



## **EVALUATING WORDPRESS OPTIMIZATION PLUGINS AND TECHNIQUES**

Lappeenranta–Lahti University of Technology LUT

Bachelor's Programme in Software Engineering, Bachelor's thesis

2025

Oliver Kuosmanen

Examiner: Associate Professor Ari Happonen

## ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Sciences

Software Engineering

Oliver Kuosmanen

### **Evaluating WordPress optimization plugins and techniques**

Bachelor's thesis

2025

54 pages, 3 figures, 7 tables and 4 appendices

Examiner: Associate Professor Ari Happonen, D.Sc. (Tech.)

Keywords: optimization, WordPress, plugin, website, web performance optimization, caching, Lighthouse, performance testing

WordPress is the most popular web content management system in the world by market share. While WordPress itself is free to use, cloud computing resources are not. To efficiently utilize these resources, optimizations should be applied to websites created with WordPress.

The aim of this bachelor's thesis is to evaluate the usefulness of WordPress optimization plugins. A quantitative experimental research methodology is used to measure the impact of optimization on typical WordPress sites.

Measurement results indicate that WordPress optimization plugins can improve the performance of websites. Additionally, installing two plugins with complementary features yielded even greater performance results. Furthermore, the features of the plugins were compared to web optimization techniques mentioned in prior research, and they largely aligned.

The technical implementations of web optimization strategies are not explored in this thesis but could be a subject for future research. Although optimization could improve server efficiency, its effects should be evaluated on a case-by-case basis through testing.

## TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUTin insinööritieteiden tiedekunta

Tietotekniikka

Oliver Kuosmanen

### **WordPressin optimointilisäosien ja -menetelmien arviointi**

Tietotekniikan kandidaatintyö

2025

54 sivua, 3 kuvaa, 7 taulukkoa ja 4 liitettä

Tarkastaja: Tutkijaopettaja Ari Happonen (TkT)

Avainsanat: optimointi, WordPress, lisäosa, verkkosivusto, verkkosivuston suorituskyvyn optimointi, välimuistiin tallentaminen, Lighthouse, suorituskykytestaus

WordPress on maailman suosituin verkkosisällönhallintajärjestelmä markkinaosuudella mitattuna. Vaikka WordPress on itsessään ilmainen, pilvipalveluiden resurssit eivät ole. Näiden resurssien tehokkaan hyödyntämisen mahdollistamiseksi WordPressillä luotuja verkkosivustoja tulisi optimoida.

Tämän kandidaatintyön tavoitteena on arvioida WordPress-optimointilisäosien hyödyllisyyttä. Kvantitatiivista kokeellista tutkimusmenetelmää käytetään mittaamaan optimointien vaikutuksia tyypillisiin WordPress-sivustoihin.

Mittaustulokset osoittavat, että WordPress-optimointilisäosat voivat parantaa verkkosivustojen suorituskykyä. Lisäksi kahden toisiaan täydentävän lisäosan asentaminen tuotti vieläkin parempia suorituskykytuloksia. Lisäosien ominaisuuksia verrattiin aiemmissa tutkimuksissa mainittuihin verkkosivujen optimointimenetelmiin, ja ne vastasivat suurelta osin toisiaan.

Verkkosivustojen optimointistrategioiden teknisiä toteutuksia ei käsitellä tässä kandidaatintyössä, mutta ne voisivat olla tulevaisuuden tutkimuksen kohteena. Vaikka optimointi voisi parantaa palvelimen tehokkuutta, sen vaikutuksia tulisi arvioida tapauskohtaisesti testaamalla.

## ACKNOWLEDGEMENTS

I would like to thank Parvus Vulpes Oy for commissioning this bachelor's thesis and for their integral role in making the experimental work possible by providing demonstration examples of WordPress websites developed by the company.

## SYMBOLS AND ABBREVIATIONS

### Dimensionless quantities

p            probability

### Abbreviations

API            Application Programming Interface

CDN            Content Delivery Network

CLS            Cumulative Layout Shift

CMS            Content Management System

CSS            Cascading Style Sheets

CSV            Comma-Separated Values

DNS            Domain Name System

FCP            First Contentful Paint

HTML          Hypertext Markup Language

HTTP          Hypertext Transfer Protocol

INP            Interaction to Next Paint

LCP            Largest Contentful Paint

LSWS          LiteSpeed Web Server

PHP            Hypertext Preprocessor

QoS            Quality of Service

RAM            Random-Access Memory

REST          Representational State Transfer

SEO            Search Engine Optimization

TBT            Total Blocking Time

WCMS          Web Content Management System

WPO            Web Performance Optimization

## Table of contents

Abstract

Acknowledgements

Symbols and abbreviations

1	Introduction .....	9
2	Related research on optimization .....	13
2.1	Web performance optimization .....	13
2.1.1	Static asset optimization .....	13
2.1.2	Server- and browser-side caching .....	15
2.1.3	Other optimization strategies which affect performance testing .....	16
2.2	Analysis of the research .....	17
3	Research methodology .....	20
3.1	Methodology .....	20
3.2	The setup of the experiment .....	20
3.3	The measurement tool, metrics and procedures .....	22
4	Technical research on WordPress optimization plugins .....	25
4.1	Conducting the experiment .....	25
4.1.1	Selection of the plugins and setting up the testing environment .....	25
4.1.2	Auditing with Lighthouse .....	26
4.2	Results of performance testing .....	27
4.2.1	Results of optimizing without caching .....	27
4.2.2	Results of optimizing with caching enabled .....	30
4.2.3	Optimizing the sites based on the previous results .....	31
5	Discussion .....	33
5.1	Comparing the plugins' optimization techniques to prior WPO Research .....	33
5.2	What do the test results mean? .....	35
5.2.1	Why do the results on the sites differ from one another? .....	35
5.2.2	Differences between plugin performance .....	36
5.2.3	The effects of using two optimization plugins at once .....	37
5.2.4	Mitigating the server-side performance loss caused by optimization plugins .....	37
5.3	Limitations and potential future research opportunities .....	38
6	Conclusions .....	41
	References .....	43

## Appendices

Appendix 1. Table of the optimization plugins' features and their cost

Appendix 2. Box and whisker chart of the test results for website 1

Appendix 3. Box and whisker chart of the test results for website 2

Appendix 4. Box and whisker chart of the test results for website 3

## Figures

Figure 1: Graph of WCMS usage statistics on websites (W3Techs 2025)

Figure 2: The command line input for running audits with Lighthouse

Figure 3: Performance score differences relative to the baseline

## Tables

Table 1: Optimization plugin ratings

Table 2: Results for website 1

Table 3: Results for website 2

Table 4: Results for website 3

Table 5: Results for website 1 with caching enabled

Table 6: Results of testing with both Speed Optimizer and Smush enabled

Table 7: Results of performance testing a large and dynamic site

# 1 Introduction

Digital transformation is changing how businesses operate by integrating technology-based solutions that induce organisational change, but it is also intertwined with technological adoption (Judijanto 2025). This technological adoption, commonly referred to as digitalization, is pushing both traditional industry and technology sectors of the economy toward more knowledge-driven service models, where data is not only collected but used to enhance company decision-making and create new value. As a consequence of this business transition, more capacity is needed for analytical review and analysis of large amounts of data. (Kortelainen et al. 2019) At the same time, it is becoming increasingly expected for service providers to offer a variety of digital content. The ability to manage and host such a diverse fleet of content is essential to maintaining a competitive advantage in the knowledge-driven information technology sector (Kinnunen et al. 2019). Consequently, there is a growing need for content management systems (CMS) in the business space.

Digitalization impacts a wide range of management systems, including those related to human resources (Vatousios & Happonen 2022), asset management (Usmani et al. 2023), warehouse management (Minashkina & Happonen 2021), change management (Salmela & Happonen 2021), and so on. This, in turn, is also pushing internal and external company CMSs to evolve (Kumar et al. 2024; Saravanan et al. 2024). As the different requirements for CMSs increase, performance may suffer due to a greater workload, which could prove difficult for business managers and content providers to handle.

Mobile applications have simultaneously become more effective at maintaining high customer engagement and experience than desktop services. Customers now expect the same seamless experience when interacting with content-providing platforms, regardless of the medium. (Khan et al. 2023) This poses a challenge for content providers on the web.

This may be why the use of web content management systems (WCMS) to serve internet users dynamic webpages has garnered more popularity in recent times. Over the past decade, the use of a WCMS has become more common than using none. The top five most widely adopted WCMSs today are WordPress, Shopify, Wix, Squarespace, and Joomla. Among these, WordPress has the largest market share by far. WordPress is used on 43.6% of all websites, while 29.6% do not use any type of WCMS. The remaining WCMSs trail far

behind WordPress in installations, as of January 1<sup>st</sup>, 2025. (W3Techs 2025) This trend is illustrated in Figure 1, where the usage was measured once each year on January 1<sup>st</sup>.

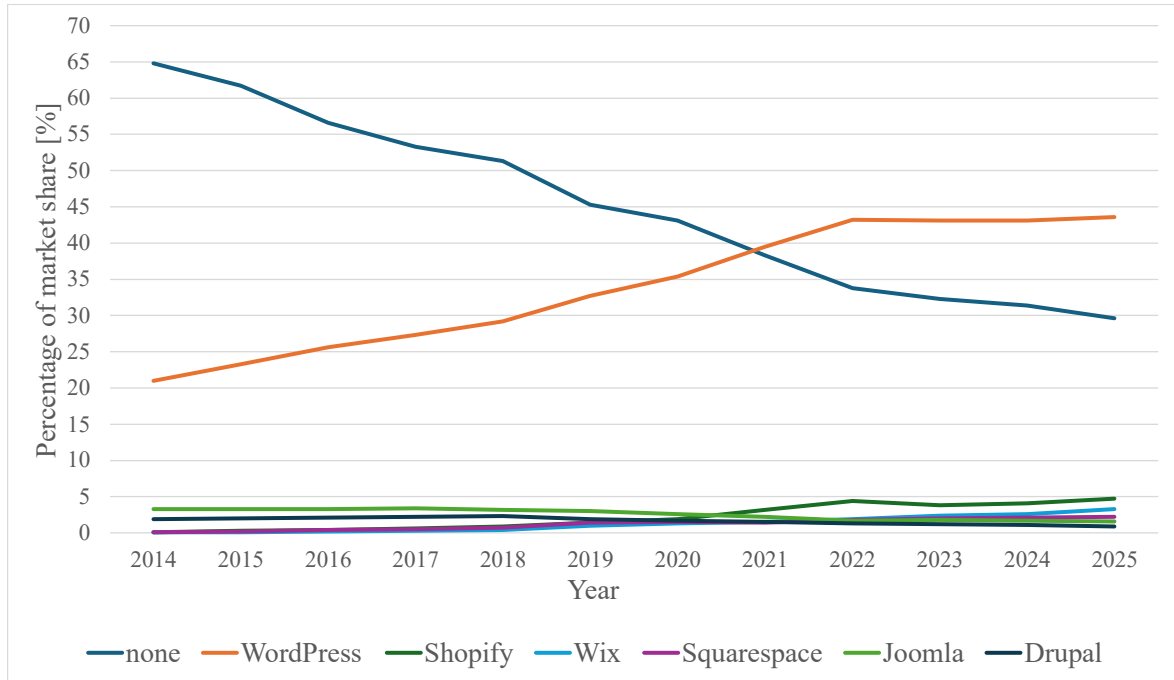


Figure 1. Graph of WCMS usage statistics on websites (W3Techs 2025)

The rising popularity of WordPress can be attributed to several factors. At its core, the WordPress WCMS is a free-to-use open-source tool that simplifies the dynamic web content creation and management process, so even users without a comprehensive set of technical skills can use it to alter website content fast and easily (Patel et al. 2011b; Kumar et al. 2021; Khalil et al. 2024). In other words, programming knowledge of languages such as Hypertext Preprocessor (PHP), which WordPress uses, is not necessary (Hakimi & Rahman 2021). Developers can also easily create WordPress posts, which are dynamic content entries that appear on their site (Schäferhoff 2025). As with all WCMSs, the benefits of having such a system, which does not require much in terms of implementation, come with the drawback of not being able to fully customise everything (Barker 2016, 18–19).

