Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА ТА ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ВЕБ-СТОРІНОК ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

DEVELOPMENT AND RESEARCH OF METHODS FOR OPTIMIZING THE PERFORMANCE OF WEB PAGES FOR MOBILE DEVICES

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗм-21
Фурсик А. І.
Керівник:
к.т.н., доцент
Ліщина Н. М.

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент Суринович О. М.
_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *магістр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«___» _____ 202__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ
Фурсик Анастасії Ігорівни

1. Тема кваліфікаційної роботи: Розробка та дослідження методів оптимізації продуктивності веб сторінок для мобільних пристроїв
Керівник роботи: Ліщина Наталія Миколаївна, доцент, к.т.н.
_____
затверджені наказом закладу вищої освіти від «29» березня 2025 року № 190/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: 04 грудня 2025 р.
3. Вихідні дані до роботи *технічне та програмне забезпечення ЕОМ*
4. Зміст розрахунково-пояснювальної записки: *аналіз проблем низької продуктивності веб-сторінок на мобільних пристроях, огляд і вибір ефективних методів оптимізації, обґрунтування застосованих технологій, опис реалізації оптимізації експериментального веб-проєкту та проведення вимірювань продуктивності.*
5. Перелік графічного матеріалу 6 рисунків, 5 лістингів коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Ліщина Н. М.*		
Теоретичне дослідження та практична реалізація	*Ліщина Н. М.*		
Експериментальне дослідження системи	*Ліщина Н. М.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Андрущак І. Є.*		
Показник запозичень тексту	___%		
Академічна доброчесність	*Ліщина Н. М.*		

7. Дата видачі завдання *«02 квітня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	Провести огляд літературних джерел по темі кваліфікаційної роботи	02.05.2025	
2	Провести аналіз загальної проблеми і вибір напрямків дослідження	24.09.2025	
3	Розробити функціональну модель та архітектуру системи	01.11.2025	
4	Описати засоби розробки об’єкта проектування	19.11.2025	
5	Практична реалізація об’єкта проектування	26.11.2025	
6	Розробити методику для проведення експерименту	05.11.2025	
7	Провести аналіз результатів експерименту	15.11.2025	
8	Здача чистового варіанту кваліфікаційної роботи на кафедру	04.12.2025	

Здобувач вищої освіти _____ Фурсик А. І.

Керівник кваліфікаційної роботи _____ Ліщина Н. М.

АНОТАЦІЯ

Фурсик А. І. Розробка та дослідження методів оптимізації продуктивності веб-сторінок для мобільних пристроїв. Рукопис.

Кваліфікаційна робота магістра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається з вступу, 3 розділів, висновків та списку використаних джерел.

У роботі досліджено сучасні підходи до підвищення швидкодії веб-сторінок на мобільних пристроях, включаючи методи оптимізації завантаження ресурсів, зменшення об'єму та кількості запитів, використання адаптивних технологій рендерингу інтерфейсу, оптимізацію зображень та впровадження технік кешування. Представлено аналіз ключових показників продуктивності, таких як LCP, FID, CLS, та інструментів їх вимірювання. Розроблено та протестовано комплекс оптимізаційних рішень на прикладі реального веб-проєкту, що дозволило оцінити ефективність впроваджених заходів та сформулювати практичні рекомендації щодо подальшого вдосконалення продуктивності.

Отримані результати демонструють значне зменшення часу відображення основного контенту, підвищення стабільності інтерфейсу та покращення взаємодії користувача з веб-ресурсом на мобільних пристроях. Проведені розробка та дослідження може бути корисним для розробників та інженерів програмного забезпечення, які працюють над підвищенням якості та ефективності веб-додатків.

Ключові слова: оптимізація продуктивності, веб-сторінки, мобільні пристрої, JavaScript-оптимізація, адаптивний дизайн, кешування, оптимізація зображень, рендеринг.

ABSTRACT

Fursyk A. I. Development and Research of Methods for Optimizing the Performance of Web Pages for Mobile Devices. Manuscript.

Master's thesis in Software Engineering, specialty 121 «Software Engineering». Lutsk National Technical University. Lutsk, 2025.

The master's thesis consists of an introduction, 3 chapters, conclusions and a list of references.

The thesis explores modern approaches to improving the performance of web pages on mobile devices, including methods for optimizing resource loading, reducing the volume and number of requests, using adaptive interface rendering technologies, optimizing images, and implementing caching techniques. An analysis of key performance indicators such as LCP, FID, CLS, and tools for measuring them is presented. A set of optimization solutions was developed and tested on a real web project, which made it possible to evaluate the effectiveness of the measures implemented and formulate practical recommendations for further performance improvement.

The results demonstrate a significant reduction in the display time of the main content, increased interface stability, and improved user interaction with the web resource on mobile devices. The development and research conducted may be useful for software developers and engineers working to improve the quality and efficiency of web applications.

Keywords: performance optimization, web pages, mobile devices, JavaScript optimization, responsive design, caching, image optimization, rendering.

ЗМІСТ

ВСТУП

Стрімке зростання кількості мобільних користувачів та домінування мобільного трафіку у світових мережах зумовлюють підвищені вимоги до швидкодії та ефективності веб-сторінок. Попри наявність великої кількості методів оптимізації, значна частина веб-ресурсів усе ще демонструє низькі показники продуктивності, особливо за умов повільного мобільного інтернету або обмежених ресурсів пристрою. Практично вирішено чимало задач, пов'язаних з кешуванням, мінімізацією ресурсів, адаптивною графікою та використанням сучасних фронтенд-фреймворків, однак залишаються прогалини щодо комплексного підходу до оптимізації, вибору ефективних стратегій під специфічні типи веб-проєктів та впливу новітніх технологій (LCP-орієнтованої оптимізації, адаптивного рендерінгу, ресурсних пріоритетів тощо). Саме ці невирішені аспекти формують наукову проблему, що потребує поглибленого дослідження.

Світові тенденції спрямовані на досягнення максимальної продуктивності веб-додатків шляхом застосування прогресивних методів оптимізації: динамічне формування контенту на стороні сервера (SSR та SSG), інтелектуальне керування завантаженням ресурсів, використання AI-алгоритмів для адаптивної оптимізації, а також орієнтація на метрики Web Vitals, які стали стандартом оцінювання якості мобільного користувацького досвіду. Такі підходи активно впроваджуються провідними компаніями, але потребують систематизації та адаптації під практичні завдання сучасних веб-платформ.

Актуальність теми зумовлена необхідністю підвищення продуктивності веб-сторінок у мобільних умовах, що безпосередньо впливає на доступність цифрових сервісів, SEO-показники, комерційну ефективність та задоволеність користувачів. Попри значний прогрес у дослідженнях, актуальним залишається питання вибору оптимальних методів з урахуванням специфіки пристроїв, мережевого середовища та архітектури веб-застосунків. Саме це й визначає

доцільність виконання даної роботи та її значення для розвитку галузі веб-технологій.

Метою дослідження є розроблення, систематизація та експериментальне обґрунтування ефективних методів оптимізації продуктивності веб-сторінок для мобільних пристроїв, спрямованих на зменшення часу завантаження, покращення ключових метрик Web Vitals та забезпечення стабільної роботи веб-додатків в умовах різної якості мобільних мереж та апаратних обмежень.

Завданнями дослідження є:

– виконання аналізу предметної області та виявлення чинників, що впливають на продуктивність веб-сторінок у мобільному середовищі;

– дослідження сучасних методів оптимізації веб-ресурсів та встановлення їх ефективності і межі застосування;

– формування комплексної методології оптимізації, що охоплює архітектурні рішення, мережеві процеси, рендеринг, роботу зі скриптами, стилями та графічними ресурсами;

– створення експериментального середовища та підбір веб-проєкту для дослідження впливу оптимізаційних підходів на ключові метрики Web Vitals і пов'язані показники продуктивності;

– проведення експериментальних вимірювань продуктивності до та після застосування різних методів оптимізації;

– порівняння ефективності окремих методів і їхніх комбінацій для різних типів веб-сторінок і архітектур.

Об'єктом цього дослідження є процес забезпечення продуктивності та швидкодії веб-сторінок у мобільних веб-середовищах.

Предметом дослідження є методи, технології та інструменти оптимізації продуктивності веб-сторінок для мобільних пристроїв, зокрема стратегії керування завантаженням ресурсів, підходи до рендерингу, мережеві механізми, адаптивні алгоритми оптимізації та техніки покращення ключових метрик Web Vitals.

У роботі: уперше запропоновано узагальнену модель оптимізації продуктивності веб-сторінок для мобільних пристроїв, що поєднує ресурсну, мережеву та структурну оптимізацію з урахуванням реальних обмежень мобільних середовищ, удосконалено підхід до оцінювання ефективності оптимізаційних заходів за допомогою системи показників, отримано подальший розвиток використання адаптивних стратегій завантаження ресурсів, що дозволяють індивідуально налаштовувати процес оптимізації під характеристики конкретного пристрою та мережі.

Результати дослідження можуть бути використані у процесі розроблення та модернізації веб-проєктів, орієнтованих на мобільну аудиторію, у компаніях, що займаються створенням високопродуктивних веб-додатків, у освітніх курсах з фронтенд-розробки та веб-оптимізації, як рекомендації для впровадження в IT-індустрії під час аудиту та підвищення продуктивності веб-сайтів.

Запропоновані методи мають високий ступінь готовності до впровадження, що підтверджено експериментальною перевіркою на реальному веб-ресурсі. Положення та результати дослідження пройшли апробацію шляхом публікування статті у «Студентському науковому віснику» випуск 54 Луцького національного технічного університету, а також були представлені тези на X Міжнародній науково-практичній конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ – 2025)», Луцьк, Україна, 23-24 трав. 2025 р. [1].

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМАТИКИ ЗА ТЕМОЮ РОБОТИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд і аналіз предметної області проблеми, результатів існуючих теоретичних та експериментальних досліджень

Розвиток мобільних технологій та зростання частки мобільного трафіку в глобальній мережі визначили нові пріоритети у розробленні веб-систем, серед яких ключовим фактором стала продуктивність веб-сторінок на мобільних пристроях. На відміну від настільних систем, мобільні пристрої характеризуються обмеженими обчислювальними ресурсами та нестабільними мережевими умовами, що значно ускладнює забезпечення стабільної швидкодії веб-додатків. Саме тому дослідження ефективних методів оптимізації продуктивності стало предметом численних наукових праць, технічних звітів індустрії та практичних експериментів.

Детальний системний аналіз наукових джерел показує, що значна частина сучасних робіт фокусується на оптимізації часу завантаження сторінки та покращенні метрик Web Vitals, серед яких найбільш критичними є Largest Contentful Paint (LCP) – показник швидкості завантаження, який вимірює час, необхідний для відображення найбільшого візуального елемента (зображення, відео або текстового блоку) у видимій частині екрана користувача, First Input Delay (FID) – метрика, яка вимірює час від першої взаємодії користувача зі сторінкою до моменту, коли браузер реагує на цю дію та Cumulative Layout Shift (CLS) – метрика, що вимірює візуальну стабільність веб-сторінки, показуючи, наскільки несподівано елементи зміщуються під час її завантаження та використання. Як зображуються ці метрики при аналізі сторінки за допомогою Google Chrome DevTools Lighthouse можна побачити на рисунку 1.1.

