INVESTIGATING THE WEBSITE SUSTAINABILITY OF FINNISH LISTED COMPANIES

Master's Thesis

Lappeenranta-Lahti University of Technology LUT

School of Engineering Sciences

Master's Programme in Software Product Management and Business

2024

Thomas Granelund

Examiners:

Associate Professor Jussi Kasurinen, D.Sc. (Tech.)
Erno Vanhala, D.Sc. (Tech.)

ABSTRACT
Lappeenranta-Lahti University of Technology LUT
LUT School of Engineering Sciences
Software Engineering

Thomas Granelund

Investigating the website sustainability of Finnish listed companies

Master's Thesis
2024

82 pages, 16 figures, 6 tables and 7 appendices

Examiners:
Associate Professor Jussi Kasurinen, D.Sc. (Tech.), Erno Vanhala, D.Sc. (Tech.)

Keywords: sustainability, CO_2 emissions, global warming, web design, optimization, programming, software development

Sustainability has become a key concern globally, influencing society, individuals, and organizations, including the software industry. Software plays a critical role in fighting environmental challenges but also contributes to energy consumption and the use of resources. This thesis examines sustainability in web design, addressing three research questions: "How is sustainability in web design defined?", "What are suitable performance metrics in sustainable web design?" and "How does a sample of Finnish Stock Exchange companies position in terms of web design sustainability?" To respond to these questions, a sample of Finnish websites was analyzed with a focus on CO_2 emissions, site speed, and performance. Additional variables included technological components of the websites, such as programming languages and content management systems (CMS). Data collection was handled through third-party tools and APIs, supported by a theoretical framework grounded in prior academic research.

The results indicate that CO_2 emission intensity across the sample was primarily influenced by the total document size (MB), with only little impact from the chosen CMS or hosting provider. The selection of programming languages was closely tied to the CMS. Based on a global green hosting register, 78% of the websites were categorized as hosted on renewable resources, while 22% relied on non-green hosting and some companies opted for self-hosted server solutions. The number of HTTP requests affected carbon footprint, highlighting the importance of optimization. Analysis of Google Lighthouse scores emphasized the need for technological efficiency and performance optimization to achieve faster rendering. Improvements in the Site Speed Index were found to enhance website performance, especially for users on slow mobile networks.

This study highlights the importance of deliberate decision-making in web design, including selecting green hosting providers and software optimization to address climate change. Through these strategies, organizations can reduce emissions and contribute to more sustainable digital practices.

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
LUTin insinööritieteiden tiedekunta
Tietotekniikka

Thomas Granelund

Selvitys suomalaisten pörssiyhtiöiden verkkosivustojen kestävästä kehityksestä

Diplomityö
2024

82 sivua, 16 kuviota, 6 taulukkoa ja 7 liitettä

Tarkastajat:
Apulaisprofessori Jussi Kasurinen (TkT), Erno Vanhala (TkT)

Avainsanat: kestävä kehitys, hiilidioksidipäästöt, ilmastonmuutos, web-suunnittelu, optimointi, ohjelmointi, ohjelmistokehitys

Kestävä kehitys on noussut keskeiseksi huolenaiheeksi kansainvälisesti, vaikuttaen yhteiskuntaan, yksilöihin ja yrityksiin, mukaan lukien ohjelmistoalaan. Ohjelmistot ovat keskeisessä roolissa ympäristöhaasteiden torjunnassa, mutta ne myös lisäävät energiankulutusta ja resurssien käyttöä. Tämä diplomityö tutkii kestävää kehitystä web-suunnittelussa kolmen tutkimuskysymyksen kautta: “Miten kestävä kehitys määritellään web-suunnittelussa?”, “Mitkä ovat soveltuvia kestävän kehityksen tehokkuusmittareita web-suunnittelussa?” ja “Miten suomalaiset pörssiyhtiöt suoriutuvat kestävän web-suunnittelun näkökulmasta?”. Kysymysten vastaamiseksi otanta suomalaisia verkkosivuja analysoitiin keskittyen hiilidioksidipäästöihin, sivuston nopeuteen sekä tehokkuuteen. Lisämuuttujina tutkittiin verkkosivustojen teknologisia komponentteja kuten ohjelmointikieltä ja sisällönhallintajärjestelmää (CMS). Tiedonkeruu toteutettiin kolmannen osapuolen työkalujen ja API-rajapintojen avulla, jota tuki aiempaan akateemiseen tutkimukseen perustuva teoreettinen viitekehys.

Tulokset osoittavat, että otannan hiilidioksidi-intensiteettiin vaikutti ensisijaisesti dokumentin kokonaiskoko (Mt). CMS-järjestelmällä ja palvelintarjoajalla oli tähän vain vähäinen vaikutus. Ohjelmointikielten valinta oli vahvasti sidoksissa CMS-järjestelmään. Tämän tutkimuksen perusteella 78 % verkkosivustoista luokiteltiin vihreiksi, eli isännöitiin uusiutuvien energialähteiden avulla. 22 % verkkosivuista turvautuivat ei-vihreään isännöintiin ja osa yrityksistä valitsivat itse ylläpitää palvelimia. HTTP-pyyntöjen määrä vaikutti hiilijalanjälkeen, mikä korostaa optimoinnin tärkeyttä. Google Lighthouse -pisteiden analysointi korostaa teknologisen tehokkuuden ja suorituskyvyn optimointia nopeamman renderöinnin saavuttamiseksi. Korkea Site Speed Index todettiin parantavan verkkosivujen suorituskykyä, erityisesti hitailla mobiiliverkoilla oleville käyttäjille.

Tämä tutkimus korostaa harkitun päätöksenteon tärkeyttä web-suunnittelussa, mukaan lukien vihreiden isännöintipalveluntarjoajien valinta ja ohjelmistojen optimointi ilmastonmuutoksen hillitsemiseksi. Näiden strategioiden avulla organisaatiot voivat vähentää päästöjä, edistäen kestävämpiä digitaalisia käytäntöjä.

ABSTRAKT

Villmanstrand-Lahtis tekniska universitet LUT
LUT:s fakultet för ingenjörsvetenskaper
Informationsteknik

Thomas Granelund

Utredning av hållbar utveckling för webbplatser hos finska börsbolag

Diplomarbete
2024

82 sidor, 16 figurer, 6 tabeller och 7 bilagor

Granskare:
Biträdande professor Jussi Kasurinen (TkD), Erno Vanhala (TkD)

Nyckelord: hållbar utveckling, koldioxidutsläpp, klimatförändring, webbdesign, optimering, programmering, mjukvaruutveckling

Hållbar utveckling har blivit en viktig angelägenhet globalt som påverkar samhällen, individer och företag, men även mjukvarubranschen. Mjukvara har en viktig roll i kampen mot utmaningar i vår omgivning men bidrar även till en förbrukning av energi och användning av resurser. Detta diplomarbete utreder hållbar utveckling i webbdesign genom tre frågor: "Hur definieras hållbar utveckling inom webbdesign?", "Vad är passande prestandamått inom hållbar webbdesign?" samt "Hur klarar sig ett sampel av finska börsbolag i hållbar webbdesign?" För att besvara dessa frågor analyserades ett sampel av finska webbsidor med fokus på koldioxidutsläpp, webbsidornas snabbhet och prestanda. Därutöver beaktades variabler i tekniska komponenter hos en webbsida, såsom programmeringsspråk och publiceringsverktyg (CMS). Datainsamling hanterades genom tredjepartsverktyg och API:s, understödda av ett teoretiskt ramverk baserat på tidigare akademisk forskning.

Resultaten visar att koldioxidintensiteten i samplet främst påverkas av dokumentets totala storlek (MB) och endast i liten grad av valet av CMS eller serverleverantör. Valet av programmeringsspråk var nära kopplat till CMS:et. Baserat på ett globalt register för gröna serverleverantörer var 78 % av webbsidorna kategoriserade under förnybara källor, medan 22 % använde sig av icke-gröna servrar och en del företag valde att själv hantera sina serverlösningar. Antalet HTTP-förfrågningar påverkade koldioxidavtrycket, vilket framhäver ett optimeringsbehov. Analysen av Google Lighthouse-poäng betonar ett behov av teknologisk effektivitet och optimering av prestanda för att uppnå en snabbare rendering. Förbättringar av Site Speed Index visade sig förbättra webbsidornas prestanda, särskilt för användare i ett långsamt mobilt nätverk.

Denna studie betonar vikten av medvetet beslutsfattande i webbdesign, inklusive val av en grön tjänsteleverantör för servrar, samt optimering av mjukvara i kampen mot klimatförändringen. Genom dessa strategier kan företag minska sina utsläpp, vilket bidrar till mer hållbara digitala tillvägagångssätt.

List of abbreviations

ASP	Active Server Pages
CDN	Content Delivery Network
CLS	Cumulative Layout Shift
CMS	Content Management System
CSS	Cascading Style Sheets
CWV	Core Web Vitals
FCP	First Contentful Paint
FID	First Input Delay
GIF	Graphics Interchange Format
HTML	HyperText Markup Language
INP	Interaction to Next Paint
LCP	Largest Contentful Paint
PHP	Hypertext Preprocessor
PNG	Portable Network Graphics
SEO	Search engine optimization
SPA	Single-page application
TBT	Total Blocking Time
TTFB	Time to First Byte
TTI	Time to Interactive

Table of contents

Figures

Tables

Appendices

1 Introduction

In today’s competitive business landscape, firms are forced to develop quality software and quickly offer customers with added value. Achieving this agility is a key challenge in firms, which is emphasized by swift development and shorter development cycles (Baskerville et al. 2003). As the internet landscape rapidly evolves, developing web services proceeds at a fast pace (Dubey et al. 2023). Similarly, customer behavior is changing (Antinoja 2021), which puts pressure on companies to show business transparency and how they adapt to optimizing their supply chains adequately (Kulkarni & Haghi 2024).

To succeed, firms must establish processes that improve teamwork, collaboration, and design teams with the right skillsets (Baskerville et al. 2003). Cross-disciplinary teams enforce collaboration between disciplines (Küng 2020), and in combination with agile methodologies, offering flexibility and methods to cope with change (Baskerville et al. 2003), organizations develop better abilities to act in a market of uncertainty (Küng 2020).

As consumer demand evolves, a preference for environmental consideration shapes the market (Kulkarni & Haghi 2024). In a landscape of digitalization and technological advancements firms are challenged to adapt their business models to better match unpredictable markets, differentiate, and offer services on-demand to customers (Küng 2017).

Sustainability, a rising topic among customers worldwide (Calero et al. 2021), has become a critical focus as firms now should aim to escalate environmental, social, and economic challenges (Kulkarni & Haghi 2024). Similarly, there is an increased concern on resource consumption and climate impact (Goyal 2015). Although software can assist firms to handle environmental challenges (Calero et al. 2021), it is key to acknowledge that software solutions may impact a degradation of our environment (Lago et al. 2015). To reach sustainability goals defined by the United Nations in the Paris Agreement, software plays a key role (Calero et al. 2022). The role of software engineering in sustainability is to ensure that it constantly, over a longer period of time, reaches set goals despite of modifications, updates, and its evolution (Becker et al. 2015).

1.1 Research purpose and background

Software has a significant role in fighting climate change but can also impact the environment negatively (Calero et al. 2021). Sustainable software often focuses on long-term usage with little resources but may still stress the environment due to energy consumption and usage of resources (Lago et al. 2015).

A demand for user friendly websites has led to a substantial increase in emissions (CO_2) and energy consumption of hosting servers (Calero et al. 2021). Thus, sustainability aspects of web design are highly topical and exploring techniques to minimize environmental degradation can be argued a key priority for firms.

This thesis investigates aspects of sustainability in web design and aims to find solutions to optimize webpages to decrease carbon emissions (CO_2). Data was programmatically collected and analyzed for a set of Finnish websites (N = 32) to review their carbon footprint, speed and performance. Similarly, technical aspects such as content management system (CMS), programming language, and hosting were included in the analysis.

Six key focus areas were analyzed to examine web page performance in terms of sustainability. Results of this study show that there is a direct relation between the site speed index and the Lighthouse score by Google. Thus, improving site speed should be prioritized by companies. Improving website speed increases overall performance but also enhances user experience especially on slow internet connections. The number of requested resources, selection of CMS and technology stack appears to have a small impact on page loading time. Nevertheless, this study proves that there are various techniques to decrease carbon footprint of a webpage, through strategic decision-making in software development.

1.2 Previous research and relevancy

Previous studies verify that sustainability is a rising topic (Kulkarni & Haghi, 2024) and that addressing sustainability concerns, such as reducing emissions, has become a key issue for companies (Calero et al. 2022). Similarly, regulation is forcing companies to put efforts on emission-aware operations (Wang et al. 2023). Studies also indicate that optimization of websites not only impacts emissions (Teke 2023) but also improves customer experience and search engine visibility (Google 2024a). Similarly, it is important to acknowledge that software can be a problem and part of the solution in environmental degradation and climate change (Calero et al. 2021). Thus, there are many ways to improve websites (Zolfaghari et al. 2020; Google 2024c), and certain standardized metrics can help comprehend performance of websites in terms of sustainability (Edgar 2023; Google 2024b).

Previous research has focused little on practical implications or measurements of webpages. Additionally, few studies focus on the Finnish market, and the performance of technologies such as single-page applications (SPAs). In this thesis, the focus is examining a sample of large Finnish enterprises and developing a toolset that can be applied in web page optimization. Although sustainability is common in research (Johnston et al. 2007), it seldom focuses on web design and its implications on environmental degradation (Kulkarni & Haghi 2024).

1.3 Research problem statement

This thesis aims to examine sustainability and web development in general by investigating performance metrics of a selected sample of Finnish websites. It aims to expand on previous knowledge and present pragmatic solutions through which web design practices and performance metrics can be reviewed.

Based on the research problem statement, this thesis aims to answer the following research questions (RQs):

RQ1	How is sustainability in web design defined?
RQ2	What are suitable performance metrics in sustainable web design?
RQ3	How does a sample of Finnish Stock Exchange companies position in terms of web design sustainability?

1.4 Research structure

Introduction

The introduction chapter gives context to the issue of sustainable web design and highlights its relevancy. Similarly, it presents what problematics it aims to address, the research purpose, limitations in previous studies, and clarifies the research problem statement.

Related research

In the second chapter, a thorough theoretical background, expanding on previous research, is presented. The content in chapter two gives an accurate definition on what sustainability is, how it relates to web development, and presents various techniques for website optimization. Additionally, it introduces various metrics and guidelines to measure sustainability and performance of a website.

Methodology

Chapter three justifies the selected research methodology and outlines the concepts of sampling and comparative analysis. Similarly, it introduces the six step Design Science Research Methodology (DSRM) by Peffers et al. (2007), as well as explains specifics of data collection, and what tools, algorithms and frameworks were utilized for this study.

Results

In the fourth chapter, results are thoroughly presented based on the selected research methodology. Results are visualized through plots in histograms and pie charts, in addition to calculations in form of algorithms. The results chapter consist of six key metrics as subchapters and is concluded by a summary of all results.

Discussion

Chapter five aims to discuss and highlight specific results and revisits key questions. Additionally, it introduces research limitations, addresses generalizability of the results, and presents potential future work and its implications in a larger context.

Conclusions

To conclude the thesis, a concise summary of all previous chapters is provided.

2 Background and related research

In this chapter, a context is given through expanding previous academic research. The content aims to thoroughly examine topics such as sustainability, innovation, web development, and implications of quick changes in a competitive environment. Additionally, technical development, consumer behavior, and web service optimization is highlighted.

2.1 Background

Global sustainability has become a pivotal concern in response to escalating environmental, social, and economic challenges worldwide (Kulkarni & Haghi 2024). As rapid urbanization, population growth, and industrialization continue, there is a simultaneous increase in resource consumption, biodiversity loss, and climate change impacts. These developments underscore an urgent need for sustainable practices across critical sectors. However, despite growing awareness and efforts, substantial challenges remain in achieving true global sustainability (Kulkarni & Haghi 2024).

A key area driving sustainability is innovation in renewable technologies. Ongoing research gives significant breakthroughs in materials and design, contributing to the improved efficiency of non-conventional energy systems (Kulkarni & Haghi 2024). Alongside technological advancements, consumer behavior is evolving (Antinoja 2021), with a growing preference for products and brands that demonstrate transparency in their supply chains and adopt circular economy principles. This shift in consumer preferences reflects an increased environmental awareness that shapes the global market (Kulkarni & Haghi 2024).

Another significant trend is the role of digitalization in sustainability efforts. From smart cities to precision agriculture, digital innovations are optimizing resource use and minimizing environmental impacts. Technologies such as artificial intelligence (AI) and the Internet of Things (IoT) are playing crucial roles in creating more efficient systems, contributing to a more sustainable future (Kulkarni & Haghi 2024). As Utesheva (2020) highlights, digital media and technological advancements have fundamentally transformed how we interact with the world. Access to real-time information and digital platforms has enabled greater connectivity, collaboration, and the ability to trace global patterns with remarkable accuracy. This transformation has shaped our values, habits, and our understanding of sustainability (Utesheva 2020).

In summary, global sustainability efforts are being shaped by a combination of technological innovation, digitalization, and evolving consumer behavior. The integration of corporate social responsibility (CSR) and strategic business models further reinforces the importance of balancing economic growth with environmental and social responsibility. As digital and sustainable practices continue to evolve, they offer both challenges and opportunities for future development. (Kulkarni & Haghi 2024)

2.2 Sustainability

Sustainability as a topic is on the rise worldwide and people are becoming increasingly aware of it. Similarly, companies, governments, charities, and the software industry are recognizing its importance (Calero et al. 2022). For example, the United Nations is emphasizing a decrease in energy consumption and emissions. Simultaneously, there is an ongoing change in the operative landscape, which changes demands on sustainable services, products, and innovations (Ukko et al. 2018). From a business perspective, sustainability has grown to be a significant consideration. Should a firm fail to operate through sustainable development, making it one of its highest priorities, it could suffer from public criticism and lose market acceptability. The concept of strategic sustainability is highly relevant as most consumers say they are willing to pay more for a green product (Calero et al. 2021).

Sustainability thinking has kickstarted initiatives to reduce power usage and carbon footprints (Calero et al. 2021) and is aligned with the Paris Agreement, which strives to minimize the global average temperature below 2°C. To achieve targets of this agreement, countries are applying voluntary reductions in emissions on a local, national level (Wang et al. 2023). A concern for the globe to sustain human actions is commonly addressed by firms of the S&P 500. Studies show that up to 78% of these companies are implementing sustainability initiatives and report environmental and social performance metrics in so called sustainability reports (Urdan & Luoma 2020).

The term sustainability can be complex to define and is often used interchangeably with corporate social responsibility, or CSR (Urdan & Luoma 2020). The concept of sustainable development was initially defined in the Brundtland Report in 1987 from the United Nations (Johnston et al. 2007). The definition is explained as "*development that meets the needs of the present without compromising the ability of future generations to meet their own needs*" (Ukko et al. 2018). It is key to understand that the term sustainability can have various understandings, and it is commonly framed differently by researchers, consultants, and executives (Urdan & Luoma 2020). According to Johnston et al. (2007), it is therefore important to reclaim sustainability as a term that is defined unambiguously.