Drawbacks of WordPress are not limited to the customisability. The research by Ferreira et al. (2025) concludes that the monolithic architecture of WordPress causes inflexibility. The researchers propose a new framework to address this issue. A paper by Kazemi et al. (2024) proposes a conceptual framework for a web-based platform designed to limit overdose risk

amongst opioid addicts to reduce fatalities and support their recovery. These studies go to show that WordPress is still an evolving tool that is being worked on to solve issues of today.

Another relevant issue in the realm of web-based applications is server cost and energy consumption (Yan et al. 2016). In cloud computing environments the customer, who's hosting the content on the web, pays in accordance with the number of computational resources and their selected pricing model. The customer essentially rents computing resources, such as the amount of random-access memory (RAM) addressable in the system and gets charged an hourly rate to utilize those resources. (Arshad et al. 2015) Some cloud services, such as Microsoft Azure, offer pricing models that support dynamic resource allocation based on real-time demand (Shethiya 2025). Therefore to get the most value out of the rented resources, the amount of computation should be minimized, while still providing the same quality of service (QoS) (Everman & Zong 2018).

The underutilization of high-end server computing resources creates unnecessary costs, which can be minimized by employing various optimization techniques. These optimizations improve the server's performance relative to the cost, because the website can run on less expensive hardware. The QoS is also positively impacted by optimization, since page performance directly affects the user experience. (Everman & Zong 2018; Shailesh & Suresh 2017)

Previous research on WCMS optimization has focused on comparing the speed, performance, or features of different WCMSs such as WordPress, Joomla and Drupal (Patel et al. 2011a; Yanney et al. 2023; Khalil et al. 2024). Although there has been research done on WCMS optimization techniques and best practices, there is a lack of quantitative studies focusing on WordPress performance optimization tools. These tools are known as plugins, which expand the functional capabilities of WordPress websites (Kumar et al. 2021; Cigoj & Blazic 2019). For example, a popular optimization plugin is WP-Optimize, which is a WordPress plugin with over a million installations that is marketed as an easy-to-use tool for the purpose of optimizing overall website performance by various means (TeamUpdraft 2025). On the other hand, the large number of plugins available to install at the click of a button inherently introduces security risks. Mesa et al. (2018) conjecture that some plugin developers might not be skilled enough to catch potential security risks before they are introduced into any codebase they are writing.

Since WordPress is the largest WCMS by market share, this thesis focuses on its optimization techniques, plugins, and best practices. As one of the limitations, this thesis will not dive into the intricate technicalities related to the functional aspects of these plugins. The purpose of this thesis is instead to empirically study the effects of the plugins to assess their practical benefits. As previously discussed, optimization has a direct impact on energy efficiency, QoS, and cost. Still, it is unclear how WordPress optimization plugins affect these factors in a typical use case. The goal of this thesis is to find out the value of these plugins and determine whether the methods they use are based on best practices outlined in research.

The research questions of this bachelor's thesis are as follows:

- How do WordPress optimization plugins affect the average website?
- Do the plugins adhere to the strategies and best practices outlined by previous research?
- Are the plugins useful enough to justify their installation?

The second chapter of this thesis describes concepts, techniques, strategies, and best practices of web performance optimization (WPO) in prior research. Different techniques are discussed, examined, and some are linked to the context of WordPress. The third chapter outlines the research methodology and tackles the practical aspects of conducting the experiment. The fourth chapter documents the process of empirically measuring the changes in performance due to different WordPress optimization plugins and techniques, after which the results are presented. In the fifth chapter, the results of the technical research are analysed and the optimization methods used by the plugins are compared against the strategies explored in chapter two, thereby answering the research questions. Additionally, the limitations and possible future research possibilities are laid out. Finally, the main findings and their potential uses in future research are summarised in the sixth chapter of the thesis.

## 2 Related research on optimization

This chapter covers previous related research on web optimization techniques and commonly agreed upon strategies within the scientific community. Optimizable targets are identified and the effectiveness of their optimizations, discussed in related research, is evaluated. The aim of this chapter is to provide a general understanding of WPO techniques used to boost various performance metrics, independent of any specific plugin or platform. While some of the optimization techniques are also relevant to the WordPress use case and are analysed in that context, the focus is on presenting general best practices that serve as the foundation for understanding the objectives and functions of optimization plugins.

Creating a WordPress website begins with an idea. The planning, structure, and technical implementation affect how well the site serves the needs of the intended userbase. (Schäferhoff 2025) This naturally ties into website optimization, as optimizing a website directly impacts user experience (Shailesh & Suresh 2017; Shethiya 2025).

### 2.1 Web performance optimization

WPO affects not only the usability of a website, but also aspects such as user retention, site traffic, and online revenue. WPO also influences search engine ranking; faster pages are ranked higher. There are many different ways to optimize a webpage ranging from minimizing the size of delivered content, such as images, to different ways of prefetching assets. (Shailesh & Suresh 2017; Shailesh & Suresh 2016; Vihervaara et al. 2016; Shethiya 2025) Based on previous studies' findings, relevant WPO techniques with a measurable impact on performance are reviewed in this section.

#### 2.1.1 Static asset optimization

Modern webpages are composed mostly of other content besides hypertext markup language (HTML), such as images and JavaScript. Because of the high quantity of this content, optimization of these assets could have a large impact on the website performance. (Shailesh & Suresh 2017) One of the largest contributing factors to a website's size is the amount and

size of images. The research by Heričko et al. (2021) found that, in a WordPress environment, using any automated form of image optimization was in all cases at least as effective as, if not better than, not using any. However, the computational overhead of using an additional plugin did, in some situations, cost more in terms of performance compared to the native WordPress image optimization features. The primary optimization technique used in image optimization is compression, which can be either lossy or lossless (Vihervaara et al. 2016). In addition to compression, deleting unnecessary images, lazy loading offscreen images, and choosing an optimal image format, such as WebP, is also good practice when optimizing these assets (WordPress 2023a; Osmani et al. 2024).

Another form of compression can be applied to minimize the size of text-based files such as JavaScript and HTML. One way of compressing HTML is by using the GZIP format, which can greatly reduce the amount of data that needs to be transmitted from the server to the end user. The smaller file size also reduces overall energy consumption, since less data is transmitted over the internet. (Vihervaara et al. 2016; Shailesh & Suresh 2017; Pyles 2022) However, compressed files must be decompressed on the client side before they can be rendered. This longer processing time can delay how quickly content is rendered. While gzip is a widely used compression tool, it is being replaced by newer compression algorithms such as Brotli, which has already been widely adopted by web browsers and servers (Alakuijala et al. 2019).

In general, reducing the number of hypertext transfer protocol (HTTP) requests lessens the latency and amount of bandwidth used. However, this does not apply to versions after HTTP/1.1, as they support request multiplexing. Reducing the number of requests made to external services for assets, such as stylesheets or third-party advertisements, can also help reduce latency, since loading those assets may become a bottleneck. Static assets such as JavaScript and cascading style sheets (CSS) should be minified and merged, when possible, to lower the number of HTTP/1.1 requests. This is not necessary when using HTTP/2 or HTTP/3. Removing unnecessary content is also a best practice when optimizing web content. This covers practices such as deleting unused images and duplicate file includes, cleaning up unnecessary data from the database, and in the case of WordPress, deactivating or entirely deleting plugins and themes that go unused. This reduces the amount of overhead and thereby increases the performance. (Shailesh & Suresh 2017; Shethiya 2025; WordPress 2023a; Pyles 2022)

### 2.1.2 Server- and browser-side caching

During normal operation, a WordPress server responds to a web request with a dynamically created HTML file. So, each time the website is requested by a browser, WordPress must retrieve the relevant content for the page from the database, apply PHP logic, and return a valid HTML document. (Patel et al. 2011b; Khalil et al. 2024) This is resource-intensive for the server, and therefore causes more latency and power draw (Tomiša et al. 2019).

Server-side caching is a technique where the web server generates this HTML document and stores it for a specific amount of time. During this time the server responds to similar requests by sending the already cached version of the website, reducing the latency and computation done by the server, thereby improving QoS under load. After the time has expired, the cached version is updated. (Everman & Zong 2018) WordPress servers may also cache results from database queries, by using server RAM, to function as an object cache (Pyles 2022). Alongside database queries, data stores like Memcached and Redis can also be used to cache application programming interface (API) responses (Shethiya 2025). Some content from a WCMS may not be cacheable because the content is unique to a specific user, such as personal information or purchase history. However, breaking the page down to fragments would allow the server to cache the static parts while inserting the dynamic content to the final resulting page at runtime (Shivakumar & Suresh 2018). WordPress also offers its own native way of storing data in a database cache using the WordPress Transients API (WordPress 2019a).

In browser-side caching, the browser caches the requested webpage for a specified amount time indicated by the HTTP header field in the server's response. When the cached version of the site expires, the browser may request the server to verify if the site has changed. If the server confirms that there have not been any changes, the cached version will be reused, which is faster than reloading the whole site. The server also benefits since responding to the query, regarding changes after the last cached version, is less resource-intensive than resending the same cached version back to the client. (Vihervaara et al. 2016; WordPress 2023b)

### 2.1.3 Other optimization strategies which affect performance testing

As stated previously, it is general good practice to only have plugins installed that are used by the WCMS. It is also recommended that the latest versions of plugins, database software, and the operating system be used, as newer software may have a positive impact on performance through bug fixes and optimizations (WordPress 2023a). When using WordPress, it is therefore important to make sure the plugins are regularly updated and maintained by their developers (Pyles 2022). This also plays a pivotal role in the security aspect of WordPress sites because the plugins may have vulnerabilities that need to be addressed, which may not be possible if the authors are not actively engaged in the maintenance of the software (Cigoj & Blazic 2019).

A more technical approach to improving website performance is prefetching. Prefetching is a way of predicting the need for an asset based on the user's behaviour and then sending a query for that asset to the server and storing the result. This would cut down on the latency as the content would be immediately accessible when needed. There are many ways to model predictive patterns such as by logging the most commonly requested content. However, research shows that sub-optimal prefetching can result in more network bandwidth consumption, if prefetching is done too frequently. Additionally, if the prefetched assets are not needed, the resources put into prefetching them are effectively wasted. So, prefetching should be done carefully only in specific contexts where it may have a benefit. Lazy loading is another way to cut down on rendering delay by only loading certain assets on-demand, rather than on the initial page load. In simple terms, lazy loading prevents the rendering of assets that are not yet in view of the user, cutting down on delay when rendering a long page of content. (Shivakumar & Suresh 2018; Pyles 2022; Shethiya 2025) This is especially important when it comes to images, since they can use up a lot of bandwidth. Lazy loading images outside the viewport, meaning everything initially offscreen, lowers the overall wait time. (Osmani et al. 2024) Prefetching and lazy loading both adhere to the best practice of asynchronously loading assets.

Speculative loading is another asynchronous loading technique that can be influenced by browser heuristics or developer-provided hints. Essentially, speculative loading predicts user actions and begins fetching assets or preparing necessary resources for fetching in advance, such as prefetching pages or establishing HTTP connections before they are needed.

Speculative loading also comes with similar drawbacks to the previously discussed prefetching technique, but there is also potential to achieve near-instantaneous page navigation if predictions turn out accurate. (Mozilla 2025) During the writing of this thesis, speculative loading was added in WordPress version 6.8 as a natively supported feature (Arntz 2025).

