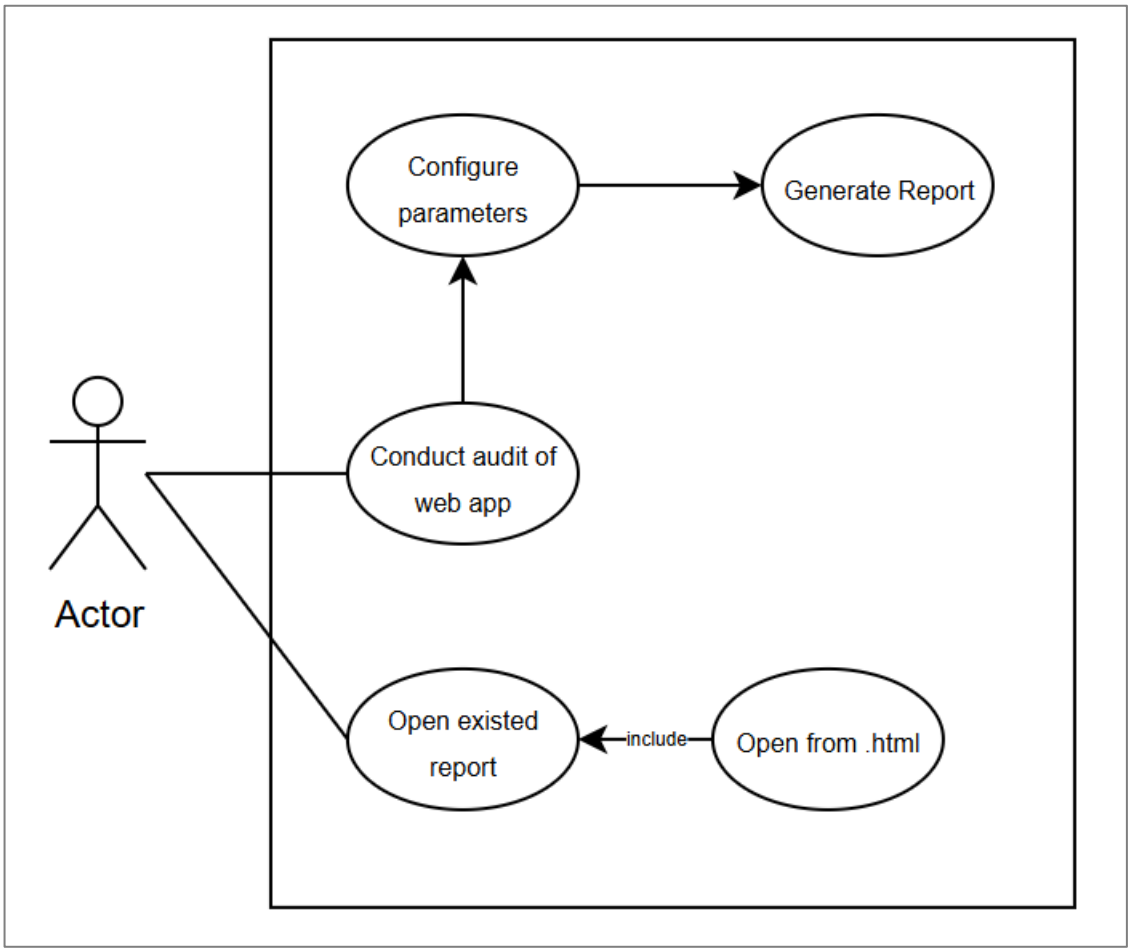


Рисунок 3.1 – Діаграма прецедентів

Взаємодія користувача з системою передбачає два сценарії використання:

– Use case 1: проведення аудиту вебзастосунку та генерації звіту;
– Use case 2: перегляд існуючого звіту.

Розглянемо кожний сценарій моделювання роботи інструменту окремо, для розуміння коректної поведінки системи.

У першому прецеденті користувач ініціює запуск інформаційної системи для проведення аудиту обраного вебзастосунку. Головна мета цього процесу – отримати детальний звіт про продуктивність вебресурсу з урахуванням основних метрик та показників, що впливають на швидкодію, стабільність і відповідність сучасним стандартам. Для коректної роботи системи користувач задаватиме різноманітні параметри, які визначають формат результату та специфіку проведення аудиту.

Основні параметри конфігурації інформаційної системи з метою проведення комплексного аудиту продуктивності web-застосунків будуть наступними (рис. 4.9):

– h (--help): це ключ, який відповідає за отримання інформації про доступні системні команди та параметри запуску. Він надає короткий опис кожного еліаса, дозволяючи користувачу швидко зорієнтуватися у можливостях інформаційної системи. Даний параметр додається автоматично для забезпечення коректної взаємодії з CLI-застосунками через бібліотеку yargs;

– u (--url): основний параметр, що використовується для вказівки цільового вебзастосунку. Після цього еліаса необхідно вказати посилання на сайт у повному форматі (включаючи протокол, DNS та домен), що дозволяє системі точно визначити ціль аудиту. Тип опції – string, тому передача інших значень неможлива;

– j (--json) – параметр, який визначає формат кінцевого звіту. Якщо під час запуску системи передати цей ключ, звіт буде згенеровано у форматі JSON. Такий підхід особливо корисний при інтеграції системи у процеси CI/CD, де автоматизований звіт на пайплайні дозволяє зручно відстежувати результати аудиту. Тип параметру – boolean; якщо ключ не задано, значення за замовчуванням – false, тобто звіт буде представлено у консольному форматі;

– r (--html-report) – ще один параметр, що впливає на формат звіту. При його використанні кінцевий результат аналізу буде представлено у вигляді інтерактивного HTML-документа. Це дозволяє детально переглядати метрики та

рекомендації в зручному візуальному форматі. Тип параметру – boolean; якщо ключ не задано, звіт у форматі HTML не генерується;

– ip – цей параметр надає можливість дізнатись поточний IP-адрес користувача та країну запуску системи. Він корисний у випадках, коли аудит проводиться з використанням проксі-серверів або з різних локацій. Інформація про IP дозволяє зафіксувати точку запуску аудиту, що особливо важливо у процесах автоматизації тестування та аналізу географії неоптимізованих метрик вебсайтів;

– x (--x-forwarded-for) – параметр, який дозволяє використовувати заголовок X-Forwarded-For для зміни IP-адреси при відправці запиту. Тип параметру – string, оскільки потрібно буде вказувати адресу проксі-сервера. Цей параметр забезпечує анонімність та можливість тестування ресурсу з різних геолокацій.

Таким чином, користувач зможе комбінувати та поєднувати різні параметри залежно від цілей аудиту. Гнучкість налаштувань дозволятиме адаптувати процес аналізу під конкретні сценарії використання, що робить систему універсальною та зручною для різних завдань. Якщо жоден параметр не передано, система буде працювати у стандартному режимі без додаткових налаштувань, кінцевим результатом котрої буде вивід репорту аудита в консольному форматі.

Другий прецедент доцільно використовувати після того, як система завершила аналіз і згенерувала звіт. Основна мета цього сценарію – отримання та перегляд результатів аудиту у зручному форматі. Зазвичай цей прецедент використовуватиметься після запуску системи з параметром, що відповідає за створення HTML-звіту. Інтерактивний звіт у форматі HTML є чудовим доповненням до глибинного аналізу вебсайту. Він дозволяє зручно переглядати основні показники продуктивності, оцінки SEO та відповідності Best Practices у вигляді наочного дашборду. Це значно полегшує розуміння результатів аудиту, надаючи користувачеві структуровану інформацію з рекомендаціями для оптимізації ресурсу.

Однак у випадках, коли необхідно зберігати результати аудиту на тривалий час або інтегрувати їх у процеси CI/CD, доцільно скористатися параметром, що

генерує JSON-відповідь. Такий формат буде не лише зручний для автоматизованої обробки, але й забезпечувати надійне збереження результатів у вигляді артефакту в пайплайні. Це дозволить порівнювати показники після кожного оновлення системи та оперативно реагувати на зміни у продуктивності вебзастосунку.

На етапі проектування розроблено діаграму станів та переходів інформаційної системи для аналізу продуктивності вебзастосунків (рис 3.2).

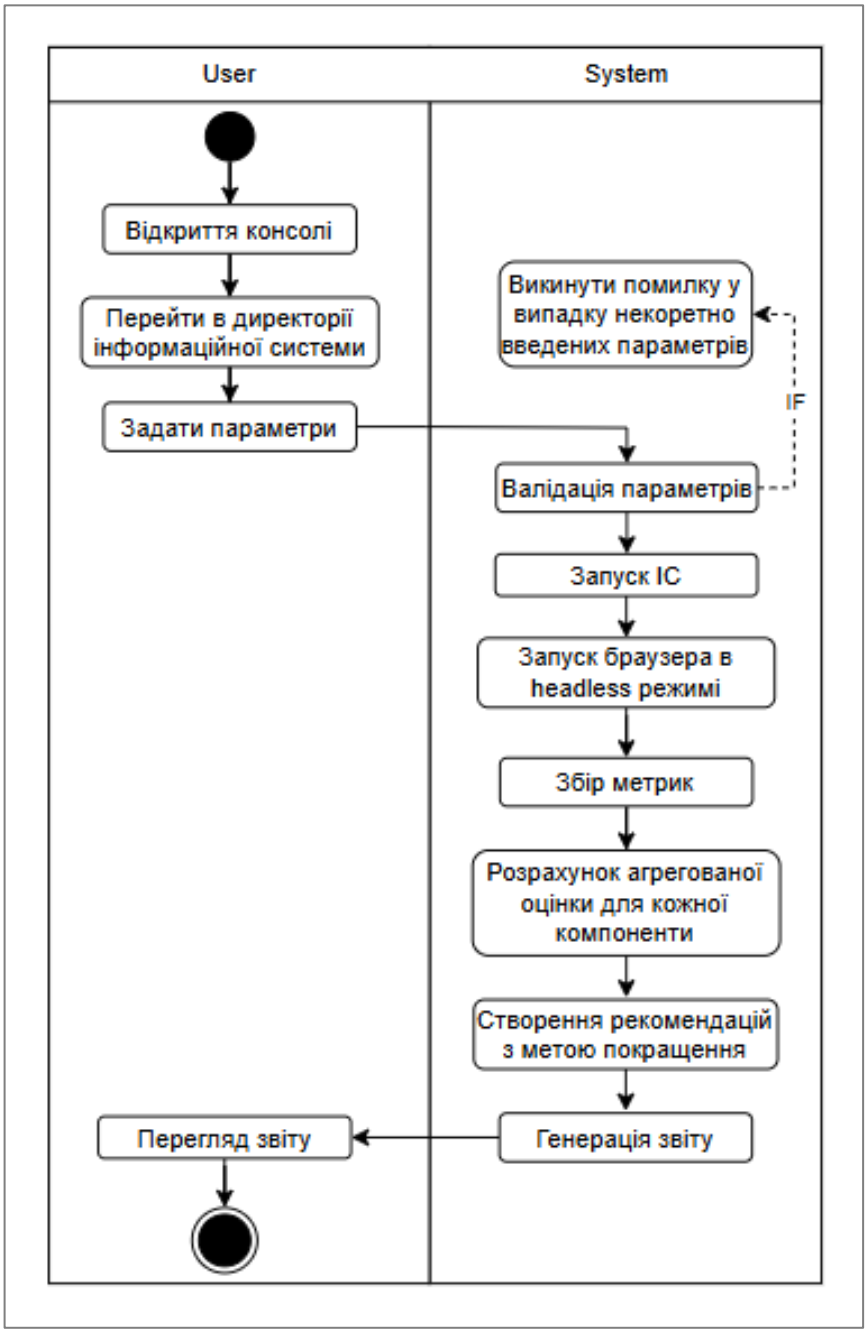


Рисунок 3.2 – Діаграма станів та переходів системи

3.2 Інструментальні засоби розробки системи

Здійснимо опис стеку технологій, які було обрано для розробки інформаційної системи аналізу та моніторингу вебзастосунків.

JavaScript – це високорівнева мова програмування, яка є стандартом для розробки інтерактивних вебзастосунків. Завдяки своїй універсальності JavaScript використовується як на стороні клієнта, так і на стороні сервера, що дозволяє створювати повноцінні вебрішення із застосуванням єдиної мовної бази. Мова підтримує об'єктно-орієнтований та функціональний підхід, що робить її гнучкою та зручною для побудови систем будь-якої складності. Активна екосистема JavaScript постійно розширюється завдяки великій кількості бібліотек і фреймворків, які спрощують розробку й оптимізують робочі процеси.

Основною причиною вибору JavaScript для реалізації інформаційної системи аналізу продуктивності вебзастосунків є його масштабованість, автономність та стабільність. Використання JavaScript дозволяє ефективно працювати із вебресурсами, обробляти дані в реальному часі, а також забезпечує кросплатформність – система може бути розгорнута на будь-якій операційній системі без значних модифікацій.

Для реалізації серверної частини було обрано Node.js – середовище виконання JavaScript на сервері, яке побудоване на базі рушія V8 від Google. Node.js дозволяє використовувати JavaScript поза браузером, що відкриває можливість створення швидких, масштабованих і асинхронних серверних застосунків. Архітектура Node.js базується на неблокуючій моделі введення-виведення, що забезпечує високу продуктивність при роботі з мережевими запитами та великою кількістю одночасних підключень [21].

У рамках даного проєкту Node.js виконуватиме ключові завдання на серверній стороні: формування та відправлення HTTP-запитів до вебресурсів, отримання відповідей від серверів, а також подальшу їхню обробку та валідацію. Крім того, Node.js забезпечує взаємодію між клієнтською та серверною частинами,

обробку даних із зовнішніх API та зберігання результатів аудиту, що дозволяє створити цілісну інформаційну систему для аналізу продуктивності вебзастосунків та ефективності їх функціонування.

Одним із ключових інструментів у побудові інформаційної системи аналізу продуктивності вебзастосунків є Puppeteer – високорівнева бібліотека для керування браузером через Node.js. Puppeteer працює поверх движку Chromium і забезпечує програмний доступ до функціональності браузера, надаючи розробникам можливість взаємодіяти із вебсторінками так, ніби це робить реальний користувач. Завдяки побудові на основі асинхронного програмування з використанням промісів (англ. Promises) [22], бібліотека дозволяє ефективно виконувати основні поставлені задачі.

Інтеграція Puppeteer у Node.js-проєкти є доволі простою завдяки сумісності екосистеми: бібліотека встановлюється як npm-пакет і легко підключається через імпорт у скрипти. Puppeteer створює інстанси браузера і сторінок, якими можна керувати за допомогою простого й зрозумілого API. У рамках розроблюваної системи аналізу продуктивності ця бібліотека виступає основним інструментом для автоматизації доступу до вебресурсів, збору даних про час завантаження, структуру DOM, аналіз SEO-метрик і відповідність Best Practices. Це дозволяє отримувати комплексну аналітику сторінок без необхідності ручного втручання.

Для створення ядра Node.js застосунку, було використано бібліотеку Yargs, оскільки вона дозволяє легко організовувати обробку аргументів командного рядка. Yargs забезпечує зручний механізм налаштування команд і параметрів, що спрощує взаємодію користувача з програмою.

Крім того, для здійснення HTTP-запитів у JavaScript буде застосовано Axios – популярну бібліотеку, яка працює як на клієнтській, так і на серверній стороні. Axios надає зручний інтерфейс для роботи з асинхронними запитами, підтримує обробку помилок, автоматично перетворює JSON-відповіді у JavaScript об'єкти і працює з промісами. Завдяки своїй простоті та зручності використання, Axios

ідеально підходить для роботи з API і забезпечує легкість у відправленні запитів до серверів.

Щоб користувач міг бачити, що система виконує запит і не «зависла», було використано бібліотеку cli-spinner. Це візуальне відображення прогресу дозволить користувачу зрозуміти, що програма працює, навіть якщо вона здійснює запити до сервера чи виконує важливі операції. Спіннер з'являтиметься під час виконання запитів, поки система очікує відповіді від серверів або виконує аналітику через Puppeteer, забезпечуючи зручний інтерфейс для користувача.

Для розробки консольного застосунку було обрано Visual Studio Code – сучасний та легкий редактор коду, розроблений Microsoft. Завдяки своїй гнучкості, універсальності та розширюваності VS Code став популярним серед розробників, особливо тих, хто працює з JavaScript. VS Code надає основні функції для роботи з кодом, такі як автодоповнення, інтелектуальні підказки (англ. IntelliSense), дебаггінг та інтеграцію з Git, що дозволяє зручно управляти версіями проекту.

Однією з ключових переваг є інтеграція штучного інтелекту Copilot, що допомагає швидше генерувати код та надає рекомендації на основі аналізу проєкту. Активна спільнота розробників постійно вдосконалює IDE, створюючи тисячі розширень, що розширюють його функціональні можливості. Цей баланс між легкістю використання та глибиною функціоналу робить VS Code відмінним вибором як для початківців, так і для досвідчених програмістів, що працюють у сфері фронтенд і бекенд-розробки.

Загальна схема стеку технологій, які були задіяні у розробці інформаційної системи аналізу продуктивності вебзастосунків, відображена на рисунку 3.3. Завдяки комбінації JavaScript, Node.js та Puppeeter забезпечується повноцінний цикл роботи системи – від збору необхідної інформації до її обробки й формування аналітичних результатів, що відповідає сучасним вимогам до продуктивності, масштабованості та надійності.

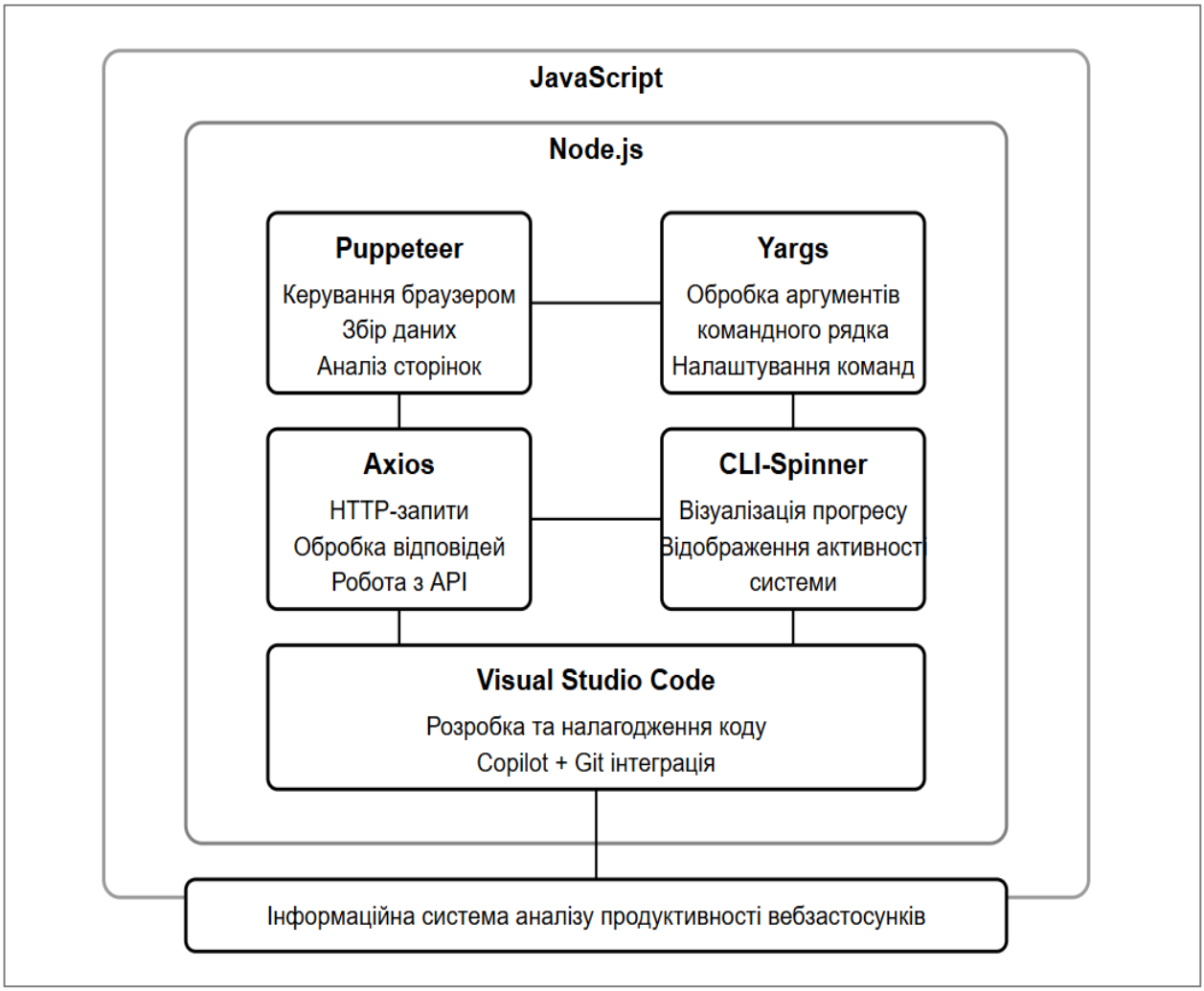


Рисунок 3.3 – Стек технологій розробки інформаційної системи

3.3 Планування асинхронної роботи CLI-застосунку

Асинхронність є важливою концепцією в програмуванні, що дозволяє виконувати кілька операцій одночасно, не блокуючи основний потік виконання програми. У контексті розробки вебзастосунків, асинхронність є критично важливою, оскільки сучасні вебсайти повинні ефективно взаємодіяти з користувачами та зовнішніми сервісами, такими як API. Особливо це актуально при виконанні аналізу продуктивності вебзастосунків, коли система постійно надсилає запити для збору і оцінки даних.