Рисунок 1.1 – Зображення метрик Web Vitals

У публікаціях Google, Mozilla та незалежних дослідників доведено, що ці показники безпосередньо впливають на користувацький досвід та конверсійні метрики, тобто такі, що відображають успішність і частоту досягнення необхідних цілей користувачами. При цьому більшість досліджень підкреслює, що основний і найбільший внесок у погіршення продуктивності роблять великі графічні ресурси, блокуючі скрипти, неефективна структура DOM та недотримання принципів адаптивного рендерингу.

У теоретичних роботах значну увагу приділено методам оптимізації мережевих процесів – зокрема, використанню HTTP/2 (новий протокол передачі даних, який дозволяє завантажувати багато файлів через одне з'єднання та швидше передавати ресурси, зменшуючи затримки), стисненню даних, кешуванню (збереження ресурсів у браузері чи на сервері, щоб повторні запити завантажувалися без повторного отримання всіх даних), а також пріоритезації ресурсів (визначення, які файли повинні завантажитися з пріоритетом першості). Окремі дослідження акцентують на ефективності застосування CDN, відомого також, як мережа розподілених серверів, які доставляють контент з найближчого до користувача вузла, що суттєво пришвидшує завантаження сторінки, особливо на мобільних пристроях, попереднього завантаження (preload, prefetch) та lazy loading – відкладене завантаження медіа даних, які демонструють значне

зменшення часу отримання критичного контенту на мобільних пристроях. У сучасних експериментальних роботах також досліджуються моделі гібридного рендерингу: переходи від класичного CSR (Client-Side Rendering) до SSR (Server-Side Rendering) і SSG (Static Site Generation), що дозволяють мінімізувати навантаження на мобільний пристрій та прискорити перший рендер.

У сучасній науковій літературі все більше уваги приділяється не лише швидкості завантаження веб-сторінок, але й енергоспоживанню мобільних пристроїв в процесі їх перегляду. Так, Dornauer B., Felderer M. у своїй роботі проводить систематичний огляд стратегій збереження енергії у мобільних веб-додатках, виділяючи підходи, які дозволяють балансувати між продуктивністю та споживанням енергії [2]. З іншого боку, робота Van Riet J., Malavolta I., Ghaleb T. A., видана 2023 року, підкреслює важливість безперервного інжинірингу та покращенню продуктивності в індустріальних веб-додатках: навіть невеликі оптимізації, впроваджені регулярно, суттєво покращують користувацький досвід [3].

Щодо сучасних технологій оптимізації, Juho Vepsäläinen, Arto Hellas та Petri Vuorimaa, у своїй роботі, пропонують системний огляд методів, включаючи серверний рендеринг (SSR), code splitting, lazy loading та мінімізацію блокуючих ресурсів – всі ці підходи є ключовими для підвищення продуктивності веб-сторінок [4].

Паралельно з цим, значний обсяг досліджень спрямований на оптимізацію графіки – використання сучасних форматів медіаданих (WebP, AVIF), адаптивних зображень, а також алгоритмів автоматичного визначення оптимальної роздільної здатності. Експериментальні результати численних авторів підтверджують, що саме оптимізація зображень забезпечує найбільший приріст продуктивності на мобільних пристроях, а отже є найбільш ефективною. Однак більшість наукових робіт сфокусована на більш вужчих аспектах оптимізації і часто не враховує комплексний вплив різних методів на реальну продуктивність кінцевої системи.

У сучасній літературі простежується загальна тенденція переходу від часткової оптимізації до системного підходу оптимізації продуктивності веб-сторінок на мобільних пристроях, що включає комбінування технічних та архітектурних рішень. У звітах Lighthouse, Web.dev та у роботах провідних дослідників показано, що максимальна ефективність досягається при одночасному впровадженні декількох, міксованих разом, взаємопов'язаних методів, тоді як окремі оптимізації демонструють обмежений ефект. Разом з тим прогалини залишаються у питаннях вибору оптимальної стратегії для конкретного типу веб-системи, визначенні впливу оптимізацій на низькоресурсних пристроях та дослідженні поведінки веб-сторінок у зовсім нестабільних мережах.

Саме визначення таких оптимальних рішень, розроблення моделі комплексної оптимізації та проведення експериментів для підтвердження її ефективності формують ключові завдання даного дослідження.

Практичні рішення на цю тему пропонують такі дослідники, як, Venkataraajalu S. R. з роботою виданою 2024 року, яка вказує на поєднання серверної оптимізації, клієнтських стратегій та інтелектуального управління ресурсами як шлях до значного прискорення веб-сайтів які відображаються на мобільних пристроях [5]. У своїй роботі того ж самого 2024 року Оліховська С. В. аналізує клієнтську частину веб-сторінок під мобільні пристрої, зокрема показники FCP, оптимізацію зображень – що дуже добре корелює з практичними викликами фронтенд-розробки [6].

Також важливим є порівняння енергоспоживання та продуктивності між нативними мобільними додатками і веб-додатками: Ruben Horn, Abdellah Lahnaoui та співавтори роботи 2023 року показують, що веб-версії часто споживають більше ресурсів, що підкреслює необхідність оптимізації для мобільного вебу [7].

Дослідницький підхід до «енергетичних патернів» у веб-коді був представлений Pooja Rani, Jonas Zellweger та інші співавтори роботи 2024 року

вони виявили певні шаблони (наприклад, відкриття ресурсів лише за потреби), які можуть знизити енергоспоживання без значних втрат продуктивності [8].

І, звісно, Олійник М. Г. у своїй кваліфікаційній роботі 2023 року застосовує модель RAIL (Response, Animation, Idle, Load) для оптимізації прогресивного веб-застосунку, показуючи реальні ефекти на продуктивність і користувацький досвід [9]. Для кращого розуміння, RAIL – це фреймворк від Google для оцінювання та оптимізації продуктивності веб-інтерфейсів із позиції реального користувацького досвіду. Він розбиває роботу веб-продукту на чотири ключові етапи взаємодії та задає цільові пороги продуктивності для кожного з них.

Узагальнення результатів проведеного огляду дозволяє стверджувати, що стан проблеми на момент виконання даної роботи характеризується наявністю значної кількості теоретичних напрацювань та практичних рекомендацій, проте відсутність універсального комплексного підходу до оптимізації для мобільних умов залишається актуальною. Це обґрунтовує необхідність подальших детальних досліджень, спрямованих на систематизацію методів, їх експериментальну перевірку та визначення найбільш ефективних комбінацій для реальних веб-проєктів.

Таким чином, аналіз існуючих досліджень показує, що сучасна наука рухається в напрямі комплексного підходу, який поєднує продуктивність, енергоефективність та архітектурні стратегії. Водночас існує чітка прогалина щодо систематизації цих підходів та їх експериментального застосування у реальному мобільному веб-середовищі.

1.2 Огляд і аналіз методів та засобів розробки оптимізації продуктивності веб-сторінок на мобільних пристроях для вирішення проблеми дослідження

Сучасні веб-технології перебувають у постійному розвитку, зумовленому зростанням вимог до якості користувацького досвіду та швидкодії веб-ресурсів, у мобільному середовищі особливо. Глобальні тенденції цифровізації призвели

до того, що більшість користувачів взаємодіє з веб-сторінками через смартфони, які, на відміну від настільних систем, характеризуються нижчою продуктивністю обчислювальних ресурсів, обмеженим енергоспоживанням та нестабільністю мережевих з'єднань. Цю статистику можна побачити на рисунку 1.2, де зображено скріншот з HTTP Archive де виведена статистика з 01.01.2020 р. по 01.11.2025 р. [10]. Внаслідок цього, ефективність методів оптимізації продуктивності веб-сторінок на мобільних пристроях стала однією з ключових проблем як практичної веб-розробки, так і фундаментальних досліджень у галузі інженерії програмного забезпечення. Аналіз сучасних методів і технологій демонструє як значні досягнення у напрямку підвищення швидкодії таких систем, так і наявність недостатньо вивчених аспектів, що потребують додаткового теоретичного узагальнення та їх практичної оцінки.

Рисунок 1.2 – Статистика відкриття веб-сайтів на мобільних та настільних пристроях

Одним із ключових напрямів оптимізації, визначених у сучасній літературі, є оптимізація мережевих процесів. Вона охоплює використання вже згаданого вище HTTP/2 та HTTP/3, стиснення даних за допомогою Brotli, що є алгоритмом з відкритим вихідним кодом, розробленим Google для швидкого завантаження веб-сторінок та Gzip – це програма та формат файлів для стиснення і

розпакування даних, який використовує алгоритм стиснення без втрат DEFLATE. Він зменшує розмір файлів, роблячи їх зручнішими для передачі через мережу або для зберігання на диску, що особливо ефективно для текстових файлів, як-от HTML, CSS та JavaScript. Також важливим є застосування CDN-мереж, також згаданого вище попереднього завантаження і передбачуваного отримання ресурсів (preload, preconnect, prefetch). Підходи, орієнтовані на покращення транспортного рівня моделі OSI, дозволяють значно скоротити час отримання критичного контенту у мобільних умовах. Дослідження Akamai – американської компанії, що надає послуги доставки контенту (CDN) та хмарні сервіси, яка використовує велику мережу розподілених серверів для прискорення доступу користувачів до веб-сайтів та захисту від кібератак, Cloudflare – американської компанії, що надає послуги мережевої безпеки та оптимізації продуктивності інтернет-ресурсів та незалежних авторів показують, що грамотне використання CDN забезпечує підвищення швидкодії у 30-70 %, особливо у регіонах із великими затримками мережі. У поєднанні з політиками кешування (Cache-Control, ETag, Service Worker caching) та адаптивним вибором ресурсів (наприклад, зображень різних розмірів) вдається мінімізувати кількість запитів та навантаження на мобільний пристрій.

Окремим напрямом досліджень є оптимізація рендерингу та обробки JavaScript, яка відіграє значну роль у контексті мобільних пристроїв через їхню нижчу продуктивність CPU. У звітах Chrome Dev Summit та численних академічних роботах показано, що саме JavaScript є найбільш ресурсоємним компонентом веб-сторінки. Основними техніками оптимізації є code splitting, tree shaking, видалення невикористаного коду (unused code elimination), застосування динамічного імпорту, а також використання надлегких фреймворків або нативних можливостей браузера. Зменшення обсягу JavaScript є критично важливим, оскільки на мобільних пристроях навіть додаткові 100-200 КБ стиснутого JavaScript можуть додавати секунди до часу інтерпретації та компіляції, що є великою оптимізацією. Експериментальні дослідження Juho Vepsäläinen, Arto Hellas та Petri Vuorimaa у вже згаданій вище роботі 2024 року

демонструють, що оптимізація JavaScript, включаючи правильне визначення пріоритетів завантаження скриптів, здатна зменшити FID (структурований файл, що містить детальну інформацію про товари або послуги інтернет-магазину, такий як назва, опис, ціна, посилання, зображення тощо) та INP ((Interaction to Next Paint) – це метрика, яку Google використовує для вимірювання швидкості реагування веб-сайту на дії користувача, такі як натискання, дотик або введення тексту) у 2-4 рази залежно від складності застосунку [11].