Another way to cut down on latency would be to use a content delivery network (CDN). A CDN is a collection of proxy servers distributed geographically in order to minimize the propagation delay of data transfers. The CDN mirrors popular static files, such as videos, which are accessed often. This means that the latency of delivering this commonly accessed content is shortened, making the QoS better for users regardless of geographical distance, since asset requests get answered by the geographically nearest server to the user. A CDN could also lessen the work of the main server by shifting away some of the workload to the network of proxy servers. (Everman & Zong 2018; WordPress 2023a; Shivakumar & Suresh 2018)

Database caching was previously mentioned in section 2.1.2, but there are other ways of optimizing a database. Large databases may cause delay in query execution, but this can be minimized with sharding, indexing, and query optimization (Shethiya 2025). Regarding the WordPress database, a paper by Stojić et al. (2024) concluded that loading in content directly from the database can, in certain situations, be faster than loading in the same data from WordPress posts.

An entirely different kind of optimization is search engine optimization (SEO). SEO techniques bump up the website in the search engine ranking, garnering more traffic to the site by making it easier to find through a search engine. Different SEO methods include, for example, using specific keywords on the site that people might search for. There are also SEO focused plugins for WordPress such as Yoast SEO. (Mulyandi et al. 2022; Hakimi & Rahman 2021)

## 2.2 Analysis of the research

There are numerous ways of conducting WPO, many of which were brought up in the previous section. Based on the current state of research and the techniques previously

examined, more general strategies for WPO can be formulated by following the principles of best practice outlined in the research.

Shivakumar and Suresh (2018) identify multiple patterns, which can be expressed as general strategies for WPO. The more assets are requested, the longer the wait times become. Large assets should be loaded asynchronously and on demand. The rendering of assets should also be optimized depending on the device used. The website design and content should be made lightweight, responsive and adaptive for the aforementioned devices. In simple terms, only ask the server for what is needed and load it when needed. Avoiding unnecessary work is a fundamental principle of WPO.

A well-thought-out implementation of WPO should adhere to most of these principles. For example, bundling assets that are critical to the first paint of the site, and then instructing the browser to preload them (Djirdeh et al. 2018). Preloading allows a developer to prioritize which assets to request first, speeding up the initial render of a page. These essential parts may include HTML, CSS, JavaScript, fonts, and small images. Dynamically generated assets may be requested afterwards, which lowers the initial rendering time, since the client does not need to wait for the slower assets before starting the rendering process. After the initial paint of critical assets, non-essentials can be deferred or lazy loaded to further optimize performance.

One way of achieving better performance would be to upgrade the hardware of the server distributing the content. Upgrading RAM speed and capacity can alleviate bottlenecks when a server is nearing the limits of what its resources can handle. Disk speed is also a factor in overall performance and may be a limiting factor depending on the amount of reads and writes. Increasing the number of servers helps handle higher server loads caused by many concurrent requests. (WordPress 2023a) While this kind of horizontal scaling increases the site's capacity to serve more users, that doesn't mean individual requests are processed faster under zero-load conditions, as response time for a single request is still determined by the performance of a single server.

As was discussed previously, WordPress dynamically builds an HTML file to respond with upon receiving a request from a client. This is why RAM and disk speed is important in the context of WCMSs like WordPress. Dynamic caching could be used to lessen the strain on resources by caching full or parts of pages. Code and database table caching are also

considered dynamic caching (Shivakumar & Suresh 2018). These various dynamic caching methods could be used to alleviate strain on limited server resources.

There are times that the QoS could be bottlenecked by low-power server hardware. Still, higher-end hardware does not always scale linearly with performance unless the upgrades are done with pinpoint accuracy: just renting access to more expensive computing resources does not equate to an equal increase in performance proportional to the price. (Yan et al. 2016) So, there needs to be evaluation done on when to optimize and when to upgrade to a server capable of higher performance. Still, most servers aren't utilizing the maximum potential of their resources, which is wasteful in terms of energy and cost (Everman & Zong 2018). Consequently, WPO is the right solution for many content providers on the web.

The general WPO practices covered in this chapter provide a basis for analysing the features of WordPress optimization plugins. Some plugins may guide a developer to use certain WPO techniques covered in this chapter, while others might automate the optimization process at the click of a button. In certain cases, a plugin might offer or recommend turning on some optimization feature, which could unintentionally degrade performance. Therefore, understanding the underlying principles of WPO is essential for the later evaluation of performance optimization plugin effectiveness in this thesis.

## 3 Research methodology

This chapter first explains the methodology in general terms, followed by a description of how the methodology is applied to this thesis' specific use case. Additionally, the measurement tools and environment chosen to conduct the empirical study are laid out as a framework for the research.

### 3.1 Methodology

This thesis will make use of a quantitative experimental research methodology. By gathering data of quantitative metrics via measurement, it is possible to identify causal relationships between different actions. For example, measuring an attribute such as delay using a scale of milliseconds could shed light on the difference between using some optimization plugin A versus B. In experimental research, this investigative approach is meant to assist in distinguishing between outcomes caused by the aforementioned actions by manipulating specific variable factors and keeping the rest as consistent as possible. The crux of experimental research is to measure the variables, manipulate them in a deliberate manner, and then measure again to note the differences between the cases under study. After the collection of sufficient data, begins the analysis of quantitative results, which may be interpreted using statistical means. (Wohlin et al. 2024)

### 3.2 The setup of the experiment

The purpose of the experiment setup is to limit the number of variables and thereby their effect on the results. The aim is to only change specific variables to accurately measure the effects of optimization. First, a baseline measurement will be taken of the website's performance without any optimizations applied. During the experiment, one hundred separate measurements will be taken of each optimization plugin's performance on the site. The measurement results will be appended to a spreadsheet. This process will then be repeated for all of the websites.

The three websites used to benchmark the effectiveness of the optimization plugins are provided by the WordPress Development Company Parvus Vulpes Oy. These specific webpages were selected by the company as they represent typical WordPress websites of differing sizes. By utilizing typical websites made with WordPress, the experiment setup is going to more closely resemble real-world applications in which WordPress optimization plugins may be leveraged. This, in turn, will make the results paint a more accurate picture of the usefulness of the optimization plugins.

The websites have been built using page builders. Page builders make the content editing process easier, since the developer does not need to write HTML or CSS themselves (WordPress 2024). One such page builder paradigm is the block editor, a website creation approach that uses a modular system of blocks to lay out a webpage. The default WordPress block editor is called Gutenberg. (WordPress 2019b) Blocks of content can be divided into two distinct groups: static and dynamic blocks. Static blocks are saved to the database as fixed HTML, and dynamic blocks are generated on the server side upon request (WordPress 2024). Consequently, by bundling critical content into static blocks, content can be preloaded to serve to the client while waiting for dynamic content to be fetched.

Given the reliance on both static and dynamic content, the web server plays a critical role in combining these elements into a final functional page. The web server requirements of WordPress are the following: PHP, MySQL or MariaDB, and HTTPS support. WordPress recommends the usage of an Apache or Nginx web server for running WordPress websites. (WordPress 2018) There are no official operating systems designed with WordPress in mind, but Ubuntu is a valid choice because of its long-term support, extensive documentation, and lightweight operation (Ferguson 2020). The websites in this thesis' experiment will be hosted locally on Local WordPress version 9.2.2, which is a free-to-use WordPress development application (Local WP n.d.). The sites will be hosted on web servers running nginx 1.26.1, PHP 8.3.0, MySQL 8.0.35, and WordPress 6.7.2. Local WordPress will also be using the HTTP/1.1 protocol, which may affect latencies negatively. This is because the HTTP/2 and HTTP/3 protocols can utilize request multiplexing, whilst HTTP/1.1 cannot (Shethiya 2025).

Keeping the original sites unchanged is crucial to keeping the experiment's results valid and accurate representations of real-world conditions. Because of this, existing caching rules are not altered. Because of possible PHP, database, and static file caching, initial cold loads from

the server may appear as outliers in the measured data, which could lead to skewed values when calculating averages. However, if a sufficiently large number of measurements are taken in series, results will still be reliable to draw conclusions from.

### 3.3 The measurement tool, metrics and procedures

Lighthouse is an open-source performance auditing tool made by Google and integrated into Chrome DevTools. Previous research has also made use of the tool to generate performance reports for the purpose of improving websites. Lighthouse generates an overall weighted performance score from 0 to 100 but also shows the individual score for five different performance metrics. Each metric is assigned a colour-coded grade, green, orange, or red, indicating fast, moderate, or slow render times, respectively. (Lehat et al. 2023) Scores ranging from 0 to 49 are red, 50 to 89 orange, and 90 to 100 green. Scores also do not fluctuate linearly, which means that improvements at lower ranges lead to a larger increase in performance score compared to the same improvement at higher levels. One aspect to keep in mind while analysing performance scores is that the score increases near-linearly between 50 and 92, after which performance improvements have diminishing returns. (Google 2019a)

The five main performance metrics that Lighthouse audits are:

- First Contentful Paint (FCP): FCP is a metric, which measures the time between when the browser begins to load the page and when the first content is rendered. In simple terms, FCP is the time it takes for the browser to display the first visible element. (Walton 2023)
- Speed Index: The Speed Index metric uses a video taken of the page while it's loading, to measure how fast the page content is displayed to the user. This metric measures the overall progression of loading a page rather than a specific moment. (Google 2019b)
- Total Blocking Time (TBT): TBT measures how long the page is blocked from responding to user actions, such as clicking different buttons on the page. (Google 2019c)

- Largest Contentful Paint (LCP): LCP measures the time between opening the page and loading in the largest asset such as an image, video, or text block. (Google 2020)
- Cumulative Layout Shift (CLS): CLS is a calculated value based on how many unexpected layout shifts, such as a button moving when the page loads, happen and by how much. (Mihajlija & Walton 2023)

The main performance metrics measured by Lighthouse mostly adhere to the Google Core Web Vitals, which are LCP, CLS and Interaction to Next Paint (INP). INP replaced First Input Delay as a Core Web Vital in September of 2024. Because Lighthouse is a lab tool, measuring field metrics may sometimes prove difficult. Despite LCP and CLS being primarily field metrics, they can still be measured in a lab environment, but the same is not the case for INP. Even so, TBT may serve as a proxy for INP because the same WPO techniques, which improve TBT in the lab, can do the same for INP in the field. (Walton 2024)

In addition to the main performance metrics used by Lighthouse to calculate the overall performance score, the server response time is also noted because of its importance in the context of this thesis' focus on WordPress. Because WordPress must dynamically create the HTML by retrieving database content and run the interpreted PHP code before responding to the HTTP request for the webpage, it causes latency in the server response time. In short, server-side delay is caused by the HTML generation process. The server response time will be used to measure the overall performance of the server-side.

The procedures for measuring the effects of optimization plugins on the three WordPress sites include the following steps:

1. Running the WordPress sites locally.
2. Ensuring all sites run on the same versions of WordPress and PHP.
3. Cloning the original sites as many times as there are optimization plugins to be tested.
4. Installing one plugin on each of the cloned sites, enabling the optimizations, and checking that the site remains functional and visually the same.
5. Creating a Python script that runs one hundred Lighthouse tests for each original and cloned site, saving the results in comma-separated values (CSV) format.

6. Opening the CSV file in a spreadsheet program and calculating averages for the different tests.
7. Conducting a t-test on each previously mentioned performance metric's results in order to compare the original site with each of its optimized clone versions, determining the statistical significance of the results.
8. Compiling the results into tables and charts to illustrate the performance differences.

By running tests back-to-back in a controlled environment, where outside influences are kept to a minimum, the data can be used to draw conclusions about the optimization plugins. A two-sample, two-tailed unequal variance Student's (1908) t-test is used to show whether the difference in results is meaningful. If a metric's p-value is lower than 0.05, there is sufficient statistical evidence that the results were not due to random variance. To do this, the spreadsheet program's built-in t-test function will be used (Microsoft n.d.).

When running Lighthouse version 12.4.0 via command line input using HeadlessChromium 134.0.0.0, a simulated network roundtrip time throttle of 150 milliseconds is used to simulate a 4G connection. This is necessary because, without any kind of throttle, running the sites locally would make the latency of page requests negligible, which wouldn't stay true to real-world conditions. In addition to the network throttle, Lighthouse makes use of simulated processor throttling by default. By emulating a commonly used slow client, such as a mobile device or low-end desktop environment, Lighthouse can provide insight on what a worst-case scenario might be like. This approach could also help identify bottlenecks and other performance-limiting factors more easily, as the effects of optimizations become more noticeable under resource-constrained conditions.