It has been estimated that there are over three hundred definitions of sustainability and sustainable development that are linked to other domains such as environmental and management disciplines (Johnston et al. 2007). Over the past four decades, there has been a significant evolution of sustainability in business which has impacted the way the term is interpreted. Most often the term sustainability has been focusing on environmental themes, including specifical programs and strategical approaches (Urdan & Luoma 2020). Additionally, it has been linked to economic and technological activities that adequately should be considered in its definition. Nevertheless, sustainable development involves subsets that describe parts of the same paradigm through universal principles and conditions (Johnston et al. 2007).

The dictionary definition of sustainability indicates that an action or activity should be capable of continuing indefinitely. The definition, however, is challenging as damaging environmental activities can be sustained to a time frame of a human life span, which can seem indefinite (Johnston et al. 2007). Calero et al. (2021) defines sustainability as an ability to operate at a stable level without overusing natural resources or causing ecological harm. In a sustainable environment, sustainability should consider future generations of humans without risking their existence (Calero et al. 2021).

2.2.1 Software sustainability

Generally, sustainability plays a great importance in today's world and software can play a vital role in achieving sustainability (Calero et al. 2022). To reach sustainability objectives, such as the 17 goals specified by the United Nations in 2015 aiming to ensure wealth while protecting our planet (Ukko et al. 2018), software has an important role (Calero et al. 2022). The role of sustainability in software engineering is to ensure that it continuously, over time, fulfils objectives despite its evolution, modifications and updates (Becker et al. 2015). In other words, firms should guarantee that software used is not only available today but is also supported and improved in the future. To aid reaching sustainability goals, firms should examine software through operational efficiency, reducing costs, accelerating scientific progress, and by ensuring a desirable reputation. To simplify, sustainable software is engineering the art of designing and developing software products that enable continuously documenting, assessing, and optimizing its features (Calero et al. 2021).

Although software often is referred to as a key ingredient to reach sustainability objectives, one should understand that technology can have a negative impact on the environment (Calero et al. 2021). In other words, software's role can be both as a part of the solution and the problem. Software sustainability focuses on producing software that can be used for a longer period while using only limited resources (Calero et al. 2022). Although software can help companies tackle environmental challenges by allowing for example videoconferencing, optimization of working processes, and reducing raw materials (Calero et al. 2021) it is often responsible for degradation of the environment (Lago et al. 2015).

Software can be analyzed in either how sustainable the development of code is, agnostic of use case, or how a particular software supports achieving sustainability goals. When analyzing technological aspects of software, also called sustainability in software, the focus is ensuring that it is developed in a sustainable manner. When examining how software functions a tool to fulfil sustainability objectives within any context, it is referred to as sustainability by software (Calero et al. 2022). These two aspects of technology imply that organizations are facing two issues. Firstly, they must ensure that the process of developing software is sustainable. Secondly, firms should verify that software products are contributed to a society that is sustainable (Calero et al. 2021).

Software can be seen as a separate aspect of sustainability that interacts with other dimensions of sustainability. For example, a human dimension can be analyzed through how sociological and psychological aspects impact software development (Lorey et al. 2022). In software, human sustainability includes development of competences and human skills that support operations of sustainable software products. Similarly, it focuses on growing an organization that promotes sustainable development and practices. Additionally, it may focus on human factors such as communication, leadership, team environment, and acknowledgement of efforts (Calero et al. 2021).

Practitioners in software classify human factors critical in achieving sustainability (Bertoa et al. 2021). Despite this, a lack of knowledge in the software community leads to a situation where sustainability techniques are seldom applied (Calero et al. 2021).

2.2.2 Measuring software sustainability

To recognize sustainability of software, one should examine a wide range of sources such as software products, systems, web applications, and data centers (Calero et al. 2021). Similarly, it is possible to compare carbon emission equivalent (CO_2e) in regular internet habits as presented in Table 1 (Graham 2020). To make adequate estimations of software sustainability, it is key to have tools and processes to measure energy efficiency (Calero et al. 2021). Most energy efficiency is to be gained in data centers (Masanet et al. 2020), especially through optimization of software. Hardware has traditionally been improved constantly, but a focus on the software development lifecycle, tools, and principles has seldom been an objective in energy efficiency optimization (Calero et al. 2021).

In general, the behavior of software has a vital role on whether energy-saving features of platforms are effective or not. Similarly, it is important to understand that energy consumption occur both directly and indirectly in relation to a software product. Direct impacts are related to energy consumption and usage of resources when software is produced or is utilized by a user. Indirect impacts are effects that together with software usage and long-term systemic effects influence energy consumption (Calero et al. 2021). Measuring sustainability is complex, as one should be aware of energy consumption and efficiency when a software is running (Liu et al. 2021). This, combined with the fact that there are no formalized methodologies or tools that would guide measurements of energy consumption, makes improving software sustainability complicated (Calero et al. 2021).

Table 1. Examples of internet habits measured in CO_2 equivalent listed by Graham (2020).

Source	**Carbon emission equivalent, g** (CO_2e)
Spam email	0.3
Regular email	4
Email with photo attachment	50
SMS text message	0.014
Publish content in X (prev. Twitter "tweet")	0.2
Google search	0.2

2.3 Web development

In the digital economy, companies are under pressure to develop high-quality software at "internet speed", delivering new systems faster and offering additional value to customers (Baskerville et al. 2003). As the internet evolves rapidly, the development of websites also proceeds quickly. Crafting web software has thus become increasingly convenient as new development platforms and frameworks are introduced (Dubey et al. 2023).

Software development in the internet era is defined by quick changes in requirements and unpredictable product complexity. To manage such challenges, software development approaches must balance flexibility with disciplined principles. The ability to adapt quickly, while maintaining structure is key in this fast-paced environment. (Baskerville et al. 2003)

In today's business landscape, where product release cycles are becoming shorter, companies must deploy new features swiftly to avoid losing customers to competitors (Baskerville et al. 2003). To increase speed in development, various backend frameworks have been created that not only make web design quicker, but also make websites scalable, prepared for high traffic and server load (Dubey et al. 2023).

As the name suggests, web development means creating web applications, and the most difficult part of this process is the initial setup. It is estimated that 60 percent of the hard work goes into a project even before you have your first response on a web page. In practice, from an end-user perspective, every time that you want to access a web page, you open your browser and type in the address that you want to visit (Nicoara 2023). To display a web page, the frontend of an application interacts with the backend by sending a request. The requests are handled through the HTTP protocol and are directed to a certain server port. As the port receives a request, it processes it and provides a response to the client. Based on this, the client renders the response and visualizes the user interface based on the received information (Dubey et al. 2023).

Once the request has reached the server, it is usually handled by a backend service, which is built to interact with the user and is typically written in languages like PHP, Java, or Python. Once the backend service has finished processing your request, it will pack everything up, usually in an HTML response, and serve it back to your browser. This response includes the requested data and the necessary display resources, such as styles, images, and scripts that your browser needs to run the application. (Nicoara 2023)

The frontend is the entire bundle of processes that happen in your browser once the data has been provided by the server. The first layer being served is HyperText Markup Language (HTML), which contains the unformatted data inside your web page (Nicoara 2023), along with all the links to the scripts required for proper rendering (MDN Web Docs 2024).

Next, the browser will execute the CSS (Cascading Style Sheets) layer (Nicoara 2023), which dictates how the web page should look (Libby 2014). Finally, the logic layer is responsible for managing interactions and data retrieval from the server without refreshing the page. The most important part of the JavaScript layer is that it can dynamically generate both HTML and CSS adapting the content and style of the page on the fly (Nicoara 2023).
The backend is everything that happens within the server. Once your browser reaches out to the server, it will start to compile the data that you want to send back to the browser. This mostly means fetching HTML and JavaScript pages and serving them, or providing plain data that it has taken from the database and processed accordingly (Nicoara 2023).

Classic websites and single-page applications (SPAs) represent different approaches to handling user interactions and data loading. The disadvantage of classic websites is that they require users to navigate away from the current page to load new data (Nicoara 2023), whereas SPAs allow content to be fetched without leaving the page (Lopez 2021).
Because the backend is entirely within the server, it holds all the information needed to run the website, processes it, and displays it to the user (Dubey et al. 2023). It also remembers

your data, even if you log in from a different machine (Nicoara 2023). The frontend, on the other hand, is the interface that brings the application to life and displays content, making it the most visible and frequently updated part of the application (Dubey et al. 2023).

2.3.1 Website optimization

As technologies evolve, it is imminent that the size of websites increases. Complex systems often include a demanding use of visuals, which means that carbon emissions caused by websites globally are increasing. As a majority of the world's population has access to the internet, it increases the total carbon emissions produced as bytes of data is transferred on the internet. (Teke 2023)

Web-based projects require precise timing, which means that it is critical for web browsers to preload all the necessary assets, such as media files, before executing the HTML document. This process can often delay user interaction, so optimizing download times should be a key focus. To address this, browser vendors are working on new standards that reduce unnecessary downloads, improve file formats, and cache assets to enhance performance. (Garaizar & Reips 2019)

The preferred method for preloading web assets is currently using the *rel="preload"* property in the link element within the header of the HTML document (MDN Web Docs 2024). This directive to the web browser forces it to fetch necessary assets as early as possible to ensure a smooth user experience. Developers can also preload assets like images directly through JavaScript, which provides flexibility in managing the resources required for rendering a webpage (Garaizar & Reips 2019).

Rendering a webpage involves multiple steps. First, web browsers interpret the JavaScript and CSS code written by developers to implement visual changes. The browser then determines which CSS rules apply to each element in the style phase, and calculates the layout, defining the space each element will occupy on the screen. This leads to the paint phase, where the browser draws the actual pixels for all visual elements such as text, images, and borders. Modern browsers further enhance performance by rendering multiple overlapping layers independently, which must be handled in the correct order to properly display the webpage. (Hikolakita 2024)

Any modification made during the rendering process forces recalculation of previous phases in the pipeline, making optimization critical. For instance, GPU-accelerated rendering allows certain CSS animations to be managed asynchronously, boosting performance (Garaizar & Reips 2019). Web developers can also trace the timing of JavaScript tasks through the developer toolkits in modern browsers, which helps reduce inefficiencies in the execution process (Mozilla Developer Network 2024).

Optimizing the style phase can be done by reducing complexity of stylesheets, such as simplifying selectors and minimizing the hierarchy of elements affected by a style change. Tools like "unused CSS" can help streamline stylesheets and improve performance. Regarding the layout phase, it is recommended to avoid making layout changes inside loops, a problem known as "layout thrashing," as this can result in multiple unnecessary recalculations. Using animations on elements with fixed or absolute positions can also help reduce layout recalculations. (Garaizar & Reips 2019)

The paint phase is typically the most resource-intensive part of the rendering process (FreeCodeCamp 2024). To minimize resources used, developers should avoid repainting areas unnecessarily by utilizing techniques like creating layers, adjusting the opacity of elements, or hiding elements not currently in use. By following these best practices, web developers can significantly improve performance and timing accuracy of their web applications (Garaizar & Reips 2019).

Modern web browsers are equipped with advanced technologies that allow for greater precision and control over content loading and rendering (Mozilla Developer Network 2024). However, they can still encounter performance issues if the complexity of the web application is not optimized effectively (Garaizar & Reips 2019).

2.3.2 Single-page applications and performance

For a long time, a client-to-server architecture has been one of the most commonly used patterns in large-scale software development. This architecture is frequent in systems running on a single machine. Its core advantage is the clear separation of concerns: the server manages complex business logic, persistent storage, and data access from third-party services, while the client focuses solely on presenting data to the end user. Over recent years, however, the frontend layer has changed significantly, moving from server-side rendering to client-side rendering using JavaScript, or more enhanced versions known as single-page applications (SPAs). (Lopez 2021)

SPAs are a relatively new concept where the entire frontend application resides on the client side, usually within the browser (Lopez 2021). The server provides only the initial HTML page, along with a set of JavaScript libraries, which the browser uses to assemble the view. A single-page application follows this pattern for web clients, running entirely in the browser once the JavaScript app is launched from the initial web page. All visual changes and page interactions are based on user actions and data received from remote APIs (Sencha 2024).

SPAs typically communicate with the backend via RESTful services (Nicoara 2023). This interaction allows the client-side to dynamically update the page without requiring a full reload (Sencha 2024). After the initial request to the server, the browser receives an empty page with the basic skeleton of the web application. The rest of the content is fetched using AJAX calls, which load necessary data. Once the data returns from the server, the HTML document rendered on the page is updated dynamically (Nicoara 2023).

SPAs are called single-paged because the server only delivers the minimal HTML required to initialize the JavaScript application. From there, all page rendering and navigation is managed by the client using JavaScript, which utilizes the History API to update content and URLs dynamically. One of the key advantages of this approach is that clients can perform tasks in the background during page transitions, eliminating the need to re-download and re-render the entire page when only the main content needs updating. (Konshin 2018)

A challenge with single-page applications is the need to initially download a large number of assets (Sencha 2024). These include JavaScript files for the app itself, CSS files for styles, images, and other media. Large-scale JavaScript apps must be modularized, leading to numerous small files, often separated by component or page. Similarly, CSS files are typically organized by components or other criteria (Konshin 2018).

The architecture of SPAs also results in significant API traffic. Any user action requiring persistence or any need to pull data from storage, such as a database, triggers an API call. The reliance on these API calls is one of the most critical performance issues of SPAs and their large initial load times. Research has shown that there is a direct link between load times and key metrics like page views and conversions. In fact, many users abandon a page if it takes longer than 2-3 seconds to load. (Konshin 2018)

Additionally, SPAs pose challenges for search engine optimization (SEO) (Sencha 2024). Search engines favor pages that load faster, and while crawlers have recently improved their ability to parse SPAs, this has historically been a challenge. Server-side rendering (SSR) helps address this by allowing the server to perform resource requests more efficiently (Konshin 2018), with the codebase pre-loaded and pre-parsed (Sencha 2024).

2.4 Programming language performance

Software language engineering provides a variety of techniques and tools to design, implement, and evolve software languages. These techniques aim to enhance programmer productivity by incorporating advanced features into language design, such as robust modular and type systems. Additionally, software language engineering seeks to ensure that software executes efficiently by developing optimizations in compilers. However, beyond improving performance, energy consumption is becoming a growing concern in the software development landscape. This issue is now important not only to computer manufacturers, but also to software engineers, programmers, and individual users. (Pereira et al. 2017)

Originally, concerns about energy efficiency focused on hardware, but this has quickly extended to software developers. This shift has led to an increasing amount of research focused on analyzing and optimizing the energy consumption of software systems (Pereira et al. 2017). The primary motivation for such work is to understand how various programming languages differ in terms of energy efficiency (Gordillo et al. 2024).

One common misconception is that energy consumption in software correlates directly with execution time. However, energy efficiency behaves differently from execution time. The study conducted by Pereira et al. (2017) highlights significant variations between programming languages in terms of energy consumption. Languages that exhibit similar energy usage may still have very different execution times (Pereira et al. 2017), demonstrating that average power consumption is not constant across different programming languages (Gordillo et al. 2024).

For instance, when considering memory usage, the top five programming languages requiring the least amount of memory on average were compiled languages: Pascal (66MB), Go (69MB), C (77MB), Fortran (82MB), and C++ (88MB). Conversely, the five languages requiring the most memory were primarily interpreted languages: JRuby (1309MB), Dart (570MB), Erlang (475MB), Lua (444MB), and Perl (437MB), with Erlang being the only exception as a non-interpreted language. (Pereira et al. 2017)

By measuring execution time, energy usage, and peak memory usage, researchers were able to establish relationships between these factors. This helped to reveal that faster execution does not always equate to greater energy efficiency. The findings suggest that developers need to consider more than just execution speed when aiming for efficiency in software. (Pereira et al. 2017)

Finally, recognizing that developers often need to balance multiple efficiency characteristics, Pereira et al. (2017) calculated the best and worst programming languages by combining the factors of energy consumption, execution time, and peak memory usage. These combined metrics offer valuable insights for developers who are constrained by limited resources and need to optimize across multiple dimensions (Pereira et al. 2017).

2.5 Green hosting

Web hosting centers, also called data centers, offer quick server availability at low cost and are suitable for a variety of digital services (Deng et al. 2012). It is estimated that there are approximately 2 billion public websites available globally (Ridenour 2022). Their quick expansion, however, leading to growing carbon emissions, has become a critical concern. The impact on climate change is evident and challenges perceptions of web hosts among the public. Thus, forerunners in web hosting continuously aim to lower their carbon footprints even though it may have negative impacts on revenue profits (Deng et al. 2012).

Controversially, it is important for service providers to ensure an adequate quality of service (QoS) with the result that high-load websites must be deployed on high-end servers. High-end servers consume substantial amounts of energy and are costly as they are often underutilized (Everman & Zong 2018). The overall energy consumption of servers has led to significant carbon emissions, and it is estimated that the total carbon footprint exceeds the footprint of the entire Netherlands. This impact can result in penalties for web hosts due to regulations (Deng et al. 2012). A key challenge in web hosting is that to satisfy availability and response time of web services, they must run on powerful and energy-intensive servers. Simultaneously, these servers continuously consume energy and operate daily, although it is estimated that server utilization is as low as 5 to 15% (Everman & Zong 2018).

For hosting providers to minimize their carbon footprint, there are two approaches. Firstly, the services consuming energy can be designed and optimized to use less energy. This does not only impact carbon footprints but also diminishes the cost of web hosting. Secondly, web hosts can decide to use energy from clean sources in comparison to using electricity powered by coal or other dirty energy source. Despite environmental benefits of using clean "green" energy, which potentially can eliminate carbon footprints of servers entirely, it inevitably will increase energy cost. This leads to a situation, where green hosts may have to implement an increased amount of hosting servers to be able to reach an adequate profit from their investments. (Deng et al. 2012)

To offset carbon footprints, hosting providers can replace dirty energy for example by investing in on-site solar panels, by utilizing power from local wind farms or simply by purchasing fully renewable energy on the energy markets. Often web hosts set an objective of carbon offsets to dirty energy, a so-called offset ratio, which is analyzed through profit over a certain period. Similarly, web hosts commonly focus efforts on green energy as much a possible within a set budget. (Deng et al. 2012)

According to Deng et al. (2012), the green web hosting boom will result in competition through quick, affordable, and green products in flexible cloud-based services. As of the internet boom, the number of websites and hosting servers has increased significantly. The web host rivalry is guided by a demand for high performance, reliability, (Everman & Zong 2018) and clean energy at a competitive and low, price (Deng et al. 2012). Despite a flourishing market, it is vital to acknowledge that due to the nature of website traffic, most web servers are still underutilized leading to increased costs and energy waste (Everman & Zong 2018).

2.6 Web typeface performance and optimization

Despite the wide compatibility of web fonts, each operating system has a defined list of preinstalled fonts. Thus, there are only a few fonts that are fully reliable (Frain 2020) and work on a wide range of devices (Richardson & Thies 2013). In comparison to typography used in print publishing, web publishing means website visitors can only display fonts that are available on their device. This leads to situations where users see websites differently compared to the publisher (Wellens 2015). Using a system font has its benefits, such as little network overhead, and no additional fonts are loaded and replacing text on the fly. On the contrary, using a system font means one never fully knows how text is being rendered on an individual user's device (Frain 2020). In other words, it may result in a page that does not look the way it was designed (Richardson & Thies 2013).