Особливе місце у дослідженнях продуктивності займає тема оптимізації зображень. Зображення становлять від 50 до 80 % обсягу типового веб-ресурсу, що підтверджено статистикою HTTP Archive. Тому сучасні наукові та індустріальні підходи рекомендують застосовувати адаптивні зображення із використанням елементів <picture>, srcset та sizes, що дозволяють завантажувати лише той обсяг графіки, який потрібен конкретному пристрою. Крім того, значну ефективність демонструє техніка lazy loading – відкладене завантаження зображень поза межами області видимості, яке дозволяє прискорити появу основного контенту та зменшити навантаження на мережу. У сучасних браузерах достатньо використати атрибут loading="lazy" або ж застосувати Intersection Observer API. У наукових роботах підтверджено, що застосування форматів AVIF та WebP дозволяє зменшити розмір графічних ресурсів на 50-80 %, а використання адаптивних стратегій скорочує LCP на 30-60 %.

Значну увагу дослідники приділяють питанням архітектури рендерингу. Класичний клієнтський рендеринг (CSR) продовжує втрачати актуальність для мобільних пристроїв через високе навантаження на ресурси смартфона та необхідність завантаження значних обсягів JavaScript перед тим, як сторінка стане інтерактивною. Натомість у сучасних дослідженнях перевага надається гібридним моделям – SSR (Server-Side Rendering), SSG (Static Site Generation), ISR (Incremental Static Regeneration), Edge-Rendering та Streaming SSR. Ці архітектурні підходи дозволяють значно зменшити час до першого відображення контенту (FCP) та часу до LCP, перекладаючи частину обчислювальних процесів на сервер, який має більшу продуктивність. Зокрема, SSR забезпечує

формування HTML на сервері, що дозволяє користувачу одразу отримати готову сторінку без необхідності виконання великих скриптів. Додаткові техніки – hydration, partial hydration та island architecture – мінімізують обсяг JavaScript, який виконується на клієнті, зменшуючи навантаження на мобільний процесор.

У літературі активно розглядається проблема оптимізації каскадних таблиць стилів (CSS). Хоча CSS менш ресурсоємний, ніж JavaScript, неправильно структуровані стилі можуть блокувати рендеринг або призводити до значних зміщень контенту (погіршуючи показник CLS). Тому сучасні підходи включають застосування критичних CSS (critical CSS), мінімізацію стилів, видалення невикористаних класів, а також перенесення некритичних стилів у асинхронні стилі. Це дозволяє уникнути блокування побудови DOM і прискорити початковий рендеринг сторінки.

У дослідженнях останніх років великий інтерес викликають теми енергоспоживання та продуктивності. У роботах Dornauer B., Felderer M. 2023 року та D. Bui 2023 року проведено комплексний аналіз енергетичної ефективності веб-сторінок, що є особливо важливим для мобільних пристроїв. Установлено, що агресивна оптимізація продуктивності, наприклад постійне перерахування DOM, виклик великої кількості подій прокрутки або використання великих фреймворків, може призводити до збільшення енергоспоживання, що впливає на час автономної роботи пристрою [12]. Таким чином, автори пропонують використовувати адаптивні алгоритми оптимізації, які враховують як продуктивність, так і енергетичні витрати.

Сучасні інструменти оцінювання продуктивності, такі як Google Lighthouse, WebPageTest, Chrome DevTools Performance, SpeedCurve та інші, відіграють ключову роль у плануванні оптимізаційних заходів, приклад цього вже було наведено вище. Вони дозволяють ідентифікувати вузькі місця, аналізувати критичний шлях рендерингу, визначати обсяг невикористаних ресурсів, оцінювати час блокування головного потоку (main thread blocking time), а також вимірювати всі ключові метрики Web Vitals. Використання Lighthouse у

поєднанні з профілюванням JavaScript дає змогу аналітично довести ефективність конкретного методу оптимізації.

У межах аналізу методів оптимізації важливо також розглянути аспект автоматизації та застосування CI/CD-підходів – набір методик, які автоматизують процеси розробки, тестування та розгортання програмного забезпечення, щоб прискорити та зробити більш надійним випуск оновлень. Сучасні системи розгортання (Netlify, Vercel, Cloudflare Pages) забезпечують автоматичне застосування оптимізацій, таких як мініфікація, агрегація ресурсів, компресія, оптимізація зображень, пріоритезація JavaScript-модулів. Це дозволяє зробити процес оптимізації системним, виключивши людський фактор.

Аналіз сучасної наукової та технічної літератури показує, що ефективна оптимізація продуктивності веб-сторінок на мобільних пристроях ґрунтується на комплексному підході, що включає архітектурні, ресурсні, мережеві, рендерингові та поведінкові техніки. Поєднання цих методів у єдину систему дає змогу досягти максимального поліпшення метрик Web Vitals, зменшення часу завантаження та підвищення стабільності інтерфейсу. У сучасних застосунках усе частіше використовуються інструменти автоматичної оптимізації, такі як Image CDN, попереднє кешування через Service Worker та динамічне формування ресурсів у залежності від умов мережі (network-aware loading).

Практичним підтвердженням доцільності цих технік є впровадження адаптивних стратегій роботи з графічними ресурсами. Зображення становлять найбільшу частку обсягу переданих даних у мобільному трафіку, тому їх оптимізація має критичне значення. Для демонстрації практичного застосування деяких методів оптимізації в лістингу 1.1 наведено приклад використання сучасних стратегій завантаження зображень. У цьому прикладі показано підтримку нових форматів (AVIF, WebP), використання lazy loading, явне визначення розмірів для запобігання зміщенню контенту (CLS), а також застосування властивості content-visibility для оптимізації процесу рендерингу. Додатково наведено приклад використання атрибутів fetchpriority та decoding,

що забезпечують гнучкіший контроль над порядком обробки ресурсів браузером.

Лістинг 1.1 – Сучасні стратегії завантаження зображення

```
<!-- Контейнер зображення з сучасними техніками оптимізації -->
<picture>
    <!-- Найсучасніший формат AVIF -->
    <source
        srcset="images/photo.avif"
        type="image/avif"
        width="600"
        height="400"
    />

    <!-- Формат WebP як fallback -->
    <source
        srcset="images/photo.webp"
        type="image/webp"
        width="600"
        height="400"
    />
</picture>

<!-- Додатковий приклад адаптивного завантаження -->
<img
    src="images/thumbnail.jpg"
    srcset="
        images/thumbnail_small.jpg 320w,
        images/thumbnail_medium.jpg 480w,
        images/thumbnail_large.jpg 800w
    "
    sizes="(max-width: 600px) 100vw, 600px"
    alt="Responsive preview"
    loading="lazy"
    decoding="async"
    width="600"
    height="350"
    style="content-visibility: auto;"
/>
```

Кінець лістингу 1.1

Підсумовуючи, застосування сучасних форматів зображень, поєднання адаптивних роздільностей, коректне визначення розмірів та раціональне використання механізмів відкладеного завантаження формують комплексну

основу для підвищення продуктивності веб-сторінок. Наведені техніки є важливими елементами сучасних фронтенд-практик, оскільки зменшують навантаження на мережу, покращують швидкодію інтерфейсу та забезпечують більш стабільний користувацький досвід на мобільних пристроях.

У лістингу 1.2 продемонстрована оптимізація JavaScript шляхом використання динамічного імпорту. Цей підхід є ключовим елементом сучасної архітектури веб-сайтів, оскільки дозволяє суттєво зменшити обсяг JavaScript-коду, який необхідно завантажувати на етапі початкового рендера. Замість традиційного підходу, де всі модулі та залежності потрапляють у глобальний бандл, сучасні браузери підтримують механізм code splitting та on-demand loading, що робить завантаження більш адаптивним і залежним від дій користувача. Це особливо важливо для мобільних пристроїв, де пропускна здатність та доступна пам'ять можуть бути обмеженими, а кожен зайвий кілобайт негативно впливає на швидкодію.

У реальних проєктах динамічний імпорт дозволяє завантажувати функціональність лише тоді, коли користувач переходить до відповідного розділу сторінки, відкриває модальне вікно, взаємодіє з формою, використовує пошук або активує інші вторинні можливості. Це дає змогу суттєво зменшити Time to Interactive (TTI) та оптимізувати First Input Delay (FID) [13]. У наведеному нижче лістингу 1.2 показано не лише базовий приклад імпорту модуля, але й розширені можливості: паралельний імпорт, умовне завантаження залежностей, попереднє передбачуване завантаження (prefetching), обробка помилок, а також розділення модулів за сценаріями використання.

Лістинг 1.2 – Динамічний імпорт через JavaScript

```
    // Динамічний імпорт після натискання кнопки
    document.getElementById("open-modal").addEventListener("click", async
() => {
        const module = await import("./modal.js");
        module.openModal();
    });

    // Умовний імпорт залежно від ширини екрана
```

```
    if (window.innerWidth < 600) {
        import("./mobile.js").then((m) => m.initMobile());
    } else {
        import("./desktop.js").then((m) => m.initDesktop());
    }

    // Паралельний імпорт двох модулів
    async function loadGallery() {
        const [gallery, lightbox] = await Promise.all([
            import("./gallery.js"),
            import("./lightbox.js")
        ]);
        gallery.initGallery();
        lightbox.initLightbox();
    }
    document.getElementById("open-gallery").addEventListener("click",
loadGallery);

    // Імпорт при появі елемента в зоні видимості
    const observer = new IntersectionObserver(async (entries) => {
        if (entries[0].isIntersecting) {
            const module = await import("./analytics.js");
            module.track();
            observer.disconnect();
        }
    });
    observer.observe(document.querySelector("#lazy-section"));
```

Кінець лістингу 1.2

Застосування динамічного імпорту забезпечує істотний виграш у продуктивності, оскільки JavaScript-код, який не потрібен користувачу одразу, не впливає на час початкового рендерингу та швидкість відгуку інтерфейсу [14]. Завдяки цьому зменшується розмір головного бандла, полегшується робота браузера, а веб-сторінка стає доступною для взаємодії значно швидше. Саме тому динамічне завантаження сьогодні вважається обов'язковим елементом оптимізованих веб-застосунків.

Узагальнюючи проведений огляд, можна стверджувати, що, як вже було зазначено вище, сучасні методи та інструменти оптимізації продуктивності веб-сторінок орієнтовані на комплексний підхід, який охоплює оптимізацію мережевих ресурсів, зменшення обсягу JavaScript, ефективне використання

архітектури рендерингу, адаптивну оптимізацію графіки та застосування автоматизованих інструментів аналізу. Проте попри значну кількість практик та інструментів, у науковій літературі існує потреба у систематизації цих методів та розробленні моделі, що інтегрує їх у єдину оптимізаційну систему. Саме ця прогалина визначає наукову новизну та практичну значущість даної кваліфікаційної роботи.