## 4 Technical research on WordPress optimization plugins

In addition to the empirical data gathered, reasoning behind the practical implementation of the research method is also discussed in this chapter. The end goal of the technical part of the research is to shed light on the effectiveness of WordPress optimization plugins.

### 4.1 Conducting the experiment

This section outlines the process of setting up the WordPress websites for the experiment. In addition to explaining the experiment setup, more information about the optimization plugins is presented. The aim of the setup is to create separate testing environments where the only variable between them is the optimization plugin used. This way it is possible to ensure that the results are valid and not due to outside influences on performance.

#### 4.1.1 Selection of the plugins and setting up the testing environment

The first part of the experiment is to choose a sample of commonly used optimization plugins to conduct the experiment with. The plugins are chosen from the official WordPress (n.d.) plugin directory with the criteria being that the plugins must each have over one million active installations and a user rating better than 4.0 out of 5. This will give a general overview of the types of optimization plugins available and installed by the average users of WordPress. Table 1 shows the number of optimization plugins available grouped by rating. The ratings have been rounded to the nearest whole number. For example, a plugin with a rating of 4.2 would get rounded to 4.

Table 1. Optimization plugin ratings

Rating out of 5	0	1	2	3	4	5
Number of plugins	679	31	19	85	229	726

The data for Table 1 was obtained through the WordPress.org API. Optimization plugins were included if they had at least one of the following tags associated with them in the

WordPress (n.d.) plugin directory: optimization, optimize, SEO, performance, pagespeed, cache, or image optimization.

Appendix 1 illustrates different main features of the optimization plugins and shows whether they are paid or free. QUIC.cloud, ShortPixel, and Smush CDN are CDN services that may be used to access more features, such as WebP image conversion, if applicable. Some plugins also allow the website developer to deploy any CDN they desire. In the case of LiteSpeed Cache, caching features require the use of a LiteSpeed Web Server (LSWS) to function (LiteSpeed n.d.). If a feature is free but limited, it means that the website developer is allowed to take advantage of the premium paid feature, but with some restrictions. For example, Smush offers free image compression, but only for images with a maximum file size of 5 megabytes. If a developer wants to compress larger files, they will need to pay for the premium subscription Smush offers.

After selecting the plugins, begins the task of choosing which optimizations to apply and applying them to each of the three sites. To do this, eight copies of the original site are made, each with a different optimization plugin installed to it. Each plugin's optimization features are enabled if it is possible to do so and doesn't affect the site visually, risk other incompatibilities, or cause unintended behaviour. This procedure is then repeated for the other sites as well. Even though every possible optimization feature is turned on, no CDNs will be used, as they introduce more unpredictability into the testing due to their variable latency, caching, and geographic location.

#### 4.1.2 Auditing with Lighthouse

Google calls running performance tests with Lighthouse auditing and the test results audits. Auditing automatically with Lighthouse can be made easy with the Node command line tool. After installing Lighthouse with Node, audits can be performed straight from the command line. (Google 2016)

```
command = f"lighthouse {URL} --quiet --chrome-flags='--headless' --output=json\  
--only-categories=performance --screenEmulation.desktop\  
--screenEmulation.width=1920 --screenEmulation.height=1080\  
--screenEmulation.deviceScale Factor=2"
```

Figure 2. The command line input for running audits with Lighthouse

With the command in Figure 2, it is possible to run an audit via the command line. If the command is paired with a for-loop and a function for writing the results to a file, the code can be run any number of subsequent times, thus automating the testing process. The flags used in the command specify that the desktop screen size should be set to full high definition, while the device scale is set to the Lighthouse recommended setting of two. This size is chosen, because the WordPress site needs to load in more content on the desktop version. This makes the impact of optimization more noticeable in the results.

## 4.2 Results of performance testing

The performance testing can be divided into two distinct groups: tests with and without caching enabled. The first subsection covers the results of optimizing without caching, while the second discusses the results of having caching enabled. This is done because caching obscures the impact of optimizations enabled on the server side. The only exception to this is the results row for the WP Rocket plugin. WP Rocket is a paid optimization plugin. WP Rocket requires caching to be enabled for the optimizations to function. This is why WP Rocket is last in the Tables, while all other plugins are in alphabetical order.

### 4.2.1 Results of optimizing without caching

After writing the test results to a CSV file, an average of the one hundred repeats of audits can be calculated for each of the metrics. Tables 2, 3 and 4 house the results of these measurements. Gray colour indicates the baseline measurements where no optimization or caching was used. The other cells are highlighted with a colour if the t-test comparing an optimization plugin to the baseline was statistically significant. A t-test result is statistically significant if probability  $p < 0.05$ . The highlight colour is green if the measured result was better, yellow if it remained the same and red if it worsened. For the performance score, a larger score is better. For the other metrics, a smaller value is better. Values of the last column are rounded to the nearest six decimal places, while all others are rounded to the nearest two. The unit *ms* means milliseconds, and the metrics without a specified unit do not have one.

Table 2. Results for website 1

Optimization Plugin	Performance Score	Server			Speed Index [ms]	TBT [ms]	CLS
		Response Time [ms]	FCP [ms]	LCP [ms]			
No optimization	<b>74.00</b>	<b>241.72</b>	<b>1598.81</b>	<b>9937.33</b>	<b>1598.81</b>	<b>0.00</b>	<b>0.019937</b>
Autooptimize	73.94	287.33	1688.41	9704.46	1699.57	31.39	0.018942
LiteSpeed Cache	95.42	615.48	1072.92	2828.91	1542.05	14.70	0.019797
Speed Optimizer	84.29	301.45	1081.82	4425.68	1651.96	0.69	0.039103
W3 Total Cache	90.12	437.80	1597.33	3533.46	1597.37	0.00	0.019245
WP Fastest Cache	74.00	244.14	1598.21	9913.50	1598.21	0.00	0.019491
WP-Optimize	80.55	300.45	1596.69	5123.38	1596.69	0.00	0.018435
Smush	91.02	258.38	1600.10	3409.64	1600.14	0.00	0.026597
WP Rocket	89.57	26.60	2143.73	3374.82	2143.73	0.13	0.041058

The results of the first website test show that, in nearly all cases, using any of the optimization plugins yields an equal or higher performance score compared to the baseline. On the other hand, most of the plugins also ended up causing more latency to the server response time. The exception to this is WP Rocket, which has caching enabled and therefore a low server response time.

Table 3. Results for website 2

Optimization Plugin	Performance Score	Server			Speed Index [ms]	TBT [ms]	CLS
		Response Time [ms]	FCP [ms]	LCP [ms]			
No optimization	<b>72.33</b>	<b>325.67</b>	<b>1772.74</b>	<b>10622.28</b>	<b>3866.12</b>	<b>6.28</b>	<b>0.000825</b>
Autooptimize	71.83	365.28	2341.65	6930.67	4886.59	39.48	0.001636
LiteSpeed Cache	88.31	620.03	1226.74	3832.75	2296.57	0.12	0.001981
Speed Optimizer	90.67	432.75	1065.94	3050.22	2420.70	3.98	0.107404
W3 Total Cache	80.12	500.43	2317.13	4771.64	2449.26	1.78	0.000495
WP Fastest Cache	72.26	333.67	1773.79	9949.75	3935.44	6.39	0.000733
WP-Optimize	73.87	662.97	1452.61	6333.45	4823.03	5.15	0.000817
Smush	84.70	504.28	1810.62	3779.53	2216.76	3.19	0.104727
WP Rocket	79.66	30.95	2437.94	4918.09	2437.94	0.26	0.001765

From the results of performance testing on website 2 shown in Table 2, it may be concluded that WP Fastest Cache had little to no effect on performance, while Autooptimize worsened five out of seven metrics. Website 3 is the simplest out of the three websites, because of its content. Site 3 only has a few images and minimal text, making it more difficult to find elements to optimize. The results of testing website 3's performance are shown in Table 4.

Table 4. Results for website 3

Optimization Plugin	Performance Score	Server			Speed Index [ms]	TBT [ms]	CLS
		Response Time [ms]	FCP [ms]	LCP [ms]			
No optimization	<b>72.14</b>	<b>384.14</b>	<b>2464.58</b>	<b>8339.46</b>	<b>2470.69</b>	<b>0.76</b>	<b>0.000005</b>
Autoptimize	75.55	461.56	1925.51	6568.45	2053.51	0.96	0.000015
LiteSpeed Cache	75.74	952.12	1486.60	6916.77	2596.15	0.28	0.000000
Speed Optimizer	75.56	610.05	1659.96	7051.61	2129.89	0.00	0.000000
W3 Total Cache	74.88	545.25	2253.84	6472.08	2291.13	1.15	0.000279
WP Fastest Cache	73.05	417.19	2454.27	7034.22	2482.60	0.33	0.000007
WP-Optimize	75.92	634.89	2108.66	6187.36	2190.90	7.32	0.000000
Smush	72.03	436.40	2828.29	6796.09	2834.60	2.56	0.000000
WP Rocket	75.00	25.19	1872.57	7336.48	1872.97	0.42	0.000143

The results of the third test show that all plugins, except for Smush, improved the website performance. During testing, Smush was unable to detect any optimizable images, which caused its low performance score. The performance scores of the different plugins give an overall sense of the usefulness of the plugins. The improvement can be seen in Figure 3, where the change in performance score for each site is graphed.

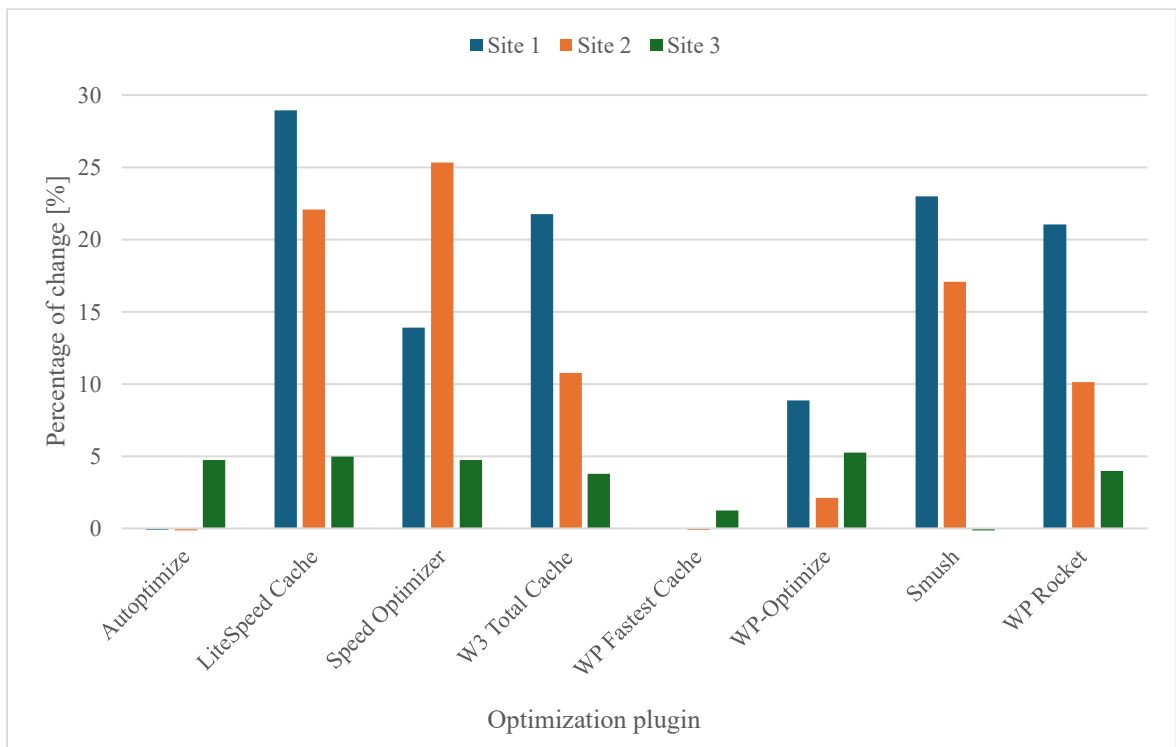


Figure 3. Performance score differences relative to the baseline

Figure 3 shows the percentage change when comparing the optimized website audit results to the baseline control measurements, where no optimizations were applied. As shown in the graph, no single plugin performs best on all sites, since there are noticeable differences between the plugins. For example, WP-Optimize did not perform well on sites 1 or 2 compared to the other plugins, but it achieved the highest score on site 3 among all plugins. Speed Optimizer improved the score more on site 2 than site 1, which is an exception when looking at the performance change of other plugins.