To counter issues with font delivery, techniques and formats have been developed. To ensure specific fonts loads correctly, the *@font-face* technique can be used. As of 2020, the only font format that is required is WOFF or WOFF2, which has efficient properties to compress font information (Frain 2020). To ensure text is correctly displayed, multiple fonts can be selected should the first choice be missing on the device (Richardson & Thies 2013).

When utilizing web fonts, the client browser can wait up to three seconds before rendering the page. If the web fonts have not loaded within this timeframe, a predefined fallback is used to ensure text is displayed before the web font is available. Preferably, the web font loads swiftly without disturbing rendering, but often a fallback is necessary (Google 2024c). This can lead to a phenomenon called Flash of Unstyled Text, or FOUT, which from a technical perspective means that a text is redrawn from a system font to the intended font (Frain 2020). In practice, this means the fallback forces the text to flash before showing the web font (Google 2024c). According to Google (2024c), a FOUT effect is preferred as the key objective of a web page generally is to provide content. Nevertheless, it is key to ensure that loading of the web font is optimized, to avoid a greater risk of users abandoning the website (Google 2024c).

Typefaces consist of various glyphs and for example Cyrillic and Greek character sets are recognizably different. Thus, it is possible to achieve a smaller file size by selecting a set that exclude certain glyphs of a language that is not to be used. Similarly, one can reduce file size of a font by excluding variable versions (Frain 2020). Variable fonts differ from "normal" fonts as they contain information, glyphs, and every variation (such as Italic, Thin, Black and Medium) compressed into one file (Monotype 2024).

To impact how a font is loaded on a website, the World Wide Web Consortium, or W3C, (W3C 2024), has introduced a CSS command called *font-display*, which allows the website to quickly show the content of a page. It forces the web client to render the page immediately with a fallback font and draws the page again only when other fonts have loaded (Google 2024c). Similarly, it is possible to place a piece of code in the *<head>* section of a HTML document, which forces the web font to load early in the rendering path without having to wait for CSS stylesheets to be loaded (Frain 2020).

Generally, there are two approaches to handle loading of fonts on a website. One option is to wait for fonts to entirely load before showing content. As soon as the font is ready, it paints text on the web page. This approach is supported by Safari and is often referred to as FOIT, or "Flash of Invisible Text". As an alternative, one can render the text through a system font, which means it will initially be visible, but then replaced by the correct font when it is available in the web client. (Frain 2020)

2.7 Content Delivery Networks (CDN)

Internet has become a way for individuals to communicate with the outer world, thus the demand for services that allow video calling, online meetings, and streaming has risen extensively (Kumar et al. 2022). Simultaneously, a growing number of internet users are requesting for low latency internet. This has impacted the development of so-called Content Delivery Networks. Content Delivery Networks or CDN’s, that are becoming increasingly popular (Zolfaghari et al. 2020), allow serving web content at scale (Contreras et al. 2023) by placing responding servers in a location as close as possible to users. This results in a reduction of traffic and requests for single content pieces. Without an implemented CDN system, individual requests for content items are serving different versions of the item (Zolfaghari et al. 2020).

A key benefit of using a CDN is making transportation paths shorter between the original content and the browser, as well as preventing any delays in transmission (Contreras et al. 2023). In practice, a content delivery network utilizes a hierarchy of surrogate and cache servers with the last level of operations placed at the end of the user network (Zolfaghari et al. 2020). In a functioning CDN system, the commonly requested content of each local network is cached from a surrogate server close to the user. CDN related servers, commonly developed and maintained by external parties, are cloned by an interconnection of servers that are distributed around the web. As a user requests content through their web browser, the request is handled by the CDN to fetch data from the closest cache server. If the content is unavailable in the cached storage, a new attempt is made in other parts of the hierarchy of servers (Zolfaghari et al. 2020).

Content delivery networks, commonly used to provide content on streaming platforms such as Netflix, YouTube, and Amazon Prime, are built to handle multiple requests from many web clients simultaneously. Service providers of CDN’s are deploying servers on a global scale, aiming to reduce latency issues by transferring contents nearer to end-users (Kumar et al. 2022). This content is stored on local edge nodes that decrease access time and improve quality of a service. Similarly, a CDN can reduce traffic on local networks and the wide internet (Zolfaghari et al. 2020).

Research of CDN’s has existed for a long time with an aim to find collaboration opportunities between service providers. The benefit with an optimized usage of CDN’s is related to a reduced traffic load of ISP networks (Contreras et al. 2023). Although CDN’s have proven benefits, simply increasing the number of servers may not be a viable solution as the establishment costs can become unfeasible. Thus, utilizing techniques from peer-to-peer (P2P) networks can result in gains in terms of storage requirements and providing content to users through edge servers near web clients (Kumar et al. 2022).

2.8 Caching

Today, consumers have high expectations on performance of web services (Gasston 2013) and move to competitors quickly should they be unsatisfied (Davison 2001). Users expect services to work offline and to save data when internet connection is lost (Gasston 2013). This puts an emphasis on solving how to fetch data from file systems and databases (Parker et al. 2010) and creates larger, expensive, web connections that should work seamlessly (Davison 2001). Similarly, it puts technical requirements on developing websites that are visited simultaneously by many users. This can to an extent be solved by adding more hardware, but it also increases costs and makes managing information across various systems harder. Simultaneously, it is a challenge to provide a smooth user experience (Parker et al. 2010).

To meet user anticipations, caching techniques have been introduced to store web files in a shared memory location (Parker et al. 2010). Caching, which has become a critical part of web development infrastructures (Davison 2001), is a technology aiming to deal with high traffic (Parker et al. 2010), bandwidth usage, connection latency, and web server loads (Davison 2001). To simplify, cache is an interface showing files that should be requested and downloaded by a user’s web browser, stored in a memory (Gasston 2013) without having to access a server’s file system (Parker et al. 2010). Ideally, this operation should work even without an internet connection (Gasston 2013).

From a technical perspective, web clients read a manifest file when a page is loaded and determines whether certain files have been loaded previously. Should a file exist in the manifest (cache), it retrieves a copy of this file. The web browser stores this information and lists all entries in the cache to be accessed the next time a page is loaded. When this technique is implemented, web browsers load cached files first, followed by any other required files to render a web page (Gasston 2013). Similarly, a cluster of servers can access a shared memory space without instability under server load, or data latency issues. Often popular files stay permanently in memory as others are put temporarily in and out of memory as they are accessed (Parker et al. 2010).

Caching technologies are also used in computer architectures to improve performance. As a computer's central processing unit (CPU) can run at high speed in comparison to system memory, CPU designers develop certain levels of cache that can operate at an elevated speed. In practice, a CPU can obtain information from the memory cache and fetch objects directly to avoid performance cost. In a website environment, servers and clients transfer data, where the web client, often a web browser, attempts to request data over the internet. The web server then responds with requested information through the hypertext transfer protocol (HTTP). Requests and corresponding responses contain headers and bodies with content. For caching, headers can specify how long a file should be stored in memory (*Expires* and *Cache-Control: max-age*). For files and resources that rarely change, an explicit date of expiration can keep the object in cache for a longer period of time without requiring a new version to be validated by the origin server. (Davison 2001)

Caching benefits all participants of a web ecosystem such as users, network managers, and content producers. This as it can reduce network usage, save costs for content consumers and producers, decrease any delays when displaying content, and lighten server loads. Smaller loads will protect hardware on the origin server and result in a shorter response time for resources that are noncached (Davison 2001). Similarly, as presented in Figure 1, cache technology can result in a substantial increase in requests per second when performing concurrent requests (Parker et al. 2010).

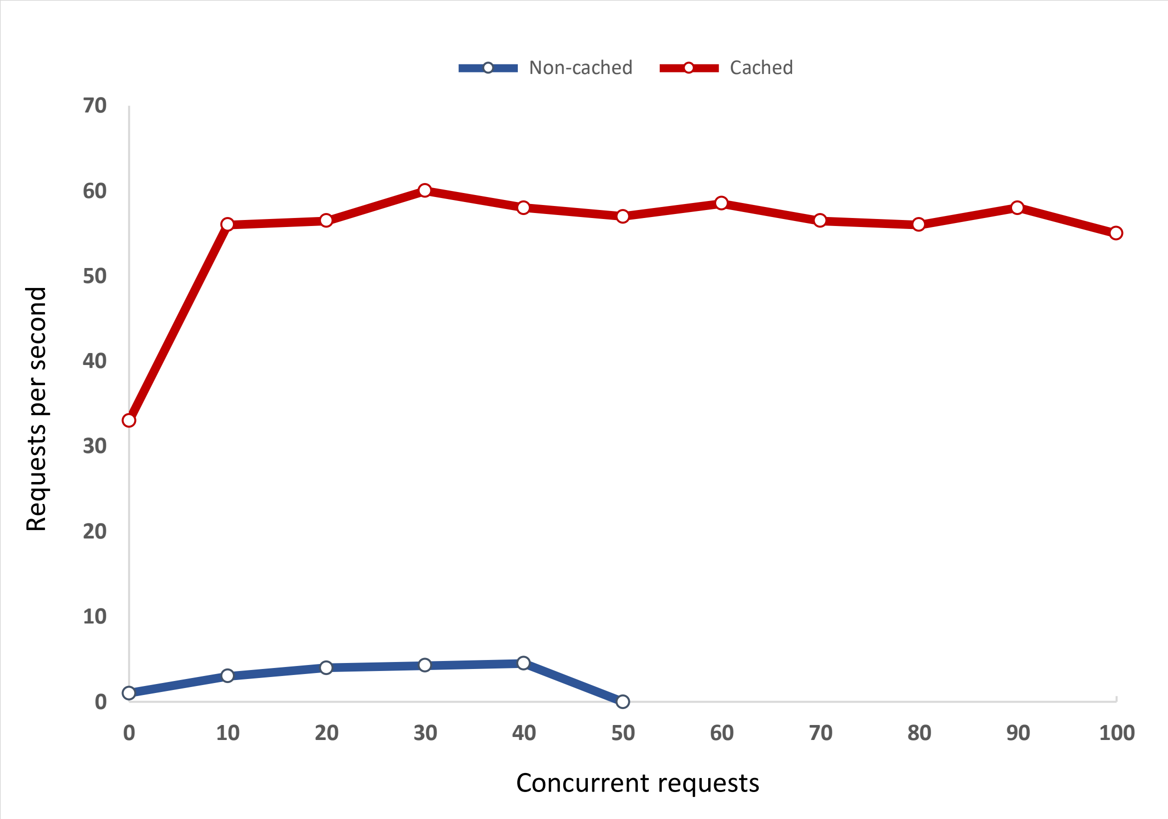

Figure 1. A performance comparison between cached (red line) and non-cached (blue line) server operations (Parker et al. 2010).

2.9 Image formats, performance and compression

The emergence of the internet has led to higher demands of high-speed internet and quicker access to online services (Hu et al. 2017). Simultaneously, as cameras are integrated in phones (Zhao et al. 2019), the amount of data is rapidly increasing (Kuruvilla 2012), and storage and bandwidth of data centers are challenged (Zhao et al. 2019). Out of these large amounts of data, images consume a substantial amount (Kuruvilla 2012), and it has been estimated that storage of data centers will increase by up to four times from 663EB to 2.6ZB (Zhao et al. 2019). As of this, images must be compressed adequately and rendered quickly by web clients (Hu et al. 2017). Similarly, research around image compression algorithms on servers of data centers is urgent (Zhao et al. 2019).

As digital images are increasing in various fields, compression has become critical when storing and transmitting images. In practice, image compression is a technique to reduce the number of bits in a digital image (Rajeswari & Rajesh 2010). A key driver for image compression is to reduce costs of data centers. Thus, large companies such as Google, eBay, and Meta are focusing on implementing efficient image formats (Zhao et al. 2019). Although there are numerous alternatives for compression of images (Hu et al. 2017), there are challenges to maintain a high standard in quality. It is important to acknowledge that images include different types of content like graphics, drawings, and photographs. This variety and combinations of images lead to requirements on reducing the expanding rate of image data on the internet (Kuruvilla 2012). Simultaneously, photos taken with mobile devices are edited, compressed, and constantly uploaded to cloud servers, which puts stress on server bandwidth. Hence, it is vital to achieve a required quality with less power consumption through compression and encoding standards (Hu et al. 2017).

In its essence, compression efficiency shows how well the coding algorithm optimizes the visual quality and number of bits necessary to display an image. Often this efficiency is aligned with feedback from users and their perception of image quality (Ginesu et al. 2012). A common challenge in the compression process is to maintain an image's visual quality. This challenge is emphasized when compression is applied in a quality-intensive context, for example where images include object, emotion, or face expression recognition. In consumer services, a loss in quality can lead to user dissatisfaction. Similarly, it may be problematic in a medical context where for example a diagnosis is made through interpretation of a compressed radiology image. Although image compression through traditional methods can have a positive effect on file size (Moradi et al. 2019) and storage (Rajeswari & Rajesh 2010), it is evident that there are side effects that will impact image clarity, clearness, and details (Moradi et al. 2019).

Various image formats have developed optimized coding algorithms to compress images that ease pressure on bandwidth and storage of data datacenters (Zhao et al. 2019). Common image formats, such as PNG, GIF, JPEG, and WebP all have attributes that can be compressed to reduce file size without reducing visual image quality (Hu et al. 2017). To address performance of these image formats, characteristics are summarized in Table 2 based on Ginesu et al. (2012), Hu et al. (2017), Kuruvilla (2012), and Zhao et al. (2019).

Table 2. Comparison of the GIF, PNG, JPEG, and WebP image formats based on Ginesu et al. (2012), Hu et al. (2017), Kuruvilla (2012), and Zhao et al. (2019).

Standard	**Release year**	**Compression algorithm**	**Benefits**	**Weaknesses**
GIF	1987	Lempel–Ziv–Welch (LZW)	Wide support including animations	Compression of colorful images
JPEG	1992	Discrete Cosine Transform (DCT)	Wide support, Standard in digital cameras	Restricted to 24-bit colors
PNG	1996	Huffman coding and LZ77	Wide support, alpha channel	Maintaining a good compression ratio
WebP	2010	Spatial prediction, LZ77, Huffman coding	Small file size, supports multimedia formats	Slow encoding, does not support lossy to lossless compression

The Graphics Interchange Format, or GIF, perhaps most known for its abilities to support animations in a 256-color palette, was developed in 1987 to support lossless data compression. GIF is a bitmap image format that is suitable for pictures with little distribution of colors. The standard aims to reduce file size without impacting the visual quality of the image. Although GIF has limitations in compression of colorful pictures (Hu et al. 2017), it can still be seen a mainstream image format (Zhao et al. 2019).

The JPEG standard, dating back to 1992 (Hu et al. 2017), has become the leading format in image compression as it is efficient, provides early diffusion, has a low complexity, and is free from licensing royalties (Ginesu et al. 2012). Similarly, JPEG is used in most digital cameras in up to 24-bit colors and 65,535 x 65,535 pixels. JPEG utilizes numerous encoding and algorithms to achieve a lossy compression (Hu et al. 2017). JPEG allows encoding photographs at a high quality (Kuruvilla 2012) and adjusting the relation between file size and image quality (Hu et al. 2017). Despite its popularity, one should understand that mixed-content images may result in a poor compression with a single compression technique (Kuruvilla 2012). One should, however, also acknowledge that JPEG has played a key role in improving the compression ratio to up to 20:1 in the last decades (Ginesu et al. 2012).

The Portable Network Graphics, commonly referred to as PNG, is using an encoding that can compress line graphics, diagrams, and rasterized text with a high compress ratio (Kuruvilla 2012). Released in 1996, as an improved version GIF, PNG has become the most used lossless image format on the web and has a wide support by web browsers and graphics programs. PNG utilizes algorithms and coding supporting up to 24-bit RGB or 32-bit RGBA colors to compress images adequately (Hu et al. 2017).

The WebP image format, released by Google in September 2010 (Ginesu et al. 2012), was developed with many of the advantages of PNG and JPEG (Hu et al. 2017) and has proven positive results with its lossy compression algorithm. In comparison to for example JPEG, WebP can reduce the file size up to 34%. In comparison to PNG, it can reduce the file size to 26% without losing image quality (Zhao et al. 2019). The WebP format, supported by most modern web browsers, is acting a promising competitor to current mainstream formats. It allows using a 24-bit color and 8-bit alpha channel and is available in both a lossy and lossless compression. Furthermore, images in WebP allow preprocessing of compression to adjust image quality. WebP is also designed to store audiovisual material and uses various algorithms to support lossless and lossy compression (Hu et al. 2017). Despite its popularity, it is important to recognize that the encoding speed of WebP is approximately 10 times slower than JPEG which creates higher demands on servers in data centers (Zhao et al. 2019). WebP as a format is rather inefficient in the coding phase but has proven benefits in decoding of images in a low resolution. Nevertheless, WebP does not support lossy to lossless compression, indicating that improvements are still necessary (Ginesu et al. 2012).

2.10 Google Core Web Vitals (CWV)

The speed and usability of a website have become critical factors influencing its search engine ranking and overall user experience (Google 2024a). Elements such as page speed, Core Web Vitals, and mobile usability significantly affect how websites perform in search results and determine how efficiently a search engine robot crawls through the site’s content (Edgar 2023). By focusing on optimizing these elements, website owners cannot only improve search engine visibility but also ensure a seamless and engaging user experience for their visitors (Google 2024a). As speed and usability continue to play a more significant role in determining rankings (Egri & Bayrak 2014), improving these areas is increasingly seen as essential for achieving strong search engine performance (Edgar 2023).

One of the most vital aspects of this optimization is Google’s Core Web Vitals, a set of metrics that assess three key aspects of user experience on a website (Edgar 2023). These metrics include First Input Delay (FID), which measures interactivity; Largest Contentful Paint (LCP), which focuses on how quickly the largest visible element of a page loads; and Cumulative Layout Shift (CLS), which evaluates the visual stability of a page by measuring how often content moves unexpectedly as it loads (Google 2024b). Unlike many other metrics that rely on data from automated bots, Core Web Vitals are based on actual user interactions, captured through field data in Google Chrome (Edgar 2023). Alternatively, developers can measure these aspects through simulated load testing, known as lab data, using tools like PageSpeed Insights (Google 2024b).

Given the increasing importance of user experience in search engine rankings, optimizing for both speed and usability is crucial (Egri & Bayrak 2014). A fast-loading website with a smooth user experience is likely to rank higher in search results, and users are more likely to stay engaged with the content (Google 2024a). By focusing on improvements in Core Web Vitals, website developers can enhance search engine performance while also creating a more responsive and user-friendly environment for visitors (Google 2024b). This dual benefit of enhancing ranking and user experience makes these optimizations a necessary part of modern website management (Edgar 2023).

2.11 Reducing website’s load time and improved mobile friendliness

To effectively optimize a website's speed, there are three key objectives that all are aligned with the broader goal of improving Core Web Vitals. First, reducing the total number of files requested is essential. Every file requested by a web browser requires a trip to the server, and even though each request may take only a few milliseconds, these add up quickly when hundreds of files are involved (Edgar 2023). By eliminating unnecessary requests, the overall load time can be significantly reduced (Egri & Bayrak 2014). For example, instead of loading five separate CSS files, combining them into a single file minimizes the number of server requests (Edgar 2023).