Інформаційна система, орієнтована на аналіз продуктивності, працює за принципом клієнт-серверної архітектури, в якій клієнт надсилає запити до сервера для отримання різних метрик і результатів. Коли користувач ініціюватиме дію, він побачить індикатор завантаження, що інформує про обробку запиту. В той же час, застосунок активно надсилатиме значну кількість запитів на сервер для проведення аудиту продуктивності вебресурсу.

Синхронність у такому випадку не дала б бажаних результатів через те, що кожен запит оброблявся б у порядку черги, очікуючи на завершення попереднього. Це призвело б до значного збільшення часу виконання операцій, досягнувши до 3 хвилин або більше, що є неприйнятним для проведення аудиту. Такий підхід негативно позначився б на користувацькому досвіді.

Крім того, користувачі могли б вважати, що система «заморожена», під час тривалого процесу, що призвело б до частішого закриття вебсайту або переходу до інших рішень. Використання асинхронності дозволяє уникнути ці проблеми, сприяючи ефективному управлінню запитами і забезпечуючи плавний, безперервний інтерфейс для взаємодії з користувачем (рис. 3.4).

У рамках розробленої інформаційної системи, бібліотека Puppeteer використовується для автоматизованого збору показників продуктивності вебзастосунків. Цей інструмент дозволяє виконувати операції над веб-сторінками, такі як навігація, введення даних, створення скріншотів, зчитування DOM-елементів і вимірювання метрик, без блокування потоку обробки запитів. Кожен запуск сценарію Puppeteer виконується у відокремленому асинхронному потоці, що суттєво підвищує продуктивність системи.

Завдяки асинхронності, з якою працює Puppeteer, можливо виконання кількох завдань одночасно без блокування основного потоку виконання програми. Це організовує паралельну обробку запитів, що дозволяє системі надсилати численні запити на сервер, здійснювати навігацію по сторінках та виконувати інші дії, не очікуючи завершення кожного з них перед початком наступного [23].

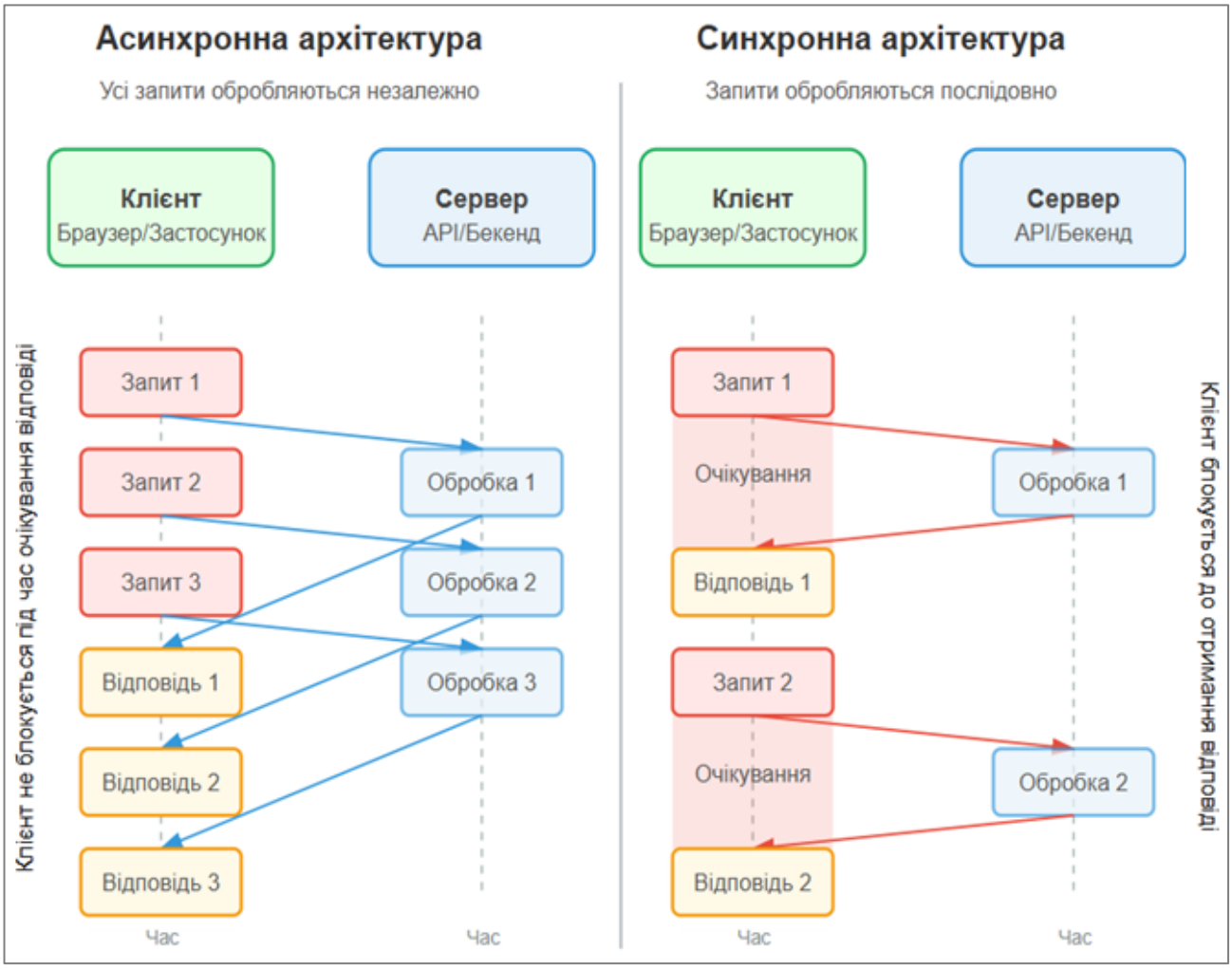


Рисунок 3.4 – Порівняння роботи асинхронної та синхронної архітектури

Наприклад, коли Puppeteer завантажує сторінку, у цей момент він не зупиняє всю програму на очікуванні відповіді – замість цього, він переходить до виконання інших запитів та завдань, таких як взаємодія з уже завантаженими елементами або аналіз інших сторінок. Це означає, що під час завантаження однієї сторінки система може продовжувати виконувати інші дії, що оптимізує використання часу.

Маючи теоретичну базу щодо того, як працює асинхронність, як саме вона реалізована в Puppeteer, а також опираючись на порівняльну статистику (рис. 3.5), було обрано проєктування запиту в асинхронній архітектурі, який відправлятиметься з клієнта на сервер.

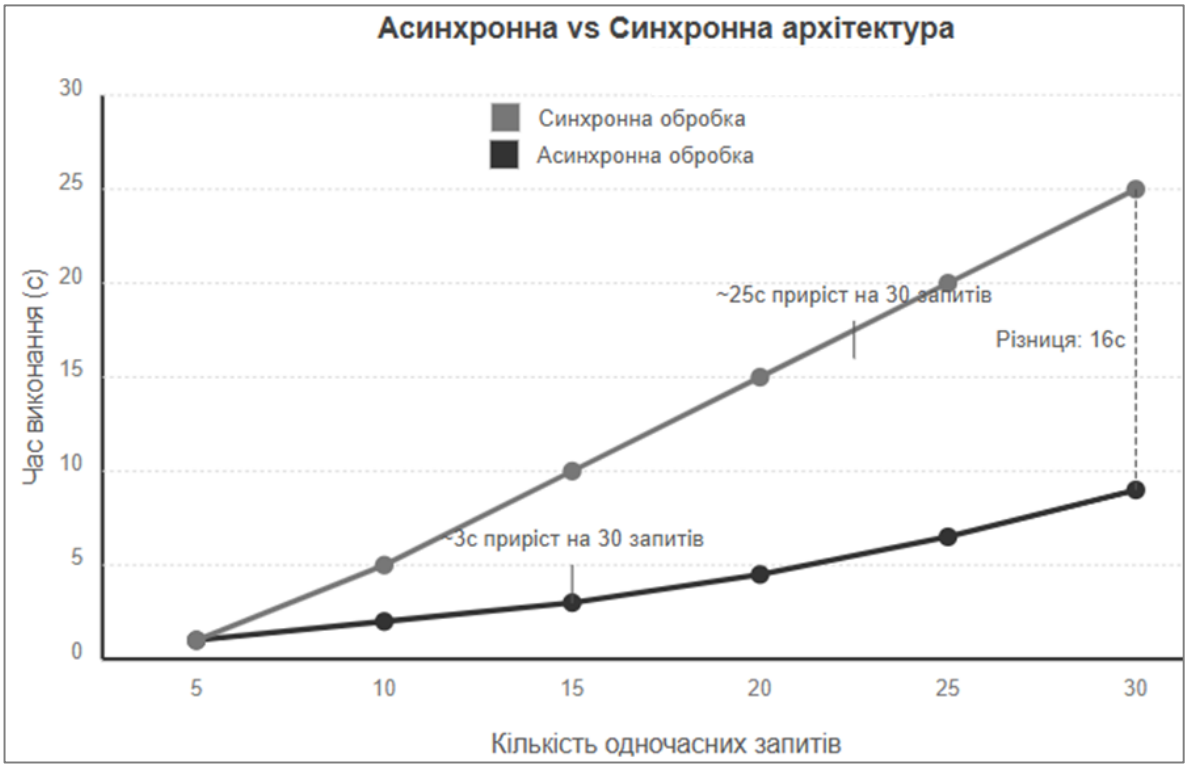


Рисунок 3.5 – Порівняння часу обробки 30 запитів до https://justmarkets.com при використанні асинхронної та синхронної архітектури

Після відправки запиту в консольному застосунку процес обробки запиту розпочнеться з клієнта, який надішле HTTP (REST) запит на сервер. Сервер, отримавши запит, ініціює завдання для Puppeteer в асинхронному режимі. Puppeteer, у свою чергу, запустить браузер (що буде асинхронною операцією, яка повертає Promise) і після готовності браузера створить нову сторінку, також асинхронно очікуючи її створення. Потім Puppeteer здійснить навігацію до цільового сайту (що також є Promise), і браузер відправить HTTP запит до цілі, отримуючи у відповідь HTML/CSS (де-інде й JavaScript код) (рис. 3.6).

Після завантаження сторінки Puppeteer виконає JavaScript на сторінці (асинхронно). Результат виконання JavaScript може призвести до взаємодії з елементами сторінки, наприклад, кліків, що можуть ініціювати AJAX/XHR запити від браузера до цілі та отримання асинхронної відповіді. Puppeteer слідкуватиме за цими процесами, збираючи необхідні дані (асинхронно) та сигналізуючи про готовність даних. Після завершення обробки сервер отримає результат від

Puppeteer (англ. Promise resolve) і, нарешті, надішле клієнту відповідь із результатами аудиту конкретних метрик.

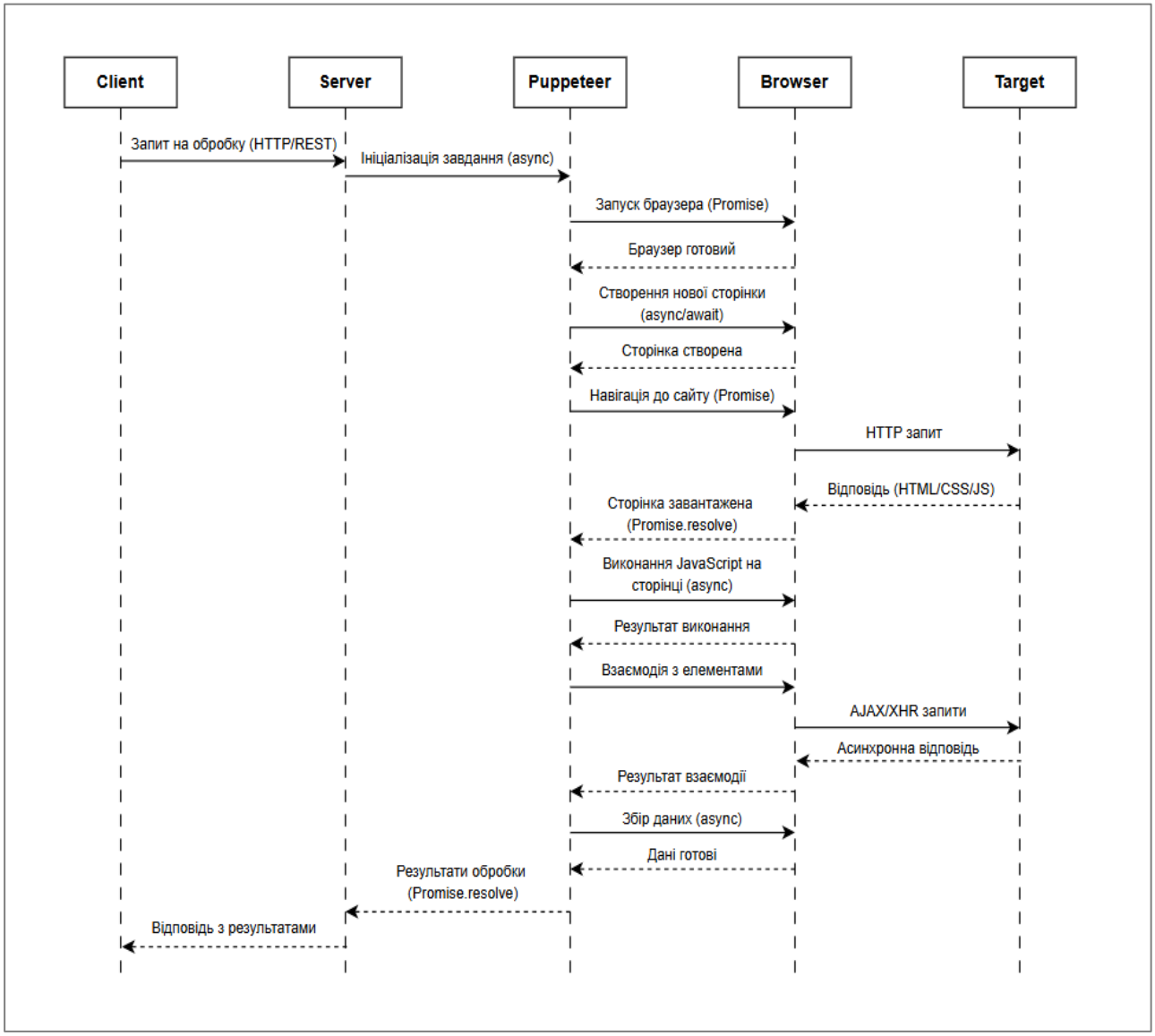


Рисунок 3.6 – Діаграма станів та переходів роботи запиту в клієнт-серверній архітектурі з використанням Puppeteer на стороні Back-End

3.4 Інтеграція модулів та формування звіту

У межах розробки CLI-застосунку Node.js виступає ядром, яке ініціалізує запуск Puppeteer, обробляє вхідні параметри користувача, взаємодіє з допоміжними бібліотеками, зчитує та записує дані, а також формує фінальний звіт у зручному

для користувача вигляді (рис. 3.7). За допомогою вбудованих модулів Node.js, як-от fs (для роботи з файловою системою), os (для визначення системного середовища) чи child_process (для запуску сторонніх процесів), програма отримує доступ до широкого спектра можливостей без потреби в надмірних залежностях.

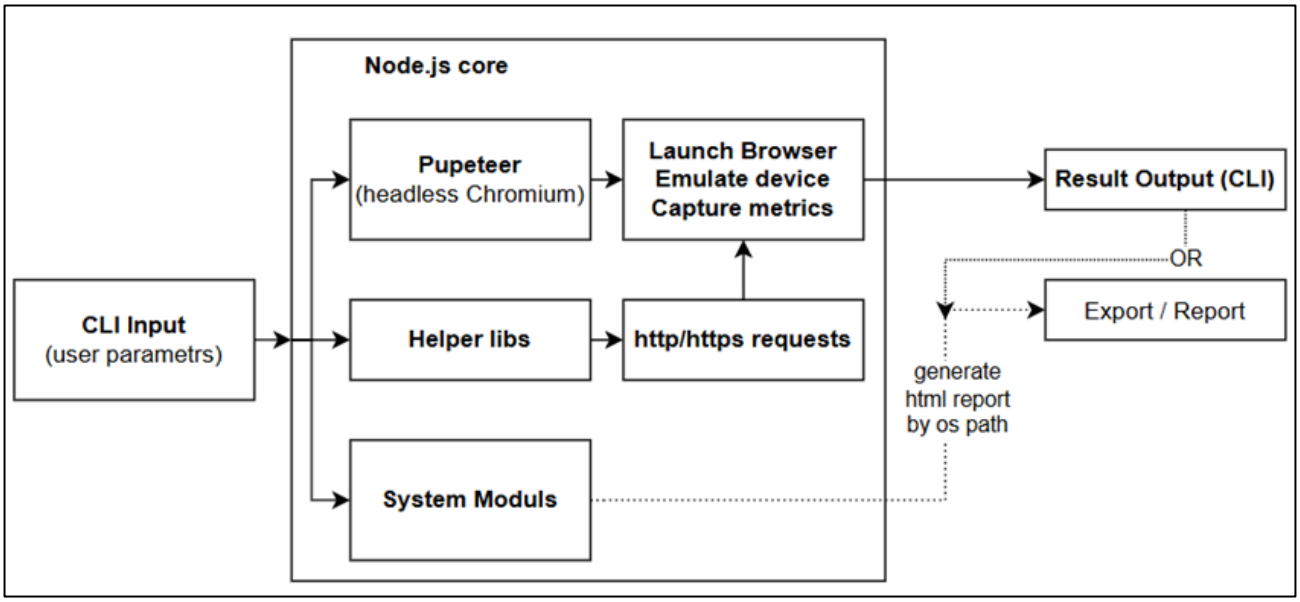


Рисунок 3.7 – Архітектура інформаційної системи

У процесі виконання асинхронних запитів частина відповідей від сервера вже буде надходити до клієнта. На цьому етапі в роботу вступлять обчислювальні модулі інформаційної системи у вигляді компонентів, які використовуватимуть дані від Puppeteer для їх аналізу, обробки та підрахунку в асинхронному порядку. Отримані результати будуть гармонічно представлені в консольному застосунку. Цей процес забезпечить інтеграцію даних та їх ефективний аналіз, що дозволить отримати оперативну та корисну інформацію в реальному часі.

Враховуючи інформацію, викладену у параграфі 3.2, було створено окрему компоненту, яка відповідає за налаштування параметрів браузера та його запуск у headless-режимі. Ця компонента реалізуватиме конфігурацію браузерного середовища Puppeteer і визначить, на основі яких критеріїв і куди саме буде здійснюватися асинхронний запит для аналізу цільового вебресурсу.