1.3 Постановка завдання на кваліфікаційну роботу магістра

На основі детального аналізу сучасних тенденцій у сфері веб-інженерії, огляду предметної області та всебічного вивчення наукових публікацій, що стосуються підвищення продуктивності веб-ресурсів, було встановлено, що однією з найбільш актуальних проблем залишається відсутність системного підходу до оптимізації веб-сторінок у мобільних середовищах. Незважаючи на значний обсяг сучасних досліджень, більшість робіт фокусуються на окремих аспектах, таких як оптимізація мережевих процесів, мінімізація часу виконання скриптів, впровадження прогресивних форматів зображень чи оптимізація алгоритмів рендерингу. Проте ці рішення зазвичай не інтегруються у цілісну архітектуру та не враховують специфіку мобільних пристроїв, що відрізняються обмеженістю ресурсів, невисокою обчислювальною потужністю, нестабільністю мережі та підвищеною чутливістю до енергоспоживання.

Виявлено, що більшість наявних підходів орієнтовані на роботу у десктопному середовищі й недостатньо адаптовані до мобільної взаємодії, де час відгуку, енергоефективність, кількість мережевих запитів та коректність рендерингу відіграють критичну роль для користувацького досвіду. Сучасні веб-додатки, що активно використовують JavaScript, розгалужені бібліотеки та багаторівневі каскадні таблиці стилів, створюють значне навантаження на головний потік, що уповільнює рендеринг і негативно позначається на таких показниках, як Largest Contentful Paint, First Input Delay/INP та Cumulative Layout Shift. Таким чином, у роботі визначено необхідність сформувати науково

обґрунтований, узагальнений і водночас практично орієнтований підхід, здатний охопити різні рівні оптимізації – від архітектури клієнтського застосунку та структури DOM до мережевої взаємодії, роботи з графічними ресурсами, механізмів кешування та енерговитрат пристрою [15].

На основі огляду досліджень і аналітичних матеріалів сформовано висновок про те, що проблема оптимізації мобільних веб-сторінок має міждисциплінарний характер, оскільки поєднує питання програмної інженерії, аналізу продуктивності, веб-архітектури, дизайну інтерфейсів, поведінкових патернів користувача та мобільних мережевих технологій. Існуючі підходи, хоча й демонструють високу ефективність у певних умовах, часто не забезпечують достатньої комплексності і не враховують взаємозалежності між структурними елементами веб-сторінки, наслідком чого стає лише часткове зменшення часу рендерингу, у той час як загальна швидкодія залишається недостатньою. Це особливо відчутно для сайта, що орієнтується на мобільну аудиторію та містить насичений графічний контент, інтерактивні елементи чи використовує фреймворки з великою кількістю залежностей.

У межах цієї кваліфікаційної роботи передбачається створення комплексної методології, що включає як теоретичне обґрунтування, так і експериментальну перевірку запропонованих методів оптимізації. Перед дослідженням було визначено, що ключовою передумовою для побудови ефективної системи оптимізації є детальна ідентифікація чинників, які найсуттєвіше впливають на продуктивність веб-сторінок на мобільних пристроях. Сюди належать параметри, пов'язані з обчислювальними можливостями процесора, обсягом оперативної пам'яті, характеристиками мобільної мережі, швидкістю дискового доступу, а також архітектурні особливості браузера, зокрема специфіка виконання JavaScript у головному потоці, обробка переробки стилів, побудова макета та процеси композиції [16]. У рамках роботи буде здійснено оцінку того, яким чином ці чинники впливають на метрики Web Vitals, на прикладні показники взаємодії та на суб'єктивне сприйняття швидкодії користувачем.

Особливе місце у постановці завдання займає систематизація методів оптимізації продуктивності, що дозволяє створити уніфіковану модель, засновану на узагальненні існуючих технологічних рішень. У розробці та дослідженні передбачається розглянути такі напрямки, як лінійна та прогресивна оптимізація мережевих запитів, застосування технік відкладеного завантаження ресурсів, мінімізація JavaScript за рахунок code splitting і tree shaking, впровадження сучасних форматів зображень, а також використання серверних підходів рендерингу – SSR, SSG та гібридних моделей. Системне поєднання цих методів дозволить сформувати концептуальну модель комплексної оптимізації, застосовану до широкого спектру веб-сайтів незалежно від їхньої архітектури.

У зв'язку з необхідністю підтвердження ефективності таких рішень передбачено проведення серії експериментальних досліджень із вимірюванням ключових показників продуктивності до та після оптимізації. До уваги беруться як стандартні метрики Web Vitals (LCP, INP, CLS, TTFB), про які вже було описано вище, так і додаткові показники, що стосуються енергоспоживання, навантаження на головний потік, частоти рефлоу, часу до готовності до взаємодії та змін у кількості мережевих запитів [17]. Отримані результати дозволять оцінити як окремі методи, так і їхні комбінації з погляду ефективності для різних типів веб-систем, зокрема динамічних SPA-додатків, статичних інформаційних ресурсів та e-commerce платформ – онлайн-системи, які дозволяють купувати та продавати товари або послуги через інтернет.

Особливу увагу в роботі приділено питанням формування практичних рекомендацій щодо підвищення продуктивності мобільних веб-сторінок, орієнтованих на розробників та інженерів, які створюють сучасні інтерфейси. Рекомендації будуть базуватися на узагальнених експериментальних даних та розробці на базі цих даних і міститимуть як архітектурні поради, так і прикладні підходи до роботи зі стилями, медіаресурсами, JavaScript-логікою та мережею.

Завершенням постановки завдання є формування висновку про те, що виконання зазначених дослідницьких етапів дозволить розробити науково обґрунтований та практично перевірений комплексний підхід до оптимізації

продуктивності веб-сторінок у мобільному середовищі, який сприятиме підвищенню швидкодії, стабільності роботи, зменшенню енергоспоживання пристроїв користувачів та формуванню стандартів розробки високоефективних веб-ресурсів майбутнього. Таке дослідження має не лише прикладне значення, але й наукову цінність, оскільки дозволяє розширити інструментарій веб-інженерії та створює фундамент для подальших наукових робіт у галузі оптимізації продуктивності веб-додатків.

Отже, для досягнення всіх цих результатів у межах виконання кваліфікаційної роботи необхідно вирішити такі завдання:

– виконати аналіз предметної області та виявити чинники, що впливають на продуктивність веб-сторінок у мобільному середовищі;

– дослідити сучасні методи оптимізації веб-ресурсів та встановити їх ефективність і межі застосування;

– сформувати комплексну методологію оптимізації, що охоплює архітектурні рішення, мережеві процеси, рендеринг, роботу зі скриптами, стилями та графічними ресурсами;

– створити експериментальне середовище та підібрати веб-проєкт для дослідження впливу оптимізаційних підходів на ключові метрики Web Vitals і пов’язані показники продуктивності;

– провести експериментальні вимірювання продуктивності до та після застосування різних методів оптимізації;

– порівняти ефективність окремих методів і їхніх комбінацій для різних типів веб-сторінок і архітектур.

РОЗДІЛ 2
ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ МЕТОДІВ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ВЕБ-СТОРІНОК ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

2.1 Обґрунтування вибору шляхів, технологій, алгоритмів і засобів вирішення поставленого завдання

Оптимізація продуктивності веб-сторінок для мобільних пристроїв є багатовимірною науково-технічною проблемою, як вже зазначалось вище, яка охоплює питання мережевої взаємодії, обчислювальної ефективності, енергоощадності, архітектури веб-додатків, моделювання поведінки користувача та роботи браузера. Для обґрунтованого вибору методів і технологій необхідним є формування такої концептуально-методичної бази, яка враховуватиме специфіку мобільних середовищ, у тому числі їхню низьку продуктивність процесорів, обмеженість оперативної пам'яті, енергетичні обмеження та високу варіативність швидкості мережі. Саме такі чинники визначають критерії та логіку добору оптимізаційних технік, що становлять предмет даного дослідження.

Першою групою рішень, які доцільно використати для реалізації поставленого завдання, є методи, пов'язані з прискоренням початкового завантаження веб-сторінки. Їх вибір ґрунтується на численних теоретичних роботах і практичних дослідженнях, які доводять, що показники Largest Contentful Paint (LCP), Time To First Byte (TTFB) та Interaction to Next Paint (INP) безпосередньо визначають сприйняття швидкодії мобільними користувачами. Нестабільність мобільних мереж зумовлює необхідність впровадження алгоритмів пріоритезації ресурсів, таких як критичний рендеринг HTML, preload ключових стилів та шрифтів, оптимізоване завантаження JavaScript-коду та зменшення часу блокування основного потоку. Технології HTTP/2 та HTTP/3 обрано у якості базових, оскільки вони забезпечують мультиплексування потоків і покращують використання мобільних мережевих каналів. Використання

протоколу QUIC, що є транспортним мережевим протоколом, який використовує UDP для забезпечення швидких і надійних інтернет-з'єднань шляхом об'єднання функцій TCP та TLS, дає змогу мінімізувати латентність під час початкового з'єднання, що особливо важливо для мобільних умов.

Другим напрямом є оптимізація виконання JavaScript-коду [18]. Браузери мобільних пристроїв характеризуються значним часом JIT-компіляції (Just-in-Time) та повільнішою роботою інтерпретатора порівняно з десктопними системами, тому ефективне керування розміром і структурою JS є критично важливим. У цьому дослідженні обґрунтовано вибір таких технік, як code splitting, tree-shaking та deferred loading, а також застосування алгоритмів мінімізації блокуючих скриптів і конструювання lightweight-логіки. Системи збірки на кшталт Webpack, Vite або Rollup виступають інструментами реалізації зазначених алгоритмів. Для аналізу та моделювання впливу виконання скриптів на головний поток обрано профілювальні інструменти Chrome Performance, які дозволяють встановити кореляцію між обсягом коду та затримками під час рендерингу.

Третім важливим напрямом є оптимізація графічних ресурсів. З огляду на те, що саме зображення становлять основну частку обсягу веб-сторінки, їх оптимізація має вирішальне значення для мобільної продуктивності. У роботі обґрунтовано вибір сучасних форматів WebP та AVIF, які забезпечують значно кращу компресію без помітної втрати якості, а також використання адаптивних алгоритмів генерації зображень з урахуванням роздільності екрану, щільності пікселів (device pixel ratio) і типу мережі. Логіка lazy loading зображень обрана як базова, оскільки вона дозволяє завантажувати тільки ті ресурси, які наразі потрібні користувачу. Додатково використовується підхід responsive images із застосуванням атрибутів srcset та sizes, що оптимізує передачу даних для різних типів пристроїв.

Архітектурні рішення також є ключовими для досягнення високої продуктивності у мобільних середовищах. На основі аналізу наукових джерел і практичних рекомендацій зроблено вибір на користь гібридних моделей

рендерингу. Така комбінація забезпечує скорочення часу до першого рендеру, рівномірний розподіл обчислювального навантаження між сервером і клієнтом та покращення стабільності роботи застосунку при нестабільних мережах. Перевірка адекватності цих моделей здійснюється шляхом порівняння показників продуктивності та побудови експериментальних конфігурацій веб-сторінок з вимірюванням метрик Web Vitals за допомогою Lighthouse, WebPageTest та профілювальних засобів браузера.

Важливим елементом дослідження є моделювання обмежених умов мобільного середовища, зокрема слабкого сигналу мережі, політики енергоспоживання та швидкості реакції інтерфейсу. Для цього застосовується throttle-емуляція мережі та процесора, що дозволяє оцінити коректність роботи алгоритмів оптимізації на пристроях із низькою продуктивністю. Такий підхід забезпечує об'єктивність результатів та дозволяє будувати моделі продуктивності, які враховують реальні сценарії використання веб-сторінки.