Second, shrinking the overall size of the website is crucial for a faster performance. The smaller the files, the faster they load (Edgar 2023). Techniques like compressing and minifying CSS and JavaScript files (Egri & Bayrak 2014) as well as removing unused code, can dramatically reduce file sizes. Additionally, optimizing images to ensure they load at proper dimensions also contributes to reducing the overall size of the website, allowing it to load quicker (Google 2024a).

Finally, loading files as efficiently as possible can make a significant difference. Caching dynamic files is an effective way to reduce the frequency of database requests, improving overall load efficiency. JavaScript files can also be loaded asynchronously, preventing them from blocking the website's load process, or deferred, allowing the browser to execute them only as all other essential elements have loaded. (Google 2024b)

Another important factor in website optimization is mobile friendliness, especially as Google now includes mobile usability as part of its Page Experience ranking factors. Without proper support for mobile users, websites risk losing significant traffic. One of the key elements to address is ensuring that text is easily readable on mobile devices without requiring to zoom. A font size of at least 12 pixels for main text is recommended, though smaller sizes, such as 9 pixels, may still pass mobile usability tests. (Google 2024a)

Additionally, adjusting content is critical so users don't have to scroll horizontally to view content. This is particularly important for creating a smooth user experience, as horizontal scrolling can be frustrating when used across the entire page (Google 2024b). Ensuring that links are placed far enough apart also plays a role in mobile usability. Tap targets, such as buttons, should be at least 48x48 pixels in size, with a minimum of 6–8 pixels between them, to ensure that users can tap the right link (Edgar 2023).

By focusing on these areas, optimizing load time and enhancing mobile usability, websites can significantly improve their performance in search rankings (Google 2024a). Additionally, these optimizations contribute to a smoother, more seamless user experience for visitors (Edgar 2023). Prioritizing key metrics like Core Web Vitals and mobile usability ensures that websites not only rank better but also meet the expectations of modern users (Google 2024b).

2.12 Measuring and monitoring guidelines

When assessing a website's page experience, it is crucial to start by reviewing key speed metrics (Egri & Bayrak 2014), including Core Web Vitals and other critical performance indicators. The first step is to identify the metric with the worst performance and investigate how to optimize it. For instance, metrics like Time to First Byte (TTFB), Fisrst Input Delay (FID), and Interaction to Next Paint (INP) can often reveal underlying issues with server configuration. In contrast, metrics such as Largest Contentful Paint (LCP) or Start Render suggest that there are inefficiencies in how files are being loaded in web browsers. It is essential to revisit these metrics after every larger code deployment and make necessary adjustments to ensure optimal performance (Edgar 2023).

While speed is undeniably important, it is not the only factor influencing search rankings. For example, Google will not prioritize a fast website if it lacks quality content. Similarly, visitors are unlikely to convert purely because a website loads quickly. The content must also be compelling enough to drive conversions, reinforcing that both speed and content substance are equally important. (Edgar 2023)

For mobile optimization, it's important to consider how the mobile version of the website is built. One should consider aspects such as responsivity and dynamic content that it served to users depending on their device (Niranga 2015). If dynamic serving is used, one should ensure that content remains equal across all devices. If the website is responsive, one should check whether any essential parts of the main text are hidden from view, potentially causing content equivalency issues between mobile and desktop experiences (Edgar 2023).

Another factor to review is the use of interstitials, such as pop-ups or modals, on the mobile version of the site. One should consider how these are loaded, for example directly as the page loads, and if it they are blocking any content from being displayed. Another aspect is determining if they are legally required or if they have a promotional intent. For promotional content, one should consider alternative ways to present this information that do not obstruct the user experience on mobile devices. (Edgar 2023)

2.13 Choosing website speed metrics

The range of metrics available for evaluating a website's speed can make the process of reviewing speed reports appear complex and overwhelming (Edgar 2023). Fortunately, this does not have to be the case (Google 2024b). A more effective approach involves contextualizing each metric within the broader framework of the website loading process, from the visitor's initial connection through to their final interaction with the site. By categorizing metrics based on key stages of loading, and reviewing them step by step, the analysis becomes more structured and meaningful as presented in Table 3 by Edgar (2023).

The first stage to evaluate is the initial connection. A critical metric at this stage is Time to First Byte (TTFB) (Google 2024b). If TTFB is slow, this signals inefficiencies in the early stages of the loading process, which can result in delays in all following stages. It is essential to address these foundational issues before proceeding. To diagnose where the problems lie, to separate the TTFB into its components, such as DNS Lookup Time, and to prioritize improvements in these areas before considering further optimization (Edgar 2023).

Once the initial connection is efficient, attention should shift to the metrics that reflect the middle stage of the loading process, specifically those related to displaying the page. This phase, often referred to as the critical rendering path, involves metrics such as DOM Content Loaded, Total Resources, Transfer Size (Edgar 2023), and First Contentful Paint (FCP) (Google 2024b). These indicators provide insight into how quickly content becomes visible and interactive for users. Any delays at this stage significantly impede the user experience and must be addressed prior to addressing optimizations at a final stage (Edgar 2023).

After optimizing the middle stages, the focus should shift to the final phases of loading, which involve completing the website load (Edgar 2023). Relevant metrics at this stage include Time to Interactive (TTI), which measures when users can begin interacting with the site, and Total Blocking Time (TBT), which can provide an explanation for any delays in TTI. Additionally, Speed Index measures how quickly content becomes visible, while Onload or Fully Loaded Time indicate when all resources on the page have been fully rendered (Edgar 2023).

Lastly, it is crucial to evaluate user experience metrics, specifically those included in Google's Core Web Vital. These metrics assess critical aspects of how users interact with the site. Largest Contentful Paint (LCP) measures when the most significant visual element in the first viewport is painted, typically the main content of the page (Google 2024b). Cumulative Layout Shift (CLS) evaluates whether unexpected layout shifts impact user experience, while Interaction to Next Paint (INP) measures the responsiveness of the page to user interactions (Edgar 2023).

Table 3. The major stages of a website load as described by Edgar (2023).

Initial connection	**Displaying the page**	**Completing website load**	**User experience**
Time to First Byte DNS Lookup Time	DCL Time Transfer Size Total Resources First Contentful Paint (FCP)	Time to Interactive (TTI) Total Blocking Time Onload Time Fully loaded Time Speed Index	Largest Contentful Paint (LCP) Cumulative Layout Shift (CLS) Interaction to Next Paint (INP)

In conclusion, it is essential to recognize that each website has unique characteristics, thus, the choice of metrics for regular monitoring should be tailored to specific goals of the site. The metrics selected should align with the business's objectives and provide a meaningful reflection of the website's user experience. Furthermore, only metrics that can be realistically improved should be tracked. This, as there is limited utility in monitoring metrics that cannot be influenced due to technical constraints or those optimizations that may not significantly enhance the website's performance. (Edgar 2023)

Monitoring chosen metrics on a regular basis is key to understanding how website speed evolves over time, particularly in response to optimization efforts and code updates. While some organizations may benefit from investing in multiple speed analysis tools, others may find sufficient value in free tools. Regardless of the tools or metrics chosen (Edgar 2023), the primary objective is to consistently measure at least some aspect of website speed (Google 2024b). This practice provides critical insights into why a website may load slowly and highlights opportunities for improvement (Edgar 2023).

2.14 Sustainable web design

The exponential growth of digital consumption has led to increased energy demands, particularly within the communication technology sector. With the uprise of smartphones, tablets, and the Internet of Everything (IoE), consumer devices now dominate global electronic markets. By 2030, estimates suggest that communication technology could consume up to 51% of global electricity, contributing significantly to greenhouse gas emissions (Andrae & Edler 2015). This rise in energy demand is combined with an increased use of cloud computing and high-speed networks, despite advancements in electricity efficiency for data transmission and storage (IBM 2022). As global internet traffic is expected to continue growing, with predictions of a threefold increase over the next five years, the need for sustainable design practices in web development is becoming more urgent than ever (Cisco 2015).

Designing for sustainability requires a holistic approach that accounts for the environmental impact of every stage in a digital product’s lifecycle. The Sustainable Web Design Model (SWDM) provides a framework for estimating and reporting greenhouse gas (GHG) emissions across various digital activities, such as web browsing, video streaming, and data transmission (Sustainable Web Design 2024). This model highlights the importance of minimizing resource consumption through efficient design practices, including optimizing content for accessibility and usability, reducing page load times, and utilizing lightweight visual assets (IBM 2022).

At the heart of sustainable web design is the reduction of energy use. This can be achieved by simplifying user experiences, lowering network requirements, and selecting eco-friendly hosting environments. By embracing a sustainability-conscious mindset, designers can create digital products that not only serve users effectively but also reduce their carbon footprint. The SWDM encourages designers to consider both operational and embodied emissions, factoring in the energy used during a product's lifecycle, from manufacturing to usage. (SVID 2018)

2.14.1 The Sustainable Web Manifesto

The Sustainable Web Manifesto, presented in Table 4 (Sustainable Web Manifesto 2024), outlines six core principles that guide the creation of sustainable digital products: Clean, Efficient, Open, Honest, Regenerative, and Resilient. These principles form the foundation for ethical and environmentally conscious web design (Greenwood & Frick 2024).

Table 4. The six core principles of the Sustainable Web Manifesto (2024).

Core Principle	**Definition**
Clean	Services should be powered by renewable energy, reducing dependency on fossil fuels.
Efficient	Products must minimize resource and energy consumption, ensuring that the least amount of materials and power are used.
Open	Ensuring transparency and accessibility in digital services is key, allowing for the free exchange of information while giving users control over their data.
Honest	Designs should not exploit or mislead users, promoting ethical interactions between consumers and digital platforms.
Regenerative	Digital products should support sustainable economies that benefit both people and the planet, moving beyond mere resource efficiency to restorative practices.
Resilient	Products must be designed to function effectively in a variety of environments, particularly in times of crisis or resource scarcity.

These principles challenge designers to rethink their role in the broader ecological system, recognizing that digital products have both immediate and long-term impacts on the environment. By adhering to these guidelines, developers can significantly reduce the environmental footprint of their products, fostering a more sustainable digital future. (Sustainable Web Manifesto 2024)

3 Methodology

In the following chapter the concepts of sampling and comparative analysis are outlined. Furthermore, the methodologies used in this study are explained in detail through justification of the sampling method, data collection, tools, algorithms, and frameworks.

3.1 Sampling

Sampling is a key concept in research, referring to the selection of a smaller, manageable subset from a larger group. This subset, or sample, contains the characteristics of the larger population. Essentially, sampling is a process that allows researchers to study a smaller group of individuals to make inferences that are likely to be applicable to the entire population. (Baral 2024)

The sample consists of the individuals who will participate in the research. To understand sampling more clearly, it is essential to distinguish between two core terms: population and sample (Baral 2024). The population refers to the entire group that a researcher aims to draw conclusions about, while the sample is the specific group of individuals from whom data will be collected (Banerjee & Chaudhury 2010).

One important sampling method is non-probability sampling, also referred to as non-random sampling. This is a technique in which samples are deliberately selected rather than chosen randomly (Baral 2024). In non-probability sampling, the researcher selects samples based on their subjective judgment rather than using a random selection process. Unlike probability sampling, non-probability sampling does not give every member of the population an equal chance of participating in the study (Novosel 2023).

Within non-probability sampling, judgmental or purposive sampling is a common approach. This method is not considered scientific, as it carries the risk of being influenced by the researcher’s preconceived opinions, which can skew the results. Despite this limitation, purposive sampling can be used in both qualitative and quantitative research methods.

However, this technique involves a high degree of ambiguity, adding complexity to the research process. (Baral 2024)

Another form of non-probability sampling is convenience sampling, which involves selecting respondents who are “convenient” to access (Novosel 2023). There is no structured pattern for acquiring these respondents; they may be approached in public spaces such as streets, workplaces, or buildings. While this may give the impression of random sampling, as people are often recruited irregularly, true random sampling involves the use of random numbers to select participants from a predefined sampling frame. Random sampling usually results in a statistically balanced selection, whereas convenience sampling introduces a significant level of bias due to the unstructured selection process (ScienceDirect 2024).

3.2 Comparative analysis

Comparative analysis refers to any attempt to identify causal relationships between variables (Ragin 1987). However, it’s important to clarify what exactly constitutes a "cause." In some cases, it refers to the presence or absence of a phenomenon, while in others, it focuses only on direct or occasioning causes that trigger outcomes. Additionally, it is essential to consider the level of abstraction at which these causal forces appear, as this influences the scope and impact of the analysis (Pickvance 2001).

In a strict sense, all analysis can be considered comparative as it involves the comparison of variables to draw conclusions. A more focused definition, however, limits comparative analysis to situations where a societal characteristic (Ragin 1987) is shown to have a measurable influence on the variable or relationship being studied (Pickvance 2001).

In comparative analysis, variables are commonly categorized into four distinct types. Dependent variables are those whose values we aim to explain, such as a specific outcome or effect in a study. Independent variables, on the other hand, are those that are suspected to explain the variation in the dependent variables (Creswell 2014). These independent variables can either be allowed to vary freely or be controlled, with their values held constant to maintain consistency for a particular unit of analysis. Uncontrolled variables are those that impact the dependent variable but remain outside the researcher's control, while controlled variables influence both the dependent and independent variables, potentially complicating the relationships in investigation (Pickvance 2001). This classification of variables is vital for understanding their roles in comparative analysis (Pickvance 2001) and provides a structured approach to analyzing the complex relationships between different factors (Creswell 2014).

3.3 Selected research methodology

This study is based on a dataset collected from a sample of Finnish websites (N = 32). The websites were selected based on the Large Cap segmentation of the leading publicly listed companies in Finland as of Q1 / 2024. The Large Cap includes companies with a market value above 1 billion euros (Nasdaq 2024) and represent various industries such as technology, banking, energy production, real estate, consumer goods, and healthcare.

As the sample was carefully, purposively selected from a list of companies, a non-probability sampling method was used. The researcher's subjective judgement was used in a combination with a convenience sampling method, where the sample of companies was conveniently available publicly on the internet. Despite limitations in purposive sampling (Baral 2024), it is key to acknowledge that these Finnish companies represent a wide variety of industries and have operated globally in five different centuries (Fiskars Oyj Abp 2024; Nokia Oyj 2024; Neste Oyj 2024; Cargotec Oyj 2024).

To examine data, a comparative analysis approach and design science discipline was utilized. A comparative analysis aims to identify relationships and causalities between variables (Ragin 1987). All analysis in this thesis may be classified as comparative, as it includes making conclusions based on comparing variables (Ragin 1987). Throughout the thesis, dependent and independent variables, and controlled and uncontrolled variables, are demonstrated in terms of sustainability in web design. For example, CMS systems used (independent variable) and carbon emissions per page view (dependent variable) are analyzed. Similarly, the utilized APIs and their crawling mechanics (controlled variable) are introduced in addition to describing how one may not impact a changing user behavior directly (uncontrolled variable).

The design science paradigm, on the other hand, strives to cover boundaries of human and organizational capabilities (Hevner et al. 2004) and develop innovative artifacts (Peffers et al. 2007). For this study, the Design Science Research Methodology (DSRM), presented in Table 5, is applied to identify a problem, defining, designing, and developing a solution as well as evaluating and communicating results (Peffers et al. 2007). Firstly, it is important to evaluate the problem domain (Hevner et al. 2004), which in this study is problematics in sustainable web design. Secondly, based on the problem a solution is design and developed as an innovative artifact (Hevner et al. 2004). In practice this means planning and producing a set of software tools, that can audit performance metrics of any web page.

Table 5. The six step Design Science Research Methodology (DSRM) as described by (Peffers et al. 2007).

Research Step	Concerns	Output to Next Step	Entry Point?
1. Identify Problem & Motivate	• Define problem • Show importance	• Inference	• Problem-Centered Initiation
2. Define Objectives of a Solution	• What would a better artifact accomplish	• Theory	• Objective-Centered Initiation
3. Design & Development	• Artifact	• How-to Knowledge	• Design & Development Centered Initiation
4. Demonstration	• Find suitable context • Use artifact to solve problem	• Metrics, Analysis, Knowledge	• Client/Context Initiate
5. Evaluation	• Observe how effective, efficient • Iterate back to design	• Disciplinary knowledge	
6. Communication	• Scholarly and professional publications		

To collect a dataset for the sample, a collection of technologies was utilized. Most data was obtained through Google’s PageSpeed Insights API, which offered a crawling tool to retrieve performance metrics in a simulated environment based on real user data (Google Developers 2024a). To receive a successful response, various variables (url, strategy, categories, and an API key) were given. The program would then return the request and store the information in a text file as visualized in Figure 2.

To obtain information about the technology stack of the sample websites (N = 32), an API endpoint by WhatCMS was used. The API provided data such as CMS, programming language, and web server in a convenient JSON format. To retrieve data from the endpoint, two parameters were given, the URL of each website and a personal API key (WhatCMS 2024) as shown in Figure 3. As the WhatCMS API was used on a restricted, “Free” license plan, the program was configured to send a request only once per 11 seconds, as the program was looping through the sample of website URL’s. In addition to the WhatCMS API, Google Chrome DevTools was used to examine network calls that would indicate what CMS system was used for a website. Similarly, the “React Developer Tools” extension in Google Chrome was used to detect web services built on React, a widely used JavaScript library that allows developing scalable and effective web applications (Bugl 2019).

For accessing the Google PageSpeed Insights API and the technology API by WhatCMS, Bash scripts (.sh) were run to loop through a sample of web addresses and send cURL request with necessary parameters. The retrieved information was then stored as text files (.txt) or structured data (JSON) for further analysis as shown in Figures 2 and 3.

To review hosting services of the sample websites, the public register of The Green Web Foundation was examined. This register, which gives information on green hosting for each site, offered a tool that was used in connection with a Google Sheet document. This document was configured to crawl webpages through the *IMPORTXML* function, a feature that can be used to import data of various types of structured data (Google 2024d). In addition to the *IMPORTXML* function, *XPATH* was used in the data retrieval process. Conditional clauses and conditional formatting made the data easier to comprehend. The data collection process for hosting information is visualized in Figure 4.

For data analysis, mean, median, and standard deviation (SD) algorithms were calculated to show results. These algorithms were used to compare data distribution and to get an overview on variances in the websites’ performance. To visualize results, Python scripts were programmed to utilize the Matplotlib and Pandas libraries. This allowed plotting findings from collected data (.csv) in a graph format with custom styling.

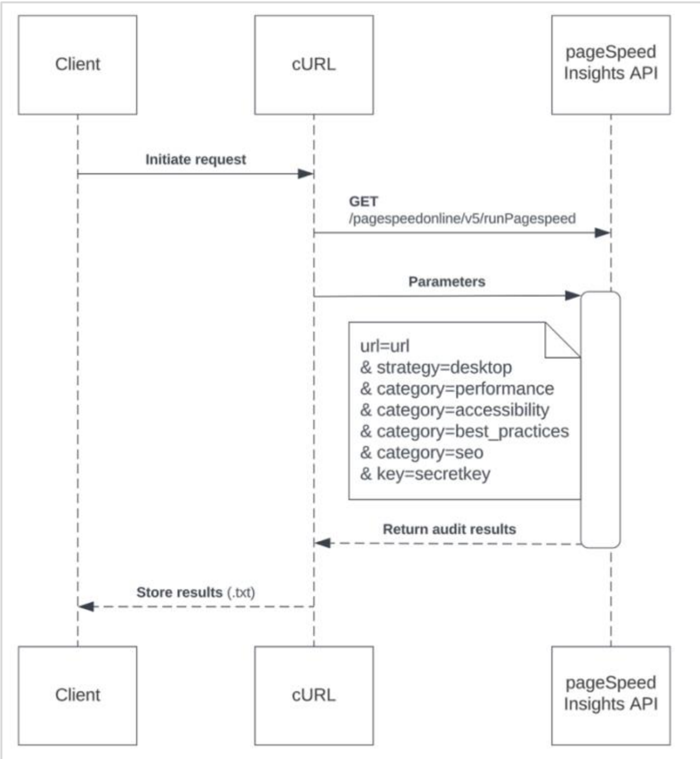

Figure 2. Sending a cURL request to Google's PageSpeed Insights API (Google Developers 2024a) to audit performance, accessibility, best practices and SEO results.