Для забезпечення зручності масштабування та повторного використання коду в реалізації інформаційної системи використовується модульна розробка – кожен функціональний блок аудиту має власний модуль, відповідальний за оцінку відповідного аспекту при аналізі.

Усі ці елементи інтегровані в єдину інформаційну систему, що забезпечує злагоджену роботу всіх модулів. Отримані дані від усіх компонентів перенаправляються до аналітичного модуля аудиту, який відіграє ключову роль у процесі глибокого аналізу результатів. Це дозволяє виконувати всебічний моніторинг вебзастосунку, виявляти потенційні проблеми, оцінювати ефективність та надавати корисні рекомендації для подальшого поліпшення його продуктивності. Завдяки такому підходу, інформаційна система є потужним інструментом, який дозволяє постійно оцінювати та вдосконалювати функціональність вебзастосунків.

Оскільки всі модулі виконують незалежні обчислення та не залежать один від одного, асинхронність дозволяє одночасно запускати всі розрахунки та значно зменшити загальний час формування аудиту. Обчислення запускаються паралельно, а потім інформаційна система збирає всі результати у єдиній точці. Такий підхід дозволяє максимально ефективно використовувати ресурси, особливо у випадку комплексного аналізу.

Оскільки в даній клієнт-серверній архітектурі є можливість відправляти кастомні headers, для підвищення рівня анонімності та захисту даних реалізована можливість використання x-forwarded-for хедера. Цей хедер дозволяє вказати проксі-сервер, через який буде відправлений запит, що забезпечує додатковий рівень захисту та контролю над мережею. Це особливо корисно для систем, які потребують маскування реального місця відправника запиту, наприклад, при перевірці завантаження ресурсу для конкретної геолокації (регіону, країни) або обробці конфіденційних даних.

Кінцевим результатом роботи інформаційної системи є аудит-звіт (репорт), що узагальнює результати всіх асинхронних обчислень та оцінок.

Генерація репорту реалізована у трьох форматах:

– консольний формат – вивід у консоль, зручний для швидкої перевірки без запуску браузера;

– JSON формат – для подальшої автоматичної обробки або інтеграції зі сторонніми системами;

– HTML візуальний звіт з деталізацією оцінок, статусів, підказками та кольоровою індикацією.

Формування звіту є результатом асинхронної обробки даних, де всі модулі інформаційної системи працюють паралельно, а після завершення їхні результати збираються в єдину структуру, яка й лягає в основу репорту. Це дозволяє зробити аналіз швидким та гнучким, а репорт інформативним та зрозумілим для кінцевого користувача.

3.5 Алгоритмізація логіки застосунку

У процесі планування роботи аналітичної складової системи визначено, що результати, отримані в процесі збору даних, розподілятимуться по спеціалізованих модулях, кожен із яких відповідатиме за конкретний аспект аудиту. Зокрема, дані надходитимуть у:

– модуль SEO для перевірки пошукової оптимізації;

– модуль Best Practices для оцінки відповідності найкращим стандартам розробки;

– модуль Performance для вимірювання швидкодії та продуктивності;

– модуль Special Features, який фокусується на аналізі технічних характеристик і унікальних особливостей вебзастосунку.

Компоненти інформаційної системи – модулі SEO, Best Practices, Performance та Special Features, виконують свою роботу за певним алгоритмом. Для застосунку інформаційної системи розроблено діаграми діяльності (рис. 3.8) для графічного відображення алгоритмів кожного модуля.

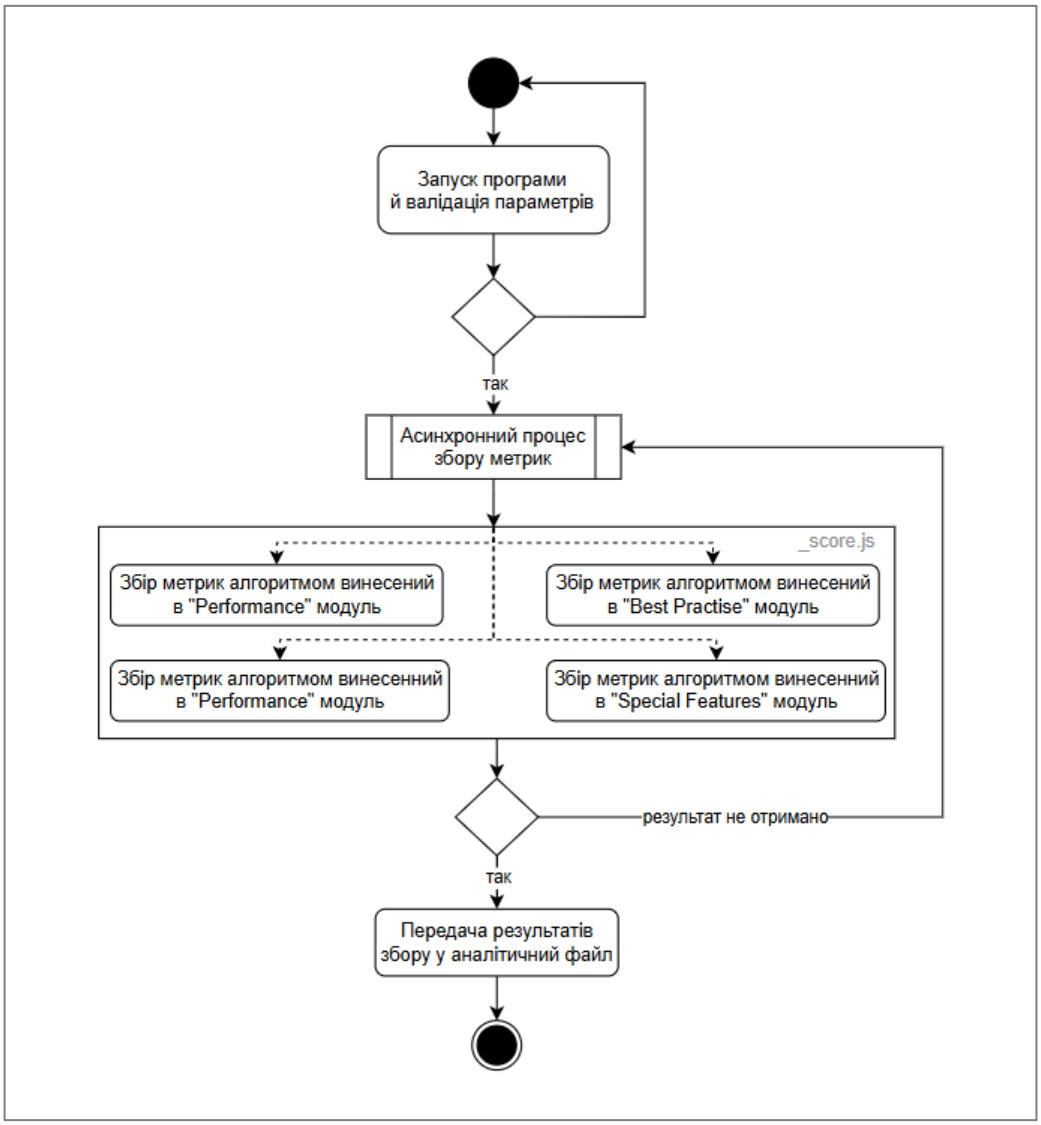


Рисунок 3.8 – Діаграма діяльності

Оцінювальних алгоритмів винесені в окремі файли з приписом «…»_score. Кожен із цих модулів є незалежним та повертає структурований об’єкт результатів.

Блок-схема підрахунку підсумкового показника продуктивності Performance S_{perf} , визначеного за формулою 2.2, відображена на рисунку 3.9.

Алгоритм отримує значення метрик LCP, INP, CLS, FCP і TTFB. Якщо метрики відсутні то повертається помилка і 0. Якщо є – початковий бал 100. Для кожної метрики визначені порогові рівні (зелений, жовтий, червоний). Залежно від

зони значення метрики, від балу віднімається відповідний штраф. Після обробки всіх метрик бал обмежується мінімумом 0 і повертається.

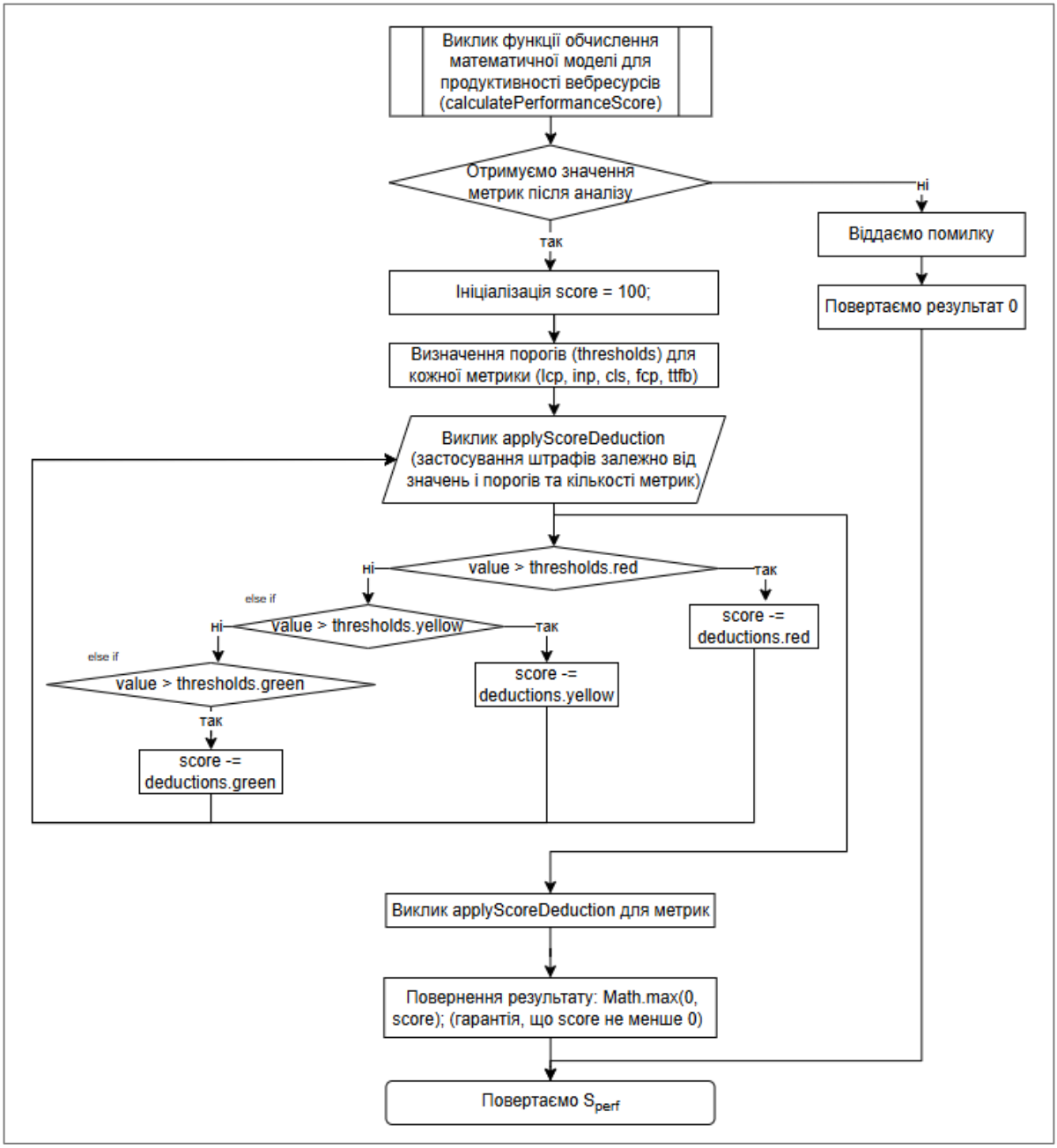


Рисунок 3.9 – Блок-схема метода підрахунку S_{perf}

Діаграма діяльності підрахунку підсумкового показника продуктивності Performance S_{perf} , визначеного за формулою 2.2, відображена на рисунку 3.10.

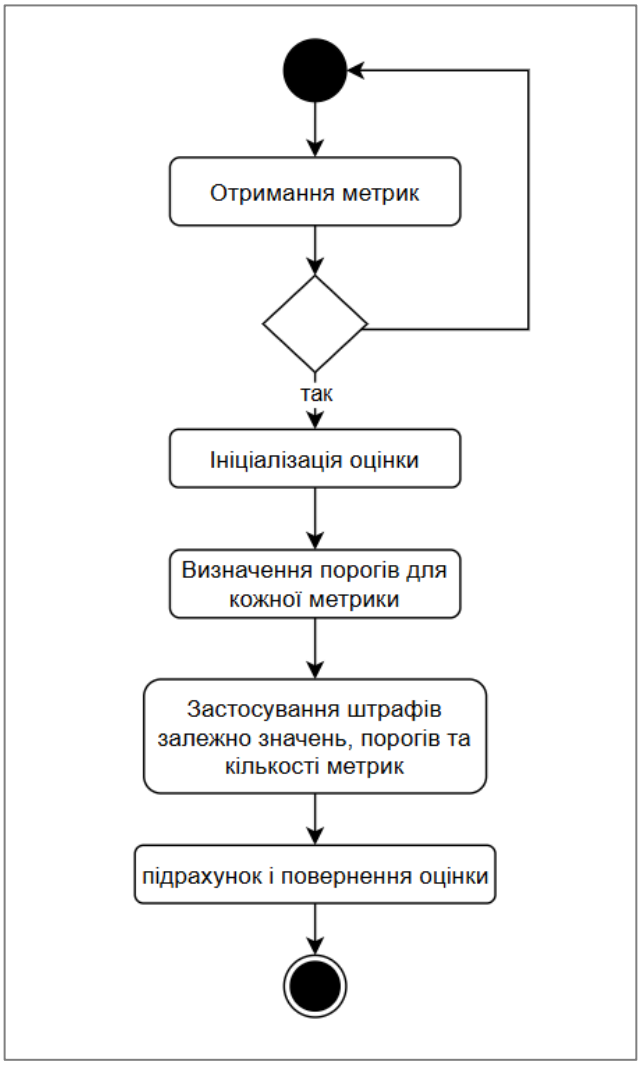


Рисунок 3.10 – Діаграма діяльності S_{perf}

Блок-схема підрахунку комплексного показника SEO-оцінки S_{SEO}, визначеного за формулою 2.4, відображена на рисунку 3.11, в котрій викликається функція для обчислення моделі SEO та отримання метрик. Якщо результати відсутні то повертається помилка і нуль. Якщо є, початковий бал встановлюється на 100, і для кожної метрики визначаються порогові рівні (червоний, жовтий, зелений). За допомогою функції applyScoreDeduction бал зменшується залежно від порогової зони метрики, причому червона дає найбільше зменшення, зелена – найменше. Після обробки всіх метрик викликається функція ще раз для додаткових коригувань. Наприкінці повертається максимум між 0 і обчисленим балом, щоб уникнути від’ємних значень.

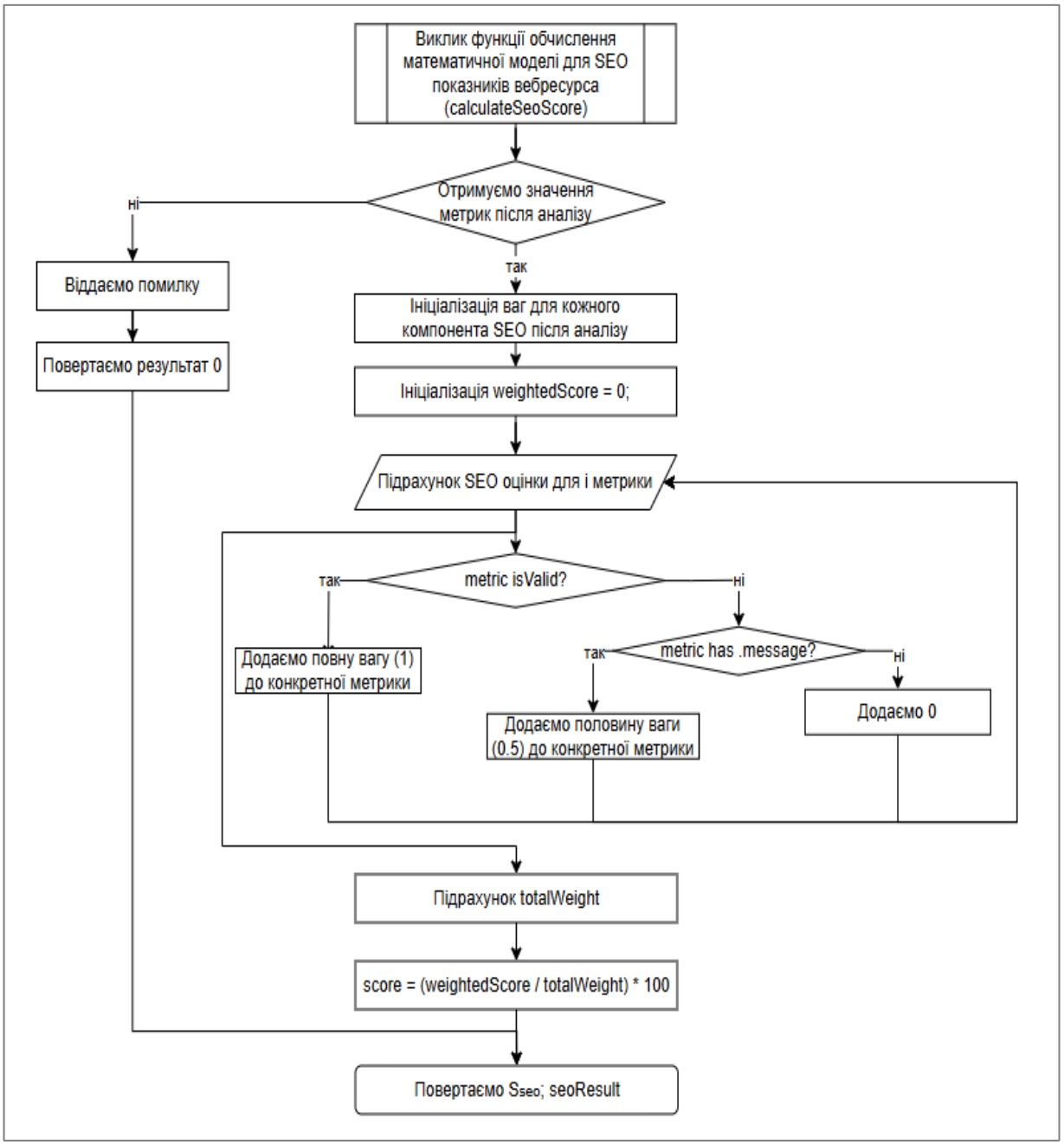


Рисунок 3.11 – Блок-схема метода для підрахунку S_{SEO}

Діаграма діяльності підрахунку комплексного показника SEO-оцінки S_{SEO} , визначеного за формулою 2.4, відображена на рисунку 3.12.