Окремим критерієм вибору технологій є вплив оптимізацій на енергоспоживання мобільних пристроїв. В 1 розділі вже було зазначено, що надмірна кількість JavaScript-операцій, постійні DOM-модифікації та часте виконання reflow спричиняють підвищене навантаження на процесор, що, у свою чергу, збільшує витрати енергії. Тому у роботі обґрунтовано застосування RAIL-моделі та energy-aware підходів, які передбачають оптимізацію тривалості та частоти обчислювальних операцій.

Комплексність завдання оптимізації продуктивності вимагає використання інтегрованих засобів моніторингу та тестування, які дозволяють оцінити валідність вибраних методів. Підтвердження адекватності моделей здійснюється через багаторазове тестування, статистичний аналіз результатів та порівняння показників до і після застосування оптимізаційних заходів.

Узагальнюючи викладене, вибір технологій, алгоритмів і методів оптимізації у рамках даного дослідження зумовлений такими чинниками, як відповідність мобільній специфіці браузерів, можливість зниження навантаження на обчислювальні ресурси, здатність забезпечити стабільну

швидкодію у різних мережевих умовах та підтверджена ефективність у сучасних наукових працях. Обраний комплекс рішень охоплює мережеві оптимізації, оптимізацію графічних ресурсів, скорочення та раціоналізацію JavaScript-коду, застосування SSR/SSG/CSR-гибридів, використання моделей оцінки продуктивності браузера та енергоспоживання, а також впровадження методів адаптивного рендерингу та обчислювальної оптимізації. Таким чином, сформовано науково обґрунтовану методологію, яка створює основу для проведення експериментальних досліджень і побудови комплексної моделі оптимізації продуктивності веб-сторінок для мобільних пристроїв.

2.2 Практична реалізація об'єкта проектування

Практична реалізація розробленого програмного рішення ґрунтувалася на створенні та оптимізації веб-ресурс, орієнтованого передусім на мобільних користувачів, що відповідає сучасним вимогам цифрової екосистеми. Основною метою було запровадити комплекс технічних і технологічних рішень, які забезпечують стабільну, швидку та передбачувану роботу веб-сторінок у реальних умовах використання, враховуючи поведінку користувачів, особливості мобільних мереж, обмеження апаратних потужностей та специфіку рендерингу інтерфейсів на різних пристроях. Відповідно до поставлених задач, у межах реалізації було побудовано інфраструктуру клієнтської частини, виконано інтеграцію з серверною логікою та проведено всебічне тестування продуктивності на основі сучасних метрик. Окрему увагу було приділено адаптації архітектури веб-сторінок для зменшення часу завантаження, підвищення інтерфейсної чуйності та мінімізації візуальних зсувів, що є критично важливими для досягнення високих значень показників LCP, FID та CLS.

Процес розробки включав проєктування логічної структури сторінок, підготовку оптимізованої графіки, упорядкування шарів DOM-системи та впровадження механізмів відкладеного завантаження ресурсів, які покращують

початкову продуктивність. Реалізована система передбачала використання актуальних підходів, спрямованих на зменшення обсягу переданих даних, оптимізацію мережевих запитів та скорочення кількості блокувальних процесів під час рендерингу. Важливою частиною роботи стала оцінка ефективності застосовуваних методів безпосередньо на робочому веб-сайті, де кожен етап оптимізації відстежувався у реальних умовах, що дозволило врахувати поведінкові сценарії, які важко відтворити в лабораторному середовищі. Оскільки вихідний програмний код був розроблений для реального робочого проекту, він захищений авторськими правами, тому у тексті не наводиться конкретний і повний метод реалізації, однак описані процеси відображають застосовані технічні підходи та отримані результати, а також можна побачити невеликі частини програмного коду, без специфікацій, які захищені цим же авторським правом.

У межах практичної реалізації було проведено адаптацію структури компонентів, стилів та ресурсів під специфіку мобільного рендерингу. Зокрема, для зменшення часу завантаження було впроваджено оптимізовану логіку завантаження медіафайлів за допомогою механізмів lazy loading, що дозволило уникнути передчасного завантаження графічних елементів, які не знаходяться у видимій зоні екрану. Для цього застосовано нативний атрибут loading="lazy" та кастомні скрипти для браузерів зі старішою підтримкою [19]. Також виконано оптимізацію форматів зображень шляхом конвертації у WebP та AVIF, що суттєво зменшило вагу файлів без помітної втрати якості, що особливо важливо при обмеженій пропускній здатності мобільних мереж.

Певна частина оптимізації вимагала додаткової роботи з DOM-системою, оскільки великі та багаторівневі дерева елементів негативно впливають на швидкість рендерингу та взаємодії. Розбиття складних компонентів на логічно автономні модулі та впровадження механізмів часткового рендерингу дали змогу зменшити кількість перерахунків стилів та зменшили навантаження на основний потік браузера. Для цього було використано оптимізовані обробники подій підхід delegation, який зменшує кількість активних listeners у DOM. В

лістингу 2.1 наведено приклад оптимізованого обробника розгортання блоків контенту, що застосовувався при інтерактивній роботі з інформаційними списками.

Лістинг 2.1 – Оптимізований розробник розгортання блоків

```
// Делегований обробник для розгортання блоків контенту
document.addEventListener("click", (event) => {
    const trigger = event.target.closest("[data-toggle]");
    if (!trigger) return;

    const blockId = trigger.getAttribute("data-toggle");

    // Мінімізуємо доступи до DOM - кешуємо елемент
    const block = document.getElementById(blockId);
    if (!block) return;

    // Керуємо станом через data-атрибут
    const isOpen = block.dataset.open === "true";
    block.dataset.open = String(!isOpen);

    // Встановлення max-height через requestAnimationFrame
    requestAnimationFrame(() => {
        if (!isOpen) {
            block.classList.add("open");
            block.style.maxHeight = block.scrollHeight + "px";
        } else {
            block.style.maxHeight = "0px";
            block.addEventListener(
                "transitionend",
                () => block.classList.remove("open"),
                { once: true }
            );
        }
    });
});

// (pre-calculation batch update)
function prepareExpandableBlocks() {
    document.querySelectorAll("[data-expand]").forEach((el) => {
        el.style.overflow = "hidden";
        el.style.transition = "max-height 0.3s ease";
        el.style.maxHeight = "0px";
    });
}

prepareExpandableBlocks();
```

Кінець лістингу 2.1

Такий підхід дозволив уникнути створення великої кількості окремих слухачів подій для кожного елементу списку, що зменшило навантаження на інтерпретатор JavaScript та позитивно вплинуло на час обробки подій, особливо на слабких мобільних процесорах.

Важливим напрямом оптимізації стала робота зі стилями та зменшенням їх блокувального впливу на рендеринг сторінки. Оскільки CSS-файли належать до ресурсів, що блокують побудову дерева рендерингу (Render Tree), їх неконтрольоване або надмірне використання призводить до збільшення часу відображення першого екрану та до погіршення показника LCP. У межах дослідження значна частина CSS була реорганізована: усунено дублікати, оптимізовано специфічність селекторів та видалено невикористані правила (unused CSS), що дозволило скоротити загальний обсяг таблиць стилів.

Для критичних стилів першого екрану було застосовано метод inline-critical CSS, який дозволив браузеру негайно сформувати базову структуру інтерфейсу без очікування зовнішніх ресурсів. Для решти стилів надано можливість завантажуватися асинхронно – браузер отримує їх за допомогою методу preload, а після завершення завантаження атрибут rel замінюється на stylesheet. Це дозволяє ефективно керувати блокувальним ефектом CSS і водночас уникати «спалаху неоформленого контенту» (FOUC). У лістингу 2.2 наведено приклад оптимізованого завантаження стилів із забезпеченням fallback-поведінки для випадків, коли JavaScript недоступний.

Лістинг 2.2 – Оптимізоване завантаження стилів

```
<!-- Критичні стилі для першого екрану -->
<style>
    /* Приклад мінімально необхідних стилів */
    body { margin: 0; font-family: system-ui; }
    .header { height: 60px; background: #fff; }
</style>

<!-- Асинхронне завантаження основного CSS -->
<link
    rel="preload"
    href="/css/main.css"
    as="style"
```

```
    onload="this.rel='stylesheet'"
>

<!-- Попереднє завантаження додаткових стилів -->
<link
    rel="preload"
    href="/css/animations.css"
    as="style"
    onload="this.rel='stylesheet'"
>

<!-- Неблокувальна вставка стилів через JS -->
<script>
    const asyncStyles = [
        "/css/theme-dark.css",
        "/css/responsive.css"
    ];

    asyncStyles.forEach((href) => {
        const link = document.createElement("link");
        link.rel = "preload";
        link.as = "style";
        link.href = href;
        link.onload = () => link.rel = "stylesheet";
        document.head.appendChild(link);
    });
</script>

<!-- Fallback для користувачів без JavaScript -->
<noscript>
    <link rel="stylesheet" href="/css/main.css">
    <link rel="stylesheet" href="/css/animations.css">
</noscript>
```

Кінець лістингу 2.2

Це рішення допомогло зменшити затримку у відображенні основного контенту та покращило показник Largest Contentful Paint, що є одним із найважливіших параметрів мобільної продуктивності.

Особлива увага приділялася оптимізації JavaScript-файлів, оскільки саме скрипти часто створюють значну затримку під час взаємодії користувача зі сторінкою. У процесі реалізації було виконано поділ коду за принципом «code splitting», що дозволило завантажувати лише ті модулі, які потрібні на конкретній сторінці [20]. Наприклад, окремі компоненти, що використовувалися

тільки у певних секціях сайту, були винесені в окремі бандли, які завантажувалися лише при переході на відповідну сторінку. Для покращення часу початкової взаємодії використано deferred-завантаження скриптів, що показано у лістингу 2.3. Основний бандл завантажується з атрибутом defer, що дозволяє браузеру не блокувати побудову DOM і зменшує затримку перед відображенням першого контенту. Додаткові модулі, такі як галерея чи відеоплеєр, завантажуються лише тоді, коли вони реально присутні на сторінці, що запобігає надлишковому використанню трафіку та знижує навантаження на процесор мобільних пристроїв. Особливо важливим є використання асинхронного завантаження аналітичного модуля, яке не впливає на критичний шлях рендерингу та дозволяє переносити необов'язкові скрипти поза основний потік виконання.