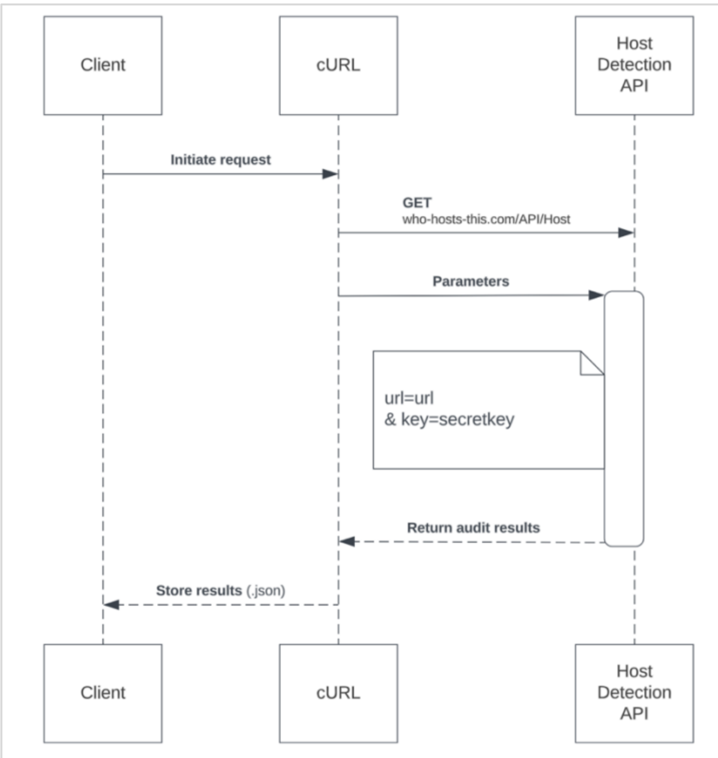

Figure 3. Sending a cURL request to the Host Detection API by who-hosts-this.com to retrieve hosting information.

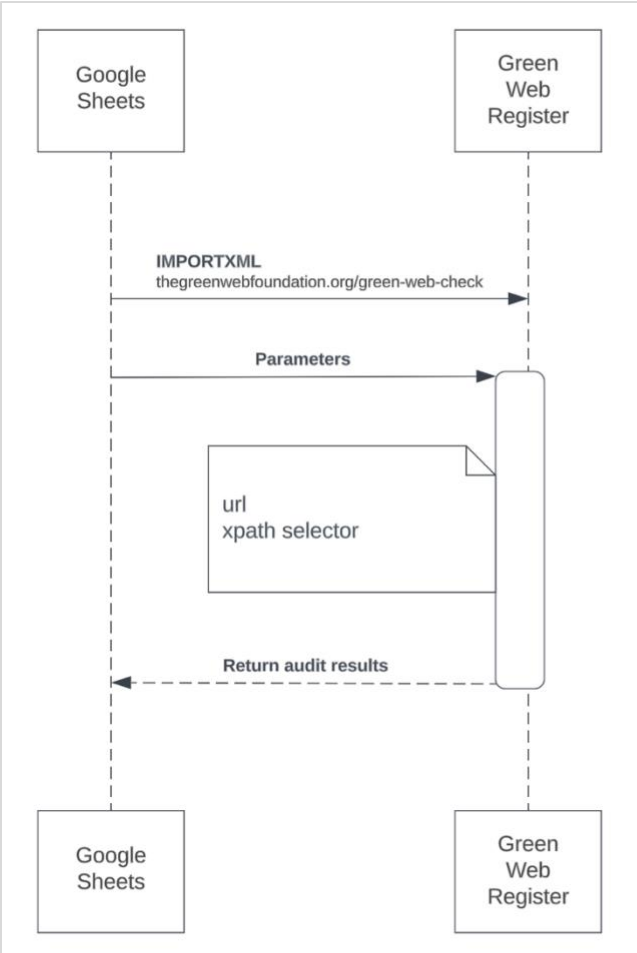

Figure 4. Retrieving hosting data from the Green Web Register (The Green Web Foundation 2024) through the IMPORTXML function in Google Sheets.

4 Results

In the following chapter, results from the study are examined through a selected set of defined sustainability related web design metrics. As the sample was limited to 32 Finnish websites, expert judgement was used to explore collected data and to select a set of six relevant areas of analysis. The assessment focus was to analyze 1) CO_2 emissions in grams, 2) programming language and CMS, 3) green hosting technologies, 4) number of requests, 5) site speed index, as well as the 6) Google Lighthouse score for each website.

4.1 Estimating CO_2 emission intensity in grams

Governments, charities, and enterprises are increasingly aware of sustainability issues (Calero et al. 2022). Similarly, institutions, such as the United Nations, are addressing sustainability challenges and are directing efforts on decreasing overall energy consumption and CO_2 emissions (Ukko et al. 2018). Furthermore, there is a rising demand on sustainable services, products, and innovation (Ukko et al. 2018) and countries are pressured to apply emission reductions on a national level (Wang et. al 2023).

As technologies advance, it is evident that website size increases. This is a result of a demanding use of visuals which impacts carbon emission globally. As the world's population grows, it inevitably increases carbon emissions as all website users produce emissions through every byte that is transferred (Teke 2023). Despite a clear correlation between website usage and emissions, it is key to acknowledge that emission calculation in data transfer is complex due to numerous variables included (Thornbush 2021).

For this study, websites were analyzed through a crawling mechanic developed by Google in the PageSpeed Insights API (Google Developers 2024a). This API was used to obtain the effective website size variable of *mainDocumentTransferSize* (MB). Through the information on downloaded resources by the web client, it was possible to calculate an average electricity intensity of internet data transmission (0.06 kWh/GB) as proposed by Aslan et al. (2018) in a regression analysis. Based on the energy intensity, the values were converted into a CO_2 emission intensity based on Finnish data (0.113 kg CO_2/kWh) of the European Environment Agency (EEA) of the European Union (European Environment Agency 2024). The CO_2 emission intensity results for all analyzed websites (N = 32) are visualized in Figure 7.

As presented in Figure 5, one can calculate how much CO_2 emissions each website load results in grams through an algorithm, where E is the CO_2 emission intensity in grams and G is the GB of a website resource load in a web client. The constant 0.06 kWh/GB refers to electricity intensity of internet data transferred (Aslan et al. 2018), and the constant 0.113 kg CO_2/kWh is based on CO_2 emission intensity for a Finnish market as defined by the European Environment Agency (EEA) (European Environment Agency 2024).

$$\boldsymbol{E = \left(\frac{G}{1000}\right) * 0.06 * 0.113 * 1000}$$

Figure 5. Calculating the CO_2 emission intensity in grams (g CO_2/kWh).

The sample (N = 32) mean in grams is **0.25** and the median lies at **0.22**. The websites of companies with the lowest carbon footprint are Telia Company (0.02g), Revenio Group Oyj (0.10g), Elisa Oyj (0.11g), Fiskars Oyj Abp (0.12g), and Kojamo Oyj (0.13g). The highest calculated CO_2 emissions in grams are found on the websites of TietoEVRY Oyj (0.52g), Huhtamäki Oyj (0.51g), Kemira Oyj (0.46g), Metso Oyj (0.46g) and Nokia Oyj (0.45g).

When calculating standard deviation (SD) for CO_2 emission intensity in grams for the analyzed web pages (N = 32), we get a value of **0.13**. This value helps understand data distribution in relation to the mean (Parrill et al. 2019). The SD, visualized in Figure 6, was obtained through subtracting the mean (**0.25**) from the squared deviation of all observations (x_i^2), summing these values ($\sum$), dividing this with the sample number (32) and by square rooting this result ($\sqrt{}$) (National Center for Biotechnology Information 2024).

$$SD\ \sigma = \sqrt{\frac{\sum (x_i - 0.25)^2}{32}} = 0.13$$

Figure 6. Calculating the standard deviation (SD) for a set of analyzed web pages in terms of CO_2 emission intensity in grams (g CO_2/kWh).

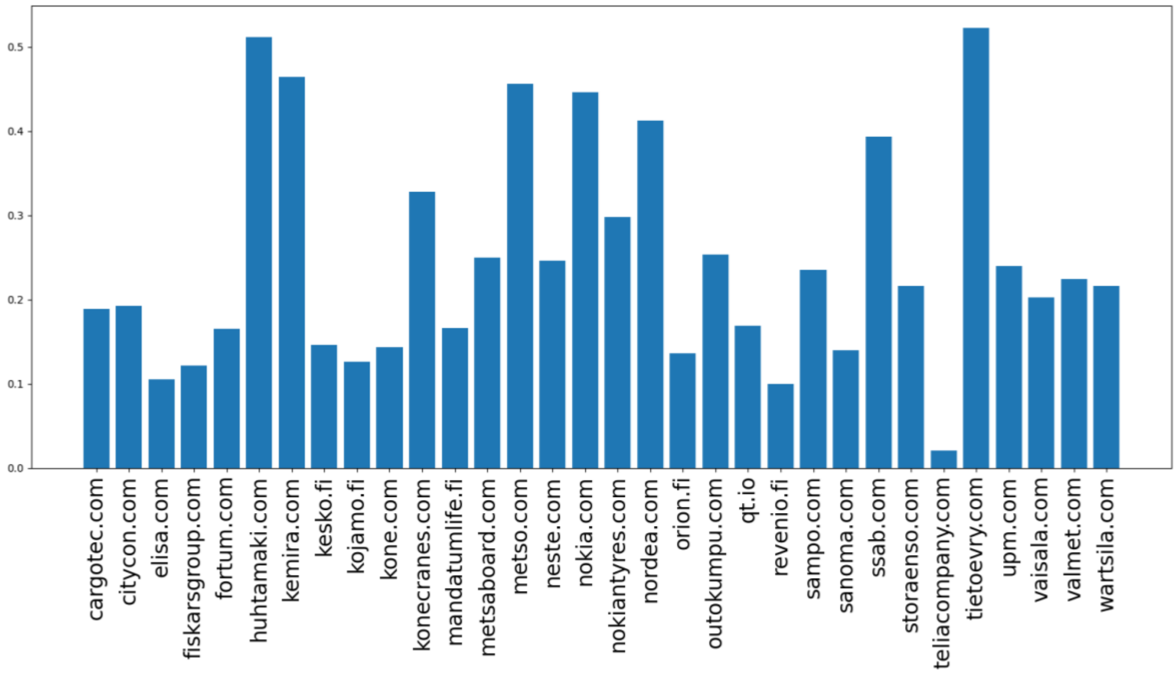

Figure 7. CO_2 emission intensity in grams (y-axis) of analyzed web pages (x-axis, N = 32).

4.2 Programming language and CMS

Content management systems (CMS's) allow editors to manage website content without having to edit HTML (Seadle 2006), and with little or no previous web design or programming experience (Christianson & Cochran 2009). CMS platforms are developed through a range of programming languages (Seadle 2006); thus, it is essential to provide a user-friendly interface to edit content without programming requirements (Christianson & Cochran 2009).

As commercial products emerge and CMS systems advance (Seadle 2006), developers must operate through well-structured environments that allow for efficient programming

(Christianson & Cochran 2009). Despite improved efficiency, the energy consumption of software has become a growing concern (Pereira et al. 2017). Thus, understanding how different programming languages differ in energy efficiency must be researched (Gordillo et al. 2024). When comparing programming languages, various metrics must be compared to obtain a realistic view on energy efficiency. For example, one should not only analyze the execution time of a programmed task, but also investigate relations to energy usage and peaks in memory usage (Pereira et al. 2017).

To collect data on the technology stack of the sample websites (N = 32), an API by WhatCMS was used. Additionally, Google Chrome DevTools and the "React Developer Tools" extension was used to examine network calls of websites.

When looking at CMS platforms used by the sample websites in Figure 8, we can conclude that 11 different systems were implemented. **34%** of the websites are using Optimizely (former Episerver). Optimizely aims to offer organizations tools to design personalized online experiences through a data-driven approach (Episerver 2021). Approximately **19%** of the websites were powered by Drupal, as **9.4%** each were using WordPress, SiteCore or React.js. A minority of web services at **3.23%** each were running in a custom-built CMS, or a formalized system in SDL Tridion, Vue.js, Contentful, HubSpot, or Sitefinity used as their CMS platform.

As visualized in Figure 9, programming languages used for web services of the sample was ASP (**50%**), PHP (**28.13%**), JavaScript (**12.5%**), ASP/JS (**3.13%),** HubL (**3.13%),** and Node.js (**3.13%**). In other words, ASP stands for half of the systems whereas PHP is placed second with just under a third of the websites. ASP, or Active Server Pages, originally developed by Microsoft soon 30 years ago, has been praised for its speed, flexibility, and efficiency. Similarly, applications built on ASP are often easy to understand and accessible to developers of all levels (Mesbah et al. 2001). Nonetheless, PHP, utilized for example on Drupal and WordPress websites, is a general-purpose scripting programming language, that is easy to learn (Hussien Odeh 2019) and gives developers tools to create content dynamically and connect to databases in web applications easily (Engebreth & Sahu 2023).

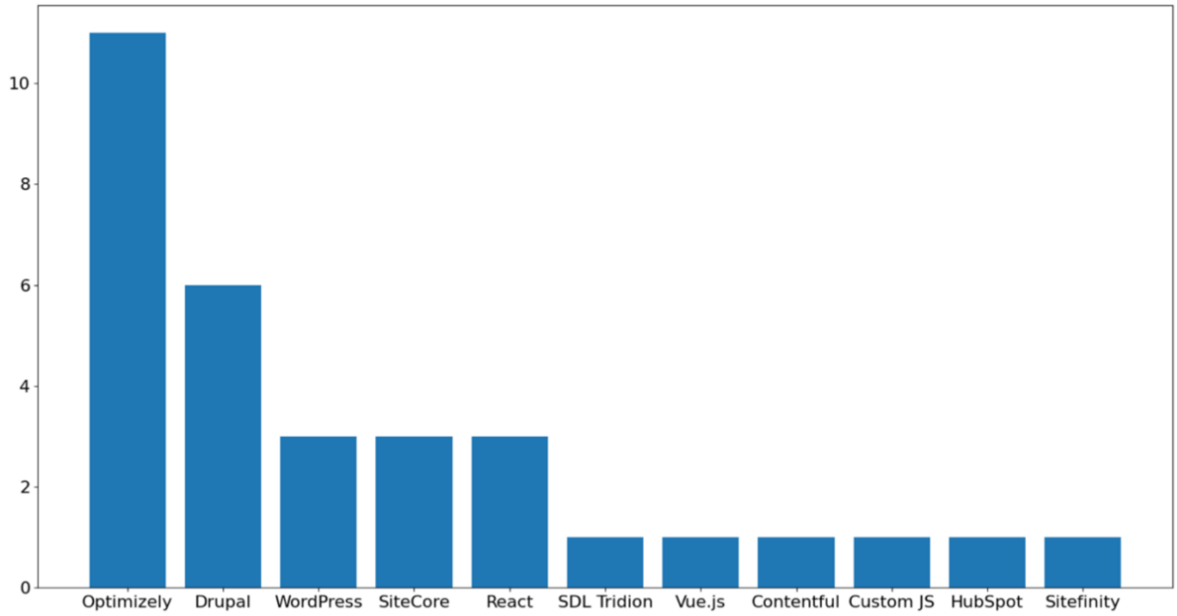

Figure 8. Number of websites published on different content management systems (CMS) in the sample (N = 32).

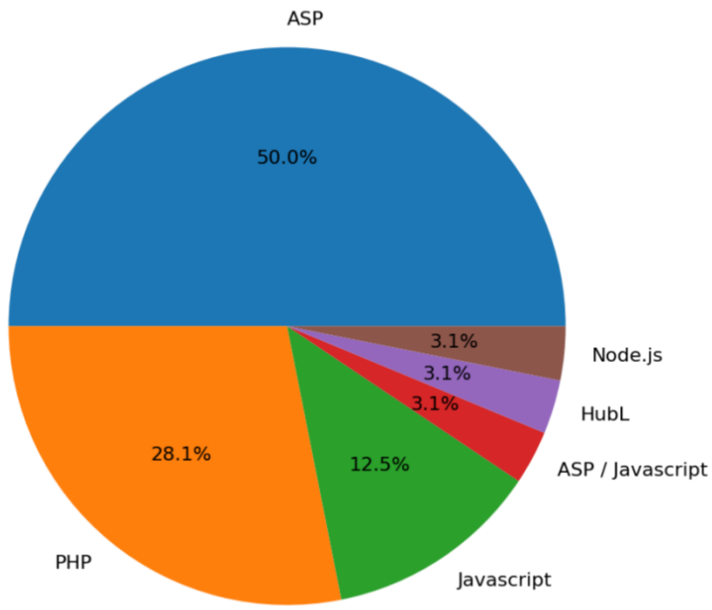

Figure 9. The division of programming languages for the websites analyzed (N = 32).

4.3 Green hosting

Data centers offer server availability suitable for hosting of various digital services (Deng et al. 2012) and it is estimated that there are nearly 2 billion websites available on a global scale (Ridenour 2022). This quick development has resulted in increasing carbon emissions and the impact on climate change is inevitable. Thus, hosting companies constantly strive to lower their carbon footprint, although it may have a negative impact on company revenue streams (Deng et al. 2012).

To track websites that run on green energy, the independent non-profit organization The Green Web Foundation, founded in 2006, maintains a global register. Through this dataset, in combination with the foundation’s open-source tools, environmental aspects of digital products can be analyzed adequately. (The Green Web Foundation 2024)

For this study, the register of The Green Web Foundation was examined to detect how many websites of the sample (N = 32) are green hosted. To obtain hosting information, a tool linked to the register was used in connection with a Google Sheet document to crawl webpages through the *IMPORTXML* function. This function can be used to import data of various types of structured data (Google 2024d). In addition to the *IMPORTXML* function, *XPATH* was used to query data in the data retrieval process. Conditional formatting in the Google Sheet document made the data easier to comprehend.

As visualized in Figure 10, we can conclude that **78%** of the sample web pages are running on green energy, whereas **22%** are not. The websites of Telia Company, Kojamo Oyj, Kesko Oyj, Kemira Oyj, Fortum Oyj, Elisa Oyj, and Citycon Oyj were not included in the register of The Green Web Foundation and were thus classified as “non-green”.