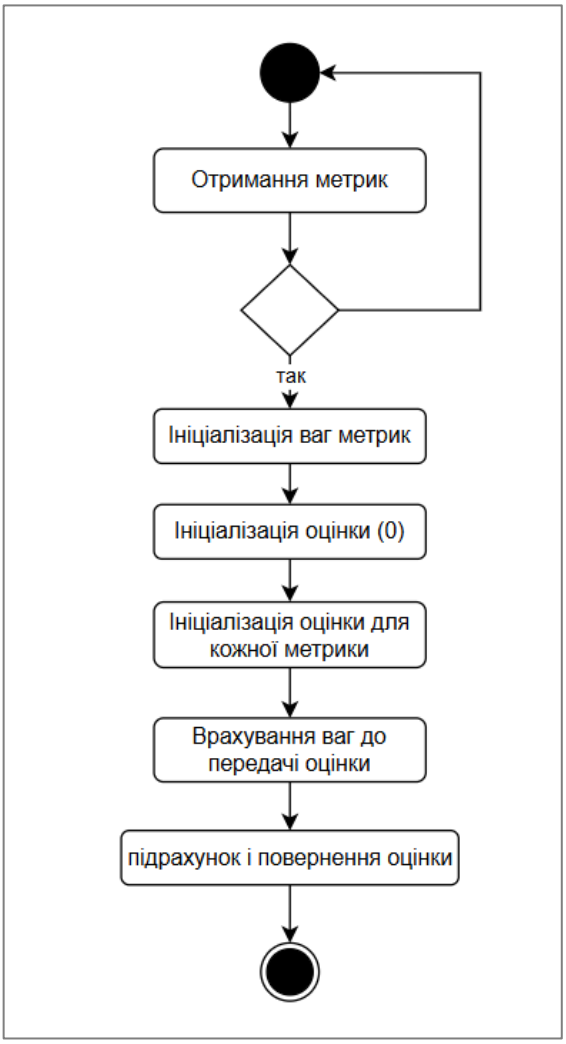


Рисунок 3.12 – Діаграма діяльності S_{SEO}

Блок-схема підрахунку оцінки Best Practices вебресурсу S_{BP} , визначеного за формулою 2.6, відображена на рисунку 3.13. Алгоритм обробляє результати аудиту для обчислення загального балу S_{BP}. Для кожного елемента визначається тип результату: якщо булевий і значення false то віднімається половина ваги елемента; якщо числовий – залежно від зон порогів (червоний, жовтий, зелений) або відхилення від 100, відбувається відповідне зниження балу. Ваги за замовчуванням рівні 1. Після обробки всіх елементів результат гарантує, що бал не буде меншим за 0, і повертається як кінцевий результат.

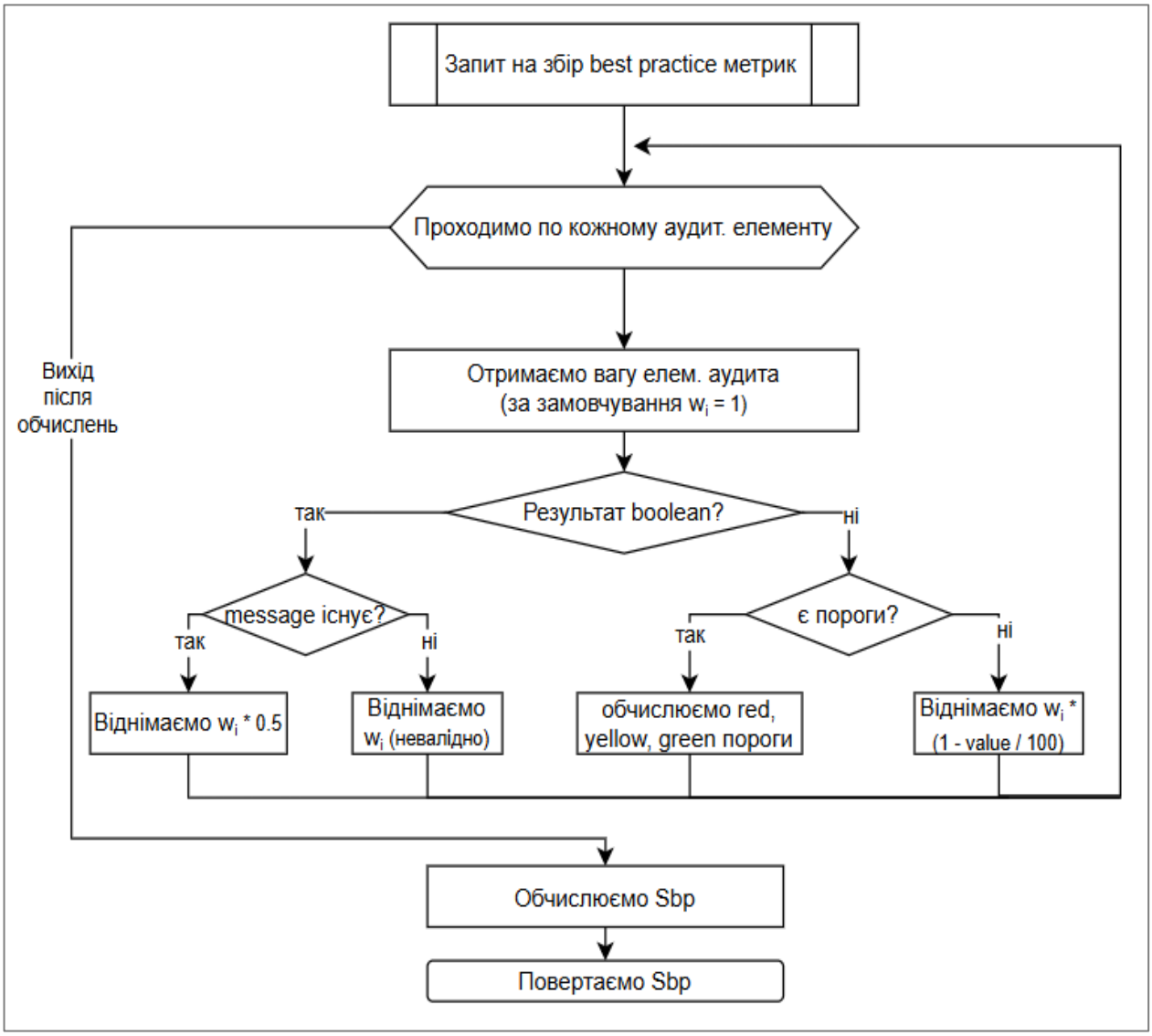


Рисунок 3.13 – Блок-схема метода підрахунку S_{BP}

Алгоритмізація логіки застосунку є важливим етапом розробки, що забезпечує структурований підхід до обробки даних і проведення аудиту вебзастосунків. Для ефективного функціонування інформаційної системи було визначено чітку логіку роботи кожного модуля, що були вищеописані, попередньо розробивши модель для підрахунку секції метрик.

Діаграма діяльності підрахунку комплексного показника SEO-оцінки S_{SEO} , визначеного за формулою 2.6, відображена на рисунку 3.14.

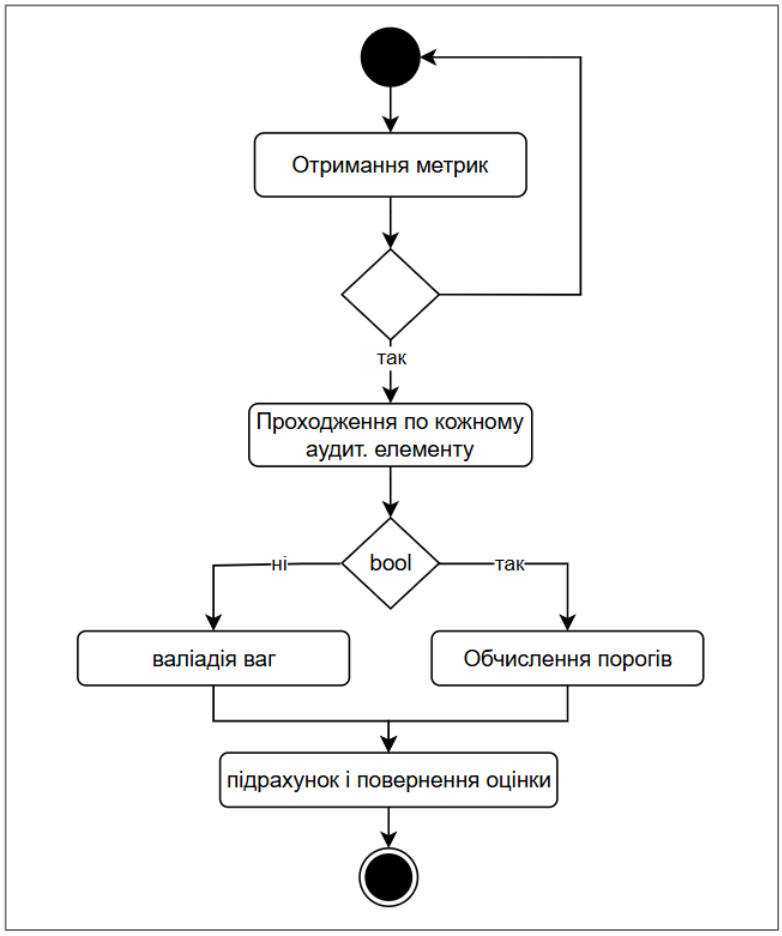


Рисунок 3.14 – Діаграма діяльності S_{BP}

Для зручності та візуалізації роботи кожного модуля було створено діаграми діяльності (рис. 3.8, рис. 3.10, 3.12 та 3.14), що наочно демонструють основні кроки алгоритмів (рис. 3.9, 3.11 та 3.13). Діаграма та блок-схеми дозволяють легко простежити весь процес обробки даних, від початкового збору показників до видачі підсумкових результатів. Це сприяє більш чіткому розумінню внутрішньої логіки роботи системи та забезпечує зручність у підтримці та модифікації застосунку.

Висновки до розділу 3

У третьому розділі було зосереджено увагу на технічних засадах та підходах до розробки інформаційної системи, реалізованої у форматі CLI-інструменту для аналізу продуктивності web-застосунків.

Вибір технологій, зокрема мови програмування JavaScript та середовища Node.js, обумовлений їх широкими можливостями для розробки високопродуктивних і масштабованих вебсистем. JavaScript забезпечує необхідну кросплатформність, підтримку функціонального програмування, а також великий вибір бібліотек і фреймворків для спрощення розробки. Node.js, зокрема, є ідеальним рішенням для серверної частини проєкту завдяки своїй асинхронній архітектурі, що дозволяє обробляти велику кількість одночасних запитів без блокування основного потоку виконання.

Ключовим інструментом для автоматизації взаємодії з вебресурсами став Puppeteer, що дозволяє ефективно керувати браузером Chromium у headless режимі, емулюючи реальні сценарії користувача. Завдяки Puppeteer система може збирати дані про продуктивність вебзастосунків, аналізувати метрики UX, SEO та перевіряти відповідність Best Practices. Інтеграція цього інструменту з Node.js створює зручну та потужну платформу для розробки системи, здатної забезпечити всебічний аудит вебресурсів.

Таким чином, вся архітектура проєкту базується на простоті, універсальності та продуктивності JavaScript, поєднаному з потужністю Node.js для реалізації серверної логіки та бібліотекою для взаємодії з браузером – Puppeter. Також важливою складовою є реалізація асинхронної обробки запитів, що дозволяє одночасно обробляти великі обсяги даних без значних затримок.

Інтеграція результатів обробки в єдиний звіт, який надається користувачу в різних форматах (консольний, JSON, HTML), забезпечує зручність і можливість подальшого використання даних для автоматизації або інтеграції з іншими системами.

Завдяки модульному підходу і інтеграції асинхронних механізмів обробки даних, система дозволяє швидко і зручно здійснювати аудит вебресурсів, а також отримувати детальні звіти з рекомендаціями для поліпшення продуктивності.

4 ПРОГРАМНА РЕАЛІАЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ПРОДУКТИВНОСТІ WEB-ЗАСТОСУНКІВ

4.1 Конфігурація бібліотеки Puppeteer для вебскрейпінгу

Під час початкової конфігурації проєкту, однією з перших залежностей була встановлена бібліотека Puppeteer за допомогою команди npm і puppeteer. Як було зазначено в розділі 3.3, Puppeeter є високорівневий інструментом для автоматизації браузера, що працює поверх Chromium або Chrome. Його ключовою перевагою є можливість керування браузером у headless-режимі, тобто без графічного інтерфейсу, що дозволяє виконувати дії у фоновому режимі. Через API Puppeteer можна відкривати сторінки, емулювати дії користувача, зчитувати DOM-елементи, збирати метрики продуктивності та багато іншого.

Одним із найпоширеніших сценаріїв використання Puppeteer є вебскрейпінг – процес автоматичного збору даних з вебсторінок. Фактично, це розширена форма звичайного пошуку інформації на вебсторінках, яка масштабується до сотень або тисяч запитів, що здійснюються за заданими параметрами. У випадку інформаційної системи це означатиме, що програма здатна в автоматизованому режимі аналізувати сторінки, вивантажувати їх вміст та витягувати з них технічну інформацію, що стосується продуктивності, структури HTML, SEO або загальних best practices. Такий підхід дозволяє масштабувати процес аналізу ресурсів [23].

Створимо окрему допоміжну директорію в проєкті – helper, де будуть зберігатись утилітарні компоненти системи котрі будуть перевикористовуватись в рамках проєкту. У межах цієї теки створимо файл analyzePageData.js де реалізуємо функцію, яка ініціалізує браузер через puppeteer.launch(). Цей метод повертає об'єкт типу Browser (через проміс), який дозволяє відкривати вкладки, взаємодіяти з контентом сторінок та аналізувати технічні характеристики ресурсу. Далі створюється нова сторінка за допомогою const page = await browser.newPage(); page

використовується для створення нової вкладки (сторінки) в запущеному браузері Puppeteer для подальшої взаємодією зі сторінкою вебресурсу.

```
export async function analyzePageData(url, deviceName) {
    const stopSpinner = startSpinner(`Loading page on ${deviceName || 'desktop'}...`);
    const browser = await puppeteer.launch({
        headless: true,
        devtools: false,
        args: ['--no-sandbox',
            '--disable-setuid-sandbox',
            '--disable-web-security',
            '--disable-features=IsolateOrigins',
            '--disable-site-isolation-trials'],
        ignoreDefaultArgs: ['--disable-extensions'],
    });
    const page = await browser.newPage();
```

Рисунок 4.1 – Ініціалізація браузера в headless-режимі та отримання сторінки для подальшої взаємодії з нею через API Puppeteer

На рис. 4.1 зображено набір параметрів для налаштування запуску браузера Puppeteer. Ці параметри контролюють, чи буде відображатися графічний інтерфейс, чи будуть відкриті інструменти розробника, а також впливають на безпеку та поведінку браузера на більш глибокому рівні [24]. Основні з них відповідають за:

– headless: true визначає, чи запускати браузер у «безголовому» режимі (без графічного інтерфейсу). Значення «true» означає, що браузер працюватиме у фоновому режимі, без відображення вікна. Це must have для проведення аудиту в рамках консольного застосунку;

– disable-web-security – цей аргумент вимикає політику одного походження (Same-Origin Policy) та інші механізми веб безпеки. Це дозволяє сторінкам завантажувати ресурси з різних доменів без обмежень;

– no-sandbox – цей аргумент вимикає sandbox-режим браузера. Sandbox – це механізм безпеки, який ізолює процеси браузера. Вимкнення sandbox може бути

необхідним у деяких обмежених середовищах (наприклад, у контейнерах Docker з подальшою інтеграцією на CICD);

– disable-features=IsolateOrigins та disable-site-isolation-trials відключають експериментальні функції та механізмі ізоляції (дозволяє обходити безпекові рішення, такі як DDos Guard, Cloudflare та інші).

У процесі дослідження зовнішніх інструментів аудиту вебзастосунків та формулювання завдань системи (розділи 1.2 і 1.3), було встановлено доцільність реалізації подвійного підходу до аналізу – у desktop та mobile режимах. Такий підхід дозволяє інформаційній системі ширше охоплювати сценарії взаємодії з вебресурсами, враховуючи різну поведінку інтерфейсів залежно від типу пристрою користувача.

Для цього в проєкті створюється файл call.js в теці helper, де описуються два окремі об'єкти конфігурації пристроїв, які виступають в ролі тригерів запуску режиму в headless-браузера, ініціалізованому Puppeteer.

Ключовим параметром у таких об'єктах є user-agent – це звичайний рядок, який браузер надсилає серверу під час HTTP-запиту для ідентифікації себе. Він містить інформацію про операційну систему, тип пристрою, версію браузера тощо [25]. У контексті роботи інформаційної системи, коректно вказаний user-agent дозволяє точно симулювати поведінку реального пристрою, забезпечуючи релевантність отриманих даних під час аудиту. Від цього залежить, який контент або структура буде повернута сервером, зокрема адаптивні стилі, поведінка скриптів, завантаження ресурсів і SEO-елементи.

На рис. 4.2. продемонстровані об'єкти mobileDevice і desktopDevice котрі описують набір параметрів, що застосовуються до headless-вкладки Puppeteer перед запуском аудиту. Параметри включають роздільну здатність екрана, масштабування, наявність сенсорного вводу та режим landscape, що критично важливо для адекватного тестування UI/UX компонентів. Такий розподіл режимів дозволяє виконати аналіз у максимально наближених до реальності умовах, підвищуючи точність діагностики продуктивності та якості сайту.

```
// helper/call.js

export const mobileDevice = {
    name: 'Pixel 8',
    userAgent: 'Mozilla/5.0 (Linux; Android 14; Pixel 8 Build/UPD1.220817.017) AppleWebKit/537.36 (KHT
    viewport: {
        width: 412,
        height: 915,
        deviceScaleFactor: 3.0,
        isMobile: true,
        hasTouch: true,
        isLandscape: false,
    },
};

export const desktopDevice = {
    userAgent: 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
    viewport: {
        width: 1280,
        height: 720,
        deviceScaleFactor: 1,
        isMobile: false,
        hasTouch: false,
        isLandscape: false,
    },
};
```

Рисунок 4.2 – Встановлення параметрів для запуску сторінок

Повернемось до analyzePageData.js файлу та реалізуємо базову умовну логіку, яка дозволить визначати, який тип пристрою повинен емулюватися під час запуску аудиту. Для цього буде використано просту конструкцію if-else, що перевірятиме параметри, передані користувачем через консольний інтерфейс. У разі відсутності конкретних параметрів, то для забезпечення повного охоплення аналізу за замовчуванням буде запускатися перевірка у двох режимах одночасно – мобільному та десктопном.

Після завершення всіх дій на сторінці браузера (збору даних), закриємо браузер Puppeteer за допомогою асинхронної функції browser.close().

4.2 Програмна реалізація модулів інформаційної системи

У процесі розробки проєкту було обрано компонентний підхід, що забезпечує структурованість, модульність та гнучкість системи. Компонентний підхід дозволяє розділити функціональність програми незалежні частини, які можна

окремо проєктувати, реалізовувати та тестувати. Це сприяє підвищенню читабельності коду, адже кожен компонент відповідає за свою, чітко визначену задачу. Також такий підхід спрощує процес підтримки та оновлення системи, оскільки внесення змін до одного модуля не впливає на решту компонент, за умови чіткого дотримання зон відповідальності між ними [26].

Однією з вагомих переваг компонентного підходу є здатність повторно використовувати раніше створені модулі. Компоненти можуть бути легко адаптовані для різних проєктів або загальних рішень, що робить їх ідеальними для Open Source ініціатив. Сторонні користувачі зможуть не лише використовувати готові компоненти в своїх власних перевірках, але й доповнювати, модифікувати їх відповідно до специфічних потреб своїх проєктів [27]. Це значно пришвидшує розробку ПЗ, адже розробникам не потрібно витрачати час на створення одних і тих самих рішень з нуля.

Для організації структури проєкту у рамках обраного підходу, буде створена відповідна директорія під назвою audits (рис. 4.3). У середині цієї директорії буде створено п’ять допоміжних папок, кожна з яких відповідатиме за певний аспект аудиту. Назви цих папок будуть такими:

– seo: папка для компонентів, що займаються перевіркою факторів, спрямованих на поліпшення пошукової оптимізації;

– performance: компоненти для оцінки швидкості завантаження та загальної продуктивності вебсторінки;

– best_practise: зібрання компонентів, що реалізують різноманітні кращі практики у розробці вебзастосунків;

– special_features: компоненти для перевірки специфічних, нестандартних можливостей вебзастосунків, які можуть бути корисні в специфічних сценаріях;

– recommendations: папка для компонентів, які формують рекомендації щодо покращення вебзастосунку на основі проведеного аудиту.

На рис. 4.4 представлено діаграму функціональної структури інформаційної системи, що забезпечує повний цикл аналізу та моніторингу вебзастосунків конкретних метрик.