The results of WP Rocket do not dramatically differ from the results of the other plugins, even though WP Rocket is a paid plugin, and therefore it would be expected to perform better than free alternatives. In the testing results, WP Rocket also had one large outlier point in its results on all sites. These points were deleted as to not outrageously skew the averages of the measured metrics. This should be taken into consideration when using WP Rocket on any website, and the phenomenon should be studied further.

Appendices 2, 3, and 4 are box and whisker charts of four performance metrics, excluding outlier points. In box and whisker charts, averages are marked with a cross, and medians are indicated by a dash inside the box. The charts depict the variability in individual test results. Some observations of note include how much some of the plugins decreased the performance metric values and the variability of them compared to the baseline. Also, if the average of a measurement is high compared to the median, it signifies that the measurement has significant outlier points.

#### 4.2.2 Results of optimizing with caching enabled

The previous tests were done without applying any kind of caching with the free plugins. Because caching lowers the latency of the server responding to a page request, it should be reflected in the server response time. The following table only contains tests of plugins which offer free caching features. According to Appendix 1, four out of eight plugins offer caching features without any fee. Table 5 contains calculated averages from one hundred repeats of Lighthouse audits with caching enabled. The cells are coloured if the t-test comparing an optimization plugin to the same optimization plugin without caching enabled was statistically significant. The results are rounded in the same way as in Tables 2, 3, and 4.

Table 5. Results for website 1 with caching enabled

Optimization Plugin	Performance Score	Server Response Time [ms]	FCP [ms]	LCP [ms]	Speed Index [ms]	TBT [ms]	CLS
Speed Optimizer	84.57	24.65	1075.51	4391.54	1236.26	0.00	0.035623
W3 Total Cache	89.73	38.61	1596.92	3580.68	1673.89	0.00	0.020812
WP Fastest Cache	74.00	245.14	1597.96	9923.13	1598.94	0.00	0.019071
WP-Optimize	80.47	24.63	1598.17	5132.85	1598.17	0.00	0.018174

The results of Table 5 show a drastic improvement in server response time compared to the same results without caching. This is to be expected, since static versions of WordPress sites are faster to deliver than dynamic ones (Tomiša et al. 2019). The only plugin that did not show any improvement was WP Fastest Cache, while W3 Total Cache and WP-Optimize significantly lowered the server response time.

#### 4.2.3 Optimizing the sites based on the previous results

The free features differ from plugin to plugin. Thus, managing to optimize all major aspects of a site with one free plugin seems unlikely. By analysing the results of the previous audits and the table in Appendix 1, a plugin combination can be formed to enhance WPO further.

As a proof of concept, a combination of two plugins will be used in a new test. Because Speed Optimizer achieved a high performance score on websites 1 and 2 according to Figure 3, that plugin will be used in the next test. Speed Optimizer does not offer image compression or WebP image support for free; therefore, Smush will be used for its image optimization features in the new test. LiteSpeed Cache scored the best on website 1, according to Figure 3, but it does not offer caching features for free, as shown in Appendix 1, which is why it will not be chosen. The results of this new test can be seen in Table 6.

Table 6. Results of testing with both Speed Optimizer and Smush enabled

Site	Caching	Performance Score	Server Response Time [ms]	FCP [ms]	LCP [ms]	Speed Index [ms]	TBT [ms]	CLS
1	Disabled	94.94	387.19	1302.67	2870.89	1559.97	14.77	0.040337
1	Enabled	95.67	31.73	1070.67	2773.32	1077.57	0.00	0.037198
2	Disabled	65.88	460.45	1068.57	4017.08	2370.18	5.26	0.557295
2	Enabled	64.71	38.5903	1067.206	4163.22079	1779.429	3.69	0.572264

In Table 6, the cells are coloured if the t-test result was statistically significant compared to the results of only using Speed Optimizer without caching on the same site. Site 3 was not tested, because Smush did not find any images to optimize there. In the case of site 1, the performance of this combination of plugins shows improvement. By enabling caching, the server response time is also shortened as could be expected by looking at the results shown in Table 5. However, the same cannot be said about the results for site 2. Both with and without caching, the performance suffered so much that the score dipped below the baseline measurement for site 2 shown in Table 3.

## 5 Discussion

This chapter discusses the possible reasons behind the test results and how they compare to previous research. In this chapter, the research questions are answered by first analysing the optimization plugins' features and then analysing the testing results. At the end of the chapter, the limitations and other risks to this research are also discussed and possible future avenues for further research possibilities are given.

### 5.1 Comparing the plugins' optimization techniques to prior WPO Research

The tested plugin features share similarities with the WPO strategies outlined by previous research, but there are also differences. This section attempts to provide a sufficient answer to the research question: Do the plugins adhere to the strategies and best practices outlined by previous research? This will be done by looking at how the features of the plugins, shown in Appendix 1, compare to the strategies discussed in the second chapter.

All the tested plugins offer some kind of static asset optimization. Many of the plugins offer minifying and merging files, which can reduce the number of HTTP requests made to the server. The optimization of web fonts may also help in reducing the number of these requests, but because requests for external fonts are made to third-party services, they could become a bottleneck in some scenarios. However, deferring these requests would prevent render-blocking. Alternatively, using system fonts circumvents the problem entirely.

The plugins leverage many of the WPO strategies outlined in the research by Shivakumar and Suresh (2018), such as reducing the size of objects sent over HTTP and making use of asynchronous loading. Prefetching is another asynchronous loading technique, which is used by some plugins. One such feature is domain name system (DNS) prefetching, which resolves DNS queries before they are needed. Pyles (2022) notes that DNS lookup time increases latency. For this reason, some plugins offer DNS prefetching, which can cut down on the webpage load time. That said, DNS caching is not offered by the plugins but is mentioned in the research by Sailesh and Suresh (2017). WP Rocket offers a link preloading feature, which implements prefetching by fetching the HTML of a target page if the user hovers over a link for 100 milliseconds. Other plugins offer general preloading of critical

assets. Preloading is considered an effective way to speed up the loading process, as discussed by Djirdeh et al. (2018). Other examples of asynchronous loading techniques utilized in many of the plugins include deferring JavaScript and lazy loading images.

As concluded by the research of Heričko et al. (2021), image optimization can make a big difference in the overall page size and thereby increase performance. Some plugins also offer image optimization and conversion to the WebP format. However, with most of the plugins tested, image optimization requires payment. The tested plugins do not offer other media optimization. Other multi-media content, such as videos, is recommended to be hosted on a CDN (Shivakumar & Suresh 2018).

One WPO strategy for optimizing static assets not mentioned in the literature is disabling emojis. A majority of the plugins tested offer this feature for free. Disabling emojis could help in optimizing a WordPress site's performance, since disabling them would lessen overhead by stopping the execution of the related scripts. Related research considers reducing overhead as a general best-practice. Some of the plugins that were investigated in the fourth chapter do offer database cleanup functions, which help cut down database bloat as stated in the whitepaper by Pyles (2022).

Additionally, the plugins have a variety of caching features available, but not all of them are free. Features like dynamic and object caching are not offered for free by most plugins tested. Most of the WPO strategies related to caching are nevertheless utilized by these plugins. The plugins did feature opcode and representational state transfer (REST) API caching, which are not extensively discussed in related research. API caching was briefly mentioned in the journal article by Shethiya (2025). The article also mentions that caching session data, database queries, and API responses requires a persistent in-memory cache such as Redis.

There is a vast array of optimization strategies discussed in WPO research, but not all of them are so easily implemented without tailoring them in specific contexts. This is partly why some optimization techniques discussed in research are not offered by the plugins. As an example, prefetching assets is not something a developer should try to implement without long consideration, because of its many possible drawbacks. In summary, the majority of optimization techniques are shared between the research findings and the implementations of the plugins, with only minor differences.

## 5.2 What do the test results mean?

From the results from Tables 2, 3, and 4, it can be deduced that increasing the amount of processing needed for server requests also increases server response time. However, using more processing power on the server side to optimize the load time on the client side does result in a better QoS for the user. Measures to minimize the impact of this phenomenon are explored in this section. Additionally, answers will be provided to the remaining research questions, specifically how WordPress optimization plugins affect the average website and whether their benefits justify their installation.

### 5.2.1 Why do the results on the sites differ from one another?

The improvements vary between the websites due to many reasons, such as differences in the amounts and types of content, as well as the plugins already installed on each site. In nearly all cases for the three websites, using an optimization plugin improved the LCP metric. The improvements to LCP and FCP stem from the fact that the plugins' optimizations reduced the amount of data needed to be loaded and processed by the client's browser. Minifying CSS, JavaScript, and HTML helps the client render the pages quicker. These techniques are applied on the server-side for each request if no caching is used. This is why the server response time is higher when using these kinds of optimizations.

In the case of image optimization, compression reduces the file size and, therefore, also reduces the load time for those assets. Additionally, the use of asynchronous rendering, by enabling lazy loading of images and deferred JavaScript, reduces the FCP rendering time. The Speed Index also improves with the optimization plugins, which indicates that the content appears faster to the user instead of gradually loading in, which may improve the user experience and therefore also the QoS. The differences between the Speed Index results could be due to the number of images used on each site. For example, site 2 uses many small images, which can be optimized, therefore contributing to a lower Speed Index score.

There may be a correlation between higher performance scores and higher server latency. The increased latency could be because the server must complete more operations before responding to a request. This could include minifying CSS and JavaScript, combining CSS files, and compressing the data with gzip before sending a response to the client. Because

Smush does not do many of those operations every time a webpage is requested, the latency is lower than with the other plugins, but the performance score is still improved over the baseline, as can be seen from Tables 2 and 3.

### 5.2.2 Differences between plugin performance

Some plugins produced better results than others. For example, WP Fastest Cache did not perform well in most of the tests, which may be due to the lack of many free features when compared to the other plugins, as shown in Appendix 1. There are also somewhat unexpected results such as the performance of Autoptimize, which can be seen in Figure 3. Autoptimize worsened performance on sites 1 and 2 but reached an almost 5% improvement on site 3. According to Appendix 1, Autoptimize offers many of the same static asset optimization features as the other plugins. Therefore, the differences can be attributed to the different ways the features are implemented by each plugin.

Differences can also be found in the variation of results. The variability of results can be seen in Appendices 2, 3, and 4. These variations should be considered when choosing an optimization plugin. The average result of two optimization plugins might be similar, but they could have vastly different amounts of variation. For example, in Appendix 4 the average Speed Index value of LiteSpeed Cache and WP Fastest Cache are very similar, but WP Fastest Cache has a much lower variability in its results. Therefore, WP Fastest Cache would be a better choice if the developer is looking for a low Speed Index time. There could be large differences in variability, which can be due to many variables. Therefore, predicting variability on one site based on the performance on other sites would be almost impossible.

Many of the differences between plugin performance can be attributed to the features they offer for free. For example, Autoptimize fared well on site 3 but did not work as well on the other two. As can be seen from Figure 3, the amount of improvement across plugins on the sites is not consistent. WP-Optimize worked best on site 1 and worst on site 2, whereas LiteSpeed Cache, W3 Total Cache, and WP Rocket worked best on site 1 and worst on site 3. There are similarities as can be seen from the chart, but there is no pattern in behaviour that could be used to predict the performance of a plugin on the sites.