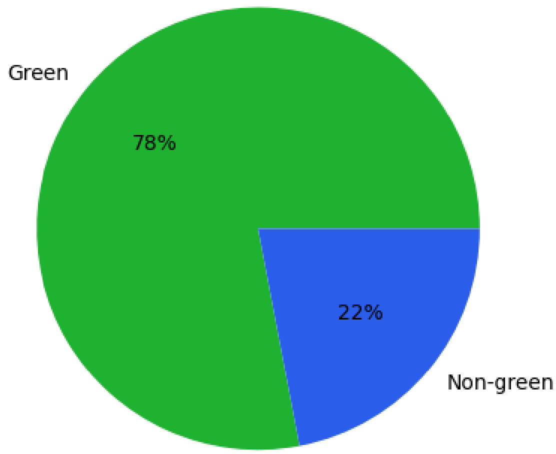

Figure 10. Insights on green hosting of the website sample (N = 32) as defined by The Green Web Foundation (2024).

4.4 Number of requests

When rendering a web page, applications communicate through sending requests between the frontend and backend service. As the HTTP request receives a response, the web browser can construct a user interface (Dubey et al. 2023) and load essential resources such as images and scripts through which the web browser can run the application (Nicoara 2023).

Optimizing download speed should be a core focus of websites, thus, it is key to preload assets before the web page is rendered. To improve download speed of websites, professionals are continuously developing new formats that allow caching and enhancing performance. For example, requests can be handled to preload most critical assets, giving flexibility when rendering a webpage. (Garaizar & Reips 2019)

To obtain the number of requests sent per page view of a website, the PageSpeed Insights API (Google Developers 2024a) by Google was used. This API allowed the usage of the variable of *numRequests*. The *numRequests* variable for all analyzed websites (N = 32) is shown in Figure 12. Based on the sample, the calculated mean of requests is **78.63** resources, and the median lies at **70.00**. Most requests were found on the websites of Sampo Oyj (**223**), UPM-Kymmene Oyj (**140**), KONE Oyj (**127**), Revenio Group Oyj (**122**), and Kesko Oyj

(**120**). Least requests were identified on the websites of Nokian Renkaat Oyj (**26**), Metso Oyj (**40**), Metsä Board Oyj A (**41**), Nordea Bank Abp (**42**), and Kemira Oyj (**46**).

To analyze data distribution in relation to the mean (Parrill et al. 2019), a standard deviation (SD) function was used. The standard deviation for the dataset, calculated in Figure 11, is **37.92**, which was obtained by subtracting the mean (**78.63**) from the squared deviation of all observations (x_i^2), summing these values ($\sum$), dividing this with the sample number (32) and by square rooting this result ($\surd$) (National Center for Biotechnology Information 2024).

$$SD\ \sigma = \sqrt{\frac{\sum (x_i - 78.63)^2}{32}} = 37.92$$

Figure 11. Calculating standard deviation (SD) for the number of requests of a sample of Finnish web pages (N = 32).

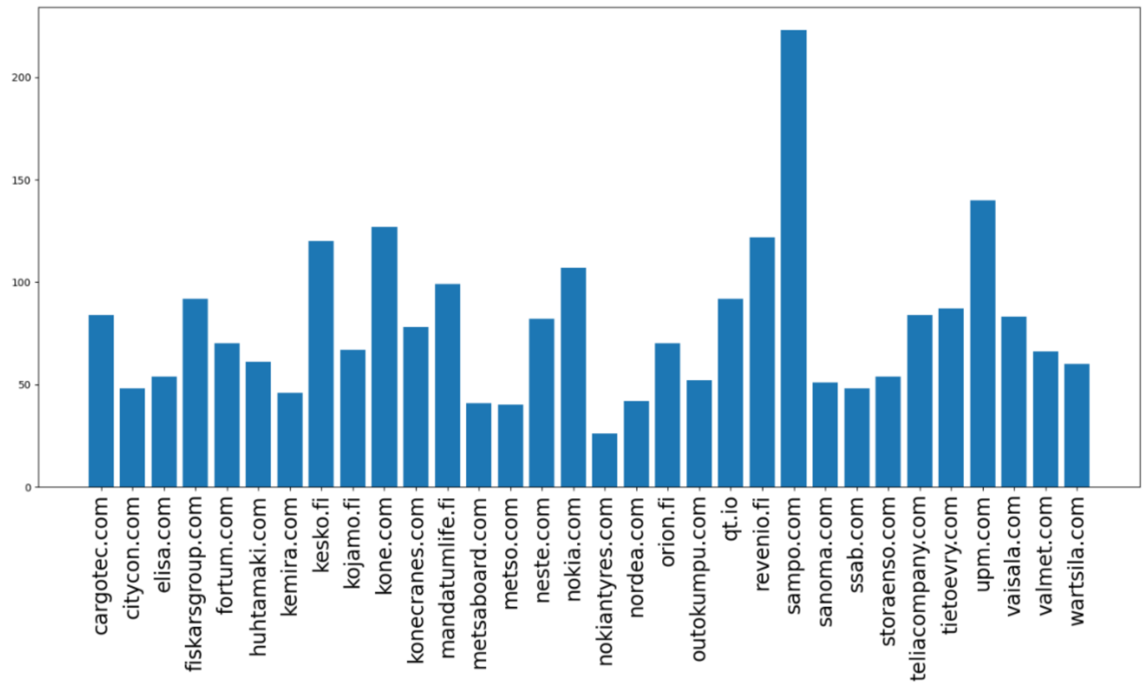

Figure 12. Number of requests per analyzed website.

4.5 Site speed index

Focusing on site speed and usability has become vital for companies (Google 2024a). In addition to improving user experience, these factors have great influence on how websites rank in search engines and how efficiently search engine robots crawl content of a web page (Edgar 2023). By optimizing these areas, firms can improve search visibility and provide a seamless and engaging visitor experience (Google 2024a).

To measure how quickly content is displayed on a web page during page load, Google has developed a metric called Site Speed Index, calculated in seconds (s). This index is one of five metrics used to track Performance in the Lighthouse report by Google (Google Developers (2024b). To retrieve site speed index values for this study, Google's PageSpeed Insights API (Google Developers 2024a) was used to fetch the *site-speed* variable and its *displayValue* attribute.

For the website sample, the mean of the site speed index was **2.12** seconds and the median **1.90** seconds. As indicated in Figure 14, the lowest site speed index was found for the websites of Sanoma Oyj (**1.0**), Citycon Oyj (**1.2**), Wärtsilä Oyj Abp (**1.2**), Elisa Oyj (**1.3**), Fiskars Oyj Abp (**1.3**), Huhtamäki Oyj (**1.3**), and Metso Oyj (**1.3**). The highest site speed index was identified for the websites of Konecranes Oyj (**4.7**), Cargotec Oyj (**4.0**), KONE Oyj (**3.2**), Outokumpu Oyj (**3.0**), and Orion Oyj (**3.0**).

When identifying the standard deviation (SD) (Parrill et al. 2019) of the site speed index sample (N = 32), we can conclude that the dataset SD is **0.86** as calculated in Figure 13. This was calculated by deducting the mean (**2.12**) from the squared deviation of each observation (x_i^2), summing these values ($\sum$), dividing this with the sample size (32) and by square rooting this result ($\surd$) (National Center for Biotechnology Information 2024).

$$\boldsymbol{SD\ \sigma} = \sqrt{\frac{\sum (\boldsymbol{x_i} - \mathbf{2.12})^2}{\mathbf{32}}} = \mathbf{0.86}$$

Figure 13. Calculating standard deviation (SD) for the site speed index of a sample of Finnish web pages (N = 32).

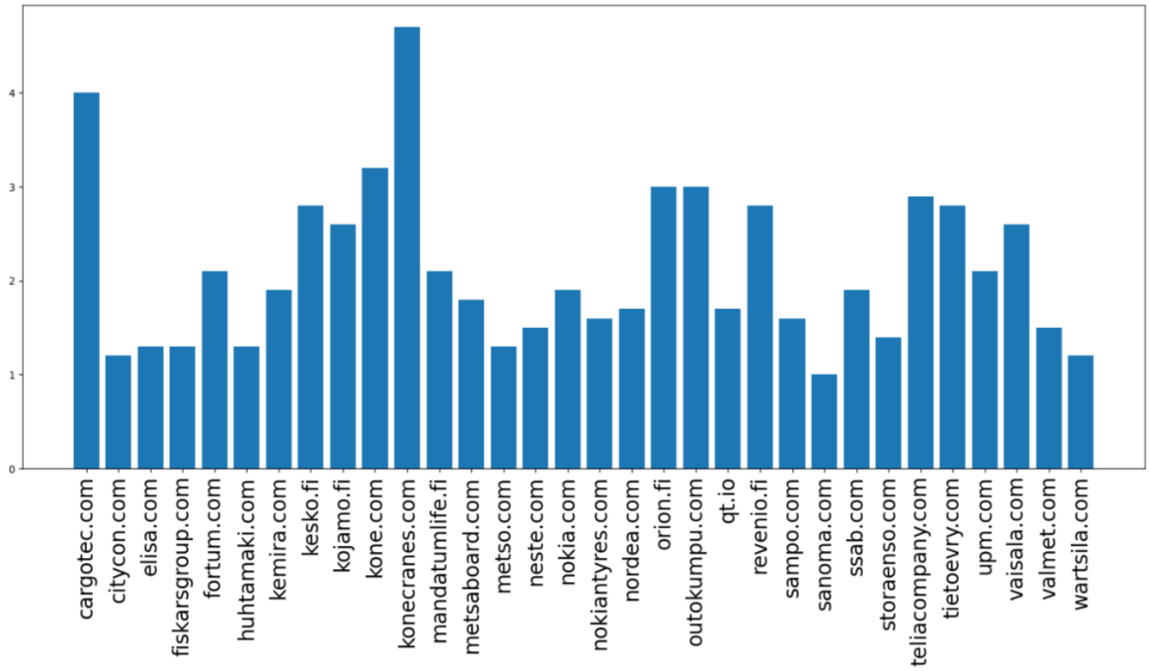

Figure 14. The site speed index ranging from 1 to 4.7 seconds for a sample of Finnish websites (N = 32).

4.6 Google Lighthouse score

A fast webpage with smooth user experience contributes to higher user engagement with content (Google 2024a). Thus, it is key to optimize websites for both speed and usability (Egri & Bayrak 2014) which also impacts search engine visibility positively (Google 2024b).

To examine quality and performance of webpages, Google offers a set of audits that generate actionable reports on how to improve through so called Lighthouse scores (Google Developers 2024c). These are obtained via simulated load testing through lab data (Google 2024b). For this study, an average Google Lighthouse score was calculated through the Google’s PageSpeed Insights API (Google Developers 2024a). This was achieved through running a script in the command line (Terminal) for each sample web page to fetch a numerical score for the *Performance*, *Accessibility*, *Best practices,* and *SEO* variables. The scale of 0 to 100%, from low to high, gives an indication on how to improve the quality of web pages (Google Developers 2024c).

The calculated mean Lighthouse score was **85.54%** and the median **88.00%**. As Figure 16 indicates, the lowest average score was found for the websites of Konecranes Oyj (**67.50%**), Revenio Group Oyj (**75%**), Outokumpu Oyj (**75.75%**), Kesko Oyj (**77.25%**), and UPM-Kymmene Oyj (**77.75%**). The highest identified scores were found for the websites of Elisa Oyj (**97.00%**), Neste Oyj (**94.25%**), Sampo Oyj (**93.25%**), Nordea Bank Abp (**93.00%**), and Nokian Renkaat Oyj (**90.75%**).

To analyze differences in comparison to the mean (Parrill et al. 2019), a standard deviation (SD) was calculated at **6.53** for the dataset. The SD value, as calculated in Figure 15, was obtained by subtracting the mean (**85.54**) from the squared deviation of each observation (x_i^2), summing these values ($\sum$), divided by the sample size (32) and by square rooting the final result ($\surd$) (National Center for Biotechnology Information 2024).

$$\boldsymbol{SD\ \sigma} = \sqrt{\frac{\boldsymbol{\sum (x_i - 85.54)^2}}{\boldsymbol{32}}} = \boldsymbol{6.53}$$

Figure 15. Calculating standard deviation (SD) of Google Lighthouse scores of a sample of Finnish web pages (N = 32).

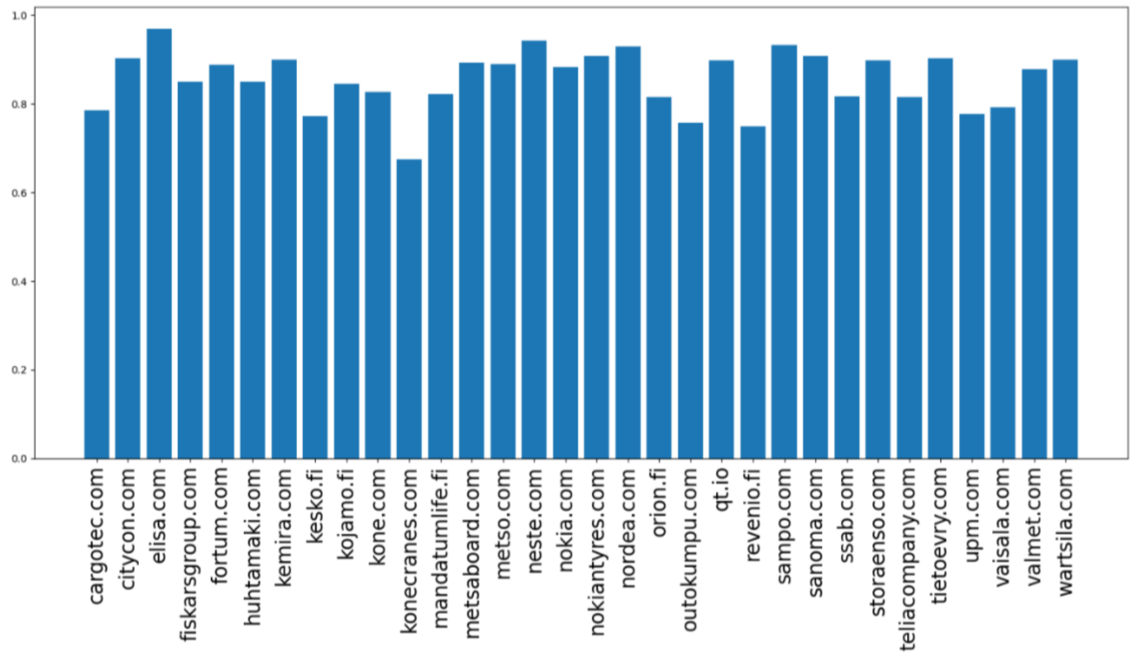

Figure 16. Average Google Lighthouse scores of a sample of Finnish websites (N = 32) rated between 0 (low) and 1 (high).

4.7 Result summary

The sample (N = 32) was analyzed through a set of metrics that focused on sustainability properties of websites. All websites were crawled programmatically to collect equivalent data for analysis. Results are shown sorted ascending by CO_2 emission intensity in Table 6 and include variables demonstrating technical characteristics of the sample.

CO_2 emission intensity in grams was calculated based on downloaded resources in a web browser, which allowed estimating the electricity intensity of data transmission (g CO_2/kWh). The results revealed little variance in emission intensity, suggesting that neither the CMS nor the hosting provider directly impact estimated emissions. Instead, the total requested document size (MB) seemed to be the primary factor.

When investigating programming language and CMS of the sample, results indicate that Optimizely, used by 34% of Finnish websites, was the most popular CMS. This was followed by Drupal, used by approximately 19% of the sample. ASP, a programming language used in all Optimizely websites, was used by 50%, followed by PHP that was used by 28% of the sample. Nevertheless, results show that the choice of programming language was closely connected to the selected CMS and React.js reveals a low emission intensity.

The results show that 78% of the websites are powered by green energy, whereas 22% were categorized as “non-green”. Similarly, the sample revealed a significant difference in the number of HTTP requests, which correlated to a slower page rendering and thus, has implications on the carbon footprint. The site speed index metric shows that most of the sample websites were rendered in less than three seconds. In contradiction, Google Lighthouse scores demonstrated considerable variance in terms of performance and quality, which highlights a need for improvements in speed and rendering techniques.

The results reveal that the number of requests, choice of CMS, and technology have little direct impact on page load times. Nonetheless, the selection of a hosting provider can play a significant role, especially when aiming for green hosting solutions.

Table 6. Result summary of the analyzed websites (N = 32) ordered by ascending emission intensity (g CO_2/kWh).

Website URL	**Emission intensity,** g CO_2/kWh	**HTTP requests,** No.	**Site speed index**, s	**Google Lighthouse score**, %	**Program. language**	**CMS / Technology**	**Green hosting,** boolean
teliacompany.com	0.02	84	2.90	82	JavaScript	React.js	FALSE
revenio.fi	0.09	122	2.80	75	Node.js	React.js	TRUE
elisa.com	0.10	54	1.30	97	JavaScript	React.js	FALSE
fiskarsgroup.com	0.12	92	1.30	85	PHP	WordPress	TRUE
kojamo.fi	0.12	67	2.60	85	PHP	WordPress	FALSE
sanoma.com	0.13	51	1.00	91	ASP	Optimizely	TRUE
orion.fi	0.13	70	3.00	82	ASP	Optimizely	TRUE
kone.com	0.14	127	3.20	83	ASP, JavaScript	SDL Tridion	TRUE
kesko.fi	0.14	120	2.80	77	ASP	Optimizely	FALSE
mandatumlife.fi	0.16	99	2.10	82	ASP	Optimizely	TRUE
fortum.com	0.16	70	2.10	89	PHP	Drupal	FALSE
qt.io	0.16	92	1.70	90	HubL	HubSpot	TRUE
cargotec.com	0.18	84	4.00	79	ASP	Optimizely	TRUE
citycon.com	0.19	48	1.20	90	PHP	Drupal	FALSE
vaisala.com	0.20	83	2.60	79	PHP	Drupal	TRUE
wartsila.com	0.21	59	1.20	90	ASP	Sitefinity	TRUE
storaenso.com	0.21	54	1.40	90	ASP	SiteCore	TRUE
valmet.com	0.22	66	1.50	88	ASP	Optimizely	TRUE
upm.com	0.23	140	2.10	78	ASP	Optimizely	TRUE
sampo.com	0.23	223	1.60	93	ASP	Optimizely	TRUE
neste.com	0.24	82	1.50	94	JavaScript	Contentful	TRUE
metsaboard.com	0.24	41	1.80	89	ASP	Optimizely	TRUE
outokumpu.com	0.25	52	3.00	76	ASP	SiteCore	TRUE
nokiantyres.com	0.29	26	1.60	91	JavaScript	Custom JS	TRUE
konecranes.com	0.32	78	4.70	68	PHP	Drupal	TRUE
ssab.com	0.39	48	1.90	82	ASP	SiteCore	TRUE
nordea.com	0.41	42	1.70	93	PHP	Drupal	TRUE
nokia.com	0.44	107	1.90	88	PHP	Drupal	TRUE
metso.com	0.45	40	1.30	89	ASP	Vue.js	TRUE
kemira.com	0.46	46	1.90	90	PHP	WordPress	FALSE
huhtamaki.com	0.51	61	1.30	85	ASP	Optimizely	TRUE
tietoevry.com	0.52	87	2.80	90	ASP	Optimizely	TRUE

5 Discussion

This chapter reflects how research objectives were met and presents generalizability of the results. Additionally, key findings and implications are summarized, and research limitations identified. Finally, potential future research topics are outlined to conclude the chapter.