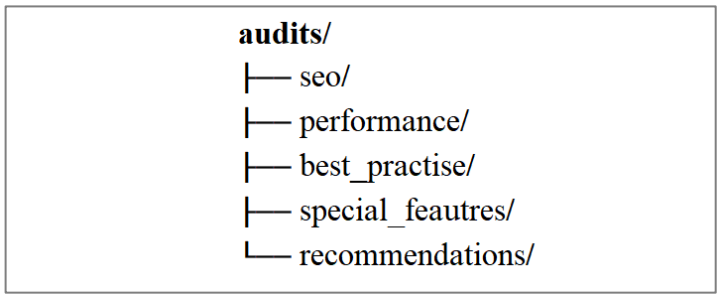


Рисунок 4.3 – Структура тек з компонентами для проведення аудиту

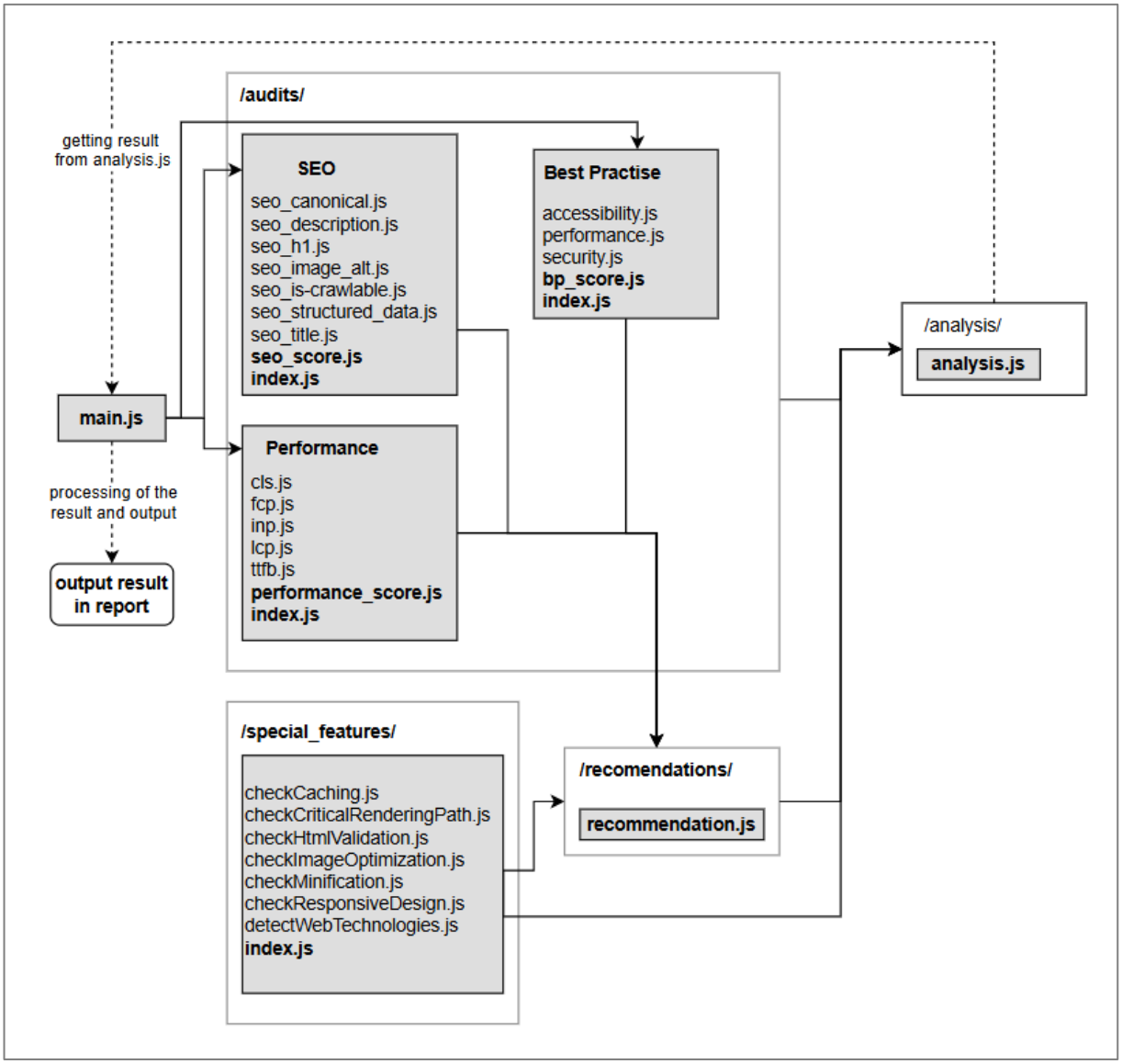


Рисунок 4.4 - Діаграма функцій CLI-застосунку

Діаграма функціональної структури демонструє взаємодію між ключовими модулями, які відповідають за збір, обробку та інтерпретацію даних, а також за

генерацію звітів і рекомендацій. Усі компоненти, відповідальні за розрахунок оцінок, дозволяють системі виконувати аудит у автоматичному режимі. Така структура забезпечує гнучкість, масштабованість і готовність до розширення функціоналу в майбутньому. Компонент analysis.js котрий відповідає за здійснення та запуск аналізу буде продемонстровано в наступному розділі.

Перейдемо до реалізації конкретних компонентів у рамках обраної структури проєкту, розпочнемо з seo/. Ця папка слугуватиме основою зберігання для компонентів, що відповідатимуть за перевірку різноманітних аспектів пошукової оптимізації вебсторінок, на основі метрик, описаних у параграфі 2.2. В кожному з цих компонентів реалізуємо методи, котрі виконуватимуть збір необхідних даних та повертати їх. Основні функції та їх призначення описано у таблиці 4.1. Ці функції автоматизовано збирають ключові метрики SEO та аспекти доступності вебзастосунку, використовуючи можливості Puppeteer для аналізу DOM-структури та метаданих сторінок. Отримані результати перевірок слугують для оцінки та моніторингу «здоров’я» (health) вебзастосунку з точки зору пошукової оптимізації та технічної реалізації.

Для зручності обробки, всі метрики передаються окремо у файл seo_score.js, де вони використовуються для підрахунку загальної оцінки після проведення аудиту SEO. Код продемонстровано в додатку А.

Перейдемо до реалізації параметрів продуктивності у папку /performance. Основні метрики для аналізу продуктивності були виведені в параграфі 2.1. Після запуску браузера за допомогою Puppeteer, він автоматично зчитуватиме та збиратиме результати виконання функцій, які написанні у відповідних компонентах. Ряд цих функцій дозволить отримувати конкретні дані про продуктивність вебзастосунку в режимі реального часу. Після завершення виконання функцій Puppeteer повертатиме зібрані результати, що значно спростить аналіз і оцінку продуктивності. Далі перенаправляємо дані для проведення оцінки продуктивності у відповідний файл performance_score.js. Файл з методом підрахунку оцінки продуктивності подано в додатку Б.

Таблиця 4.1 – Функції в директорії /seo

Назва функції	**Призначення**
checkCanonical	Перевіряє наявність та коректність канонічного URL на сторінці для уникнення проблем з дублюванням контенту
checkDescription	Аналізує наявність та вміст мета-тегу description для SEO та відображення в результатах пошуку
checkH1	Перевіряє наявність та релевантність заголовка першого рівня (H1) для SEO та структури сторінки
checkImageAlt	Контролює наявність та описовість атрибутів alt для зображень з метою доступності та SEO
checkIsCrawlable	Визначає, чи доступна сторінка для індексації пошуковими роботами (перевіряє robots.txt, мета-теги noindex тощо)
checkTitle	Аналізує наявність та релевантність тегу title для SEO та відображення у вкладці браузера
checkStructuredData	Перевіряє наявність та валідність структурованих даних (schema.org) для покращення відображення в пошукових результатах

Таблиця 4.2 – Функції в директорії /performance

Назва функції	**Призначення**
calculateCLS	Міряє візуальні стрибки сторінки
calculateFCP	Зчитує час до першого відображення контенту
calculateINP	Дізнається затримку реакції на дії користувача
calculateLCP	Вимірює час до відображення найбільшого елемента
calculateTTFB	Отримує час від запиту до першого байта від сервера

Запуск всіх вищеописаних функцій помістимо у файлі index.js. Усередині цього файлу реалізована функція, яка запускає всі аудити паралельно за допомогою

Promise.all. Такий підхід оптимізує час збору та аналізу метрик, що забезпечує швидку віддачу оцінки продуктивності.

Наступним кроком є реалізація метрик, які дозволять оцінити дотримання кращих практик (best practices) вебзастосунків. Ці метрики служитимуть критеріями для оцінки якості коду та відповідності встановленим стандартам. Ряд метрик best practise, описаних у параграфі 2.3, реалізовано у відповідній теці /best_practise. Перейдемо до огляду основних функцій (табл. 4.3).

Таблиця 4.3 – Функції в директорії /best_practise

Назва функції	**Призначення**
checkAccessibilityBestPractices	Функція аналізує різні аспекти HTML-структури, атрибутів елементів та контенту, щоб виявити потенційні проблеми, які можуть ускладнити використання сайту для людей з обмеженими можливостями (наприклад, відсутність alt атрибутів для зображень, неправильна структура заголовків, проблеми з контрастністю кольорів тощо
checkPerformanceBestPractices	Функція перевіряє вебсторінку на відповідність найкращим практикам продуктивності (виявлення застарілих API, перевірку наявності сторонніх cookie, аналіз співвідношення сторін зображень, перевірку роздільної здатності зображень, наявність мета-тегу viewport, тощо
checkSecurityBestPractices	Функція аналізує наявність HTTPS, заголовки безпеки (Content Security Policy, Strict-Transport-Security), вразливості в застарілих бібліотеках, безпеку cookie та інші фактори, що впливають на безпеку вебзастосунку

Для підрахунку оцінки дотримання кращих практик усі компоненти передаються в bp_score.js, при цьому спочатку викликаються з файлу index.js, що дозволяє виконувати їх асинхронно та паралельно, використовуючи багатопоточність. Лише після завершення цих перевірок дані передаються до методу обчислення оцінки (bp_score.js). Код підрахунку оцінки дотримання best practise подано в додатку В.

Останнім аудиторським модулем є аналіз технічних складових вебпроєктів. Створимо теку /special_features в котрій реалізуємо функціональним підходом методи для аналізу технічної складової. Основні функції аудиту технічної сторони описано у таблиці 4.4.

Таблиця 4.4 – Функції в директорії /special_features

Назва функції	**Призначення**
checkCaching	Перевіряє, чи правильно налаштовано кешування для швидшого повторного завантаження сторінки
checkCriticalRenderingPath	Аналізує, чи оптимізовано завантаження видимого контенту для швидшого першого відображення
checkHtmlValidation	Перевіряє HTML-код на наявність синтаксичних помилок
checkImageOptimization	Контролює, чи оптимізовано розмір та формат зображень для кращої продуктивності
checkMinification	Перевіряє, чи стиснуто (мініфіковано) файли CSS та JavaScript для зменшення їх розміру
checkResponsiveDesign	Аналізує, чи коректно відображається сайт на різних розмірах екранів
detectWebTechnologies	Визначає, які технології (CMS, фреймворки, бібліотеки) використовуються на сайті (за допомоги якого інструмента був побудований вебзастосунок)

Основні компоненти технічного аналізу, визначені у параграфі 2.4, є критично важливими, оскільки даний функціонал виконує комплексну оцінку технічних аспектів. Мета цього аналізу полягає в ідентифікації можливих прогалин і технічних проблем, які можуть негативно вплинути на якість продукту, його ефективність та стабільність. Цей модуль не має за мету визначення оцінки технічних складових, оскільки його основна задача полягає в тому, щоб підсвітити аспекти, які потребують покращення у вебресурсах для забезпечення безперебійної та якісної роботи системи. Такий підхід дозволяє ідентифікувати та усунути слабкі місця, що, в свою чергу, підвищує загальну надійність і продуктивність проєкту. Зосереджуючи увагу на оптимізації технічних компонентів, модуль сприяє створенню більш ефективних і стабільних рішень, що сприятиме досягненню гарних стандартів якості у розробці та експлуатації вебресурсів.

Після завершення збору даних кожним з компонентів, наступним кроком є обробка та подальша інтерпретація результатів. Усі ці результати імпортуються до окремого модуля – recommendations.js. Саме тут буде зосереджена основна логіка надання рекомендацій для аналізу: спеціально розроблена функція прийматиме на вхід отримані метрики продуктивності, доступності, SEO та інших показників. На основі цих даних функція генерує чіткі, релевантні рекомендації щодо підвищення оцінок для майбутніх аудитів, усунення проблем і загального покращення стану вебзастосунку. Метод рекомендацій також ідентифікує критичні помилки або недоліки, які безпосередньо впливають на кінцеву якість ресурсу. Такий компонентний підхід дозволяє централізовано керувати аналізом і забезпечує наочну видачу «інструкцій» (результатів), необхідних для оптимізації сайту після кожного проведеного аудиту.

4.3 Збірка проєкту

Збірка консольного застосунку для аудиту та моніторингу вебзастосунків реалізована у вигляді логічно впорядкованої, компонентної структури, що поєднує

ряд модулів, кожний з яких виконує окрему функцію в межах єдиного процесу. Центральну роль у цьому процесі відіграє бібліотека Puppeteer, що забезпечує запуск headless-браузера для автоматичного збору показників із цільових вебресурсів. Завдяки можливості безпосередньої взаємодії з елементами сторінки, Puppeteer дозволяє отримувати актуальні дані в режимі реального часу, що слугує основою для подальшого аналізу.

Після збору даних результати передаються до модуля analysis.js, який відповідає за первинну обробку отриманих метрик. Цей файл містить набір функцій, що аналізують критичні показники, перевіряють їх відповідність базовим критеріям, і структурують інформацію для подальшого використання.

Аналіз здійснюється окремо по кожній оцінювальній групі, що дозволяє точно визначити проблемні місця та зрозуміти природу втрати продуктивності. Далі оброблені результати передаються до recommendations.js, де реалізований логічний фунціонал інтерпретації даних. Саме тут формуються висновки та рекомендації щодо покращення якості вебресурсу. Ці поради базуються на виявлених недоліках і мають практичне спрямування – вони враховують сучасні вимоги до вебсторінок з точки зору безпеки, продуктивності, доступності тощо.

Всю систему координує основний файл main.js, що слугує вхідною точкою застосунку (додаток Г). Саме він ініціалізує запуск усіх компонентів, об’єднує окремі модулі в єдиний ланцюжок, і відповідає за підготовку повного звіту. Завдяки цьому запускаючи лише один файл, користувач отримує повноцінний аудит сайту за всіма критеріями.

Компонентна архітектура проєкту сприяє його масштабованості, легкому тестуванню й обслуговуванню. Кожен компонент існує у вигляді окремого файлу з чітко визначеним функціоналом, що дозволяє розширювати можливості застосунку без ризику порушити існуючу логіку [28].

Такий підхід добре узгоджується з принципами Open Source, оскільки забезпечує прозорість, зрозумілість коду та потенціал для зовнішнього внеску від інших розробників.

Завдяки такій структурі проєкт має можливість оперативно реагувати на змінювані умови ринку та потреби користувачів. Регулярне оновлення даних і функціональних можливостей гарантує, що інструмент залишається актуальним і цінним для різних категорій користувачів, від індивідуальних підприємців до компаній, забезпечуючи ефективний аналіз web-застосунку.

4.4 Інтерфейс інформаційної системи та аналіз продуктивності вебресурсів

Інтерфейс інформаційної системи, що відповідає за аудит і моніторинг вебзастосунків, реалізовано у кількох форматах, орієнтованих на різні потреби користувача. Основним способом взаємодії є консольний режим, що підходить для швидкого запуску, перевірки та аналізу.

Оскільки консольний інтерфейс не передбачає складного візуального оформлення, результати виводяться у вигляді лаконічного тексту з ключовими метриками (рис. 4.5), які легко зчитуються та інтерпретуються вручну. Такий підхід дозволяє оперативно оцінити загальний стан вебзастосунку без зайвого перевантаження інформацією.

Для більш детального технічного аналізу та інтеграційного використання передбачено альтернативний формат виводу – у вигляді структурованого JSON-об'єкта. Достатньо додати флаг -j (або --json) при запуску застосунку, і користувач отримає повний звіт із усіма зібраними метриками, деталізацією помилок, а також технічними рекомендаціями (рис. 4.6).

Це особливо корисно в рамках автоматизованих сценаріїв або при інтеграції в CI/CD-процеси, де після кожного оновлення системи потрібно перевірити, чи не вплинули нові зміни на ключові показники продуктивності.

```
Desktop Analysis:
Performance Score: 81
LCP: 3900 ms
INP: 0 ms
CLS: 0.00732061794704861
FCP: 3783.2999999988824 ms
TTFB: 2263.39999999851 ms
SEO Score: 79
Technical Aspects: {
  caching: {
    isCachingConfigured: true,
    message: 'Caching is properly configured for all resources.'
  },
  criticalRenderingPath: {
    isCriticalPathOptimized: true,
    message: 'Critical rendering path is optimized.'
  },
  htmlValidation: {
    isValid: true,
    issuesCount: 0,
    message: 'HTML validation: No potential issues found.'
  },
  imageOptimization: { areImagesOptimized: true, message: 'Images are optimized.' },
  minification: {
    areResourcesMinified: false,
    message: 'Some resources are not minified.'
  },
```

Рисунок 4.5 – Стандартний вивід результату аудиту консольного застосунку


```
  "seoScore": {
    "score": 79,
    "results": {
      "title": {
        "title": "Trade Forex, CFDs, Gold & Oil | Justmarkets",
        "isValid": true,
        "message": "Title is valid. Length: 43 characters."
      },
      "description": {
        "description": "Sign up and trade with the best forex broker! ► JustMarkets offers the lowest spreads, high le
verage up to 1:3000 and more than 90 financial instruments for trading  No requotes  Fast execution.",
        "isValid": true,
        "message": "Meta description is valid. Length: 196 characters."
      },
      "h1": {
        "h1Count": 1,
        "isValid": true,
        "message": "Exactly one H1 heading found."
      },
      "imageAlt": {
        "totalImages": 35,
        "imagesWithoutAlt": 22,
        "isValid": false,
        "message": "22 out of 35 images are missing alt attributes."
      },
```


Рисунок 4.6 – JSON вивід результату аудиту консольного застосунку

Для покращення взаємодії з користувачем та інформування його про поточний статус роботи системи, у консольному застосунку реалізовано

візуалізацію процесу завантаження за допомогою бібліотеки cli-spinner (рис. 4.7). Цей елемент виконує роль індикатора, який демонструє, що система розпочала збір та обробку даних, тим самим підтверджуючи користувачу, що процес виконання запиту активний і результат буде представлено найближчим часом.

```
// helper/spinner.js

import cliSpinners from 'cli-spinners';

export function startSpinner(text) {
    const spinner = cliSpinners.bouncingBar;
    let i = 0;
    const interval = setInterval(() => {
        process.stdout.write(`\r${spinner.frames[i = ++i % spinner.frames.length]} ${text}`);
    }, spinner.interval);

    return () => {
        clearInterval(interval);
        process.stdout.write('\r');
    };
}
```

Рисунок 4.7 – Реалізація індикатора завантаження й обробки результату

Спіннер використовується для двох незалежних потоків аналізу – десктопної та мобільної версій. Бібліотека cli-spinner також реагує на зміну стану запиту: вона сигналізує про старт процесу збору даних, а після завершення – автоматично припиняє анімацію, інформуючи користувача про завершення операції. Це створює відчуття контрольованого процесу та покращує загальне UX-враження при роботі з інструментом (рис. 4.8).

```
C:\Users\Пользователь\Desktop\Uni\cli3>node --no-warnings ps5.js -u https://justmarkets.com
[  ==] Loading page on desktop...


C:\Users\Пользователь\Desktop\Uni\cli3>node --no-warnings ps5.js -u https://justmarkets.com
Page loaded on desktop successfully.
[ ===] Loading page on mobile...
```

Рисунок 4.8 – Індикація завантаження й оброки результатів під час виконання

Усвідомлюючи важливість наочної інтерпретації результатів аудиту, в рамках інформаційної системи було реалізовано функціонал автоматичного формування HTML-звітів за певним параметром при запуску аудиту [29] (--html-report або -r). Після завершення аналізу всі зібрані метрики, технічні висновки та рекомендації структуруються в єдиний звіт, який зберігається у форматі HTML-файлу.

```
C:\Users\Пользователь\Desktop\Uni\cli>node --no-warnings ps.js -h
Options:
      --version          Show version number                          [boolean]
  -u, --url              URL of the website to analyze       [string] [required]
  -j, --json             Output the result as JSON    [boolean] [default: false]
      --ip               Show current IP address and location (may reflect X-For
                         warded-For if present)       [boolean] [default: false]
  -w, --no-warnings      Hide experimental warnings during JSON module import
                                                      [boolean] [default: false]
  -r, --html-report      Generate an HTML report      [boolean] [default: false]
  -x, --x-forwarded-for  Set the X-Forwarded-For header with the specified IP ad
                         dress                                         [string]
  -h, --help             Show help                                    [boolean]
```

Рисунок 4.9 – Перелік параметрів

HTML-звіт, що генерується після завершення аудиту вебзастосунку, виступає як візуальна панель моніторингу – інтерактивний дашборд, який відображає всі основні метрики у структурованому та зручному для аналізу вигляді.