As stated before, there is a pattern of caching reducing the server response time. According to Table 5, the only exception to this is WP Fastest Cache, which had the least positive effect on performance score out of all the tested plugins as can be seen in Figure 3. The three other plugins each had an over 90% improvement in server response time compared to the same plugin without caching enabled. Speed Optimizer even showed improvement in other metrics.

### 5.2.3 The effects of using two optimization plugins at once

Speed Optimizer was chosen for a new test, where two plugins were simultaneously enabled, because of its performance on sites 1 and 2. Smush was the second plugin chosen because it offers image optimization features for free, which Speed Optimizer does not. Site 3 was not tested because it had no images for Smush to optimize. The results of this test were shown in Table 6 and indicate large improvements on site 1, both with and without caching. The performance score of Speed Optimizer and Smush with caching enabled was over ten points greater than Speed Optimizer without Smush. This proves that there could be major benefits to using a combination of optimization plugins.

However, the results were the opposite on site 2. The performance score of Speed Optimizer and Smush went down by over 20 points, when compared to Speed Optimizer without Smush. This could be for many reasons, such as incompatibilities with the existing plugins on site 2. This reiterates the fact that the effects of optimization plugins depend on the site. Evaluation of the usefulness of an optimization plugin should be done on a case-by-case basis, because the effectiveness is not entirely predictable.

### 5.2.4 Mitigating the server-side performance loss caused by optimization plugins

As can be seen from the results presented in subsection 4.2.2, caching reduced the latency on the server side. Because static file, database, and other types of caching reduce the computational burden on the server, it naturally follows that the server latency also decreases (Tomiša et al. 2019). Best practices for optimizing the backend of WordPress sites were briefly covered in the second chapter, where techniques such as updating software and disabling unnecessary plugins were discussed.

However, caching is not the only way to improve the performance of a server. As noted by Everman and Zong (2018), utilizing higher end servers to fix backend performance issues comes with diminishing returns. Their study shows that efficient hardware utilization is key in conjunction with caching and CDN usage. Similarly, Shivakumar and Suresh (2018) also identify CDN deployment as a supporting piece in the overall backend performance, along with other server infrastructure configuration strategies, such as load balancing and fine-tuning settings. Load balancing also has many other benefits besides distributing the computational load across many servers, such as improved redundancy and fault tolerance, but it requires additional expertise from the website administrator (Shethiya 2025; WordPress 2023a).

The Codeable Editorial Team (2024) go more in-depth regarding more backend optimization techniques in the context of WordPress. Their use of the term backend is used mainly to refer to the administrative dashboard on WordPress and not the server side as a whole, but the two do overlap. The team mentions many of the same best practices discussed in research, such as cleaning the WordPress database, which is also mentioned in the whitepaper by Pyles (2022). The Codeable Editorial Team also apply the general best practice of monitoring, mentioned by Shivakumar and Suresh (2018), by advising to disable plugins that consume excessive amounts of processing resources. This type of resource monitoring is a proactive measure for improving QoS (Shailesh & Suresh 2017). Monitoring with performance measuring tools, such as Lighthouse and New Relic, is still necessary even after optimization has been completed to ensure sites retain a high QoS. This is because performance monitoring solutions allow the website administrators to notice not only bottlenecks, but also slow queries and inefficient code (Shethiya 2025).

### 5.3 Limitations and potential future research opportunities

The limitations and constraints of this thesis are mostly due to the scope of the experiment and what it is trying to achieve. Although the experiment was designed to limit the number of variables, the total elimination of them would be nearly impossible. However, future research about WordPress optimization could delve deeper into the details of optimizing specific types of sites or the benefits of specific optimization techniques, thereby limiting the variability in testing results. The aim of this thesis is to be a stepping stone toward more

specific research topics about WordPress optimization plugins, and therefore, the conclusions about the plugins and their features are only to be treated as general strategies rather than absolute truth about their effectiveness in all situations.

One specific research topic could be about optimizing particularly large websites or ones with a great amount of dynamic content. As a proof of concept, one small additional test will be conducted on an e-commerce website provided by Parvus Vulpes Oy. The site was not used in previous testing. This WordPress site is larger in scale than the previous three and features lots of dynamic content. The site will be tested both without optimization plugins as well as with Speed Optimizer and WP Rocket. Speed Optimizer will not have caching enabled while WP Rocket will. The cells are coloured green or red if the t-test is statistically significant, as was done in Tables 2, 3, and 4. The results of this small test can be seen in Table 7.

Table 7. Results of performance testing a large and dynamic site

Optimization Plugin	Performance Score	Server			Speed Index [ms]	TBT [ms]	CLS
		Response Time [ms]	FCP [ms]	LCP [ms]			
No optimization	<b>38.25</b>	<b>2837.61</b>	<b>3497.19</b>	<b>28652.82</b>	<b>8104.22</b>	<b>1173.16</b>	<b>0.006676</b>
Speed Optimizer	62.75	2747.78	2283.56	15974.71	6998.23	24.27	0.107168
WP Rocket	42.22	1525.50	3470.29	21165.31	7328.67	811.32	0.006636

The results from Table 7 suggest that there are performance improvements still possible to be achieved on large dynamic sites by using WordPress performance optimization plugins. In the case of optimizing sites with dynamic content, utilizing the techniques discussed in section 5.2.4 could prove helpful. The effectiveness of optimizations could vary from the results of the experiment conducted in this thesis, but this test acts as a proof of concept to aid in possible future research regarding this topic.

Future research could take pricing into consideration when evaluating WordPress optimization plugins. This thesis did not include any pricing information about the tested plugins, as pricing can vary depending on many factors. Nonetheless, studying the value of more paid optimization plugins compared to free ones could provide valuable information. Additionally, different approaches of applying optimizations were not considered. One approach could be to maximize performance at the expense of looks, for example by applying lossy compression to images. Similarly, balancing backend and frontend

optimization could also be studied. Another approach could be to find a balance between performance and having a visually appealing user interface. These ideas could serve as a foundation for future research topics on WordPress optimization.

Depending on the situation, a WordPress site may behave very differently than during the lab conditions of this thesis' experiment. This fact leads into another limitation of this thesis, namely the unknown performance of the optimized sites during varying levels of server load. The environment used in the experiment was, in terms of server load, a best-case scenario. Although processing and network limiters were used in the experiment to more accurately resemble real-world conditions, further research should be conducted about the effects of server load on the effectiveness of optimizations.

Additionally, the websites tested in this thesis were selected as a small sample of common websites. In order to gain a broader understanding of the impacts of optimization techniques, more testing and data is needed. The causalities of certain results should also be tested further in more detail in future research. One possible research topic could be about comparing different results of the same optimization feature on a variety of plugins.

## 6 Conclusions

This thesis focused on performance testing WordPress optimization plugins to find out about their effects on typical WordPress websites, and how closely their methods align with existing research about website optimization. A quantitative experimental research methodology was used to test and measure the effectiveness of eight plugins with varying feature sets. Additionally, optimization strategies and best practices from existing research were identified and compared against the methods used by the plugins.

Because WordPress is the most widely used web content management system in the world, powering over 40% of websites, optimizing the performance of WordPress sites can help lower costs for administrators. While there are many WordPress optimization plugins available, there is a lack of research evaluating their effectiveness. Therefore, this research on the methods these plugins' use provides valuable insight for WordPress developers looking to increase the performance of their sites.

It was found that the performance optimization plugins tested did, in fact, improve the performance of WordPress sites in many cases. However, the different implementations and content of the websites introduced variability into the results. In some cases, performance remained the same or even worsened due to the increased overhead. Because of this, optimization plugins should not be used if a website is already optimized using similar strategies to those offered by the plugins, if the website content is not performance-critical, or if the added strain on the backend outweighs the performance benefits. It was also noted that increased performance on the client side came at the expense of higher server-side latency due to this added strain, although this could be mitigated by enabling caching.

Additionally, it was concluded that by choosing two plugins whose features complement each other, an even greater increase in performance is possible. However, it was also shown that the opposite is possible: the combination of two plugins can degrade the performance compared to using only one of the two plugins. Optimization also had to be done carefully as to not have the original site change visually or cause other unintended behaviour, which did occur with some optimizations. Therefore, applying optimizations to WordPress sites should be done systematically by testing and monitoring the effects they have on the performance.

The results of this research indicate that there are substantial benefits to trying out WordPress optimization plugins, whether they are free or not. These plugins generally follow the theoretical strategies outlined in related research, which is also a strong indicator of their usefulness in practice. Experimentation with these plugins to see what works best on a target site could bring many improvements not only to performance, but also to the resource use and QoS. Limiting unnecessary resource consumption can reduce server load, improve latency, and contribute to a more sustainable information technology sector.

The results of the experimentation demonstrate that there is value to be gained from using optimization plugin features on websites built with WordPress. Companies and developers might want to investigate what kinds of efficiency benefits and knowledge they could gain from utilizing optimization plugins on their WordPress sites. However, these optimization plugins cannot be used for everything, as it is still up to the developers to identify bottlenecks, define a caching strategy, consider scalability, and understand optimal database usage.

One aspect this thesis did not address was the internal mechanics and plugin-specific implementations of the optimization features, which could be a subject for future research. A future research topic could be to test and analyse the differences between plugin implementations of specific WPO techniques. Another topic could be examining these kinds of plugins from a security perspective: is it wise to let external tools access website source code? As a limitation of this work, the results of the testing are not generalizable to all cases, as there is no definitive pattern of behaviour regarding which plugins will work best on each site. Overall, this thesis aims to serve as a stepping stone for further investigation into WordPress optimization plugins and their practical benefits.

## References

- Alakuijala, J., Farruggia, A., Ferragina, P., Kliuchnikov, E., Obryk, R., Szabadka, Z. & Vandevenne, L. 2019. Brotli: A General-Purpose Data Compressor. *ACM Transactions on Information Systems*. 37 (1). Available at DOI: 10.1145/3231935
- Arntz, F. 2025. Speculative Loading in 6.8. Cited 21 May 2025. Available at <https://make.wordpress.org/core/2025/03/06/speculative-loading-in-6-8/>
- Arshad, S., Ullah, S., Khan, S.A., Awan, M.D. & Khayal, M.S.H. 2015. A survey of Cloud computing variable pricing models. In 2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE). Barcelona: IEEE. 27–32. Available at URN:ISBN:978-989-758-143-4
- Barker, D. 2016. *Web Content Management: Systems, Features, and Best Practices*. First edition. Sebastopol, CA: O'Reilly Media, Inc.
- Cigoj, P. & Blazic, B.J. 2019. An Intelligent and Automated WCMS Vulnerability-Discovery Tool: The Current State of the Web. *IEEE Access*. 7, 175466–175473. Available at DOI: 10.1109/ACCESS.2019.2957573
- Codeable Editorial Team 2024. Diagnose & Resolve Slow WordPress Backend Issues Fast. Cited 4 Apr 2025. Available at <https://www.codeable.io/blog/wordpress-backend-slow/>
- Djirdeh, H., Wagner, J. & Mihajlija, M. 2018. Preload critical assets to improve loading speed. Cited 12 Apr 2025. Available at <https://web.dev/articles/preload-critical-assets>
- Everman, B. & Zong, Z. 2018. GreenWeb: Hosting High-Load Websites Using Low-Power Servers. In Ninth International Green and Sustainable Computing Conference (IGSC). Pittsburgh, PA: IEEE. 1–6. Available at URN:ISBN:978-1-5386-7466-6
- Ferguson, M. 2020. The Pros and Cons of Running WordPress on Ubuntu. Cited 15 Apr 2025. Available at <https://www.wpexplorer.com/pros-cons-wordpress-ubuntu/>

Ferreira, D., Pereira, T., Mendes, I. & Amaral, A. 2025. WordPress Architecture Modernization Projects. In Pereira, T., Kiesler, N., Impagliazzo, J., & Santos, H. (eds.). Internet of Everything. Guimarães: Springer Nature Switzerland. IOECON: International Conference on Internet of Everything. 121–131. Available at URN:ISBN:978-3-031-84425-6