Лістинг 2.3 – Використання deferred-завантаження скриптів

```
    <!-- Основний бандл застосунку завантажується з атрибутом defer,
         що дозволяє браузеру продовжувати парсинг HTML без блокування --
>
    <script src="/js/app.core.min.js" defer></script>

    <!-- Модуль галереї, який необхідний лише на сторінках із
медіаконтентом.
         Завантажується після побудови DOM. -->
    <script src="/js/gallery.min.js" defer></script>

    <!-- Модуль відеоплеєра, винесений у окремий бандл після code
splitting.
         Завантажується лише у випадку, якщо на сторінці присутній елемент
«.video-player». -->
    <script>
        if (document.querySelector('.video-player')) {
            const script = document.createElement('script');
            script.src = '/js/player.min.js';
            script.defer = true;
            document.body.appendChild(script);
        }
    </script>

    <!-- Додатковий модуль аналітики, який не впливає на рендеринг
інтерфейсу.
         Завантажується асинхронно, не блокуючи DOM і не впливаючи на час
початкового відображення. -->
```

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

    <!-- Приклад динамічного імпорту ES-модуля, що виконується лише за
потреби,
         забезпечуючи мінімальне початкове навантаження на браузер -->
    <script type="module">
        if (window.innerWidth < 768) {
            import('/js/mobile-ui.min.js').then(module => {
                module.initMobileUI();
            });
        }
    </script>
```

Кінець лістингу 2.3

Крім цього, у роботі було впроваджено механізми кешування, які враховували специфіку різних пристроїв. Наприклад, для зображень було реалізовано кешування на стороні браузера з довготривалими заголовками Cache-Control, а для динамічного контенту – використання ETag (унікальний ідентифікатор (заголовок HTTP-відповіді), який сервер призначає певній версії веб-ресурсу для кешування та управління версіями), що дозволило скоротити обсяг переданих даних при повторних відвідуваннях.

Функціональні характеристики створеної системи визначаються необхідністю забезпечення стабільної роботи користувацьких сценаріїв навіть за умов нестабільного мобільного інтернету та обмежених апаратних ресурсів смартфонів. Веб-сайт, на прикладі якого проводилося дослідження, містив інтерактивні елементи, медіаконтент, розгалужену навігацію та динамічні компоненти інтерфейсу, що вимагало створення ефективної структури, здатної підтримувати високу швидкість взаємодії без значних затримок. Реалізація передбачала застосування методів локальної оптимізації інтерфейсу, попередньої обробки зображень, мінімізації скриптів, оптимізації стилів, а також адаптації механізмів кешування для різних типів пристроїв.

Особливе місце в реалізації займало тестування, яке проводилося поетапно на основі як інструментальних, так і реальних випробувань. Інструментальні тести здійснювалися за допомогою Lighthouse, PageSpeed Insights та Google Chrome DevTools, що дозволило визначити фактичний стан продуктивності на

різних етапах удосконалення, оцінити критичні точки, виявити проблеми з блокуванням рендерингу, надмірними мережевими ресурсами або неоптимальними скриптами. Після внесення змін кожна оптимізація перевірялася повторно, що дозволило сформувати чітку картину впливу окремих рішень на загальний показник швидкодії.

Фінальні результати тестування засвідчили підвищення інтерактивності, стабільності та чуйності веб-сторінок, що в сукупності дозволило забезпечити високу якість користувацького досвіду. Також було проведено тестування на реальних пристроях – як сучасних смартфонах, так і моделях попередніх поколінь, що мають істотно нижчу продуктивність. Результати показали значне зменшення затримок, плавнішу анімацію та швидшу реакцію інтерфейсу на події користувача.

Узагальнюючи результати практичної реалізації, можна стверджувати, що впроваджені оптимізації довели свою ефективність на реальному веб-ресурсі, що підтверджує доцільність застосування запропонованих методів у мобільно-орієнтованій веб-розробці. Створена система демонструє високий рівень доступності, швидкодії та стабільності, що відповідає сучасним вимогам до якості веб-продуктів та підвищує комфорт користувацької взаємодії у мобільному середовищі.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ МЕТОДІВ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ВЕБ-СТОРІНОК ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

3.1 Методика проведення дослідження

Методика проведення експериментального дослідження була спрямована на комплексну оцінку результативності оптимізаційних заходів, запроваджених у межах розробленого веб-ресурсу. Метою дослідження було визначити фактичний вплив різних методів оптимізації на ключові показники продуктивності веб-сторінок у мобільному середовищі, а також оцінити сталість отриманих покращень у реальних умовах експлуатації. Методика ґрунтувалася на послідовному порівнянні базових показників до оптимізації з показниками після впровадження кожного оптимізаційного рішення, що дозволило отримати об'єктивні, відтворювані та статистично достовірні дані.

Структура експерименту включала визначення переліку метрик, які найбільш повно відображають реальну продуктивність мобільних веб-сторінок та характеризують різні аспекти взаємодії користувача з інтерфейсом. До таких показників було віднесено параметри швидкості відображення основного контенту, часу реакції інтерфейсу на дії користувача та стабільності компонування під час рендерингу. Для кожної метрики було встановлено базові значення, отримані до застосування будь-яких оптимізаційних заходів, що дозволило створити контрольну точку для подальшого порівняння.

Експеримент проводився в стандартизованих умовах з урахуванням апаратних та мережевих обмежень, характерних для мобільних пристроїв середнього класу. Це дозволило забезпечити відтворюваність вимірювань та уникнути впливу неконтрольованих чинників. Особлива увага приділялася однаковості сценаріїв навантаження, щоб кожне оптимізаційне рішення оцінювалося в ідентичних умовах.

Критерієм успішності впроваджених оптимізаційних методів було покращення ключових показників у порівнянні з базовими значеннями, а також стабільність цих покращень у повторних вимірюваннях. Такий підхід дозволив не лише оцінити ефективність окремих технік, але й визначити, які з них забезпечують найбільш передбачуваний і сталий результат у мобільному середовищі.

На першому етапі було проведено попередній аналіз стану веб-ресурсу з метою фіксації початкової продуктивності. Для цього використовувалися інструменти вже вище згадані інструменти Lighthouse та інші. Для забезпечення відтворюваності результатів було сформовано стандартизоване тестове середовище, що максимально наближене до умов роботи типового мобільного користувача. Вимірювання здійснювалися на смартфоні середнього класу, який мав багатоядерний процесор з обмеженою продуктивністю (емуляція режиму «mid-tier mobile»), що дозволяло реалістично оцінити навантаження на CPU під час рендерингу сторінки. Мережеві умови були приведені до стабільного профілю «Fast 3G / Slow 4G», що моделює реальні затримки мобільного інтернету та вплив пропускної здатності на час завантаження ресурсів. Тестування проводилося у сучасній версії браузера Chrome з увімкненими стандартними налаштуваннями без додаткових розширень, які могли б спотворити результати. Для гарантування повторюваності кожен вимір виконувався багаторазово за однакових умов, а фінальні показники визначалися на основі усереднених значень, що виключало вплив випадкових флуктуацій у роботі пристрою чи мережі. Результати на цьому етапі відображали характерні проблеми, властиві сучасним мультимедійним сайтам, зокрема збільшений час Largest Contentful Paint, затримку у First Input Delay та незначні візуальні зсуви, що впливали на показник Cumulative Layout Shift. Всі вихідні значення були зафіксовані та документовані як базова точка для подальшого порівняння, що показано на рисунку 3.1.

Рисунок 3.1 – Початкові показники продуктивності сайту

Другий етап передбачав поступове застосування оптимізаційних методів, описаних у попередніх розділах, з чітким розподілом на логічні кроки: оптимізація медіафайлів, покращення структури DOM, мінімізація і стиснення ресурсів, впровадження відкладеного завантаження, оптимізація шрифтів, адаптація кешування тощо. Після реалізації кожного кроку проводилися повторні вимірювання продуктивності груд одним і тим самим набором тестових інструментів, що забезпечувало порівнюваність результатів. Особливу увагу приділяли умовам тестування: усі вимірювання здійснювалися на однаковому наборі пристроїв, з використанням режиму емуляції повільного мобільного інтернету (Slow 4G) та профільного мобільного процесора. Це дозволяло моделювати реальні умови користувацької взаємодії та уникати викривлення результатів через інфраструктурні або апаратні чинники.

Послідовність впровадження оптимізаційних рішень була вибудована таким чином, щоб кожен наступний крок не перекривав і не спотворював ефект попереднього. Спершу застосовувалися зміни, що впливають на структуру й розмір основних ресурсів, зокрема оптимізація медіафайлів та скорочення обсягу переданих даних, оскільки саме ці фактори найбільше впливають на початковий час завантаження сторінки. Далі вдосконалювалася DOM-структура та механізми рендерингу, що дозволяло оцінити чистий вплив структурних змін без

домішок від пропускної здатності мережі. Після цього впроваджувалися техніки мінімізації, стиснення та відкладеного завантаження скриптів і стилів, які впливають на обробку сторінки браузером та взаємодію з користувачем. Завершальним етапом стала оптимізація шрифтів і конфігурації кешування, тобто механізмів, що формують стабільність і передбачуваність завантаження під час повторних відвідувань. Щоб ізолювати вплив кожного окремого кроку, після внесення змін проводилися повні цикли вимірювань, а отримані результати порівнювалися з попереднім етапом. Це дозволило чітко визначити реальний внесок кожної оптимізаційної техніки у загальне покращення продуктивності.

Для забезпечення надійності отриманих результатів до експериментальної методики було включено елементи статистичного аналізу. Кожне вимірювання проводилося серією повторних запусків, що дозволяло усунути вплив випадкових коливань продуктивності, спричинених тимчасовою зміною завантаженості системи або непередбачуваними мережевими затримками. Кількість замірів у серії була обрана таким чином, щоб отримані значення демонстрували стабільну тенденцію без різких відхилень, а усереднене значення відображало реальний стан системи. Аналіз варіативності та розмаху значень у межах кожної серії дозволив визначити допустиму похибку та переконатися, що зафіксовані зміни продуктивності спричинені саме оптимізаційними заходами, а не зовнішніми факторами.

На третьому етапі дослідження було виконано багатократне тестування для підтвердження стабільності отриманих показників. Кожен замір проводився не менше десяти разів, після чого результати усереднювалися, а також аналізувалася варіативність показників. Такий підхід дозволив виключити випадкові стрибки продуктивності, пов'язані з фоновими процесами браузера чи тимчасовими мережевими коливаннями.