У верхній частині вікна (рис. 4.10) представлено три ключові показники: Performance (продуктивність), SEO та Best Practices, кожен із яких відображається у вигляді півкруглих графіків із кольоровими індикаторами, що сигналізують про рівень ефективності – від зеленого (висока якість) до червоного (критичні недоліки). Нижче розміщено Core Web Vitals, де для метрик LCP, FCP, CLS створено прогрес-бари з візуальним підсвічуванням зони належності: good, needs improvement або poor (рис 4.11).

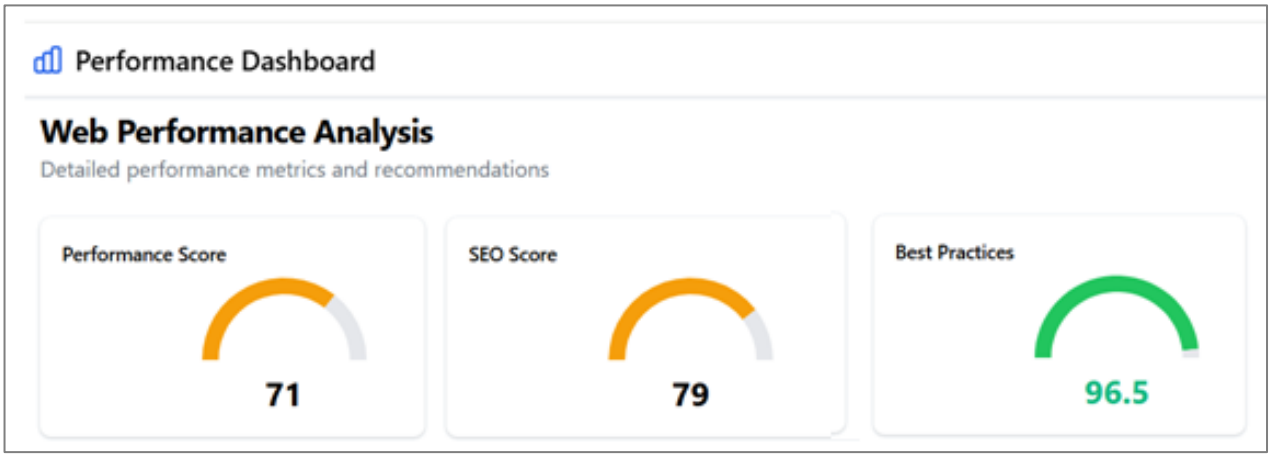


Рисунок 4.10 – Оцінка аудиту вебзастосунка

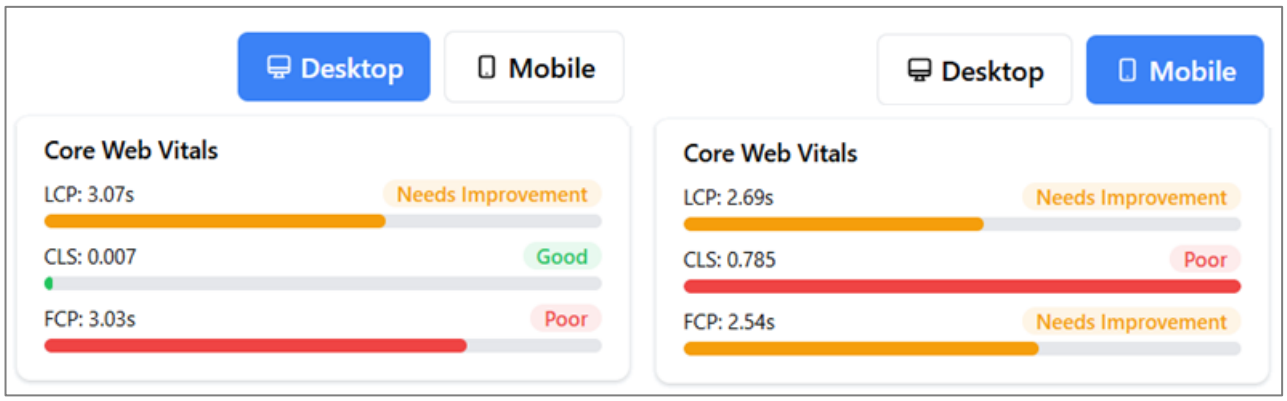


Рисунок 4.11 – Вивід метрик продуктивності (завантаження сторінки)

Окрема гістограма продуктивності демонструє час завантаження за ключовими параметрами у секундах, що дозволяє швидко оцінити загальну динаміку швидкодії (рис 4.12).

Інструмент також містить порівняльну діаграму «Desktop vs Mobile», що дає змогу миттєво зіставити рівень продуктивності між версіями сторінки для різних пристроїв (рис. 4.13). Такий підхід дозволяє виявити оптимізаційні прогалини, специфічні для конкретного середовища користувача.

Окрім графічних елементів, у вкладках нижче представлено поглиблений звіт по кожному з напрямів: рекомендації, технічні аспекти, SEO та Best Practices. Кожен пункт супроводжується кольоровими індикаторами, іконками та

деталізованими повідомленнями (рис. 4.14 – рис. 4.17), які чітко вказують, які саме елементи потребують доопрацювання.

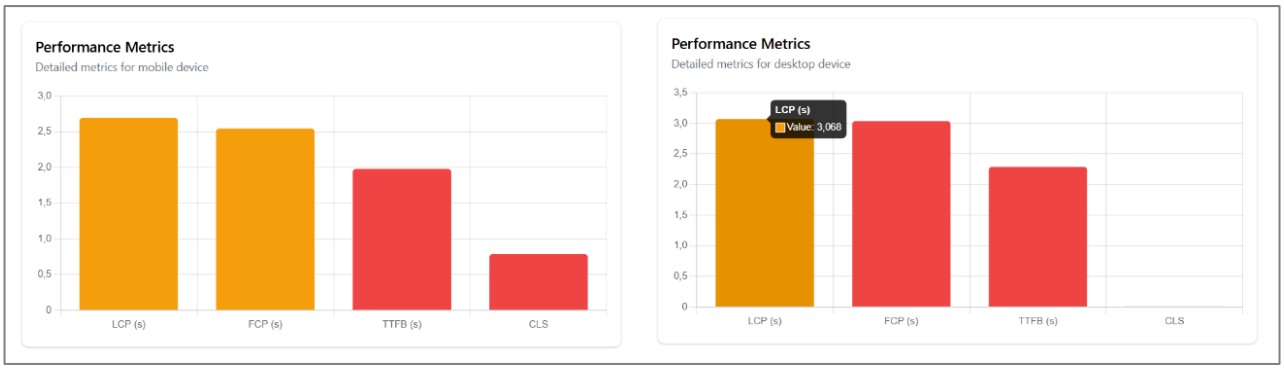


Рисунок 4.12 – Гістограма метрик продуктивності (завантаження сторінки)

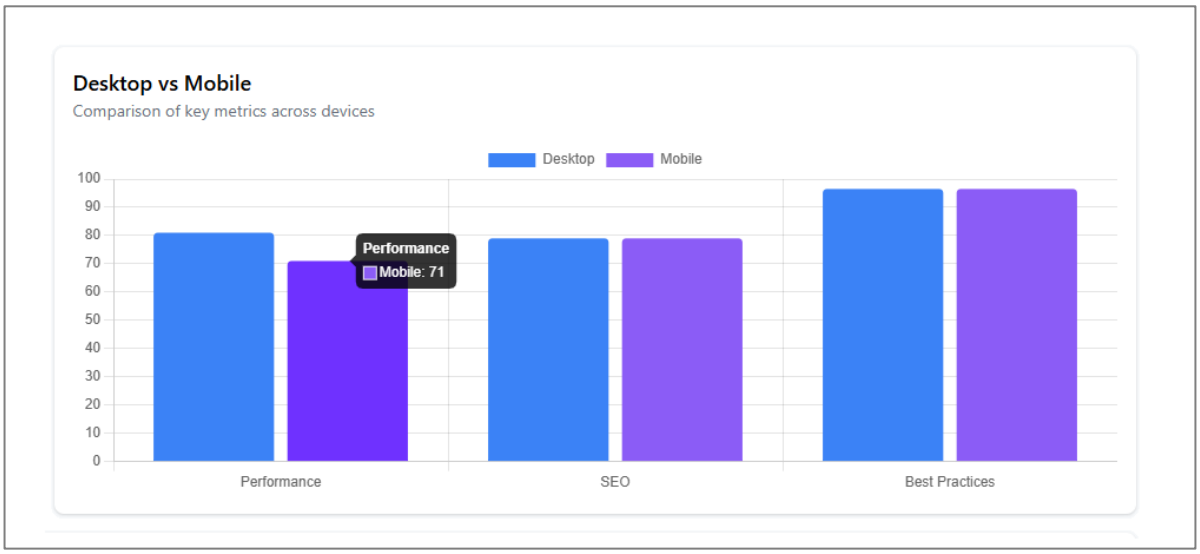


Рисунок 4.13 – Порівняльна діаграма метрик для різних типів пристроїв

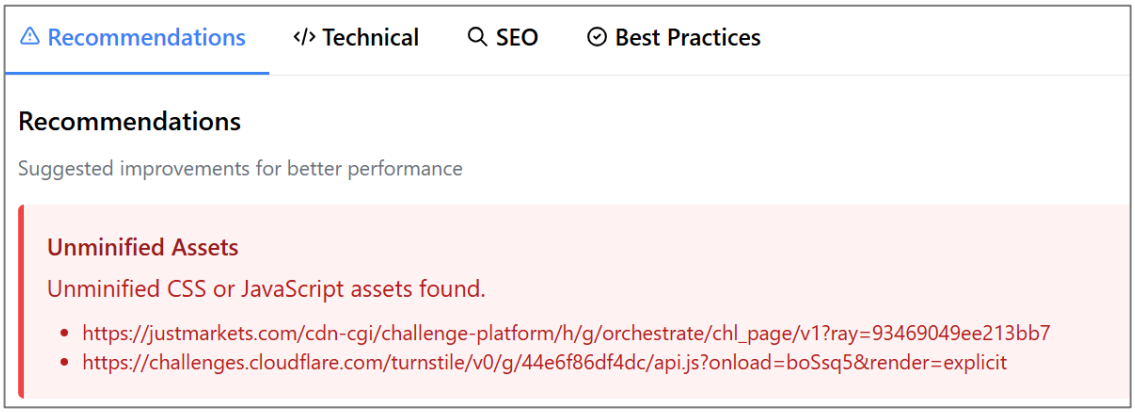


Рисунок 4.14 – Текстові рекомендації за результатом перевірок метрик

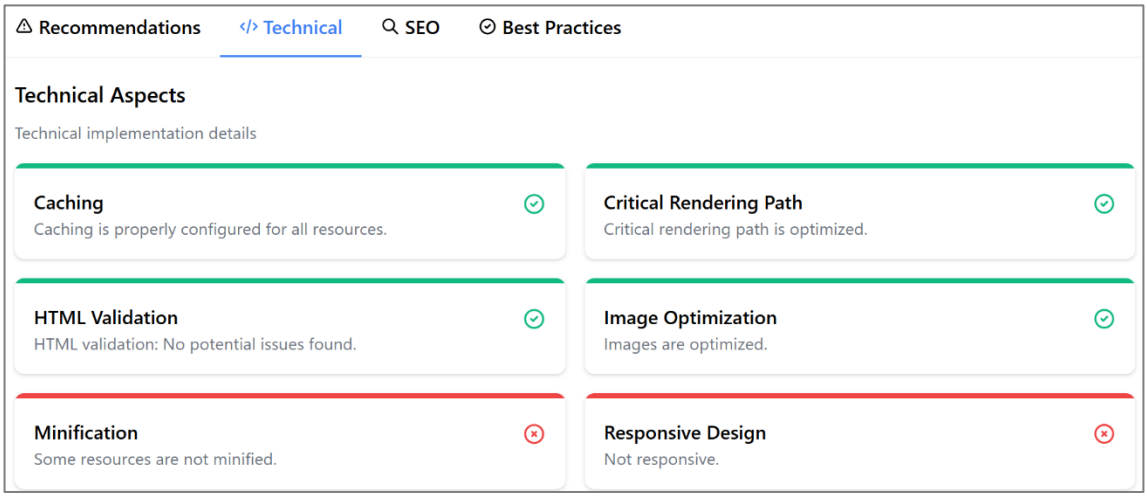


Рисунок 4.15 – Текстові рекомендації та результат технічного аналізу

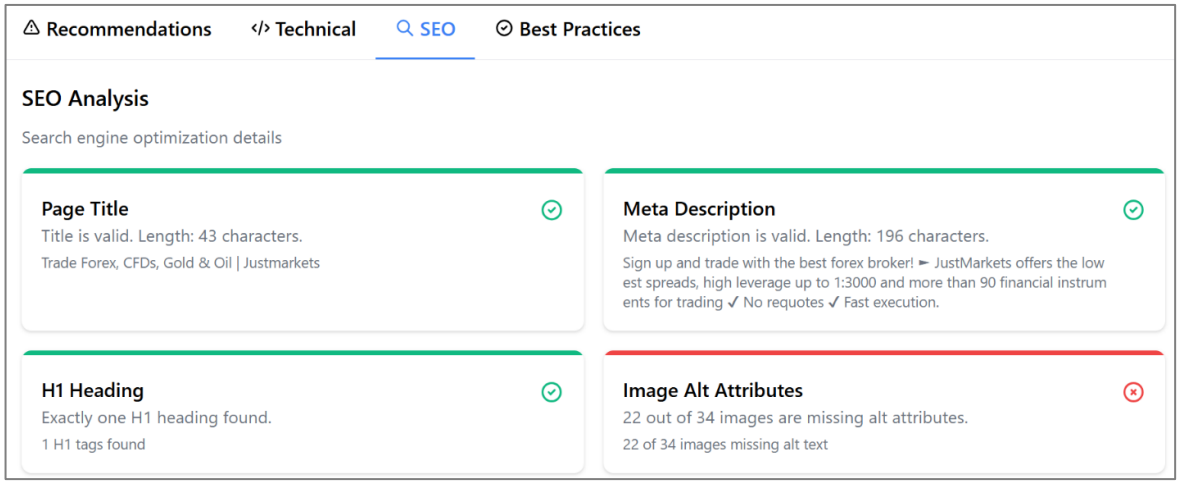


Рисунок 4.16 – Текстові рекомендації та результат SEO-аналізу

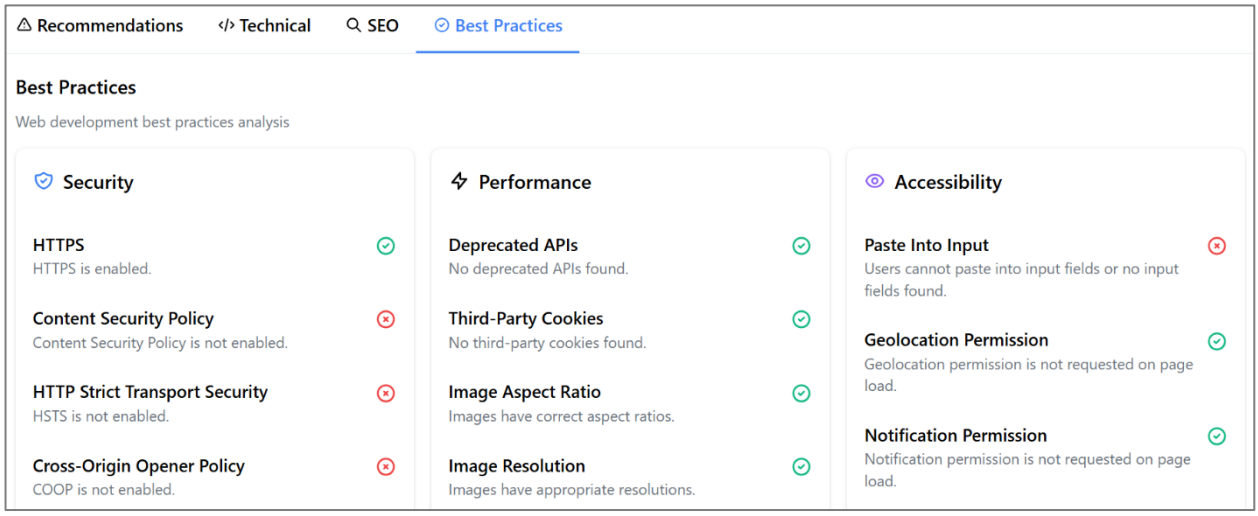


Рисунок 4.17 – Текстові рекомендації та результат аудиту Best Practices

Такий підхід подачі інформаційні HTML-репортом дозволяє створити візуалізований дашборді, для внутрішнього використання, дозволяючи не лише зафіксувати результати аудиту, але й зробити їх зрозумілими як для розробників, так і для менеджерів й простих користувачів. Це значно полегшує прийняття рішень щодо оптимізації сайту та його технічного вдосконалення. Розподіл інтерфейсів дозволяє гнучко використовувати систему як вручну, так і в автоматизованому режимі, залежно від контексту.

Після проведення детального аудиту вебресурсу https://justmarkets.com, який вже був згаданий у розділі 1.2 при огляді популярних інструментів для аналізу продуктивності, було отримали низку важливих висновків щодо його поточного стану та потенційних напрямків для оптимізації. Отримані результати Performance чітко демонструють значну різницю в продуктивності вебресурсу між десктопною та мобільною версіями. Як видно з представлених даних (рис. 4.13), показники продуктивності на десктопних пристроях значно вищі, ніж на мобільних. Це свідчить про те, що основні оптимізаційні роботи, спрямовані на покращення процесу завантаження сторінки та контенту, були переважно сфокусовані на десктопній версії.

Гістограми (рис. 4.11, 4.12) наочно ілюструють ті аспекти, які залишаються неоптимізованими та потребують особливої уваги для подальшого покращення. Ідентифікація цих вузьких місць є ключовим кроком для підвищення загальної швидкості завантаження та відгуку вебсайту на мобільних пристроях.

Що стосується SEO показників, то вони є ідентичними для обох версій ресурсу та становлять 79 балів. Це є достатньо високою оцінкою, що свідчить про хороший рівень оптимізації для пошукових систем. Однак, існують певні моменти, які залишаються неоптимізованими і потенційно можуть применшувати значимість ресурсу в пошуковій видачі. Конкретні рекомендації щодо усунення цих недоліків та подальшого покращення SEO позицій представлені на рис. 4.16. Впровадження цих рекомендацій допоможе підвищити видимість ресурсу та залучити більше органічного трафіку.

В той же час, метрики Best Practices знаходяться на високому рівні, що є позитивним показником. Це свідчить про те, що вебресурс дотримується найкращих практик у галузі веб-розробки та є достатньо оптимізованим з технічної точки зору. Високий рівень Best Practices сприяє стабільності, безпеці та зручності використання ресурсу, що є важливим фактором як для користувачів, так і для пошукових систем.

Загалом, аудит вебресурсу https://justmarkets.com виявив сильні сторони, зокрема високі показники продуктивності на десктопі та хороший рівень Best Practices, а також області, що потребують покращення, особливо в частині продуктивності мобільної версії та деяких аспектів SEO. Подальші роботи з оптимізації, спрямовані на усунення виявлених недоліків, дозволять значно підвищити загальну ефективність ресурсу, покращити користувацький досвід та збільшити його видимість у пошукових системах.