Google 2019.a. Lighthouse performance scoring. Cited 28 Mar 2025. Available at <https://developer.chrome.com/docs/lighthouse/performance/performance-scoring>

Google 2019.b. Speed Index. Cited 7 May 2025. Available at <https://developer.chrome.com/docs/lighthouse/performance/speed-index>

Google 2019.c. Total Blocking Time. Cited 7 May 2025. Available at <https://developer.chrome.com/docs/lighthouse/performance/lighthouse-total-blocking-time>

Google 2020. Largest Contentful Paint. Cited 7 May 2025. Available at <https://developer.chrome.com/docs/lighthouse/performance/lighthouse-largest-contentful-paint>

Google 2016. Introduction to Lighthouse. Cited 20 Feb 2025. Available at <https://developer.chrome.com/docs/lighthouse/overview>

Hakimi, H.A. & Rahman, N.A. 2021. Developing the COVID-19 Malay Corpus Using WordPress Content Management System (CMS). In 2021 Fifth International Conference on Information Retrieval and Knowledge Management (CAMP). Kuala Lumpur: IEEE. 75–83. Available at URN:ISBN:978-1-6654-1237-7

Heričko, T., Čučko, Š., Šumak, B. & Brdnik, S. 2021. Web Performance Tuning of WordPress-Based Websites Through Automatic Image Optimization. In Neven, V., Pergler, E., & Grd, P. (eds.). Varaždin: University of Zagreb Faculty of organization and informatics. Central European Conference on Information and Intelligent Systems. 343–350. Available at <https://www.proquest.com/docview/2604878601/abstract/F74A448238194CD7PQ/1>

- Judijanto, L. 2025. The Impact of Digital Transformation on Business Models: A Bibliometric Study. *The Eastasouth Management and Business*. 3 (2), 255–267. Available at DOI: 10.58812/esmb.v3i02.424
- Kazemi, A., Boyd, M., Choi, F., Tai, A.M.Y., Tsang, V.W., To, T., Kim, J., Jang, K., Shams, F., Schreiter, S., Cabanis, M. & Krausz, R.M. 2024. Architecture and Development Framework for a Web-Based Risk Assessment and Management Platform Developed on WordPress to Address Opioid Overdose. *JMIR Formative Research*. 8. Available at DOI: 10.2196/49759
- Khalil, M.M., Ghazi, H.M., Habib, M.I., Shahzad, F. & Elahi, A.I. 2024. Guideline for Selecting the Right Content Management System (RCMS) for Web Development: A Comprehensive Approach. *Journal of Computing & Biomedical Informatics*. Available at DOI: 10.56979
- Khan, I., Hollebeek, L.D., Fatma, M., Islam, J.U., Rather, R.A., Shahid, S. & Sigurdsson, V. 2023. Mobile app vs. desktop browser platforms: the relationships among customer engagement, experience, relationship quality and loyalty intention. *Journal of Marketing Management*. 39 (3–4), 275–297. Available at DOI: 10.1080/0267257X.2022.2106290
- Kinnunen, S.K., Happonen, A., Arola, S.M. & Kärri, T. 2019. Traditional and extended fleets in literature and practice: definition and untapped potential. *International Journal of Strategic Engineering Asset Management*. 3 (3), 239–261. Available at DOI: 10.1504/IJSEAM.2019.108467
- Kortelainen, H., Happonen, A. & Hanski, J. 2019. From Asset Provider to Knowledge Company—Transformation in the Digital Era. In Mathew, J., Lim, C.W., Ma, L., Sands, D., Cholette, M.E., & Borghesani, P. (eds.). *Asset Intelligence through Integration and Interoperability and Contemporary Vibration Engineering Technologies*. Cham: Springer International Publishing. Lecture Notes in Mechanical Engineering. 333–341. Available at URN:ISBN:978-3-319-95710-4
- Kumar, A., Kumar, A., Hashmi, H. & Khan, S.A. 2021. WordPress: A Multi-Functional Content Management System. In 2021 10th International Conference on System Modeling

& Advancement in Research Trends (SMART). Moradabad: IEEE. 158–161. Available at URN:ISBN:978-1-6654-3970-1

Kumar, A., Narayan, A., Sharma, V., Prasad, A.A., Sami, M. & Jamnadas, H. 2024. Decoding the Web CMS Landscape: A Comparative Study of Popular Web Content Management Systems. *International Journal of Computers and Their Applications*. 31 (4), 281–291. Available at <https://www.researchgate.net/publication/388195104>

Lehat, M.L., Abu Bakar, N.F., Jamil, A.A., Mohd Shafee, C.M.N., Shamala, P. & Rosnan, S. 2023. Assessing Web Performance of Malaysian University Website. In *2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*. Kuala Lumpur: IEEE. Available at DOI: 10.1109/ICRAIE59459.2023.10468308

LiteSpeed 2025. LiteSpeed Web Server - Apache Alternative - LiteSpeed Technologies. Cited 21 May 2025. Available at <https://www.litespeedtech.com/products/litespeed-web-server>

Local WP 2025. Features. Cited 15 Apr 2025. Available at <https://localwp.com/features/>

Mesa, O., Vieira, R., Viana, M., Durelli, V.H.S., Cirilo, E., Kalinowski, M. & Lucena, C. 2018. Understanding vulnerabilities in plugin-based web systems: an exploratory study of wordpress. In *Proceedings of the 22nd International Systems and Software Product Line Conference*. Gothenburg: ACM. 149–159. Available at URN:ISBN:978-1-4503-6464-5

Microsoft 2025. T.TEST function. Cited 23 Mar 2025. Available at <https://support.microsoft.com/en-us/office/t-test-function-d4e08ec3-c545-485f-962e-276f7cbed055>

Mihajlija, M. & Walton, P. 2023. Cumulative Layout Shift (CLS). Cited 7 May 2025. Available at <https://web.dev/articles/cls>

Minashkina, D. & Happonen, A. 2021. A Systematic Literature Mapping of Current Academic Research Connecting Sustainability into the Warehouse Management Systems Context. In Sheu, Dr.G.Y. (eds.). *Current Approaches in Science and Technology Research*. Book Publisher International (a part of SCIENCEDOMAIN International). 52–80. Available at URN:ISBN:978-93-91215-45-3

Mozilla 2025. Speculative loading. Cited 21 May 2025. Available at

[https://developer.mozilla.org/en-US/docs/Web/Performance/Guides/Speculative\\_loading](https://developer.mozilla.org/en-US/docs/Web/Performance/Guides/Speculative_loading)

Mulyandi, M.R., Septiani, N., Yusup, M., Riza Chakim, M.H., & Nursohit 2022.

Optimizing SEO (Search Engine Optimization) Implementation on a Website Content Management System. In 2022 IEEE Creative Communication and Innovative Technology (ICCIIT). Tangerang: IEEE. Available at DOI: 10.1109/ICCIIT55355.2022.10118592

Osmani, A., Djirdeh, H., Bynens, M. & Pollard, B. 2024. Browser-level image lazy loading for the web. Cited 20 May 2025. Available at <https://web.dev/articles/browser-level-image-lazy-loading>

Patel, S.K., Rathod, V.R. & Parikh, S. 2011.a. Joomla, Drupal and WordPress - a statistical comparison of open source CMS. In 3rd International Conference on Trendz in Information Sciences & Computing (TISC2011). Chennai: IEEE. 182–187. Available at URN:ISBN:978-1-4673-0133-6

Patel, S.K., Rathod, V.R. & Prajapati, J.B. 2011.b. Performance Analysis of Content Management Systems Joomla, Drupal and WordPress. International Journal of Computer Applications. 21 (4), 39–43. Available at DOI: 10.5120/2496-3373

Pyles, J. 2022. 15 Ways To Optimize WordPress From The Inside Out. Austin, TX: WP Engine. Cited 11 Feb 2025. Available at <https://wpengine.com/wp-content/uploads/2017/08/15WaysToOptimizeWP-1.pdf>

Salmela, E. & Happonen, A. 2021. Are You doing the Right Things? Initial Stages of Individuals Change Management: Identify, Acknowledge and Make a Courageous Move Framework. In Singh, Dr.R. (eds.). Selected Topics in Humanities and Social Sciences. Book Publisher International (a part of SCIENCEDOMAIN International). 12–25. Available at URN:ISBN:978-93-91215-74-3

Saravanan, S., Manivannan, S.K., Venusamy, K. & Sathiyamoorthy, C. 2024. A Study On Content Management System On B2B And B2C Websites By Using CMS Tool. In 10th International Conference on Communication and Signal Processing (ICCSPP). Melmaruvathur: IEEE. 165–170. Available at URN:ISBN:979-8-3503-5306-8

- Schäferhoff, N. 2025. How to Make a Website: The Complete, No-Code Beginner's Guide. Cited 24 Apr 2025. Available at <https://wordpress.com/blog/2025/01/29/how-to-make-a-website/>
- Shailesh, K.S. & Suresh, P.V. 2017. An analysis of techniques and quality assessment for Web performance optimization. *Indian Journal of Computer Science and Engineering*. 8 (2), 61–69. Available at <https://www.ijcse.com/docs/INDJCSE17-08-02-100.pdf>
- Shailesh, K.S. & Suresh, P.V. 2016. Web performance optimization through smart resource and asset optimizations. In Hoda, M.N. (eds.). *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi: IEEE. 277–281. Available at URN:ISBN:978-93-80544-21-2
- Shethiya, A.S. 2025. Scalability and Performance Optimization in Web Application Development. *Integrated Journal of Science and Technology*. 2 (1). Available at <https://ijstpublication.com/index.php/ijst/article/view/1>
- Shivakumar, S. & Suresh, P.V. 2018. A Survey and Analysis of Techniques and Tools for Web Performance Optimization. *Journal of Information Organization*. 8 (2). Available at DOI: 10.6025/jio/2018/8/2/31-57
- Stojić, D., Vujičić, D., Damnjanović, Đ. & Marković, D. 2024. PERFORMANCE COMPARISON OF WORDPRESS POSTS AND MYSQL DATABASE. *UNITECH – SELECTED PAPERS*. 2024. Available at DOI: 10.70456/GNOI2010
- Student 1908. The Probable Error of a Mean. *Biometrika*. 6 (1), 1–25. Available at DOI: 10.2307/2331554
- TeamUpdraft, D. 2025. WP-Optimize – Cache, Compress images, Minify & Clean database to boost page speed & performance. Cited 3 Feb 2025. Available at <https://wordpress.org/plugins/wp-optimize/>
- Tomiša, M., Milković, M. & Čačić, M. 2019. Performance Evaluation of Dynamic and Static WordPress-based Websites. In *2019 23rd International Computer Science and Engineering Conference (ICSEC)*. Phuket: IEEE. 321–324. Available at URN:ISBN:978-1-7281-2544-2

Usmani, U.A., Happonen, A. & Watada, J. 2023. Advancements in Industry 4.0 Asset Management: Interoperability and Cyber Security Challenges and Opportunities. In Arai, K. (eds.). Proceedings of the Future Technologies Conference (FTC) 2023, Volume 4. Cham: Springer. Lecture Notes in Networks and Systems. 468–488. Available at URN:ISBN:978-3-031-47447-7

Vatousios, A. & Happonen, A. 2022. Transforming HR and Improving Talent Profiling with Qualitative Analysis Digitalization on Candidates for Career and Team Development Efforts. In Arai, K. (eds.). Intelligent Computing. Cham: Springer International Publishing. Lecture Notes in Networks and Systems. 1149–1166. Available at URN:ISBN:978-3-030-80118-2

Vihervaara, J., Loula, P. & Tuominen, T. 2016. Performance Gains from Web Performance Optimization - Case Including the Optimization of Webpage Resources in a Comprehensive Way: In Majchrzak, T.A., Traverso, P., Monfort, V., & Krempels, K.-H. (eds.). Proceedings of the 12th International Conference on Web Information Systems and Technologies. Rome: SCITEPRESS - Science and and Technology Publications. 188–193. Available at URN:ISBN:978-989-758-186-1

W3Techs 2025. Historical yearly trends in the usage statistics of content management systems. Cited 31 Jan 2025. Available at [https://w3techs.com/technologies/history\\_overview/content\\_management/all/y](https://w3techs.com/technologies/history_overview/content_management/all/y)

Walton, P. 2023. First Contentful Paint (FCP). Cited 7 May 2025. Available at <https://web.dev/articles/fcp>

Walton, P. 2024. Web Vitals. Cited 28 Mar 2025. Available at <https://web.dev/articles/vitals>

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B. & Wesslén, A. 2024. Experimentation in Software Engineering. Second edition. Berlin, Heidelberg: Springer.