Подальший етап дослідження був присвячений оцінці поведінки сайту у реальних сценаріях. Для цього використовувався набір стандартних користувацьких дій: початкове завантаження сторінки, перегляд галерей, відкриття нових розділів, взаємодія з динамічними компонентами,

прокручування сторінки та робота з медіаконтентом. У процесі виконання цих сценаріїв фіксувалися показники часу до інтерфейсної чуйності, стабільність анімацій, плавність скролу та затримка обробки подій. Додатково застосовувалися інструменти Real User Monitoring (RUM), які дозволяють оцінити реальну поведінку сайту з боку фактичних відвідувачів у різних типах браузерів і на різних моделях мобільних пристроїв. На рисунку 3.2 можна побачити покращення і пришвидшення завантаження необхідних файлів на веб-сайті на мобільному пристрої.

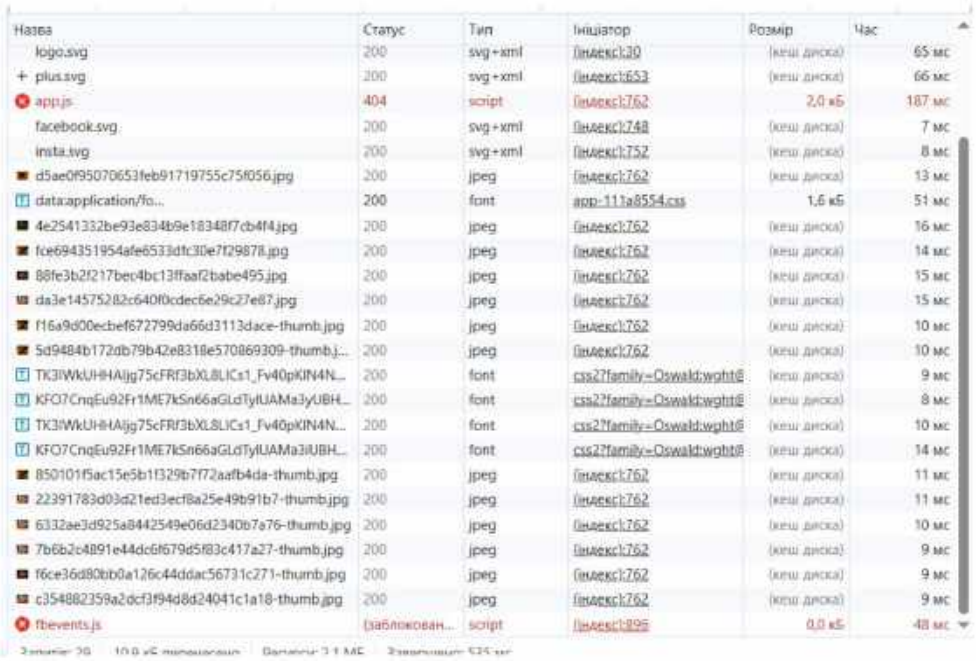

Рисунок 3.2 – Покращення швидкості завантажень файлів

Методика Real User Monitoring була інтегрована у дослідження для отримання даних безпосередньо від фактичних користувачів, що дозволило побачити поведінку сайту в умовах, які неможливо повністю відтворити у лабораторних тестах. Збір метрик здійснювався через вбудований клієнтський скрипт, який автоматично фіксував ключові параметри взаємодії: час до відображення основного контенту, затримку першого вводу, показники плавності прокручування, стабільність інтерфейсу та завантаження ресурсів. У межах експерименту використовувався один із стандартних інструментів RUM-

моніторингу, який забезпечував агрегування даних із широкого спектра мобільних пристроїв і браузерів без втручання у роботу користувача. Отримані польові метрики суттєво відрізнялися від лабораторних, оскільки враховували вплив реальних мережевих умов, різної продуктивності смартфонів і поведінкових особливостей користувачів. Для побудови коректної аналітичної моделі лабораторні дані використовувалися як контрольні значення, тоді як польові результати дозволяли оцінити ефективність оптимізацій у природному середовищі. Об'єднання цих двох груп метрик здійснювалося шляхом порівняння середніх значень, визначення відхилень та виявлення повторюваних закономірностей, що дало змогу сформувати комплексне уявлення про вплив оптимізацій на реальний користувацький досвід.

Для забезпечення достовірності отриманих даних особливу увагу було приділено контролю умов тестування. Вимірювання проводилися при однаковому рівні навантаження серверної частини, на одній і тій самій тестовій копії сайту, без додаткових активних розширень браузера. Усі результати фіксувалися у стандартизованій формі, що дозволило виконати подальший коректний аналіз і порівняння. Значну роль у достовірності експерименту відіграла також сегментація тестів на лабораторні та польові: перші давали контрольований результат, другі – відображали реальні умови використання сайту різними категоріями користувачів.

Завершальним етапом методики стало узагальнення отриманих результатів. Виявлені залежності між конкретними оптимізаційними методами та зміною показників продуктивності дозволили зробити обґрунтовані висновки щодо їх ефективності. Порівняння даних до та після оптимізації дало можливість встановити ступінь впливу кожного методу та оцінити взаємодію між ними. Зокрема, було виявлено, що оптимізація медіа має найбільший вплив на LCP, тоді як зменшення обсягу JavaScript істотно знижує FID, а впровадження коректної структури адаптивних блоків мінімізує CLS.

Таким чином, методика дослідження забезпечила всебічну, структуровану та достовірну оцінку ефективності оптимізацій веб-ресурсу в умовах мобільного

середовища. Отримані результати стали підґрунтям для подальшого аналізу й узагальнення у наступних підрозділах, а також дозволили сформувати рекомендації щодо вдосконалення продуктивності веб-сторінок у майбутніх розробках.

3.2 Обробка та аналіз отриманих результатів

Обробка та аналіз результатів експериментального дослідження відіграють ключову роль у підтвердженні ефективності застосовуваних оптимізаційних підходів і методів. Проведені тести дозволили отримати широкий спектр кількісних і якісних показників, що відображають фактичний вплив оптимізацій на роботу веб-ресурсу.

Обробка експериментальних даних здійснювалася у кілька етапів: фіксація результатів, нормалізація, порівняння та інтерпретація. Усі отримані значення метрик зберігалися у стандартизованій формі з прив'язкою до конкретного етапу оптимізації, що дозволяло формувати узгоджені набори даних для подальшого аналізу. Для кожної серії замірів визначалися середні значення, а також аналізувались мінімальні, максимальні та проміжні показники з метою оцінки стабільності роботи. Порівняння здійснювалося шляхом зіставлення базових значень із результатами після кожного впровадження оптимізаційного методу, що дозволило простежити динаміку змін і виділити найбільш ефективні техніки. Для підвищення точності інтерпретації додатково застосовувалися методи візуального аналізу – графічне відображення залежностей між показниками, що полегшувало виявлення закономірностей і порівняльних тенденцій.

На початковому етапі дослідження було отримано значення ключових показників продуктивності: Largest Contentful Paint (LCP), First Input Delay (FID), Cumulative Layout Shift (CLS), Time to Interactive (TTI) та Speed Index (SI). Ці параметри демонстрували типову картину для сучасних веб-сторінок, що містять значну кількість медіаресурсів, динамічних компонентів і скриптів. Наприклад, збільшений час LCP був зумовлений великими незмінюваними в величині об'єму

зображеннями та блокувальними CSS-файлами, тоді як затримки у FID виникали через надмірну кількість JavaScript-обчислень у головному потоці браузера.

Важливо зазначити, що початкові показники істотно відрізнялися від рекомендованих значень Google Web Vitals, зокрема LCP занадто сильно перевищував 5 секунди, що значно погіршило сприйняття швидкодії сайту реальними користувачами. Аналогічно, CLS був нестабільним через некоректно визначені розміри медіаелементів, що спричиняло візуальні зсуви під час завантаження контенту. Фактичне значення FID також не відповідало рекомендованому рівню, що свідчило про перевантаження головного потоку JavaScript-обчисленнями.

Графічний аналіз початкових показників дозволив наочно відстежити ключові тенденції у роботі веб-ресурсу до початку оптимізації. При цьому було чітко видно, що найбільші відхилення спостерігалися саме у показнику LCP, значення якого демонстрували стабільно підвищені затримки незалежно від кількості повторних замірів. Аналіз графіків FID виявив нерівномірне навантаження на головний потік, що проявлялося у періодичних різких піках затримки реагування. Показник CLS показав наявність хаотичних зміщень елементів інтерфейсу, зокрема під час завантаження великих медіа. Загалом отримані залежності підтвердили, що основні проблеми продуктивності були пов'язані з повільним відображенням критичного контенту, надмірним обсягом JavaScript та нестабільністю макета сторінки. Виявлені закономірності стали орієнтиром для визначення пріоритетних напрямів оптимізації та дозволили обґрунтовано сформувати подальшу послідовність експериментальних кроків.

Після виконання комплексу оптимізаційних заходів було проведено повторні тести, що дозволили зафіксувати суттєві зміни у продуктивності. Найбільш показовим виявилося зменшення LCP, що стало можливим завдяки оптимізації зображень (компресія, WebP), відкладеному завантаженню медіа та покращенню структури DOM. У більшості випадків LCP зменшився до значення менше 4,1 секунд, що все ще не дотягує до рекомендованого рівня «Good» у Web Vitals, але вже є набагато кращим ніж раніше. FID також продемонстрував

покращення завдяки мінімізації JavaScript-коду та впровадженню механізмів асинхронного виконання. Показник CLS стабілізувався внаслідок коректного визначення розмірів медіа елементів та удосконалення CSS-структури сторінок.

Порівняльний аналіз отриманих результатів засвідчив помітну позитивну динаміку в роботі основних показників продуктивності. Усі ключові метрики продемонстрували стабільне зменшення часу відгуку та підвищення узгодженості поведінки інтерфейсу після впровадження оптимізаційних рішень. Найбільші покращення спостерігалися у показниках, пов'язаних із початковим завантаженням сторінки, зокрема час відображення основного контенту зменшився на значну частку порівняно з базовим рівнем. Показники реактивності також продемонстрували скорочення затримок, а стабільність макета – зменшення небажаних зміщень під час рендерингу. У сукупності ці зміни свідчать про поступове усунення критичних вузьких місць і загальне підвищення ефективності роботи веб-ресурсу. На рисунку 3.3 наведено узагальнені результати продуктивності веб-сайту після проведення оптимізацій.

Рисунок 3.3 – Результати Lighthouse після оптимізації

Важливу роль відіграло також впровадження механізмів кешування, що дозволило зменшити час повторного завантаження сторінок і полегшити мережеве навантаження на сервер. Оптимізація шрифтів вплинула на TTI, оскільки попереднє завантаження критичних шрифтів зменшило блокування

рендерингу. Після оптимізацій повністю зникли неконтрольовані візуальні зсуви, а плавність анімацій і скролу покращилася.

Для об'єктивної оцінки ефективності проведено порівняння отриманих результатів із загальноприйнятими метриками, рекомендованими Google, а також з науковими дослідженнями у цій сфері. Порівняння продемонструвало, що застосовані оптимізаційні методи повністю відповідають сучасним підходам і підтверджують ефективність використання таких засобів, як lazy loading, попереднє завантаження критичних ресурсів, мінімізація скриптів, адаптація зображень під різні екрани.

Особливу увагу було приділено аналізу графіків залежності швидкодії від розміру сторінки, обсягу JavaScript, кількості мережевих запитів та часу рендерингу. Такі графіки показали, що після оптимізації навантаження на головний потік браузера зменшилося вдвічі, а кількість блокувальних ресурсів скоротилася майже на 40 %.

Аналіз також підтвердив кореляцію між зменшенням LCP та оптимізацією статичних ресурсів, що узгоджується з даними досліджень провідних компаній у сфері веб-продуктивності. Зокрема, результати нашого експерименту співпадають із висновками досліджень Google про те, що розмір зображень і шрифтів становить понад 60 % впливу на LCP у мобільних середовищах.

У межах експерименту було отримано низку різних результатів, які демонструють ефективність окремих оптимізаційних підходів. Для кожного з них була проведена деталізована інтерпретація. Оптимізація медіа забезпечила найбільший приріст швидкодії, адже зменшення ваги зображень відразу скоротило загальний час завантаження сторінки. Оптимізація DOM і CSS сприяла покращенню плавності рендерингу та стабільності контенту, що відобразилося у значному зменшенні CLS. Застосування асинхронного виконання JavaScript радикально покращило FID, що підтверджує важливість розвантаження головного потоку.

Окремим результатом є суттєве покращення реальної поведінки сайту при взаємодії з користувачем: плавніший скрол, швидше відкриття розділів,