4.5 Тестування й оцінка якості системи

Тестування є невід’ємною частиною життєвого циклу розробки будь-якої програмної системи. Його головна мета це забезпечення надійності, стабільності та відповідності функціоналу заявленим вимогам. За допомогою тестування можна виявити помилки й недоліки ще на ранніх етапах розробки, що допомагає уникнути збоїв під час експлуатації системи та гарантує позитивний користувальницький досвід при взаємодії програмою [30].

Ретельно протестований застосунок легше масштабувати, підтримувати і інтегрувати з іншими рішеннями без ризику порушити логіку роботи або втратити важливі дані. Перевірка системи включала не лише стандартні робочі сценарії, але й тестування крайових ситуацій (edge case), виявлення потенційних уразливостей та аналіз можливих нестабільних або некоректно реалізованих ділянок під час отримання метрик чи генеруванні результату відпрацювання системи.

Під час реалізації інформаційної системи для аналізу та моніторингу вебзастосунків було розроблено окремий ряд сценаріїв тестування (декілька тестів з цього ряду представлено в табл. 4.5), що охоплює всі основні функціональні можливості проєкту.

Таблиця 4.5 – Основні критичні сценарії перевірки під час тестування

№	Назва	Сценарій	Очікуваний результат
1	Запуск системи	Користувач вводить всі необхідні параметри та запускає програму на виконання	Користувач отримує результат аудиту у обраному форматі виводу даних.
2	Генерація репорту	Користувач додає параметри для генерації репорту	Обраний тип репорту успішно генерується.
3	Користувач вводить невладні дані	Користувач забув додати параметр -u (URL) під час запуску аудиту	Повідомлення про помилку буде виведено в консольний застосунок.
4	Підміна локації	Користувач хоче відправити запит на проведення аудиту з іншої країни	IP змінюється через проксі, запит відсилається з іншої країни.
5	Збір даних	Під час аудиту збираються всі метрики	Метрики збираються та виводяться в консоль.
6	Валідація даних у HTML репорт	Користувач проводить аудит з параметром генерації HTML репорту	Результати аналізу підтягуються в HTML репорт.
7	Аудит для 2х і більше сторінок	Користувач вводить для аналізу 2+ сторінок	Повідомлення про помилку буде виведено в консольний застосунок.
8	Паралелізація запиту	Користувач запускає програму на аудит	Система надсилає ряд асинхронних паралельних запитів на ресурс.

У процесі тестування застосовувалися різноманітні техніки тест-дизайну, які охоплювали як позитивні сценарії (тобто ситуації, коли система правильно реагує на очікувані дії користувача), так і негативні сценарії (ситуації, в яких користувач вводить некоректні або непередбачувані дані, а система при цьому поводиться стабільно та без збоїв). Такий підхід дозволив виявити слабкі місця у логіці, непрацюючі ділянки коду, а також вразливості, які могли б вплинути на цілісність або безпеку системи.

Під час тестування було опрацьовано загалом 67 сценаріїв, що охоплювали основні функціональні можливості системи. Під час проведення тестування було виявлено 2 критичних дефекти, які серйозно впливали на працездатність системи (при генерації будь-якого звіту виникав збої через некоректний тип даних у компоненті виводу, що впливало на продуктивність; в першочерговій версії метрики Best Practise не використовували асинхронний підхід, що збільшило роботу програми на ~10 секунд), а також 5 некритичних недоліків (аналіз відбувався навіть без вказання параметра `-u`; підміна геолокації не працювала при використанні проксі; тощо) що впливали на зручність і точність роботи застосунку.

З метою забезпечення коректної роботи системи, знайдені дефекти були своєчасно проаналізовані та виправлені у процесі подальшої розробки. Це дозволило підвищити стабільність роботи системи, її здатність обробляти дії користувачів та зменшити ризики виникнення збоїв у майбутньому.

Процес тестування демонструє, що саме ретельна перевірка відповідає за валідацію якості продукту і є ключовим фактором у створенні надійної та ефективної інформаційної системи. В результаті систематичне тестування допомагає не тільки виявити та усунути дефекти, але й закладає основу для подальшого вдосконалення і розвитку програмного забезпечення.

Підтвердженням високої якості та значної конкурентоспроможності розробленого застосунку для аналізу та моніторингу продуктивності вебзастосунків є результати порівняльного аналізу з існуючими програмними засобами, представлені в табл. 4.6. Цей аналіз демонструє, що розроблений

інструмент не лише відповідає, але й перевершує функціональність багатьох популярних рішень, які представлені на ринку.

Як видно з Таблиці 4.6, CLI-інструмент (останній стовпець) забезпечує повне покриття всіх ключових метрик, які є критично важливими для комплексного аналізу продуктивності сучасних вебрішень. На відміну від багатьох конкурентів, які мають обмеження у вимірюванні SEO, Best Practices, або наданні конкретних рекомендацій (Suggestions), застосунок охоплює всі ці аспекти. Проведений аналіз чітко показує, що розроблений інструмент задовольняє та покриває всі вимоги в сучасному цифровому світі веб-рішень. Він надає повний спектр даних, необхідних для глибокого розуміння стану веб-ресурсу, від технічної продуктивності до оптимізації для пошукових систем та дотримання найкращих практик.

Таблиця 4.6 – Програмні засоби для аналізу та моніторингу продуктивності вебзастосунків

Програмний засіб	**SE Ranking**	**Web Page Test**	**PageSpeed Insights**	**GTmetrix**	**Pingdom Tools**	**CLI**
Performance	+	+	+	+	+	+
SEO	+	–	+	–	–	+
Best Practise	–	+	+	–	–	+
Suggestions	–	–	+	–	–	+
GEO runs	–	+ (обмежено)	–	+ (обмежено)	+ (обмежено)	+ (без обмежень)
3rd API for run	+	+	own API	+	own API	own API
Free or Paid	Free plan, but paid	Free plan, but paid	Free	Free plan, but paid	Free plan, but paid	Free
Audit report	–	+	+	+	+	+
Час аналізу, *секунд*	~20	~35-40	~20	~25	~10-15	~15

Особливо варто відзначити, що консольний застосунок забезпечує можливість запуску аналізу з різних географічних локацій без будь-яких обмежень, на відміну від багатьох комерційних рішень, де ця функція часто обмежена або доступна лише за додаткову плату. Важливим аспектом є також доступність рішення. На відміну від багатьох конкурентів, які пропонують лише обмежений безкоштовний план або повністю платний доступ, інструмент є повністю безкоштовним, що робить його доступним для широкого кола користувачів, незалежно від їхнього бюджету.

Незважаючи на широкий спектр функціональності, час аналізу, який становить приблизно 15 секунд, є цілком конкурентним і не поступається багатьом іншим інструментам. Крім того, інструмент надає детальний звіт про аудит, що дозволяє користувачам легко інтерпретувати отримані результати та планувати подальші дії.

З рисунку 4.18, який підсумовує один із ключових показників таблиці 4.7, а саме «час виконання» (час виконання аудиту від відправлення запиту до отримання результату), видно значну перевагу розробленого консольного застосунку.

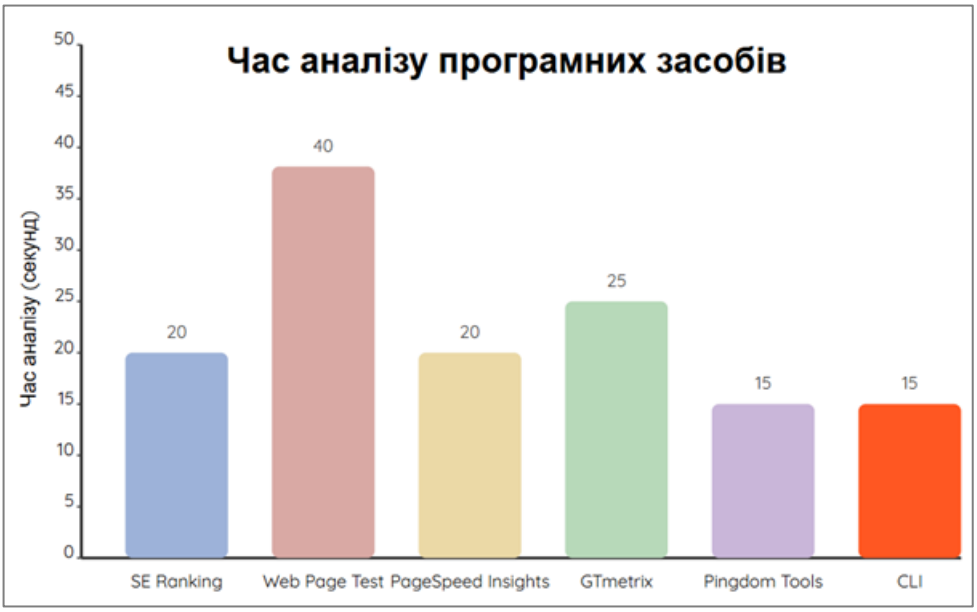


Рисунок 4.18 – Час аналізу різних програмних засобів для аудиту вебсайтів

Проведений аналіз демонструє, наскільки сторонні рішення, незважаючи на свою простоту та меншу функціональність, поступаються більш функціонально насиченій, продуманій та повноцінній, але при цьому швидкій та стабільній інформаційній системі.

Як видно з таблиці 4.6, прямим конкурентом розробленого консольного застосунку є інструмент від Google – PageSpeed Insights. З метою оцінки якості системи проведено порівняльний аналіз між створеною інформаційною системою та вебінструментом PageSpeed Insights. Основною метою такого порівняння є визначення відповідності отриманих показників продуктивності, стабільності та оптимізації вебзастосунку з метриками, що пропонує PageSpeed Insights. Для аналізу обрано вебресурс https://justmarkets.com, а оцінювання проводилося як для мобільної, так і для десктопної версії, що дозволяє забезпечити комплексний підхід до порівняння.

За отриманими результатами (рис. 4.19, рис. 4.20) можна побачити, що показники розробленої інформаційної системи є наближеними до продукту від Google, що свідчить про високу точність і коректність оцінки. Незначні відмінності у бік зниження в деяких метриках консольного застосунку пояснюються частково відсутністю оновлення новітніх показників та індикаторів в PageSpeed Insights, відповідно до останніх трендів та стандартів продуктивності вебзастосунків.

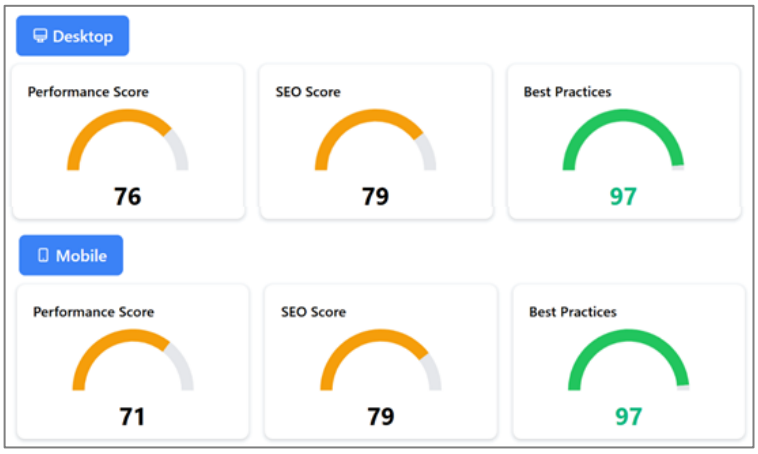


Рисунок 4.19 – Результати аудиту консольної ІС для https://justmarkets.com

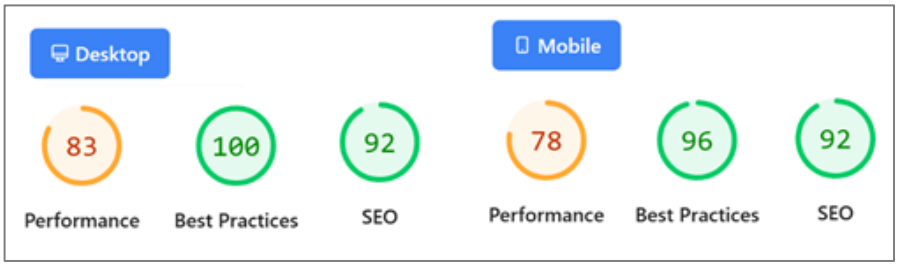


Рисунок 4.20 – Результати аудиту PageSpeed Insight для https://justmarkets.com

Така ж картина порівняльного аналізу представлена у таблицях 4.7. З таблиці порівняння швидкості завантаження (табл 4.7) для типу пристроїв Mobile можна побачити, що показник CLS (Cumulative Layout Shift), який має значний вплив на загальну оцінку продуктивності, дещо знизив остаточний бал Performance аудиту інформаційної системи аналізу та моніторингу web-застосунків. Це чітко простежується при порівнянні результатів на рис. 4.19 та 4.20.

Таблиця 4.7 (а) – Порівняльний аналіз показників завантаження сторінки (Desktop)

Показник / Інструмент	**PageSpeed Insight**	**Консольний застосунок**
LCP (секунд)	4,7	4,6
FCP (секунд)	4,5	4,1
TTFB (секунд)	3,4	2,5
CLS (показник зміщення)	0,09	0,004

Таблиця 4.7 (б) – Порівняльний аналіз показників завантаження сторінки (Mobile)

Показник / Інструмент	**PageSpeed Insight**	**Консольний застосунок**
LCP (секунд)	4,0	2,9
FCP (секунд)	1,8	2,6
TTFB (секунд)	2,0	1,9
CLS (показник зміщення)	0,1	0,8

Така ситуація вказує на необхідність глибшого аналізу роботи Puppeteer як інструменту для збирання метрик, адже точність та коректність вимірювань безпосередньо впливають на достовірність отриманих результатів.

Попри це, отримані показники є задовільними, що свідчить про загальну коректність функціонування розробленої інформаційної системи. Зокрема, кінцеві рекомендації, сформовані на основі проведеного аудиту, є чіткими, структурованими та спрямованими на підвищення рівня продуктивності вебзастосунку, що підтверджує доцільність використання ІС у практичному середовищі.

Таким чином, порівняльний аналіз однозначно підтверджує, що розроблений застосунок є висококонкурентним рішенням, яке пропонує розширений функціонал, гнучкість, доступність та ефективність, що робить його цінним інструментом для тих, хто займається моніторингом вебсайтів.

Висновки до розділу 4

У четвертому було детально розглянуто процес розробки, налаштування та реалізації інформаційної системи для аналізу та моніторингу продуктивності web-застосунків. Особлива увага приділялася компонентній архітектурі, що забезпечує гнучкість, масштабованість і можливість повторного використання модулів системи. Розроблені модулі охоплюють різні аспекти аналізу, такі як SEO, продуктивність, Best Practise, технічна складова та рекомендації, що дозволяє проводити комплексний аудит вебресурсів у автоматичному режимі.

Значну роль у реалізації системи відіграє бібліотека Puppeteer, яка забезпечує автоматизацію браузера для збору необхідних метрик у двох режимах – десктопному та мобільному. В процесі налаштування враховувалися важливі параметри запуску браузера, зокрема, режим headless, налаштування безпеки та емулювання різних пристроїв через user-agent і параметри екрана. Це дозволяє

отримувати більш точні та релевантні дані, що враховують поведінку користувачів на різних платформах.

У рамках системи реалізовано інтуїтивно зрозумілий інтерфейс, який підтримує як консольний режим для швидкого аналізу, так і генерацію HTML-звітів для візуалізації результатів. Такий підхід забезпечує зручність використання як для технічних фахівців, так і для менеджерів або кінцевих користувачів.

Велику увагу приділено процесу тестування, включаючи позитивні та негативні випадки. Це сприяє виявленню та усуненню критичних і некритичних дефектів, що підвищує надійність та стабільність системи. Проведене тестування підтвердило важливість систематичної перевірки для забезпечення високої якості та відповідності системи поставленим вимогам.

Загалом, розділ присвячений розробці та тестуванню продемонстрував цілісність та ефективність обраного підходу до створення інформаційної системи для аналізу та моніторингу продуктивності web-застосунків, що є гнучкою, масштабованою та зручною у використанні, з високим потенціалом для подальшого розвитку та вдосконалення у форматі Open Source.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи створено інформаційну систему для аналізу та моніторингу продуктивності web-застосунків, що дозволяє комплексно оцінювати ключові показники швидкодії, стабільності та відповідності сучасним стандартам. Розроблена система забезпечує централізоване збирання метрик продуктивності та генерацію рекомендацій для покращення.

Для досягнення поставленої мети виконано наступні завдання:

– проведено аналіз предметної області та сучасних програмних рішень у сфері аудиту продуктивності вебзастосунків;

– обґрунтовано вибір технологій та методів аналізу й моніторингу вебресурсів, зокрема вибрано стек на основі JavaScript, Node.js та Puppeteer;

– розроблено архітектуру системи з урахуванням потреб гнучкості та масштабованості, що дозволяє інтеграцію з CI/CD процесами;

– реалізовано програмну систему у вигляді CLI-застосунку для аудиту продуктивності вебзастосунків з підтримкою формування звітів у форматах HTML, JSON та консольного виведення;

– виконане тестування розробленої системи на реальних вебресурсах для підтвердження її працездатності, ефективності та точності оцінок.

Проведене дослідження в рамках APM систем показало насиченість ринку різноманітними, різносортними рішеннями. Проте, багато з них мають спільні риси та взаємозалежності, часто покладаючись на синтетичні методи аналізу та тестування ресурсів.

Аналіз існуючих технологічних рішень підтвердив доцільність використання програмної мови JavaScript. Її безпосередня інтерпретація браузером та наявність однобічного API, зрозумілого браузером, забезпечує просту, швидку та ефективну взаємодію. Відповідно до цього, програмна реалізація консольного застосунку була виконана на мові JavaScript у середовищі VS Code, що забезпечило високу адаптивність, легке використання та інтеграцію з вже вбудованими пакетами для

розробки. Інтеграція JavaScript стала ключовим зв'язуючим рішенням між бібліотекою Node.js, що обробляє серверну логіку, та бібліотекою Puppeteer, яка забезпечила стійку роботу з браузером для валідації та збору необхідних метрик для подальшого аналізу.

У розробленому консольному застосунку для оцінки якості продуктивності реалізовано використання трьох основних моделей метрик: Performance, SEO та Best Practices. Для кожної групи метрик розроблено окремі функції, що виконують всебічний аналіз отриманих результатів та видають інтегровану оцінку за 100-бальною шкалою, де 100 є найвищим показником. Окрім ортодоксального збору критично важливих метрик, значну увагу було приділено безпосередньо технічним особливостям вебзастосунків. Проведене тестування значно підвищило стабільність, надійність та точність роботи застосунку.

Результати проведеного дослідження різносторонніх вебзастосунків для виявлення продуктивності надали критичне розуміння того, якою система не повинна бути в моменті видачі аудиту. Завдяки глибокому всебічному огляду, в подальшій реалізації кінцевого рішення було запропоновано три рішення для виводу результату проведеного аудиту інформаційної системи: консольний вивід, структурований JSON-об'єкт та інтерактивний HTML-звіт. Такий різноманітний підхід до представлення результатів розширює можливості валідації та результативності аналізу, дозволяючи користувачам обирати найбільш зручний формат залежно від їхніх потреб – від швидкої оцінки до детального технічного аналізу та інтеграції в автоматизовані процеси.

У результаті, розроблений консольний застосунок демонструє здатність значно спростити та систематизувати процес збору критично важливих показників web-ресурсів. Його конкурентною перевагою та унікальним місцем в ієрархії рішень є модель розповсюдження, оскільки застосунок розроблений за методологією Open Source та розповсюджується безоплатно. Це відкриває нові перспективи для оптимізації аудиторських рішень як частин APM систем, сприяючи підвищенню рівня клієнтського обслуговування, одночасно підвищуючи

бізнес-показники та задоволеність клієнтів за рахунок покращення якості та швидкості вебзастосунків.