WordPress 2023.a. Optimization. Cited 11 Feb 2025. Available at <https://developer.wordpress.org/advanced-administration/performance/optimization/>

WordPress 2019.a. Transients. Cited 20 May 2025. Available at <https://developer.wordpress.org/apis/transients/>

WordPress 2023.b. Cache. Cited 11 Feb 2025. Available at <https://developer.wordpress.org/advanced-administration/performance/cache/>

WordPress 2024. Static or Dynamic rendering of a block. Cited 12 Apr 2025. Available at <https://developer.wordpress.org/block-editor/getting-started/fundamentals/static-dynamic-rendering/>

WordPress 2019.b. Block Editor Handbook. Cited 12 Apr 2025. Available at <https://developer.wordpress.org/block-editor/>

WordPress 2018. Server Environment. Cited 15 Apr 2025. Available at <https://make.wordpress.org/hosting/handbook/server-environment/>

WordPress 2025. WordPress Plugins. Cited 25 Mar 2025. Available at <https://wordpress.org/plugins>

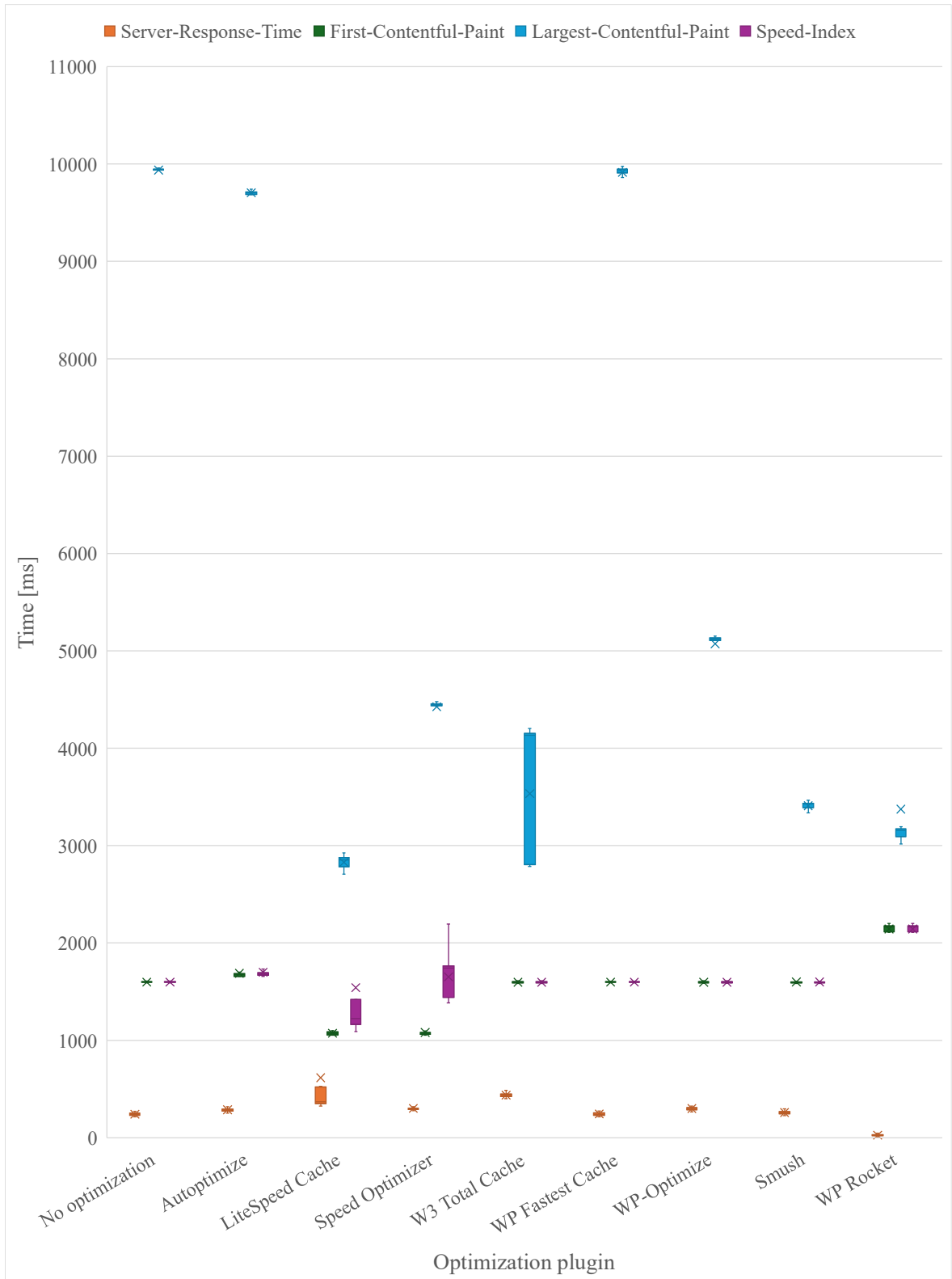
Yan, G., Ma, J., Han, Y. & Li, X. 2016. EcoUp: Towards Economical Datacenter Upgrading. *IEEE Transactions on Parallel and Distributed Systems*. 27 (7), 1968–1981. Available at DOI: 10.1109/TPDS.2015.2477827

Yanney, E., Simpson, T. & Boadi, K.S. 2023. Using Performance Metrics to Guide the Selection of a Website Content Management System -The Case of Joomla, Drupal and WordPress. *Information and Knowledge Management*. 13 (1), 36–47. Available at DOI: 10.7176/IKM/13-1-04

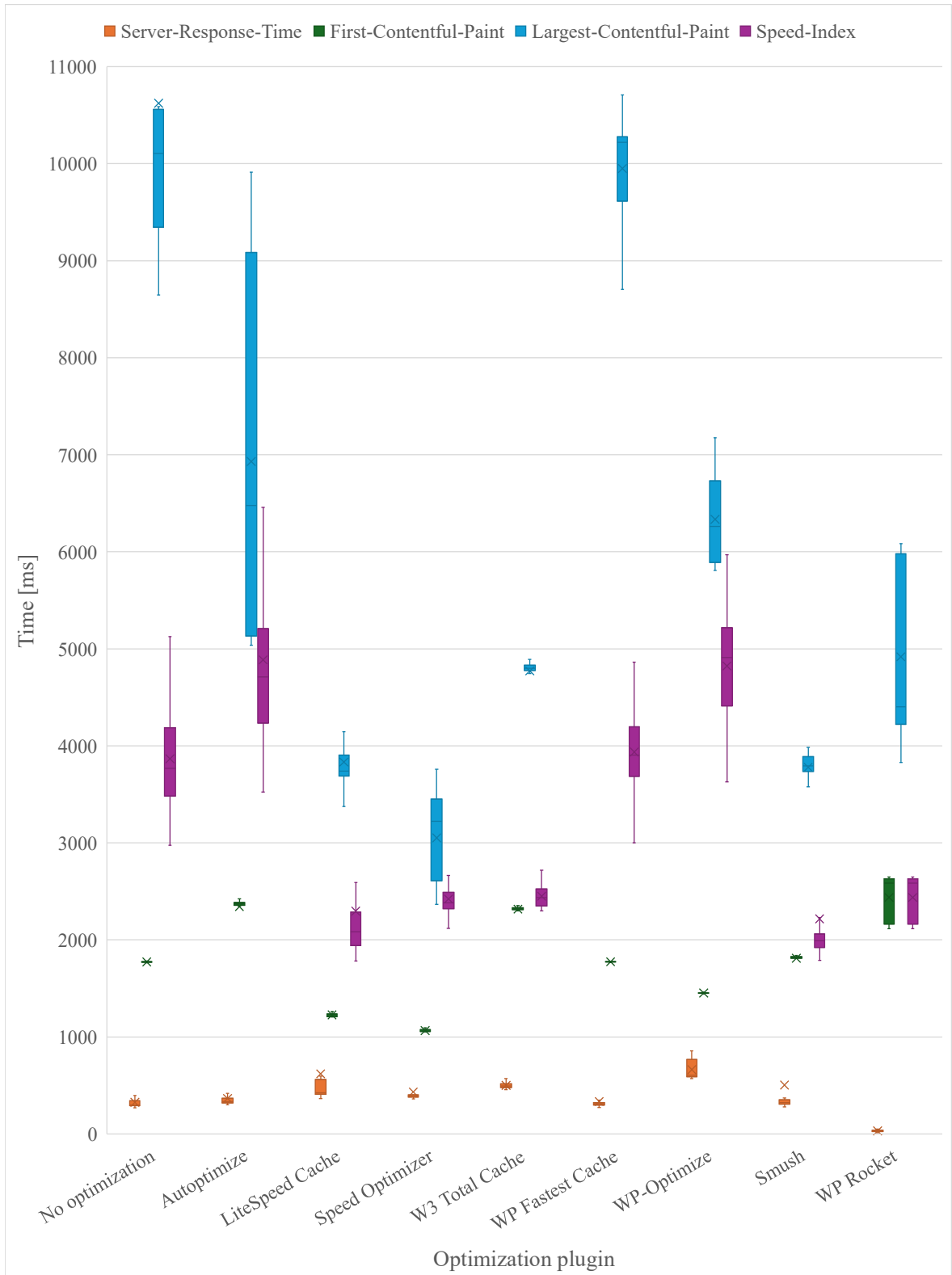
Appendix 1. Table of the optimization plugins' features and their cost

Optimization feature	Autooptimize	LiteSpeed Cache	Speed Optimizer	W3 Total Cache	WP Fastest Cache	WP-Optimize	Smush	WP Rocket
Dynamic caching		LSWS	Paid			Free		
File-based caching	Paid	LSWS	Free	Free	Free	Free		Paid
Browser caching		LSWS	Free	Free	Free	Free		Paid
Opcode cache				Free				
REST API cache		LSWS		Paid				
Object cache		LSWS	Paid	Free				
Database cleanup		Free			Paid	Free		Paid
Preload requests				Paid		Paid		Paid
DNS prefetch		Free	Free					Paid
CDN	ShortPixel	Any		Any	Any		Smush CDN	Any
Gzip compression	Paid		Free		Free	Free		
Minify CSS files	Free	Free	Free	Free	Free	Free		Paid
Combine CSS files	Free	Free	Free	Free	Free	Free		
Preload combined CSS		Free	Free			Free		
Minify JavaScript	Free	Free	Free	Free	Paid	Free		Paid
Combine JavaScript	Free	Free	Free	Free	Free	Free		Paid
Defer render-blocking JavaScript	Free	Free	Free	Paid	Paid	Free		Paid
Minify HTML	Free	Free	Free	Free	Free	Free		
Web fonts optimization	Free	Free	Free		Paid	Free		Paid
Disable emojis	Free	Free	Free	Free	Free			
Remove query string from static resources	Free	Free	Free	Free				
Image compression	Paid	QUIC.cloud	Paid		Paid	Free	Free (limited)	
WebP image support	Paid	Free	Paid	Free (limited)	Paid	Free	Paid	Paid
Lazy load images	Free	Free	Free	Free	Paid	Paid	Free	Paid

Appendix 2. Box and whisker chart of the test results for website 1



### Appendix 3. Box and whisker chart of the test results for website 2



Appendix 4. Box and whisker chart of the test results for website 3