5.1 Research questions and objectives

The purpose of this study was to examine sustainable web design as a topic and to analyze a sample of Finnish websites based on a defined set of metrics. It is key to recognize that when assessing sustainability in web design, one cannot bypass emissions as a key metric. This, despite a certain complexity involved in measurement. Thus, additional metrics were considered such as site speed and green hosting, which may be easier to grasp.

The outlined research questions of this study were the following: “How is sustainability in web design defined?” (RQ1), “What are suitable performance metrics in sustainable web design?” (RQ2), and “How does a sample of Finnish Stock Exchange companies position in terms of web design sustainability?” (RQ3). These questions were addressed through developing a theoretical framework based on previous literature, and a comparative analysis of the sample in alignment with the Design Science Research Methodology (DSRM).

To answer the first research question (RQ1), academic research was initially examined by defining the concept of sustainability in a larger context. As there are numerous definitions of sustainability, various aspects were included before its relation to software development was outlined. This general software sustainability perspective, often focusing on its role as a problem and solution in environmental challenges, was then clarified through examples of internet habits (Table 1) and its relation to web design. Finally, the Sustainable Web Manifesto was presented to highlight short and long-term impact of sustainable web design.

The second research question (RQ2) was approached through assessing previous literature on website speed and performance. Concepts such as Google's Core Web Vitals (CWV) was introduced as an example for measuring responsiveness and visual stability of a website. Furthermore, earlier studies were examined to understand measuring and monitoring, as well as selecting website related performance metrics. This content gave adequate tools to define suitable metrics used for the purpose of this study.

To answer the third research question (RQ3), a sample of Finnish websites (N = 32) was analyzed based on the previously defined sustainability metrics. These websites were assessed programmatically through specific tools to obtain information on for example used technology, efficiency, and web server hosting. The selected tools, including two APIs, were customized to give desired output for analysis. It quickly became evident that separate tools would be needed, as the defined metrics addressed different aspects of web design sustainability. Certain tools provided outputs that directly could be used for analysis, whereas others gave results that required calculations, for example when addressing the mean, median, and standard deviation (SD). Through plots, results were visualized to display how companies (N = 32) position in terms of web design sustainability.

5.2 Key findings and implications

This study identifies key aspects of emission intensity, performance, and sustainability of the analyzed websites. Top examples of low-emission websites show a combination of efficient content management systems (CMS), combined with fast load speeds and optimized web design. Conversely, no clear patterns could be identified between high emissions and specific hosting providers, or programming languages. An increased utilization of green hosting services indicates that the industry is shifting towards sustainability. A variance in website size (MB) and the number of requests highlights the significance of efficient network calls and web server optimization. This, to achieve website efficiency and to fight environmental issues.

These findings have practical implications for web developers, hosting providers, and sustainability advocates. A lightweight design and tools to manage web requests are essential to reduce emissions and improve website speeds. In practice, this could involve compressing images, using caching mechanisms, and by selecting a green host. Similarly, developers should aim to utilize technologies that support website efficiency, while hosting providers could further invest in solutions powered by renewable energy. For sustainability advocates, this research highlights a need to educate stakeholders on emission impacts and encourage the adoption of sustainability metrics in companies.

This study adds to previous literature on sustainable web design by combining technical aspects of software with awareness of environmental issues. It challenges the assumption that specific technologies automatically ensure sustainability and direct operations to tailored optimizations that focus on both performance and ecological goals. It also provides a pragmatic approach and presents customizable tools to analyze sustainability of websites. Similarly, it addresses the current state of websites of large Finnish enterprises. These insights contribute to existing academia addressing the environmental impact of digital infrastructure and support global sustainability initiatives such as the Paris Agreement.

Benefits of performance optimization in reduced environmental impact and improved user experience is key for the future of internet. While certain technologies show potential, achieving sustainability requires a broader focus on resource efficiency and active optimization strategies. Collaboration in the digital ecosystem is critical to integrate sustainability into web development practices, and to foster responsibility in an environmentally friendly digital landscape.

5.3 Limitations and generalizability

When crawling websites of companies, it is vital to acknowledge that only a selection of metrics was examined. These metrics were selected based on previous academic research and conclusions of the author. Similarly, one should recognize that for example greenhouse gas emissions should not solely be selected as an indicator. For example, digitalization could also include analysis of a digital service's lifecycle. This could include resource consumption, energy and water, waste production, and pollution (The Green Web Foundation 2022). In simple terms it is key to comprehend that analysis of a firm's sustainable actions are impacted by and involve numerous variables.

Another limitation was that academia proposes different estimations on kWh/GB impact of websites and only one estimation was selected. Thus, the author has chosen a peer reviewed study with an estimate to give an adequate calculation for this study. It is key to mention that estimations may change over time as the digital business landscape evolves. Furthermore, emission calculations were based on producing energy, a combined average of all energy sources in Finland, and not all users are visiting the sample websites from Finland.

Equally important is recognizing that the fetched, analyzed data only captures one moment of time and will inevitably change, should similar research be performed at a later occurrence. Similarly, the analyzed data is relying on third party browser tools and APIs that may have limitations themselves. There could for example exist a variety of estimates when comparing mobile and desktop devices. Similarly, one could challenge the validity of the register of The Green Web Foundation, which may have shortcomings in the dataset. Similarly, one should acknowledge that the APIs by Google use “lab tests” to analyze web pages in a simulated environment (Google Developers 2024a).

The study results are based on technical website performance of a targeted sample of companies (N = 32), which means it is not reflecting a larger market or regional trend. Simultaneously, one should remember that emission calculations of data transfer is challenging and complicated due to numerous variables included (Thornbush 2021).

As the sample include single-page applications (SPAs), that dynamically update content without a complete refresh of the webpage (Sencha 2024), one should recognize that calculations such as website size (MB), rendering speed, and number of requests may only be applicable on the initial load. Similarly, one should acknowledge that web services often combine several programming languages, so it may, from an outsider’s view, be difficult to completely identify and analyze the technology stack of a website.

Another aspect impacting performance of web pages are various external third-party scripts, such as web analytics tracking and advertising pixels. These are loaded in the client by users that have accepted cookie consent of the website. As the utilized tools do not consider these in their audits, this aspect of sustainability is lacking.

The analyzed sample (N = 32) of websites consist of large Finnish enterprises mostly in the business-to-business (B2B) sector. This may have impacted the technology used, server setup and overall resources available for developing each firm's web presence. Should this study have comprised a larger sample of companies, including small and medium-sized companies, the results and for example selected CMS systems would likely have differed. Similarly, one could argue that the number of CMS systems included in the analyzed sample is too small to be statistically significant.

Finally, it is important to mention that the results of this study do not reflect the sample companies' actual actions, impact and sustainability efforts, i.e. strategies fighting climate change and preparing sustainable actions.

5.4 Future work

A comparative evaluation of content management systems (CMS) and their default installations could give insights into their environmental footprint. Analyzing the network calls of an initial page load without customization would show the energy efficiency of each system at its most basic level. Similarly, the increasing popularity of JavaScript-based tools, such as React, Vue.js, and Node.js, justifies further research. By evaluating a wider range of web services, one could identify emerging technologies and find common features among them. This focus on new technologies would not only show trends but also help advance web service technology research. Similarly, using a global sample would provide a broader view on environmental impacts of modern websites.

Expanding the focus to include global, widely used digital services is another possible direction. By selecting a larger sample based on the number of active users, such as the most popular social media or e-commerce platforms, one could examine ecological aspects of the systems. A similar approach could be applied to mobile applications to explore their environmental impact, their technology stack, and possible solutions for making them more sustainable.

Research comparing different web server or database solutions could also shed light on environmental efficiency. By comparing various options, one could measure differences in energy use and emissions. Adding qualitative research could indicate whether environmental factors influence decisions in web service development. Although studies on programming language performance already exist, one could argue there still is a need to focus on their use in a web service development context. Examining trends in programming technologies could lead to useful insights, especially in identifying tools and practices that are environment friendly. Similarly, from a technology perspective, performance comparison between self-hosting and using third-party hosting services could be valuable. This could indicate whether managed hosting services result in any environmental benefits or whether self-hosting setups have a higher potential in terms of sustainability.

A historical perspective could add more depth to research. Yearly comparisons of specific web services could show how their environmental impact may change over time. Additionally, investigating trends in popularity of technologies and how different industries use them, could reveal patterns and motivations behind their implementation.

Interviews or surveys with web development experts could offer detailed insights into why certain systems are chosen. For listed companies, looking at public tendering process materials could indicate popularity of certain CMS systems and may highlight reasoning behind the selection. Similarly, analyzing archived websites could reveal how often websites are renewed completely and shed light on reasons for these changes.

Finally, it could be helpful to discover the role of sustainability in web design beyond technical aspects. Investigating whether companies address sustainability when designing their websites could reveal their awareness of environmental impact. Studying how sustainability fits into broader business operations could provide a better picture of its relevance in web service practices.

6 Conclusions

This study aimed to examine sustainable web design by addressing three research questions: 1) How is sustainability in web design defined? 2) What are suitable performance metrics in sustainable web design? and 3) How does a sample of Finnish Stock Exchange companies position in terms of web design sustainability? These objectives were reached through an analysis of Finnish websites (N = 32), evaluating emissions, site speed, and performance metrics, while considering aspects such as programming languages, web hosting, and CMS.

The findings revealed that, although there is an increased attention to sustainability, gaps in implementation still exist. Inadequate practices continue to increase energy consumption and emissions, although modern browsers continuously improve website rendering. The results emphasize a need for efficient technologies that address environmental impacts.

The theoretical framework contributes by describing sustainability in web design and defining measurable metrics, adding to previous academic discourse on sustainable software practices. Practical applications include strategies for developers and organizations to match digital operations with environmental goals, showing the potential to reduce emissions and meet changing consumer expectations. Comparisons with prior studies confirm the dual role of software in both reducing and contributing to environmental challenges. Issues, such as varying levels of adoption across organizations, highlight the need for standardized development practices and optimization of digital services.

Limitations include a focus on Finnish companies, depending on third-party tools in data collection, and a lack of historical data and user perspectives. This may affect the generalizability of findings. Future research could address these gaps by exploring a global context, emerging web development technologies, and the behavioral drivers behind sustainable web design decisions.

This study emphasizes a societal and industry-wide importance of sustainable web design. By applying environmentally conscious practices, organizations can meet sustainability goals while fostering innovation and addressing climate change.

References

Andrae, A.S.G. & Edler, T. (2015). On global electricity usage of communication technology: trends to 2030. Challenges, 6(1), pp. 117-157.

Antinoja, R. (2021). Optimizing supply chain sustainability: A case study. Master's Thesis. Lappeenranta-Lahti University of Technology LUT.

Aslan, J. et al. (2018). Electricity Intensity of Internet Data Transmission: Untangling the Estimates. Journal of industrial ecology. 22 (4), pp. 785–798.

Banerjee, A. & Chaudhury, S. (2010). Statistics without tears: Populations and samples, Industrial Psychiatry Journal, 19(1), pp. 60–65. doi: 10.4103/0972-6748.77642.

Baral, K. (2024). Types of Sampling. Department of Political Science, Sonada Degree College, University of North Bengal.

Baskerville, R. et al. (2003). Is "Internet-speed" software development different?. IEEE Software, 20(6), pp. 70–77.

Becker, C. et al. (2015). Sustainability and Software Engineering: Key Issues and Challenges. In Proceedings of the 37th International Conference on Software Engineering (ICSE 2015). IEEE, pp. 530-540.

Bertoa, M.F. et al. (2021). Impact of Human Factors on Software Sustainability. In: Proceedings of the 2nd International Workshop on Green and Sustainable Software, IEEE Press, pp. 46-53.

Bugl, D. (2019). Learn React Hooks: Build and Refactor modern React.js Applications Using Hooks. 1st edition. Birmingham, England; Packt.

Calero, C. et al. (2021). Software Sustainability. Switzerland: Springer International Publishing AG. pp. 329–348.

Calero, C. et al. (2022). Software, Sustainability, and UN Sustainable Development Goals. IT professional. Vol. 24 (1), pp. 41–48.

Cargotec Oyj. (2024). Our Story and History. Available at: https://www.cargotec.com/en/about-Cargotec/our-story-and-history/history (Accessed: 9 November 2024).

Christianson, C. & Cochran, J. (2009). ASP.NET 3.5 Content Management System development: build, manage, and extend your own Content Management System. 1st ed. Birmingham, U.K: PACKT Publishing.

Cisco. (2015). Global Internet Traffic Growth Projections. Cisco Systems. Available at: https://blogs.cisco.com/sp/2015-cisco-vni-complete-forecast-update-key-trends-include-mobility-m2m-and-multimedia-content (Accessed: 26 September 2024).

Contreras, L. M. et al. (2023). Analysis of network function sharing in Content Delivery Network-as-a-service slicing scenarios. International journal of network management. Vol. 33 (4).

Creswell, J.W. (2014). Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. 4th ed. Thousand Oaks, CA: Sage.

Davison, B. D. (2001). A Web caching primer. IEEE internet computing. Vol. 5, pp. 38–45.

Deng, N. et al. (2012). Adaptive green hosting, in Proceedings of the 9th international conference on Autonomic computing. 2012 New York, NY, USA: ACM. pp. 135–144.

Dubey, A. et al. (2023). Customized Framework for Backend Using Node JS, in 2023 International Conference on Sustainable Emerging Innovations in Engineering and Technology (ICSEIET). 2023 IEEE. pp. 01–05.

Edgar, M. (2023) Speed Metrics Guide: Choosing the Right Metrics to Use When Evaluating Websites.

Egri, G. & Bayrak, C. (2014). The Role of Search Engine Optimization on Keeping the User on the Site, Procedia Computer Science, Vol. 36, pp. 335-342.

Engebreth, G. & Sahu, S. K. (2023). PHP 8 Basics For Programming and Web Development. 1st ed. Berkeley, CA: Apress.

Episerver. (2021). Episerver Optimizely named a leader in 2021 Gartner Magic Quadrant for Digital Experience Platforms. [online] Available at: https://www.prnewswire.com/news-releases/episerver-optimizely-named-a-leader-in-2021-gartner-magic-quadrant-for-digital-experience-platforms-301220540.html (Accessed: 23 September 2024).

European Environment Agency. (2024). Greenhouse gas emission intensity of the economy. Available at: https://www.eea.europa.eu/en/analysis/indicators/greenhouse-gas-emission-intensity-of-1?activeAccordion=546a7c35-9188-4d23-94ee-005d97c26f2b (Accessed: 25 October 2024).

Everman, B. & Zong, Z. (2018). GreenWeb: Hosting High-Load Websites Using Low-Power Servers, in 2018 Ninth International Green and Sustainable Computing Conference (IGSC). 2018 IEEE. pp. 1–6.

Fiskars Oyj Abp. (2024). About Us. Available at: https://fiskarsgroup.com/about-us/ (Accessed: 9 November 2024).

Frain, B. (2020). Responsive Web Design with HTML5 and CSS - Third Edition. Third edition. Packt Publishing.

FreeCodeCamp. (2024). An introduction to Web Performance and the Critical Rendering Path. freeCodeCamp. Available at: https://www.freecodecamp.org/news/an-introduction-to-web-performance-and-the-critical-rendering-path-ce1fb5029494/ (Accessed: 5 October 2024).

Garaizar, P. & Reips, U.D. (2019) Best practices: Two web-browser-based methods for stimulus presentation in behavioral experiments with high-resolution timing requirements, Behavior Research Methods, 51(3), pp. 1441–1453.

Gasston, P. (2013). The modern Web: multi-device Web development with HTML5, CSS3, and JavaScript. 1st edition. San Francisco: No Starch Press.

Ginesu, G. et al. (2012). Objective assessment of the WebP image coding algorithm. Signal processing. Image communication. Vol. 27 (8), pp. 867–874.

Google. (2024a). Why speed matters. Available at: https://web.dev/learn/performance/why-speed-matters (Accessed: 25 September 2024).

Google. (2024b). Web Vitals: Essential metrics for a healthy site. Available at: https://web.dev/articles/vitals (Accessed: 25 September 2024).

Google. (2024c). Using web fonts: Optimizing font loading. Google Fonts Knowledge. Available at: https://fonts.google.com/knowledge/using_type/using_web_fonts - optimizing-font-loading (Accessed: 26 August 2024).

Google. (2024d). IMPORTXML - Google Docs Editors Help. Available at: https://support.google.com/docs/answer/3093342 (Accessed: 27 October 2024).

Google Developers. (2024a). Get started with Google PageSpeed Insights. Available at: https://developers.google.com/speed/docs/insights/v5/get-started (Accessed: 25 October 2024).

Google Developers. (2024b). Speed Index. Available at: https://developer.chrome.com/docs/lighthouse/performance/speed-index (Accessed: 31 October 2024)

Google Developers. (2024c). Overview | Lighthouse. Available at: https://developer.chrome.com/docs/lighthouse/overview (Accessed: 31 October 2024)

Gordillo, A. et al. (2024). Programming languages ranking based on energy measurements. Software quality journal.

Goyal, M. R. (2015). Sustainable practices in surface and subsurface micro irrigation. 1st ed. Toronto, Ontario: Apple Academic Press.

Graham, H. (2020). Why your internet habits are not as clean as you think, BBC Future, 5 March. Available at: https://www.bbc.com/future/article/20200305-why-your-internet-habits-are-not-as-clean-as-you-think (Accessed: 5 November 2024).

Greenwood, T., & Frick, T. (2024). Sustainable Web Manifesto. Available at: https://corporatedigitalresponsibility.net/sustainable-web-manifesto (Accessed: 18 October 2024).

Hevner, A. R. et al. (2004). Design Science in Information Systems Research, MIS Quarterly, Vol. 28(1), pp. 75–105

Hikolakita, (2024). How Modern Browsers Render Websites: A Behind-the-Scenes Look. Dev.to. Available at: https://dev.to/hikolakita/how-modern-browsers-render-websites-a-behind-the-scenes-look-5bfn (Accessed: 5 October 2024).

Hu, J. et al. (2017). Comparative performance analysis of web image compression. 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI). 2017 IEEE. pp. 1–5.

Hussien Odeh, A. (2019). Analytical and Comparison Study of Main Web Programming Languages – ASP and PHP, TEM Journal, 8(4), pp. 1517–1522.

IBM (2022). Design for Sustainability Position Paper. Available at: https://www.ibm.com/design/practices/design-for-sustainability/design-for-sustainability-positionpaper.pdf (Accessed: 13 July 2024).

Johnston, P. et al. (2007). Reclaiming the definition of sustainability. Environmental science and pollution research international. Vol. 14 (1), pp. 60–66.

Konshin, K. (2018). Next.js Quick Start Guide: Server-Side Rendering Done Right. 1st ed. Birmingham: Packt Publishing.

Kulkarni, S. & Haghi, A.K. (2024) Global Sustainability: Trends, Challenges & Case Studies. 1st edn. Cham: Springer Nature

Kumar, A. et al. (2022). Software-defined content delivery network at the edge for adaptive video streaming. International journal of network management. pp. 32 (6).

Küng, L. (2017). Strategic Management in the Media - Theory to Practice. 2nd Edition. SAGE Publications Ltd., London.