Поставлені завдання виконано, однак у подальшому функціонал системи може бути розширено за рахунок додавання нових модулів для більш детального аналізу специфічних аспектів вебзастосунків, зокрема модулів для моніторингу безпеки (виявлення вразливостей, тестування на стійкість до DDoS-атак), а також для оцінки доступності (Accessibility Score) відповідно до стандартів WCAG.

Ключовим покращенням системи може стати впровадження окремого модуля прогнозної аналітики – використання алгоритмів машинного навчання для передбачення потенційних проблем або збоїв у системі на основі історичних даних. Такий модуль дозволить проактивно виявляти вузькі місця та запобігати критичним ситуаціям ще до їх виникнення, що значно підвищить стабільність та надійність роботи вебзастосунків. Додатково створивши функціонал для автоматичних рекомендацій за допомоги аналізу штучного інтелекту шляхом створення системи автоматизованих пропозицій щодо покращення на основі зібраних даних. Це дозволить розробникам оперативно реагувати на виявлені недоліки та впроваджувати необхідні оптимізації без затримок

Розширення функціоналу також може охоплювати підтримку різних форматів звітів (наприклад, PDF або інтеграція з базами даних для збереження історії аудиту), а також можливість порівняння звітів між собою для відстеження динаміки змін у продуктивності після оновлень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. McMillan S. Performance.Now() 2023 Debrief. *Conference conclusions*, 2023. Retrieved from: stuart-mcmillan.com/blog/webperf/perfnow2023.html (дата звернення: 20.04.2025)

2. Analyzing the impact of website speed on user engagement, 2024. URL: uptime.com/blog/analyzing-the-impact-of-website-speed-on-user-engagement (дата звернення: 20.04.2025)

3. Vessum S. Amazon PageSpeed Case Study. URL: conductor.com/academy/page-speed-resources/faq/amazon-page-speed-study/ (дата звернення: 21.04.2025)

4. MDN. Web performance. URL: https://developer.mozilla.org/en-US/docs/Web/Performance (дата звернення: 21.04.2025)

5. Heger C., Hoorn A., Mann M., Okanovic D. Application Performance Management: State of the Art and Challenges for the Future. *Proceedings of the International Conference on Software Engineering (ICSE)*, 2017 – 429-432 pp. Retrieved from: https://doi.org/10.1145/3030207.3053674

6. Ji S., Li Q., Cao W., Ko C. Quality Assurance Technologies of Big Data Applications. *Proceedings of the International Conference on Big Data*, 2020. Retrieved from: https://doi.org/10.3390/app10228052

7. Xilogianni, C., Doukas, F., & Drivas, I. Speed Matters: What to prioritize in Optimization for Faster Websites. *Analytics*, 1(2), 2022 – 175-192 pp. Retrieved from: https://doi.org/10.3390/analytics1020012

8. Ale, N.K. Integration Performance Testing into CI/CD Pipelines for Test Automation. *Journal of Scientific and Engineering Research*, 7(6), 2020 – 273-278 pp. Retrieved from: jsaer.com/download/vol-7-iss-6-2020/JSAER2020-7-6-272-278.pdf

9. Netcraft: February 2025 Web Server Survey. URL: netcraft.com/blog/february-2025-web-server-survey (дата звернення: 22.04.2025)

10. Coursaris, C., Osch, W., Lores-Nicolas, C., Molina-Castillo F., & Rapp, N. Driving Website performance using Web Analytics: A Case Study. *In Proceedings of the 19th Americas Conference on Information Systems*, 2013.

11. Bansal D. How SEO Makes Website Loads Faster and Helps in User Engagement. *Journal of Digital Marketing, Vol. 6, Issue 2*, 2024 – 11-13, 15-20 pp.

12. Google Documentation. Understanding Core Web Vitals and Google search result. Retrieved from: developers.google.com/search/docs/appearance/core-web-vitals

13. Kravchenko Y., Leshchenko O., Trush O. Optimizing and Improvement a Web Application Using Open-Source Tools. In *Proceedings of the International Conference on Web Technologies, Vol. 3624, paper 30*, 2023 – 374-378 pp.

14. Ah Zau Marang. Analysis of web performance optimization and its impact on user experience. *Thesis*, 2018. – 19 p.

15. Munyaradzi Z., Maxmillian G., Amanda M. Effects of Web Page Contents on Load Time over the Internet. *International Journal of Science and Research (IJSR) Vol. 2, Issue 9*, 2013. – 77-78 pp.

16. Google Lighthouse documentation. Performance Audit: Speed Index. URL: https://developer.chrome.com/docs/lighthouse/performance/speed-index (дата звернення: 24.04.2025)

17. Ranjpour R. The Significant Role of SEO in Effective Web Marketing. *The 2024 Association of Marketing Theory and Practice Proceedings, 23*, 2024. – 14-15 pp.

18. Poturak M., Keco D. Influence of Search Engine Optimization on business performance. *International Journal of Research in Business and Social Science (2147-4478), Center for the Strategic Studies in Business and Finance, vol. 11(4)*, 2022. – 63 p.

19. Vepsalainie J., Hellas A., Vuorimaa P. Overview of Web Application Performance Optimization Techniques, *Web Information Systems and Technologies* 2024. DOI: 10.1007/978-3-031-89621-7_3 (дата звернення 26.04.2024)

20. Manhas, J. A Study of Factors Affecting Websites Page Loading Speed for Efficient Web Performance. *International Journal of Creative Research Thoughts (IJCRT), E-ISSN: 2347-2693, Vol. 1, Issue 3*, 2013. – 33-35 pp.

21. Chhetri N. A Comparative Analysis of Node.js (Server-Side JavaScript). *Culminating Projects in Computer Science and Information Technology. 5*, 2016. – 28 p.

22. Puppeteer library documentation. URL: https://pptr.dev/guides/javascript-execution (дата звернення: 27.04.2025)

23. Zenrows. How to do Web Scrapping with Puppeteer and NodeJS, 2023. URL: zenrows.com/blog/puppeteer-web-scraping (дата звернення: 29.04.2025)

24. Kusuma R. Puppeteer args in launch() method, 2024. URL: hayageek.com/puppeteer-args-in-launch-method/ (дата звернення: 29.04.2025)

25. Senol A., Acar G. Unveiling the Impact of User-Agent Reduction and Client Hints: A Measurement Study. *WPES '23: Proceedings of the 22nd Workshop on Privacy in the Electronic Society*, 2023 – 4-19 slides.

26. Luo X., Zheng X., Liao C. Research on the Modular Design Method and Application of Prefabricated Components Based, *Journal Buildings, № 12, p. 2980,* 2024.

27. OSG. Starting an Open-Source Project Guideline. URL: https://opensource.guide/starting-a-project/ (дата звернення: 29.04.2025)

28. Mustafic A. Modular Application Architecture – Inheritance, 2017. URL: goetas.com/blog/modular-application-architecture-inheritance (дата звернення: 29.04.2025)

29. Almasi S., Ahmadi H., Sohrabei S. Usability Evaluation of Dashboards: A Systematic Literature Review of Tools. *Biomed research international*, 2023 – 5-6 pp.

30. ISTQB Glossary. Test process. URL: https://glossary.istqb.org/en_US/term/test-process (дата звернення: 30.04.2025)

31. Chrome UX Reporter (CrUX methodology). URL: developer.chrome.com/docs/crux/methodology (дата звернення: 30.04.2025)

32. Allison R., Hayes, C. Comprehensive framework to evaluate websites: literature review and development of GoodWeb. *Journal of Medical Internet Research (JMIR), Vol. 3, No 4,* 2019. – 3 p.

33. Kumar N., Kumar S., Rajak R. Website Performance Analysis and Evaluation using Automated Tools. In *5th International Conference on Electrical, Electronics, Commutations, Computer Technologies & Optimization Techniques*, 2021 – 210-214 pp.

34. Gangurde R., Kumar B. Web page prediction using genetic algorithm and regression based on web content features. *International Conference on Electronics and Sustainable Communication Systems (ICESC),* 2020. – 68-74 pp.

35. Najadat H., Alodibat S. A review of website evaluation using web diagnostic tools and data envelopment analysis. *Bulletin of Electrical engineering and informatics (BEEI), Vol. 10, No 1*, 2021. – 258-265 p.

36. Drivas I., Sakas D., Giannakopoulos G. Big data analytics for search engine optimization. *Big Data Cognitive Computing (BDCC), 4(2), 5*, 2020;

ДОДАТОК А

JavaScript-скрипт для розрахунку метрик SEO

```
// audits/seo/seo_score.js

import { checkTitle } from './seo_title.js';
import { checkDescription } from './seo_description.js';
import { checkH1 } from './seo_h1.js';
import { checkImageAlt } from './seo_image_alt.js';
import { checkCanonical } from './seo_canonical.js';
import { checkIsCrawlable } from './seo_is-crawlable.js';
import { checkRobots } from './seo_robots.js';

/**
 * Calculates the overall SEO score based on various SEO checks.
 * @param {import('puppeteer').Page} page - Puppeteer page object.
 * @param {string} url - The URL of the page being analyzed.
 * @returns {Promise<{score: number, results: object}>} - SEO score and detailed results.
 */
export async function calculateSeoScore(page, url) {
   try {
      const titleResult = await checkTitle(page);
      const descriptionResult = await checkDescription(page);
      const h1Result = await checkH1(page);
      const imageAltResult = await checkImageAlt(page);
      const canonicalResult = await checkCanonical(page);
      const isCrawlableResult = await checkIsCrawlable(page, url);

      const seoResults = {
         title: titleResult,
         description: descriptionResult,
         h1: h1Result,
         imageAlt: imageAltResult,
         canonical: canonicalResult,
         isCrawlable: isCrawlableResult,
      };

      const weights = {
         title: 15,
        description: 15,
         h1: 10,
         imageAlt: 10,
         canonical: 10,
         isCrawlable: 10,
      };

      let weightedScore = 0;

      weightedScore += weights.title * (titleResult.isValid ? 1 : titleResult.message ? 0.5 : 0);
      weightedScore += weights.description * (descriptionResult.isValid ? 1 : descriptionResult.message ? 0.5 : 0);
      weightedScore += weights.h1 * (h1Result.isValid ? 1 : h1Result.message ? 0.5 : 0);
      weightedScore += weights.imageAlt * (imageAltResult.isValid ? 1 : imageAltResult.message ? 0.5 : 0);
      weightedScore += weights.canonical * (canonicalResult.isValid ? 1 : canonicalResult.message ? 0.5 : 0);
      weightedScore += weights.isCrawlable * (isCrawlableResult.isCrawlable ? 1 : 0);

      const totalWeight = Object.values(weights).reduce((a, b) => a + b, 0);
      const score = (weightedScore / totalWeight) * 100;
```

```
 return {
            score: Math.round(score),
            results: seoResults,
        };
    } catch (error) {
        console.error('Error calculating SEO score:', error);
        return {
            score: 0,
            results: {},
        };
    }
}
```

ДОДАТОК Б

JavaScript-скрипт для розрахунку метрик Performance

```
// audits/performance/performance_score.js

/**
 * Calculates the overall performance score based on individual metrics with green, yellow, and red zones.
 * @param {object} metrics - Object containing LCP, INP, CLS, FCP, and TTFB metrics.
 * @returns {Promise<number>} - The calculated performance score.
 */
export async function calculatePerformanceScore(metrics) {
  try {
    const { lcp, inp, cls, fcp, ttfb } = metrics;
    let score = 100;

    // Define thresholds for green, yellow, and red zones
    const lcpThresholds = { green: 2500, yellow: 4000, red: 5000 };
    const inpThresholds = { green: 200, yellow: 250, red: 260 };
    const clsThresholds = { green: 0.1, yellow: 0.25, red: 0.4 };
    const fcpThresholds = { green: 1000, yellow: 3000, red: 4500 };
    const ttfbThresholds = { green: 600, yellow: 1800, red: 3000 };

    const applyScoreDeduction = (value, thresholds, deductions) => {
      if (value > thresholds.red) {
        score -= deductions.red;
      } else if (value > thresholds.yellow) {
        score -= deductions.yellow;
      } else if (value > thresholds.green) {
        score -= deductions.green;
      }
    };

    applyScoreDeduction(lcp, lcpThresholds, { green: 3, yellow: 8, red: 15 });
    applyScoreDeduction(inp, inpThresholds, { green: 3, yellow: 8, red: 15 });
    applyScoreDeduction(cls, clsThresholds, { green: 3, yellow: 8, red: 15 });
    applyScoreDeduction(fcp, fcpThresholds, { green: 3, yellow: 8, red: 10 });
    applyScoreDeduction(ttfb, ttfbThresholds, { green: 3, yellow: 8, red: 10 });

    return Math.max(0, score);
  } catch (error) {
    console.error('Error calculating performance score:', error);
    return 0;
  }
}
```

ДОДАТОК В

JavaScript-скрипт для оцінки Best Practise

```
// audits/best-practice/bp-score.js

/**
 * Calculates the best practices score with weighted checks, starting from 100.
 * @param {object} bpResults - Best practices audit results.
 * @param {object} weights - Weights for each audit.
 * @param {object} thresholds - Thresholds for each audit (optional).
 * @returns {number} - Best practices score.
 */
export function calculateBestPracticesScore(bpResults, weights = {}, thresholds = {}) {
  let score = 100; // Start with 100%
  let totalWeight = 0;

  for (const category in bpResults) {
    for (const audit in bpResults[category]) {
      const result = bpResults[category][audit];
      const weight = weights[audit] || 1;
      totalWeight += weight;

      if (typeof result.isValid === 'boolean') {
        if (!result.isValid) {
          if (result.message) {
            score -= weight * 0.5; // Deduct for warnings
          } else {
            score -= weight; // Deduct for errors
          }
        }
      } else if (typeof result.isValid === 'number') {
        // Apply thresholds if provided
        if (thresholds[audit]) {
          const { green, yellow, red } = thresholds[audit];
          if (result.isValid > red) {
            score -= weight; // Deduct full weight for red zone
          } else if (result.isValid > yellow) {
            score -= weight * 0.5; // Deduct half weight for yellow zone
          } else if (result.isValid < green) {
            score -= weight * (1 - (result.isValid / green));
          }
        } else {
          score -= weight * (1 - (result.isValid / 100));
        }
      }
    }
  }

  return Math.max(0, score); // Ensure score is not negative
}
```

ДОДАТОК Г

JavaScript-скрипт для запуску аудиту

```
import yargs from 'yargs';
import { hideBin } from 'yargs/helpers';
import { analyzePageData } from './helper/analyzePageData.js';
import { getIpInfo } from './helper/helper.js';
import fs from 'fs';
import path from 'path';
import { fileURLToPath } from 'url';
import { dirname } from 'path';
import axios from 'axios';

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);

const argv = yargs(hideBin(process.argv))
  .option('url', {
    alias: 'u',
    description: 'URL of the website to analyze',
    type: 'string',
    demandOption: true,
  })
  .option('json', {
    alias: 'j',
    description: 'Output the result as JSON',
    type: 'boolean',
    default: false,
  })
  .option('ip', {
    description: 'Show current IP address and location (may reflect X-Forwarded-For if present)',
    type: 'boolean',
    default: false,
  })
  .option('no-warnings', {
    alias: 'w',
    description: 'Hide experimental warnings during JSON module import',
    type: 'boolean',
    default: false,
  })
  .option('html-report', {
    alias: 'html',
    description: 'Generate an HTML report',
    type: 'boolean',
    default: false,
  })
  .option('x-forwarded-for', {    // run: node ps5.js -u ... -xff (or --x-forwarded-for) <country_ip_address>
    alias: 'xff',
    type: 'string',
    description: 'Set the X-Forwarded-For header with the specified IP address',
  })
  .help()
  .alias('help', 'h')
  .argv;

async function analyzePageDataWithHeaders(url, device, headers = {}) {
  try {
    // Assuming your analyzePageData function can accept an optional headers object
```

```
        const result = await analyzePageData(url, device, { headers });
        return result;
    } catch (error) {
        console.error(`Error analyzing ${url} for ${device}:`, error.message);
        return {};
    }
}

export async function main() {
    if (argv.ip) {
        const ipInfo = await getIpInfo();
        console.log(`Current IP: ${ipInfo.ip}, Country: ${ipInfo.geoData.country}`);
        return;
    }

    const headers = {};
    if (argv['x-forwarded-for']) {
        headers['X-Forwarded-For'] = argv['x-forwarded-for'];
        console.log(`Setting X-Forwarded-For header to: ${argv['x-forwarded-for']}`);
    }

    const desktopResult = await analyzePageDataWithHeaders(argv.url, 'desktop', headers);
    const mobileResult = await analyzePageDataWithHeaders(argv.url, 'mobile', headers);

    const combinedResult = {
        url: argv.url,
        desktop: desktopResult,
        mobile: mobileResult,
    };

    if (argv.json) {
        console.log(JSON.stringify(combinedResult, null, 2));
    } else {
        console.log('\nDesktop Analysis:');
        if (desktopResult && desktopResult.performanceMetrics) {
            console.log(`Performance Score: ${desktopResult.performanceMetrics.score}`);
            console.log(`LCP: ${desktopResult.performanceMetrics.lcp?.lcp} ms`);
            console.log(`INP: ${desktopResult.performanceMetrics.inp?.inp} ms`);
            console.log(`CLS: ${desktopResult.performanceMetrics.cls?.cls}`);
            console.log(`FCP: ${desktopResult.performanceMetrics.fcp?.fcp} ms`);
            console.log(`TTFB: ${desktopResult.performanceMetrics.ttfb?.ttfb} ms`);
            console.log('SEO Score:', desktopResult.seoScore?.score);
            console.log('Technical Aspects:', desktopResult.technicalAspects);
            console.log('Best Practices Score:', desktopResult.bestPractices?.score);
            console.log('Recommendations:');
            Object.entries(desktopResult.recommendations || {}).forEach(([key, value]) => {
                console.log(`- ${key}: ${value?.message}`);
                (value?.items || []).forEach(item => console.log(`  - ${item}`));
            });
        } else {
            console.log('Desktop analysis data is not available.');
        }

        console.log('\nMobile Analysis:');
        if (mobileResult && mobileResult.performanceMetrics) {
            console.log(`Performance Score: ${mobileResult.performanceMetrics.score}`);
            console.log(`LCP: ${mobileResult.performanceMetrics.lcp?.lcp} ms`);
            console.log(`INP: ${mobileResult.performanceMetrics.inp?.inp} ms`);
            console.log(`CLS: ${mobileResult.performanceMetrics.cls?.cls}`);
            console.log(`FCP: ${mobileResult.performanceMetrics.fcp?.fcp} ms`);
```

```
            console.log(`TTFB: ${mobileResult.performanceMetrics.ttfb?.ttfb} ms`);
            console.log('SEO Score:', mobileResult.seoScore?.score);
            console.log('Technical Aspects:', mobileResult.technicalAspects);
            console.log('Best Practices Score:', mobileResult.bestPractices?.score);
            console.log('Recommendations:');
            Object.entries(mobileResult.recommendations || {}).forEach(([key, value]) => {
                console.log(`- ${key}: ${value?.message}`);
                (value?.items || []).forEach(item => console.log(`  - ${item}`));
            });
        } else {
            console.log('Mobile analysis data is not available.');
        }
    }

    if (argv['html-report']) {
        console.log('Generating HTML report...');

        const templatePath = path.join(__dirname, 'performance-dashboard.html');
        let htmlTemplate;

        try {
            htmlTemplate = fs.readFileSync(templatePath, 'utf8');
        } catch (error) {
            console.log('HTML template not found. Creating template file...');
            const defaultTemplatePath = path.join(__dirname, 'node_modules/web-performance-
analyzer/templates/performance-dashboard.html');
            htmlTemplate = fs.readFileSync(defaultTemplatePath, 'utf8');
            fs.writeFileSync(templatePath, htmlTemplate, 'utf8');
        }

        const htmlReport = htmlTemplate.replace('__DATA_PLACEHOLDER__', JSON.stringify(combinedResult));
        const hostname = new URL(argv.url).hostname;
        const timestamp = new Date().toISOString().replace(/[:.]/g, '-');
        const reportFilename = `${hostname}-report-${timestamp}.html`;

        fs.writeFileSync(reportFilename, htmlReport, 'utf8');
        console.log(`HTML report generated: ${reportFilename}`);
    }
}
```