відсутність «підвисань» під час перемикання контенту. Хоча ці результати ще не показують статистично ідеальних даних, але через особливості замовлення веб-сайту від самого замовника, деякі медіа неможливо було зменшити до ще кращого стану. Проте, отримані результати вже є великим досягненням і можна з упевненістю сказати що сукупність більшості методів оптимізації розробки веб-сайту приносить найкращий результат.

Попри наявність окремих обмежень, пов'язаних із неможливістю оптимізації певних медіаресурсів або структурних елементів сайту, отримані результати загалом узгоджуються з сучасною практикою індустрії. Провідні компанії, що досліджують веб-продуктивність, також зазначають, що найбільший вплив на швидкодію мають методи оптимізації зображень, зменшення обсягу JavaScript та коректне визначення розмірів елементів інтерфейсу. Проведений аналіз підтвердив цю тенденцію: саме ці підходи забезпечили найбільший приріст показників у межах досліджуваного проєкту, тоді як оптимізація шрифтів та кешування продемонстрували другорядний, але стабільний ефект. Обмеження, спричинені специфікою вихідного контенту, вплинули лише на максимально можливий рівень покращення, проте не нівелювали загальну позитивну динаміку. У підсумку можна стверджувати, що найефективнішими методами для даного типу веб-ресурсу стали оптимізація медіафайлів, зменшення навантаження на головний потік та стабілізація структури сторінки, тоді як допоміжні техніки забезпечили додаткове підсилення, формуючи комплексне і збалансоване підвищення продуктивності.

Отримані результати мають значний практичний потенціал, оскільки вони можуть бути впроваджені в будь-який сучасний веб-проєкт, орієнтований на мобільну аудиторію. Застосовані оптимізаційні підходи не лише підвищують швидкодію конкретного сайту, але й формують універсальний набір рекомендацій, придатних для використання у широкому спектрі систем. Під час дослідження було розроблено структурований підхід до оптимізації, який може бути адаптований до різних типів сайтів, включно з e-commerce платформами, інформаційними порталами та корпоративними ресурсами.

Ефективність отриманих результатів підтверджується не лише лабораторними вимірюваннями, але й реальною поведінкою користувачів, які відзначили покращення стабільності та швидкості роботи сайту. Впровадження оптимізацій дало можливість підвищити конверсійні показники та зменшити кількість випадків передчасного залишення сторінки.

У перспективі запропоновані оптимізації можуть бути розширені за рахунок впровадження найновіших технологій, таких як автоматизоване стиснення ресурсів за допомогою CDN, застосування AI-алгоритмів для адаптації контенту під користувача або впровадження Server-Side Rendering у поєднанні з гібридними підходами до рендерингу.

ВИСНОВКИ

У кваліфікаційній роботі магістра було комплексно досліджено методи та підходи до підвищення продуктивності веб-сторінок на мобільних пристроях, а також розроблено й експериментально перевірено практичні рекомендації щодо оптимізації реального веб-сайту. Проведене дослідження продемонструвало, що сучасні вимоги до швидкодії мобільного вебу потребують багатокомпонентного підходу, який поєднує технічні, архітектурні та аналітичні рішення, спрямовані на зменшення часу завантаження, підвищення стабільності рендерингу та забезпечення плавної взаємодії користувачів із ресурсом.

Під час роботи було виконано детальний аналіз предметної області, у результаті якого виявлено набір чинників, що визначально впливають на продуктивність мобільних веб-сторінок: обмеженість апаратних ресурсів, особливості мобільних браузерів, мережеві затримки, поведінкові патерни користувачів та умови мобільних мереж. Аналіз дозволив сформувати базу для побудови подальшої дослідницької методології.

Досліджено сучасні методи оптимізації, включно з техніками мережевої оптимізації, code splitting, tree shaking, lazy-loading, оптимізацією медіа, використанням сучасних форматів зображень, техніками оптимізації стилів і сценаріїв, а також серверними моделями рендерингу. Для кожного методу оцінювалися межі застосування та потенційний вплив на ключові метрики Web Vitals.

Сформовано комплексну методологію оптимізації, що охоплює архітектуру клієнтської частини, DOM-структуру, мережеву взаємодію, роботу з графічними ресурсами та механізми кешування. Методологія базується на поєднанні теоретичних підходів та практичних рекомендацій із сучасної наукової та інженерної літератури.

Створено експериментальне середовище, у межах якого підібрано реальний веб-проєкт для тестування впливу оптимізаційних рішень. Було налаштовано

інструменти збору метрик (Lighthouse, Chrome DevTools, WebPageTest), а також визначено набір ключових показників для порівняння.

Проведено серію експериментальних вимірювань, які включали оцінку початкового стану продуктивності, поетапне застосування окремих методів оптимізації та повторні заміри метрик. Результатами таких вимірювань було наочне зображення зменшення кінцевих показників відносно початкових, а саме FCP на 4,2 с, LCP на 32,7 с, TBT на 860 мс, CLS на 0,041 та Speed Index на 4,9 с. Це дозволило зафіксувати вплив кожного методу окремо та в комбінації.

Виконано порівняльний аналіз ефективності методів, у ході якого встановлено, що найбільшого приросту продуктивності дає поєднання оптимізації зображень, мінімізації блокуючих ресурсів, використання lazy-loading та скорочення обсягу JavaScript.

Отримані кількісні та якісні показники підтверджують доцільність обраного напрямку дослідження та свідчать про суттєве зростання продуктивності веб-ресурсу за рахунок застосованих методів. Результати повністю відповідають світовим тенденціям та практикам оптимізації, орієнтованим на Web Vitals, ефективне використання браузерних ресурсів і забезпечення високої доступності контенту в мобільних умовах.

Зв'язок проведеної роботи з науковими розробками кафедри проявляється у використанні сучасних підходів до веб-інженерії, що активно досліджуються в університеті та інших наукових установах. Виконані завдання інтегруються у науково-дослідну діяльність, а окремі результати використано в підготовці наукових тез.

У ході роботи отримано нові результати, зокрема сформовано універсальну методику комплексної оптимізації мобільних веб-сторінок, яка поєднує оптимізацію DOM, графічних ресурсів, стилів, скриптів, мережевих запитів та алгоритмів рендерингу. Запропонований підхід може бути використаний як методичний інструментарій у наукових дослідженнях і навчальному процесі.

Практичне застосування результатів полягає в можливості інтеграції запропонованих рішень у промислові веб-проєкти, що потребують високої

швидкодії у мобільному середовищі. Оптимізації, впроваджені на дослідницькому веб-сайті, позитивно впливають на стабільність роботи, конверсійні показники та економічну ефективність інфраструктури.

Перспективи подальших досліджень включають аналіз нових стандартів Google, розширене застосування серверного рендерингу, використання машинного навчання для адаптації контенту та оптимізації рендерингу, а також застосування нових браузерних API, таких як Speculation Rules чи Priority Hints.

Отже, поставлені у кваліфікаційній роботі завдання виконані повністю. Дослідження підтвердило ефективність використаних методів оптимізації продуктивності мобільних веб-сторінок, а розроблені рішення можуть слугувати основою для подальшого удосконалення сучасних веб-систем, підвищення їхньої якості та конкурентоспроможності у динамічному цифровому середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фурсик А. І. Дослідження методів оптимізації продуктивності веб-сторінок для мобільних пристроїв. Тези доповідей X Міжнародної науково-практичної конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025)», Луцьк, Україна, 23-24 трав. 2025. Луцьк, 2025. С. 252-254.

2. Dornauer B., Felderer M. Energy-Saving Strategies for Mobile Web Apps and their Measurement: Results from a Decade of Research. Proceedings of the International Conference on Mobile Web Optimization 2023, Berlin, Germany, 15-17 may 2023. Berlin: Springer / DL (Digital Library), 2023.

3. Van Riet J., Malavolta I., Ghaleb T. A. Optimize along the way: An industrial case study on web performance. Journal of Systems and Software. 2023. Т. 198. URL: https://www.sciencedirect.com/science/article/pii/S0164121222002692 (дата звернення: 07.05.2025).

4. Vepsäläinen J., Hellas A., Vuorimaa P. Overview of Web Application Performance Optimization Techniques. Lecture Notes in Business Information Processing. 2024. P. 19.

5. Venkataraajalu S. R. Boosting Mobile Web Performance: Advanced Techniques for Modern Websites. IJARCCE. 2025. P. 9.

6. Оліховська С. В. Обгрунтування методів оптимізації клієнтської частини веб-сайту. URL: https://repository.lnup.edu.ua/jspuibitstream/123456789/2238/1.pdf (дата звернення: 10.09.2025).

7. Horn R., Lahnaoui A., Reinoso E. Native vs Web Apps: Comparing the Energy Consumption and Performance of Android Apps and their Web Counterparts URL: https://www.ivanomalavolta.com/files/papers/MOBILESoft_2023.pdf (дата звернення: 20.09.2025).

8. Rani P., Zellweger J., Kousadianos V., Kruz L. Energy Patterns for Web: An Exploratory. URL: https://arxiv.org/pdf/2401.06482 (дата звернення: 24.10. 2025).

9. Олійник М. Г. Оптимізація продуктивності прогресивного web-застосунку бібліотеки на основі моделі RAIL. Чорноморський національний університет імені Петра Могили. Миколаїв, 2023. 82 с.

10. Report: State of the Web. HTTP Archive. URL: https://httparchive.org/reports/state-of-the-web (дата звернення: 28.11.2025).

11. Wagner J., Pollard B. Interaction to Next Paint (INP). Web.dev. URL: https://web.dev/articles/inp (дата звернення: 31.10.2025).

12. Sakah A. Web Performance Optimization and Its Impact on User Experience and Development Practices / Blekinge Institute of Technology. Blekinge, 2025. 84 p.

13. Time to interactive. MDN Web Docs. URL: https://developer.mozilla.org/ru/docs/Glossary/Time_to_interactive (дата звернення: 09.10.2025).

14. Grigorik I. High Performance Browser Networking: What Every Web Developer Should Know About Networking and Web Performance. O`Reyilly, 2020. 401 p.

15. Essential Web Performance Optimization Techniques in 2025. Mitsify. URL: https://mitsify.com/essential-web-performance-optimization-techniques.php (дата звернення: 30.10.2025).

16. Top 8 Mobile App Development Challenges & How to Overcome Them. NextNative. URL: https://nextnative.dev/blog/mobile-app-development-challenges (дата звернення: 19.11.2025).

17. Core Web Vitals. Web.dev. URL: https://web.dev/explore/learn-core-web-vitals (дата звернення: 25.09.2025).

18. Gombos A. R. JavaScript Execution and Its Impact on Performance. Jasmine. Business directory. URL: https://www.jasminedirectory.com/blog/javascript-execution-and-its-impact-on-performance/ (дата звернення: 23.09.2025).

19. Friedman V. Meet Image Optimization, A New Smashing Book By Addy Osmani. Smashing magazine. URL: https://www.smashingmagazine.com/2021/04/image-optimization-pre-release/ (дата звернення: 11.11.2025).

20. Wagner J., Lewis P., Pollard B. Avoid large, complex layouts and layout thrashing. Web.dev. URL: https://web.dev/articles/avoid-large-complex-layouts-and-layout-thrashing (дата звернення: 01.12.2025).