Küng, L. (2020). Hearts And Minds: Harnessing Leadership, Culture, And Talent To Really Go Digital. Reuters Institute. Available at: https://reutersinstitute.politics.ox.ac.uk/sites/default/files/2020-11/Kueng%20-%20Hearts%20and%20Minds%20FINAL.pdf. [Accessed: 20 October 2024)

Kuruvilla, A. M. (2012). Tiled Image Container for Web Compatible Compound Image Compression, in 2012 Eighth International Conference on Signal Image Technology and Internet Based Systems. 2012 IEEE. pp. 182–187.

Lago, P., et al., (2015). Green and Sustainable Software Engineering. In: International Workshop on Green and Sustainable Software (GREENS), Florence, Italy. IEEE, pp. 1-8.

Libby, A. (2014). Learning Less.js: Develop Attractive CSS Styles Efficiently, Using the Less CSS Preprocessor. 1st ed. Birmingham: Packt Publishing.

Liu, C., et al. (2021). Energy-Aware Software Engineering. IntechOpen.

Lopez, D. A. P. (2021) Full-stack Web Development with Jakarta EE and Vue.js: Your One-stop Guide to Building Modern Full-stack Applications with Jakarta EE and Vue.js. 1st ed. New York, NY: Apress L. P.

Lorey, T. et al. (2022). Social Science Theories in Software Engineering Research. arXiv preprint arXiv:2202.07519.

Masanet, E. et al. (2020). Recalibrating Global Data Center Energy-Use Estimates. Science, Vol. 367(6481), pp. 984-988.

MDN Web Docs. (2024). rel=preload - HTML: HyperText Markup Language. Available at: https://developer.mozilla.org/en-US/docs/Web/HTML/Preloading_content (Accessed: 5 October 2024).

Mesbah, A. et al. (2001) ASP.NET Web Developer’s Guide. 1st ed. Rockland, MA: Syngress Publishing, Inc.

Monotype. (2024). The Evolution of Typography with Variable Fonts. Available at: https://www.monotype.com/resources/articles/the-evolution-of-typography-with-variable-fonts (Accessed: 5 October 2024).

Moradi, M. et al. (2019). Concept-Aware Web Image Compression Based on Crowdsourced Salient Object Detection, in 2019 5th International Conference on Web Research (ICWR). 2019 IEEE. pp. 221–227.

Mozilla Developer Network, (2024). User Timing API. Available at: https://developer.mozilla.org/en-US/docs/Web/API/User_Timing_API (Accessed: 5 October 2024).

Nasdaq (2024). OMX Helsinki All-Share Index Consumer Price Index Gross. Available at: https://indexes.nasdaqomx.com/Index/Overview/OMXHLCPI (Accessed: 5 November 2024).

National Center for Biotechnology Information. (2024). Standard Deviation. Available at: https://www.ncbi.nlm.nih.gov/books/NBK574574/ (Accessed: 25 October 2024).

Neste Oyj. (2024). Who We Are. Available at: https://www.neste.com/about-neste/who-we-are (Accessed: 9 November 2024).

Nicoara, R. (2023). How to be a Web Developer: A Complete Beginner’s Guide on What to Know and Where to Start. 1st ed. Berkeley, CA: Apress.

Niranga, S. S. (2015). Mobile web performance optimization : deliver a better mobile user experience by improving and optimizing your website-- follow these practical steps for cutting-edge application development. Birmingham: Packt Publishing.

Nokia Oyj. (2024). Our Locations: Finland. Available at: https://www.nokia.com/about-us/careers/our-locations/finland/ (Accessed: 9 November 2024).

Novosel, L. M. (2023). Understanding the Evidence: Non-Probability (Non-Random) Sampling Designs. Urologic nursing. Vol. 43 (5), pp. 245-.

Parker, A. et al. (2010). Using caching and optimization techniques to improve performance of the Ensemble website. BMC bioinformatics. Vol. 11 (1), pp. 239–239.

Parrill, F. et al. (2019). Gesturing standard deviation: Gestures undergraduate students use in describing their concepts of standard deviation. The Journal of mathematical behavior. Pp. 531–.

Peffers, K. et al (2007). A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems. Vol. 24 (3). pp. 45-77.

Pereira, R. et al. (2017). Energy efficiency across programming languages: how do energy, time, and memory relate?, in Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2017). New York, NY: Association for Computing Machinery, pp. 256–267. Available at: https://doi.org/10.1145/3136014.3136031 (Accessed: 14 September 2024).

Pickvance, C. G. (2001) Four varieties of comparative analysis. Journal of housing and the built environment. [Online] 16 (1), 7–28.

Ragin, C.C., (1987). The Comparative Method: Moving Beyond Qualitative and Quantitative Strategies. University of California Press, Berkeley.

Rajeswari, R. & Rajesh, R. (2010). Image compression techniques – a survey. Advances in Computer Science and Engineering 5 (2). pp. 231-263
Richardson, T. & Thies, C. (2013). Multimedia Web design and development: using languages to build dynamic web pages. 1st ed. Dulles, Virginia; Mercury Learning and Information.

Ridenour, A. (2022). How many websites are on the Internet? . Available at: https://www.howtogeek.com/845500/how-many-websites-are-on-the-internet/ (Accessed 6 Dec. 2024).

ScienceDirect. 2024. Convenience Sampling. [online] Available at: https://www.sciencedirect.com/topics/computer-science/convenience-sampling [Accessed 25 September 2024].

Seadle, M. (2006). Content management systems. Library hi tech. Vol. 24 (1), pp. 5–7.

Sencha, (2024). Building Single-Page Applications (SPAs) With JavaScript Frameworks. Available at: https://www.sencha.com/blog/building-single-page-applications-spas-with-javascript-frameworks/ (Accessed: 5 October 2024).

Sustainable Web Design. (2024). Estimating Digital Emissions. Available at: https://sustainablewebdesign.org/estimating-digital-emissions/ (Accessed: 13 July 2024).

Sustainable Web Manifesto. (2024). Sustainable Web Manifesto. Available at: https://www.sustainablewebmanifesto.com (Accessed: 13 July 2024).

SVID. (2018). Design – Sustainability Guide. Available at: https://sustainabilityguide.eu/ecodesign/design/ (Accessed: 13 July 2024).

Teke, F. (2023). The Impact of Web Development on Global Carbon Emissions. 1st ed. London: Green Digital Publishing.

The Green Web Foundation. (2022). The Fog of Enactment: Conclusion. [online] Available at: https://www.thegreenwebfoundation.org/publications/report-fog-of-enactment/#conclusion (Accessed: 25 July 2024).

The Green Web Foundation. (2024). About. Available at: https://www.thegreenwebfoundation.org/about/ (Accessed: 27 October 2024).
Thornbush, M. J. (2021). The Ecological Footprint As a Sustainability Metric: Implications for Sustainability. 1st edition. Cham: Springer International Publishing AG.

Ukko, J. et al. (2018). Sustainable development: Implications and definition for open sustainability.

Urdan, M. S. & Luoma, P. (2020). Designing Effective Sustainability Assignments: How and Why Definitions of Sustainability Impact Assignments and Learning Outcomes. Journal of management education. Vol. 44 (6), pp. 794–821.

Utesheva, A. (2020) Designing Products for Evolving Digital Users: Study UX Behavior Patterns, Online Communities, and Future Digital Trends.

W3C. (2024). World Wide Web Consortium (W3C). Available at: https://www.w3.org/ (Accessed: 26 August 2024).

Wang, J. et al. (2023). How does the internet economy affect CO_2 emissions? Evidence from China. Applied economics. Vol. 55 (4), pp. 447–466.

Wellens, P. (2015). Practical web development: learn CSS, JavaScript, PHP, and more with this vital guide to modern web development. 1st ed. Birmingham, England: Packt Publishing. pp 44-49

WhatCMS. (2024). Documentation - What CMS? Available at: https://whatcms.org/Documentation (Accessed: 27 October 2024).

Zhao, Y. et al. (2019). An Optimization Method of WebP Images Lossy Compression Algorithm for FPGAs, IEEE. pp. 705–711.

Zolfaghari, B. et al. (2020). Content Delivery Networks: State of the Art, Trends, and Future Roadmap. ACM computing surveys. Vol. 53 (2), pp. 1–34.

Appendix

Appendix 1. Data collection through Google's PageSpeed Insights API.

```bash
#!/bin/bash

# Define the input file
INFILE=urls.txt

# Read the input file line by line
while read -r LINE
do
    printf '%s\n' "$LINE"
    curl -o response_"$LINE".txt "https://www.googleapis.com/pagespeedonline/v5/runPagespeed?url=https://"$LINE"&strategy=desktop&category=performance&category=accessibility&category=best_practices&category=seo&key=secretkey"

done < "$INFILE"

```

Appendix 2. Data collection through a technology endpoint API by WhatCMS.

```bash
#!/bin/bash

# Define the input file
INFILE=urls.txt

# Read the input file line by line
while read -r LINE
do
    printf '%s\n' "$LINE"

    curl -o CMS/response_"$LINE".json -G https://whatcms.org/API/Tech \
    --data-urlencode key="secretkey" \
    --data-urlencode url=""$LINE""
    sleep 11

done < "$INFILE"

```

Appendix 3. The sample dataset was analyzed through technical performance metrics.

	A	B	C	D	E	F	G	H	
1	Nasdaq - Hel - Large Cap - Company	URL	CDN	Cache	CMS	Green hosting	Hosting provider	PageSpeed Insights API, Performance	PageSpeed Insig
2									
3	Cargotec Oyj	https://www.cargotec.com/	Cloudflare	TRUE	Optimizely, Episerver	TRUE	Optimizely	0,62	0,86
4	Citycon Oyj	https://www.citycon.com/	NA	TRUE	Drupal	#N/A	UpCloud	0,93	0,87
5	Elisa Oyj	https://www.elisa.com/	NA	TRUE	React.js	#N/A	Elisa	0.94	0,98
6	Fiskars Oyj Abp	https://www.fiskarsgroup.com/	NA	TRUE	WordPress	TRUE	Hetzner	0,86	0,81
7	Fortum Oyj	https://www.fortum.com/	Fastly	TRUE	Drupal	#N/A	Google	0,76	0,98
8	Huhtamäki Oyj	https://www.huhtamaki.com/	Cloudflare	TRUE	Optimizely, Episerver	TRUE	Optimizely	0,93	0,77
9	Kemira Oyj	https://www.kemira.com/	Amazon CloudFront	TRUE	WordPress	#N/A	Kemira	0,80	0,88
10	Kesko Oyj	https://www.kesko.fi/	NA	TRUE	Optimizely, Episerver	#N/A	Elisa	0,61	0,69
11	Kojamo Oyj	https://kojamo.fi/	NA	FALSE	WordPress	#N/A	UpCloud	0,81	0,76
12	KONE Oyj	https://www.kone.com/	Akamai	TRUE	SDL Tridion	TRUE	Microsoft Corporation	0,51	0,95
13	Konecranes Oyj	https://www.konecranes.com/	Cloudflare	TRUE	Drupal	TRUE	Netmagic	0,20	0,87
14	Mandatum	https://www.mandatumlife.fi/	Cloudflare	TRUE	Optimizely, Episerver	TRUE	Optimizely	0,81	0,78
15	Metsä Board Oyj A	https://www.metsaboard.com/	Cloudflare	TRUE	Optimizely, Episerver	TRUE	Microsoft Corporation	0,91	0,96
16	Metso Oyj	https://www.metso.com/	Azure Front Door	TRUE	Vue.js	TRUE	Microsoft Corporation	0,83	0,95
17	Neste Oyj	https://www.neste.com/	Azure Front Door	TRUE	Contentful	TRUE	Microsoft Corporation	0,87	0,98
18	Nokia Oyj	https://www.nokia.com/	Akamai	TRUE	Drupal	TRUE	Google	0,75	0,82
19	Nokian Renkaat Oyj	https://www.nokiantyres.com/	Cloudflare	TRUE	Custom JS	TRUE	Amazon	0,96	0,86
20	Nordea Bank Abp	https://www.nordea.com/	Cloudflare	TRUE	Drupal	TRUE	Nordea Bank AB	0,87	0,93
21	Orion Oyj	https://www.orion.fi/	Cloudflare	TRUE	Episerver	TRUE	Optimizely	0,56	1
22	Outokumpu Oyj	https://www.outokumpu.com/	NA	TRUE	SiteCore	TRUE	Microsoft Corporation	0,44	0,83
23	Qt Group Oyj	https://www.qt.io/	Cloudflare	TRUE	HubSpot	TRUE	Amazon	0,93	0,81
24	Revenio Group Oyj	https://www.revenio.fi/	Google	TRUE	React.js	TRUE	Google	0,41	0,76
25	Sampo Oyj	https://www.sampo.com/	Cloudflare	TRUE	Episerver	TRUE	Microsoft Corporation	0,95	0,97
26	Sanoma Oyj	https://www.sanoma.com/	Cloudflare	TRUE	Episerver	TRUE	Amazon	0,95	0,95
27	SSAB	https://www.ssab.com/	Cloudflare	TRUE	SiteCore	TRUE	TELE2	0,80	0,7
28	Stora Enso Oyj	https://www.storaenso.com/	Akamai	TRUE	SiteCore	TRUE	Microsoft Corporation	0,96	0,9
29	Telia Company	https://www.teliacompany.com/	Amazon CloudFront	TRUE	React.js	#N/A	Telia Sweden	0,34	1
30	TietoEVRY Oyj	https://www.tietoevry.com/	Cloudflare	TRUE	Optimizely, Episerver	TRUE	Optimizely	0,69	1
31	UPM-Kymmene Oyj	https://www.upm.com/	Cloudflare	TRUE	Optimizely, Episerver	TRUE	Optimizely	0,85	0,71
32	Vaisala Oyj	https://www.vaisala.com/	Cloudflare	TRUE	Drupal	TRUE	Vaisala Oyj	0,86	0,9
33	Valmet Oyj	https://www.valmet.com/	Cloudflare	TRUE	Episerver	TRUE	Optimizely	0,90	0,91
34	Wärtsilä Oyj Abp	https://www.wartsila.com/	Amazon CloudFront	TRUE	Sitefinity	TRUE	Amazon	0,91	0,96
35									

Appendix 4. Data visualization and analysis was handled in Python3 through the Matplotlib and Pandas library.

```
#Imports
import matplotlib.pyplot as plt
import pandas as pd

#Emission Metrics
df = pd.read_csv("emission_c02.csv")
fig, ax = plt.subplots()

sites = df["site"]
n = mylabels = df["emission_c02"]

ax.bar(sites, n)
ax.set_xticklabels(sites, rotation=90, fontsize=20)

plt.show()

```

```
#Imports
import matplotlib.pyplot as plt
import pandas as pd

#CWV Metrics
df = pd.read_csv("cwv.csv")
fig, ax = plt.subplots()

sites = df["www"]
n = mylabels = df["cwv"]

ax.bar(sites, n)
ax.set_xticklabels(sites, rotation=90, fontsize=20)

plt.show()

```

Appendix 5. The site speed index metric was collected and analyzed through Google's PageSpeed Insights API.


```
"speed-index": {
  "id": "speed-index",
  "title": "Speed Index",
  "description": "Speed Index shows how quickly the contents of a page are visibly populated. [Learn more about the Speed Index metric](https://developer.chrome.com/docs/lighthouse/performance/speed-index/).",
  "score": 0.92,
  "scoreDisplayMode": "numeric",
  "displayValue": "1.2s",
  "numericValue": 1213.6474352236105,
  "numericUnit": "millisecond"
},
```


Appendix 6. The site speed index for the sample of analyzed websites (N = 32).

www	**speed-index.score**	**www**	**speed-index.score**
cargotec.com	4	nokiantyres.com	1.6
citycon.com	1.2	nordea.com	1.7
elisa.com	1.3	orion.fi	3
fiskarsgroup.com	1.3	outokumpu.com	3
fortum.com	2.1	qt.io	1.7
huhtamaki.com	1.3	revenio.fi	2.8
kemira.com	1.9	sampo.com	1.6
kesko.fi	2.8	sanoma.com	1
kojamo.fi	2.6	ssab.com	1.9
kone.com	3.2	storaenso.com	1.4
konecranes.com	4.7	teliacompany.com	2.9
mandatumlife.fi	2.1	tietoevry.com	2.8
metsaboard.com	1.8	upm.com	2.1
metso.com	1.3	vaisala.com	2.6
neste.com	1.5	valmet.com	1.5
nokia.com	1.9	wartsila.com	1.2

Appendix 7. Review of the Green Web Foundation's register on green hosts implemented through Google Sheets and the *IMPORTXML* function.

	A	C	D	E	
1	url	crawl_url_no_www	is_green (www)	is_green (no_www)	
2	www.cargotec.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
3	www.citycon.com	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	No evidence found for	#N/A
4	www.elisa.com	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	No evidence found for	#N/A
5	www.fiskarsgroup.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
6	www.fortum.com	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	No evidence found for	#N/A
7	www.huhtamaki.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
8	www.kemira.com	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	No evidence found for	#N/A
9	www.kesko.fi	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	No evidence found for	#N/A
10	www.kojamo.fi	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	No evidence found for	#N/A
11	www.kone.com	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	We've found evidence	TRUE
12	www.konecranes.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
13	www.mandatumlife.fi	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
14	www.metsaboard.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
15	www.metso.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
16	www.neste.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
17	www.nokia.com	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	We've found evidence	TRUE
18	www.nokiantyres.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
19	www.nordea.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
20	www.orion.fi	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
21	www.outokumpu.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
22	www.qt.io	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	We've found evidence	TRUE
23	www.revenio.fi	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
24	www.sampo.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
25	www.sanoma.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
26	www.ssab.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
27	www.storaenso.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	We've found evidence	TRUE
28	www.teliacompany.com	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	No evidence found for	#N/A
29	www.tietoevry.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
30	www.upm.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
31	www.vaisala.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
32	www.valmet.com	https://www.thegreenwebfoundation.org/green-web-c	We've found evidence	No evidence found for	TRUE
33	www.wartsila.com	https://www.thegreenwebfoundation.org/green-web-c	No evidence found for	We've found evidence	TRUE
34					
35					
36		Based on The Green Web Foundation	**78.13%**	is green	
37		Based on The Green Web Foundation	**21.88%**	is not green	

Appendix 8. JSON response of the technology endpoint API by WhatCMS.


```
{
    "request": "https://whatcms.org/API/Tech?key=[redacted]
    "result": {
        "code": 200,
        "msg": "Success"
    },
    "results": [
        {
            "name": "Episerver",
            "id": "23",
            "version": "",
            "categories": [
                "E-commerce",
                "CMS"
            ],
            "url": "//whatcms.org/c/Episerver"
        },
        {
            "name": "Optimizely Content Management",
            "id": 4640,
            "version": "",
            "categories": [
                "Other CMS",
                "CMS"
            ],
            "url": "//whatcms.org/c/Optimizely-Content-Management"
        },
        {
            "name": "Microsoft ASP.NET",
            "id": 1548,
            "version": "4.0.30319",
            "categories": [
                "Web Framework"
            ],
            "url": "//whatcms.org/c/Microsoft-ASP-NET"
        },
        {
            "name": "Cloudflare",
            "id": 1174,
            "version": "",
            "categories": [
                "CDN"
            ],
            "url": "//whatcms.org/c/Cloudflare"
        }
    ],
```